EnsarRoot: The framework for simulation and data analysis for ENSAR

IOP Conf. Series: Journal of Physics: Conf. Series **1024** (2018) 012038 doi:10.1088/1742-6596/1024/1/012038

EnsarRoot: The framework for simulation and data analysis for ENSAR

Pablo Cabanelas[1,2], Héctor Alvarez-Pol[1,2], Elisabet Galiana[1,2] and Yago González-Rozas[1,2] for the SATNuRSE workpackage of ENSAR2

[1] Dpt. de Física de Partículas, Universidade de Santiago de Compostela, E-15782 Santiago de Compostela, Spain

[2] Instituto Galego de Física de Altas Enerxías (IGFAE), Dpt. Física de Partículas, E-15782 Universidade de Santiago de Compostela, Spain

E-mail: pablo.cabanelas@usc.es

Abstract. EnsarRoot is the simulation and analysis framework for the ENSAR project. It provides the software infrastructure and examples to develop an analysis and simulation code for Nuclear Physics experiments. It is written in C++, based on the FairRoot framework and implements data structure and geometry definitions based on ROOT. Dealing with the transport engines is done through the Virtual Monte Carlo concept. The code includes examples of different kind of detectors, specific event generators and event viewers, among other features.

1. Introduction

ENSAR and ENSAR2 (second phase) represent the integrating activity for European scientists who are performing research in Nuclear Physics [1]. It is funded by the European Commission within the HORIZON 2020 Programme, and the core aim is to provide access to large-scale nuclear research facilities in Europe, as GSI/FAIR, CERN/ISOLDE, GANIL and many others. EnsarRoot has been developed under the ENSAR Joint Research Activity SiNuRSE (Simulations for Nuclear Reactions and Structure in Europe), which continues as SATNuRSE (Simulations and Analysis Tools for Nuclear Reactions and Structure in Europe) in ENSAR2.

EnsarRoot is the dedicated framework for the ENSAR project. A software framework is a piece of code, a set of classes, scripts and templates, providing a given functionality that can be selectively completed or modified by additional user-written code, in this case to perform simulations and data analysis for the description of Nuclear Physics experiments. The purpose of a so structured framework is to improve the efficiency of creating new software, increase the reliability of a new application and reduce the programming effort. It is written in C++, it is based on the FairRoot framework libraries [2], which are common to many experiments at GSI-FAIR, and it loads the ROOT [3] libraries providing a ROOT-like output structure. EnsarRoot implements different examples of detectors and experimental setups, and uses the Virtual Monte Carlo concept for running simulations [4].

In the next paragraphs, a general description of the software is presented, and also some words are devoted to implemented cases.

2. The EnsarRoot Framework

EnsarRoot is a suitable framework for simulation and analysis of small and medium scale Nuclear and Particle Physics experiments, publicly available [5] under a GNU LGPL licence [6]. As already mentioned, it is written in C++, and CMake [7] is used as a build system. It runs in many Linux distribution and Mac OSX, so that users could install it in any linux box or select some configurations that have been previously tested. It is based on the FairRoot library to implement a set of base classes. It serves as the core where other developments can be implemented, that is, an user code for specific detectors could be added, based on existing examples, both for the simulation and for the development of additional analysis tasks.

For the simulation, the interface with the transport engines Geant3 and Geant4 [8, 9] is done through the concept of Virtual Monte Carlo, and the implementation we include derives from FairRoot. This concept allows the user to build a simulation independently from the Monte Carlo implementation itself.

The reconstruction is done event by event and is organized in tasks. For each event, several tasks or algorithms are accomplished, such as digitization and hit registration in the active volumes, previously defined in the simulation section. The users can create their own algorithm inheriting from existing example tasks or from the `FairTask` class. Being ROOT based, the initialization of tasks is done with ROOT steering macros and no executable files are needed. The input-output workflow is arranged also in ROOT structures like TFile, TTree or TClonesArray. It makes use of `TGeo` ROOT file format for geometry and navigation.

