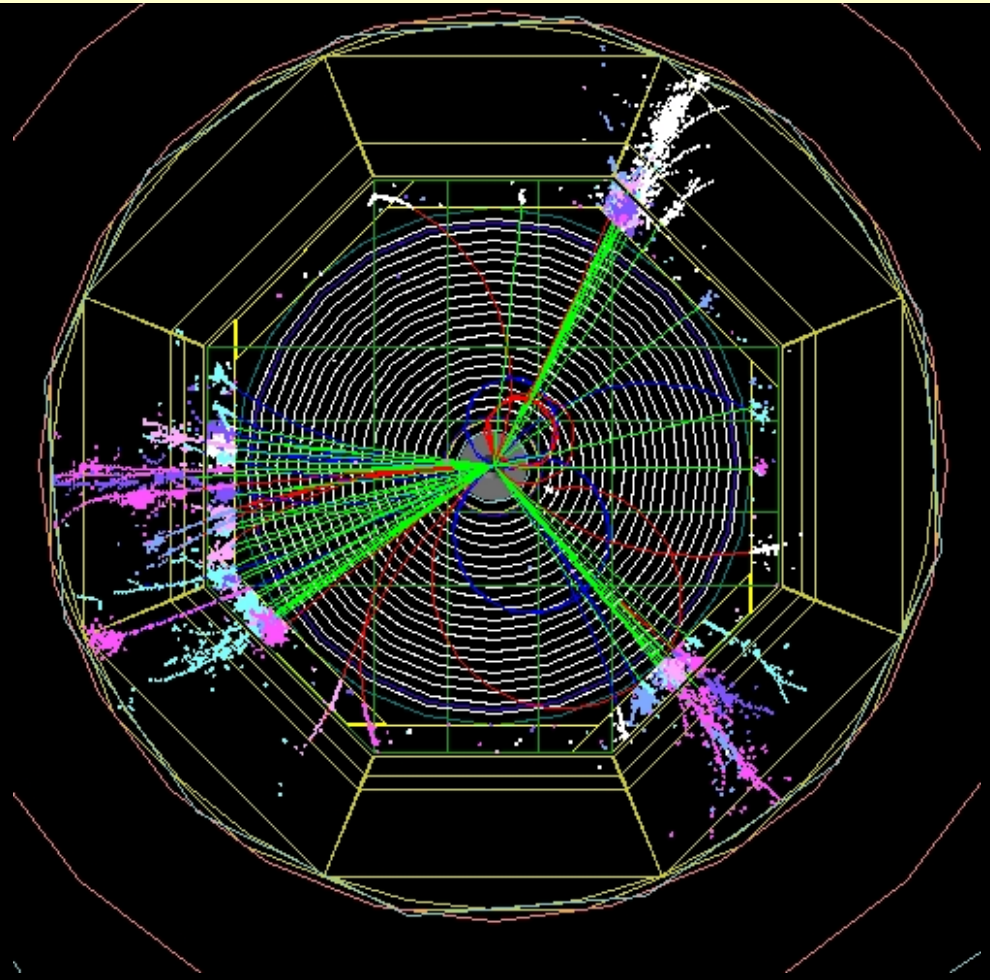
Geant4 simulation for the FLC detector models with Mokka

