GENERAL INTRODUCTION

Training course at International User Conference on Medicine and Biology applications

Bordeaux, 8-11 October 2013

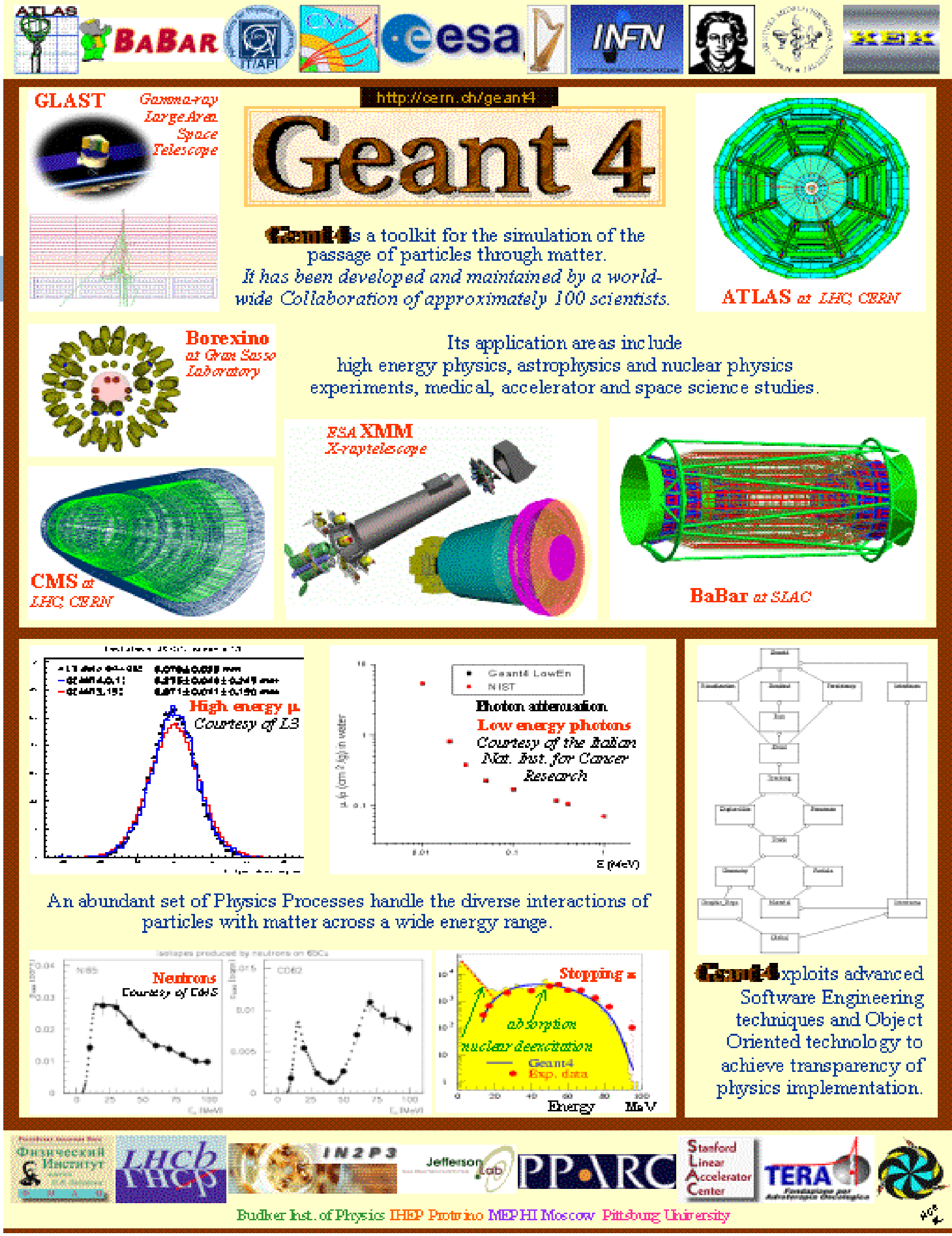
V. Ivanchenko

adaptation of the original lecture of

Makoto Asai (SLAC)

Outline

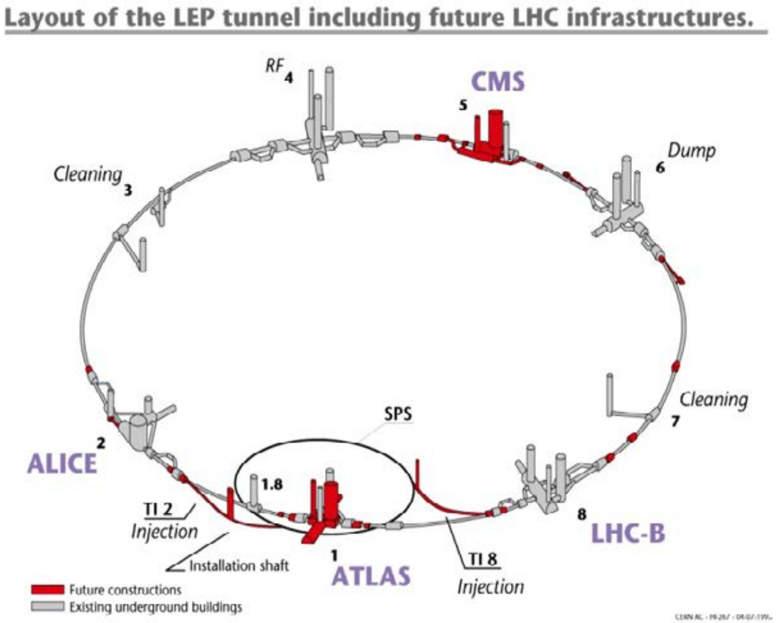
- General introduction and brief history
- Highlights of user applications
- Geant4 license
- Geant4 kernel
  - Basic concepts and kernel structure
  - User classes

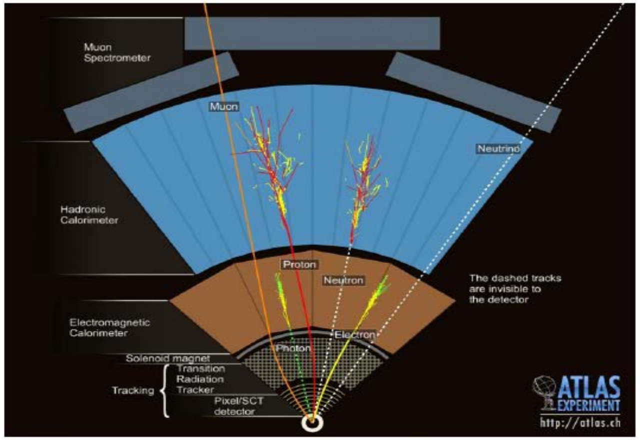


Geant4 History

- Dec 1994 - Project start
- Apr 1997 - First alpha release
- Jul 1998 - First beta release
- Dec 1998 - First Geant4 public release - version 1.0
- …
- June 2007 - Geant4 version 9.0
- ….
- Dec 3d, 2012 - Geant4 version 9.6
  - May 24th, 2013 - Geant4 9.6-patch02 release
- Dec 6th, 2013 - Geant4 version 10.0

- Geant4 developments were strongly supported by HEP community
  - LHC experiments are the goal
  - CERN, SLAC, KEK, TRIUF, IN2P3, INFN….
- Space and medical user communities also contribute significantly to Geant4 developments





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 and Boolean solids
  - Placement, replica, divided, parameterized, reflected and grouped
  - XML interface
- Everything is open to the user - user may become a developer
  - Choice of physics processes/models
  - Choice of GUI/Visualization/persistency/histogramming technologies

