Journal of Physics: Conference Series **331** (2011) 072056 doi:10.1088/1742-6596/331/7/072056

Visual Physics Data Analysis in the Web Browser

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

RWTH Aachen University, Physikalisches Institut 3A, 52062 Aachen, Germany

E-mail: erdmann@physik.rwth-aachen.de

Abstract. The project VISPA@WEB provides a novel graphical development environment for physics analyses which only requires a standard web browser on the client machine. It resembles the existing analysis environment available from the project Visual Physics Analysis VISPA, including the connection and configuration of modules for different tasks. High level logic can be programmed using the Python language, while performance-critical tasks can be implemented in C++ modules. The use cases range from simple teaching examples to highly complex scientific analyses.

1. Introduction

In the last couple of years web based applications, e.g. online steet maps or modern web based mail clients, became increasingly popular. Compared to conventional desktop applications they have two major advantages. There is no need to install the software locally, and the applications have access to all the resources provided by the server and the connected computing center. Both advantages apply to analysis software as well. Experiment specific software for example can have special requirements, both in software as in hardware, and therefore can be hard to install on the physicists laptop. Nevertheless it is often convenient to have access to these resources over the internet as many collaborations are spread around the world. Established solutions like SSH and VNC have some limitations, e.g., high bandwith requirements or the need for system accounts. Especially the possibility to grant access without system accounts allows setups for teaching or demonstration purposes where non privileged users need access to these resources. As a result of the ongoing competition for the fastest JavaScript [1] engine between web browser developers, we now have a very fast and efficient language at hand which allows complex applications to be executed within the web browser. Together with many mature and feature rich open source libraries the effort required to implement highly interactive graphical user interfaces is minimized. Using these technologies we have developed a web based application prototype as competitive reimplementation of the VISPA [2, 3] desktop application. The VISPA project constitutes a development environment for data analysis, e.g., in high energy physics. In the following sections we first introduce VISPA and the underlying analysis framwork PXL [4, 5] and we then describe the main features of the web based analysis software VISPA@WEB [6].

2. VISPA and PXL

VISPA (Visual Physics Analysis) is a graphical development environment which enables physicists to prototype, execute, and verify data analysis of any complexity. The key idea

of VISPA is to develop physics analyses in a flow based manner, where modules with incoming and outgoing data ports are connected and configured. It provides a multipurpose window with visual tools to design and execute modular analyses, create analysis templates, and browse physics event data. VISPA and PXL are utilized in high energy and astroparticle physics and may be extended to other fields of research. The underlying framework PXL (Physics eXtension Library) is a C++ class collection for high level analyses, with specializations for high energy and astroparticle physics. It is intended for experiment independent data handling and has been continuously developed since 2006 as the successor of the PAX toolkit [7]. The library includes classes representing physics objects, convenience containers, a robust and fast I/O format and a complete Python [8] interface. Additionally it provides the infrastructure for module based analyses. This module system is optimized for a flow based analysis design typically found in particle physics analyses where each event is processed individually.

3. Client-Server Architecture

VISPA@WEB is developed using the typical client server architecture (Fig. 1). All data and analyses are located on the server while the client requests information for each element of the analysis when needed, that way only the information needed for presentation is send to the client using AJAX, Asynchronous JavaScript and XML [9]. The server encodes the requested information into JSON (JavaScript Object Notation) [10], a text based data format, and sends it to the client. All changes the user makes to the analysis are as well transmitted to the server via JSON. Those changes are then validated and applied to the actual analysis by the server. The server is programmed in Python using the built-in webserver and the PXL Python

