

# Views On CINT

Difficult to maintain due to complexity

Core part of ROOT with amazing capabilities – and people tend to ignore both facts

In the middle of a major transformation:

- Dictionary: on demand, no call wrappers, persistency
- Interpreter: Script pre-compilation and persistency, OO “byte” code for better maintenance
- CINT Objects, thread safety

# Views On ROOT

Few generic developments needed in “my” area:

- Live help system: THtml doc to the prompt
- THtml from Reflex DB, globals, friends, etc.

Generic:

- We should go beyond “users = HEP”. But how? Blocker is probably usability.
- Development too much driven by requests
- Growth is the worst improvement possible – find mechanism to integrate (external) packages without blowing up ROOT source

# GOOI – Graphical rOOt Interface

We need a non-expert, intuitive, standard UI

- prompt interface optional
- C++ optional
- optional knowledge of classes, collections,...

Use reflection to visualize data format (c.f. TBrowser)

Use graphical interface to design analysis, convert to C++, e.g. with Zdenek Culik's C++ generator

Parts re-usable (selections, algos, control histos)

Auto-iteration (“foreach electron”)

Select input data, save, load, run, view results

# GOOI – Graphical rOOt Interface

The screenshot displays the GOOI - TestAnalysis\_GOOI.root window. The interface includes a menu bar (File, Edit, View, Window, Help), a toolbar, and a status bar (TODO: layout dialog bar). The main workspace shows a workflow diagram with the following components:

- Input:** A green dashed box labeled `ptsumf`.
- Data flow:** A yellow arrow pointing from the `ptsumf` box to the output box.
- Output with x as input:** A red dashed box labeled `TH1F Sum of pT` with a small green box containing 'x' next to it.
- Condition:** A light blue dashed box containing the text "Condition: `nch > 4`".
- Filtered Output:** A red dashed box labeled `TH1F Sum of pT for nch > 4` with a small green box containing 'x' next to it.

The workflow is contained within a larger light blue dashed box. A yellow arrow also points from the `ptsumf` box to the filtered output box, indicating that the data flow is filtered by the condition `nch > 4`.

On the left side, there is a file browser showing the directory structure:

- ROOT Files
  - mlpHiggs.root
    - bg\_filtered
      - acoln
      - acopl
      - minvis
      - msumf
      - nch
      - ptsumf
      - qelep
    - sig\_filtered

Below the file browser is a "Tools" section with the following items:

- TH1F
- TH2F
- Condition
- ForEach

The status bar at the bottom left shows "Ready" and the bottom right shows "NUM".

# GOOI - Objects

The screenshot displays the GOOI (Graphical Object-Oriented Interface) environment for ROOT. The main workspace shows a workflow diagram with the following components:

- ForEach: Jets**: A block that iterates over a collection of jets. A callout box labeled "Collection reference" points to this block.
- Condition: Muons**: A block that filters jets based on the presence of muons.
- @HasMuon**: A block that takes two inputs, "IN: jet" and "IN: muons", and produces an output "OUT: hasMuon".
- Re-used algorithm block**: A TH2F histogram titled "Jet Energy vs emf for muon-jets". It has two inputs, "x" and "y", and one output. A callout box explains it is a "Re-used algorithm block with 2 inputs, one output".

On the left side, the "Tools" panel contains several blocks: TH1F, TH2F, Condition, and ForEach. The "ROOT Files" panel shows a directory structure for "mlpHiggs.root" containing sub-directories "Jets" and "Muons".