PROCEEDINGS OF SCIENCE

Visual Physics Analysis - Applications in High Energy and Astroparticle Physics

M.Brodski, M.Erdmann, R.Fischer, A.Hinzmann*, T.Klimkovich, D.Klingebiel, M.Komm, G.Müller, T.Münzer, J.Steggemann, T.Winchen

RWTH Aachen University, Physikalisches Institut 3A, 52062 Aachen, Germany
erdmann@physik.rwth-aachen.de

VISPA (Visual Physics Analysis) is a development environment to support physicists in prototyping, execution, and verification of data analysis of any complexity. The key idea of VISPA is to develop physics analyses using a combination of graphical and textual programming. In VISPA, a multipurpose window provides visual tools to design and execute modular analyses, create analysis templates, and browse physics event data at different steps of an analysis. VISPA aims at supporting both experiment independent and experiment specific analysis steps. It is therefore designed as a portable analysis framework for Linux, Windows and MacOS, with its own data format including physics objects and containers, thus allowing convenient transport of analyses between different computers. All components of VISPA are designed for straightforward integration with experiment specific software to enable physics analysis with the same graphical tools. VISPA has proven to be an easy-to-use and flexible development environment in high energy physics as well as in astroparticle physics analyses.

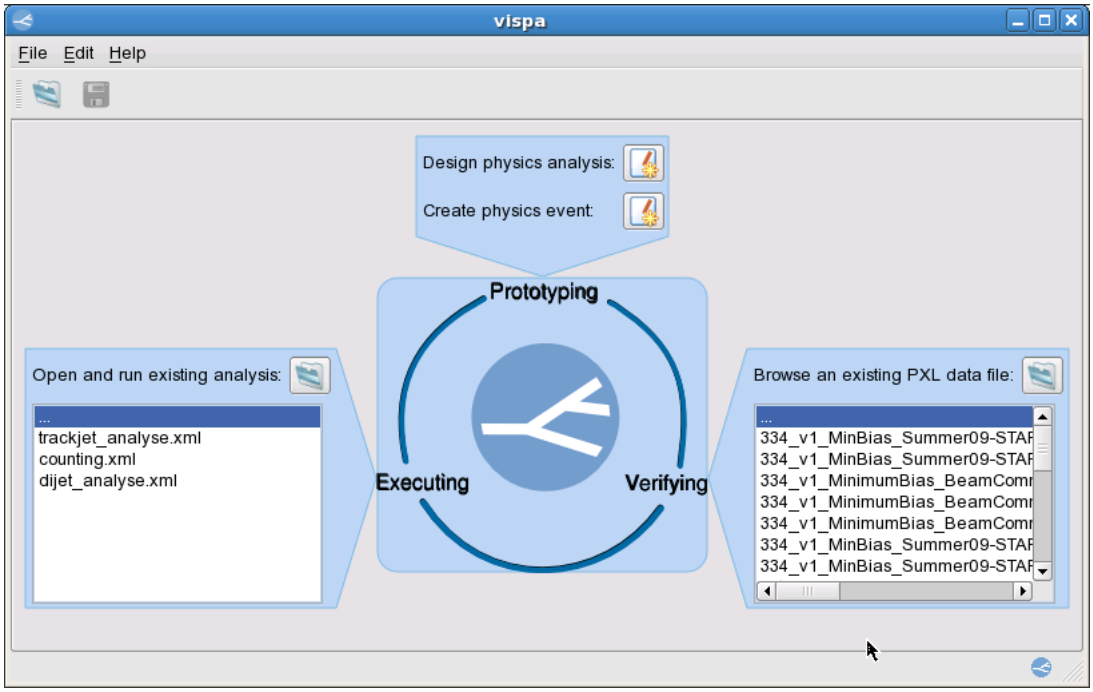
13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research, ACAT2010
February 22-27, 2010
Jaipur, India

*Speaker.

PoS(ACAT2010)064

1. Introduction

Nowadays graphical tools are commonly used to support physics analyses. Among the most widely used tools are integrated development environments such as Eclipse [1] for code development as well as data browsing tools such as the ROOT-TBrowser [2] and event displays in high energy physics. The concept of the Visual Physics Analysis (VISPA) project [3] is to incorporate graphical tools for analysis development including data browsing in an **integrated development environment for physics analysis**. The goal of VISPA is to allow physicists to spend more time on solving physics problems than on technical implementations. Therefore VISPA enables development of an entire analysis in a single integrated development environment. In the startup screen of VISPA, shown in Figure 1, a typical analysis cycle is depicted, consisting of prototyping, execution and verification. VISPA supplies graphical tools for each of these steps.