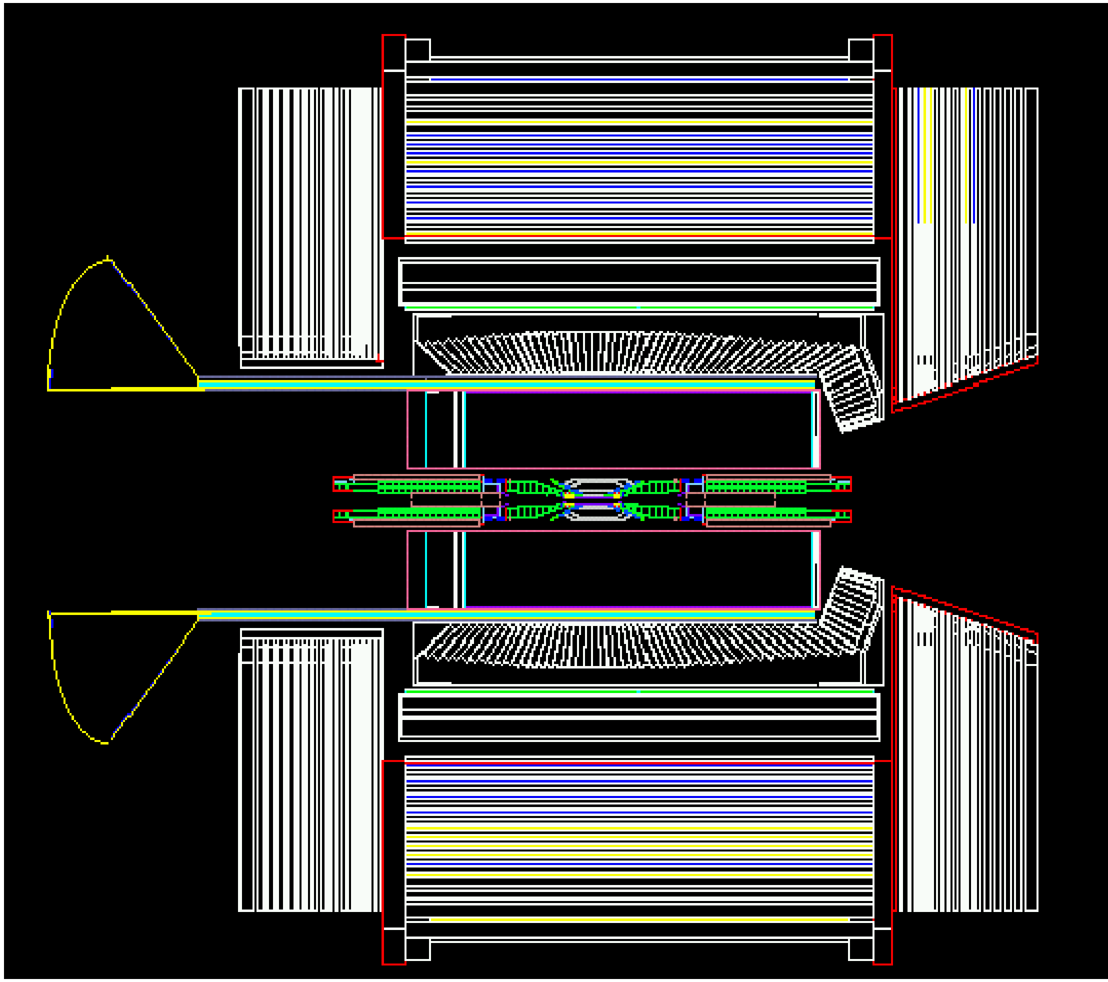
HIGHLIGHTS OF USERS APPLICATIONS

To provide you some ideas how Geant4 would be utilized...

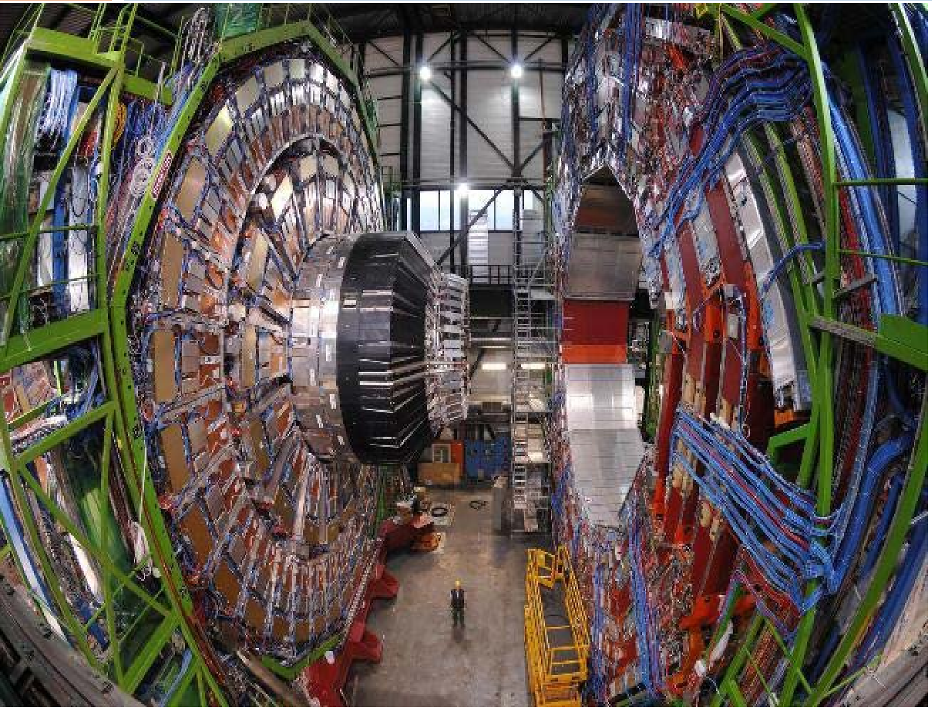
BaBar

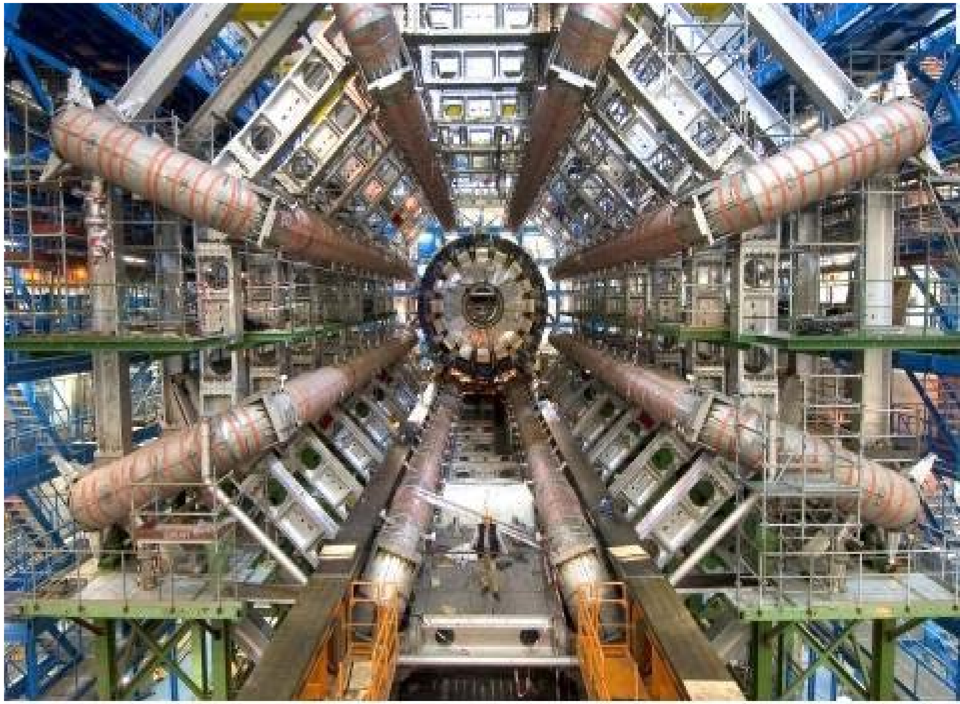
Courtesy of D.Wright (SLAC)

