Monte Carlo Technique

Particle scattering and absorption

- ❒ involves random processes
- ❒ uses complicated multidimensional integrals

Use Monte Carlo technique to do the multidimensional integration

Solution of a problem as a parameter of a hypothetical population and constructing a sample of the population to obtain estimates of the parameter

$\Rightarrow$ use random numbers to construct a sample

We may need to carry out an integration:

$$\mathrm{I} = \int_0^1 \mathrm{dx_n} \cdots \int_0^1 \mathrm{dx_1 F(x_1, \cdots x_n)}$$

Then with a set of random numbers x_1, $\cdots$ x_n in the range $0 - 1$, determine F and this F will be an unbiased estimate of I

Repeat this estimate for a large number of times and $\bar{\mathrm{F}}$ will converge to the value of I

Random Variables

- It can have more than one value (generally any value within a range)
- One cannot predict in advance which value it will take
- The distribution of the variable may be well known

Distribution of a random variable $\Rightarrow$ probability of having a specific value

Probability distribution function is given by

$$g(u)du = P[u < u' < u + du]$$

Integrated distribution function:

$$G(u) = \int_{-\infty}^{u} g(u')du'$$

$$g(u) = \frac{dG(u)}{du}$$

G(u) increases monotonically with u. Normalisation of g is determined by

$$\int_{-\infty}^{\infty} g(u)du = 1$$

Truly random numbers

Sequence of truly random numbers is completely unpredictable and hence irreproducible

Can be generated only through random physical processes (radioactive decay, arrival time of cosmic ray particles, $\cdots$)

Quasi random numbers

Sequence do not appear random (high degree of correlation) but give right answers to Monte Carlo integration

These use strict mathematical formula and provide fast convergence of integration

These are of limited use

Pseudo random numbers

Sequence generated according a strict mathematical formula, but indistinguishable from a sequence generated truly randomly

Most simulation programs use pseudo random numbers. The heart of the simulation process is generation of "Uniform Deviates": random numbers which lie within a specified range (0 to 1) with any number just as likely as the other

- ❑ Start with a number of r digits. The first random is the middle r/2 digits. Square this number and again take the middle r/2 digits for the next random number and so on. This procedure is machine dependent, has large correlation and also has small period.
- ❑ Multiplicative Linear Congruental Generator: This is the most common random number generator which generates a sequence of integers between 0 and m−1 (a large number) using the recurrence relation

  $$r_i = \mathrm{mod}(a \cdot r_{i-1} + b, m)$$

  where a is the multiplier; b is an additive constant; r_0 is the starting value and m is the modulus.

This is very fast and is transportable with a proper choice of a, b and m. But it is not free from sequential correlation and has a short period (at most can have a period of m)

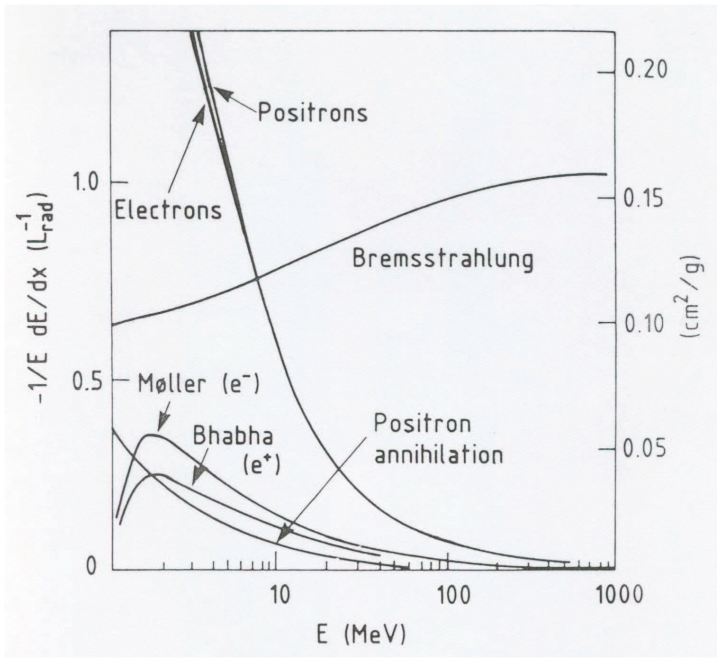
Lower order bits are much less random than the higher order bits. So for a random number in the range of 1–10, it is better to use $1 + \mathrm{int}(10. * \mathrm{r_i})$ rather than $1 + \mathrm{mod}(\mathrm{int}(100000000 * \mathrm{r_i}), 10)$.

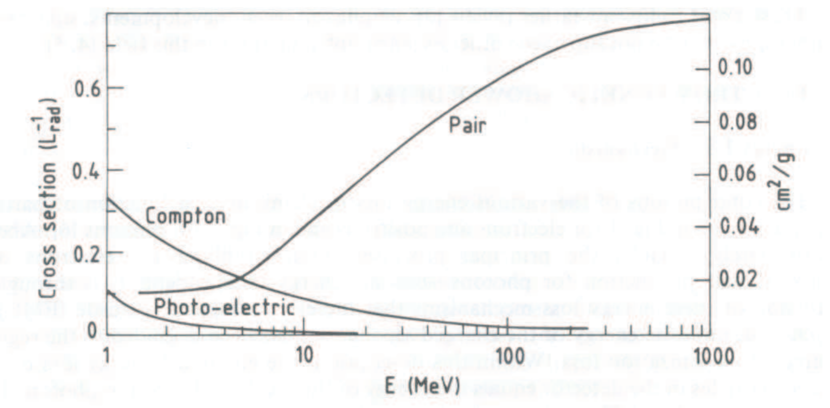
This can be improved by first making a table of random numbers generated using MLCG and then picking randomly from this table.

- Subtraction Method: Subtracting two randomised numbers provide transportable random numbers of rather large period.
  - Initialise a table in a slightly random order with numbers that are not strictly random
  - Randomise them by subtracting a number not especially random
  - Take the difference between two numbers in the table which are apart
  - Update the table position with this number
  - Go to the next sequence of the table

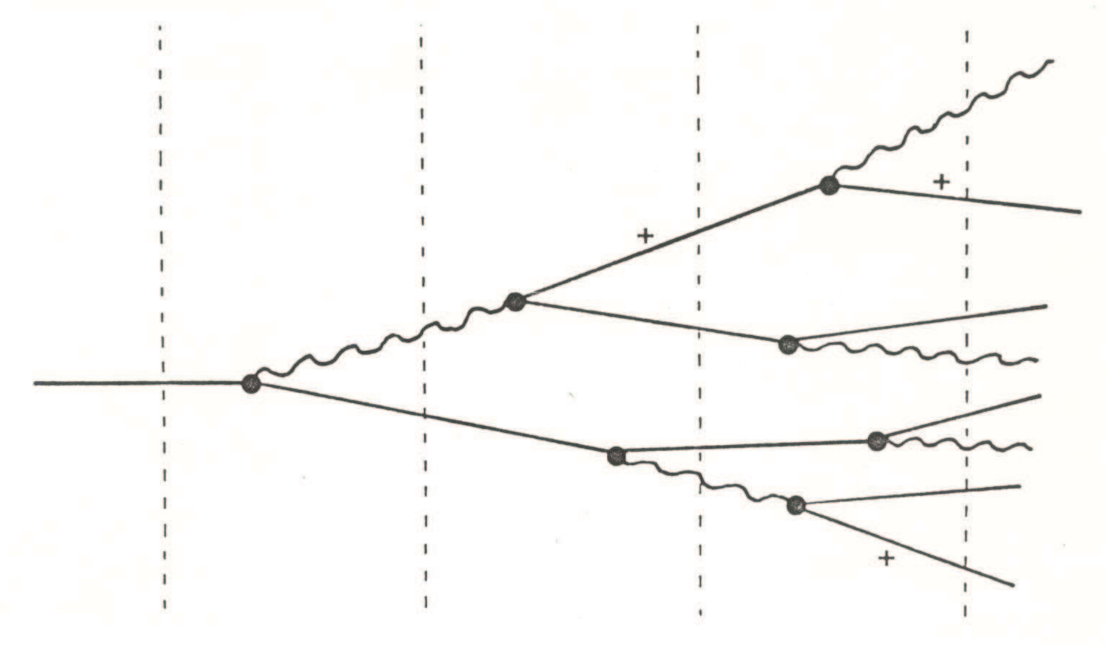
GEANT uses a generator based on subtraction method: RANECU

