PROCEEDINGS OF SCIENCE

VISPA: a Novel Concept for Visual Physics Analysis

Oxana Actis, Martin Erdmann, Robert Fischer, Andreas Hinzmann, Matthias Kirsch, Tatsiana Klimkovich*, Gero Mueller, Matthias Plum, Jan Steggemann
RWTH Aachen University
E-mail: oxana.actis@physik.rwth-aachen.de,
erdmann@physik.rwth-aachen.de,
robert.fischer1@rwth-aachen.de,
hinzmann@physik.rwth-aachen.de,
matthias.kirsch@physik.rwth-aachen.de,
klimkovich@physik.rwth-aachen.de,
gero.mueller@physik.rwth-aachen.de,
matthias.plum@rwth-aachen.de,
jan.steggemann@rwth-aachen.de

VISPA is a novel graphical development environment for physics analysis, following an experiment-independent approach. It introduces a new way of steering a physics data analysis, combining graphical and textual programming. The purpose is to speed up the design of an analysis, and to facilitate its control.
As the software basis for VISPA the C++ toolkit Physics eXtension Library (PXL) is used which is a successor project of the Physics Analysis eXpert (PAX) package. The most prominent features of this toolkit are the management of relations, a copyable container holding different aspects of physics events, the ability to store arbitrary user data, and a fast I/O.
In order to support modular physics analysis, VISPA provides a module handling system using the above mentioned event container as the interface. Several analysis modules are provided, e.g. a module for automated reconstruction of particle cascades. All modules can be steered through Python scripts. Physicists can easily write their own modules to the module handling system or extend the existing ones.
In this paper the concept of VISPA will be presented.

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

*Speaker.

PoS(ACAT08)070

1. Introduction

In recent years major progress has been achieved in development of experiment-specific software analysis frameworks, typically based on C++ and specific configuration languages (e.g. [1, 2]). In some of them a dynamically-typed programming language Python is being increasingly used for an analysis environment [3, 4]. The issue of visualization while performing a physics analysis has been attempted in the past [5] and is considered again in the context of simplifying the software technical demands of performing a physics analysis. With the new project VISPA (Visual Physics Analysis) [6, 7], the step in between visualizing the measured objects in the detector (event display) and physics distributions (histograms) is graphically supported. In this step, the primary tasks of a physicist are prototyping, executing, and verifying a physics analysis. This is an iterative procedure until the analysis is finalized.

VISPA facilitates prototyping, performing, and verifying a physics data analysis by combining graphical and textual programming. This combination has been shown to speed up design and development in other fields, e.g. hardware control using the LabView program [8].

To deploy a visual environment for physics analysis, VISPA provides a multi-purpose window tool with a three column structure, a navigator panel, a window for graphical displays, and a property panel (Fig. 1). For the text-based programming, both the C++ and the Python languages are supported. VISPA has been developed independently of experiment-specific software with a well defined interface to connect to any high energy physics experiment. The interface is based on the software package PXL which provides a general container to hold all relevant information of a high energy physics event [9].

For designing a physics analysis, VISPA provides a graphical module steering which enables physicists to add, connect, and configure the analysis modules. Based on a plug-in mechanism, modules are already provided for the purpose of reading/writing event data, or for individual user analysis code. The latter can be coded directly within a corresponding editor. The visualization of the module steering facilitates verification of the analysis structure and communication between physicists doing analysis (Fig. 1). When using the Python language in their analysis code, the user programs can be executed without compiling and linking. For verification of the analysis, VISPA provides an event browser to inspect the overall event structure, to display particle decay cascades, and the properties of each particle (Fig. 2).

This contribution is organized as follows: first, the underlying C++ toolkit PXL is described. Then, the module steering system is discussed, followed by a description of the available analysis modules. Finally, the VISPA graphical user interface is described.

2. The PXL Toolkit

The C++ toolkit PXL has been developed since 2006 [9]. It is the successor of the PAX toolkit, which was developed from 2002 to 2007 [10, 11]. PXL provides all necessary features for an experiment-independent high level physics analysis with emphasis on an easy user syntax. Particularly, PXL enables the reconstruction of decay trees and the handling of analyses with reconstruction ambiguities. PXL offers an extensible collection of physics objects, representing particles, vertices, events, and collisions. In the analysis of an event containing reconstructed data,

