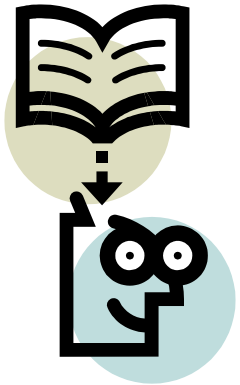
Tecniche sperimentali *Offline*

Simulazione

Ricostruzione

Analisi

Un gruppo di fisici vuole costruire un rivelatore



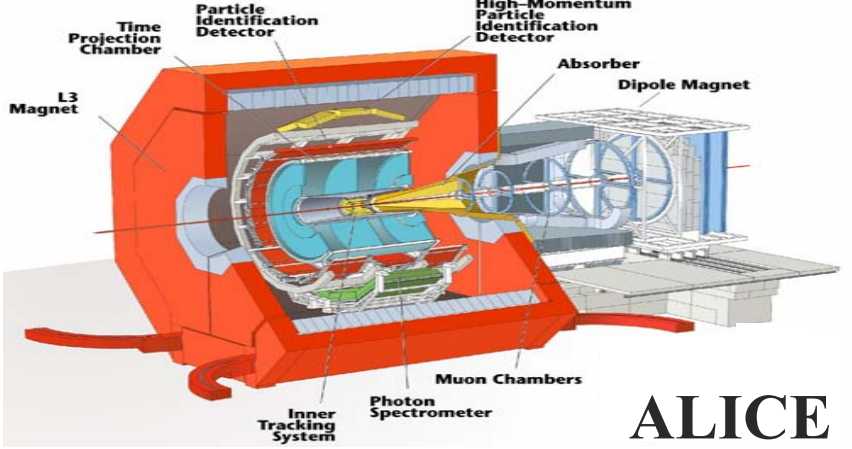
Quale tipo di Fisica ?

Quali tipi di tecniche di rivelazione ?

In che ambiente opereremo ?