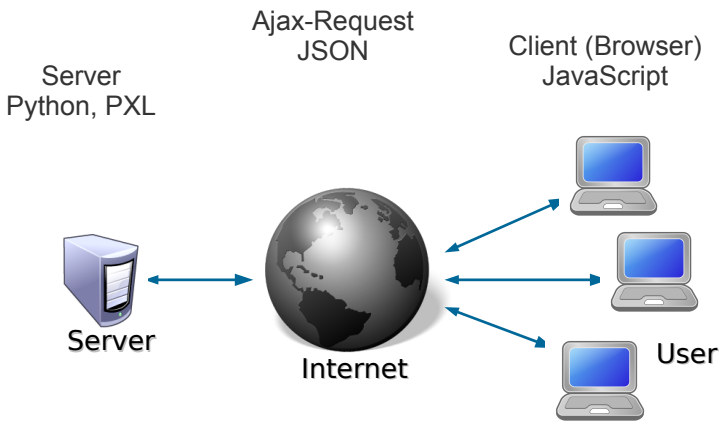

Figure 1. Client Server architecture implemented in VISPA@WEB.

bindings. Essential systems are the components that associate analysis instances with users and that handle the concurrent execution of analyses. The client is programmed using HTML/CSS and JavaScript, facilitating several JavaScript libraries.

4. Virtual Filesystem

VISPA@WEB implements a virtual fileystem where each user has read/write access to his own private folder and read only access to global data. For convenience and for security reasons the

user does not have access to the complete servers filesystem. Instead every user has his own working directory, where he can upload data and source files to. Also stored in this directory are source files created with the offered online editor, as well as output files of the analyses executed on the server. Additionally the user has read access to a global directory containing scripts and data shared by all users. All files can be downloaded to the local computer.

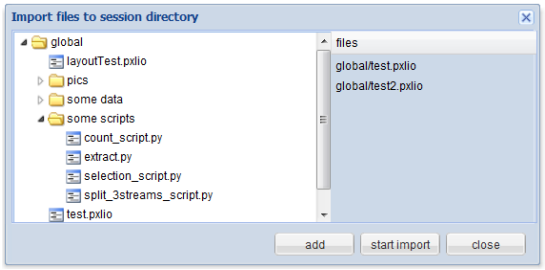

Figure 2. Files from the shared global folder are imported into the user folder.

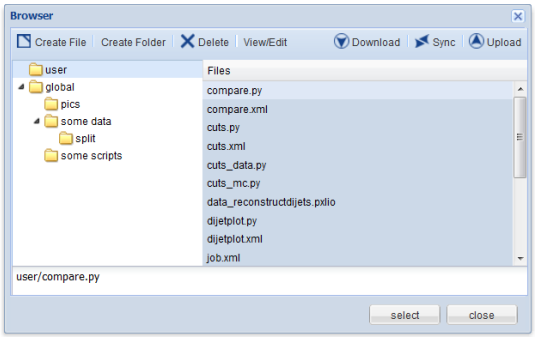

Figure 3. The file open dialog showing the user folder and the global folder.

5. Designing a Physics Analysis

A central functionality of VISPA@WEB is designing physics analyses. It therefore resembles most of the features and concepts known from the VISPA Qt [11] based desktop application where instead of textual programming the analysis flow is designed by connecting and configuring different modules. On the left side of the window (Fig. 4), a list with the available modules is

