NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

RE-ENGINEERING PORTABILITY OF THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)

by

Recep Erdinc Yetkin
and
Sotero Enriquez
March, 1997

Thesis Advisors: Luqi
Valdis Berzins

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1997	Master's Thesis

4. TITLE AND SUBTITLE RE-ENGINEERING PORTABILITY OF THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)	5. FUNDING NUMBERS
6. AUTHOR(S) Recep Erdinc Yetkin and Sotero Enriquez	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	

13. ABSTRACT *(maximum 200 words)*

The Computer-Aided Prototyping System (CAPS) currently runs only on SPARC workstations running SunOS version 4.1.3. This limits the usefulness of CAPS, since Sun Microsystems has publicly announced that they have no interest in continuing support for SunOS version 4.x. A solution to this problem is to port CAPS to a PC platform running the Linux operating system.

Towards this end, the graphical editor portion of CAPS was ported onto a 100Mhz Pentium, with 32 MB of RAM, Linux 3.0, running Motif 2.0 on Xwindows. Modifications to both, the Makefile and the graphical editor source code were required for a successful compilation. These modifications were items such as having to compile using various compilers, providing pointers to the Motif and Xwindows Libraries needed to produce the static builds of the graphical editor, and a number of recompilations of the Linux kernel.

As a result of these efforts, the graphical editor, a functional component of CAPS, was successfully ported to this system. The software database, project control and execution support components still remain to be ported as a future development.

14. SUBJECT TERMS CAPS, Linux, Computer-Aided Prototyping, Software Port	15. NUMBER OF PAGES 343
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

RE-ENGINEERING PORTABILITY OF THE COMPUTER AIDED PROTOTYPING SYSTEM (CAPS)

Recep Erdinc Yetkin
Ltjg, Turkish Navy
B.S., Turkish Naval Academy, 1991

Sotero Enriquez
Lieutenant, United States Navy
B.S., University of New Mexico, 1990

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE
IN
COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1997

Authors: _____
Recep Erdinc Yetkin

_____
Sotero Enriquez

Approved by: _____
Luqi, Thesis Advisor

_____
Valdis Berzins, Thesis Advisor

_____
Ted Lewis, Chairman
Department of Computer Science

ABSTRACT

The Computer-Aided Prototyping System (CAPS) currently runs only on SPARC workstations running SunOS version 4.1.3. This limits the usefulness of CAPS, since Sun Microsystems has publicly announced that they have no interest in continuing support for SunOS version 4.x. A solution to this problem is to port CAPS to a PC platform running the Linux operating system.

Towards this end, the graphical editor portion of CAPS was ported onto a 100Mhz Pentium, with 32 MB of RAM, Linux 3.0, running Motif 2.0 on Xwindows. Modifications to both, the Makefile and the graphical editor source code were required for a successful compilation. These modifications were items such as having to compile using various compilers, providing pointers to the Motif and Xwindows Libraries needed to produce the static builds of the graphical editor, and a number of recompilations of the Linux kernel.

As a result of these efforts, the graphical editor, a functional component of CAPS, was successfully ported to this system. The software database, project control and execution support components still remain to be ported as a future development.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

I. INTRODUCTION

Today, the computer industry is one of the main developing research areas. This comes from the fact of the wide usability of computers in almost every area and makes them look like one of the "must have" devices in the future. With this increasing usability of computers in daily life, faster cheaper and more reliable hardware helps the software to become a dominant factor in terms of the customer needs. In the early days of the computing industry, only few people could answer the question, "What is software ?" In those days, project managers were focused on the hardware costs rather than software costs. The software can be seen as a logical rather than a physical system element. It consists of data structures, program procedures, and documentation. There are many growing areas for the software issue. System software, real-time system software, business, scientific, and engineering software can be given as examples [Ref. 1]. Since hardware prices are decreasing as demand for the software increases, software should be cost-effective to develop, meet user requirements and needs, and be reliable. Consequently, there is a great need to improve software productivity and reliability [Ref. 2].

A. BACKGROUND

The Computer Aided Prototyping System (CAPS) is an ongoing software engineering project at the Naval Postgraduate School. The aim of CAPS is to help design real-time systems. Real-time systems are defined as:

> ...those systems in which the correctness of the system depends not only on the logical result of computation, but also on the time at which the results are produced. [Ref. 3]

Command control, flight control, and some defense systems are common examples of real-time systems. The software of these type of systems, measure, analyze, and control certain events. From this definition we can conclude that such a system should be fairly complex and full of fault tolerance. If errors are not properly debugged, severe results can occur. CAPS Release 1 currently runs on sun Sparc 4.1.3. Since the announcement made by the Sun company (http://www.sun.com), stating that there is no further development support for the Sun Operating System, the portability of CAPS to different platforms has become an issue. This thesis focuses on enhancing the portability of CAPS to a more cost-effective platform. It researches the underlying system software, related tools and examines the feasibility of porting CAPS to the Linux Operating System.

1. Software Engineering Problems

There are many problems which can effect the system from a software engineering point-of-view. One of these problems today is the management of large-scale software packages with software teams. The coordination and consistency between the teams and packages become essential tasks for the software engineers.

A common task presented to software engineers is an achievement of both goals and needs specified during the requirement analysis. As pointed out by Luqi:

> ...as systems get larger and serve more diversified user communities, formulating requirements that accurately represent the customer's needs becomes the limiting factor in producing useful software. [Ref. 1]

The errors caused by large-scale packages is a big problem. For example, due to software program failures, the bank of New York was required to pay $5 million [Ref. 4]. A mix-up with the target timing data in a patriot missile had caused it to miss an airborne Scud missile. This resulted in killing 28 Americans during the Gulf War. [Ref. 5]

In a typical software development process, while customer needs and constraints for the systems implementations are documented in the requirement analysis phase, the proposed system interface is formalized and developed during the functional specification phase. The relation between the activities are shown as in Figure 1. [Ref. 6]

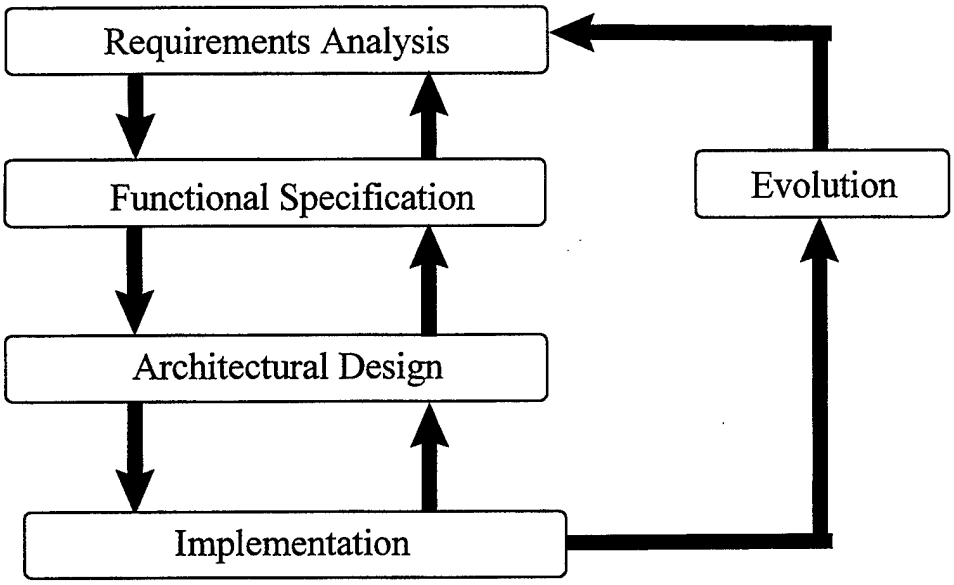

Figure 1. Software Development Process

Much of the time, the problem statement given by the customer is vague, and incomplete. The task of the software engineer is to analyze the customer's needs and test the feasibility of his goals. During this process, the prototype is designed and built to accomplish the listed goals and requirements. If the needs and/or requirements are changed

by the customer once, the prototype is implemented, the cost of reengineering and the re-development can be astronomical and may cause time overruns, or worse, cancellation of good projects due to customer's inability to state the requirements.

Faults in large scale software projects are usually derived from errors during the requirement analysis phase. These errors or omissions are not normally detected prior to the release [Ref. 7]. The cost of debugging errors rises dramatically later in the software process when the error is detected. These errors are resource-intensive and can quickly deplete the project's budget. A system is therefore needed that can document the customer's needs and document the project's status the life-cycle of software development. This reduces errors, and maintenance cost.

In FY 1995, The Department of Defense's (DoD) investment only in software was $42 billion. In addition to this cost, most of these large scale software projects suffered from overbudget, and overtime delays. According to the data verified by DoD, specifically 53% of these types of projects were stricken by cost or time overruns, 31% were canceled and only 16% were developed within the perspective of budget and goals. [Ref. 8]

2. Prototyping

Software engineering requires that a system being developed be fully specified. The reason for this is that changes, either in the customer needs or requirements, during software-development process are a source of problems and increase maintenance. One solution to this problem is to use prototyping. As pointed out by Luqi:

> A significant improvement in software technology is needed to improve programming productivity and software reliability. Computer-aided, rapid

> prototyping via specification and reusable components is a promising approach that makes this improvement possible. In this approach, the traditional; software life cycle is replaced by a life cycle with two phases: rapid prototyping and automatic program generation. [Ref. 9]

Prototyping is used for understanding and criticizing the proposed systems and explore the new possibilities that computer solutions can bring to their problems in a punctual and very cost-effective manner. Prototyping on one hand gives the opportunity to see and determine the needs for the proposed system for the designers, on the other hand gives the feedback to the customer of what the system will look like. As seen in the previous Figure 1, prototyping occurs during the evolution phase. The system is in a modification-demonstration-evolution loop until it satisfies the customer. An iterative approach can be seen here in order to design with improved feedback from the customer.

From the view point of designer and customer needs, prototyping builds a common platform which each party can communicate interactively on the proposed system. Along with discussions, clarifications, verifications or if necessary changes can be made to the system. Only after the agreement is reached by both parties, does the approved prototyping and the development go into the production phase. The prototyping process which is shown in Figure 2, depicts the prototyping phases. [Ref. 10] The shaded areas depict the cycles that are done to get the requirements prior to implementation.

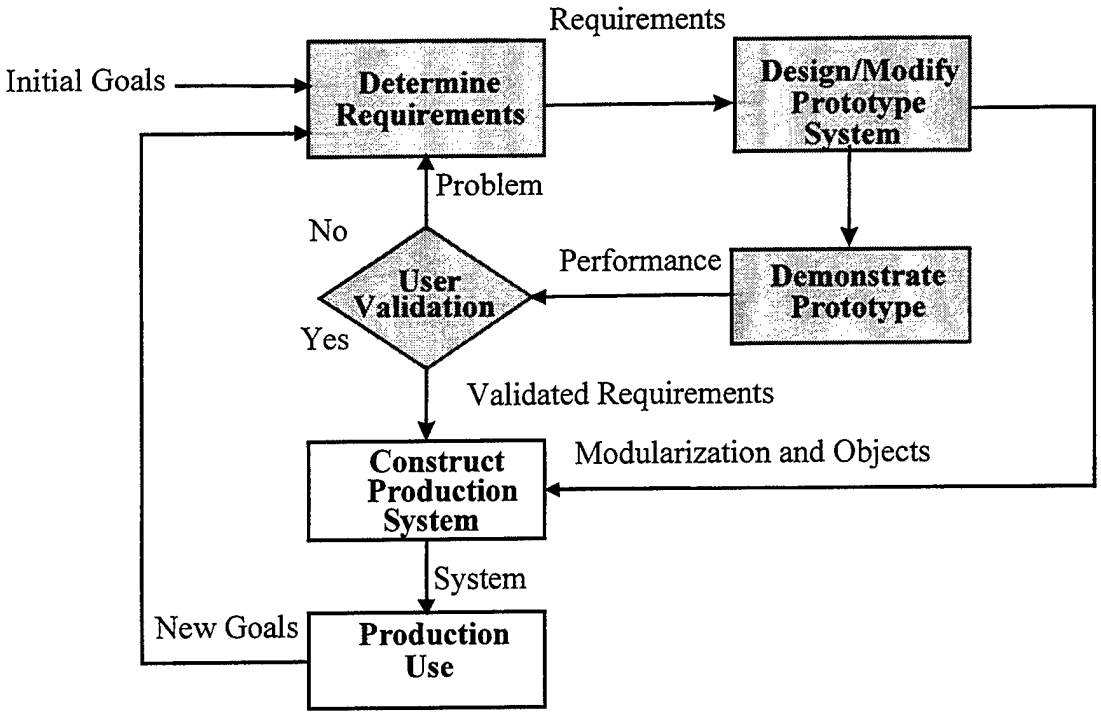

Figure 2. Prototyping Process

B. PURPOSE

The purpose of this study is to demonstrate the port of the CAP system from its existing platform of the sun Sparc 4.1.3 to the Linux O.S.

The Linux O.S. is chosen for various reasons. From an economical standpoint, it makes sense to be able to use Linux due to its very low cost relative to the expensive Sun O.S. 4.1.3. that it currently runs on. In addition, the Linux O.S. is well supported and documented as freeware online and is therefore available to everyone throughout the world. There is a wide research support group, specifically interested in the Linux O.S. These research groups include newsgroups, bulletin boards and other related entities which can be accessed by all users requiring support. After Linux was introduced and made available on

the Internet, Linux users, being mainly PC users, proposed changes to Linux that enhanced its user-friendly capabilities.

C. SCOPE

This study is directed at demonstrating a port of the CAP system to a similar Unix-like operating system. As indicated a port to one of the popular PC Operating Systems such as Windows 95 or Windows NT was not considered.

D. METHODOLOGY

The methodology used in this study consisted of the development of the port and its practical demonstration using the tools provided by both operating systems.

E. ORGANIZATION

This chapter describes the current state of software development, the software engineering problem and the need for prototyping. This need is especially evident in large-scale software projects. Chapter II gives an overview of the installation, networking, and problems concerned with the Linux O.S. Chapter III and IV describe the installation procedures for Xwindows and Motif 2.0 for Linux O.S. These libraries are utilized intensively by CAPS system.

Chapter V is a detailed look into CAPS. It also provides a quick review of the tools and steps required to port the graph editor tool to the Linux O.S. Chapter VI outlines work that is related to our thesis. This related work covers portability issues as well as system

hierarchy requirements. Chapter VII is a summary of our conclusions and discusses possible future enhancements.

II. INTEGRITY OF LINUX OPERATING SYSTEM

A. OVERVIEW OF LINUX

1. What is LINUX?

Today, the computer industry together with software and hardware is one of the fastest developing areas in the world. This stems from the great demand of the market users. As hardware prices go down, the software prices for supporting those platforms begin to go up. Now, a free widely implemented and available operating system can convert a relatively cheap machine into a very cost-effective workstation:

> For professionals who use Unix-based workstations in their work place, Linux permits virtually identical Unix-based working environments on their personal home machines. For cost-conscious educational institutions, especially in developing nations, Linux can create world-class computing environments from inexpensive, easily maintained PC clones. For university students, especially in science and engineering, Linux provides an essentially cost-free path into Unix and X Windows. [Ref. 11]

An Operating System (O.S.) is one of the four major components of a computer system. The others are hardware, application programs and the users. Hardware and application programs which are utilized by users, have to be consistent with the main program of a computer. This main program is called the operating system. The O.S. of a computer provides resource coordination and control for the application programs owned by the users. The O.S. is a control program:

> A control program controls the execution of user programs to prevent errors and improper use of the computer. [Ref. 12]

Linux is a full-featured 32 bit O.S. It is mainly characterized as a UNIX clone for PC's. It is freely distributed at no cost and copyright belongs to Linus B. Torvalds. Unlike the other programs, Linux is developed using an open and distributed model rather than a closed and centralized model. This open-distributed model provides rapid usability and testing for the users. In narrow technical words, Linux can be seen as an operating system kernel providing the basic services.

Linux can turn any 386, 486, or Pentium PC into a workstation and provides all the capabilities of the UNIX O.S. Multi-tasking, multi-users, password security, file protection systems are all supported by Linux. We can use the same commands in Linux as in UNIX such as ls, more, cd, mkdir, etc. Today, with its rapid widespread development, Linux is capable of running Xwindows, TCP/IP networking, and motif. Almost all software application packages have been ported to Linux or are headed in this direction. [Ref. 13]

2. Brief History

In the 1970's, UNIX was one of the main operating systems that provides a multi-tasking, multi-user system for both the micro-computers and the mainframes. Despite its cost, it became the most widely used O.S. In 1991, the first version of the Linux was created. It was born of a project conducted by Linus Torvalds, who was a graduate student in Finland. (torvalds@kruuna. helsinki. fi) After making some progress on this O.S. , he made it available on the Internet. This allowed other users to access its capabilities and soon it became a distributed O.S. model that could be utilized by other users. As he got assistance through the net, Linux released the official version in mid-1992. It was becoming widespread with

hundreds of contributors creating hardware component's drivers, adding system features, debuggers and error checkers. Linux was then made available to anyone wanting to use it by placing it under a General Public License (GPL). This license allowed changes made by others to become readily available for everyone.

From 1993 to 1994, volunteers began to contribute their efforts to the project. It was not only students but a wide range of theorists, scientists and engineers also made a contribution to its development. As a result of the GPL, Linux today has the same properties as any other commercial 32 bit O.S. In addition to its cost-effectiveness, it has much more features compared to other O.S. 's on the market.

3. System Features

In every O.S., much of the basic operating system commands are located together and are commonly referred to as the kernel. The rest of this chapter is focused on the properties of the Linux O.S. kernel. The kernel, which controls the coordination between the user's application programs and the hardware, schedules the processes for multi-tasks. This allows the Linux to support one of its powerful features: multi-tasking. Multi-tasking provides the capability of access by multiple users into a single machine each running a different program. Linux can be referred to as a UNIX O.S. for the PC's, since it supports most UNIX features.

> Linux is a complete, multitasking, multi-user operating system (just like the other versions of UNIX) This means that many users can be logged into the same machine at once, running multiple programs simultaneously [Ref. 13].

a. Software Issues

Today, almost every standard implementation of the UNIX O.S. has been ported to Linux. For example, in Linux, we use the same commands like ls, lpr, more, cd, etc. as in UNIX. In addition emacs and vi text editors can be used in Linux. In UNIX, programs are usually ported by C or C++ compilers. In Linux, one compiler is used for both C and C++. This compiler is the gnat C compiler (gcc). Many other compilers for the various programming languages like Ada, Fortran, Pascal, and Lisp are also available with Linux. As stated in the beginning, all source code for the O.S. is available and is free under the GPL and is therefore considered public domain. This includes the kernel, all drivers, development tools and user programs. All of these programs are freely distributable.

b. Hardware Issues

In order to port or run a program in an environment one need to know the capabilities and the requirements of the system. Since Linux's capabilities were expanded by the users, we can safely conclude that much of the hardware which it supports is readily available to most users. For the last few years there has been an increasing tendency in support for various types of hardware. Today, a wide range of products can be found in the computer's industry. Because of this, it is very difficult to keep track and provide support for those devices. These devices show differences once compared to one another. One can list the requirements according to the devices as follows:

(1) Motherboard and CPU. Linux currently supports systems with an Intel 80386 or better. This means all models of 386's, 486's or better should work properly. The system motherboard uses ISA, EISA, VESA or PCI bus architecture.

(2) Memory. Linux uses very little memory as compared to other O.S.'s. To provide a quick comparison of this issue, Linux requires 2MB while the win95 O.S. requires 8 MB, win NT and OS/2 each require 8MB. As we are all aware the higher the memory, the faster the programs will execute. Although 2MB is a minimal requirement, it is recommended to have at least 4MB for efficient execution. In Linux users allocate some portion of their hard drive as a swap space. This is regarded as virtual memory. This area is used to swap out inactive portions of the code to the disk. This allows more memory to be dedicated to the running program and enables larger programs to be executed.