- BaBar at SLAC is the pioneer experiment in HEP in use of Geant4
  - Started in 2000
  - Simulated ~$2*10^{10}$ events so far
  - Produced at 20 sites in North America and Europe



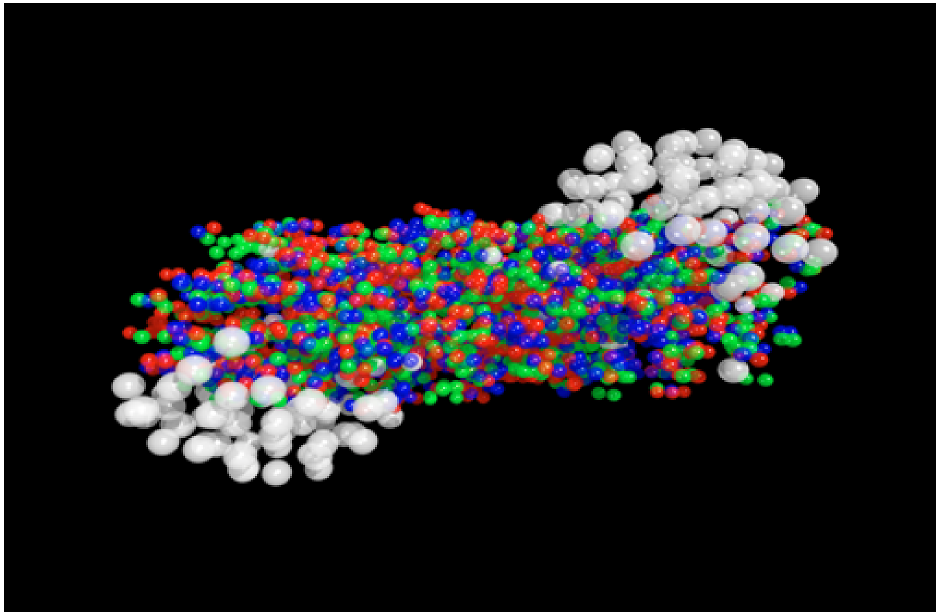
Geant4 now used by all LHC detectors

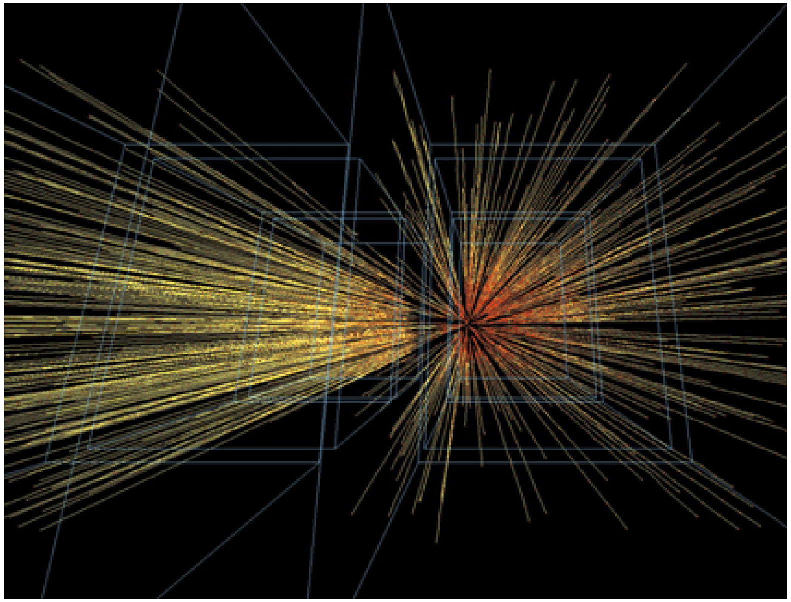




- ATLAS, CMS – greatest detectors
- LHCb, ALICE – large specific detectors

Pushing G4 to the limits: Heavy Ions





Events with > 50000 particles/event in detector acceptance

Albert De Roeck (CERN)27

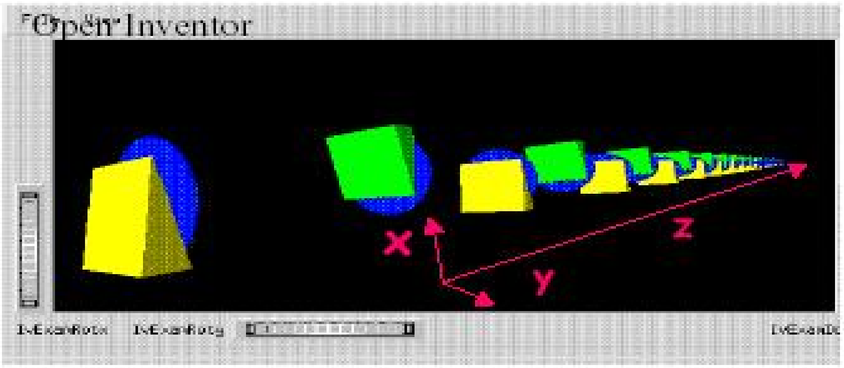
Geant4 for beam transportation

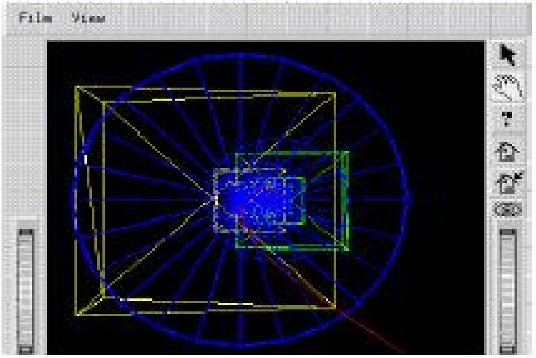
Example: Helical Channel

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

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

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



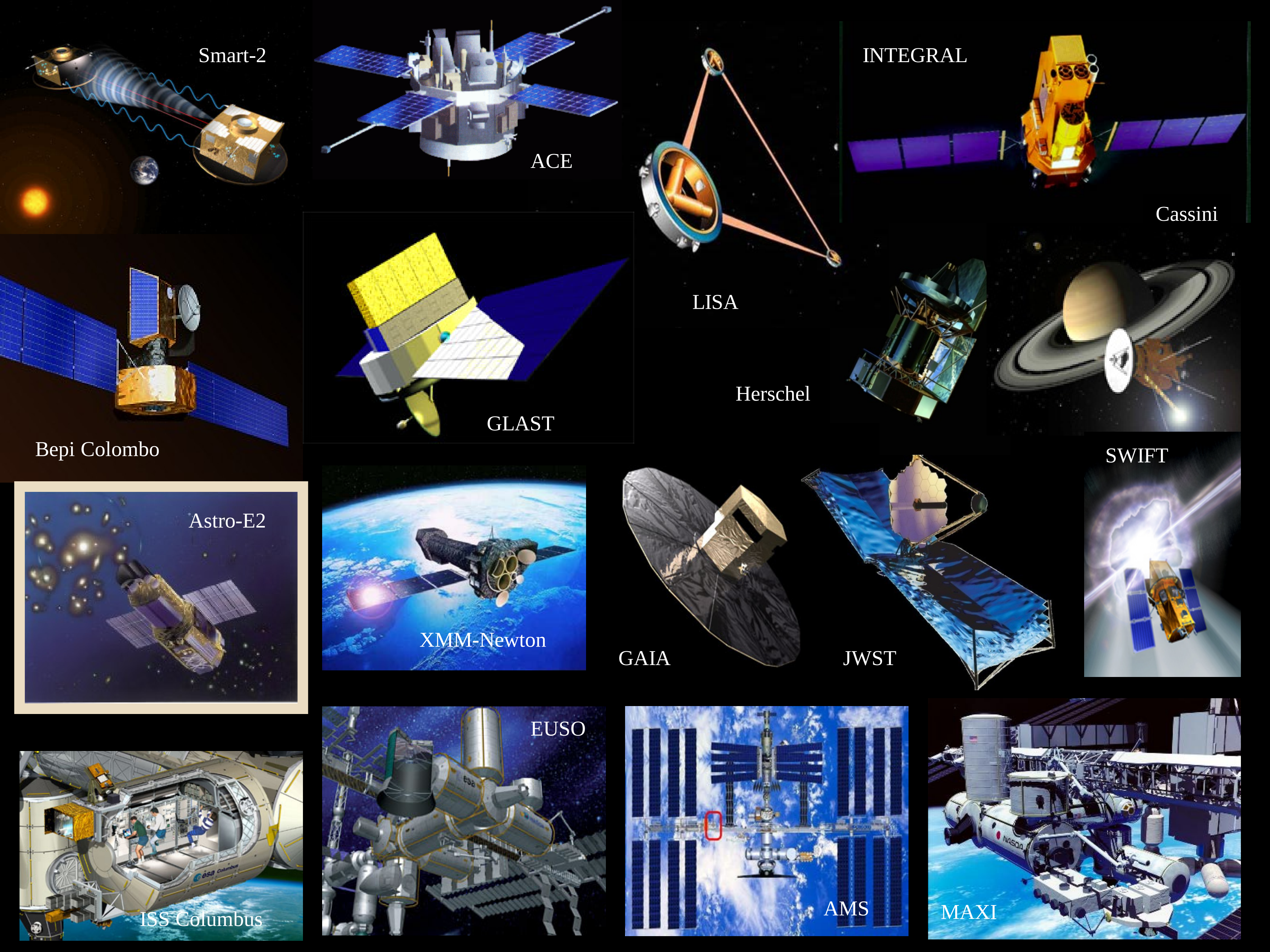


Other simulations:

- 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)

G4 Users Meeting, February 21st , 2002 *V. Daniel Elvira, Fermilab*

Courtesy of V.D.Elvira (FNAL)


Smart-2
ACE
LISA
INTEGRAL
Cassini
Bepi Colombo
GLAST
Herschel
SWIFT
Astro-E2
XMM-Newton
GAIA
JWST
EUSO
ISS Columbus
AMS
MAXI

PlanetoCosmics

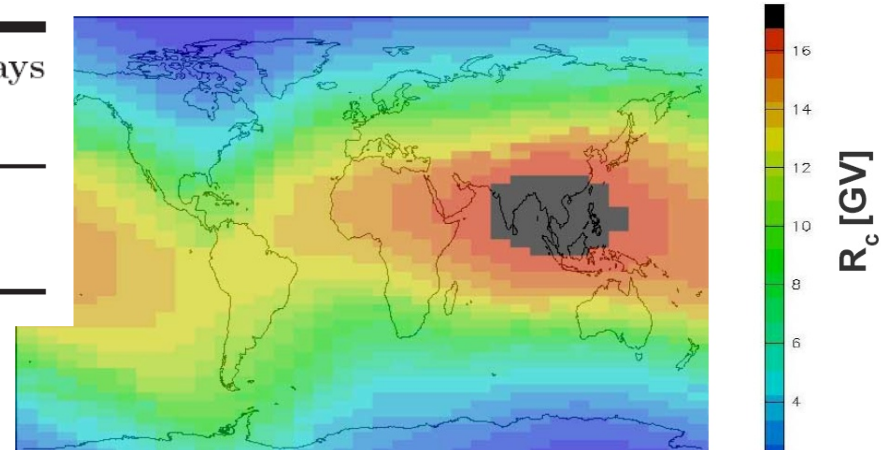
Geant4 simulation of Cosmic Rays in planetary Atmo-/Magneto- spheres

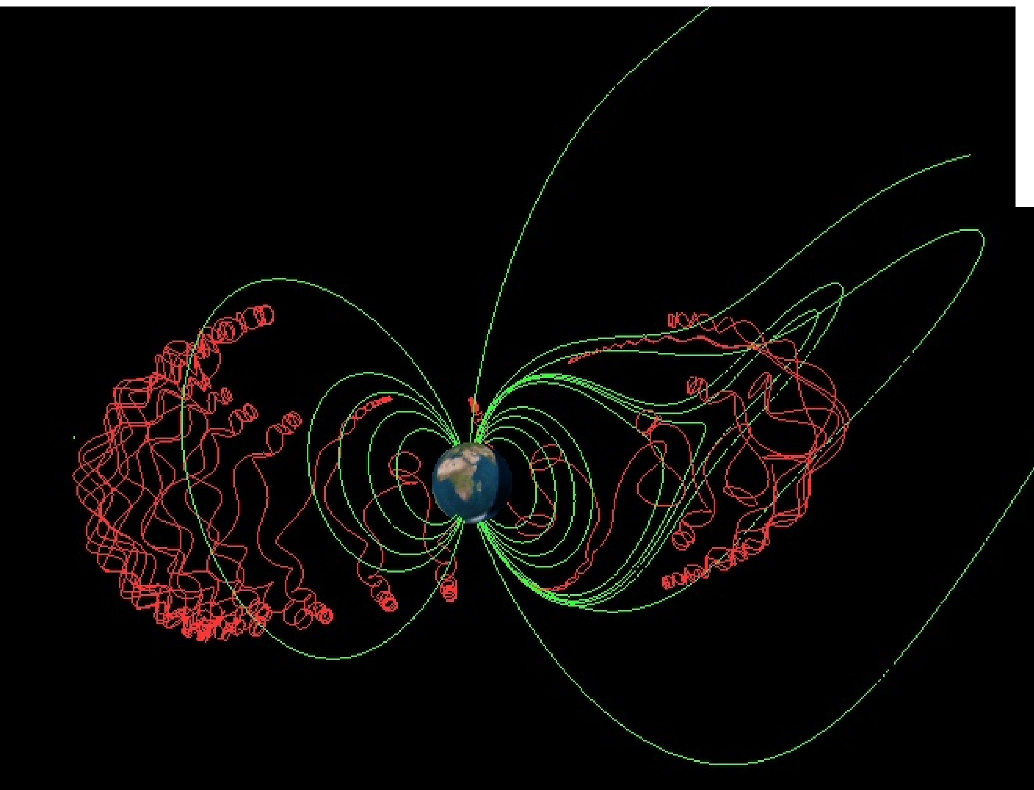
28th International Cosmic Ray Conference — 4277