2.1. Software arrangement

The software package is arranged in modules. There are some common base modules which contain the features to be called by the different implementations:

- *ensarbase*: contains the `EnsarDetector` class. This class inherits from `FairDetector` class, and implements the local reference system and the global positioning of any implemented detector in the framework.
- *ensardata*: contains a list of the active detector systems, as well as classes `EnsarMCStack` and `EnsarMCTrack`, which are responsible for the storage of the output data of the transport engine in ROOT branches.
- *ensargen*: contains the different developed event generators to be called in the simulation initialization (see 2.2).
- *ensarevtdisplay*: contains all the features of a dedicated EVE event display (see 2.3), which are initialized through the class `EnsarEventManager`.

There are another two common modules: *gconfig* and *macros*. The first contains all the needed scripts for the configuration of the selected transport engine (Geant3 or Geant4), and the second, general example scripts for running the simulation and analyzing data.

Making use of these common base modules, the user implements its own system or application in a dedicated module. Each system or application module contains three major submodules:

- The main system class is located in the *detector* submodule and should inherits from the `EnsarDetector` class. This class transform the primary output of the transport engine and register the detector information in the corresponding ROOT TTree branch.
- The *data* submodule contains the dedicated classes for managing the data.
- The corresponding geometry files are stored in the *geometry* submodule.

Besides, the *macros* submodule contains the root scrips for creating the geometry, running simulations and perform the data analysis.

EnsarRoot provides templates for the construction of some of the most commonly used Nuclear Physics detectors, that is, complete system modules for experimental arrangements with NaI and CsI scintillator crystals, plastic scintillators, High Purity Germanium Detector, Silicon and Double-sided Silicon Strip Detectors or Resistive Plate Chambers.

Each system module can be activated and deactivated in the `CMakeLists` file before the compilation.

2.2. Dedicated Event Generators

Some specific event generators has been developed and implemented in EnsarRoot. Some interesting cases are:

- Standard Ion Generator: any given ion can be set as the initial transported particle by calling the `EnsarIonGenerator` class, which has an interface to the `FairIon` class, passing A, Z and charge state as arguments to define an ion.
- Generator for gamma-correlations in proton induced reactions: a dedicated event generator for correlated gamma particle emission has been embedded in the framework through classes `EnsarCascadeGen` and `EnsarSiGenerator`.
- Giant and Pygmy Dipole Resonance Generator: a complete generator has been implemented in the submodule *CascadeGen*, with easy running and parameter handling scripts; the output is produced in a proper ascii EnsarRoot format.
- CRY (Cosmic Ray Air Showers) Generator interface: the output of the CRY Cosmic Rays event generator [10] can be directly plug into the frame by calling the interface `EnsarCryAsciiGenerator`.

2.3. Event Display

It has been implemented an useful event display based on the EVE package of ROOT [11]. It draws not only the general view of detectors geometry and event tracks, but also their projections in different axis using the `TEveProjections` features. Also, a dedicated calorimeter representation has been implemented using the `TEveCalo`, which allows a topological study of the event in an user-friendly environment. Fig. 1 shows a simulated event in a CsI calorimeter using the event display.

3. Implementation Cases

Besides the templates for the most common cases of single particle detectors, some specific complex cases have been implemented in EnsarRoot and are delivered in the package. The two most important cases are presented in the following subsections.

3.1. The CALIFA Demonstrator Experiment

CALIFA [12] is the calorimeter for the detection of gammas and light charged particles of the R3B experiment [13] at FAIR. A dedicated experiment for testing the CALIFA Barrel Demonstrator has been carried out in November of 2016 at the tandem accelerator of the CTN/IST facilities, in Lisbon, Portugal [14]. Two sets of 64 CsI(Tl) crystals each and a HPGe detector were exposed to gamma rays emitted in proton reactions over Al targets. The full setup was simulated and implemented in the module *ctn*. The analysis of the simulated data and its comparison with the real data is being done with EnsarRoot. Figures 2 and 3 show the geometry implementation of the setup detectors.