(3) Hard Drive Requirements. In order to run a program, especially an O.S., one must have some amount of free space on the hard drive. Linux has the capability of supporting various and multiple hard drives on one machine. The space requirements for Linux O.S. depend on each person 's individual needs and goals. As a rule, 10-20 MB is sufficient to support the basic features. However, keep in mind that if one desires to run many programs or wants to expand the platform, then more memory will no doubt be a necessity. If the machine is going to be used by multiple users, one needs to give each user a sufficient working area. This will increase the space requirements for the system.

(4) Monitor And Video Requirements. VGA, EGA, CGA Hercules Super VGA IBM monochrome video cards and monitors are all supported. If videos and monitors are working under another O.S., one can conclude that they are also compatible with Linux. As stated later in the installation section of this chapter, in order to install Linux, one needs to know some specific device features currently in use on the platform such as vertical and horizontal sync values of the monitors and the type of graphic card installed.

(5) Pointing Device, CD-ROM Drivers. Like most of other O.S.'s, the pointing device, usually a mouse, is important to the ease-of-use and user friendly capabilities of the platform. In Linux, our particular installation utilizes a mouse, especially in graphical environments like Xwindows. Linux supports all types of pointing devices such as Microsoft, Logitech, PS/2 etc. Also Linux supports SCII and IDE drivers for CD-ROM's. A lot of CD-ROM drivers have already been ported and provided with Linux. Linux uses the standard ISO-9600 file system for CD-ROM's.

B. INSTALLATION OF LINUX

1. How and Where to Get Linux ?

As stated earlier, Linux is a freely distributable O.S. Therefore, there is no one official distributor for the software. The distributors of Linux O.S. can be obtained via mail, disks, CD-ROM's or ftp from the Internet. In this section we examine where one can obtain the source code from the different distributors.

a. Via Mail or E-Mail Orders

Depending on your system, one can order the software either from the Internet or from some of the distributors represented in Appendix A. Although Linux is free, one may incur some fees for providing this software on disks or CD-ROM's. On the other hand one can borrow it from anyone who already has it downloaded from the Internet. Typically CD-ROM's ordered via e-mail/mail contain some necessary documentation related to the software. In particular, some FAQ's, HOWTO's and demos are usually provided.

b. Via FTP

Linux is a widespread developing O.S. It is continually undergoing development. This development is continually being refined by users who wish to expand its capabilities. The reason for this development is the fact that changes can be made by anyone and are made available to others on the Internet. This strong "information highway" network allows people to communicate and share the desired information easily and rapidly. One of the easiest places to obtain Linux is via ftp sites. Available ftp sites for obtaining Linux are listed in Appendix B. If one chooses to download the Linux O.S. from the ftp site, ensure that the "binary" mode is selected for file transfer. Additionally, one can download the Linux straight to a floppy drive in order to have a backup on hand. The necessary information in each site is determined by a README file.

2. Installation Process

Installing the software is different and it is dependent on the distributor of the code. In our system, we used the slackware 3.0 version of Linux [Ref. 14]. In order to install the slackware version one is required to do the following:

- Create Boot and Root Disks,
- Create partitions with "fdisk" command in Linux for preparing the hard drive for set up,
- Run the setup program that will interactively lead one through the software installation.

a. Creating Boot and Root Disks

In the installation process the first step is to boot the installation media which is typically a floppy disk. In most versions of Linux O.S., one will be entered as " root" when "login" prompt is presented after booting the floppy. In order to boot Linux, one begins at the floppy disk. This floppy disk is for installation.

> Many distributors give you the option of installing LILO on your hard drive. LILO is a program that resides on your drive's master boot record. It is able to boot a number of operating system including MS-DOS and Linux and allows you to select at startup time which to boot [Ref. 13].

In our system we used the slackware version of Linux. In this version once the system prompt is displayed the system will begin to detect your devices.

b. Preparing The Hard Drive

Before setting up the hard drive you need to know some specifications of your particular platform. The best way is to write down every circuit card's specifications in

your system, if available. Besides these cards you need to know the model and brand name of your devices such as CD-ROM driver, video card, monitor. Our system specifications are shown in Appendix C.

After booting up from the floppy you will be prompted by the login prompt. After entering as a root you are ready to repartition the hard drive.

(1) Repartitioning the Hard Drive. In order to create partitions we will use the "fdisk "program. It allows one to create partitions in the hard drive. Using the fdisk command is easy. The commands and the equivalent meanings are as follows:

Command (m for help) m
Command Action

a	Toggle a bootable flag
d	Delete a partition
l	List known partition types
m	Print the menu
n	Add a new partition
p	Print the partition table
q	Quit without saving changes
t	Change a partition system id
u	Change display and entry units
v	Verify the partition table
w	Write table to disk and exit

In Linux, partitions are named according to their disks. For example the first partition on the disk /dev/hda will be /dev/hda1, the second will be /dev/hda2 and so on. If you want to make some logical partitions they should start from /dev/hda5 and so on. The repartition procedure used, including the size and number of partitions is outlined in Appendix D.

(2) Preparing The Swap Space. Swap portions are used for virtual memory. Many distributors require you to create and activate swap space before installing

the software [Ref. 13]. The command "mkswap" is used for creating the swap space. For example when you type:

```
# mkswap -c /dev/hda2 10278
```

The arguments /dev/hda2 and 10278 give the name of the swap partition and the size of swap partition respectively. The option "-c" is the command is used for checking bad blocks. Finally, to enable the swapping one needs to activate by typing in " swapon /dev/hda2."

A typical swap space is two times the amount of RAM that your system is using. This swap space allows Linux to temporarily store program data pages that have been modified in the swap space when they are of lower priority or waiting for input/output operations to complete and make room for higher priority programs that are active in memory. It is only necessary for Linux to store read/write pages in the swap space because fixed text pages and unmodified initialized data are available from the original program file. The decision about the size of the swap space is based on the number of users logged on at a time and the expected read/write page requirements of the software. For example, a database server processing queries from a large number of users would require more swap space than a large number of users who have requirements for E-mail.

3. Running Setup

After running fdisk, you should reboot the system with the boot disk. After logging in as "root," the system prompt "#" will be displayed. When one see the prompt, type "setup." The setup menu shown in Figure 3 will be displayed.

```
# setup
```

Figure 3. The Linux Setup Menu

You may continue by selecting "target" from the menu. One will be given a list of all partitions with the system ID of 83 (Linux native). At this point one should enter the partition which will be used as the root partition (/dev/hda1). Next, select the formatting options. For the Linux System it is possible to display the DOS partitions in order to select a particular O.S. at the time of boot. When prompted, select "yes" then enter the partition name for your MS-DOS partition /dev/hda1. We used "/dos." Now one can enter "q" and save what one did.

The installation process continues by choosing the "source "option from the setup menu as shown in Figure 4. If you intend to install the Linux O.S. from the CD-ROM, select the "CD-ROM" option.

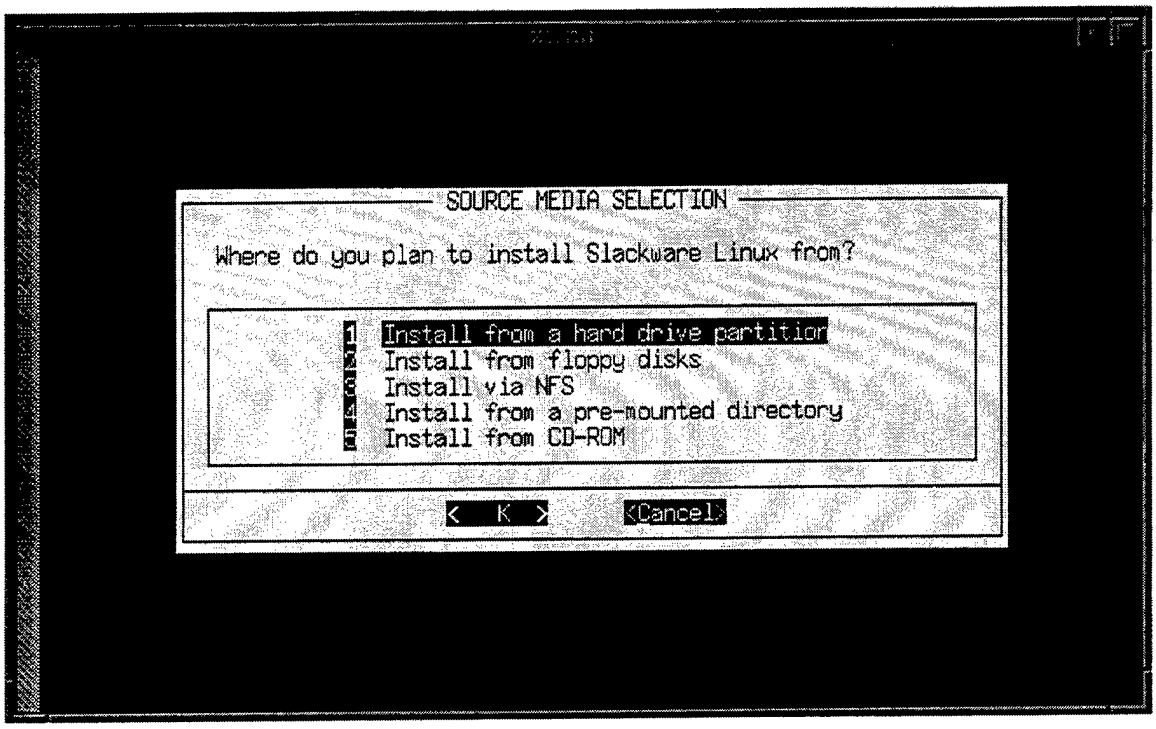

Figure 4. Installing Media

The setup will now begin installing Linux to the hard drive. For recovering from an emergency crash down of the system, a boot disk is recommended in order to reboot. This simple boot disk uses LILO on the hard drive. LILO is a boot loader which can be used to boot into either Linux, MS-DOS, Win95 or any other operating system. The following setup has to be done according to the system features:

Modem Configuration	: No (We used the Ethernet card instead of modem)
Mouse Configuration	: Yes
Custom Screen Fonts	: No

Setup Modem Speed : Cancel

Configuring LILO

Choose "Begin"

Choose "MBR"

Choose " Forever " option that the prompt will wait

Type /dev/hda1

Type "lx"(command for starting the Linux O.S.)

Choose Install

After Installation, a special configuration file named "xf86Config" is required to run. When one opens this file in a preferred editor, one should follow the instructions enclosed with the system.

Mouse Protocol Number : 2 (Mouse system)

Clear DTR and Clear DTS : N

Emulate 3-Button : N

Mouse Device : Enter

Alt Keys : Enter

Monitor Horizontal Synch : 31.5 MHz

Monitor Vertical Synch : 76 Hz

Monitor Identifier Name : Unisys

Video Card : ATI Mach 8

Screen Types : 5

Symbolic Links : Y

/var/X11R6/bin : Y

Ramdac : Generic 8 Bit

After modification of the "xf86 configure" file we are ready to run the Linux O.S.

C. NETWORKING AND COMMUNICATION

Today, it is not enough to have a computer system working as a stand-alone unit. It is also essential to be able to connect, and communicate with the outside world. Connecting to the world, (networking), is the main body of this communication. In this section, we will discuss configuring the network environment and the system.

1. TCP/IP

TCP/IP is a protocol which runs successfully for most of the networked computers. Protocols like TCP/IP provide the rules for communication between the computers. They contain information about handling errors, correction criteria, and data formats.

Transmission Control Protocol (TCP) resides above Internet Protocol (IP) in the protocol layer as shown in Figure 5.

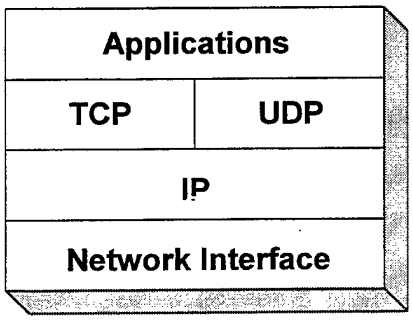

Figure 5. Conceptual Layering TCP, IP AND UDP

In some applications UDP (User-Defined Protocol) is used in place of TCP. UDP is not as reliable as TCP because it does not send/receive acknowledgments. In applications, where reliability is not an issue, UDP is better to use because it's faster then TCP.

On a TCP/IP network, each machine has an identifier called an IP address. This identifies the computer to the hosts. The IP address consists of two parts, the network and host portion as shown in Figure 6.

The size of these two depends upon the network size. [Ref. 15] In our system, luqipc, domain name for the computer, has host number as 131.120.1.212. By using the netstat -rn command we can see the following kernel routing Table 1. This command for

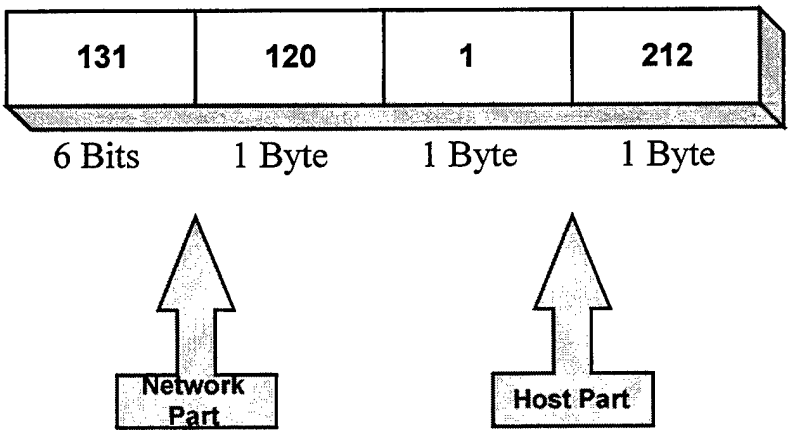

Figure 6. IP Address Breakdown

Linux generates a listing of the active network connections in our machine. The "netstat" program is an invaluable tool which is used to monitor the TCP/IP network.

luqipc# netstat -rn

Destination	Gateway	Genmask	Iface
131.120.1.0	0. 0. 0. 0	255.255.255.0	2 eth0
127.0 0.0	0. 0. 0. 0	255.0.0.0	1 lo

Table 1. Kernel Routing Table in "Luqipc"

2. Configuring TCP/IP With Ethernet

To connect the ethernet card to the LAN, one needs to have an IP address which can be assigned by the System Administrator or network access provider. Setting up the network configuration for Linux is straightforward. If you are configuring loopback mode (i.e., no SLIP, no ethernet card) the IP address should be 127.0.0.1. The reason that we said "no" in the Table 2 to the loopback is "loopback-only" systems do not have a broadcast address by typing "netconfig" at the system prompt [Ref. 13]. The following table will be displayed.

Domain Name	Luqipc
Host Name	cs. nps. navy. mil
Are we using Loopbacks	No
IP Address	131.120.1.212
Gateway	131.120.1.212
Netmask	255.255.255.0

Table 2. Netconfig Table

In addition to this setup we can add the the hosts that we want to the /etc/hosts file

131.120.1.212 luqipc. cs. nps. navy. mil

131.120.1.213 luqinotebook. cs. nps. navy. mil

3. Hardware Requirements

To use TCP/IP to provide connectivity between hosts, a connection can be obtained through the use of a modem. If a TCP/IP connection via an Ethernet is needed, than the added hardware requirement is an Ethernet networking card. The following Ethernet cards following cards are supported by Linux O.S.

3com 3c503, 3c503/16

Novell	NE1000, NE2000
Western Digital	WD8003, WD8013
Hewlett-Packard	HP 27245, HP27247, HP27250 [Ref. 13]

With each improved kernel, more products are supported. More information about this area can be obtained at the Internet site *http://jaka.nn.com/lpd/HOWTO/Ethernet-HOWTO.html*

4. World Wide Web

For the Linux O.S., there is also an available version of the Netscape internet browser. You can simply download the browser from the homepage *http://www.netscape.com/.* After selecting the Linux product and downloading it to your hard drive under the root directory, one needs to unzip and unttar the file. Use the following commands to accompish this:

gunzip <file name> to open the zipped file.

tar xvf <file name>

To open the tared file. After untaring one is ready to run the internet browser as shown in Figure 7.

D. SHUTTING DOWN THE SYSTEM

When one is done with the system, a proper shutdown is required. In order to shutdown the system properly, The command *"# shutdown -r now. "* should be entered. If the system was suddenly turned off without entering the proper command, all data could be lost or corrupted. The following options are valid for the shutdown command:

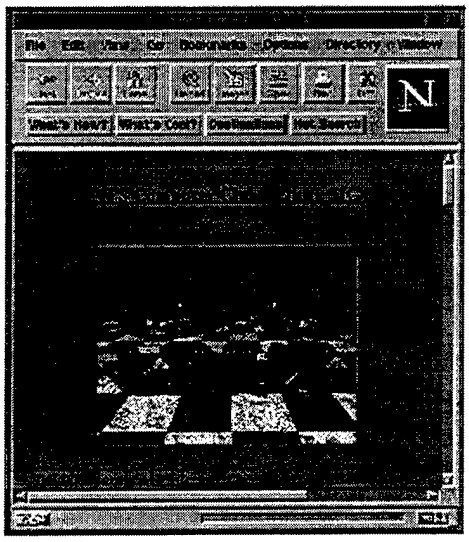

Figure 7. Netscape Internet Browser Netscape Running under Linux O.S.

shutdown -r now : Reboot the system after shutdown,

shutdown -c now : Cancels the current shutdown operation.

A detailed list of shutdown options can be found in the manual page of the Linux O.S.

E. PROBLEMS ENCOUNTERED

1. Unknown Information about the System

This section describes the problems encountered in installing the Linux O.S. As stated in the beginning of this chapter, one needs to know some basic specification of certain devices. It is recommended that one first list all the hardware devices on the platform and then list their corresponding features. Table 3 lists the specifications of the "luqipc" system.

If one is unaware of some of these specifications, one should look in the databases provided in the documentation section of the distributor. Our initial problem was finding the video card and and monitor information. After consulting with the vendors, sending the

problem to the related Linux newsgroups, and searching in the data bases we managed to obtain the missing information.

System Specifications of the "Luqipc" in CAPS Project	
Motherboard	Pentium 100 w/32Mb RAM, 512Kb Cache
Controller	Onboard EIDE and Floppy
Video Card	ATI Mach 8 DRAM 1024 Kb
Monitor	Unisys
	Horizontal Sync 31.5 MHz.
	Vertical Sync 76 Hz
Sound Card	Creative S. B 16 Bit w/ speaker
Mouse	Microsoft Compatible, 3 Button generic
Hard Drive	WD 2.1 GB
CD-ROM	Creative 4x speed IDE
Network	3com Ethernet card
Standard 1.44 Mb 3.5 Floppy and Keyboard	

Table 3. Luqipc System Specifications

2. CD-ROM Driver Support

After installing Linux O.S., we encountered problems in trying to access our CD-ROM. The problem was that Linux was not recognizing our cd-rom driver. We searched through the net and found that we needed to recompile the kernel [Ref. 16]. In order to compile the kernel, one should follow the commands descibed below after getting the system prompt " #. "

make config : Which asks various questions about the drivers

make dep : To gather dependencies for each file and include them in the various makes file

make clean : To clear from object files

make : To build the kernel

The above listed steps may take some time, it is dependent upon each individual platform setup. After recompiling the kernel, the CD-ROM was recognized and the Linux installation was complete. [Ref. 17]

III. X11R6

A. WHAT IS X11R6?

The X Window System, X11R6 also referred to as 'X', is an industry-standard software system that supports the development of portable graphical user interfaces. One of its main advantages is its unique device-dependent architecture, so one can run programs on any hardware that supports the X protocol without having to modify, recompile, or relink those programs. Instead of enforcing a particular user interface style (or "look and feel"), X provides basic mechanisms to support many different styles, which are implemented as libraries of widgets (or interaction objects) on top of basic X functionality. One of those styles is the Open Software Foundation (OSF)'s Motif widget set. X uses a client-server architecture in which the X Server is the server which handles all the physical input and output devices. This provides a portable layer between applications and display hardware. A client is an application that uses the facilities provided by the X Server and communicates with it via a network connection. Therefore, multiple clients can connect to one server, and a client can connect to multiple servers. Both the X server and clients are processes that can execute on the same machine or different machines.

