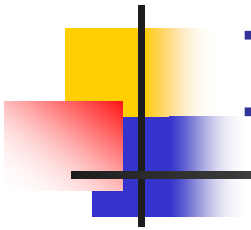
principi e applicazioni

Luciano Pandola

INFN-LNGS

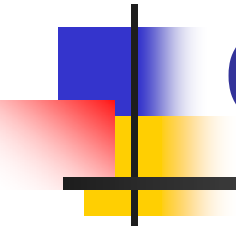
Corso INFN su C++, ROOT e Geant4

Presentazione basata su contributi di M. Antonello, G.A.P. Cirrone, A. Dotti, M.G. Pia e F. Romano



Introduzione e scopo

- Questa lezione **non è un corso di Geant4**
  - *non* vi metterà nelle condizioni di scrivere da zero un'intera simulazione Geant4
  - ci vorrebbe un corso di *alcuni giorni*
- Lo scopo è mostrare come i concetti visti nel corso (OO, ereditarietà, interfacce astratte) trovano un'applicazione reale in un software complesso e di ampio utilizzo in fisica
- Dare un'idea delle potenzialità e funzionalità di Geant4, per chi fosse interessato ad approfondire



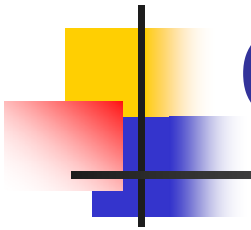
Cos’è Geant4

Codici Monte Carlo

- MCNP (principalmente neutroni)
- Penelope ($e^{\pm}$ e gamma)
- PETRA (protoni)
- EGSnrc ($e^{\pm}$ e gamma)
- PHIT (protoni/ioni)
- FLUKA (tutte le particelle)
- **Geant4**

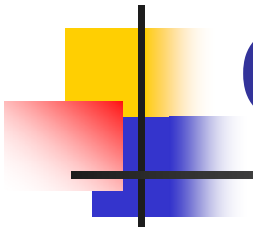
GEometry ANd Tracking

Nucl. Inst. and Methods Phys. Res. A, **506**, 250
Transaction on Nuclear Science **53**, 270



Geant4

- Scritto in **linguaggio C++**
  - **Pensato** per essere Object Oriented
  - Flessibilità grazie alle interfacce astratte
- **Open Source**
  - Scaricabile (codice sorgente) liberamente da Internet
- E' un **toolkit**, ossia un insieme di strumenti che l'utente può utilizzare per la propria simulazione
- **`http://geant4.cern.ch`**



Cosa offre?

- Varietà e flessibilità per definire geometrie anche complesse
  - Forme geometriche base
  - Rappresentazione di solidi tramite le superfici di confine
  - Operazioni Booleane
- Varietà e flessibilità nella definizione di elementi e materiali
- Un'enorme varietà di particelle e di **processi fisici** disponibili
  - Comprese particelle instabili e ioni

Di cosa ho bisogno?

- **C++**
  - Conoscenza base necessaria: perché Geant4 è un insieme di librerie in C++ (!)
  - Non è richiesta una grandissima esperienza di C++
- **Object oriented technology (OO)**
  - Richiesta una conoscenza base
  - Più esperienza necessaria se si vogliono scrivere applicazioni più complesse
- **Sistemi operativi Linux/Unix**
  - Sono i sistemi operativi standard per Geant4
  - Principali comandi di shell
  - Come compilare un programma dal sorgente

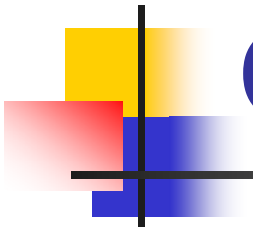
Il kit Geant 4

- Codice
  - **~1M** di linee, in crescita
  - scaricabile pubblicamente da web
- Documentazione
  - 5 manualli
  - scaricabili pubblicamente da web
- Esempi
  - distribuiti con il codice
  - diverse applicazioni complete di set-up sperimentali realistici (sebbene semplificati)
- Piattaforme
  - Linux, Windows, MacOS
- Sofware commerciale
  - Nessuna dipendenza
  - Possibili interfacce
- Software libero
  - gmake, g++
  - CLHEP
- Grafica & (G)UI
  - OpenGL, X11, OpenInventor, DAWN, VRML...
  - OPACS, GAG, MOMO...

Dove lo trovo?

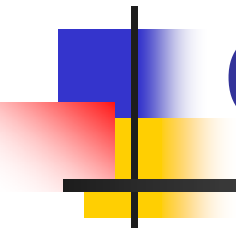
- Sul sito di Geant4 **http://geant4.cern.ch** si possono scaricare
  - Il codice sorgente C++
  - Le librerie pre-compilate, per alcune piattaforme
- Consiglio: scaricare e ricompilare sempre il codice sorgente

1. Scaricare il codice sorgente dal sito di Geant4
2. Scaricare i files del database usato per i modelli fisici (sempre dal sito di Geant4)
3. Scaricare e installare le librerie CLHEP
   - **http://cern.ch/clhep**



Come lo installo?

- Due alternative:
  - **[Consigliato]** Utilizzare lo **script configure** (**`./configure`**) fornito con Geant4 per definire le variabili di ambiente
    - Vengono fatte all'utente delle **domande** a schermo per definire il tipo di sistema e le impostazioni da utilizzare
    - **`./configure -build`**
  - Definire manualmente le variabili di ambiente appropriate **prima** di iniziare la compilazione di Geant4



Costruire un'applicazione Geant4: principi generali

Dicevamo che è un toolkit: e allora?

- Non esiste il concetto di "Geant4 default"
- L'utente **deve** fornire le informazioni necessarie per configurare la propria simulazione (e.g. geometria, processi fisici, etc.)
- L'utente **deve** scegliere quali degli strumenti di Geant4 usare
- Guida: un certo numero di esempi
  - Novice examples: overview degli strumenti di Geant4
  - Advanced examples: strumenti di Geant4 in applicazioni realistiche

Geant4: concetti base

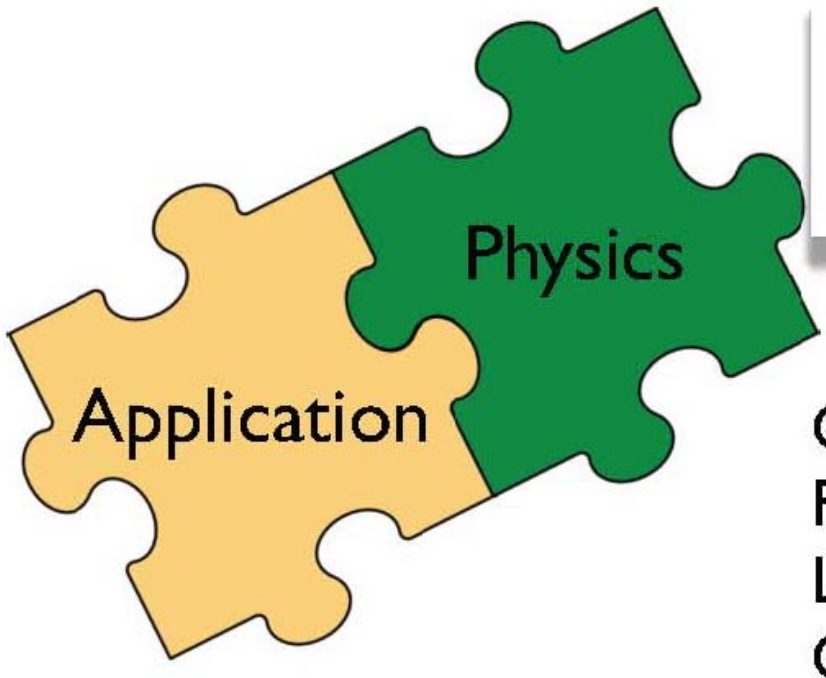
- Cosa l'utente **DEVE** fare:
  - Descrivere il proprio setup sperimentale
  - Definire le particelle primarie da generare per la propria simulazione (tipo, energia, posizione, direzione)
  - Decidere quali particelle e quali modelli fisici devono essere usati fra tutti quelli disponibili in Geant4, e qual è la precisione che si vuole ottenere (scelta dei cut di produzione di particelle secondarie)

Geant4: concetti base

- Cosa **SI PUÒ** voler fare:
  - Interagire con il kernel di Geant4 per controllare la simulazione (selezionare eventi, uccidere particelle, etc.) ed accedere alle informazioni di interesse
  - Visualizzare il setup sperimentale e le tracce simulate
  - Produrre istogrammi, ntuple e oggetti per la succesiva analisi offline

Costruzione di un'applicazione

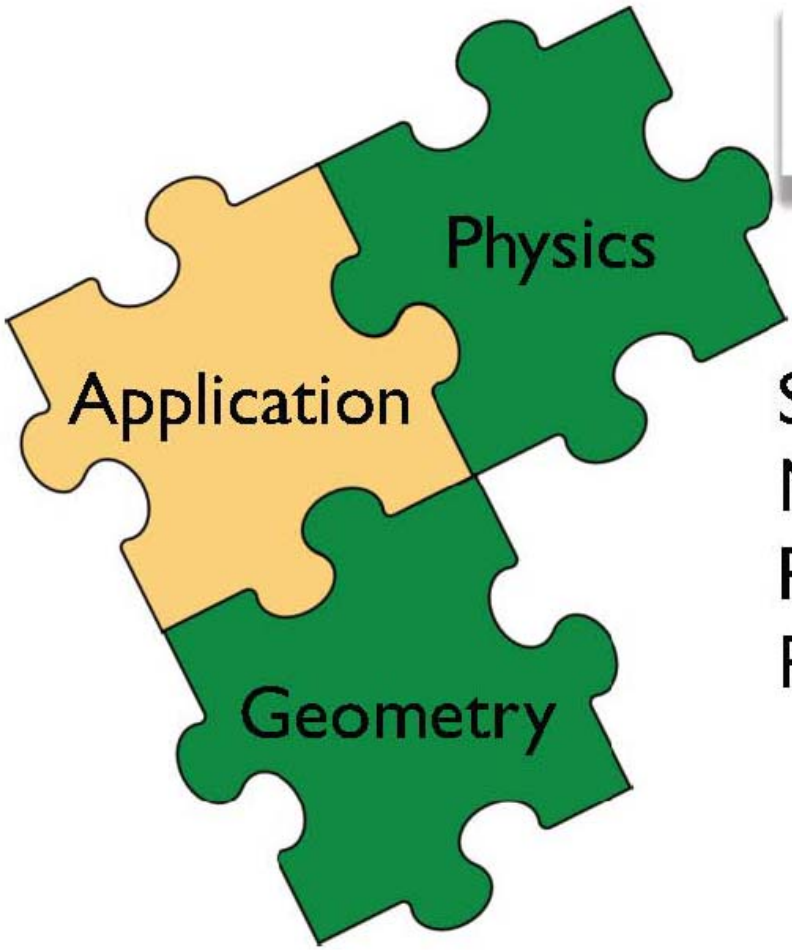
Costruzione di un'applicazione



From Geant4:
One of the provided Physics lists or build/tailor your own

QGSP_BERT
FTFP_BERT
LHEP
QGSP_BIC
CHIPS
....

