Visual physics analysis – from desktop to physics analysis at your fingertips

H-P Bretz, M Erdmann, R Fischer, A Hinzmann, D Klingebiel, M Komm, J Lingemann, M Rieger, G Müller, J Steggemann, T Winchen

RWTH Aachen University, III. Physikalisches Institut A, 52062 Aachen, Germany

E-mail: `rfischer@physik.rwth-aachen.de`

Abstract. Visual Physics Analysis (VISPA) is an analysis environment with applications in high energy and astroparticle physics. Based on a data-flow-driven paradigm, it allows users to combine graphical steering with self-written C++ and Python modules. This contribution presents new concepts integrated in VISPA: layers, convenient analysis execution, and web-based physics analysis. While the convenient execution offers full flexibility to vary settings for the execution phase of an analysis, layers allow to create different views of the analysis already during its design phase. Thus, one application of layers is to define different stages of an analysis (e.g. event selection and statistical analysis). However, there are other use cases such as to independently optimize settings for different types of input data in order to guide all data through the same analysis flow. The new execution feature makes job submission to local clusters as well as the LHC Computing Grid possible directly from VISPA. Web-based physics analysis is realized in the VISPA@Web project, which represents a whole new way to design and execute analyses via a standard web browser.

1. Introduction

Visual Physics Analysis (VISPA) is an integrated analysis development environment for high-energy and astroparticle physics [1, 2]. The goal is to provide one environment that addresses the typical cycle of designing, executing, and verifying of analyses.

In VISPA, analyses are designed and represented visually as an analysis flow. This flow consists of individual modules and connections among them via a variable number of input and output ports. During the execution, data sets are guided through the flow on an event by event base. Depending on internal criteria, user programmed modules decide to which output port the current event will be propagated. Along with the attached connection to the respective port, this also determines the next module to handle the event. Once an event is not forwarded to another module anymore, the analysis loop continues with the next event in the data set.

The modular structure simplifies the exchange of whole analyses or even parts of it between physicists due to the clear module interface and the fact that modules are independent from each other. In order to make the design process more flexible, VISPA supports C++ and Python modules. While C++ offers better runtime performance, Python modules are faster to prototype due to advantages of interpreted code compared to compiled code. Part of the flexibility is also the option to include arbitrary external packages.

Analyses in VISPA employ three main paradigms, see figure 1. Within modules, physics objects are accessible via the physics library PXL [3,4] in an object-oriented manner. The analysis flow based on the module system implements a data-flow driven paradigm. Visual programming and steering are provided within one integrated graphical user interface (GUI).

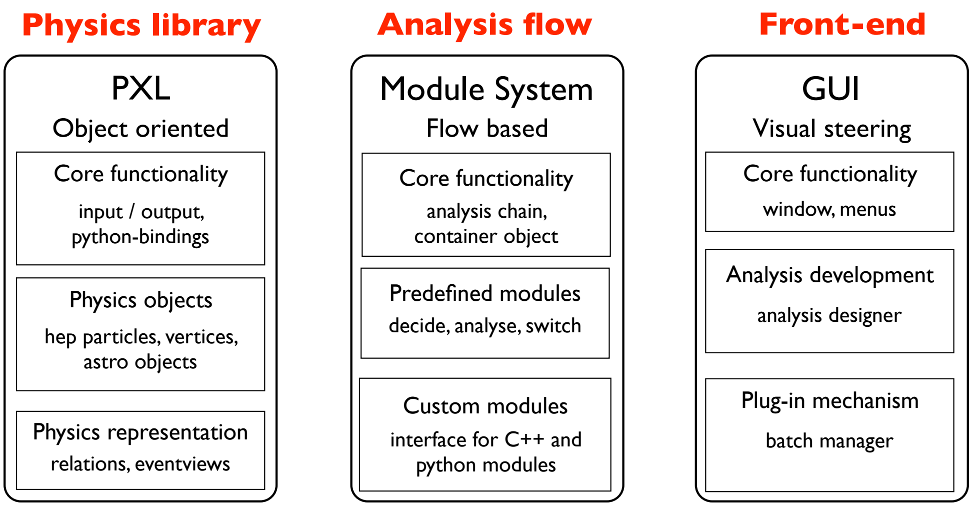


Figure 1. VISPA is based on three main components. The physics library PXL, the module system, and the GUI are summarized along with some of their core features.

