



Axel Naumann, CERN, PH-SFT for the ROOT Team

ROOT 6

ACAT 2014, Prague



Content

- ❖ The “6”
- ❖ C++ support
- ❖ TTreeReader
- ❖ TFormula 2.0
- ❖ Graphics 2.0
- ❖ Current Plans



The “6” in ROOT 6

Assessing the Past

- ❖ Heart of ROOT ≤ 5 was CINT: served amazingly well
- ❖ Limitations
 - ❖ C++ support: coverage of C++ constructs, C++11, correctness, diagnostics
 - ❖ scalability: too many types brought ROOT to a limit
 - ❖ robustness: ill-formed code could make ROOT crash
 - ❖ design: difficult to use in multi-threaded programs

cling clang llvm

- ❖ ROOT 6 has new interpreter cling
 - ❖ using production-grade compiler (clang+llvm) as hidden library; interpreting from #includes without dictionaries
 - ❖ just-in-time compiles to “shared library in memory”: virtual functions, function pointers - all native!
 - ❖ robust after four years of development
- ❖ ROOT now inherits C++ features from clang: diagnostics! C++14 support! correctness! speed!

Example

- ❖ Correctness; support of language features

```
root [0] #include <vector>
root [1] #include <map>
root [2] #include <string>
root [3] vector<map<string, vector<string>>> a;
root [4] a.push_back(map<string, vector<string>>());
root [5] a[0]["A"].push_back("B");
root [6] a[0]["A"][0].length() == 1
(bool) true
```

Example

❖ Quality of diagnostics

```
[cling]$ int MissSpelled = 1;
[cling]$ printf("%g\n", MissSpelled);
input_line_6:2:17: error: use of undeclared identifier 'MissSpelled';
did you mean 'MisSpelled'?
printf("%g\n", MissSpelled);
                ^~~~~~
                MisSpelled

input_line_5:2:6:      'MisSpelled' declared here
int MissSpelled = 1;
  ^

input_line_6:2:17: warning: format specifies type 'double' but the ar
gument has type 'int' [-Wformat]
printf("%g\n", MissSpelled);
      ~~      ^~~~~~
      %d
```

C++ Support

C++ Standards

- ❖ C++ Standards are published at higher frequency: C++11 feels “just out”, C++14 is about to come
- ❖ C++ Standards are implemented “live”: C++14 *already* available in GCC, clang
- ❖ Experiments ask for C++11 support
- ❖ Cling gives sustainable way to adapt: rely on clang!

Why C++ 11?

- ❖ Increased clarity of code
- ❖ Increased robustness
- ❖ Increased performance
- ❖ Increased standard library size (a good thing!)
- ❖ Increased appeal to contributions!

TTreeReader

Accessing TTree Data in the Past

- ❖ Many TTree interfaces are fragile (void*& etc)
- ❖ Painful to extract data from existing TTree
 - ❖ painful to teach!
- ❖ Several key TTree improvements disabled by default
- ❖ Needed fast, robust, usable interface

Using the TTreeReader

```
1. TTreeReader myReader("ntuple", myFile);
2. TTreeReaderValue<Float_t> myPx(myReader, "px");
3. TTreeReaderArray<Jet> myJets(myReader, "jets");
4. while (myReader.Next()) {
5.     myHist->Fill(*myPx);
6.     for(auto &&j: myJets) { myHist2->Fill(j.E()); }
7. }
```

- ❖ Diagnoses type mismatch
- ❖ Recommended way of accessing TTree data for mere humans

TFormula 2.0

We have a Compiler-Library!

- ❖ Why not use just-in-time compiler? Thus: the all new TFormula; part of ROOT 6.04
 - ❖ based on cling / clang / llvm: compiles code - speed! diagnostics!
- ❖ ROOT 6 (with current TFormula) enables full-fledged C++ in TFormula / TF1
 - ❖ TF1("CosICan",
[(double* x, double*p) { return p[0]*cos(x[0]); },
0., 1., 1)

Graphics 2.0

Graphics to Latex

- ❖ Can store graphics as Latex document
 - ❖ simply `canvas->Print("plot.tex")`
- ❖ Resulting file can be included in Latex documents, matching fonts and styles
- ❖ Uses PGF / TikZ