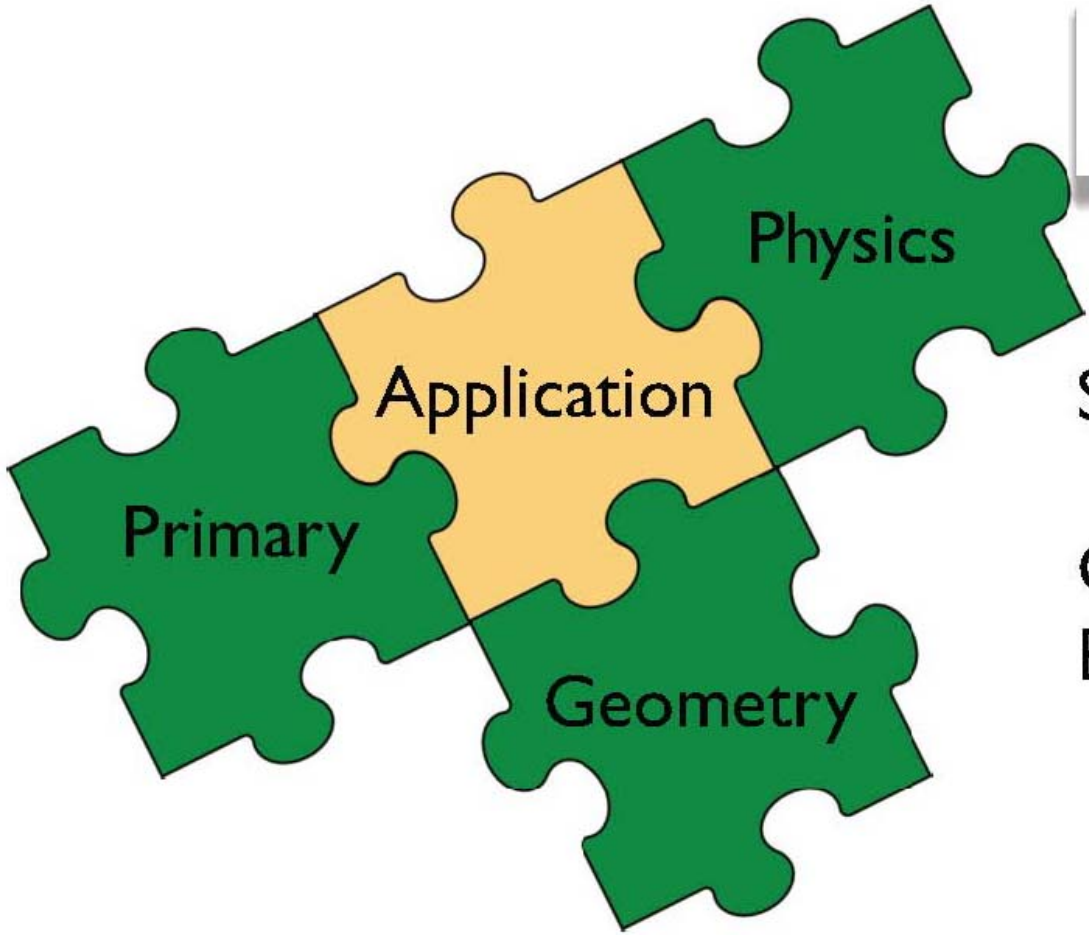
Costruzione di un'applicazione



Describe your detector using G4 geometry

Shapes,
Materials,
Placements,
Read-out geo

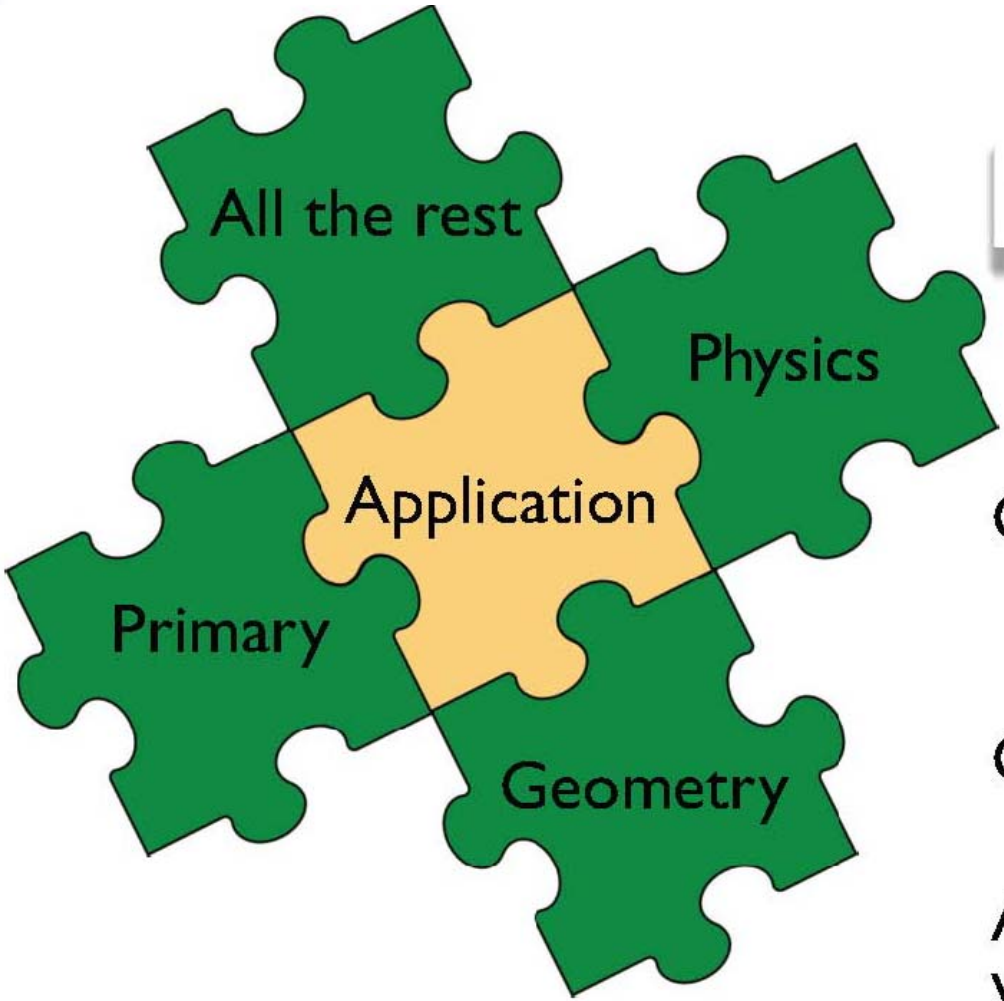
Costruzione di un'applicazione



Describe the source of radiation

Simple Gun
test-beam like
Generic source
External input
HepMC files

Costruzione di un'applicazione



Add all the rest

G4UserActions
interact with simulation
G4Hits/Digits
read-out
Analysis
Visualization

Classi utente obbligatorie

- Classi utente obbligatorie in *qualunque* applicazione utente basata su Geant4:

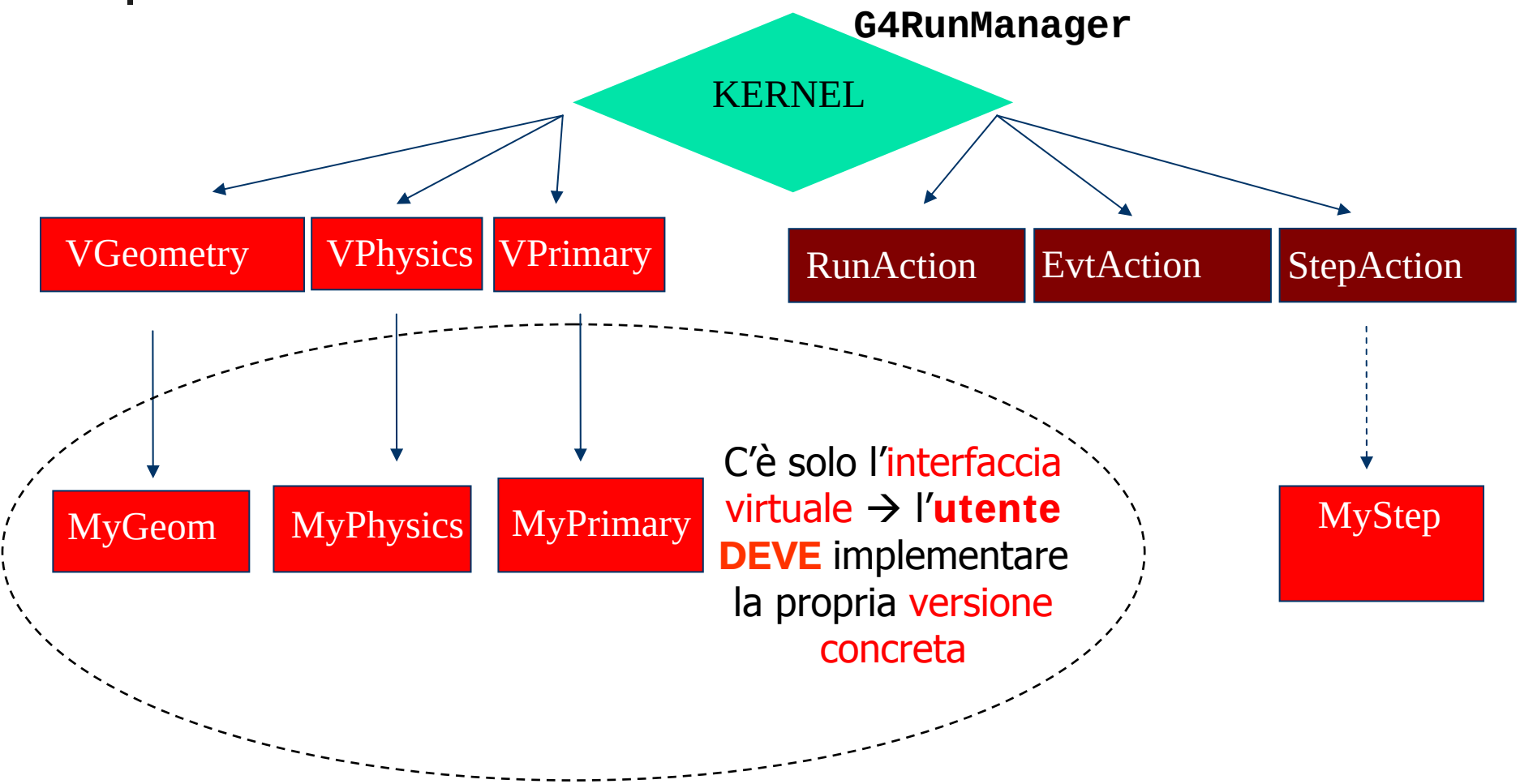
  1. derivata da **`G4VUserDetectorConstruction`**
     descrive il setup sperimentale
  2. derivata da **`G4VUserPhysicsList`**
     selezione la fisica da utilizzare ("Physics List")
  3. derivata da **`G4VUserPrimaryGeneratorAction`**
     produce gli eventi primari

- Classi **puramente virtuali** (solo interfaccia!)

Classi utente facoltative

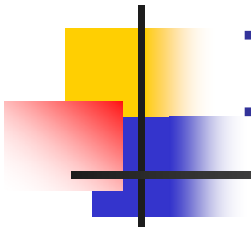
- Alcune classi utente possono essere scritte e registrate nel kernel per affettuare operazioni al livello del loop della simulazione (**ACTION CLASSES**)
- Devono ereditare da:
  - **`G4UserRunAction`**
  - **`G4UserEventAction`**
  - **`G4UserStackingAction`**
  - **`G4UserSteppingAction`**
- Le classi madre non sono virtuali, quindi l'implementazione dei metodi virtuali non è necessaria
  - esiste un default → "fai niente"

Concetto generale di Geant4



Il main() di Geant4

- Geant4 **non** fornisce un **`main()`**
  - Geant4 è un toolkit!
  - Il main() è parte dell'applicazione utente
- Nel proprio main(), l'utente deve
  - Costruire il **`G4RunManager`**
  - Registrare al **`G4RunManager`** le **classi obbligatorie** derivate (via i loro puntatori) scritte dall'utente
    - **`runManager->SetUserInitialization (new MyDetectorConstruction);`**



Il main() di Geant4

- L'utente **PUÒ** definire nel proprio main():
  - Classi utente facoltative (e.g. User Actions)
  - VisManager, sessione di (G)UI
- L'utente deve anche farsi carico di recuperare e salvare l'informazione di interesse della simulazione (energie, etc.)
  - Geant4 **non lo farà** di default
- Curare di cancellare l'istanza del **`G4RunManager`** alla fine dell'esecuzione

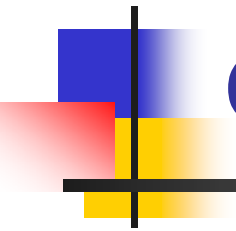
Un esempio di main()

```
{
  // Construct the default run manager
  G4RunManager* runManager = new G4RunManager;
 // Set mandatory user initialization classes
  MyDetectorConstruction* detector = new MyDetectorConstruction;
  runManager->SetUserInitialization(detector);
  MyPhysicsList* physicsList = new MyPhysicsList;
  runManager->SetUserInitialization(myPhysicsList);

  // Set mandatory user action class (Primary Generator)
  runManager->SetUserAction(new MyPrimaryGeneratorAction);

  // Set optional user action classes (e.g. only a few of them)
  MyEventAction* eventAction = new MyEventAction();
   runManager->SetUserAction(eventAction);
  MyRunAction* runAction = new MyRunAction();
   runManager->SetUserAction(runAction);
..
  delete runManager;
}
```

