LHC++

Status and evaluation

M. Stavrianakou

ATLAS SOFTWARE WORKSHOP
26/02/98

HEPEXPLORER future

Dino Ferrero Merlino, CERN IT/ASD

General issues

- **Objectivity**
  Plan to move HEPExplorer to Objectivity V 5.0 if V 5.0 is available on the major platforms (as it should be)
- **STL**
  Plan to replace RW with STL (assuming we can use aCC on HP)

HistOOgram Library

- Current implementation is frozen (bug fixing & small modifications allowed, helper classes can be prototyped)
- Current implementation will be ported to NT 4.0, DEC-UNIX 4.0, AIX 4.1
- Transient and persistent versions available on all platforms
- Analysis and 1st implementation of a new design: Templated histogram, STL, generic interface

HEPInventor library status

- 1D representations (stairs,line, curve & error plots)
- Page layout (zones), log scale, grid, vector postscript (1D graphics)
- Fill color,line color, zone background color,text size
- Support for HistOOgrams from Objectivity

HEPInventor improvements

- 1D representation:
    - Markers, line type, line width, min/max
    - Plot modes: superimpose, cumulate, subtract ...
- 2D representations:
    - Contour, box plot, scatter plot, arrow plot, color plot
    - Lego, surface

More HEPInventor improvements

- Information on plots:
  - Histogram statistics, Fit parameters
  - Date/time, Legend
  - Annotations (arrows, lines and text on the plot)
- Picking

HEPExplorer modules: optimizations

- Performance (CPU, I/O)
  - Profile current modules and (possibly) improve them
  - Reduce modularity to improve performance (if required)
- System resource usage
  - Assess modules memory footprint
  - Reduce memory usage by:
    - basic optimizations
    - Objectivity cache customization
  - Minimize the use of other system resources (e.g. reduce modularity to use less processes)

New HEPExplorer modules

- Functional replacement for NT/LOOP (over Tag Collections)
- New fitting module based on NAG minimizers
- Improved graphics modules:
  - page layout module, zone layout module (multiple zones on same page)
  - PostScript module: convert an OpenInventor geometry/file to PostScript (vector postscript on any 2D/3D plot)
- Identify new experiment-oriented modules (collaboration with CMS/Atlas/...)

IRIS Explorer

- Porting to:
  NT 4.0
  DEC-UNIX 4.0
  AIX 4.1
- Script language survey

Plans for future releases of HEPEXPLORER, HEPINVENTOR, IE
=> *feedback requested on priorities!*

A consistent release of LHC++ is foreseen for June/July 1998. Depending on the actual release of the underlying products, a following release could be made in early 1999 (using OpenInventor 2.5 and IRIS Explorer 4.0), or towards the middle of the year, using also Objectivity/DB version 6.

This is a list of possible enhancements to HEPInventor/HEPExplorer/IRIS Explorer. Enhancements for the HEPInventor category will be split in three sections:

- Features guaranteed to be in June's release
- Features that could be in June's release.
- Features that will likely be implemented on the next release.

For each items of the second and third category, please assess the importance of the feature according to when you would like to see this feature implemented, either in the forthcoming June 98 release or later on.

HEPInventor improvements (class library level)

- **Features that will be implemented on June:**
  - 1D : Markers, line type, line width, min/max (when e.g. plotting a histogram)
  - 2D : Box plot, scatter plot, color plot
  - Information on plots: histogram statistics, fit parameters, date/time, Legend
- **Features that could be implemented on June:**
  - 2D : Contour, box plot, scatter plot, arrow plot, color plot Lego, surface
- **Features that could be implemented on next release:**
  - Annotations (arrows, lines and text on the plot)
  - Simple picking (e.g. pick on a bin and see corresponding content/error)

HEPExplorer modules:

- Optimization of performance and system resources usage for existing modules.
- Functional replacement for NT/LOOP (over Tag Collections)
- New fitting module based on NAG minimizers
- Improved graphics modules: Page/zone layout module (multiple zones on same page)
- PostScript module: convert an OpenInventor geometry to vector PostScript

