SFT-CORE-ROOT
Program of work

Rene Brun

18 March 2005

Preliminary remarks

- This preliminary program of work is mainly for ourselves to clarify several issues following the merge of SEAL and ROOT.
- We need more discussions before a public presentation end of March.
- I had no time to discuss this week with all package managers several points that must still be clarified.
- This presentation does not cover important packages, such as the geometry or the VMC.
- We will not have the time to go through all the slides (100!)
- Many thanks to Fons, Ilka, Lorenzo, Markus, Olivier, Philippe, Richard for their contribution.

Team structure

- **BASE**: System & CVS & DOC & Releases & Test suite & newsgroups
  Fons, Philippe, Pere,Bertrand, Ilka, Axel, Jose, Rene
- **DICTINT**: CINT, Dictionary, Interpreters
  Masa, Philippe, Markus, Stefan, Wim
- **IO**: Basic I/O & Trees
  Markus, Philippe
- **PROOF**: PROOF & xrootd & GRIDs
  Fons, Andreas, Maarten, Derek, Gerri, Marek , Guenter+ BaBar
- **MATH**: Maths,Histograms,Functions, Linear Alg
  Lorenzo, Andras, Anna,Eddy
- **GUI**
  Ilka, Valeriy, Bertrand, Valeri, Fons
- **GRAPHICS**: 2-D & 3-D graphics & Geometry
  Olivier, Richard, Andrei, Mihaela, Timur, Bruno

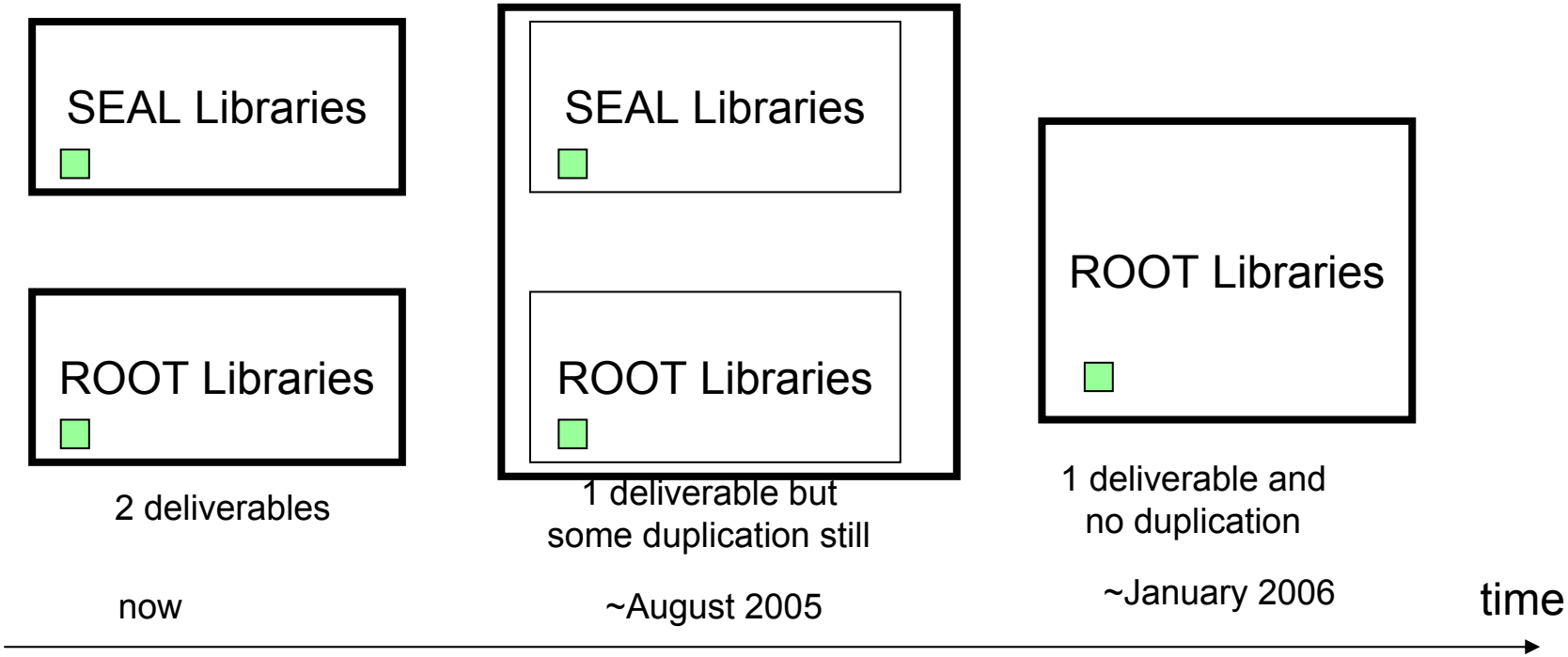
Ilka Antcheva
Maarten Ballinjtin MIT 80%
Bertrand Bellenot ALCAN 50%
Bruno Belbute 100%
Marek Biskup
Rene Brun
Philippe Canal FNAL
Olivier Couet
Christophe Delaere BE 10%
Valeri Fine BNL 10%
Markus Frank LHCb 50%
Gerri Ganis
Andrei Gheata ALICE 80%
Mihaela Gheata unpaid 50%
Masa Goto JP ??%
Ivana Hrivnocova ORSAY 10%
uenGter Kickinger 100%
Anna Kreshuk
Wim Lavrijsen LBL 50%
Sergei Linev GSI 60%
Jose Lo 75%
Pere Mato
Richard Maunder
Lorenzo Moneta
Axel Naumann FNAL
Eddy Offermann RENTEC 50%
Valeriy Onuchin
Timur Pocheptov JINR 80%
Fons Rademakers
Stefan Roiser
Andras Zsenei

+xrootd

BaBar

SEAL + ROOT Migration

- Adiabatic changes towards experiments
  - Experiments need to see libraries they use currently will evolve from current usage today towards a unique set
- Details be planned in the Programme of Work
  - Will be extra tasks in order to complete migration



ROOT Base and SealBase

- **Fons Rademakers**
- **Ilka Antcheva**
- **Bertrand Bellenot**
- **Philippe Canal**
- **Jose Lo**
- **Pere Mato**
- **Axel Naumann**

Merging SealBase with ROOT

- Make an inventory of what exists in SealBase and not in ROOT
- Port/add missing features to ROOT
- Provide a migration guide to help people migrate to ROOT base classes

SealBase

- SealBase provides about 80 classes that provide basic infrastructure functionality:
  - a set of classes encapsulating and abstracting OS and libc concepts and functions
    - file system, pipe, socket, inet addresses, system info, user info, process info, shared library handling, i/o multiplexing, error handling, time, memory mapped files, signal handling, logging, etc.
  - a set of utility classes:
    - timestamps, bit handling, regexps, string operations, tasks, callbacks, uuids, etc.

ROOT Base

- ROOT provides basically the same functionality in a much smaller number of classes.
- The main OS abstraction is via the TSystem class, which has two concrete implementations:
  - TUnixSystem
  - TWinNTSystem
- Most OS access goes via direct calls to TSystem (not via intermediate abstraction classes)
- Some higher level OS concepts are provided via their own classes, like sockets, multiplexers, timestamps, etc, but internally they all call TSystem
- In addition we also have a large set of utility classes for concepts like uuids, signal/slots, strings, regexps, tasks, white board, bit handling, etc.

ACLiC

The 'Automatic Compiler of Libraries for Cint' the tools implementing support for:

.L mymacro.C+

- Needs to be updated to properly support for MacOS's library idiosyncrasies
  - Probably requires introduction of support for autotools' .la meta shared libraries files
- Extended support more flexibility in the location of the generated library
- Enhance compilation speed for result of MakeClass/MakeSelector/MakeProxy

Coding practice and conventions

- Review existing coding practice and conventions including exception and error handling and better usage of associative patterns.
- See Fons' talk

AutoDocumentation

- Reimplementation of THtml to remove existing limitations
- new HTML presentation layer with more functionality
- save/retrieve documentation in a root file
- documentation available in ROOT session

User's Guide

- User's Guide is following ROOT production versions
  - Current - "User's Guide v4.0/08"

- New version in June/July
  - New chapter "Linear Algebra"
  - Chapters for updating:
    - "Getting Started"
    - "Graphics and Graphical User Interface"
- New chapters will be introduced: Mathlib..