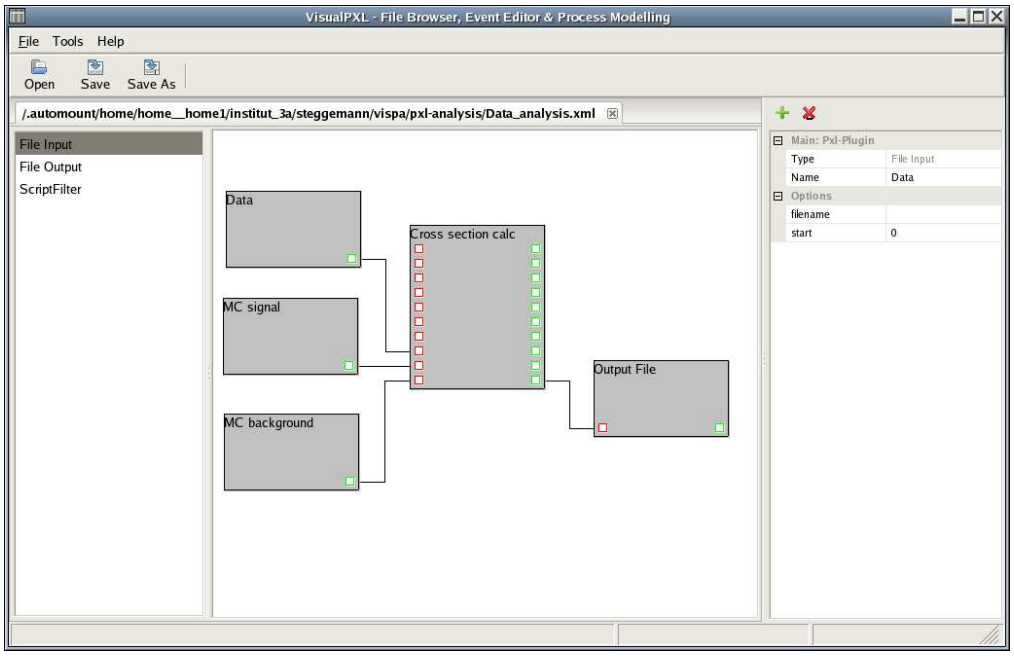

Figure 1: Analysis design within VISPA graphical platform.

new information can be added to each object by means of user event data. Between all objects, relations can be established, e.g. to build up decay trees, or to associate reconstructed particles with generated particles. The class pxl::Event represents an entire physics event and pxl::EventView is a special view of this physics event. These classes act as containers for physics objects, holding the relations between them, and as the standard interface to algorithms. Copies of these classes preserve all contained information such as the relations between particles. These features efficiently support the evaluation of hypotheses. PXL includes all objects into its I/O scheme, where a file is made up of compressed binary data chunks, thus guaranteeing file integrity during job execution. To enable the usage of all PXL objects and their methods within Python programs, a Python extension PyPXL is provided.

3. Module Steering

The design of physics analyses with VISPA is based on the decomposition of analyses into modules. Whereas a simple analysis may require only a few modules which are connected serially, a complex analysis requires more modules and more sophisticated streams of data. The VISPA module steering system controls the data flow as well as the selection and settings of analysis modules. The module selection is based on a plug-in mechanism, guaranteeing extensibility and efficiency as only libraries for the analysis modules deployed in the current analysis need to be provided. The data flow is managed by connections between sources (data input) and sinks (data output) of the respective modules. This permits the usage of multiple data streams as needed in complex physics analyses. Any state of the analysis can be stored and received back either in XML

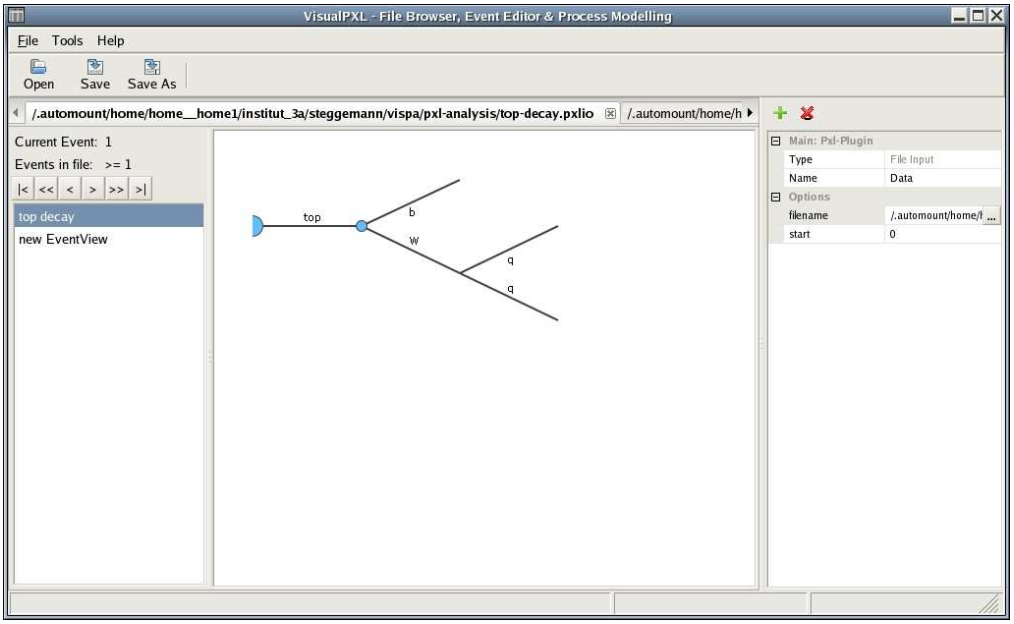

Figure 2: Event tree browser within the VISPA graphical platform.

format or in Python. The analysis can be executed either in batch mode, or interactively from the VISPA graphical platform.