(a)

Figure 1: Startup screen of VISPA.

The **fields of application** for VISPA are in high energy physics and astroparticle physics and may extend in the future. As an example, VISPA has been used for the development of a method for measuring cosmic magnetic fields with ultra high energy cosmic ray data [4]. This analysis consists of a linear chain of Python [5] and C++ modules which process the cosmic rays originating from randomly generated universes. Another example is a development of an electroweak top quark analysis within the CMS experiment where simulated collision events pass through a logical tree of modules. This analysis makes use of the auto-reconstruction process [6] module that automatically reconstructs multiple versions of decay trees using a steering template of the parton process on input.

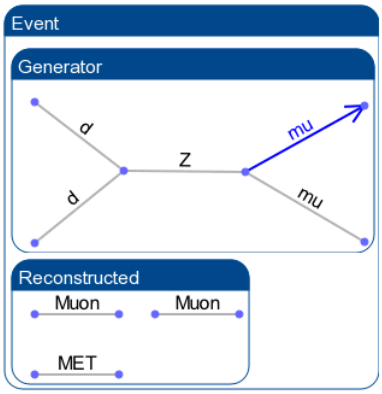
A key feature of VISPA is that it is designed for any level of complexity and user experience. It enables highly flexible analysis concepts according to the needs of the individual analyses. The data format allows any data to be stored and user extensions to be included. Simultaneously a minimal learning time and the possibility to design basic analysis with little knowledge of programming supports student-level analyses. The visualization of modular analysis steps encourages development of well structured analyses. VISPA has been used in hands-on exercises of the Elementary Particle Physics lecture series (4th year) at the RWTH Aachen University (2009).

The two **main components of VISPA** are the analysis toolkit PXL (Physics eXtension Library) and the graphical user interface of VISPA. In the following two sections, first the analysis toolkit and secondly the graphical user interface will be explained in detail.

2. The Physics eXtension Library (PXL)

The Physics eXtension Library (PXL) [7] is the underlying analysis software for VISPA. It is a C++ toolkit offering a collection of tools to support experiment independent physics analysis. PXL has been continuously developed since 2006 as the successor of the PAX toolkit [8]. The main components of PXL will be discussed in the following. Each component is accessible within VISPA, but can be used separately as well.

PXL provides a set of **interfaces for physics analysis** in high energy and astroparticle physics. They include physics objects (e.g. pxl::Particle), containers (e.g. pxl::Event), object relations (e.g. the pxl::Relations of pxl::Particles) and an interface allowing for the addition of user specific data to any of these objects (e.g. pxl::UserRecords). Instances of these objects can be organized in the following sense. A pxl::Event can hold several pxl::EventViews which themselves contain pxl::Particles and their relations to construct, e.g., Feynman diagrams, or particle decay trees. Different pxl::EventView's are useful, e.g., for comparisons of generated particles and objects reconstructed in a detector. Figure 2 shows an example of a representation of a high energy physics event in PXL.



(a)

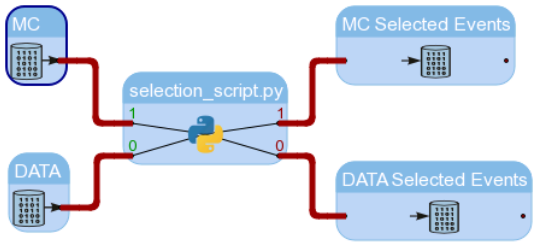
Figure 2: Example for an event containing two views of the event filled with particles and their relations.

All classes of PXL are available in the Python language as well. This feature is essential for the implementation of analysis modules in the Python programming language. The **PyPXL** interface

is wrapped around the C++ classes using the SWIG [9] tool.

PXL has its own fast, compressed and flexible **I/O format**. All objects in PXL are derived from the pxl::Serializable which defines serialization (deserialization) to (from) disk. It is designed for simple splitting and merging of data at the file level and allows exchange of data between all common platforms (Linux, Windows and MacOS).

