VISPA: Visual Physics Analysis Environment

Tatsiana Klimkovich for the **VISPA** group

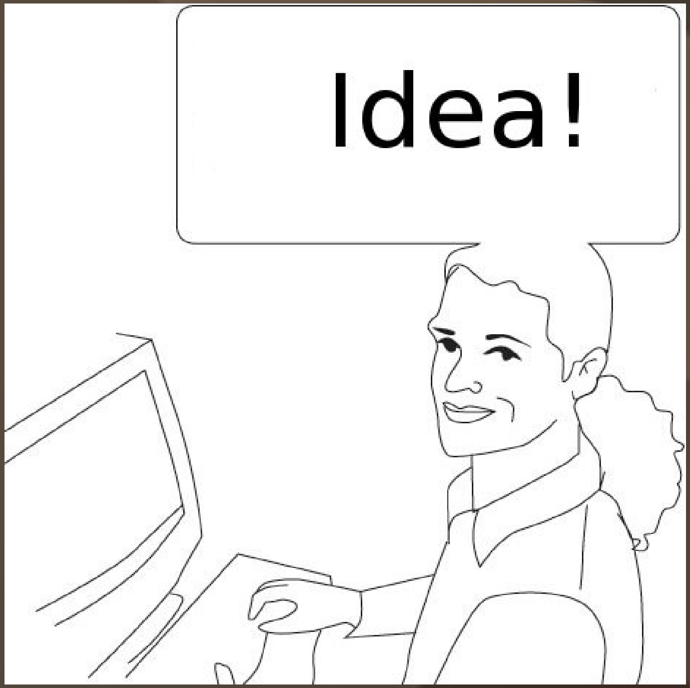
(O.Actis, M.Erdmann, R.Fischer, A.Hinzmann, M.Kirsch, G.Müller, M.Plum, J.Steggemann)

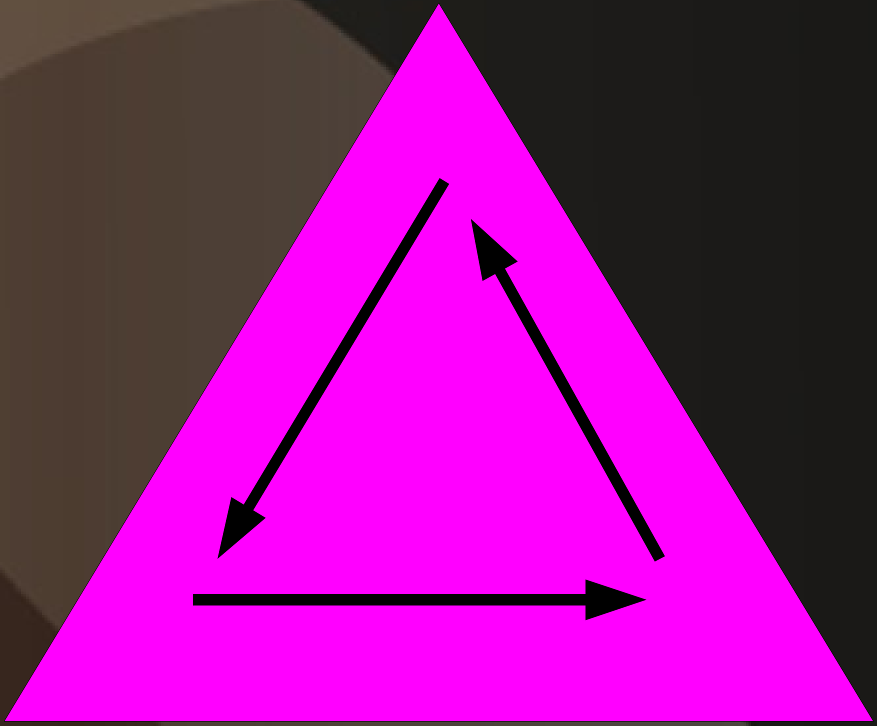
DESY Computing Seminar, 27 October 2008

Contents

- Physics Analysis in High Energy Physics experiment
- Novel Concept for HEP analyses: VISPA
- Look & feel with analysis example
- PXL: underlying functionality
- Python interface to C++ functionality
- Autolayout algorithm
- Autoprocess: tool for 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. **H1OO** in H1, **CMSSW** in CMS etc.
- On top of them more analysis specific software has been developed and used (e.g. **H1Lt** and **Marana** in H1 which work with Mods/Hat and user trees)
  - Analysis specific
  - Reusable, require less time for start
  - The outputs are smaller, more analysis specific
  - Reduce time for making analysis

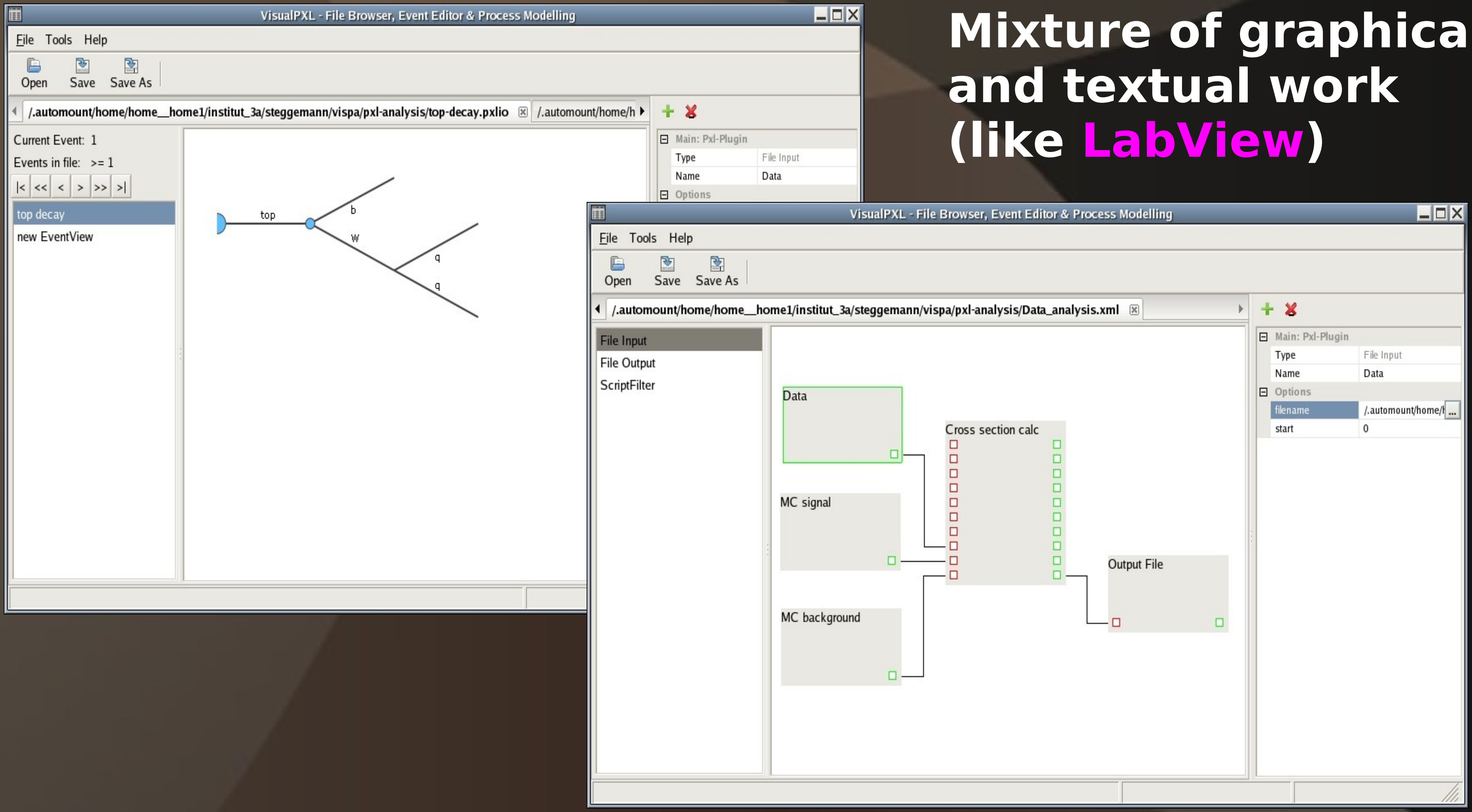
The wish list of the analyser

- 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 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**)



/.automount/home/home__home1/institut_3a/steggemann/vispa/pxl-analysis/top-decay.pxlio

/.automount/home/h

Current Event: 1

Events in file: >= 1

top decay

new EventView

Main: Pxl-Plugin

Type	File Input
Name	Data

Options

/.automount/home/home__home1/institut_3a/steggemann/vispa/pxl-analysis/Data_analysis.xml

Main: Pxl-Plugin

Type	File Input
Name	Data

Options

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

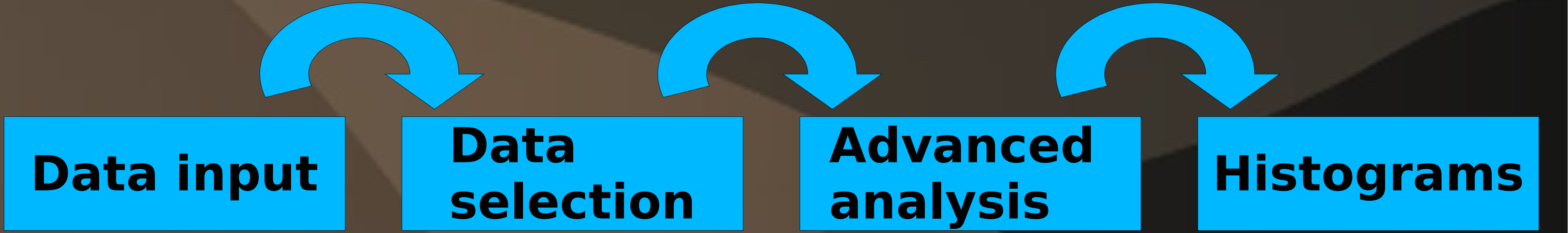
VISPA: Main Features

Development environment for HEP analyses

(first prototype)

