

Stanford  
Linear  
Accelerator  
Center



# Introduction to Geant4

Makoto Asai (SLAC Computing Services)

Geant4 Tutorial Course @ Fermi Lab

October 27<sup>th</sup>, 2003

# Geant4

# Contents

- ▶ General introduction and brief history
- ▶ Highlights of user applications
- ▶ Geant4 kernel
  - ▶ Basic concepts and kernel structure
  - ▶ User classes

The background of the slide is a microscopic image of plant tissue, likely a cross-section of a stem. It shows a complex network of cell walls, with prominent circular and polygonal structures that could be xylem vessels or other specialized cells. The overall appearance is that of a dense, fibrous biological structure.

# General introduction and brief history

# Geant 4

# What is Geant4?

- ▶ Geant4 is the successor of GEANT3, the world-standard toolkit for HEP detector simulation.
- ▶ Geant4 is one of the first successful attempt to re-design a major package of HEP software for the next generation of experiments using an Object-Oriented environment.
- ▶ A variety of requirements also came from heavy ion physics, CP violation physics, cosmic ray physics, astrophysics, space science and medical applications.
- ▶ In order to meet such requirements, a large degree of functionality and flexibility are provided.
- ▶ G4 is not only for HEP but goes well beyond that.

# Flexibility of Geant4

- ▶ In order to meet wide variety of requirements from various application fields, a large degree of functionality and flexibility are provided.
- ▶ Geant4 has many types of geometrical descriptions to describe most complicated and realistic geometries
  - ▶ CSG, BREP, Boolean
  - ▶ STEP compliant
  - ▶ XML interface
- ▶ Everything is open to the user
  - ▶ Choice of physics processes/models
  - ▶ Choice of GUI/Visualization/persistency/histogramming technologies

# Physics in Geant4

- ▶ It is rather unrealistic to develop a uniform physics model to cover wide variety of particles and/or wide energy range.
- ▶ Much wider coverage of physics comes from mixture of theory-driven, parameterized, and empirical formulae. Thanks to polymorphism mechanism, both cross-sections and models (final state generation) can be combined in arbitrary manners into one particular process.
  - ▶ Standard EM processes
  - ▶ Low energy EM processes
  - ▶ Hadronic processes
  - ▶ Photon/lepton-hadron processes
  - ▶ Optical photon processes
  - ▶ Decay processes
  - ▶ Shower parameterization
  - ▶ Event biasing technique

# Physics in Geant4

- ▶ Each cross-section table or physics model (final state generation) has its own applicable energy range. Combining more than one tables / models, one physics process can have enough coverage of energy range for wide variety of simulation applications.
- ▶ Geant4 provides sets of alternative physics models so that the user can freely choose appropriate models according to the type of his/her application.
- ▶ Several individual universities / physicists groups are contributing their physics models to Geant4. Given the modular structure of Geant4, developers of each physics model are well recognized and credited.

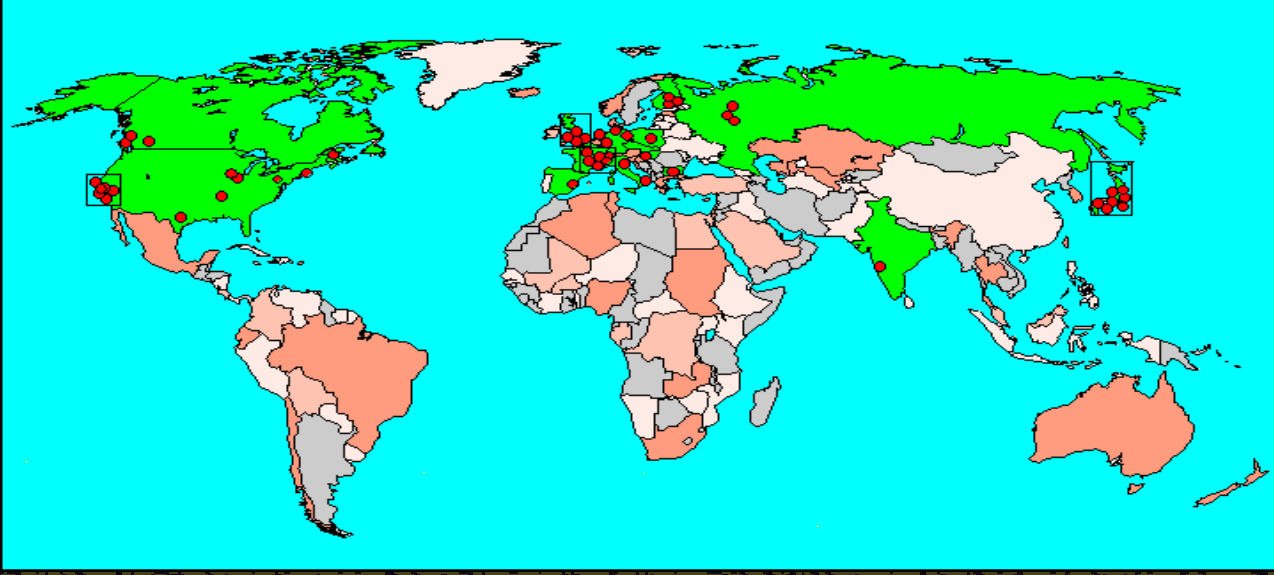
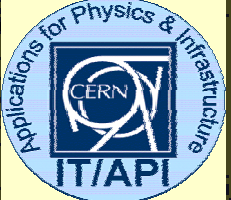
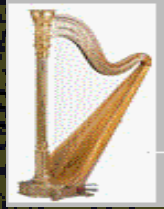
# Geant4 – Its history and future

- ▶ Dec '94 - Project start
- ▶ Apr '97 - First alpha release
- ▶ Jul '98 - First beta release
- ▶ Dec '98 - Geant4 0.0 release
- ▶ Jul '99 - Geant4 0.1 release
- ▶ ...
- ▶ Jun '03 - Geant4 5.2 release
- ▶ Dec '03 - Geant4 6.0 release (planned)
- ▶ We currently provide two to three public releases and monthly beta releases in between public releases every year.



# Geant4 Collaboration

HARP

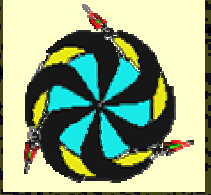


Lebedev



Univ. Barcelona

PPARC



Collaborators also from non-member institutions, including  
Budker Inst. of Physics  
IHEP Protvino  
MEPHI Moscow  
Pittsburg University



Helsinki Inst. Ph.

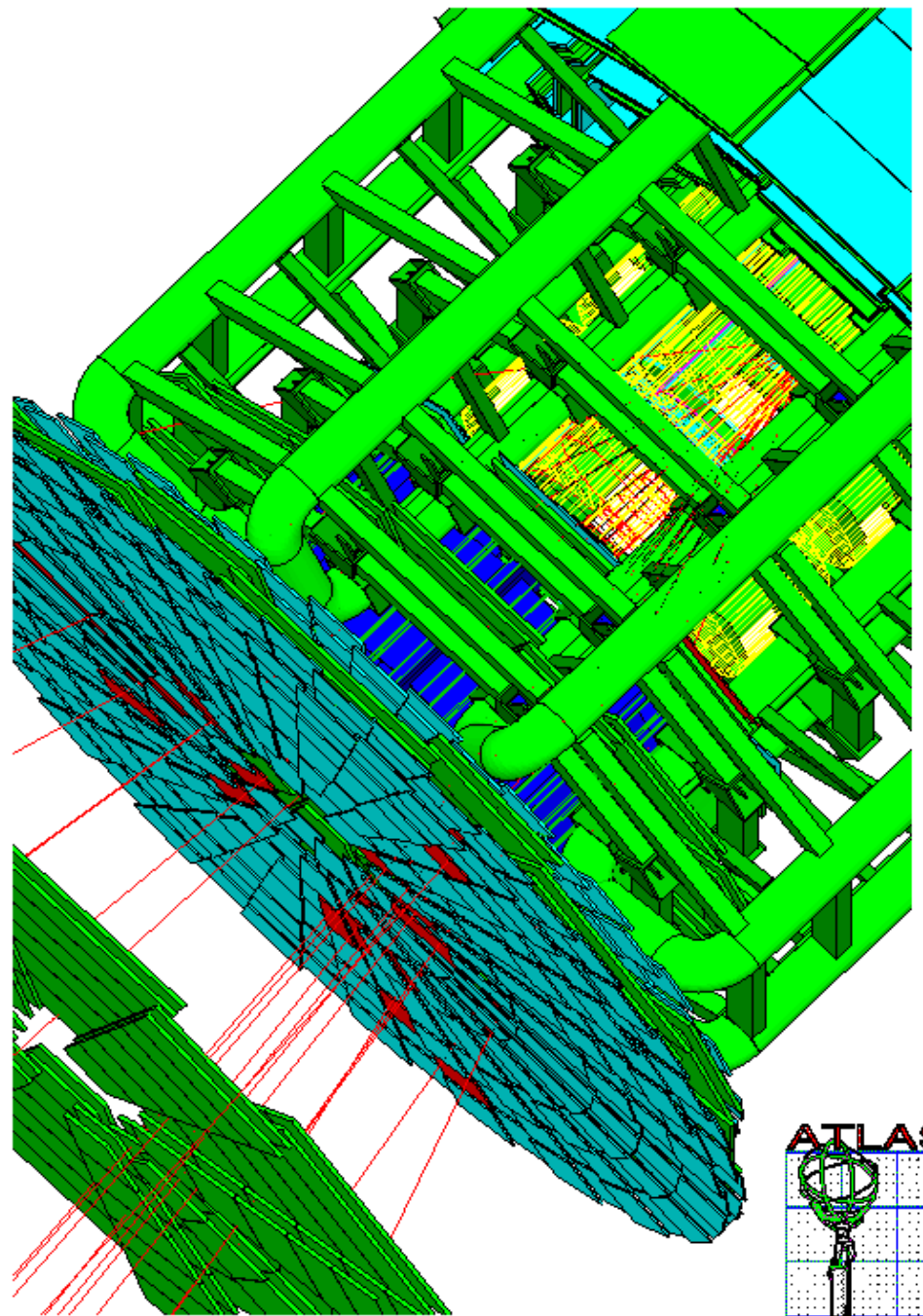
A microscopic image of plant cells, showing a complex network of cell walls and large, circular vacuoles. A white grid is overlaid on the image. The text is centered in the upper half of the image.