In X, a display is a single X server process (so, display and server are interchangeable terms) and a screen is a single hardware output device. A single X display can support many screens. The X server controls all resources used by the window systems (e. g., windows, bitmaps, fonts, colors). When a client application needs a service from the

X server, it sends a request to the server. For example, a client could request the server to create or destroy windows or display text or graphics in a window. The server places requests in a queue and takes care of them on a first-come, first-served basis. Clients do not wait for the server to respond to requests (i. e., the client doesn't "hang" until the request is handled). The most fundamental resource in X is the window—a rectangular section of the screen that has a background color and border. Any client that has the window's resource identifier (resource ID) can request the server to manipulate the window (e.g., change its position, size, color, etc.). X organizes windows hierarchically in what is called a window tree. The top window is called the root window; the X server automatically creates a root window for each screen it controls. The root window occupies the entire physical screen and cannot be moved or resized.

Every window (except for the root window) has a parent window (or ancestor) and may also have children (or descendants or "subwindows"). The X11R6 environment running on the Linux O.S. using the "fvwm" window manager can be seen in Figure 8.

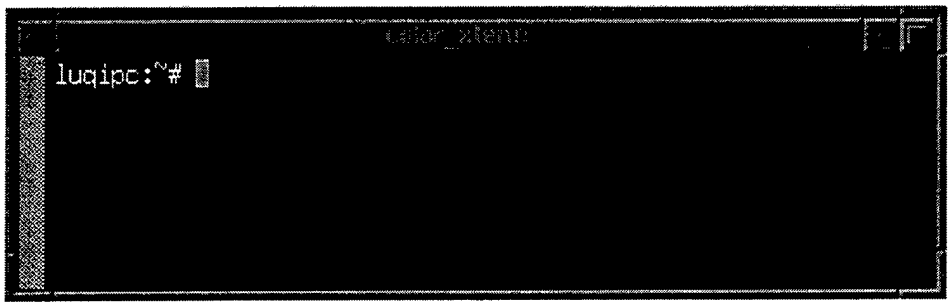

Figure 8. Xwindows xterm Running Under Linux O.S.

The X server clips portions of children that lie outside the bounds of the parent window. X allows siblings (windows with the same parent) to overlap. Clients may request the server to change the stacking order by raising or lowering windows. Each X window, including the root window, has its own integer coordinate system, starting from (0,0) in the upper left corner. The x coordinate increases towards the right and the y coordinate increases towards the bottom. The units used in the coordinate system are pixels and they are dependent on the screen resolution e.g. 840x1024. When the server creates a window, it allocates and initializes the data structures that represent the window within the server but does not actually display that window on the screen. Clients can issue a map request to make the window visible. But the window may not actually become visible if 1) it is obscured by another window on the screen, 2) one or more of its ancestors is not mapped, or 3) it is completely clipped by an ancestor.

A window manager allows the user to control the size and location of windows on the screen. The window manager is ordinarily a client application (e. g., mwm is the Motif Window Manager).

B. LOCATION

X11R6 can be obtained free or for a small nominal fee as on the Internet cites as listed in Appendix E. The installation of the X11R6 should be placed right below the '/usr/ directory. It contains various sub-directories such as 'bin', 'lib'. One important thing to remember is to ensure that the "/usr/x11r6/bin" is on your path. To ensure accomplishing this one must edit the '. cshrc' file.

C. FEATURES

This section describes some of the new features and changes in the X Consortium distribution. Release 6 contains much new functionality in the following areas. Each functionality is explained under its own section. [Ref. 18]

1. Software Features

The following are new X Consortium standards in Release 6.

a. X Image Extension

Release 6 fully supports all JPEG compression formats. The JPEG compression and decompression code is based on the Independent JPEG Group's (IJG) JPEG software. This software was chosen as a basis for implementation of JPEG compression and decompression. It achieves portability and flexibility without sacrificing performance.

b. Inter-Client Communications Conventions Manual (ICCCM)

Release 6 includes version 2.0 of the ICCCM which contains a large number of changes in window management, selections, session management and resource sharing.

(1) Window Management. The interface between the user and the Xwindows server is conducted via the window manager. It interfaces all calls and requests for X services. All requests to the X server are received, processed and replied to the users display. The window manager determines which process ID's and in what sequence are output to the Xserver.

(2) Selections. A number of new targets for Encapsulated PostScript and for the Apple Macintosh PICT structured graphics format have been added. A selection requester can now pass parameters in with the request. Another new facility is manager selections. This use of the selection mechanism is not to transfer data, but to allow clients known as managers to provide services to other clients. Version 2.0 also specifies that window managers should hold a manager selection.

(3) Resource Sharing. A prominent new addition in version 2.0 is the ability of clients to take control of color map installation under certain circumstances. Earlier versions of the ICCCM specified that the window manager had exclusive control over color map installation. This proves to be inconvenient for certain situations, such as when a client has the server grabbed. Version 2.0 allows clients to install color maps themselves after having informed the window manager. This enables the X server to be more efficiently utilized by other clients sharing its resources.

c. Inter-Client Exchange Protocol

ICE provides a common framework to build protocols on. It supplies authentication, byte order negotiation, version negotiation, and error reporting conventions. It supports multiplexing multiple protocols over a single transport connection.

d. X Session Management Protocol

The X Session Management Protocol (XSMP) provides a uniform mechanism for users to save and restore their sessions using the services of a network-based session manager. It is built on ICE. A new protocol, rstart, greatly simplifies the task of

starting applications on remote machines. It is built upon already existing remote execution protocols such as remote shell (rsh). The most important feature that it adds is the ability to pass environment variables and authentication data to the applications being started.

e. Input Method Protocol

Some languages need complex pre-editing input methods, and such an input method may be implemented separately from applications in a process called an Input Method (IM) Server. The IM Server handles the display of pre-edit text and the user's input operation. The Input Method (IM) Protocol standardizes the communication between the IM Server and the IM library linked with the application.

The IM Protocol is a completely new protocol. The following new features have been added:

- The IM Server can support any of several transports for connection with the IM library.
- Both the IM Server and clients can authenticate each other for security.
- A client can initiate string conversion to the IM Server for reconversion of text.
- A client can specify some keys as hot keys, which can be used to escape from the normal input method processing regardless of the input method state.

f. X Logical Font Descriptions

The X Logical Font Description has been enhanced to include general 2D linear transformations, character set subsets, and support for polymorphic fonts.

g. SYNC Extension

The Synchronization extension lets clients synchronize via the X server. This eliminates the network delays and the differences in synchronization primitives between operating systems. The extension provides a general Counter resource; clients can alter the value of a Counter, and can block their execution until a Counter reaches a specific threshold. Thus, for example, two clients can share a Counter initialized to zero, one client can draw some graphics and then increment the Counter, and the other client can block until the Counter reaches a value of one and then draw some additional graphics.

h. BIG-REQUESTS Extension

The standard X protocol only allows requests up to 2^{18} bytes long. A new protocol extension, BIG-REQUESTS, has been added that allows a client to extend the length field in protocol requests to be a 32-bit value. This is useful for PEX and other extensions that transmit complex information to the server.

i. XC-MISC Extension

A new extension, XC-MISC, allows clients to get back ID ranges from the server. Xlib handles this automatically under the covers. This is useful for long-running applications that use many IDs over their lifetime.

j. Kerberos

There is a new authorization scheme for X clients, MIT-KERBEROS-5. It implements MIT's Kerberos Version 5 user-to-user authentication. As with any other

authentication protocol, xdm sets it up at login time, and Xlib uses it to authenticate the client to the X server.

k. Internationalization

Internationalization (also known as I18N, there being 18 letters between the i and n of the X Window System, which was introduced in R5 and has been substantially improved in the R6 version. The R6 I18N format is based mainly on the ANSI C and POSIX models. Most of its capabilities are derived from the Xlib. In the R5 version, a basic left to right, non-context sensitive codeset language was introduced. Unfortunately, it had some shortcomings. It was still unable to cross all possible language and cultural conventions. To date R6 does not fully cover all language and cultural conventions, although it does make substantial improvements in this area.

The additional support is mainly in the area of text display. R5 originally introduced the idea of a font set. R6 took this idea and expanded it to include a more generalized method of displaying context input and output. The result is a general framework to enable bi-directional text and context sensitive text display.

l. Fresco

R6 includes the first sample implementation of Fresco, a user interface system specified using CORBA IDL and implemented in C++. Fresco is not yet a Consortium standard or draft standard, but is being distributed as a work in progress.

m. LBX (Low Bandwidth X)

The X Consortium is working to define a standard for running X applications over serial lines, wide area networks, and other slow links. This effort, called Low Bandwidth X (LBX), aims to improve the startup time, performance, and interactive feel of X applications run over low bandwidth transports. LBX does this by interposing a pseudo-server (called the proxy) between the X clients and the X server. The proxy caches data flowing between the server and the clients, merges the X protocol streams, and compresses the data that is sent over the low bandwidth wire. The X server at the other end uncompresses the data and splits it back out into separate request streams. The target is to make many X applications transparently usable over 9600 BPS modems.

2. Hardware Requirements

There are specific hardware requirements that X11R6 imposes on video, monitors and memory. Although various vendors will state that the video cards and monitors should work and are compatible, this is not necessarily the case. Specific video card specifications must be obtained. This could be accomplished by one of three methods. First one can get the specs from the vendor, second one can call the technical support line for the video card to obtain the specifications or lastly, one can run a software program such as "superprobe" to get the specs identified.

Hardware requirements for running Xwindows is a 486 machine with at least 4 Mb of RAM, although a significant performance increase could be realized by having 8Mb or more memory. A list of supported SVGA chipsets can be seen in Appendix F. [Ref. 13]

IV. MOTIF

A. OVERVIEW OF MOTIF

Motif is a widely-accepted set of user interface guidelines developed by the Open Software Foundation (OSF) around 1989 which specifies how an X Window System application should "look and feel." OSF/Motif, as it is more formally called, includes the Motif Toolkit (also called "Xm" or the "Motif widgets"), which enforce a policy on top of the X Toolkit Intrinsics ("Xt"). Xt is really a "mechanism not policy" layer, and Xm provides the specific "look and feel." For example, Xt does not insist that windows have titlebars or menus, but it provides hooks for developers of specific toolkits (Motif, OpenLook, Athena widgets) to take advantage of. In addition to widgets, OSF/Motif includes the Motif Style Guide document which details how a Motif user interface should look and behave to be "Motif compliant."

B. MOTIF COMPONENTS

Typically, when motif is purchased as a bundle from a particular developer, it is not much different than other available motif packages. The most typical components included with most motif bundles are as follows:

- The mwm Window Manager
- Sample. mwmrc files. (Resource files)
- libXm, libMrm, libUil, libWmWsl

- UIL Compiler, Header and Include Files and everything you need to compile and link Motif Applications.
- On-line Manual Pages
- OSF/Motif demo program and associated documentation including all libraries and source code.
- The OSF/Motif Users Guide
- The full OSF documentation set in both postscript and ascii including the User Guide, Style Guide, Programmers Guide, Widget Guide, comprising over a thousand pages.

C. SYSTEM REQUIREMENTS

Motif requires at least 15MB of free disk space in order to run efficiently. Version 1.1.18 or higher should be used along with libc 4. 4. 4. In addition, XFree86 2.0 or 3.1 is the optimum choice. The use of libc. so. 4.6.27, or better is strongly recommended.

D. INSTALLATION

First, Motif is not free. Once purchased, Motif is a 100% OSF/Motif port to Linux. One will find everything one needs to develop and use Motif applications, the mwm window manager, uil compiler, and shared and static libraries. In addition, the full OSF documentation set is included.

When installing, the CD-ROM contains previous releases of Motif in the directory Moo-Prev. This is done in order to maintain backwards compatibility as Linux changes occur. This contains the binaries from the last two Motif releases and is there for those that are using older Linux versions or distributions with mixed X11R5 and X11R6 libraries.

The current version of Motif is in a directory Motif/Moo-3 and these are the binaries and Installation scripts that were used.

The Moo-ELF directory contains binaries for ELF4 and ELF5 versions of Linux. ELF Linux is changing rapidly and, no doubt, there will be updates to this release to match the new developments.

1. Installation Steps

a. Read the FAQ directory.

b. Logon as root.

c. Ensure that the file systems you are going to install into are writable (i.e., are not on a CD-ROM). Linux CD-ROM distributions seem to vary in the directory layouts used. One should change the permissions of the following directories in order to be able to write to them:

- /usr/X386/include

- /usr/X386/lib

- /usr/X386/bin

On some systems the X386 directory may be called X11R6 and on others it may be a symbolic link to a directory X11R6.

d. Run the Install script in the Motif/Moo-3/disks directory. Watch out for any error messages that appear. Errors may occur due to a lack of disk/inode space or some attempt to write to a CD-ROM. There may also be broken-pipe warnings on some Linux systems. These can be ignored.

Install is now complete. To start X and invoke the mwm window manager make certain that /usr/X11R6/bin is in your path. Once X has been invoked, the following X windows is displayed to the user by the Motif window manager (mwm).

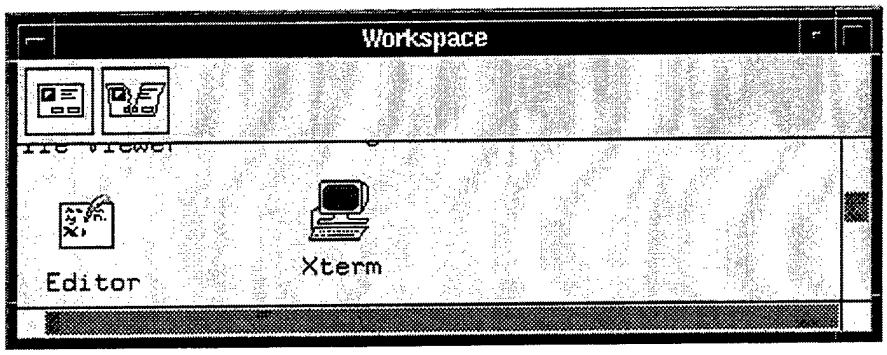

Figure 9. Motif Windows Manager's Workspace Display

E. PROBLEMS ENCOUNTERED

1. Warnings About Locale's Not Being Defined

There seems to be a problem with the Locale's of some Linux's. It can be fixed with the following script:

```
#!/bin/sh
#
# Work around mwm bug; mwm does not like LC_CTYPE set to ISO-8859-
  1
unset LC_CTYPE
exec mwm. original
```

Where mwm. original is the mwm supplied with the distribution.

2. ELF5 changes

If using the August 1995 or earlier version of Motif, changes have to be made to some of the files in order to comply with updates to the newer release of Linux. These

changes and the required files and procedures can be found on the Internet at http:// www. motif.com.

Linux is switching to a different format for executables, object files and object code libraries, known as `ELF5' (the old format is called `a. out'). This new format has many advantages, including better support for shared libraries and dynamic linking. Both a. out and ELF binaries can coexist on a system. However, they use different shared C libraries, both of which will have to be installed.

There is a patch to get 1.2. x to compile using the ELF compilers and produce ELF coredumps. It can be found at ftp site tsx-11. mit. edu under /pub/packages/GCC. You do not need the patch merely to run ELF binaries. Version 1.3. x and later do not require this patch.

V. CAPS

A. OVERVIEW OF CAPS

1. What is CAPS?

The Computer Aided Prototyping System (CAPS) is a set of tools used to develop prototypes of real-time systems. It can be used to satisfy system requirements, feasibility studies, design of large embedded systems, and customer needs. CAPS has a capability of determining feasibility of a large software projects prior to implementation. This saves time and money for the costs of development of the projects. The cost-effectiveness comes from the fact that it can determine missing system requirements or unfeasibility of certain large-scale projects. The tools that make up CAPS are broken down to four separate groups: software base, editors, execution supports and project control. These components are explained later on in this chapter. The tools that consist of CAPS are presented to user in the form of a user-friendly Graphical-user-interface (GUI) as presented in Figure 10. CAPS is a public domain software which produces fast and reliable code for the prototyping systems. From a software engineer point of view, it saves time and money in the further implementations. CAPS enables the user to test the timing feasibility of the requirements, automates assistance for project planning, scheduling, task assignment, create code via both graphical and text editors, incorporates the reuse of well tested Ada code which resides in the software base.

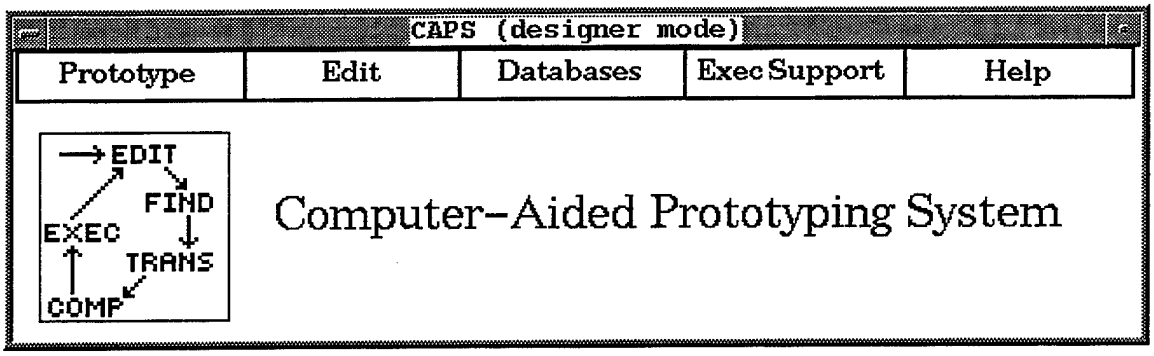

Figure 10. The CAPS User Interface

In general [Ref. 3]

- Automates software development,
- Improves software quality,
- Reduces development time,
- Decreases life-cycle costs,
- Supports mil-std 498 for software acquisition and development,
- Aids in the dod software reuse initiative (sri), and
- Provides better overall project management.

Some of the benefits CAPS provides the user are [Ref. 19]:

- Graphic model representation of the software design,
- Non-procedural annotations, requiring no specific programming language knowledge during the design and specification phase,
- Automatic generation of code free of syntax errors and interface consistency errors,
- Automatic generation of schedules to meet strict real-time deadlines,
- Support for computer-aided generation of graphical user interfaces and simple animation of prototype behavior,
- Fast and early feedback to the user through an executable prototype,

- Early detection of errors in the requirements phase,
- The ability to firm up requirements before production through iterative assessment and modification of graphical representations, as well as requirements changes after delivery,
- Easy modification of software designs to meet the customer's changing needs,
- Requirements tracing through facilities for recording dependencies, requirements to specification, and preserving design information, and
- Computer-aided assessment of hardware/software tradeoffs relative to different types of hardware.

2. History

Originally CAPS began as an idea of Prof. Luqi when she was a Ph. D. student at Minnesota University. In 1986, she realized that much of the time spent in software engineering was in the debugging stage, instead of in the requirement analysis stage. It was well known to the software engineering field that a requirement oversight could be very costly when found to be unfeasible during the debugging stage. If found during this stage, many resources could have already been dedicated. The project therefore would have sustained a considerable loss in order to find this unfeasibility. Prof. Luqi theorized that if a tool was made available that was capable of quickly determining the feasibility of each requirement, resources could then be utilized more cost-effectively. In addition, much of these resources were used to ascertain the correctness of large-scale projects. This is commonly referred to as verification. Verification is an important required task that engineers, scientists and mathematicians must undergo prior to implementation. This task is done in order to ensure and document the project feasibility.

