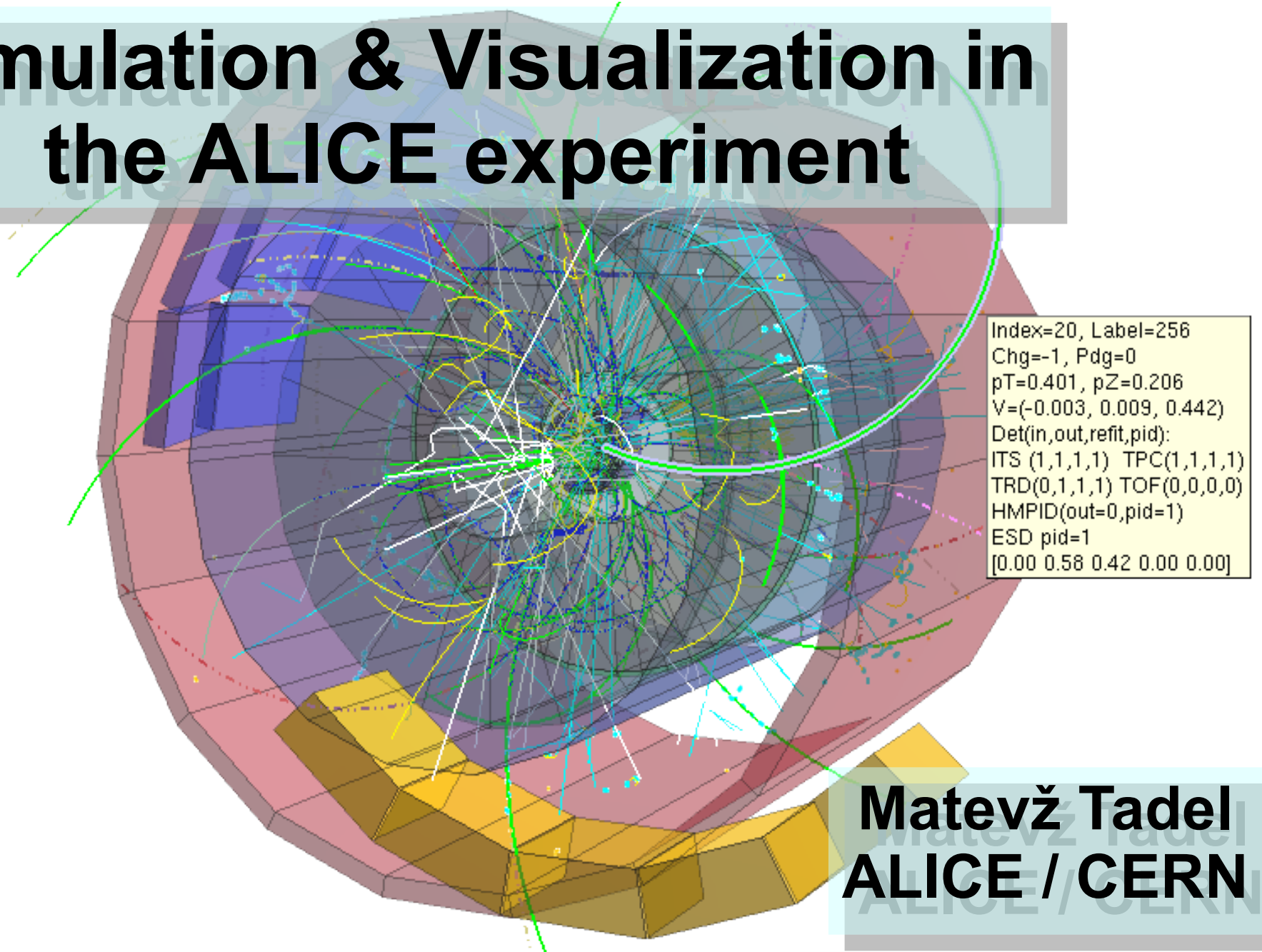
Simulation & Visualization in the ALICE experiment