Un’occhiata al kernel e alle tre componenti obbligatorie



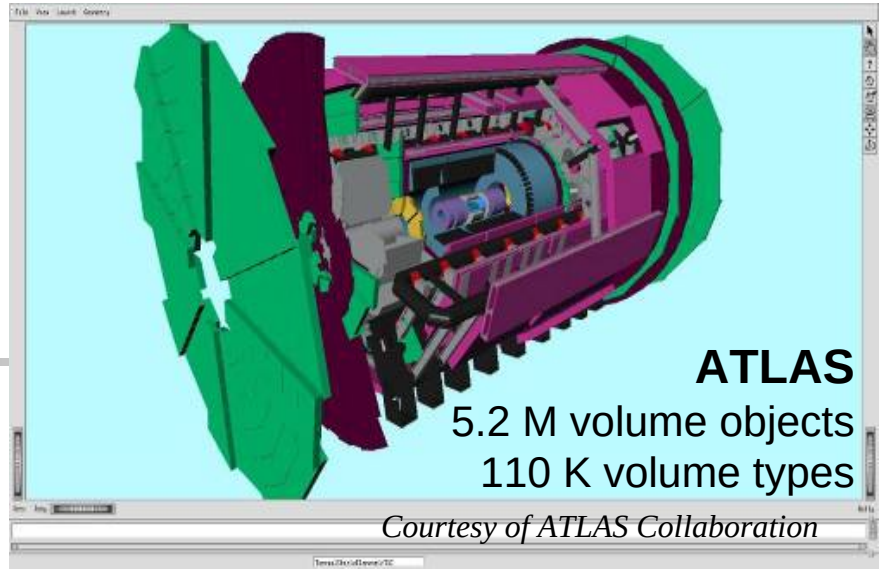
Il kernel – Run ed eventi

- **Eventi multipli**
  - Più di un primario
    - diverse posizioni, energie, tempi
  - Possibilità di simulare il pile-up
- Run multipli nella stessa esecuzione
  - diverse geometrie, materiali, etc
- Meccanismo di stacking flessibile
  - Tre livelli di default (urgente, postponed, killed): possibile fare **studi di trigger**, tracciare eventi soltanto se si verificano particolari condizioni
  - **Tagli** in energia, tempo, etc. possono essere definiti dall'utente

Il kernel – il tracciamento

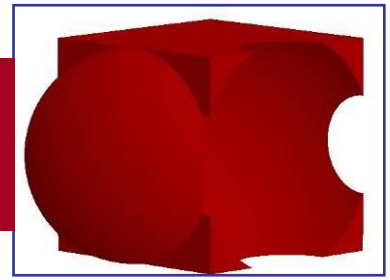
- **Disaccoppiato dalla fisica**
  - Tutti in processi fisici sono gestiti attraverso la **stessa interfaccia astratta** → **`G4VProcess`**
  - Il tracking "non sa" quale processo in particolare sta gestendo (Compton, etc.)
  - Flessibilità dell'approccio OO
- Independente dal tipo di particella
  - Tutte le particelle sono tracciate fino ad energia zero
- Consente di aggiungere nuovi modelli di fisica (user-custom) senza necessità alterare il tracciamento

Geometria

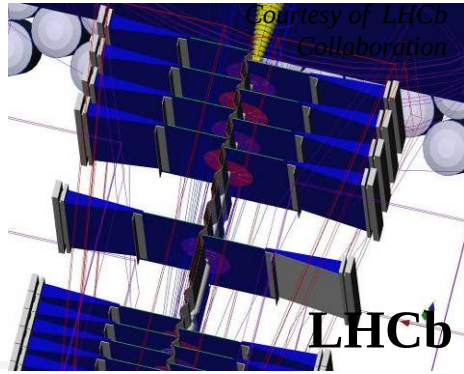


- Scopo
  - Descrizione dettagliata del rivelatore
  - Navigazione efficiente
- **Tre** livelli concettuali:
  - Solido: forma, dimensioni
  - LogicalVolume: materiale, sensibilità, volumi figli, visualizzazione, campi magnetici etc,
  - PhysicalVolume: posizione e rotazione
    - Un volume logico può essere piazzato più volte per originare volumi fisici diversi

Operazioni booleane



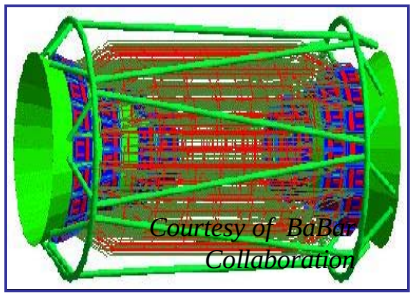
Solidi in Geant4

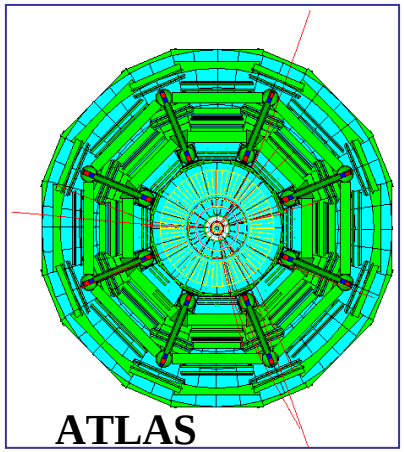


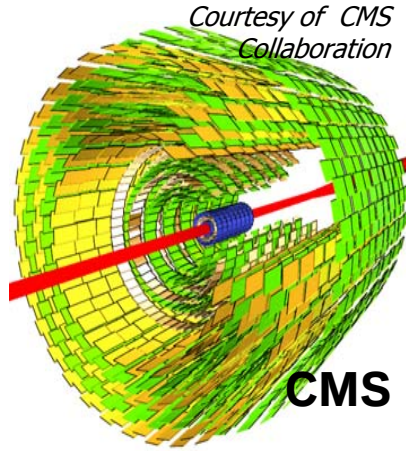
Rappresentazioni multiple
Stessa interfaccia astratta
G4VSolid

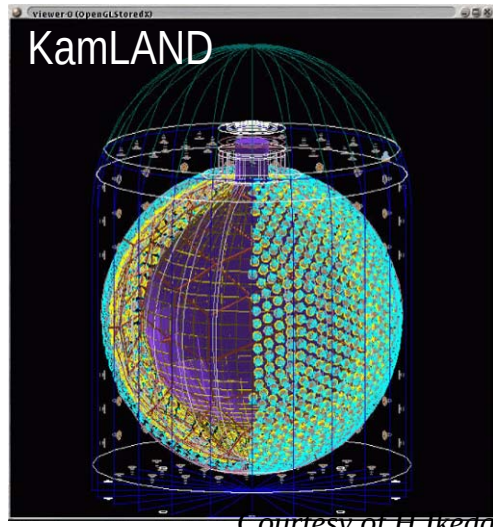
- CSG (**C**onstructed **S**olid **G**eometries)
  - solidi semplici (box)
- STEP extensions
  - poliedri, sfere, cilindri, coni, toroidi, etc.
- BREPS (**B**oundary **REP**resented **S**olids)
  - Volumi definiti dalle loro superfici di confine

BaBar

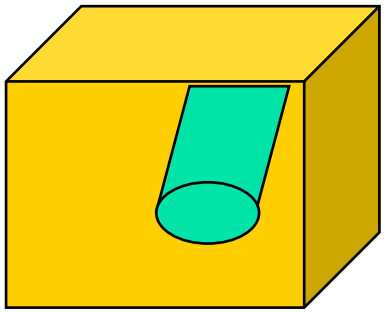




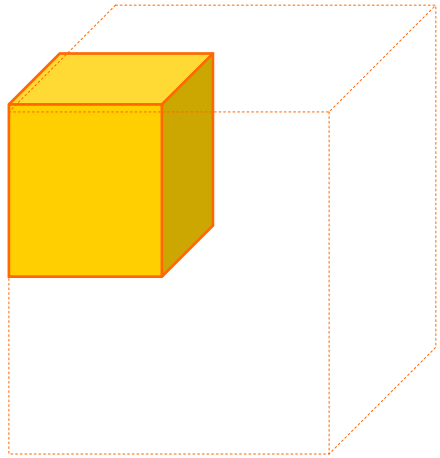


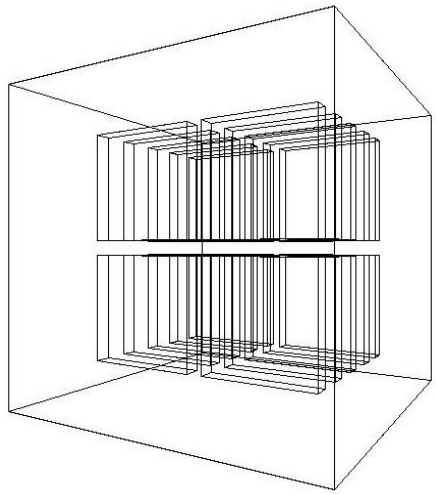


Volumi fisici

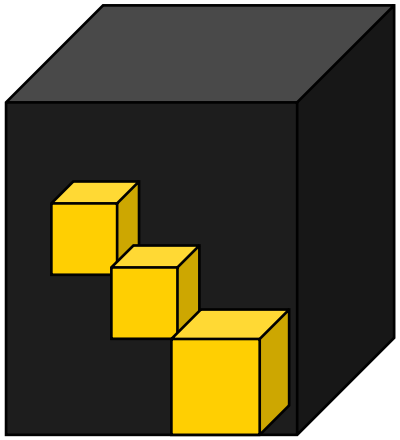


placement

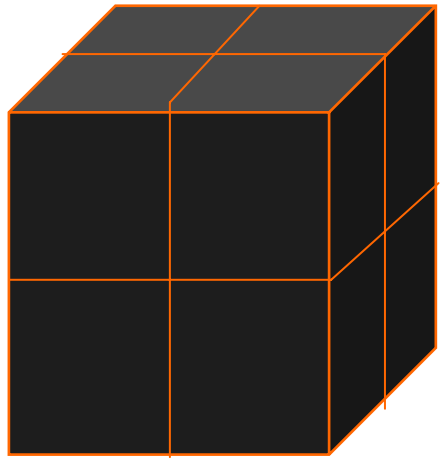




assembled



parameterised

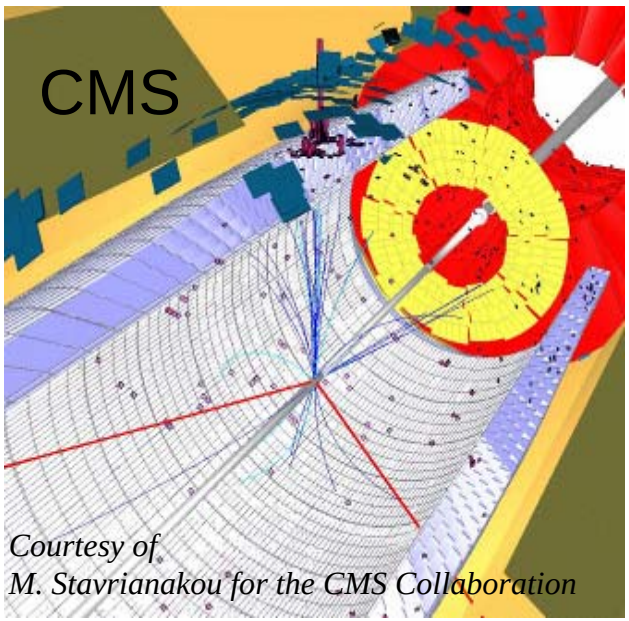


replica

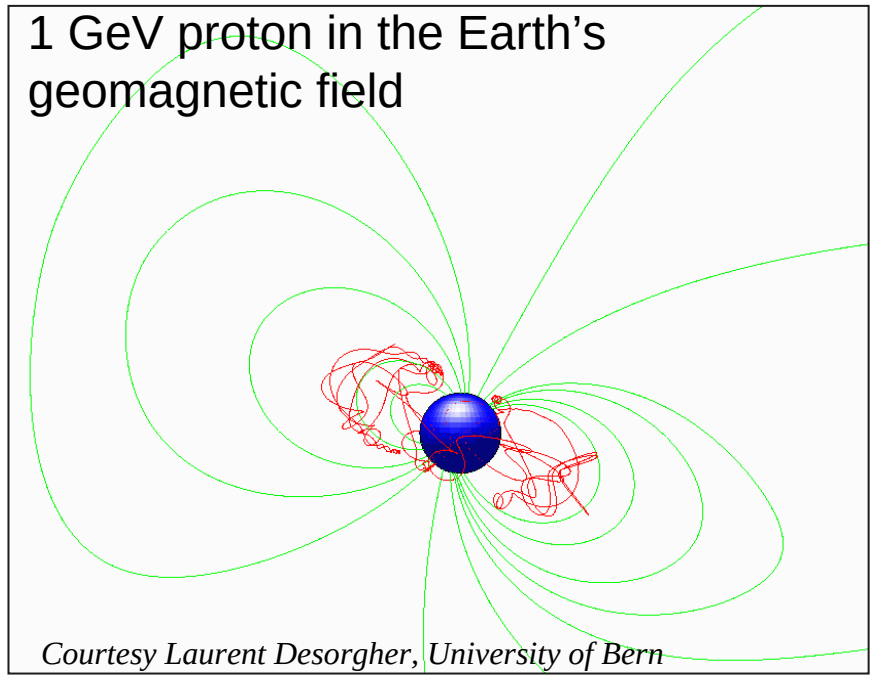
Flessibilità per descrivere geometrie complesse