PXL provides a **framework for modular physics analyses** in which each module can have multiple sinks and sources to control the data flow. Figure 3 shows an example of a simple analysis in PXL. The data flow is from left to right, starting from an input or generator module, and visualized by connection lines. For data exchange between the modules, a common interface is defined. In this example from high energy physics the interface is the pxl::Event, but in principle any pxl::Serializable can be exchanged. PXL analyses can be saved and loaded in XML format. They can be executed both on a batch system using the executable pxlrun as well as interactively using the graphical user interface of VISPA.



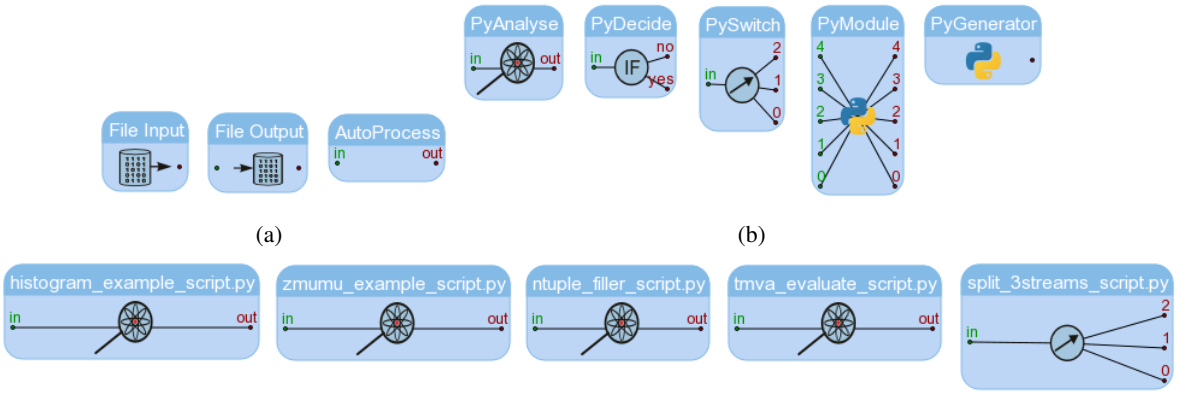
(a)

Figure 3: Analysis which filters events from data and Monte Carlo (MC) input using the same selection module and writes them out to two files.

PXL delivers a variety of **module interfaces and examples**. From these interfaces users derive their analysis specific modules. Is it possible to use both C++ and Python modules in the same analysis. C++ modules are used for performance-sensitive analysis tasks, while Python modules are used for fast prototyping and analysis logic. Figure 4 summarizes the standard modules available in PXL. They include a set of examples which explain the access to common tools (e.g. plotting histograms using PyROOT [2]).

3. Visual development of physics analyses

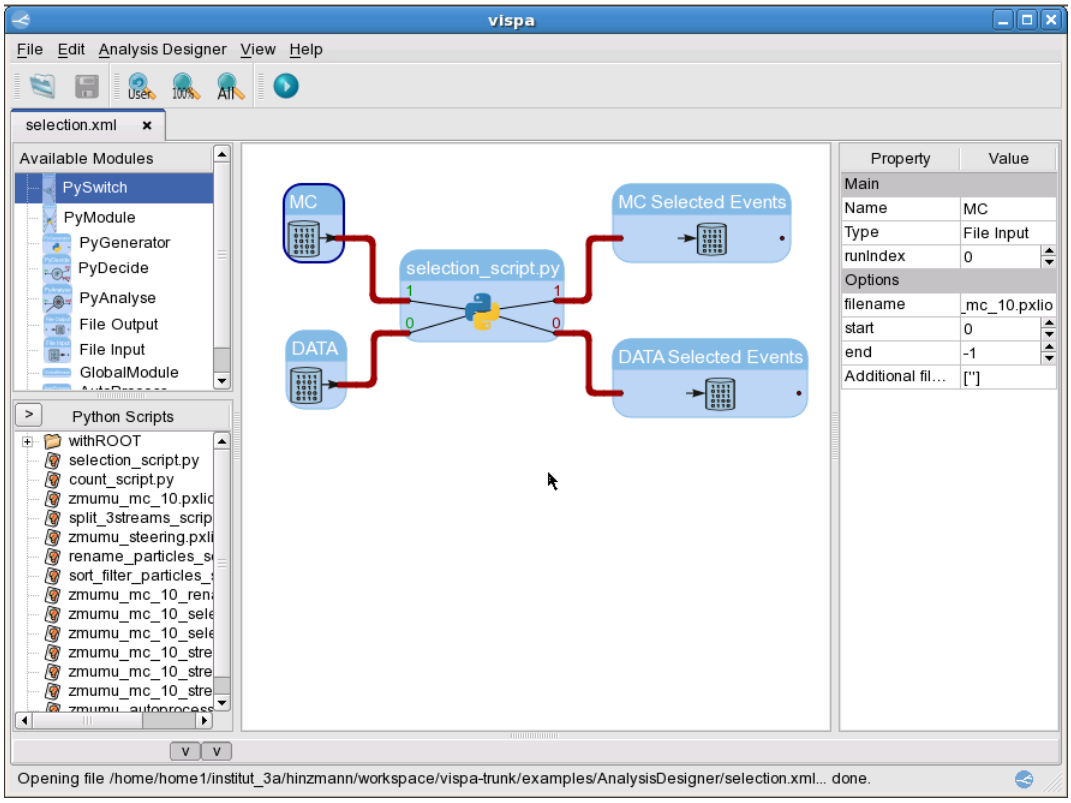
VISPA supplies a graphical user interface for **visual development of analyses**, the Analysis Designer, shown in Figure 5. The Analysis Designer allows the design of new analyses by connecting and configuring a chosen set of modules. Analyses can be saved as XML files and opened again for editing. The main functionalities of the Analysis Designer are described in the following. It lists the available C++ and Python modules which can be inserted into an analysis using drag and drop. Connection lines between modules can also be drawn using drag and drop. The parameters of the modules can be modified using the properties grid on the right side of the window. A double click on a module opens its script for editing. A double click on input and output modules opens the data browser for inspection of the corresponding file. On execution of the analysis in the Analysis Designer the output is shown in an extra section at the bottom of the window.