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

Structure of Physics Analysis

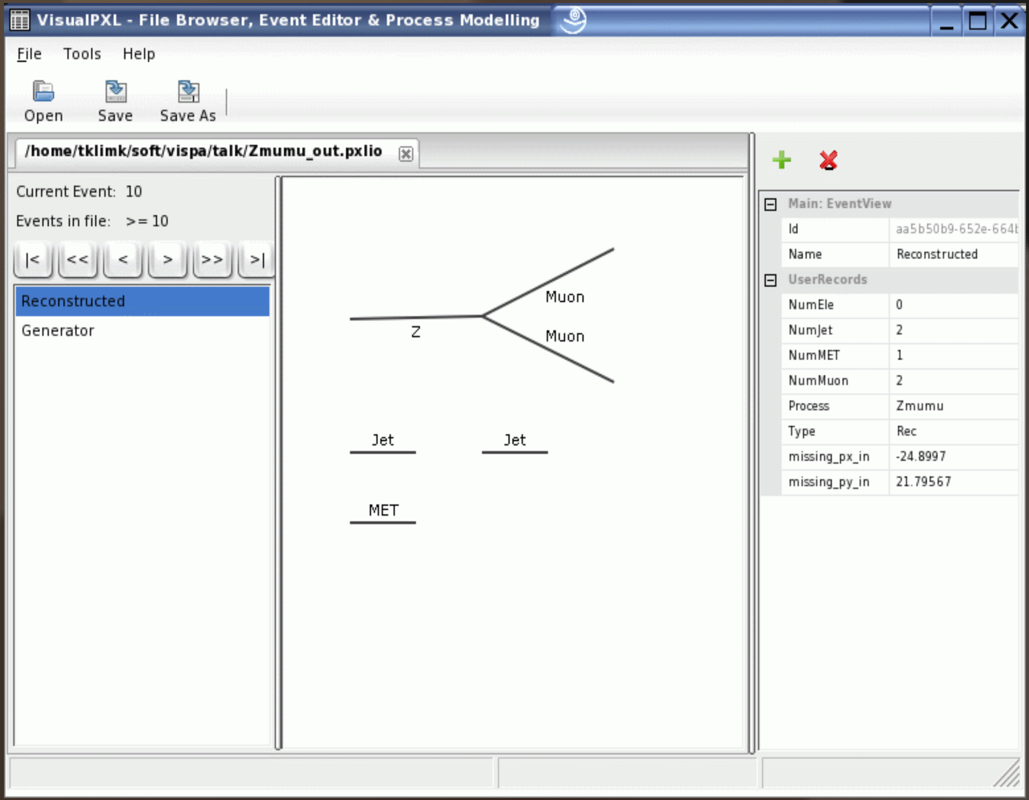


Data input

Data selection

Advanced analysis

Histograms



/home/tklimk/soft/vispa/talk/Zmumu_out.pxlio

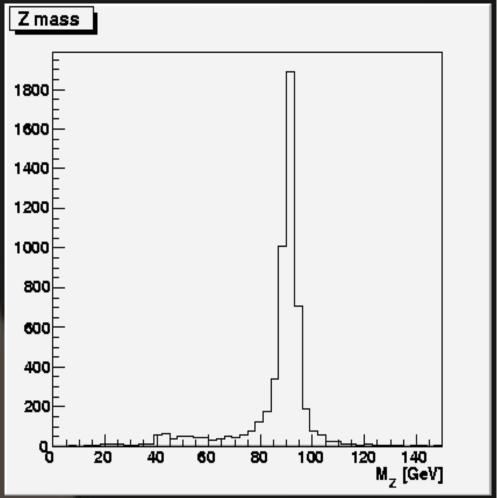
Current Event: 10

Events in file: >= 10

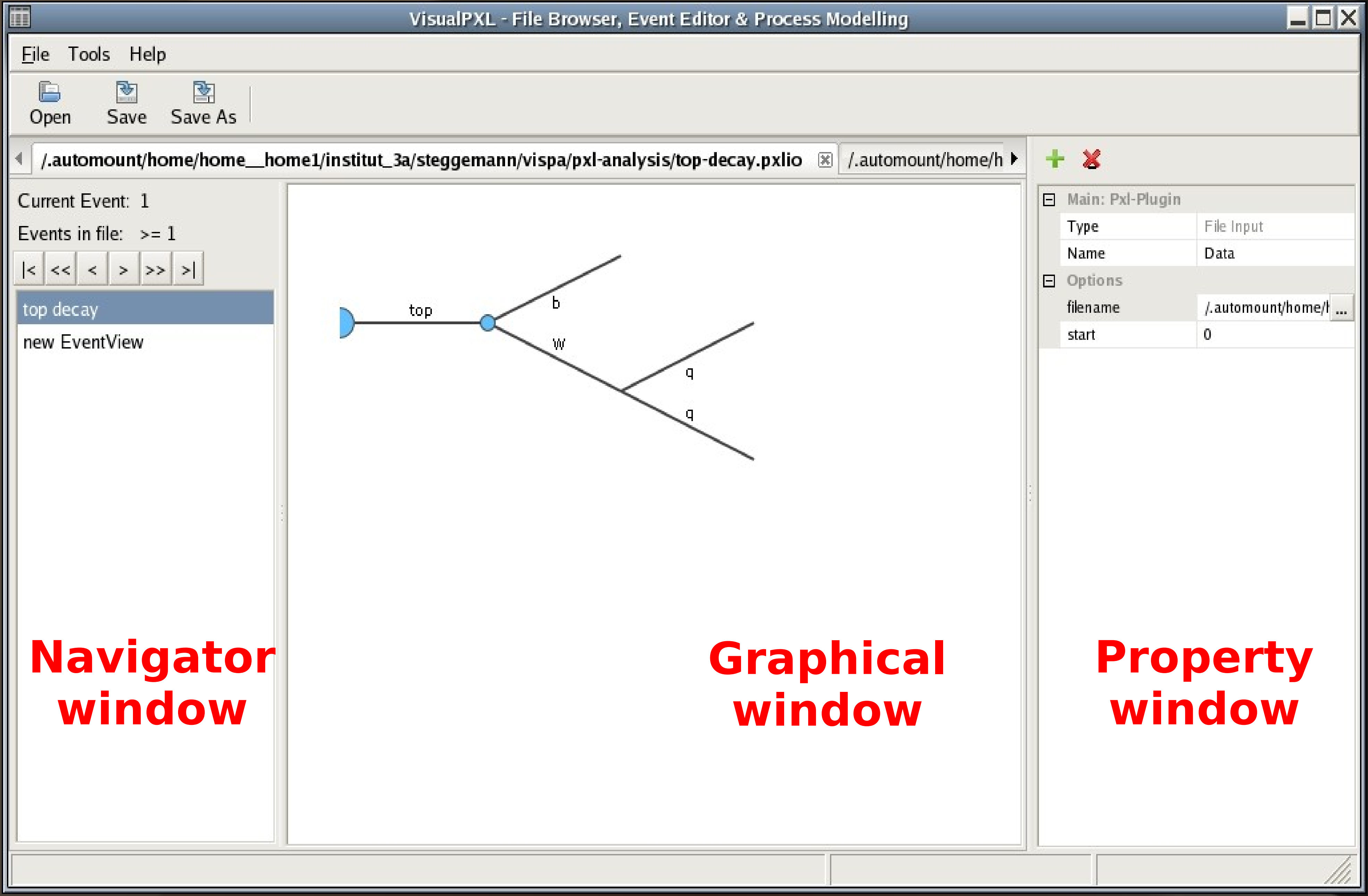
Reconstructed

Generator

Main: EventView	
Id	aa5b50b9-652e-664
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
/.automount/home/h
|<
<<
<
>
>>
>|
top decay
new EventView
top
b
W
q
q

Current Event: 1

Events in file: >= 1

⊟ Main: Pxl-Plugin	
Type	File Input
Name	Data
⊟ Options	
filename	/.automount/home/
start	0

Navigator window

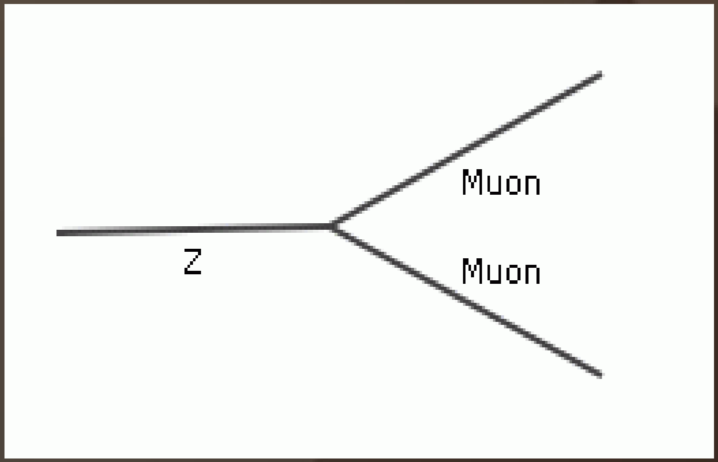
Graphical window

Property window

Closer consideration:

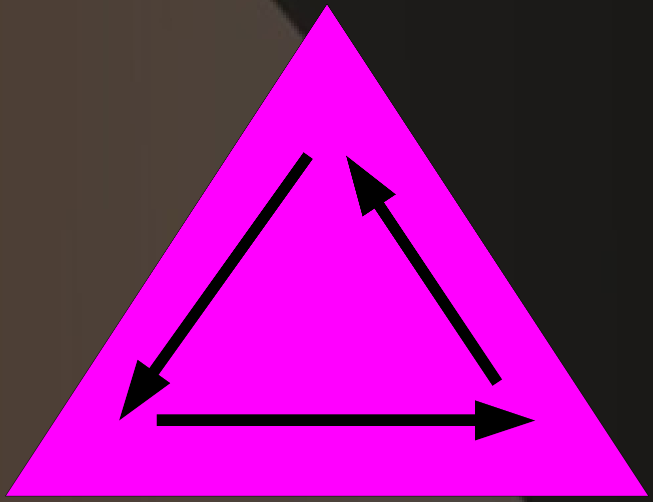
Very simple example analysis

Z --> mu mu



Sample at LHC energies (in pxlio format)

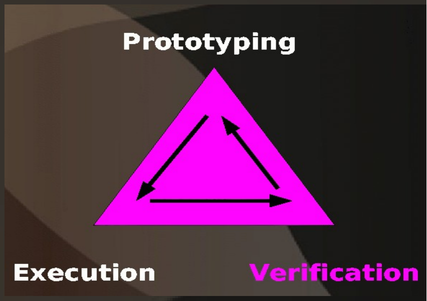
Prototyping

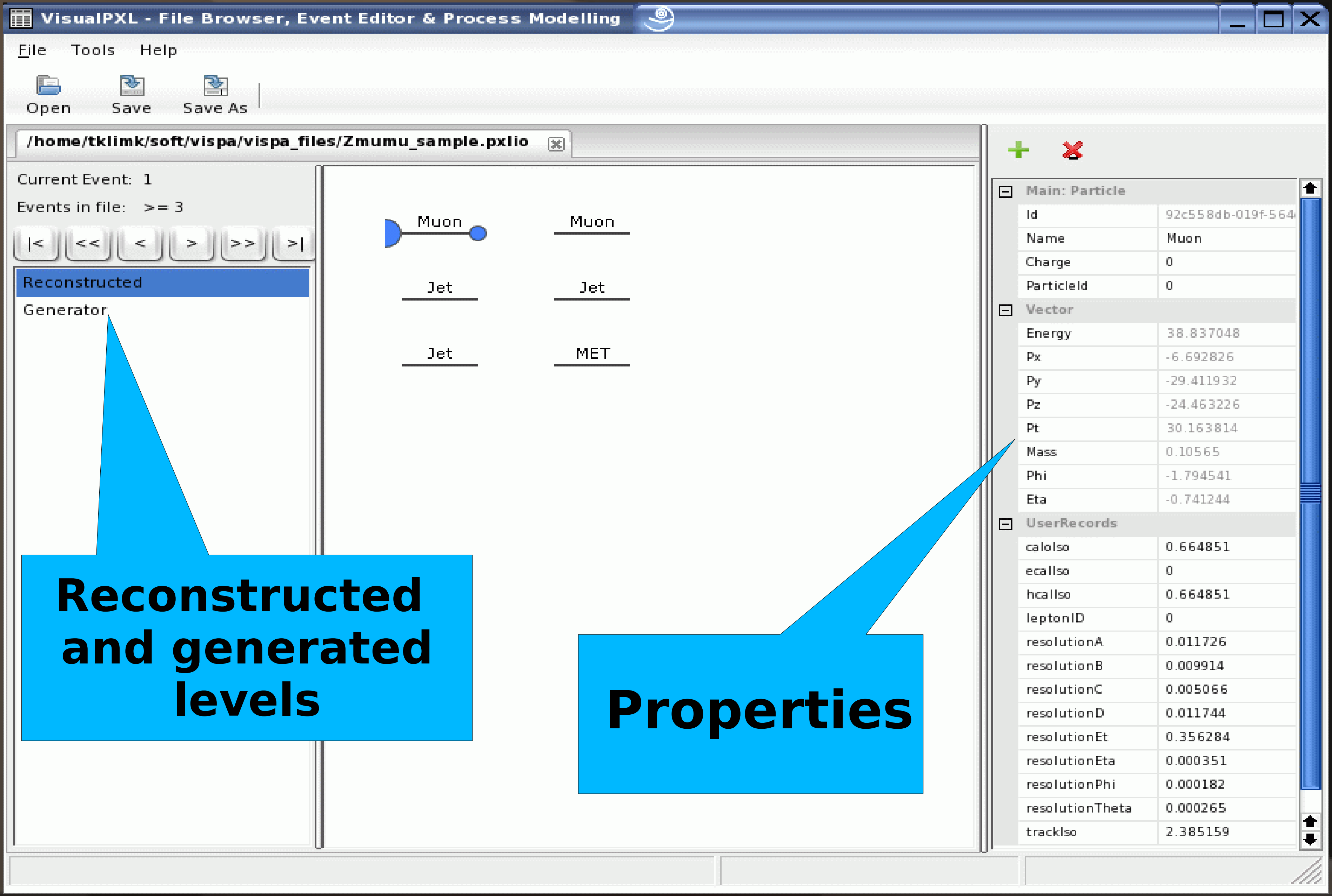


Execution

Verification

Z boson reconstruction from muons
First step: inspect an input file





/home/tklimk/soft/vispa/vispa_files/Zmumu_sample.pxlio

Current Event: 1

Events in file: >= 3

Reconstructed

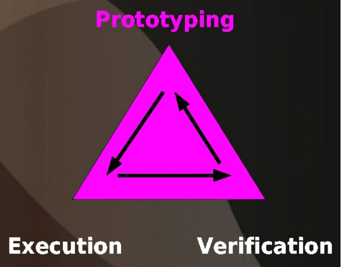
Generator

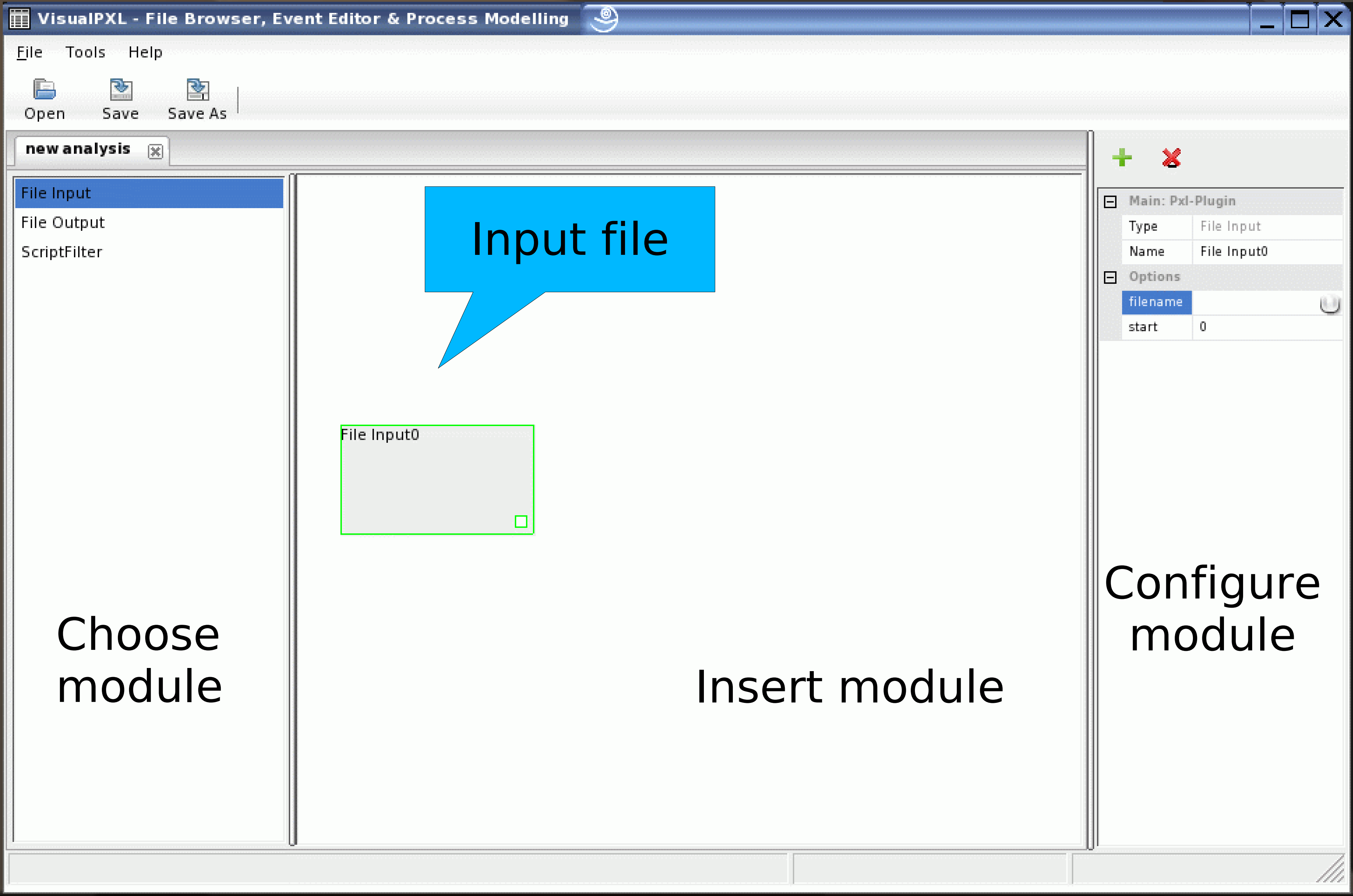
Main: Particle	
Id	92c558db-019f-564
Name	Muon
Charge	0
ParticleId	0
Vector	
Energy	38.837048
Px	-6.692826
Py	-29.411932
Pz	-24.463226
Pt	30.163814
Mass	0.10565
Phi	-1.794541
Eta	-0.741244
UserRecords	
caloIso	0.664851
ecalIso	0
hcalIso	0.664851
leptonID	0
resolutionA	0.011726
resolutionB	0.009914
resolutionC	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 add Modules





