

# **VISPA: a Novel Concept for Visual Physics Analysis**

**Tatsiana Klimkovich** for the **VISPA** group

(O.Actis, M.Erdmann, R.Fischer, A.Hinzmann, M.Kirsch,  
G.Müller, M.Plum, J.Steggemann)

ACAT2008, Erice, November 2008

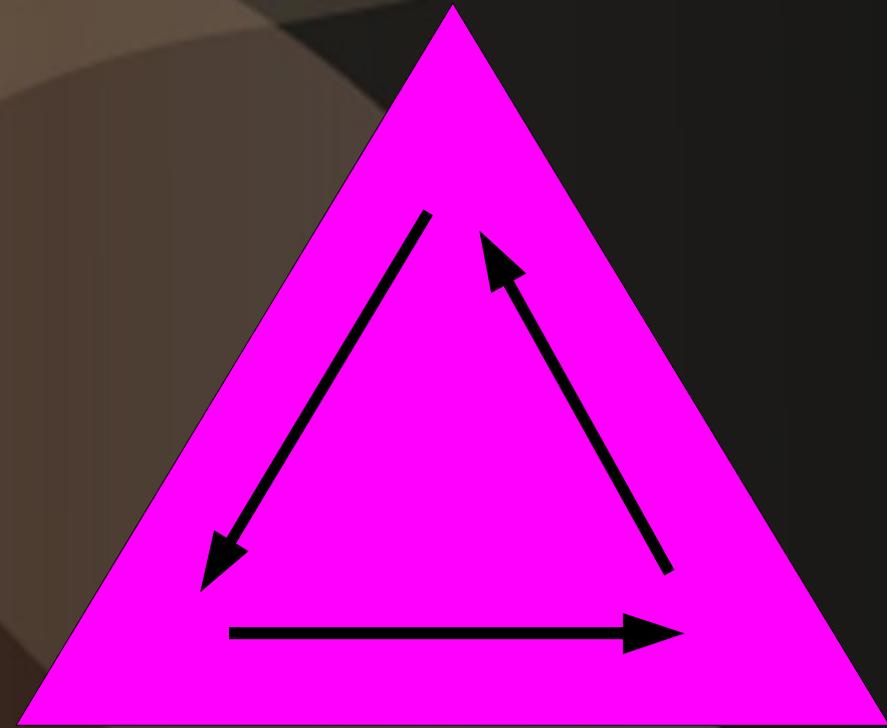
# Contents

- **Physics Analysis in High Energy Physics experiment**
- **Novel Concept for HEP analyses: VISPA**
- **Look & feel with analysis example**
- **VISPA key ingredients:**
  - **PXL: underlying functionality**
  - **Python interface to C++ functionality**
  - **Module steering system**
  - **Autoprocess: automatic decay chain reconstruction**

# High Energy Physics Analysis



**Prototyping  
(design)**



**Execution  
(steering)**

**Verification**

# High Energy Physics Analysis

- During last years big achievements have been done in developing analysis software for the experiments
- Experiments have different software frameworks e.g. **H100** in H1, **CMSSW** in CMS, **ATHENA** in ATLAS etc.
- On top of them **more analysis specific software** has been developed and used:
  - Requires less time to start
  - The output files are smaller
  - Reduce time for making analysis
  - Allows to perform analysis on the laptop

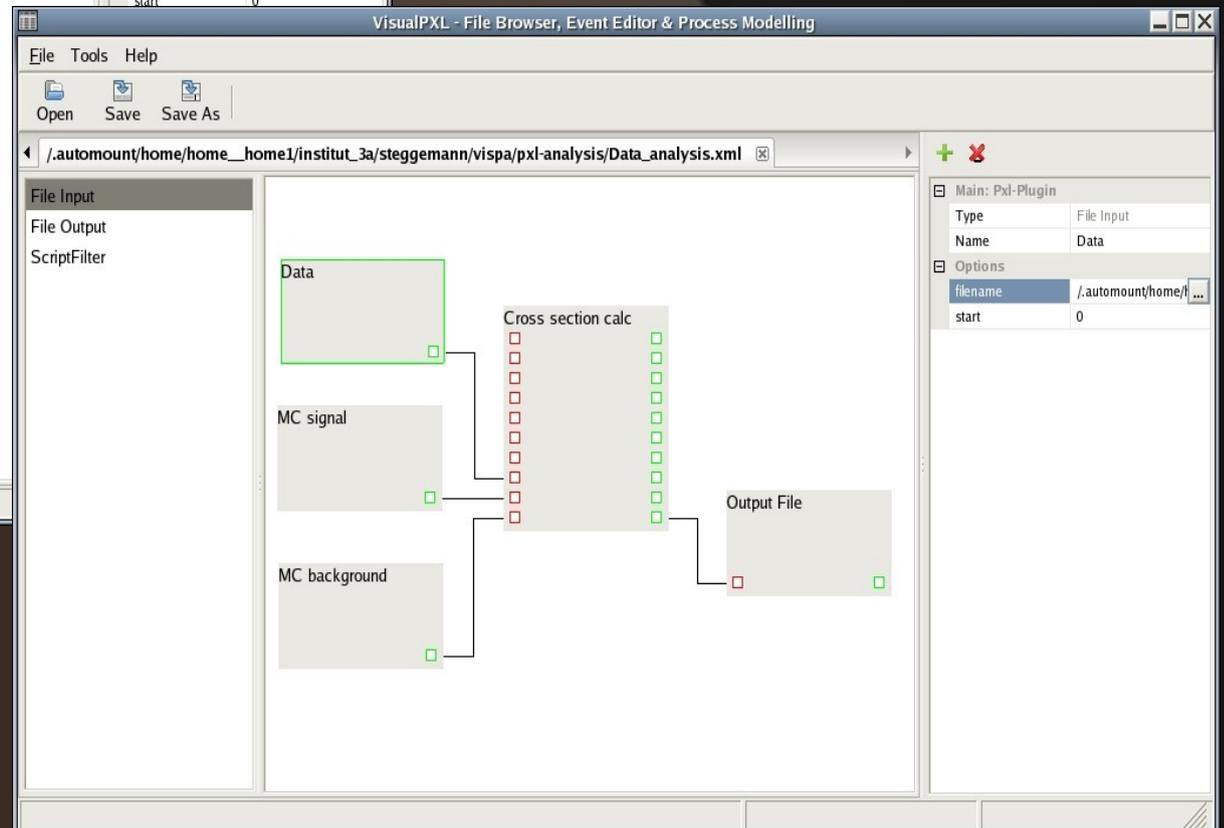
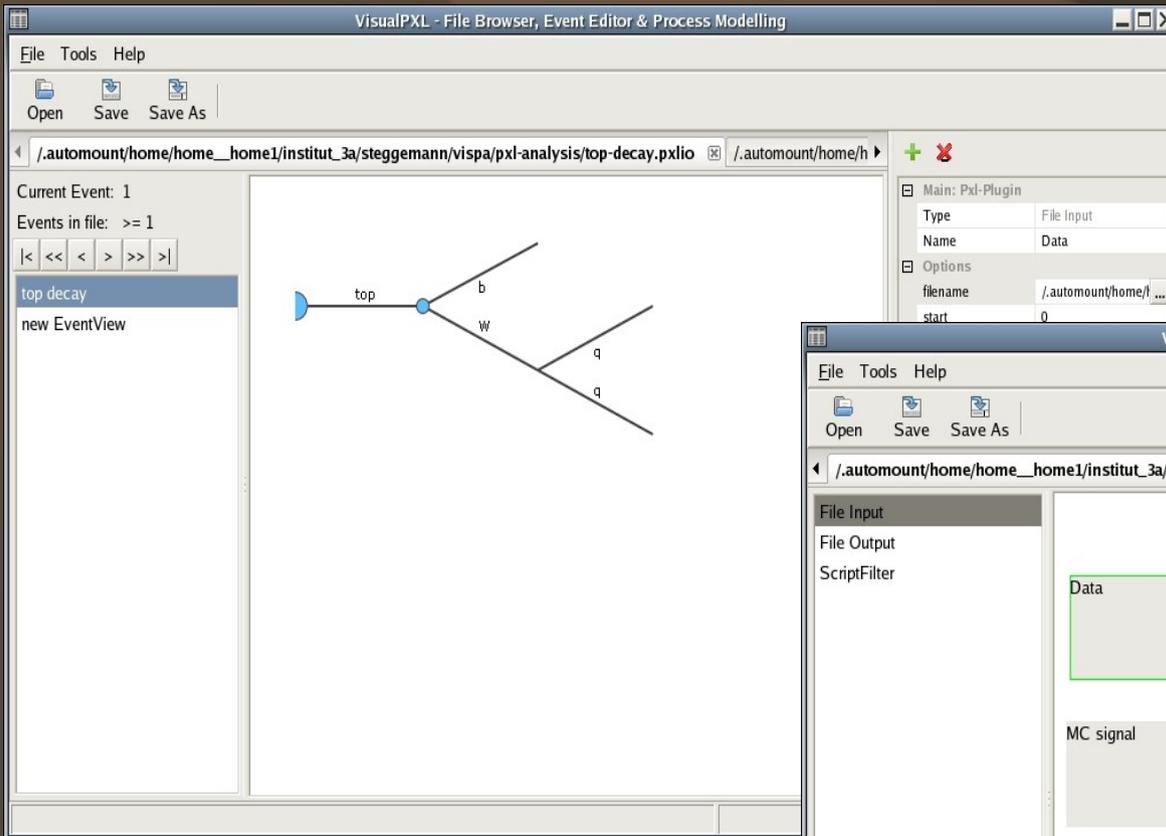
# The wish list of the analyser