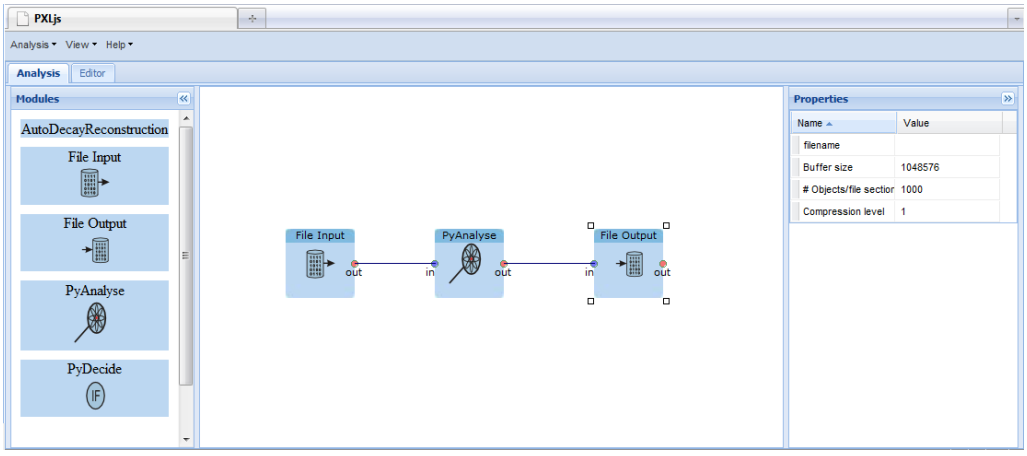

Figure 4. Designing analysis with VISPA@WEB

displayed. This list is dynamically loaded from the server. Modules can be added via drag and drop to the current analysis, displayed in the center of the window. Each module has sinks, for incoming data elements, and sources, for outgoing data elements. Each source can be connected to one sink while each sink can have multiple incoming connections. The data elements are created in special input modules and then sequentially passed to the connected modules. On

the right side of the window the properties of a module selected by the user are listed. They can be modified and are synchronized with the server. For options representing filenames a file selection dialog showing the virtual filesystem on the server is provided.

6. Editing Analysis Scripts

In order to cope with complex and innovative physics analyses, we incorporated high level scripting modules which can be edited by the user (Fig. 5). These scripts are written in the Python programming language, and can be edited online with the provided browser based editor. In these scripts PXL as well as ROOT [12] and every other software which is available in Python can be used. The major advantages are the platform independence and the simplicity of the Python language, so custom analysis steps can be added easily.

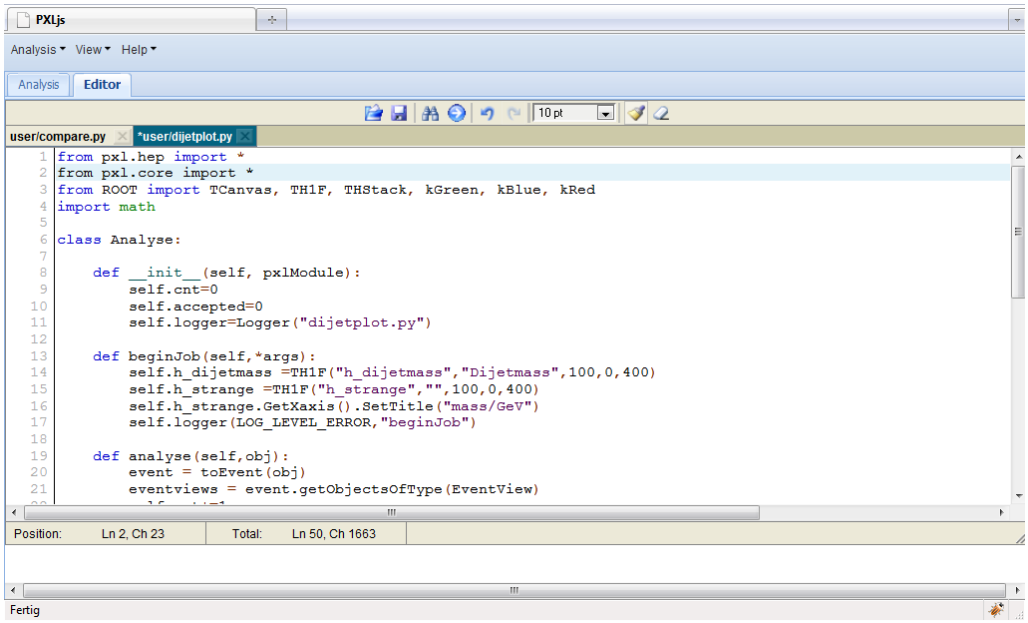

Figure 5. Script editor of VISPA@WEB

7. Browsing and Verifying Data

Another important aspect of visual physics analysis is the visualization of data. In many situations the physicist needs to know the exact contents of data files. Therefore VISPA@WEB allows users to quickly browse the input and output files of the analysis (Fig. 6). In the left column of the window the list of objects contained in the current event is shown. In the center view a visual representation of the selected object is presented, where particle decays are layouted automatically. On the right column the properties of each selected object are listed. Only the data for the currently displayed event is loaded. In this way it is possible to open files of arbitrary size. PXL I/O files have the unique feature to be readable even if the analysis does not finish successfully.

