Geant4 theory

Alessandra Filippi
INFN Torino
filippi@to.infn.it

Summary

- Introduction to Geant4
  – General concepts and description of a Gant4 application

- Examples and exercises using Geant4 Geant4

  Definition of the steps necessary for a simulation layout:
    - Geometry
    - Physical processes
    - Generated primary particles
    - Visualization
    - How to get the information on released energies, doses, …

Introduction to Geant4

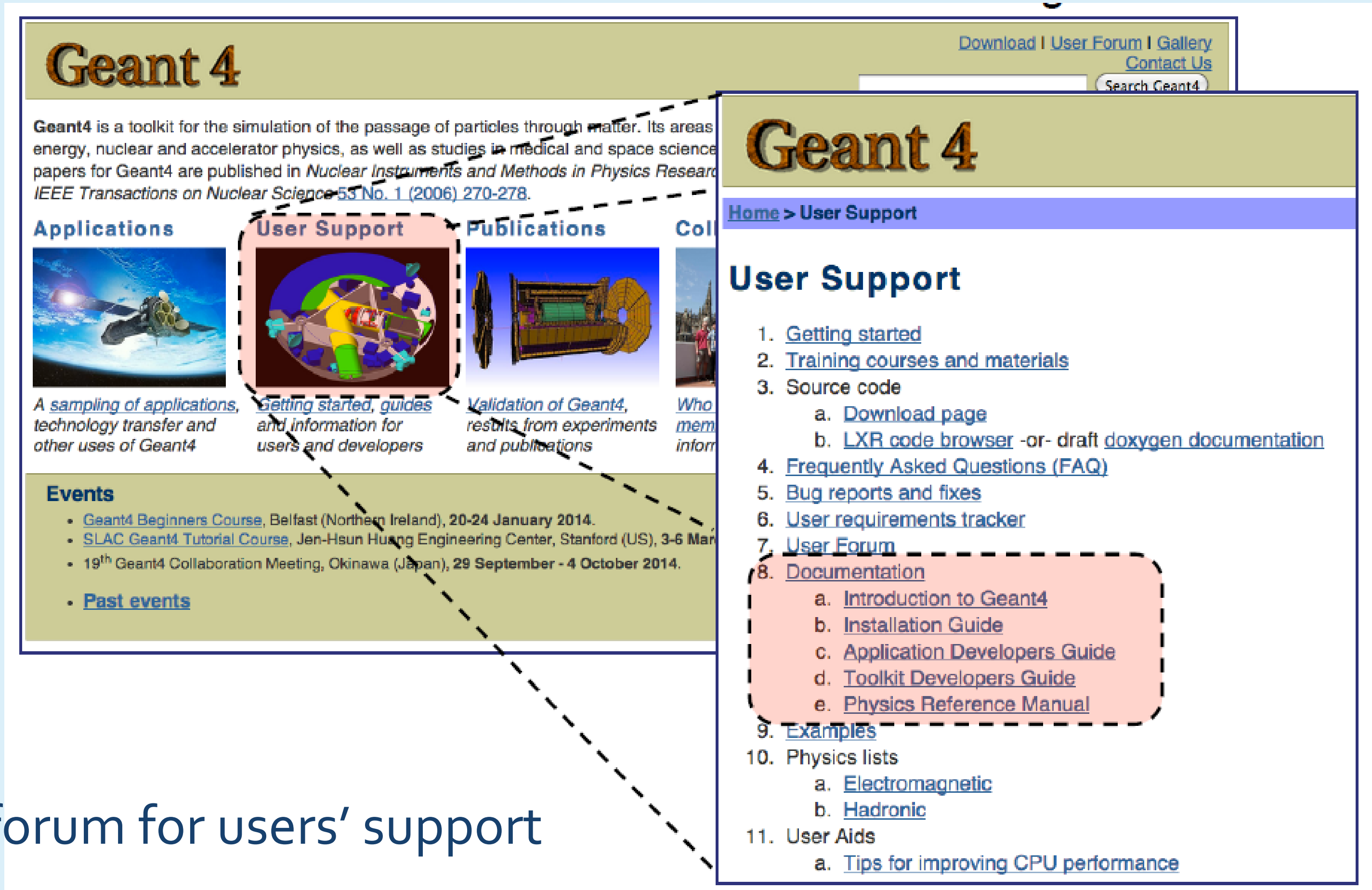
- Geant4 is a very wide and complex toolkit
- Course purpose:
  - Provide a general overview on its usage
  - Give some first tools to start using it
  - Get acquaintance with basic concepts and tools used by Geant4

How to use Geant4

- The users must build some applications (classes and code snippets in c++) based on the existing framework
- Additional instructions can be provided by command line
- Nothing to be invented from scratch! (the framework is too complex to think about doing this)
  - Based on working applications existing design
    - Start typically from an example
  - Changes wherever needed
  - Search for missing information on the Reference Manual (and browse the forum to get some idea when they run out...)
- The user must:
  - Provide for application inizialization
  - Define due actions through suitable User* classes
- First (not trivial) task: framework installation

Distribution

- Geant4 is open-source
- Freely available on http://geant4.cern.ch
  - Source codes, libreries, associated data files (for the simulation of cross sections)



- User forum for users' support

Which information is delivered by Geant4?

- Geant4 **generates particles**, one at a time, and **transports them one by one through sequential steps**, taking into account the physical process to which they can be subject, their interactions with materials and possible electromagnetic fields in which they swim, until the particle
  - Loses all its kinetic energy
  - Disappears due to some interaction
  - Reaches the volume boundaries

- There is a way to interact with the transport process inquiring what is happening it every step and extracting the results from the simulation
  - At the beginning and at the end of the transport
  - At the end of every step along the transport
  - When the particle enters a sensible volume in the setup
    - **Sensible volume: a volume in which we are interested in tracking the particle and determining whether it may have produced a hit**

Which information do we need to pass to GEant4 to define a setup?

- **The user must pass three essential pieces of information**:
  1. Setup **geometry** and **materials** description
  2. **Kinetic information** and on the **primary particles** whose interaction with materials we want to simulate
  3. **Physical processes choice** for the interactions which may occur

- Other possible (not mandatory) pieces of information:
  - Presence of electric and magnetic fields
  - Actions one need to take along the transport (for instance: not to track given particles – not always all generated/produced particles are interesting)
  - Actions one needs to take when the particles enters a given sensible volume

How to we pass the setup information to Geant4?

- Some user-classes which interface to the Geant4 kernel are needed
  - The Geant4 kernel is the “base package” which is not modified by the user, and contains all relevant libraries
- Interact with the kernel through line commands (UI)
  - We will work along this path
    - We will prepare some “macros” containing suitable groups of instructions

What a developer should know to use Geant4 properly?

- C++
  - Geant4 is implemented in c++ language
  - Not necessary to be super-experts
- Object-Oriented technologies
  - Base concepts (depper knowledge neede only if complex applications are needed)
- unix/linux operating system
  - Standard Geant4 working environment, basic knowledge needed at least to compile c++ code and use the most elementary unix commandsed utilizzare i comandi unix più elementari
  - Geant4 can run in Windows through emulators
    - cgwin: some unix knowledge always necessary
- Root as final data analysis tool
  - Some outputs can also be analyzed quickly with excel or jupyter notebooks (python)

Geant4 kernel

Geant4 kernel: Run, Event, Tracking

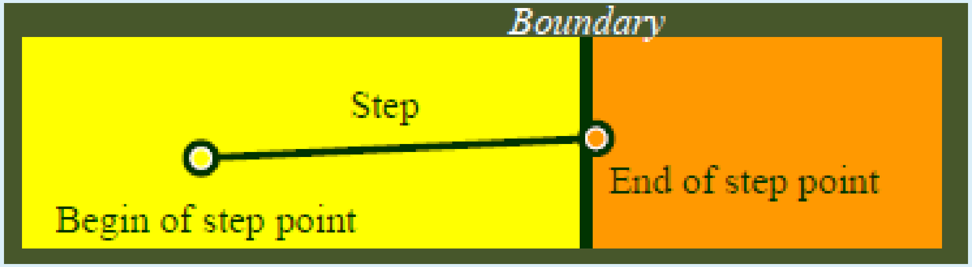
- The Geant4 kernel manages the interaction between the fundamental elements of the simulation **: run, event, track, step, hit, trajectory**
  - It provides a framework for geometry and physical processes representation
- **Run**: collection of **Events** which share the same detectors and physics setting
  - Detector/physics settings fixed in a run
  - **G4RunManager manages G4Run**
- **Event: contains the simulated primary particles**
  - They are generated, stored in a stack and further processed (tracking through detectors/materials, ...)
  - **G4EventManager manages the event processing, G4Event**
- **Tracking: single particles transportation on the volume of interest**
  - Deciouples with requested physical processes (one can add additional ones without interfering with the tracking process)
  - Independent on the particle type
  - Throshold for the production of secondary particles (cuts): NO cuts on tracking
    - **All particles are tracked until zero range**
    - The user can require cuts on the physical observables: energy, TOF, ...

Geant4 kernel: track, step

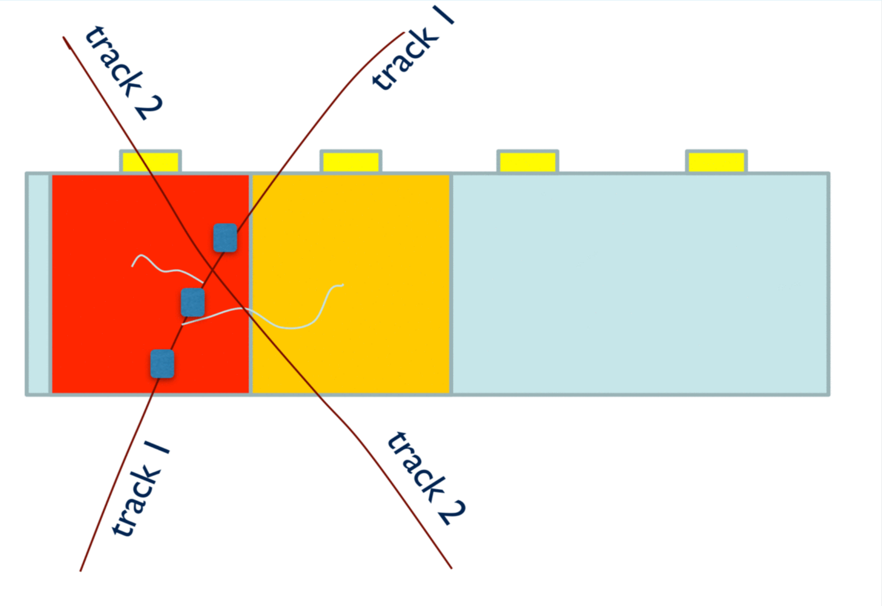
- **Track**: particle snapshot
  - **It delivers the particle position and physical observables**
  - NOT a step collection
  - It is released and cancelled when:
    - The particle gets out of the "World" volume (global reference system)
    - The particle disappears (for instance after a decay)
    - The particles' kinetic energy goes to zero and no other process is required "AtRest"
    - The user deletes it
  - The Track object does not persist beyond the end of the event
    - To keep it in memory use the Trajectory object
  - **G4TrackingManager** manages the track processing, and the track is a **G4Track object**
- **Track vs Trajectory**
- **Step**: track "delta"

Geant4 kernel: step

- **Step**: has two points (start and stop)
  - Brings the differential information on the along the path travelled by the particle
    - Kinetic energy loss, TOF along the step…
  - **In every point the volume and the material being traversed is known**
  - If a step is limited by the end of a volume, the end point lies on the boundary and belongs to the next logical volume (this way the materials of both volumes are known, allowing the simulation of phenomena that occur at the material interface, such as transition radiation or refraction)
  - **G4SteppingManager** manages the step processing, the step being given by a G4Step instance



Geant4 kernel: hit



- The definition of a hit is generally linked to the time window over which the signal deposited by a particle is integrated.

