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

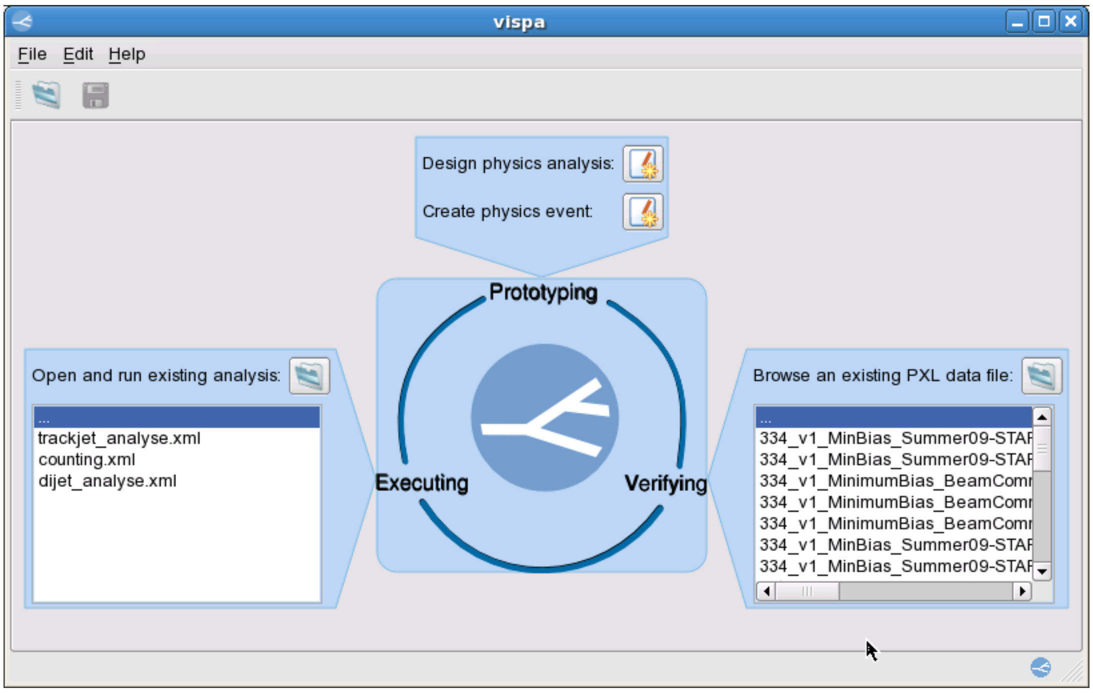
VISPA (Visual Physics Analysis) is a development environment to support physicists in prototyping, execution, and verification of data analysis of any complexity. The key idea of VISPA is to develop physics analyses using a combination of graphical and textual programming. In VISPA, a multipurpose window 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, supporting Linux, Windows and MacOS, with its own data format including physics objects and containers, thus allowing easy transport of analyses between different computers. All components of VISPA are designed for easy 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.

1. Introduction

Nowadays graphical tools are commonly used to support physics analyses. Among the most used tools are integrated development environments such as Eclipse [1] for code development and data browsing tools such as the ROOT-TBrowser [2] and event displays in high energy physics. The concept of Visual Physics Analysis (VISPA) [3] is an **integrated development environment for physics analysis** that provides graphical tools for all steps of analysis development including data browsing. The goal of VISPA is to enable physicists to develop an entire analysis in a single integrated development environment and spend more time on physics problems than on implementation. 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.


vispa
File Edit Help
Design physics analysis:
Create physics event:
Prototyping
Executing
Verifying
Open and run existing analysis:
...
trackjet_analyse.xml
counting.xml
dijet_analyse.xml
Browse an existing PXL data file:
...
334_v1_MinBias_Summer09-STAF
334_v1_MinBias_Summer09-STAF
334_v1_MinimumBias_BeamComi
334_v1_MinimumBias_BeamComi
334_v1_MinimumBias_BeamComi
334_v1_MinBias_Summer09-STAF
334_v1_MinBias_Summer09-STAF

(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 in an analysis 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 content of random generated universes. Another example is a complex single top analysis performed within the CMS experiment where simulated collision events pass through a logical tree of modules. This analysis makes use of the Autoprocess [9] module that automatically reconstructs multiple versions of decay trees using a steering tree as input.