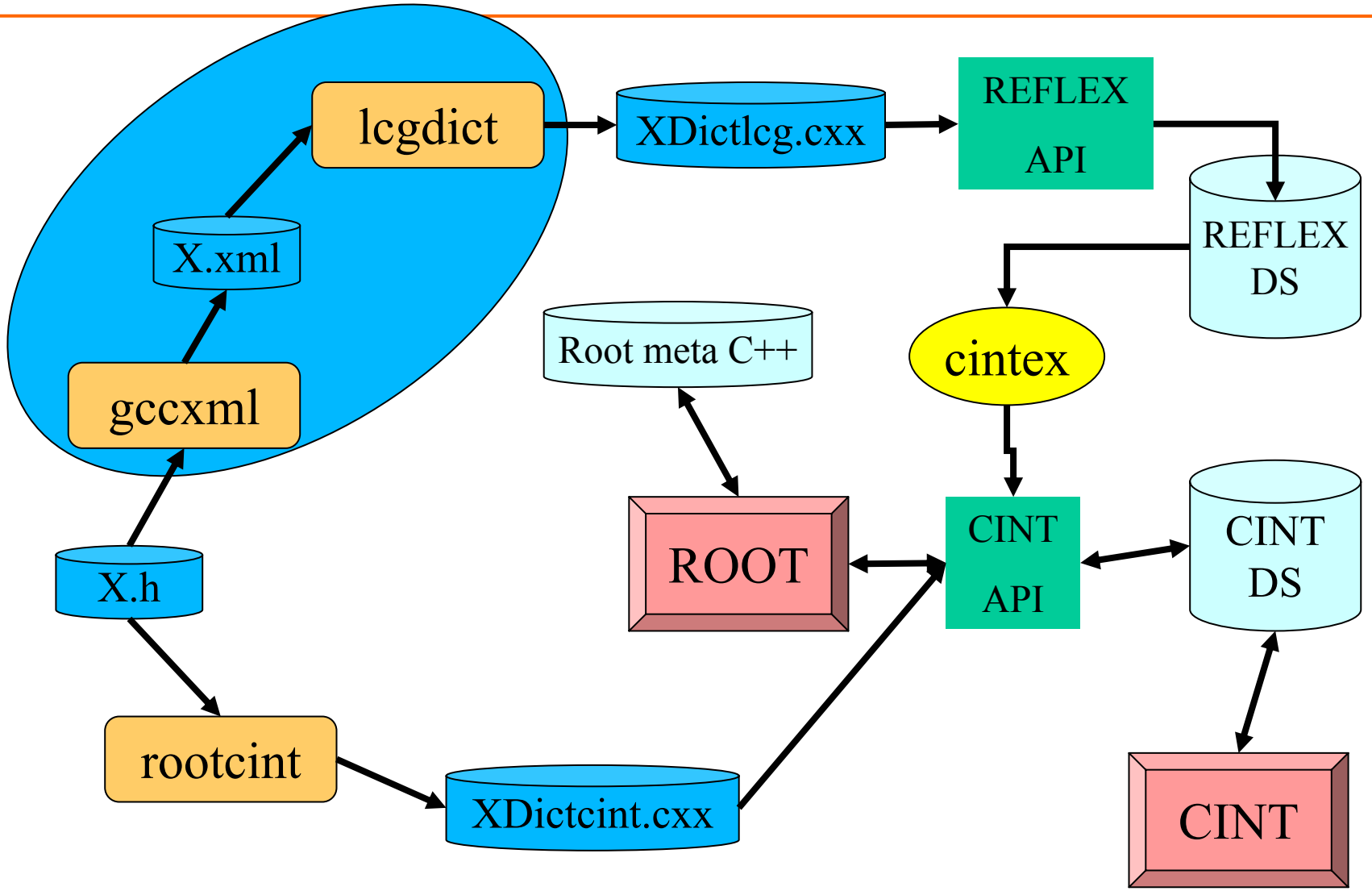
DICTINT
Dictionary, Reflex,CINT,Python

Philippe Canal
Masa Goto
Stefan Roiser
Wim Lavrijsen

CINT, Dictionaries & Interpreters

- Main Goal
  - Improve the conformity of the dictionary generator with the C++ standard without sacrificing performance.
- Move to CINT 6 (enhanced code executor)
- Understand the issues surrounding the incorporation of Reflex (and gcc_xml)
  - Portability of gcc_xml to all the CINT supported platform
  - Size and ease of installation of gcc_xml distribution
  - Backward compatibility issues of the CINT API
  - Performance issues (as far as interpreter is concerned)
  - Structure of the collaboration with Masa (to CVS or not to CVS).
  - Update of the ROOT Meta classes to leverage Reflex API

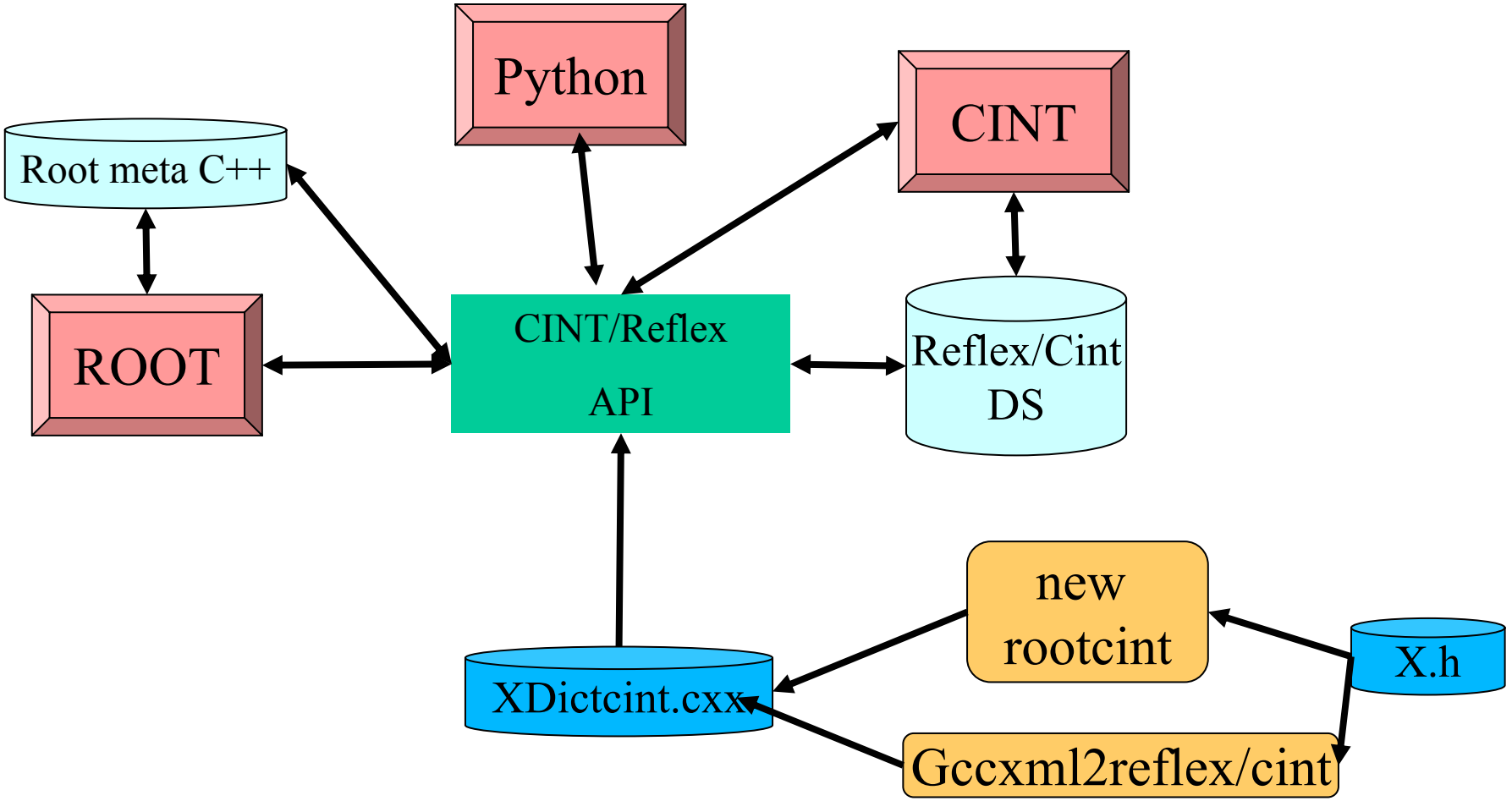
Dictionaries : situation today



CINT/Reflex workshop

- A very important workshop has been scheduled for May 2->7 to discuss the integration of Reflex and CINT.
  - Fons, Markus, Masa, Philippe, Rene, Stefan
- We hope to converge on a C++ DS taking advantage of Reflex and the current redesign of CINT by Masa.
- If successful, Cintex will not be required anymore.