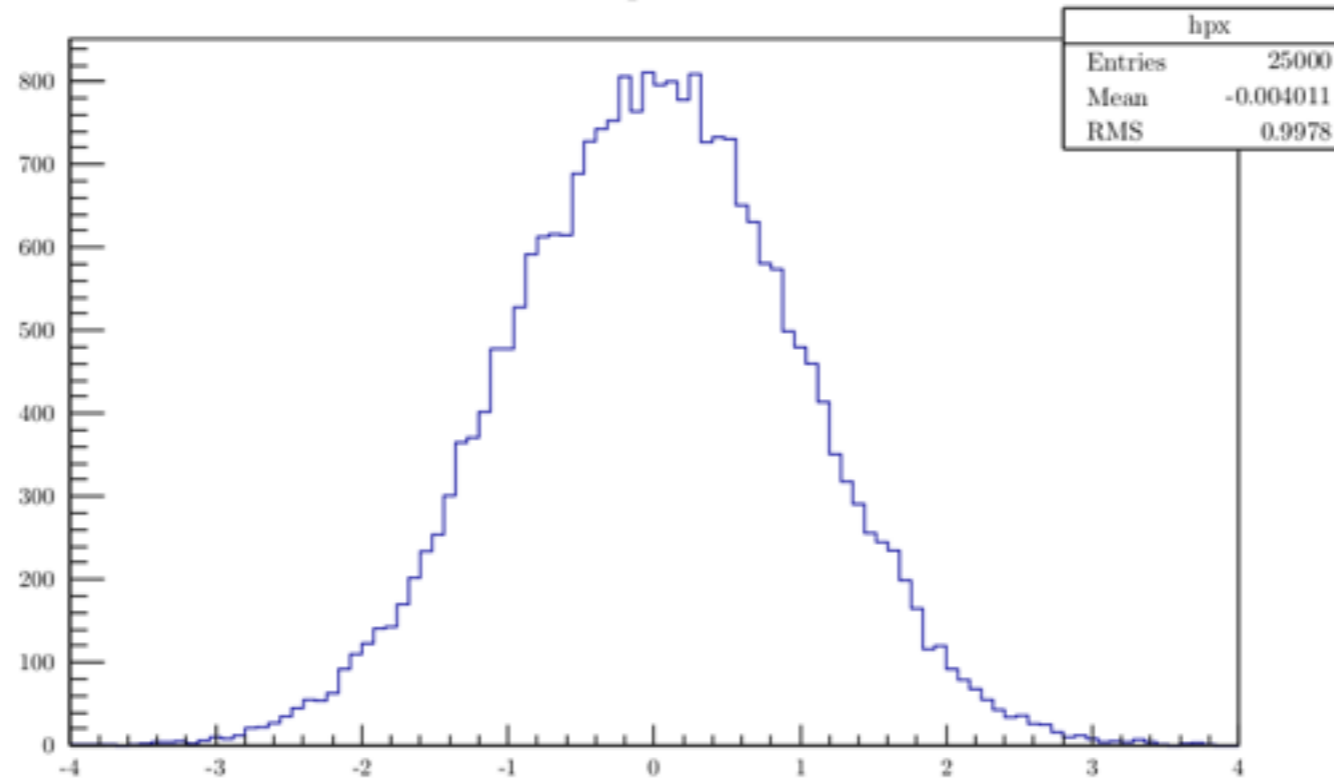
Graphics to LaTeX

A simple LaTeX example