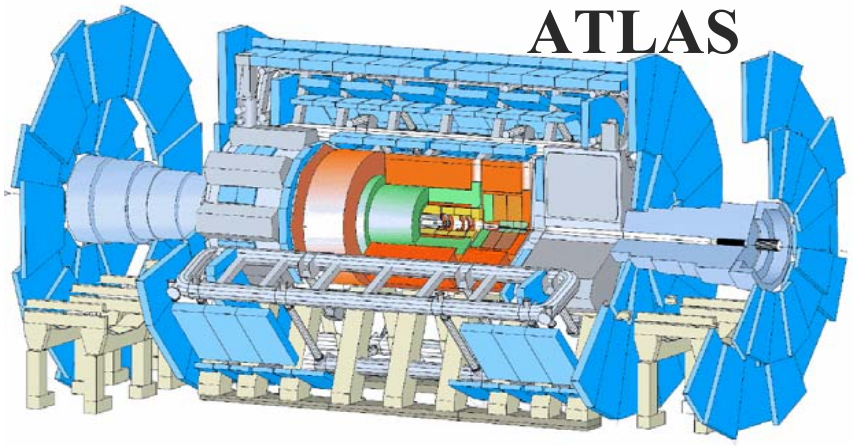
….guardiamoci intorno…

Esperimenti LHC

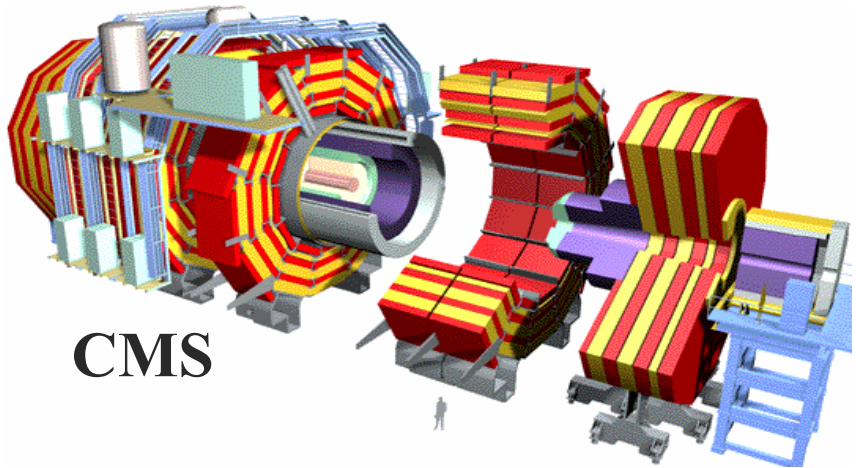


ALICE

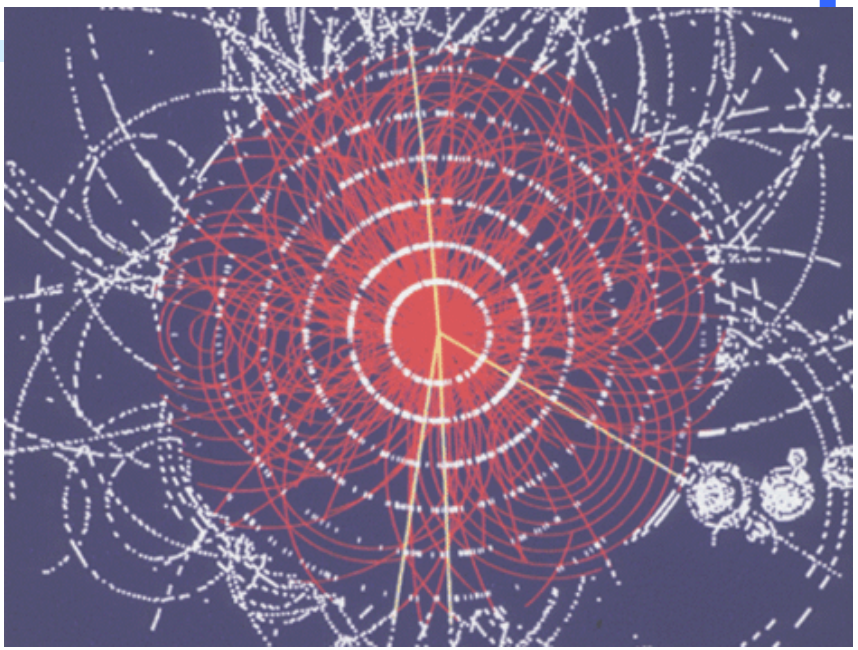
ATLAS

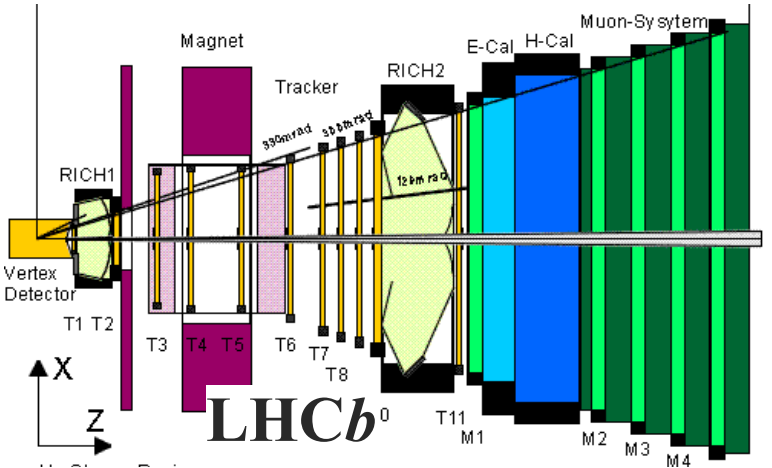


CMS

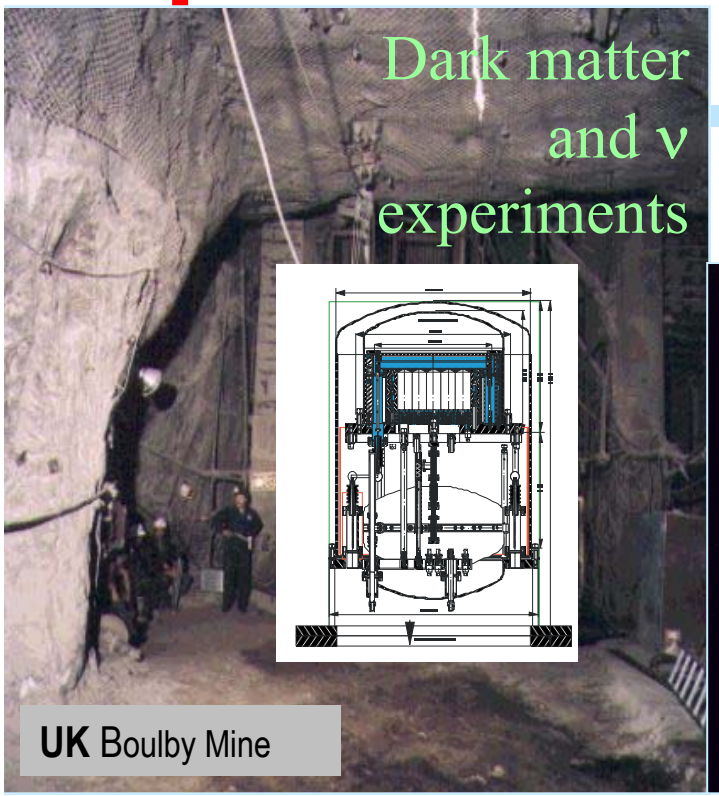


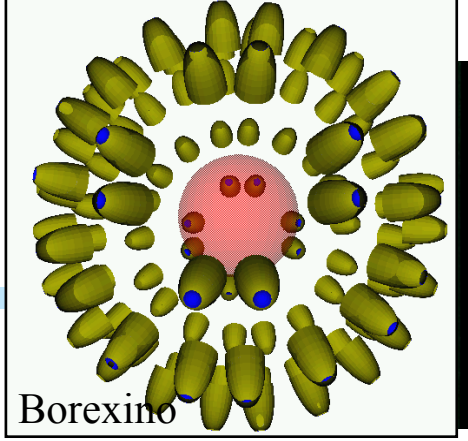
CMS

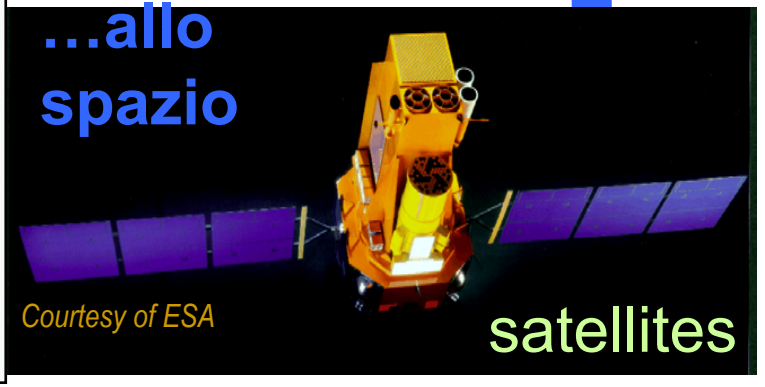


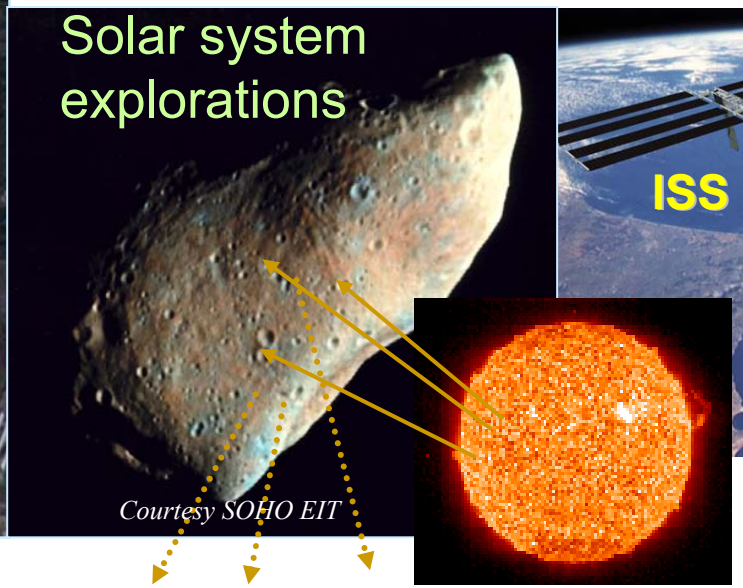


Dai sotterranei profondi…





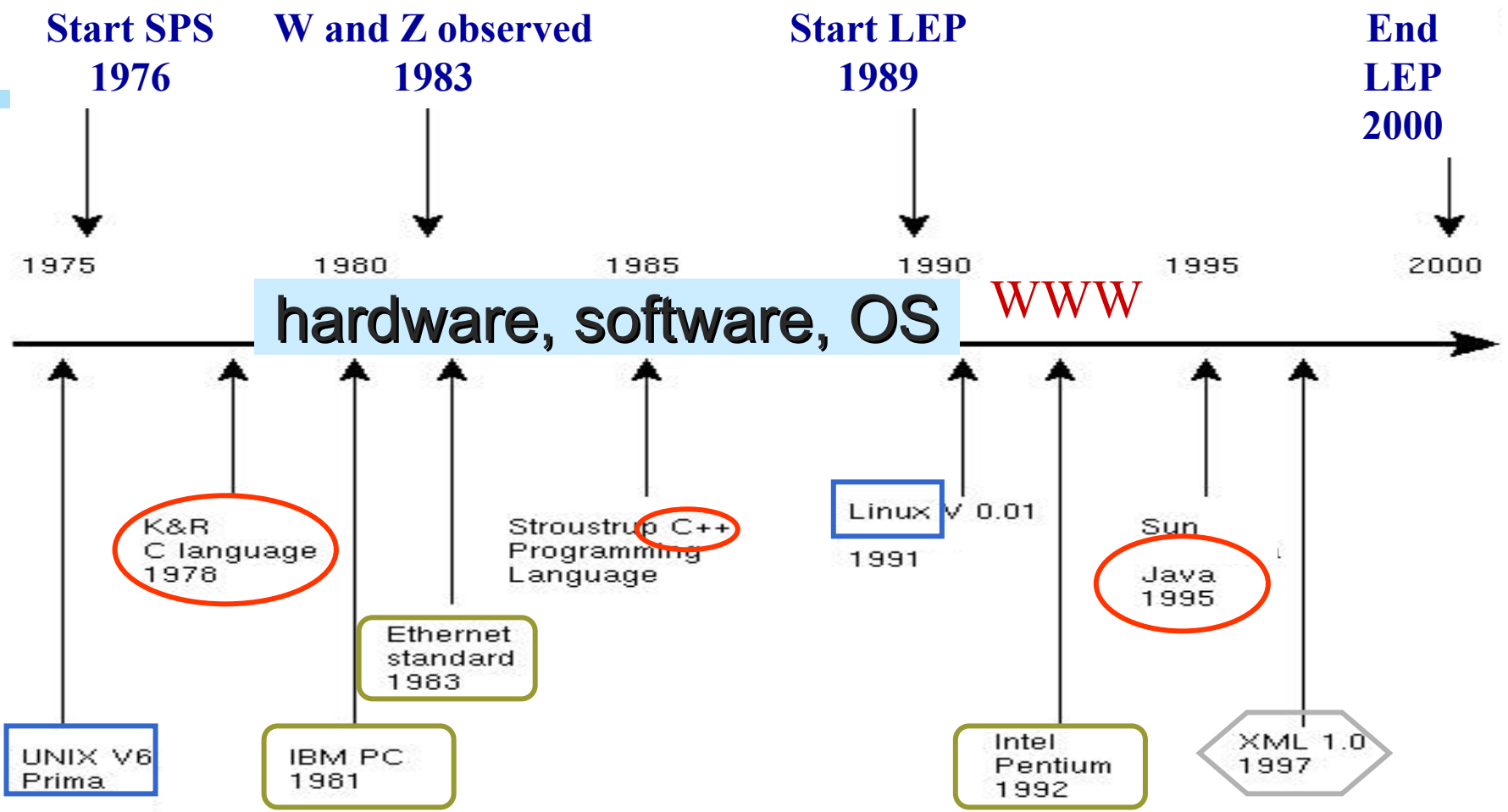




Comportano molte richieste da diversi tipi di applicazioni

Tipi di rivelatori, e ambienti sperimentali — La fisica va dal eV al PeV

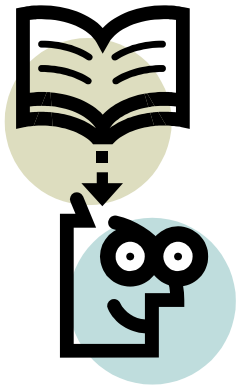
…in un ambiente di calcolo che cambia rapidamente



…e in cui esistono richieste molto diverse

La simulazione

La simulazione gioca un ruolo importante in vari domini e fasi di un esperimento.



- Progettazione dell'apparato
- Valutazione e definizione delle potenzialita' fisiche del progetto (capacita' di scoperta)
- Valutazione dei potenziali rischi del progetto
- Calcolo delle caratteristiche.
- Sviluppo, test ed ottimizzazione del software di ricostruzione e di analisi
- Calcolo e validazione dei risultati di fisica.

La simulazione dei rivelatori

- La simulazione e' una realta' virtuale. Si usa per progettare i rivelatori durante la fase di R&D e per capire la risposta del rivelatore in fase di analisi e studio della fisica.
- Per costruire questa realta' virtuale abbiamo bisogno di un modello di interazione tra particella e materia, ma anche di un modello per descrivere la geometria e i materiali e per propagare le particelle elementari nel rivelatore.
- Abbiamo anche bisogno di descrivere la sensibilita' del rivelatore per riprodurre *raw data*.

Cosa ci serve?

- Apparato sperimentale
- Tracciamento delle particelle nei materiali
- Interazione delle particelle con la materia
- Risposta del rivelatore
- *Run* ed *event control*
- Accessori *(generatore di numeri casuali, le informazioni sulle particelle PDG, costanti fisiche, sistemi di unita' di misura, etc.)*

Da dove prendiamo questi modelli?

Questo e' l'elenco dei codici Monte Carlo presentati alla Conferenza MC2000, Lisbona, Ottobre 2000:

EGS4, EGS5, EGSnrc
MCNP, MCNPX, A3MCNP, MCNP-DSP, MCNP4B
Penelope
Geant3, Geant4
Tripoli-3, Tripoli-3 A, Tripoli-4
Peregrine
MVP, MVP-BURN
MARS

Moltissimi codici non sono distribuiti pubblicamente

NMTC
HERMES
FLUKA
EA-MC
DPM
SCALE
GEM
MF3D
MCU
MORSE
TRAX
MONK
MCBEND
VMC++
LAHET
RTS&T-2000

Pacchetti integrali o pacchetti specialistici?

Pacchetti Specialistici coprono un dominio di simulazione specifico

Pro:

- L'argomento specifico e' trattato in maniera dettagliata.
- A volte il pacchetto si basa su collezioni di dati sperimentali specifici.
- Il codice e' semplice, generalmente e' relativamente facile da installare ed usare.

Contro:

- Tipicamente un esperimento copre molti domini, non solo uno.
- I domini sono spesso interconnessi

Pacchetti Integrali coprono molti domini di simulazione

Pro:

- Lo stesso ambiente fornisce tutte le funzionalita'

Contro:

- E' difficile assicurare una dettagliata copertura di tutte le componenti alla stesso livello di (alta) qualita'
- Monolitico: prendi tutto o niente
- Limitato non permette l'utilizzo di modelli alternativi
- Generalmente difficile da installare e usare
- Difficile mantenimento ed evoluzione

L'approccio *Toolkit*

Un *toolkit* e' un insieme di componenti compatibili

- Ogni componente e' specializzata per una funzionalita' specifica
- Ogni componente puo' essere sviluppato indipendentemente fino ad alti livelli di dettaglio
- I componenti possono essere integrati a vari gradi di complessita'
- I componenti possono lavorare insieme per gestire domini interconnessi
- E' facile fornire (e usare) componenti differenti.
- L'applicazione di simulazione puo' essere adattata dall'utente a seconda delle sue necesita'
- Mantenimento ed evoluzione – sia per i componenti che per l'applicazione dell'utente – e' fortemente facilitato

...ma qual'e il prezzo da pagare?

- L'utente e' investito di una responsabilita' maggiore.
- Deve valutare criticamente e decidere di cosa ha bisogno e cosa vuole usare.

L'approccio *Toolkit*

… e' un esempio di Globalizzazione

Divisione di richieste e funzionalita' da diversi campi.

Dal punto di vista concettuale ricorda la filosofia della programmazione ad oggetti.

La programmazione ad oggetti (OO)

- Consente di programmare tramite aggregati di variabili (chiamati oggetti).
- Si basa sull'incapsulamento delle variabili all'interno di oggetti e sulla creazione di codice che descrive il comportamento interno ed esterno di tali oggetti.
- La progettazione a oggetti parte definendo un universo (dominio del problema) entro il quale verra' creata un'applicazione. All'interno dell'universo vengono identificati gli oggetti. Quindi vengono definiti i contenuti e i comportamenti di tali oggetti e create le classi per ogni tipo di oggetto. I modelli di comportamento di ogni tipo di oggetto vengono incorporati nei metodi. L'ereditarieta' e la composizione specificano le relazione tra le classi.