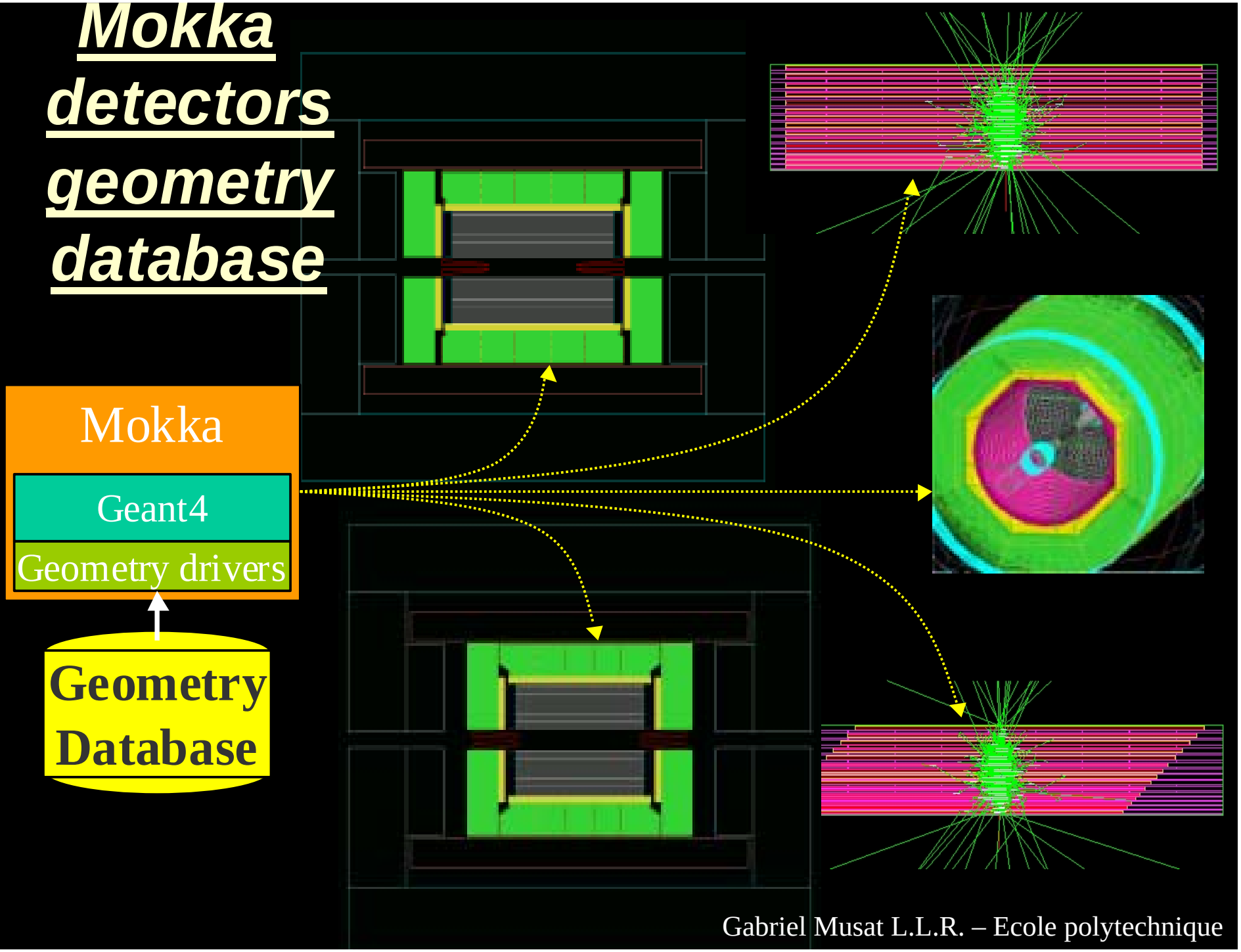


Gabriel Musat L.L.R. – Ecole polytechnique

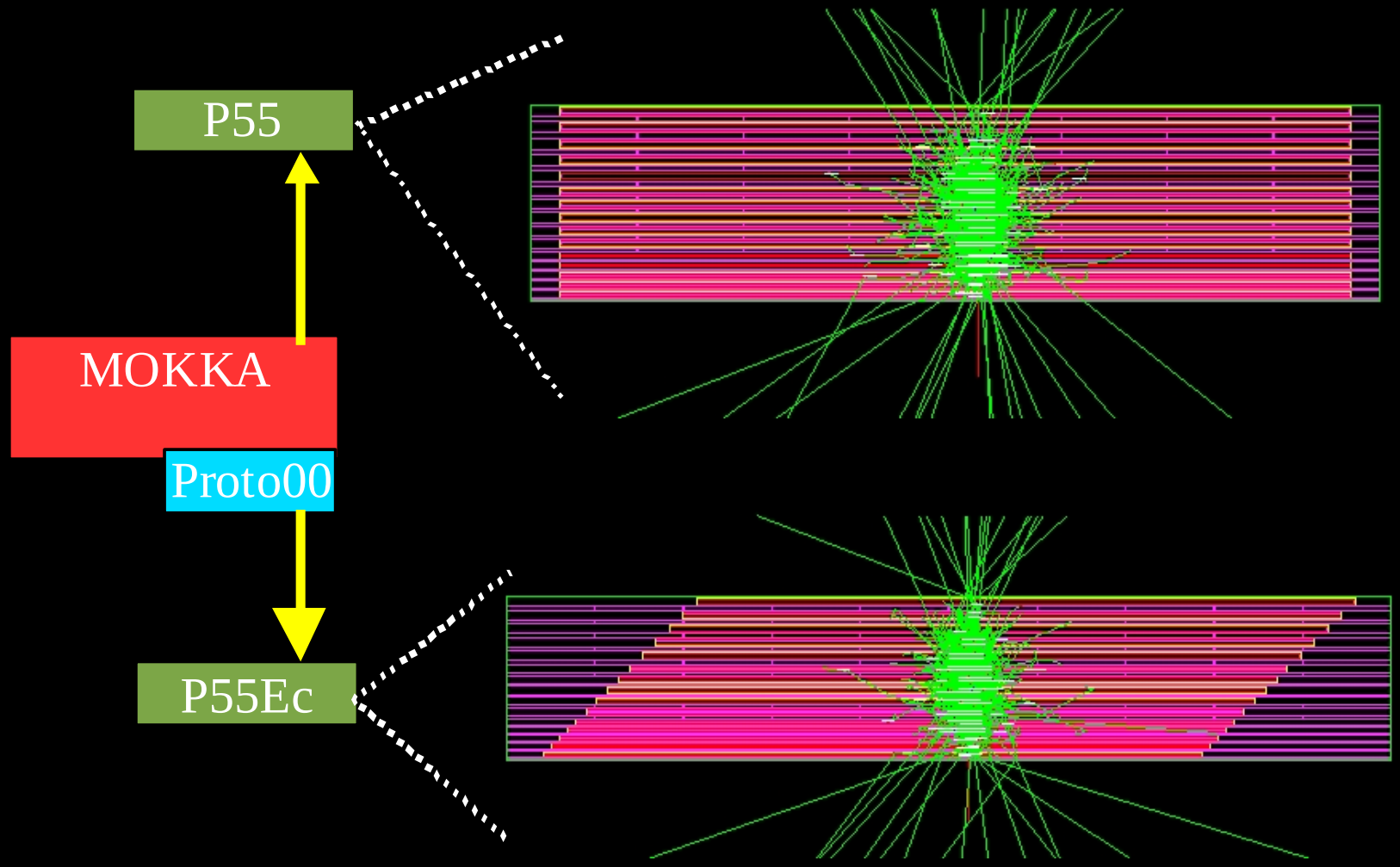
What is Mokka?