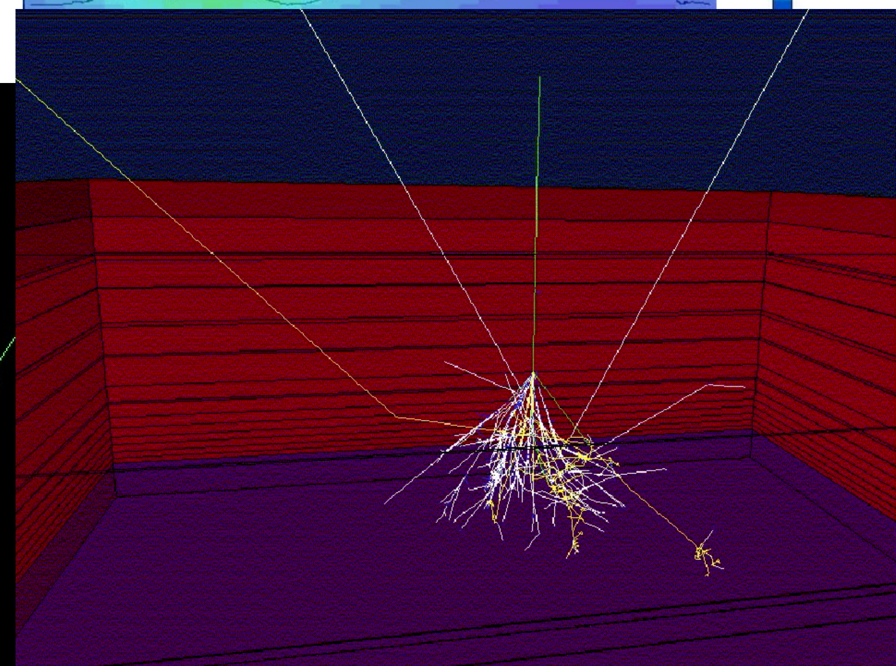
Geant4 Simulation of the Propagation of Cosmic Rays through the Earth's Atmosphere

L. Desorgher, E. O. Flückiger, M. R. Moser, and R. Bütikofer
Physikalisches Institut, University of Bern, CH-3012 Bern, Switzerland