La simulazione dei rivelatori in ambiente OO

- Geant4 e' un Object-Oriented *toolkit* che fornisce funzionalita' richieste per le simulazioni in HEP ed altri campi.
- Segue i principi dell' *Object-Orientation* cio' significa che vuole essere un simulatore di realta' virtuale:
  - Facile da sviluppare e mantenere
  - Ben modularizzato
  - Leggibile e comprensibile ad altri collaboratori

Cosa possiamo simulare con GEANT4?

Geant4 fornisce un modello per simulare il passaggio di particelle nella materia

- *Ci sono altri tipi di componenti di simulazione, come generatori di eventi fisici, generatori della risposta elettronica dei rivelatori, etc.*
- *Spesso la simulazione di un esperimento complesso e' composta da molte di queste componenti interfacciate l'una con l'altra.*

Parte 1

Concetti di base

Run

Event

Track

Step

Trajectory

Run

- In analogia agli esperimenti reali, un *run* di Geant4 inizia con "*Beam On*".
- All'interno del *run*, l'utente non puo' cambiare
  - Geometria del rivelatore
  - Caratteristiche dei processi fisici
    - il rivelatore e' inaccessibile durante un *run*
- Concettualmente, un *run* e' una raccolta di eventi che dividono condizioni comuni del rivelatore.

Event

- All'inizio del *processing*, un evento contiene particelle primarie. Queste sono spinte in uno *stack*.
- Quando lo *stack* si svuota, il *processing* dell'evento e' completo.
- La classe G4Event rappresenta un evento. Alla fine della fase di *processing* la classe contiene i seguenti oggetti.
  - La lista dei vertici e delle particelle primarie
  - L'insieme delle traiettorie (*optional*)
  - L'insieme degli *Hits*
  - L'insieme dei *Digits* (*optional*)

Track

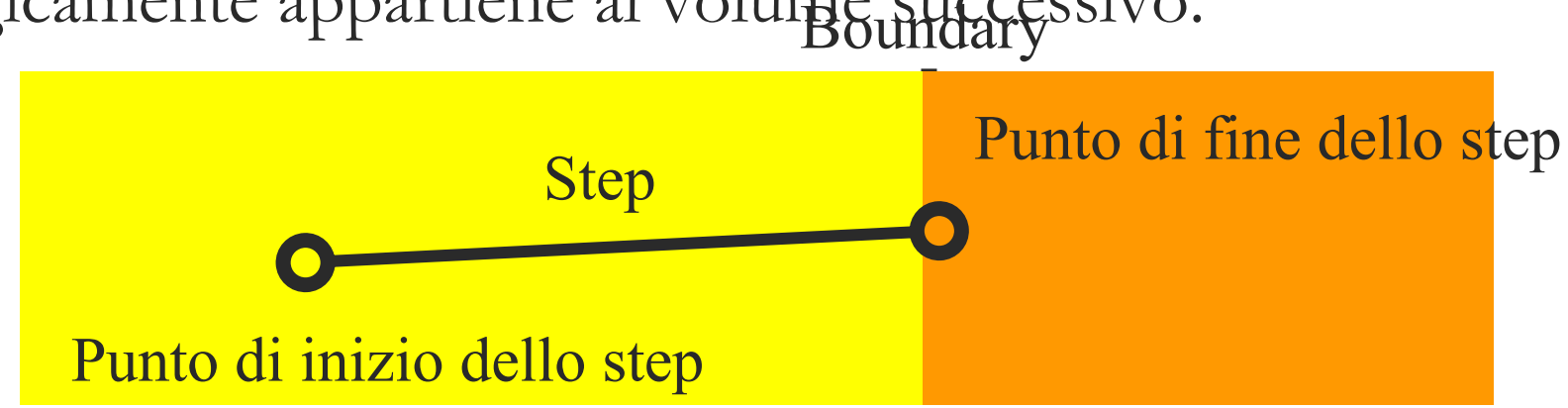
- Una Traccia *(Track)* e' un po' come una fotografia istantanea fatta ad una particella.
- Lo *Step* e' una informazione infinitesima ("delta") della traccia.
  - La Track non e' una raccolta di *steps*.
- Una Track viene cancellata quando
  - Esce dal nostro volume (universo)
  - Scompare (es. decade)
  - Ha energia cinetica nulla e non ci sono processi ``a riposo"
  - L'utente decide di ucciderla

Track

- Una track e' formata da tre strati di classi oggetto.
  - G4Track
    - Posizione, volume, lunghezza di traccia, ToF globale
    - ID di se' stessa e della traccia madre (che l'ha generata)
  - G4DynamicParticle
    - Momento, energia, tempo locale, polarizzazione
    - Canale di decadimento prefissato
  - G4ParticleDefinition
    - Comune a tutte le G4DynamicParticle dello stesso tipo
    - Massa, tempo di vita, carica, altre quantita' fisiche
    - Tabella contenente i modi di decadimento

Step

- Uno *Step* ha due punti e le informazioni infinitesimali ("*delta*") della particella (energia persa nello *step*, Time-of-Flight dello *step*, etc.).
- Ogni punto conosce il volume in cui si trova. Nel caso in cui uno *step* sia limitato dal contorno di un volume, il punto finale fisicamente e' sul bordo e logicamente appartiene al volume successivo.



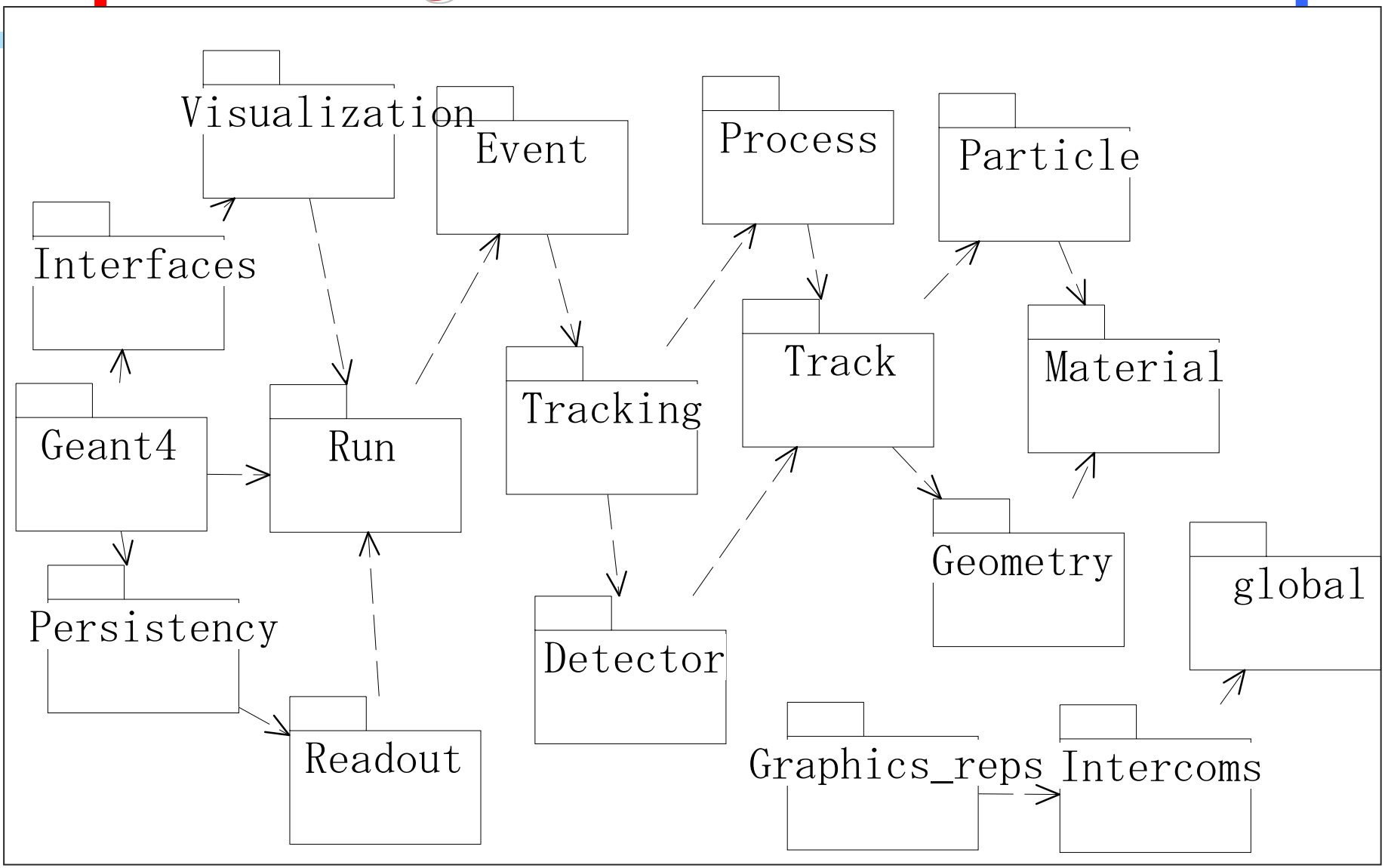
Trajectory

- Una traiettoria e’ un archivio contenente la storia di una traccia. Conserva l’informazione di tutti gli *step* compiuti dalla traccia come oggetti della classe: G4TrajectoryPoint.
- Generalmente non e’ consigliabile salvare le traiettorie delle particelle secondarie generate in uno sciame a causa del consumo di memoria.
- L’utente puo’ creare la sua propria classe traiettoria a partire dalle classi G4VTrajectory e G4VTrajectoryPoint per registrare ogni informazione aggiuntiva necessaria alla simulazione.