- To have an **easy way to develop analysis**
- To start fast
- To dedicate **minimal time for learning**
- To have **modular structure of the software**
- To have many **reusable components**
- To have **small summary data sets (ntuples)**
- To have **clear picture of what you are doing**
- To **transmit your knowledge to other people**

# VISPA: Visual Physics Analysis

## Novel Concept of making physics analysis

Mixture of graphical and textual work (like LabView)



# VISPA: Main Features

**Development environment for HEP analyses  
(first prototype)**

- **Combines graphical and textual steering**
- **Module steering**
- **Works for any experiment**

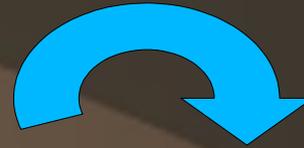
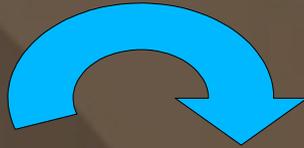
# Structure of Physics Analysis

Data input

Data selection

Advanced analysis

Histograms



VisualPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

/home/tklimk/soft/vispa/talk/Zmumu\_out.pxlio

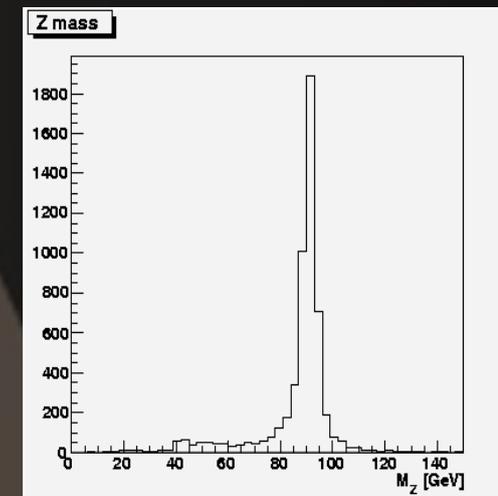
Current Event: 10  
Events in file: >= 10

Reconstructed  
Generator

Diagram labels: Z, Muon, Muon, Jet, Jet, MET

Main: EventView

Main: EventView	
Id	aa5b50b9-652e-664t
Name	Reconstructed
UserRecords	
NumEle	0
NumJet	2
NumMET	1
NumMuon	2
Process	Zmumu
Type	Rec
missing_px_in	-24.8997
missing_py_in	21.79567



# Multipurpose Window

VisualPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

/.automount/home/home\_\_home1/institut\_3a/steggemann/vispa/pxl-analysis/top-decay.pxlio

Current Event: 1  
Events in file: >= 1

top decay  
new EventView

Graphical window

Main: Pxl-Plugin

Type	File Input
Name	Data

Options

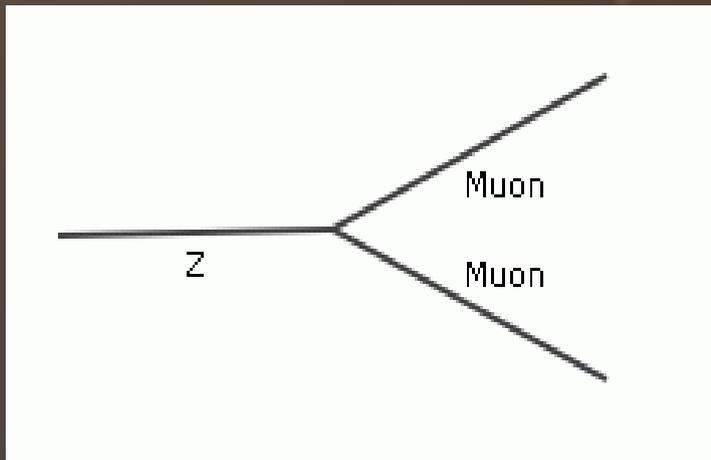
filename	/.automount/home/h ...
start	0

Property window

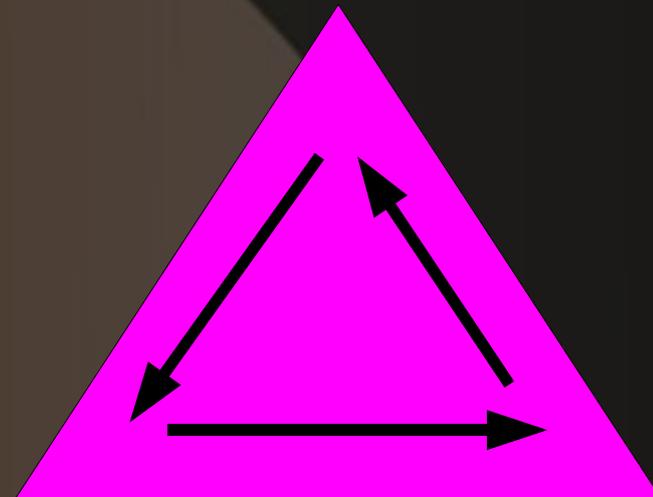
# Closer consideration:

Very simple  
example analysis

$Z \rightarrow \mu \mu$



Prototyping



Execution

Verification

Sample at LHC energies

# Z boson reconstruction from muons

## First step: inspect an input file

Prototyping



Execution

Verification

VisualIPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

/home/tklimk/soft/vispa/vispa\_files/Zmumu\_sample.pxlio

Current Event: 1  
Events in file: >= 3

Reconstructed  
Generator

Muon Muon  
Jet Jet  
Jet MET

Main: Particle

Id	92c558db-019f-564
Name	Muon
Charge	0
ParticleId	0

Vector

Energy	38.837048
Px	-6.692826
Py	-29.411932
Pz	-24.463226
Pt	30.163814
Mass	0.10565
Phi	-1.794541
Eta	-0.741244

UserRecords

calIso	0.664851
ecalIso	0
hcalIso	0.664851
leptonID	0
resolutionA	0.011726
resolutionB	0.009914
resolutionC	0.005066
resolutionD	0.011744
resolutionEt	0.356284
resolutionEta	0.000351
resolutionPhi	0.000182
resolutionTheta	0.000265
trackIso	2.385159

Reconstructed and generated levels

Properties

# Use GUI to design analysis

Prototyping



Execution Verification

The screenshot shows the VisualPXL software interface. The title bar reads "VisualPXL - File Browser, Event Editor & Process Modelling". The menu bar includes "File", "Tools", and "Help". The toolbar contains "Open", "Save", and "Save As" buttons. The main workspace is titled "new analysis" and contains a "File Input" module. A blue speech bubble labeled "Input file" points to the module. A blue box labeled "Choose module" is positioned on the left, and another blue box labeled "Insert module" is at the bottom center. On the right, a configuration panel for "Main: Pxl-Plugin" shows the module type as "File Input", name as "File Input0", and options for "filename" and "start" (set to 0). A blue box labeled "Configure module" is positioned on the right side of the configuration panel.