- **Within a given time window, all steps occurring within a certain detector (sensitive volume) constitute a hit.**

- If a second track crosses the volume within the same time window, separate hits are not created; instead, its steps are added to the steps of the previous track, unless the FLUX mode is selected (in which case the hits from the two tracks remain separate).

Geant4 kernel: particle

- **Particle**: a particle information can be included in three classes containing different features

- **G4Track**
  - Class repreenting a particle which must be tracked
  - **Delivers geometrical information** on position, direction, …

- **G4DynamicParticle**
  - **Delivers dynamical information**: momentum, energy, spin, … and represents one particle only
  - Every **G4Track** instance is linked to a unique **G4DynamicParticle** object

- **G4ParticleDefinition**
  - Delivers "static" information: charge, mass, lifetime, decay channels, …
  - The involved processes are managed by **G4ProcessManager**
  - A **G4ParticleDefinition** holds for all **G4DynamicParticle** objects with the same kind of particle

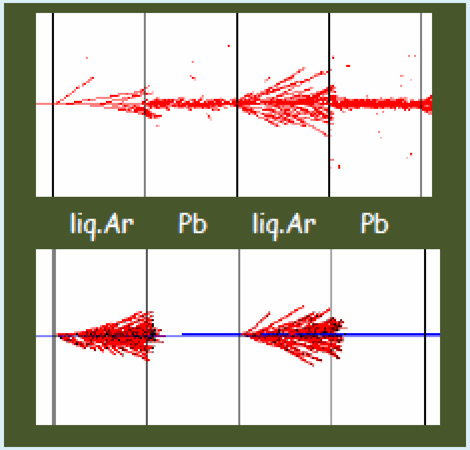
Geant4 kernel: process

- **Process**: Each particle is subject to a series of physical processes contained in a user-provided list: the selection of a process occurs randomly, weighted by the probability of the process occurring (given by its cross section)
  - At the end of each step, all activated processes are invoked to calculate the λ **interaction length**
    - A random number is drawn according to the cumulative distribution of the cross sections, and the corresponding interaction length is calculated
    - The interaction length of a process is related to its cross section, which is tabulated as a function of a certain number of (discrete) variables.
    - For most of the processes: $1/\lambda = \Sigma = N\rho\sigma/A$
      - For in-flight decays: $\lambda = c\beta\gamma\tau$
    - The process with the shortest interaction length is selected, which then determines the step length for the next step.
    - To every process one of these status, or a combination thereof, is assigned:
      - AtRest
      - AlongStep (continuous process)
      - PostStep (discrete process)

Cuts: energia vs range

- **Cut**: represents **a production threshold**, and is evaluated on the ***range***
  - Only for processes with an infrared divergence (**electrons, positrons, photons**)
  - These are not cuts on the tracking and do not enter Geant4 by default
  - Secondary particles are generated only within a specific range
  - Primary particles are tracked until zero range

- Example: 500 MeV/c protons in 4 mm LAr + 4mm Pb
  - Energy cut (Geant3)
    - E_{cut} = 450 KeV
  - Range cut (Geant4)
    - R_{cut} = 1.5 mm



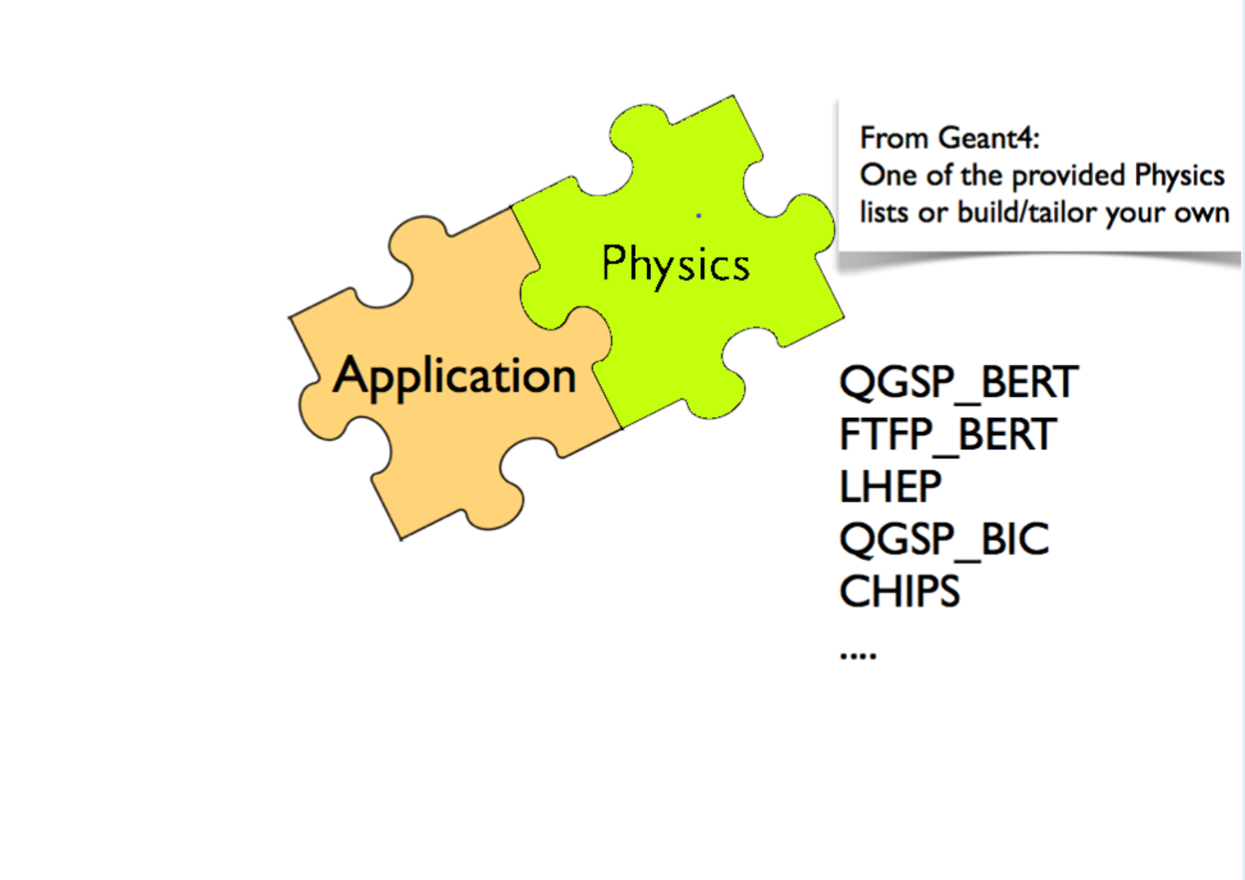
Geant4 units

- Geant4's internal system of units is completely hidden from both the user code and the Geant4 source code
- **Every number in input must be multiplied by its proper unit**
  - radius = 10.0 * **cm**;
  - kineticEnergy = 1.0 * **GeV**;
    - The numerical value is obtained dividing it by its unit
- The user can freely change the working units without affecting the result

How is a Geant4 application structured?

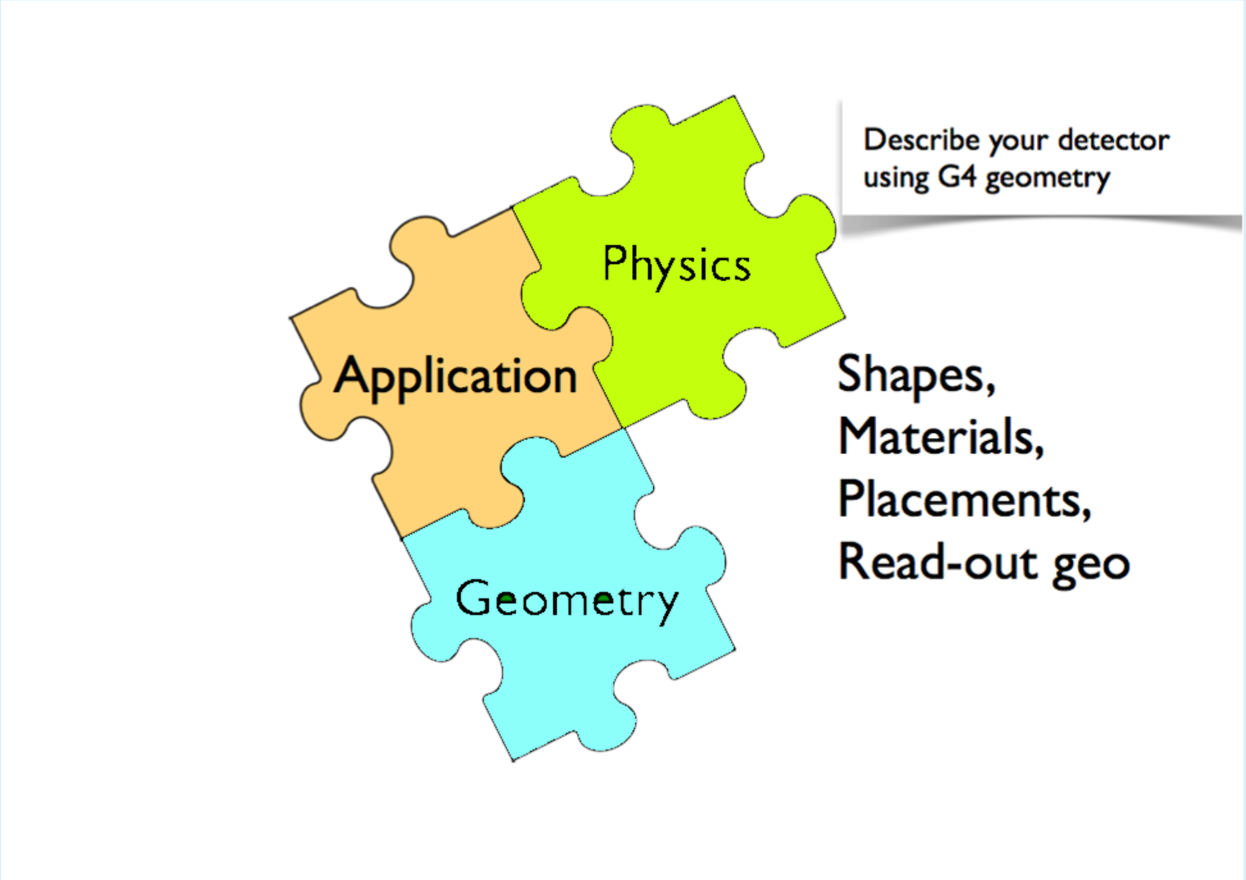
The user must provide a certain number of specifications to the kernel to define the problem

How is a Geant4 application structured?