# Highlights of Users Applications

**Geant 4**

# Geant4 in HEP

- ▶ ATLAS (CERN-LHC)
- ▶ 22 x 22 x 44 m<sup>3</sup>
- ▶ 15,000 ton
- ▶ 4 million channels
- ▶ 40 MHz readout

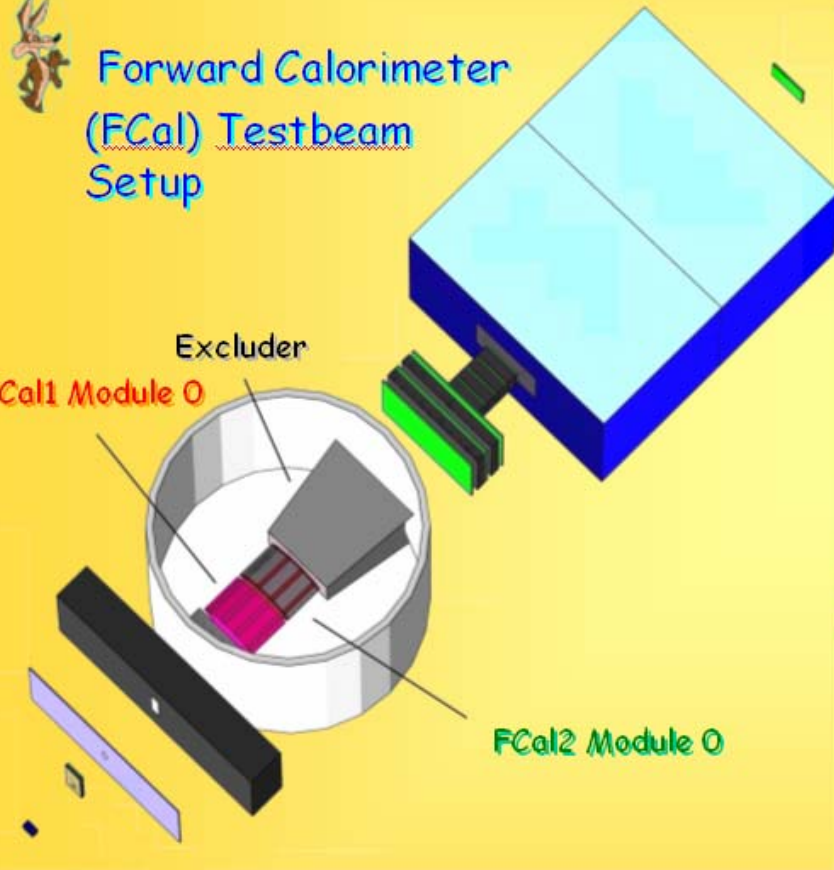




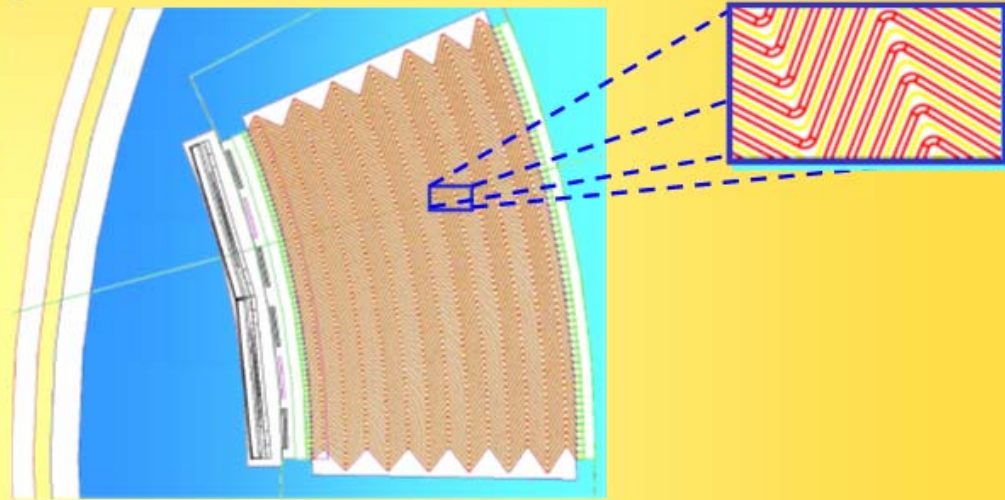
## Geant4 Setups (2)



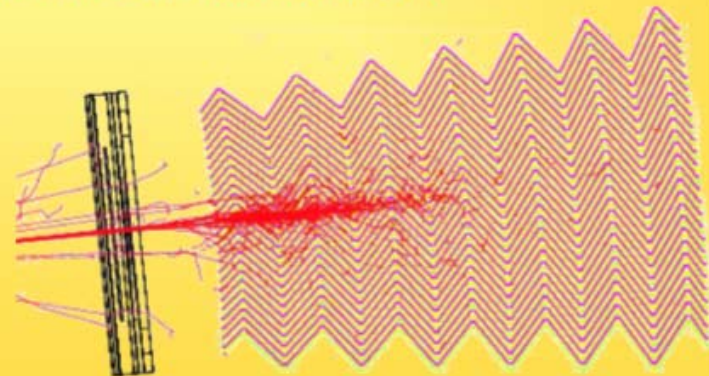
Forward Calorimeter  
(FCal) Testbeam  
Setup



Electromagnetic Barrel Accordion Calorimeter



10 GeV Electron Shower

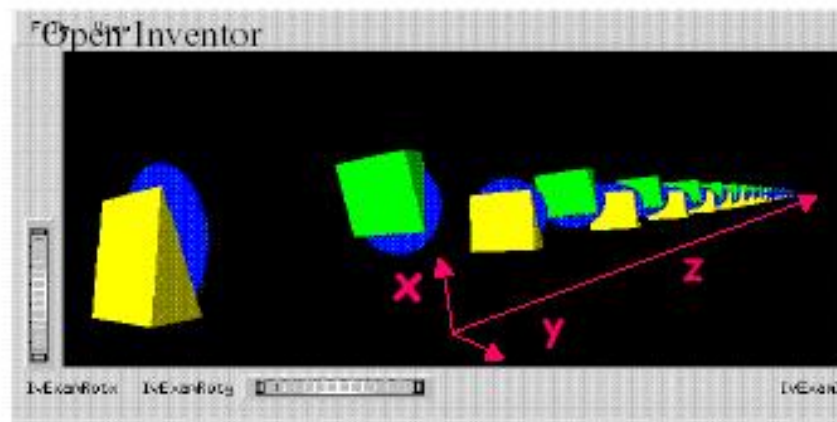


# Geant4 for beam transportation

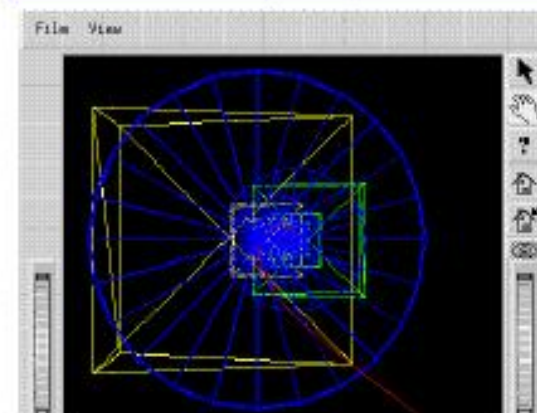
## Example: Helical Channel

Published in proc. of PAC 2001  
(Fermilab-Conf-01-182-T)

72 m long solenoidal + dipole field with wedge absorbers and thin cavities



$$B_{xy} = B_T \cos, \sin \left( \frac{2p}{L} z \right) \quad B_z = B_0$$

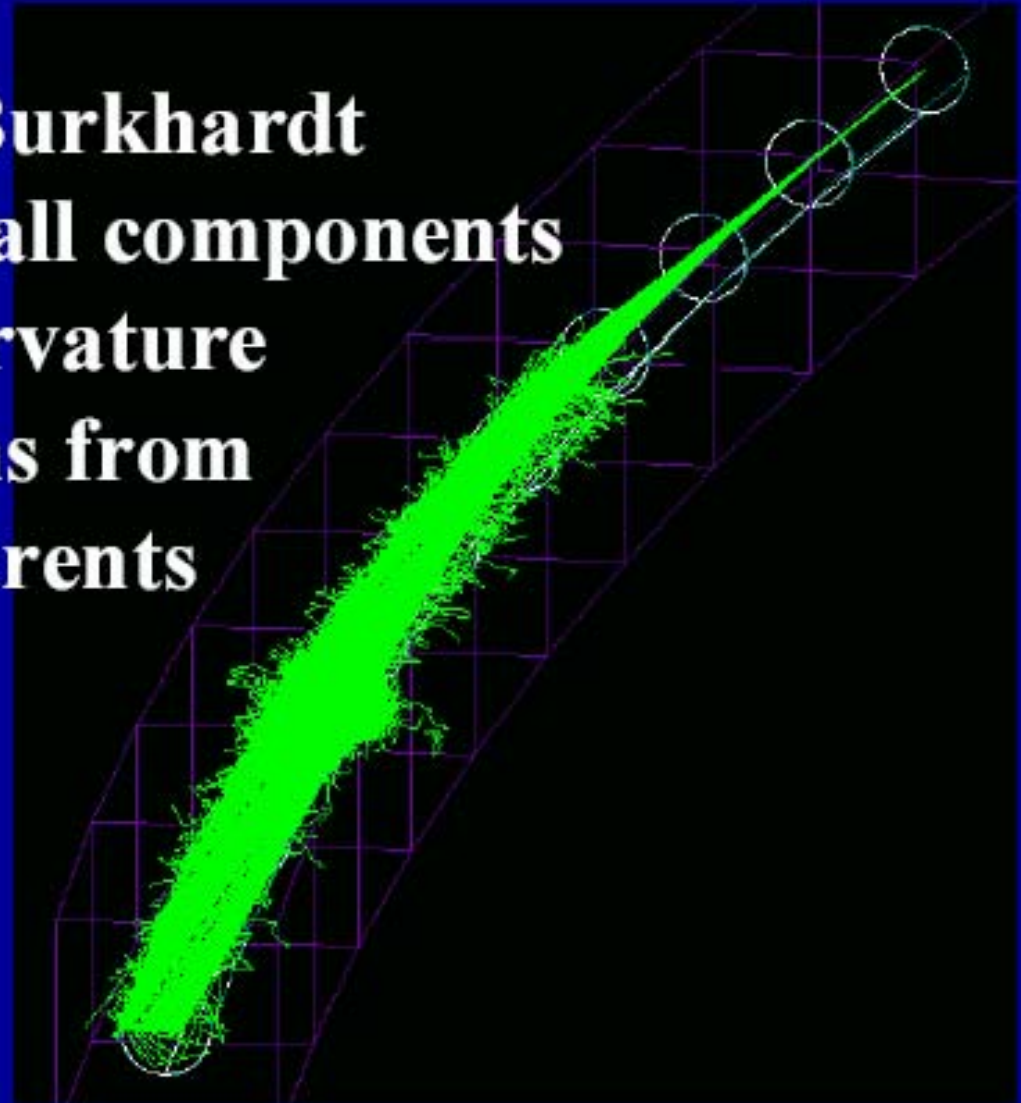


### Other simulations:

- Alternate Solenoid Channel (sFoFo), published in proceedings of PAC2001 and Feasibility Study II for a Neutrino Factory at BNL (2001)
- Bent Solenoid Channel, presented at Emittance Exchange Workshop, BNL 2000
- Low Frequency r.f. Cooling Channel, presented at International Cooling Experiment Workshop, CERN 2001
- Cooling Experiment (MICE) Simulation (in progress)