Main: Pxl-Plugin	
Type	File Input
Name	File Input0
Options	
filename	<input type="text"/>
start	0

# Complete Analysis Design

Prototyping



Execution Verification

VisualPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

new analysis

File Input  
File Output  
ScriptFilter

## Analysis module

Input file

Output file

File Input0

ScriptFilter1

File Output2

Choose other modules

Connect the modules

Configure modules

Main: Pxl-Plugin	
Type	ScriptFilter
Name	ScriptFilter1
Options	
filename	
script	
eventview	False
parameter	

# Create Python Analysis Module

Prototyping



Execution Verification

The screenshot shows the VisualPXL software interface. The main workspace displays a workflow diagram with three components: 'File Input0', 'ScriptFilter1', and 'File Output2'. 'File Input0' is connected to 'ScriptFilter1', which is connected to 'File Output2'. The 'ScriptFilter1' component is highlighted with a green border and contains a vertical list of red and green checkboxes. A blue callout box with the text 'Edit user analysis' points to the 'ScriptFilter1' component. The interface includes a menu bar (File, Tools, Help), a toolbar (Open, Save, Save As), and a right-hand panel with a tree view and a properties table.

VisualPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

new analysis

File Input  
File Output  
ScriptFilter

Analysis module

Input file

Output file

File Input0

ScriptFilter1

File Output2

Edit user analysis

Main: Pxl-Plugin

Type	ScriptFilter
Name	ScriptFilter1
Options	
filename	
script	
eventview	False
parameter	

# Create Python Analysis Module

Prototyping



Execution Verification

VisualPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

new analysis

File Input  
File Output  
ScriptFilter

**Analysis module**

Input file

ScriptFilter1

Output file

File Input0

File Output2

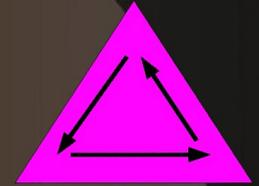
**Rapid prototyping of the analysis**

```
for particle in particles:  
    if particle.getName() == 'Muon':  
        histo_muon_pt.Fill(particle.getPt() )
```

Main: Pxl-Plugin

Type	ScriptFilter
Name	ScriptFilter1
Options	
filename	
script	
eventview	False
parameter	

# Run Analysis



**Export the analysis (as XML or Python)**  
→ **batch, GRID**

VisualIPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

analysis.xml

File Input  
File Output  
ScriptFilter

Analyse

Analyse File: analysis.xml

Select...

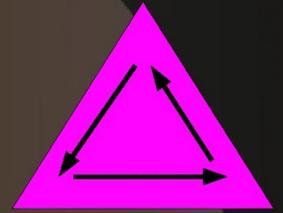
```
plugin /usr/local/libexec/pxl-plugins-2.0/scriptfilter-plugin.so loaded
plugin /usr/local/libexec/pxl-plugins-2.0/io-plugin.so loaded
::: initialize plugin 'File Input0' ...
::: initialize plugin 'ScriptFilter1' ...
*** starting analysis ***
::: initialize plugin 'File Output2' ...
plugins loaded...
connections established...
begin loop
end loop
File Input0: read 6000 Events.
File Output2: wrote 6000 Events.
Info in <TCanvas::Print>: ps file %numu.ps has been created
*** finishing analysis
```

Process Close

**Or run the analysis interactively**

# Verify Analysis Output

Prototyping



Execution

Verification

VisualPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Save Save As

/home/tklimk/soft/vispa/talk/Zmumu\_out.pxlio

Current Event: 10  
Events in file: >= 10

Reconstructed  
Generator

Z

Muon  
Muon

Jet Jet

MET

Main: EventView

Id	aa5b50b9-652e-664t
Name	Reconstructed

UserRecords

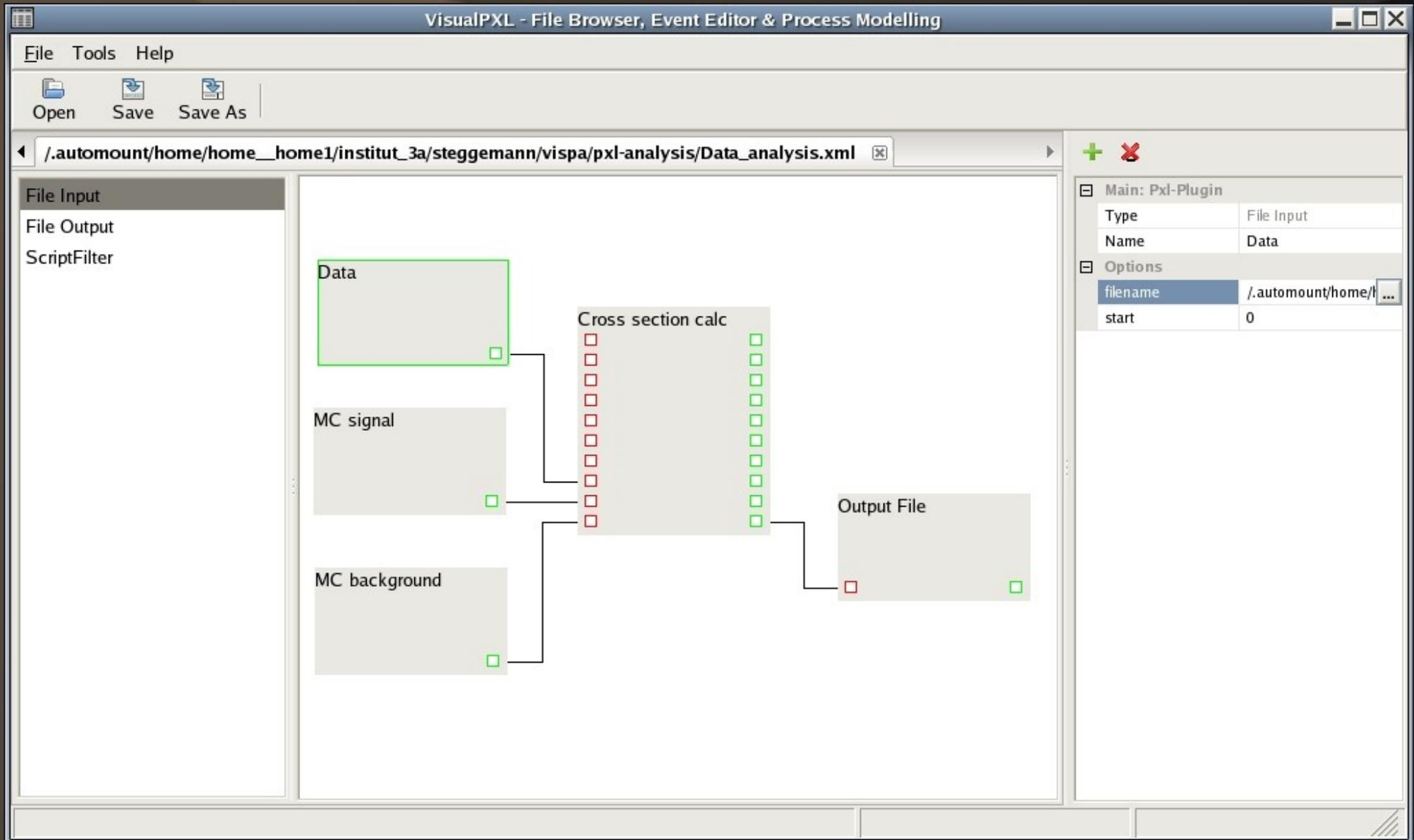
NumEle	0
NumJet	2
NumMET	1
NumMuon	2
Process	Zmumu
Type	Rec