Main: Pxl-Plugin

Type	File Input
Name	File Input0

Options

filename	
start	0

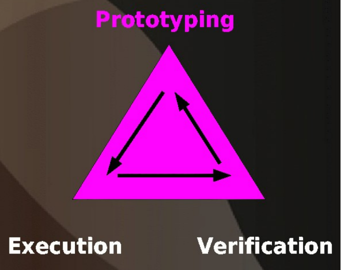
Input file

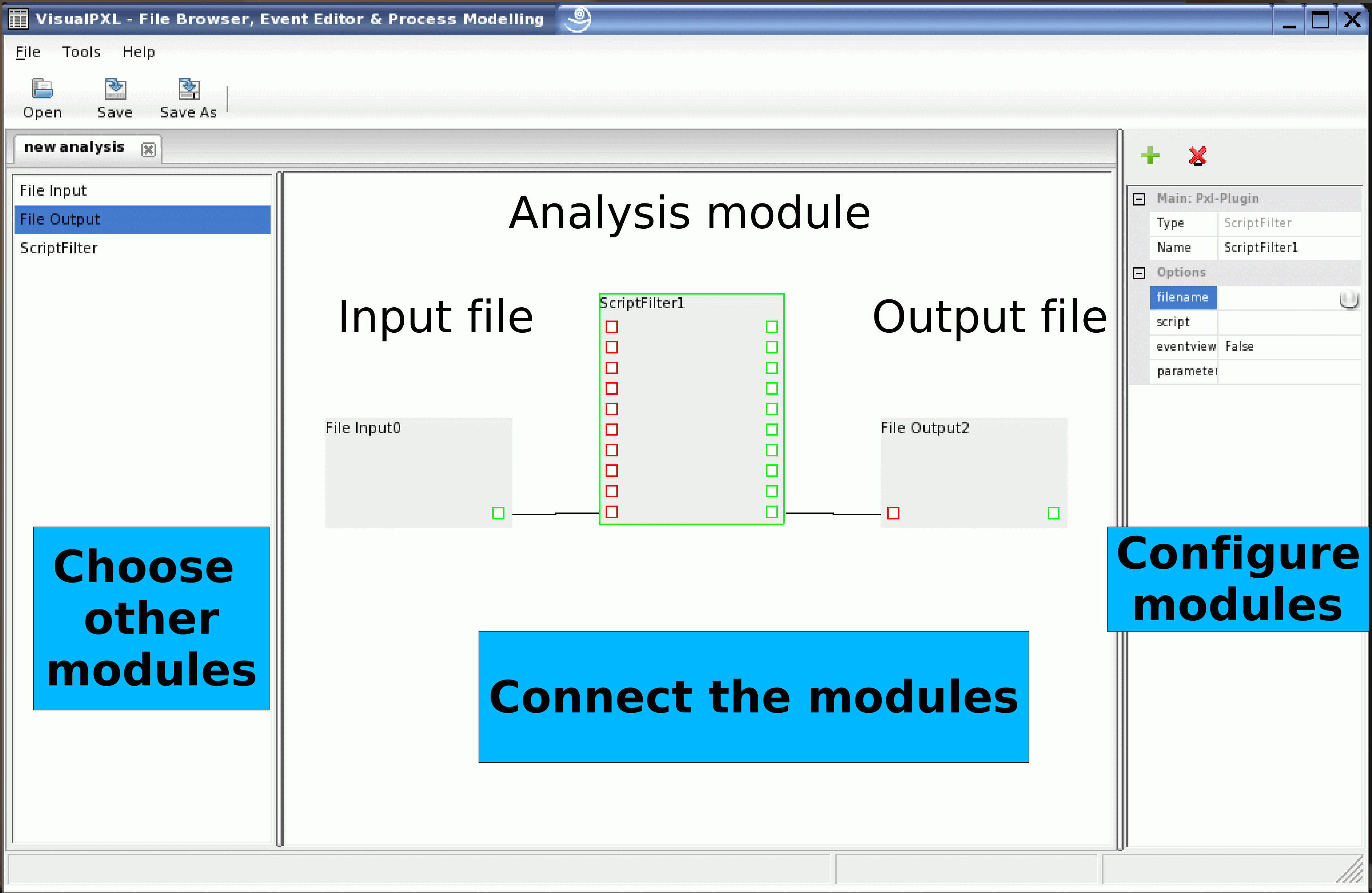
Choose module

Insert module

Configure module

Complete Analysis Design





Analysis module

Input file

Output file

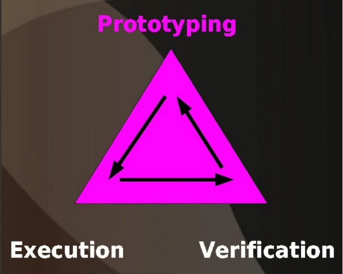
Main: Pxl-Plugin	
Type	ScriptFilter
Name	ScriptFilter1
Options	
filename	
script	
eventview	False
parameter	