Matevž Tadel
ALICE / CERN

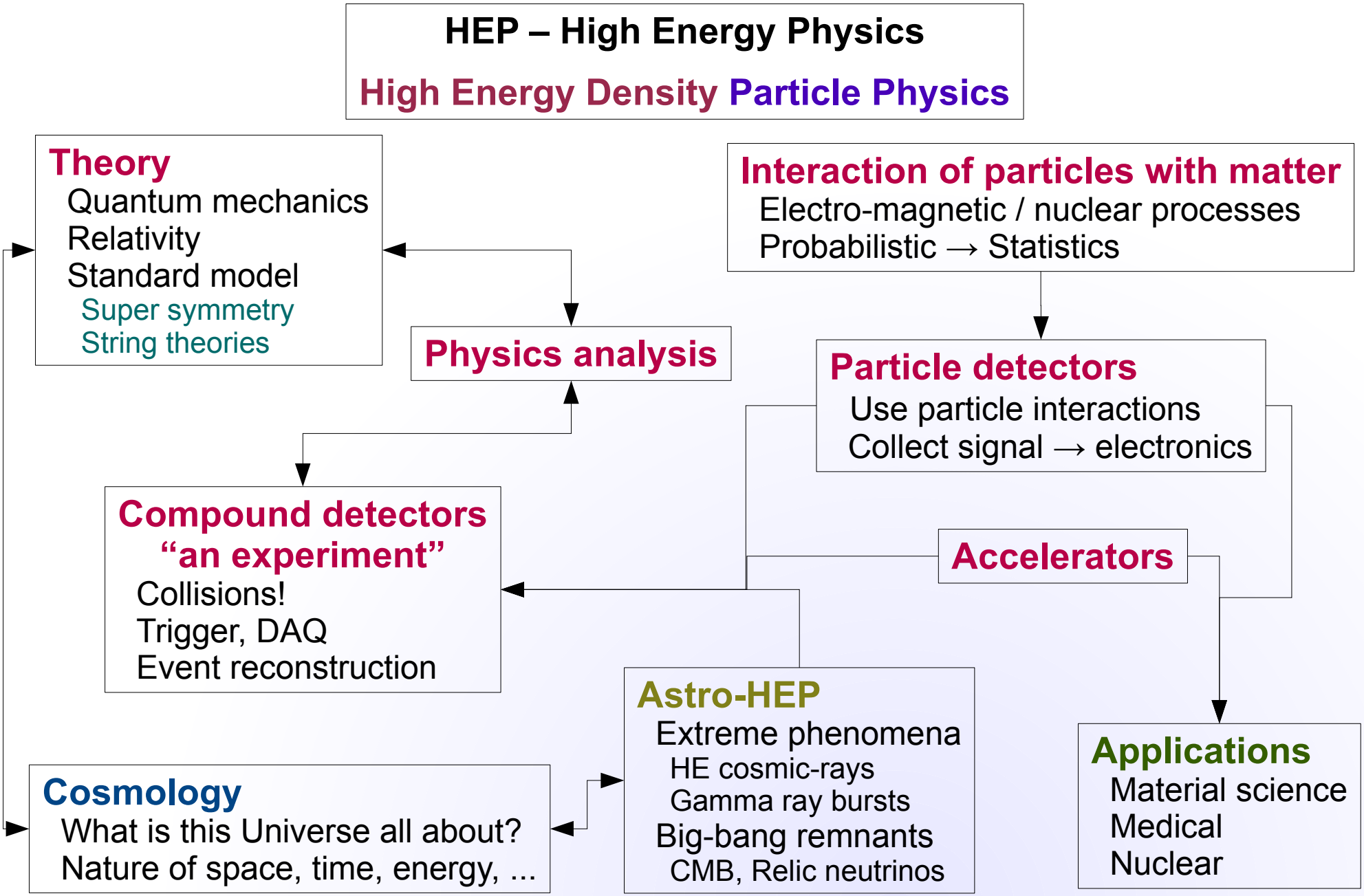
Overview

Expected audience: CS and some physics students around graduation.

- Introduction
  - HEP, role of Simulation and Visualization
- Simulation
  - What it means, detector geometry description, history.
  - Simulation in ALICE including minimal intro to ROOT and AliRoot.
- Visualization
  - Requirements for event-visualization.
  - Implementation in ROOT (EVE) and AliRoot (AliEve).
- Conclusion

Introduction

Down the Rabbit Hole


HEP – High Energy Physics
High Energy Density Particle Physics
Theory
Quantum mechanics
Relativity
Standard model
Super symmetry
String theories
Interaction of particles with matter
Electro-magnetic / nuclear processes
Probabilistic → Statistics
Physics analysis
Particle detectors
Use particle interactions
Collect signal → electronics
Compound detectors
"an experiment"
Collisions!
Trigger, DAQ
Event reconstruction
Accelerators
Astro-HEP
Extreme phenomena
HE cosmic-rays
Gamma ray bursts
Big-bang remnants
CMB, Relic neutrinos
Applications
Material science
Medical
Nuclear
Cosmology
What is this Universe all about?
Nature of space, time, energy, ...

Simulation

Theory
- Quantum mechanics
- Relativity
- Standard model
  - Super symmetry
  - String theories

Interaction of particles with matter
- Electro-magnetic / nuclear processes
- Probabilistic → Statistics

Physics analysis

Particle detectors
- Use particle interactions
- Collect signal → electronics

Compound detectors
"an experiment"
- Collisions!
- Trigger, DAQ
- Event reconstruction

- Design of detectors and experiments
- Operation of detector
  - Understand how detector behaves
  - Calibration, alignment
- Input for analysis
  - Understand what detector sees
  - Estimate background
  - Acceptances
  - Trigger efficiencies

Visualization

Theory
- Quantum mechanics
- Relativity
- Standard model
  - Super symmetry
  - String theories

Interaction of particles with matter
- Electro-magnetic / nuclear processes
- Probabilistic → Statistics

Physics analysis

Particle detectors
- Use particle interactions
- Collect signal → electronics

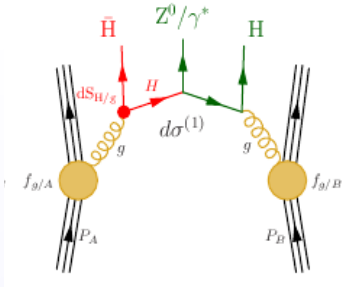
Compound detectors
"an experiment"
- Collisions!
- Trigger, DAQ
- Event reconstruction

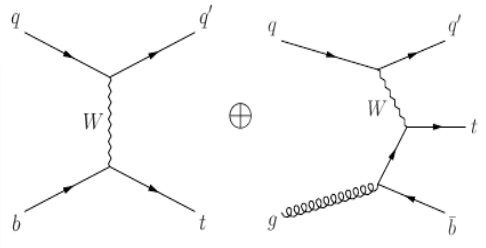
- Display graphs, histograms, correlations
- Detector structure visualization
- Understand:
  - detector operation
  - reconstruction algorithms
  - manifestations of physics processes
- Presentations, publications, outreach

Simulation

Simulation – What does it mean

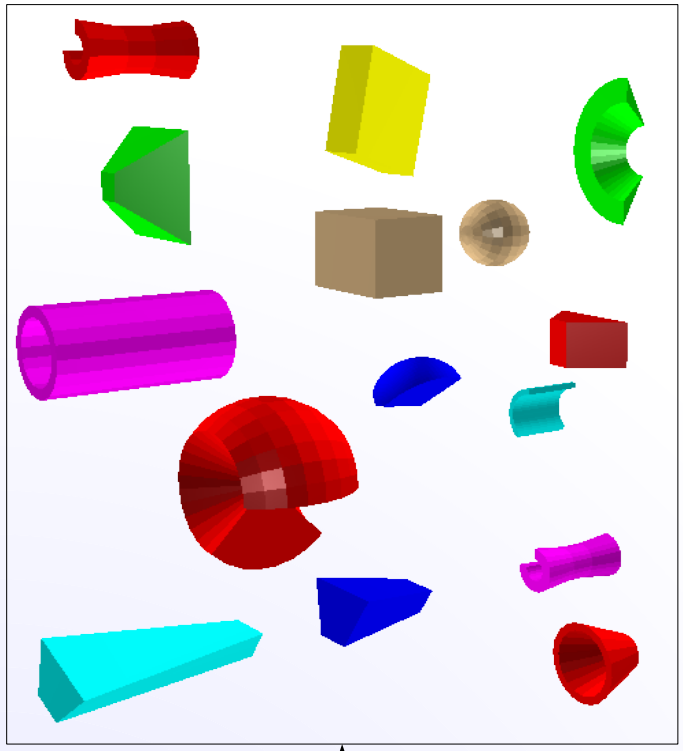
1. Event / hard process simulation
   - Incoming beam
   - QED / QCD processes
   - outgoing "**primary**" particles





2. **Particle transport**
   - propagate particles through the detector
   - simulate material interaction, creation of new particles
   - deposition of energy in the detector → **"hits"**
3. Detector / electronic response simulation
   - signal collection – electrons or photons
   - electronic response + noise → **"digits"**