Dictionaries : situation in the future



Meta package

- Improve the support for loading a library implementing a nested collection after the collection has already been emulated.
  - Introduce either a TClassRef or insure that TClass objects are never deleted
- Test (or decide to abandon the idea) changing ProcessLine to return a Long64_t
- Check if the autoloader now works for classes in namespaces
- Add a TClass:SetCanSplit or equivalent
- Add support for virtual derivation

CINT and Dictionaries

- Interpreted vs. Compiled function
  - Resolve issues of asymmetry in their usage
- #include <stdio.h> vs. CINT's embedded functions
- Improve support for typedef to function type
- Improve support for using statement when used for class and function templates.
- Add proper support for private and protected inheritance
- Resolve issues with non-qualified name
  - list<std::list<B> > vs list<list<B> >
- Investigate a few hard to reproduce core dumps.

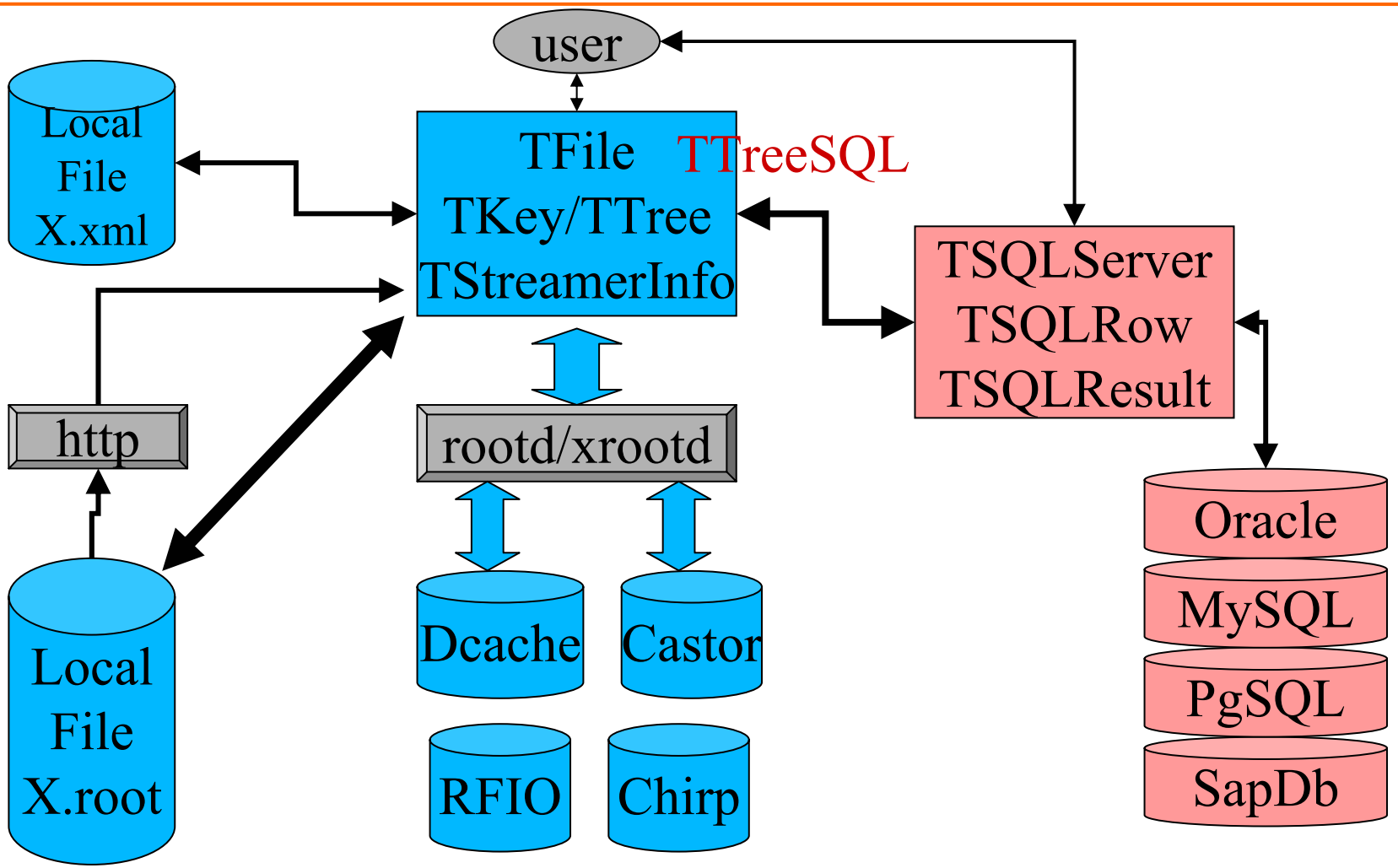
I/O
Basic I/O, RDBMS interfaces, Trees

Philippe Canal
Markus Frank

New RDBMS interface: Goals

- Access any RDBMS tables from TTree::Draw
- Create a Tree in split mode ➔ creating a RDBMS table and filling it.
- The table can be processed by SQL directly.
- The interface uses the normal I/O engine, including support for Automatic Schema Evolution.
- Convergence between RAL interface and the TSQL interfaces

File types & Access in 4.04/xx



TTree with SQL database back-end

- Upload in CVS repository of first version of TTreeSQL
  - support the TTree containing branches created using a leaf list (eg. hsimple.C).
- Add an interface to create the proper TTree object depending on the backend
  - Something like TTree::Open using the Plugin Manager
- Extend TTreeSQL to support TBranchElement
- Implement proper schema evolution support
  - The main design problem is how to save/retrieve the TProcessID/TStreamerInfo.
  - One possibility is to use the same mechanism currently in use in TXMLFile

General I/O improvements

- Upgrade support for C-style array of pointers
  - Currently support neither polymorphism nor duplicate references to the same object
- Strengthen support for typedefs when the library is not available.
- Review content of the ClassDef and ClassImp macros.

TTree

- Introduce support for bitmap indices
  - Should speed up specific query by more efficiently pre-selecting entries
  - Work in collaboration with Kurt Stockinger and John Wu (post Helmut work)
- Add support for class containing a sub-object of the same type as one its base class

TTreeFormula

Used by TTree::Draw and TTree::Scan to efficiently retrieve a TTree's data

- Update to support automatic dereferencing of reference objects including TRef and POOL refs.
- Improve support for casting the object type
- Allow intermixing of C++ function and TFormula functions in TFormula
- Fix TFormula::GetExpFormula when containing a method call
- Add support for calling method with non-numerical arguments
- Add support for TList, TObjArray as collections
  - Need a TVirtualCollectionProxy for each
- Fix a couple of oddities in the array dimension handling

- Add a TTree::Draw interface referring to the histogram by address rather than by name.

MakeProxy

MakeProxy generate a file implementing a class deriving from TSelector with which a C++ function can be run in a context where the name of the branches are useable as a C++ variable.

