ALICE Simulation Architecture

Linear Collider Workshop
November 15

René Brun
CERN
ALICE Event/100

Front View of a simulated event with only 1/100 of the expected multiplicity
ALICE Event/100

Side View of a simulated event with only 1/100 of the expected multiplicity

Estimated size of one raw event = 40 Mbytes

Simulated event with hits = 1.5 Gbytes

Time to simulate one event = 24 hours
Situation in 1998

- CERN official line = Objy + G4 + LHC++
- G4 & LHC++ not ready to use
- Existing simulation based on G3.
- G3, PAW, ROOT authors join Alice
- Idea is to move to G4 in the medium term
- Vital to have comparisons between G3 and G4
- Decision to develop a Virtual MC with:
  - Common input kinematics
  - Common Geometry
  - Common Hits/Digits output
A brief history

Software before June 1998

the galice package
(Geant3, Zebra, PAW)
no coordination
between detectors

New software organisation since 1998

Adoption of a unique framework (Sep 98)
Creation of AliRoot (Nov 98)

Online & Offline use ROOT (99)
Very fast transition to C++/OO
All detectors included

MDCs, Physics Performance Report (00/01)
The Virtual MC

AliRun

Detector Code

AliMC

TGeant3

TGeant4

TFluka

G3 geometry

G4 geometry
The AliRoot System

- Run Control
- Virtual MC
- Generators
- Fast MC
- FLUKA
- Geant3.21
- Geant4
- Geometry Database

Root particle stack

Transport engine selected at run time

Virtual MC

ROOT DB

Geometry Database

Selected Transport engine
Virtual Monte Carlo

- Both G3 and G4 simulations are based on the virtual Monte Carlo interface (abstract class AliMC)
  - Both G3/G4 applications are defined from the same source
  - Same input, geometry, hits, control

- Implementation of AliMC for G3 = TGeant3 class

- Implementation of AliMC for G4 = TGeant4 class
  - Each domain is covered by its manager class: geometry, physics, stepping, visualisation
  - Each manager uses corresponding categories of G4

- “T”-classes contain only ALICE independent code
AliRoot Modules

- Modularity of AliRoot reflected in the physical structure of the shared libraries
The AliRun Manager

```
AliRun --> AliVMC
  +fVMC

AliVMC
  MC Run Management()

AliMC
  Geometry Management Methods()
  Physics Management Methods()
  Step Management Methods()
  Vis Management Methods()
  G3 Specific Prototypes()

AliGeant3
  G3 Specific Prototype()

TG4RunManager (from g4me)

AliGenerator

AliGenHIJINGpara

and others

AliModule

AliDetector

AliTPC

AliTPCvN
```
Tracking schema

- FLUKA Step
- GUSTEP
- Geant4 StepManager
- AliRun::StepManager
- Module Version StepManager
  - Add the hit
- Disk I/O
- Root
AliGeant4/TGeant4

**TGeant4**
- Functions of general usage
  - G3 cuts and process controls
  - User limits base class
- MC functions for geometry definition
  - XML converter
- Sensitive detector ABC and management
- Modular physics list and physics constructors
  - Special processes implementing G3 cuts and controls
  - Mapping of G4 particles, physics processes

**AliGeant4**
- *global*
- *geometry*
- *digits+hits*
- *physics*
- Functions of general usage
- Detector construction
  - delegation to AliModule (STEER)
- Sensitive detector
  - delegation to AliModule (STEER)
AliGeant4/TGeant4

**TGeant4**
- MC functions for access to track in stepping (step manager)
- Stepping action ABC
- Tracking action ABC

**AliGeant4**
- Stacking Action
- Stepping Action
- Tracking Action
- Event Action
  - calls to AliRun (STEER)

**Primary Generator Action**
- delegation to AliGenerator (STEER)
- Run Action
  - calls to AliRun (STEER)

**Event**

**Run**

**Visualization**

**Interfaces**

**Root geometry browser**
void Config() {
    // ******* Create the output file
    TFile *rootfile = new TFile("galice.root","new");

    // ******* GEANT3 parameters
    TGeant3 *geant3 = new TGeant3("Interface to Geant3");
    geant3->SetSWIT(4,10);
    geant3->SetDEBU(0,0,1);
    geant3->SetDCAY(1);
    geant3->SetPAIR(1);

    // ******** Initialize event generator setup
    AliGenHIJINGpara *gener = new AliGenHIJINGpara(500);
    gener->SetMomentumRange(0,999);
    gener->SetPhiRange(0,360);
    gener->SetThetaRange(10,170);
    gener->SetSigma(0,0,0); //Sigma in (X,Y,Z)

    // ******** TOF parameters
    AliTOF "TOF" = new AliTOFv0("TOF","coarse geometry");

    // ******** TRD parameters
    AliTRD "TRD" = new AliTRDv1("TRD","TRD detailed geometry ");
    TRD->SetGasMix(0);

    // ******** TPC parameters
    AliTPC "TPC" = new AliTPCv1("TPC","detailed TPC");
    AliTPCParam *param = TPC->GetDigitParam();
    param->SetInnerRadiusLow(83.9);
    param->SetInnerRadiusUp(141.3);
    param->SetOuterRadiusLow(146.9);
    param->SetOuterRadiusUp(249.4);
    param->SetDeadZone(1.15);
    // ...
}