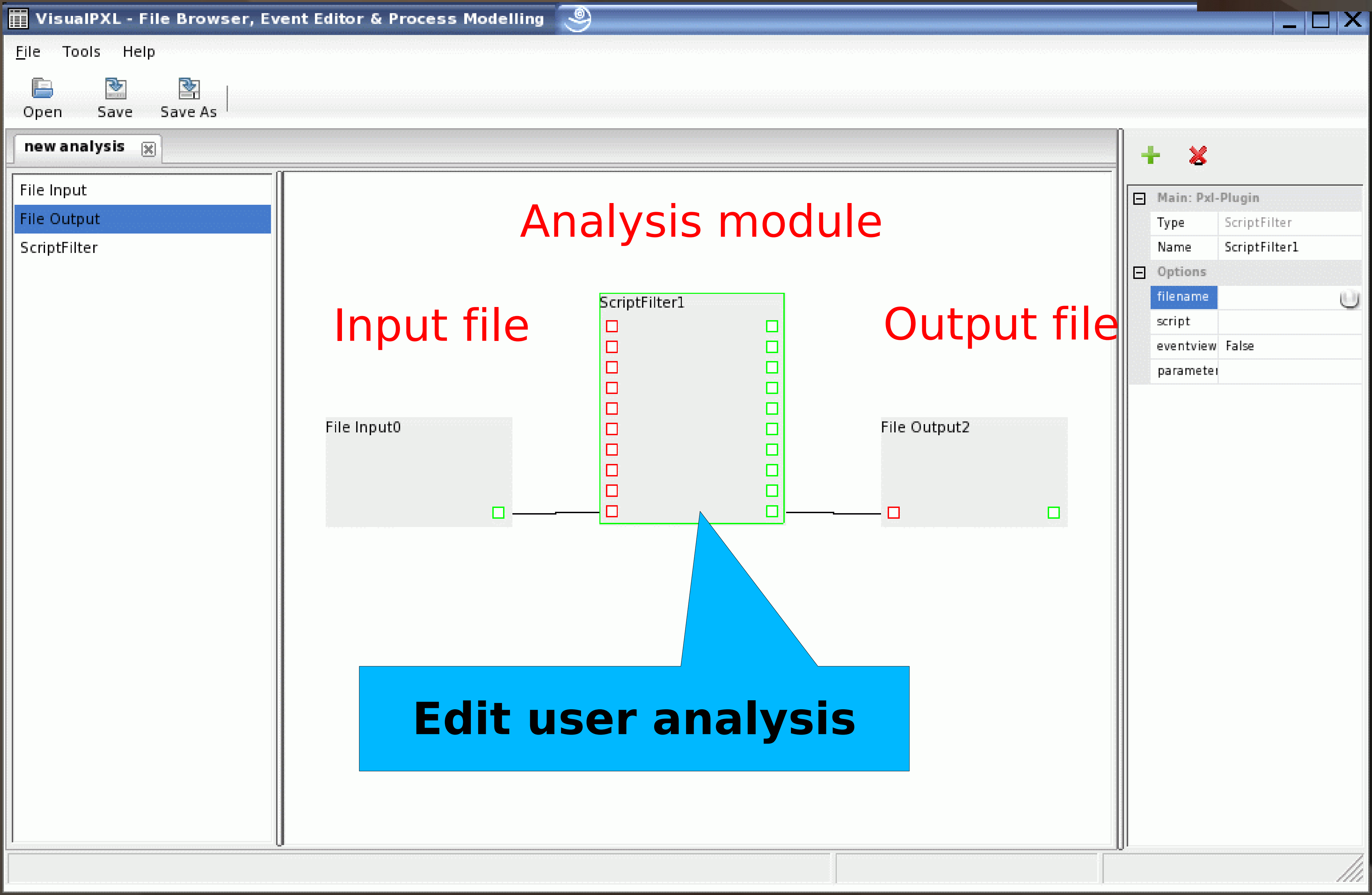
Choose other modules

Connect the modules

Configure modules

Create Python Analysis Module





Main: Pxl-Plugin

Type	ScriptFilter
Name	ScriptFilter1

Options

filename	
script	
eventview	False
parameter	

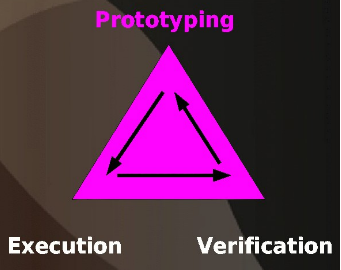
Analysis module

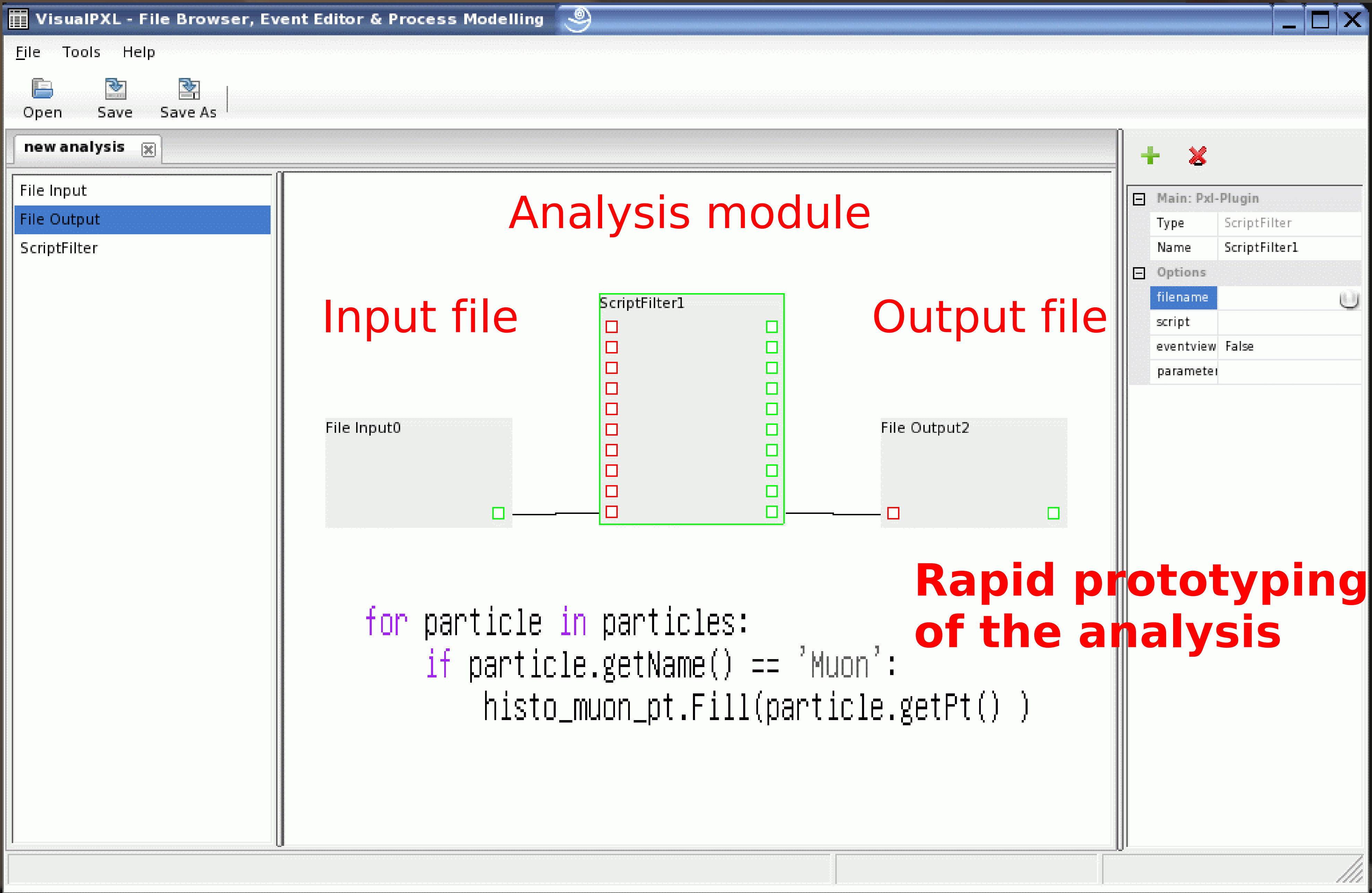
Input file

Output file

Edit user analysis

Create Python Analysis Module





Type: ScriptFilter
Name: ScriptFilter1
eventview: False

Analysis module

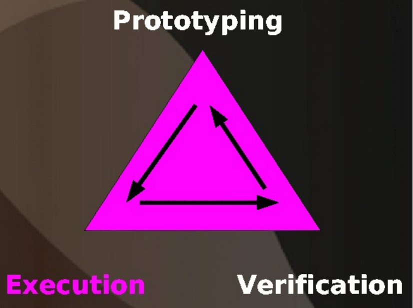
Input file

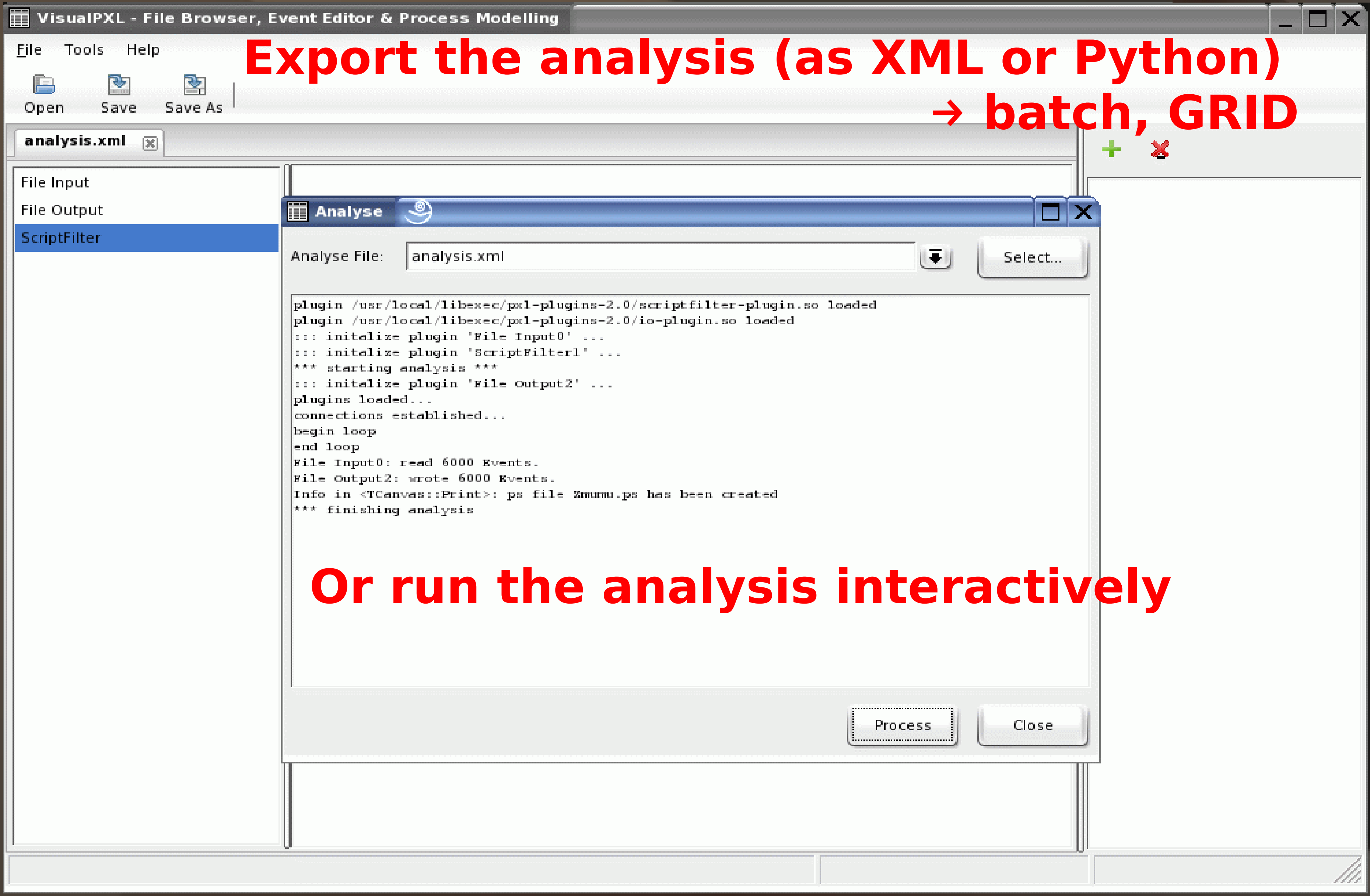
Output file

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

Rapid prototyping of the analysis

Run Analysis





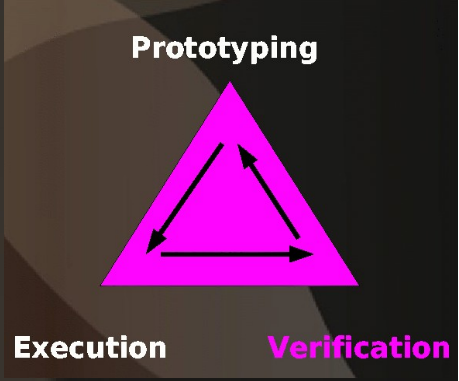
Analyse File: analysis.xml

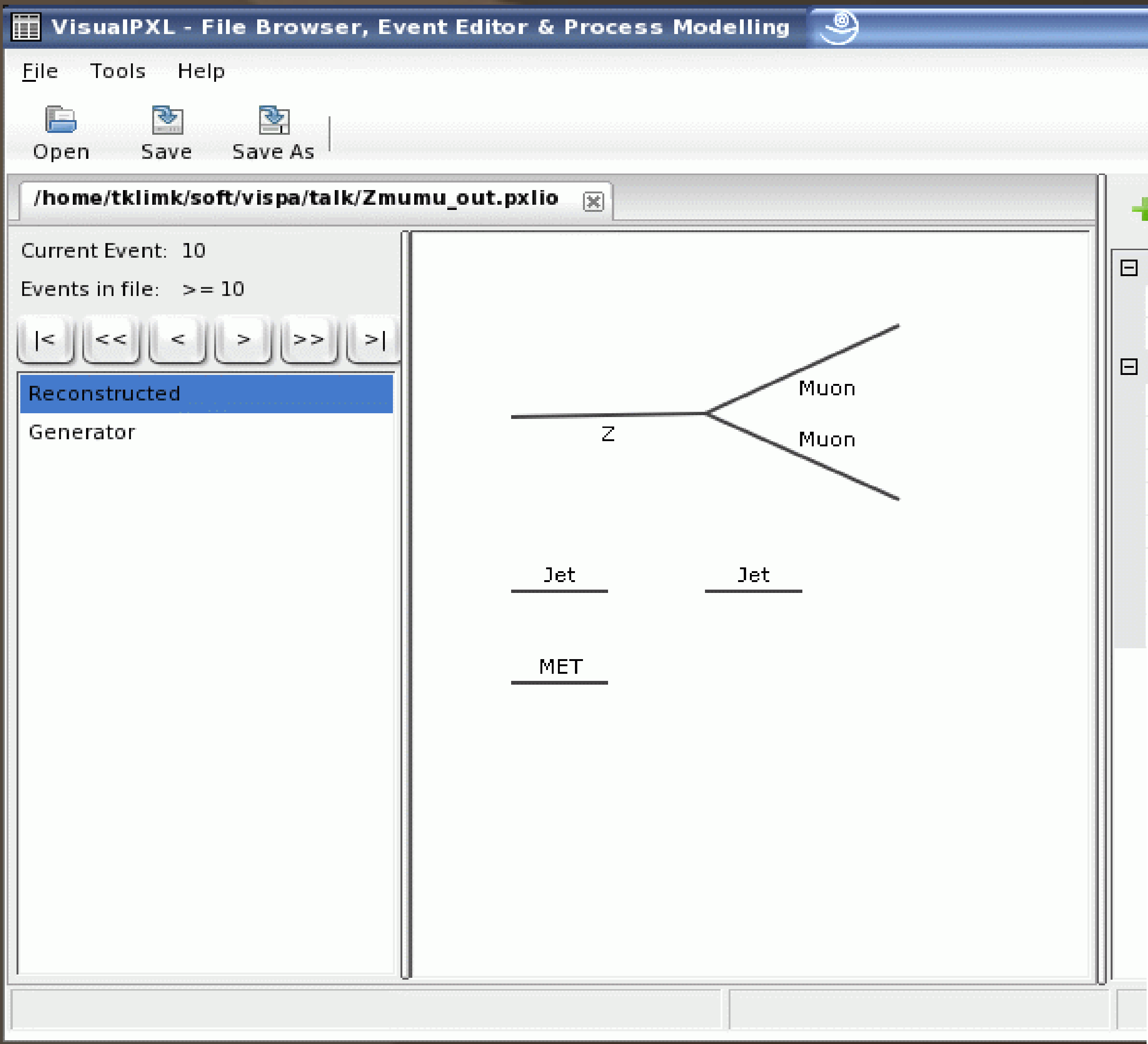
```
plugin /usr/local/libexec/pxl-plugins-2.0/scriptfilter-plugin.so loaded
plugin /usr/local/libexec/pxl-plugins-2.0/io-plugin.so loaded
::: initalize plugin 'File Input0' ...
::: initalize plugin 'ScriptFilter1' ...
*** starting analysis ***
::: initalize plugin 'File Output2' ...
plugins loaded...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

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

Or run the analysis interactively

Verify the Analysis Output




VisualPXL - File Browser, Event Editor & Process Modelling
File
Tools
Help
Open
Save
Save As
/home/tklimk/soft/vispa/talk/Zmumu_out.pxlio
|<
<<
<
>
>>
>|
Reconstructed
Generator
Z
Muon
Muon
Jet
Jet
MET

Current Event: 10

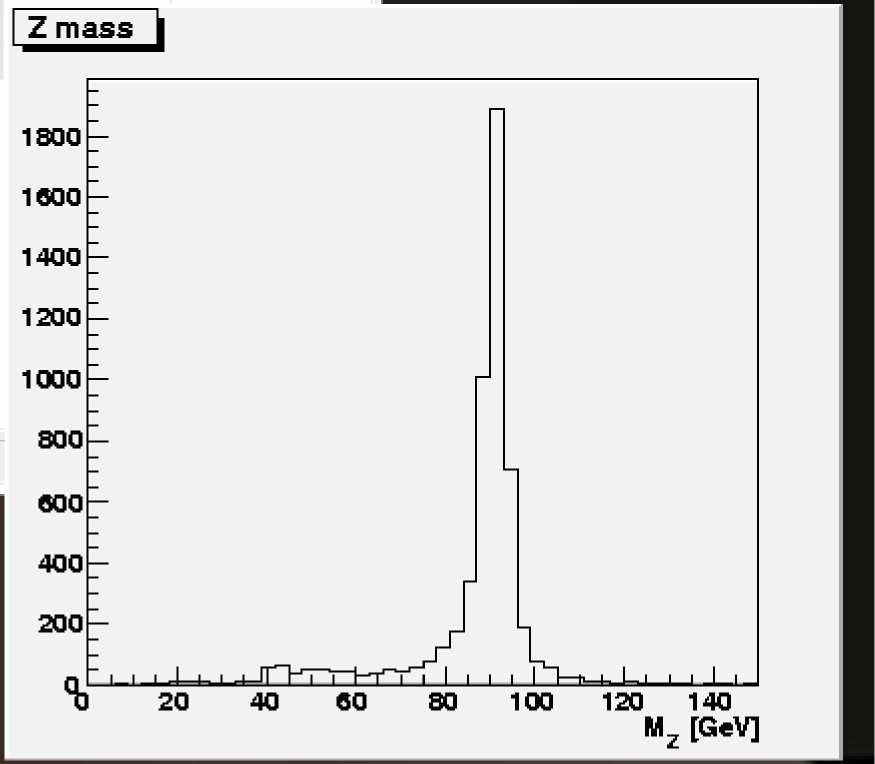
Events in file: >= 10

Main: EventView

Id	aa5b50b9-652e-664b
Name	Reconstructed

UserRecords

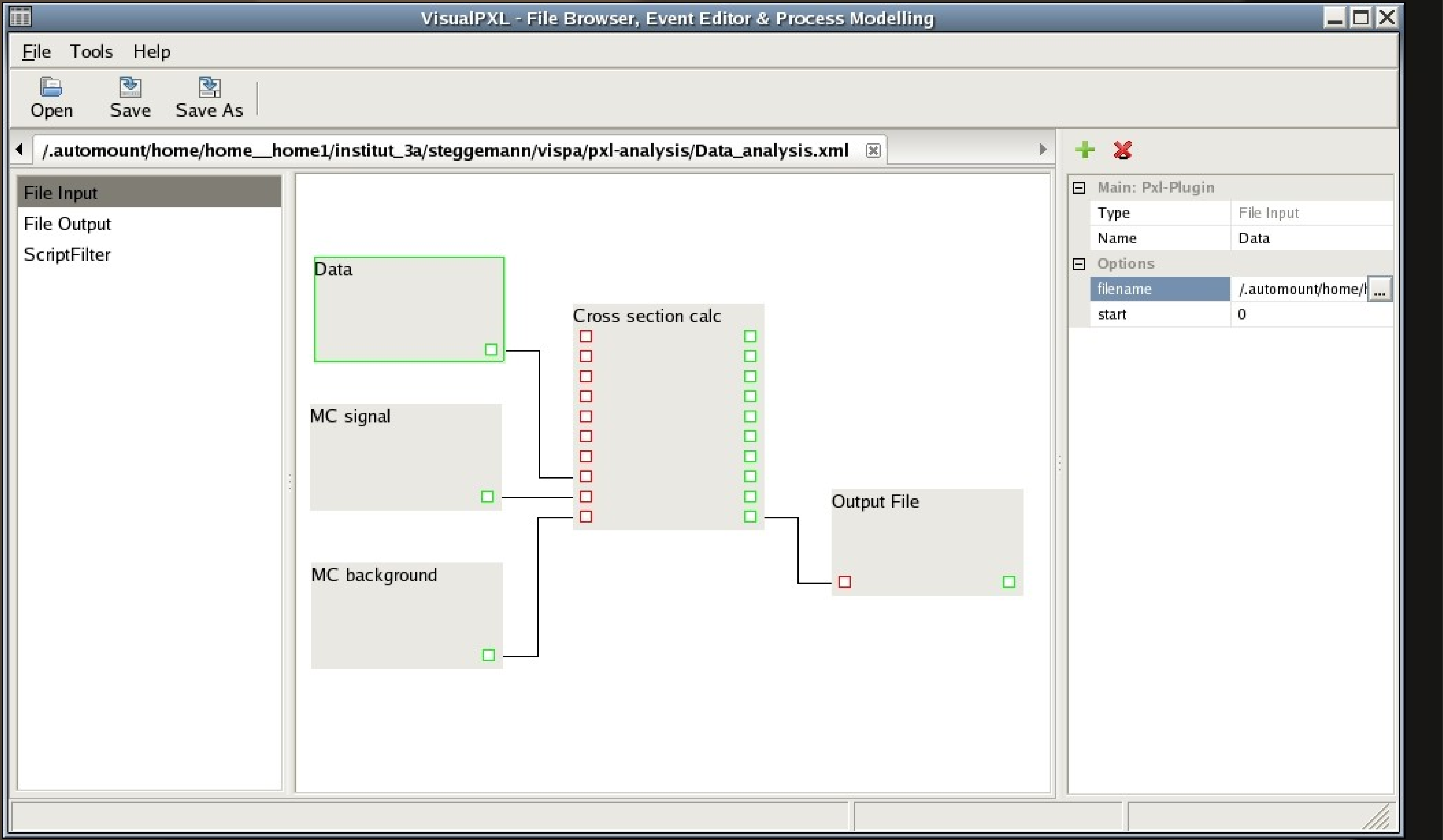
NumEle	0
NumJet	2
NumMET	1
NumMuon	2
Process	Zmumu
Type	Rec



Create histogram using **PyROOT**

If needed repeat prototyping, execution, verification

More complex analysis