Z mass

Create histogram using PyROOT

If needed repeat prototyping, execution, verification

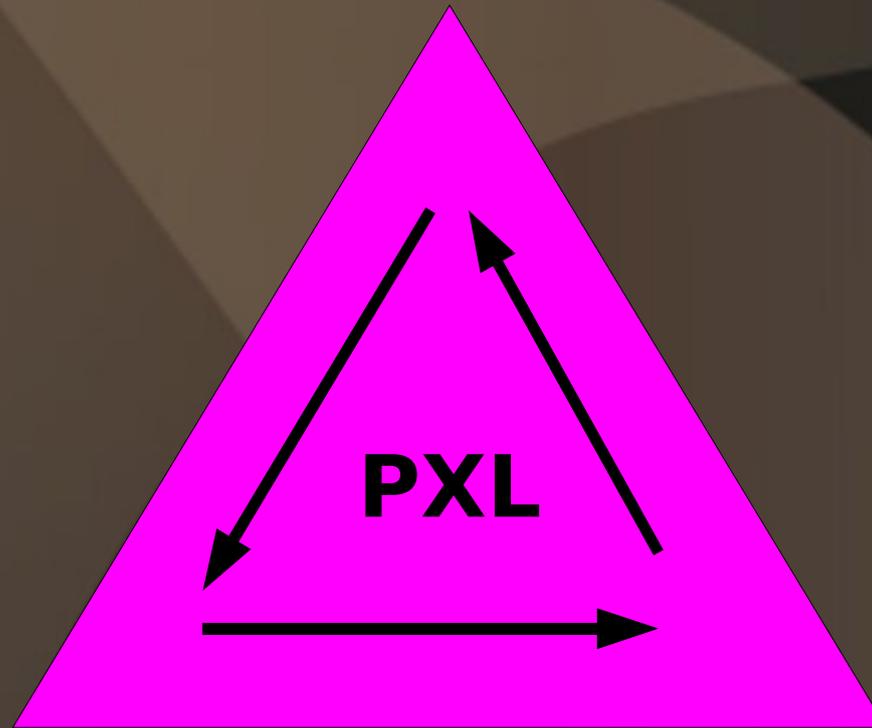
# More complex analysis





# Summary: Analysis flow with VISPA

**Prototyping:** Graphical platform  
**Analysis modules:** Python or C++



**Execution**

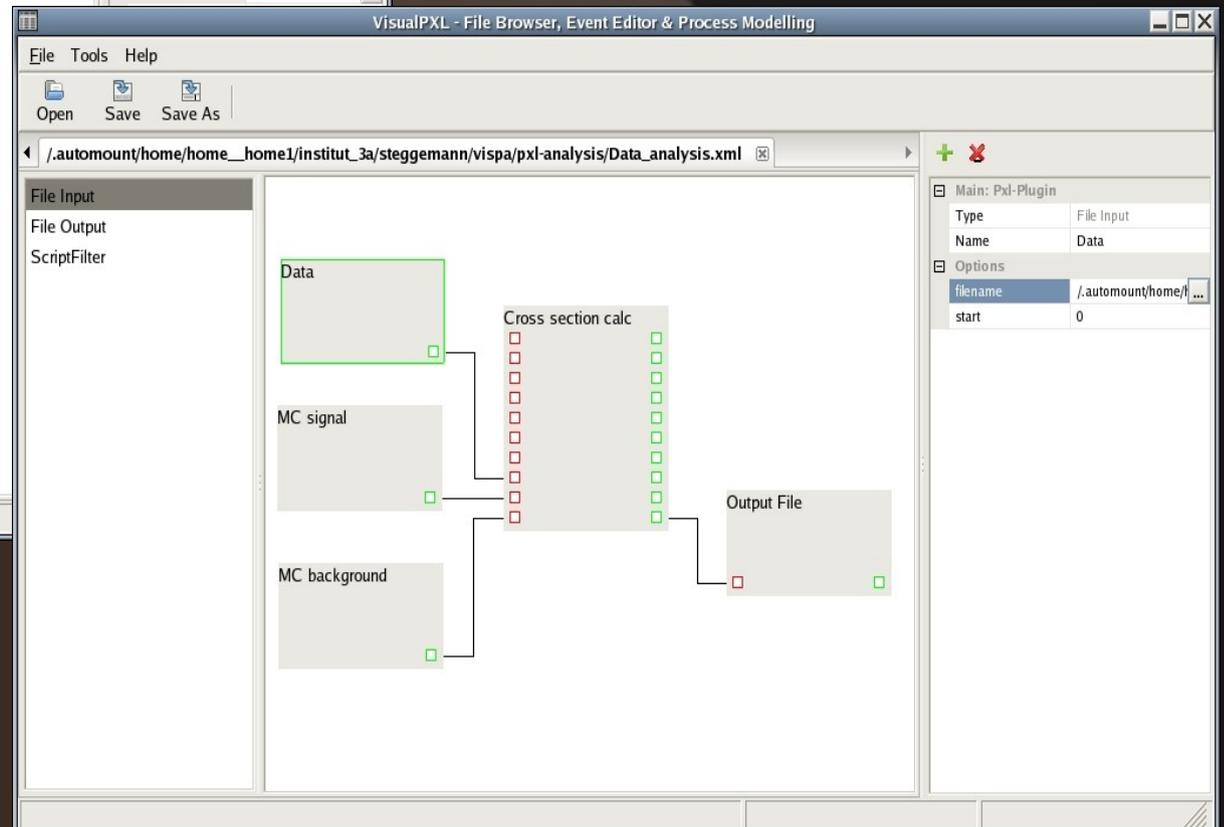
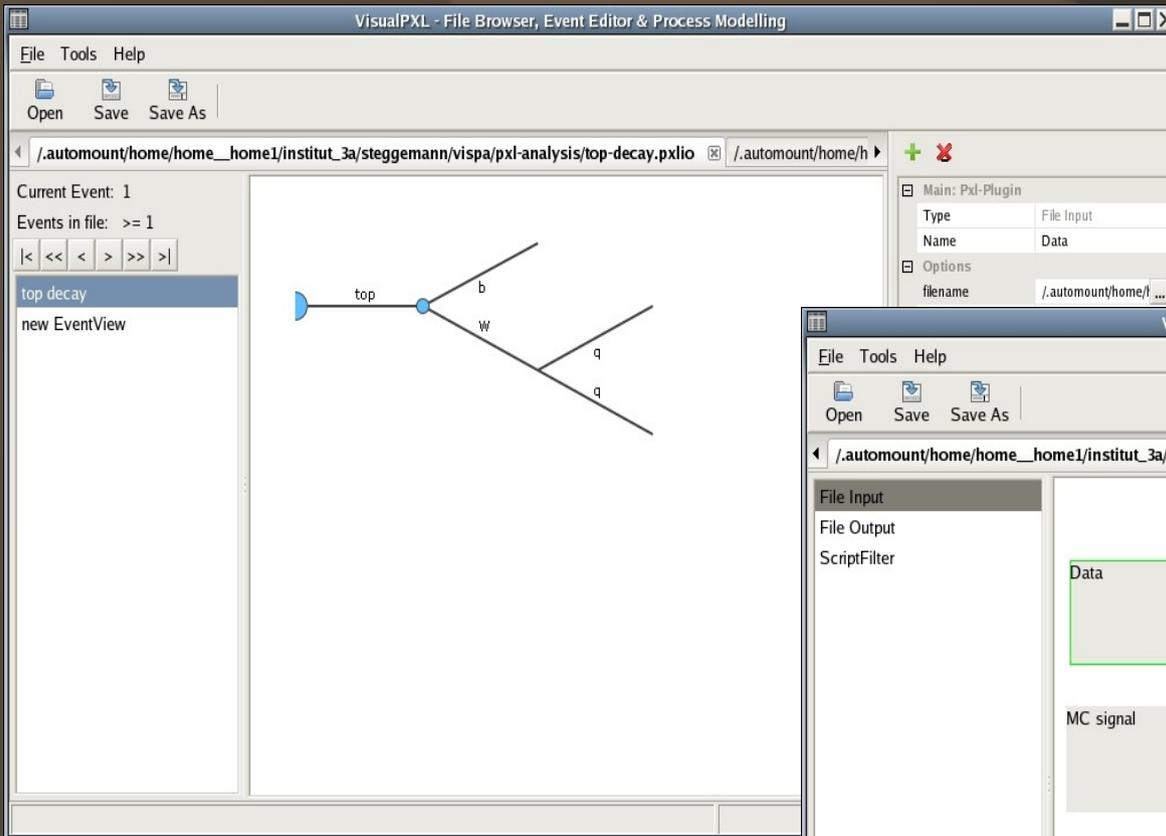
**(XML or Python steering)  
(interactive or batch)**

