FROG: The Fast & Realistic OPENGL Event Displayer

Loïc Quertenmont[*][†]
Center for Particle Physics and Phenomenology (CP3)
Université catholique de Louvain
Chemin du cyclotron 2, B-1348-Louvain-la-Neuve - Belgium
E-mail: `loic.quertenmont@cern.ch`

FROG: The Fast & Realistic OPENGL Event Displayer is a generic framework dedicated to visualisation of events in high energy experiment. It is suitable to any particular physics experiment or detector design. The code is light and fast and can run on various operating systems, as its object-oriented structure (C++) relies on the cross-platform OPENGL and GLUT libraries. Moreover, FROG does not require installation of third party libraries for the visualisation. This documents describes the features and principles of FROG version 1.106, its working scheme and numerous functionalities such as: 3D and 2D visualisation, graphical user interface, mouse interface, configuration files, production of pictures of various format, integration of personal objects, etc. Finally, several examples of its current applications are presented for illustration.

XII Advanced Computing and Analysis Techniques in Physics Research
November 3-7, 2008
Erice, Italy

[*]Speaker.
[†]Funded by the "‘Fonds de la Recherche Scientifique"’ (FNRS.

1. Introduction

In high energy physics experiments, the possibility to visualise each event is crucial for several reasons. Understanding the event topologies can help in developing better data analysis algorithms. It can also be used as a powerful debugging tool for the simulation and reconstruction softwares.
A visualisation tool has to be :

- fluid : draw more than 60 frames per second
- fast : scan hundreds of events in few seconds *(required for commissioning purposes)*
- light : the entire package should not exceed 10 MB *(required for outreach purposes)*
- easily upgradable
- an intuitive debugging tool
- able to provide nice illustrations for communications

Satisfying simultaneously the above requirements could require heavy usage of all the resources of a computer. In general, these resources are the main limitation of the 3D visualisation. In large experiments such as CMS [5], complex algorithms are used to reconstruct and analyse physics data. Since these algorithms are processor and memory consuming, a fast visualisation tools should be decoupled from simultaneous physics reconstruction calculations.
The philosophy of FROG [1] [2] is to divide the software into a Producer and a Displayer (Figure 1). It has an impact on several other elements of the software, like the input file format, the operating system portability and the structure of the software.

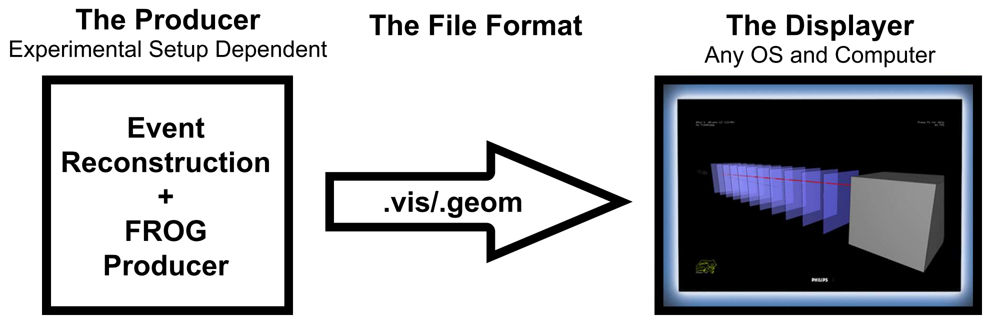

Figure 1: Factorisation of FROG into the Producer and Displayer components. The Producer is integrated in the experimental software. It extracts, processes and stores the data required by the Displayer, using the FROG File Format for data encapsulation. The Displayer is completely disconnected from the experimental software.

2. The FROG File Format

A dedicated FROG File Format (FFF) is needed in order to make the two parts of the code communicate with each other. While it has to be highly compact, the decoding time of the files has to remain as reduced as possible. Finally, the FFF should also be flexible enough. Its binary encoding ensures its universality with respect to the computer type and the operating systems.

3. The FROG Producer

The Producer is the interface between the physics software of the detector/experiment[1] and the FROG Displayer.
The FROG Producer builds the Events and the Geometry of a particular detector and stores them respectively into `.vis` and `.geom` files. The Producer is the only part of FROG that has to be interfaced to the user needs, and to the software and data format of the user's experiment. Some examples of Producer's implementation are available online as tutorials [6]. The Producer is in general a small piece of C++ code. Its task is to extract and process once for all the data required by the Displayer. For instance, it can convert the position of a hit from local coordinates of the detector to global coordinates. Such computations are in general relatively fast but can slow down the 3D visualisation when repeated many times. Since the Producer does not use specific graphical libraries, it can be run on parallel on a computer cluster and the resulting `.vis` file can then be merged in one unique file. The experiment software only needs to include the definition of the FROG classes in order to produce the `.vis` and `.geom` files. This reduced FROG package, containing only the class definition, takes of the order of 0.5MB on disk and is freely distributable.