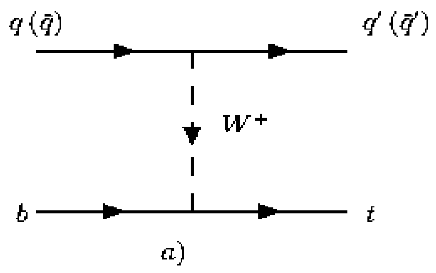
/.automount/home/home__home1/institut_3a/steggemann/vispa/pxl-analysis/Data_analysis.xml

⊟ Main: Pxl-Plugin

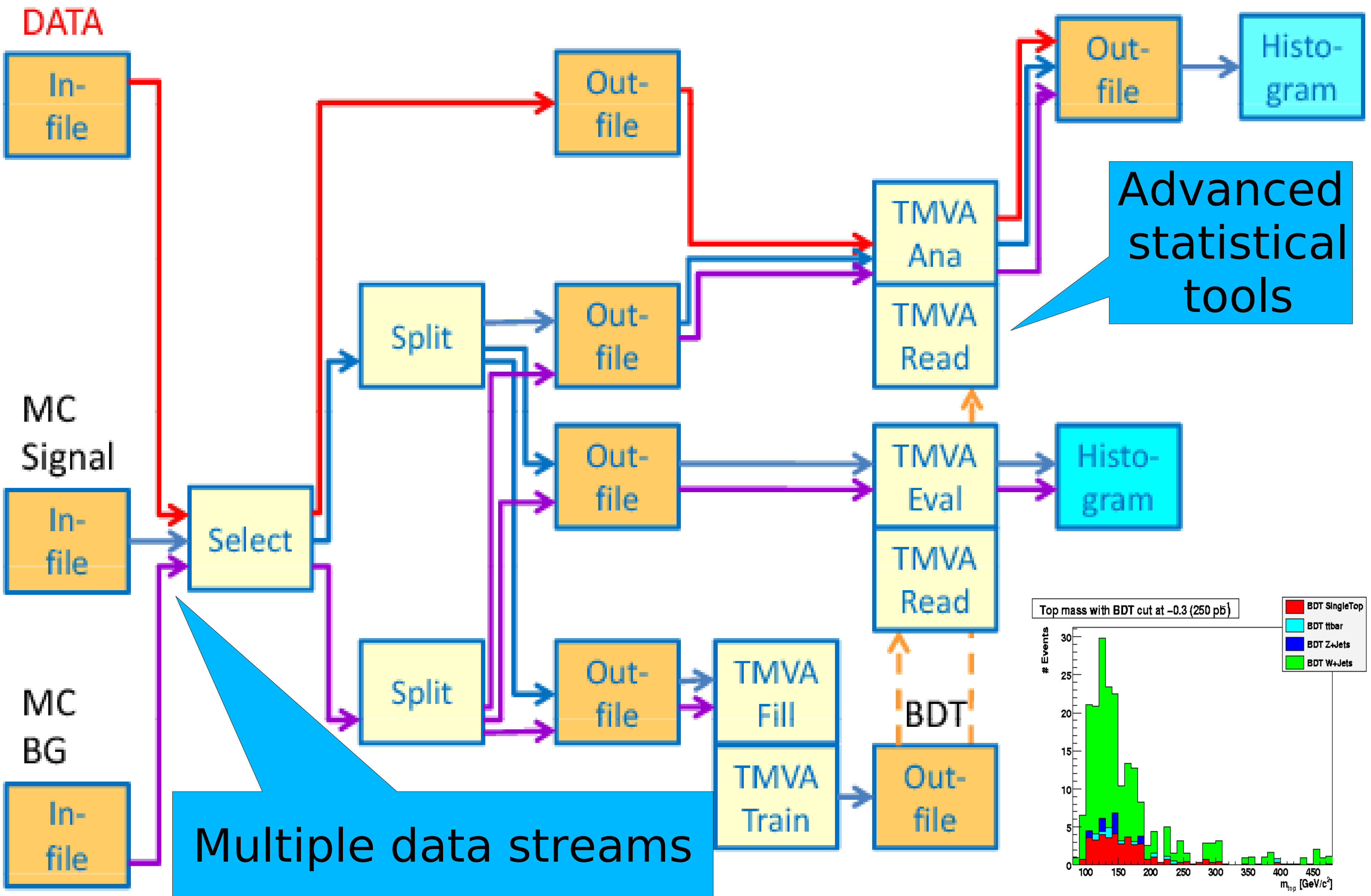
Type	File Input
Name	Data

⊟ Options

filename	/.automount/home/h
start	0



Single top quark analysis

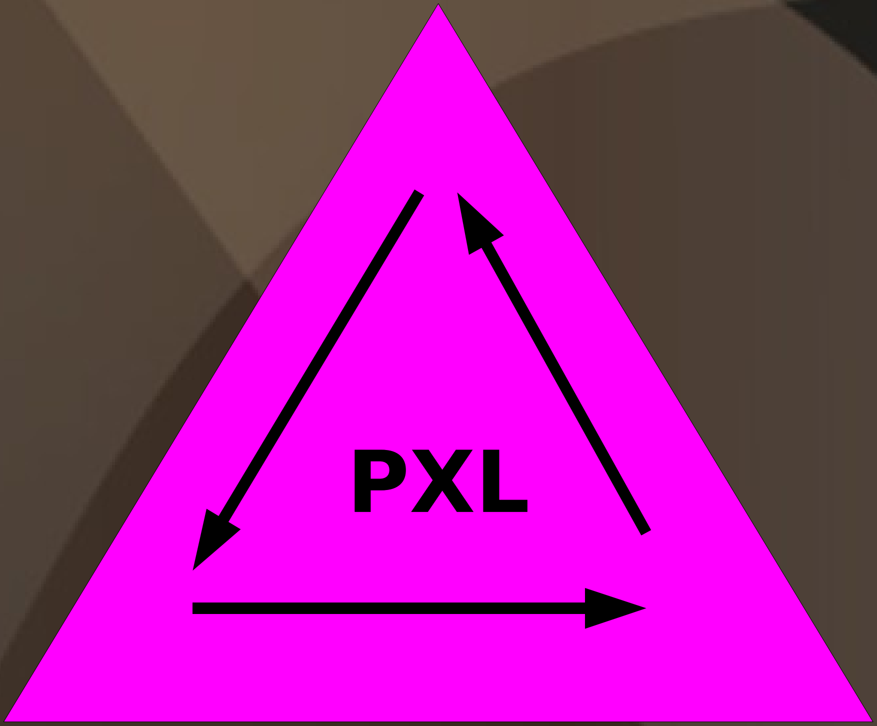


Advanced statistical tools

Multiple data streams

Summary: Analysis flow with VISPA

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

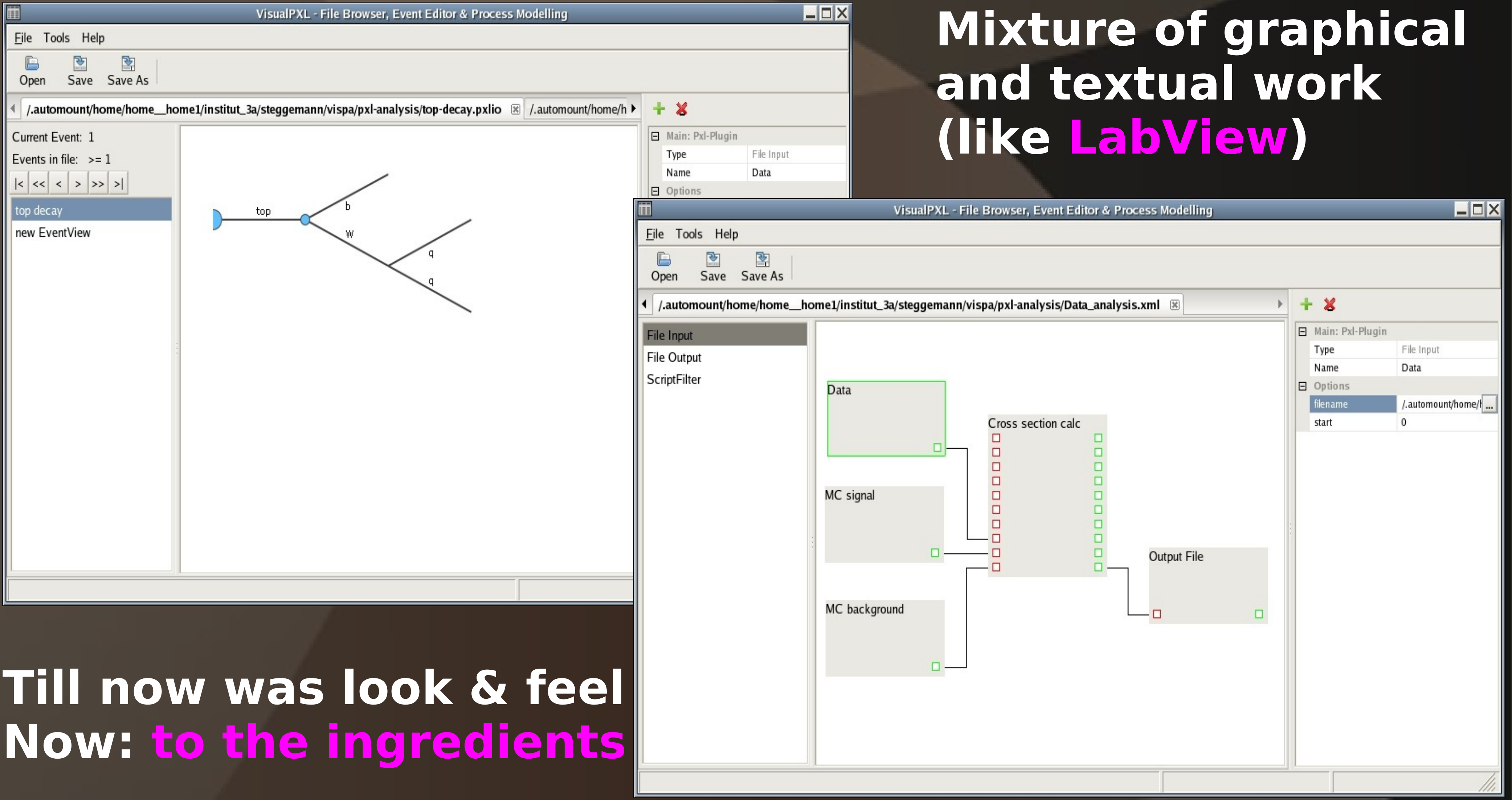


Execution
(XML or Python steering)
(interactive or batch)

Verification
(Event Browser)
(ROOT histograms)

VISPA: Visual Physics Analysis

Novel Concept of making physics analysis



VisualPXL - File Browser, Event Editor & Process Modelling

/.automount/home/home__home1/institut_3a/steggemann/vispa/pxl-analysis/top-decay.pxlio

/.automount/home/h

Current Event: 1

Events in file: >= 1

Main: Pxl-Plugin	
Type	File Input
Name	Data
Options	

VisualPXL - File Browser, Event Editor & Process Modelling

/.automount/home/home__home1/institut_3a/steggemann/vispa/pxl-analysis/Data_analysis.xml

Main: Pxl-Plugin	
Type	File Input
Name	Data
Options	
filename	/.automount/home/h
start	0

Mixture of graphical and textual work (like LabView)

Till now was look & feel

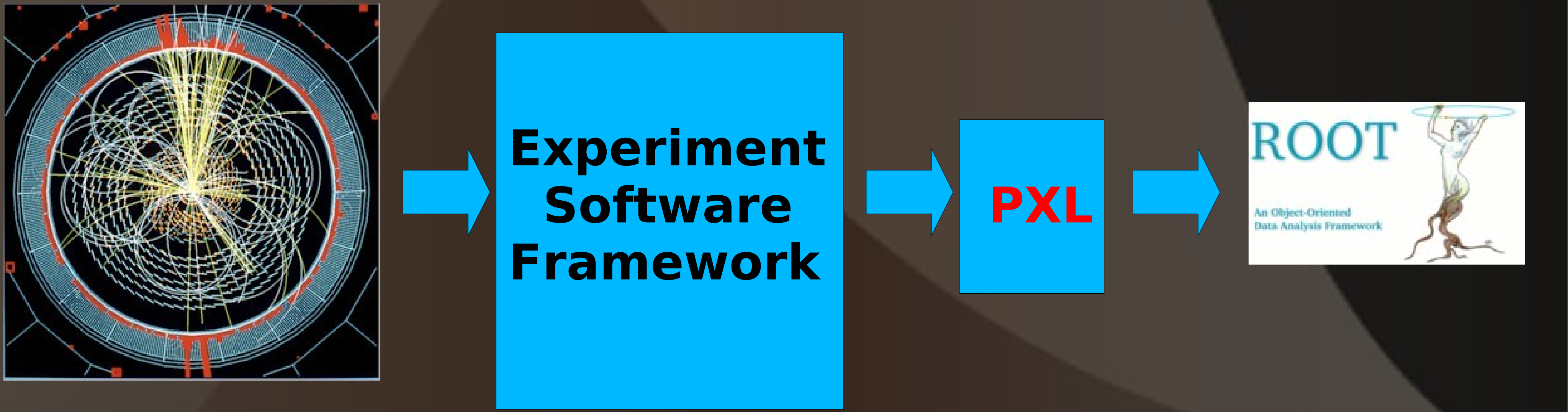
Now: to the ingredients

VISPA Packages

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

PXL (Physics eXtension Library)

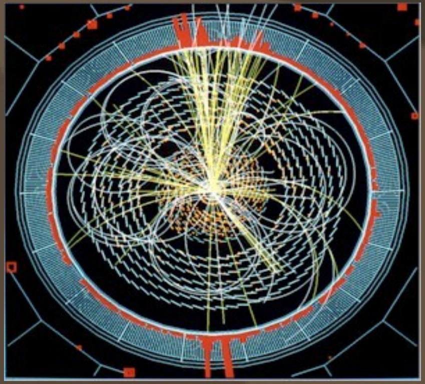
- C++ toolkit for high-level physics analysis
- Provides underlying functionality for Visual Physics Analysis (VISPA)
- Version 2.0 (2008)
- Successor of PAX (Physics Analysis Expert) (2002-2007)

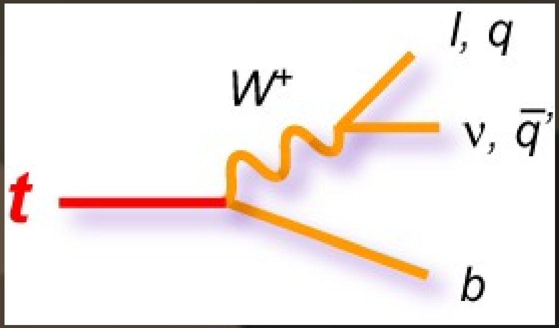


Experiment Software Framework

PXL

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, arbitrary pointers)