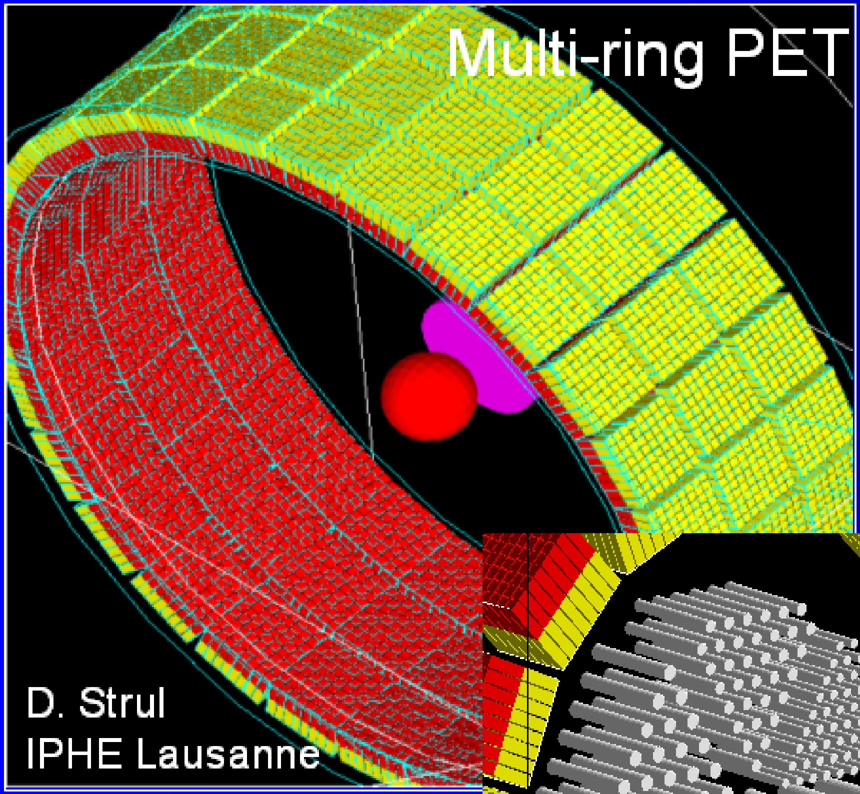
Cutoff Rigidities vs position

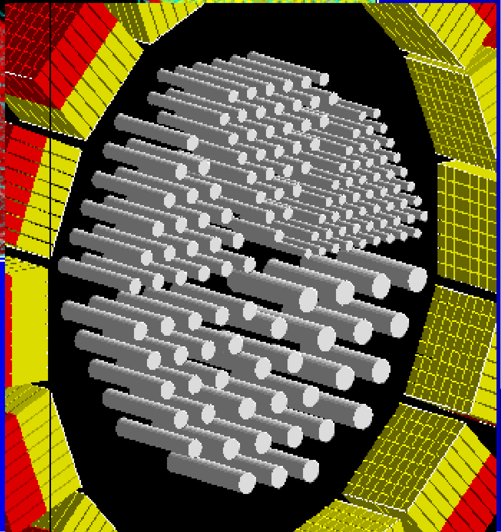


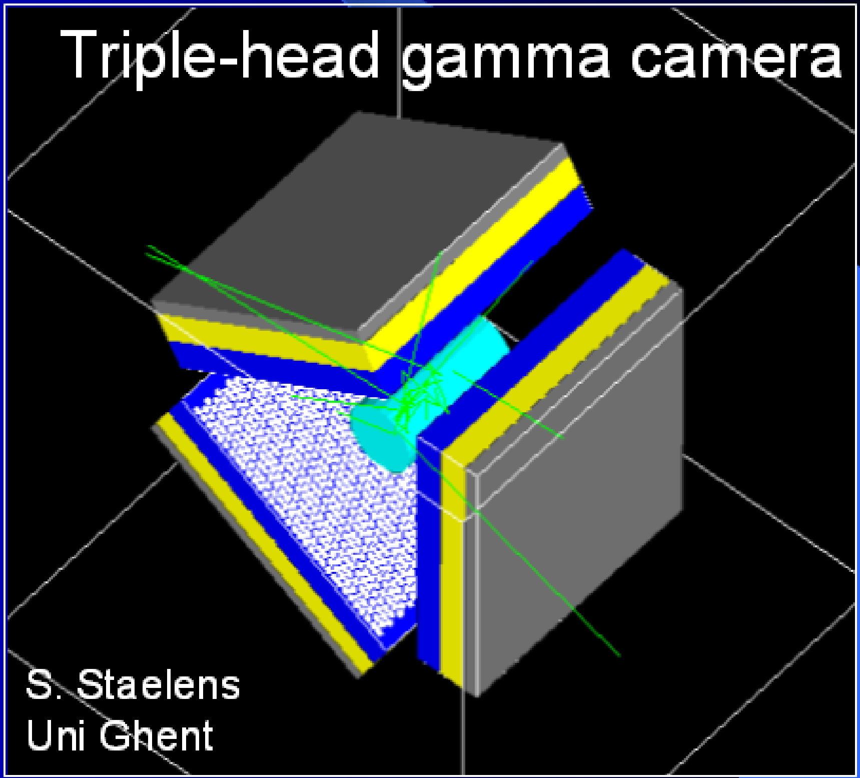




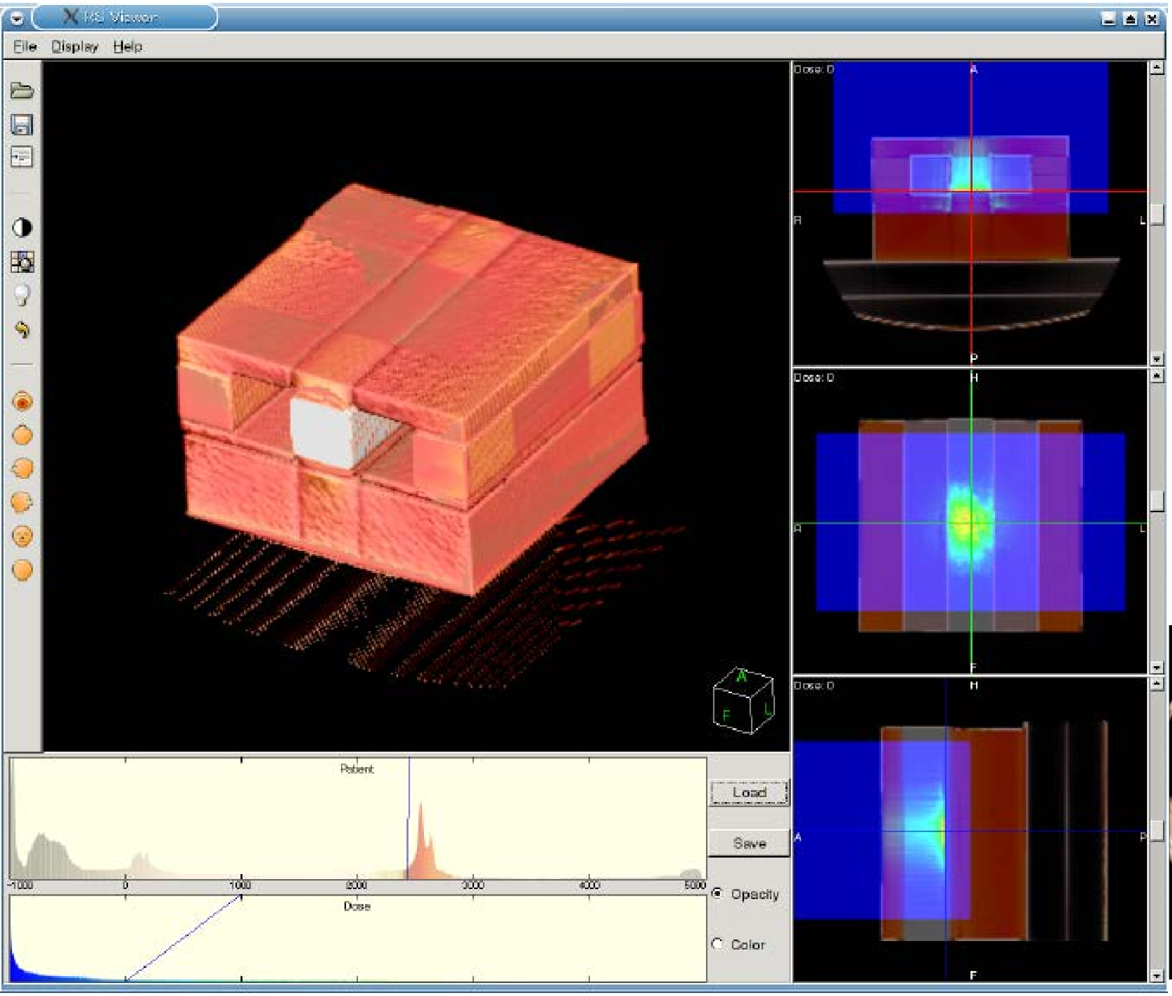
Geometry examples of GATE applications

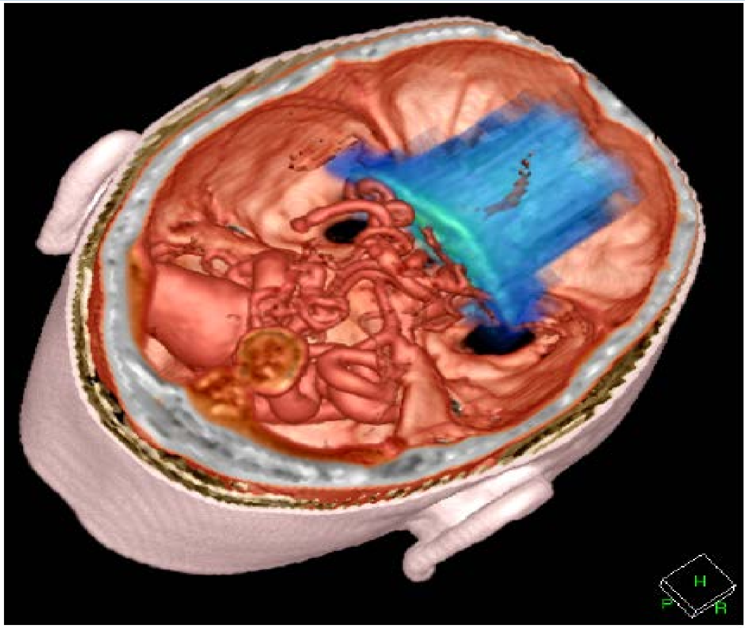