- Add support for stl containers
- Allow interpretation of the result
  - Need upgrade CINT to properly support cast operators
- Provide an implementation of MakeSelector and MakeClass using MakeProxy

PROOF
xrootd

- **Fons Rademakers**
- **Maarten Ballantjin**
- **Marek Biskup**
- **Derek Feichtinger (ARDA)**
- **Gerri Ganis**
- **Guenter Kickinger**
- **Andreas Peters (ARDA)**

BaBar

Original Goals

- Interactive parallel analysis on local cluster
- Transparency
  - same selectors, same chain Draw(), etc. on PROOF as in local session
- Scalability
  - quite good and well understood up to 1000 nodes (most extreme case)
  - extensive monitoring capabilities
  - MLM (Multi-Level-Master) improves scalability on wide area clusters
- Adaptability
  - partly achieved, system handles varying load on cluster nodes
  - MLM allows much better latencies on wide area clusters
  - not yet coming and going of worker nodes

New Additional Goals

- Support for “interactive” batch mode
  - allow submission of long running queries
  - allow client/master disconnect and reconnect
- Support “hostile” grid environments
  - startup of agents via Grid job scheduler
  - agents calling out to master (firewalls, NAT)
  - dynamic master-worker setup

Disconnect / Reconnect

- Authentication, sessions token
- State issues
- Transparency issues
  - TSelector::Begin(), Terminate()
  - objects created in Terminate(), output lists, etc.
  - storage of intermediate results

Grid Interfacing

- Grid catalog
  - data set creation
  - meta data, #events, time, run, etc.
- proofd agent creation
  - agents call out to (no-incoming connection)
- Config file generation / fully dynamic
- Coming and going of worker nodes
- Grid aware packetizer
- Scheduled execution
- Limiting processing to specific part of the data set

Performance Issues

- Read ahead interface in (x)rootd
- Using and understanding xrootd
- Cache index on master, optimizes repeat queries
  - assign different sessions with same data set to same worker nodes, reuse in memory files
- Monitoring
- Proofbench
- Query estimator

Authentication, Authorization

- New xrootd authentication plugins
- Certificates (login and user name)
  - single experiment wide login
  - user name used for sandbox
- Authorization to sandbox and shared global space
  - not to other user's sandboxes under same account

Robustness

- Get rid of OOB
- Split in two processes
  - protocol, authentication processor (proofd)
  - proofserv
- Communicate via priority based message queue
- Threaded or forked?
- xrootd code reuse?
- Periodic output list reporting to the master

Usability

- Selector with dynamic variables
- Selector proxy support
  - (TTree::MakeProxy – better name)
- Tree friends
- Event lists
- Compressed bit slice indices
- All possible draw options of the Draw() interface

Usability

- Browser interface to:
  - proof sessions
  - on demand histograms (single shot, timer driven feedback)
  - monitoring histograms
  - other input/output list objects
- Stop / resume capability

Usability

- Limiting processing to specific part of the data set
  - allow meta data in the TDSet (time, run, conditions, etc)
  - be independent from catalog
  - allows creation of new reduced data sets
- Remote display package, all canvases created in the master are send back to the client

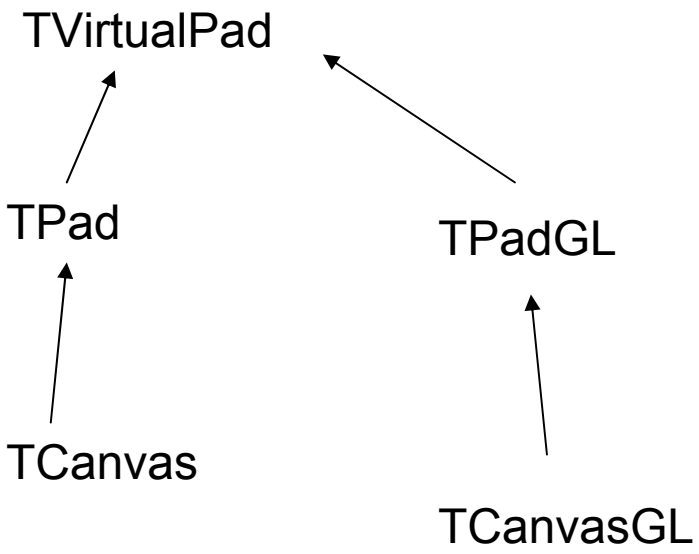
Conclusion

- We have started an acceleration in the PROOF development towards achieving the new exciting goals that will hugely enhance the data analysis experience of very large data sets
- A first firm milestone is a demo at SC'05

ROOT Graphics

- Olivier Couet
- Valeri Fine
- Richard Maunder
- Valeriy Onuchin

TPadGL / TCanvasGL

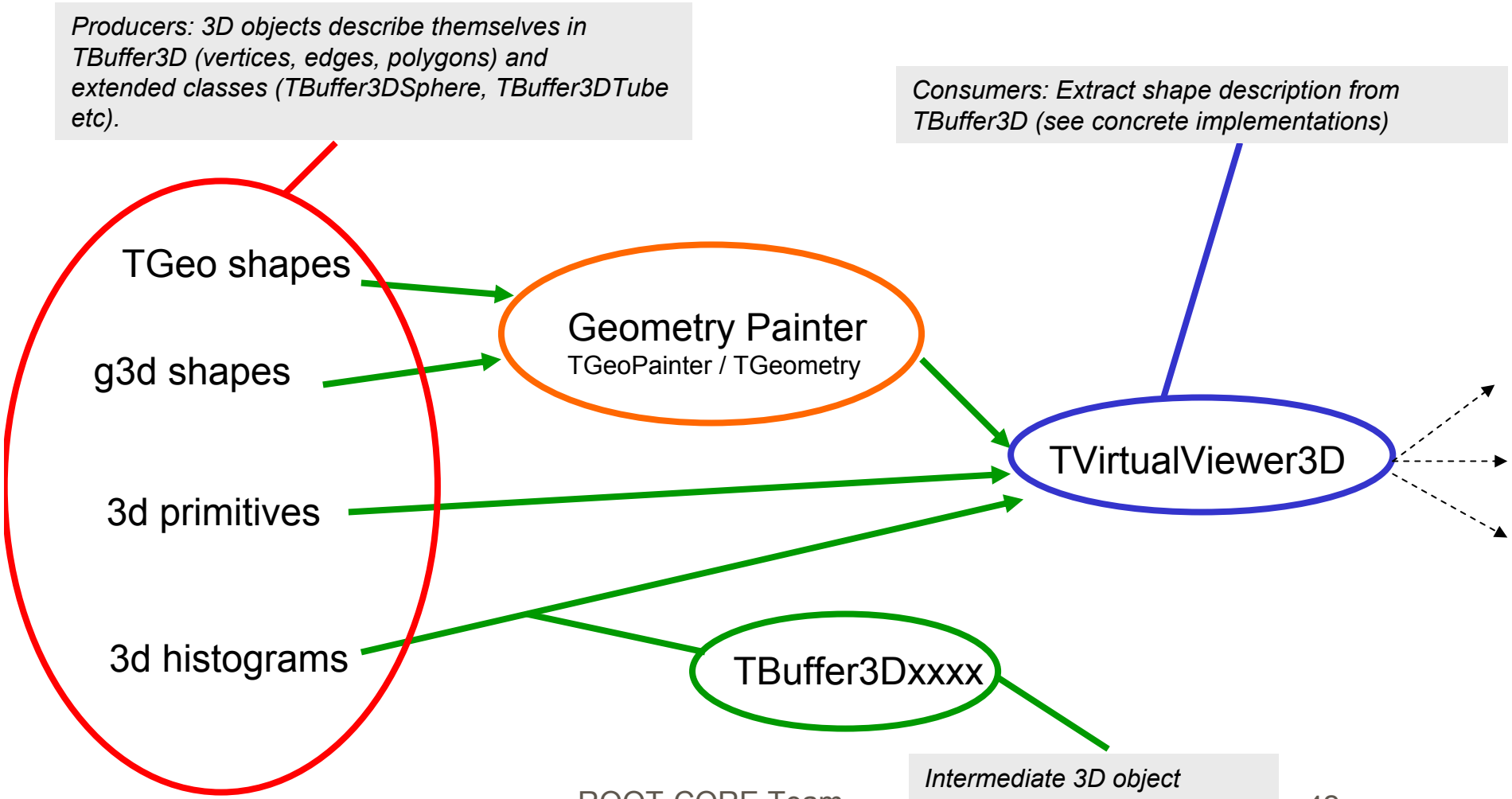


Inheritance diagram

Once TPad will be split, a new version Based on GL will be implemented. It will do:

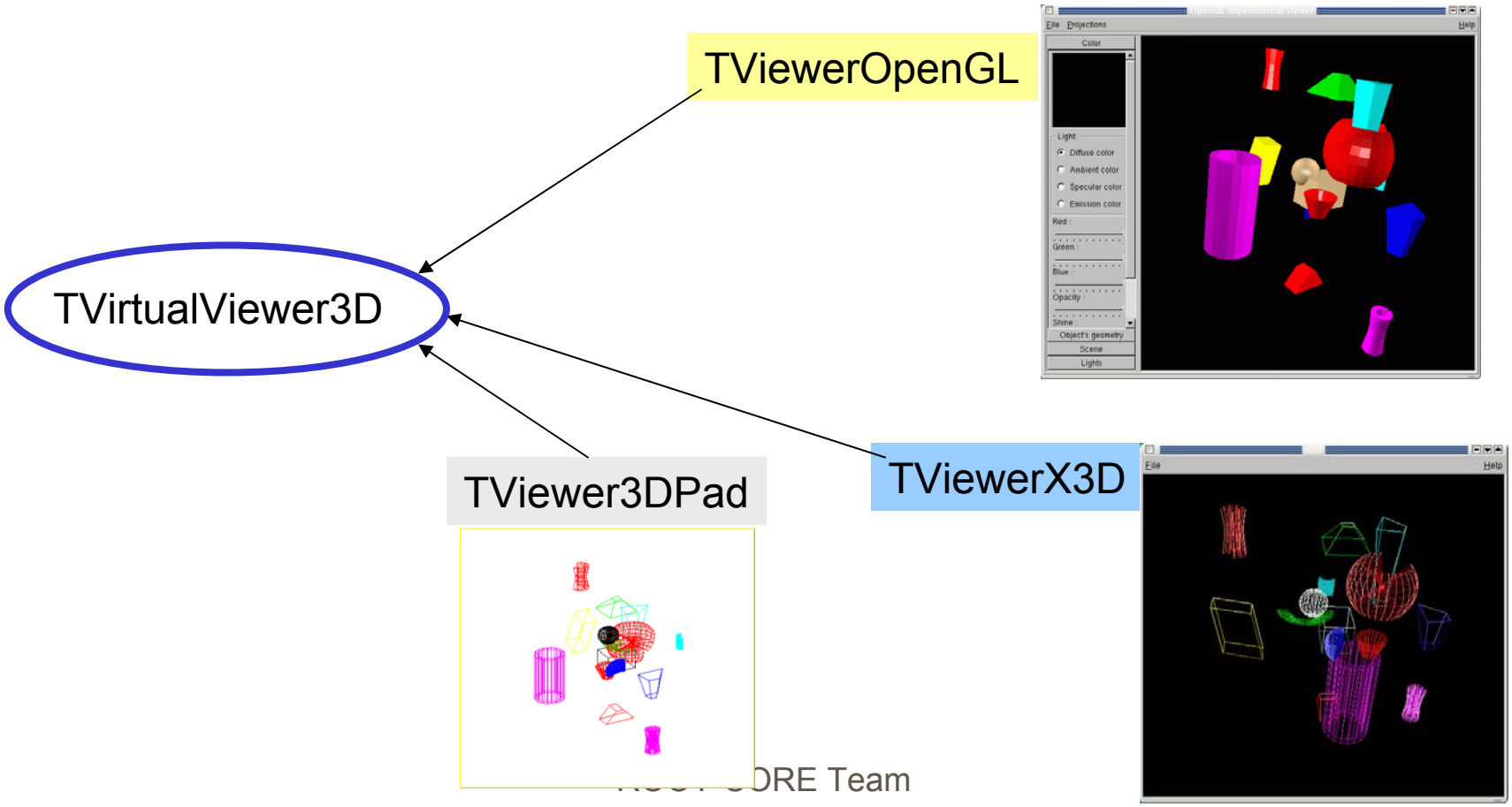
- 2D graphics *(the equivalent of gVirtualX is needed for OpenGL).*
- 3D scene rendering (TGeo) *(see 3D viewers slides)*
- 3D representation (Lego etc ..)

3D scenes rendering



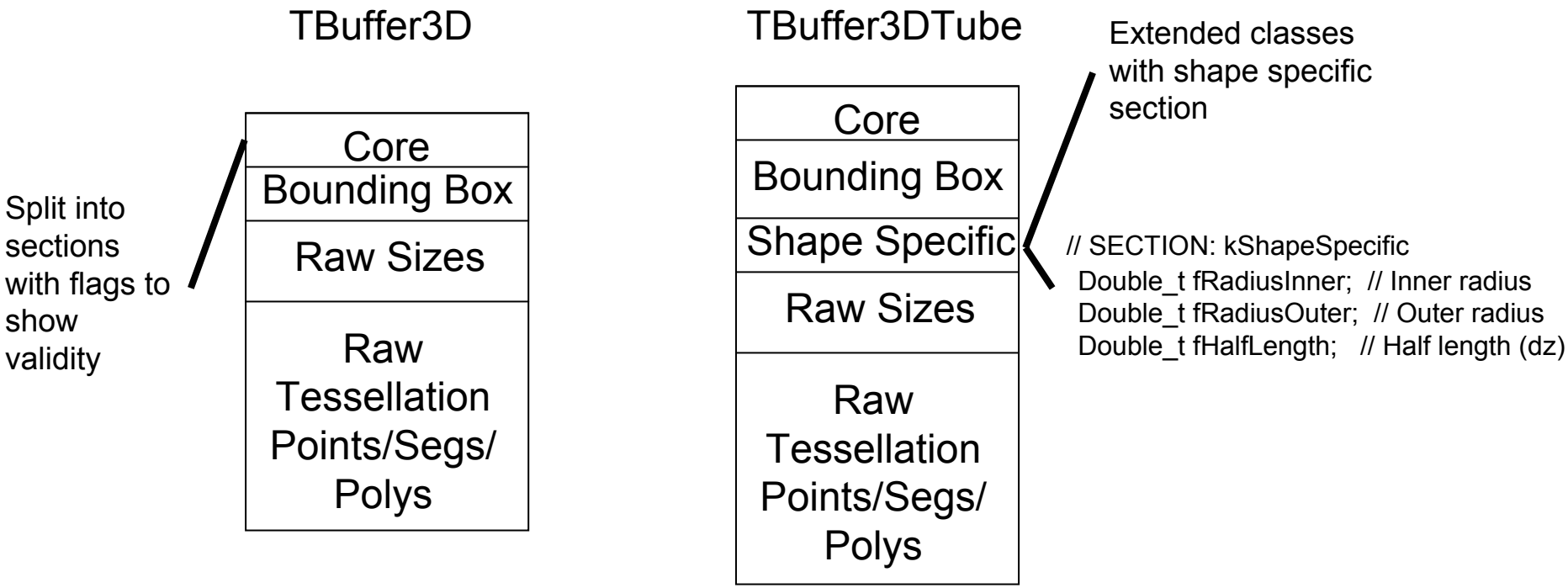
3D Viewers

3 concrete viewers inherit from the virtual interface TVirtualViewer3D.



Viewer Infrastructure Changes

3D Buffer Classes



- Producers fill cheap sections automatically and expensive parts (tessellation) on demand of viewer.
- Addition of bounding box, local/master reference frame and translation matrix.

Viewer Infrastructure Changes - cont

TVirtualViewer3D

- Viewer preference for local frame positions – producer shapes not obliged to meet request.*
- Viewer interest in child objects – should producer send?
- Simple objects: "3D primitive, at this 3D location"
- Placed & Template objects: "Placed copy (with unique ID) of this template 3D primitive" – enables viewer side caching of the unique shapes. Equivalent to:

TVV3D:	*Template*	*Placed*
Geant4 :	Logical Volume	Physical Volume
TGeo:	Volumes	Nodes

* All producing shapes must be able to generate buffer in master reference frame, and all consuming viewers be capable of displaying them.