**Verification**

**(Event Browser)  
(ROOT histograms)**

# VISPA: till now was look & feel Now: to the ingredients

Mixture of graphical  
and textual work  
(like LabView)

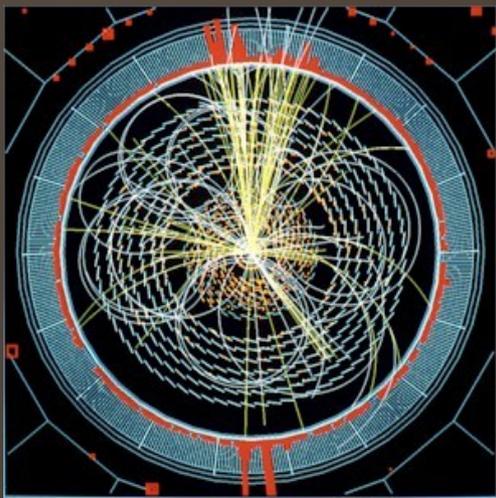


# VISPA Key Components

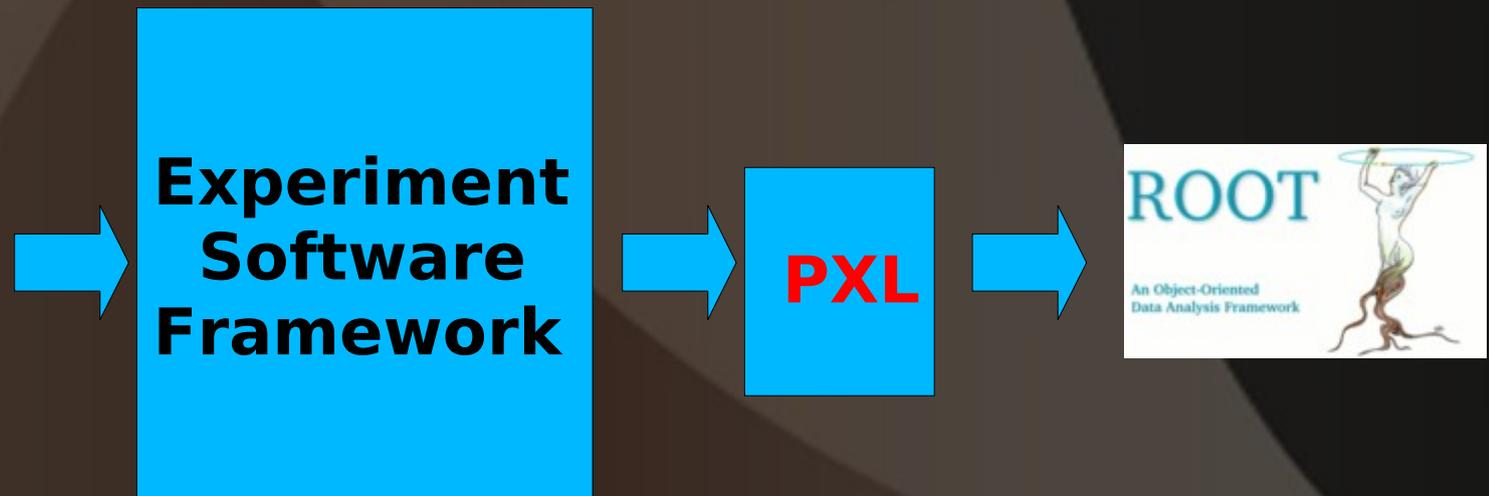
- **PXL**: C++ package providing underlying functionality
- **PyPXL**: Python interface to PXL
- **Module steering system**
- **Autoprocess**: automatic decay chain reconstruction

# PXL (Physics eXtension Library)

- **C++** toolkit for high-level physics analysis
- Provides underlying functionality for Visual Physics Analysis (VISPA)
- Version 2.0 (2008)
- Successor of **PAX** (Physics Analysis Expert) (2002-2007)

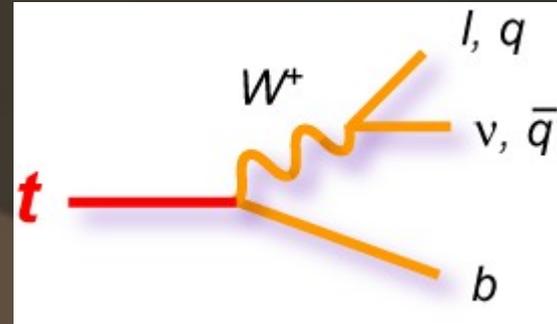
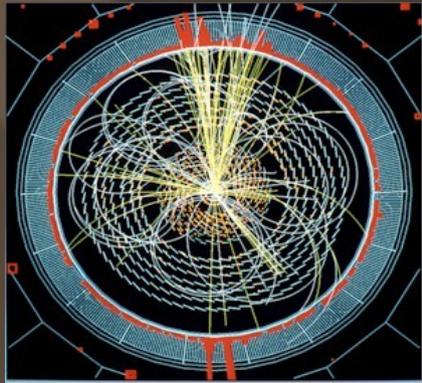


Tatsiana Klimkovich



ACAT2008, Erice, Sicily

# PXL key components: Event Container



- Particles (`pxl::Particle`)
- Vertices (`pxl::Vertex`)
- Collisions (`pxl::Collision`)
- User data (`pxl::UserRecord`)
- Their **relations** and **roles**

**Physics  
objects**

**Event View**

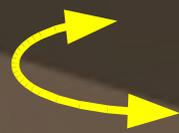
`pxl::EventViews`

**Event container** `pxl::Event` can hold several `pxl::EventViews`

Allows **deep copies** (physics objects with redirected relations, data members, user records)

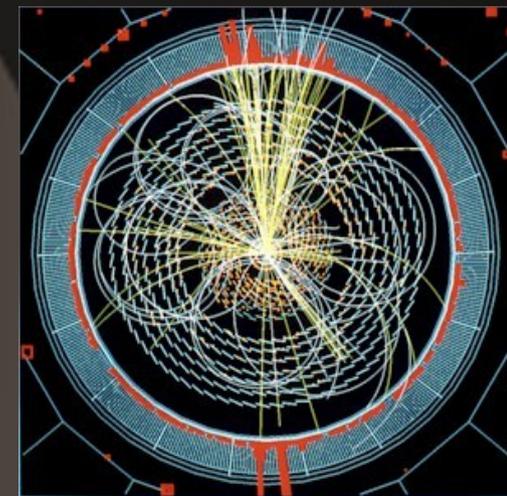
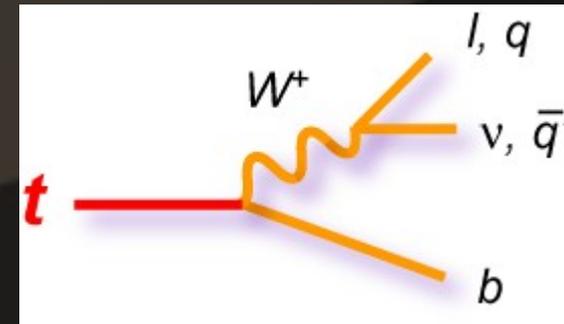
# PXL key components: UserRecord

