# Data Analysis, Presentation, and Visualization

Joseph Boudreau [a],

[a] *University of Pittsburgh, Pittsburgh, PA*

The status of simulation, reconstruction, analysis and visualization code for HEP is reviewed in light of presentations to the data analysis session of CHEP '97.

*Key words:* Analysis; Simulation; Reconstruction; Visualization

## 1 Overview

Software for simulation, reconstruction, analysis and visualization of particle physics data is at a turning point. A new generation of large experiments is expected to turn on in the year 1999: dedicated B facilities such as BELLE, BaBar, HERA-B, and CLEO III; new runs of CDF and D0 at the Tevatron; and the heavy ion experiments STAR and PHENIX. The papers presented at this conference show a HEP community that is beginning to break with HEP's FORTRAN past. The bulk of the material presented at CHEP '97 in the area of data analysis software was based upon the C++ language.

The field of particle physics is in a difficult period of retooling for future experiments. The people involved are not simply building reconstruction, simulation, or event display programs; rather, due to the reusability of the C++ language and the "toolkit" nature of the software, their efforts will determine the way that particle physicists program for many years into the future.

This summary will discuss four of the major areas of activity that have been reported at this conference, namely: analysis frameworks, simulation, reconstruction, and visualization.

## 2 Analysis Frameworks

An *analysis framework*, loosely speaking, is a program that schedules and runs pieces of simulation, reconstruction, or analysis code. These pieces of code have a common interface and act on event data. Analysis frameworks were presented from BaBar, HERA-B, STAR, PHENIX, BELLE, ATLAS, and CLEO.

The common abstractions of "module" and "path" underlie many analysis frameworks. A *module* is a piece of code that runs under the analysis framework; the user community extends the pool of available modules over the lifetime of an experiment. A *path* is an ordered sequence of modules. The framework executes each module in the path, in order. These two notions have spread around the world and are adopted by nearly every experiment now. The implementations are quite different, however. STAR, PHENIX, HERA-B, and BELLE all allow the use of either FORTRAN, C, or C++ within a module; in BELLE the multilanguage capability is achieved through the use of wrappers, whereas in the other cases a data definition language (DDL) is used instead. STAR, PHENIX, and BELLE use active dynamic linking in their frameworks, in two ways: first, to load different user interfaces at run time; and second, to load modules without relinking the entire analysis code. In such systems, users need only build object libraries. These multilanguage environments, while allowing users to program in their choice of programming language, severely restrict object-oriented design; specifically, OO design cannot straddle module boundaries.

An object-oriented incarnation of an analysis framework comes from the BaBar collaboration, where the notions of "module" and "path" have been captured in C++ classes. The event itself is also a class. This introduces a technical problem: since the event class is certain to be redefined, every module in the system seems doomed to require frequent recompilation. This problem is circumvented in the BaBar system by designing the event class as a dictionary of other objects. Basic classes for analysis have also been written for BaBar: these form the Beta package discussed here by Bob Jacobsen. The notions that underlie Beta are those of: 1) "candidate", which may be simple or composite, 2) "operator", a way of combining candidates (e.g., four-vector addition, vertex fitting) to make composite candidates, and 3) "finder", a factory class for identifiying and creating candidates. Beta is designed to be complete, versatile, and accessible to a wide range of users so it becomes the common basis of analysis code in BaBar.

Toby Burnett presented a rather different sort of framework, ARVE, for the ATLAS experiment. This framework allows for early debugging of reconstruction code. It consists of a drawing area and a graphical user interface to which users can easily add their own menus and commands. ARVE draws detector elements and tracks, and intersects tracks with detector elements. To draw detector elements, ARVE relies only on an abstract public interface that all such ATLAS elements are supposed to respect. To create intersections, it requires other services that may eventually have to be stripped away.

Tony Johnson has built and demostrated here a client-server analysis prototype using Java. Users write and debug Java code on a local, client machine, then transmit the code to a remote server where it has access to all the experimental data and can load additional object code. The output of the analysis, in the form of histograms, is transmitted back over the network to the client machine for further analysis.

Finally, we note that many of the frameworks discussed above must solve the problem of data access. Chris Jones from the CLEO experiment has discussed one solution. The main abstraction is "the state of the detector at an instant of time", or a *frame*. They have designed a system to serve frames to an analysis framework. The system assembles the frame from multiple *records*, each record being a parcel of related information (calibration, geometry, digitization, etc) coming from different sources known as *streams*. A feature of the system is that the streams themselves can be connected flexibly to different data sources, or extended.