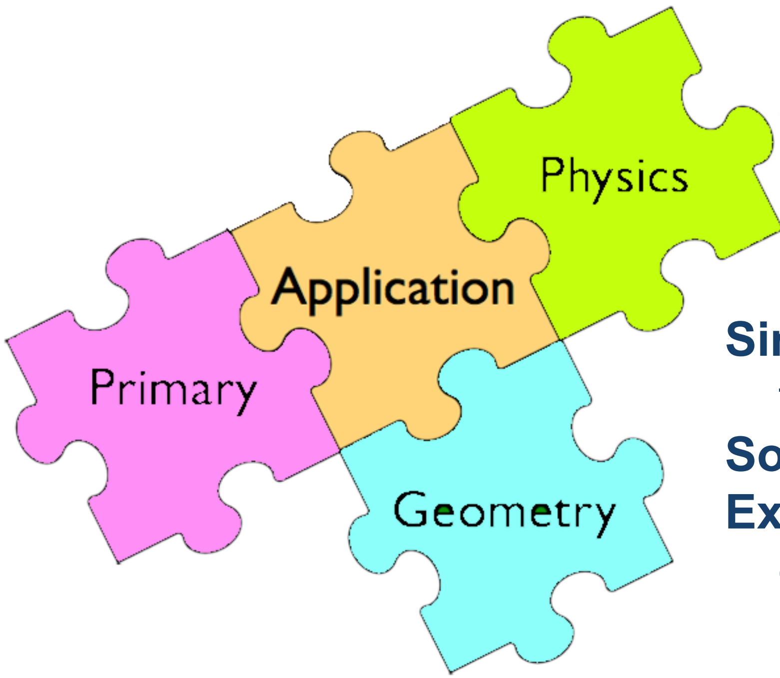
1. The user must provide the list of physical processes to which the particles will be subjected (not all processes are of interest, depending on the specific problem)

How is a Geant4 application structured?



2. The users must provide the setup geometric description

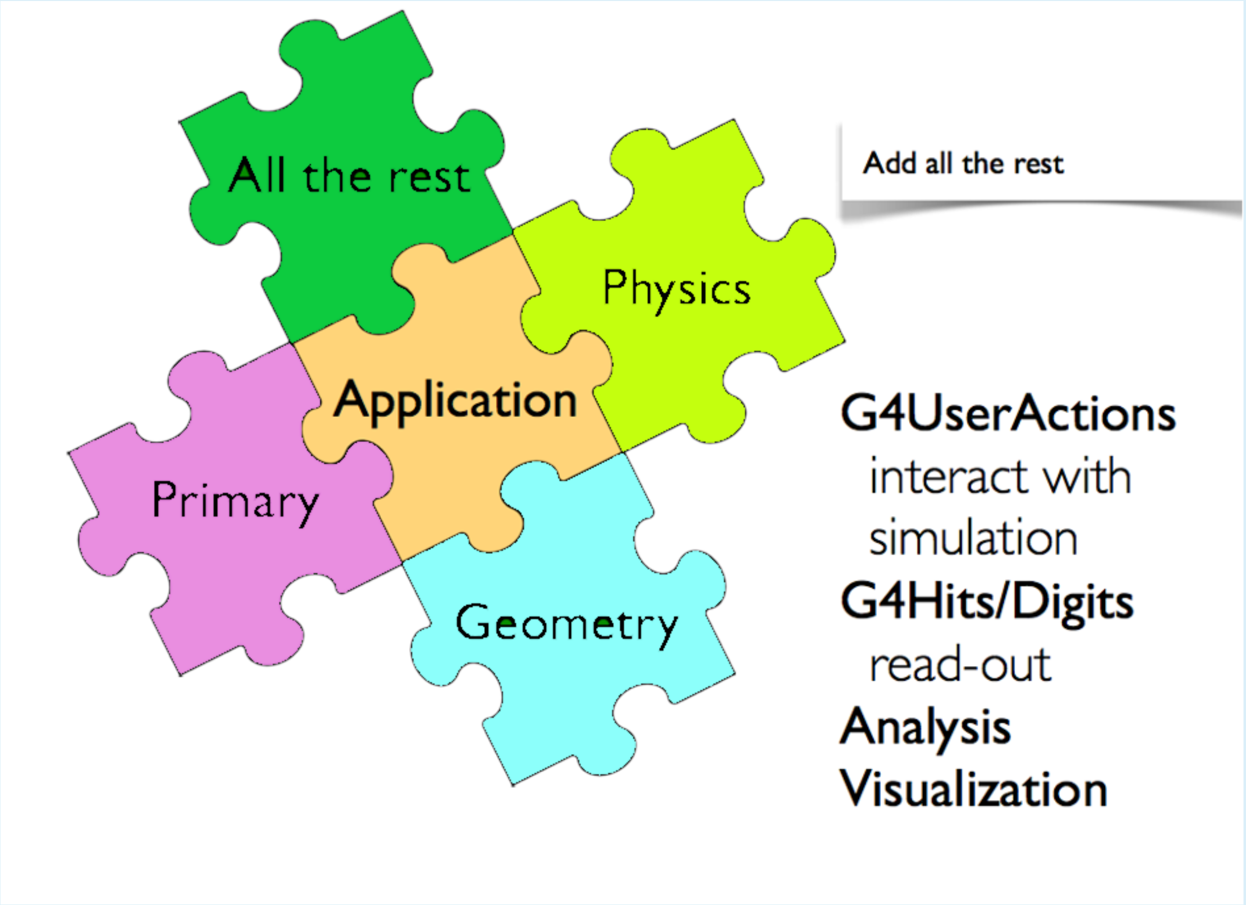
How is a Geant4 application structured?



Simple Gun
test beam like
Source geometry
External input
data files produced by external generators

3. The user must provide a list of the primary particles (for instance: beam type or source and its spatial distribution)

Come è formata un'applicazione Geant4



4.The user must provide instructions on the output format

What must a user do?

- Define the framework
  - setup geometry
    - Materials
    - Shapes and dimensions
    - Hit objects: either sensitive (i.e., capable of storing information about the energy deposited by a particle as it passes through) or non-sensitive
  - Starting interacting particles
    - beam
    - Single particle
    - Energy and direction distributions
    - Kind of particles
  - Physical processes that affect the particle, including any secondary particles it may generate

User Interface (interactive Geant4): a first approach

Geant4 UI commands

- Every UI command consists of
  - A directory
  - A command
  - Parameters: string (" "), boolean, integer, double; they can be omitted (default: ! is placeholder for a default value)
    - `/run/verbose 1`
    - `/vis/viewer/flush`
    - `/dir/command ! Second`

- the GUI commands can be launched
  - Interactively from a command line
  - From a macro file
  - From a hard-coded implementation: for instance in a main()
    - `G4UImanager* UI = G4UImanager::GetUIpointer();`
    - `UI->ApplyCommand("/run/verbose 1");`

- Some commands are available only in some Geant4 states
  - `/run/BeamOn` **only available if the application is in** *Idle* state

Macro Files

- ascii files containing a list of UI commands
- The commands must be written using their full path
- The # symbol comments a line – the commented lines can be transcribed using the instruction
  - **`/control/verbose 2`**
- macros can be run:
  - Interactively or calling them from another macro file
    - **`/control/execute macro.mac`**

(G)UI available commands

- Dull list available in *Users's Guide For Application Developers*:
  - **/control/manual [directory]**

- One can define **alias'es** (which can be used recursively):
  - **/control/alias [name] [value]**

```
/control/alias file1 /diskA/dirX/fileXX.dat
/control/alias file2 /diskB/dirY/fileYY.dat
/control/alias run 1
/myCmd/getFile {file{run}}
```

- Loops:
  - **/control/loop [macroFile] [counterName] [initialValue] [finalValue] [stepSize]**
  - **/control/foreach [macroFile] [counterName] ["valueList"]**
    - Example:

      /control/loop myRun.mac Ekin 10. 20. 2.

      - In **myRun.mac**:
        - **/control/foreach mySingleRun.mac pname "p pi- mu-"**
        - **In mySingleRun.mac:**
          - **/gun/particle {pname}**
          - **/gun/energy {Ekin} GeV**
          - **/gun/beamOn 100**

“shell” commands

- Some shell-like commands are available and can be entered at the prompt
  - They are interpreted at run-time and are not passed to the Geant4 kernel
  - They cannot be used inside a macro
    - `cd, pwd`
    - `ls`
    - `history`
    - `!history-ID`
      - to recall the # HistoryID command
    - `?Uicommand`
      - shows the current value of the parameters for the issued command
    - `help [Uicommand]`
    - `exit`

Geant4: materials and geometry

Materials

- One can define different types of materials:
  - **Description of the properties of atoms:**
    - Isotopies ⇒ **G4Isotope**
    - Elements ⇒ **G4Element**
      - These classes describe the properties of atoms: atomic number, number of nucleons, molar mass, shell energies, cross sections per atom, etc
  - **Descriptin of the macroscopic properties of materials**: G4Material
    - Molecules ⇒ **G4Material**
      - Constructed from different elements (composition by number of atoms)
      - Composite materials and mixtures ⇒ **G4Material**
      - Composition by mass fractions

- To every materials some attributes associated :
  - Temperature
  - Pressure
  - Physical state
  - density
  - Radiation length
  - Absorption length
  - ...

```
a = 14.01*g/mole;
G4Element* elN  =
   new G4Element(name="Nitrogen",symbol="N",z= 7.,a);
a = 16.00*g/mole;
G4Element* elO  =
   new G4Element(name="Oxygen",symbol="O",z= 8.,a);
density = 1.290*mg/cm3;
G4Material* Air =
   new G4Material(name="Air",density,ncomponents=2);
Air->AddElement(elN, 70.0*perCent);
Air->AddElement(elO, 30.0*perCent);
```

```
G4Element* elC  = …;    // define "carbon" element
G4Material* SiO2 = …;   // define "quartz" material
G4Material* H2O = …;    // define "water" material

density = 0.200*g/cm3;
G4Material* Aerog =
   new G4Material("Aerogel",density,ncomponents=3);
Aerog->AddMaterial(SiO2,fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O ,fractionmass=37.4*perCent);
Aerog->AddElement (elC ,fractionmass= 0.1*perCent);
```

NIST material data-base

- NIST material database imported in Geant4 and available
  - http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/Appendix/materialNames.html
- UI commands to import materials
- Best accuracy for material description
  - Elementary materials
  - Composite material
  - HEP e nuclear materials
- It is possible to build NIST composite materials and user-defined materials
- UI commands:
  - **`/material/nist/printElement`**
  - **`/material/nist/listMaterials`**