4. Analysis Modules

A variety of analysis modules is provided for tasks of different complexity within a physics analysis. File operations are handled by input and output file modules. For user-demanded tasks within an analysis, a generic Python analysis module is provided, where the Python analysis code can be edited interactively within the VISPA graphical platform. An example of a C++ based complex analysis module is the automated reconstruction of particle cascades, a task arising in the reconstruction of particles with combinatorial ambiguities [12]. Given a template of a particle decay cascade and reconstructed particle data as input, this module generates all possible reconstruction versions. For events simulated with a Monte Carlo generator it supports finding the reconstructed version corresponding to the correct decay chain.

5. Graphical User Interface

The VISPA multi-purpose window tool serves as the graphical user interface for the design and steering of analyses, and for the browsing of complete physics events. It provides a common user interface for these different tasks (Figs. 1, 2). The frame on the left-hand side is used for the selection of modules or events, the main window displays the analysis modules or the event content, and the frame on the right-hand side shows the properties of the item selected in the main window. All objects in the main frame can be selected and moved like in popular software (drag-and-drop). Icons for opening and saving data files as well as other analysis modules are provided. By using

a tabbed document interface, several files can be opened in parallel. Browsing physics event data allows the verification of physics analyses on an event-by-event basis. This can be done at any stage of a progressing analysis and includes all information added by the user in the course of the analysis. The correctness of each object can be explored by selecting it, and inspecting its contents in the property grid on the right-hand side. In addition, the visualization of decay trees allows to check if all relations have been established correctly. To display complex decay trees, the VISPA graphical browser incorporates an algorithm for their proper layout. The algorithm is based on a model of physical forces, like spring forces, or gravity. Each starting and end point of a particle is provided with a node subjected to these forces. Using an iterative procedure, the positions of the nodes are optimized with respect to balanced forces. This algorithm results in a well-distributed view of, e.g., asymmetric decay trees.

6. Conclusion

A novel physics analysis environment, VISPA, has been presented. VISPA facilitates prototyping, performing, and verifying a physics data analysis by combining graphical and textual programming. It provides the so far missing graphical support for physicists in the step between displaying events, and visualizing physics distributions.

7. Acknowledgements

We are very grateful for financial support of the Ministerium für Innovation,Wissenschaft, Forschung und Technologie des Landes Nordrhein-Westfalen, the Bundesministerium für Bildung und Forschung (BMBF), and the Deutsche Forschungsgemeinschaft (DFG).

References

[1] A. B. Meyer [H1 Collaboration], *A new object-oriented physics analysis framework for the H1 experiment*, Prepared for *International Europhysics Conference on High-Energy Physics (HEP 2001)*, Budapest, Hungary, 12-18 Jul 2001.

[2] F. Fabozzi, C. D. Jones, B. Hegner and L. Lista, *Physics analysis tools for the CMS experiment at LHC*, CERN-CMS-NOTE-2008-015.

[3] G. Barrand, M. Frank, P. Mato, E. de Oliveira, A. Tsaregorodtsev and I. Belyaev, In *Interlaken 2004, Computing in high energy physics and nuclear physics*, 377-380.

[4] C. D. Jones, L. Luca and B. Hegner, J. Phys. Conf. Ser. **119** (2008) 032027.

[5] B. Ferrero Merlino, *The LHC++ environment*, CERN-OPEN-2000-191.

[6] O. Actis *et al.*, *Visual Physics Analysis (VISPA) - Concepts and First Applications*, arXiv:0810.3609 [physics.data-an].

[7] *VISPA (Visual Physics Analysis)*, http://vispa.sourceforge.net.

[8] National Instruments, *LabView*, http://www.ni.com/labview.

[9] M. Erdmann, G. Mueller, J. Steggemann, *Physics eXtention Library 2.0*, http://pxl.sourceforge.net.

[10] M. Erdmann, D. Hirschbuehl, Y. Kemp, P. Schemitz and T. Walter, *User oriented design in high energy physics applications: Physics analysis expert*, Prepared for *14th Topical Conference on Hadron Collider Physics (HCP 2002)*, Karlsruhe, Germany, 29 Sep - 4 Oct 2002.

[11] S. Kappler *et al.*, IEEE Trans. Nucl. Sci. **53** (2006) 506 [arXiv:physics/0512232].

[12] O. Actis *et al.*, *Automated Reconstruction of Particle Cascades in High Energy Physics Experiments*, arXiv:0801.1302 [physics.data-an].