# Synchrotron Radiation

**Generator of H. Burkhardt**  
**Implemented for all components**  
**Based on local curvature**  
**Individual photons from**  
**individual parents**



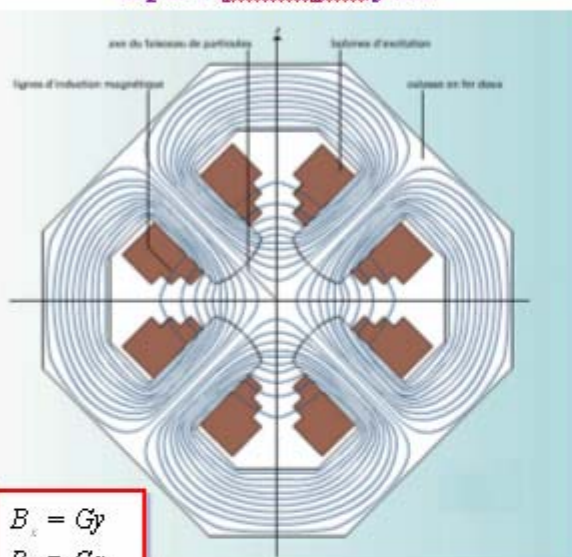
2/22/2002

# Ray tracing in perfect quadrupoles

In our microbeam line, four quadrupoles to focus the beam

- Focus<sub>1</sub> Defocus<sub>2</sub> F<sub>2</sub> D<sub>1</sub> "Russian" configuration
- Quad length = 15 cm, gap radius = 1 cm, distance between quads = 4 cm
- $G_1 = -G_4 = 5.8928$  T/m and  $G_2 = -G_3 = -14.6466$  T/m
- proton or alpha beam
- gaussian T = 8 MeV or 2 MeV (standard deviation is 4 keV)
- angular divergence : 0.5 mrad
- gaussian position distribution of 10  $\mu\text{m}$  FWHM

## A pure quadrupole field

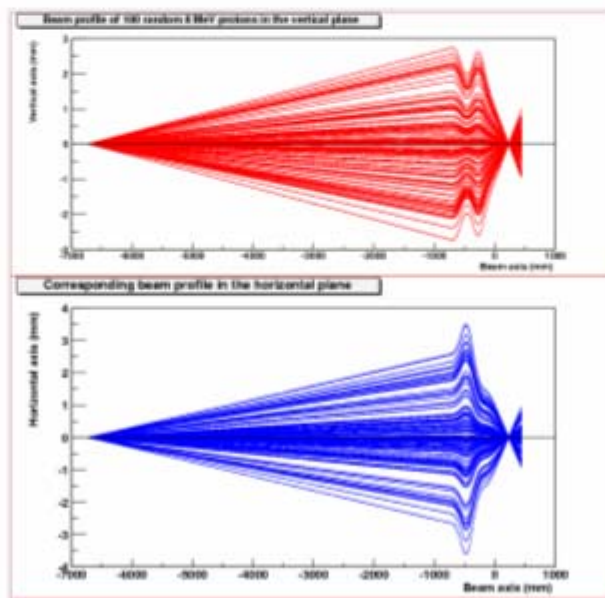
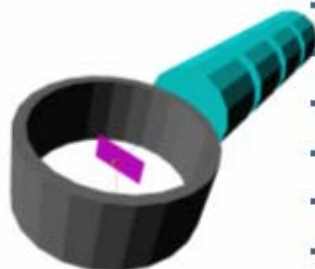


$$\begin{aligned} B_x &= Gy \\ B_y &= Gx \\ B_z &= 0 \end{aligned}$$

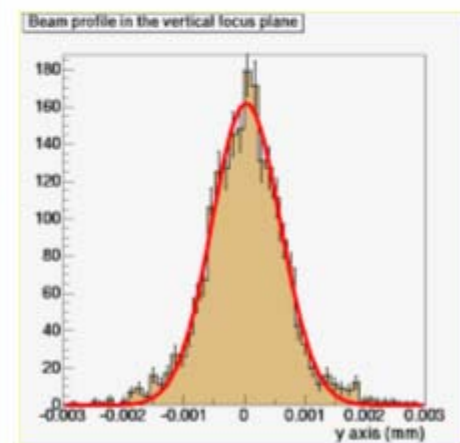
$G$  is the field gradient

GEANT4.1+P01 predicts :

- Focus plane position :  $230.15 \pm 0.05$  mm
- FWHM of beam in image plane :  $1.3$   $\mu\text{m}$
- same prediction as the OXRAY code :  
focus plane position :  $230.1 \pm 0.1$  mm  
FWHM =  $1.1$   $\mu\text{m}$



GEANT4 4.1+P01



GEANT4 4.1+P01

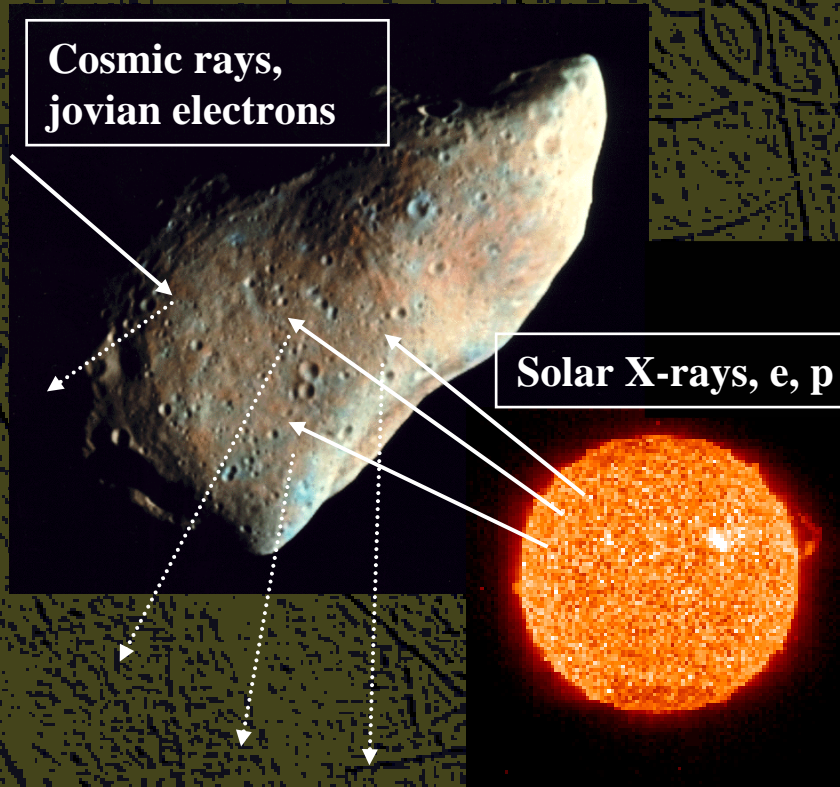
# Geant4 in space science



ESA Space Environment & Effects Analysis Section

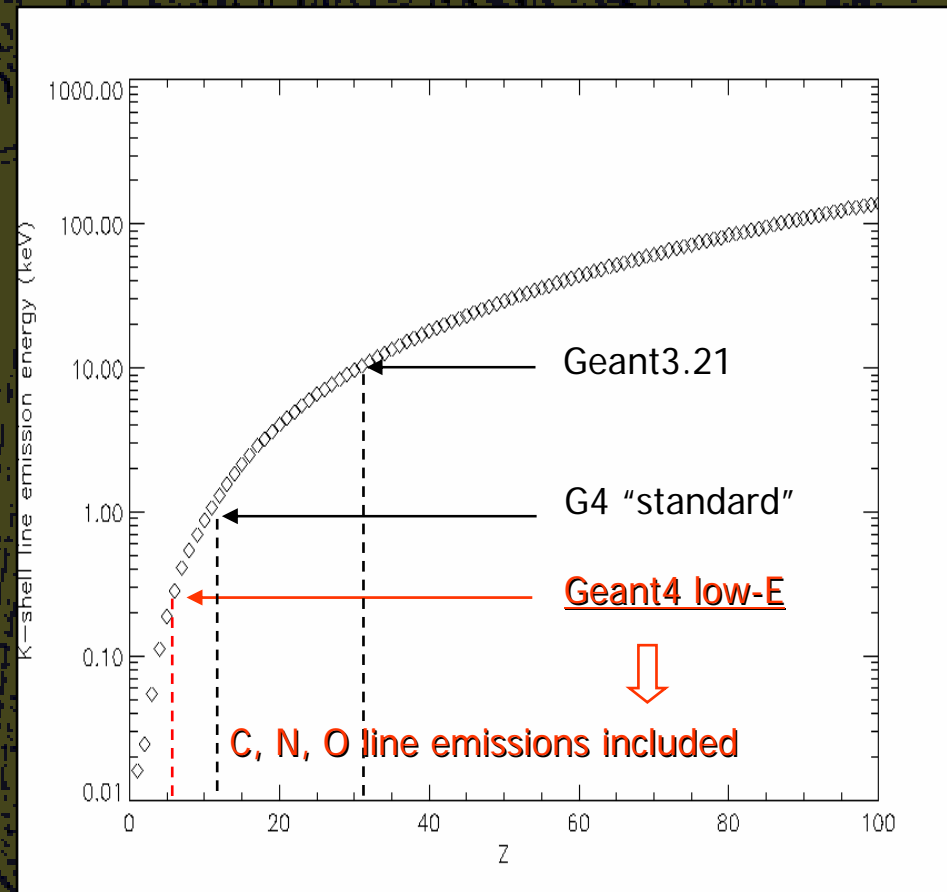


## X-Ray Surveys of Asteroids and Moons



Courtesy SOHO EIT

Induced X-ray line emission:  
indicator of target  
composition  
(~100  $\mu\text{m}$  surface layer)