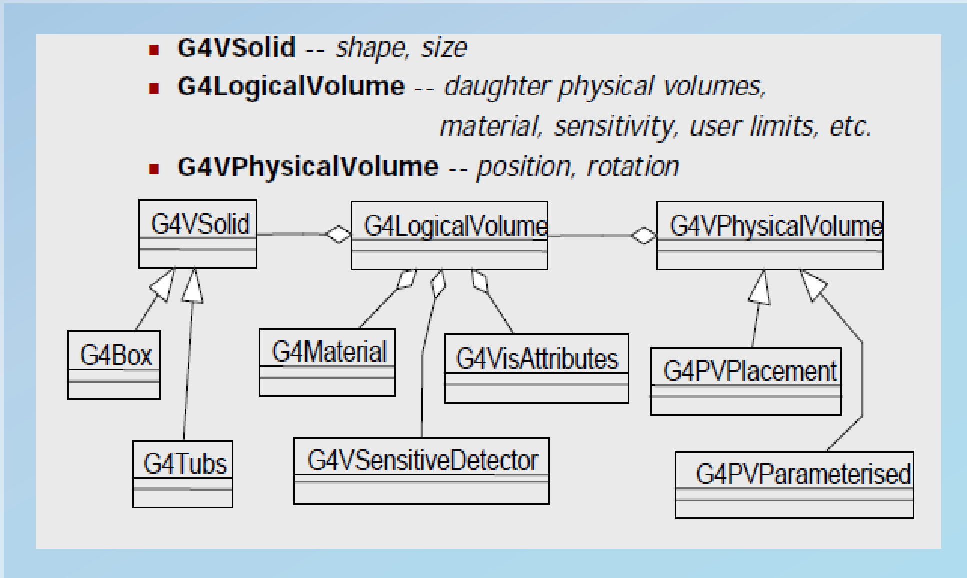
Geometry

- To provide a detailed description of the experimental setup
- **Three conceptual layers**:
  - **Solid**: shape, dimensions
  - **LogicalVolume**: materials, yes/no sensible volume, level in the volume hierarchy
  - **PhysicalVolume**: position, rotation

- Every "detector" (or target) is described in the **Construct() method of G4VUserDetectorConstruction** and is realized in a modular way for every component or sub-component
  - Definition of necessary materials
  - Definition of solid shapes required for the objects' descriptions
  - Volume building and placement in the space
  - Definition and identification of sensible volumes
  - Definition and placement of possible electromagnetic fields
  - Definition of visualisation attributes to associate to every element in a detector

Volume creation

- Choose the solid shape and its dimensions: **Solid**
  - Box, Cylinder, ...
  - See the reference manual for the available solid shapes and how to define them
  - Note: the dimensions refer to the center of the solid (so they generally need to be divided by 2)
- Add shape properties: **Logical Volume**
  - Material
  - e/m fields
  - sensibile volume (y/n)
- Locate the volume: **Physical Volume**
  - In the "world" reference system or another volume ("mother")
  - In a position
  - Multiple copies of the same volume can be positioned using specific functions

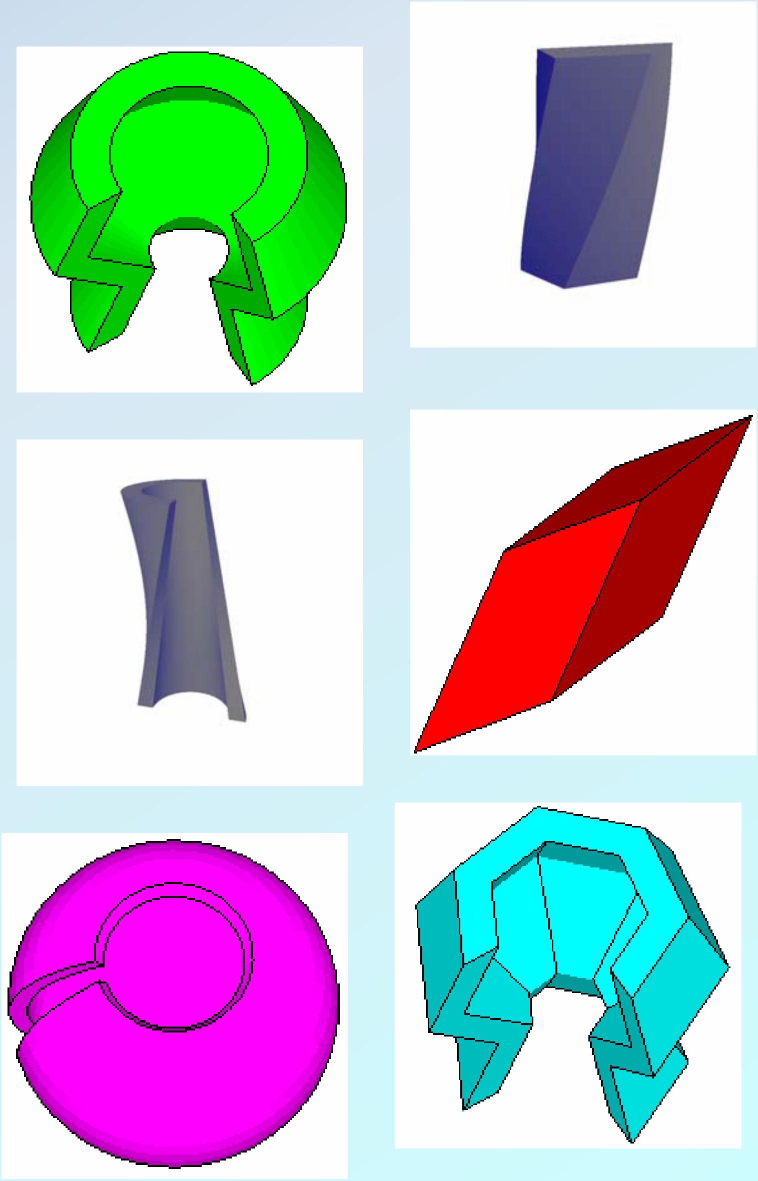


- **World**: A single physical volume that represents the entire experimental area and contains all other volumes
  - **Mandatory**

Solidi: G4VSolid

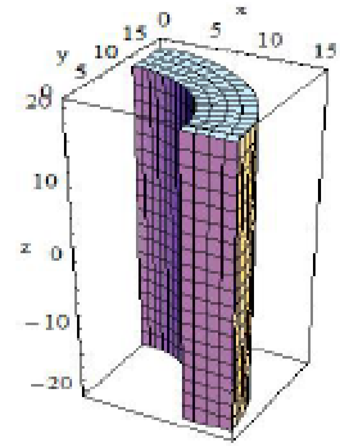
- Several solid shape definitions available in Geant4:
  - CSG (Constructed Solid Geometry)
    - G4Box, G4Tubs, G4Cons, G4Trd
  - Specific CSG-like solids
    - G4Polycone, G4Polyhedra,…
  - BREP (Boundary REPresented) solids

- Multiple volumes can be associated to form more complex volumes: Boolean Solids
  - G4UnionSolid
  - G4SubtractionSolid
  - G4IntersectionSolid
    - Two solids at a time, plus an optional rotation of the second solid relative to the first
    - Navigation through composite solids is time-consuming

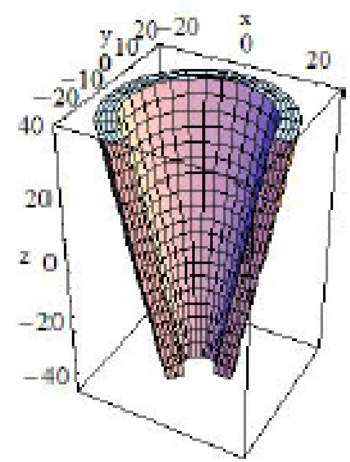


CSG: G4Tubs, G4Cons

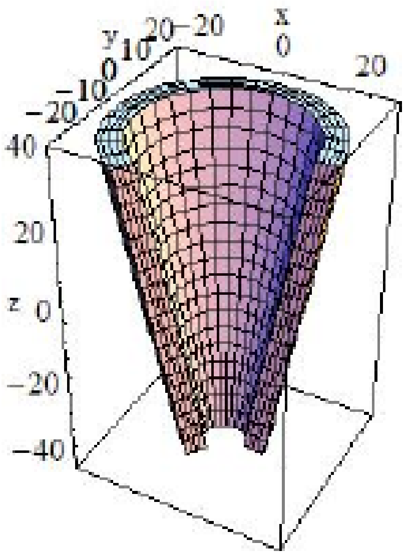
```
G4Tubs(const G4String& pname,   // name
             G4double  pRmin,   // inner radius
             G4double  pRmax,   // outer radius
             G4double  pDz,     // Z half length
             G4double  pSphi,   // starting Phi
             G4double  pDphi);  // segment angle
```



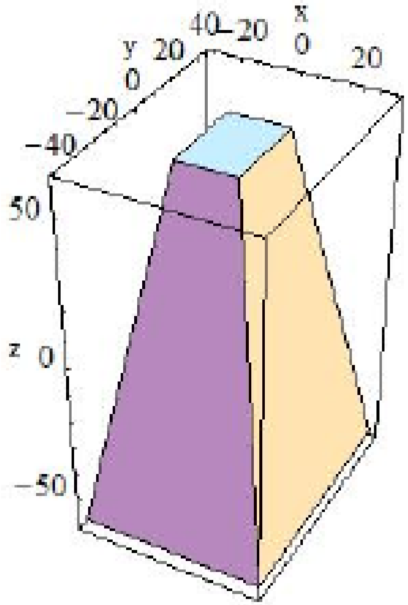
```
G4Cons(const G4String& pname,   // name
             G4double  pRmin1,  // inner radius -pDz
             G4double  pRmax1,  // outer radius -pDz
             G4double  pRmin2,  // inner radius +pDz
             G4double  pRmax2,  // outer radius +pDz
             G4double  pDz,     // Z half length
             G4double  pSphi,   // starting Phi
             G4double  pDphi);  // segment angle
```



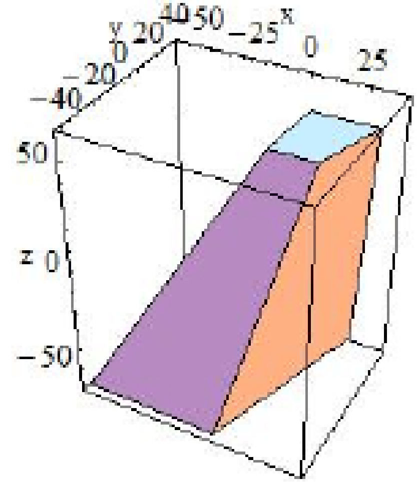
Other CSG solids



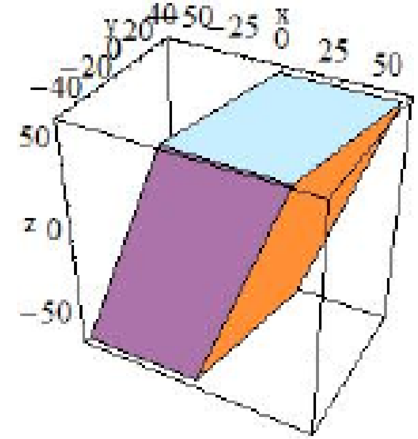
G4Cons



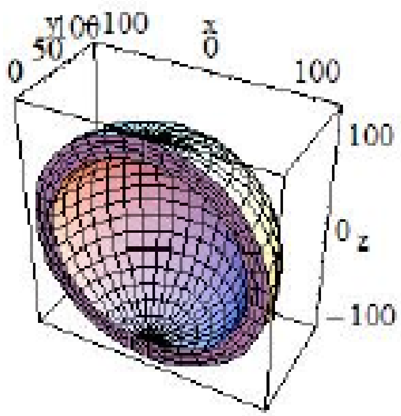
G4Trd



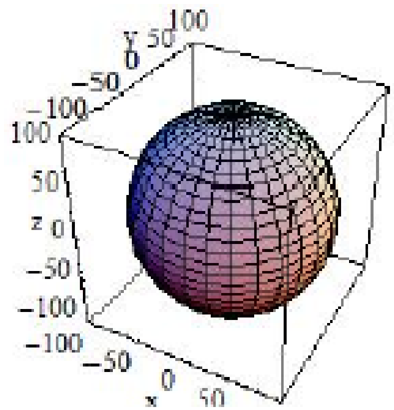
G4Trap



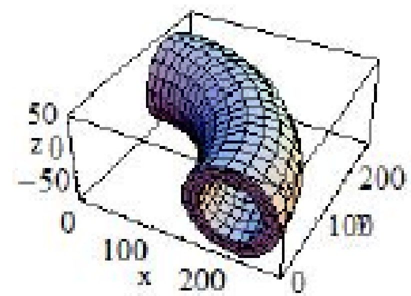
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)



G4Torus

Consult to [Section 4.1.2 of Geant4 Application Developers Guide]for all available shapes.