- PXL physics object is not only a fourvector
- PXL physics object has also a **UserRecord** where user data are stored:
  - these are **pairs of names and all basic C++ types** (int, double, string, ...)
  - it can be e.g. data from the condition databases etc.
- Deploys **Copy-On-Write** mechanism
- Flexible and simple extension of objects



# PXL key components: Relation Management

- **“Hard” relations:**
  - Mother, daughter, and flat relations
  - Between objects within the same event container
  - Safe removal in case of object deletion
- **“Soft” relations:**
  - Between any objects
  - Have an arbitrary name (*string*)



# PXL key components: Input / Output

- Main class **pxl::Serializable**
- **Fast, Flexible**
- **Small file size:** use **ZLIB** library for data compression
- **Information chunks to structure files**
- **Each object knows how to stream itself**
  - methods “serialize” and “deserialize”
- **Simple inclusion of user classes into I/O scheme**

# Python interface to PXL

- **Why Python?**
  - Python code is easy to read
  - Less code compared to C++
  - Dynamic typing
  - Automatic memory management
  - Python does not need compilation
  - Has an interactive mode for testing
    - Object oriented, works on multiple platforms, open source
- Use of **SWIG** for automatic transfer

# Python popularity in HEP

- Starts to be **more popular** for doing physics analyses
- **Bender – LHCb** Python-based physics analysis application
- Possible to perform analysis with Python in **CMS**
- Some use in **D0**

# Module Steering System

- **Data flow**

- each module has a number of **sources and sinks**
- interface between modules: **PXL event container**

- **Modules**

- plug-in mechanism
- interactive creation of PYTHON modules

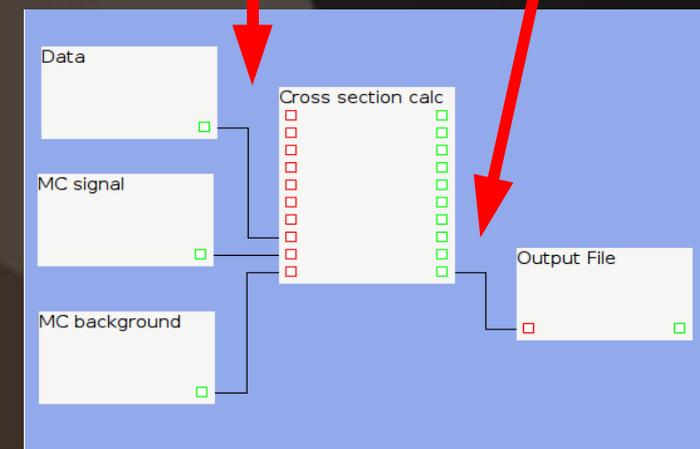
- **XML configuration**

- exchange format
- save and restore any state of the analysis

- **PYTHON configuration**

- high flexibility
- easy-to-read

**Interface:  
Event Container**



# Autoprocess

The screenshot shows the VisualPXL software interface. The main window displays a workflow diagram with three main components: 'Input file', 'ScriptFilter', and 'Output file'. The 'Input file' is connected to the 'File Input' module, which is connected to the 'ScriptFilter' module. The 'ScriptFilter' module is connected to the 'File Output' module, which is connected to the 'Output file'. A red arrow points from the title 'Autoprocess' to the 'File Input' module. Three yellow callout boxes are overlaid on the interface: 'Choose further modules' points to the left sidebar, 'Configure modules' points to the right sidebar, and 'Connect the modules' points to the central workflow diagram.

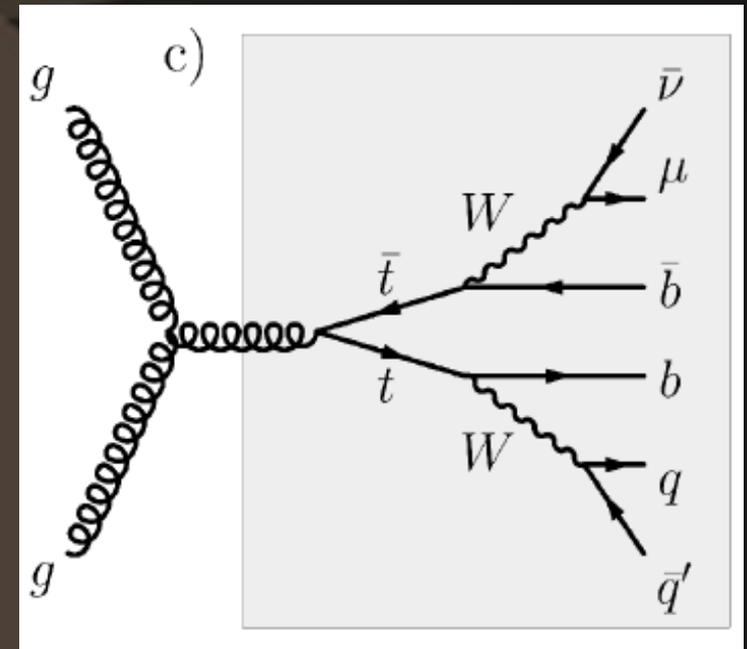
**Choose further modules**

**Configure modules**

**Connect the modules**

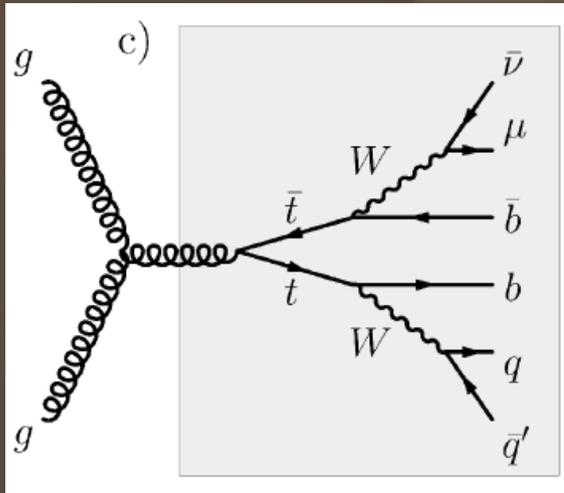
# Autoprocess

- In some physics analysis (Top, Higgs, SUSY) a reconstruction of the whole decay chain often needed
- Several possible configurations need to be built
- **Autoprocess** is an algorithm for automated reconstruction of particle cascades
- Avoid programming reconstruction code for every physics process individually



