Mantis

the Geant4-based simulation specialization of the CMS COBRA framework

M. Stavrianakou, *FNAL, Chicago, USA*, P. Arce, *CIEMAT, Madrid, Spain*,
S. Banerjee, *TIFR, Bombay, India*, T. Boccali, *SNS, Pisa, Italy,*
A.De Roeck, V. Innocente, M. Liendl, *CERN, Geneva, Switzerland*,
T. Todorov, *IReS, Strasbourg, France*

MANTIS: Modular **A**rchitecture '**N** **T**oolkit **I**n **S**imulation
Mantis = Greek work for oracle (as the *prophet* in an oracle)

Overview

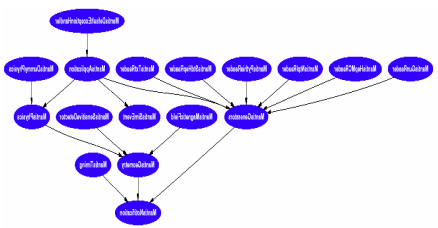
- **Mantis** = CMS Geant4-based Simulation Framework = specialization of the **COBRA** framework, which implements the CMS OO architecture; basis for the CMS-specific simulation program OSCAR
- provides the infrastructure for the selection, configuration and tuning of all essential simulation elements: geometry construction, sensitive detector and magnetic field management, event generation and Monte Carlo truth, physics, particle propagation and tracking, run and event management, and user monitoring actions
- experimental set-up built by Mantis using COBRA Detector Description Database, **DDD**, which allows transparent instantiation of any layout (full or partial CMS simulation, test beam set-ups etc)
- persistency, histogramming and other important services available using standard COBRA infrastructure; transparent to user applications

Requirements and Design

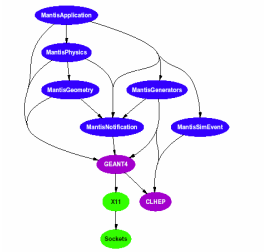
- architecture-driven approach in CMS COBRA framework
- modular design, maximal synergy with COBRA, minimal assumptions and constraints as to requirements of specific CMS simulation implementations
  - extensibility and configurability: abstract interfaces allow implementation of new modules for all major simulation components (geometry modeling, generators, physics, user actions etc)
  - minimal need for coding, compilation and linking on user side, as well as run-time flexibility and configurability
  - development, testing and debugging of individual components without unnecessary overheads; e.g. geometry modeling and visualisation decoupled from physics simulation and event processing
  - maximal synergy with other CMS software products and tools for visualization (IGUANACMS), dependency analysis (IgNominy), profiling (IgProf), testing (OVAL) etc
- support for simulation productions: compatibility with CMS infrastructure to ensure minimal deployment effort, incl. specific features such as
  - metadata initialization
  - capability to suspend and resume interrupted run (esp. important for Condor farms)

Mantis dependencies

Mantis system



MantisApplication



Metrics summary

Packages/Levels =17/5
CCD=52.0, ACD=3.0
NCCD=0.9

Application and Run management

- relies on COBRA for program's *main;* COBRA takes over and handles the application; the core of the program consists of a *SimApplication* and a *SimEvent*
- the *SimApplication* instantiates and launches an event source factory the *SimEventSource*, a COBRA abstract factory - statically built singleton – which is an abstract reader of simulated events from the original source
- the *SimEventSource* instantiates the Mantis *RunManager*
- the *RunManager* instantiates a *G4RunManagerKernel* and controls standard components such as the selection and instantiation of generator, magnetic field and physics lists and interfaces to the run, event, stacking, tracking and stepping actions; it also handles
  - storage and retrieval of random number seeds for the run and events
    - facilitates debugging of rare crashes in time-consuming physics events
  - the storage and retrieval of the cross-section tables built for a given detector configuration and physics list
    - reading pre-built cross-section tables reduces initialization time by factor ~4
- the *SimEvent*, inheriting from the COBRA *SimEvent*, manages the Monte Carlo truth – common for and sharable by all CMS applications (reconstruction, visualization etc); it consists of an interface to the transient event. (be it Geant3, Geant4, fast simulation) and an interface to a Geant4 event
  - the Monte-Carlo truth, which is assembled at the *EventAction*, consists of
    - the main event, its assigned weight and its type as specified by a set of parameters, the event ID (run number, event number) and the four-vector of the collision vertex
    - all particles with their tracks, vertices and decay trees from the original generator event
    - all tracks produced during the Geant4 simulation, that have been flagged for saving at various points of the actual CMS simulation: they have produced hits in the sensitive detectors or they have been identified as important for the interaction history and the eventual reconstruction of the full tree
  - the Monte-Carlo truth is organized so as to allow navigation from hits to their corresponding tracks and parent vertices