PXL key components: UserRecord

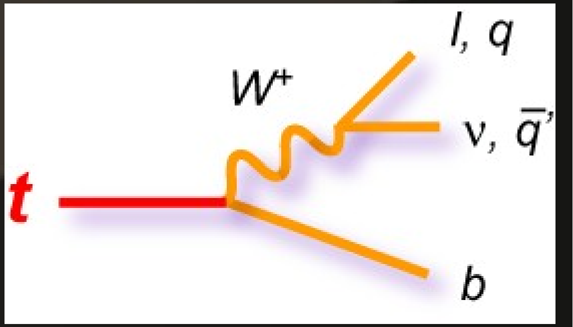
- PXL physics object is not only a fourvector
- PXL physics object has also a UserRecord where user data are stored:

  -- these are pairs of names and all basic C++ types (int, double, string)

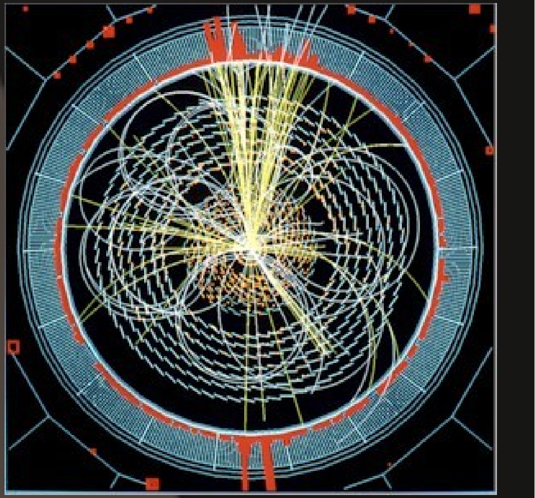
  -- it can be e.g. data from the condition databases etc.

- Deploys Copy-On-Write mechanism
- Flexible, extensible 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:**
  - **Have an arbitrary name *(string)***
  - **Between any objects**
  - **Provided but not guaranteed**

Input / Output

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

PXL: Summary

- PXL provides underlying functionalities for VISPA
- Powerful C++ tool with key ingredients as
    - event container
    - relation management
    - user record
    - fast I/O
- Works for any experiment

Python interface to PXL: why Python?

- Less Python code compared to C++
- Dynamic type allocation
- Python has automatic memory management
- C++ is compiled to an executable, Python does not need compilation
- Python has an interactive mode for testing
- Python code is easy to read
- Both object oriented, work on multiple platforms
- Python is 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

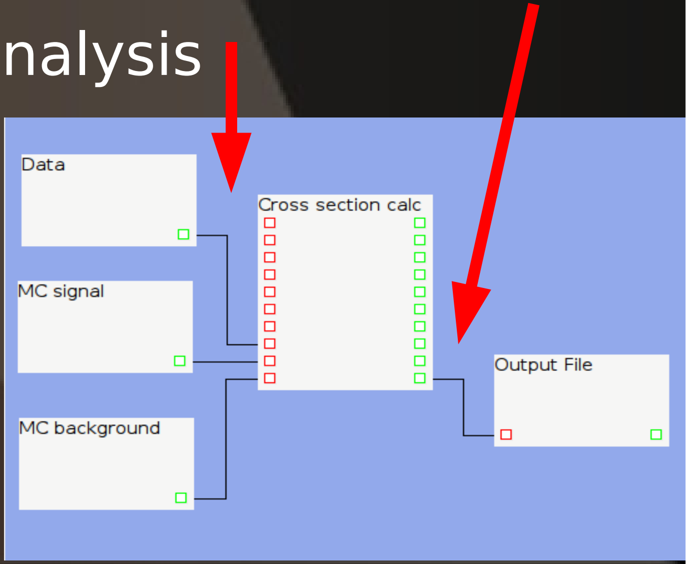
  ➔ save and restore any state of the analysis

- **PYTHON configuration**

  ➔ high flexibility

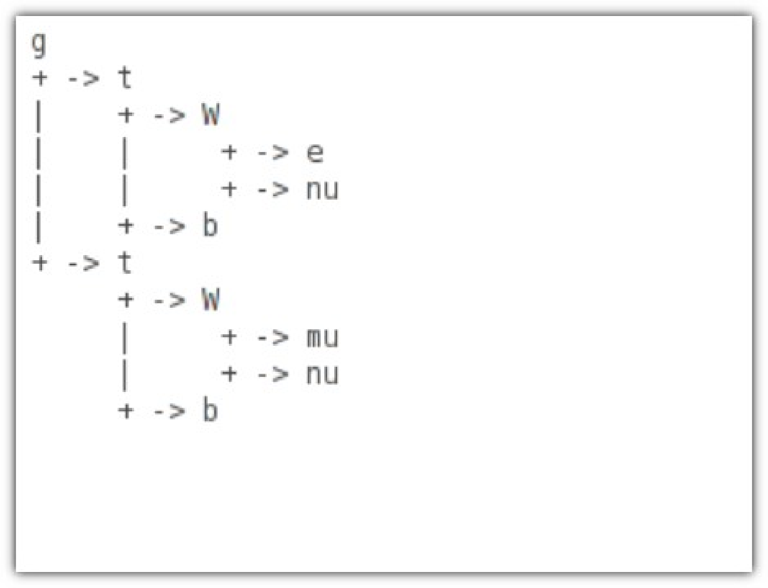
  ➔ easy-to-read

Interface: Event Container



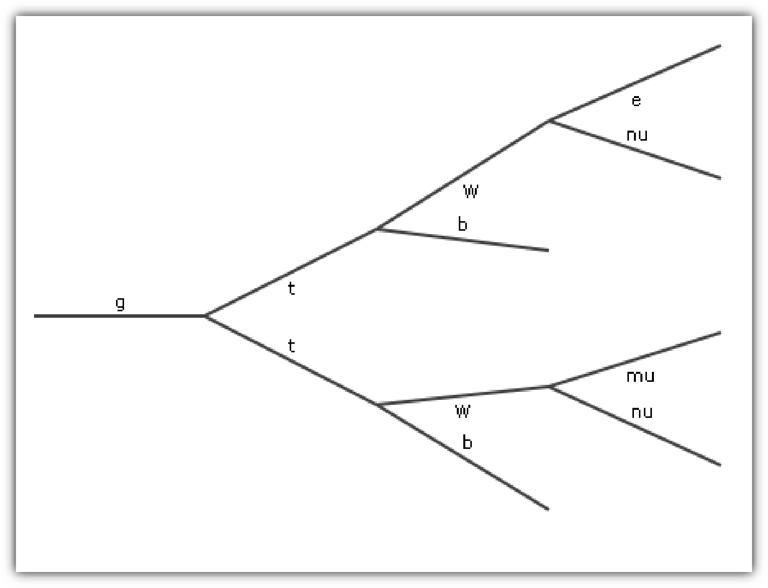
Graphical platform: autolayout algorithm

Hard to read text output

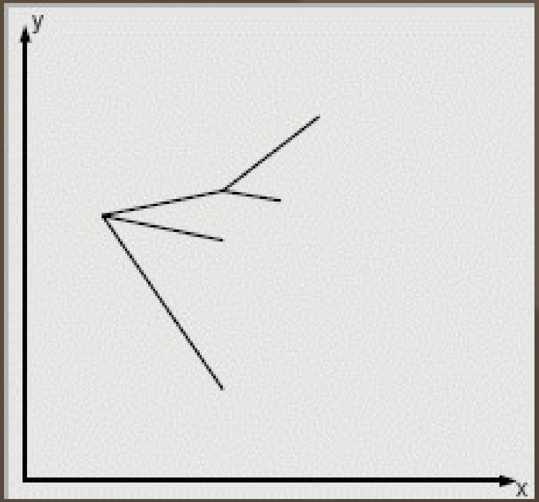


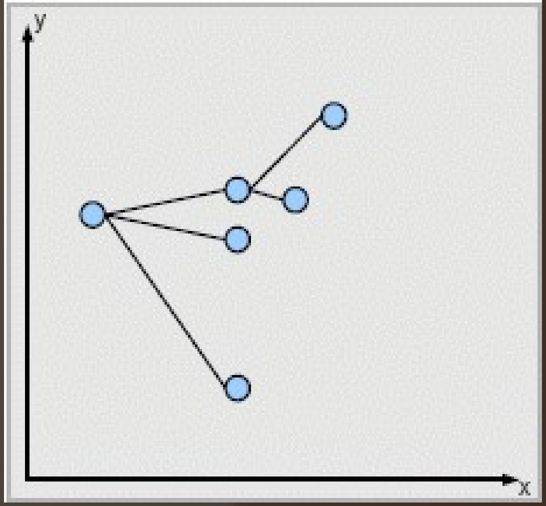
```
g
+ -> t
|   + -> W
|   |    + -> e
|   |    + -> nu
|   + -> b
+ -> t
    + -> W
    |    + -> mu
    |    + -> nu
    + -> b
```

Desired, easy to read decay tree