Electromagnetic Shower





- At energies above 100 MeV, $e^{\pm}$ loses energy mainly through bremsstrahlung emitting photons
- At energies above 100 MeV, photons interact with matter mainly through pair production (generating $e^{\pm}$)
- At high energies, $\sigma(E) \sim$ constant



- ❒ $e^+/e^-/\gamma$ cascades (degrading energy in each stage) mainly through successive bremsstrahlung and pair production steps
- ❒ number of particles in the shower increase till the energy of the particles reach $E \longrightarrow \epsilon_c$, critical energy
- ❒ Beyond this ionisation/excitation takes over and the shower decays out

Energy loss due to radiation is given by:

$$-\frac{\mathrm{dE}}{\mathrm{dx}} = \frac{\mathrm{E}}{\mathrm{L_R}}$$

where $\mathrm{L_R}$ is termed radiation length. $\mathrm{L_R}$ (in g cm^{-2}) can be written down as:

$$-\frac{1}{\mathrm{L_R}} = 4\alpha \mathrm{N_A} \mathrm{Z}^2 \mathrm{r_e}^2 \left[\ln(183 \cdot \mathrm{Z}^{-\frac{1}{3}}) + \frac{1}{18}\right]$$

For large Z (Z $>$ 13), $\mathrm{L_R} \simeq 180\frac{\mathrm{A}}{\mathrm{Z}^2}$ (better than 20%)

Low energy end of shower is generated through collision:

$$-\frac{\Delta \mathrm{E}}{\Delta \mathrm{x}}\Big|_{\mathrm{collision}} = -\frac{\epsilon_\mathrm{c}}{\mathrm{L_R}}$$
$$\text{where } \epsilon_\mathrm{c} \simeq \frac{550}{\mathrm{Z}} \text{ (MeV)}$$

The expression for ϵ_c is accurate to 10% for Z $>$ 13. From these

	Incident $e^{\pm}$	Incident γ
Shower maximum $t_{max}(L_R)$	$\ln\left(\frac{E}{\epsilon_c}\right) - 1$	$\ln\left(\frac{E}{\epsilon_c}\right) - 0.5$
Centre of gravity $t_{Med}(L_R)$ (Half energy absorbed)	$t_{max} + 1.4$	$t_{max} + 1.7$
98% Shower containment $t_{98}(L_R)$	$t_{max} + 4\lambda_{Att}$	$t_{max} + 4\lambda_{Att}$

λ_{Att} corresponds to the exponential $\exp(-t/\lambda_{Att})$ decay of the shower after the maximum

$$\lambda_{Att} \simeq (3.4 \pm 0.5) L_R$$

Transverse shower size is determined by:

- ❑ Typical angle of bremsstrahlung emission at high energy
- ❑ Multiple scattering at low energy

At t_{max}, lateral size is limited to L_R. Multiple scattering makes the low energy particles diverge increasing the lateral size

Shower profile is determined by the Moliere radius ρ_M. 95% of energy deposited is contained in a cylinder of radius $2\rho_M$.

$$\rho_M = 21\frac{L_R}{\epsilon_c} \simeq 7\frac{A}{Z}\, g \cdot cm^{-2}$$

Maximum number of track segments:

$$\mathrm{n_{max}} = \frac{\mathrm{E}}{\eta}$$

where η is the threshold for observing elements. In calorimeters energy is measured by estimating the number of track segments. Thus intrinsic shower fluctuation:

$$\frac{\sigma}{\mathrm{E}} = \frac{\sigma(\mathrm{n_{max}})}{\mathrm{n_{max}}} \sim \mathrm{E}^{-\frac{1}{2}}$$

Hadronic Shower

- ❑ Similar to electromagnetic shower, but greater variety and complexity due to hadronic processes
- ❑ Typical scale is collision length $\lambda = \frac{A}{N_A \rho \sigma_{aA}}$
- ❑ Additional nuclear effects due to evaporation, fission, $\cdots$
- ❑ (Semi)Leptoonic decays cause additional energy loss due to undetected ν's
- ❑ Considerable energy fraction goes to π^0 $(\rightarrow \gamma\gamma)$ and hence there are spots for electromagnetic shower developments

⇒ Large fluctuation in shower development

Shower maximum $t_{max}(\lambda)$	$0.2 \ln E + 0.7$
95% Shower containment $t_{95}(\lambda)$	$t_{max} + 2.5\lambda_{Att}$
Transverse dimension R_{95}	$\sim \lambda$

Slow decay of shower after shower maximum is controlled by

$$\lambda_{Att} \simeq \lambda E^{0.13}$$

Detector Simulation

GEANT - a general toolkit for detector simulation has been available for more than 20 years