The requirement analysis in large-scale programs such as air traffic control, defense systems, C3I have to undergo countless hours of feasibility testing before implementation could even begin. Prof. Luqi pointed out that rapid prototyping was the answer to this problem. Such a tool would require a language that could quickly outline the general requirements and display whether or not such requirements were feasible. She conceptualized a computer aided prototyping system, CAPS. In 1990 the first version of this theory was implemented. It was capable of rapidly prototyping a large-scale software engineering project in order to test the feasibility or unfeasibility of each requirement. During this time frame, Ada was the required language for all DoD applications and therefore, the software databases was constructed with mainly Ada applications. The first version of CAPS was Release 1. It was made available to the public on a CD-ROM in February 1996, and became an integral part of the Public Ada Library (PAL).

3. System Requirements

CAPS Release 1 runs on a Sun Sparc station running Sun O.S. 4.1.1 or later with Xwindows(X11R4 or X11R5), Motif 1.1.2 or later, Sun Ada Compiler 1.1, and TAE+ V5.3 and 130 Mb disk space [Ref. 19]. The supported software and hardware platforms for the envisioned Linux version are as follows:

Linux (Full Installation)	400 Mb
CAPS Release 2(with no source code)	47.4 Mb
CAPS Release 2(with source code)	150.3 Mb
Gnat Ada Compiler	14.8 Mb
TAE + V 5.31	48 Mb
Motif 2.0	15 Mb

Table 4. The System Software Requirements for the Envisioned Linux Version

Processor	Pentium 100 Mhz
HDD	2.5 Gbyte EIDE
CD-ROM DRV	Creative CD-ROM Drv 4 X
RAM	32 Mb
SWAP SPACE	64 Mb
GRAPHIC CARD	Chips 64545
MONITOR	Unisys 19 " SVGA

Table 5. The Hardware Specs For The "Luqipc"

In order to run CAPS effectively the following specs are recommended :

- 32 Mb of memory to avoid paging and inaccurate timing results.
- 64 Mb swap space to allow applications to be created.

4. Components

CAPS components are divided into four major groups, software base, project control, editors and execution support as shown in Figure 11.

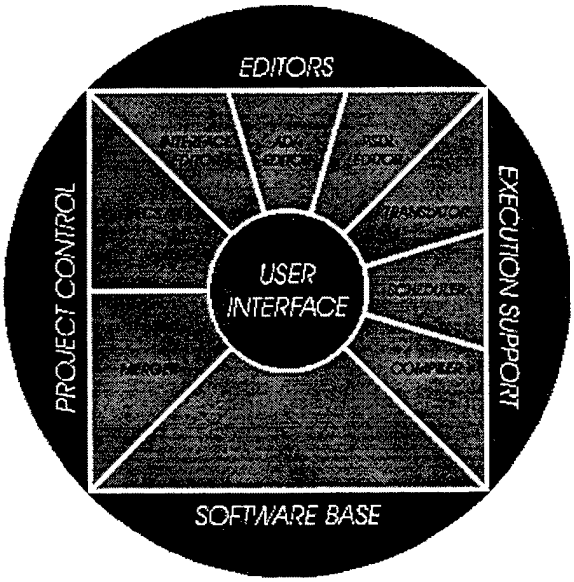

Figure 11. The Main CAPS Components

a. Software Base

The software base consists of an ADA program database which can be reused in the development environment as in Figure 11.

b. Editors

(1) PSDL. CAPS prototyping is mainly developed through the use of PSDL (Prototyping System Description Language) editors. The PSDL consists of the Syntax Directed Editor (SDE), the graphical editor and the graphical viewer. The PSDL uses the graphical editor to depict the operators, data flow streams and program constraints. This graph is converted to PSDL code and can be modified with the use of the SDE as shown in Figure 12. The SDE is created using the Synthesizer generator.

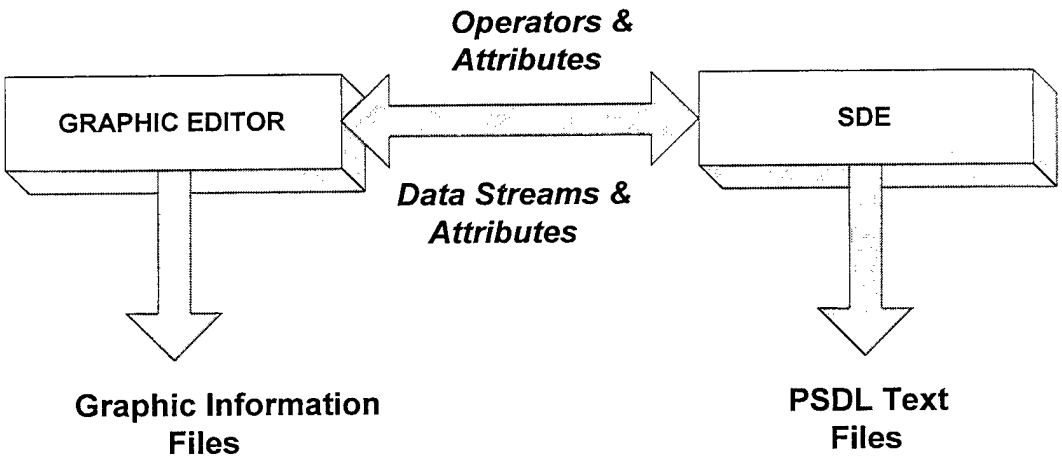

Figure 12. The Communication Between the SDE and the Graphic Editor

The PSDL editor is pictured in Figure 13.

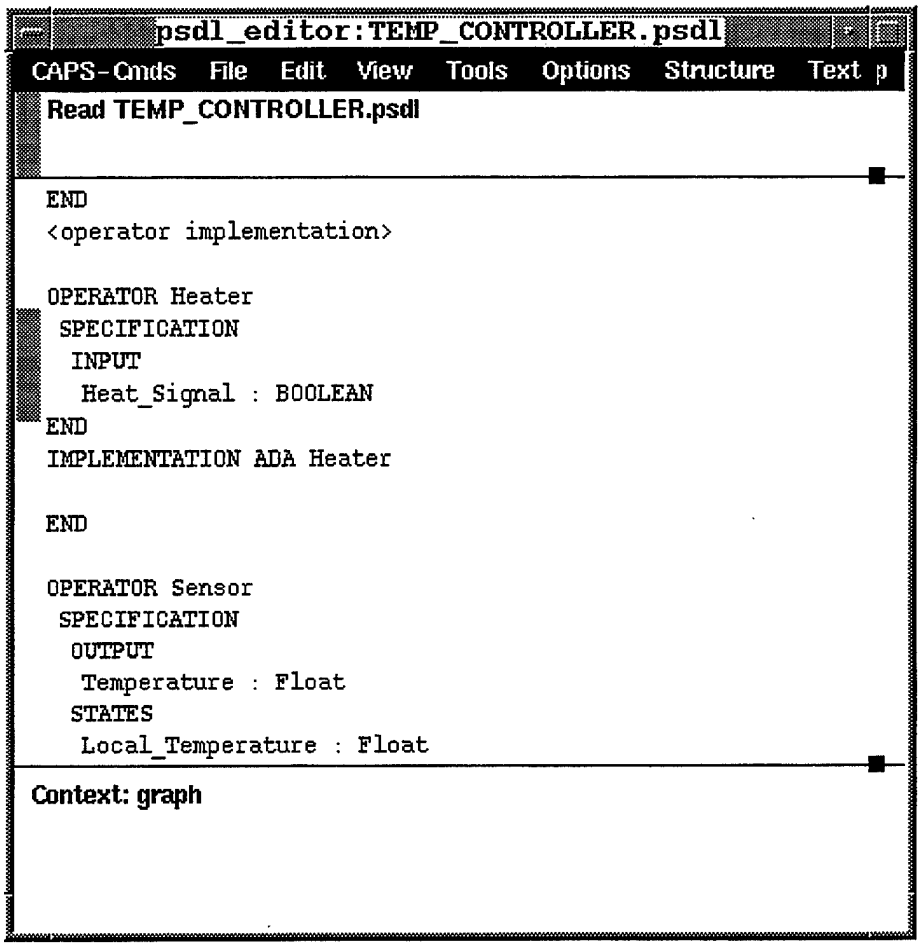

Figure 13. PSDL SDE

(2) Text Editor. Although the text editor is not exclusively a CAPS tool, CAPS provides text-editing facilities. Emacs, vi and the Ada Verdix SDE can be selected by the users in order to edit text files.

(3) Interface Editor. Within the CAPS environment TAE Plus is utilized to construct graphical user interfaces for prototyping being developed. The GUI is constructed by the user. TAE generates code from the GUI that was developed by the user.

The interface editor is used to integrate the TAE GUI code with the generated Ada code from the SDE.

c. Execution Support

Execution support is divided into three main areas. The translator, scheduler and compiler.

(1) The Translator. The translator converts the generated PSDL code into ada packages. In order to accomplish this, the PSDL code provided to the translator must be complete. These constructed 'packages' are placed in the 'supervisor module' of the prototype. It is important to note that the translator does not create packages for user-defined types or atomic operators. In short, the translator generates Ada code from PSDL code.

(2) The Scheduler. The function of the scheduler is to determine the feasibility of timing constraints within the program. The scheduler generates two separate schedules, one static and one dynamic. The static schedule is used to schedule events that must occur in a preset sequence. The dynamic schedule is used to schedule events that are not time-dependent or dependent upon other events. They can be scheduled anytime during the program. Initially, the static schedule takes precedence. The scheduler will sequence the events in accordance with the static schedule requirements, if feasible. If unfeasible, diagnostic information will be displayed to the user in order for him to determine the timing constraints that require modification. Once the static schedule is generated, the events in the

dynamic schedule can be placed within the program's non time-critical time slots. The scheduler has now incorporated the time constraints into the prototype.

(3) The compiler. The compiler is used to compile the generated Ada code. The prototype must have been successfully translated and scheduled prior to the compilation process.

d. Project Control

Project Control is a control mechanism utilized by the project manager to assure proper development of the prototype. During development of large-scale projects communications is a big problem due to the fact that meeting between individual teams are required in order to insure consistency and cooperation. In order to accomplish this effectively, an automated system is required to support this. Project Control is divided into two main areas Evolution Control System (ECS) and the Merger.

(1) ECS. ECS is a large data base that allows team members and project members to access to the current status of prototype development. Each team can enter their current state of development into the database in order for other team to easily access their data to insure project consistency. In addition, the project manager designates the required steps that must be accomplished in order to successfully create the prototype. These steps are automatically scheduled and assigned to available designers.

(2) The Merger. The main purpose of the merger is to merge different PSDL programs into a single PSDL prototype. This allows parallel development of the prototype to insure creativity of design. Initially, teams concurrently develop their

assigned PSDL tasks. Once completed these individual tasks are combined via the merger. The merger reviews these individual tasks and makes one of two decisions. If differences can be reconciled, the merger will overwrite certain tasks, if not, the merger will display error messages to the user. In order to complete the merging process, the user must make decisions for solving the differences. Once all differences are resolved, the merger generates a single PSDL prototype.

B. PORTING CAPS

1. Building Underlying Platforms

Currently CAPS Release 1 runs on Sun Sparc workstations running Sun O.S. version 4.1.3 which is a version of UNIX. This limits the use of CAPS due to the unavailability of workstations to the common user. If a PC version was made available to the public domain, CAPS would get much wider usage. This is the direction we chose to proceed.

After examining the different O.S. on the market, we concluded that the Linux O.S. would best suit our needs. The reason is that the Linux O.S. is considered a Unix-Based O.S. for PC's as described in Chapter II. Since CAPS release runs on a Unix platform, choosing Linux requires significantly less software modification than choosing a non-Unix O.S. such as Windows 95. In addition, Linux is available as freeware on the Internet. Another factor for choosing Linux was its cost. It is free. This thesis focuses on porting CAPS to different platforms. In order to do this we examined the underlying software support by CAPS. Currently CAPS has the layers depicted in Figure 14.

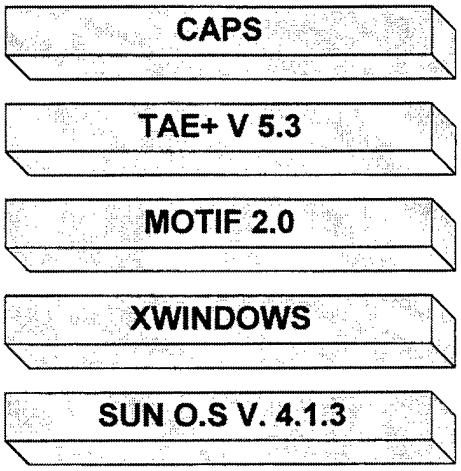

Figure 14. The Current CAPS Release 1 Layers

Our goal was to build the following layers compatible to PC running Linux O.S. in Figure 15.

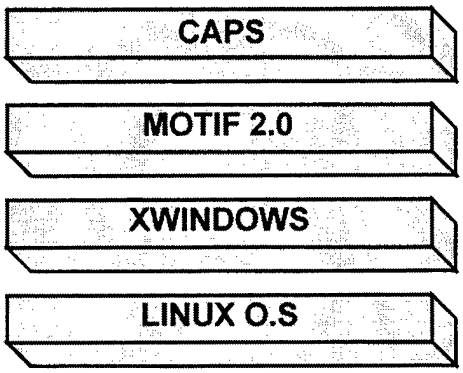

Figure 15. The Envisioned Version of CAPS Layers

One of the underlying software of CAPS component is TAE+. Based on our research results we determined that TAE+ is not available for a PC version and is not scheduled to be supported by a PC. At this stage of development, we decided that CAPS could be successfully ported to a PC without TAE support.

2. Porting Graph Editor

The Graphic Editor is used to graphically depict the prototype. The functions that the graphic editor makes available to the user are: circles, squares, lines and the ability to select and label properties. The CAPS graphic editor ported to the Linux O.S. is in Figure 16.

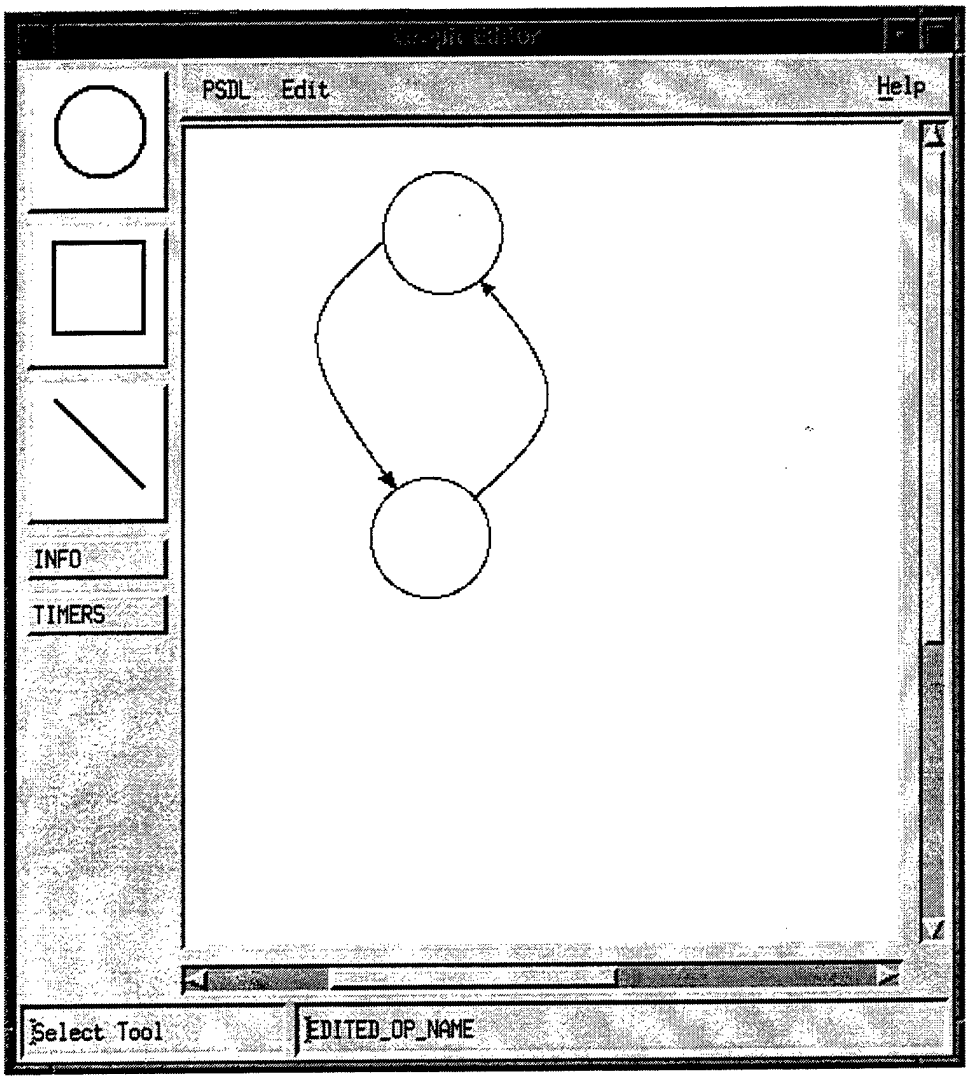

Figure 16. Graph Editor Ported to Linux O.S.

The circles are commonly used to draw circular operators which represent software components. The squares are used to represent external input/output to the prototypes. Data streams are shown as lines between operators showing direction of data flow. The "Select" option allows the user to grab the objects and manually move them on the screen. The "Properties" option is used to label the selected objects.

Once the graphic representation has been completed by the user, the graphic editor will generate PSDL code. This code requires timing and control constraints to be entered via the SDE.

The CAPS graphic editor source code from Release 1 was loaded into the "Luqipc." A new make file was created to enable the compilation of the source code on the PC. The created Make file is presented in Appendix G. Using "gcc" for the Linux O.S. the source code was compiled. As expected, it did not compile. The debugging process required much modification of the code in order to successfully compile.

After making the required modifications to the source code, the code successfully compiled. The resultant executable was run on the PC and properly displayed the graphic editor interface. More modifications were accomplished in order to ensure fully functionality of operators, streams and properties. The modified source code can be found in Appendix H.

As a result of these efforts, a graphic editor was built on a new, platform which runs on the public domain Linux O.S. In order to run the graphic editor under the Linux O.S., the following steps are required.

```
#luqipc cd /graph

#luqipc sde
```

This will bring up the graphic editor as in Figure 16.

VI. RELATED WORK

In this chapter, we briefly discuss and review some recent related work in CAPS. Our thesis includes adapting, porting and extending of some of the concepts and structures introduced in earlier works.

CAPS is a large scale system. Large systems change in relatively small increments. Each module directly or indirectly affects the main components. Maintaining the integrity and portability of such a large systems like CAPS requires a good configuration management system. The main purpose of this configuration system is to allow inter-dependencies between modules.

CAPS which has the version 1 runs on the Sun Sparc Workstations running the Sun Operating System 4.1.3. As the computer industry both software and hardware grows quickly with new research developments, portability becomes an important issue. In order to use CAPS effectively, portability of components to various platforms in a cost-effective manner became a critical issue in these efforts. To be able to use the CAPS on various platforms will lead to a substantial increase in the usability and growth of the CAPS environment. As a result of portability efforts, there also exits a CAPS version of Solaris 2.5 Sun Sparc workstations [Ref. 19]. The same re-engineering and software hierarchy principles were used.