Parte 2

Struttura del *toolkit* Geant4

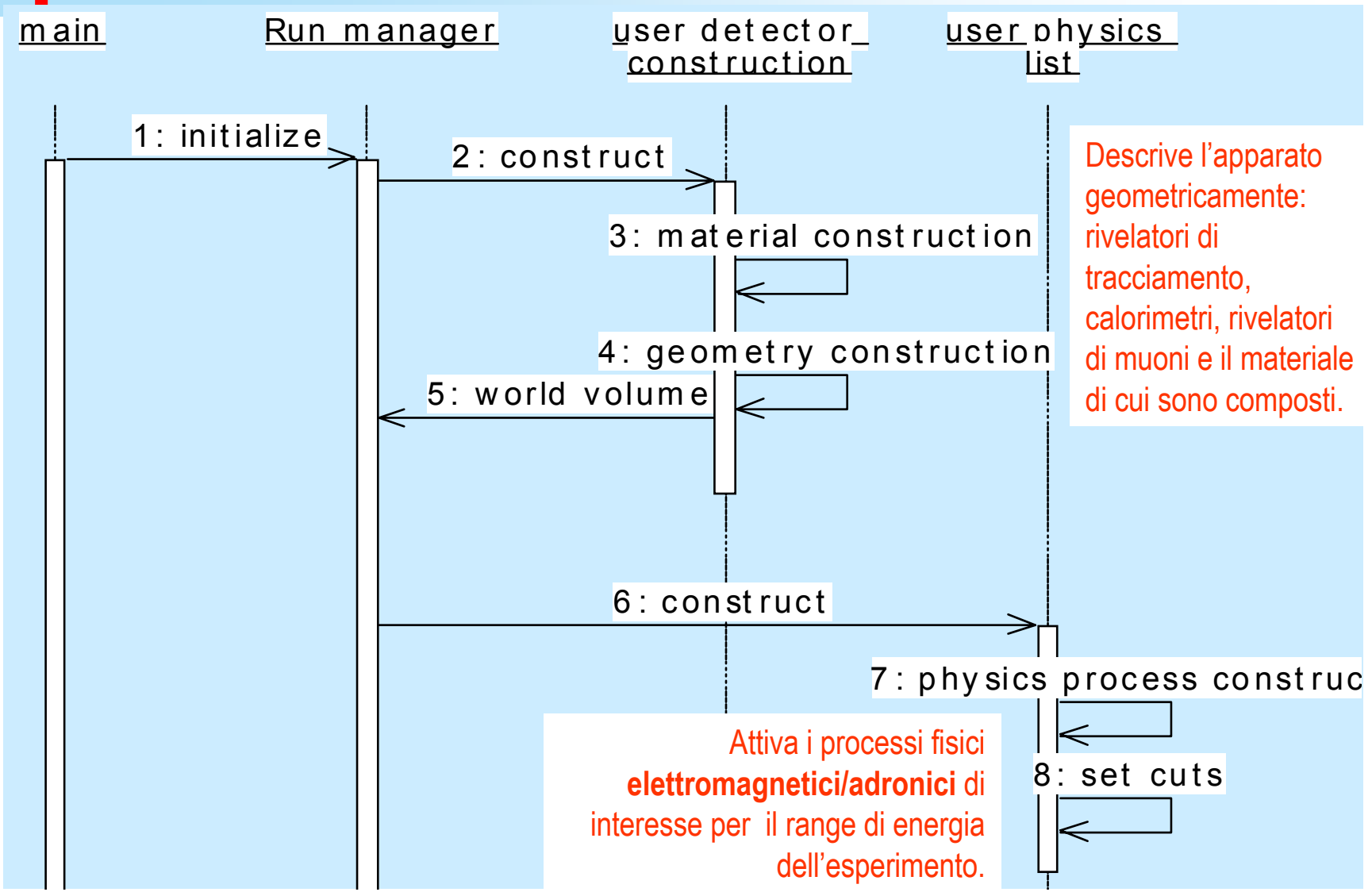
Struttura globale di Geant4



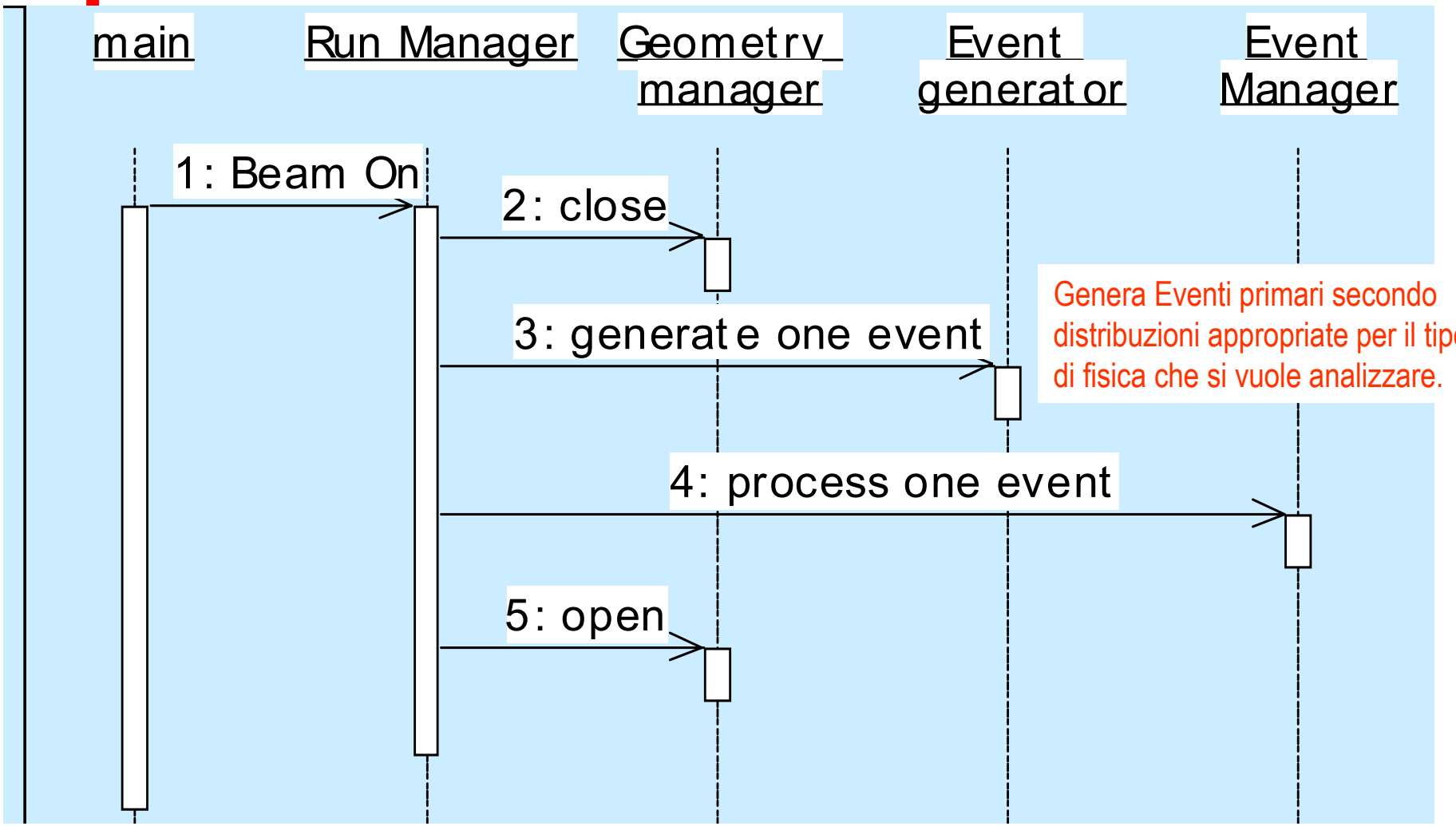
Come funziona Geant4

- Inizializzazione
  - Costruzione di materiali e geometria
  - Costruzione di particelle, processi fisici e calcoli di tabelle di sezioni d'urto.
- "Beam-On" = "Run"
  - Chiude la geometria e la ottimizza
  - Loop sugli eventi

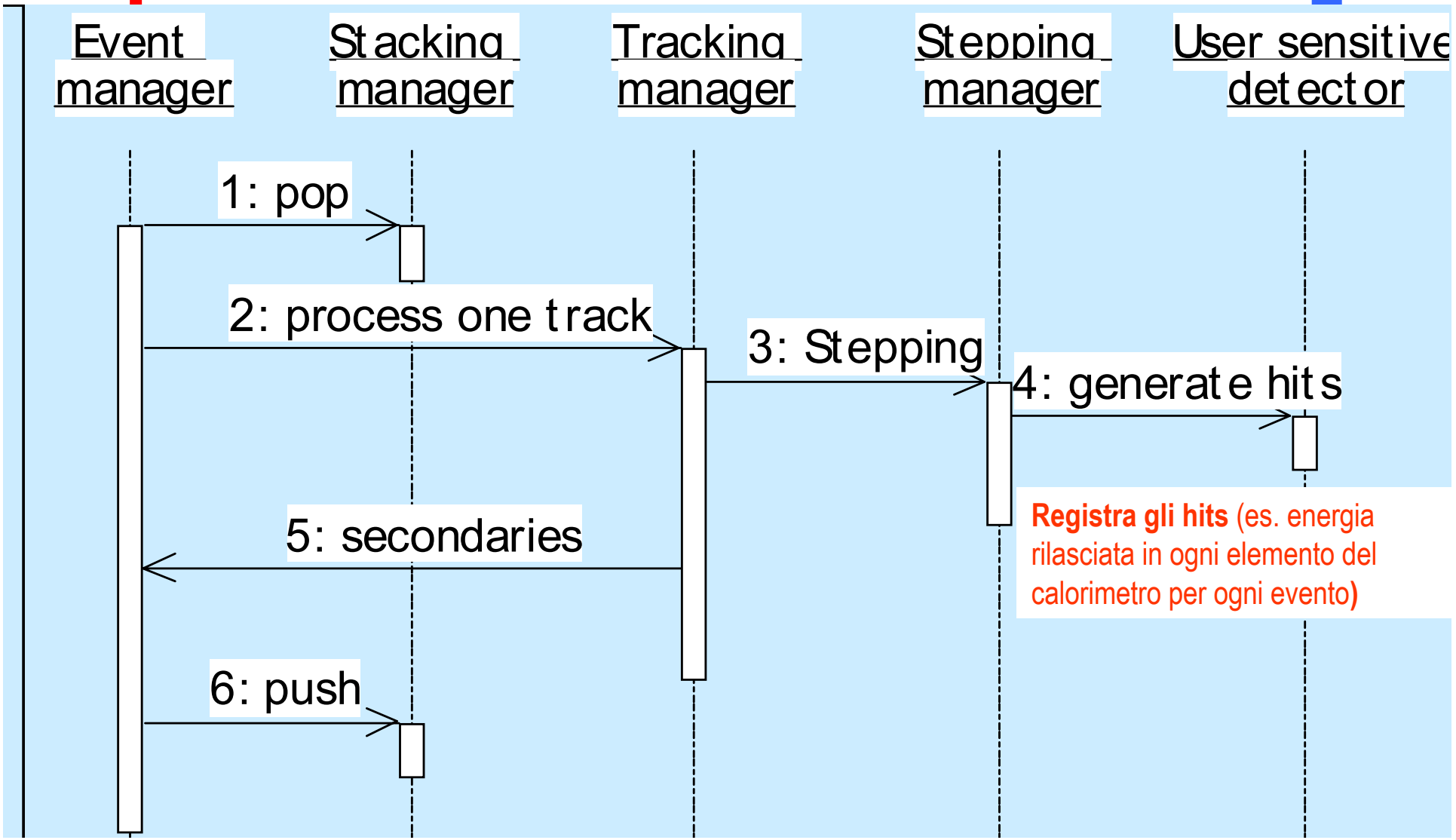
Inizializzazione



Beam On


main
Run Manager
Geometry_ manager
Event_ generator
Event Manager
1: Beam On
2: close
3: generate one event
4: process one event
5: open
Genera Eventi primari secondo distribuzioni appropriate per il tipo di fisica che si vuole analizzare.