In addition to the the physics objects, PXL also manages relations among these objects and handles input / output operations. Part of the module system are a container object and the analysis chain itself, through which the container is guided. It also contains predefined modules, e.g. a simple switch module, which can be used to split a data stream. User interfaces to C++ and Python build the foundation for user-written modules. The graphical front-end is written in Python [5] and uses the Qt framework [6] with the Python-bindings PyQt [7]. While the core part only provides common GUI elements, such as menus, toolbars and dialogs, the user interacts with plug-ins that are loaded into the GUI. Subsequently, two plug-ins, which provide core functionality to the analyst, are presented, see figure 2.

The analysis designer is the tool used to create and configure the analysis flow.

The PXL data browser visualizes data files event by event in order to verify them.

VISPA has been successfully used for published analyses in high-energy and astroparticle physics [8, 9].

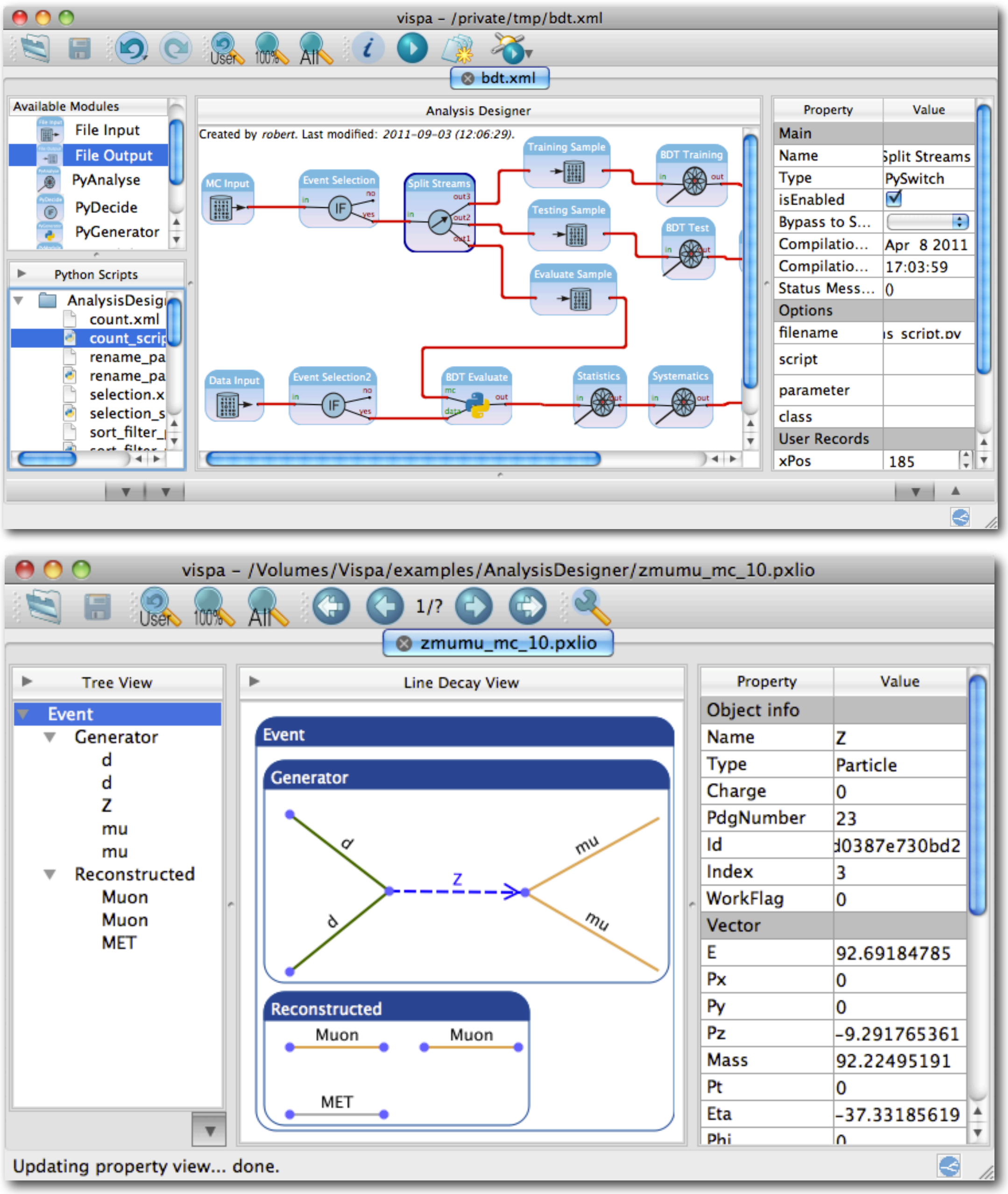
As outlined so far, VISPA integrates the analysis design and verification steps of the development cycle. In the following section, the project's scope is extended to further improve analysis handling and comfortably integrate the execution step. Finally, with VISPA@Web a new perspective to visual physics analysis via a web-interface is opened.