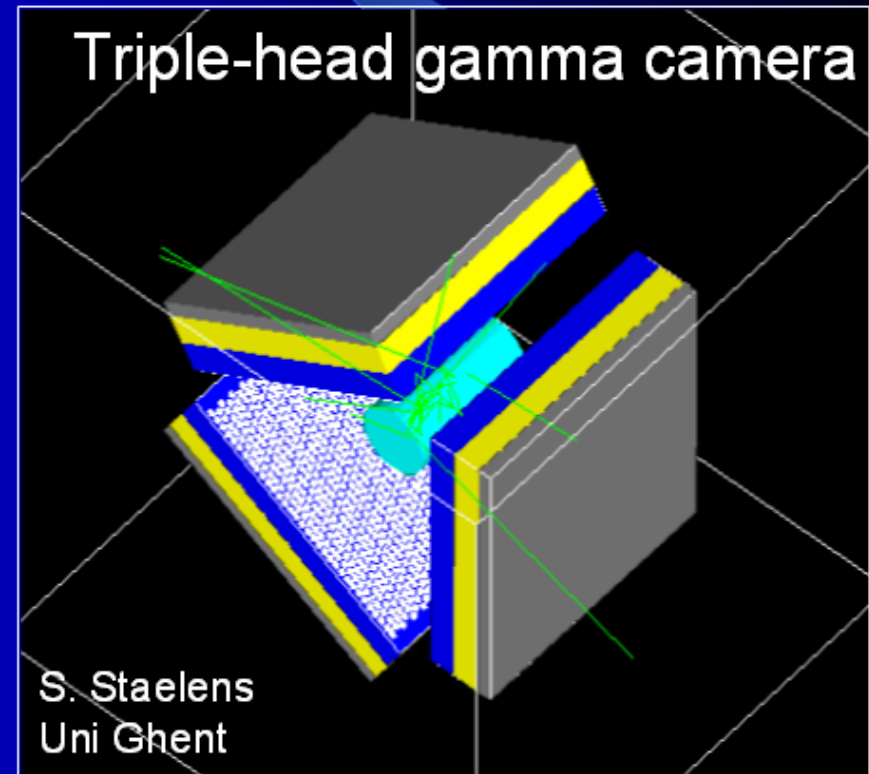
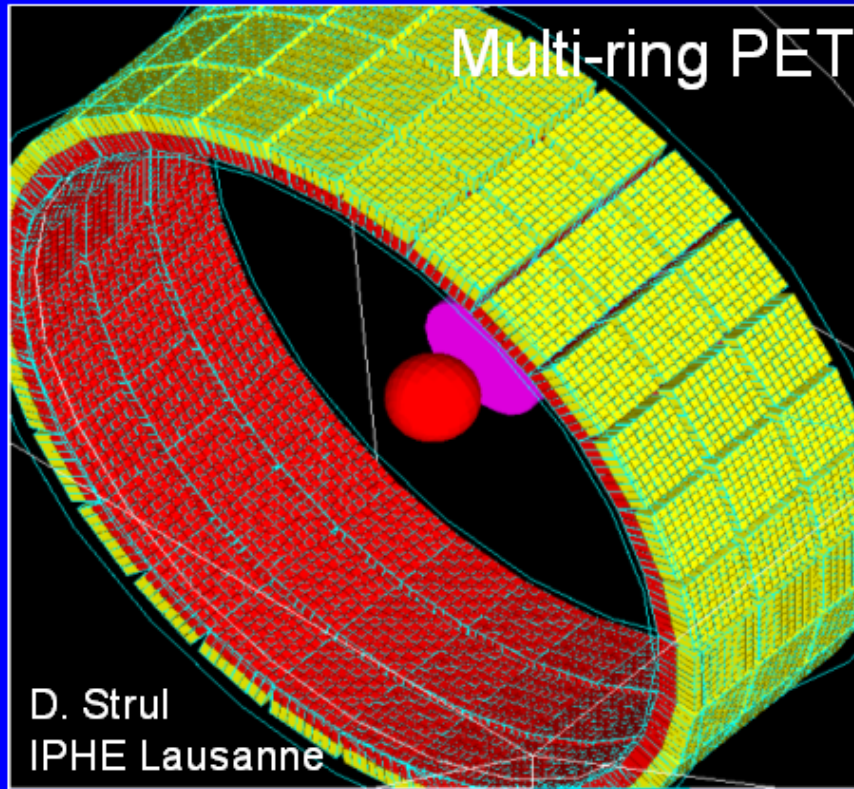




***DESIRE*** (Dose Estimation by Simulation of the ISS  
Radiation Environment)

- KTH Stockholm, ESTEC, EAC, NASA Johnson
- Prediction of the ambient energetic particle environment (**SPENVIS** & additional models)
- Construction of COLUMBUS geometry in **Geant4**
- Radiation transport, including secondary particle production, through the geometry
- Calculation of astronaut radiation doses

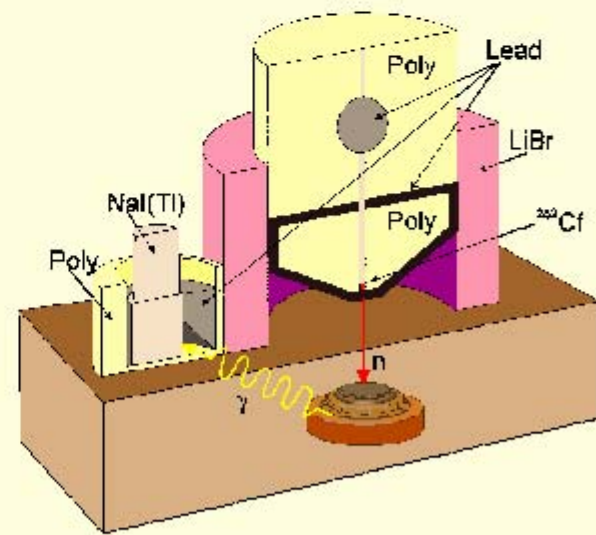
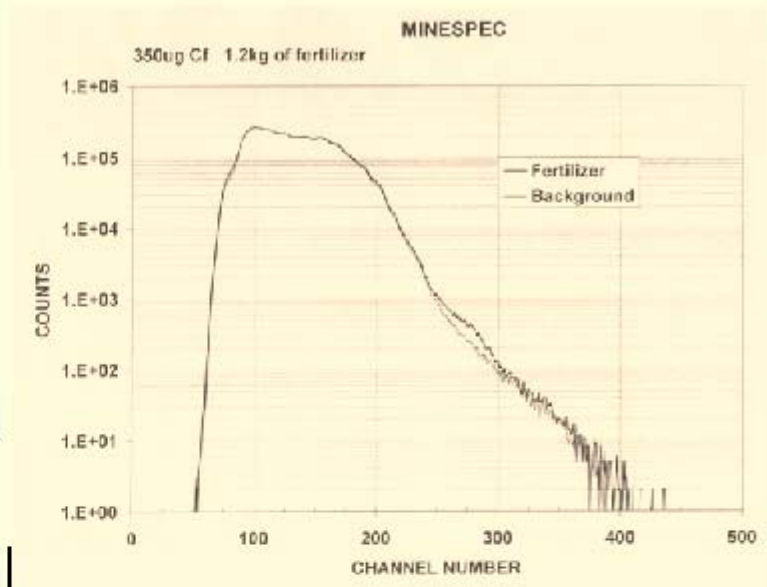
# Geometry examples of GATE applications





# Thermal Neutron Activation

- TNA detects explosive by properties of constituents
  - High concentration of N
  - Does not ID explosive
- Can confirm presence of all surface laid or shallow AT mines in few seconds to 1 minute
- AT up to 20 cm deep and large AP mines in < 5 minutes