GEANT4,the latest arrival. is based on Object Oriented Technology

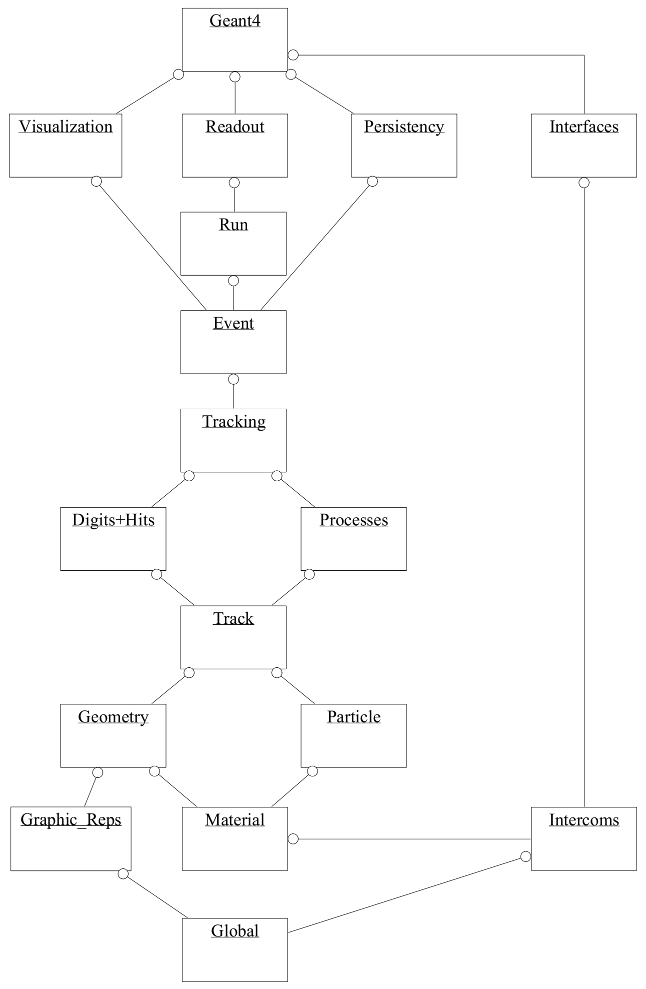
Use the experience of GEANT3

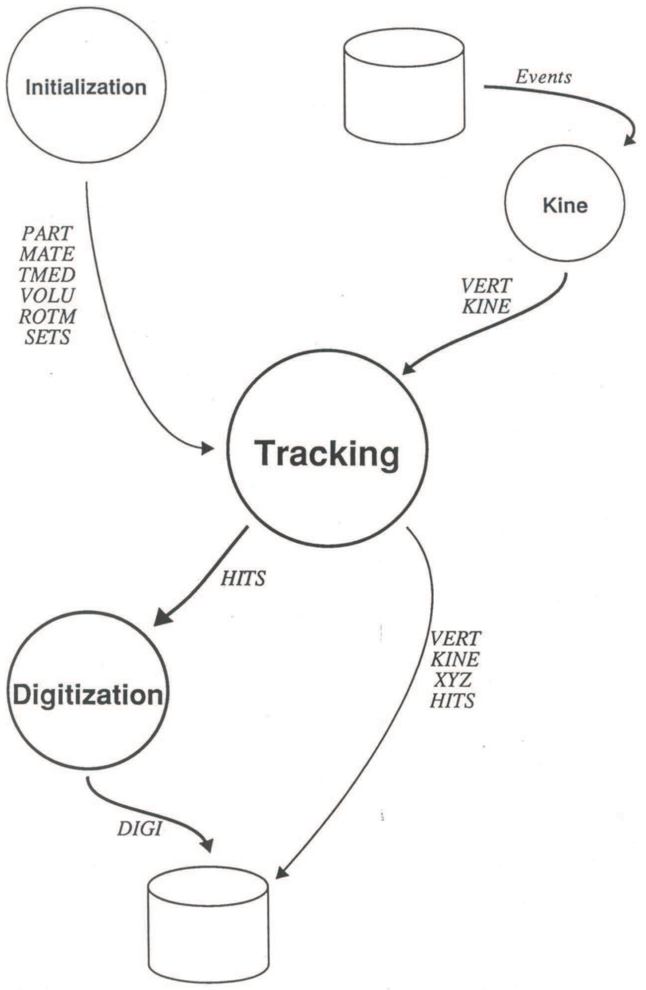
Complete re-analysis, re-design performed

It has a modular structure divided into sub-domains linked with a uni-directional flow of dependencies

Collaboration of $>$ 100 people from all over the world has contributed to this

Production version exists since 1999





Old Paradigm

Data Structures

PART, MATE, TMED, VOLU, ROTM, SETS during initialisation

VERT, KINE as event input

HITS, DIGI, XYZ (also new VERT, KINE) as event output

New Paradigm:

Objects

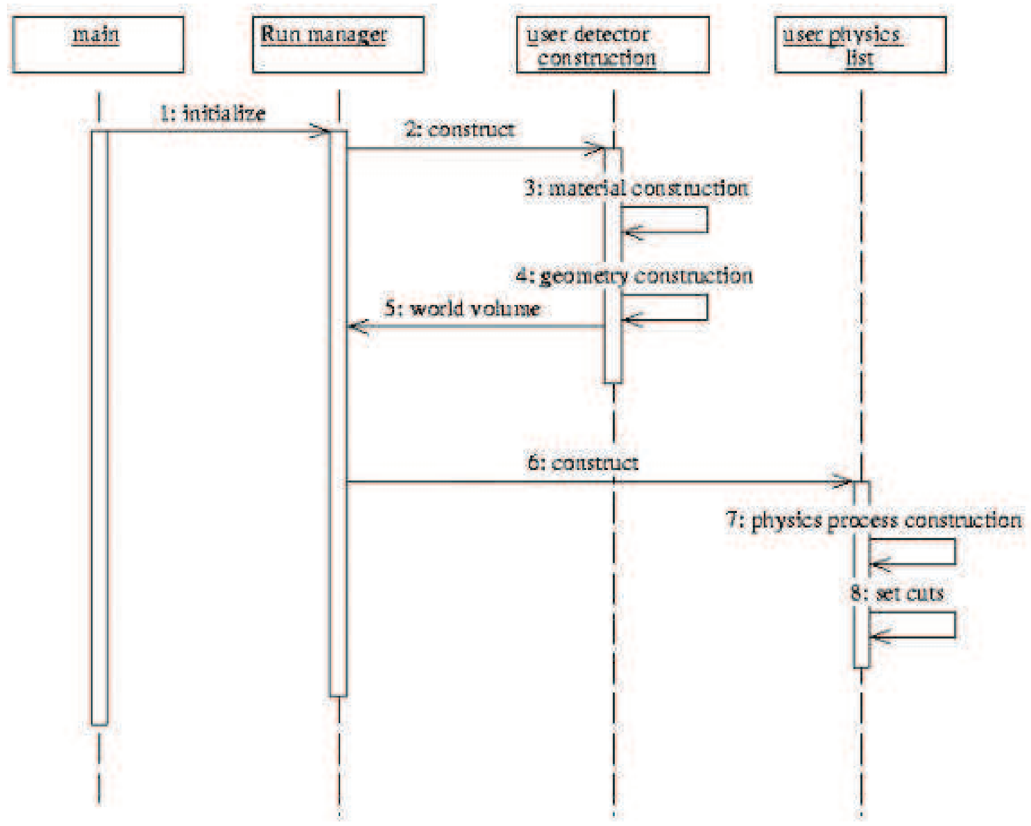
ParticleTable, Material, Solid, LogicalVolume, PhysicalVolume, SensitiveDetector during initialisation

Track as event input

Hit, Digit, Trajectory (also new Track) as event output

Initialisation

- Carry out the construction of detector geometry including all material properties

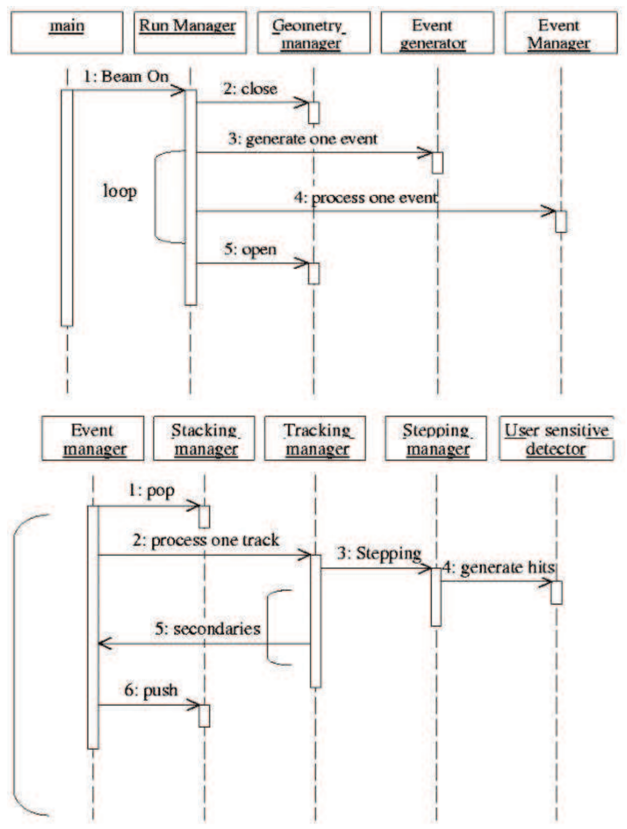


- Complete tables of particles with their static properties
- Initialise physics processes, prepare tables of cross sections, ranges etc in view of approximations and cuts

- Take the input and output attributes and make appropriate action

Event Loop

- ❑ Geometry has to be closed before event loop starts
  - ✧ Done during initialisation step in GEANT3
  - ✧ Done just before the first event in GEANT4



- ❑ Get inputs from event generators; particles with 4-momenta and space-time information at the start
- ❑ Trace particles through detector media and take care of the particle-medium interaction
- ❑ Store energy deposits in media which can transform deposited energy to detectable signal
- ❑ Store relevant parameters which can be used for future analysis
- ❑ Keep track of all particles, primary or secondary

Termination

- ❑ Produce plots/ histograms etc.
- ❑ Prepare run statistics

What Simulation Tool Kit provides?

- ❒ Geometry Modeller
- ❒ Particle table with particle properties
- ❒ Physics processes with cross sections, final state products, kinematics of particles produced in interactions, $\cdots$
- ❒ Stepping through detector material and finding out what is to be done in which step
- ❒ Provision for I/O capability

What User has to provide?