8. Conclusion

We presented an experiment independent analysis software based on modern web technologies. In its current state it resembles the core features of the desktop equivalent VISPA and allows users to design and execute a simple high energy physics analysis.

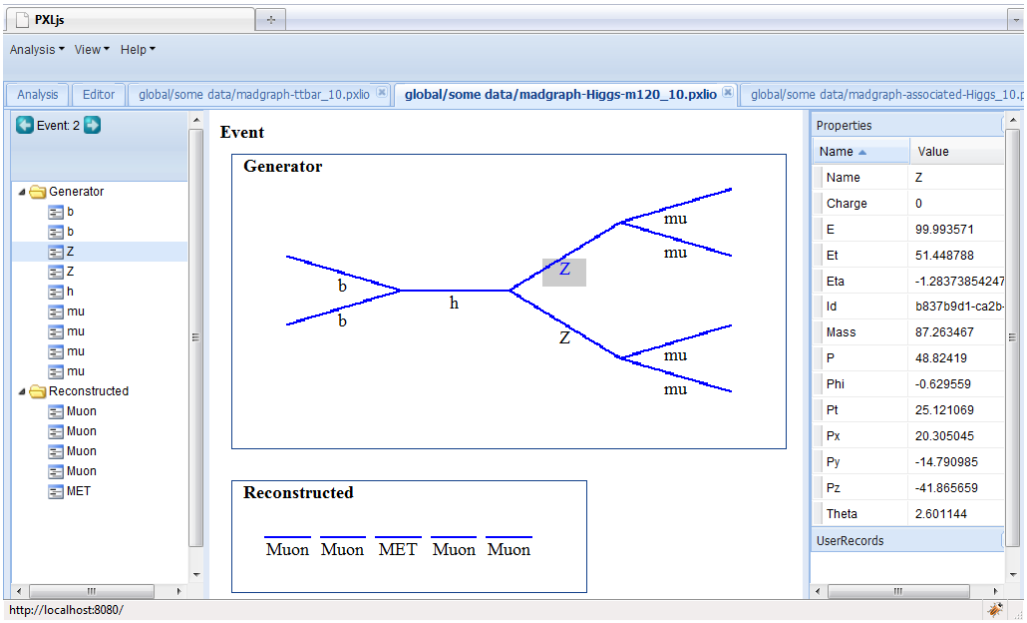

Figure 6. Browsing data files with VISPA@WEB

9. Acknowledgements

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

References

[1] ECMAScript language, ECMA-262 standard (Edition 3), `http://www.ecmascript.org/`
[2] VISPA (Visual Physics Analysis), `http://vispa.sourceforge.net/`
[3] M. Brodsky et al. Visual Physics Analysis - Applications in High Energy and Astroparticle Physics, Proc. of 13th Int. Workshop on Adv. Comp. and Anlys. Techn. in Phys. Res. (ACAT2010), Jaipur, India, Feb. 2010
[4] PXL (Physics Extension library), `http://pxl.sourceforge.net/`
[5] 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]
[6] M. Komm Development of the VISPA@WEB Program and Measurement of the Dijet Mass with CMS, Jul. 2010, B.Sc. thesis, RWTH Aachen University
[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] G. van Rossum et al. Python Language Website, `http://www.python.org/`
[9] J. J. Garrett Ajax: A New Approach to Web Applications, Feb 2005, `http://www.adaptivepath.com/ideas/essays/archives/000385.php`
[10] D. Crockford RFC 4627 The application/json Media Type for JavaScript Object Notation (JSON), `http://www.ietf.org/rfc/rfc4627.txt`
[11] Nokia Corporation Qt 4.6 Whitepaper, `http://developer.qt.nokia.com/wiki/QtWhitepaper`
[12] R. Brun, F, Rademakers ROOT - An Object Oriented Data Analysis Framework, Proc. AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81-86. `http://root.cern.ch/`.