Fermi National Accelerator Laboratory

FERMILAB-Conf-89/142

PAW at Fermilab

CORE Based Graphics Implementation of HIGZ *

Harald Johnstad
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510 U.S.A.

June 1989

* Presented at the 1989 Conference on Computing in High Energy Physics, Oxford, England, April 10-14, 1989.

PAW at Fermilab

CORE Based Graphics Implementation of HIGZ

Harald Johnstad
Fermi National Accelerator Laboratory[1]
Batavia, Illinois 60510.

Abstract

The Physics Analysis Workstation system (PAW) is primarily intended to be the last link in the analysis chain of experimental data. The graphical part of PAW is based on HIGZ (High Level Interface to Graphics and Zebra), which is based on the OSI and ANSI standard Graphics Kernel System (GKS). HIGZ is written in the context of PAW. At Fermilab, the CORE based graphics system DI-3000 by Precision Visuals Inc., is widely used in the anlaysis of experimental data. The graphical part of the PAW routines has been totally rewritten and implemented in the Fermilab environment.

1 Introduction

Personal workstations equipped with a 1 Mbit bitmap display, a speed of more than 1 MIPS, with at least 4 Mbytes of main memory and 150 Mbytes of local disk space are now available for less than 10.000 dollars, for example on a VAXstation 2000 or an Apollo DN3000.

In order to exploit the full functionality of such workstations, the Physics Analysys Workstation (PAW) project [1] was launched at CERN in the beginning of 1986.

The PAW environment is optimized to assist physicists in the analysis and presentation of data. It provides interactive graphical presentation and statistical and mathematical tools. PAW has been designed to exploit the good response time of workstations with their powerful graphics capabilities and attractive user interface in integrated environments. Good communications between large mainframes, used as centralized CPU's and file servers, and the workstations, are an important attribute of this system.

Workstations are generally part of a computer network. PAW allows for remote login and remote shell commands via the TCP/IP protocol (Transmission Control Protocol/Internet Protocol). This allows for file transfer between computers at execution time of 1 Mbyte or more per sec.

The first public release of the GKS implementation of PAW was made at the beginning of 1988 at CERN. The Fermilab implementation of PAW, with the graphics interface completely rewritten for the DI-3000 graphics system, was released at the end of 1988.

[1]Fermilab is operated by Universities Research Association, Inc., under contract with the U.S. Department of Energy

2 HIGZ - High Level Interface to Graphics and Zebra

An European effort was launched in the early 80's to restrict High Energy Physics users to using only graphics systems which mediate the transition from user programs to devices in a standardized way. This would allow for the portability of applications programs between systems using standard graphics.

However, the acceptance of the GKS standard outside Europe was seen as rather modest at the time when the PAW system was launched at CERN. Also GKS does not offer effective solutions to many requirements in the High Energy Physics community, and to the graphical output of PAW in particular. For example, recording large volumes of graphical information in compact form with a convenient access method, is not an integral part of GKS.

When the PAW project was launched at CERN, it was therefore decided to define a graphics interface, called HIGZ (High Level Interface to Graphics and Zebra), to be written in the context of PAW [2].

3 GKS - Graphics Kernel System

The Graphical Kernel System (GKS) is an international standard, adopted by the International Standards Organization (ISO), and by the American National Standards Institute (ANSI), for a 2D interactive graphics system. GKS provides for true portability of graphics code. The GKS Appendix E and the ANSI Computer Graphics Metafile (CGM) Standard format allow graphics information to be transported between different computer systems in a transparent way.

A GKS defined object undergoes a series of transformation that map the primitives from the world coordinate system to the device coordinate system of each workstation. The normalization transformation maps primitives from the user-defined world coordinate system to the normalized device coordinate system. The normalized device coordinate unit square is defined to be in the range (0,0) to (1,1) in X and Y.

The workstation transformation maps images from the normalized device coordinate system to the device coordinate system. The boundaries of the workstation window must fall within the normalized device coordinate unit square in the range (0,0) to (1,1) in X and Y.

Graphical pictures are defined using graphical output primitives, drawing primitives and text primitives. Each type of output primitive has an associated set of primitive attributes. Output primitives are defined in a 2D world coordinate system.

GKS accepts graphics input using logical input functions, and using a physical input device.

4 DI-3000 - Device Independent Graphics System

DI-3000 is based on the CORE graphics system [3]. CORE is a predecessor to GKS, but was never itself standardized, and many incompatible implementations were produced. The lower level of CORE is similar to GKS. The higher level of CORE is different from GKS.