## 3 Simulation

In the area of simulation, the most significant development of the last year is the progress that has been made in the GEANT4 project [1], which is completing its alpha-version at the time of this conference. GEANT4 is a complete object-oriented redesign of the GEANT program library for simulation of particle interactions with matter, (referred to here as GEANT3) [2]. Reports on three aspects of the GEANT4 project were given in this session: hadronic interactions, geometry, and visualization.

GEANT4 classes for hadronic interactions include both *interactions*, and *interaction models*. An interaction, elastic scattering for example, can have one or more models, each with its own validity range. Design goals include transparency and extensibility: it should be apparent to all users which objects control the behaviour of his program, and it should be straightforward to modify a program with additional processes or models. The accuracy of the models has been improved through the incorporation of new experimental data and better theoretical models, and the energy ranges have been extended. The GEANT4 collaboration is now doing extensive comparisons between their simulation, GEANT3, and experimental data. The collaboration intends to publish the results of the comparisons in a series of physics papers.

GEANT4 geometry supports two kinds of solids: constructed solids (boxes, spherical shells, etc.), and boundary representations (BREPs)[7][8], which describe any $2^{nd}$ order or NURBS surface. BREP allows the description of arbitrary shapes, and lets GEANT4 talk to CAD programs. A solid is held by each *logical volume*, which can have daughter volumes. A *physical volume* is a placed logical volume. Physical volumes support replication, where the shape, size, position or rotation of the placed volumes can be parameterized. To optimize the the calculation of track intersections with geometrical boundaries, GEANT4 slices the geometries, and if necessary then dices them again in one or two orthogonal directions. The overall performance gain over GEANT3 is reported to be significant, about a factor of two for the simplest geometries and more for complicated geometries. BREP, CAD interface, replication, and speed all constitute significant improvement over GEANT3 geometry.

The visualization component of GEANT4 consists of four graphics systems: the Fukui renderer, OPACS, Open GL, and Open Inventor. Each has particular strengths, and users can choose from among them at compile-time and at run-time. The Fukui Renderer, alias DAWN, generates vectorized postscript of an especially high technical and aesthetic quality. OPACS is a set of widgets that includes user interaction and GL-compliant rendering. Open GL is a very wide graphics standard for which commercial and free implementations are available; because many video cards implement Open GL in hardware, commercial Open GL can take advantage of graphics acceleration. And finally Open Inventor provides the most interactive user interface with real-time rotation and zoom, and like Open GL, takes advantage of graphics acceleration. Open Inventor is the only graphics system for which a license is a firm requirement.

## 4 Reconstruction

Three kinds of progress were reported in the area of event reconstruction. First, there has been progress in extending the Kalman Filter algorithm for track fitting to cases with non-Gaussian system and process noise. Second, there has been progress in exploiting the Kalman filter in very difficult tracking environments. Third, there has been progress in implementing very sophisticated tracking algorithms within large-scale object-oriented reconstruction projects.

The Kalman filter is a fitting technique used in track reconstruction to cope with significant multiple scattering [3]. This technique has come of age with B-physics. B physics has led to the almost universal use of silicon vertex detectors, which add material while improving resolution. To obtain the best resolution and accurate error matrices, one needs the Kalman filter.

The presentation of Rudolf Fruehwirth is an investigation of the properties of a simple, one-dimensional “toy” fitter which extends the classic Kalman filter to cases of non-Gaussian noise, such as multiple scattering and bremsstrahlung. The effects of bremsstrahlung can be important if a low-resolution momentum measurement is innovated with a high-resolution measurement *after* passage through a thin layer of material. This occurs, for example, for electrons in the usual silicon + drift chamber tracking system, and also for high energy electrons when the energy measurement in the calorimeter improves the momentum measurement in the drift chamber. A fast, approximate method (Gaussian superposition)[4][5] and a truly optimal method (Hastings-Metropolis)[6] were studied. The faster method proves numerically nearly as good. These studies await a full, five-dimensional generalization and tests with real detector data.

In a separate presentation, David Brown discussed treatment of non-Gaussian tracking errors, both from tails in the hit resolution function, and from the Moliere tails in multiple scattering. He uses Gaussian superposition, and finds, in a study with data from the ALEPH detector, an improvement of 3% in impact parameter resolution. By contrast note that for the same data, a 24% improvement results in using the Kalman filter rather than a simple helix fit. His presentation also described an object-oriented design of a Kalman filter track fitter: the central class is the "site" (an abstract base class) and its subclasses: bend, material, and hit.