Event processing



Classi utente

- Classi di inizializzazione
  - Usate in fase di inizializzazione
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList
- Classi d'azione
  - Usate in fase di loop sugli eventi
    - G4VUserPrimaryGeneratorAction
    - G4UserRunAction
    - G4UserEventAction
    - G4UserStackingAction
    - G4UserTrackingAction
    - G4UserSteppingAction

Parte 3

Descrizione del rivelatore

Materiali

Geometria del rivelatore

Volumi sensibili

Hits

Definizione di Materiali

- Si possono definire diversi tipi di materiali:
  - isotopi <> G4Isotope
  - elementi <> G4Element
  - molecule <> G4Material
  - composti e miscele <> G4Material
- Attributi associati:
  - temperatura, pressione, stato, densita'

Costruzione del rivelatore

- Deriva le classi concrete da una classe di base astratta `G4VUserDetectorConstruction`.
- Implementa il metodo `Construct()`:
  - Questo metodo va modularizzato in accordo ad ogni componente del rivelatore o del sottorivelatore:
    - Construisci tutti i materiali necessari
    - Definisci forme/solidi richiesti per descrivere la geometria
    - Costruisci e posiziona i volumi secondo la geometria del tuo rivelatore
    - Identifica i rivelatori sensibili e specifica i volumi dei rivelatori che vanno associati a questi.
    - Associa il campo magnetico a regioni del rivelatore
    - Definisci attributi di visualizzazione per gli elementi del rivelatore.

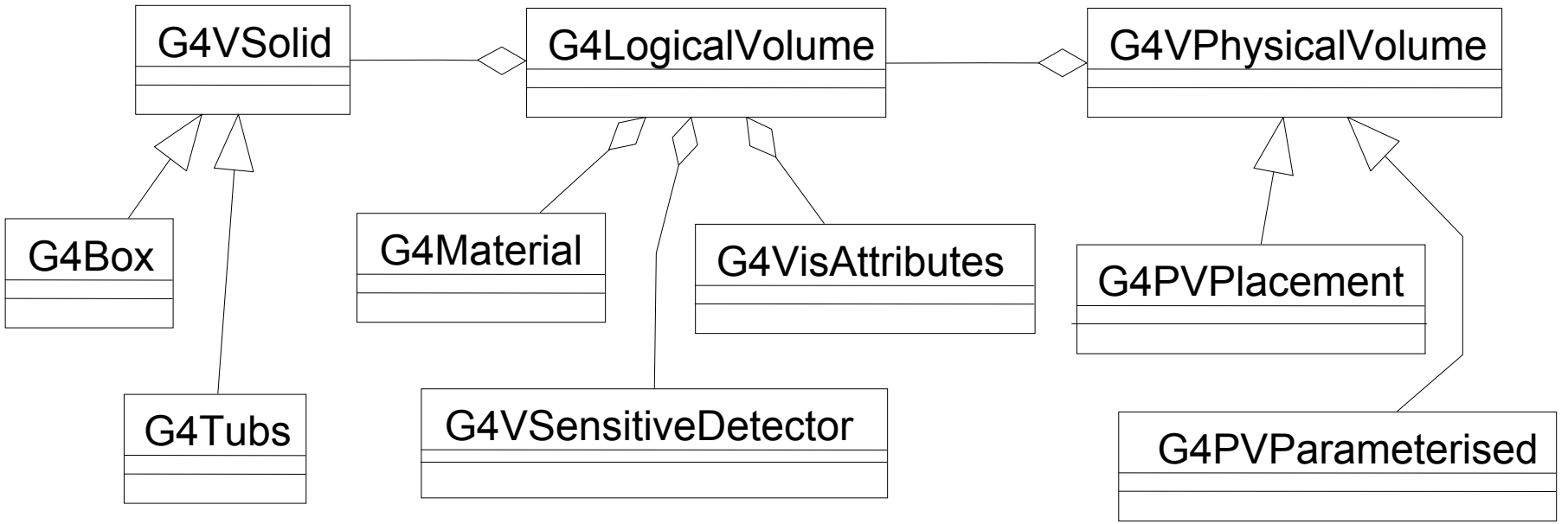
Crea un Volume

- Comincia con Shape & Size
  - Box 3x5x7 cm, sphere R=8m
- Aggiungi le proprieta':
  - material, B/E field,
  - Rendilo sensibile
- Posizionalo in un altro volume
  - in un posto
  - Ripetutamente usando una funzione

- *Solido*
- *Volume-Logico*
- *Volume-Fisico*

La geometria del rivelatore

- Tre stadi concettuali
  - **G4VSolid** -- *forma, dimensioni*
  - **G4LogicalVolume** –*Attributi fisici dei volumi, materiale, sensibilita', etc.*
  - **G4VPhysicalVolume** -- *posizione, rotazione*



Definizione della geometria

- Strategia di base

```
 // World
G4Box* solidWorld = new G4Box("World",                       // Il suo Nome
                  1.2*m,1.2*m,1.2* m);                     // La sua Dimensione

G4LogicalVolume* logicWorld = new G4LogicalVolume(solidWorld, //Il suo Solido
                                  defaultMaterial,            //Il suo Materiale
                                  "World");                   //Il suo Nome

G4VPhysicalVolume* physiWorld = new G4PVPlacement(0,          //Senza rotazione
                                     G4ThreeVector(),         //Posizione (0,0,0)
                                  "World",                    //Il suo Nome
                                  logicWorld,                 //Il suo volume logico
                                  0,                          //Il suo volume ``madre''
                                  false,                      //nessuna operazione booleana
                                  0);                         //numero di copia
```

- Deve esistere un unico volume fisico che rappresenta l'area sperimentale e che contiene tutti gli altri:
  - Il volume World

Definizione della geometria

- Strategia di base

```
 // Calorimeter
G4Box* solidCalor = new G4Box("Calorimeter",           // Il suo Nome
                  1.*m,1.*m,1.* m);                    // La sua Dimensione

G4LogicalVolume* logicCalor = new G4LogicalVolume(solidCalor, //Il suo Solido
                                  AbsorberMaterial,           //Il suo Materiale
                                "Calorimeter");               //Il suo Nome

G4VPhysicalVolume* physiCalor = new G4PVPlacement(0,          //Senza rotazione
                                G4ThreeVector(),     //Posizione (0,0,0)
                                  "Calorimeter",              //Il suo Nome
                                  logicCalor,                 //Il suo volume logico
                               physiWorld,                    //Il suo volume ``madre''
                                  false,              //nessuna operazione booleana
                                  0);                 //numero di copia
```

- All'interno del volume fisico si possono posizionare altri volumi: es. *Calorimeter*

G4LogicalVolume

```
G4LogicalVolume(G4VSolid *pSolid, G4Material
   *pMaterial,
                  const G4String& name,
                  G4FieldManager *pFieldMgr=0,
                  G4VSensitiveDetector *pSDetector=0,
                  G4UserLimits *pULimits=0);
```

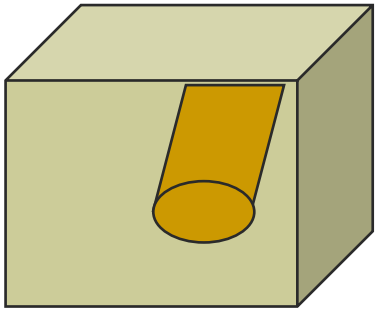
- Contiene tutte le informazione del volume eccetto la posizione:
  - Forma e dimensione (G4VSolid)
  - Materiale, sensibilita', attributi di visualizzazione
  - Posizione dei volumi ``figli''
  - Campo Magnetico
  - Parametrizzazione dello sciame
- Volumi fisici dello stesso tipo possono dividere lo stesso volume logico.

G4VPhysicalVolume

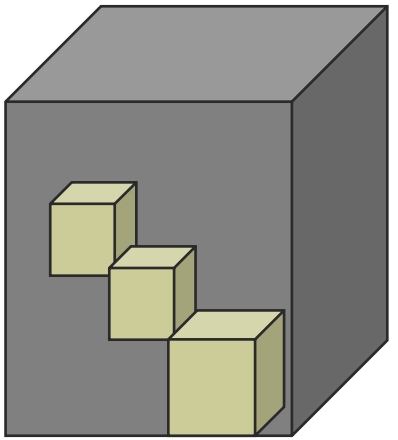
- G4PVPlacement 1 Posizionamento = 1 Volume
  - Un volume puo' essere posizionato una volta in un volume madre
- G4PVParameterized 1 Parametrizzazione = Molti Volumi
  - Parametrizzazione con un numero di copia
    - Forma, dimensione, materiale, posizione e rotazione possono essere parametrizzate, implementando una classe concreta `G4VPVParameterisation`.
    - La parametrizzazione (nell'attuale versione) puo' essere usata solo per volumi che
      - non hanno volumi ``figli''
      - sono identici in grandezza & forma.
- G4PVReplica 1 Replica = Molti Volumi
  - Divide un volume in pezzi piu' piccoli (se ha simmetria)

Volumi Fisici

- *Placement:* Un volume posizionato
- *Repeated:* Un volume posizionato molte volte
  - puo' essere costituito da un numero N di volumi
  - Replica: semplice ripetizione, simile ad una divisione (alla G3)
  - Parametrizzazione
- Un volume **mother** puo' contenere
  - **molti** volumi posizionati **OR**
  - **un** volume *repeated*



posizionamento

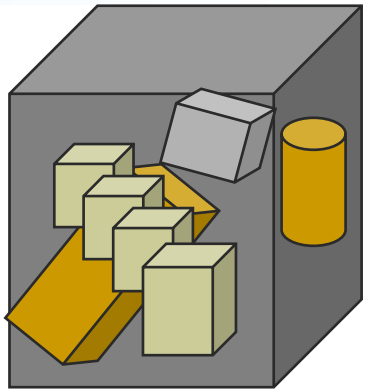


ripetizione

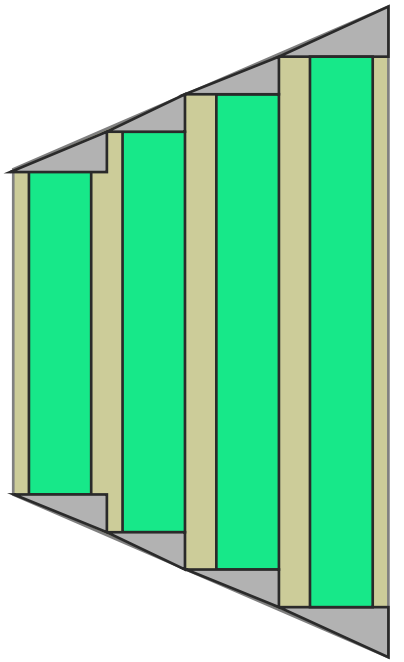
```
G4PVPlacement(G4RotationMatrix *pRot,
              const G4ThreeVector &tlate,
              const G4String &pName,
              G4LogicalVolume *pLogical,
              G4VPhysicalVolume *pMother,
              G4bool pMany,
              G4int pCopyNo);
```