2. New developments

2.1. Improved handling of complex analysis designs

Analyses often contain steps that the analyst would like to execute independently. There might also be parts of the analysis that should run several times with only slightly different configuration, e.g. for different types of data sets. These are two examples of complex problems during the analysis process that could be addressed by creating separate analysis configurations. In order to integrate such cases into one design, we introduce a new layering technique into VISPA.

The idea is to create one analysis flow as usual, however, containing all steps. This analysis flow is regarded as the base layer. New layers can be created upon the base layer and will automatically inherit all settings from the underlying layer. Within each layer, module options can be changed and modules



Created by *robert*. Last modified: *2011-09-03 (12:06:29)*.

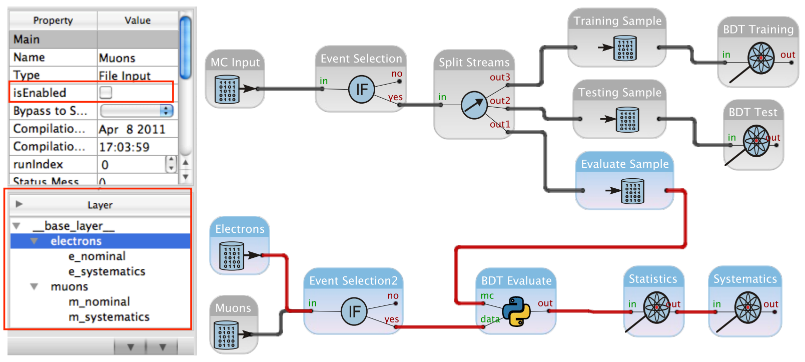
Property	Value
Main	
Name	Split Streams
Type	PySwitch
isEnabled	✓
Bypass to S...	
Compilatio...	Apr 8 2011
Compilatio...	17:03:59
Status Mess...	0
Options	
filename	s_script.py
script	
parameter	
class	
User Records	
xPos	185

Property	Value
Object info	
Name	Z
Type	Particle
Charge	0
PdgNumber	23
Id	d0387e730bd2
Index	3
WorkFlag	0
Vector	
E	92.69184785
Px	0
Py	0
Pz	-9.291765361
Mass	92.22495191
Pt	0
Eta	-37.33185619
Phi	0

Updating property view... done.

Figure 2. Top: The picture shows the analysis designer, which consists from left to right of a list of all available analysis modules, the workspace where the actual analysis flow is designed and visualized, and a property view to configure options of the selected module. Bottom: Here, the PXL data browser is depicted showing a physics event. On the left-hand side there is a tree view representation, while the center shows a Feynman-like diagram. The property view shows detailed information about a selected physics object.

can be enabled, disabled, and bypassed. By just leaving a subset of all modules active, while all other modules are deactivated, the first example is covered. In the case of the second example, certain module options could be changed to reflect the requirements of a certain data set. For the execution, the analyst selects the layer that should be run. Figure 3 shows an example analysis flow using layers.



Property	Value
Main	
Name	Muons
Type	File Input
isEnabled	☐
Bypass to S...	
Compilatio...	Apr 8 2011
Compilatio...	17:03:59
runIndex	0
Status Mess	0

Layer
- __base_layer__
  - electrons
    - e_nominal
    - e_systematics
  - muons
    - m_nominal
    - m_systematics

Figure 3. Left: The property view of the analysis designer is extended by a layer control box in the bottom. Right: An analysis flow with activated and deactivated (greyscale) modules. The currently active analysis path is realized as a layer.