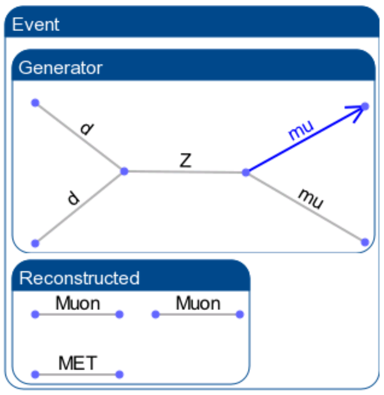
A key feature of VISPA is that it is designed for any level of complexity and user experience. It provides a highly flexible analysis concept and data format for expert analyses. At the same time a minimal time of learning and the possibility to design easy analysis with little knowledge of programming supports student level analyses which also profit from the facts that VISPA forces structuring of analyses and provides understanding of data. Another field of application for VISPA is teaching since it allows quick and easy implementation of simple problems. VISPA was already used in the “Elementary particles” (4th year students) hands-on exercises at RWTH Aachen University (2009).

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

2. The Physics eXtension Library (PXL)

The Physics eXtension Library (PXL) [8] is the underlying analysis framework for VISPA. At the same time 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 [7]. The main components of PXL will be discussed in the following. Each component is part of the framework for VISPA, but can be used separately as well.

PXL provides a set of **interfaces for physics analysis** for high energy and astroparticle physics. They include physics objects (e.g. pxl::Particle, pxl::UHECR), containers (e.g. pxl::Event, pxl::BasicContainer), 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. the pxl::UserRecords of a pxl::Event). Figure 2 shows an example of a representation of a high energy physics event in PXL.



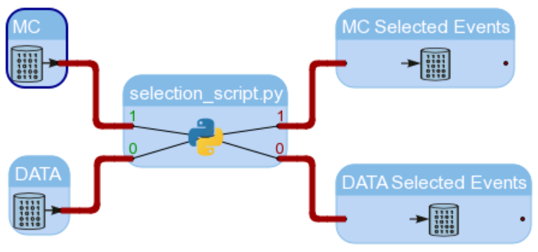
(a)

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 Python 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 SWIG [6].

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

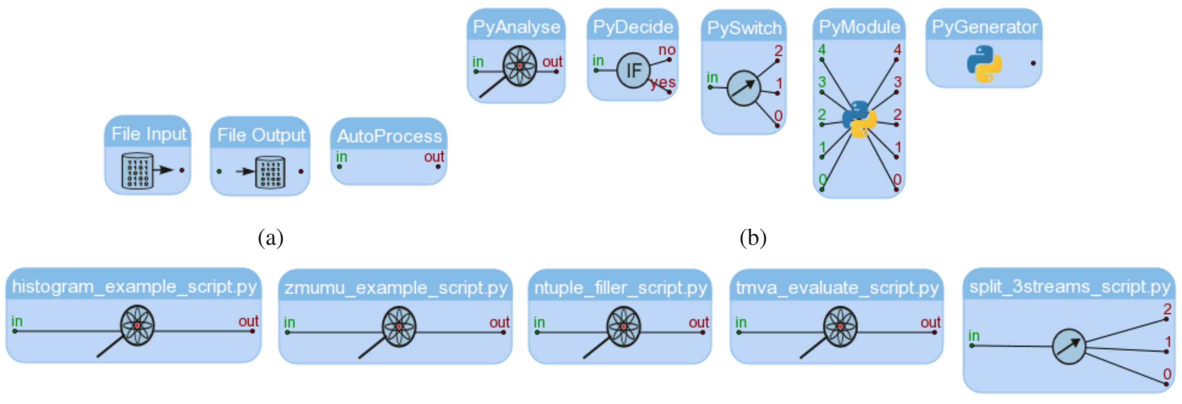
PXL provides a **framework for modular physics analyses**. Each module can have multiple sinks and sources to control the data flow. Figure 3 shows an example of a simple analysis in PXL. The data flow is from left to right, starting from an input or generator module, and visualized by connection lines. A common interface is defined for data exchange between all modules. 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 batch using the executable pxlrun and interactively using the graphical user interface of VISPA.



(a)

Figure 3: Analysis which selects events from data and MC input using the same module and writes them out to two files.

PXL delivers a variety of **module interfaces and examples**. 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 using PyROOT [2]).