Campi elettromagnetici

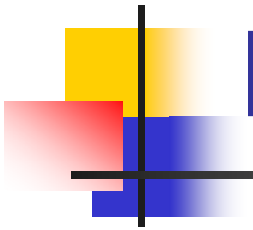
- Geant4 consente di tracciare particelle cariche in **campi elettromagnetici** di intensità e direzione variabile
- Circa 2 volte più veloce di Geant3



Courtesy of
M. Stavrianakou for the CMS Collaboration

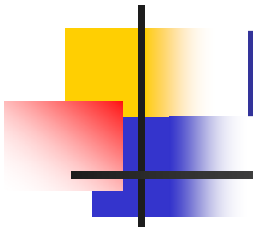


Courtesy Laurent Desorgher, University of Bern



Fisica

- Interfaccia astratta per i processi fisici
  - Trattamento uniforme dei processi elettromagnetici e adronici
  - Trasparente per il tracking
- Distinzione fra **processi** e **modelli**
  - Uno stesso processo (e.g. Scattering Compton, o scattering inelastico di n) può essere descritto da **modelli diversi**
  - Modelli possono essere complementari (e.g. funzionare ad energie diverse) oppure alternativi
- Sistema **aperto**
  - Gli utenti possono implementare i **propri modelli**, ereditando dalle opportune classi base



Fisica

- Trasparenza
  - Calcolo della sezione d'urto indipendente dal calcolo dello stato finale
  - Calcolo dello stato finale indipendente dal tracking
  - Le unità di misura sono usate esplicitamente all'interno del codice (CLHEP)
- Cuts
  - **Non** ci sono cut di **tracciamento**, ma solo di **produzione**.
  - Particelle secondarie (δ e soft γ) **generate** solo se hanno **range superiore** ad un valore impostato dall'utente

Scelta dei processi fisici

- Geant4 non ha alcuna particella né processo fisico di default
- L'utente può:
  - Scrivere una **propria classe** derivata da **`G4VUserPhysicsList`** in cui implementa i metodi virtuali
    - **`ContructParticles(), ConstructProcesses(), SetCuts()`**
    - Definisce particelle, processi e cuts
  - Utilizzare una delle **"physics list" già pronte** fornite da Geant4
    - Physics lists ottimizzate per applicazioni specifiche e regolarmente testate da Geant4
    - Consigliato per utenti non esperti

Particelle primarie

- I **vertici primari** e le **particelle primarie** devono essere inserite in un evento (**`G4Event`**) prima che l'evento possa essere processato. C'è bisogno di:
  - **`G4PrimaryVertex`**
    - punto di partenza nello spazio e nel tempo
  - **`G4PrimaryParticle`**
    - momento, polarizzazione, tipo di particella
    - per le particelle instabili: contiene la lista di particelle secondarie
- Nello stesso evento ci possono essere più vertici e/o più particelle
- Il tutto va fatto nel metodo utente **`GeneratePrimaries()`** nella classe concreta che deriva da **`G4VUserPrimaryGeneratorAction`**

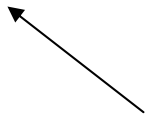
G4VUserDetectorConstruction

La geometria-utente deve **ereditare** da qui:

```
class G4VUserDetectorConstruction
{
  public:
    G4VUserDetectorConstruction();
    virtual ~G4VUserDetectorConstruction();

    virtual G4VPhysicalVolume* Construct() = 0;

…
}
```



Metodo **puramente virtuale**

Deve restituire il puntatore al **G4VPhysicalVolume** del **volume mondo**

G4VUserPhysicsList

La physics list dell'utente deve **ereditare** da qui:

```
class G4VUserPhysicsList
{
  public:
    G4VUserPhysicsList();
    virtual ~G4VUserPhysicsList();
    virtual void ConstructParticle() = 0;
    virtual void ConstructProcess() = 0;
    virtual void SetCuts() = 0;
…
}
```

Tre metodi **puramente virtuali**

Devono curare la definizione delle particelle, dei processi fisici e dei tagli da utilizzare

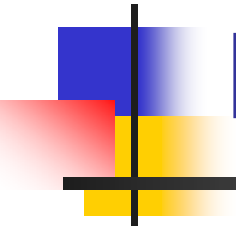
G4VUserPrimaryGeneratorAction

La classe-utente per la generazione degli eventi primari deve **ereditare** da qui:

```
class G4VUserPrimaryGeneratorAction
{
  public:
    G4VUserPrimaryGeneratorAction();
    virtual ~G4VUserPrimaryGeneratorAction();
    virtual void
      GeneratePrimaries(G4Event* anEvent) = 0;
};
```

Metodo **puramente virtuale**

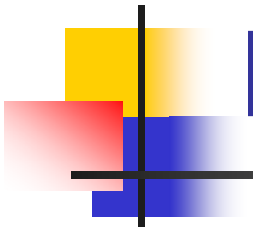
Deve generare lo **stato iniziale** associato all'oggetto **G4Event** (passato in input)



Le User Actions facoltative

Classi utente facoltative

- Varie **classi base concrete**, i cui metodi virtuali possono essere riscritti dall'utente, consentono di controllare la simulazione a vari livelli:
  - **`G4UserRunAction`**
  - **`G4UserEventAction`**
  - **`G4UserStackingAction`**
  - **`G4UserSteppingAction`**
- L'utente può eseguire ogni operazione che desidera ai vari livelli
  - E.g. fare qualche operazione ad ogni step
- I puntatori delle action class utente devono essere registrati al **`G4RunManager`**, come gli altri
  - **`runManager->SetUserAction(new MyEventActionClass);`**



Metodi delle action classes

G4UserRunAction

- `BeginOfRunAction(const G4Run*)` // booking degli istogrammi
- `EndOfRunAction(const G4Run*)` //salvataggio istogrammi

G4UserEventAction

- `BeginOfEventAction(const G4Event*)` //inizializzazione evento
- `EndOfEventAction (const G4Event*)` //analisi online evento

G4UserSteppingAction

- `UserSteppingAction(const G4Step*)`

//uccidere, sospendere, posporre la traccia, disegnare lo step, ...

Metodi delle action classes

G4UserStackingAction

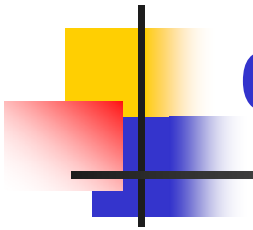
- `PrepareNewEvent()` //reset del controllo di priorità
- `ClassifyNewTrack(const G4Track*)`

// Chiamato ogni volta che una nuova traccia viene registrata

// (si può uccidere, posporre, etc)

- `NewStage()`

// Chiamato quando lo stack "Urgent" è vuoto (re-classify, abort event)



G4UserRunAction

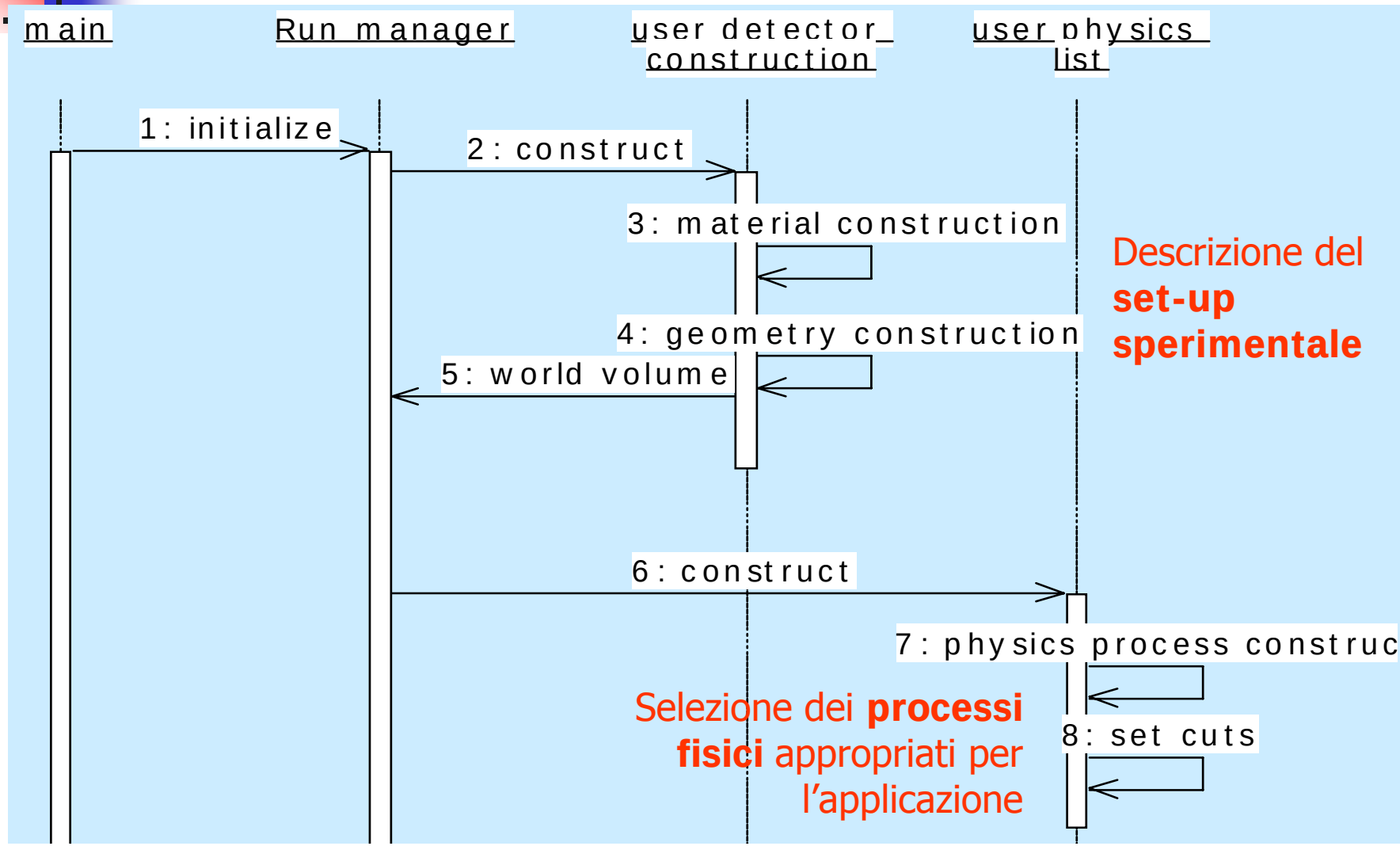
Le **action class** dell'utente hanno **classi base** di questo tipo:

```
class G4UserRunAction
{
  public:
    G4UserRunAction();
    virtual ~G4UserRunAction();
    virtual void BeginOfRunAction(const G4Run* aRun);
    virtual void EndOfRunAction(const G4Run* aRun);
};
```