Infrastructure and Services

Geometry

- detector description in CMS is handled in the CMS Detector Description Database, DDD, a COBRA subsystem
- Mantis provides mechanisms to convert DDD solids and materials to their Geant4 counterparts as well as the logical and physical volumes needed to build the Geant4 geometry for the chosen description
- the DDD SpecPars mechanism allows the definition of special parameter sets (extra attributes, field parameters, range cuts etc) to be associated with selected detectors

Magnetic field

- inherits from *G4MagneticField* and implements the Geant4 *GetFieldValue method*, so that any standard CMS magnetic field can be loaded by the COBRA *CMSMagneticFieldLoader* and passed to Geant4
- is a *world volume observer*, i.e. instantiates a field when notified that the detector has been built
- allows choice and configuration of G4MagIntegratorStepper (G4ClassicalRK4, G4SimpleHeum, G4SimpleRunge, G4HelixImplicitEuler etc), chord-finder and propagator, using the DDD SpecPars mechanism
- allows modeling, instantiation and configuration of local field managers for chosen detectors and particles, again using the DDD SpecPars mechanism

→ local field managers, an important Geant4 feature, handle particles that are either of little interest or unlikely to escape a given detector or set of volumes and can therefore be propagated with relaxed criteria as to the accuracy of the stepping and chord finding

→ this treatments allows a moderate (propagation in field anyway accounts for ~10% of processing time) performance improvement depending on the type of study and particles involved

example use case: all particles other than muons or charged pions of $E_{kinetic}>E_{threshold}$, specified by the user, in all calorimeter volumes

Sensitive detectors

- sensitive detectors inherit from G4VSensitiveDetector and their concrete implementations in the actual CMS application implement the Geant4 ProcessHits method
- the sensitive detectors are registered to the Geant4 sensitive detector manager but they are instantiated and "attached" to their corresponding geometrical volumes at run time according to the set-up described in the configuration DDD/XML file; the facility is *world volume observer*, i.e. it must be notified that the geometrical detector has been constructed before it can be instrumented

→ possibility of "instrumenting" (making sensitive) any volume, for prototyping purposes or in order to facilitate studies of energy losses in dead materials or specific parts of the detector

Hits and hit collections

- hit processing and collection is handled outside the framework by the detectors themselves, based on COBRA classes common for and sharable by all CMS applications; these common classes are managed by the COBRA *Profound* package
- COBRA read-out factories handle the hit formatting as will required for the digitization, which is handled outside the simulation, by the CMS *ORCA* reconstruction program

Generators

- an abstract generator factory based on the COBRA GeneratorInterface packages and services provides a *trigger* method to produce a *RawHepEvent* or *HepMC::GenEvent*; the RunManager instantiates the chosen generator (also referred to as "reader"), if any
- all generators can be run-time configured in terms of

→ the first event to be read (automatically determined in the case of interrupted run resumption),

→ the maximum number of events they can return depending on the total available (or a sensible number in the case of on-the-fly generation) and the first event esp. if non-zero or if the run is being resumed

→ the event vertex generator to be used (none, flat, Gaussian, test-beam specific)

→ the run and event numbering scheme to apply

available generators (some provided for backwards compatibility to allow reading of already available generator samples)

- *EventGunReader*: a particle gun with run-time choice and configuration of particle type, η and φ ranges, energy and p_t ranges with a given distribution (flat, Gaussian etc)
- *EventNtplReader* and *EventTxtReader*: read CMS generated physics events from *HBOOK* ntuple or ASCII file
- *EventPythiaReader*: read a *Pythia6* event generated on the fly, using the COBRA *Pythia6Interface*
- *EventStdHepReader*: read an event in *StdHep* format from an ASCII file
- *EventHepMCReader*: read a *HepMC::GenEvent* from any type of input file (ASCII, *POOL* database, on-the-fly etc)