- ❑ Define detector geometry with material definition
- ❑ Identify components of the detector which will generate signals and attributes required to compute these
- ❑ Interface event generators to the simulation tool kit
- ❑ Store information for further usage
- ❑ Convert energy deposits to detector signals

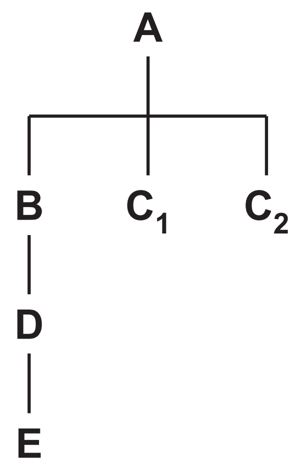
What is special about GEANT3/GEANT4?

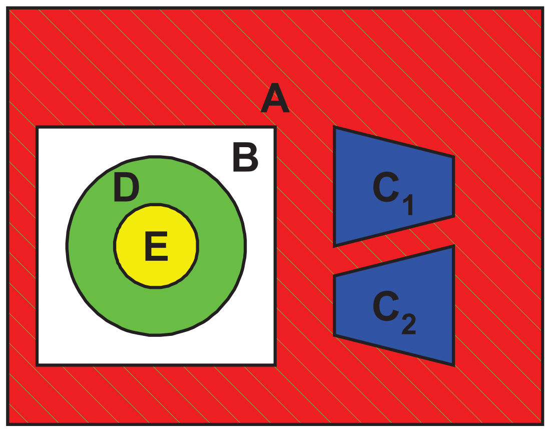
- ❒ Particle transport is automatically taken care of – good interface between geometry and tracking
- ❒ Covers physics over a wide energy region – applicable to space science, high energy and nuclear physics, medical imaging

Modelling of a Detector

Detector is modelled by a Geometrical Shape and its material content ⇒ “Volume”

Several volumes can describe different components of the detector system. Put them together in a hierarchical structure





Composite Volume ≡ Experimental Setup

Defining a Material

Material has a Name, effective Atomic Number and Weight, Density, Radiation (L_R) and absorption (λ) length

Can be defined by specifying the attributes

If L_R, λ are not known but the chemical composition known, one can furnish these information and GEANT can compute the required attributes for the application

GEANT3/GEANT4 has slightly different way of communicating these information

GEANT3 has provision for

- ❑ defining pseudo-elements
  GSMATE (IMAT, NAME, A, Z, DENSITY, RADL, ABSL, UBUF, NW)
- ❑ defining mixtures by weight/atomic proportions
  GSMIXT (IMAT, NAME, A, Z, DENSITY, NCOMPONENT, W)
  with A(1, $\cdots$ NCOMPONENT), Z(1, $\cdots$), W(1, $\cdots$) are atomic weight, number and proportion of 1 $\cdots$ NCOMPONENT components. If $W_i > 0$, the proportion is by weight - otherwise it is by atomic proportion

GEANT4 can in addition define the state, isotopic properties, $\cdots$

- ❑ defining pseudo-elements
  G4Material (Name, Z, A, Density, State, Temperature, Pressure)
- ❑ defining a mixture of elements in atomic or weighted proportion
  G4Element (Name, Symbol, Z, A)
  G4Material (Name, Z, A, Density, State, Temperature, Pressure)
  $\Rightarrow$ AddElement (Element, nAtom)
  $\Rightarrow$ AddElement (Element, fraction)
- ❑ defining a mixture of materials by weighted proportion
  $\Rightarrow$ AddMaterial (Material, fraction)

Tracking Medium

This concept exists only in GEANT3: A material acquires additional properties in the context of tracking

Air in the Tracker (thin gaps) may be different in tracking properties from Air between coil and yoke (large gap)

For tracking in GEANT3, one is required to define for each medium

- ➜ Maximum turning angle in a magnetic field
- ➜ Maximum displacement due to multiple scattering
- ➜ Precision for tracking
- ➜ Maximum fractional energy loss
- ➜ Minimum step size due to multiple scattering, energy loss, magnetic field
- ➜ control for discrete processes (modes, cutoffs)

```
GSTMED (ITMED, NAME, IMAT, ISVOL, IFIELD, FIELDM, TMAXFD,
            STEMAX, DEEMAX, EPSIL, STMIN, UBUF, NW)
GSTPAR (ITMED, PARAMETER_NAME, PARAMETER_VALUE)
```

A Complete Volume

A volume is defined by its shape, dimensional parameters and its material content. In GEANT3, the material content is replaced by the tracking medium. In GEANT4, shape with dimensional parameters is called a Solid and association of a Solid and Material is called a LogicalVolume

GEANT3

A set of simple shapes are supplied. They are adequate to describe any complicated detector setup with reasonable detail

Box, Tube, Trapezoid, Sphere, $\cdots$

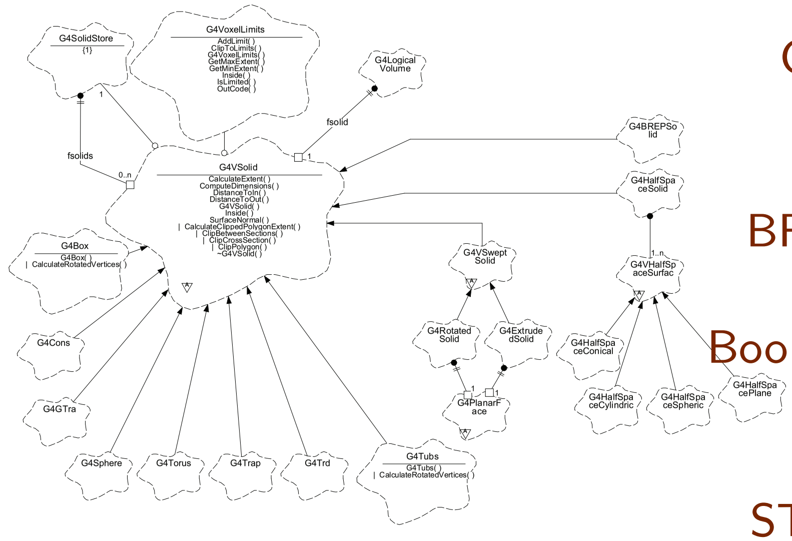
Mostly described as a CSG (Computed Solid Geometry). For certain shapes, internal computation is done using BREP (Boundary Representation)

Dimensional parameters are measured in cm – often half lengths are used. All angles (inclination among surfaces) are measured in degrees

GSVOLU (NAME, SHAPE, ITMED, PAR, NPAR, IVOL)

PAR(1···NPAR) contain the dimensional parameters. The actual meaning differs from shape to shape

GEANT4



There are several ways of defining solids

CSG	G4Box, G4Trd, G4Trap, G4Tubs, G4Cons, G4Sphere, G4Polycone, ···
BREP	G4BrepSolidPcone, ··· (much slower navigation)
Boolean	Solids made out of adding, subtracting, intersecting several solids: G4RotateSolid, ···
STEP	To import from the CAD system

Though internally, a convention of unit system is used, the recommendation is not to remember them and use the units explicitly:

```
double length = 5.*cm;
double angle = 30.*deg;
```

Definition of Detector Setup

To define a set up, one needs to

- ❑ define a Master or World reference system
- ❑ position the various components with respect to each other

GEANT3 uses the same Volume definition for positioned volumes but there are several hooks for positioning. The first created Volume defines the Master Reference System (MRS) and is the container of all Volumes

GEANT4 uses the concept of PhysicalVolume which is a LogicalVolume positioned in a Mother (PhysicalVolume or LogicalVolume) with a translation vector and rotation matrix (optional). For the top level volume (defining the World refernce system) the reference Mother Volume is a Null

One useful way of defining daughter volume is by dividing an existing mother volume into n equal parts along a chosen axis (Cartesian, Cylindrical, Polar)

- ✧ In GEANT3 this operation simultaneously takes care of (a) creation of new volume, (b) positioning the newly created volume in mother.
- ✧ In GEANT4 the creation and positioning is done in two separate steps