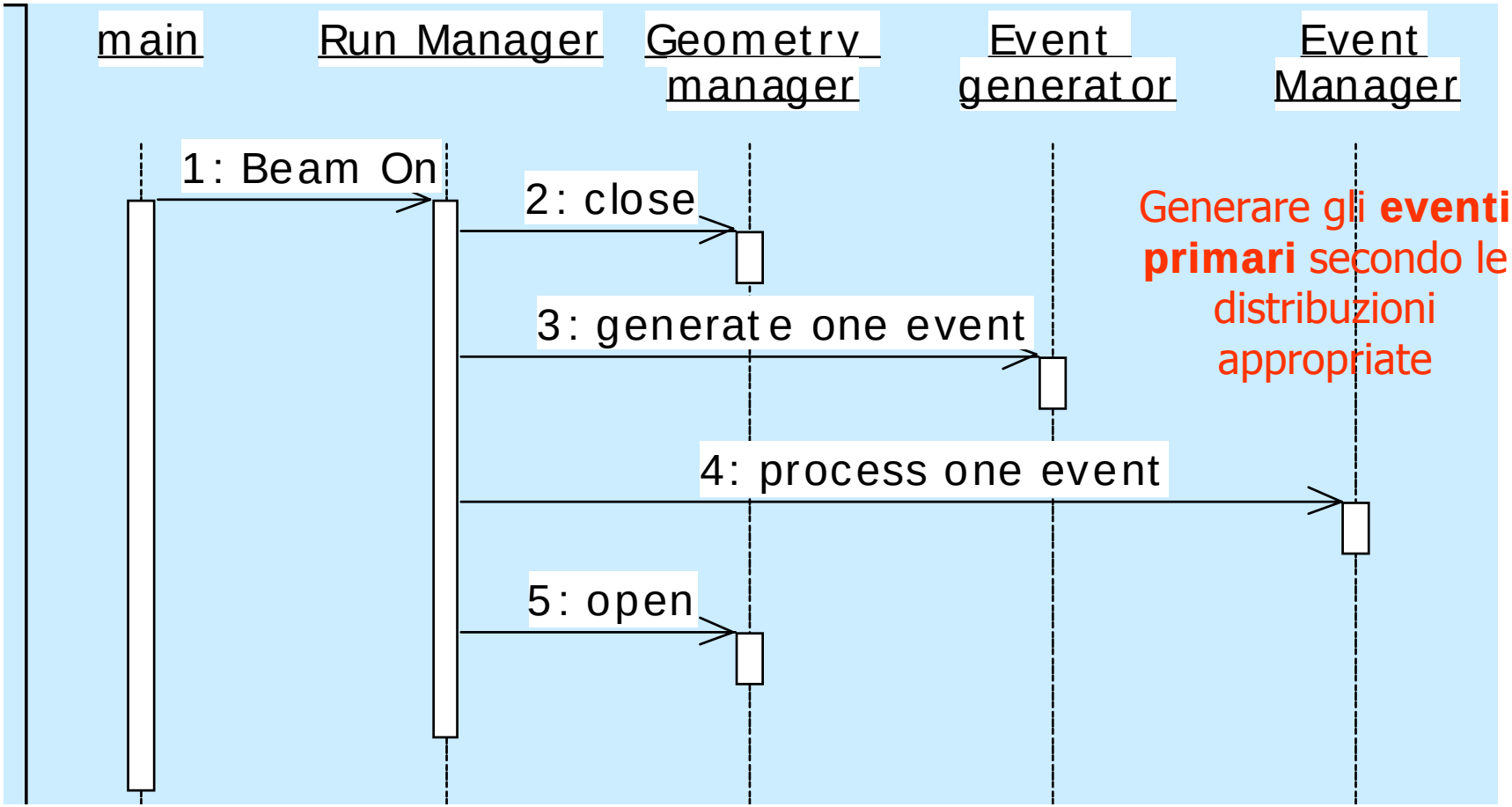
Metodi **virtuali** ma non "assolutamente" virtuali

Possono essere riscritti, ma c'è comunque un default (che è vuoto)

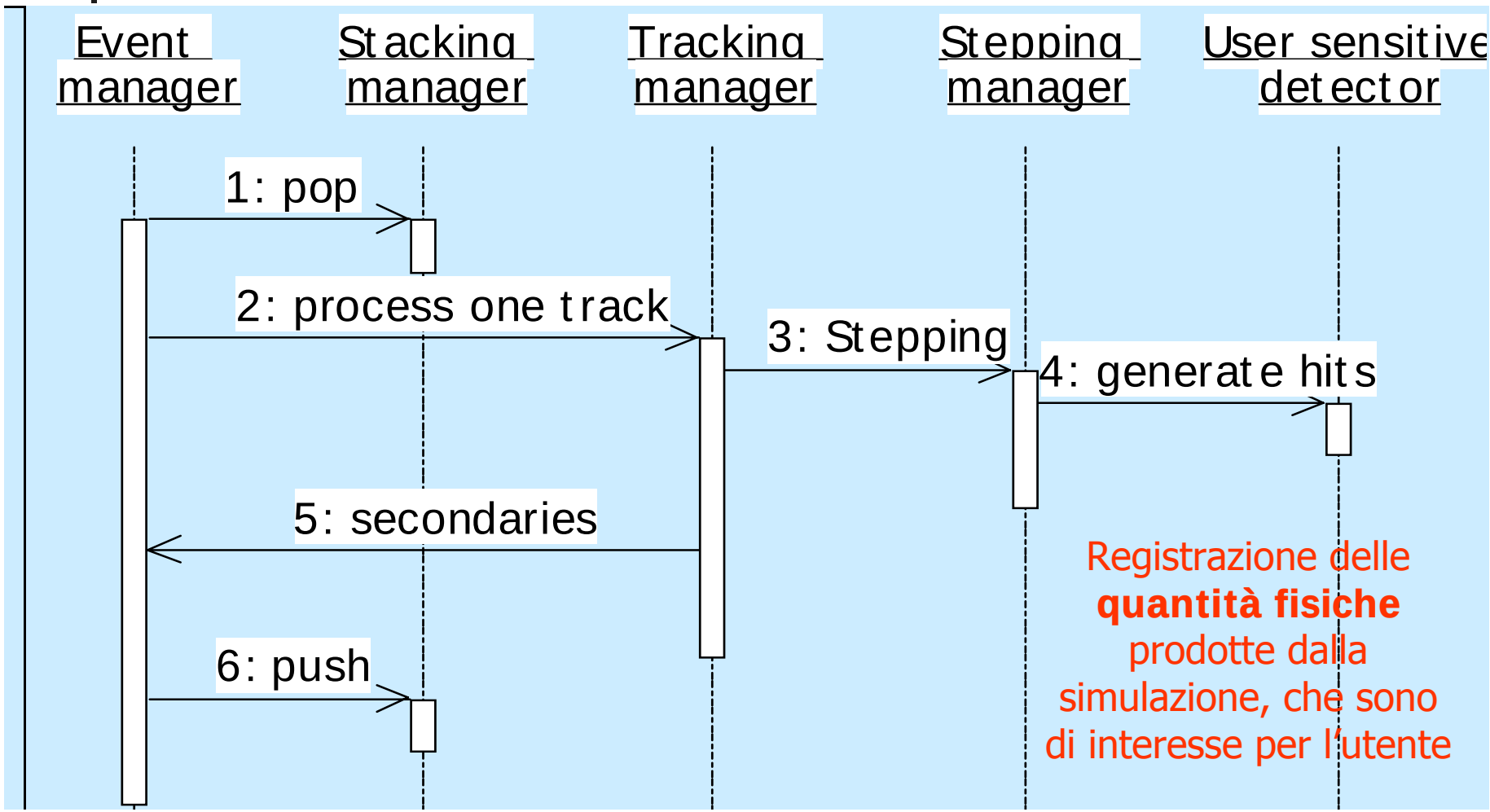
Loop di inizializzazione

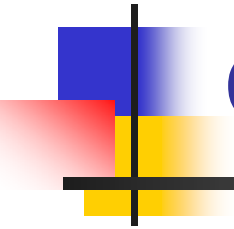


Loop di tracciamento (BeamOn)

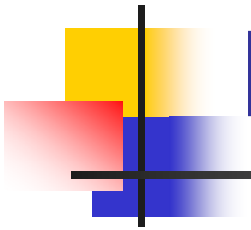


Loop di event processing





Visualizzazione, interfacce grafiche

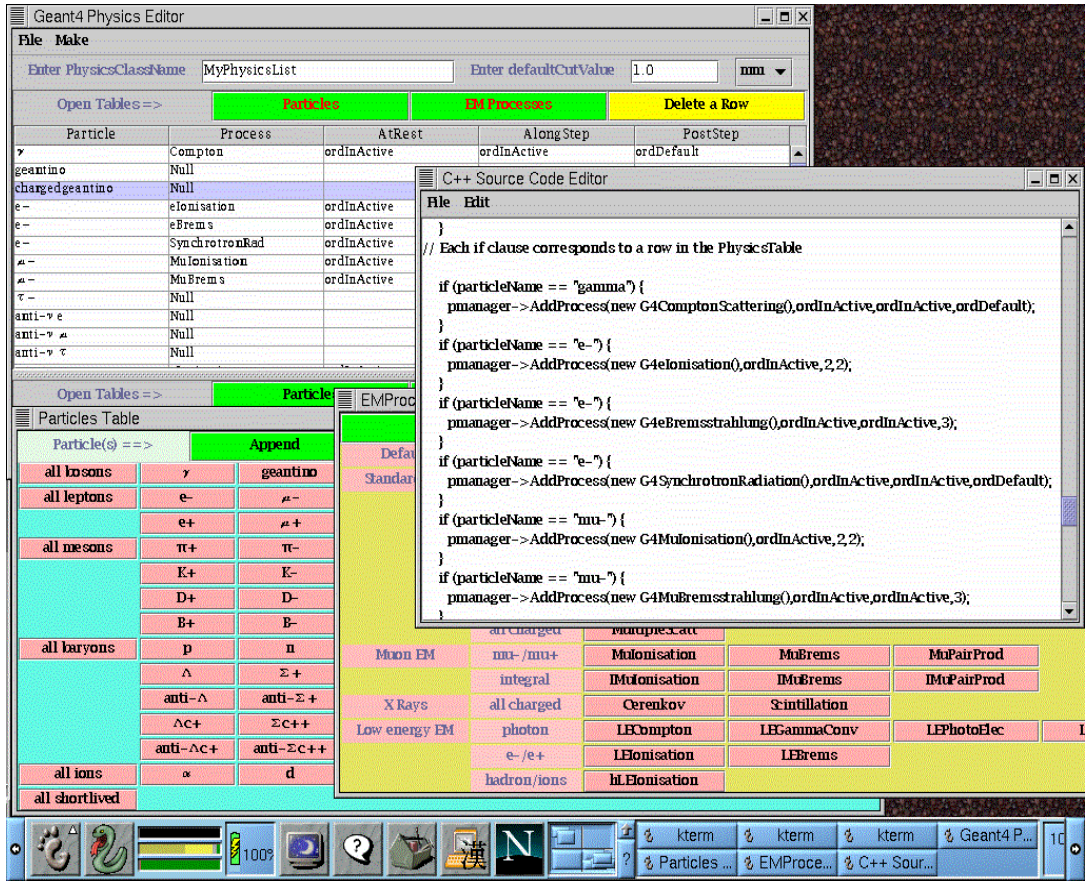


Facoltativo: (G)UI

- Per controllare la simulazione (interattivo o batch) può essere utile una **interfaccia utente** (possibilmente grafica)
- Nel **`main()`** va instanziata una classe derivata da **`G4UISession`** fornita da Geant4 (scelta a seconda di cosa è installato sul proprio sistema) e chiamato il metodo SessionStart():
  - **`mysession -> SessionStart();`**
- Geant4 fornisce:
  - **`G4UIterminal`** (shell csh o tcsh-like)
  - Varie interfacce grafiche, eventualmente system-dependent o che richiedono librerie esterne (e.g. Qt)
  - Possibilità di batch usando macro files in **ASCII**

Interfacce utente

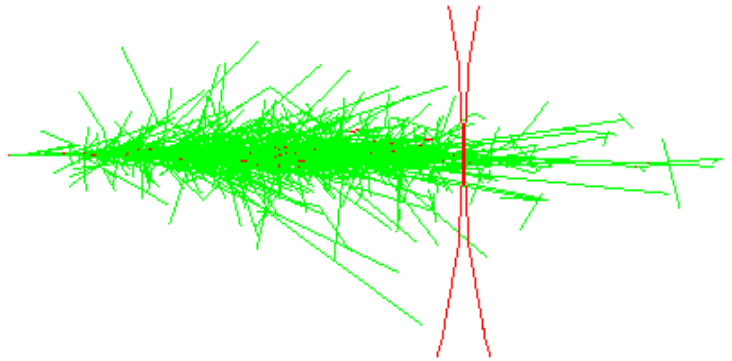
- Implementazioni multiple disponibili, tutte gestite via interfacce astratte
- Disponibile una **linea di comando** (batch e terminale)
- **GUIs**
  - X11/Motif, GAG, MOMO, OPACS, Java
  - Possono dipendere da software esterni



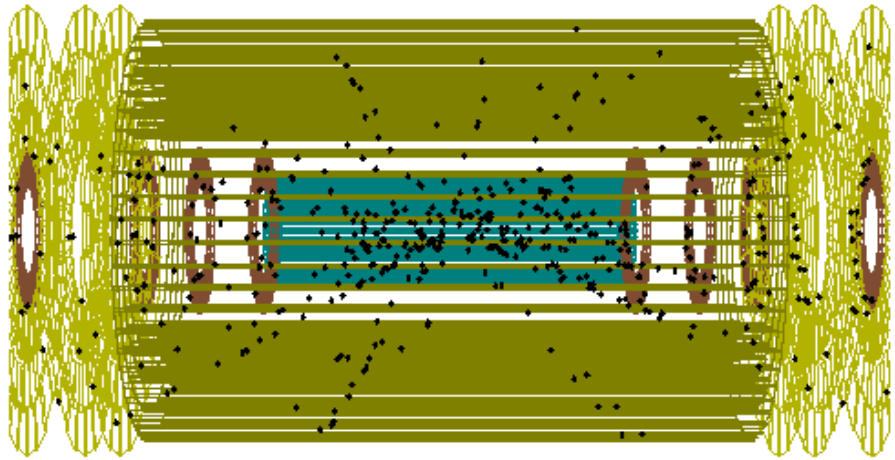
Facoltativo: visualizzazione

- Nel **`main()`** va instanziata la classe **`G4VisExecutive`** di Geant4 e chiamato il metodo **`Initialize()`**
- Vengono attivati dal **`G4VisExecutive`** i sistemi grafici che sono **disponibili sul proprio sistema**, a seconda delle variabili di ambiente impostate
  - Alcuni driver possono richiedere delle **librerie esterne** (Qt, OpenGL,)
- La lista dei sistemi grafici resi disponibili dal **`G4VisExecutive`** viene creata al momento della compilazione di Geant4 (./configure)

Visualizzazione

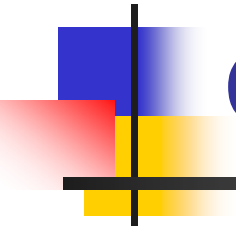


- Geant4 fornisce le interfacce per molti driver grafici
  - Dawn
  - Wired
  - RayTracer
  - OpenGL
  - VRML
  - Qt
  - ....