- Singolo volume posizionato rispetto al volume *mother*
- In un sitema di riferimento ruotato e traslato rispetto al sistema di coordinate del volume

Volumi Fisici Parametrizzati



- Le funzioni scritte dall'utente definiscono:
  - La grandezza del solido (dimensione)
    - Function `ComputeDimensions(…)`
  - Dove e' posizionato (trasformazione)
    - Function `ComputeTransformations(…)`
- Optional:
  - Il tipo di solido
    - Function `ComputeSolid(…)`
  - Il materiale
    - Function `ComputeMaterial(…)`
- Limitazioni:
  - Non sono permessi volumi ``figlia" a meno di casi speciali



- Applicazioni mediche
  - Si misura il materiale nei tessuti animali
  - G4geometry: Volumi a materiale variabile
- Rivelatori Complessi
  - Molte volumi sono ripetuti.
  - Regolari o irregolari

G4PVParameterised

```
G4PVParameterised(const G4String& pName,
                  G4LogicalVolume* pLogical,
                  G4VPhysicalVolume* pMother,
                  const EAxis pAxis,
                  const G4int nReplicas,
                  G4VPVParameterisation *pParam);
```

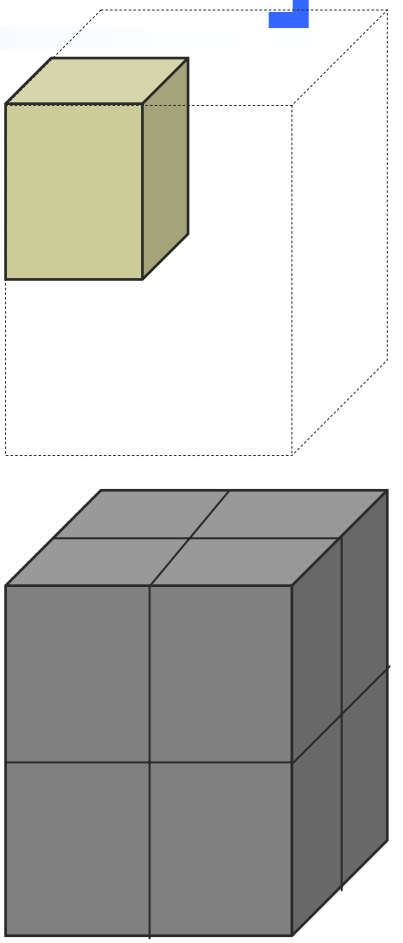
- Replica il volume *nReplicas* volte usando la parametrizzazione `pParam`, entro il volume madre *pMother*
- Il posizionamento delle repliche avviene lungo l'asse cartesiano specificato

Pameterisation: example - 1

```
G4VSolid* solidChamber = new G4Box("chamber", 100*cm, 100*cm, 10*cm);
G4LogicalVolume* logicChamber =
   new G4LogicalVolume(solidChamber, ChamberMater, "Chamber", 0, 0, 0);
G4double firstPosition = -trackerSize + 0.5*ChamberWidth;
G4double firstLength = fTrackerLength/10;
G4double lastLength  = fTrackerLength;
G4VPVParameterisation* chamberParam =
   new ChamberParameterisation( NbOfChambers, firstPosition,
                                ChamberSpacing, ChamberWidth,
                                firstLength, lastLength);
G4VPhysicalVolume* physChamber =
   new G4PVParameterised( "Chamber", logicChamber, physTracker,
                          kZAxis, NbOfChambers, chamberParam);
```

Volumi Fisici Replicati

- Il volume *mother* e' suddiviso in repliche, tutte di forma e dimensioni simili.
- Rappresenta molti elementi di rivelatore che differiscono solo nel loro posizionamento.
- Le repliche possono essere lungo:
  - Gli assi cartesiani (X, Y, Z) – perpendicolari all'asse di replicazione
    - Il sistema di coordinare e' al centro di ogni replica.
  - Asse radiale (Rho) – sezioni di tubi centrati all'origini e non ruotati
    - Il sistema di coordinate e' uguale a quello *mother*
  - Asse Phi (Phi) – sezioni phi di tubi.
    - Il sistema di coordinate e' ruotato in maniera tale che l'asse X biseca l'angolo sotteso da ogni volume.



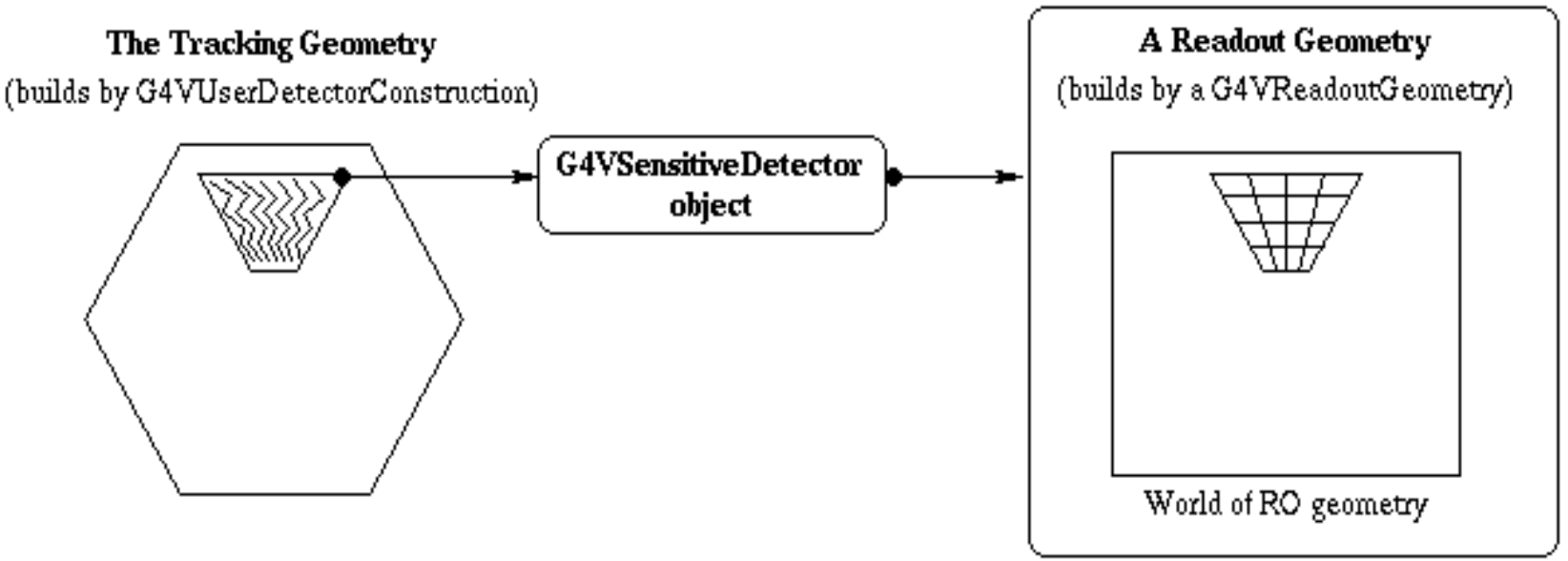
repeated

Replica: esempio!

```
G4double tube_dPhi = 2.* M_PI;
G4Tubs* tube =
  new G4Tubs("tube", 20*cm, 50*cm, 30*cm, 0., tube_dPhi*rad);
G4LogicalVolume * tube_log =
  new G4LogicalVolume(tube, Ar, "tubeL", 0, 0, 0);
G4VPhysicalVolume* tube_phys =
  new G4PVPlacement(0,G4ThreeVector(-200.*cm, 0., 0.*cm),
                    "tubeP", tube_log, world_phys, false, 0);
G4double divided_tube_dPhi = tube_dPhi/6.;
G4Tubs* divided_tube =
  new G4Tubs("divided_tube", 20*cm, 50*cm, 30*cm,
             -divided_tube_dPhi/2.*rad, divided_tube_dPhi*rad);
G4LogicalVolume* divided_tube_log =
  new G4LogicalVolume(divided_tube, Ar, "div_tubeL", 0, 0, 0);
G4VPhysicalVolume* divided_tube_phys =
  new G4PVReplica("divided_tube_phys", divided_tube_log, tube_log,
                  kPhi, 6, divided_tube_dPhi);
```

Geometria di lettura

- La geometria di lettura e' una geometria virtuale e artificiale che puo' essere definita in parallelo alla geometria reale del rivelatore.
- La geometria di lettura e' un *optional*. Puo' essercene piu' di una. Ognuna di queste deve essere associata ad un rivelatore sensibile.



Hits

- Si possono salvare diversi tipi di informazione implementando la propria classe di Hits concreti.
- Per esempio:
  - Posizione e tempo dello step.
  - Momento ed energia della traccia
  - Rilascio di energia dello step
  - Informazione geometrica.
  - O una combinazione delle precedenti

Digits

- Il Digit rappresenta l'uscita del rivelatore (per esempio il conteggio dell'ADC/TDC, il segnale di trigger).
- Un Digit e' prodotto da uno o piu' hits tramite un'implementazione completa derivata da G4VDigitizerModule.