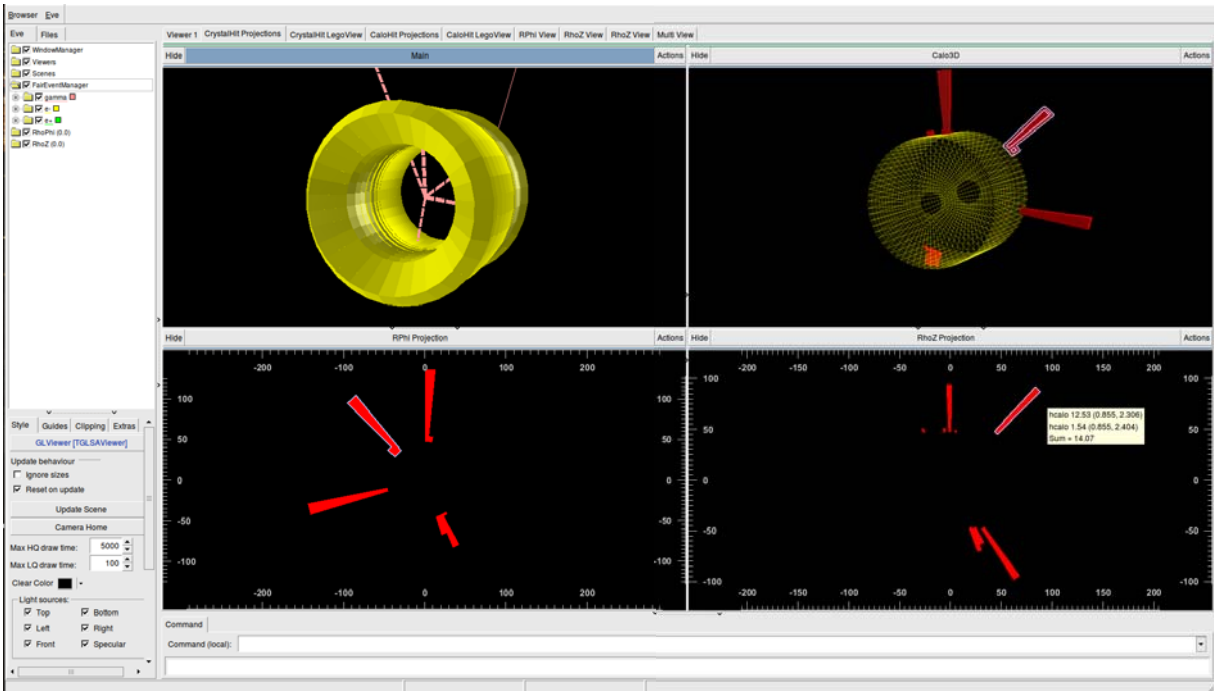

Figure 1. Example of a simulated event in a CsI calorimeter using the implemented event display of EnsarRoot

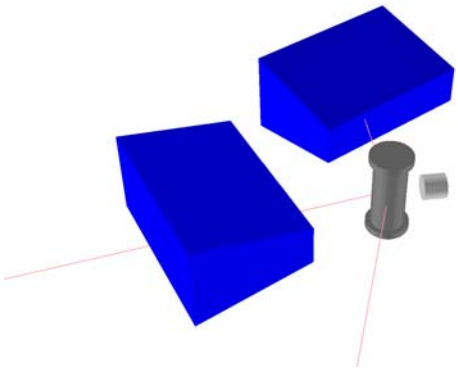

Figure 2. Geometry implementation of the setup for the The CALIFA Demonstrator Experiment, with the two segments of the demonstrator, the HPGe detector and a reaction chamber

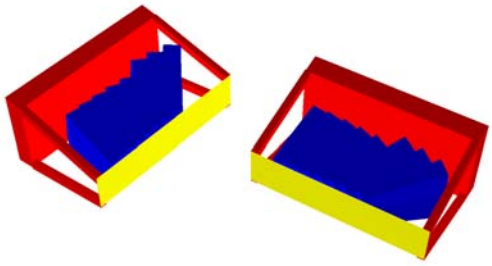

Figure 3. Detail of the inner part of the CALIFA Demonstrator segments. The crystals and the inner structure are visible