When a daughter is positioned inside a mother, the extent inside the mother occupied by the daughter gets filled up with material/medium of the daughter

⇒ Can build up a tree like a Russian doll

GEANT4 Hooks

```
G4PhysicalVolume* volume = new G4PVPlacement (rot,
            G4ThreeVector(xpos*cm,ypos*cm,zpos*cm),
            G4LogicalVolume* current, NAME,
            G4LogicalVolume* mother, false, copyNumber);
```

creates a PhysicalVolume volume by positioning copy copyNumber of LogicalVolume current inside the mother volume mother with a translation vector (G4ThreeVector) and a rotation matrix (G4RotationMatrix* rot)

If one needs to define the rotation matrix exactly in the same way as in GEANT3, namely by specifying θ_i, ϕ_i of the three axes, one needs:

```
G4ThreeVector nAxis(sin(thetaN*deg)*cos(phiN*deg),
                        sin(thetaN*deg)*sin(phiN*deg), cos(thetaN*deg));
G4RotationMatrix* rot = new G4RotationMatrix();
rot→rotateAxes (xAxis, yAxis, zAxis);
rot→invert();
```

GEANT4 Hooks (contd)

For dividing a parent volume, one needs to create LogicalVolume by standard steps (defining Solid and then LogicalVolume) and then position multiple replica through:

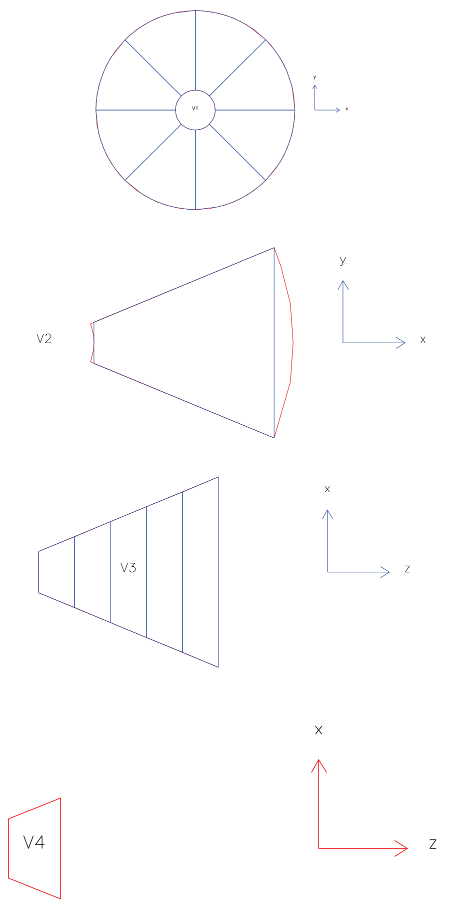
```
G4PhysicalVolume* volume = new G4PVReplica (NAME,
            G4LogicalVolume* current, G4LogicalVolume* mother,
            kAxis, nDivision, width, offset);
```

This is somewhat more general than the example given for GEANT3 – but GEANT3 also has similar functionality

The tree of physical volumes is instantiated at the time of tracking (G4VTouchable) and the TouchableHistory will provide the unique identification of a volume

Example

Construct Geometry of a cylindrical drift chamber with 8 sectors each having 5 cells



- Define volume V1 as a tube with inner and outer radii R1, R2 and half length L
- Divide the tube into 8 parts azimuthally and each section is called V2
- Define a trapezoid of half length L, width $(\mathrm{R2}\cos\frac{\pi}{8} - \mathrm{R1})$ and two edges of dimensions $2\mathrm{R1}\tan\frac{\pi}{8}$, $2\mathrm{R2}\sin\frac{\pi}{8}$. Position this volume V3 inside V2 with proper translation and rotation matrix
- Divide the trapezoid V3 into 5 parts along z-axis. Each new part V4 will be a cell

Sensitive Detector

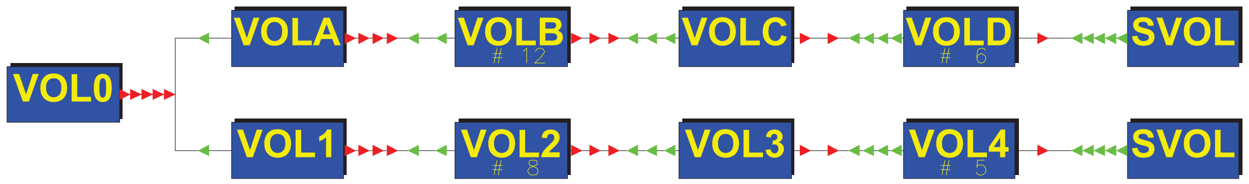
Certain parts of the setup are sensitive to passage of particles through them. Energy deposited due to energy loss of particles in these parts is converted to detectable signals (scintillation light, ionisation in gas detectors, $\cdots$). These parts need to be declared as sensitive so that special action can be taken in these parts at the time of tracking

GEANT3

A Volume is declared sensitive by associating it to the name of a Set, The user has to supply a minimum set of information so that appropriate action can be taken at the time of tracking.

- ✧ The path way in the geometry tree in order to identify the volume uniquely

  Let us consider the volume tree



SVOL is declared sensitive. The unique pathway have 4 nodes: VOL2, VOLB, VOL4, VOLD with a multiplicity as given by $(0 \cdots 8)$, $(0 \cdots 12)$, $(0 \cdots 5)$, $(0 \cdots 6)$

- Type of detection volume
  If it is tracking type or calorimetric type

- Buffer size for storing information

GEANT4

A LogicalVolume is declared Sensitive by attaching a SensitiveDetector to it through a specific method. Since SensitiveDetector is user specific , this gives maximum flexibility and there is no need of supplying additional information. Sensitive Detector can find the unique volume location by examining the TouchableHistory

Hit

During tracking, the user needs to store either transient or persistent information which will be required to compute detector signal on completion of the tracking phase (of all particles in that event) $\Rightarrow$ Hit

GEANT3

It is required to declare at the time of initialisation

- ❑ What are required to be stored as HIT
  Local/Global coordinates, time of flight, energy deposit, $\cdots$
- ❑ What are to be stored as detector response
  branch/crate/channel ID, TDC, ADC, $\cdots$
- ❑ Some user attributes required to compute this response
  TOF Cutoff/Integration time/ $\cdots$

GSDETV (CHSET,CHVOLU,IDTYPE,NWH,NWD,ISET,IDET)

declares the volume CHVOLU as a sensitive detector belonging to the set CHSET of type IDTYPE and with initial buffer size of NWH for hits and NWD for digits

GEANT4

It is not required to declare what HIT is or what a digitised signal would be. There will be corresponding Classes which will be invoked by the SensitiveDetector whenever required. But parameters required for computing these - the Constants - should be in the data base and SensitiveDetector would be using them from the data base

The Sensitive Detector has to provide 3 services:

SD::initialize(G4HCofThisEvent*) at the start of the event (called with the pointer to the Hit Container)

SD::ProcessHits(G4Step*,G4TouchableHistory*) called every time a track makes a step in one of the touchables of a volume belonging to the sensitive detector

SD::EndOfEvent(G4HCofThisEvent*) at the end of the tracking. The user should fill up the hit container with HITs with at least the minimal information regarding the unique cell ID and quantities required to compute DIGITs

Particles

Particle are specified by Name and/or a code

GEANT3 has its special set of codes and the list is somewhat restricted

GEANT4 uses PDG encoding and the list is rather large

Particles are characterised by their static properties: mass, spin, life time, decay modes

In principle, only stable and long lived particles ($\tau > 10^{-13}$ s) needed in the simulation toolkit. The remaining particles need not be created in particle-medium interaction. This, however, sets some limitation on the hadron physics model