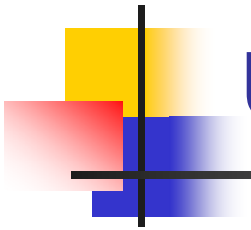


- Alcuni driver sono appropriati per produrre plot di alta qualità (= articoli), altri per il debug della geometria

Riassunto – scrivere un main di Geant4



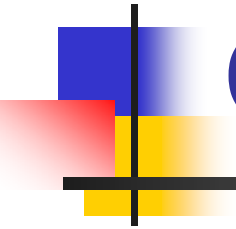
Ricettina generale per i nuovi utenti – parte 1



1. Disegnare la propria applicazione a seconda delle proprie necessità… questo richiede un po' di riflessione preliminare!
   - Cosa si vuole che l'applicazione faccia?
2. Creare le **proprie classi** derivate **obbligatorie** per definire la geometria, la fisica e le particelle primarie
   - **`MyDetectorConstruction`**
   - **`MyPhysicsList`**
   - **`MyPrimaryGeneratorAction`**

Ricettina generale per i nuovi utenti – parte 2

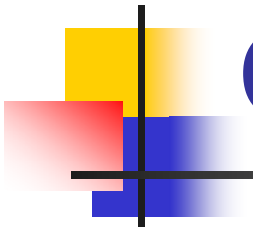
3. Creare eventualmente le proprie **classi opzionali** derivate (*action classes*)
   - **`MyUserRunAction, MyUserEventAction`**
4. Creare il **`main()`** file
   - Instanziare il **`G4RunManager`**
   - Registrare al RunManager le classi utente obbligatorie ed eventualmente le action classes facoltative
   - Eventualmente, inizializzare l'interfaccia utente (UI) e la visualizzazione
5. Utenti esperti possono fare molto di più, ma concettualmente il processo rimane lo stesso



Capacità extra

Alcune delle possibilità offerte da Geant4

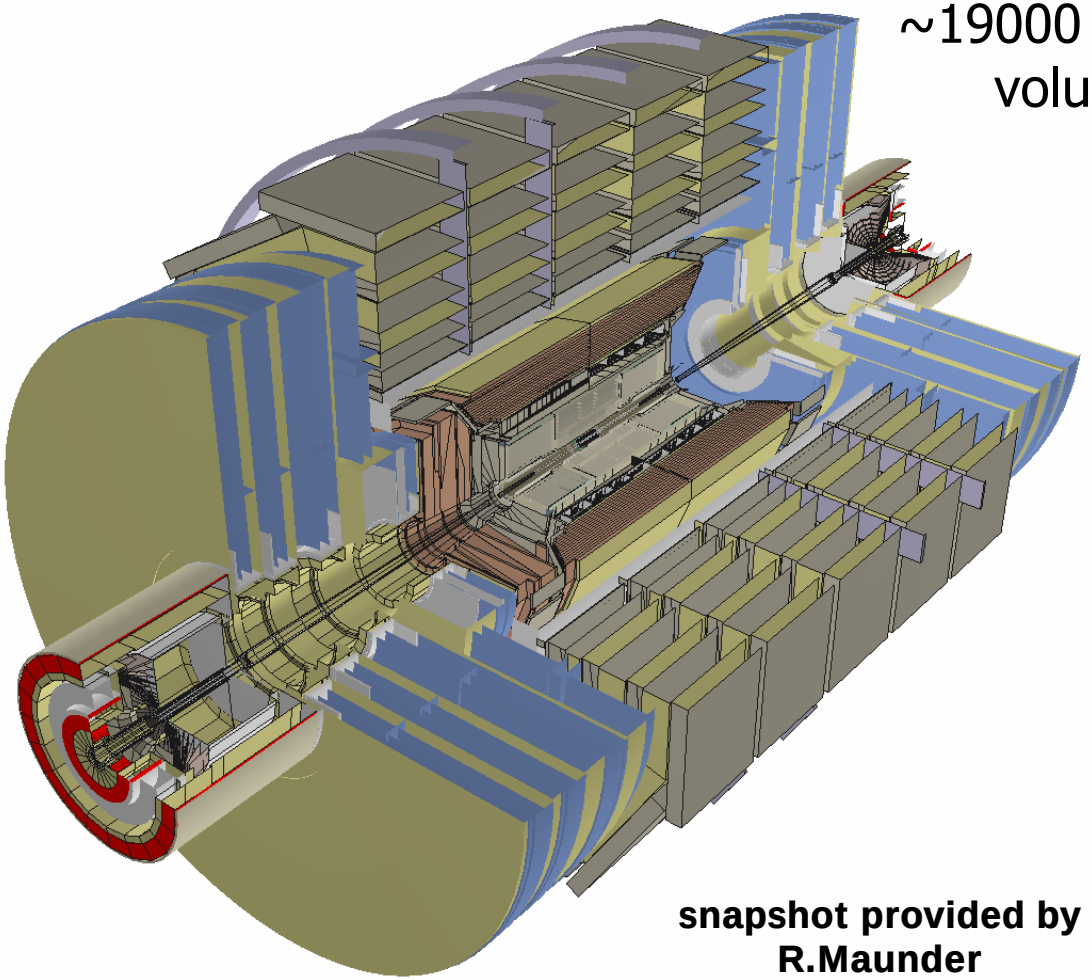
- Il trasporto di ogni particella è fatto **'step-per-step'** considerando le possibili interazioni con materiali e campi
- Il trasporto di una particella continua fino a quando
  - (1) raggiunge energia cinetica nulla (e non ha possibili interazioni in quiete, e.g. cattura o decadimento); (2) sparisce in una interazione; (3) raggiunge i bordi del world volume
- Geant4 consente di accedere alle informazioni sul trasporto/processi e di recuperare i risultati di interesse (USER ACTIONS)
  - All'inizio e alla fine del trasporto
  - Ad ogni step del trasporto
  - Se la particella raggiunge un volume dichiarato "sensibile"
  - .....



GDML

- Possibile importare/esportare geometrie complesse in Geant4 attraverso un linguaggio (tipo XML) chiamato **GDML** (Geometry Description Mark-up Language)
- GDML è utilizzato in diverse applicazioni Geant4:
  - evita di dover scrivere una geometria hard-coded
  - consente di girare la stessa applicazione con geometrie diverse (caricandole da files diversi)
- alcuni programmi (commerciali) consentono di **esportare** geometrie CAD in GDML (via i files STEP)
  - non è immediato ma consente di avere geometrie Geant4 a partire dai disegni CAD

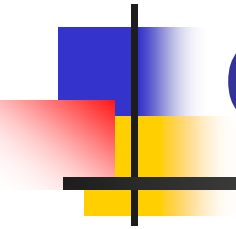
GDML



~19000 physical volumes

snapshot provided by R.Maunder

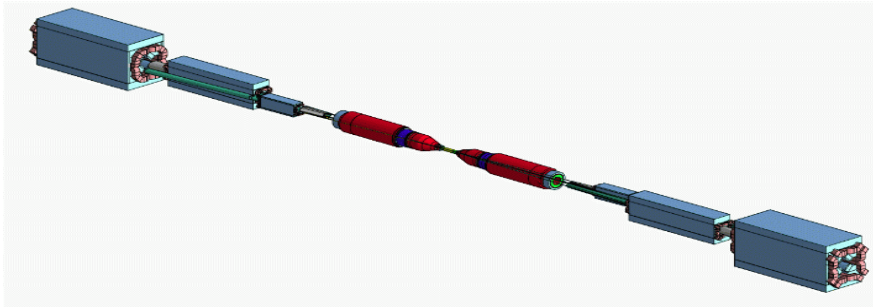
- Geometria di **CMS** importata via GDML
- Anche ROOT è in grado di leggere e disegnare geometrie in formato GDML

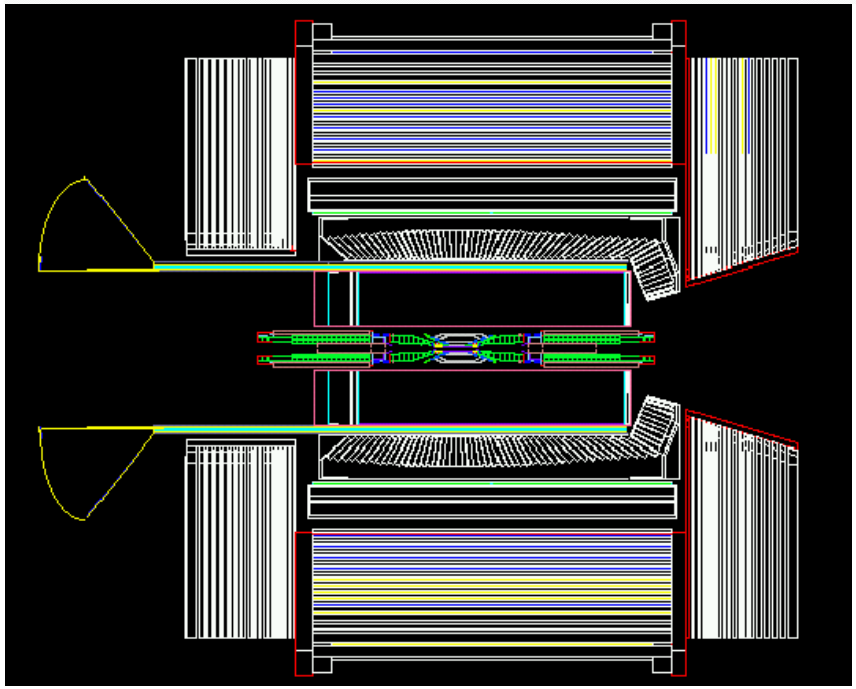


Qualche applicazione di Geant4

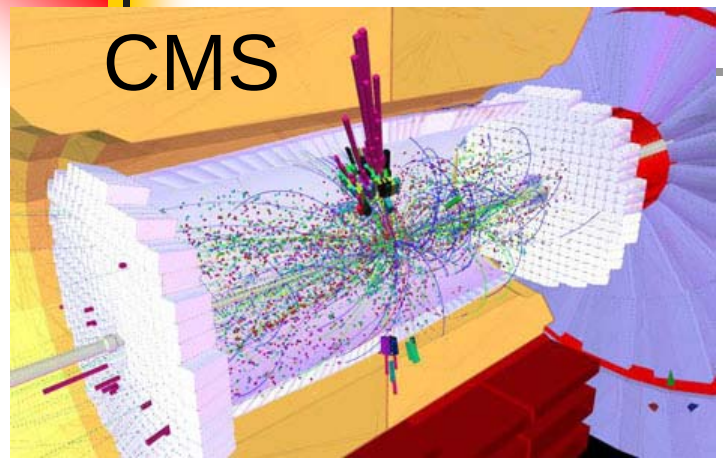
La storia: BaBar (@ SLAC)

- Esperimento pioniere in HEP per l'utilizzo di Geant4
- Iniziato nel 2000
- Simulati oltre $2x10^{10}$ eventi