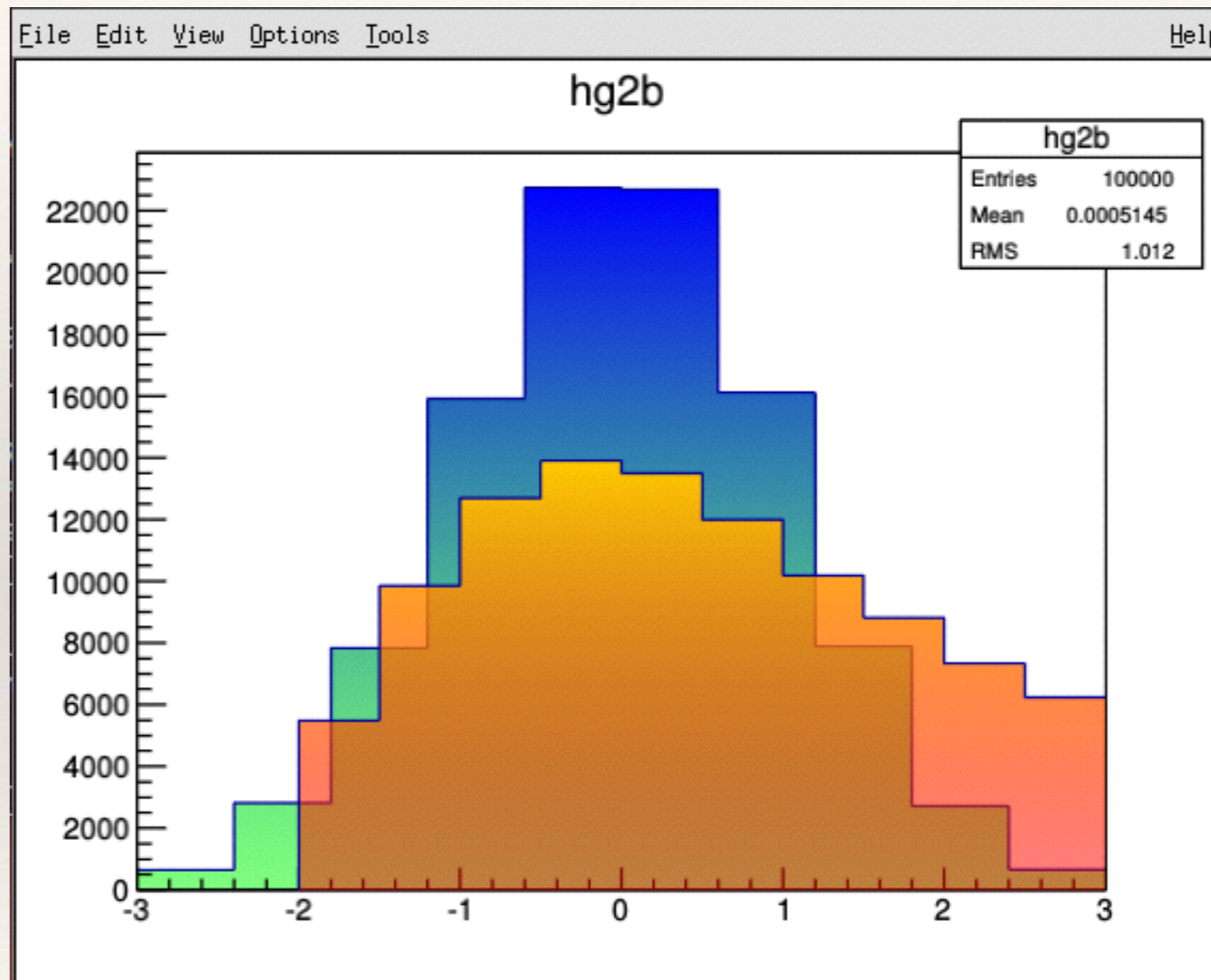
August 28, 2014

The following image has been generated using the TTeXDump class:

This is the px distribution



Transparency and Shading

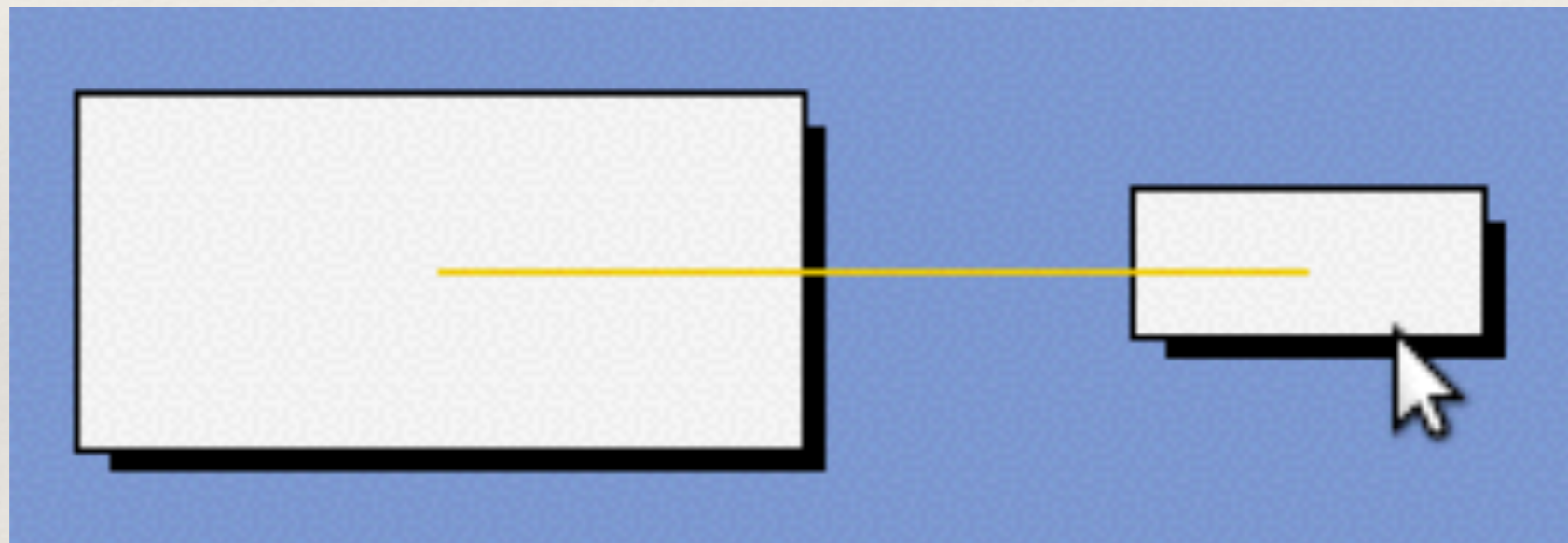


Transparency and Shading

- ❖ Requires support from graphics engine
 - ❖ currently implemented for OSX / Cocoa, OpenGL
 - ❖ planning to make OpenGL default - at least in ROOT 6
- ❖ Also currently no proper PDF export yet

Graphics UI

- ❖ Implemented current interactivity features:
 - ❖ guides for object placement



- ❖ smooth axis zoom

Current Plans

Old Interface Jargon

- ❖ C++ has evolved past TObject*-based interfaces; see also feedback from last ROOT workshop and experiments
- ❖ issues are ownership, type safety, lack of interface clarity cause crashes in user code instead of compile time errors
- ❖ little information on threading; difficult to optimize

Examples for Old Interface Jargon

- ❖ `tree->Branch("event", &pEvent);`
 - ❖ what should `pEvent` be (very difficult to answer!)
 - ❖ who owns whatever `pEvent` points to?
- ❖ `new TTree("T", "a tree");`
 - ❖ relies on context: current file!
 - ❖ might change ownership of current file
 - ❖ who owns that tree - the file?

New Interface Jargon

- ❖ Current C++(14): more precise and expressive interfaces
 - ❖ type safety through templates (no problem with cling)
 - ❖ ownership through shared / unique pointers
 - ❖ multithreading: const methods == no global state
 - ❖ maybe `shared_ptr<Event> pEvent = tree->MakeBranch<Event>("event");`
- ❖ Plan: start with new histogram interfaces, old will use new behind the scenes

Old and New Interfaces

- ❖ Will allow for transition period
- ❖ First time for ROOT, many open questions:
 - ❖ how do we deprecate?
 - ❖ how do we smooth, encourage, and track migration?
 - ❖ how important is write old, read new? And vice versa?
- ❖ We need discussions and feedback - else we just do what we want! ;-)

Releases

- ❖ ROOT 6.00 published May 30, 2014 - require C++11 now
- ❖ ROOT 6.02 scheduled for end September; targeted to LHC frameworks for Run 2
- ❖ ROOT 6.04 scheduled for early 2015, plans:
 - ❖ new JIT engine for exceptions, inline asm
 - ❖ hopefully better (CPU, memory) dictionaries / type database (utilizing clang's "C++ modules")
 - ❖ new TFormula; R interface

In Related News...

More!

- ❖ You just saw Vassil's clad which we hope to make available in ROOT soon!
- ❖ More ROOT in track 1 on Tuesday:
 - ❖ Vectorization libraries VDT, VC (Sandro)
 - ❖ Investigating alternative analysis approaches (Vassil)

Conclusion

- ❖ ROOT 6 is here!
 - ❖ wealth of new features due to new interpreter
 - ❖ still limitations, most notably in unloading
- ❖ ROOT 6 opens new doors, also for ROOT
 - ❖ interface modernization just starting
 - ❖ needs your feedback!