At the level of software, the components must be reusable for portability. Components have certain properties that identify different variations of the component

called categorical properties. The implementation languages and system environments categories at the software-base as shown in Figure 17. [Ref. 9]

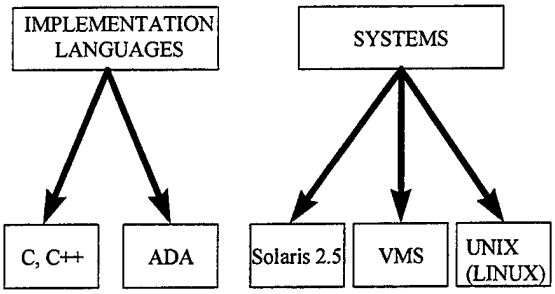

Figure 17. The Categorization from the View of Implementation Languages and System

As depicted in Figure 17, the categorization of implementation of languages in systems is directly related to the CAPS system layer model as shown in Figure 18. To increase the portability of CAPS requires us to look at other O.S.'s capable of supporting the CAPS hierarchy. Solaris 2.5, Linux O.S. and other versions of Sun O.S.'s are possible candidates.

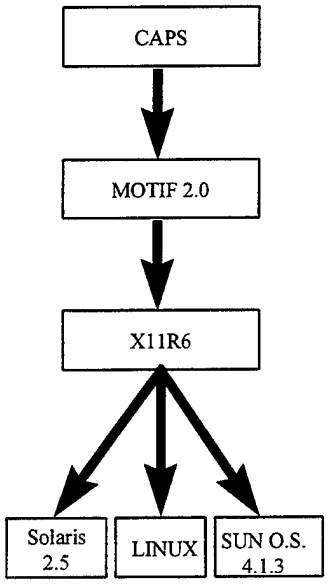

Figure 18. CAPS Layer Model

VII. FUTURE RESEARCH DEVELOPMENT AND CONCLUSION

A. FUTURE RESEARCH AND DEVELOPMENT

There is no doubt that, today, the computer industry is one of the most rapidly developing research entities in the private sector among the other industries. While computer hardware prices continue to decline, software programs used to conduct the interface between man and machine, continue to skyrocket with no end in sight. No matter how powerful the hardware you have, a main program is required in order to run the applications. This can be accomplished with a strong operating system.

Today, a new, freely distributed O.S., called Linux, is available on the Internet. The huge network power of the Internet has allowed Linux, the opportunity to be downloaded and modified by persons wishing to add their improvements and enhancements without having to worry about license and copyright laws. This allows Linux to continually grow as new enhancements are added monthly and sometimes even weekly. Sometimes these enhancements improve Linux's use of other applications while other times the kernel is upgraded. With each improved kernel, more device drivers are made available. For the future development of Linux, it is necessary to keep up with the latest version of this free O.S. Newsgroups and the ftp sites are available source of information on the latest developments. In addition, help and assistance are also available at these sites if needed.

The goal of this thesis is to port Computer Aided Prototype System (CAPS) components into the Linux O.S. and use a similar software hierarchy as the one used on the UNIX Sun workstations at the Naval Postgraduate School.

Currently, CAPS is running under the Sun operating system. The current version of CAPS is built atop of Xwindows, Motif 2.0 and TAE+. Due to the budget constraints TAE+ source code was not purchased. If more research funds can be obtained it may be possible to obtain the TAE+ original source code and compile under the gcc on Linux O.S. This is one source of possible future research and development which may be taken advantage of.

According to our research, the company administrator that owns TAE+ has stated that a move to make TAE+ available to the Linux O.S. has been considered. If this can be done, future researchers in this area can use the pre compiled binaries to install TAE+ on this current Linux platform. Information concerning TAE+ can be found at

http://www. tae+. com

This thesis addresses the need and portability requirements of CAPS components and builds the fundamental hierarchy for CAPS. Future researchers will eventually need to port the second part of CAPS, which is the SDE, to the Linux O.S. The source code is available in the "luqipc" system.

In summary, we can list the areas of potential future development and research.

1. Ada 95 Compiler Integrity

CAPS Release 1 produces Ada83 code and uses the Ada 83 compiler. Potential research areas for using the Ada 95 compiler can be considered. Future researchers should

select an Ada 95 compiler and attempt to use this compiler in their interface. If successful, Ada 95 enhanced capabilities over Ada 83 could be available to users which could take advantage of this functionality and features. One potential problem can be the inconsistency between the Ada 95 compiler and the CAPS interface tools. In addition, the software engineer database would have to be updated with Ada 95 code vice Ada 83 code.

2. CAPS on a Intel X86 Architecture

There is currently an available version of the Solaris 2.5 O.S. running on the Intel X86 platform. This X86 platform is a portable computer notebook. The future work may involve porting caps components to this platform by using the same methods and principles as described in [Ref. 16].

3. New Platforms

In the rapid changing world of computer industry, new platforms and new kernel upgrades for these platforms need to be investigated as possible new platforms to support CAPS. These possible environments can include, but are not limited to keeping up with the latest versions of Linux. This can be done by staying in contact with Linux newsgroup, linux. announcements.

In addition, Solaris 2.5.1, CDE, and the latest versions of Windows and Motif 2.0 can be upgraded.

New and improved ideas of computing are continually being discovered. One must stay in touch with these new discoveries in order to see if CAPS can exploit these areas and

therefore increase its widespread use as a software engineering tool that can greatly improve the effectiveness of engineering new software.

B. CONCLUSION

In conclusion, a summary of the accomplishments that were made follows:

- Determined that the CAPS environment is a very cost-effective and efficient tool in software engineering tool.
- Examined the current state of CAPS environment and its platform hierarchy.
- Researched various portability issues to determine the type of platform required to significantly increase access to the CAPS environment by users.
- Determined that an increase in user access to CAPS could be accomplished by porting it to an environment which is free or, at the very least, much less expensive than the current UNIX version.
- Determined that the Linux O.S. accomplished above step.
- Determined that the hierarchy of our platform would have to forego the TAE+ component, due to budget constraints.
- Built the new platform hierarchy using Linux, Xwindows, and Motif.
- Compiled and debugged the graph editor until properly configured.
- Suggested similar areas that could be further researched and discussed.
- Outlined similar accomplishments related to this thesis research topic

APPENDIX A. SOME LINUX MAIL/E-MAIL DISTRIBUTORS IN U.S.A

1. Infomagic Inc
   11950 N Highway 89
   Flagstaff , AZ 86004
   http;//www.infomagic.com
   e mail : orders@infomagic.com

2. Red hat Software
   3203 Yorktown Ave. Suite 123
   Durham , NC 27713
   http://www.redhat.com

3. Walnut Greek CD-ROM
   4041 Pike lane , Ste D
   Concord, CA 94520
   http://www.cdrom.com

4. Craftswork Linux Intel
   http://www.craftswork.com/products
   e mail: info@clbooks.com

5. Yggdrasil Computing Inc
   4880 Stevens Greek Blvd Suite 205
   San Jose , CA 95129 -1034
   http://www.yggdrasil.com

6. Linux Journal
   http://www.ssc.com/lj/

7. Fintonic Linux Systems
   1360 Willow Rd Suite 205
   Menlo Park CA 94025
   e-mail : linux@fintronic.com

8. Cheap Bytes
   P.O. Box 2714
   Lodi, CA 95241
   http://www.cheapbytes.com
   e mail : sales@cheapbytes.com

9. Linux Systems Labs
   18300 Tara Drive
   Clinton Twp, MI 48036
   e mail : info@lsl.com
   http://www.lsl.com

APPENDIX B. SOME FTP SITES FOR LINUX O.S.

Site Name	IP Address	Directory
sunsite.unc.edu	152.2.22.81	/pub/linux
ftp.funet.fi	128.214.248.6	/pub/linux
ftp.cc.gatech.edu	130.207.9.21	/pub/linux
ftp.redhat.com	199.183.24.251	/pub
ftp.cdrom.com	165.113.58.253	/pub/linux
ftp.debian.org	130.207.7.21	/pub/linux
ftp.io.org	198.133.36.5	/pub/systems/linux
ftp.sun.ac.za	146.232.212.21	/pub/linux
ftp.is.co.za	196.4.160.12	/linux
prep.ai.mit.edu	18.159.0.42	/pub/gnu
tsx-11.mit.edu	18.86.0.44	/pub/linux
kirk.bond.edu.au	131.244.1.1	/pub/OS/linux
ftp.win.tue.nl	131.155.70.100	/pub/linux
ftp.denet.dk	129.142.6.74	/pub/OS/linux
ftp.ibp.fr	132.277.60.2	/pub/linux
ftp.infomagic.com	165.113.211.2	/pub/linux/sunsite
ftp.wit.com	128.104.30.210	/systems/unix/linux
ftp.yggdrasil.com	192.244.216.52	/mirrors/linux/sunsite
ftp.rge.com	157.255.178.12	/pub/linux/sunsite
ftp.metu.edu.tr	144.122.199.41	/pub/linux/sunsite
ftp.idiscover.co.uk	194.207.26.011	/pub/Linux/sunsite
ftp.uu.net	137.39.1.9	/systems/unix/linux

APPENDIX C. LUQIPC SYSTEM SPECIFICATIONS

System Specifications of the "Luqipc" in CAPS Project	
Motherboard	Pentium 100 w/32Mb RAM, 512Kb Cache
Controller	Onboard EIDE and Floppy
Video Card	ATI Mach 8 DRAM 1024 Kb
Monitor	Unisys
	Horizontal Sync 31.5 MHz.
	Vertical Sync 76 Hz
Sound Card	Creative S. B 16 Bit w/ speaker
Mouse	Microsoft Compatible, 3 Button generic
Hard Drive	WD 2.1 GB
CD-ROM	Creative 4x speed IDE
Network	3com Ethernet card
Standart 1.44 Mb 3.5 Floppy and Keyboard	

APPENDIX D. REPARTITIONING HARD DRIVE

For repartitioning the hard drive, we begin to use the "n" command to create a new partition

Command (m for help): n
Command action
e extended
p primary partition (I - 4)
p

Here we're being asked if we want to create an extended or primary partition. In most cases you want to use primary partitions, unless you need more than four partitions on a drive.

Partition number (I - 4) : 1
First cylinder (1 - 264): 1
Last cylinder or + sizem or +sizek ([ll - 264): 264

Command (m for help): p
Disk /dev/hda : 255 heads, 63 sectors, 264 cylinders
Units = cylinders of 16065 * 512 bytes
Device Boot Begin Start End Blocks Id System
/dev/hdal

I 1 264 2120548 83 Linux native

To quit fdisk and save the changes to the partition table, use the "w" con-tmand. fdisk WITHOUT saving changes, use the "q" conunand.

Command **(m** for help): w

fdisk /dev/hda

Command (m for help): p
Disk /dev/hda: 16 heads, 63 sectors, 1048 cylinders
Units = cylinders of 1008 * 512 bytes
Device Boot Begin Start End Blocks Id System
Command (m for help):

Next we use the "n" command to create a new partition

Command (m for help): n
Command action
e extended
p primary partition (I - 4)
p

Here we're being asked if we want to create an extended or primary partition. In most cases you want to use primary partitions, unless you need more than four partitions on a drive.

Partition number (I - 4) : I
First cylinder (1 - 1048): 1
Last cylinder or + sizem or +sizek (I - 1048): 508

Command (m for help): p
Disk /dev/hda : 16 heads, 63 sectors, 1048 cylinders
Units = cylinders of 1008 * 512 bytes
Device Boot Begin Start End Blocks **Id** System
/dev/hdal

I 1 508 256000 83 Linux native

Note that the DOS partition (here, /dev/hdal) has type "Linux native". We need to change the type of the swap partition to "MS - DOS" so that the installation program will recognize it as such. In order to do this, us the fdisk "t" conimand:

Command (m for help): t
Partition number (1 - 4): 1
Hex code (type L to list codes): 6

If you use "L" to list the type codes, you'll find that 6 is the type corresponding to MS-DOS.

Command **(m** for help): **p**
Disk /dev/hda : 16 heads, 63 sectors, 1048 cylinders
Units = cylinders of 1008 * 512 bytes
Device Boot Begin Start End Blocks **Id** System
/dev/hdal

I 1 508 256000 6 DOS 16 bit >= 32M

Next we use the "n" command to create a new partition. Linux does not recognize swap partitions more than 127.5 Mbyte, because of this we are going to create 3 different swap partitions.

Command (m for help): n
Command action
e extended
p primary partition (2 - 4)

p

Partition number (I - 4) : 2
First cylinder (509 - 1048): 509
Last cylinder or + sizem or +sizek (509 - 1048): 691

Command (m for help): p
Disk /dev/hda : 16 heads, 63 sectors, 1048 cylinders
Units = cylinders of 1008 * 512 bytes
Device Boot Begin Start End Blocks Id System

/dev/hdal 1 1 508 256000 6 DOS 16 bit >= 32M
/dev/hda2 509 509 691 92232 83 Linux native

Note that the Linux swap partition (here, /dev/hda32 has type "Linux native". We need to change the type of the swap partition to "Linux swap" so that the installation program will recognize it as such. In order to do this, us the fdisk "t" command:
Command (m for help): t
Partition number (I - 4): 2
Hex code (type L to list codes): 82

If you use "L" to list the type codes, you'll find that 82 is the type corresponding to Linux swap.
Command (m for help): n
Command action
e **extended**
p primary partition (1 - 4)

p

Partition number (1 - 4) : *3*
First cylinder (692 - 1048): 692
Last cylinder or + sizem or +sizek (509 - 1048): 874

Command (m for help): t
Partition number (I - 4): 3
Hex code (type L to list codes): 82

Command (m for help): n
Command action
e extended
p primary partition (I - 4)
p

Partition number (I - 4) : 4
First cylinder (875 - 1048): 875

Last cylinder or + sizem or +sizek (875 - 1048): 1048 /* luqipc
1050

Command (m for help): t
Partition number (1 - 4): 4
Hex code (type L to list codes): 82

Command (m for help): p
Disk /dev/hda : 16 heads, 63 sectors, 1048 cylinders
Units = cylinders of 1008 * 512 bytes

Device	Boot	Begin	Start	End	Blocks	Id	System
/dev/hdal		1	1	508	256000	6	DOS 16 bit >= 32M
/dev/hda2		509	509	691	92232	82	Linux swap
/dev/hda3		692	692	874	92232	82	Linux swap
/dev/hda4		875	875	1048	87696	82	Linux swap #luqipc

1050 88704

Command (m for help): w

We need to reboot the computer to be able to use new partition.

APPENDIX E. SOME MOTIF MAIL/E-MAIL DISTRIBUTORS

1. Infomagic Inc
   11950 N Highway 89
   Flagstaff , AZ 86004
   http;//www.infomagic.com
   e mail : orders@infomagic.com

2. Red hat Software
   3203 Yorktown Ave. Suite 123
   Durham , NC 27713
   http://www.redhat.com

3. Metro Link Incorporated
   4711 North Powerline Road
   Fort Lauderdale, FL 33309
   Email: holly@metrolink.com

4. X Inside Incorporated
   1801 Broadway, Suite 1710
   Denver, CO 80202
   Email: sales@xinside.com

5. Lasermoon Ltd.
   The Forge, Fareham Road
   Wickham, Hants, England PO17 5DE
   email orders@lasermoon.co.uk

APPENDIX F. SOME COMPATIBLE SVGA CHIPSETS

1. Tseng ET3000, ET4000AX, ET4000/W32
2. Western Digital/Paradise PVGA1
3. Western Digital WD90C00, WD90C10, WD90C11, WD90C24, WD90C30, WDC90C31, WD90C33
4. Genova GVGA
5. Trident TVGA8800CS, TVGA8900B, TVGA8900C, TVGA8900CL, TVGA9000, TVGA9000i , TVGA9100B, TVGA9200CX, TVGA9320, TVGA9400CX, TVGA9420
6. ATI 1800, 18800-1, 28800-2, 28800-4, 2880-5, 28800-6, 68800-3, 68800-6, 68800AX, 68800LX, 88800
7. NCR 77C22, 77C22E, 77C22E+
8. Cirrus Logic , CLGD5420, CLGD5422, CLGD5424, CLGD5426, CLGD5428, CLGD5429, CLGD5430, CLGD5434, CLGD6205, CLGD6215, CLGD6225
9. Compaq AVGA
10. OAK OTI067, OTI077
11. Advance Logic AL2101
12. ATI Mach8, Mach12
13. Western Digital WD90C31, WD90C33
14. Tseng Et4000/W32, ET4000/W32I, ET4000/W32p

APPENDIX G. MAKE FILE

```
# edit_graph Makefile
#
#   sde represents a C program, not a C++ program.  However, no
#   provisions have been made here to use the C compiler.  But this
#   should not present a problem since the syntax-directed editor
#   program that is provided should be compatible with the C++ compiler.
# DEBUG is currently turned OFF
# sde   : makes executable
# clean : cleans up *.o, *~ and sde
.SUFFIXES: .o .C .c .h .H
#CC = CC
#LIBS = -lXm -lXt -lXext -lX11 -lm
#DEBUG = -g
CC = g++
LIBS = -L/usr/X11R6/lib -lXm -lXpm -lXt -lXext -lX11 -lm -g++ -gcc
DEBUG = -g -DGE_DEBUG
C++FLAGS = $(DEBUG) -DFUNCPROTO -DXTFUNCPROTO
CFLAGS = $(DEBUG) -D_NO_PROTO
all: sde
.c.o:
        cc $(CFLAGS) -c $*.c
GOBJECTS= sde.o graph_editor.o operator_object.o stream_object.o spline_object.o
graph_object_list.o font_table.o graph_object.o stream_property_menu.o
operator_property_menu.o ge_utilities.o llist.o windows.o
sde: $(GOBJECTS)
        $(CC)  -g -o sde $(GOBJECTS) $(LIBS)
sde.o: sde.c ge_interface.h graph_editor.h resources.h ge_defs.h
        $(CC) $(C++FLAGS) -c sde.c
graph_editor.o: graph_editor.h graph_editor.C spline_object.H operator_object.H
stream_object.H graph_object.H graph_object_list.H graph_editor.C ge_defs.h resources.h
ge_interface.h font_table.H stream_property_menu.H operator_property_menu.H
ge_utilities.H windows.H
        $(CC) $(C++FLAGS) -c graph_editor.C
operator_object.o: operator_object.C operator_object.H graph_object.H ge_defs.h
resources.h ge_interface.h font_table.H graph_object_list.H ge_utilities.H llist.H
        $(CC) $(C++FLAGS) -c operator_object.C
```

```
stream_object.o: stream_object.C stream_object.H spline_object.H graph_object.H
ge_defs.h resources.h ge_interface.h operator_object.H graph_object_list.H font_table.H
ge_utilities.H
	$(CC) $(C++FLAGS) -c stream_object.C

stream_property_menu.o:	stream_property_menu.C stream_property_menu.H
ge_defs.h graph_object_list.H graph_object.H stream_object.H graph_object.H
ge_interface.h
	$(CC) $(C++FLAGS) -c stream_property_menu.C

operator_property_menu.o: operator_property_menu.C stream_property_menu.H ge_defs.h
graph_object_list.H graph_object.H operator_object.H graph_object.H stream_object.H
ge_interface.h mini_sde.C
	$(CC) $(C++FLAGS) -c operator_property_menu.C


spline_object.o: spline_object.C spline_object.H graph_object.H ge_defs.h resources.h
stream_object.H operator_object.H font_table.H ge_interface.h
	$(CC) $(C++FLAGS) -c spline_object.C

graph_object.o: graph_object.C graph_object.H font_table.H ge_defs.h resources.h
	$(CC) $(C++FLAGS) -c graph_object.C


graph_object_list.o: graph_object_list.C graph_object_list.H graph_object.H
operator_object.H stream_object.H ge_defs.h resources.h font_table.H llist.H ge_interface.h
ge_utilities.H
	$(CC) $(C++FLAGS) -c graph_object_list.C

font_table.o: font_table.C font_table.H ge_defs.h resources.h
	$(CC) $(C++FLAGS) -c font_table.C

ge_utilities.o: ge_utilities.C ge_utilities.H ge_interface.h
	$(CC) $(C++FLAGS) -c ge_utilities.C

llist.o: llist.C llist.H ge_defs.h ge_interface.h ge_utilities.H
	$(CC) $(C++FLAGS) -c llist.C

mini_sde.o: mini_sde.C
	$(CC) $(C++FLAGS) -c mini_sde.C

windows.o: windows.C windows.H
```

```
	$(CC) $(C++FLAGS) -c windows.C


clean:
	rm -f *.o *~*
	rm -f sde
```