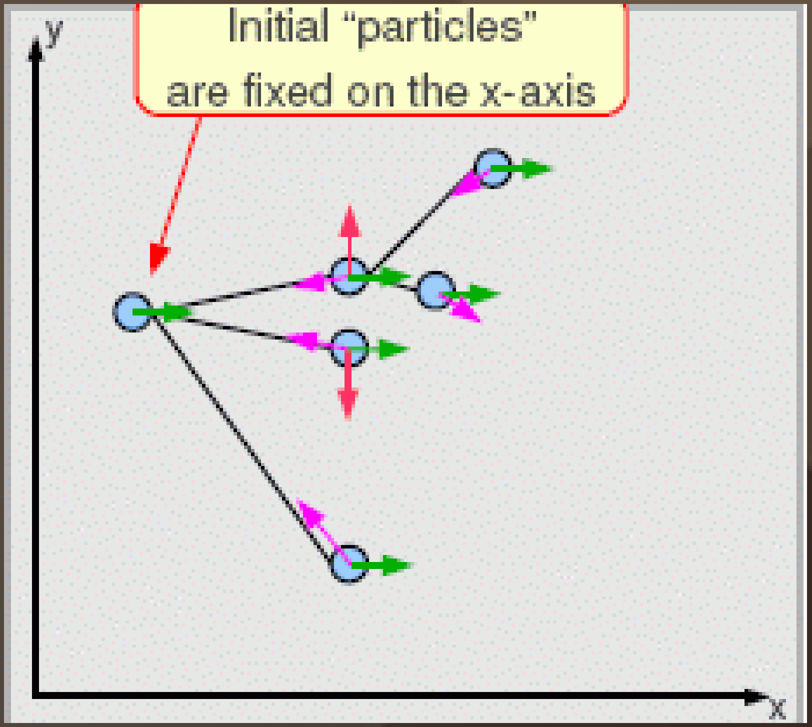
Autolayout algorithm





- Simple particle decay chain in a random initial layout
- Create nodes with the following properties: mass, position, velocity, applied forces

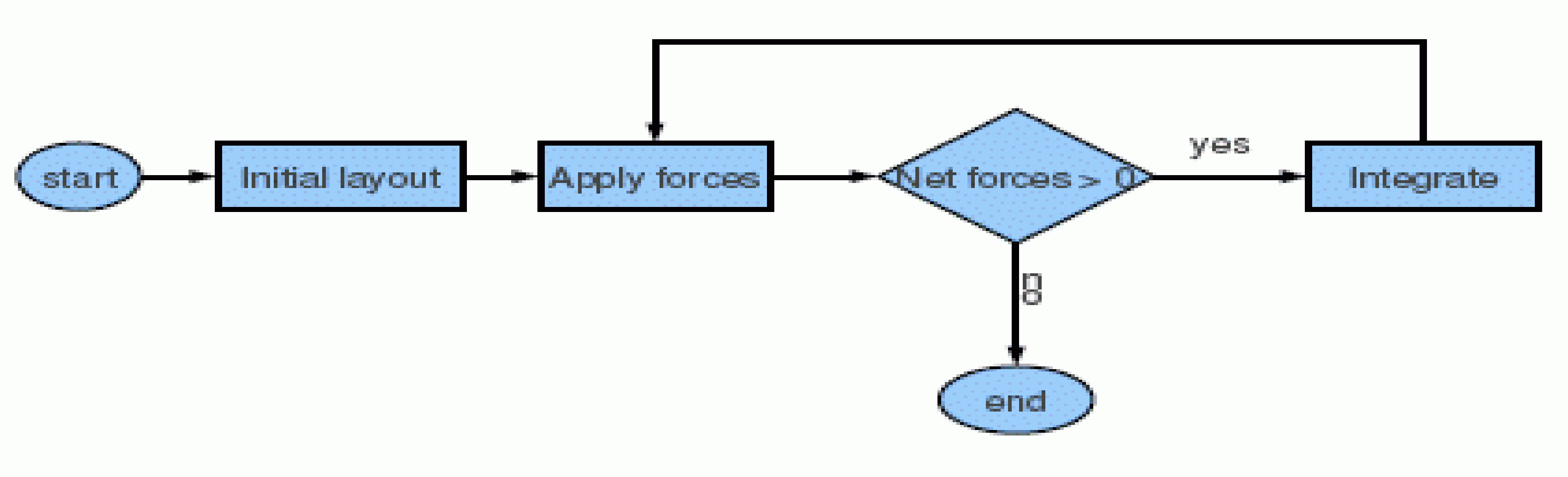
Autolayout algorithm based on model forces



Initial "particles" are fixed on the x-axis

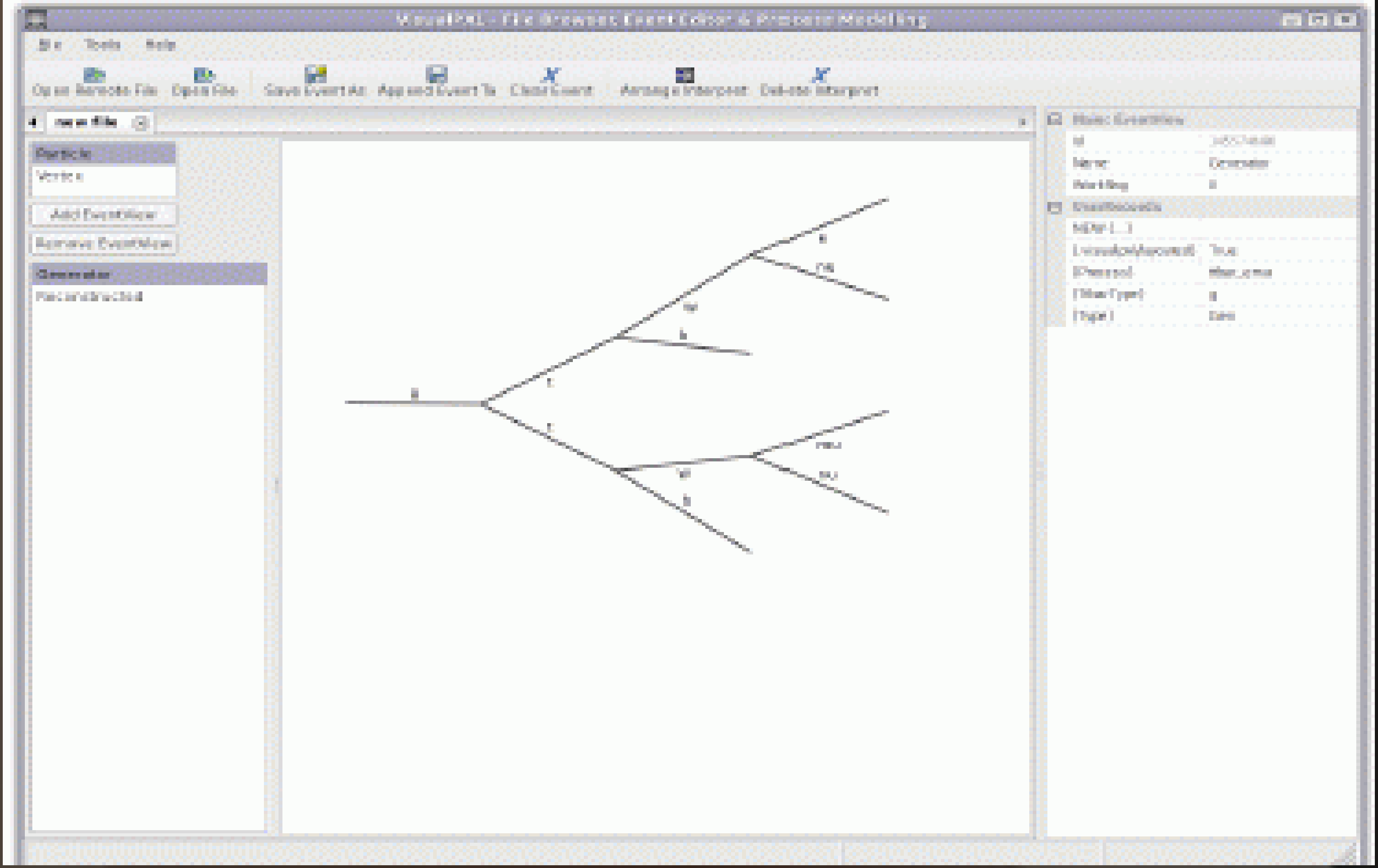
- **Constant forces** to all nodes
- **Repelling forces** between close node
- **Spring forces** to all daughter nodes
- **Friction forces** to all moving nodes

Autolayout algorithm based on model forces



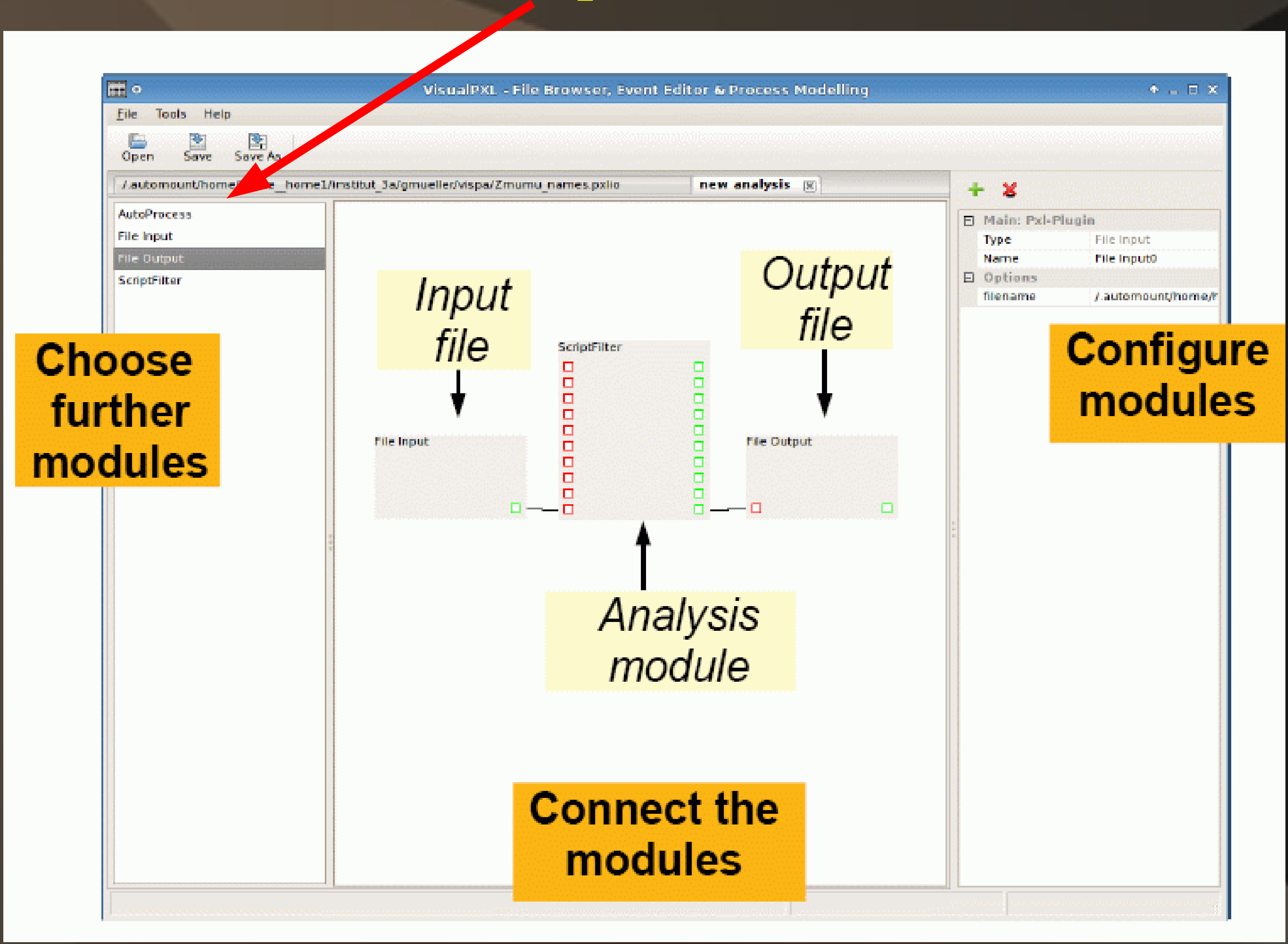
After some iterations all nodes moved into the positions where all forces on the node cancel each other out. After this the system is in stable state

Autolayout algorithm based on model forces: final result



Name	Generator
WorkFlag	0

Autoprocess



Type	File Input
Name	File Input0
filename	/automount/home/r

Input file

Output file

Analysis module

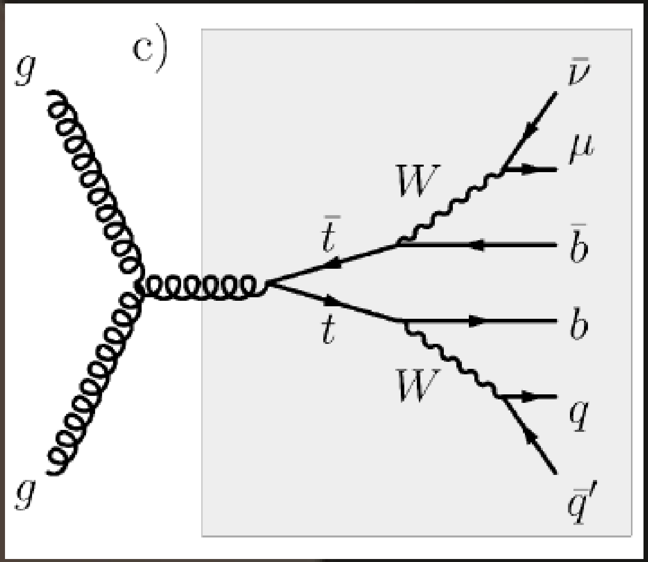
Choose further modules

Configure modules

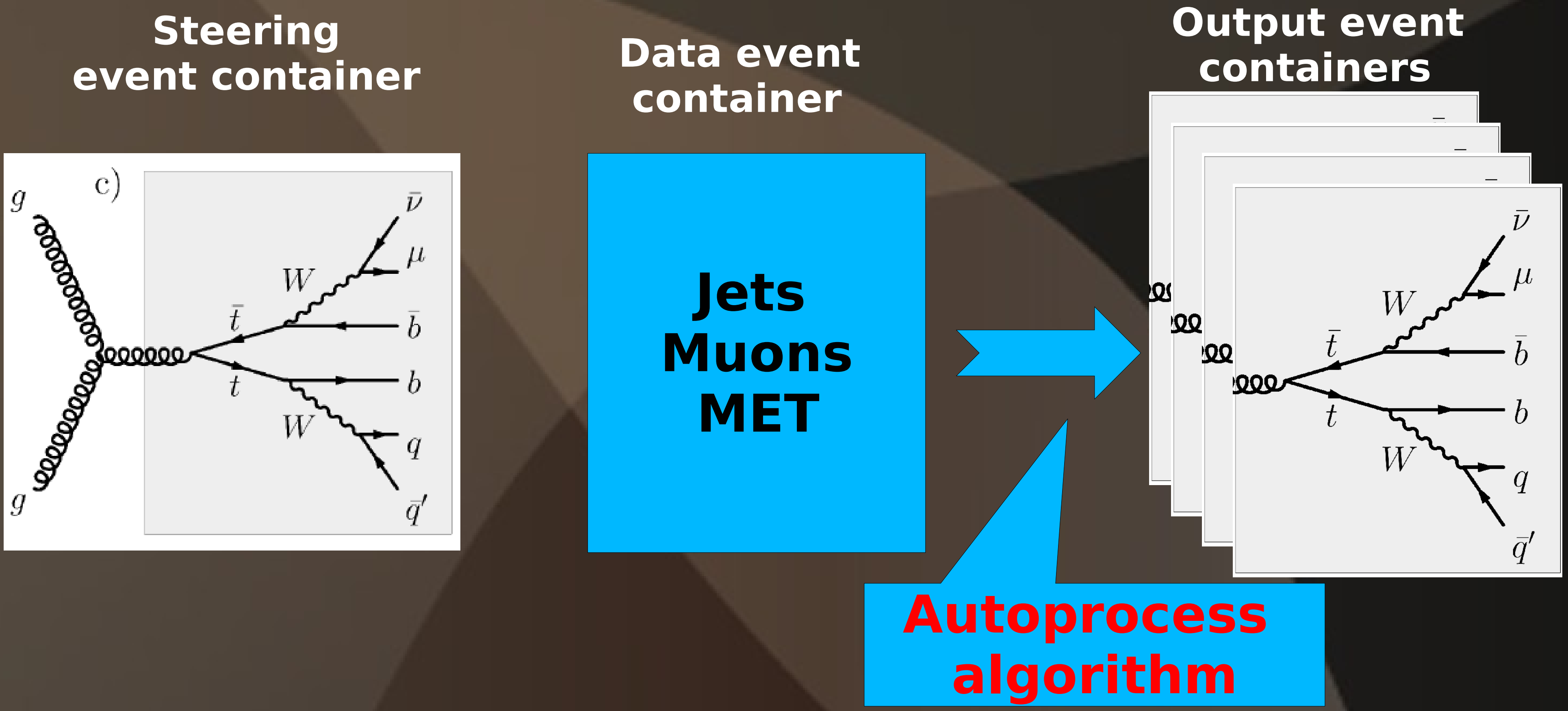
Connect the modules

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



Autoprocess



Steering event container

Data event container

Output event containers

Jets
Muons
MET

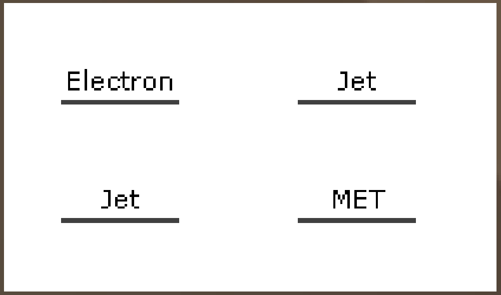
Autoprocess algorithm

- Steering by an event container* holding (part of a) Feynman diagram
- Intuitive for physicist

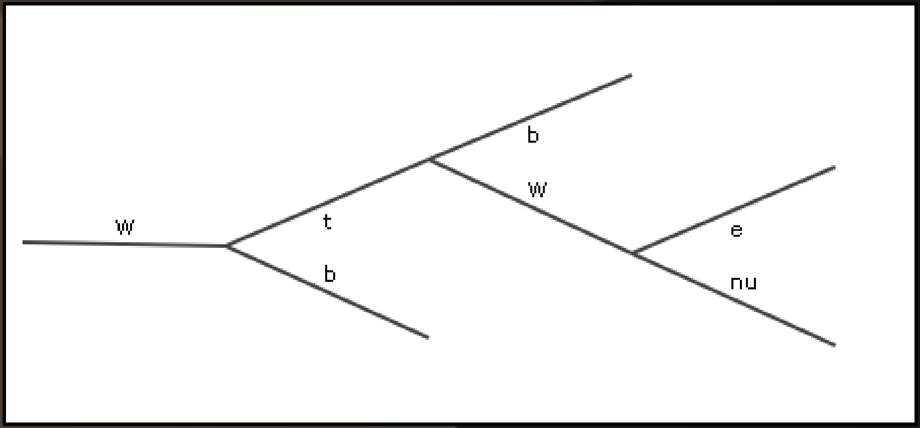
*PXL event container and particles (including relations)

Example: top decay

Reconstructed particles



Steering template

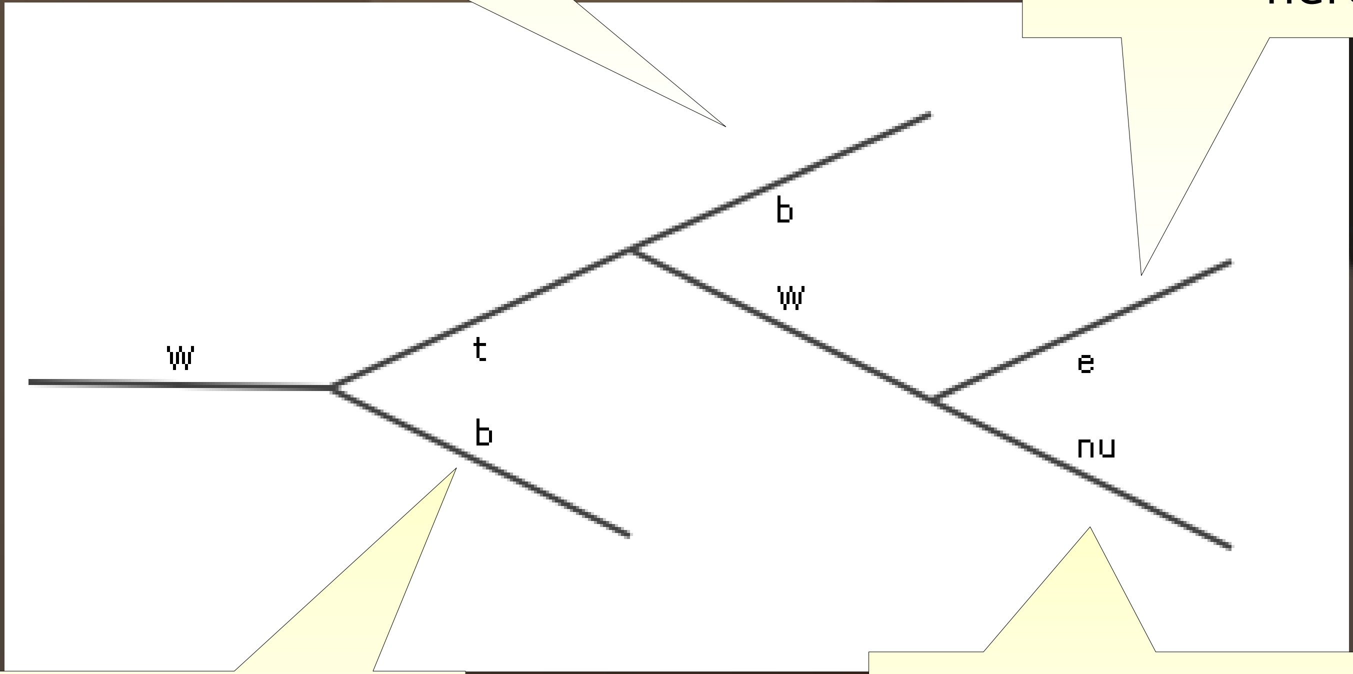


- **First step: “design” the event template of a physics process**

Designing the event template

Place particles with the name “Jet” here

Place reconstructed particles with the names “Electron” or “Muon” here



