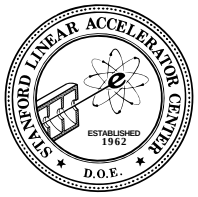

SLAC-PUB-5921
September 1992
(E/I)

Hippoplotamus*

**Michael F. Gravina, Paul F. Kunz,
Tomas J. Pavel, and Paul E. Rensing**

**Stanford Linear Accelerator Center
Stanford University
Stanford, CA 94309, U.S.A.**

Abstract

Hippoplotamus is a library package which manages and displays tables of data. It is written in ANSI C and has been tested on a large number of computer architectures. Hippo has a number of design features which make it unique. Data is saved in a binary, machine-independent format using the industry-standard XDR. Hippo supports graphics on many devices, but does not use any high-level graphics package. Even though it is written in ordinary C, the functions are all coded in an "object-oriented" manner. All the display functions have been optimized to allow for maximum interactivity.

1. Description

Hippoplotamus (or Hippo for short) is an n-tuple management and display package with object-oriented flavor. The package is written in ANSI C and comes in two parts. The first part is the n-tuple manager and it is designed to be user friendly, since it is likely to be incorporated into a user's analysis programs. In principle, the user only needs to know three functions in the package: one for creating an empty n-tuple, one for filling it, and one for writing it to a disk file. Optionally, the user might use two additional functions to give the n-tuple a title and label the columns. FORTRAN bindings for the n-tuple functions are provided with the package. Utilities to convert files in a simple text format to Hippo binary format and vice versa are also provided as part of the package, as well as a utility to convert n-tuples in HBOOK4[1] format to Hippo format.

The display part of the Hippo package is designed to be friendly to the programmer who implements an interactive n-tuple display program. Thus, it consists of a library of functions to create and manipulate displays of the n-tuple data in the form of 1D and 2D histograms, x-y plots, scatter plots, and strip charts. These functions are at a rather low level. For example, there is a function to change the number of bins in a histogram display while leaving all other characteristics alone. In a display application which has a graphical interface (such as HippoDraw, see ref [2]), a slider which controls the number of bins could easily be connected to this function. Other sliders could be hooked to the functions which control the lower and upper limits, *etc*. All calculations which are required to create a display, such as binning the data, are delayed until a request for the display to be plotted is made. If changes which might affect the bins have been made to the display since the last time it was plotted, the bins are automatically updated before being plotted.

The entire Hippo package, including the external utilities, is about 10K lines of code. It has proven to be highly portable using very few conditional compilation statements. It was developed on the NeXT and Sun platforms, but has been tested on IBM RS/6000, DECStation, Silicon Graphics, DEC VAX/VMS, and IBM VM/CMS. Since it is self-contained (apart from requiring XDR, which is discussed in a following section), it could be

* Work supported by Department of Energy, contract DE-AC03-76SF00515.

Contributed paper to the Conference on Computing in High Energy Physics, Annecy, France, September 21-25, 1992.

```
#include <stdio.h>
#include <math.h>
#include "hippo.h"
main()
{
   ntuple ntlist[2];
   float x, y;
   int rc;

   ntlist[0] = h_new(2);
   h_setNtTitle( ntlist[0], "My Tuple");
   h_setAllNtLabel( ntlist[0], "Angle",
"Sin" );
   for ( x = 0; x < 1000.; x++ ) {
      y = sin( x );
      rc = h_fill( ntlist[0], x, y );
   }
   ntlist[1] = NULL;
   h_write("test", NULL, ntlist);
}
```

Figure 1. Basic n-tuple creation steps.

used easily for many data collection purposes, for example, small embedded processors used in experiments.

2. Code Example

Figure 1 shows an example C program which generates an n-tuple and writes it to a file. The call to `h_new()` creates a new empty n-tuple structure, which will have 2 column, and returns a pointer to it. The user then uses this pointer in all subsequent calls which deal with this n-tuple. The two following calls in the example set the title of the n-tuple and label the columns. Then, the program puts data into the n-tuple using `h_fill()`, which can take a variable list of arguments, depending on the n-tuple's number of columns. As the program is finishing, it saves the n-tuple using `h_write()`, which writes a `NULL`-terminated list of n-tuples to the named file.

3. Important Features

3.1 Object-oriented approach

Hippoplotamus is written in ANSI C, but uses an object-oriented approach. The primary objects are `ntuples` and `displays`. Almost every function in Hippoplotamus has as its first argument either an `ntuple` or a `display`, on which the function performs some operation. Hippo itself maintains absolutely no state. This means that it is the responsibility of the user or application to keep track of the existing `ntuples` and `displays`, while the `ntuple` or `display` maintains everything there is to know about an n-tuple or plot.

There are certain advantages of this approach. One is the enormous saving in bookkeeping work. Once a new structure is created, Hippo does not remember anything about it. When a function is called to perform an operation, it works only on the current structure. Thus, writing the code to handle dozens of n-tuple is the same as to handle one. Another advantage is flexibility in how multiple n-tuples and plots are managed. It is up to the user or application to decide what is the best way to manage and group `ntuples` and `displays`.