Simulation – Detector Geometry

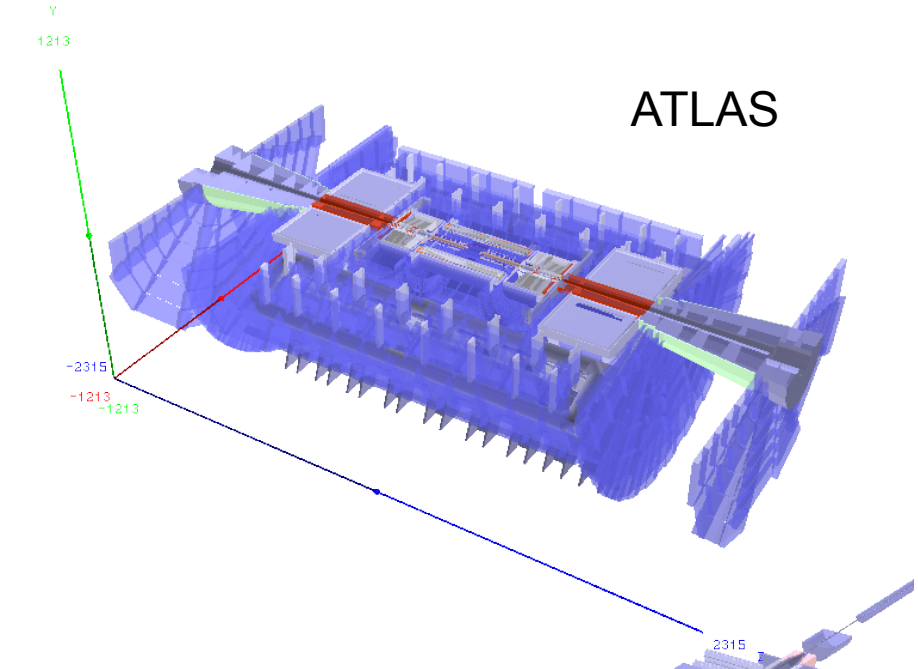


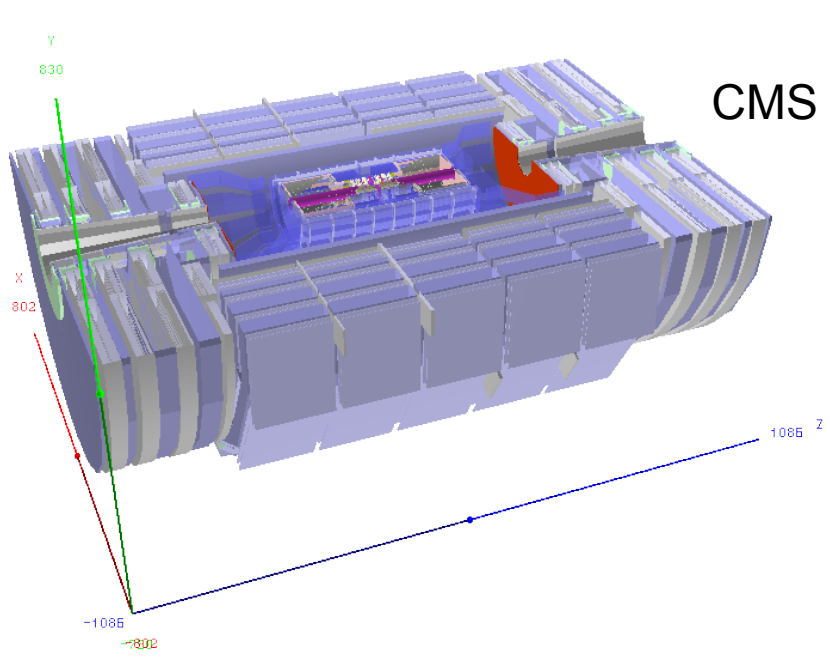
Demo:

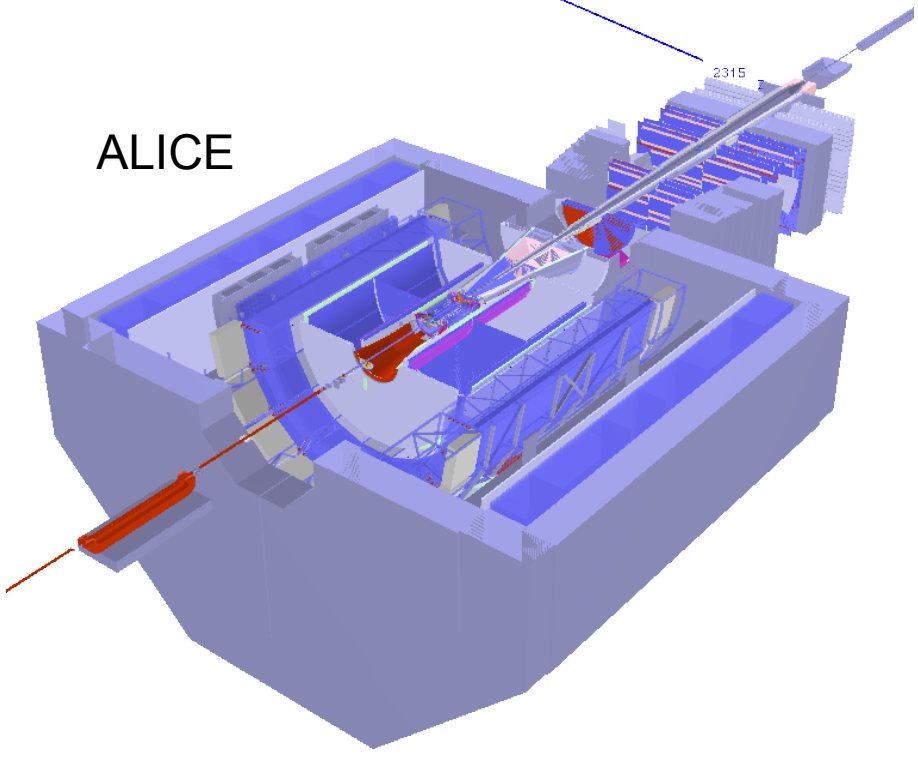
- CMS, ALICE – full geometry
- ALICE – ITS
  - full ITS, SSD detail
  - materials used, sensitive volumes