2.2. *Convenient analysis execution*

Analyses are executed in different computing environments. While basic tests may be run locally on a desktop or laptop computer, there might be resource requirements, e.g. computing power or storage needs, that can only be covered by batch systems. VISPA provides a batch system plug-in making execution in these different environments transparently accessible. Based on an analysis developed with the analysis designer, a batch job is created. At this stage it is possible to override module options, e.g. to scan over ranges or a list of parameters. Use cases are for instance to try different cut values or to give a list of input files.

Once the job is configured, it is submitted to one of VISPA's batch managers. For each supported back-end batch system, a separate batch manager exists. These managers make sure to configure the batch job correctly for the corresponding back-end. Currently, a local batch manager, a condor batch manager and a grid batch manager are included. The local batch manager executes the analysis on the local machine running VISPA. The condor batch manager supports the condor batch system [10], while the grid batch manager submits jobs to the world wide computing grid [11]. The mechanism described allows the user to design the analysis and the batch job independent from the desired back-end. Submitting the job is the only part of the mechanism in which the back-end becomes important.

2.3. *Web-based physics analysis*

Software packages need to be installed and maintained. This has two consequences. First of all, the packages are available only on computers where they are installed. Secondly, the software has to be kept up-to-date on each machine. A complete new perspective of accessing physics analysis is VISPA@Web, which is the realization of the VISPA concept accessible via a web browser. Since VISPA@Web follows a server-client approach, the software has to be installed once only to serve several users. At the same time, each analysis is stored on the server and is, therefore, accessible from any computer connected to the internet without special requirements on the client side. While it is already fairly easy to exchange analysis modules with VISPA, this is even easier for analysts who use the same server. Thus, collaboration efforts are further enhanced.

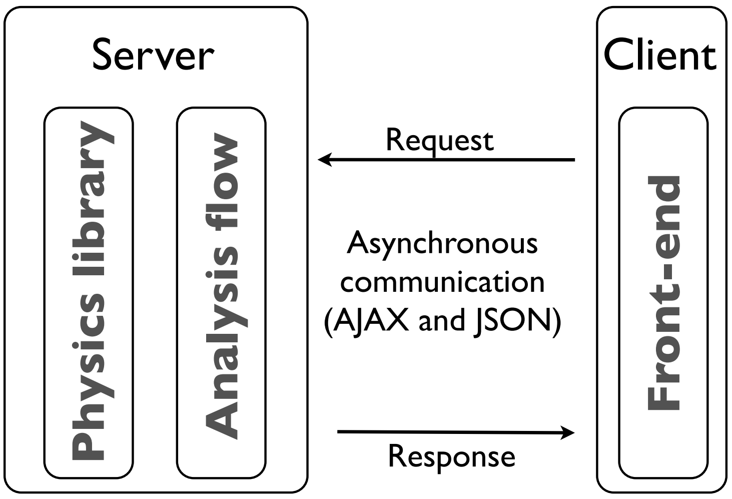
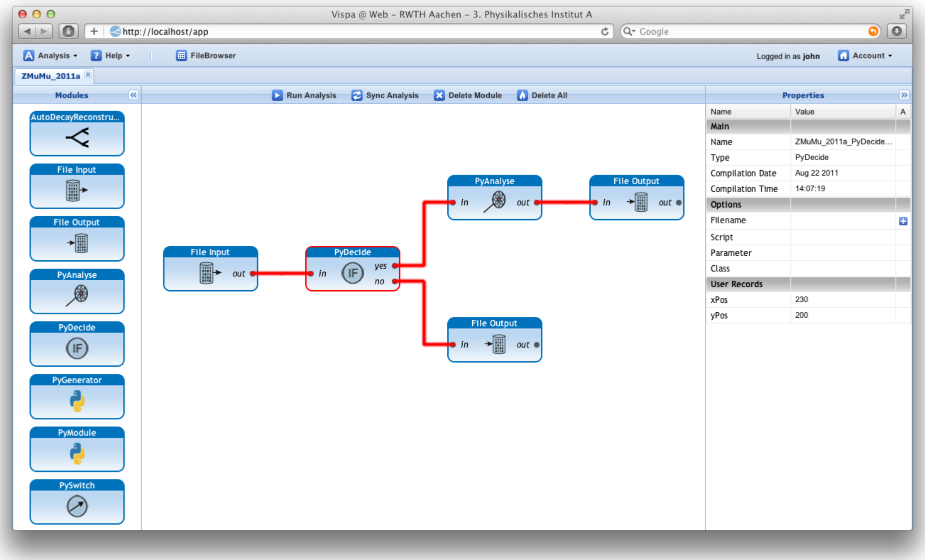


Figure 4. VISPA@Web consists of the same components as the desktop application as seen in figure 1. Here these components are devided between server and client side.