APPENDIX H. CAPS SOURCE

```
/* ********************************************************

Name:        font_table.h
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 12 Sep 92
Remarks:  Specification for the FontTable object.

      The FontTable contains all the necessary information
     about the fonts used in the graph editor, allowing
     the program to refer to them by a font id#.

      It initializes with six predefined font names,
     although more could easily be added.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
     Changed file to a .H from .h for C++.

******************************************************** */

#ifndef FONT_TABLE_H
#define FONT_TABLE_H 1

#include <stdlib.h>
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <Xm/MessageB.h>
#include <string.h>
#include "ge_defs.h"

struct _font_record {
  char *_name_ptr;
  XFontStruct  *_font_ptr;
  int _height;
};

class FontTable {
private:
  struct _font_record _font_table[MAXFONTS + 1];
  static Widget _error_tgt;

public:
  static void set_error_tgt(Widget widget) {_error_tgt = widget;}
  static void error_box(char *error_message);

  FontTable();
  ~FontTable();
  void init(Display *display_ptr);
  int width(int font_id, char *in_string);
  int height(int font_id);
  Font font_id(int font_id);
  char *font_name(int font_id);
};

#endif
```

```
/* ********************************************************

Name:
Author:      Capt Robert M. Dixon
Program:
Date Modified:
Remarks:  Implementation of the FontTable object.

      The FontTable contains all the necessary information
     about the fonts used in the graph editor, allowing
     the program to refer to them by a font id#.

      It initializes with six predefined font names,
     although more could easily be added.


Credits:   Portions of code are adapted from the following:

      Heller, Dan, Motif Programming Manual, O'Reilly and
       Associates, 1991.

      Johnson, Eric, and Reichard, Kevin, X Window
       Applications Programming, MIS Press, 1989.

      Young, Douglas, Object Oriented Programming With C++
       and OSF/Motif, Prentice-Hall, 1992.


********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
     Changed include file to a .H from .h for C++.

 @2   KBM    27 Aug 96
     g++ with -Wall complained about width() not having a
         return statement.

********************************************************
*/
#include <stream.h>
#include "font_table.H"          // @1

// Initializes static class variable.

Widget FontTable::_error_tgt = NULL;

// Displays the error message in a Motif error dialog box.

void FontTable::error_box(char *error_message) {
 static Widget error_dialog = NULL;
 Arg arg[1];
 XmString t;

 if(_error_tgt != NULL) {
  if(error_dialog == NULL) {
   error_dialog = XmCreateMessageDialog(_error_tgt, "error",
                                arg, 0);
   XtVaSetValues(XtParent(error_dialog),
            XtNtitle, "Error",
            NULL);
   XtUnmanageChild(XmMessageBoxGetChild(error_dialog,
                          XmDIALOG_HELP_BUTTON));
  }
  t = XmStringCreateSimple(error_message);
  XtVaSetValues(error_dialog,
           XmNmessageString, t,
           NULL);
```

```
  XmStringFree(t);
  XtManageChild(error_dialog);
 }
 else
  cout <<
   "Error Target Widget must be set by calling module.\n"
    << endl;
}

FontTable::FontTable() {
 int i;

 for(i = 1; i <= MAXFONTS; i++) {
  _font_table[i]._name_ptr = NULL;
  _font_table[i]._font_ptr = NULL;
 }
}


FontTable::~FontTable() {
 int i;

 for(i = 1; i <= MAXFONTS; i++) {
  delete _font_table[i]._name_ptr;
  delete _font_table[i]._font_ptr;
 }
}

// Initializes the font table using fonts defined for the
// given display.

void FontTable::init(Display *display_ptr) {

 _font_table[0]._name_ptr =
              strdup("variable");
_font_table[COURIERBOLD10]._name_ptr =
              strdup("*adobe-courier-bold-r*100*");
_font_table[COURIERBOLD12]._name_ptr =
              strdup("*adobe-courier-bold-r*120*");
_font_table[COURIERBOLD14]._name_ptr =
              strdup("*adobe-courier-bold-r*140*");
_font_table[COURIERMED10]._name_ptr =
              strdup("*adobe-courier-medium-r*100*");
_font_table[COURIERMED12]._name_ptr =
              strdup("*adobe-courier-medium-r*120*");
_font_table[COURIERMED14]._name_ptr =
              strdup("*adobe-courier-medium-r*140*");

_font_table[0]._font_ptr =
 XLoadQueryFont(display_ptr, _font_table[0]._name_ptr);
_font_table[COURIERBOLD10]._font_ptr =
 XLoadQueryFont(display_ptr,
          _font_table[COURIERBOLD10]._name_ptr);
_font_table[COURIERBOLD12]._font_ptr =
 XLoadQueryFont(display_ptr,
          _font_table[COURIERBOLD12]._name_ptr);
_font_table[COURIERBOLD14]._font_ptr =
 XLoadQueryFont(display_ptr,
          _font_table[COURIERBOLD14]._name_ptr);
_font_table[COURIERMED10]._font_ptr =
 XLoadQueryFont(display_ptr,
          _font_table[COURIERMED10]._name_ptr);
_font_table[COURIERMED12]._font_ptr =
 XLoadQueryFont(display_ptr,
          _font_table[COURIERMED12]._name_ptr);
_font_table[COURIERMED14]._font_ptr =
 XLoadQueryFont(display_ptr,
          _font_table[COURIERMED14]._name_ptr);

_font_table[0]._height =
```

```
  _font_table[0]._font_ptr->ascent +
   _font_table[0]._font_ptr->descent;

 _font_table[COURIERBOLD10]._height =
  _font_table[COURIERBOLD10]._font_ptr->ascent +
   _font_table[COURIERBOLD10]._font_ptr->descent;

 _font_table[COURIERBOLD12]._height =
  _font_table[COURIERBOLD12]._font_ptr->ascent +
   _font_table[COURIERBOLD12]._font_ptr->descent;

 _font_table[COURIERBOLD14]._height =
  _font_table[COURIERBOLD14]._font_ptr->ascent +
   _font_table[COURIERBOLD14]._font_ptr->descent;

 _font_table[COURIERMED10]._height =
  _font_table[COURIERMED10]._font_ptr->ascent +
   _font_table[COURIERMED10]._font_ptr->descent;

 _font_table[COURIERMED12]._height =
  _font_table[COURIERMED12]._font_ptr->ascent +
   _font_table[COURIERMED12]._font_ptr->descent;

 _font_table[COURIERMED14]._height =
  _font_table[COURIERMED14]._font_ptr->ascent +
   _font_table[COURIERMED14]._font_ptr->descent;
}

// Returns the width of the given string in the given font
// in pixels.

int FontTable::width(int font_id, char *in_string) {

 if(in_string == NULL)
  return 0;
 else
  if(font_id > MAXFONTS)
   error_box("Font Table entered with font too big.");
  else
   return XTextWidth(_font_table[font_id]._font_ptr,
          in_string, strlen(in_string));
 return 0;                                          //@2
}

// Returns the height of the given font in pixels.

int FontTable::height(int font_id) {

 if(font_id <= MAXFONTS)
  return _font_table[font_id]._height;
 else {
  error_box("Font Table entered with font too big.");
  return 0;
 }
}

// Returns font information needed by some X functions.

Font FontTable::font_id(int font_id) {

 if(font_id <= MAXFONTS)
  return _font_table[font_id]._font_ptr->fid;
 else {
  error_box("Font Table entered with font too big.");
  return 0;
 }
}

// Returns the name of the given font.
```

```
char *FontTable::font_name(int font_id) {

 if(font_id <= MAXFONTS)
  return _font_table[font_id]._name_ptr;
 else {
  error_box("Font Table entered with font too big.");
  return NULL;
 }
}
```

```
/* ********************************************************

Name:        ge_defs.h
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 12 Sep 92
Remarks:  Provides the common type definitions and defines
      for the modules in the graph_editor program.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
    Converted UP back to -1 from 1 so that it is compatible with
        exiting write_to_disk() method.

********************************************************
*/
#ifndef ge_defs_h
#define ge_defs_h 1

typedef int BOOLEAN;
enum CLASS_DEF{GRAPHOBJECT, OPERATOROBJECT,
STREAMOBJECT,
        GRAPHOBJECTLIST, SPLINEOBJECT};

enum TOOL_STATE{OPERATOR_TOOL, TERMINATOR_TOOL,
STREAM_TOOL,
         MET_TOOL, LATENCY_TOOL, SELECT_TOOL};

enum DRAW_STYLE{SOLID, DOTTED, ERASE};

typedef int GE_STATUS;
typedef int EXTERN_STATUS;
typedef int COLOR;

typedef struct xypair {
 int x, y;
} XYPAIR;

#define TRUE 1
#define FALSE 0
#define ENDED 2
#define SUCCEEDED 1
#define FAILED 0
#define INPUT_LINE_SIZE 100
#define CIRCLE_BEGIN 0
#define FULL_CIRCLE 360 * 64
#define HANDLESIZE 5

/*  Return codes used to tell the syntax-directed editor
 what to do with the incoming data, and whether any
 level changes are required.
*/

/* 7/30/96  MTS                  */
/* change UP to 1 and define DOWN to 2 */
#define DOWN 2
#define UP  -1   /* @1 Put back to -1 for return to parent   KBM */
#define SAME 0

#define ERROR -3
#define NOUPDATE -4

#define NO_EXTERNAL 0
#define FROM_EXTERNAL 1
#define TO_EXTERNAL 2
```

```
#define HITFUDGE 5

#define NULL_VALUE 0
#define MAXDELETEDOPS 100
#define MAXTEXTLINES 100

#include "resources.h"
#endif
```

```
/* ********************************************************

Name:        ge_interface.h

********************************************************
*/
#ifndef GE_INTERFACE_H
#define GE_INTERFACE_H 1

#include "ge_defs.h"

#define US 0            /* microsec */
#define MS 1            /* millisec */
#define SECOND 2        /* sec */
#define MINUTE 3        /* min */
#define HOUR  4         /* hours */

#define UNDEFINED_TIME -1  /* constant representing undefined time
value */

#define UNPROTECTED  0    /* no data trigger   */
#define BY_SOME      1    /* triggered by some */
#define BY_ALL       2    /* triggered by all  */

#define NTC        0    /* non time critical */
#define PERIODIC   1    /* periodic          */
#define SPORADIC   2    /* sporadic          */


typedef int BOOL;       /* 0 means false, 1 means true */
typedef int OP_ID;      /* unique identifier numbers for
                    OPERATORs, 1 .. # ops */
typedef int ST_ID;      /* unique identifier numbers for
                    streams, 1 .. # streams */
```

```
/*********************************************************
*/
/* typdef for an ID linked list                          */
/*********************************************************
*/
typedef struct id_list {
  char* ID;
  struct id_list* NEXT;
}ID_NODE, *ID_LIST;


/*********************************************************
*/
/* typedef for an OPERATOR                               */
/*********************************************************
*/
typedef struct op_str {
  OP_ID id;                 /* cannot be changed after creation */

  /* info about location of operator icon */
  int
      x,                    /* x_position of the circle's center point */
      y,                    /* y_position of the circle's center point */
      radius,
      color;

  /* info about the label */
  char     *label;
  int
      label_font,
      label_x_offset,       /* actual x_position = x + label_x_offset */
      label_y_offset;       /* actual y_position = y + label_y_offset */
```

```
/* info about timing constraints */
int
    timing_type;        /* NTC, PERIOD, SPORADIC     */

/* info about MET */
int
    met,              /* UNDEFINED_TIME if no met */
    met_unit,          /* US, MS, SECOND, MINUTE, HOUR */
    met_font,
    met_x_offset,      /* actual x_position = x + met_x_offset */
    met_y_offset;      /* actual y_position = y + met_y_offset */
ID_LIST
    met_reqmts;        /* requirements trace        */

/* info about PERIOD */
int
    period,           /* UNDEFINED_TIME if no period */
    period_unit;       /* US, MS, SECOND, MINUTE, HOUR */
ID_LIST
    period_reqmts;     /* requirements trace        */

/* info about FINISH WITHIN */
int
    fw,               /* UNDEFINED_TIME if no finish within */
    fw_unit;           /* US, MS, SECOND, MINUTE, HOUR */
ID_LIST
    fw_reqmts;         /* requirements trace        */

/* info about MAXIMUM RESPONSE TIME */
int
    mrt,              /* UNDEFINED_TIME if no maximum response
time */
    mrt_unit;          /* US, MS, SECOND, MINUTE, HOUR */
ID_LIST
    mrt_reqmts;        /* requirements trace        */
```



```
/* info about MINIMUM CALLING PERIOD */
int
    mcp,              /* UNDEFINED_TIME if no minimum calling
period */
    mcp_unit;          /* US, MS, SECOND, MINUTE, HOUR */
ID_LIST
    mcp_reqmts;        /* requirements trace        */

/* info about triggering */
int
    trigger_type;      /* UNPROTECTED, BY_SOME, BY_ALL */
char*
    if_condition;      /* triggered_if condition, set to TRUE
                       if not specified,
                       parameter to be forwarded to mini-sde
                       EDIT_IF_COND(char* if_condition)    */
ID_LIST
    trigger_reqmts;    /* requirements trace        */

/* info about output guard */
char*
    output_guard_list;   /* parameter to be forwarded to mini-sde
                       EDIT_OUTPUT_GUARD(char* out_guard_list) */

/* info about exception */
char*
    exception_list;    /* parameter to be forwarded to mini-sde
                       EDIT_EXCEPTION(char* exception_list) */

/* info about timer op */
char*
    timer_op_list;     /* parameter to be forwarded to mini-sde
                       EDIT_TIMER_OP(char* timer_op_list) */
/* info about key_words for operator spec */
```

```
ID_LIST
    key_word_list;      /* list of keywords f*/

/* info about informal description operator spec */
char*
    operator_informal_desc;

/* info about formal description operator spec */
char*
    operator_formal_desc;

/* operator properties */
BOOL
    is_composite,
    is_terminator;

/* info for house keeping */
BOOL
    is_deleted,
    is_new,     /* is_new is set by the GE and
                        cleared by the SDE' New_Operator procedure
*/
    is_modified;

}OP_NODE, *OPERATOR;


/*********************************************************
*/
/* typedef for a linked list of OPERATOR                */
/*********************************************************
*/
typedef struct op_list_node {
 OPERATOR op;
 struct op_list_node *next;
}OP_LIST_NODE, *OP_LIST;


/*********************************************************
*/
/* typedef for SPLINE linked list                       */
/*********************************************************
*/
typedef struct spline_node {
 int
    x,          /* each (x,y) represent an interval control point */
    y;          /* for the spline, not including end points.     */
 struct spline_node*
    next;
}SPLINE_NODE, *SPLINE_PTR;


/*********************************************************
*/
/* typedef for a STREAM                                 */
/*********************************************************
*/
typedef struct st_str{
 ST_ID id;      /* cannot be changed after creation */

 /* info about label */
 char *label;
 int
    label_font,
    label_x_offset,  /* actual x_position = x_mid_point + label_x_offset
*/
    label_y_offset;  /* actual Y_position = y_mid_point + label_y_offset
*/
                /* x_mid_point = 0.5 * ( from->x + to->x)       */
                /* y_mid_point = 0.5 * ( from->y + to->y)       */
```

```
 /* the two endpoints of the stream */
 OPERATOR
     from,
     to;

 /* linked list of SPLINE_NODE defining the shape of the edge on the
display */
 SPLINE_PTR
     arc;            /* null pointer if the arc is a
                        straight line */


 /* info about LATENCY */
 int
     latency,              /* UNDEFINED_TIME if not specified */
     latency_unit,           /* US, MS, SECOND, MINUTE, HOUR */
     latency_font,
     latency_x_offset, /* actual x_position = x_mid_point +
latency_x_offset */
     latency_y_offset; /* actual Y_position = y_mid_point +
latency_y_offset */

 /* stream type */
 char*
     stream_type_name;

 /* initial value if it is a state stream */
 char*
     state_initial_value;

 /* stream visible properties */
 BOOL
     is_state_variable;

 /* info for house keeping */
 BOOL
     is_new,         /* is_new is set by the GE and
                        cleared by the SDE' Make_Edge_List
                                function */
     is_modified,
     is_deleted;

}ST_NODE, *STREAM;


/*********************************************************
*/
/*typedef for a linked list of STREAM                    */
/*********************************************************
*/
typedef struct st_list_node {
  STREAM st;
  struct st_list_node *next;
}ST_LIST_NODE, *ST_LIST;


/*********************************************************
*/
/* typedef for the graph description structure           */
/*********************************************************
*/
typedef struct graph_desc_node{
  char* edited_op_name;      /* name of the current operator being edited
*/
  char* parent_op_name;      /* name of the parent of the current
operator */
  char* edited_op_spec;      /* PSDL specification of the current operator
*/
  OP_LIST operator_list;
```

```
 ST_LIST stream_list;
 ID_LIST timer_list;
 char*   graph_informal_desc;
}GRAPH_DESC_NODE, *GRAPH_DESC;


/*********************************************************
*/
/* typedef for the ACTION type                           */
/*********************************************************
*/
typedef struct action_node {
 BOOL save;              /* true if need to save change when exit */
 BOOL reinvoke;            /* true if need to reopen GE          */
 char*  next_op;         /* name of the next operator to be edited,
                         only meaningful if reinvoke == TRUE   */
} ACTION_NODE, *ACTION;

#endif
```

```
/* ******************************************************

Name:        ge_utilities.H
Author:      Ken Moeller
Program:     graph_editor
Date Modified: 5 Sep 96
Remarks:     This file contains several utility functions for
          working with ge_interface.  While ge_interface
               is a c program, this utility uses references in
               function calls and is thus a C++ program.

******************************************************
Modification History:

 @1   KBM      5 Sep 96
     New file.

******************************************************
*/

#ifndef GE_UTILITIES_H
#define GE_UTILITIES_H 1

#include <stddef.h>
#include <stdio.h>
#include <malloc.h>
#include <string.h>
#include "ge_interface.h"


char* time_unit_str(int time, int unit);
char* make_str(char* ptr);

void initOP(OP_LIST OP);
void initST(ST_LIST ST);
OPERATOR operator_id_ptr(OP_LIST ptyr, int id);
void clear_ID_LIST(ID_LIST &ptr);
void operator_delete(OP_LIST &ptr);
void spline_delete(SPLINE_PTR &ptr);
void stream_delete(ST_LIST &ptr);
void graph_desc_clear(GRAPH_DESC desc);

#endif
/* ******************************************************

Name:        ge_utilities.C
Author:      Ken Moeller
Program:     graph_editor
Date Modified: 5 Sep 96
Remarks:     This file contains several utility functions for
          working with ge_interface.  While ge_interface
               is a c program, this utility uses references in
               function calls and is thus a C++ program.

******************************************************
Modification History:

 @1   KBM      5 Sep 96
     New file.

******************************************************
*/

#include "ge_utilities.H"



/*******************************************************
 * time_unit_str(int time, int unit)
 *
```

```
 * This routine returns a pointer to a string which contains the time
 * and units based on the input parameters.
 *******************************************************/
char* time_unit_str(int time, int unit) {
 char buffer[INPUT_LINE_SIZE];

 if (time == UNDEFINED_TIME)
  return strdup("");

 switch(unit) {
  case US:
   sprintf(buffer, "%d us", time);
   break;
  case MS:
   sprintf(buffer, "%d ms", time);
   break;
  case SECOND:
   sprintf(buffer, "%d sec", time);
   break;
  case MINUTE:
   sprintf(buffer, "%d min", time);
   break;
  case HOUR:
   sprintf(buffer, "%d hr", time);
   break;
 }
 return strdup(buffer);
}



/*******************************************************
 * make_str(char* ptr)
 *
 * This routine uses strdup to make a copy of the string located at ptr.
 * However, this routine will return NULL if ptr is NULL.
 *******************************************************/
char* make_str(char* ptr) {

 if (ptr == NULL)
  return NULL;
 else {
  return strdup(ptr);
 }
}



/*******************************************************
 * initOP(OP_LIST OP)
 *
 * This routine initializes the pointers of an OP_LIST to NULL.
 * N.B.  This routine does NOT recover any dynamic memory allocated
in
 * the OP_LIST.  It is assumed that this routine is called after a
 * malloc is used to allocate an OP_LIST.
 *******************************************************/
void initOP(OP_LIST OP) {

 OP->op->id              = 0;
 OP->op->label           = NULL;
 OP->op->met_reqmts      = NULL;
 OP->op->period_reqmts   = NULL;
 OP->op->fw_reqmts       = NULL;
 OP->op->mrt_reqmts      = NULL;
 OP->op->mcp_reqmts      = NULL;
 OP->op->if_condition    = NULL;
 OP->op->trigger_reqmts  = NULL;
 OP->op->output_guard_list  = NULL;
 OP->op->exception_list  = NULL;
```

```
 OP->op->timer_op_list        = NULL;
 OP->op->key_word_list        = NULL;
 OP->op->operator_informal_desc = NULL;
 OP->op->operator_formal_desc  = NULL;

 OP->next = NULL;
}



/*********************************************************
 * initST(ST_LIST ST)
 *
 * This routine initializes the pointers of an ST_LIST to NULL.
 * N.B.  This routine does NOT recover any dynamic memory allocated
in
 * the ST_LIST.  It is assumed that this routine is called after a
 * malloc is used to allocate an ST_LIST.
 *********************************************************/
void initST(ST_LIST ST)  {

 ST->st->id              = 0;
 ST->st->label            = NULL;
 ST->st->from             = NULL;
 ST->st->to              = NULL;
 ST->st->arc              = NULL;
 ST->st->stream_type_name   = NULL;
 ST->st->state_initial_value = NULL;

 ST->next = NULL;
}



/*********************************************************
 * operator_id_ptr(OP_LIST ptr, int id)
 *
 * Given an OP_LIST and an ID, this routine will search the OP_LIST
for
 * the first OPERATOR with a matching ID and return a pointer to that
 * OPERATOR.  If a matching ID cannot be found in OP_LIST, NULL
is
 * returned.
 *********************************************************/
OPERATOR operator_id_ptr(OP_LIST ptr, int id) {

 while (ptr != NULL) {
  if (ptr->op->id == id)
   return ptr->op;
  ptr = ptr->next;
 }
 return NULL;
}



/*********************************************************
 * clear_ID_LIST(ID_LIST &ptr)
 *
 * This routine will traverse an ID_LIST an free all allocated memory.
 * Upon exit, ptr is assigned the value NULL.
 *********************************************************/
void clear_ID_LIST(ID_LIST &ptr) {
 ID_LIST delNext;

 while (ptr != NULL) {
  delNext = ptr->NEXT;
  free(ptr->ID);
  free((char*)ptr);
  ptr = delNext;
```

```
 }
}



/*********************************************************
 * operator_delete(OP_LIST &ptr)
 *
 * This routine deletes each OPERATOR in an OP_LIST in turn,
recovering
 * all allocated memory.  ptr is returned as NULL.
 *********************************************************/
void operator_delete(OP_LIST &ptr) {
 OP_LIST delNext;

 while (ptr != NULL) {
  delNext = ptr->next;

  free(ptr->op->label);
  free(ptr->op->if_condition);
  free(ptr->op->output_guard_list);
  free(ptr->op->exception_list);
  free(ptr->op->timer_op_list);
  free(ptr->op->operator_informal_desc);
  free(ptr->op->operator_formal_desc);
  clear_ID_LIST(ptr->op->met_reqmts);
  clear_ID_LIST(ptr->op->period_reqmts);
  clear_ID_LIST(ptr->op->fw_reqmts);
  clear_ID_LIST(ptr->op->mrt_reqmts);
  clear_ID_LIST(ptr->op->mcp_reqmts);
  clear_ID_LIST(ptr->op->trigger_reqmts);
  clear_ID_LIST(ptr->op->key_word_list);

  free((char*) ptr->op);
  free((char*) ptr);
  ptr = delNext;
 }
}



/*********************************************************
 * spline_delete(SPLINE_PTR &ptr)
 *
 * This routine deletes each SPLINE_NODE in an linked list pointed to
by
 * SPLINE_PTR.  Allocated memory is recovered.  ptr is returned as
NULL.
 *********************************************************/
void spline_delete(SPLINE_PTR &ptr) {
 SPLINE_PTR delNext;

 while (ptr != NULL) {
  delNext = ptr->next;
  free((char*) ptr);
  ptr = delNext;
 }
}



/*********************************************************
 * stream_delete(ST_LIST & ptr
 *
 * This routine deletes each STREAM in an ST_LIST in turn, recovering
 * all allocated memory.  ptr is returned as NULL.
 *********************************************************/
void stream_delete(ST_LIST &ptr) {
 ST_LIST delNext;
```

```
 while (ptr != NULL) {
  delNext = ptr->next;

  free(ptr->st->label);
  free(ptr->st->stream_type_name);
  free(ptr->st->state_initial_value);
  spline_delete(ptr->st->arc);

  free((char*) ptr->st);
  free((char*) ptr);
  ptr = delNext;
 }
}



/*********************************************************
 * graph_desc_clear(GRAPH_DESC desc)
 *
 * This routine clears the contents of a GRAPH_DESC_NODE,
recovering all
 * allocated memory.  However, the GRAPH_DESC_NODE is kept in
memory.
 * All pointers out of the GRAPH_DESC_NODE are set to NULL.
 *********************************************************/
void graph_desc_clear(GRAPH_DESC desc) {
 OP_LIST op_ptr, delNextOP;
 ST_LIST st_ptr, delNextST;

 /* First recover all char* */
 free(desc->edited_op_name);     desc->edited_op_name     = NULL;
 free(desc->parent_op_name);     desc->parent_op_name     = NULL;
 free(desc->edited_op_spec);     desc->edited_op_spec     = NULL;
 free(desc->graph_informal_desc); desc->graph_informal_desc =
NULL;

 /* Next the operators */
 op_ptr = desc->operator_list;
 while (op_ptr != NULL) {
  delNextOP = op_ptr->next;
  operator_delete(op_ptr);
  op_ptr = delNextOP;
 }
 desc->operator_list = NULL;

 /* Then the streams */
 st_ptr = desc->stream_list;
 while (st_ptr != NULL) {
  delNextST = st_ptr->next;
  stream_delete(st_ptr);
  st_ptr = delNextST;
 }
 desc->stream_list = NULL;

 /* Last, the timer list */
 clear_ID_LIST(desc->timer_list);
 desc->timer_list = NULL;
}
```

```
/* ********************************************************

Name:        graph_editor.h
Author:      Ken Moeller
Program:     graph_editor
Date Modified: 15 Aug 96
Remarks:  Include file for sde to obtain procedure definition
      based on the new calling method.

          Several global symbols are defined.  These are used
      since the syntax-directed editor will call edit_graph
      multiple times in a session, however, we want edit_graph
      to remember data from call to call.

********************************************************
Modification History:

 Baseline taken from Ken Moeller

 @1   KBM    26 Aug 96
     Moved next_action_ptr and current_graph_ptr to .C file
          since g++ expected them to be declared extern.

 @2   KBM    11 Sep 96
     Moved psdl_modified into GraphObjectList.
********************************************************
*/
#ifndef GRAPH_EDITOR_H
#define GRAPH_EDITOR_H 1

#include "ge_interface.h"

static BOOL     return_sde_flag;   // flag to break out of motif loop
static BOOL     motif_initialized; // only do motif init once

int edit_graph(
               /* in out parameter */
               GRAPH_DESC current_graph,

               /* out parameter */
               ACTION next_action);

#endif
/********************************************************

Name:        graph_editor.C
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 21 Sep 92
Remarks:  graph_editor.C is the main program for the
      CAPS 'xx graph editor.  Depending on the command line
      parameters, it allows either viewing only or full
      editing of a graph passed by the CAPS '93 syntax-
      directed editor.

       graph_editor accepts one command line
      parameter.  Invoked with the -v parameter, it runs in
      view mode, allowing only viewing of the graph.
      Otherwise, it allows full screen editing.

       When invoked, graph_editor looks for an input
      text file specifying the attributes of the graph.
      This file is currently named 'gedatatransfile.txt'.
      If this file is not found the editor uses a blank
      canvas.

       Unless specified otherwise, on termination
      the editor writes the attributes of the current
      graph in an output text file currently named
      'gedatatransfile2.txt'.
```