Rivelatore sensibile

- Un volume logico diventa sensibile se ha un puntatore alla classe concreta derivata da G4VSensitiveDetector.
- Un rivelatore sensibile o
  - Costruisce uno o piu' oggetti hits o
  - Accumula valori agli hit esistenti

usando informazioni date da un oggetto G4Step

Parte 4

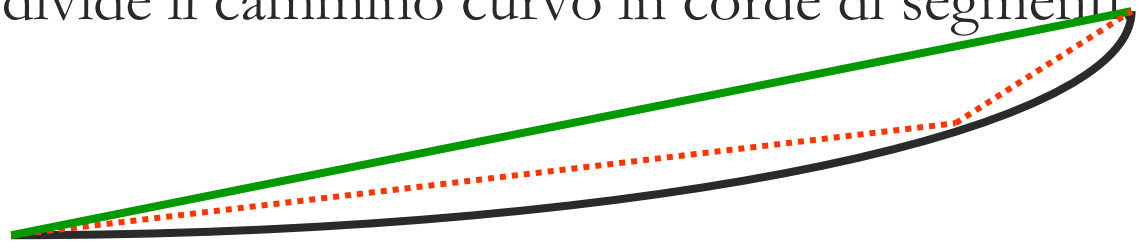
Campo magnetico

Campo Magnetico

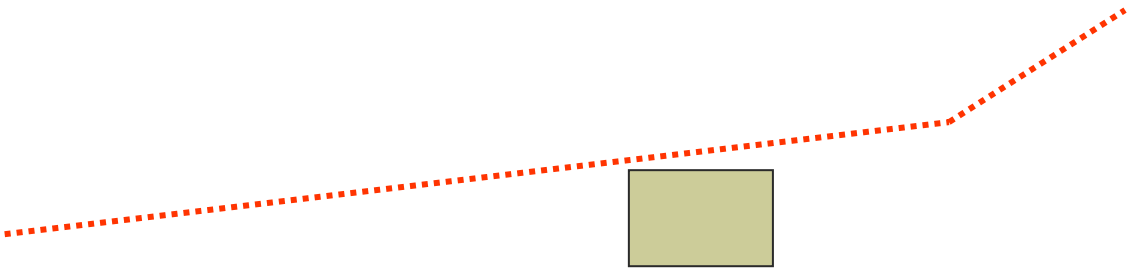
- Per propagare una particella all'interno di un campo (magnetico, elettrico o entrambi), dobbiamo risolvere l'equazione del moto della particella nel campo
- Si utilizza il metodo Runge-Kutta per l'integrazione delle equazioni differenziali del moto.
  - Sono disponibili diversi steppers Runge-Kutta.
- In casi specifici si possono usare altre soluzioni:
  - In un campo uniforme utilizzando l'equazione analitica.

Campo magnetico

- Usando il metodo per calcolare il moto della traccia in un campo Geant4 divide il cammino curvo in corde di segmenti

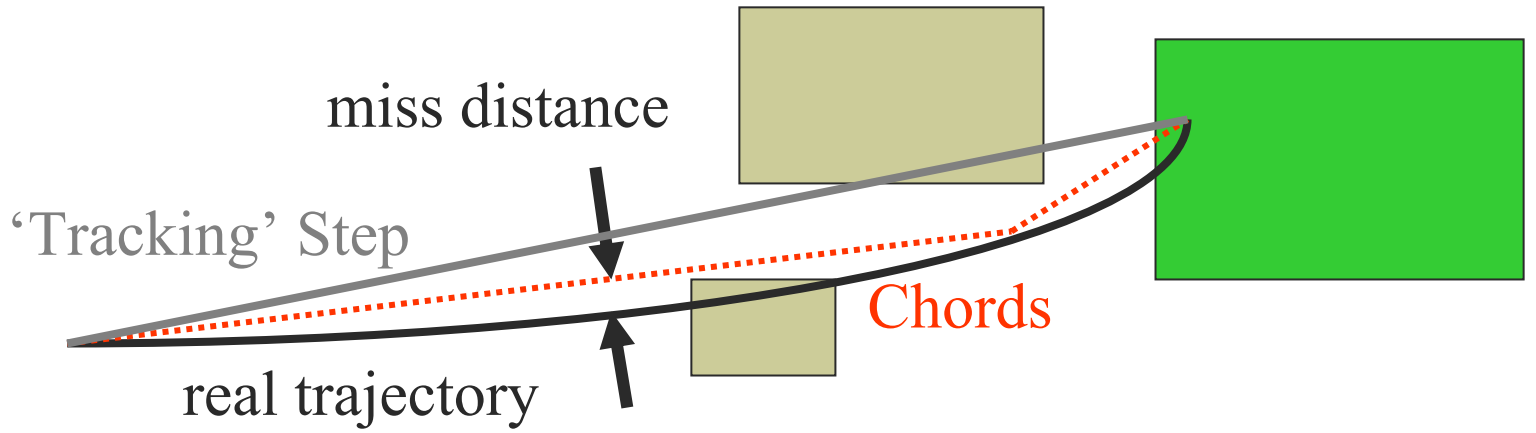


- Si determinano cosi' i segmenti di corda che si approssimano al cammino curvo.
- Si utilizzano le corde per chiedere all'oggetto *Navigator* se la traccia ha o non ha attraversato un volume



Step ed accuratezza

- Si puo' definire l'accuratezza dell'intersezione con un volume,
  - Definendo un parametro noto come *miss distance*
    - Questo parametro e' una misura dell'errore in approssimazione con cui una traccia interseca un volume
    - Default "miss distance"= 3 mm.
- Uno step fisico puo' essere generato da piu' corde
  - In alcuni casi puo' consistere di pezzi di elica che girano su se stessi.



Campo Magnetico: esempio

Classe campo Magnetico

- Campo uniforme :
  - Usa un oggetto del G4UniformMagField class

```
#include "G4UniformMagField.hh"
#include "G4FieldManager.hh"
#include  "G4TransportationManager.hh“

G4MagneticField* magField= new
   G4UniformMagField( G4ThreeVector(1.0*Tesla, 0.0,
   0.0 );
```

- Campo Non-uniforme :
  - Crea una classe concreta derivante da G4MagneticField

Campo Magnetico: esempio

Di a Geant4 di usare il campo

- Trova il Field Manager

```
G4FieldManager* globalFieldMgr=
  G4TransportationManager::
    GetTransportationManager()
                ->GetFieldManager();
```

- Assegna il campo per questo FieldManager,

```
globalFieldMgr->SetDetectorField(magField);
```

- E crea un ChordFinder.

```
globalFieldMgr->CreateChordFinder(magField);
```

In pratica: esempio

```
G4VPhysicalVolume*  ExN04DetectorConst
{
   //------------------------------------
   //  Magnetic field
   //------------------------------------

   static G4bool fieldIsInitialized =
   if(!fieldIsInitialized)
   {
      ExN04Field* myField = new ExN04Fi
      G4FieldManager* fieldMgr
         = G4TransportationManager::GetT
         ->GetFieldManager();
      fieldMgr->SetDetectorField(myFiel
      fieldMgr->CreateChordFinder(myFie
      fieldIsInitialized = true;
   }
```

Crea un proprio campo

Crea una classe con un metodo chiave che calcola il valore del campo nel punto

Point [0..2] position
Point[3] time

```
void ExN04Field::GetFieldValue(
   const double Point[4],
     double *field) const
{
  field[0] = 0.;
  field[1] = 0.;
  if(abs(Point[2])<zmax &&
   (sqr(Point[0])+sqr(Point[1]))<rmax_sq
   )
  { field[2] = Bz; }
  else
  { field[2] = 0.; }
}
```

Campi globali e locali

- Un *field manager* viene associato al '*world*'
  - Assegnato in G4TransportationManager
- Altri possono essere sovrascritti
  - Associando un field manager al volume logico
    - Questo e' propagato ai volumi figlia

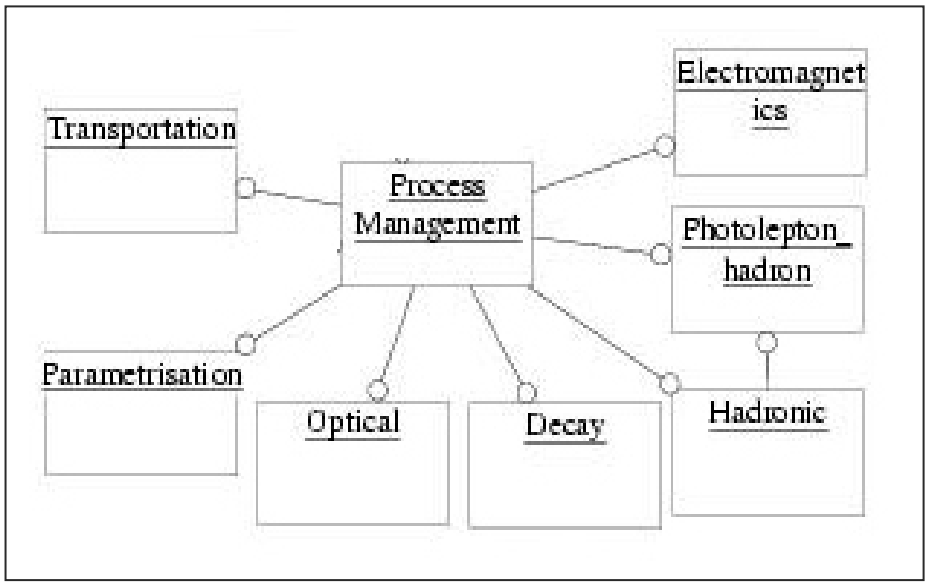
```
G4FieldManager* localFieldMgr=
    new G4FieldManager(magField);
logVolume->setFieldManager(localFieldMgr,
  true);
```

Dove *true* permette di estendere il campo a tutti i volumi contenuti.

Parte 5
Fisica Elettromagnetica in Geant4

Processi

- I processi descrivono come una particella interagisce con il materiale, con la traccia e con il volume
- 3 tipi base
  - Processo da fermo
    - (decadimento a riposo)
  - Processo continuo
    - (ionizzazione)
  - Processo discreto
    - (decadimento in volo)
- Il trasporto e' un processo
  - Interagente con i contorni del volume



Il processo che richiede il piu' piccolo cammino di interazione limita lo step

Fisica Elettromagnetica

Ha a che fare con:

- Elettroni e positroni
- γ, X-ray e fotoni ottici
- muoni
- Adroni carichi
- ioni

energy loss {

multiple scattering
Cherenkov
transition radiation
ionisation
Bremsstrahlung
annihilation
photoelectric effect
Compton scattering
Rayleigh effect
γ conversion
e^+e^- pair production
refraction
reflection
absorption
scintillation
synchrotron radiation
fluorescence
Auger effect *(in progress)*

↘ Estensioni ad alta energia

- Fondamentale per esperimenti LHC, ed esperimenti su raggi cosmici etc.