(a) (b)

(c)

Figure 4: (a) Standard C++ modules, (b) Standard Python modules, and (c) Example modules.

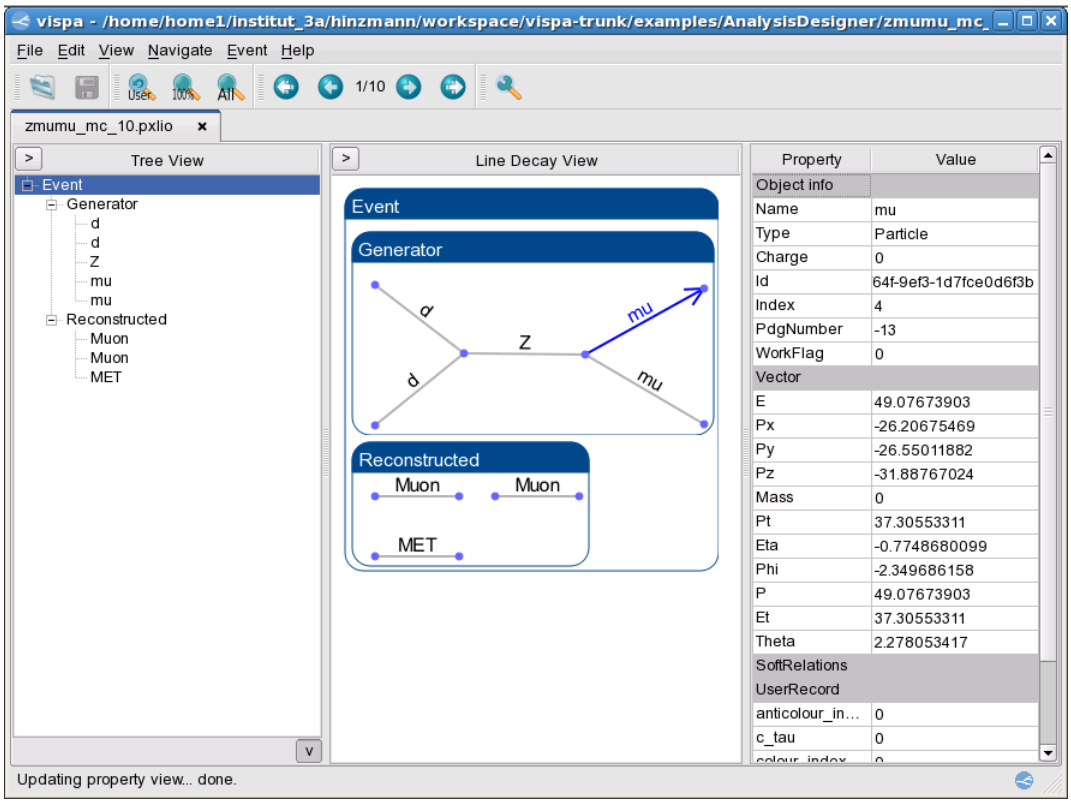


(a)

Figure 5: Screen shot of the Analysis Designer.