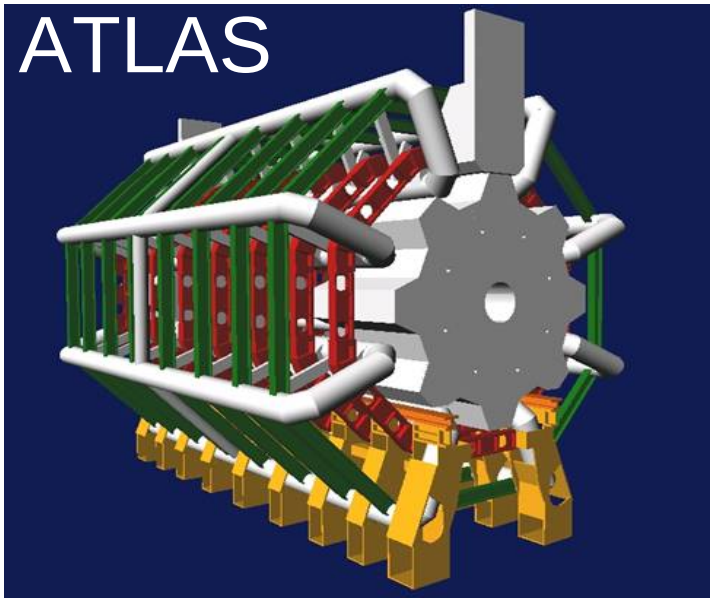
Simulazioni di LHC

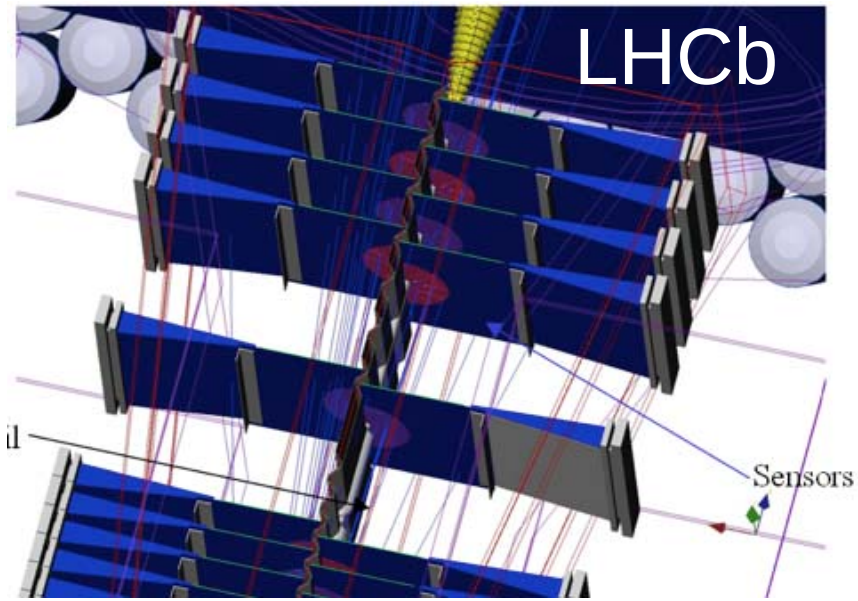


Produzione su **GRID** (computing distribuito)

Generati più di **$2 \cdot 10^9$ eventi**

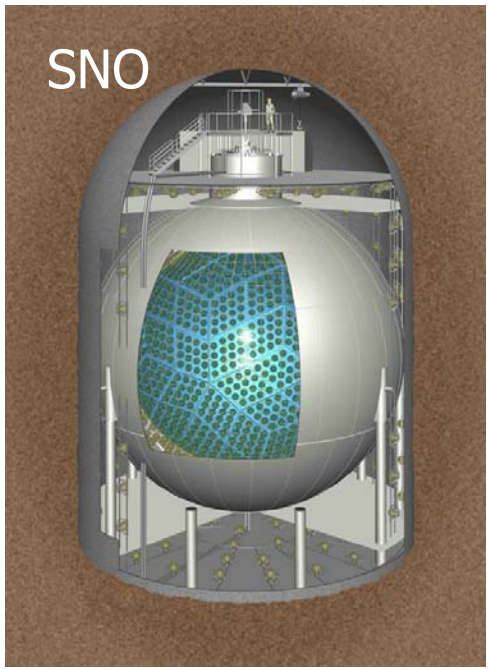
Geant4 utilizzato per **sviluppare** gli algoritmi di **analisi**, correggere accettanza, estrarre curve di calibrazione

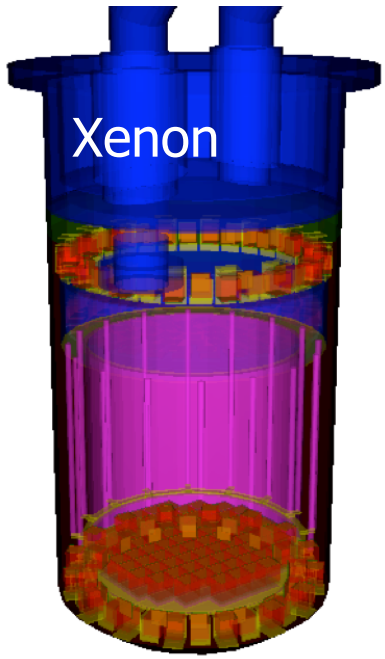


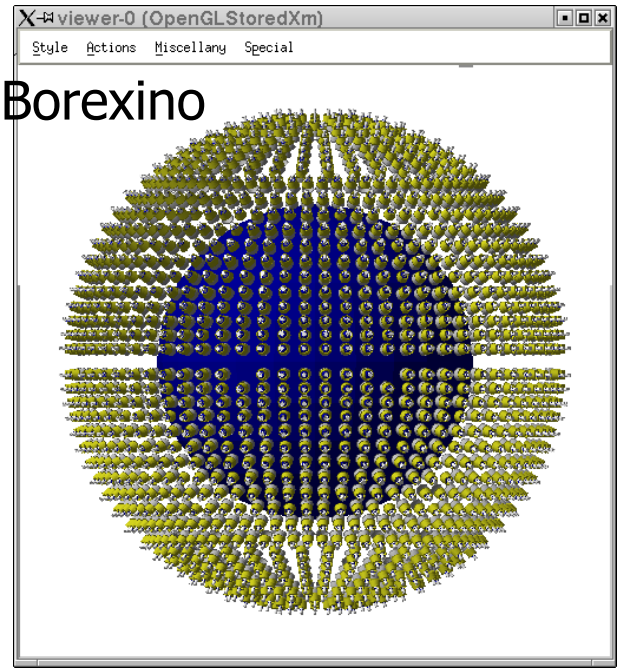


Esperimenti di eventi rari

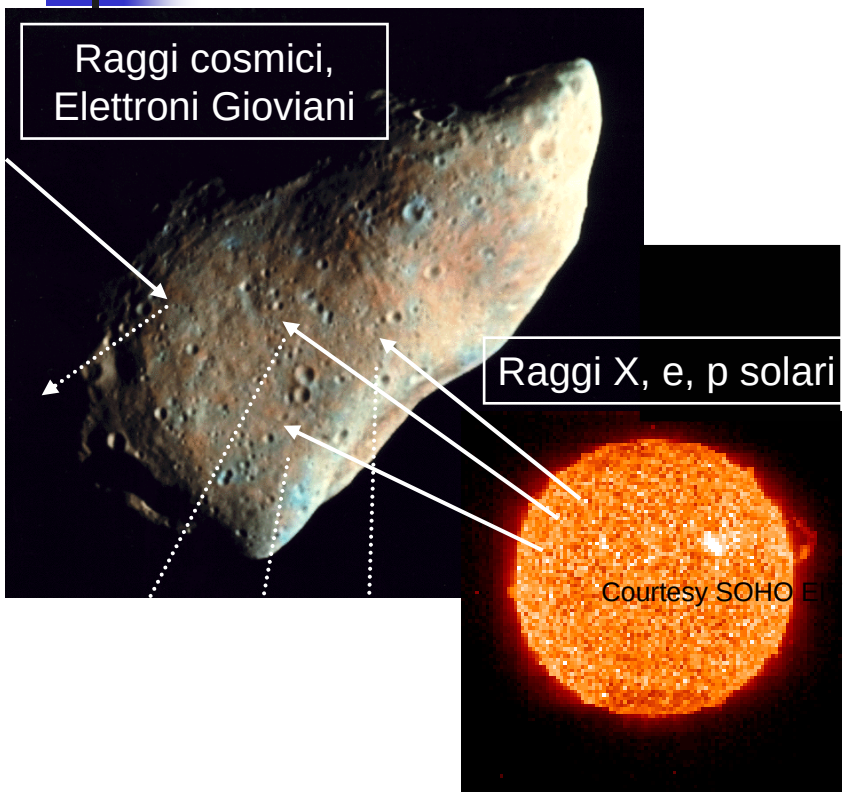
- Scopo principale è la simulazione delle sorgenti di fondo (radioattività, muoni, neutroni)



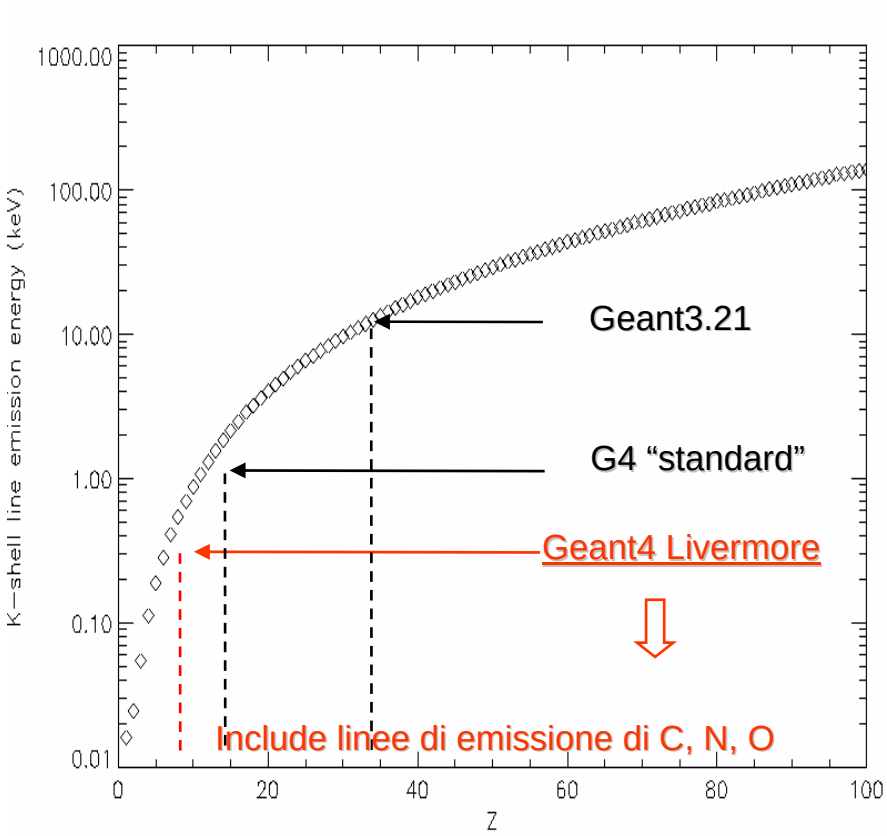




Applicazioni spaziali

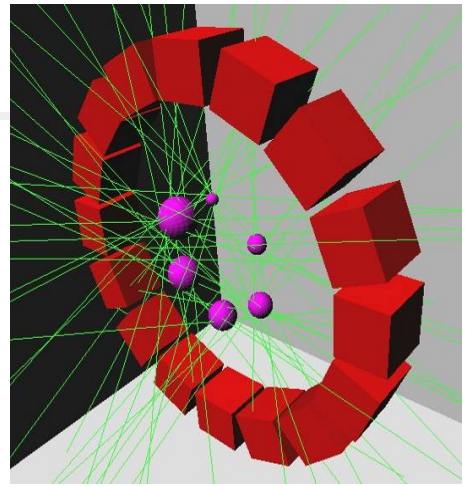


X-Ray Surveys degli Asteroidi e della Luna

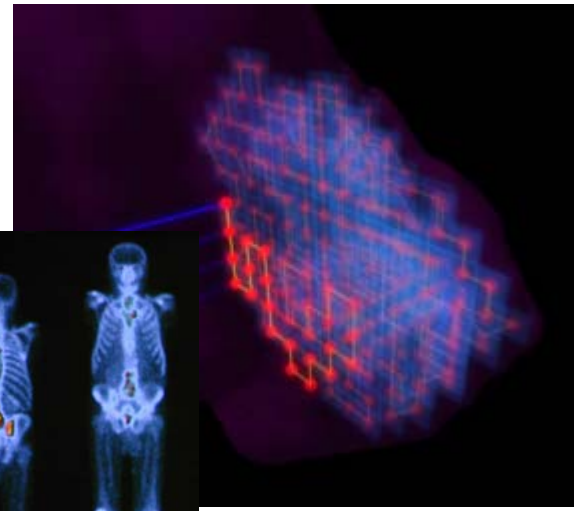


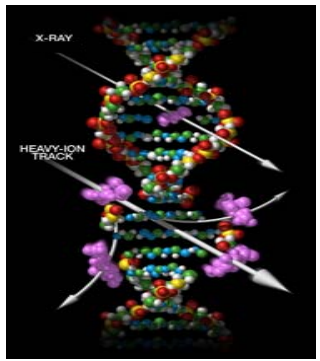
Emissione indotta di raggi X: indicatore della composizione del bersaglio (~100 μm dalla superficie)