GEANT4

- Most commonly used particles are somewhat unique and each such particle is described by a static object
  G4Gamma::GammaDefinition();
  G4Gamma::Gamma();
- Several particles are described through name, PDG code. eg. Gluons, Quarks, Di-Quarks, Leptons, Mesons, Baryons, $\cdots$. These are invoked through
  G4ParticleTable::FindParticle(code/name);
- Some ions or short lived particles are created by the process. They are activated also through special methods of G4ParticleTable

Particles are to be initiated at the same time as the physics process initiation

Interface to Event Generators

At the start of the event user has to provide the complete kinematical information of an Event:

- In GEANT3 subroutine GUKINE has to take care of this
- In GEANT4 user has to declare a PrimaryGeneratorAction as an user action of the RunManager. This has to provide the interface in one of its method GeneratePrimaries

GEANT3

One needs to produce a string of vertices and tracks. Vertex could refer to the beginning or the end of a track (moving particle) and track is a particle with the added information of its kinematics

GSVERT (VERT,BEAM,TARGET,UBUF,NW,NVTX)

creates a vertex NVTX at VERT(1$\cdots$ 3) to be the collision of two tracks BEAM and TARGET

GSKINE (PLAB,PARTICLE,NVTX,UBUF,NW,NTRK)

creates a track NTRK originating from vertex NVTX of particle code PARTICLE and of 3-momentum PLAB

GEANT4

Method GeneratePrimaries creates primary vertics and primary particles

```
new G4PrimaryVertex(particlePosition, particleTime);
```

creates a vertex at a given position (3-vector) and at a time

```
G4PrimaryParticle* particle = new G4PrimaryParticle
                                  (particleDefinition,px,py,pz);
particle→setPolarization(polarization);
particle→setMass(mass);
particle→setCharge(charge);
vertex→setPrimary(particle);
```

creates a primary particle, sets up its mass, charge, ⋯ and associates it to its origin

Processes

When a particle starts its journey through the detector, there will be several competing processes the particle can go through. They are broadly divided into 3 categories:

Transportation: Moving along a straight line (neutral or media with no em field) or along a curve (charged in magnetic field) crossing volume boundaries

Continuous Process: Particle kinematics get modified but the particle retains its identity (continuous energy loss, multiple scattering, $\cdots$)

Discrete Process: Particle undergoes interaction or decay producing new particles and may lose its own identity

These processes are treated differently in a simulation toolkit. The control is done through

GEANT3: Activated or de-activated globally or locally in selected media through data card control

GEANT4: Processes need to be registered at initialisation time and activated during tracking particles

Transportation

It cannot be switched off in GEANT3. Here the Step size is computed and user routine GUSWIM is called which can provide linear transport or transportation in a magnetic field (along a helix or Runge-Kutta integration for inhomogeneous field)

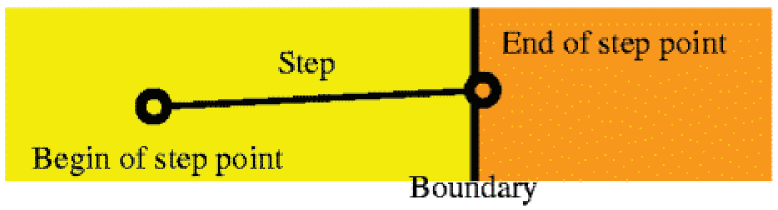
In GEANT4 transportation is not treated any different from other processes. This process has to be registered during initialisation and particles should know how to be transported

Discrete Processes

There are many discrete processes in electromagnetic, weak and strong interactions. Some of these are implemented in the different versions of GEANT. Implementation inside GEANT4 is more complete

In GEANT3 incorporation of a process is controlled by explicit flag with values 0, 1 or 2 signifying the process to be ignored, to be active with generation of secondaries or to be active with no secondary generation

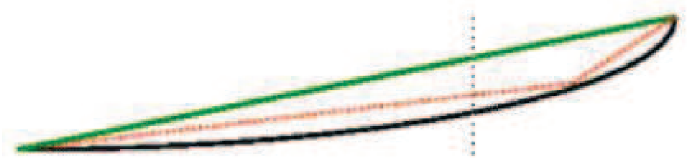
Tracking in GEANT4

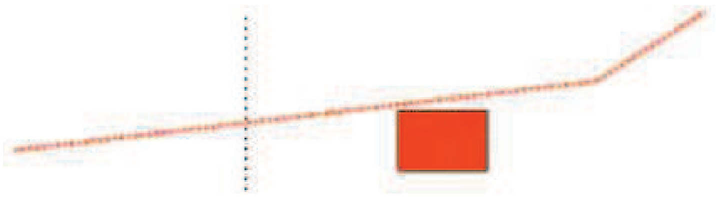


- Here a Step has two points and also ‘delta’ information of a particle (energy loss in that step, time spent in the step, $\cdots$)
- Each point knows the volume. In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume
- It does not make two steps at a boundary
- At each end of step a control is given to the method UserSteppingAction of an object G4UserSteppingAction (or a class derived from it and registered to the ActionManager)
- If it is a sensitive detector, a control is given to the method ProcessHits of the appropriate sensitive detector (for the logical volume)
- Any process (including processes supplied by user) will be asked to take appropriate action AlongStep, PostStep, AtRest

- User can also take action on a track either at the start of tracking or at the end

GEANT4 provides a more careful and detailed approach in swimming particles in an electromagnetic field





- The equation of motion is integrated over a path length using a Runge-Kutta method or some variations of this.
- In a uniform field, analytical solution exists and are used. In a nearly uniform field, perturbation is applied
- The path is calculated using a chosen integration method and then it is broken into linear chord segments that closely approximate the curved path
- The chords are used to interrogate the navigator to find out whether the track has crossed a volume boundary

User Action

Apart from describing the detector in terms of passive and active elements, the user has to take care of certain things during simulation at the time of tracking and post tracking:

- ❑ Take care of the secondaries produced in the discrete processes
- ❑ Store transient Hits at the time of tracking with information to be used for producing detector response later
- ❑ Computer detector response in all sensitive detectors starting from the Hits stored in the event
  - ✧ group hits for individual readout channel
  - ✧ convert energy loss to pulse height; position and time to drift time, · · ·
  - ✧ partition signal into a number of readout channels; generate wire #/pad #/strip #
  - ✧ take care of special effects; non-uniformity, attenuation, · · ·
  - ✧ see effect due to merging: saturation, multi-hit capability, · · ·
  - ✧ add background due to other physical process: electronic noise, radio-activity, beam induced, · · ·
  - ✧ put in detector efficiency, intrinsic resolution, · · ·

User Action in GEANT4

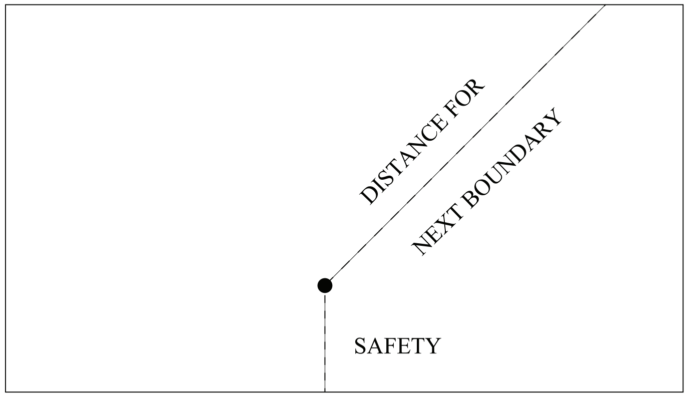
- During tracking, user gets control at several places:
  - For secondaries produced in discrete processes, appropriate action is to be taken in G4UserStackingAction
  - For steps inside a sensitive detector store hits using information from Step and TouchableHistory in the method ProcessHit
  - For deciding to store track information for future use use the methods PreUserTrackingAction and PostUserTrackingAction of G4UserTrackingAction
  - For steps inside any medium, sensitive or not, user can take action in UserStepAction
- The user has to do pretty much the same thing after the completion of tracking as in GEANT3 and this can be interfaced in G4UserAction through the methods BeginOfEventAction and EndOfEventAction