Place particles with the name “Jet” here

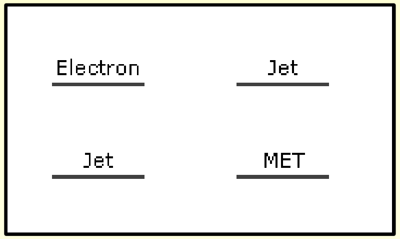
Use the MET for this particle and calculate the Pz-Solutions using a W-mass of “80 GeV”

Do it in Python, can apply cuts...

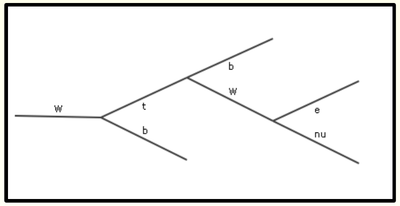
Example

Input:

Reconstructed particles



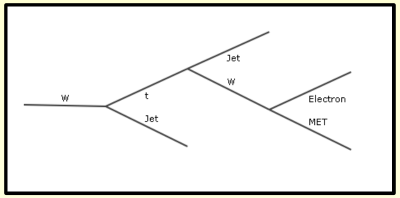
Steering template



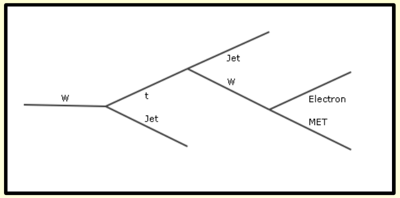
Autoprocess

Output:

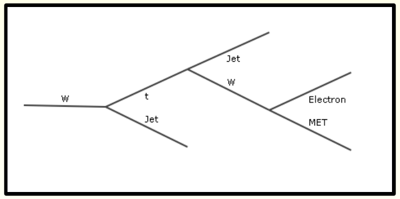
m_top = 187 GeV



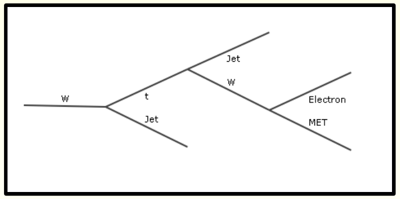
m_top = 215 GeV



m_top = 246 GeV

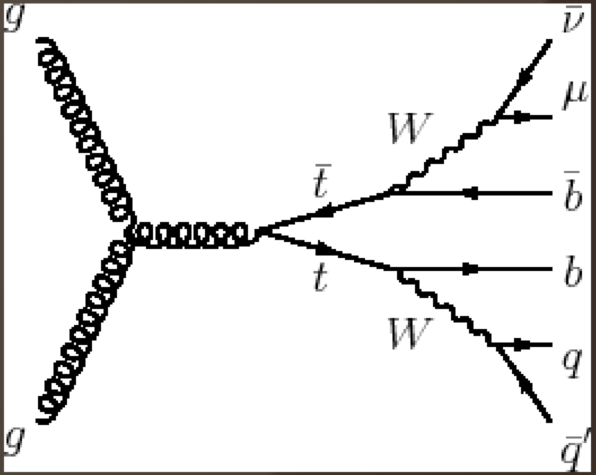


m_top = 173 GeV

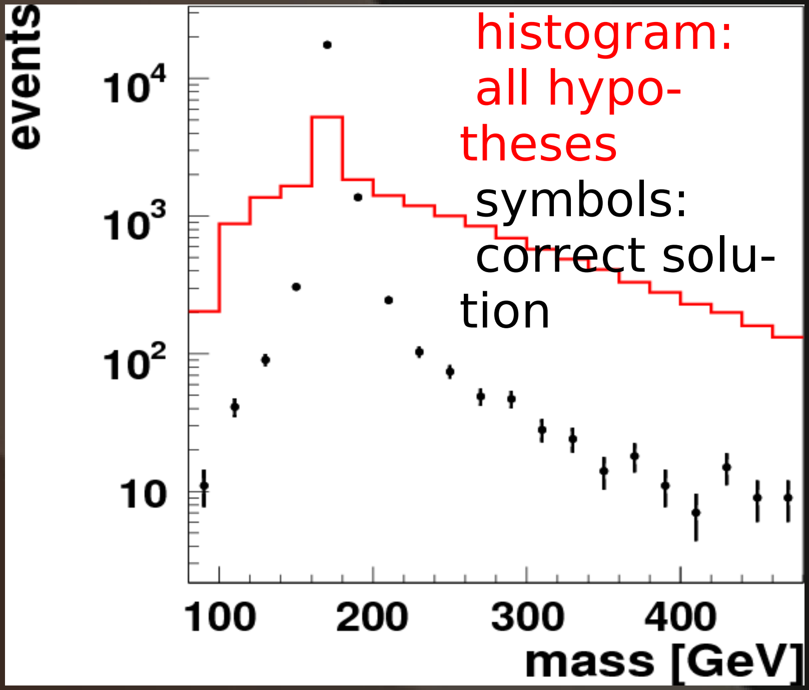


Autoprocess: summary

- 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
  - First applications for CMS analysis
- PXL provides underlying functionalities for VISPA
  - Powerful C++ tool with key ingredients as event container, relation management, user record and fast I/O
- Autolayout algorithm
- Autoprocess: powerful tool for automatic calculation of particle cascades

Summary II

- All software is continuously maintained
- Fully documented:

  - Manual for PXL:

  http://pxl.sourceforge.net/manual.pdf

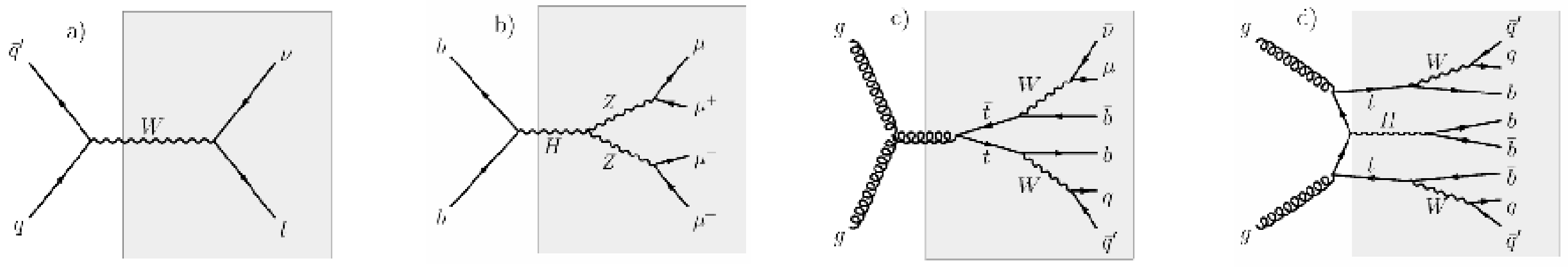
  - Doxygen
- Available online at http://pxl.sourceforge.net
- Look & feel version for MAC is provided
- Publications:

  http://arxiv.org/abs/0810.3609

  http://arxiv.org/abs/0801.1302v2

Backup

Autoprocess: performance

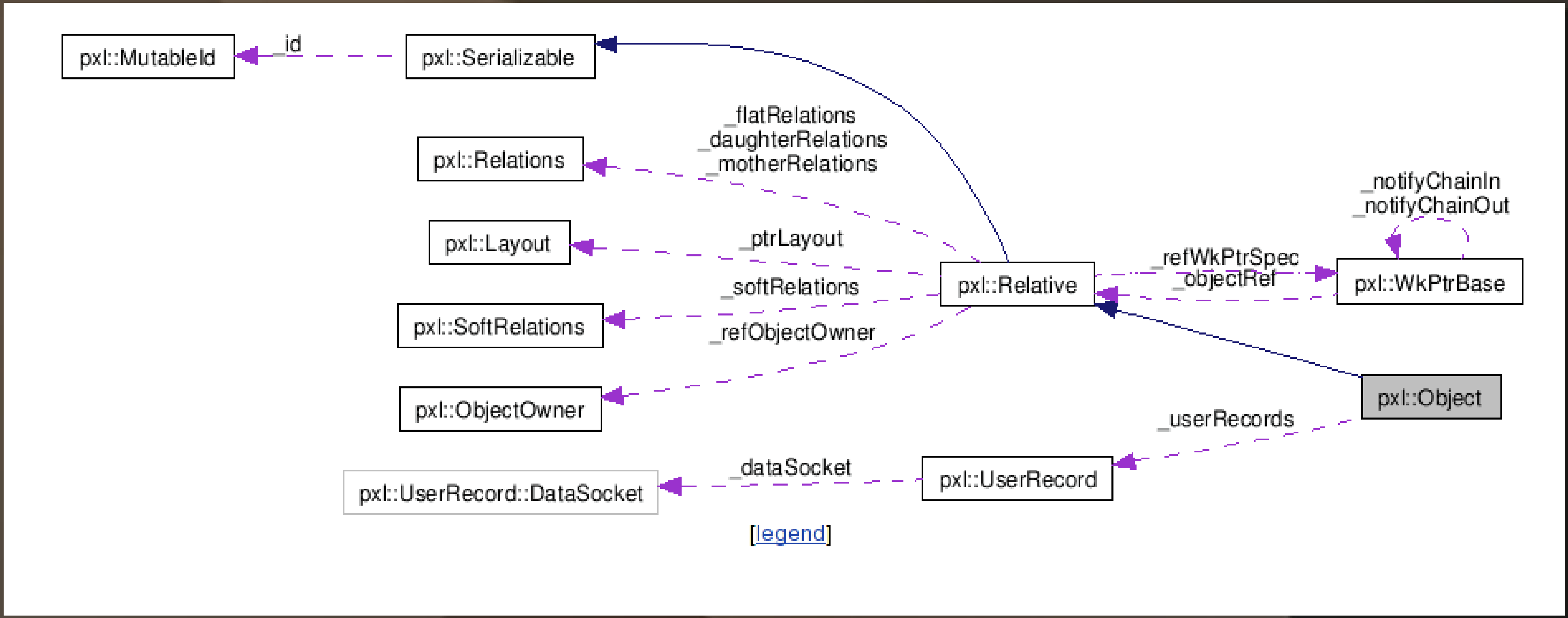


Physics process	number of particles	number of vertices	number of configurations	Time/event [ms]	Mem. alloc. [MByte]
$W \to l\nu$	3	1	2	0.06	< 1
$H \to 4\mu$	7	3	3	0.4	< 1
$t\bar{t} \to \mu\nu 4j$	11	5	24	2.3	< 1
$Ht\bar{t} \to 8j$	13	5	5040	411	36

on a standard personal computer:

- ~20 µs per reconstructed decay vertex
- ~0.6 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)***