# Autoproccess

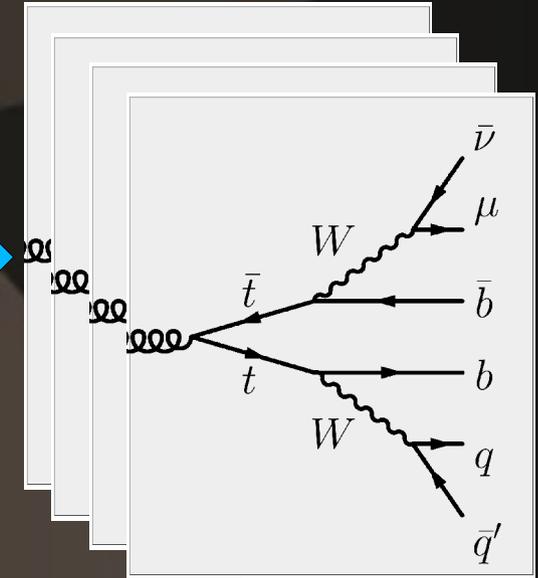
Steering  
event container



Data event  
container



Output event  
containers

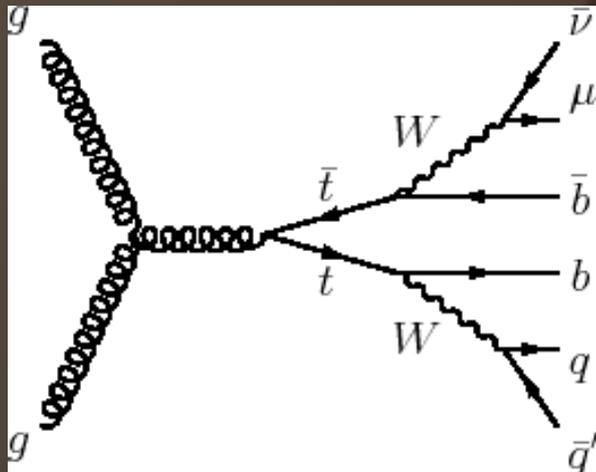


Autoproccess  
algorithm

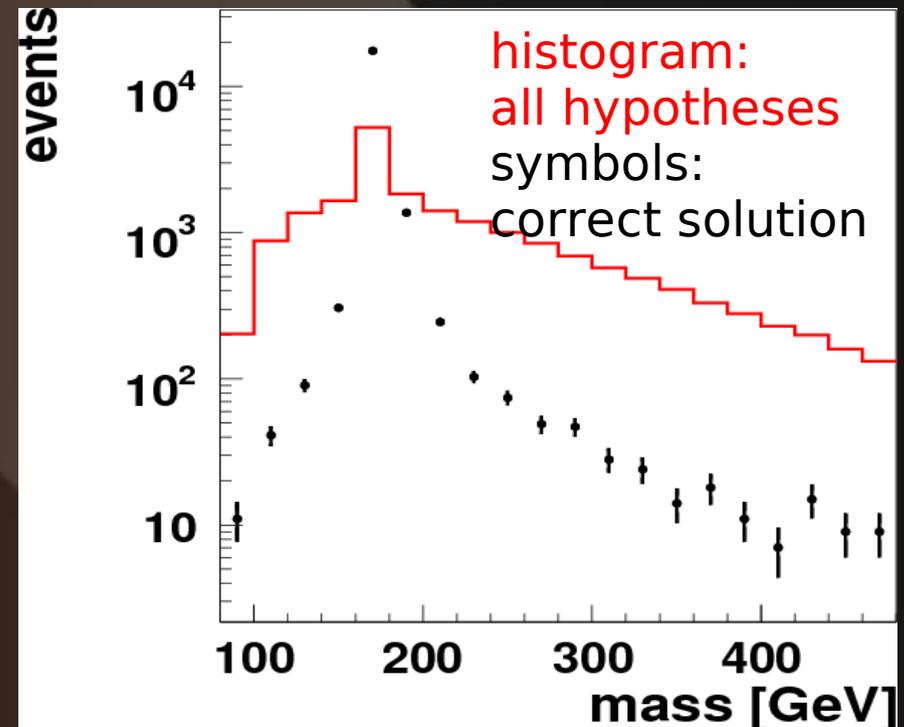
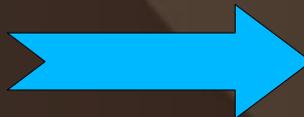
- **Steering** by an **event container**\* holding (part of a) Feynman diagram
- **Intuitive for physicist**  
\*PXL event container and particles (including relations)

# Autoproces

- Tool for automatic calculation of all event configurations (hypotheses)
- Well suited for e.g. top physics
- Good performance in **CPU time** and **memory consumption**



24 hypotheses



# Summary

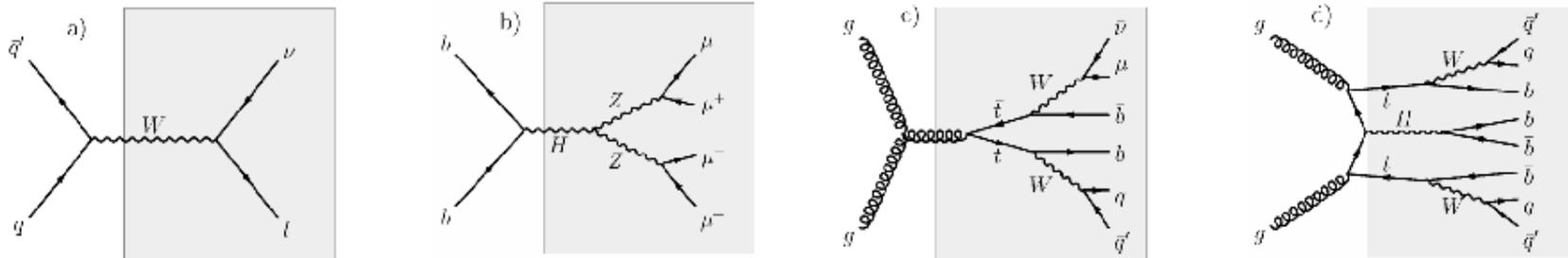
- **VISPA** is a novel **graphical analysis** design package for high energy physics analyses
  - Combines **visual and textual** programming
  - Allows fast **prototyping, execution, and verification** of an analysis
  - Can work for **any experiment**
  - First applications for CMS analysis
- **Key components:**
  - **PXL:** underlying functionalities for VISPA
  - **Module steering** system
  - **Autoprocess:** automatic calculation of **particle cascades**

# Summary II

- **All software is continuously maintained**
- **Fully documented:**
  - **Manual for PXL:**  
<http://pxl.sourceforge.net/manual.pdf>
  - **Doxygen**
- **Available online at <http://pxl.sourceforge.net>**
- **Look & feel **version for MAC is provided****
- **Publications:**  
<http://arxiv.org/abs/0810.3609>  
<http://arxiv.org/abs/0801.1302v2>

# Backup

# Autoproces: performance

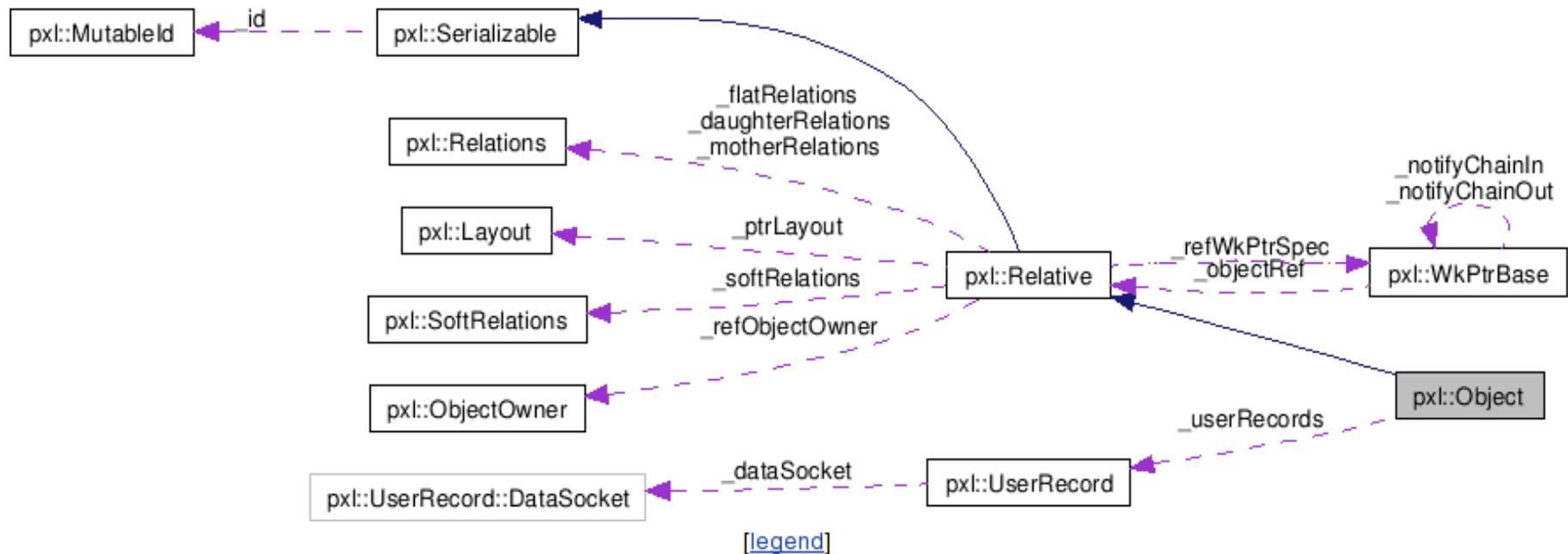


Physics process	number of particles	number of vertices	number of configurations	Time/event [ms]	Mem. alloc. [MByte]
$W \rightarrow l\nu$	3	1	2	0.06	< 1
$H \rightarrow 4\mu$	7	3	3	0.4	< 1
$t\bar{t} \rightarrow \mu\nu 4j$	11	5	24	2.3	< 1
$Ht\bar{t} \rightarrow 8j$	13	5	5040	411	36

on a standard personal computer:

- $\sim 20 \mu\text{s}$  per reconstructed decay vertex
- $\sim 0.6 \text{ kByte}$  per reconstructed particle in the decay trees

# PXL Objects Structure

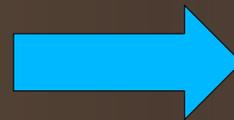
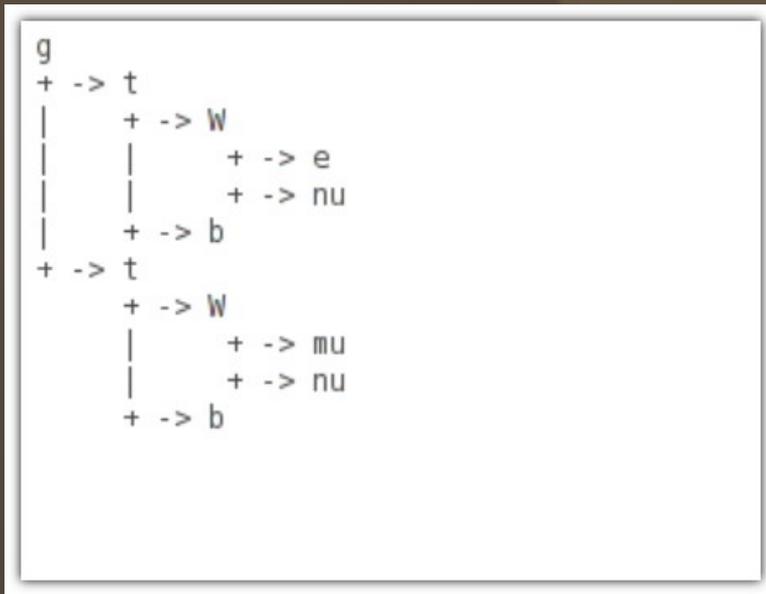


- Inheritance and composition

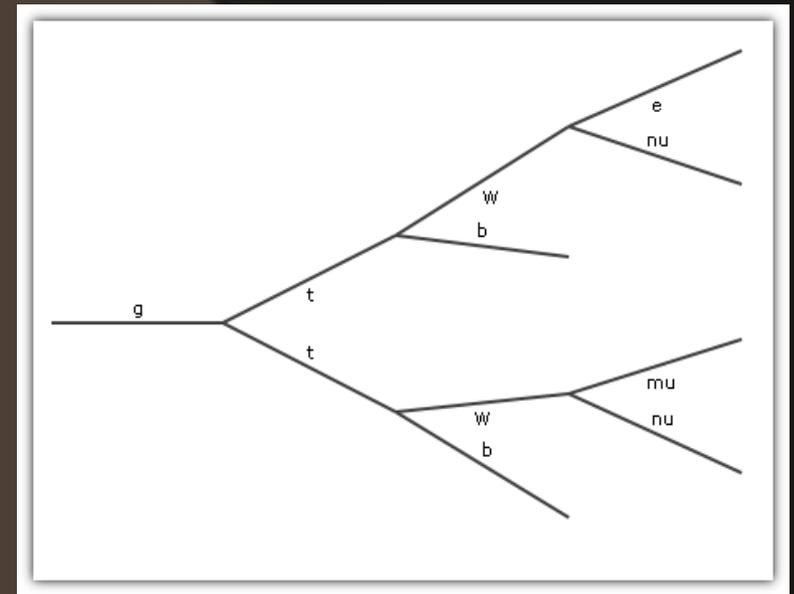
- I/O (*pxl::Serializable*)
- relations (*pxl::Relative*)
- User data (*pxl::Object*)
- Object container (*pxl::ObjectManager*)

# Graphical platform: autolayout algorithm

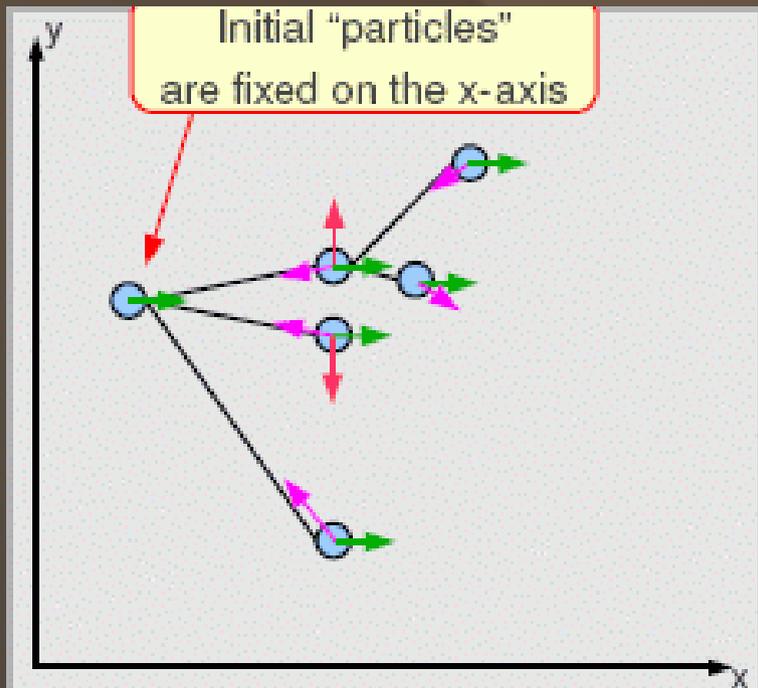
Hard to read text output



Desired, easy to read decay tree



# Autolayout algorithm based on model forces



**Constant forces** to all nodes

**Repelling forces** between close nodes

**Spring forces** to all daughter nodes

**Friction forces** to all moving nodes

- After some iterations all nodes moved into the positions where **all forces on the node cancel each other**
- After this the system is **in stable state**

# Autolayout algorithm based on model forces: final result

VisualPXL - File Browser, Event Editor & Process Modelling

File Tools Help

Open Remote File Open File Save Event As Append Event To Clear Event Arrange Interpret Delete Interpret

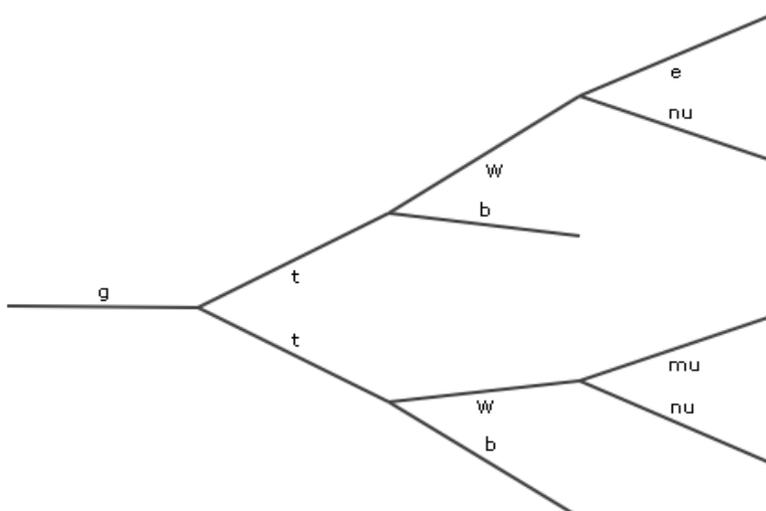
new file

Particle  
Vertex

Add EventView

Remove EventView

Generator  
Reconstructed



Main: EventView

Id	145574680
Name	Generator
Workflag	0

UserRecords

NEW {...}	
{visualpxl/layouted}	True
{Process}	ttbar_emu
{TtbarType}	g
{Type}	Gen