- Developped at L.L.R. since the end of 1999.
- Now being developped in an informal collaboration
- Implementation of almost all detector pieces

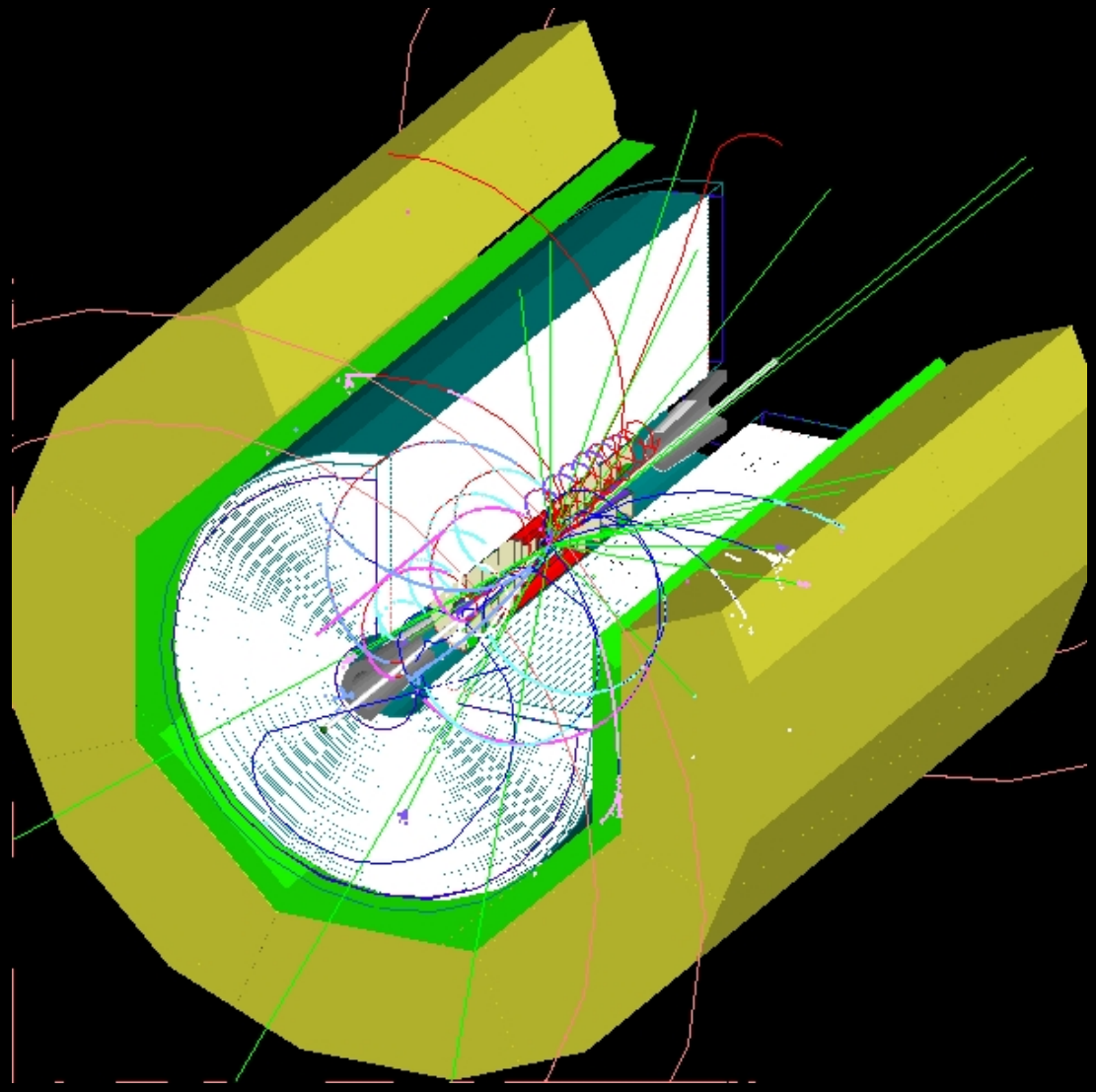
Mokka detectors geometry database



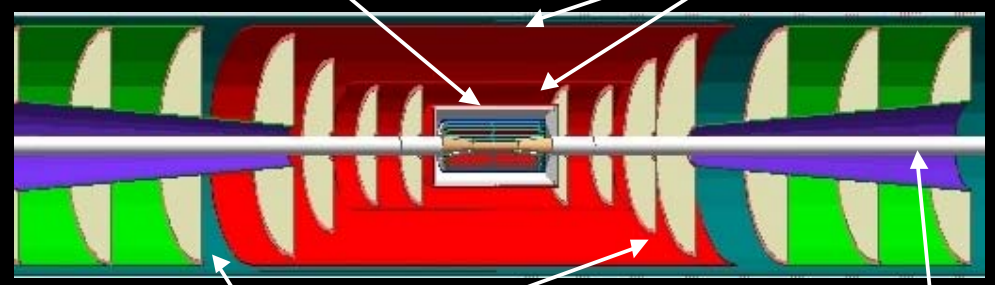
Mokka geometry drivers and databases



Last Detector Model: “D09M1”