4. The FROG Displayer

The Displayer has the unique function of displaying the content of the `.geom` and `.vis` files. It does not depend on the software of the experiment. The Displayer can be distributed and run on different platforms : Windows, Linux and MacOS are currently supported. The only requirement to execute the Displayer is to use some graphic libraries like OPENGL [3] and GLUT [4]. FROG is completely generic in the sense that it has been designed to be usable for any experiment and also to allow each user to easily upgrade/add some classe definitions in order to use FROG for his particular case. It can then be used for any physics experiments and eventually also for other purposes (e.g. in cosmology, biology, etc.).
All the style parameters are loaded from the FROG configuration file (`FROG/config.txt`). The geometry and the current event are displayed in the different 2D or 3D views. The Displayer can easily render more than 60 frames per second. The mouse clicks are handled in order to outline (flashing) and print out information of the clicked object. The Figure 2 shows a screenshot taken by the Displayer for a fictitious tracking experiment composed of eleven tracking layers, a beam source and an Ending Block that stops the beam.

5. The FROG Features

FROG is fully configurable by a set of parameters defined in an ASCII file (`FROG/config.txt`). This file contains, amongst other things[2]:

- the path to the input `.vis` file and/or input `.geom` file
- the first event to display
- the objects' colour, styles and thresholds

[1]Two examples of detector software are CMSSW in the CMS experiment and Athena in the ATLAS experiment.
[2]For a complete list of features, please see [1]

- the views to be used at the beginning

The paths in the configuration can be either a local path or a URL pointing to a `.vis/.geom` files or to gzipped files (`.vis.gz/.geom.gz`). In the second case, the files are automatically downloaded and uncompressed if needed. Many types of view are available (e.g. 3D, 2D, LegoPlot, Text, etc.) and users can easily code new types of view suiting better their needs. FROG can be used to take screenshots (this is done with key <`ENTER`>) in various picture format. Both the vectorial picture formats (e.g. PS, EPS, TEX, PDF, SVG and PGF) and the pixelized picture formats (e.g. PNG) are supported thanks to the GL2PS [7] and PNGLIB [8] libraries. FROG can be used online to have quickly clues on the quality of data taken by the experiment. This is transparent to the user, and usage of FROG online and offline only differs by the fact that in the online case, the Producer and the Displayer are used simultaneously. It is indeed possible to read the .vis file while the Producer is still pushing events into the same file.

6. Applications

FROG is already used in different experiments and environments, some of them are shown in this section. The two examples described in this section show that FROG can be used in many different applications.

6.1 GASTOF: The Ultra-Fast Gas Time-of-Flight Detector

GASTOF [9][10] detectors are Cherenkov gas detectors that will be located at 420 m from the CMS [5] and ATLAS [11] interaction point (IP), as part of the FP420 project [12] for the LHC [13]. The aim of these detectors is to reduce backgrounds due to accidental coincidence of events detected in the central detectors and in the FP420 detectors on each side of the IP. To achieve that, the z-coordinate of the event vertex measured by the central detectors is compared to the vertex reconstructed by measuring the time difference of forward proton arriving to the GASTOF detectors on two sides of the given IP. Cherenkov photons produced by high energy protons traversing gas medium are reflected by a mirror onto a very fast photomultiplier. The Figure 3 shows a simulated GASTOF event displayed by FROG.

6.2 CMS: The Compact Muon Solenoid

The Compact Muon Solenoid [5] is one of two general-purpose particle physics detectors built on the Large Hadron Collider (LHC) [13]. It is capable of studying many aspects of protons collisions with a center of mass energy up to 14 TeV. A FROG Producer has been created as an additional module of the CMS Software (CMSSW) [14]. The displayed geometry is guaranteed to be exactly the same as the geometry used by CMSSW for the event reconstruction since it is automatically extracted from the CMS database by the Producer. This is important for debugging purposes. Jets, from QCD events and other energetic processes, are very important at hadron colliders and challenge event display applications. In FROG, jets can be displayed in many different ways and one of them is represented on Figure 4.

7. Summary

The goal of the FROG project is to create a new event display able to display quickly a significant fraction of the event data in a high energy physics experiment. As a high rendering speed is desirable, the FROG software has been split into two parts: the Producer and the Displayer. In addition to the speed enhancement, this structure coupled with a custom file format brings many interesting features, like the portability of the framework on several operating systems, the independence with respect to the experiment or environment, the possibility to share easily visualisation data, etc. Another important application of FROG is to display the data in an online experiment, as the Producer and the Displayer can work simultaneously.