IRIS Explorer

- Porting to: NT 4.0 DEC-UNIX 4.0 AIX 4.2
- Script language survey

Other issues

- **for the general status of LHC++***(J. Shiers),* **CLHEP** *(E. Tcherniaev)* **and Gemini** *(S. Szkutnik)* see **http://wwwinfo.cern.ch/asd/lhc++/meetings.html**
- **Geant4** *(S. Giani)*

  see following presentation
- **PYTHIA7** *(L. Lonnblad)*

  see http://www.thep.lu.se/tf2/staff/leif/Pythia7.html

  http://www.thep.lu.se/tf2/staff/leif/CLHEP-PDT/

  (a suggestion for a common interface to particle properties to be included in CLHEP)

A user analysis scenario *(from CERNLIB to LHC++)*

TRT testbeam straw alignment

- open histogram file
- initialise (read alignment constants, calculate initial straw positions etc)
- book new histograms
- loop over straws
    - normalise (divide straw profile histogram with layer profile histogram)
    - fit normalised histogram with user defined function (cubed Gaussian with 3 parameters, where initial value of mean comes from normalised histo)
    - check fit quality and mask bad fit
    - calculate residuals (fit mean - initial straw position)
    - fill new histograms
- plot residuals vs corresponding straw layers and fit straight line or parabola
- loop over planes
    - get mean value of residuals in plane
    - if there are no good fits in a plane calculate the value from the parabolic or straight line fit
- write residuals to a file
- (optional) display and/or postscript output histos and fit results

How to implement and run (?)

- **with CERNLIB**
  - interactively with PAW: develop (and save) a kumac (a kind of script) using PAW 'vectors', sigma, comis etc
  - in batch
    - with PAW and the kumac (...)
    - in FORTRAN using HBOOK, HPLOT etc

- **with LHC++**
  - interactively with Iris Explorer:
    - make (and save) a map by connecting existing IE modules (lots of 'clicking')
    - write an IE script (???)
    - encapsulate C++ code (using Objy, HistOO etc) in a new module (possibly with minimum interface to IE)
  - in batch
    - with Iris Explorer (???)
    - in C++ (using Objy, HistOO etc) using the non-IE part of the module code

- **other...**

A PAW kumac

```
MACRO AllStraws MaxStraws=1280
.......
* Prepare the histograms and vectors
  exec clearvector hight means width ystr res phi
  vector/create ystr(1280)          r; ...........
...........
* Fit he individual straws
  do Istraw = 1,[MaxStraws]
    exec StrawFit [Istraw]
    vector/copy par(1) hight([Istraw]); vector/copy par(2) means([Istraw]); vector/copy par(3) width([Istraw])
  enddo
* Get correct sign on the width
   sigma width = abs(width)
* Take away all bad fits
  application sigma
    irow  = int(array(1280,0#1279)/16)+1; cutl  = 1.9 + 0.0*irow/80; mask  = 999999.*(abs(width-cutl) gt 0.2)
    hight = hight + mask; means = means + mask; width = width + mask
  exit
* Calculate straw positions and fill histograms
  exec StrawPos [RunNumber]
* Fit a parabola or a straight line to the y position of the track
  exec Parabola
* exec straight
.........
```

```
MACRO StrawFit
.......
*  Get the correct numbers of the histograms
   LayerNumber = int(([Strawnumber]-1)/16) + 1; StrawHisto  = [Strawnumber] + 8000
   LayerHisto  = [LayerNumber]+7000; NormProfile = [StrawHisto]+2000
*  Continue if histogram exist.
........
   divide [StrawHisto] [LayerHisto] [NormProfile] ! ! E
*  Declare vectors
   vector/create Error(3) r; vector/create ST(3) r; vector/create PMIN(3) r; vector/create PMAX(3) r
*  The first guess on the fit parameters
   vector/create par(3) r 1. $HINFO([StrawHisto],'MEAN') 2.
*  Fit the straw profile
   HISTOGRAM/FIT [NormProfile] [rd6path]/strawfit.f Q0 3 par ST PMIN PMAX Error
.......
```

```
MACRO WriteResiduals
.......
*  Only use residuals with a value below 10 mm
   sigma mask = abs(res) lt 10.; sigma res  = mask*res
*  Loop on roofs and planes and copy residuals for the plane into working vector 'block'
       ilow  = ([IRoof]-1)*256 + ([iPlane]-1)*16 + 1; ihigh = [ilow] + 15
       vector/copy res([ilow]:[ihigh]) block
* Get mean value of residuals in plane into output vector
       sigma block = sumv(block)/sumv(1-del(block))
* If no good fits in a plane use value from parabola fit
       if block(16)=0 then
         row    = irow([ilow]); fitvar = parab(1)+parab(2)*[row]+parab(3)*[row]*[row]
         vector/input block(16) [fitvar]
       endif
       vector/copy block(16) out([iPlane])
......
```