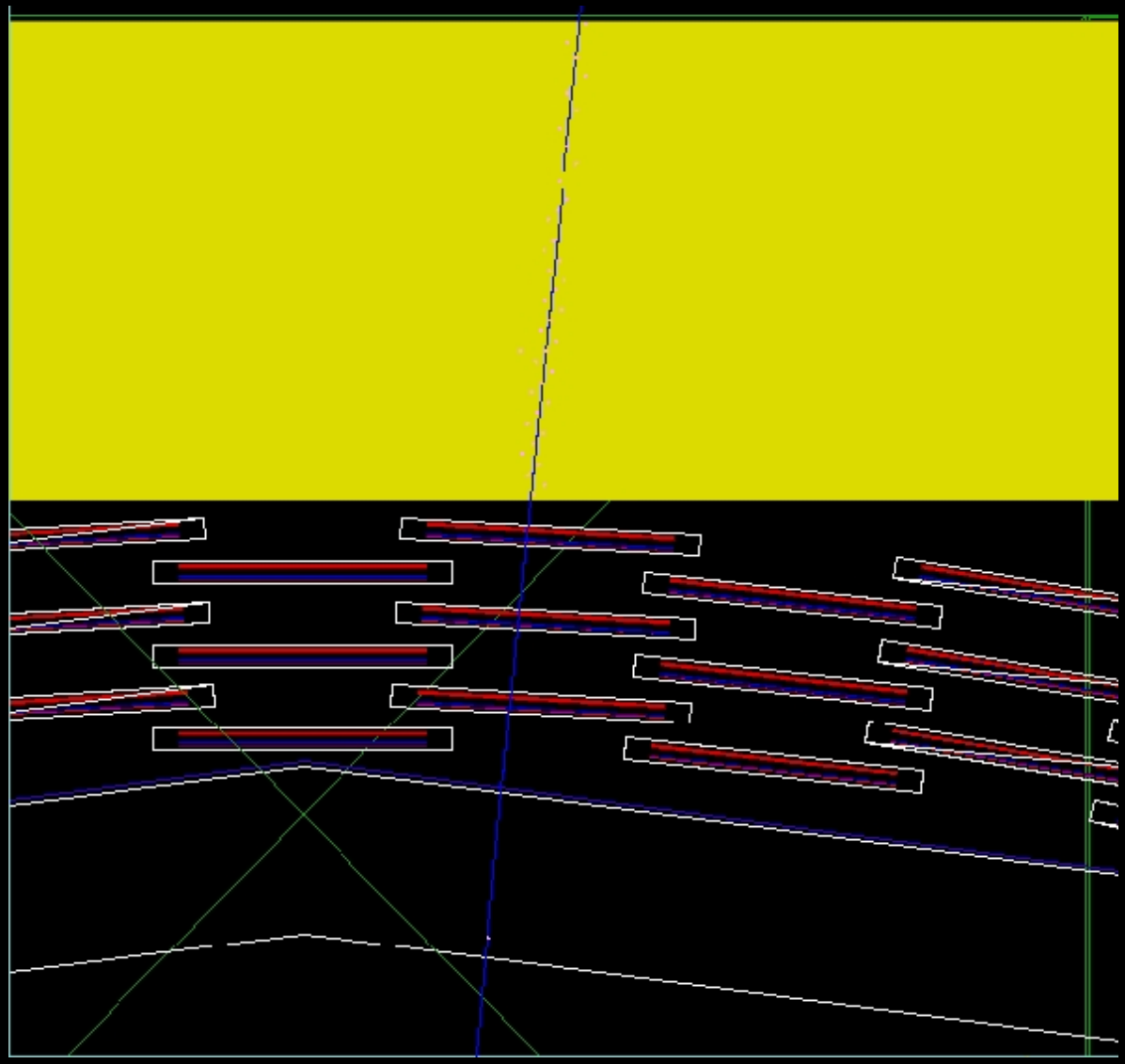


Mokka - the inner tracker devices

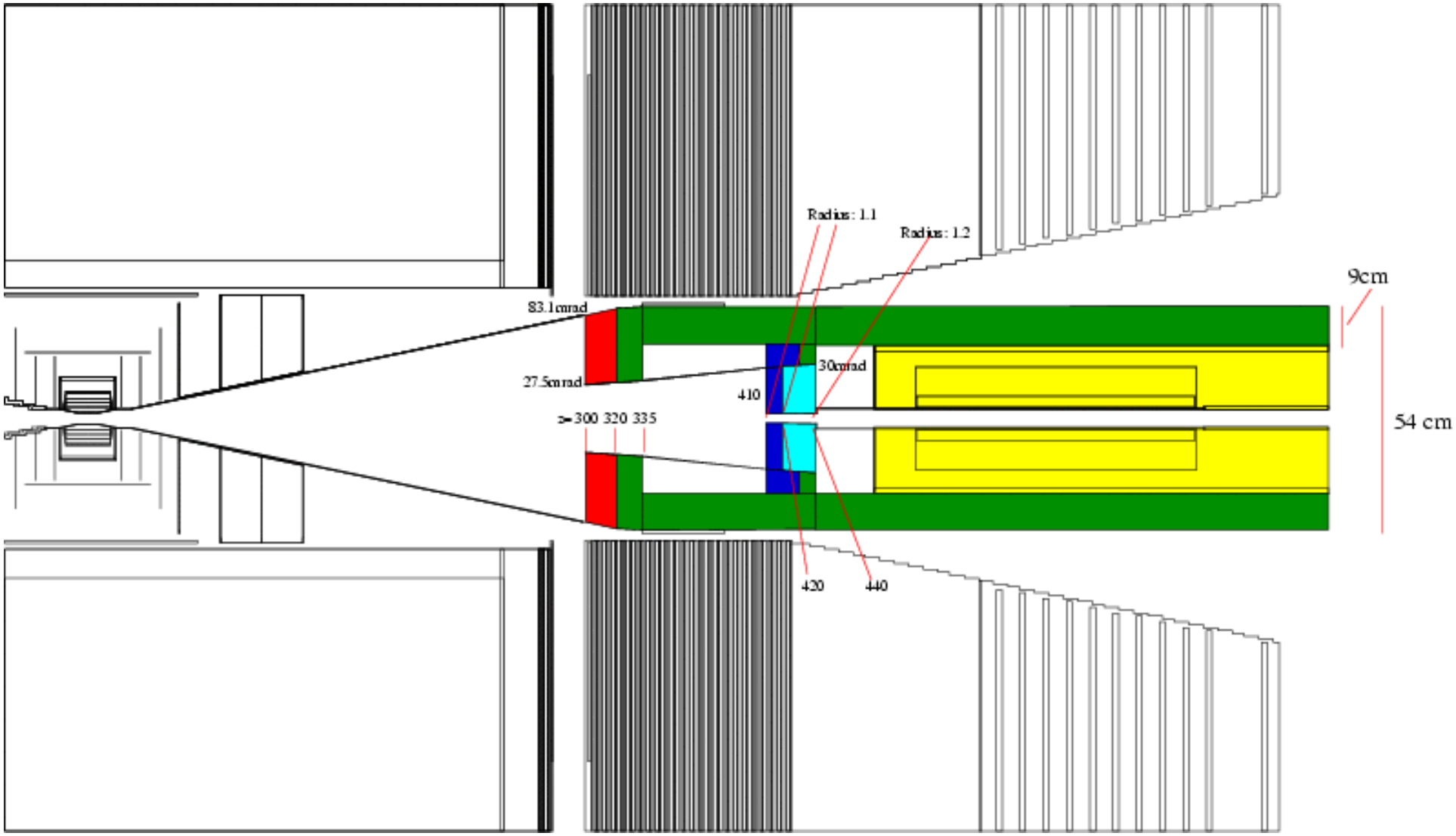


- All the tracking detectors (VXD, SIT, FTD and TPC) collect hits in a new tracking standard hit format
- The tube and inner tracking detector dimensions and materials come from the Brahms release 2.05 geometry.