Logical volumes

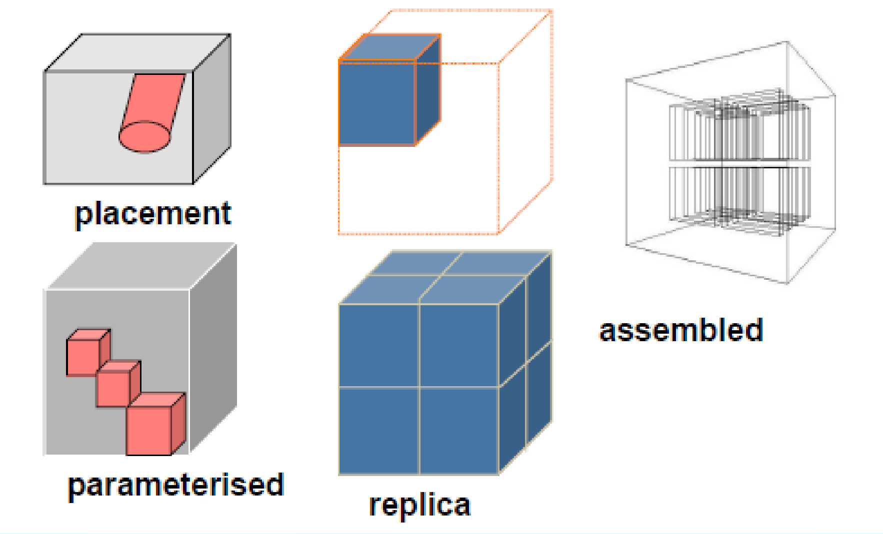
```
G4LogicalVolume(G4VSolid* pSolid, G4Material* pMaterial,
                const G4String& name, G4FieldManager* pFieldMgr=0,
                G4VSensitiveDetector* pSDetector=0,
                G4UserLimits* pULimits=0,
                G4bool optimise=true);
```

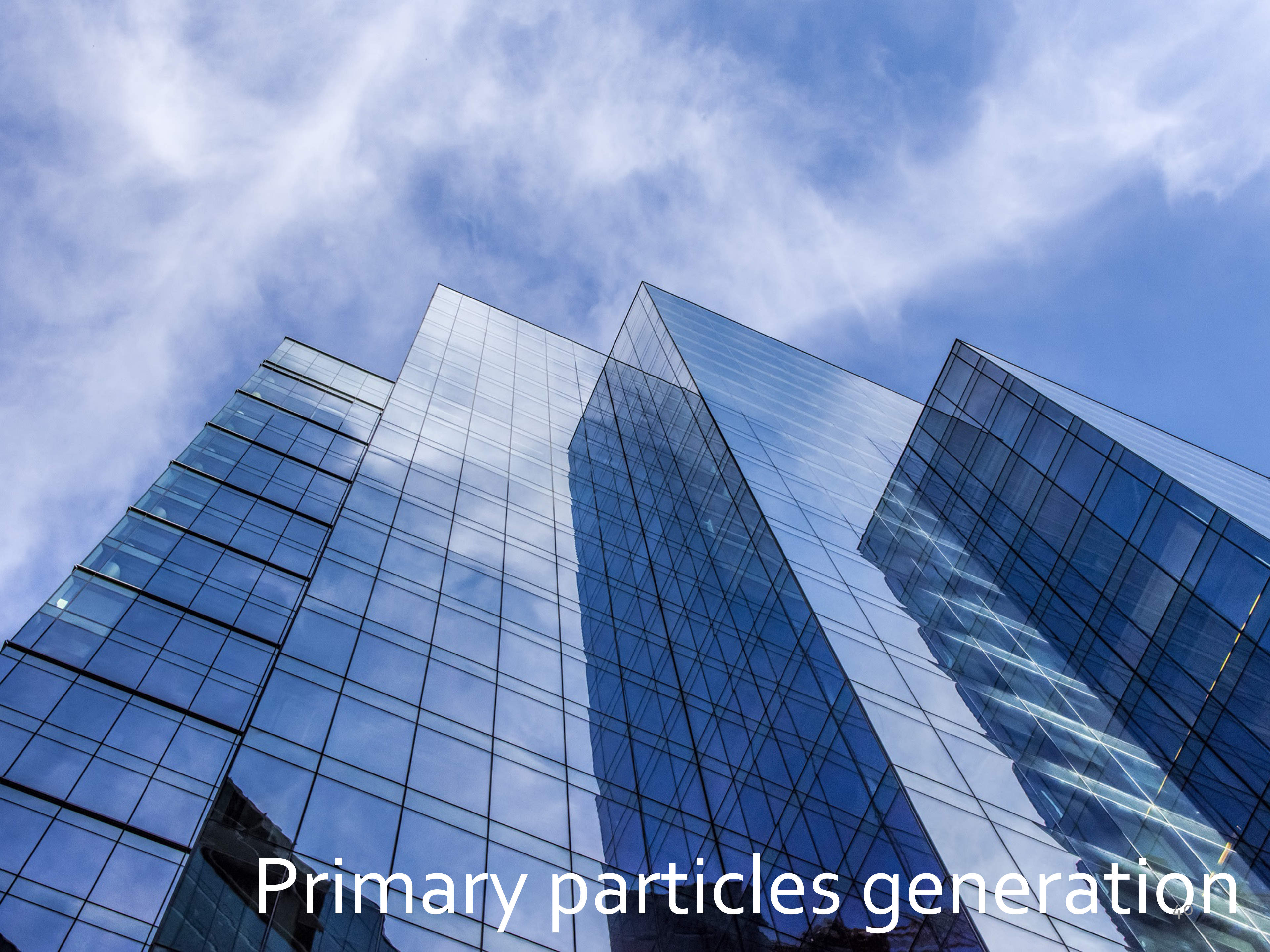
- The definition of a logical volume contains **ALL the information about the volumes except their position**
  - Shape and dimensions (**G4VSolid**)
  - Materials, sensitivity, visualization attributes
  - Position of "daughter" volumes
  - Fields, user-set boundaries
  - Showers parameterizations

- Physical volumes of the same type cannot share the same logical volume

Physical volumes

- G4PVPlacement
  - One volume at a time
- G4PVParameterised
  - Multiple volumes, parameterized accoring the the copy number
  - One can parameterize:
    - Shape
    - Dimensions
    - Materials
    - Position
    - Rotation
  - Volumes must not have further «daughters» and they must be all equal in shape and dimensions
- G4PVReplica
  - 1 replica: multiple volumes
  - The volume is subdivided into smaller pieces (with a symmetry
  - Same repetition
- A “mother” volume can contain
  - Several placements of the same volume *or*
  - A volume repetition





Primary particles generation

Types of particles generators

- **Particle gun**
  - It shoots ONE primary particle at a time, with a given energy, from a certain point at a specific time along a certain direction
  - This can be repeated for multiple primary particles within a single event

- **HEP interface**
  - Reads an ascii file produced by an external event generator and builds **G4PrimaryParticle** object**s**
    - The ASCII file contains the Fortran common block /HEPEVT/ used by almost all external event generators
    - The interaction point must be provided by the user because the placement of the primary particle is not defined

- **General Particle Source GPS**
  - Designed to replace **G4ParticleGun with greater flexibility**
  - It allows specifying multiple sources, each with an independent definition of particle type, position, direction and energy distribution
    - The vertex can be placed on a surface
    - Particles' momentum and directions and be chosen from a distribution
  - Distributions defined through UI commands

G4GeneralParticleSource (GPS)

- Main Commands:
  - `/gps/pos/type`
    - planar, point, …
  - `/gps/ang/type`
    - iso, planar wave, …
  - `/gps/energy/type`
    - monoenergetic, linear, User defined, …
- Differences against ParticleGun:
  - UI control
  - Possibility to generate from surfaces or volumes
  - Ability to randomly generate kinetic energy, position, and direction, even according to a distribution not defined analytically (histogram)
  - More powerful than ParticleGun (which generates one particle at a time)
  - Useful for generating radioactive sources

- Source 1: point-like source, 100 MeV proton, along z
  - /gps/pos/type point
  - /gps/particle proton
  - /gps/energy 100 MeV
  - /gps/direction 0 0 1
- Source 2: plane source (2x2 cm), 100 MeV proton, along z
  - /gps/pos/type plane
  - /gps/pos/shape square
  - /gps/pos/centre x y z
  - /gps/pos/Halfx
  - /gps/pos/Halfy
- Source 3: gaussian-like (sigmax and sigmay = 2cm), 100 MeV proton, along z
  - /gps/pos/shape Circle
  - /gps/pos/centre x y z
  - /gps/pos/sigmax 2 cm

Physics processes selection

Physics processes selection (G4VUserPhysicsList)

- Geant4 has not a default list of physics processes or particles
- It is necessary to explicitly define in which processes the particle can be involved, even for simple transport
  - It is necessary to define all required particles
  - It is necessary to define all processes and assign each to the appropriate particle
- Generally, the user chooses from pre-packaged 'physics lists', which combine relevant physical processes and models for a specific type of application
  - It is possible to create 'physics lists' tailored to one's own needs.
  - It is possible to add processes or models to existing physics lists
- One must have a good understanding of the processes to be simulated (and which processes are of interest in the specific case)
  - In no application is there interest in simulating EVERYTHING.
- **Mandary object** to be provided in every application

Which physical processes are available in Geant4?