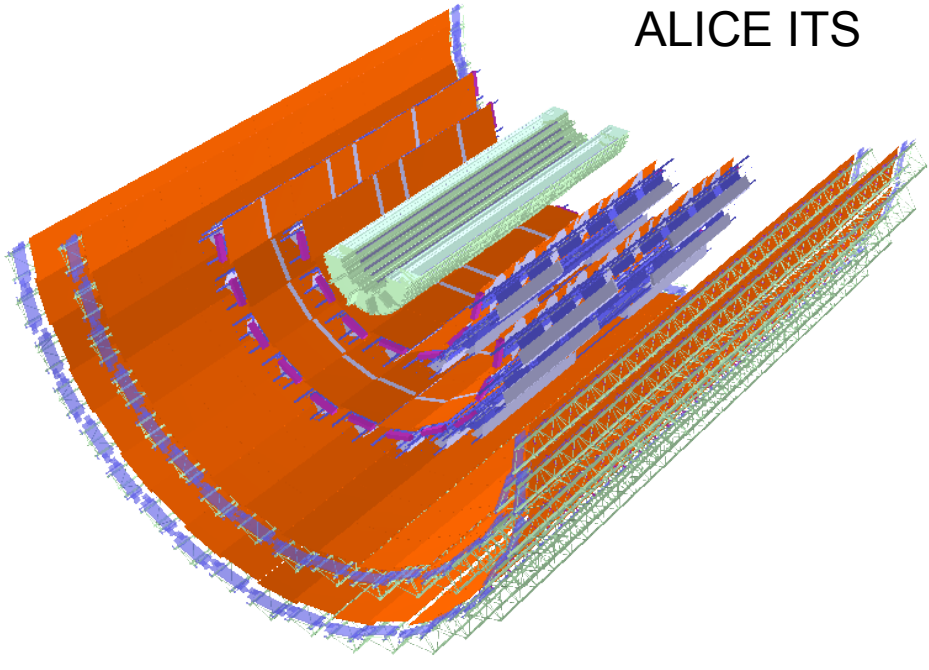
General remarks:

- Hierarchy, replication
- LHC detectors have a couple million volumes!
- Composed of basic shapes (~20)
- CSG supported by modern engines

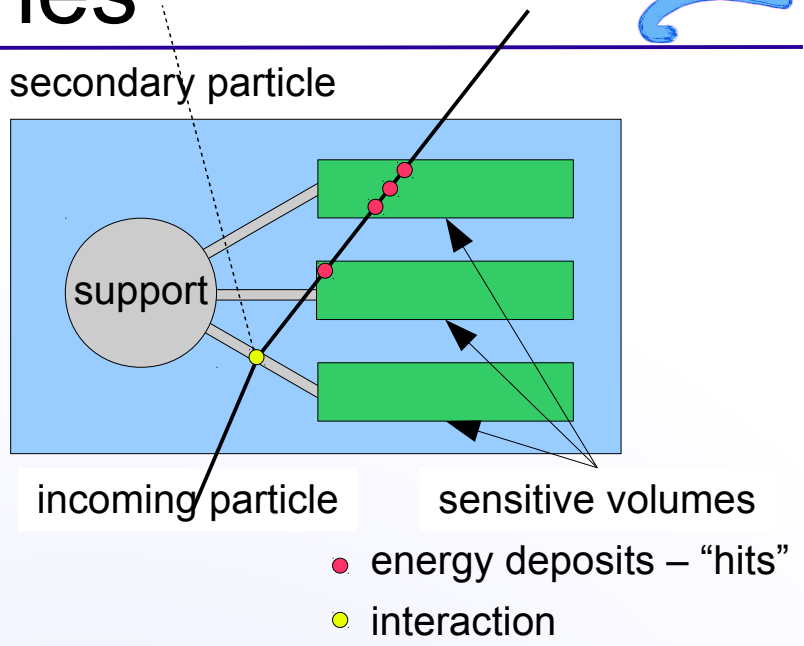

ATLAS
Y
1213
-2315
-1213
-1213
2315


CMS
Y
830
X
802
1085
Z
-1085


ALICE


ALICE ITS

Simulation – Transport engines



1. Geometry, magnetic field
2. Particle propagation / steering
   photons, leptons, barions, ions
3. Catalog of physics processes

From keV → TeV !

- in flight decay (unstable particles)
- EM: ionization, Compton scattering, photo-electric effect, bremsstrahlung, delta-ray emission, pair production / annihilation, photon conversion, ...
- hadronic / nuclear: (in)elastic scattering, capture, excitation, fission, ...
- special: Čerenkov photons, low-energy neutrons, polarized leptons / photons, ...

Processes need to be properly represented statistically!

All of them "compete" on every propagation step.

Simulation – History, before you were born

Before '75: EGS – Electron Gamma Shower. Still developed at SLAC.

'75: GEANT1 – basic EM processes

No geometry – called user code to get material, distance to boundary.

That code was a forest of if statements.

'78: GEANT2 – new EM processes, trivial hadronic processes

Used by fixed target experiments: NA3, NA4, NA10, Omega.

In '80 – data-cards (files) for geometry description.

'81: GEANT3 – Use API to define geometry and materials.

Memory management via ZEBRA (merge of ZBOOK, HYDRA and BOS)

Excellent EM processes – import EGS (with its main developer).

Several hadronic packages: TATINA, GEISHA, FLUKA.

Used by practically all HEP experiments until 2000 (including LEP and LHC)

Simulation – History, recent developments

'90+ FLUKA taken out from GEANT3, all processes written from scratch

- Geometry from MARS – flat (no hierarchy, replication), input via data-cards
- Statistically correct – allows weighting of particles to improve coverage
- Excellent physics for all energies including low-energy neutrons and ions
- Used for accelerator design, radiation studies, medical research.
- Was used by ALICE and FAIR (GSI) – licensing issues – future unclear.

'95 GEANT4 development starts

- In essence, complete rewrite of GEANT3 into C++:
  - extensions of geometry package (CSG, NURBS)
  - improvements to EM processes (multiple scattering, ionization to 1keV, polarized EM)
  - several hadronic packages; includes low-energy neutrons
- Used by all recent HEP experiments (excluding Tevatron, RHIC, Belle).
- Medical research – GATE package (emission tomography), hadron therapy.
- Space research – ESA: shielding, radiation monitoring, effects on electronics.

Simulation – History – Remarks

Development effort:

EGS – 15 manYears

GEANT3 – 30 mY (without hadronic packages)

GEANT4 – 150 mY, ~30mY core; still taking 30mY/year

Other packages on the market:

CORSIKA – high-energy cosmic-ray air showers

MARS – accelerator design, radiation studies

MCNP – flux oriented, excellent neutron physics;
nuclear bomb calculations

Simulation – ALICE – demo

Simulated pp events at 14TeV

- small event: [0]
  - Primary particles
  - $K^+ \rightarrow$ decay
  - $\pi^+ \rightarrow$ nuclear interaction
  - $\gamma \rightarrow$ mini EM shower
- **big** event: [8]
  - momentum selection → most particles have low momenta
    - forward region → polluted by beam remnants
    - transverse region → observe “clean” physics
  - ITS / TPC hits → central barrel detectors of ALICE
    - discrete .vs. continuous tracking

ALICE offline computing

Online / DAQ

Transport engines
- Geant3
- Geant4
- Fluka

VMC

Grid → AliEn
- storage management
- file catalog
- conditions DB
  - calibration
  - alignment
- job management

AliRoot
- Event generation
- Simulation / Digitization
- Reconstruction
- Analysis

ROOT
- The **only** external dependency of AliRoot!
  Completely unlike other experiments.
- If something is missing … it is added :)

ROOT

ROOT – the ultimate OO meta-framework for HEP.