Mokka - the SET

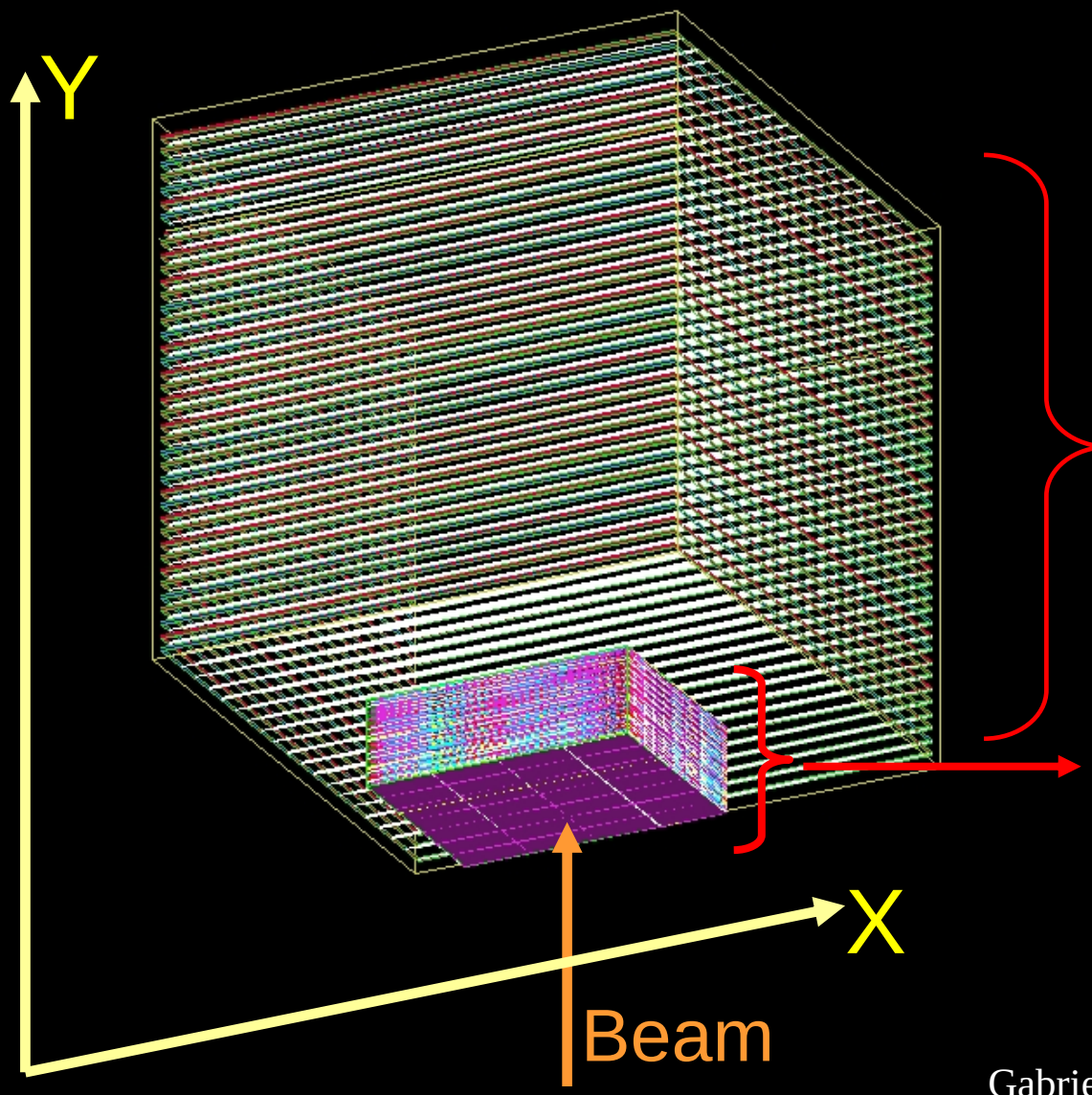


Mokka detectors - new mask



Karsten Buesser, January 2003

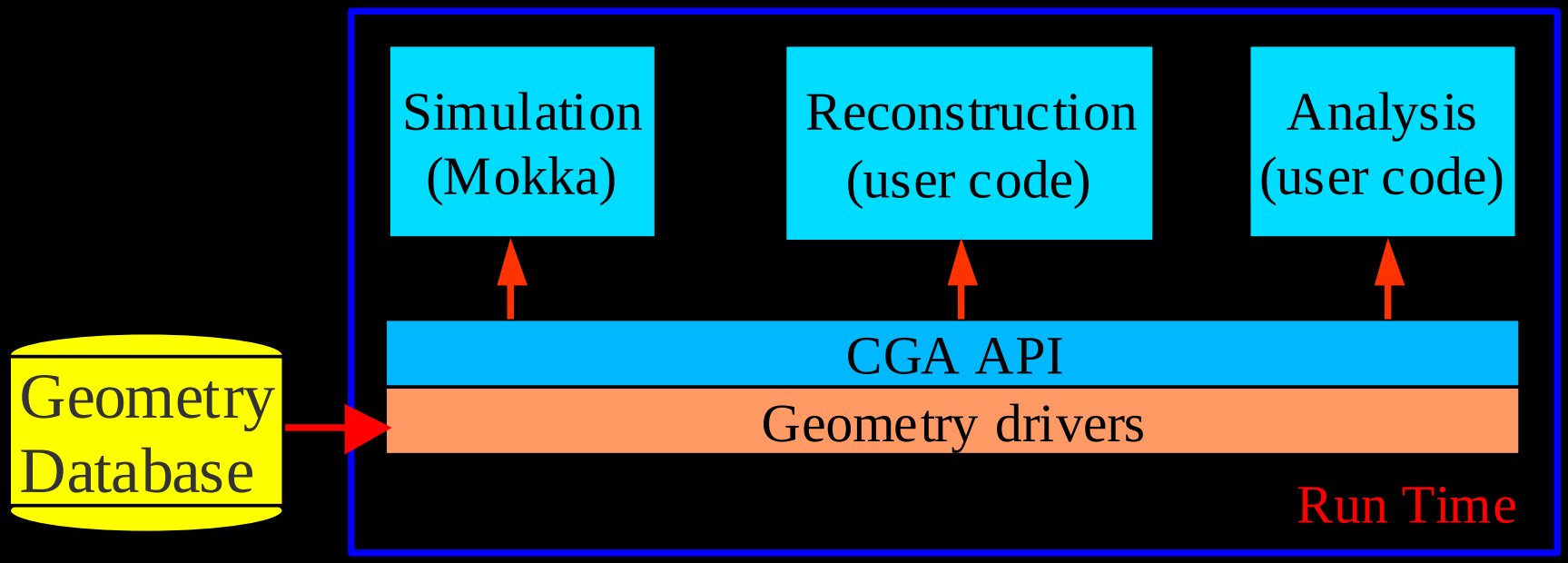
Mokka - the Prototype



Hcal prototype :
a Hcal module in a cube format (~1 m^3), built with RPCs