In addition of the previous features, FROG is based on an object oriented design. Therefore, it can display almost any imaginable object without significant degradation of the Displayer's speed. It can work on almost any recent computer with one of the following Operating Systems: Linux, MacOS or Windows. The size of the full FROG package, including the source code, is less than 4 Mb. It is really suitable for outreach applications since it can be freely distributed and installed in only few clicks. Moreover, if FROG is used to process the experiment data online, the latest available data can be automatically downloaded from the internet on any computer by the FROG Displayer. One of the main goal of the Displayer is to allow a large number of users to visualise event data in quasi-real time.

In conclusion, FROG can be used in many applications: it is highly tunable and allows a large number of users to visualise event data. The software is already used by several physicist communities for outreach and detector debugging.

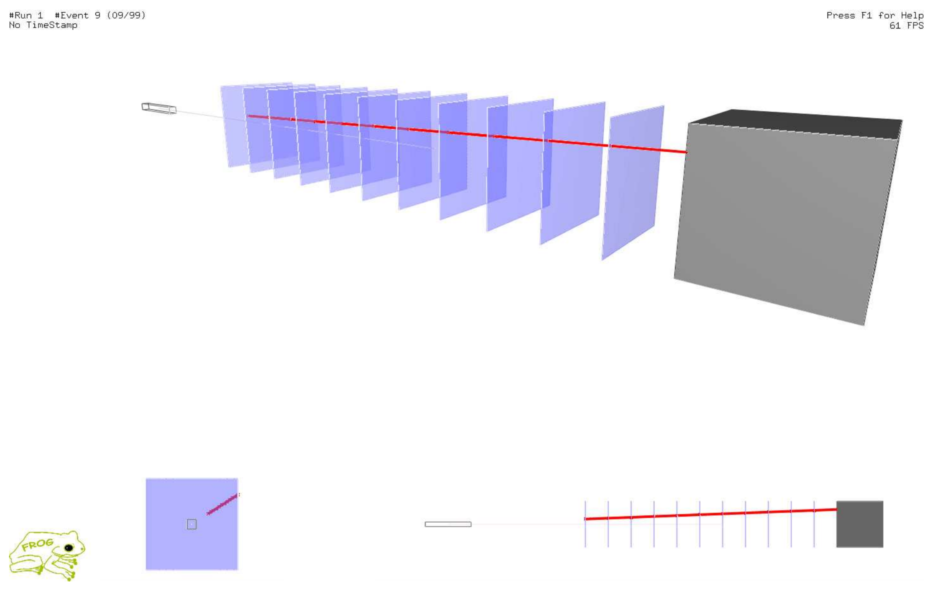

Figure 2: Three different FROG Display views: big 3D view (top), 2D longitudinal (bottom right), transversal view (bottom left). The Particle Gun (grey), the Ending Block (grey) and the eleven tracking layers (blue) are visible. The particle track is shown in red.

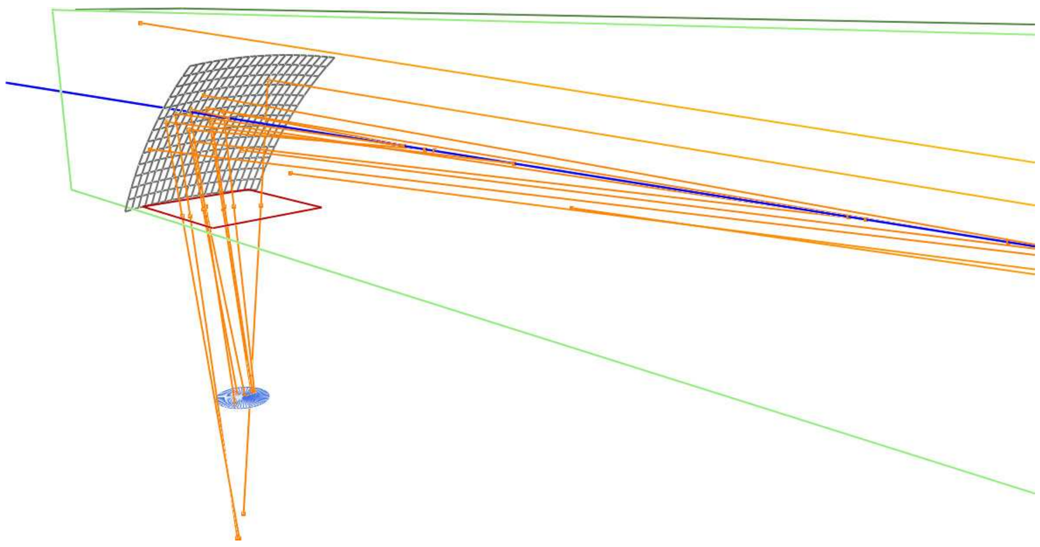

Figure 3: 3D view zoomed on the GASTOF mirror. The proton trajectory is represented by the blue line while photon rays are shown in yellow. The Cherenkov photon production is well visible along the proton path. The majority of the produced photons are reflected by the curved mirror and focused on the photocathode.