DI-3000 is an integrated system of graphics software tools for 2D and 3D graphics, developed by Precision Visuals Inc., Boulder, Colorado. The primary feature of DI-3000 is its device independence, by which the application program references one or more virtual graphics devices; the application program user chooses a physical graphics device to be associated with the referenced virtual device at program load time. Several virtual devices may be referenced simultaneously.

The application program defines graphics objects using 2D or 3D graphics output primitives, such as moves, draws, characters, polygons, etc. Primitives may be defined in a 2D or 3D absolute or relative world coordinate system. These primitives are the fundamental building blocks that generate an image on a graphics device. Output primitives can be drawing primitives or text primitives.

Each type of output primitive has an associated set of primitive attributes, different for drawing and text primitives. Drawing attributes are used to define the general characteristics of the various drawing primitives. Each attribute is defined by a device independent integer index number. Text primitives define character strings that are output as graphics primitives on a graphics display device.

An object in its 3D world coordinate system undergoes a series of transformations that map the object onto the display surface of a physical graphics device. The viewing pipeline perform four distinct transformations:

The modeling transformation manipulates the object within the 3D world system, such as scaling, rotation, translation, and shearing.

The viewing transformation maps primitives from the world coordinate system to the virtual coordinate system, mapping 3D objects to a 2D or 3D image. The virtual coordinate system is usually 2D and has units in the range (-1,-1) to (1,1) in X and Y. For devices that support a 3D virtual device interface, DI-3000 can define a 3D virtual coordinate system, with its units in the range (-1,-1) to (1,1) for X, Y, Z.

For 2D applications, the viewing transformation maps a rectangular window in the world coordinate XY-plane into a rectangular viewport in the virtual coordinate system. For 3D applications, defining the viewing transformation involves defining the position, orientation, and line of sight of a virtual camera in the 3D world coordinate system.

The image transformation manipulates the image of an object on 2D or 3D virtual device surface by scaling, rotation, and translation.

The device transformation is the final step in the viewing pipeline and consists of mapping the object from virtual coordinates to physical device coordinates. The device transformation allows the application program to display an object concurrently on several devices that have different display dimensions, and yet have the object sized appropriately to each display.

DI-3000 supports graphics input using several virtual input fucntions, such as button, locator, valuator, keyboard, stroke and pick. A device driver sends the virtual input request to a physical input device. Several input devices may be used, such as cross hair cursor, light pen, tablet, and mouse.

DI-3000 supports device independent metafiles, which are transportable among devices and among DI-3000 installations. The DI-3000 metafile is a graphics device, which has the virtual device number 0, and which has its own special device driver. Pictures from DI-3000 metafiles may be postprocessed using a metafile translator. DI-3000 supports the Precision Visuals' proprietary metafile (PVIM) format. Conversion from CGM format to PVIM format, and from PVIM format to CGM format is also fully supported.

5 HIGZ with DI-3000

DI-3000 has been in use at Fermilab since before GKS became an ANSI and OSI world graphics standard. A large user community at Fermilab and associated universities had made strong commitments towards using DI-3000 for long term projects at the time the PAW project was launched at CERN. It was therefore decided to adapt the GKS based HIGZ graphics interface to the totally different DI-3000 graphics system.

Since the HIGZ calling sequences are identical to those of GKS, all the HIGZ routines with calls to GKS routines could be isolated on the HIGZ source PAM file in collaboration with CERN. The translation could then be made by completely rewriting a well defined set of routines (known as the IDI3000 routines), and at the same time insist that the PAW developers at CERN would not introduce GKS calls elsewhere in HIGZ, or any of the other integrated packages of the PAW system. This quickly became an exercise in collaborative efforts between two major High Energy Physics laboratories on software development, which turned out to be very successful.

The HIGZ/DI-3000 uses GKS parameters in many locations, with the DI-3000 parameters deeply hidden within subroutines. The following is a summary of these conversions.

The GKS virtual square ranges from (0,0) to (1,1) in X and Y, whereas the DI-3000 virtual square ranges from (-1,-1) to (1,1) in X, Y, Z. The HIGZ/DI-3000 virtual mapping is performed into the (0,0) to (1,1) quadrant in X and Y for 2D graphics, thus using only a subset of the DI-3000 virtual space.

As mentioned above, DI-3000 uses device number 0 for the DI-3000 metafile. This forced a redefinition of the metafile control routine, IGMETA, in HIGZ.

HIGZ/DI-3000 uses the concept of a 'primary' device, usually device number 1. The device initially selected by the HIGZ initialization routine, IGSSE, is deemed the 'primary' device in HIGZ/DI-3000, and graphics scaling and positioning are done with respect to the corresponding set of parameters for viewport and window transformations.

Secondary devices may be initialized from within the application program. The graphics output to secondary devices is scaled to present the entire viewspace of device number 1 on

each secondary device. This is a minor inconvenience where different device aspect ratios are involved.