Viewer Infrastructure Changes - cont

Together these allow:

- Filling of only the sections a viewer needs for a specific shape.
- Rejection of objects off screen before tessellation.
- Efficient, high quality native viewer tessellation of supported shapes, with fall back of producer side tessellation for unsupported one.
- Repeated geometry expansions, with termination on viewer request.
- Various viewer side caching schemes – e.g. retain all large/costly shapes, extract finer details as current view requires and performance permits.
- Ensures code outside viewer is free of viewer specific dependencies/branches.

- == Higher quality, faster rendering and interaction in OpenGL and high performance viewers, + backward compatibility with pad and legacy x3d viewer.

Viewer Infrastructure Changes

TBuffer3D

- Split into sections – core, bounding box, shape specific, raw tessellation (points/segs/polys) with flags to show validity.
- Extended classes with abstract shape descriptions e.g. TBuffer3DTube:

```
// SECTION: kShapeSpecific
Double_t fRadiusInner;  // Inner radius
Double_t fRadiusOuter; // Outer radius
Double_t fHalfLength;    // Half length (dz)
```

- Producers fill cheap parts automatically and expensive parts (tessellation) on demand of viewer.
- Addition of bounding box, local/master reference frame and translation matrix.

OpenGL Viewer

- In the future will be the main 3D viewer. Has to be updated to take advantage of the infrastructure changes.
- Use the full power of OpenGL (lights, transparency, interactivity, anti-aliasing, hardware acceleration …)
- PS/PDF output using *gl2ps*
- Native rendering of uncut solid spheres and various tubes already in place.
- Soon composite shape support

In the future:

- Convert to local frame, with template shape and OpenGL display list caching.
- Level of detail support – adjust tessellation for object size.
- Animation of objects.
- etc

2D graphics and others issues

Graphics output

Graphics outputs can be generate using:

- PostScript: Stable. No major developments foreseen
- PDF: The future. Very likely will replace PS in the medium term.
- SVG: not complete yet. More and more requests.

Vector

- TAsimage package:
  - many pixel formats,
  - Works in batch mode,
  - Markers are missing,
  - Could be use to generate output for ray tracing.

Pixel

Astronomers requests

- Reverse Y (and X) axis. This requires changes in many places.
- New projections: AITOFF, MERCATOR etc… It is available for some representations but still missing for COL plots for instance.

TLatex

- #ell Calligraphic “l” : ℓ
- #matcal{}: Calligraphic font
- #v{}
- #perthousand{} : ‰
- German umlaut

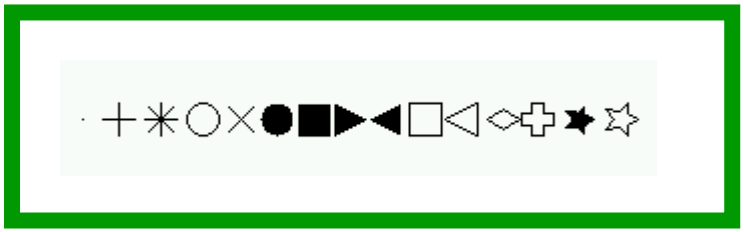
These symbols are not easy. Either the font is not (yet) in our TTF and PostScript drivers, or the character is available in one but not it the other.

Graphics test suite

- Automatic check: generate PostScript and compare the number of lines with a reference. Not very precise.
- Visual check: More accurate but need more time and manpower.

Markers

- We need more markers. The current list is not enough.
- User defined markers
- 3D markers



Markers currently available

Graphical User Interface

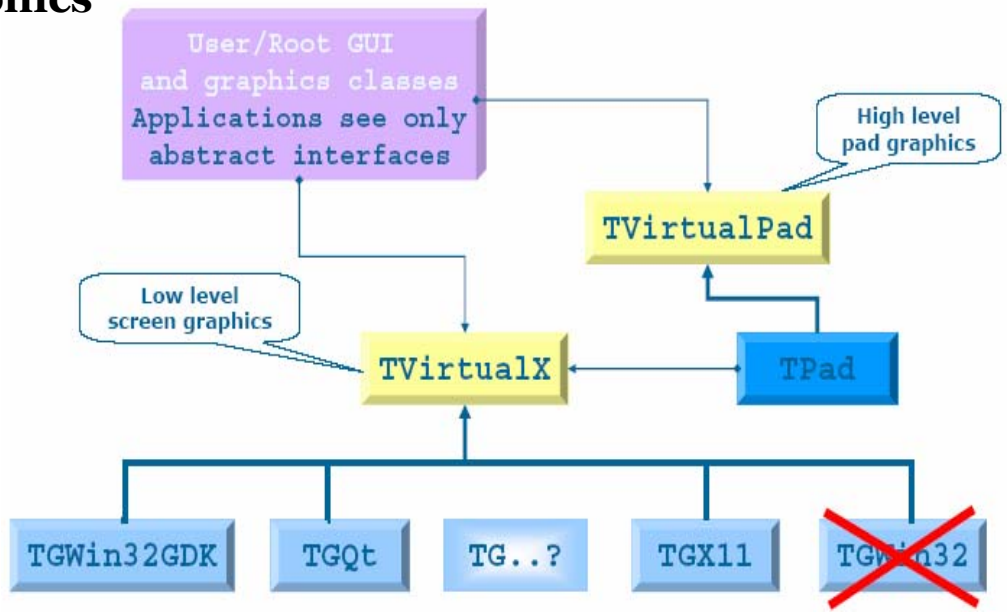
- Ilka Antcheva, Bertrand Bellenot, Valeri Fine, Valeriy Onuchin, Fons Rademakers

Overview

- Main Goal
- GUI Widgets
- Graphics Editor
- GUI Builder
- Undo/Redo Tools

Main Goal

- **Cross-platform GUIs – consistent look everywhere**
- **All machine dependent low graphics calls abstracted via TVirtualX**
  - **X11**
  - **Win32GDK**
  - **Qt layer – provided as a standard ROOT "plug-in" shared library that allows users to turn it on/off at run time with no changes in the user's code**

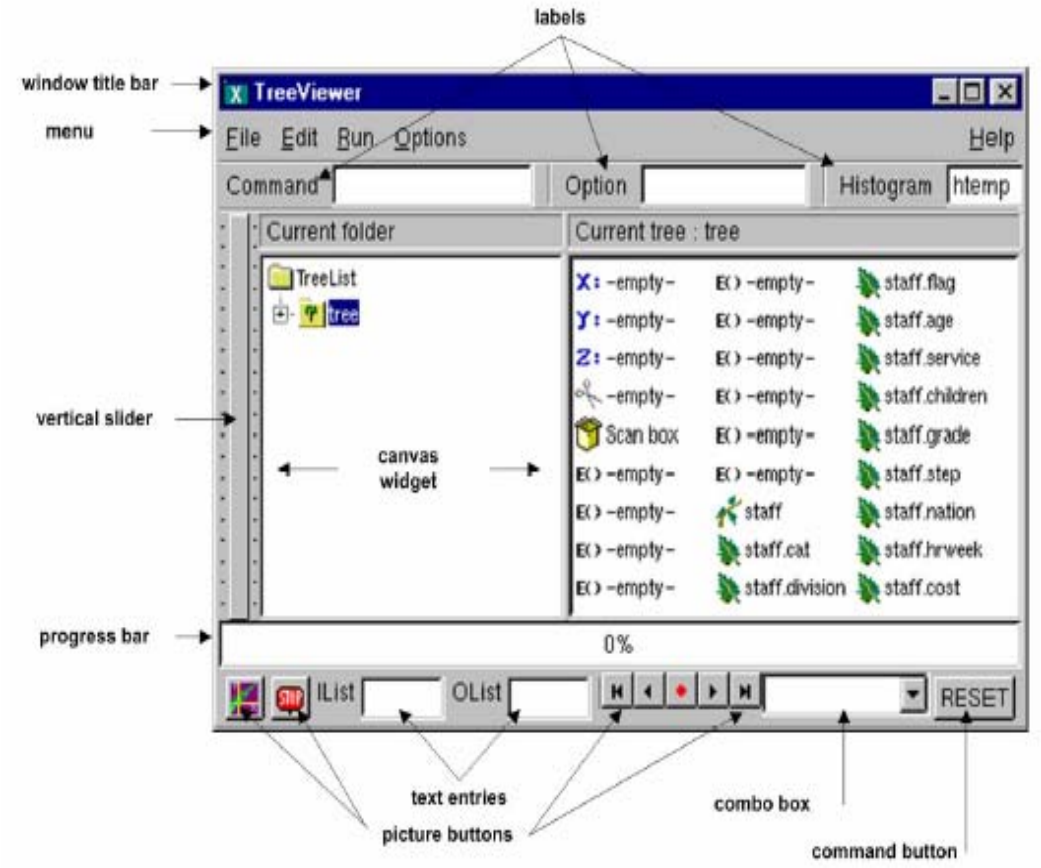


- **Improve the GUI design and performance; modify and iterate as much as necessary**
- **Integrate all system components: software, documentation, help functions, tutorials**