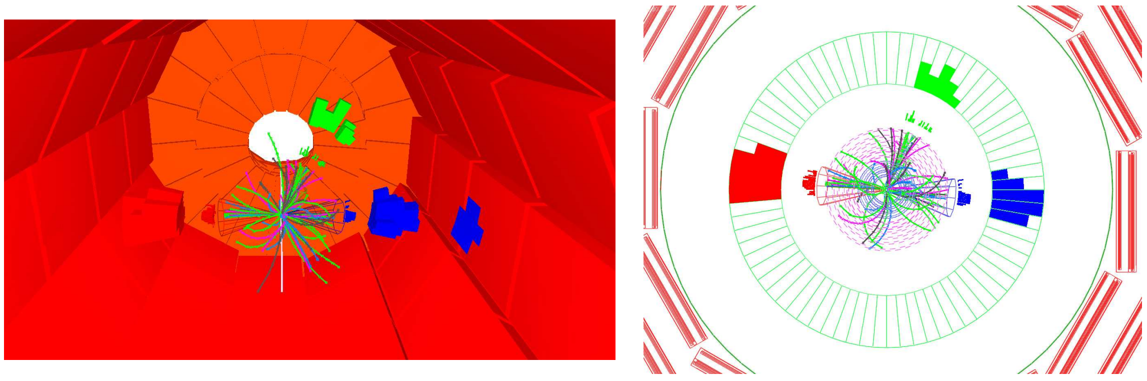

Figure 4: A 3D (left) and transversal (right) view of a QCD Di-jets event in CMS. The jets are represented by a cone pointing towards the calorimeter hits associated to the jet. The hardest P_T jet is red, the second one is blue and the third one is green. In this particular event, a third jet likely coming from a soft radiation of the second jet is also visible. In the 3D view, the geometry of the muon chambers is also shown.

References

[1] L. Quertenmont and V. Roberfroid, "**FROG: the Fast and Realistic OpenGL Displayer**," `http://projects.hepforge.org/frog/`, 2009. *arXiv:hep-ex/09012718v1*.

[2] L. Quertenmont and V. Roberfroid, "**FROG: the Fast and Realistic OpenGL Displayer: CMS Note in Preparation**," 2009.

[3] "**OPENGL: Open Graphical Library**," `http://www.opengl.org`.

[4] "**GLUT: OPENGL Utility Toolkit**," `http://www.opengl.org/resources/libraries/glut/`.

[5] M. Della Negra, A. Petrilli, A. Hervé, and al., "**CMS: The Compact Muon Solenoid, CMS Physics: Technical Design Report 1**," 2006. *CERN/LHCC 2006-001*.

[6] L. Quertenmont and Roberfroid, "**FROG Tutorials**," `http://projects.hepforge.org/frog/Tutorial/Tutorial.html`.

[7] "**GL2PS: an OPENGL to PostScript printing library**," `http://www.geuz.org/gl2ps/`.

[8] "**PNGLIB: Portable Network Graphics Library**," `http://www.libpng.org/`.

[9] L. Bonnet, T. Pierzchala, K. Piotrzowski, and P. Rodeghiero, "**GASTOF: Ultra-fast ToF forward detector for exclusive processes at the LHC**," 2006. *arXiv:hep-ph/0703320; CP3-06-18*.

[10] L. Bonnet and al., "**GASTOF: Paper in preparation**," 2009. .

[11] T. A. Collaboration, "**ATLAS** detector and physics: Performance technical design report," 1999. *CERN/LHCC 1999-14/15*.

[12] M. Albrow, R. Appleby, and al., "**FP420: The FP420 R&D Project: Higgs and New Physics with forward protons at the LHC**," 2008. *arXiv:hep-ex/08060302*.

[13] M. Benedikt, P. Collier, V. Mertens, J. Poole, and K. Schindl, *LHC Design Report*. Geneva: CERN, 2004.

[14] M. Della Negra, A. Petrilli, A. Hervé, and al., "**CMS: The Compact Muon Solenoid, CMS Physics: Technical Design Report 2**," 2006. *CERN/LHCC 2006-021*.