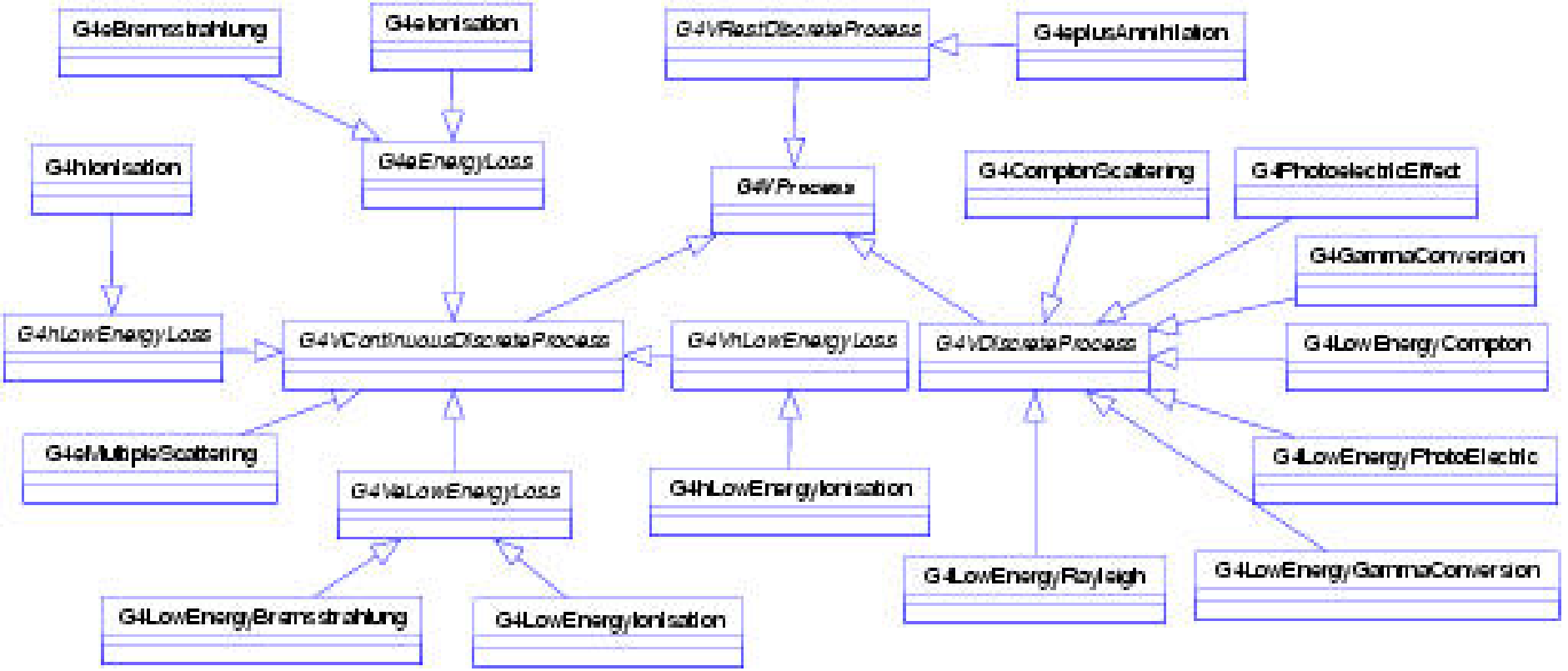
↘ Estensioni a bassa energia

- Fondamentale per esperimenti medici, neutrini, spettroscopia etc.

↘ Modelli alternativi per lo stesso tipo di fisica

OO design

Diagramma classi della fisica elettromagnetica



Sono possibili modelli alternativi che obbediscono alla stessa interfaccia astratta.

Soglie di produzione

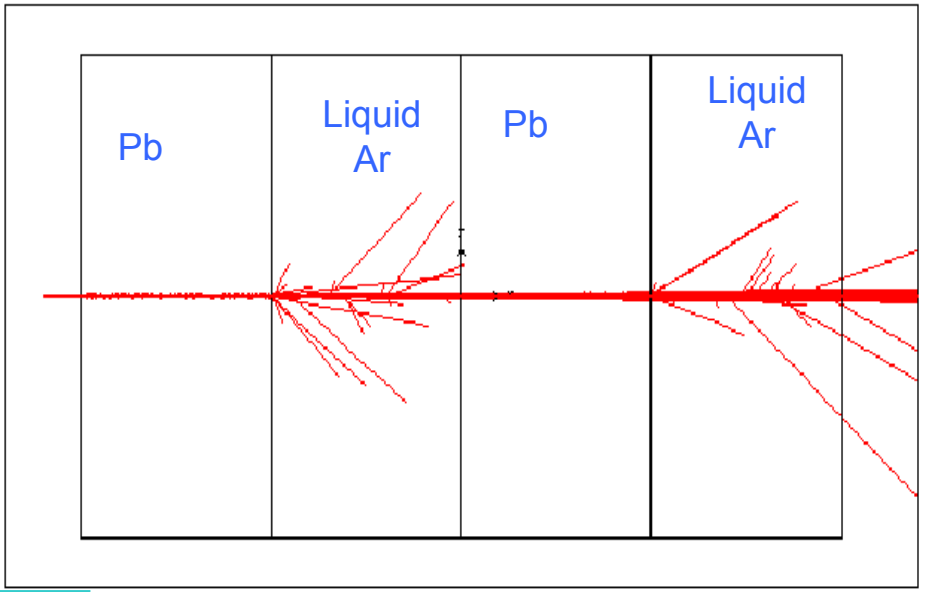
- Non ci sono tagli nel tracciamento solo soglie di produzione
  - Le soglie per produrre i secondari sono espresse in **range**, universale per tutti i mezzi
  - Convertito in energia per ogni particella e materiale

- E' piu' sensato utilizzare il *range cut-off*
  - Range di 10 keV gamma in Si ~ pochi cm
  - Range di 10 keV elettrone in Si ~ pochi micron

Effetto delle soglie di produzione

Geant 4

500 MeV protone incidente



Soglia in *range*: 1.5 mm

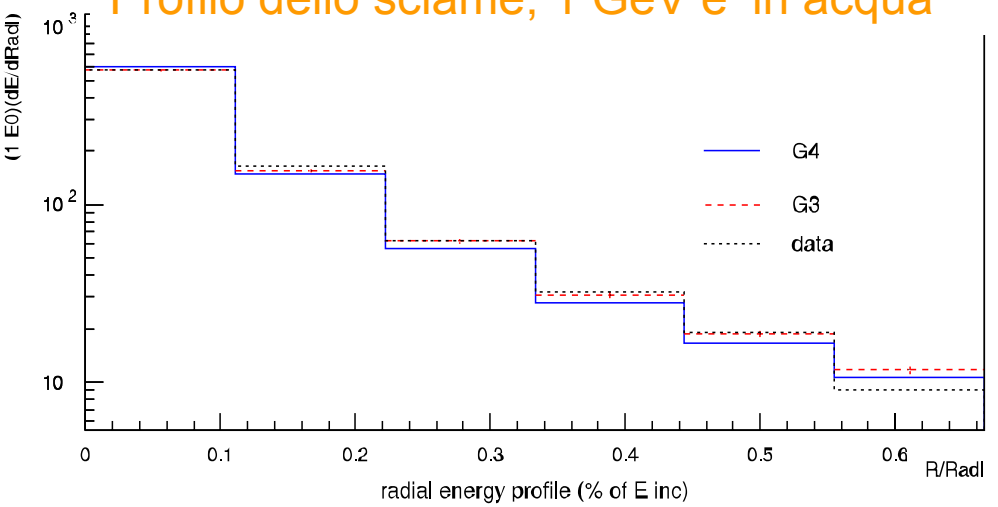
455 keV energia dell'elettrone in Ar liquido

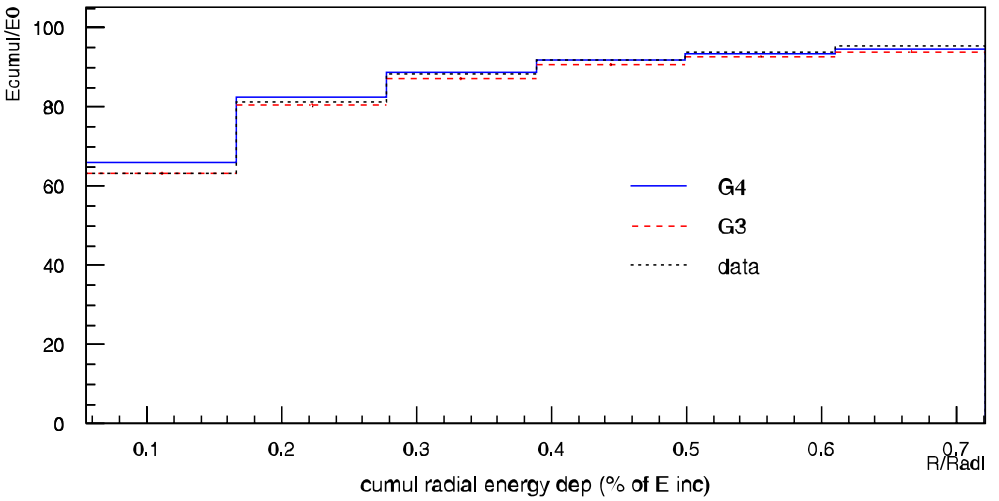
2 MeV energia dell'elettrone in Pb

Processi elettromagnetici standard

- Fotoni
  - Compton scattering
  - γ conversione
  - Effetto fotoelettrico
- Elettroni e Positroni
  - Bremsstrahlung
  - ionizzazione
    - Perdita di energia continua da Bremsstrahlung e ionizzazione
  - δ ray produzione
  - Annichilazione di positroni
  - Radiazione di sincrotrone

Profilo dello sciame, 1 GeV e^- in acqua





GammaRayPhysicsList

```
...
if (particleName == "gamma") {
      // gamma
      pManager->AddDiscreteProcess(new G4PhotoElectricEffect());
      pManager->AddDiscreteProcess(new G4ComptonScattering());
      pManager->AddDiscreteProcess(new G4GammaConversion());

    } else if (particleName == "e-") {
      // electron
      pManager->AddProcess(new G4MultipleScattering(),-1, 1,1);
      pManager->AddProcess(new G4eIonisation(),       -1, 2,2);
      pManager->AddProcess(new G4eBremsstrahlung(),   -1,-1,3);

    } else if (particleName == "e+") {
      // positron
      pManager->AddProcess(new G4MultipleScattering(),-1, 1,1);
      pManager->AddProcess(new G4eIonisation(),       -1, 2,2);
      pManager->AddProcess(new G4eBremsstrahlung(),   -1,-1,3);
      pManager->AddProcess(new G4eplusAnnihilation(),  0,-1,4);
...
 SetCutValue(cutForGamma, "gamma");
 SetCutValue(cutForElectron, "e-");
 SetCutValue(cutForElectron, "e+");
```

Seleziona I processi fisici (da attivare) per ogni particella

Stabilisci le soglie di produzione

Bibliografia

- Geant4 web home page
  http://wwwinfo.cern.ch/asd/geant4/geant4.html
- Geant4-Italy web home page
  http://www.ge.infn.it/geant4/
- Geant4 User Documentation
  http://wwwinfo.cern.ch/asd/geant4/G4UsersDocuments/Overview/html/index.html
- Geant4 results and publications http://wwwinfo.cern.ch/asd/geant4/reports/reports.html
- RD44 web home page
  http://wwwinfo.cern.ch/asd/geant/geant4.html