- Electromagnetic interaction
  - standard HEP processes in the range ~keV to PeV
  - Low energy processes: ~100 eV – PeV
  - Geant4-DNA: ~eV range (water only)
  - Optical photons

- Weak interaction
  - decays
  - Nuclear radioactive decays

- Hadronic physics
  - Standard string interaction from zero to ~TeV
  - Electro- and photonuclear interactions in the range 10 MeV to ~TeV
  - Neutron interactions (with high precision) from thermal energies to ~20 MeV

- For each process, multiple models may be available, each with different cross sections
  - Continuous update based on new data/measurements

Predefined modular Physics Lists

- Some "standard" EM physics constructors:
  - G4EmStandardPhysics - standard, for "higher energy physics"
  - G4EmStandardPhysics_option1 - for HEP, fast but not precise settings
  - G4EmStandardPhysics_option2 - for HEP, experimental
  - G4EmStandardPhysics_option3 - for medical and space science applications
  - G4EmStandardPhysics_option4 - most accurate EM models and settings
- Some "low energy" EM physics constructors:
  - G4EmLivermorePhysics
  - G4EmLivermorePolarizedPhysics
  - G4EmPenelopePhysics
  - G4EmDNAPhysics
- The complete list can be found in your toolkit:

  geant4/source/physics_lists/constructors/ ==> all built in CTR

  geant4/source/physics_lists/constructors/electromagnetic

  geant4/source/physics_lists/constructors/hadron_elastic

  geant4/source/physics_lists/constructors/hadron_inelastic
- More information at:

  geant4/source/physics_lists/constructors/xxx/README

  http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

Physics lists packages

- Pre-packaged, ready-to-use physics lists are available for use in certain reference cases
- Each list includes different combinations of hadronic and EM (electromagnetic) physics
  - Hadronic interactions are complex to handle, follow different models, and choosing the most appropriate one for a given case is not trivial.
- Available lists can be browsed in:
  - `/geant4/source/physics_lists/lists/include`
- They should be understood as a model or starting point; the user must then validate their use in their application and decide whether the set of processes is sufficient, if some can be removed, or if others need to be added
- Lists provided and maintained by large experiments, with very stable information

Packaged physics lists /I

- **Some Hadronic options:**
  - "QGS" Quark Gluon String model (> ~15 GeV)
  - "FTF" FRITIOF String model (> ~5 GeV)
  - "BIC" Binary Cascade model (< ~10 GeV)
  - "BERT" Bertini Cascade model (< ~10 GeV)
  - "P" `G4Precompound` model used for de-excitation
  - "HP" High Precision neutron model (< 20 MeV)
- **Some EM options:**
  - No suffix: standard EM i.e. the default `G4EmStandardPhysics` constructor
  - "EMV" `G4EmStandardPhysics_option1` CTR: HEP, fast but less precise
  - "EMY" `G4EmStandardPhysics_option3` CTR: medical, space sci., precise
  - "EMZ" `G4EmStandardPhysics_option4` CTR: most precise EM physics
- **Name decoding:** String(s)_Cascade_Neutron_EM
- **The complete list of pre-packaged physics list with detailed description can be found in the documentation ("*Guide for Physics Lists*"):**
  - http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsListGuide/html/index.html

Packaged physics lists /II

- **FTFP_BERT:**
  - Recommended by Geant4 developers for HEP applications
  - Includes the standard EM physics i.e. `G4EmStandardPhysics` CTR
  - "FTF" FRITIOF string model (> 4 GeV)
  - "BERT" Bertini Cascade model (< 5 GeV)
  - "P" `G4Precompound` model used for de-excitation

- **QGSP_BIC_HP(_EMZ):**
  - Recommended for medical applications (experimental QGSP_BIC_AllHP)
  - "QGS" Quark Gluon String model (> 12 GeV)
  - "FTF" FRITIOF String model (9.5 - 25 GeV)
  - "P" `G4Precompound` model used for de-excitation
  - "BIC" Binary Cascade model (200 MeV - 9.9 GeV)
  - "HP" High Precision neutron model (< 20 MeV)
  - "EMZ" `G4EmStandardPhysics_option4` CTR (or EMY that's a bit less precise)

Electromagnetic interactions

- A process can be described using different models, which are alternative and may be complementary
- Different models can be combined in order to use the one most appropriate for the considered energy range
- Each interaction is described by a class following the following naming convention:
  - `G4ModelNameProcessNameModel`
- For instance for Compton interaction different models exist:
  - `G4KleinNishinaComptonModel`
  - `G4LivermoreComptonModel` (low energy)
  - `G4PenelopeComptonModel` (low energy)
  - `G4LivermorePolarizedComptonModel` (low energy+polarization)
  - `G4PolarizedComptonModel` (Klein-Nishina+polarization)

EM interactions packages

Package	Description
Standard	γ-rays, $e^{\pm}$ up to 100 TeV, Hadrons, ions up to 100 TeV
Muons	Muons up to 1 PeV
X-rays	X-rays and optical photon production
Optical	Optical photons interactions
High-Energy	Processes at high energy (> 10 GeV). Physics for exotic particles
Low-Energy	Specialized processes for low-energy (down to 250 eV), including atomic effects
Polarization	Simulation of polarized beams

Particle	Process	G4Process
Photons	Gamma Conversion in $e^{\pm}$	`G4GammaConversion`
	Compton scattering	`G4ComptonScattering`
	Photoelectric effect	`G4PhotoElectricEffect`
	Rayleigh scattering	`G4RayleighScattering`
$e^{\pm}$	Ionisation	`G4eIonisation`
	Bremsstrahlung	`G4eBremsstrahlung`
	Multiple scattering	`G4eMultipleScattering`
e^{+}	Annihilation	`G4eplusAnnihilation`

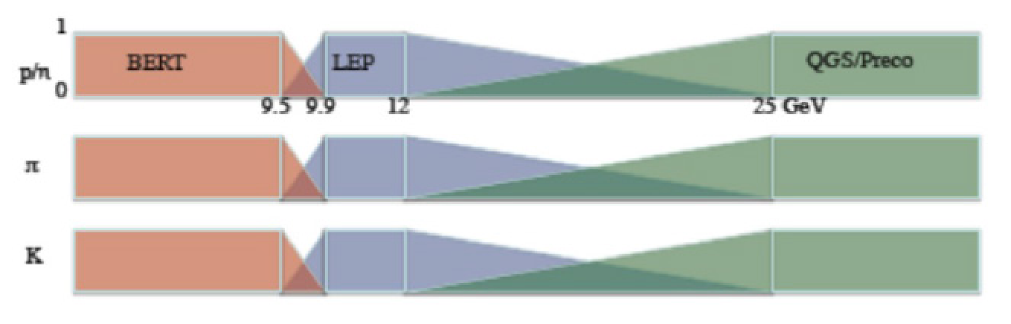
Which EM packages for low-energt processes

- Low energy EM models: Livermore or Penelope
- To be used when:
  - A precise treatment of electromagnetic showers and low-energy (~keV) interactions is required
  - One is interested in atomic effects, such as X-rays from fluorescence, Doppler broadening, etc
  - One may be dealing with a simulation that is very CPU-intensive
  - One wants to compare the results with other simulations (for example, using different models)
- Not advisable when:
  - One is interested in electromagnetic physics at the MeV scale (and above)
    - The same results as with the standard EM models are obtained, but with much lower efficiency

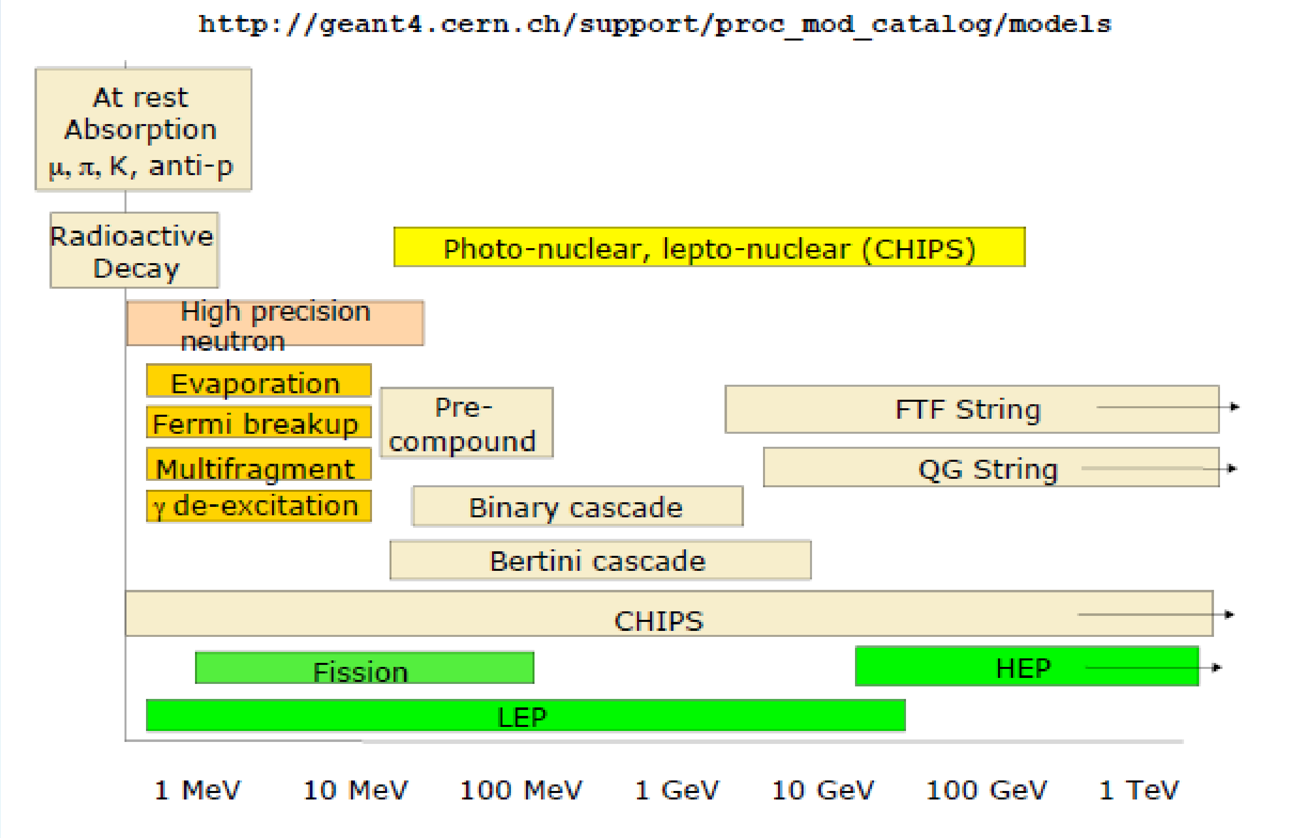
Hadron physics

- three energy ranges
  - < 100 MeV
  - Resonances and decays region: 100 MeV-10 GeV
  - > 20 GeV (QCD strings)
- In every range many models can be used, and many of them are phenomenological
- A few families of lists:
  - QGS
    - Based on the Quark-Gluon String model for high-energy hadronic interactions of protons, neutrons, pions, and kaons.
  - FTF
    - Based on FRITIOF (a string-like model) for protons, neutrons, pions, and kaons
  - Other specialized lists
- Default cross sections:
  - For low energy neutrons:
    - G4NDL available as external data file, with/without thermal cross-sections
  - Proton and neutron interactions cross sections: 20 MeV-20 GeV
  - Ion-nucleus cross sections: E/A < 1 GeV
  - Isotopes production data
  - G4NeutronData libraries