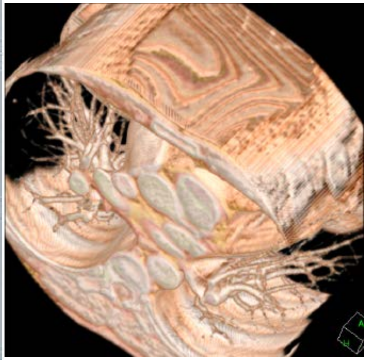


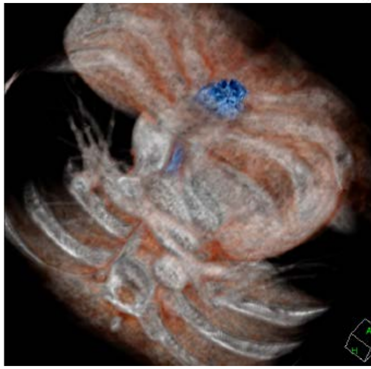


Screen shots of gMocren





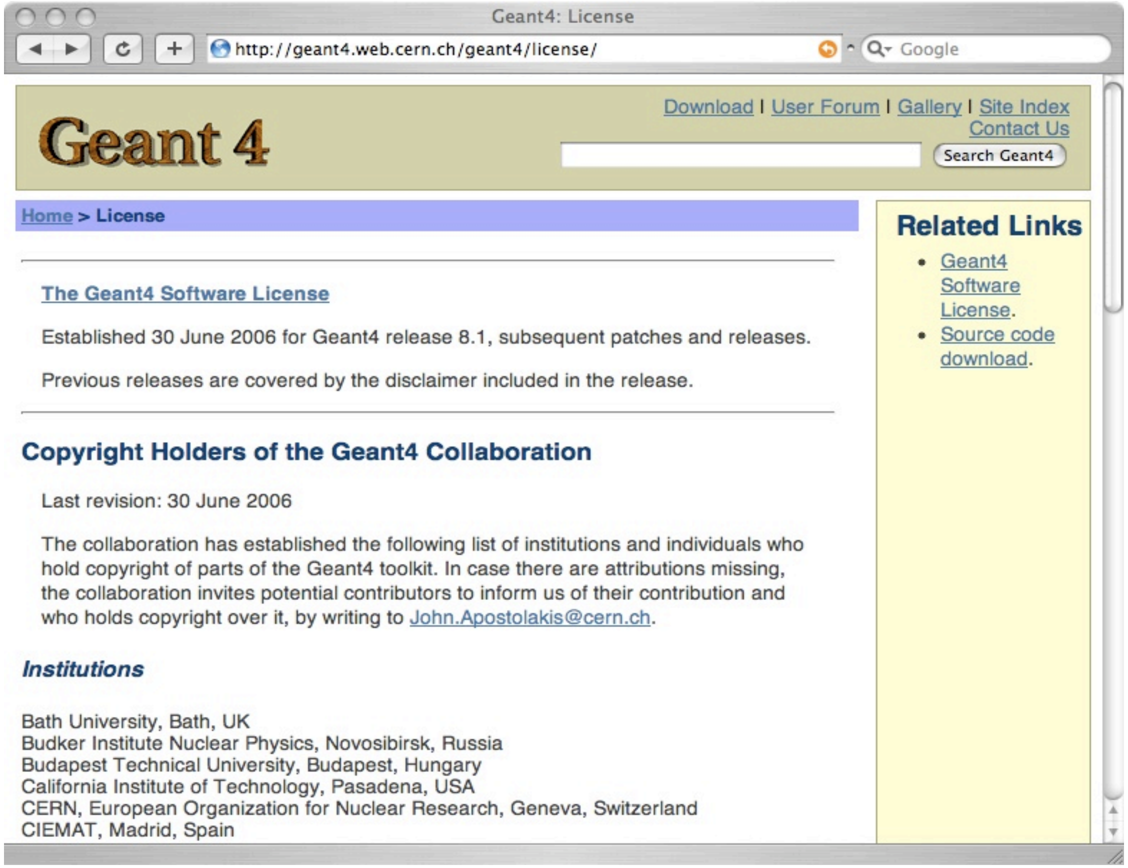




GEANT4 LICENSE

The Geant4 License

In response to user requests for clarification of Geant4's distribution policy, the collaboration established a license.