General Comments:

The XmProcessTraversal function is called numerous places in an attempt to keep the keyboard input focus in the drawing window. This allows the editor to respond to the delete and backspace key. This works with varying degrees of success.

Credits: Portions of code are adapted from the following:

Barakati, Naba, X Window System Programming, SAMS, 1991.

Heller, Dan, Motif Programming Manual, O'Reilly and Associates, 1991.

m

Johnson, Eric, and Reichard, Kevin, X Window Applications Programming, MIS Press, 1989.

Young, Douglas, Object Oriented Programming With C++ and OSF/Motif, Prentice-Hall, 1992.

****

Modification History:

Baseline taken from Robert M. Dixon

@1 KBM 15 Aug 96
Changed include file to a .H from .h for C++.

@2 KBM 15 Aug 96
Made many of the Motif symbols global so that the information is available over several calls to edit_graph. Motif initialization is only performed once.

@3 KBM 15 Aug 96
Methods renamed: write_to_disk -> to_psdl
build_from_disk -> from_psdl

@4 KBM 15 Aug 96 Req #.#
Start of an implementation for an I-bar. Currently, this implementation goes back to an arrow when you are over a handle...for moving and reshaping. However, the problem is that it does this when the handle is not displayed. This needs to be corrected.

@5 KBM 15 Aug 96
Change to a watch cursor when you leave edit_graph and return to sde. Go back to normal after drawing has been refreshed on return to edit_graph.

@6 KBM 15 Aug 96 Req #.#
Implemented the new labels for menus. Have put comments in the code where pop-up menus are required to propt the user to save code. A new symbol has been added psdl_modified which will indicate that the graph has been changed and should be saved. NOTE: Currently, this symbol is not being set when changes are made. This will have to be added.

@7 KBM 15 Aug 96 Req #.#
Implemented as a function call. Not a main program.

@8 KBM 15 Aug 96
Had to change Motif loop so that the edit_graph function could return to calling routine.