Optimisation

- ❑ Large fraction of detector simulation time is spent in tracking of particles in the detector
- ❑ Big part of the tracking time goes in geometry to find out minimum distance along a direction to the volume boundary

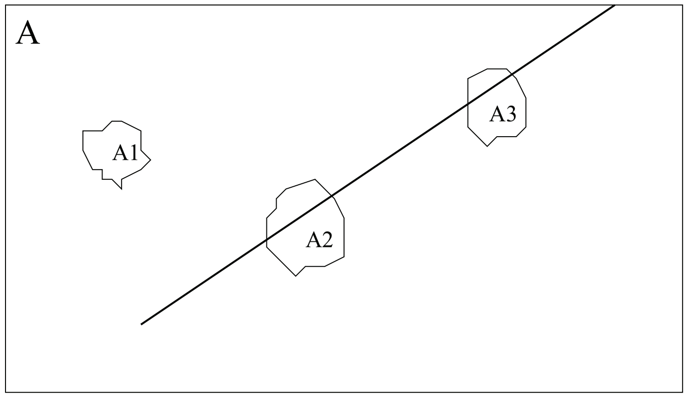
$\Rightarrow$ Use tricks to optimise this

From GEANT side



Introduce the concept of Safety: closest distance to a boundary. It is often easier to compute and on many occasions it is sufficient to know Safety alone (when other limiting distances are smaller). One has to be conservative and need not be exact in computing Safety

From User side



In computing distance to boundary one needs to compute:

- ❑ distance to volume boundary where the point is (A)
- ❑ distances tp the boundaries of all daughter volumes (A1, A2, A3) from the point

Minimum of all real positive roots $\Rightarrow$ answer

- ➢ Avoid putting too many volumes in a Mother

  It is better to have many levels with small multiplicity in each level
- ➢ Position using the Division technique – it is very fast to find out which division the point is associated with

GEANT4 specific

It is done automatically by creating voxels along a given axis to minimise number of daughters per voxel in the search process. This is extended to second or thord dimension to improve upon the ratio of # daughters per voxel. User has some control on the voxelisation procedure (set smartness)