The AliRoot configuration is built at run time by executing a Root macro
"gAlice->Init(" Config.C")"
void Config() {

    // ******* Create the output file
    TFile *rootfile = new TFile("galice.root","new");

    // ******* GEANT3 parameters
    TGeant3 *geant3 = new TGeant3("Interface to Geant3");
    geant3->SetSWIT(4,10);
    geant3->SetDEBU(0,0,1);
    geant3->SetDCAY(1);
    geant3->SetPAIR(1);

    // ******** Initialize event generator setup
    AliGenHIJINGpara *gener = new AliGenHIJINGpara(500);
    gener->SetMomentumRange(0,999);
    gener->SetPhiRange(0,360);
    gener->SetThetaRange(10,170);
    gener->SetSigma(0,0,0);  //Sigma in (X,Y,Z)

    // ******** TOF parameters
    AliTOF *TOF  = new AliTOFv0("TOF","coarse geometry");

    // ******** TRD parameters
    AliTRD *TRD  = new AliTRDv1("TRD","TRD detailed geometry ");
    TRD->SetGasMix(0);

    // ******** TPC parameters
    AliTPC *TPC  = new AliTPCv1("TPC","detailed TPC");
    AliTPCParam *param = TPC->GetDigitParam();
    param->SetInnerRadiusLow(83.9);
    param->SetInnerRadiusUp(141.3);
    param->SetOuterRadiusLow(146.9);
    param->SetOuterRadiusUp(249.4);
    param->SetDeadZone(1.15);

    ....

}
Alirun modules
Example of a complete detector system integrated into AliRoot:

- **AliModule**
- **AliDetector**
- **AliMUON**
- **AliMUONChamber**
- **AliMUONResponse**
- **AliMUONResponseV0**
- **AliMUONResponseTrigger**
- **AliMUONSegmentation**
- **AliMUONSegmentationV0**
- **AliMUONSegmentationV01**
- **AliMUONSegmentationV02**
- **AliMUONSegmentationV04**
- **AliMUONSegmentationV05**
- **AliMUONSegmentationTrigger**
- **AliMUONSegmentationTriggerX**
- **AliMUONSegmentationTriggerY**
Algorithms organized in tasks
(ROOT class TTask)
Objects organized in Folders in memory (Root class TFolder)
Objects organized in Trees in the ROOT output files
Interface to Jetset and Pythia

- **TPythia derived from TGenerator**
  - Access to Pythia and Jetset common blocks via class methods
  - Implements TGenerator methods

- **AliPythia derived from TPythia**
  - High level interface to Jetset and Pythia
  - Tailored to our special needs:
    - Generation of hard processes (charm, beauty, J/ψ..)
    - Selection of structure function
    - Forced decay modes
    - Particle decays … and more
AliGenPythia Configuration

Example

- Simple configuration of particle generation with Pythia
- AliGenerator objects are made persistent
  - Parameters are available in output together with generated events.

```c
// Example for Charm Production with Pythia

AliGenPythia *gener = new AliGenPythia(10);
gener->SetPhiRange(0,360);
gener->SetYRange(2,4);
gener->SetPtRange(0,1000000);
gener->SetOrigin(0,0,0);
gener->SetVertexSmear(perTrack);
gener->SetSigma(0,0,5.6);
gener->SetProcess(charm_unforced);
gener->ForceDecay(semielectronic);
gener->SetPtHard(0,3);
```
A view of the problem

- Monte Carlo
- Raw data
- Hits
- Digits
- Summable Digits
- Track candidates
- Segments
- Reconstructed space points
- Particles
- Comparison

Information vs. Processing
Ex: PHOS simulation/Reconstruction
Situation fall 2001

- Despite a considerable investment in G4:
  - We have not been able to simulate a complete event with G4.
  - The physics (hadronics in particular) is not mature. More on this point by Federico.
  - The geometry package is disappointing: bugs, speed, functionality.
  - The visualisation is too primitive.

- Our production still based on G3.

- Decision to invest in FLUKA rather than G4:
  - We will again consider G4 when more mature & functional.
  - We maintain the G4 interface.

- Our initial choice of a Virtual MC validated.
Geometry Package

- We have a mini-workshop today and tomorrow where the design of a new geometry package is being discussed.

- This new geometry package is much better than G3 and G4 (power, simplicity).

- Designed for speed

- Can be used in simulation, reconstruction, DAQ, visualisation, independently of the MC package.

- We want to use it with G3, FLUKA and G4.
ALICE Off-line Project

Introduction

Welcome to the home page of the ALICE Off-line Project. This page and the following contain the description of the features of ALICE Off-line environment.

The ALICE Off-line Project has started developing the current framework in 1998. The decision was taken at that time to build the simulation tool for the Technical Design Report of the ALICE detector using the C programming technique and C++ as an implementation language. This led to the choice of ROOT as framework and GEANT 3.21 as simulation code. A prototype was quickly built and put in production. The experience with this was positive, and in November 1998 the ALICE Off-line project adopted ROOT as the official framework of ALICE Off-line.

**AliRoot** is the name ALICE Off-line framework for simulation, reconstruction and analysis. It uses the ROOT system as a foundation on which the framework and all applications are built.

Except for large existing libraries, such as GEANT 3.21 and Jdetectors, and some remaining legacy code, this framework is based on the Object Oriented programming paradigm, and it is written in C++.