The data browser of VISPA, shown in Figure 6, gives a **visual representation of all data** contained in a PXL data file. It draws decay trees according to the relations of the displayed objects. All properties including user defined data of any object can be inspected using the properties grid on the right side of the window. As opposed to typical event displays in high energy physics, this browser is able to show every single object and parameter in the file including user defined data on

an event by event basis. Further, the data browser also serves as an editor for event templates, that can be used, for example, to create steering files for the auto-reconstruction process module.



(a)

Figure 6: Screen shot of the PXL Data Browser

In order to allow physics analysis using a single integrated development environment both **experiment independent and experiment specific analysis steps** are supported by VISPA. For this purpose the graphical user interface codebase has undergone a major redesign recently. The graphical platform is now based on a plugin mechanism and a large collection of graphical components to allow straightforward implementation and integration with experiment specific software. An example for a successful integration with experiment specific software is the Configuration Editor [10] which is a tool to edit the job configuration files used in the CMS experiment for simulation and reconstruction.

In physics analyses it is essential to be able to easily share work. An important feature of VISPA is **portability of analyses**. The well defined interface for modules allows the sharing and reuse of common modules within a working group. VISPA further allows the exchange of complete analyses by an automated tar-ball creation integrated in the graphical user interface. The exchange between different platforms is also possible since the graphical user interface is based on the platform independent application framework PyQt4 [11, 12]. VISPA is available on Linux, Windows and MacOS.

In 2009 VISPA has undergone a review on **software quality and performance** in collaboration with the Institute for Software Engineering, RWTH Aachen University. Measures such as the

McCabe metric for function complexity have shown good maintainability for PXL. The study has also triggered several performance improvements, in particular in the field of the I/O compression.

A new idea to perform VISPA analyses using a web browser has been recently proposed and implemented as a proof of principle. **VISPA@Web** shall allow server-based analyses using a graphical user interface implemented as a web page. The advantage of this concept is that no local user installation is needed and, therefore, it can be easily used in teaching applications, for example.

4. Summary

VISPA is a graphical development environment for physics analyses with applications in high energy and astroparticle physics. Its availability on all common platforms (Linux, Windows and MacOS) and its support for both Python and C++ modules makes it a powerful tool for collaborative work on physics analysis at any degree of complexity.

References

[1] *Eclipse*, http://www.eclipse.org/.

[2] R. Brun, F. Rademakers *ROOT - An Object Oriented Data Analysis Framework*, *Proc. AIHENP96 Workshop, Lausanne (1996)*, *Nucl. Inst. & Meth. in Phys. Res. A* **389** (1997) 81, http://root.cern.ch/.

[3] O.Actis et al. *Visual Physics Analysis (VISPA) - Concepts and First Applications*, *Proc. 34th Int. Conf. High Energy Physics (ICHEP 2008), Philadelphia, Pennsylvania* [arXiv:0810.3609], http://vispa.sourceforge.net.

[4] M. Erdmann, P. Schiffer *Measuring Cosmic Magnetic Fields with Ultra High Energy Cosmic Ray Data*, *Astropart. Phys.* **33** (2010) 201 [arXiv:0904.4888].

[5] *Python Programming Language*, http://www.python.org.

[6] O. Actis et al. *Automated Reconstruction of Particle Cascades in High Energy Physics Experiments* (2009) [arXiv:0801.1302].

[7] *Physics eXtension Library (PXL)*, http://pxl.sourceforge.net.

[8] S. Kappler et al. *The PAX Toolkit and its Applications at Tevatron and LHC*, *IEEE Trans. Nucl. Sci.* **53** (2006) 506 [arXiv:physics/0512232].

[9] *Simplified Wrapper and Interface Generator (SWIG)*, http://www.swig.org.

[10] M Erdmann et al. *Visualization of the CMS Python Configuration System*, *J. Phys.: Conf. Ser.* **219** (2010) 042008, http://twiki.cern.ch/twiki/bin/view/CMS/SWGuideConfigEditor.

[11] *Qt - A cross-platform application and UI framework*, http://www.qtsoftware.com.

[12] *PyQt*, http://www.riverbankcomputing.co.uk.