# Full Simulations

For next generation linear  
collider experiment

LCD Full Sim

GISMO

C++

BRAHMS

GEANT3

FORTRAN

JIM

GEANT3

FORTRAN

LCDROOT/LCDG4

MOKKA

JUPITER

Common GEANT4  
executable

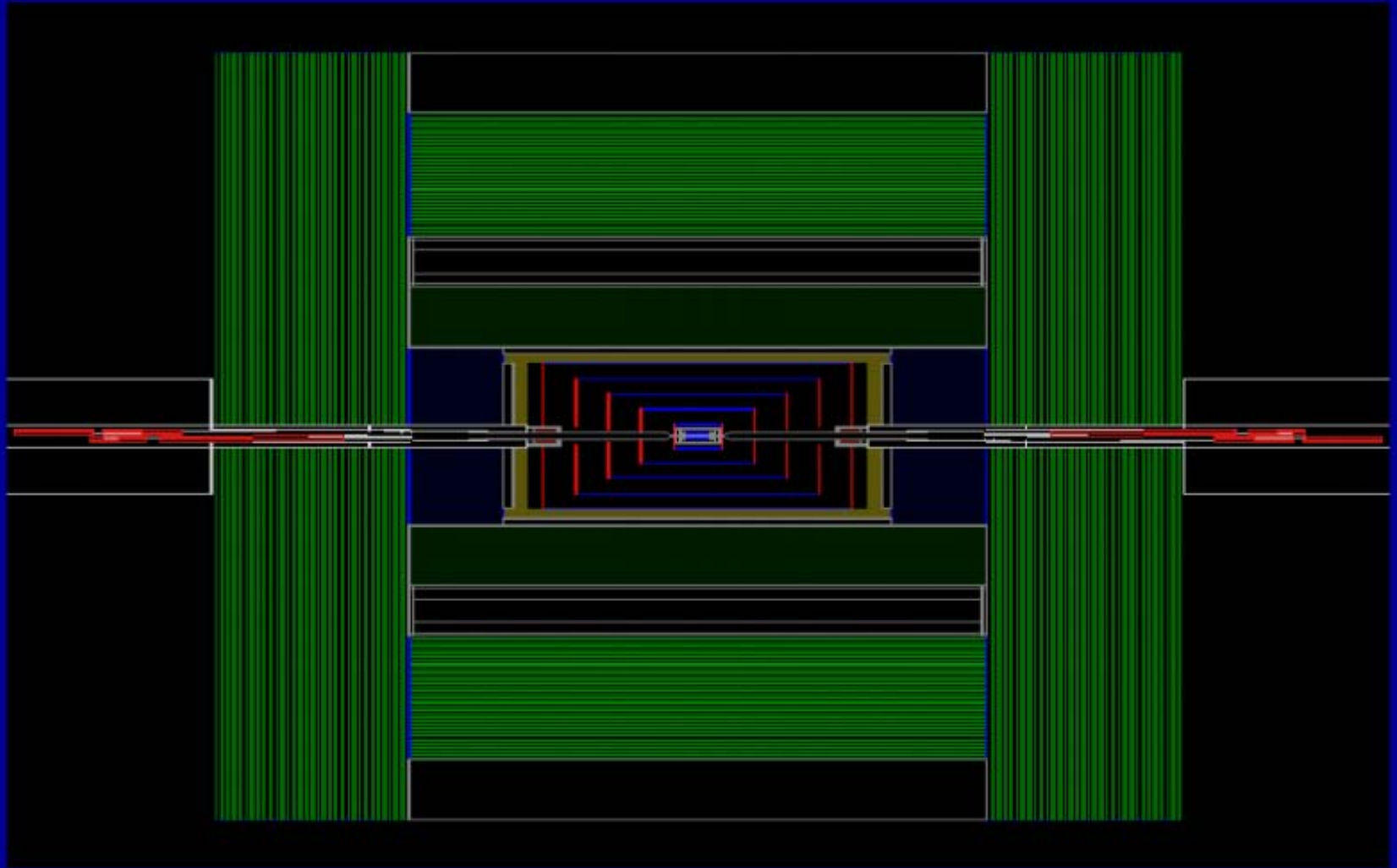
Runtime geometry

Generic Hit output

# LCD/Mokka

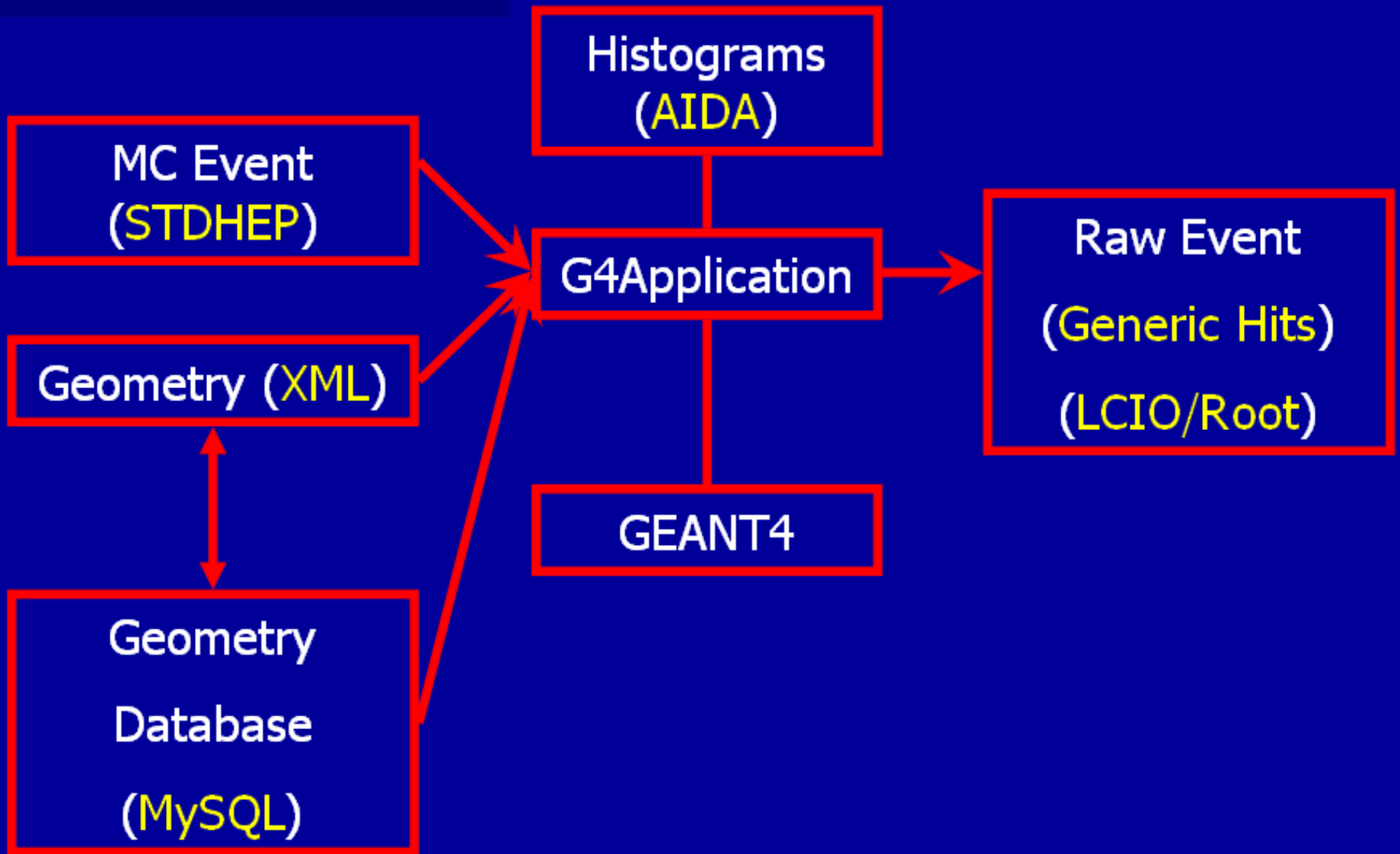
- First version of mysql / xml interface exists
- SD detector fully modeled including beamline elements.
- Several TESLA detector versions modeled.
- LCIO output implemented in beta version.
- Interfaces to HEPEVT and STDHEP and background files implemented.
- Interface to AIDA integrated.

# SD in Mokka




Courtesy of N.Graf (SLAC)

# LC Detector Full Simulation



Courtesy of N.Graf (SLAC)

A microscopic image of plant tissue, likely a leaf cross-section, showing various cellular structures. A white grid is overlaid on the image. The text "Basic concepts and kernel structure" is centered in the upper half of the image.

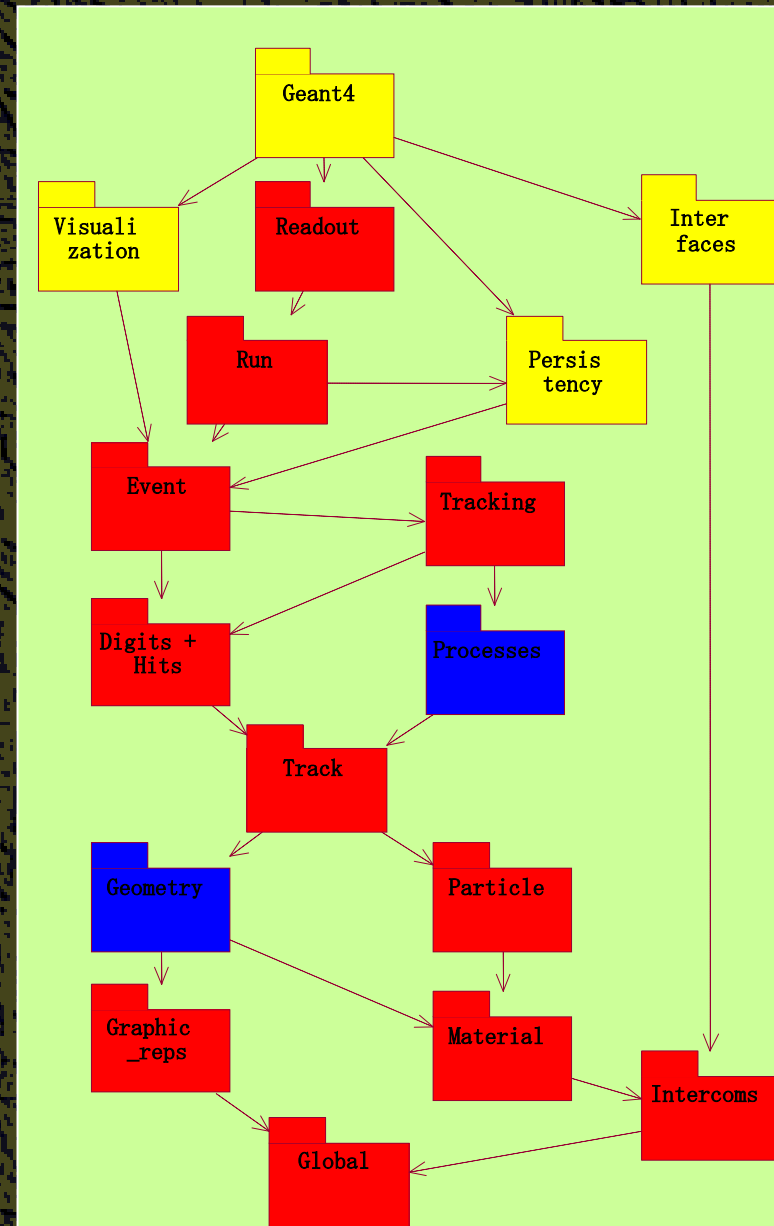
Basic concepts  
and kernel structure

Geant 4



# Geant4 kernel

- ▶ Geant4 consists of 17 categories.
  - ▶ Independently developed and maintained by WG(s) responsible to each category.
  - ▶ Interfaces between categories (e.g. top level design) are maintained by the global architecture WG.
- ▶ Geant4 Kernel
  - ▶ Handles run, event, track, step, hit, trajectory.
  - ▶ Provides frameworks of geometrical representation and physics processes.



# Run in Geant4

- ▶ As an analogy of the real experiment, a run of Geant4 starts with "Beam On".
- ▶ Within a run, the user cannot change
  - ▶ detector geometry
  - ▶ settings of physics processes
    - > detector is inaccessible during a run
- ▶ Conceptually, a run is a collection of events which share the same detector conditions.
- ▶ At the beginning of a run, geometry is optimized for navigation and cross-section tables are calculated according to materials appear in the geometry and the cut-off values defined.
- ▶ **G4RunManager** class manages processing a run, a run is represented by **G4Run** class or a user-defined class derived from G4Run.

# Event in Geant4

- ▶ At beginning of processing, an event contains primary particles. These primaries are pushed into a stack.
- ▶ When the stack becomes empty, processing of an event is over.
- ▶ **G4EventManager** class manages processing an event.
- ▶ **G4Event** class represents an event. It has following objects at the end of its processing.
  - ▶ List of primary vertexes and particles (as input)
  - ▶ Hits collections
  - ▶ Trajectory collection (optional)
  - ▶ Digits collections (optional)

# Track in Geant4

- ▶ Track is a snapshot of a particle.
  - ▶ It has only position and physical quantities of current instance.
- ▶ Step is a “delta” information to a track.
  - ▶ Track is not a collection of steps.
- ▶ Track is deleted when
  - ▶ it goes out of the world volume
  - ▶ it disappears (e.g. decay)
  - ▶ it goes down to zero kinetic energy and no “AtRest” additional process is required
  - ▶ the user decides to kill it
- ▶ No track object persists at the end of event.
  - ▶ For the record of track, use trajectory class objects.
- ▶ **G4TrackingManager** manages processing a track, a track is represented by **G4Track** class.

# Step in Geant4

- ▶ Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).
- ▶ Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume.
  - ▶ Because one step knows materials of two volumes, boundary processes such as transition radiation or refraction could be simulated.
- ▶ **G4SteppingManager** class manages processing a step, a step is represented by **G4Step** class.



# Particle in Geant4

- ▶ A particle in Geant4 is represented in three layers of classes.
- ▶ **G4Track**
  - ▶ Position, geometrical information, etc.
  - ▶ This is a class representing a particle to be tracked.
- ▶ **G4DynamicParticle**
  - ▶ "Dynamic" physical properties of a particle, such as momentum, energy, spin, etc.
  - ▶ Each G4Track object has its own and unique G4DynamicParticle object.
  - ▶ This is a class representing an individual particle (which is not necessarily to be tracked).
- ▶ **G4ParticleDefinition**
  - ▶ "Static" properties of a particle, such as charge, mass, life time, decay channels, etc.
  - ▶ G4ProcessManager which describes processes involving to the particle
  - ▶ All G4DynamicParticle objects of same kind of particle share the same G4ParticleDefinition.