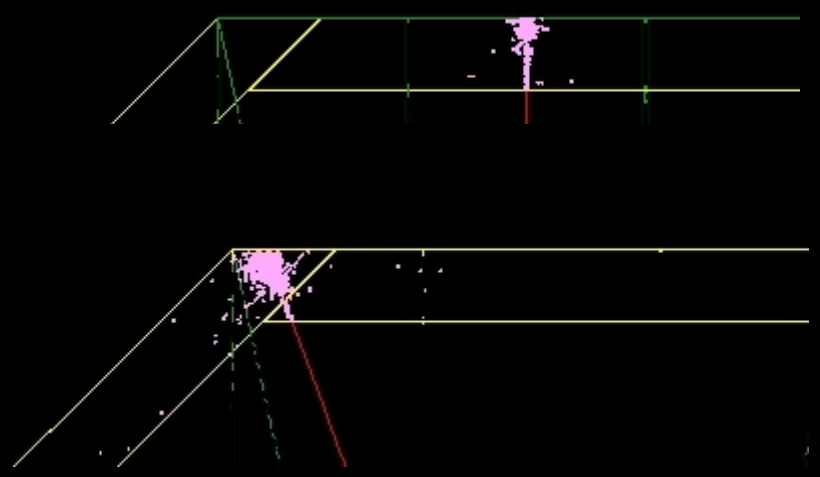
Ecal prototype:
old prototype already available in Mokka

Mokka - Common Geometry Access API (F77, C++,C, Java)



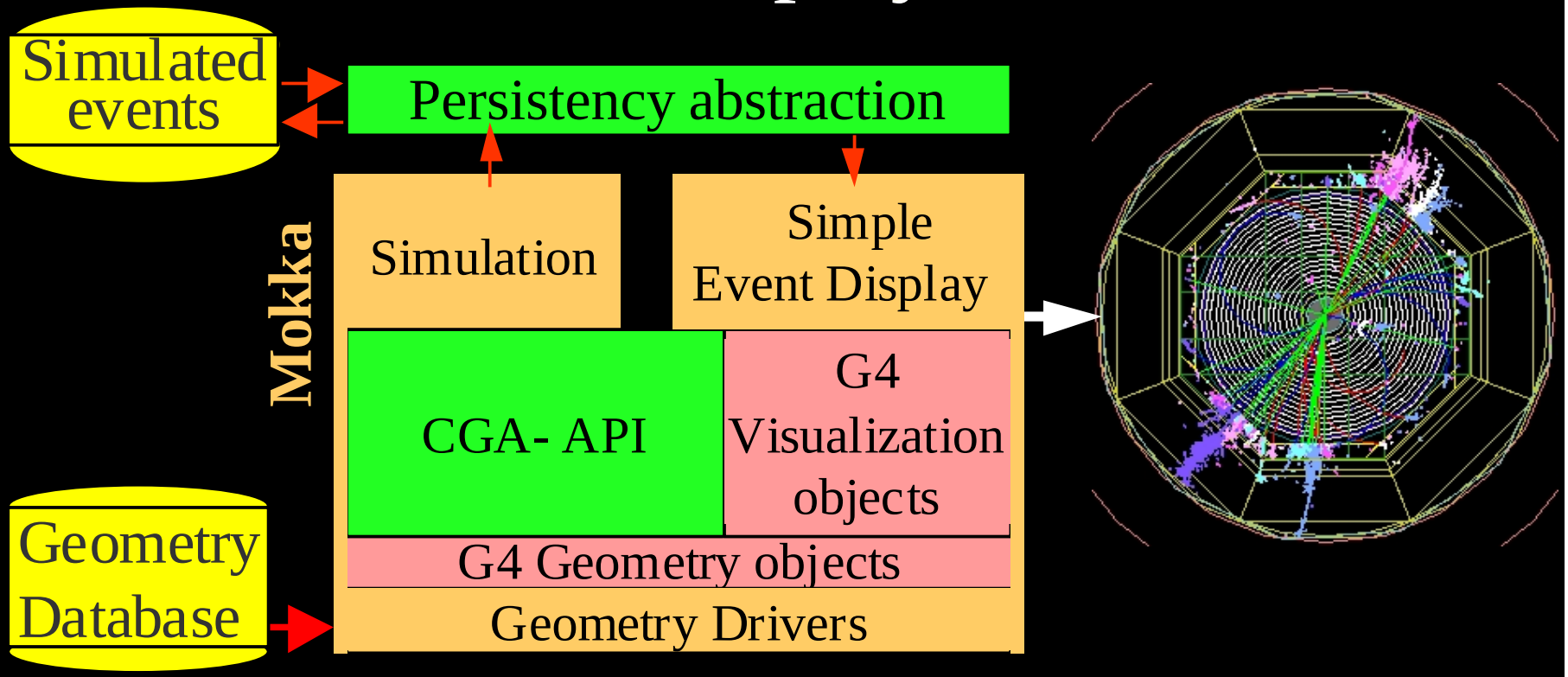
- Implements some reconstruction utilities.