GUI Widgets (1)

Current Status

- Based on the XClass library from Hector Peraza
- Provide standard components for an application environment with windows 'look and feel'
- Object-oriented, event-driven programming model
  - Containers - use of ROOT container classes for fast object look up
  - Any widgets can be extended via inheritance
  - Layout managers
  - Signals/slots communication
  - Conventional model



GUI Widgets (2)

Next Steps

- Not finished GUI tasks
  - Keyboard navigation

    menu hot keys

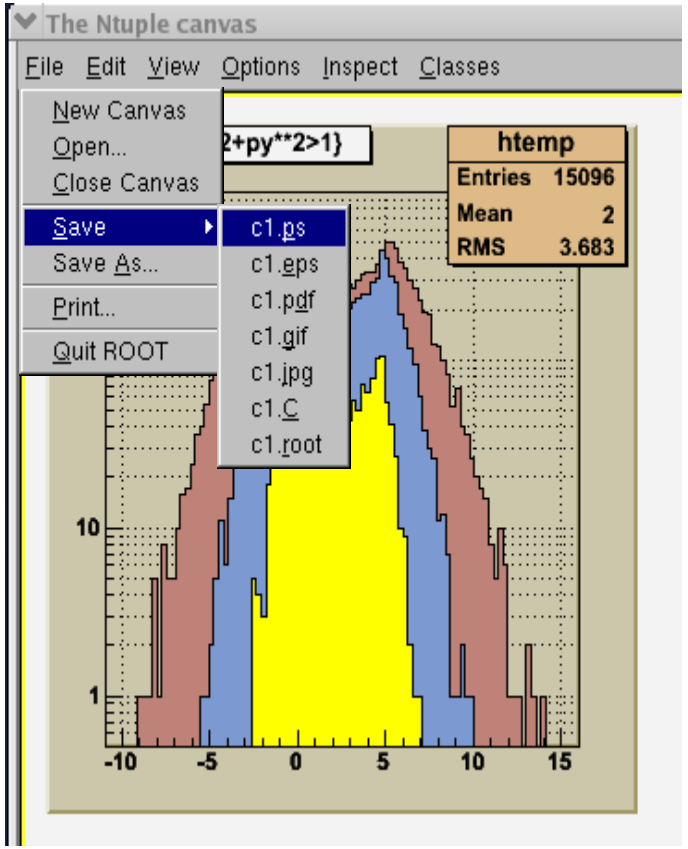
    Alt+F pops up File menu

    S activate Save - pops a submenu

    P does not create c1.ps

    dialogs - Enter = OK, Esc = Cancel

    combo boxes: up/down arrows, Home, End, PgUp, PgDn
- Cleanup tools

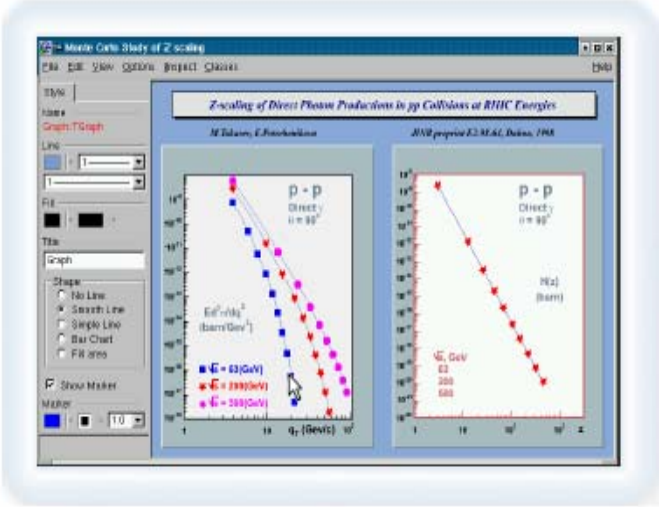


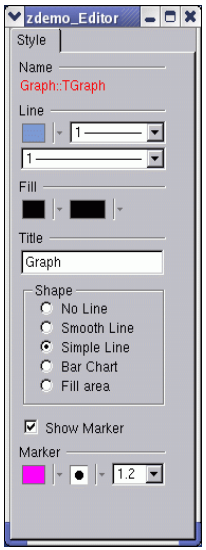
GUI Widgets (3)

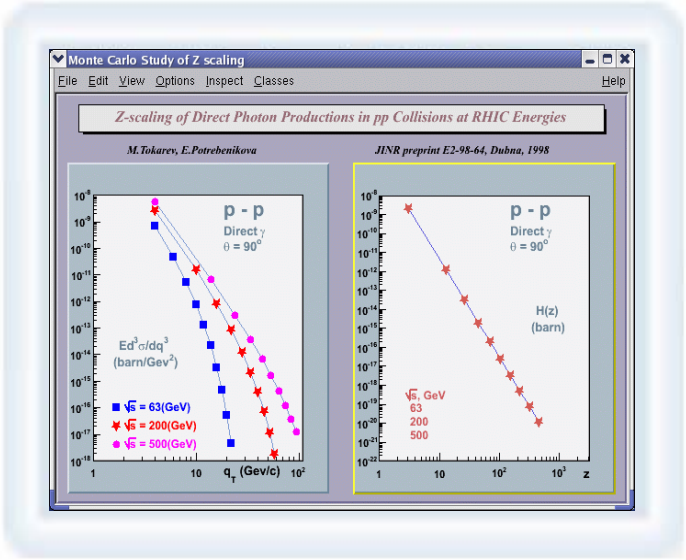
- Code optimization
  - Layout algorithms
  - GUI Dialogs
- Qt layer
  - Validation tests of interaction with ROOT GUI classes with the same ‘look and feel’
  - To solve problems with so-called "popup widget", like menus, drop down combo boxes, etc. and actions that require full-scale X11-like mouse pointer grabbing
- Complete the reference documentation of GUI classes

Graphics Editor (1)

- ROOT graphics editor can be:
  - Embedded – connected only with the canvas in the application window
  - Global – has own application window and can be connected to any created canvas



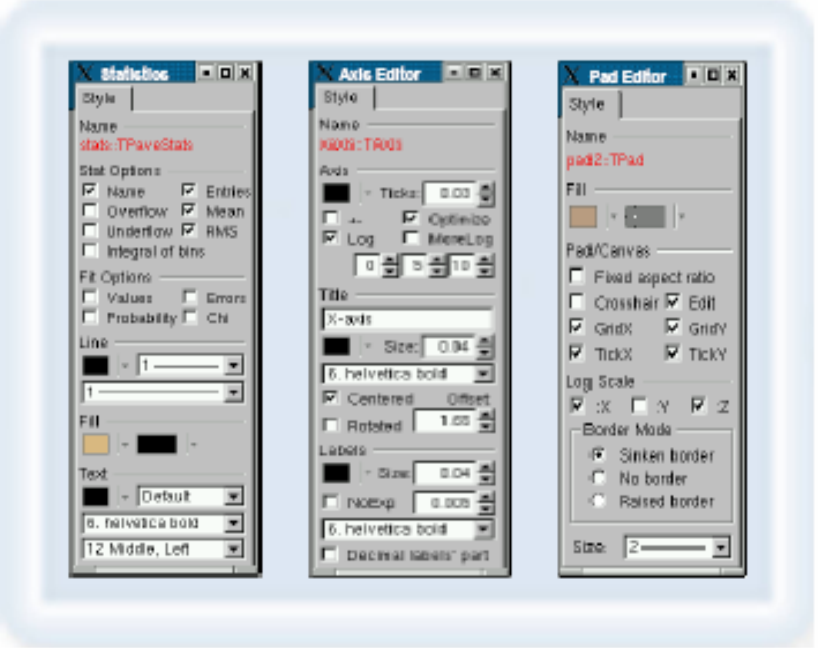




Graphics Editor (2)

- Modular – it loads the corresponding object editor TxxxEditor according to the selected object Txxx in the canvas respecting the class inheritance.
- Can be extended easily by any user-defined object editor - this makes GUI design easier and adaptive to the users' profiles
- Rules to follow:
  - Derive the code object editor from the base class **TGedFrame**
  - Correct naming
  - Register the object editor in the list TClass::fClassEditors
- This way the GUI complexity is reduced by hiding some GUI elements and revealing them when necessary.