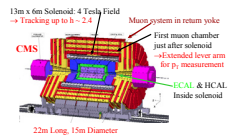
Physics

- an abstract physics list factory allows run-time selection and configuration of specific physics lists,
- physics cuts (i.e. range cuts) are implemented as cuts per region (set of volumes)
- the regions and volumes they contain (typical scheme distinguishes between "sensitive" and "dead" regions) and the cut values for electrons, positrons and photons are read from a DDD/XML file at run-time

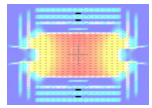
Observables and user actions

- the Geant4 mechanisms for the *UserActions* are employed to dispatch (using the COBRA implementation of the *Dispatcher-Observer* pattern) quantities such as the beginning and end of run, event, track and step
- user monitoring is implemented in the form of Observers of one or more of these quantities with access to the dispatched pointer via the COBRA Observer::upDate method

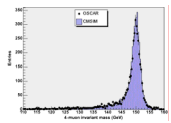
CMS Detector Layout



CMS Magnetic Field



CMS Physics



H→ZZ=150 GeV →4μ, M_H=150 GeV

What Mantis gets for free… Be lazy…

→ application control and interfaces to common services as described previously

→ persistency: has allowed transparent transition from Objectivity/DB to ROOT to POOL

→ histogramming and statistical analysis: has allowed transparent transition from HBOOK to Anaphe, to AIDA and ROOT

→ monitoring; including production monitoring and GRID interfaces

→ geometry descriptions from DDD and Geometry

→ visualization from IGUANACMS

→ S/W configuration management with CVS and SCRAM and quality assurance with IgNominy, IgProf, OVAL etc

Is Mantis really extensible?

sure seems so… so far experience with

- support for new types of physics lists
  - recent addition of a parameterized e/m shower simulation (*G4Flash*)
- support for new generator input formats and sources
  - migration from CMS *RawHepEvent* to *HepMC::GenEvent,* with input from ASCII, POOL etc
- support for new magnetic field and local field managers
- straightforward adoption of common CMS tools and compliance with CMS standards
- continuous evolution following and in close collaboration with Geant4
- success of the CMS OO simulation *OSCAR*, based on *Mantis*

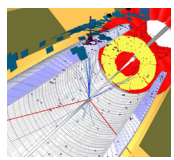
Configuration and datacards examples

In the card file

```
GoPersistent = true
ExtraPackages = MyPhysics:MantisNtplReader:MyExceptionHandler
RunManager:RestorePhysicsTables = true
DDDConfigFile = OSCARconfiguration.xml
EventNtplReader:NtplFileName = minbias.ntpl
Generator:ApplyEtaCuts = true
Generator:MinEtaCut = -5.5
Generator:MaxEtaCut = 5.5
Generator:VertexGenerator = GaussianEventVertexGenerator
MantisMagneticField:UseMagneticField = true
MagneticField:Name = CMSIMField
NumberOfEventsToBeProcessed = 10
```

In OSCARconfiguration.xml

```
<Include>
    <File name="CMS/CMSGeometry/cms.xml"  url="."/>
    <File name="ECal/ECalGeometry/ecal.xml"  url="."/>
    <File name="Materials/materials.xml"  url="."/>
    <File name="Rotations/rotations.xml"  url="."/>
    <File name="ECal/ECalSensDet/ecalsens.xml"  url="."/>
    <File name="ProductionCuts/EcalProdCuts.xml"  url="."/>
    <File name="PropagationInField/FieldParameters.xml"  url="."/>
</Include>
<Root fileName="cms.xml" logicalPartName="OCMS"/>
```



visualization with IGUANACMS

Summary and outlook

So far so good!

Next…
partial event simulation: when you need 50+ M events per channel…
e.g. H→ZZ→4l: μμμμ, eeμμ, eeee
→ simulate event without leptons, superimpose leptons simulated with correct kinematics

And then?
FLUKA interface via *FLUGG*?
common framework for all CMS simulations?
→ from fast (*FAMOS*) to parameterized (*G4FLASH*) to full simulation (*OSCAR*)…