It allows efficient handling and analysis of **LARGE** data-sets.

- introspection, C++ interpreter, object serialization;
- containers, advanced OO ntuples;
- OS interface: files (also remote), network, threads, GUI;
- graphics:
  - data analysis → graphs, histograms
  - 3D graphics → event / data visualuzalization
- mathematical libraries:
  - special functions, FFT, statistical modeling, multi-variate analysis, ...

At the end of ends all LHC experiments use ROOT for **data-storage** (~ 2 PByte / year / experiment !) and **data-analysis**.

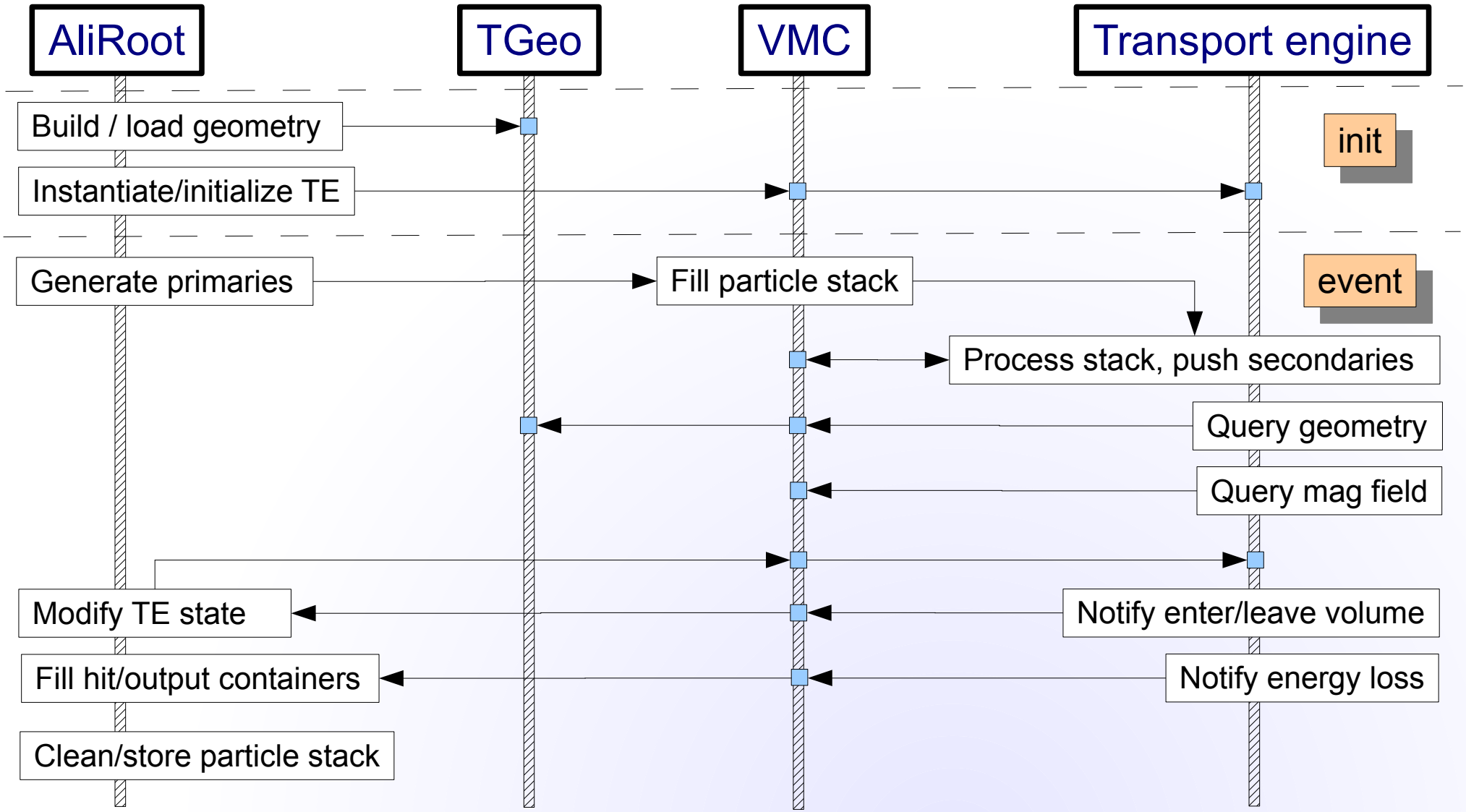
ROOT geometry package – **TGeo**

Features:

- all standard shapes
- volume, subbranch instancing
- CSG shapes and assemblies
- materials, tracking properties (cuts, process selection)
- supports navigation and boundary queries
- alignment support
- overlap detection
- can be serialized in a ROOT file (ALICE ~ 1.2 MByte)

Developed by ALICE in collaboration with ROOT.

Simulation – ALICE, Virtual Monte Carlo



Simulation – ALICE – Benefits of VMC

Benefits for experiment:

1. Use any supported transport code!
2. Shared hit processing / output management code.
3. Compare results directly – from physics perspective.

With using TGeo:

1. Same geometry for all of them – no conversions.
2. Reuse the geometry for reconstruction and visualization.

Transport code becomes physics engine only!

Does not have to care about:

- Geometry.
- Output of particle stack / hits.

Simulation – Notes for other LHC experiments

All:

- used GEANT3 until ~'04, now only use GEANT4
- have “home-grown” geometry description
- produce in-memory representation to:
  - apply misalignment
  - feed Geant4
  - extract subset of geometry needed for reconstruction

ATLAS: GeoModel (C++) → execute & build → Geant4

CMS: DDD (XML) → parse & build → Geant4

LHCb: DDDB (XML) → parse & build → Geant4

Visualization

Event Visualization – What is it?

"**Visualization of**" and "**GUI to**":

- detector geometry
- event data:
  - simulation records: kinematics, hits, digits
  - raw data
  - reconstructed objects: clusters, tracks, V0's ...
  - physics objects: b-tags, Z^0/H-candidates, ...
- reconstruction & analysis algorithms
- calibration and alignment data

To be used:

- **by experts:** visual debugging, development of algorithms
- **by non-experts:** understand detector, event structure, ...
- for presentations, demonstrations ... outreach

Visualization in ALICE: ROOT – EVE – AliEve

Too many elements, too many use-cases for one application!