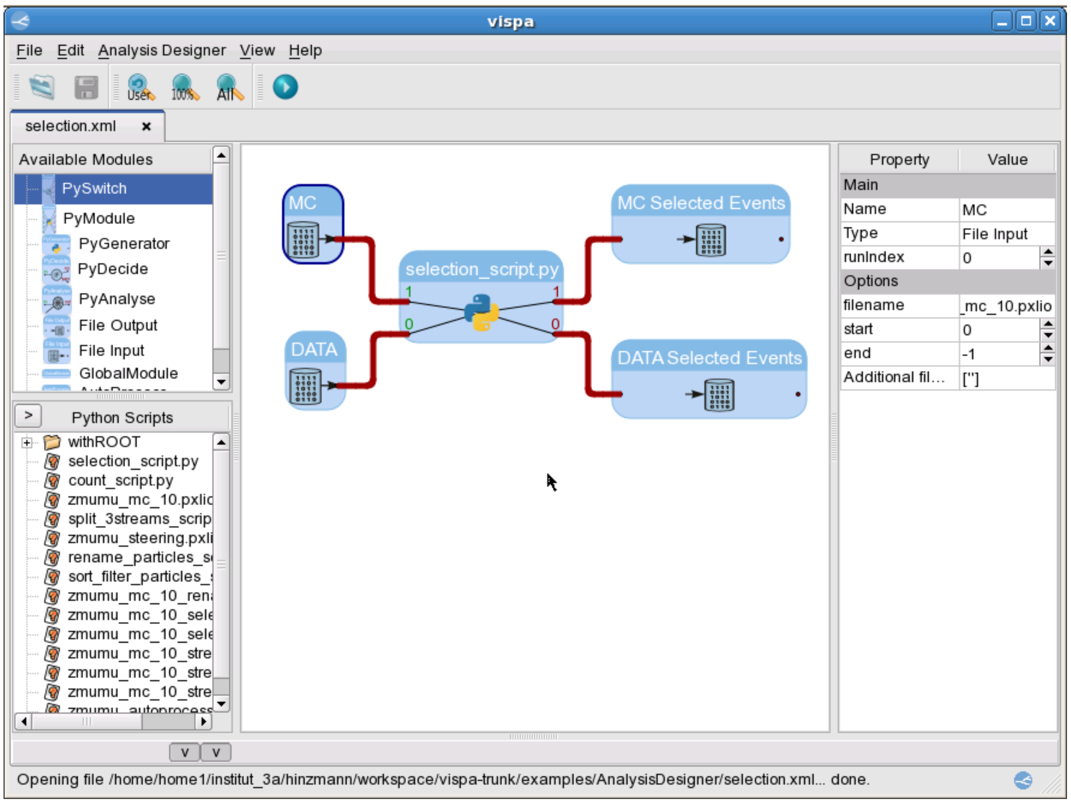


(c)

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

3. Visual development of physics analyses

VISPA supplies a graphical user interface for **visual development of analyses** which is shown in Figure 5. It lists all available C++ and Python modules which can be inserted into the analysis using drag and drop. Connection lines between modules are also drawn via drag and drop. The parameters of modules are modified using the right part 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 described in the following.



Property	Value
Main	
Name	MC
Type	File Input
runIndex	0
Options	
filename	_mc_10.pxlio
start	0
end	-1
Additional fil...	['']

Opening file /home/home1/institut_3a/hinzmann/workspace/vispa-trunk/examples/AnalysisDesigner/selection.xml... done.