Pattern recognition at HERA-B, and at STAR are in a mature state. While one is tempted at first to think of STAR as the most challenging environment for pattern recognition, the 3D hits in the STAR TPC makes the problem fairly tractable. In HERA-B typical events can have up to 200 tracks and the detectors up to 20% occupancy. Here, the favoured technique is candidate-following with a Kalman filter which bifurcates when ambiguities occur, and uses $\chi^2$ to select from among competing candidates. This technique finds the important $J/\psi K_s^0$ tracks amidst hundreds of tracks in the full event.

The BaBar tracking code is another significant use of object-oriented design in HEP. Their track objects are highly functional and can intersect themselves with surfaces and refit themselves. They are be reusable within BaBar in many contexts (reconstruction, alignment, analysis). Tracks have trajectories, but do not know the parameterization of the trajectory, even though they carry out a track fit. This decoupling is accomplished through the use of "differential numbers", which are entities having a value and derivatives with respect to some underlying parameter set. Differential numbers are used to automatically carry out difficult calculations that arise during a track fit, such as computing derivatives of a residual with respect to track parameters. Another feature is a high degree of polymorphism in the system. Different kinds of track fitters can be used in a fit, and the system can evolve by adding new kinds of fitters. Much of this code is already in a working state.

Today's tendency is to use very modern programming technology to make the most refined techniques in precision track reconstruction available to the latest generation of particle detectors.

## 5 Visualization

New developments in visualization are: immersive systems, data visualization programs (Explorer), 3D class libraries (Inventor), visualization class libraries for particle physics (Hepvis), and a working C++ event display program.

At Fermilab, an immersive visualization system consisting of a high-end graphics computer and a head-mounted device is under construction. Paul Lebrun showed examples of complicated events in which immersive systems help the observer to comprehend the event visually. The Fermilab group has used the Explorer program as the basis for the software to run this system. They have now begun to user Inventor directly, and have used it to build models of the CDF and D0 detectors.

The Hepvis project is an effort to extend the Inventor[9] class library with particle physics classes, such as scene objects that represent tracks. Smart nodes, encapsulating user interaction also play a part in the Hepvis library. For example, they allow programmers to implement a detector as a hierarchy of more and more detailed shapes that users can navigate by clicking. Basic shapes, interactive nodes, and specialized HEP viewers are all part of Hepvis, which aims to facilitate the development of fully functional event displays in C++.

A novel approach to the design of an event display program was discussed by CLEO collaborators. Their design is "user-centric", and aims to make the interface to the event display match their user's mental model. The resulting product, implemented in C++, has some very attractive features; for example, a Motif window displays, iconically, the hierarchy of objects in the rendering area. One can point and click in the motif window or in the render area to select an object. Part of the project included follow-up studies to determine if their interface indeed matched the mental model of their users.

For multi-event visualization, CERN is preparing a histogram class library. The project aims to provide native C++ histograms as objects for use in other C++ programs. One- and two-dimensional histograms derive from an abstract base class, which uses a "partition" class to determine the histogram binning. It is foreseen that the histograms are stored in an Object-Oriented database and are viewed with data visualization system such as Explorer.

## 6 Conclusion

The effort to retool the software for simulation, reconstruction, analysis and visualization of particle physics data is in full effervescence as the CHEP conference convenes in 1997. It will take another few years before one can finally say what the fruit of this tremendous activity will mean to the community of particle physics who, for better or worse, will work with these tools for many years to come. I hope for a bright future for all of these important endeavors.

## 7 Acknowledgements

I would like to thank the organizers of CHEP 97 for a fun and interesting conference and congratulate them for their spectacular recovery from the fire that damaged the original conference location. I would also like to thank the participants of session A for their cooperation in helping me prepare this summary.

## References

[1] Pointer to a set of useful documents may be found on "The GEANT4 Home Page", URL http://wwwcn.cern.ch/asd/geant/geant4.html. See also the presentations by S. Giani, J. Allison, J. Apostolakis, H. P. Wellish, G. Barrand, and S. Tanaka in these proceedings.

[2] "GEANT - Detector Description and Simulation Tool", CERN Program Library Long Writeup W5013

[3] "Application of Kalman Filtering to Track and Vertex Fitting", Nucl. Instrum. Methods Phys. Res. A262(1987) 444.

[4] G. Kitagawa, Am. Inst. Statist. Math. 46 (1994), 605

[5] R. Fruehwirth, Comp. Phys. Comm. 100 (1997), 1.

[6] S. Chib. and E. Greenberg, The American Statistician 49 (1995), 327

[7] M. E. Mortenson, *Geometric Modeling*, John Wiley and Sons, Inc., 1985.

[8] M. Mntyl, *An Introduction to Solid Modeling*, Computer Science Press, 1988.

[9] J. Wernecke and the Open Inventor Architecture Group, *The Inventor Mentor*, Addison-Wesley, 1994.