# Tracking and processes

- ▶ Geant4 tracking is general.
  - ▶ It is independent to
    - ▶ the particle type
    - ▶ the physics processes involving to a particle
  - ▶ It gives the chance to all processes
    - ▶ To contribute to determining the step length
    - ▶ To contribute any possible changes in physical quantities of the track
    - ▶ To generate secondary particles
    - ▶ To suggest changes in the state of the track
      - ▶ e.g. to suspend, postpone or kill it.

# Processes in Geant4

- ▶ In Geant4, particle transportation is a process as well, by which a particle interacts with geometrical volume boundaries and field of any kind.
  - ▶ Because of this, shower parameterization process can take over from the ordinary transportation without modifying the transportation process.
- ▶ Each particle has its own list of applicable processes. At each step, all processes listed are invoked to get proposed physical interaction lengths.
- ▶ The process which requires the shortest interaction length (in space-time) limits the step.
- ▶ All processes are derived from **G4VProcess** abstract base class. Each particle has its individual **G4ProcessManager** class object which holds a vector of assigned processes.

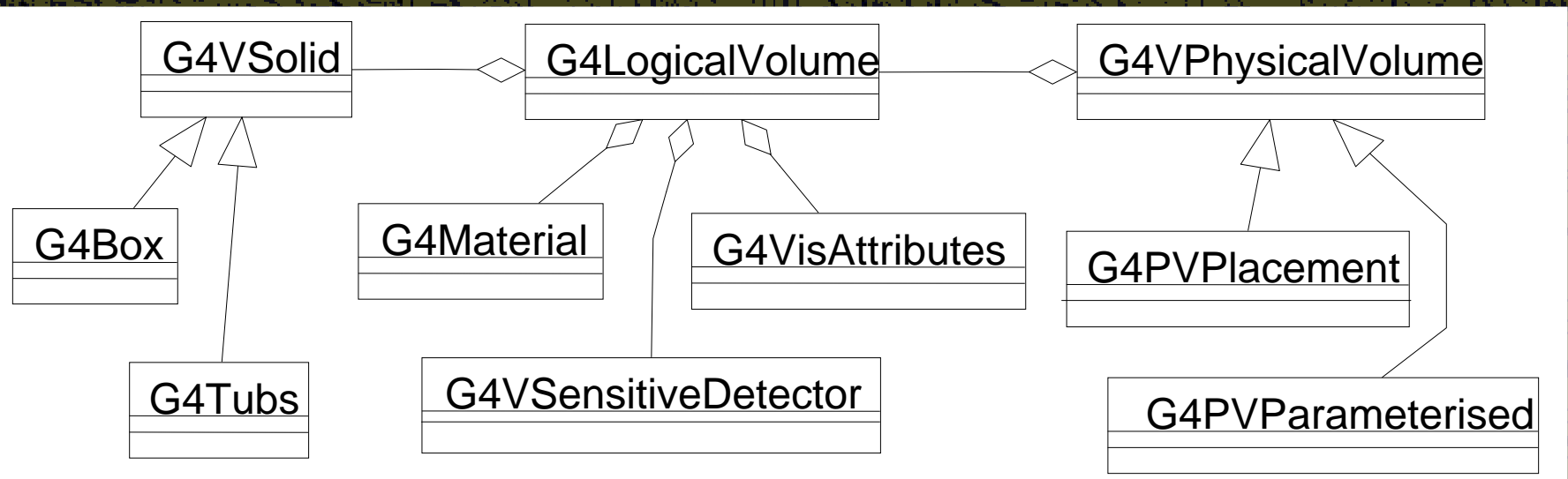


# Process and step

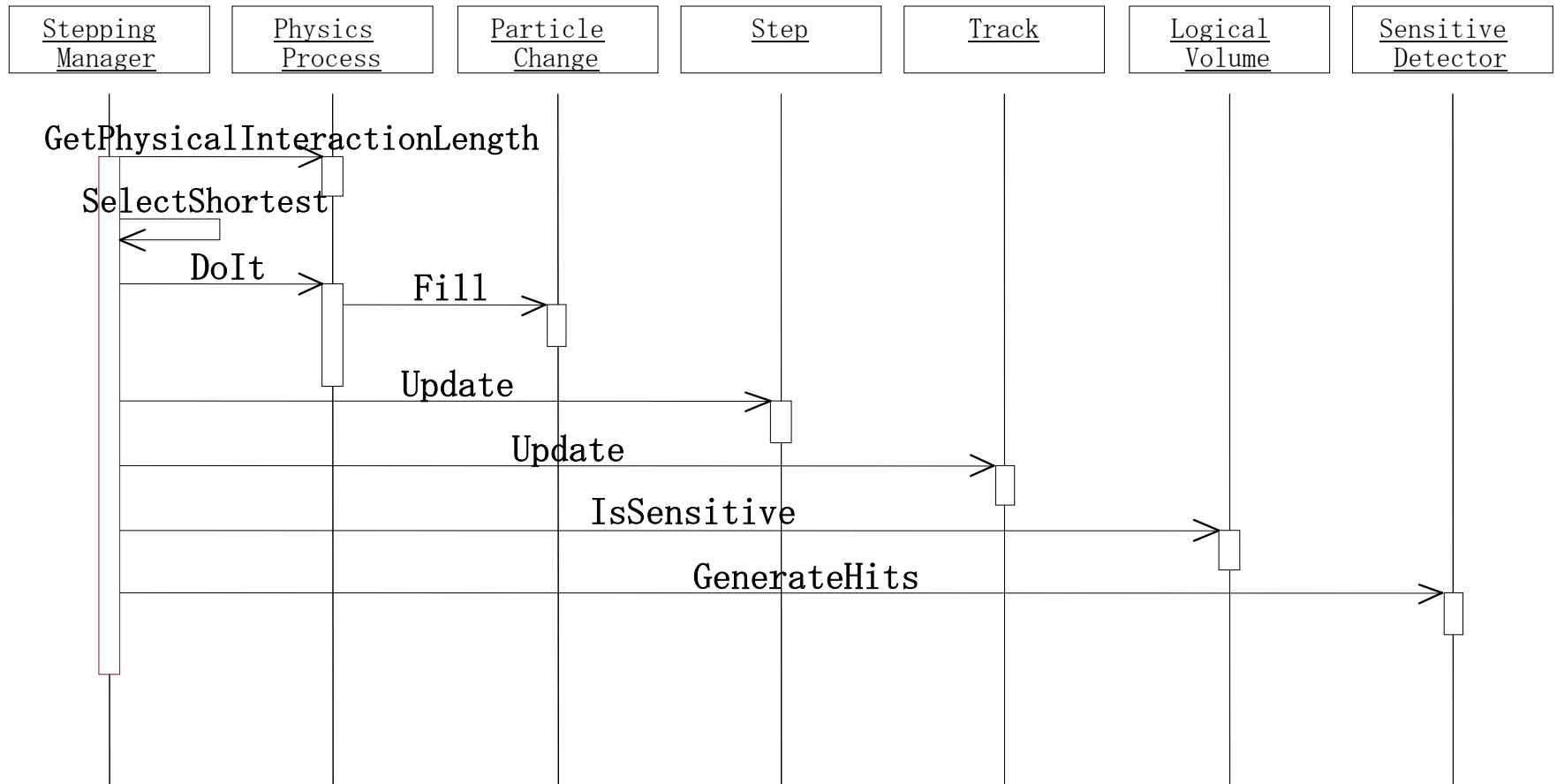
- ▶ Each process has one or combination of the following natures.
  - ▶ AtRest
    - ▶ e.g. muon decay at rest
  - ▶ AlongStep
    - ▶ e.g. Celenkov process
  - ▶ PostStep
    - ▶ e.g. decay on the fly
- ▶ Each process involving to a step replies a concrete object of **G4ParticleChange** which affects on a step/track.

# Volume

- ▶ Three conceptual layers
  - ▶ **G4VSolid** -- *shape, size*
  - ▶ **G4LogicalVolume** -- *daughter physical volumes, material, sensitivity, user limits, etc.*
  - ▶ **G4VPhysicalVolume** -- *position, rotation*
- ▶ Hierarchical three layers of geometry description allows maximum reuse of information to minimize the use of memory space.
- ▶ Detector sensitivity should be described by the user in his/her concrete implementation of **G4VSensitiveDetector** and set to G4LogicalVolume.



# How Geant4 runs (one step)

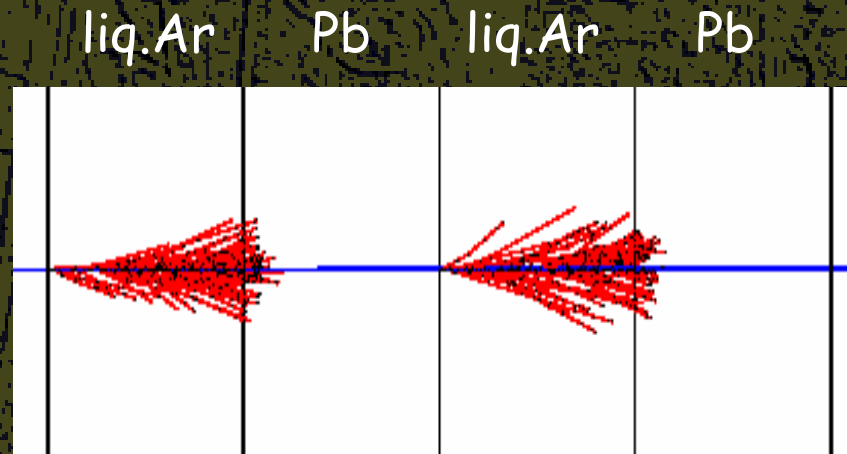
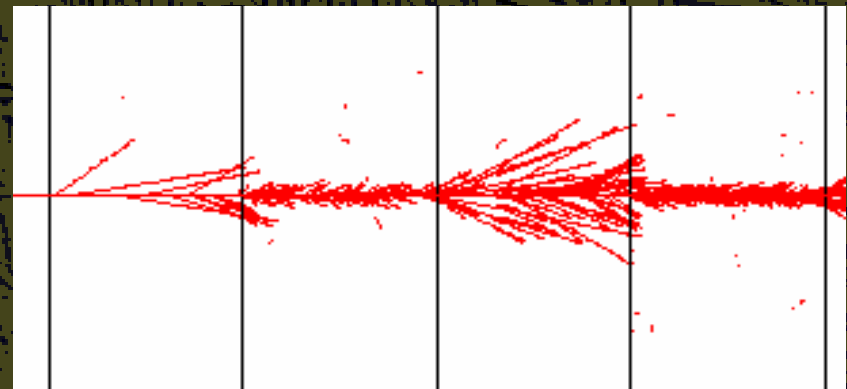