VISPA@Web employs the same components as the desktop version, see figure 4. However, while the front-end is the web interface, the physics library and analysis flow run on the server. Server and client communicate in an asynchronous manner using the AJAX technique [12]. Therefore, the web page does not have to be completely reloaded each time the user interacts with the front-end. The interactions are rather transferred to the server in the background in the JSON format [13] and are immediately applied to the analysis design. The user can just shut down his browser and reconnect, even from a different computer, and will find all of his analyses in the same state as when he quit the session before.



Name	Value	A
Main		
Name	ZMuMu_2011a_PyDecide...	
Type	PyDecide	
Compilation Date	Aug 22 2011	
Compilation Time	14:07:19	
Options		
Filename		
Script		
Parameter		
Class		
User Records		
xPos	230	
yPos	200	

Figure 5. VISPA@Web screenshot. The interface is designed to resemble the desktop application in order to enable users familiar with VISPA to get started immediately.

Besides AJAX, which is provided by the ExtJS package [14], VISPA@Web also uses other up-to-date web technologies. The front-end uses HTML5 [15] and CSS [16].

Currently, VISPA@Web is in a beta phase. Basic analyses can be designed and executed. The next steps planned are to perform a multi-user field test and to integrate the aforementioned batch system plug-in in order to allow the system to scale with the number of users.

3. Summary

VISPA is a well-tested analysis development environment with applications in high-energy and astroparticle physics. Within the project, new technologies are explored. Layers unify and simplify the handling of likewise analysis steps while they increase the flexibility at the same time. The integrated batch system plug-in makes job submission to multiple back-ends effortless. Both layers and the batch system are consequent improvements based on the foundations of VISPA. VISPA@Web represents a new perspective and demonstrates that web-based physics analysis is possible.

References

[1] Actis O, Erdmann M, Fischer R, Hinzmann A, Kirsch M, Klimkovich T, Müller G, Plum M and Steggemann J 2008 (*Preprint* `0810.3609v1`) URL `http://arxiv.org/abs/0810.3609v1`

[2] Brodski M *et al.* 2010 *Proceedings ACAT2010, Jaipur, India, PoS(ACAT2010)* **064**

[3] 2011 Physics eXtension Library (PXL) URL `http://pxl.sourceforge.net`

[4] 2010 PXL Class Documentation URL `http://pxl.sourceforge.net/doxygen3a`

[5] 2011 Python Programming Language URL `http://www.python.org`

[6] 2008 Qt - A cross-platform application and UI framework URL `http://www.qtsoftware.com`

[7] 2010 Pyqt URL `http://www.riverbankcomputing.co.uk`

[8] Erdmann M and Schiffer P 2010 *Astropart. Phys.* **33** 201 (*Preprint* `0904.4888`)

[9] 2011 PARametrized Simulation Engine for Cosmic rays (PARSEC) URL `http://www.physik.rwth-aachen.de/parsec`

[10] Thain D, Tannenbaum T and Livny M 2005 *Concurrency - Practice and Experience* **17** 323–356

[11] Donno F and Litmaath M 2008 Data management in WLCG and EGEE. Worldwide LHC Computing Grid Tech. Rep. CERN-IT-Note-2008-002 CERN Geneva

[12] Garrett J J 2005 Ajax: A new approach to web applications URL `http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications`

[13] Crockford D The application/json media type for javascript object notation (json) URL `http://tools.ietf.org/html/rfc4627`

[14] Sencha Inc 2011 JavaScript Framework for Rich Apps in Every Browser URL `http://www.sencha.com/products/extjs/`

[15] Hickson I *et al.* Html5 - a vocabulary and associated apis for html and xhtml URL `http://www.w3.org/TR/html5/`

[16] Bos B, Lie H and Lilley C The text/css media type URL `http://tools.ietf.org/html/rfc2318`