3.2 File format

The n-tuple data are saved to files in a machine-independent format. Hippo uses the industry standard XDR[3] format which is available on all UNIX machines and on other systems as part of a TCP/IP networking package. XDR is also available with Sun's free RPC 4.0 source distribution. Thus, to transfer a Hippo file between machines of different architectures, the standard FTP program can be used in binary mode, or one could write a client-server interface. Also, files that reside on NFS-mounted file systems can be shared even by machines of different architectures. When XDR is not installed, a plain ASCII text representation of the n-tuple can be used.

The performance cost of using XDR is generally not significant. The files are no bigger than one would get using direct binary format, since Hippo data is mostly floating point numbers for which XDR adds no overhead. Reading an XDR file from disk can be about 5 times slower than reading a similar direct binary file. However, on machine architectures whose floating point format is known to be the same as XDR's canonical format (*i.e.* IEEE floating point), Hippo can read the majority of the file directly into memory, bypassing the XDR conversion routines. Even if this can not be done, the time penalty is only paid when the file is read in, which is generally done only once during a program's execution.

3.3 Graphics Drivers

Hippoplotamus provides routines for displaying plots on a variety of graphics devices. Supported are ordinary PostScript to a file, Display PostScript (for NeXT

computers), plain X-Windows, X-Windows under InterViews, UNIXPlot (Tektronix 4014 and others), and line printer mode.

The package does not make use of any high-level graphic packages such as GKS. Since there are only a small number of basic functions which need to be coded, it is straightforward to write the set for a new driver. In particular, there are basic routines to define the local co-ordinate system, to plot text at a given position, to draw a line connecting an arbitrary list of points, and to plot a list of symbols at specified positions. In addition there are a few routines which perform more complicated functions, such as plotting tick marks, which can be performed faster if done as a group. All calls to plotting functions are made through a set of routines which simply select the correct routine depending on the currently selected graphics driver. Thus Hippoplotamus achieves a high degree of portability because it does not depend on licensing or availability of external packages.

3.4 Cuts and Functions

Hippo supports the application of cuts to the n-tuple data used in displays. Each display has a null-terminated list of pointers to functions which determine whether a row should be used. Standard cut functions, such as less than some value or within two bounds, are provided. The user can also provide his own functions. Also, each display can have a null-terminated list of function pointers that are used to draw an arbitrary function on the plot.

This system of using pointers to functions which have a specific format allows great flexibility while not compromising the performance; no expression parser is needed to provide flexibility. One application that uses Hippo (HippoDraw[2]) invokes the C compiler and uses dynamic linking to allow the user to input an arbitrary function. While not portable, this technique can be applied on a wide variety of platforms.

4. Future Plans

We have recently developed a set of routines to deal with 3 dimensional plots, *i.e.* 3D histograms (legoplots), mesh plots, and scatterplots, with full perspective. All the calculations to transform the 3D co-ordinates to 2D are done in reasonably well-optimized C code. It only relies on the ability of the graphics device to draw filled polygons when doing plots with hidden lines. The user or application has full control over the view angles and distance, plus the amount of perspective. At the moment, we have only implemented the graphics code on a NeXT computer, but we expect the code to work on most other platforms. Our experience with the NeXT has shown us that the time needed to perform the transformation from 3D to 2D co-ordinates is frequently quite small compared to the time needed to draw the lines. Hopefully, the use of perspective in 3D plots will enhance the view of the data in subtle ways.

5. Summary

The Hippoplotamus library is an n-tuple management and display package which was designed to be light-weight and to allow a high degree of interactivity. It breaks new ground in High Energy Physics in that it is written in C and that it has an object-oriented flavor. Hippo is fast, easy to use, and self-contained (apart from XDR). The source distribution is available by anonymous FTP at HEPLIB.SLAC.Stanford.EDU.

Acknowledgments

The initial design of Hippoplotamus was done by Jonas Karlsson when he was employed as a summer student. Tony Johnson wrote the X-Windows graphics driver. Other contributions to the design of Hippo came from William Shipley and Gary Word. The routines to perform the matrix transformations in the 3D plotting code came from Douglas M. Bates and Murray K. Clayton of the University of Wisconsin.

References

[1] R. Brun and D. Lienart, *HBOOK User Guide: CERN Computer Center Program Library Long Writeup: Version 4*, CERN-Y250, Oct 1987. 108pp.

[2] M.F. Gravina, P. F. Kunz, and P. E. Rensing, *Proc. HippoDraw in Proc. Computing in High Energy Physics*, Annecy, Sept 1992 (CERN Report) and *SLAC-PUB-5922*, (SLAC) 1992.

[3] Sun Microsystems, Inc., *XDR External Data Representation Standard*, RFC1014, (see also man pages on UNIX systems).