The CGA API and the reconstruction



- Example, a 50 GeV pi in ECAL :
  - the total number of X0 from the entrance point in ECAL is required

Mokka - a (very) Simple Event Display



- Relies on the Geant4 standard visualisation.
- It's very simple but at least you see something…

For the collaboration on Mokka development:

* CVS repository on pollin1.in2p3.fr

* simplified of the detector-driver developpement procedure

* documentation included in the standard distribution of Mokka

News from the collaboration on Mokka development

* Aims:

- improving the models of different detector pieces

- detailed prototype simulation

- improving the framework

Prototype simulation

* a new implementation of TB by Jeremy MCCormick
  - GEM implementation of HCAL

* more detailed Ecal prototype implementation
  in progress

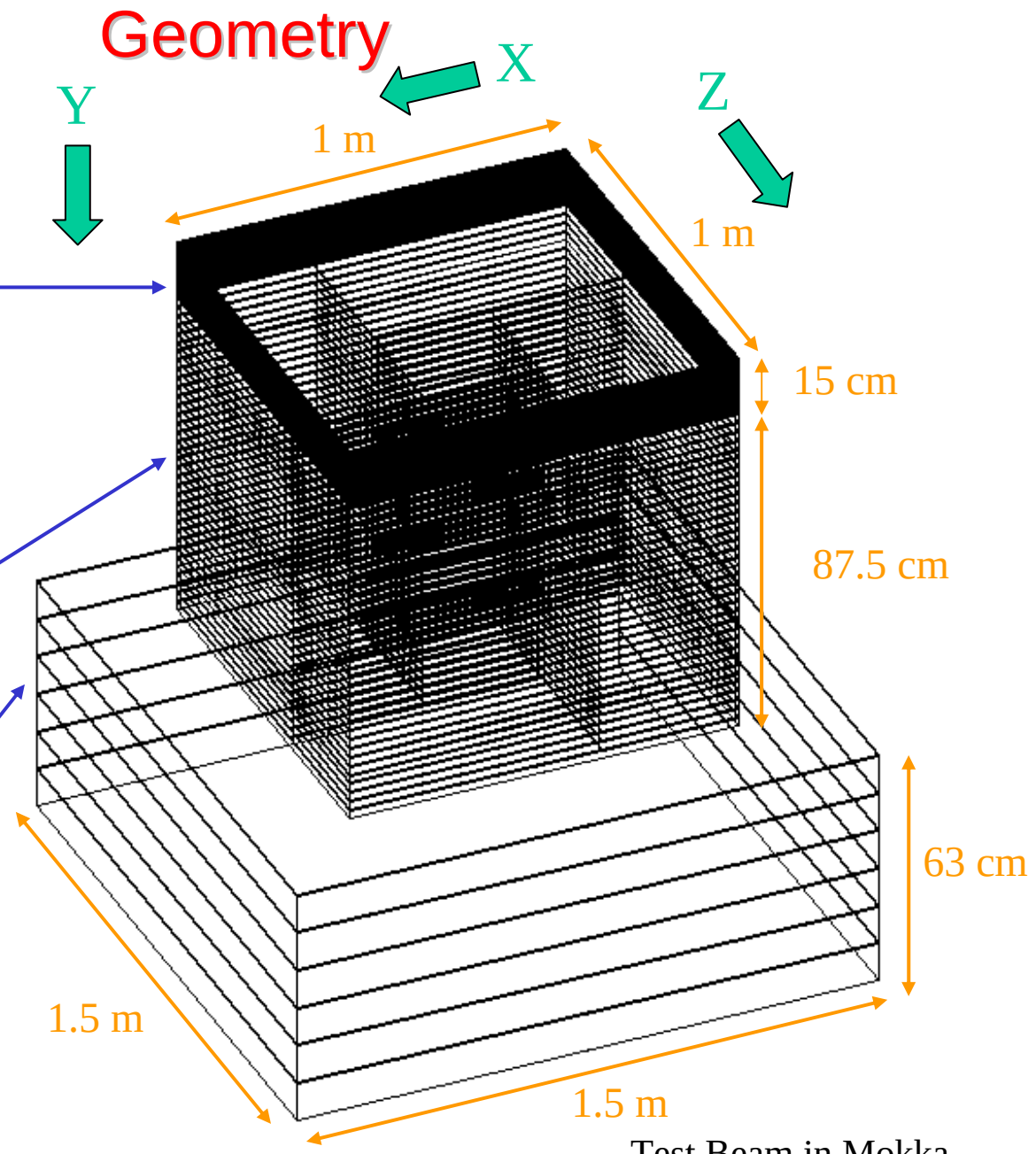
Geometry

Single layer widths in mm

ECal	
Tungsten	2.5
G10	0.5
Silicon	0.4
Copper	1
Air	0.6

HCal	
Polystyrene	5
Steel	25

Tail Catcher	
Polystyrene	5
Steel	100



Test Beam in Mokka

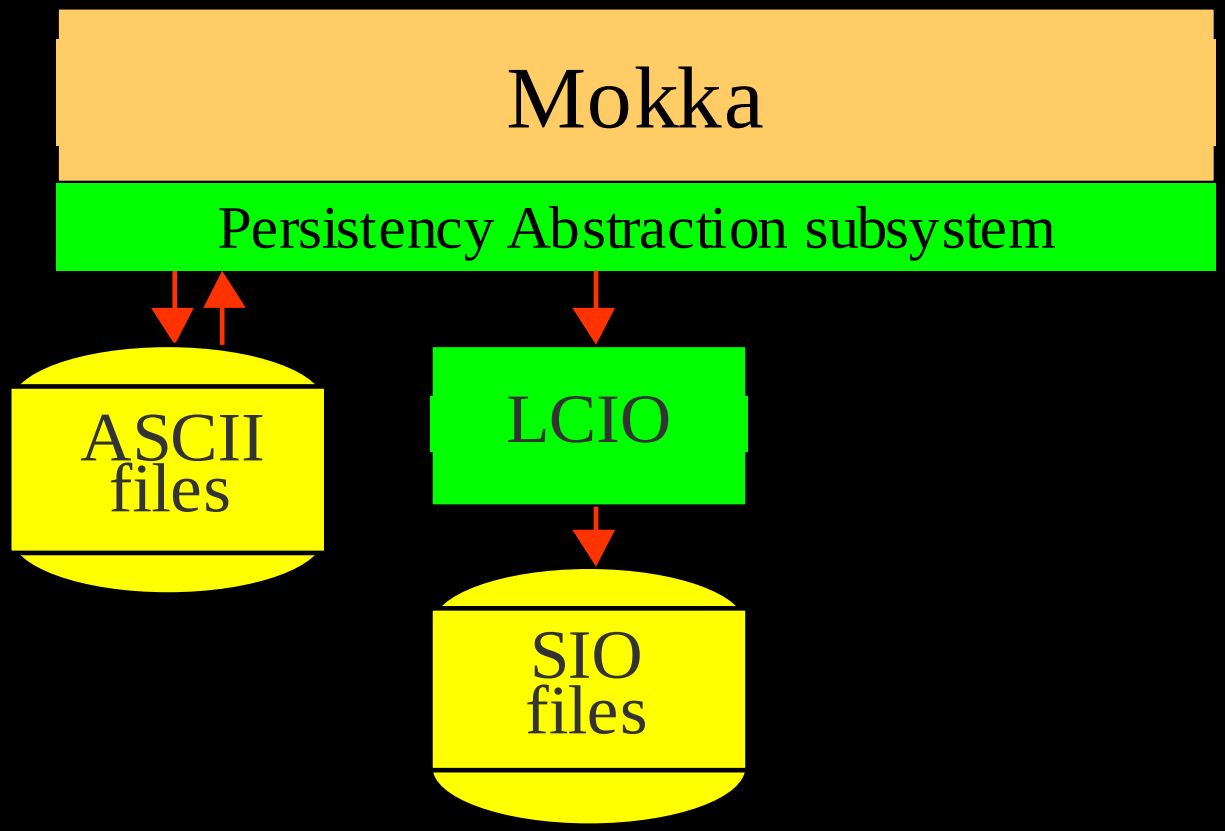
Jeremy McCormick/Nicadd

Improving the framework (I)

* Mokka output in both LCIO and ASCII format

* Keeping up-to-date with LCIO evolution

Mokka persistency model



Improving the framework (II)

* compression of the six Cal hit indices

P S M I J K X Y Z E PID PDG **CELLID FLAG**

in a single word in both ASCII files (CELLID) and LCIO files (CellID0 index of SIMCALORIMETERHIT)