# Cuts in Geant4

- ▶ A Cut in Geant4 is a **production threshold**.
  - ▶ Only for physics processes that have infrared divergence
  - ▶ Not tracking cut, which does not exist in Geant4
- ▶ Energy threshold must be determined at which discrete energy loss is replaced by continuous loss
  - ▶ Old way:
    - ▶ Track primary particle until cut-off energy is reached, calculate continuous loss and dump it at that point, stop tracking primary
    - ▶ Create secondaries only above cut-off energy, or add to continuous loss of primary for less energetic secondaries
  - ▶ Geant4 way:
    - ▶ Specify range (which is converted to energy for each material) at which continuous loss begins, track primary down to zero range
    - ▶ Create secondaries only above specified range, or add to continuous loss of primary for secondaries of less energetic **and** not reaching to the volume boundary

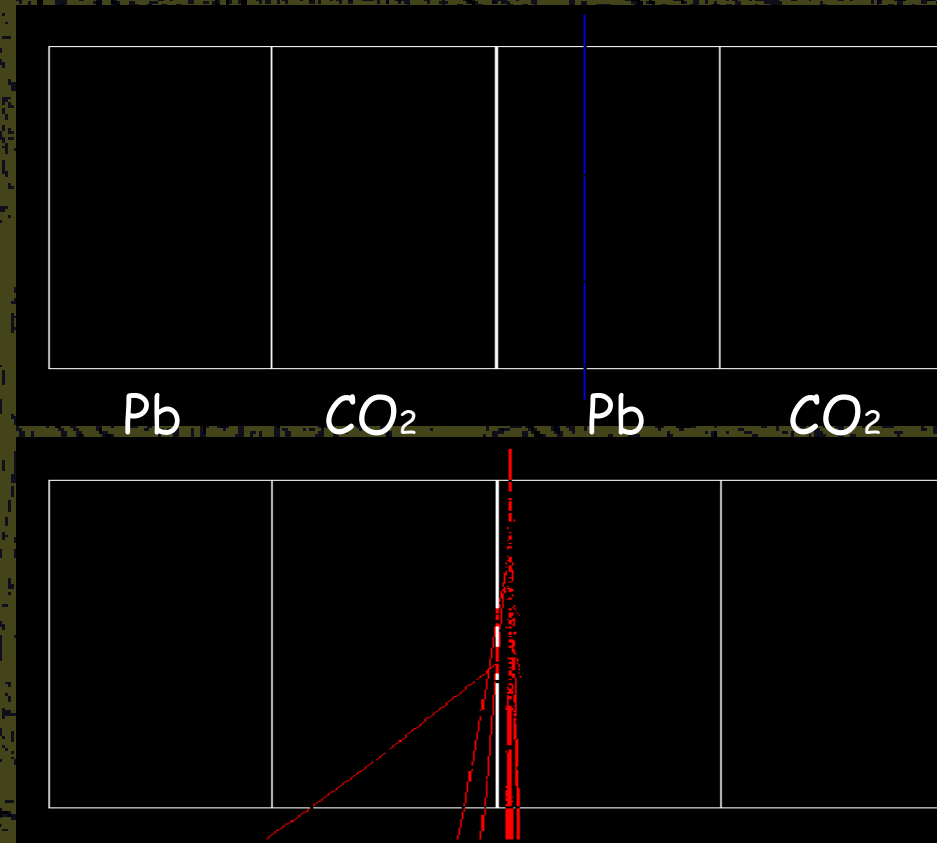
# Energy cut vs. range cut

- ▶ 500 MeV/c proton in liq.Ar (4mm) / Pb (4mm) sampling calorimeter
- ▶ Geant3 (energy cut)
  - ▶ Ecut = 450 keV
- ▶ Geant4 (range cut)
  - ▶ Rcut = 1.5 mm
  - ▶ Corresponds to  
Ecut in liq.Ar = 450 keV,  
Ecut in Pb = 2 MeV



# Range cut vs. geometrical safety

- ▶ Even though a secondary is less energetic than the defined range cut, it can penetrate to the next volume (and actual range can be longer than the range cut) if it is born close to the geometrical boundary.
- ▶ Range cut is applied **only if** the range of the particle is shorter than the geometrical safety.
  - ▶ Such particle cannot penetrate.
  - ▶ Geometrical safety is the isotropic shortest distance to the geometrical boundary.



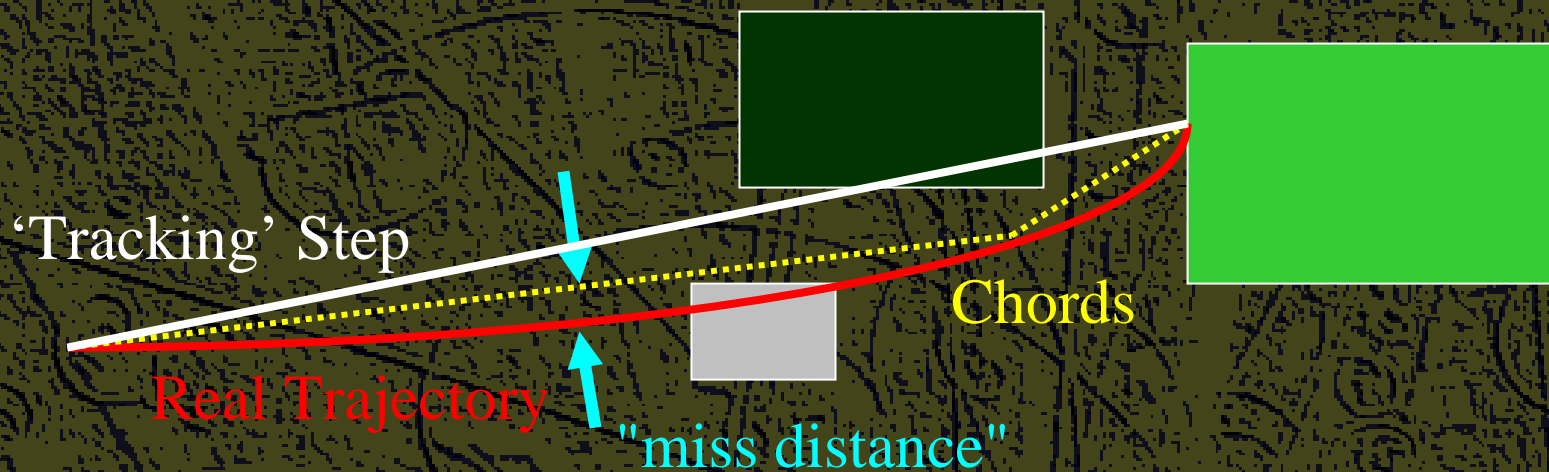
# Field integration

- ▶ In order to propagate a particle inside a field (e.g. magnetic, electric or both), we solve the equation of motion of the particle in the field.
- ▶ We use a Runge-Kutta method for the integration of the ordinary differential equations of motion.
  - ▶ Several Runge-Kutta 'steppers' are available.
- ▶ In specific cases other solvers can also be used:
  - ▶ In a uniform field, using the analytical solution.
  - ▶ In a nearly uniform field (BgsTransportation/future)
  - ▶ In a smooth but varying field, with new RK+helix.
- ▶ Using the method to calculate the track's motion in a field, Geant4 breaks up this curved path into linear chord segments.
  - ▶ We determine the chord segments so that they closely approximate the curved path.



# Tracking in field

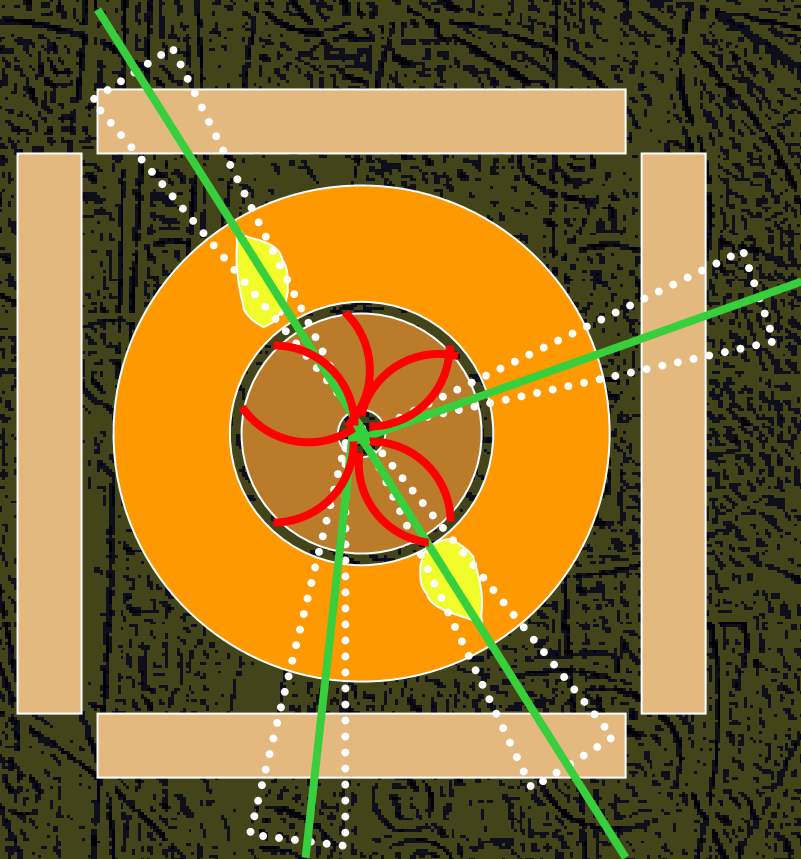
- ▶ We use the chords to interrogate the **G4Navigator**, to see whether the track has crossed a volume boundary.
- ▶ User can set the accuracy of the volume intersection,
  - ▶ By setting a parameter called the "miss distance"
    - ▶ It is a measure of the error in whether the approximate track intersects a volume.
- ▶ One physics/tracking step can create several chords.
  - ▶ In some cases, one step consists of several helix turns.