Our solution: (in 2005)

1. Build an extendible framework, not a monolithic application, call it:
   EVE – **Event Visualization Environment**
2. Use and extend ROOT functionality: GUI, OpenGL for 3D graphics
3. Use ROOT philosophy: build a modular, loosely coupled class toolbox
4. Provide new elements as needed:
   i. Put basic / low-level development back into ROOT
   ii. Build composite / top-level elements from those – but still keep them general
   iii. Put only ALICE specific code into AliRoot

And this worked well:

- EVE is now used by ALICE, CMS, FAIR, K2K, NA62, 2 medical projects
- considered by ILC experiments

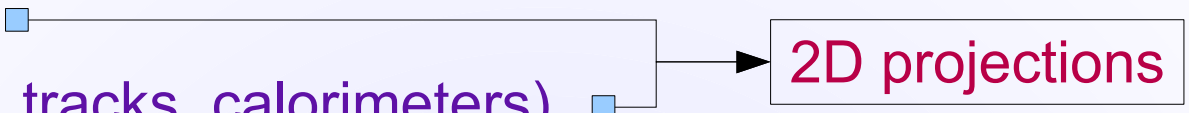
Visualization – EVE overview

Framework for object management:

- hierarchy / structure
- interaction, visualization → GUI / OpenGL (2D/3D)
- creation and coherency of 2D projections
- object selection, highlight

Toolkit satisfying HEP requirements:

- geometry – TGeo
- sim/rec data (points, tracks, calorimeters)
- raw-data visualization
- overlays, view markup
- window management



Visualization – EVE classes I.

Geometry:

Direct usage via TGeoPainter: requires geometry

Extracted shape-data: standalone, fully configurable

Hits, clusters:

TEvePointSet: per-point TRef (optionally owned by the object)

TEvePointSetArray: an array of point-sets - interactive histogram

Select on external criteria provided during filling.

Trajectories, particles, tracks:

TEveTrack: supports extrapolation in arbitrary magnetic field

Can specify position/momentum at:

- arbitrary reference points (enter/leave certain volume)
- points of daughter creation (with daughter momentum) and decay
- 2D

TEveTrackList – an array of tracks, allows drawing style changes for all

Interactive selection on standard track parameters: pT, p, Chi^2

Visualization – EVE classes II.

Digits, raw-data:

TEveQuadSet – set of 2D rectangles, lines or hexagons

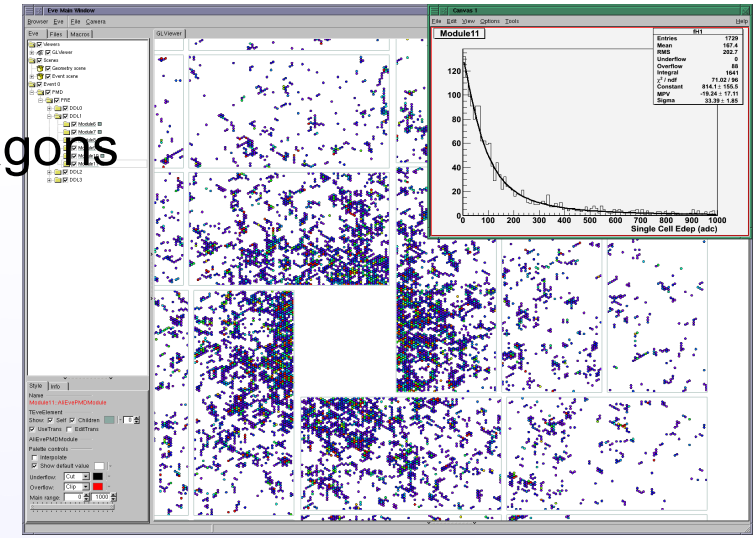
Used for silicon detectors, TRD, TOF, PMD, …

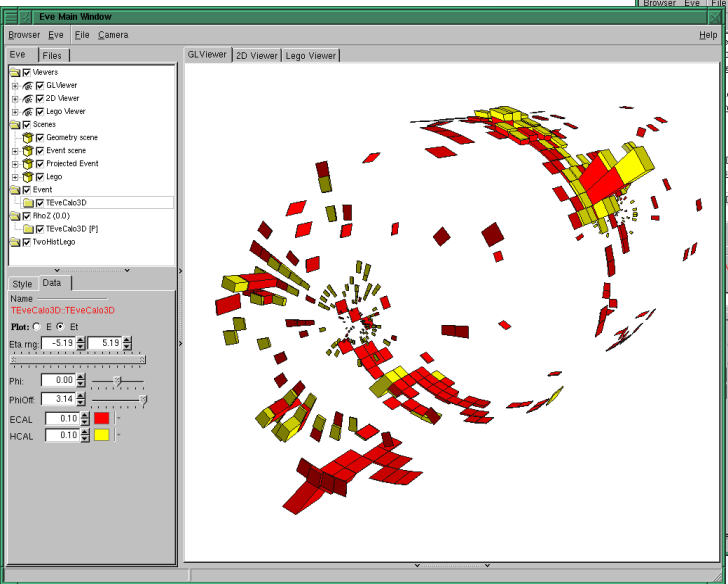
TEveBoxSet – set of 3D boxes

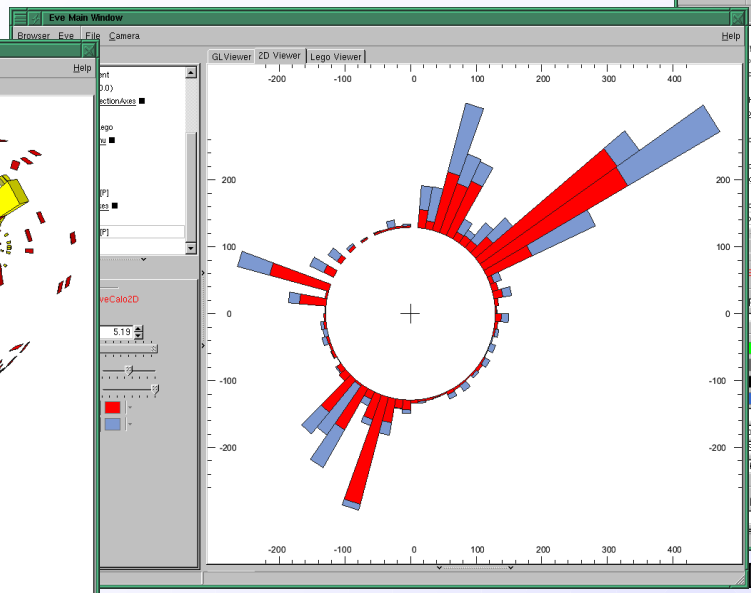
Calorimeters:

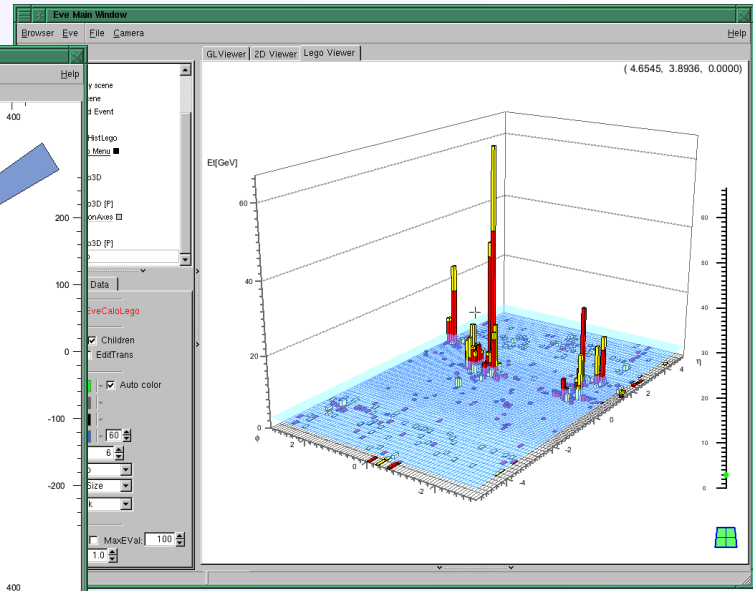
TEveCaloData – 2D histogram stack or list of cells

TEveCaloViz – 3D, projections, lego









Visualization – AliEve structure

1. **Core classes**
   - **Data & event managers** – connect to ALICE data files
     event selection
     macro managements – what / how to show on every event
   - Custom viz. classes – tracks, kinks, V0s, cascades, HF
   - Custom tools – kinematics helper, track counter, track fitters
2. **Raw-data visualization**
   Specialized classes to read / display raw-data.
   Some detectors use TEveQuad/BoxSet objects directly.
3. **CINT macros**
   - Obtain event-data handles from AliEveEventManager.
   - Create EVE objects.
   - Register them to EVE scenes.

Visualization – AliEve Demo

Run 104799, 11.Dec.2009, pp@900GeV [chunk 019.10]

- run / event info
- event selection – number of tracks
- kinks, V0s
- TPC raw-data [ev 307]
  - sectors, pads
  - time selection
  - time histogram from individual pad

Visualization – Other LHC experiments

ATLAS:

- **Atlantis** – Java (was Fortran); XML input; supports 2D projections
- **Persint** – Qt + OpenGL; custom input file; specialized for Muon spectrometer
- **VP1** – Qt + Coin3D; integrated

CMS:

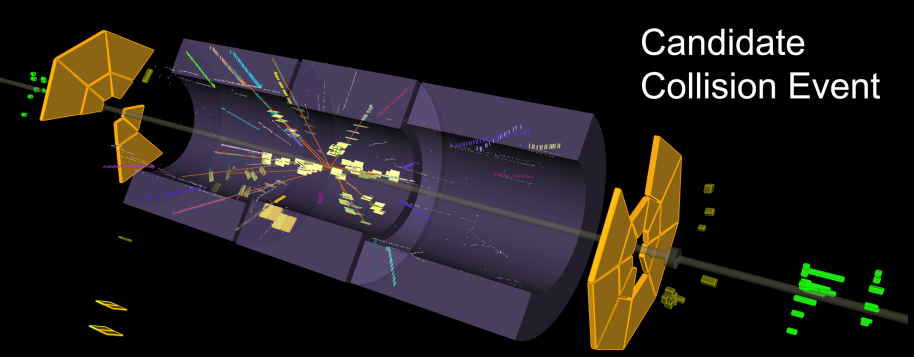
- **Iguana** – Qt + Coin3D; integrated
- **iSpy** – Qt + Coin3D; custom input file
- **Fireworks** – ROOT/EVE; integrated with "light framework"

  Being extended to include / provide required features of Iguana / iSpy

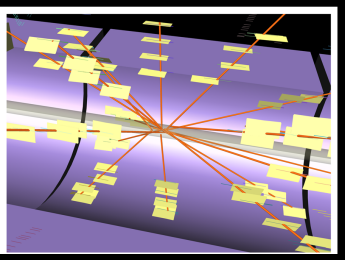
LHCb:

- **Panoramix** – Qt + Coin3D; inversely integrated; 3D views only

  Python interface / scripts for operation and configuration



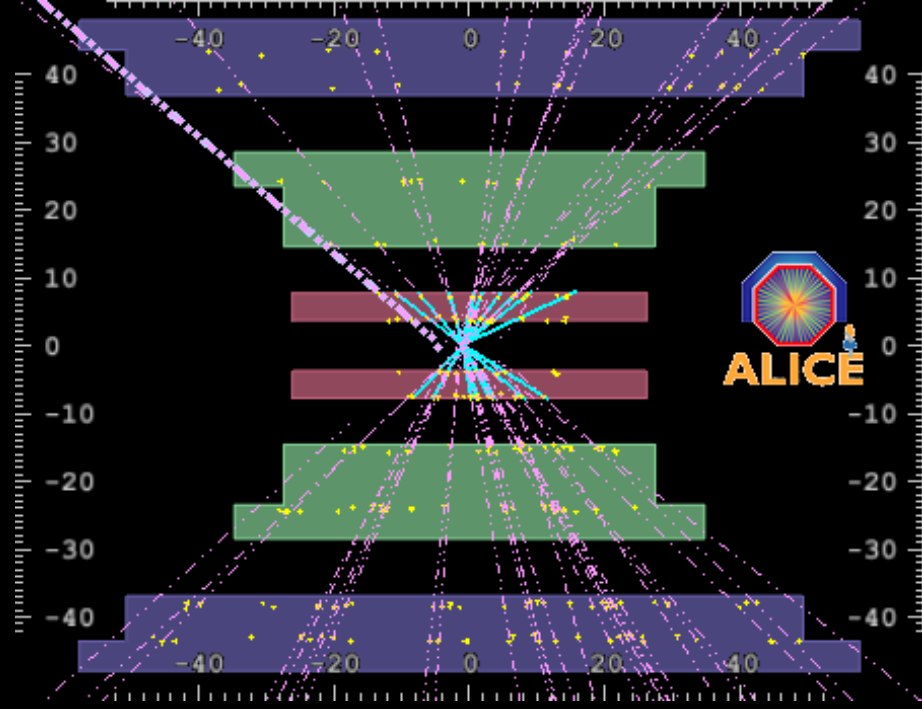
Candidate
Collision Event

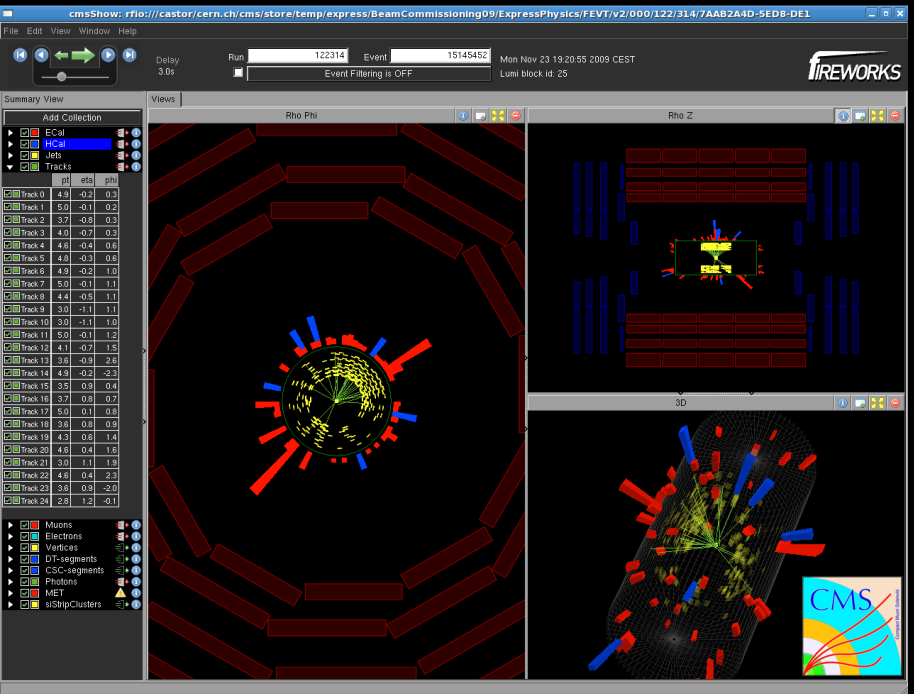


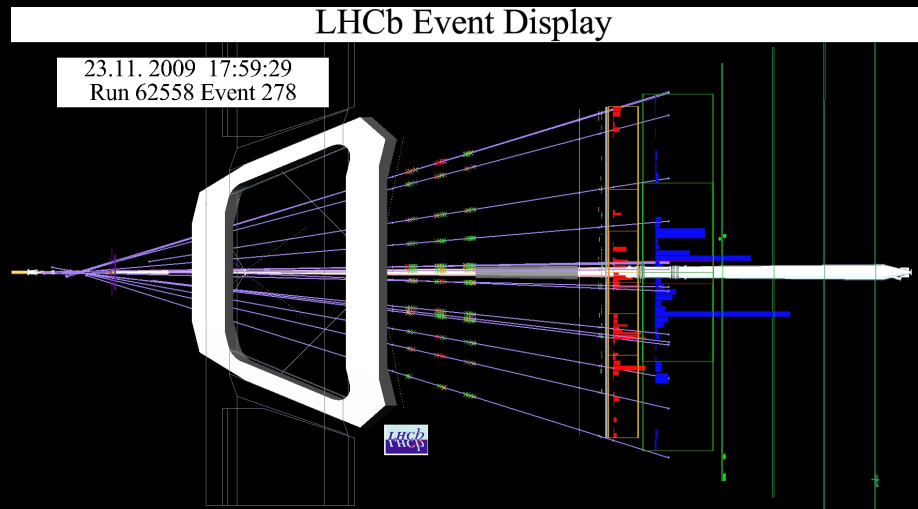
ATLAS
EXPERIMENT

2009-11-23, 14:22 CET
Run 140541, Event 171897

http://atlas.web.cern.ch/Atlas/public/EVTDISPLAY/events.html







Conclusion

This was really  **just a low fly-by**

over simulation and visualization.

So I hope you don't feel like this guy.

Try it yourself!

- Download ROOT: http://root.cern.ch/

  See the instructions on the download page,

  set the environment.
- cd root/tutorials/eve
- Run geometry demos:

  root geom_atlas.C

  root geom_cms.C
- Run ALICE event demos:

  root alice_vsd.C
- EVE documentation: http://root.cern.ch/root/htmldoc/GRAF3D_EVE_Index.html

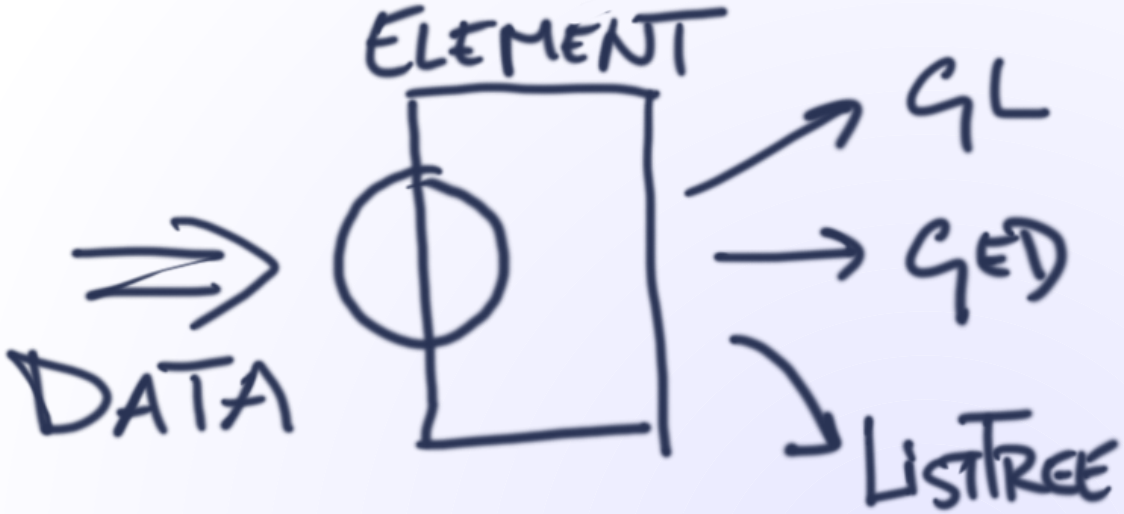
Have fun!

Details

Visualization – EVE base-class I.

Base-class **TEveElement** holds together:

- visualization data – optional back-references
  - use existing EVE-classes or sub-class for full control
- GL-renderer
- GED (object editor)
- List-tree entries /browsable representations



Visualization – EVE base-class II.

Hierarchy – each element knows its:

- **Children** – `list<TEveElement*>`
  - Rendering control
  - Collection management
    - Sub-classes can override Add/RemoveElements()
    - Control state of children (e.g. TEveTrackList p_T selection)
- **Parents –** also `list<TEveElement*>`!
  - Propagation of change / redraw requests
  - Multiple control agents can share an element:
    - Scenes, Selection, ...
  - Used for reference-counting.