3.2. The TRAGALDABAS Detector

The TRAGALDABAS Detector [15] is a cosmic ray telescope based on Resistive Plate Chambers located at the Facultade de Física of the University of Santiago de Compostela, Spain. It is devoted to the understanding of the atmospheric showers arriving to the Earth surface. The system was implemented in the framework in the module *tragaldabas*, and all the simulations, performance analysis, efficiency and acceptance studies are being done with EnsarRoot. A picture of the system is shown in Fig. 4, and Fig. 5 shows a simulated event where the cosmic

rays are generated with CRY and are propagated with the `EnsarCryAsciiGenerator` interface.

Figure 4. Picture of the TRAGALDABAS Detector System based on four planes of Resistive Plate Chamber detectors

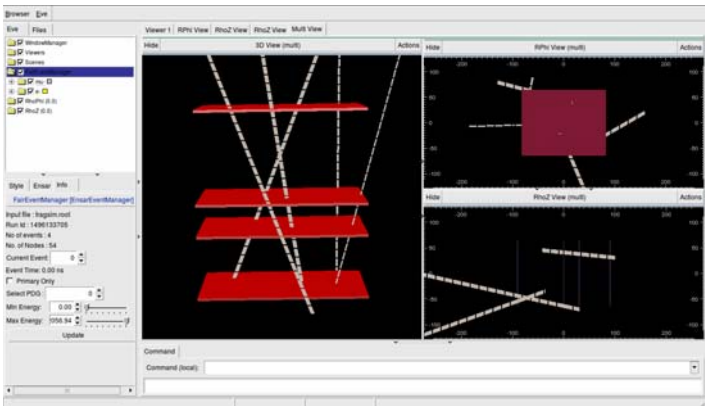

Figure 5. Example of a simulated air shower event over the TRAGALDABAS Detector in EnsarRoot

4. Summary

EnsarRoot, a general purpose simulation and event reconstruction framework for nuclear experiments has been developed. It is based on the ROOT package and the FairRoot library. It provides a set of classes and templates which allows the user to implement its own simulation in an easy way. It uses the Virtual Monte Carlo concept for the transport engines and implements several dedicated event generators. Input and output data for analysis is arranged in ROOT data structures. Different systems and experiments were already implemented in EnsarRoot and are delivered with the package as examples.

Acknowledgments

This project has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 654002 (ENSAR2), and from Plan Galego de Investigación, Innovación e Crecemento (I2C) of Xunta de Galicia under project No. POS-B/2016/015. The authors also thanks Enrique Nacher for the development of the Giant and Pygmy Dipole Resonance generator.

References

[1] European nuclear science and applications research - 2 URL `http://www.ensarfp7.eu/`
[2] Bertini D *et al.* 2008 *J. Phys.: Conf. Ser.* **119** 032011
[3] Brun R and Rademakers F 1997 *Nucl. Inst. and Meth. in Physics Research* A **389** 81
[4] Hrivnacova I *et al.* 2003 (*Preprint* `cs.SE/0306005`)
[5] URL `https://github.com/EnsarRootGroup/EnsarRoot`
[6] URL `https://opensource.org/licenses/lgpl-license`
[7] URL `https://cmake.org/`
[8] Brun R *et al.* 1987 GEANT3 CERN-DD-EE-84-1
[9] Agostinelli S *et al.* 2003 *Nucl. Inst. and Meth. in Physics Research* A **506** 250
[10] C Hagmann D L and Wright D 2007 *IEEE Nucl. Sci. Conf. Record* **24** 36
[11] URL `https://root.cern.ch/eve`
[12] Cortina D *et al.* 2014 *Nuclear Data Sheets* **120** 99
[13] URL `http://www.gsi.de/r3b`
[14] Cabanelas P *et al.* 2017 *GSI Scientific Report 2016* **RESEARCH-NUSTAR-KR-9** 219
[15] Garzón J A *et al.* 2017 (*Preprint* `astro-ph.HE/1701.07277`)