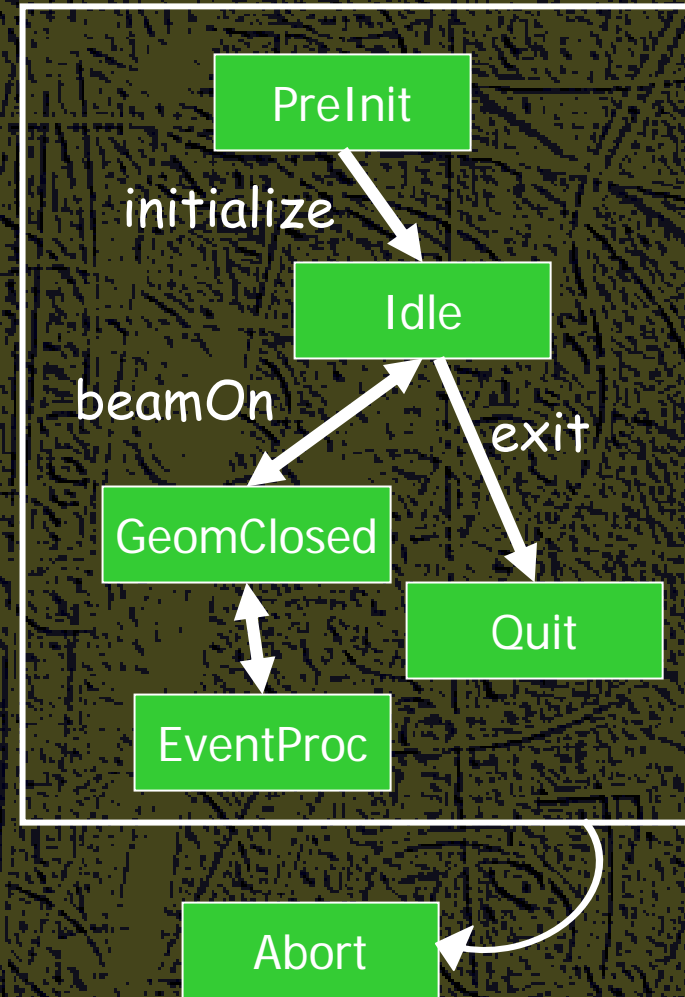
# Stack

- ▶ Track is a class object, thus it is easy to treat suspending or postponing tracks. For example,
  - ▶ Suspend tracks at the entrance of calorimeter, i.e. simulate all tracks in tracking region before generating showers.
  - ▶ Suspend a “looper” track after certain time and postpone it to next event.
  - ▶ Prioritized tracking without performance cost
- ▶ Stacks are managed by **G4StackManager** with user's G4UserStackingAction.
- ▶ Well-thought prioritization/abortion of tracks/events makes entire simulation process much more efficient.



# Geant4 as a state machine

- ▶ Geant4 has six application states.
  - ▶ G4State\_PreInit
    - ▶ Material, Geometry, Particle and/or Physics Process need to be initialized/defined
  - ▶ G4State\_Idle
    - ▶ Ready to start a run
  - ▶ G4State\_GeomClosed
    - ▶ Geometry is optimized and ready to process an event
  - ▶ G4State\_EventProc
    - ▶ An event is processing
  - ▶ G4State\_Quit
    - ▶ (Normal) termination
  - ▶ G4State\_Abort
    - ▶ A fatal exception occurred and program is aborting



# Unit system

- ▶ Internal unit system used in Geant4 is completely hidden not only from user's code but also from Geant4 source code implementation.

- ▶ Each hard-coded number must be multiplied by its proper unit.

```
radius = 10.0 * cm;
```

```
kineticE = 1.0 * GeV;
```

- ▶ To get a number, it must be divided by a proper unit.

```
G4cout << eDep / MeV << " [MeV]" << G4endl;
```

- ▶ Most of commonly used units are provided and user can add his/her own units.
- ▶ By this unit system, source code becomes more readable and importing / exporting physical quantities becomes straightforward.
  - ▶ For particular application, user can change the internal unit to suitable alternative unit without affecting to the result.

# G4cout, G4cerr

- ▶ G4cout and G4cerr are *ostream* objects defined by Geant4.
  - ▶ G4endl is also provided.
- ▶ Some GUIs are buffering output streams so that they display print-outs on another window or provide storing / editing functionality.
  - ▶ The user should not use `std::cout`, etc.
- ▶ The user should not use `std::cin` for input. Use user-defined commands provided by `intercoms` category in Geant4.

User classes

Geant 4

# User classes

- ▶ Initialization classes
  - ▶ Invoked at the initialization
    - ▶ **G4VUserDetectorConstruction**
    - ▶ **G4VUserPhysicsList**
  - ▶ Action classes
    - ▶ Invoked during an event loop
      - ▶ **G4VUserPrimaryGeneratorAction**
      - ▶ G4UserRunAction
      - ▶ G4UserEventAction
      - ▶ G4UserStackingAction
      - ▶ G4UserTrackingAction
      - ▶ G4UserSteppingAction
- ▶ **main()**
  - ▶ Geant4 does not provide *main()*.

Note : classes written in **yellow** are mandatory.

# Describe your detector

- ▶ Derive your own concrete class from **G4VUserDetectorConstruction** abstract base class.
- ▶ In the virtual method *Construct()*,
  - ▶ Instantiate all necessary materials
  - ▶ Instantiate volumes of your detector geometry
  - ▶ Instantiate your sensitive detector classes and set them to the corresponding logical volumes
- ▶ Optionally you can define
  - ▶ Regions for any part of your detector
  - ▶ Visualization attributes (color, visibility, etc.) of your detector elements

# Select physics processes

- ▶ Geant4 does not have any default particles or processes.
  - ▶ Even for the particle transportation, you have to define it explicitly.
- ▶ Derive your own concrete class from **G4VUserPhysicsList** abstract base class.
  - ▶ Define all necessary particles
  - ▶ Define all necessary processes and assign them to proper particles
  - ▶ Define cut-off ranges applied to the world (and each region)
- ▶ Geant4 provides lots of utility classes/methods and examples.
  - ▶ "Educated guess" physics lists for defining hadronic processes for various use-cases.



# Generate primary event

- ▶ Derive your concrete class from **G4VUserPrimaryGeneratorAction** abstract base class.
- ▶ Pass a G4Event object to one or more primary generator concrete class objects which generate primary vertices and primary particles.
- ▶ Geant4 provides several generators in addition to the G4VPrimaryParticlegenerator base class.
  - ▶ G4ParticleGun
  - ▶ G4HEPEvtInterface, G4HepMCInterface
    - ▶ Interface to /hepevt/ common block or HepMC class
  - ▶ G4GeneralParticleSource
    - ▶ Define radioactivity

# Optional user action classes

- ▶ All user action classes, methods of which are invoked during “Beam On”, must be constructed in the user’s *main()* and must be set to the RunManager.
- ▶ **G4UserRunAction**
  - ▶ G4Run\* GenerateRun()
    - ▶ Instantiate user-customized run object
  - ▶ void BeginOfRunAction(const G4Run\*)
    - ▶ Define histograms
  - ▶ void EndOfRunAction(const G4Run\*)
    - ▶ Store histograms
- ▶ **G4UserEventAction**
  - ▶ void BeginOfEventAction(const G4Event\*)
    - ▶ Event selection
    - ▶ Define histograms
  - ▶ void EndOfEventAction(const G4Event\*)
    - ▶ Analyze the event

# Optional user action classes

## ▶ `G4UserStackingAction`

- ▶ `void PrepareNewEvent()`
  - ▶ Reset priority control
- ▶ `G4ClassificationOfNewTrack ClassifyNewTrack(const G4Track*)`
  - ▶ Invoked every time a new track is pushed
  - ▶ Classify a new track -- priority control
    - ▶ Urgent, Waiting, PostponeToNextEvent, Kill
- ▶ `void NewStage()`
  - ▶ Invoked when the Urgent stack becomes empty
  - ▶ Change the classification criteria
  - ▶ Event filtering (Event abortion)

# Optional user action classes

## ▶ **G4UserTrackingAction**

- ▶ void PreUserTrackingAction(const G4Track\*)
  - ▶ Decide trajectory should be stored or not
  - ▶ Create user-defined trajectory
- ▶ void PostUserTrackingAction(const G4Track\*)

## ▶ **G4UserSteppingAction**

- ▶ void UserSteppingAction(const G4Step\*)
  - ▶ Kill / suspend / postpone the track
  - ▶ Draw the step (for a track not to be stored as a trajectory)

# The main program

- ▶ Geant4 does not provide the *main()*.
- ▶ In your *main()*, you have to
  - ▶ Construct G4RunManager (or your derived class)
  - ▶ Set user mandatory classes to RunManager
    - ▶ G4VUserDetectorConstruction
    - ▶ G4VUserPhysicsList
    - ▶ G4VUserPrimaryGeneratorAction
- ▶ You can define VisManager, (G)UI session, optional user action classes, and/or your persistency manager in your *main()*.

# Select (G)UI

- ▶ In your *main()*, according to your computer environments, construct a G4UIsession concrete class provided by Geant4 and invoke its *sessionStart()* method.
- ▶ Geant4 provides
  - ▶ G4UITerminal -- C- and TC-shell like character terminal
  - ▶ G4GAG -- Tcl/Tk or Java PVM based GUI
  - ▶ G4Wo -- Opacs
  - ▶ G4JAG -- Interface to JAS (Java Analysis Studio)
  - ▶ G4UIBatch -- Batch job with macro file

# Visualization

- ▶ Derive your own concrete class from G4VVisManager according to your computer environments.
- ▶ Geant4 provides interfaces to graphics drivers
  - ▶ DAWN -- Fukui renderer
  - ▶ WIRED
  - ▶ RayTracer -- Ray tracing by Geant4 tracking
  - ▶ OPACS
  - ▶ OpenGL
  - ▶ OpenInventor
  - ▶ VRML

# Environment variables

- ▶ You need to set following environment variables to compile, link and execute Geant4-based simulation.
  - ▶ Mandatory variables
    - ▶ G4SYSTEM – OS (e.g. Linux-g++)
    - ▶ G4INSTALL – base directory of Geant4
    - ▶ G4WORKDIR – your temporary work space
    - ▶ CLHEP\_BASE\_DIR – base directory of CLHEP
  - ▶ Variables for physics processes in case corresponding processes are used
    - ▶ G4LEVELGAMMADATA - photon evaporation
    - ▶ G4LEDATA - cross-sections for Low-E EM module
    - ▶ G4RADIOACTIVEDATA - radioactive decay
    - ▶ NeutronHPCrossSections - neutron cross-section
  - ▶ Additional variables for GUI/Vis/Analysis



# (Graphical) User Interfaces

- ▶ Geant4 kernel is independent to any specific GUI technology.
- ▶ Geant4 provides several alternative (G)UIs or interfaces to external GUI packages. The user can choose one or more of them according to computer environment / need.
  - ▶ Character terminal (csh and tcsh(bash)-like terminal)
  - ▶ Xm, Xaw, Win32, variations of the upper terminals by using a Motif, Athena or Windows widget to retrieve commands
  - ▶ GAG, a fully Graphical User Interface and its extension GainServer of the client/server type
  - ▶ OPACS, an OPACS/Wo widget manager implementation in conjunction with the OPACS visualization system.
  - ▶ JAG, an interface to JAS (Java Analysis Studio)
  - ▶ User can connect his/her own GUI to Geant4

# Visualization

- ▶ Geant4 kernel is independent to any specific visualization technology.
- ▶ Geant4 provides several alternative visualization drivers or interfaces to external visualization drivers. The user can choose one or more of them according to computer environment / need.
  - ▶ OpenGL viewers
  - ▶ FukuiRenderer (DAWN)
  - ▶ VRML builder
  - ▶ WIRED
  - ▶ Wo, Xo (OPACS)
  - ▶ OpenInventorX (OIX)
  - ▶ RayTracer
  - ▶ User can connect his/her own visualization driver to Geant4
- ▶ Some example figures are given with introduction of users applications in this presentation