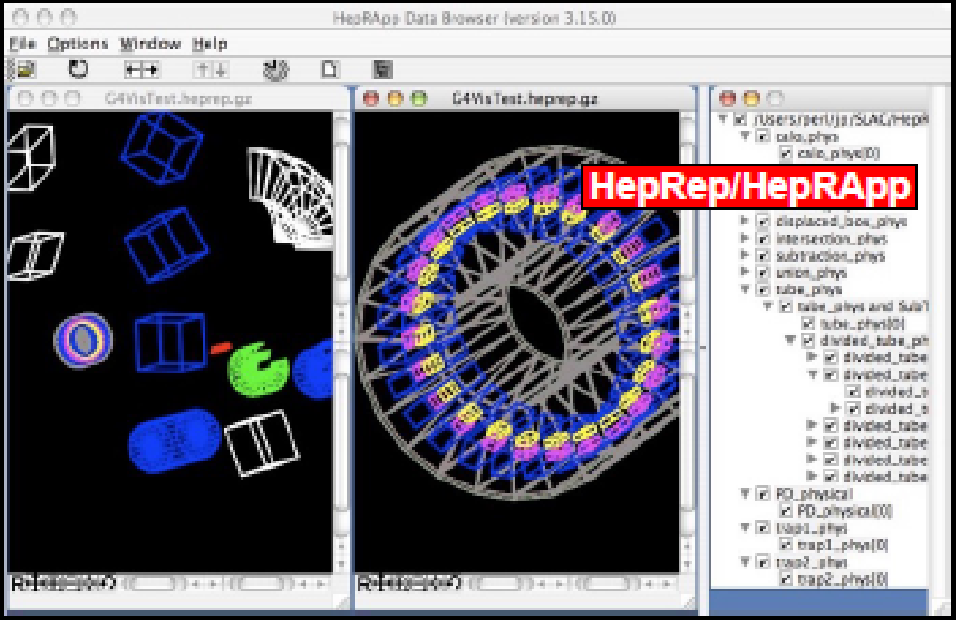
Available hadronic processes and models

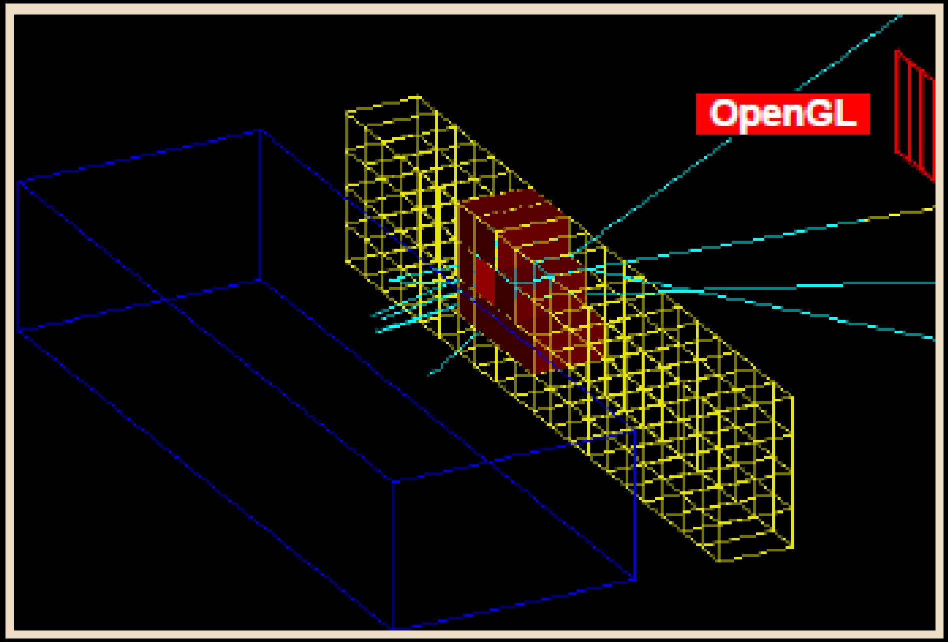


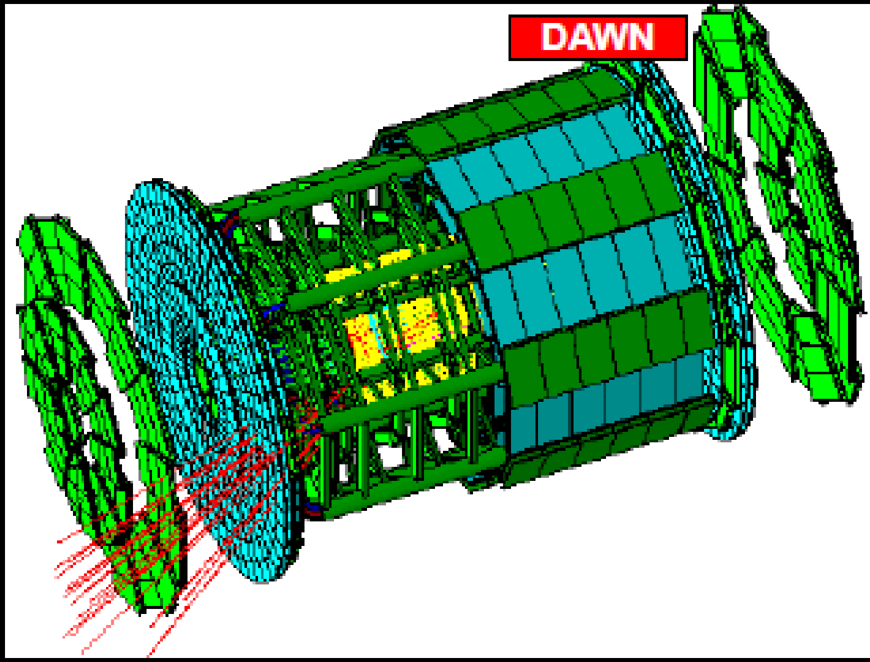
Match between several hadronic models

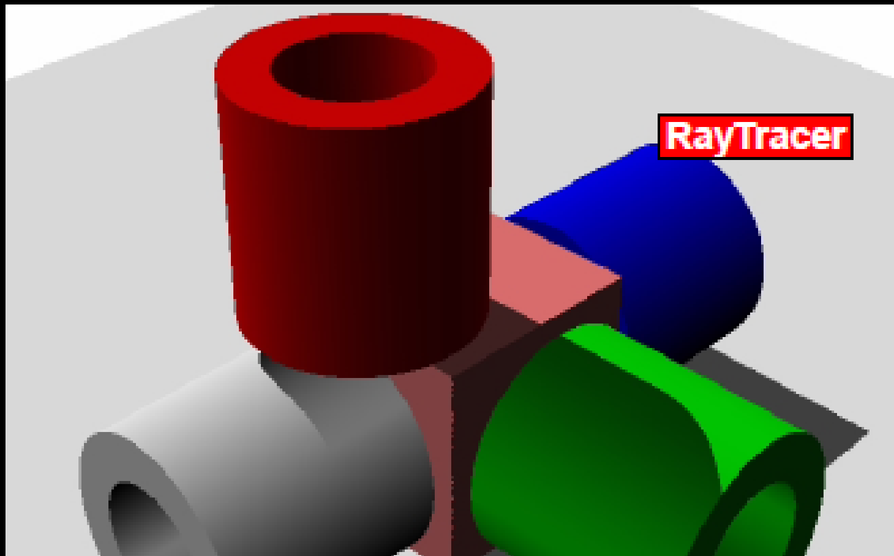


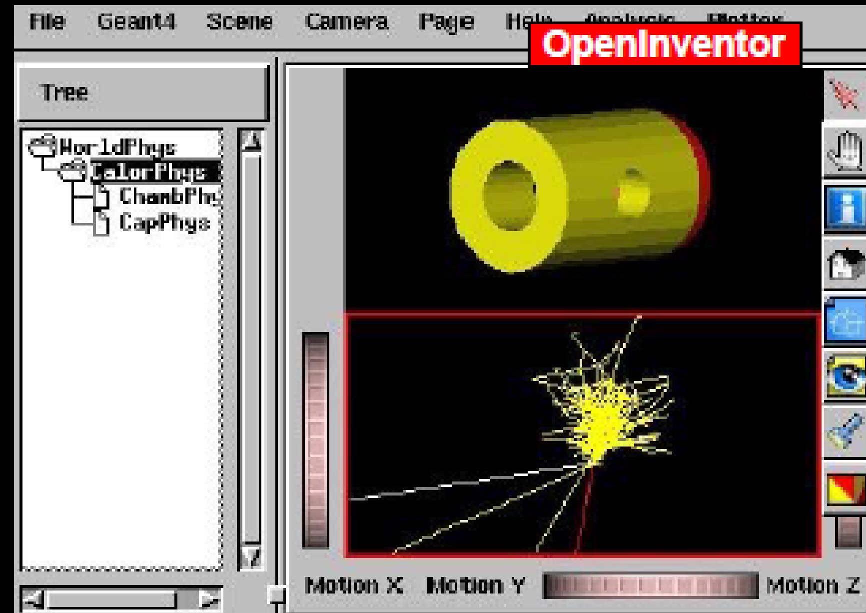
Visualization

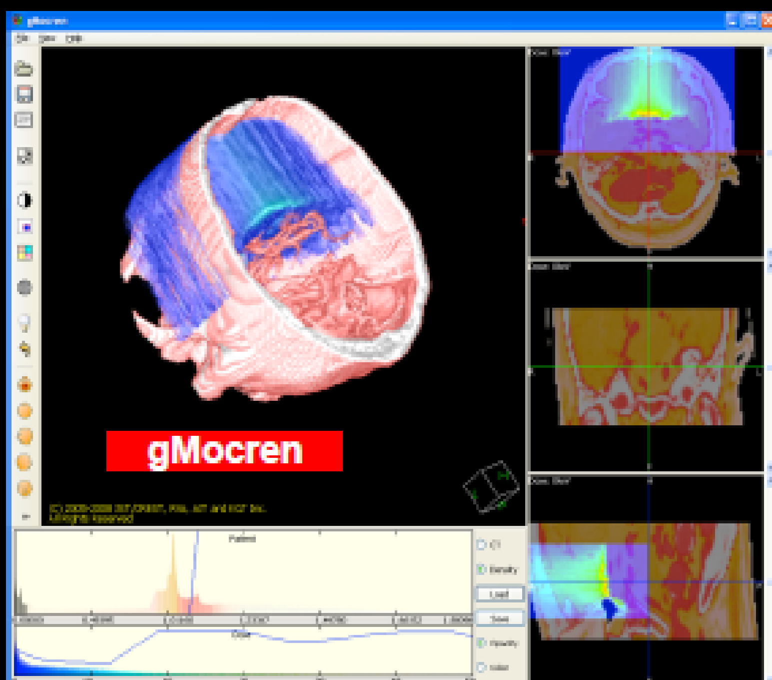

HepRApp Data Browser (version 3.15.0)
File Options Window Help
G4VisTest.heprep.gz
G4VisTest.heprep.gz
HepRep/HepRApp