HIGZ/DI-3000 supports both HIGZ and DI-3000 metafiles.

Color index numbers 0 through 7 are interpreted as GKS values. These numbers are translated internally into DI-3000 numbers, using a translation table for the defined colors: normal, red, green, yellow, blue, magenta, cyan, white, and compliment respectively.

GKS color intensities, red, green, and blue, are translated into the the corresponding DI-3000 values for: hue, saturation, and lightness respectively.

Certain color changes in GKS, such as associations between graphics elements types (line, marker, text), are usually not possible in DI-3000, and a global color change is done instead.

The polygon interior style used by HIGZ/GKS is somewhat different from that used in HIGZ/DI-3000. This is not a major concern, other than that some macro input files to PAW may have to be edited before they can be used by a DI-3000 application.

The linestyle and linewidth attributes are essentially the same in the GKS and the DI-3000 implementations of HIGZ, except that a 'dashdot' line in GKS becomes a 'longdash' line in DI-3000. Nonvalid values used by HIGZ/GKS, have been protected and reset to defaults in HIGZ/DI-3000.

Not all marker values scale in HIGZ/DI-3000, some internal HIGZ marker values override the DI-3000 values.

Text attributes are not always the same in HIGZ/GKS and HIGZ/DI-3000. For example, text vertical justification values in GKS ranges from 0 to 5 (normal, top, cap, half, base, bottom), and ranges from 1 to 3 (bottom, center, top) with HIGZ/DI-3000. These differences have been mapped by choice.

6 Implementation of HIGZ and PAW at Fermilab

The DI-3000 versions of HIGZ and PAW have been implemented successfully, with the exception of a few choices of mapping of attributes as outlined above. PAW using HIGZ/DI-3000 was first released at Fermilab at the end of 1988 when the IDI3000 routines in HIGZ became an integrated part of the HIGZ source PAM file, now being distributed from CERN. Fermilab has the responsibility for maintenance, bug fixing and further development of the DI-3000 dependencies in the PAW system.

PAW with DI-3000 is being used with a large number of drivers for workstations, as well as terminals. The many drivers in use include Codonics, Tektronix, Tektronix color, Envison, Macintosh, Talaris, Seiko, VT240, VT340, VAXstation, and PostScript (Apple Laserwriter). Several bugs in the various graphics device drivers software had to be fixed during the development of HIGZ/DI-3000.

PAW with HIGZ/DI-3000 is being used in production at Fermilab on VAX/VMS and Amdahl with VM/CMS, and at a number of collaborating institutions. It is expected that

the attractive and advanced features of this system, will be used by most experiments at Fermilab for several years to come.

Various HIGZ/GKS implementations have been used at Fermilab as well, mainly for testing purposes. The performance report between HIGZ/GKS and HIGZ/DI-3000 show no significant differences.

7 Future Plans

Future plans include adding full 3D capabilities in HIGZ and PAW with DI-3000, graphics application with DECwindows, X windows, and VAX/ULTRIX; remote login, remote file sharing (with Unix), using TCP/IP, and hopefully OSI (Open System Interface) in the future.

8 Conclusions

HIGZ has been implemented successfully at Fermilab in the DI-3000 CORE based graphics environments with no major performance differences with the original CERN HIGZ/GKS implementation.

The CERN/Fermilab collaboration on this project has been constructive and fully successful.

Acknowledgements

It is a pleasure to thank L.Roberts for invaluable help in sorting out bugs and fixes during the evolution of this project. Thanks are due to E.Lessner and A.Napier for constructive criticism and numerous testing of HIGZ/DI-3000 and PAW on several different systems and devices. Thanks are also due to the Graphics Support group at Fermilab for advice on the usage of DI-3000, in particular to C.O'Reilly for fixing bugs in the DI-3000 drivers, and for help with the color attribute routine. Thanks also to P.Constanta-Fanourakis for her advice on usage of the DI-3000 normalization routines.

Finally, it is a pleasure to thank R.Brun, F.Carminati, O.Couet, and P.Zanarini, at CERN for tremendous help and patience in the many discussions and questions on the implementation, testing, and distribution of this product.

I wish to express my sincere thanks to my wife for her silent patience during long, solitude hours during the development of this project.

References

[1] R. Brun, O. Couet, N. Cremel, C. Vandoni, P. Zanarini, PAW – Physics Analysis Workstation, *CERN Program Library Q121*, September, 1988.

[2] R.Brun, O.Couet, C.Vandoni, P.Zanarini, PAW, a General Purpose Portable Software Tool for Data Analysis and Presentation, This Conference.

[3] The CORE Graphics System, SIGGRAPH-ACM Computer Graphics 13, 3, (August 1979).