- ❑Makes clear the user's wide-ranging freedom to use, extend or redistribute Geant4, even as part of some for-profit venture.
- ❑Simple enough that you can read and understand it.

- http://cern.ch/geant4/license/

Geant4 License Highlights

- Establishes the Geant4 copyright
- Prohibits others from claiming that they are Geant4

- If you develop something in or based on Geant4 and give it away, Geant4 can have it for free, too
- Any documentation you produce must refer to Geant4
- You cannot patent the parts already written by the collaboration

- We don't claim that it works, and we're not responsible if it doesn't

Geant4 general references

- Recent Improvements in Geant4 Electromagnetic Physics Models and Interfaces. V. Ivanchenko et al., Progress in NUCLEAR SCIENCE and TECHNOLOGY, Vol. 2, pp.898-903, 2011.
- • Geometry and physics of the Geant4 toolkit for high and medium energy applications. J. Apostolakis et al., Radiation Physics and Chemistry 78: 859-873, 2009.
- • Geant4 developments and applications. J. Allison et al., IEEE Trans. Nucl. Sci. 53 (1): 270-278, 2006.
- • Geant4: A Simulation toolkit. By GEANT4 (S. Agostinelli et al.), Nucl. Instrum. Meth. A506 (3): 250-303, 2003.

BASIC CONCEPTS AND KERNEL STRUCTURE

Geant 4

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.

```
G4cout << "Hello Geant4!" << G4endl;
```

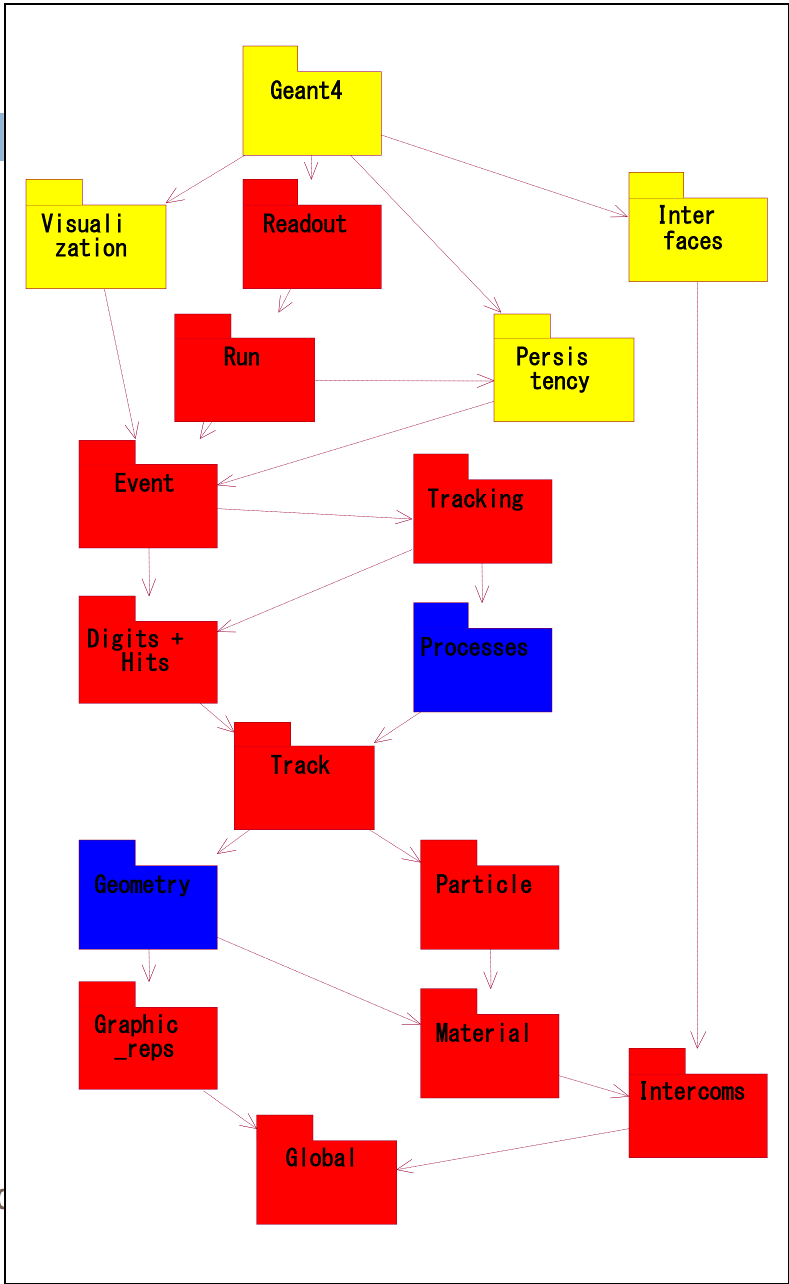
- Some GUIs are buffering output streams so that they display print-outs on another window or provide storing / editing functionality.
  - The user should not use std::cout, etc.
- The user should not use std::cin for input. Use user-defined commands provided by intercoms category in Geant4.
  - Ordinary file I/O is OK.

Terminology

- Step – the smallest unit of Geant4 simulation, a particle is transported from one point to another
- Trajectory and TrajectoryPoint – collection of steps and step points
- Process – the physics that happens along a step
- Track – a snapshot of a particle at some point along its path (not the same as trajectory)
- Event – a collection of info from tracks and particle trajectories
- Run – a collection of events

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 setup
  - settings of physics processes
- Conceptually, a run is a collection of events
  - A run consists of one event loop.
- 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.
  - A run class may have a summary results of the run.
- G4UserRunAction is the optional user hook.

Event in Geant4

- An event is the basic unit of simulation in Geant4.
- At beginning of processing, primary tracks are generated. These primary tracks are pushed into a stack.
- A track is popped up from the stack one by one and "tracked". Resulting secondary tracks are pushed into the stack.
  - This "tracking" lasts as long as the stack has a track.
- When the stack becomes empty, processing of one event is over.
- G4Event class represents an event. It has following objects at the end of its (successful) processing.
  - List of primary vertices and particles (as input)
  - Hits and Trajectory collections (as output)
- G4EventManager class manages processing an event. G4UserEventAction is the optional user hook.