An IE module in C++

```
// various include files e.g.
#include "HEAdaptors/HEInputContainer.h"
........
extern "C" {
   void allfits              ( void );
   void initHook             ( void );
   void removeHook           ( void );
}
static HEInContainer *inCont;
static HESessionHelper *helper;
//---------------------------------------------------------------------------
// Hook function for Explorer module - executed once when the module starts
void initHook(void)
{
inCont    = new HEInContainer ("inCont");
helper = new HESessionHelper(); // Open FDDB
if (helper != 0) {
      if (helper->fDbCheck () == 0) { // Check if OO_FD_BOOT is defined
.........
//---------------------------------------------------------------------------
// Hook function for Explorer module - executed when the module stops
void removeHook(void)
{ delete inCont; delete helper; }
```

```
// This function is the kernel of the module.

void allfits (void)
{
.....
MNINIT(5, 6, 7); // initialise Minuit
GLOBAL = new GlobalDataFit(); // global data fit object
// get the fitting function
GLOBAL->model->.....
// create the parameter set
GLOBAL->SetOfParameters->....
// adjust the Minuit settings (if necessary)
GLOBAL->currentSettings->....
// initialise straw positions....
double ystr[MaxStraw]; strawpos(ystr);

helper->startRead();

  // get the container
  ooHandle(ooContObj) locContH;
  if (inCont->lookUp())
    if ((inCont->getValue()).isValid()) locContH = inCont->getValue();
.....
  // book new histos
  HepRef(Histo1D) histo_max  = new(locContH) Histo1D("Straw heights",50,0.,2.);
.....
```

```
// declare iterators over 1D histograms
ooItr(Histo1D) histI1; ooItr(Histo1D) histI2;
// find and divide histos
......
hist2->div(*hist1);
.....
// get fit data (values, errors, coords) from resulting histo
myData *mySet = new myData(hist2);
GLOBAL->globalData = mySet;
// set the parameters for the fit using the mean of the incoming histo
.....
current->setValue(hist2->compute_mean(filling_time_stats)); // MinParameter *current
.....
GLOBAL->SetOfParameters->setIntoMinuit();
.....
// fit and continue analysis
int ierrflag = GLOBAL->launchMinuitFit();
if (!ierrflag) { GLOBAL->SetOfParameters->getFromMinuit();
                 GLOBAL->extractParamValues(); ..............

          ..........
```

(haven't yet implemented the parabolic fit, the file output, the postscript output etc)

What we have learnt from this exercise

- this is just one example from a wide spectrum of analysis scenarios => we need more use cases
- quite a few things CAN be done with LHC++ now but debugging, enhancements and extensions are necessary in order to provide a comfortable working environment for the physics user, for example
    - Hbook2Objy (histogram names)
    - histogram operations (e.g. division)
    - ‘batch’ fitting methods
    - histogram ‘factories’ (i.e. optimise histogram I/O and hide Objectivity implementation)
    - improved graphics (zones, markers etc)
    - postscript output
- we need to understand and formulate our requirements and ideas for the analysis environment (GUI, scripting, batch etc) *(also in the context of the control domain)*