Tecniche Monte Carlo per applicazioni mediche



- Perché viene usato il Monte Carlo:
  - Calcolo di distribuzioni di dose, anche in geometrie complesse
  - Simulazione realistica di disomogeneità
  - Verifica dei sistemi di piani di trattamento o dei sistemi dosimetrici
  - Studi e ottimizzazione dei fasci
  - Simulazione di strumenti diagnostici innovativi
  - Studi di danni biologici da radiazione



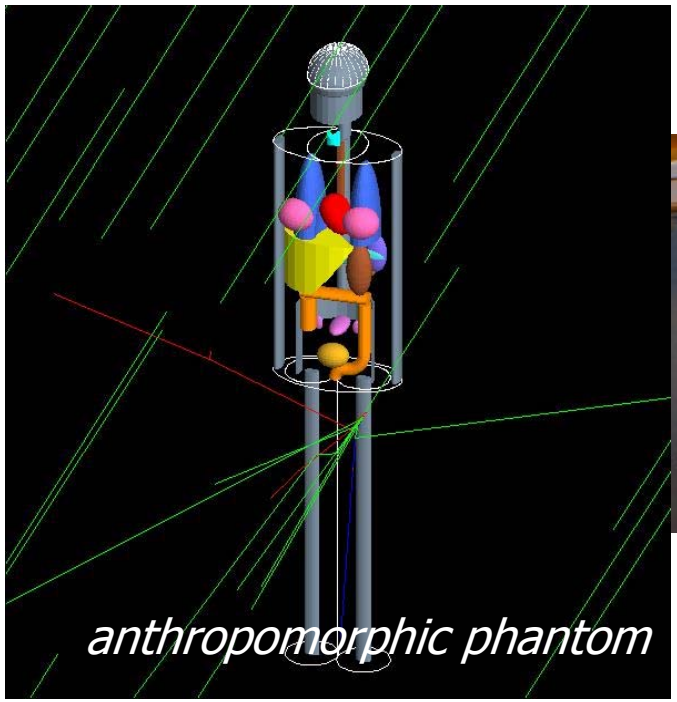


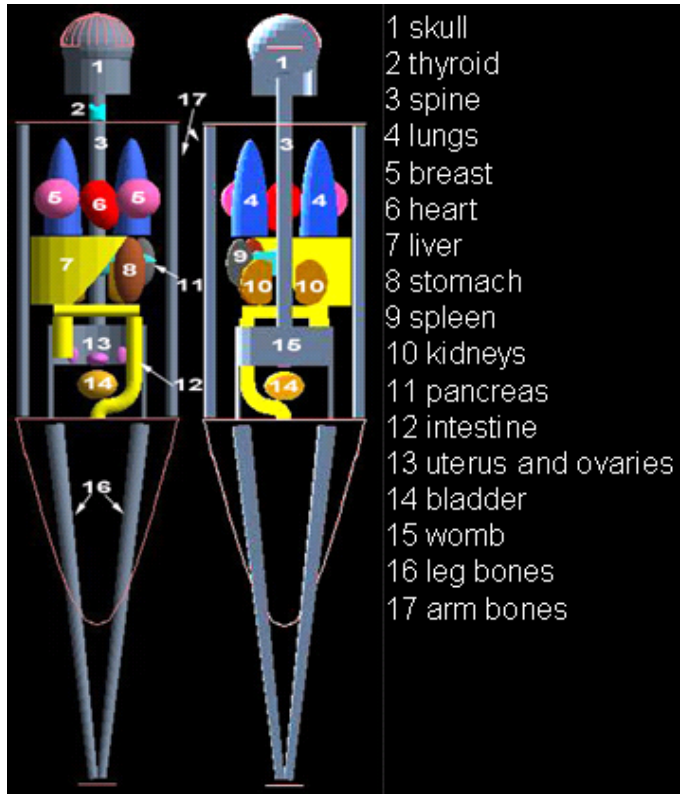
Approccio più preciso attualmente disponibile per queste applicazioni

Total Body Irradiation (Advanced Example di Geant4)

Studio principale in radioprotezione è la **dose accumulata negli organi più a rischio**

Modelli antropomorfi in Geant4 per la valutazione della dose depositata negli organi critici

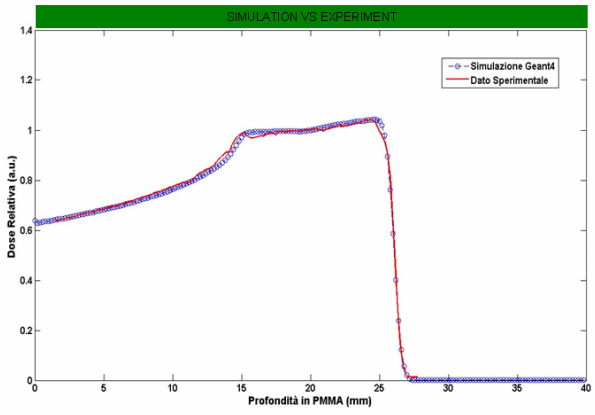


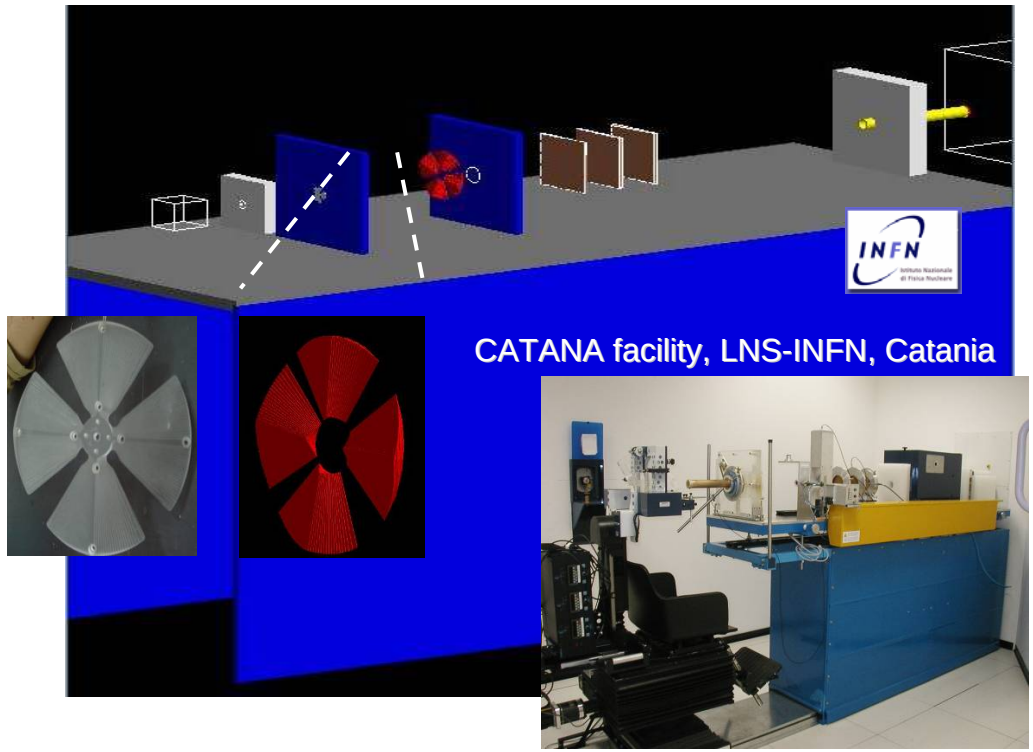


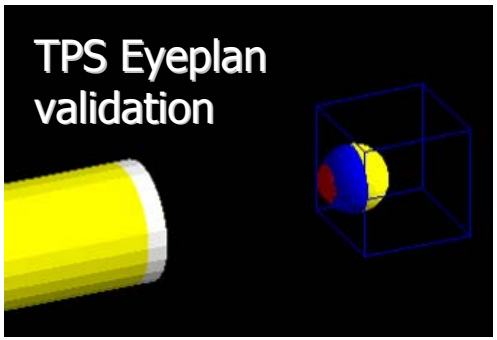
`$G4WORKDIR/examples/advanced/human_phantom`

Hadrontherapy (Advanced Example di Geant4)

- Simulazione di una linea di fascio di protoni per il trattamento dei **tumori oculari** (CATANA)



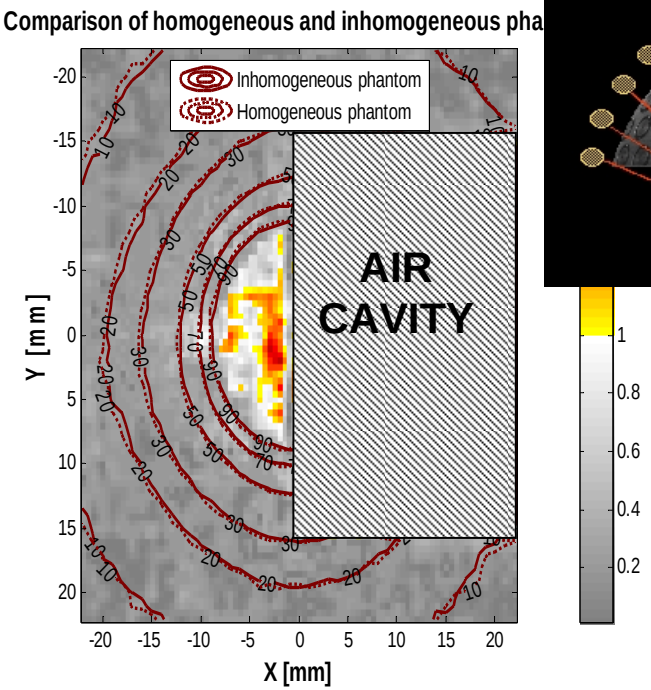


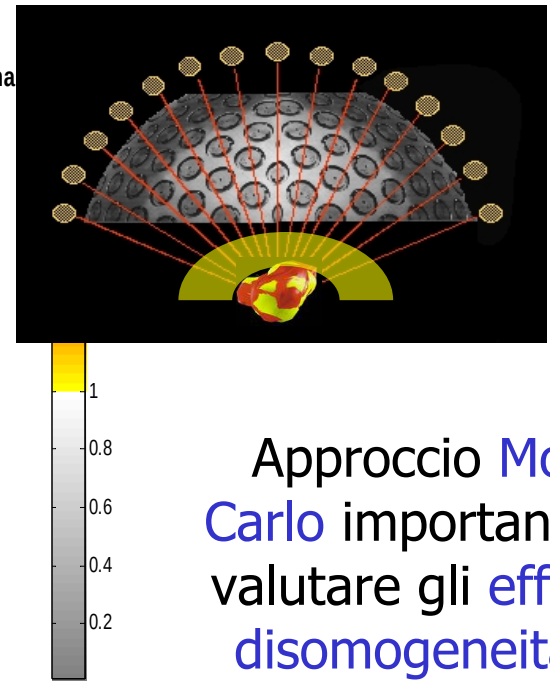


`$G4WORKDIR/examples/advanced/hadrontherapy`

Gamma Knife (Advanced Example di Geant4)

Radioterapia con Gamma Knife usata oer trattare tumori cerebrali spesso inaccessibili alla chirurgia convenzionale → **un solo irraggiamento** ad alta dose





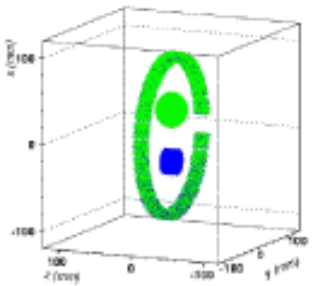
Approccio Monte Carlo importante per valutare gli effetti di disomogeneità → fino al **4%**

201 sorgenti di ^{60}Co in un emisfero (i γ convergono tramite collimatori)

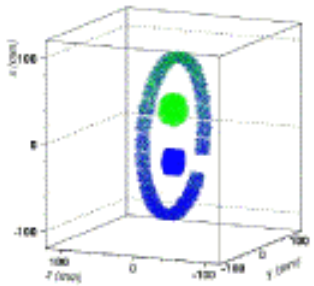
Geant4 Application for Tomographic Emission (GATE)

- Descrizione e gestione accurata di fenomeni **dipendenti dal tempo** (movimento della sorgente o del rivelatore, decadimento delle sorgenti)
- Modellizzazione accurata della risposta dei rivelatori

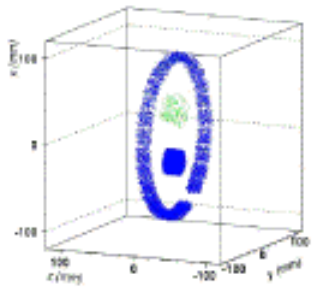
Simulation of the decay of O-15 (in green) and C-11 (in blue) sources throughout 3 time frames



0 - 2 min



7 - 9 min



14 - 16 min