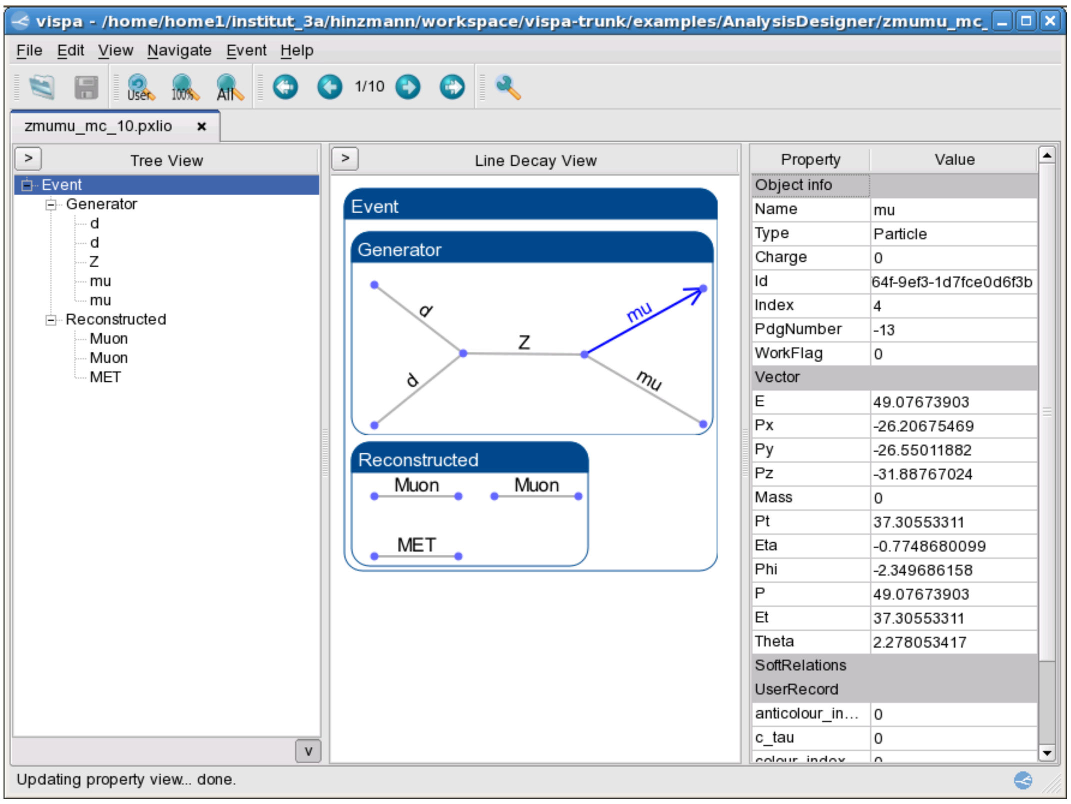
(a)

Figure 5: Screen shot of the Analysis Designer

The data browser of VISPA, shown in Figure 6, gives a **visual representation of all data** in a PXL data file. It is able to draw decay trees according 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 on an event by event basis. Further the data browser can be used as an editor for event templates, e.g. as input for the Autoprocess module as a steering file for the decay tree reconstruction.

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

In physics analyses it is essential to be able to easily share work. An important feature of



Property	Value
Object info	
Name	mu
Type	Particle
Charge	0
Id	64f-9ef3-1d7fce0d6f3b
Index	4
PdgNumber	-13
WorkFlag	0
Vector	
E	49.07673903
Px	-26.20675469
Py	-26.55011882
Pz	-31.88767024
Mass	0
Pt	37.30553311
Eta	-0.7748680099
Phi	-2.349686158
P	49.07673903
Et	37.30553311
Theta	2.278053417
SoftRelations	
UserRecord	
anticolour_in…	0
c_tau	0

```
Updating property view... done.
```

(a)

Figure 6: Screen shot of the PXL Data Browser

VISPA is the **portability of analyses**. The well defined interface for modules allows to share and reuse common modules within a group. VISPA further allows to exchange complete analyses by an automated tar-ball creation integrated in the graphical user interface. The exchange between different platforms is also possible. Therefore the platform independent application framework PyQt4 [11, 12] was chosen as a base for the graphical user interface. VISPA is available on all common platforms (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 easy maintainability and low error rate for PXL. The study has also triggered several performance improvements in particular in the field of I/O.

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 installation is needed and it can be easily used in teaching.

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 make it a powerful tool for collaborative work on physics analysis on any degree of complexity.

References

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

[2] *ROOT - An Data Analysis Framework*, http://root.cern.ch/.

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

[4] M. Erdmann, P. Schiffer *Measuring Cosmic Magnetic Fields with Ultra High Energy Cosmic Ray Data*, accepted by *Astroparticle Physics* (2010) [arXiv:0904.4888]

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

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

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

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

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

[10] O.Actis et al. *Visualization of the CMS Python Configuration System* in proceedings of *17th Int. Conf. Computing in High Energy and Nuclear Physics (CHEP 2009), Prague, Czech*, 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.