The **FLAG** is used by the CGA interface to set the right sensitive detector that will provide the cell center coordinates.

In the LCIO files it is stored in the two user bytes of the collection flag.

Improving the framework (III)

* two new methods of the CGA interface that are used to decode cell center coordinates from the CellID word:

  - CGASetSD selects the sensitive detector corresponding to the flag of the LCIO collection or to the FLAG stored at the rightmost in the ASCII hit files

  - CGACellIndex uses the CellID0 of the LCIO hit or the CELLID from the ASCII file to calculate the cell center coordinates.

Improving the framework (IV):

* Recent work of Frank Gaede

  - steering-files

  - plugins

Steering files

* alternative to command-line options (still available)

* commands of the form: /Mokka/init

* initializing user-defined variables:

  /Mokka/init/userInitDouble MyCutEnergy 1000 KeV

* activating a G4 physics list:

  /Mokka/init/physicsListName QGSP_BERT

* usage: Mokka mokka.steer

Plugins

* for user analysis during simulation
* plugins have user action methods that are called
    - at begin/end of run or event
    - during stepping or tracking

* steering command:

    /Mokka/init/registerPlugin MyPlugin

Improving the framework: TODO list

* improve the Event Display
    - selection of detector pieces
    - selection of hits, tracks
    - 2D views
* defining materials in the DB
* improve the documentation
* etc...