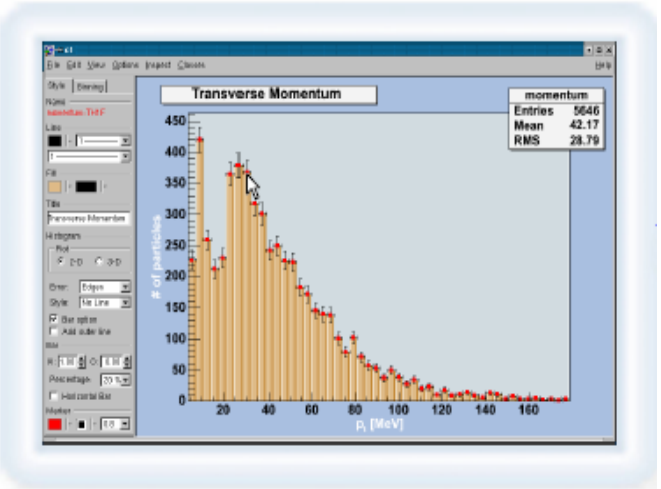
Different Object Editors



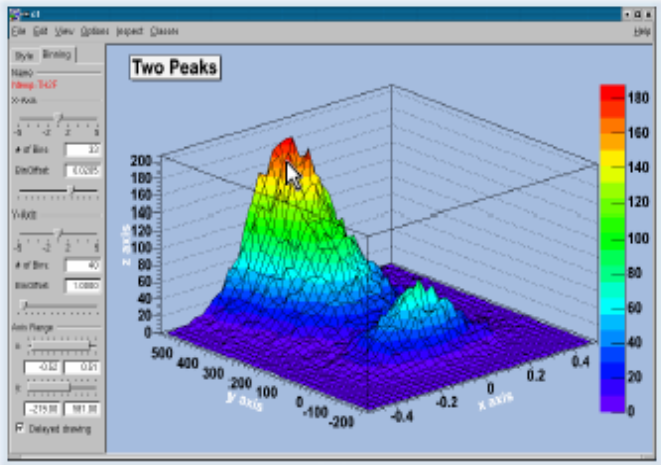
Graphics Editor (3)

- Global graphics editor
  - To show the related canvas title
  - Close, help buttons
  - Include ROOT commands in tool tips of the GUI widgets
- Hide/Show objects in a canvas
- New object editors:
  - **TAttPad**
  - **TSpline**
  - **TFn** draw panel
- Style manager – summer student project

TH1 Editor



TH2 Editor



GUI Builder (1)

- Tests and validation of the current version
  - Lay out a GUI quickly by dragging components
  - Using Ctrl+S or SaveAs dialog, users can generate C++ code in a macro that can be edited and executed via the CINT interpreter. It reproduces all widget in use and the GUI layout.

```
root[0] .x example.C
```

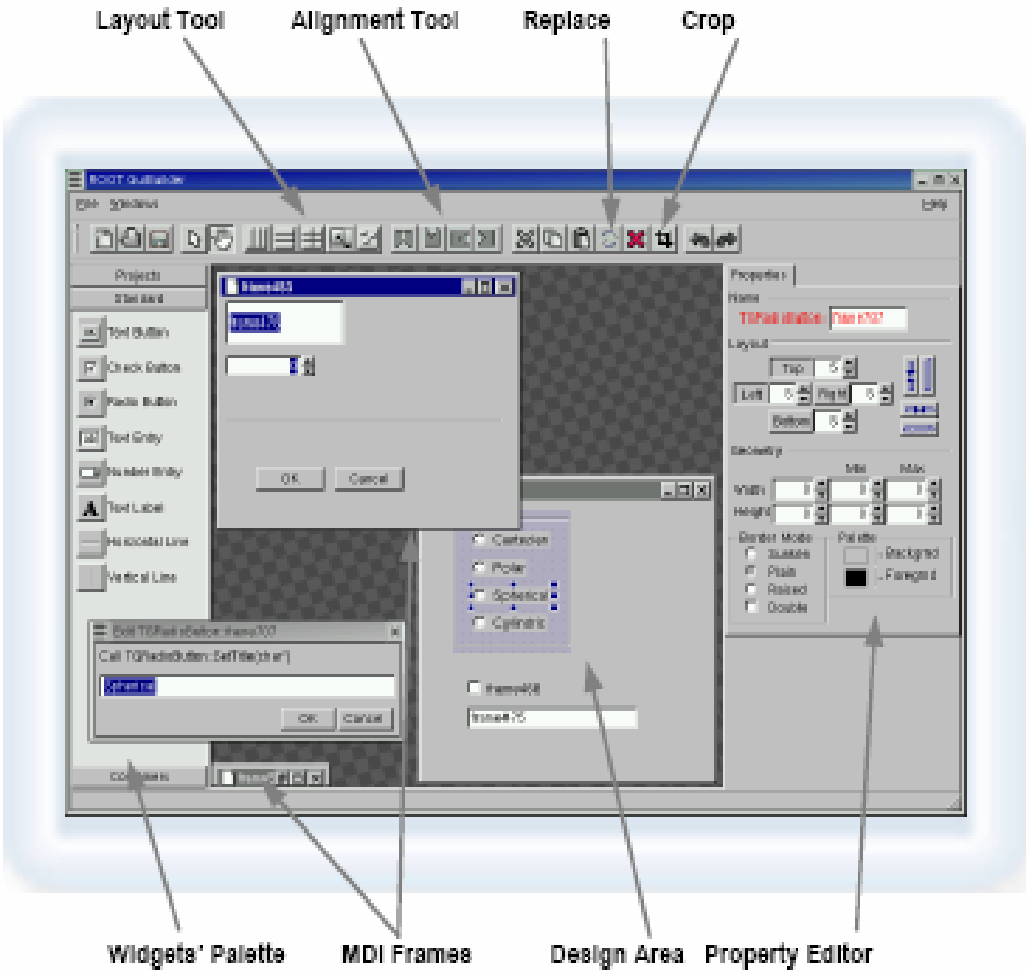
```
// transient frame
TGTransientFrame *frame2 = new TGTransientFrame(gClient->GetRoot(),760,590);

// group frame
TGGroupFrame *frame3 = new TGGroupFrame(frame2,"curve");
TGRadioButton *frame4 = new TGRadioButton(frame3,"gaus",10);
frame3->AddFrame(frame4);

frame2->SetWindowName("Fit Panel");
frame2->MapSubwindows();
frame2->Resize(frame2->GetDefaultSize());
frame2->MapWindow();
}
```

GUI Builder (2)

- Next steps – develop a set of tools for creating GUIs
  - Completing GUI components for selection: Button group, combo/list boxes, double sliders, list view, list tree, shutter, etc.
  - Tools for signals/slots
  - Templates for several basic types of GUIs (as tutorials)



Undo/Redo Tools

- Allow users to recover from mistakes - very important of GUI
  - Confirmation of destructive actions: Overwrite, Delete, etc
  - A stack of states/actions to go back

- Tests and validation of **TQCommand**, **TQCommandHistory, TQUndoManager**

GUI HowTo's and Tutorial

- GUI examples
  - In the User's Guide
  - How to examples on the web
  - GUI tutorials
- ROOT Graphics Editor
  - How to page
  - Guides for object editor user interface development

SEAL-ROOT Math Plans for 2005

- **Math work package**

- Andras Zsenei, Anna Kreshuk, Lorenzo Moneta, Eddy Offermann

See separate presentation by Lorenzo