A Minimal Example

```
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "ExN01DetectorConstruction.hh"
#include "ExN01PhysicsList.hh"
#include "ExN01PrimaryGeneratorAction.hh"

int main() {

  // Construct the default run manager
  G4RunManager* runManager = new G4RunManager;

  // set mandatory initialization classes
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  runManager->SetUserInitialization(new ExN01PhysicsList);

  // set mandatory user action class
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);

  // Initialize G4 kernel
  runManager->Initialize();

  // get the pointer to the UI manager and set verbosities
  G4UImanager* UI = G4UImanager::GetUIpointer();
  UI->ApplyCommand("/run/verbose 1");
  UI->ApplyCommand("/event/verbose 1");
  UI->ApplyCommand("/tracking/verbose 1");

  // start a run
  int numberOfEvent = 3;
  runManager->BeamOn(numberOfEvent);

  // job termination
  delete runManager;
  return 0;
}
```

```
#ifndef ExN01DetectorConstruction_H
#define ExN01DetectorConstruction_H 1

class G4LogicalVolume;
class G4VPhysicalVolume;

#include "G4VUserDetectorConstruction.hh"

class ExN01DetectorConstruction : public G4VUserDetectorConstruction {
  public:

    ExN01DetectorConstruction();
    ~ExN01DetectorConstruction();

    G4VPhysicalVolume* Construct();

  private:
    // Logical volumes
    //
    G4LogicalVolume* experimentalHall_log;
    G4LogicalVolume* tracker_log;
    G4LogicalVolume* calorimeterBlock_log;
    G4LogicalVolume* calorimeterLayer_log;

    // Physical volumes
    //
    G4VPhysicalVolume* experimentalHall_phys;
    G4VPhysicalVolume* calorimeterLayer_phys;
    G4VPhysicalVolume* calorimeterBlock_phys;
    G4VPhysicalVolume* tracker_phys;
};
#endif
```

```
#include "ExN01DetectorConstruction.hh"

#include "G4Material.hh"
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "G4PVPlacement.hh"
#include "globals.hh"

ExN01DetectorConstruction::ExN01DetectorConstruction() :
 experimentalHall_log(0), tracker_log(0), calorimeterBlock_log(0),
 calorimeterLayer_log(0), experimentalHall_phys(0), calorimeterLayer_phys(0),
 calorimeterBlock_phys(0), tracker_phys(0) {;}

ExN01DetectorConstruction::~ExN01DetectorConstruction() {}

G4VPhysicalVolume* ExN01DetectorConstruction::Construct() {

  //------------------------------------------------------ materials
  G4double a;  // atomic mass
  G4double z;  // atomic number
  G4double density;

  G4Material* Ar =
  new G4Material("ArgonGas", z= 18., a= 39.95*g/mole, density= 1.782*mg/cm3);
  G4Material* Al =
  new G4Material("Aluminum", z= 13., a= 26.98*g/mole, density= 2.7*g/cm3);
  G4Material* Pb =
  new G4Material("Lead", z= 82., a= 207.19*g/mole, density= 11.35*g/cm3);

  //------------------------------------------------------ volumes
  //------------------------------ experimental hall (world volume)
  //------------------------------ beam line along x axis
  G4double expHall_x = 3.0*m;
  G4double expHall_y = 1.0*m;
  G4double expHall_z = 1.0*m;
  G4Box* experimentalHall_box
  = new G4Box("expHall_box",expHall_x,expHall_y,expHall_z);
experimentalHall_log = new G4LogicalVolume(experimentalHall_box,
                                           Ar,"expHall_log",0,0,0);
experimentalHall_phys = new G4PVPlacement(0,G4ThreeVector(),
                                   experimentalHall_log,"expHall",0,false,0);
//------------------------------ a tracker tube

G4double innerRadiusOfTheTube = 0.*cm;
G4double outerRadiusOfTheTube = 60.*cm;
G4double hightOfTheTube = 50.*cm;
G4double startAngleOfTheTube = 0.*deg;
G4double spanningAngleOfTheTube = 360.*deg;
G4Tubs* tracker_tube = new G4Tubs("tracker_tube",innerRadiusOfTheTube,
                                  outerRadiusOfTheTube,hightOfTheTube,
                                 startAngleOfTheTube,spanningAngleOfTheTube);
tracker_log = new G4LogicalVolume(tracker_tube,Al,"tracker_log",0,0,0);
G4double trackerPos_x = -1.0*m;
G4double trackerPos_y = 0.*m;
G4double trackerPos_z = 0.*m;
tracker_phys = new G4PVPlacement(0,
           G4ThreeVector(trackerPos_x,trackerPos_y,trackerPos_z),
           tracker_log,"tracker",experimentalHall_log,false,0);

//------------------------------ a calorimeter block
G4double block_x = 1.0*m;
G4double block_y = 50.0*cm;
G4double block_z = 50.0*cm;
G4Box* calorimeterBlock_box = new G4Box("calBlock_box",block_x,
                                        block_y,block_z);
calorimeterBlock_log = new G4LogicalVolume(calorimeterBlock_box,
                                           Pb,"caloBlock_log",0,0,0);
G4double blockPos_x = 1.0*m;
G4double blockPos_y = 0.0*m;
G4double blockPos_z = 0.0*m;
calorimeterBlock_phys = new G4PVPlacement(0,
           G4ThreeVector(blockPos_x,blockPos_y,blockPos_z),
           calorimeterBlock_log,"caloBlock",experimentalHall_log,false,0);
//------------------------------ calorimeter layers
```

```
  G4double calo_x = 1.*cm;
  G4double calo_y = 40.*cm;
  G4double calo_z = 40.*cm;
  G4Box* calorimeterLayer_box = new G4Box("caloLayer_box",
                                        calo_x,calo_y,calo_z);
  calorimeterLayer_log = new G4LogicalVolume(calorimeterLayer_box,
                                           Al,"caloLayer_log",0,0,0);
  for(G4int i=0;i<19;i++) {// loop for 19 layers
    G4double caloPos_x = (i-9)*10.*cm;
    G4double caloPos_y = 0.0*m;
    G4double caloPos_z = 0.0*m;
    calorimeterLayer_phys = new G4PVPlacement(0,
               G4ThreeVector(caloPos_x,caloPos_y,caloPos_z),
               calorimeterLayer_log,"caloLayer",calorimeterBlock_log,false,i);
  }
  //------------------------------------------------------------------
  return experimentalHall_phys;
}
```

```
#ifndef ExN01PhysicsList_h
#define ExN01PhysicsList_h 1

#include "G4VUserPhysicsList.hh"
#include "globals.hh"

class ExN01PhysicsList: public G4VUserPhysicsList {
  public:
    ExN01PhysicsList();
    ~ExN01PhysicsList();

  protected:
    // Construct particle and physics process
    void ConstructParticle();
    void ConstructProcess();
    void SetCuts();

};

#endif
```

```
#include "ExN01PhysicsList.hh"
#include "G4ParticleTypes.hh"


ExN01PhysicsList::ExN01PhysicsList() {;}

ExN01PhysicsList::~ExN01PhysicsList() {;}

void ExN01PhysicsList::ConstructParticle() {
  // In this method, static member functions should be called
  // for all particles which you want to use.
  // This ensures that objects of these particle types will be
  // created in the program.

  G4Geantino::GeantinoDefinition();
}

void ExN01PhysicsList::ConstructProcess() {
  // Define transportation process

  AddTransportation();
}

void ExN01PhysicsList::SetCuts() {
  // uppress error messages even in case e/gamma/proton do not exist
  G4int temp = GetVerboseLevel();
 SetVerboseLevel(0);
  //  " G4VUserPhysicsList::SetCutsWithDefault" method sets
  //    the default cut value for all particle types
  SetCutsWithDefault();

  // Retrieve verbose level
  SetVerboseLevel(temp);
}
```

```
#ifndef ExN01PrimaryGeneratorAction_h
#define ExN01PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"

class G4ParticleGun;
class G4Event;

class ExN01PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction {
  public:
    ExN01PrimaryGeneratorAction();
    ~ExN01PrimaryGeneratorAction();

  public:
    void GeneratePrimaries(G4Event* anEvent);

  private:
    G4ParticleGun* particleGun;
};

#endif
```

```
#include "ExN01PrimaryGeneratorAction.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "globals.hh"

ExN01PrimaryGeneratorAction::ExN01PrimaryGeneratorAction() {
  G4int n_particle = 1;
  particleGun = new G4ParticleGun(n_particle);

  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4String particleName;
  particleGun->SetParticleDefinition(particleTable->FindParticle(particleName="geantino"));
  particleGun->SetParticleEnergy(1.0*GeV);
  particleGun->SetParticlePosition(G4ThreeVector(-2.0*m, 0.0, 0.0));
}

ExN01PrimaryGeneratorAction::~ExN01PrimaryGeneratorAction() {
  delete particleGun;
}

void ExN01PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent) {
  G4int i = anEvent->GetEventID() % 3;
  G4ThreeVector v(1.0,0.0,0.0);
  switch(i) {
    case 0:
      break;
    case 1:
      v.setY(0.1);
      break;
    case 2:
      v.setZ(0.1);
      break;
  }
  particleGun->SetParticleMomentumDirection(v);
  particleGun->GeneratePrimaryVertex(anEvent);
}
```

```
#****************************************************************
# Define system dependent variables
#************************************
if (`uname` == "Linux" ) then
  setenv G4SYSTEM    Linux-g++
  setenv myarch Linux
  if ("$?LD_LIBRARY_PATH" == 0) then
    setenv LD_LIBRARY_PATH "/usr/lib:/usr/local/lib:/usr/local/lib/X11:/usr/explorer/lib:/usr/lib/Motif1.2"
  else
    setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:/usr/lib:/usr/local/lib:/usr/local/lib/X11:/usr/explorer/lib:/usr/lib/Motif1.2"
  endif
else if (`uname` == "SunOS" ) then
  setenv G4SYSTEM    SUN-CC
  setenv myarch sun
else
  echo ERROR: Unknown system `uname`! Exiting...
  exit -1
endif
#*****************************************************************
#*** General GEANT4 directory variables
#****************************************************************
#***** where the GEANT4 package is
setenv G4VERS    geant4.5.2.p02
setenv G4INSTALL /afs/cern.ch/sw/geant4/releases/share/${G4VERS}
echo Geant4 installation path set to ${G4INSTALL}

setenv G4LIB /afs/cern.ch/sw/geant4/releases/specific/i386_redhat73/gcc-3.2/${G4VERS}/lib/
echo Geant4 libraries path set to ${G4LIB}

#***** where your compiled files will go to
setenv G4WORKDIR ${PWD}
echo working directory set to ${G4WORKDIR}

#********************  on Linux OSPACE must not be used, on SUN CC-5 not either
if ( $?G4USE_OSPACE )  unsetenv G4USE_OSPACE

#*****************************************************************
#*** Debugging options
```

```
#******************************************************************
#setenv G4DEBUG 1
unsetenv G4DEBUG
if ("$?G4DEBUG" == 1) then
  echo GEANT4 Debug option set
endif
setenv VISDEBUG 1
if ("$?VISDEBUG" == 1) then
  echo GEANT4 vis Debug option set
endif
setenv CPPVERBOSE 1
#if ("$?CPPVERBOSE" == 1) then
  echo Compile verbose option set
endif
if ("$?G4_NO_VERBOSE" == 0) then
  echo Verbose Run mode
else
  echo Quiet Run mode
endif
#******************************************************************
#*** External libraries location:
#******************************************************************
if ( $G4SYSTEM == "Linux-g++"  )  then
 setenv CLHEP_BASE_DIR /afs/cern.ch/sw/lhcxx/specific/redhat73/gcc-3.2/CLHEP/1.8.0.0/
else if ( $G4SYSTEM == "SUN-CC"  )  then
  setenv CLHEP_BASE_DIR /afs/cern.ch/sw/lhcxx/specific/Solaris-7/CC-5.2/3.6.2
endif

if ("$?CLHEP_BASE_DIR" == 1) then
  echo CLHEP base path set to ${CLHEP_BASE_DIR}
  setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${CLHEP_BASE_DIR}/lib"
endif

setenv G4LEVELGAMMADATA       ${G4INSTALL}/data/PhotonEvaporation
setenv NeutronHPCrossSections ${G4INSTALL}/data/G4NDL3.7
setenv G4RADIOACTIVEDATA      ${G4INSTALL}/data/RadiativeDecay
setenv G4LEDATA               ${G4INSTALL}/data/RadiativeDecay/G4EMLOW
setenv LD_LIBRARY_PATH "/usr/local/gcc-alt-3.2/lib:${LD_LIBRARY_PATH}:/afs/cern.ch/sw/lhcxx/specific/redhat61/Mesa/3.2/lib/:${CLHEP_BA

#setenv G4VIS_USE_OPENGLXM    1
#setenv OGLHOME /cmsoo/cms/external/lhcxx/specific/redhat61/Mesa/3.2/
#setenv G4UI_USE_XAW  1
setenv PATH .:${PATH}

## UI & VISUALIZATION
setenv G4VIS_NONE 1
#setenv G4UI_USE_XM 1
#setenv G4UI_USE_XAW 1
#setenv G4VIS_USE_DAWN 1
#setenv G4VIS_USE_OPENGLX 1
#setenv G4VIS_USE_OPENGLXM 1
#setenv G4VIS_USE_VRML 1

setenv G4ANALYSIS_USE 1
setenv OGLHOME /afs/cern.ch/sw/lhcxx/specific/rh73_gcc32/Mesa/3.2
#
```