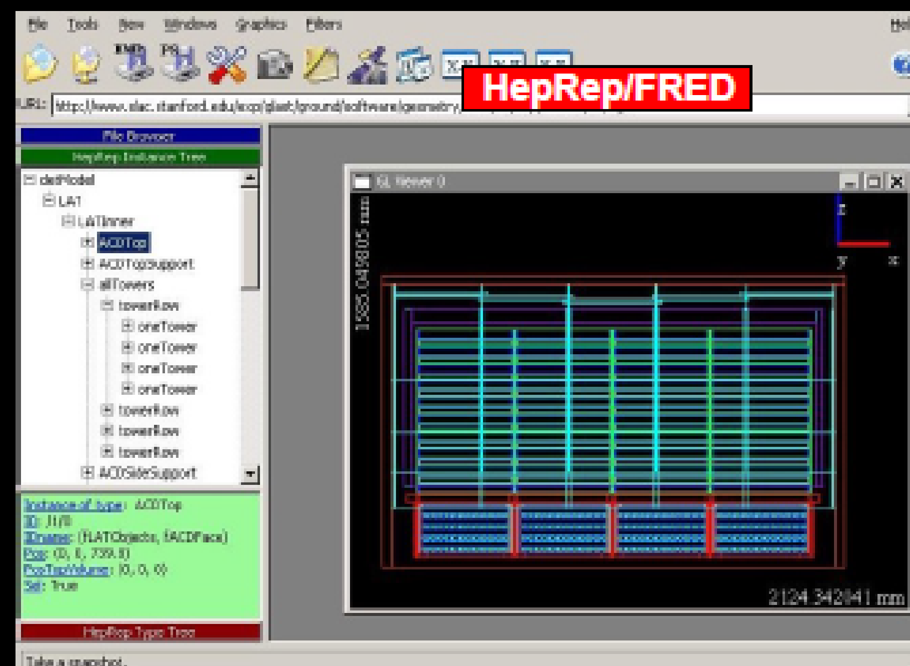

OpenGL

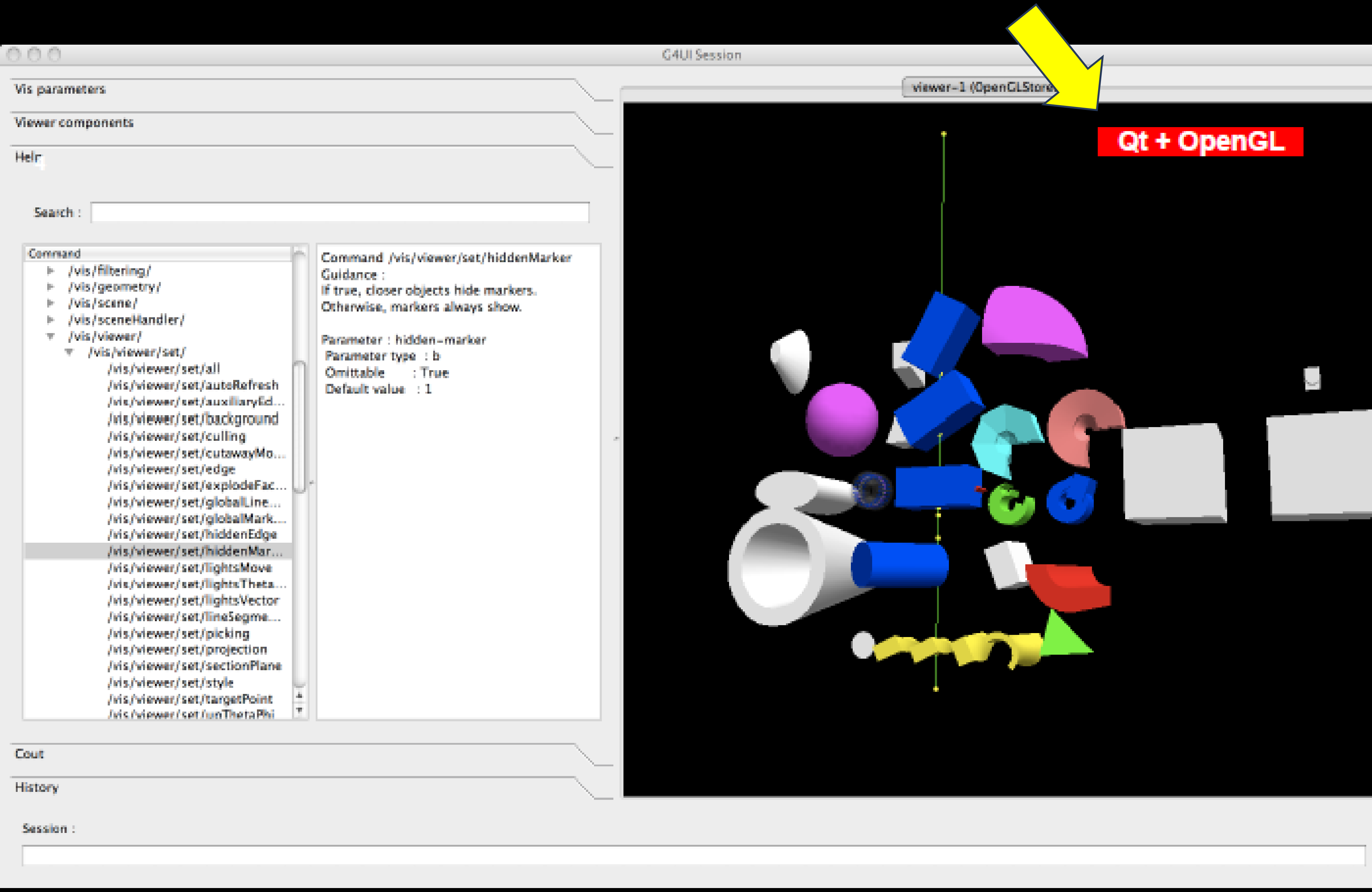

DAWN


RayTracer


File
Geant4
Scene
Camera
Page
OpenInventor
Tree
WorldPhys
ChambPhys
CapPhys
Motion X
Motion Y
Motion Z


gMocren


HepRep/FRED
Take a snapshot.


G4UI Session
Vis parameters
Viewer components
Help
Search :
Command
/vis/filtering/
/vis/geometry/
/vis/scene/
/vis/sceneHandler/
/vis/viewer/
/vis/viewer/set/
/vis/viewer/set/all
/vis/viewer/set/autoRefresh
/vis/viewer/set/auxiliaryEd...
/vis/viewer/set/background
/vis/viewer/set/culling
/vis/viewer/set/cutawayMo...
/vis/viewer/set/edge
/vis/viewer/set/explodeFac...
/vis/viewer/set/globalLine...
/vis/viewer/set/globalMark...
/vis/viewer/set/hiddenEdge
/vis/viewer/set/hiddenMar...
/vis/viewer/set/lightsMove
/vis/viewer/set/lightsTheta...
/vis/viewer/set/lightsVector
/vis/viewer/set/lineSegme...
/vis/viewer/set/picking
/vis/viewer/set/projection
/vis/viewer/set/sectionPlane
/vis/viewer/set/style
/vis/viewer/set/targetPoint
Command /vis/viewer/set/hiddenMarker
Guidance :
If true, closer objects hide markers.
Otherwise, markers always show.
Parameter : hidden-marker
Parameter type : b
Omittable : True
Default value : 1
Cout
History
Session :
Qt + OpenGL

Visualization

- It is possible to request the visualization of different elements
  - Geometry of the detectors/setup
  - Particles trajectories
  - Hits in the detectors
- One can visualize:
  - Lines
  - 3D markers
  - Text
- Several drivers are available
  - OpenGL
  - OpenInventor
  - X11
  - Postscript
  - DAWN
  - HepRec
  - VMRL
  - …
- All drivers are managed through abstract interfaces

Visualization control

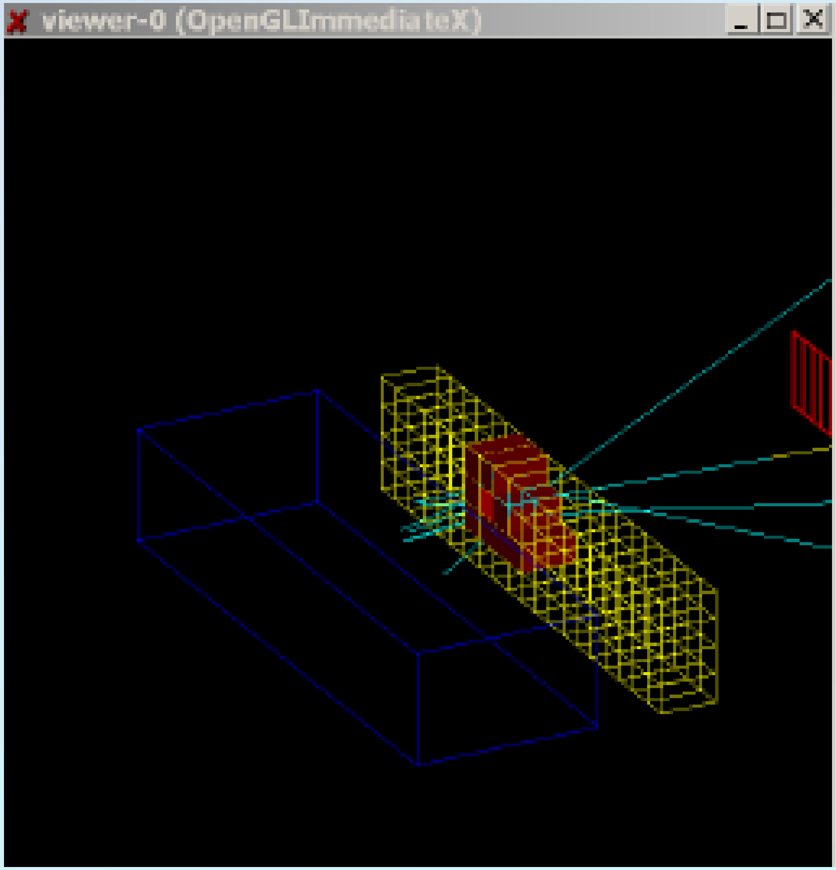
- Geant4 applications do not depend on the visualization driver one wants to use
- The visualization control occurs in the c++ application or through UI commands
- Some drivers work natively inside Geant4
  - OpenGL
  - OpenInventor
  - **RayTracer**
  - **ASCIITree**
- Some other drivers write output files which need to be decoded by another application
  - **HepRepFile**
  - **DAWNFILE**
  - **VRMLFILE**
  - **gMocrenFile**
- It is possible to use more than one diver at a time
- Some drivers are included in Geant4 by default, while others must be included by adding the appropriate flag during the compilation of the application/kernel

Detectors, tracks, hits visualization

- Detectors
  - Every logical volume has a **G4VAttributes** object in association which contains information
    - Mother volume and her daughters' visibility
    - Color, type and line thickness
    - Wire-frame/solid-style mode flag
- Tracks: **G4Trajectory**->Draw() to visualize tracks
  - The track color depends on the user settings, which can override the Geant4 default
- Hits: **G4VHit** has a Draw() method for which some attributes can be set
  - Colored marker
  - marker style (full/empty)
  - Hit volume color

Example: OpenGL

```
/vis/open OGL
/vis/drawVolume
```



- Direct control fromGeant4
- Use GL libraries
- Fast response
- Some elementary operations are possible
  - Zoom
  - Rotations
  - Translations
- graphics can be saved in vector eps format
- more recent: Qt + OpenGL

Tutorials available on the web

- DAWN
  - http://geant4.slac.stanford.edu/Presentations/vis/G4DAWNTutorial/G4DAWNTutorial.html
  - http://geant4.kek.jp/~tanaka/DAWN/About_DAWN.html
  - DAWNCUT
    - http://geant4.kek.jp/~tanaka/DAWN/About_DAWNCUT.html
  - DAVID
    - http://geant4.kek.jp/~tanaka/DAWN/About_DAVID.html
  - And more…
    - http://geant4.kek.jp/~tanaka/
- gMocren
  - http://geant4.kek.jp/gMocren
- HepRApp
  - http://www.slac.stanford.edu/~perl/HepRApp
  - http://geant4.slac.stanford.edu/Presentations/vis/G4HepRAppTutorial/G4HepRAppTutorial.html
- OpenGL
  - http://geant4.slac.stanford.edu/Presentations/vis/G4OpenGLTutorial/G4OpenGLTutorial.html
- OpenScientist
  - http://openscientist.lal.in2p3.fr
- Qt
  - http://geant4.in2p3.fr/spip.php?rubrique25&lang=en