Track in Geant4

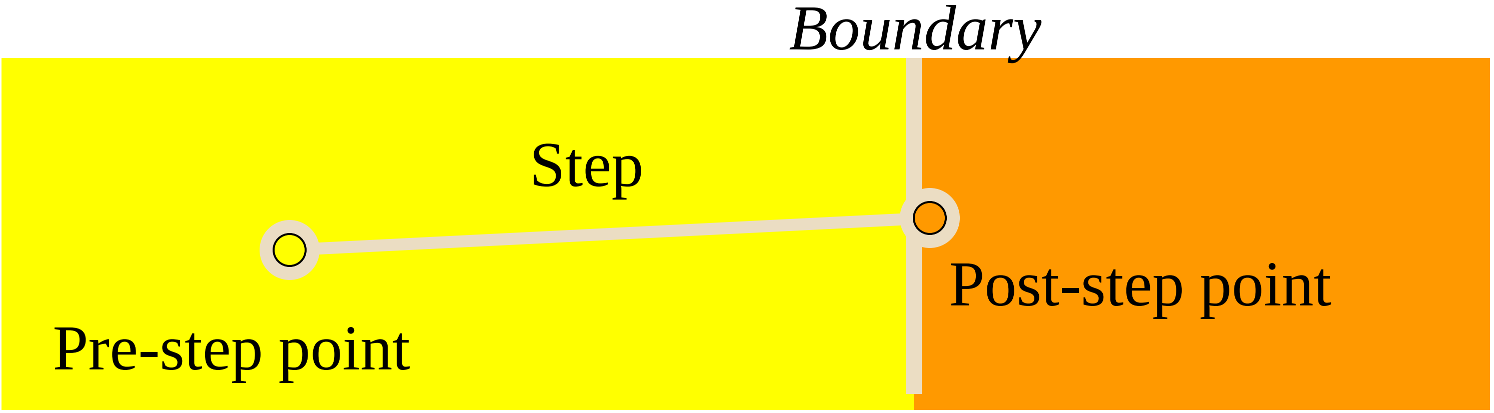
- Track is a snapshot of a particle.
  - It has physical quantities of current instance only. It does not record previous quantities.
  - Step is a "delta" information to a track. Track is not a collection of steps. Instead, a track is being updated by steps.
- Track object is deleted when
  - it goes out of the world volume,
  - it disappears (by e.g. decay, inelastic scattering),
  - it goes down to zero kinetic energy and no "AtRest" additional process is required, or
  - the user decides to kill it artificially.
- No track object persists at the end of event.
  - For the record of tracks, use trajectory class objects.
- G4TrackingManager manages processing a track, a track is represented by G4Track class.
- G4UserTrackingAction is the optional user hook.

Track status

- At the end of each step, according to the processes involved, the state of a track may be changed.
  - The user can also change the status in UserSteppingAction.
  - Statuses shown in green are artificial, i.e. Geant4 kernel won't set them, but the user can set.
- fAlive
  - Continue the tracking.
- fStopButAlive
  - The track has come to zero kinetic energy, but still AtRest process to occur.
- fStopAndKill
  - The track has lost its identity because it has decayed, interacted or gone beyond the world boundary.
  - Secondaries will be pushed to the stack.
- fKillTrackAndSecondaries
  - Kill the current track and also associated secondaries.
- fSuspend
  - Suspend processing of the current track and push it and its secondaries to the stack.
- fPostponeToNextEvent
  - Postpone processing of the current track to the next event.
  - Secondaries are still being processed within the current event.

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, and G4StepPoint classes.
- G4UserSteppingAction is the optional user hook.



Step Status

- Status is attached to each G4StepPoint to show how step was determined
  - use PostStepPoint to get status of current step
  - PreStepPoint has status of previous step
- Step status codes:
  - fWorldBoundary : step at edge of world volume
  - fGeomBoundary : step limited by a volume boundary other than the world
  - fAtRestDoItProc, fAlongStepDoItProc, fPostStepDoItProc : step is limited by one three types of process
  - fUserDefinedLimit : step limited by user
  - fUndefined : step not defined yet

Trajectory and trajectory point

- Track does not keep its trace. No track object persists at the end of event.
- G4Trajectory is the class which copies some of G4Track information. G4TrajectoryPoint is the class which copies some of G4Step information.
  - G4Trajectory has a vector of G4TrajectoryPoint.
  - At the end of event processing, G4Event has a collection of G4Trajectory objects.
    - /tracking/storeTrajectory must be set to 1.
- Keep in mind the distinction.
  - G4Track ←→ G4Trajectory, G4Step ←→ G4TrajectoryPoint
- Given G4Trajectory and G4TrajectoryPoint objects persist till the end of an event, you should be careful not to store too many trajectories.
  - E.g. avoid for high energy EM shower tracks.
- G4Trajectory and G4TrajectoryPoint store only the minimum information.
  - You can create your own trajectory / trajectory point classes to store information you need. G4VTrajectory and G4VTrajectoryPoint are base classes.

Particle in Geant4

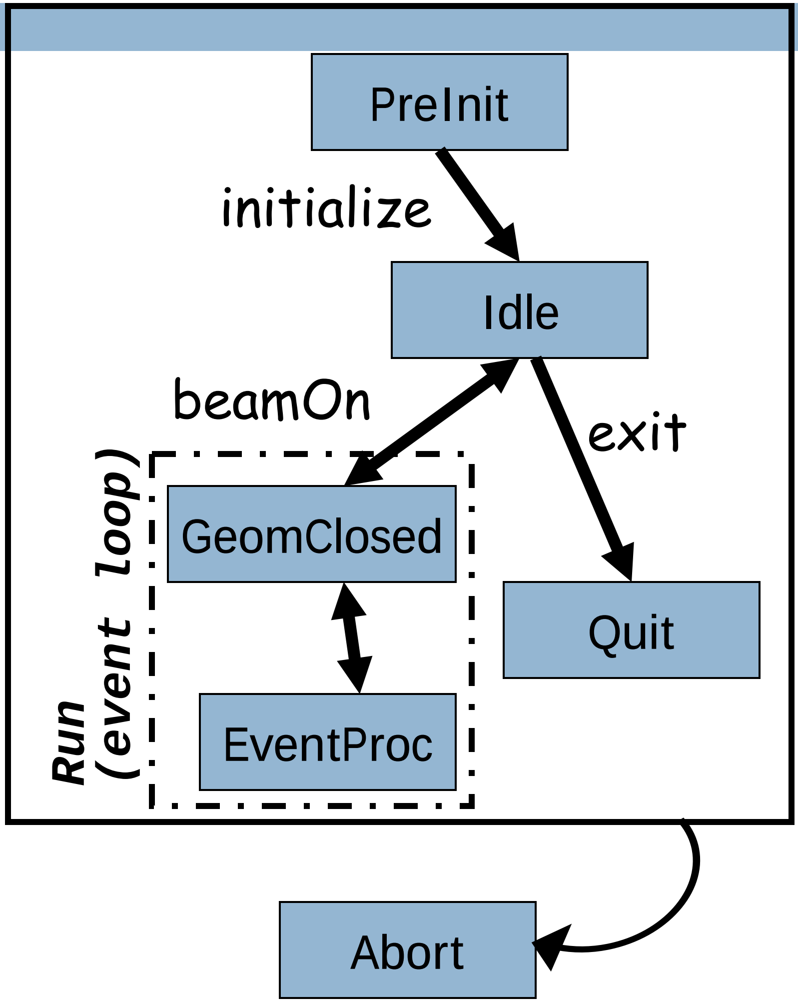
- A particle in Geant4 is represented by three layers of classes.
- G4Track
  - Position, geometrical information, etc.
  - This is a class representing a particle to be tracked.
- G4DynamicParticle
  - "Dynamic" physical properties of a particle, such as momentum, energy, spin, etc.
  - Each G4Track object has its own and unique G4DynamicParticle object.
  - This is a class representing an individual particle.
- 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.

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



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()*.

Batch Mode

- A Geant4 simulation can be executed in a batch mode.
  - A macro file consists of a series of UI commands
  - A macro file can be specified as an argument.

```
$ task2a myrun.mac >& myrun.log  (csh)
# task2a myrun.mac > myrun.log 2>&1 (bash)
```

- To enable batch mode,
  - In your main(),

```
G4UImanager* UI = G4UImanager::GetUIpointer();
G4String command = “/control/execute ”;
G4String fileName = argv[1];
UI-> applyCommand(command+fileName);
```

Thank you for your attention!