@9 KBM 20 Aug 96
Found that hit() did a selection and caused handles to be drawn over any object that had been "hit". In order to correct,

```
        had to add new functions over() and over_handle() which does
        not select anything.

@10  KBM    22 Aug 96
    Changed print menu option to print the toplevel window and
    not just drawing_a.  Also added a XC_watch cursor.

@11  KBM    26 Aug 96
    Had to add PointerMotionMask to normal_mask in draw_stream().

@12  KBM    26 Aug 96
    Moved next_action_ptr and current_graph_ptr from .h file into
        this file since g++ expected them to be declared extern.

@13  KBM    29 Aug 96
    Ref ECP #8, Got the code from MT Shine, stream_propert_dialog.
    Added hooks in for stream_property_dialog and for
        operator_property_menu.  operator_property_menu
        is currently commented out.  stream_property_dialog
        does work.  Added an event for Btn3dn.

@14  WH     30 Aug 96
    Ref ECP #7. The operator_property_menu hook is added

@15  KBM    30 Aug 96
    Modifications for stream latency.

@16  WH     31 Aug 96
    Standardized comments added to each method.

@17  KBM    11 Sep 96
    Moved psdl_modified into GraphObjectList and provided
        methods for access.

@18  KBM    13 Sep 96
    Added code to initialize the graphic editor window.

@19  KBM    13 Sep 96
    Added dialog boxes for save options...now use windows.H
        and windows.C.
**********************************************************/

#include <stdlib.h>
#include <stream.h>

#include <X11/cursorfont.h>                    //@4
#include <X11/Xatom.h>
#include <Xm/DialogS.h>
#include <Xm/DrawingA.h>
#include <Xm/DrawnB.h>
#include <Xm/BulletinB.h>
#include <Xm/PushB.h>
#include <Xm/Form.h>
#include <Xm/LabelG.h>
#include <Xm/List.h>
#include <Xm/MainW.h>
#include <Xm/MessageB.h>
#include <Xm/RowColumn.h>
#include <Xm/PanedW.h>

#include <Xm/PushBG.h>
#include <Xm/ScrolledW.h>
#include <Xm/SelectioB.h>
#include <Xm/Text.h>
#include <Xm/TextF.h>
#include <Xm/ToggleBG.h>
#include <Xm/Xm.h>                             //@4

#include "graph_editor.h"
#include "ge_defs.h"
```

```
#include "ge_interface.h"
#include "graph_object_list.H"
#include "operator_object.H"
#include "spline_object.H"
#include "stream_object.H"
#include "stream_property_menu.H"                    //@13
#include "operator_property_menu.H"                  //@14
#include "ge_utilities.H"
#include "windows.H"

//  MAXCOLORS is the number of colors defined to the editor.
// To add or subtract colors, this value must be modified.

#define BUTTONWIDTH 75

//  graph_editor has a number of global variables due to
// Motif's use of callback functions. Since these functions
// have fixed formal parameter lists, global variables must
// be used to pass some data between functions.

//  All drawing commands are executed on both drawing_a and
// drawing_area_pixmap. drawing_a is the visible canvas, while
// drawing_area_pixmap provides a backup.  When the canvas needs
// to be redrawn, the drawing in drawing_area pixmap is merely
// copied back onto the canvas.

//  colors[] is a list of predefined X colors. To use others,
// consult an X reference giving allowable color names. Using
// the predefined colors allows the user to specify color
// preferences in X resource text files.

//  graphic_list is a GraphObjectList containing all the
// visible operators and streams.

//  selected_object_ptr always points to the object selected
// (i. e. with handles around it) on the drawing canvas.

//  num_del_ops is the number of deleted operators, and
// del_op_id is an array of identifiers for deleted operators.

//  The Resrcs struct, resources[], and options[] are used
// by Motif for parsing the command line options.

static Widget toplevel, main_w, menubar, rowcol, scrolled_win,   //@2
    op_button, term_button,
    stream_button, info_button, timer_button;
static XtAppContext app;
static Pixmap op_button_pixmap, term_button_pixmap,
    stream_button_pixmap, property_button_pixmap,
    select_button_pixmap;
static XGCValues gcv1, gcv2, gcv3;
static Screen *screen_ptr;
static XtActionsRec actions;
static String translations =                          //@4@13
      "<Btn1Down>:  draw(down)\n\
       <Btn1Up>:    draw(up) \n\
       <Btn1Motion>: draw(motion)\n\
       <Btn3Down>:  draw(btn3dn)\n\
       <Motion>:    draw(moved)\n\
       <Key>:       draw(key)\n\
       <Key>Tab:    draw(tab)";
static unsigned long gc_mask;
static Window root_window, toplevel_window;

GC std_graphics_context, dotted_context, erase_context;
Dimension  width, height;
Pixmap drawing_area_pixmap;
Widget drawing_a, graph_title, tool_indicator;
BOOLEAN state_stream = FALSE, alt_selected = FALSE;
char* colors[] = {"Aquamarine", "Black", "Blue", "BlueViolet",
```

```
            "Brown", "CadetBlue", "Coral",
            "CornflowerBlue", "Cyan", "DarkGreen",
            "DarkOliveGreen", "DarkOrchid",
            "DarkSlateBlue", "DarkSlateGrey",
            "DarkTurquoise", "DimGrey", "Firebrick",
            "ForestGreen", "Gold", "Goldenrod", "Grey",
            "Green", "GreenYellow", "IndianRed", "Khaki",
            "LightBlue", "LightGrey", "LightSteelBlue",
            "LimeGreen", "Magenta", "Maroon",
            "MediumAquamarine", "MediumBlue",
            "MediumOrchid", "MediumSeaGreen",
            "MediumSlateBlue", "MediumSpringGreen",
            "MediumTurquoise", "MediumVioletRed",
            "MidnightBlue", "Navy", "Orange", "OrangeRed",
            "Orchid", "PaleGreen", "Pink", "Plum", "Red",
            "Salmon", "SeaGreen", "Sienna", "SkyBlue",
            "SlateBlue", "SpringGreen", "SteelBlue",
            "Tan", "Thistle", "Turquoise", "Violet",
            "VioletRed", "Wheat", "White", "Yellow",
            "YellowGreen"};
unsigned long color_table[MAXCOLORS + 1];
TOOL_STATE tool_state = SELECT_TOOL;

//**KM  added next two lines to support ECP #8                @13
GraphObjectList graphic_list;
GraphObject* selected_object_ptr = NULL;

Display *display_ptr;
Window draw_window;
int default_color = WHITE;
int default_font = COURIERBOLD12;
int num_del_ops = 0, del_op_id[MAXDELETEDOPS];

ACTION_NODE*    next_action_ptr;  // make these two symbols
visible to //@12
```

```
GRAPH_DESC_NODE* current_graph_ptr; // all of graph_editor

StreamObject    st_being_updated;          //@13
OperatorObject  op_being_updated;          //@13

struct _resrcs {
 int viewer;
} Resrcs;

/*********************************************************
* static used for XtResources
*********************************************************/

static XtResource resources[] = {
 {"viewer", "Viewer", XmRBoolean, sizeof (int),
  XtOffsetOf(struct _resrcs,viewer), XmRImmediate, False},
};

/*********************************************************
* returns viewer options
*********************************************************/

static XrmOptionDescRec options[] = {
 {"-v", "viewer", XrmoptionNoArg, "True"},
};

/*********************************************************
* Change the cursor to cursor_id
*********************************************************/

void change_cursor(Widget w, int cursor_id) {
 Cursor C;

 C = XCreateFontCursor(XtDisplay(w), cursor_id);
XDefineCursor(XtDisplay(w), XtWindow(w), C);
```

```
 XFlush(XtDisplay(w));
}

/*********************************************************
* Return cursor to previous shape
*********************************************************/

void normal_cursor(Widget w) {
 XUndefineCursor(XtDisplay(w), XtWindow(w));
 XFlush(XtDisplay(w));
}

/*********************************************************
* error_box displays a Motif dialog box with the given
* error message. The dialog box is created the first time
* it is used. Subsequent calls change the text string and
* redisplay the dialog box.
*********************************************************/

void error_box(char *error_message) {
 static Widget error_dialog = NULL;
 Arg arg[1];
 XmString t;

 if (error_dialog == NULL) {
  error_dialog = XmCreateMessageDialog(drawing_a, "error",
              arg, 0);
  XtVaSetValues(XtParent(error_dialog),
            XtNtitle, "Error",
            NULL);
  XtUnmanageChild(XmMessageBoxGetChild(error_dialog,
             XmDIALOG_HELP_BUTTON)); // Hide help button
 }
 t = XmStringCreateSimple(error_message);
 XtVaSetValues(error_dialog,
          XmNmessageString, t,
          NULL);
 XmStringFree(t);
 XtManageChild(error_dialog);
}

/*********************************************************
*  Initializes the color table.
*********************************************************/
void initialize_color_table(Screen *screen) {
 Colormap color_map = DefaultColormapOfScreen(screen);
 XColor color, unused;
 int i, screen_depth = DefaultDepthOfScreen(screen);

 if (screen_depth > 1) {        //  a color screen
  for (i = 1; i <= MAXCOLORS; i++) {
   if (!XAllocNamedColor(display_ptr, color_map,
     colors[i - 1], &color, &unused))
      cout << "Allocated unknown color: " << colors[i - 1]
        << endl;
   color_table[i] = color.pixel;
  }
 }
 else {    //  a black and white screen //
  for (i = 1; i <= MAXCOLORS; i++) {
   if (strcmp(colors[i - 1], "White") != 0)
    color_table[i] = BlackPixelOfScreen(screen);
   else
    color_table[i] = WhitePixelOfScreen(screen);
  }
 }
}

/*********************************************************
*  Executes menu options from the 'psdl' menu. This is
```

```
*  called by either the menu callback function, if the
*  pulldown menus are used, or by the draw() function,
*  if the alt-key combinations are used.
*********************************************************/

void handle_psdl_options(int item_no) {                    //@6
 char buffer[INPUT_LINE_SIZE];
 int  action;
 static PrintCmd prn = {NULL,0};

 Quest_Script abort_script = {"","Abort changes made to graph?", "Yes",
"No",
                              "Cancel", BTN2};
 Quest_Script save_script  = {"","Save changes made to graph?", "Yes",
"No",
                              "Cancel", BTN1};
 if (prn.printer == NULL) prn.printer = make_str("");   // Default value

 XFlush(display_ptr);
 switch(item_no) {

 case 0:  // Save

  next_action_ptr->save    = TRUE;
  next_action_ptr->reinvoke = TRUE;
  next_action_ptr->next_op =
make_str(graphic_list.edited_op_name());
  return_sde_flag         = TRUE;                          //@8

  graphic_list.to_psdl();
  break;

 case 1:  // Restore from Save
  action = YES;                 // Default action if not modified
  if (graphic_list.psdl_modified())
   action = AskUser(app,drawing_a, abort_script);
  switch(action) {
   case YES:
        next_action_ptr->save    = FALSE;
        next_action_ptr->reinvoke = TRUE;
        free(next_action_ptr->next_op);
        next_action_ptr->next_op =
make_str(graphic_list.edited_op_name());
        return_sde_flag = TRUE;                            //@8
        break;

   case NO:
        return_sde_flag = FALSE;  // Aborted operation, do nothing
        break;
   }

  break;


 case 2:  // Go to Parent

  if (graphic_list.parent_op_name() == NULL) {
   error_box("No parent node defined");

   break;
  }

  action = NO;   // Default action if not modified
            // This is not the default save option, see save_script
  if (graphic_list.psdl_modified())
   action = AskUser(app,drawing_a, save_script);
  switch(action) {
   case YES:
        next_action_ptr->save    = TRUE;
        next_action_ptr->reinvoke = TRUE;
```

```
            free(next_action_ptr->next_op);
            next_action_ptr->next_op =
make_str(graphic_list.parent_op_name());
            return_sde_flag = TRUE;                          //@8
            break;

        case NO:
            next_action_ptr->save    = FALSE;
            next_action_ptr->reinvoke = TRUE;
            free(next_action_ptr->next_op);
            next_action_ptr->next_op =
make_str(graphic_list.parent_op_name());
            return_sde_flag = TRUE;                          //@8
            break;

        case CANCEL:
            return_sde_flag = FALSE;
            break;
      }

      break;

    case 3:  // Decompose
      if (selected_object_ptr == NULL)
        error_box("Please select an operator");

      else {
        if (selected_object_ptr->is_a() == OPERATOROBJECT) {

            action = NO;   // Default action if not modified
                  // This is not the default save option, see save_script
            if (graphic_list.psdl_modified())
              action = AskUser(app,drawing_a, save_script);
            switch(action) {
          case YES:
            next_action_ptr->save    = TRUE;
            next_action_ptr->reinvoke = TRUE;
            free(next_action_ptr->next_op);
            next_action_ptr->next_op = make_str(selected_object_ptr-
>name());
            return_sde_flag = TRUE;                          //@8
            break;

          case NO:
            next_action_ptr->save    = FALSE;
            next_action_ptr->reinvoke = TRUE;
            free(next_action_ptr->next_op);
            next_action_ptr->next_op = make_str(selected_object_ptr-
>name());
            return_sde_flag = TRUE;                          //@8
            break;

          case CANCEL:
            return_sde_flag = FALSE;
            break;
          }

          break;

        }
        else
          error_box("Please select an operator");
      }

      break;

    case 4:  // Print
      AskPrint(app,drawing_a, &prn);
      if (prn.answer == OK) {
        graphic_list.draw();     // Redraw the display to get rid of print box
```

```
    sprintf(buffer, "xwd -frame -id %d | xpr -gray 2 -device ps | lpr %s ",
            XtWindow(toplevel), (prn.printer == NULL ? " " :
prn.printer)); //@10
    // Note: space is needed at the end of the string for unix

    change_cursor(toplevel,XC_watch);                    //@10
    system (buffer);
    normal_cursor(toplevel);                             //@10
   }

   break;

  case 5:  // Return to SDE

   action = NO;   // Default action if not modified
            // This is not the default save option, see save_script
   if (graphic_list.psdl_modified())
    action = AskUser(app,drawing_a, save_script);
   switch(action) {
    case YES:
         next_action_ptr->save    = TRUE;
         next_action_ptr->reinvoke = FALSE;
         next_action_ptr->next_op  = NULL;
         return_sde_flag = TRUE;                         //@8
         break;

    case NO:
         next_action_ptr->save    = FALSE;
         next_action_ptr->reinvoke = FALSE;
         next_action_ptr->next_op  = NULL;
         return_sde_flag = TRUE;                         //@8
         break;

    case CANCEL:
    default:
         return_sde_flag = FALSE;
         break;
   }

   break;

  default:

   break;
  }
 }


/*********************************************************
*  This function is called by the window manager when a
*  menu option from the 'psdl' pulldown menu is selected.  Its
*  address is passed as a parameter to X when the menus are
*  defined.
*********************************************************/

static void psdl_menu_cb(Widget, XtPointer client_data, XtPointer) {
 int item_no = (int) client_data;

 handle_psdl_options(item_no);
}

/*********************************************************
*  This function is called when a selection is made from
* the list box displayed in the 'edit:Color' menu.
*********************************************************/
void color_list_cb(Widget widget, XtPointer,
            XtPointer cb_struct_ptr) {
 XmListCallbackStruct *list_struct_ptr =
  (XmListCallbackStruct *) cb_struct_ptr;
```

```
 if (selected_object_ptr != NULL) {
  if (selected_object_ptr->is_a() == OPERATOROBJECT) {
   selected_object_ptr->erase();
   selected_object_ptr->set_color(
                    list_struct_ptr->item_position);
   selected_object_ptr->draw(SOLID);
  }
 }
 else
  default_color = list_struct_ptr->item_position;
 XtDestroyWidget(widget);
}

/*********************************************************
*  This function is called when a selection is made from
*  the list box displayed in the 'edit:Font' menu.
*********************************************************/

void font_list_cb(Widget widget, XtPointer,
          XtPointer cb_struct_ptr) {
 XmListCallbackStruct *list_struct_ptr =
  (XmListCallbackStruct *) cb_struct_ptr;

 if (selected_object_ptr != NULL) {
  selected_object_ptr->erase();
  selected_object_ptr->set_object_font(
                    list_struct_ptr->item_position);
  selected_object_ptr->draw(SOLID);
 }
 else {
  default_font = list_struct_ptr->item_position;
  graphic_list.set_default_font(default_font);
 }
 XtDestroyWidget(widget);
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
}

/*********************************************************
*  This function is called when a selection is made from
*  the list box displayed in the 'edit:Undelete Operator'
*  menu.
*********************************************************/

static void op_list_cb(Widget widget, XtPointer,
          XtPointer cb_struct_ptr) {
 XmListCallbackStruct *list_struct_ptr =
  (XmListCallbackStruct *) cb_struct_ptr;

// The last entry in the list is 'Cancel'.
 if (list_struct_ptr->item_position != num_del_ops + 1) {
  graphic_list.set_undeleted(OPERATOROBJECT,
          del_op_id[list_struct_ptr->item_position - 1]);
  graphic_list.draw();
 }
 XtDestroyWidget(widget);
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
}

/*********************************************************
*  Executes menu options from the 'Edit' menu. This is
*  called by either the menu callback function, if the
*  pulldown menus are used, or by the draw() function,
*  if the alt-key combinations are used.
*********************************************************/

void handle_edit_options(int item_no) {
 int i, num_items = XtNumber(colors);
 XmStringTable color_list, font_list, op_list;
 Widget list_box, op_box;
 char *del_op_str[MAXDELETEDOPS];
```

```
switch(item_no) {
case 0:  // Color
   color_list =
    (XmStringTable) XtMalloc(num_items * sizeof(XmString *));
   for (i = 0; i < num_items; i++)
    color_list[i] = XmStringCreateSimple(colors[i]);
   list_box =
        XmCreateScrolledList(drawing_a, "Colors", NULL, 0);
   XtVaSetValues(list_box,
    XmNitems, color_list,
    XmNitemCount, num_items,
    XmNvisibleItemCount, 8,
    NULL);
   for (i = 0; i < num_items; i++)
    XmStringFree(color_list[i]);
   XtFree((char *) color_list);
   XtAddCallback(list_box, XmNdefaultActionCallback,
            color_list_cb, NULL);
   XtManageChild(list_box);
   break;
case 1:  // Font
   font_list =
     (XmStringTable) XtMalloc(MAXFONTS * sizeof(XmString *));
   for (i = 0; i < MAXFONTS; i++)
    font_list[i] =
       XmStringCreateSimple(graphic_list.font_name(i + 1));
   list_box =
        XmCreateScrolledList(drawing_a, "Fonts", NULL, 0);
   XtVaSetValues(list_box,
    XmNitems, font_list,
    XmNitemCount, MAXFONTS,
    XmNvisibleItemCount, 7,
    NULL);
   for (i = 0; i < MAXFONTS; i++)
    XmStringFree(font_list[i]);
   XtFree((char *) font_list);
   XtAddCallback(list_box, XmNdefaultActionCallback,
            font_list_cb, NULL);
   XtManageChild(list_box);
   break;
case 2:  // Undelete Operator
   if (selected_object_ptr != NULL) {
    selected_object_ptr->unselect();
    selected_object_ptr = NULL;
   }
   graphic_list.get_del_op_list(del_op_str, del_op_id,
                    num_del_ops);
   op_list = (XmStringTable)
         XtMalloc((num_del_ops + 1) * sizeof(XmString *));
   for (i = 0; i < num_del_ops; i++)
    op_list[i] = XmStringCreateSimple(del_op_str[i]);
   op_list[num_del_ops] = XmStringCreateSimple("Cancel");
   op_box = XmCreateScrolledList(drawing_a, "Undelete",
                    NULL, 0);
   XtVaSetValues(op_box,
            XmNitems, op_list,
            XmNitemCount, num_del_ops + 1,
            XmNvisibleItemCount, 7,
            NULL);
   for (i = 0; i < num_del_ops + 1; i++)
    XmStringFree(op_list[i]);
   XtFree((char *) op_list);
   XtAddCallback(op_box, XmNdefaultActionCallback,
            op_list_cb, NULL);
   XtManageChild(op_box);
   break;
default:
   break;
}
```

```
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
}

/*********************************************************
*  This function is called by the window manager when a
*  menu option from the 'Edit' pulldown menu is
*  selected.  Its address is passed as a parameter to X when
*  the menus are defined.
*********************************************************/

static void edit_menu_cb(Widget, XtPointer client_data,
                         XtPointer) {
 int item_no = (int) client_data;

 handle_edit_options(item_no);
}

/*********************************************************
*  This function is called by the window manager when a
*  menu option from the 'Help' pulldown menu is
*  selected.  Its address is passed as a parameter to X when
*  the menus are defined.
*********************************************************/

static void help_menu_cb(Widget, XtPointer, XtPointer) {};

/*  ================== Not USED !!!! ====================

void set_color(Widget widget, char *color) {
 Display *dpy = XtDisplay(widget);
 Colormap cmap = DefaultColormapOfScreen(XtScreen(widget));
 XColor col, unused;

 if (!XAllocNamedColor(dpy, cmap, color, &col, &unused)) {
  error_box("Can't allocate color");
  return;
 }
 XSetForeground(dpy, std_graphics_context, col.pixel);
}
*/

//  Builds the top line menu bar.


/*********************************************************
*  This function is called to build the menu bar with the desired widgets.
*
*********************************************************/

void build_menu_bar(Widget &main_w, Widget &menubar) {
//@6
 // 8/4/96 KBM  Updated for label changes in Req 4
 //          Also changed callback names to reflect new labels.

 XmString psdl_menu, save_opt, restore_opt, goto_parent_opt,
        decompose_opt, print_opt, return_sde_opt,
       edit_menu, color_opt, font_opt, undelete_opt,
       help_menu;
 Widget widget;

 psdl_menu       = XmStringCreateSimple("PSDL");
  save_opt       = XmStringCreateSimple("Save");
  restore_opt    = XmStringCreateSimple("Restore from Save");
  goto_parent_opt = XmStringCreateSimple("Go to Parent");
  decompose_opt  = XmStringCreateSimple("Decompose");
  print_opt      = XmStringCreateSimple("Print");
  return_sde_opt = XmStringCreateSimple("Exit to SDE");

 edit_menu       = XmStringCreateSimple("Edit");
  color_opt      = XmStringCreateSimple("Color");
```

```
 font_opt      = XmStringCreateSimple("Font");
 undelete_opt  = XmStringCreateSimple("Undelete Operator");

help_menu      = XmStringCreateSimple("Help");

menubar = XmVaCreateSimpleMenuBar(main_w, "menubar",
   XmVaCASCADEBUTTON, psdl_menu, NULL,
   XmVaCASCADEBUTTON, edit_menu, NULL,
   XmVaCASCADEBUTTON, help_menu, 'H', NULL);
if (widget = XtNameToWidget(menubar, "button_2"))
 XtVaSetValues(menubar, XmNmenuHelpWidget, widget, NULL);

XmVaCreateSimplePulldownMenu(menubar, "psdl_menu", 0,
                psdl_menu_cb,
 XmVaPUSHBUTTON, save_opt, 'S', NULL, NULL,
 XmVaPUSHBUTTON, restore_opt, 'R', NULL, NULL,
 XmVaPUSHBUTTON, goto_parent_opt, 'G', NULL, NULL,
 XmVaPUSHBUTTON, decompose_opt, 'D', NULL, NULL,
 XmVaPUSHBUTTON, print_opt, 'P', NULL, NULL,
 XmVaPUSHBUTTON, return_sde_opt, 'E', NULL, NULL,
 NULL);

XmVaCreateSimplePulldownMenu(menubar, "edit_menu", 1,
                edit_menu_cb,
 XmVaPUSHBUTTON, color_opt, 'C', NULL, NULL,
 XmVaPUSHBUTTON, font_opt, 'F', NULL, NULL,
 XmVaPUSHBUTTON, undelete_opt, 'U', NULL, NULL,
 NULL);

 XmVaCreateSimplePulldownMenu(menubar, "help_menu", 2,
help_menu_cb,
  XmVaPUSHBUTTON, help_menu, 'H', NULL, NULL,
  NULL);

 XmStringFree(psdl_menu);
 XmStringFree(save_opt);
 XmStringFree(restore_opt);
 XmStringFree(goto_parent_opt);
 XmStringFree(decompose_opt);
 XmStringFree(print_opt);
 XmStringFree(return_sde_opt);
 XmStringFree(edit_menu);
 XmStringFree(color_opt);
 XmStringFree(font_opt);
 XmStringFree(undelete_opt);
 XmStringFree(help_menu);
}

/*********************************************************
*  Createpush buttons used to select the tools.
*********************************************************/

void make_buttons(Widget &rowcol, Widget &op_button,
          Widget &term_button, Widget &stream_button,
          Widget &info_button,
          Widget &timer_button,
          Pixmap &op_button_pixmap,
          Pixmap &term_button_pixmap,
          Pixmap &stream_button_pixmap,
          Display *display_ptr, Screen *screen_ptr) {
 Window root_window = RootWindowOfScreen(screen_ptr);
 unsigned int screen_depth = DefaultDepthOfScreen(screen_ptr);

 op_button_pixmap = XCreatePixmap(display_ptr, root_window,
             BUTTONWIDTH, BUTTONWIDTH, screen_depth);
 term_button_pixmap = XCreatePixmap(display_ptr, root_window,
             BUTTONWIDTH, BUTTONWIDTH, screen_depth);
 stream_button_pixmap = XCreatePixmap(display_ptr, root_window,
             BUTTONWIDTH, BUTTONWIDTH, screen_depth);
 XFillRectangle(display_ptr, (Drawable) op_button_pixmap,
```

```
        erase_context, 0, 0, BUTTONWIDTH, BUTTONWIDTH);
XFillRectangle(display_ptr, (Drawable) term_button_pixmap,
        erase_context, 0, 0, BUTTONWIDTH, BUTTONWIDTH);
XFillRectangle(display_ptr, (Drawable) stream_button_pixmap,
        erase_context, 0, 0, BUTTONWIDTH, BUTTONWIDTH);

XSetLineAttributes(display_ptr, std_graphics_context, 2,
          LineSolid, CapButt, JoinMiter);
XDrawArc(display_ptr, (Drawable) op_button_pixmap,
      std_graphics_context, 10, 10, 45, 45, CIRCLE_BEGIN,
      FULL_CIRCLE);
XDrawRectangle(display_ptr, (Drawable) term_button_pixmap,
        std_graphics_context, 10, 10, 45, 45);
XDrawLine(display_ptr, (Drawable) stream_button_pixmap,
       std_graphics_context, 10, 10, 55, 55);
// XDrawString(display_ptr, (Drawable) property_button_pixmap,
//         std_graphics_context, 5, 35, "Properties", 10);
// XDrawString(display_ptr, (Drawable) select_button_pixmap,
//         std_graphics_context, 10, 35, "Info", 6);
op_button = XtVaCreateManagedWidget("op_button",
                    xmDrawnButtonWidgetClass,
                    rowcol,
                    XmNrecomputeSize, FALSE,
                    XmNpushButtonEnabled, TRUE,
                    XmNwidth, BUTTONWIDTH,
                    XmNheight, BUTTONWIDTH,
                    XmNlabelType, XmPIXMAP,
                    XmNlabelPixmap, op_button_pixmap,
                    NULL);
term_button = XtVaCreateManagedWidget("term_button",
                    xmDrawnButtonWidgetClass,
                    rowcol,
                    XmNrecomputeSize, FALSE,
                    XmNpushButtonEnabled, TRUE,
                    XmNwidth, BUTTONWIDTH,
                    XmNheight, BUTTONWIDTH,
                    XmNlabelType, XmPIXMAP,
                    XmNlabelPixmap, term_button_pixmap,
                    NULL);
stream_button = XtVaCreateManagedWidget("stream_button",
                    xmDrawnButtonWidgetClass,
                    rowcol,
                    XmNrecomputeSize, FALSE,
                    XmNpushButtonEnabled, TRUE,
                    XmNwidth, BUTTONWIDTH,
                    XmNheight, BUTTONWIDTH,
                    XmNlabelType, XmPIXMAP,
                    XmNlabelPixmap, stream_button_pixmap,
                    NULL);
XmString info_text = XmStringCreateSimple("INFO");
XmString timer_text = XmStringCreateSimple("TIMERS");

info_button = XtVaCreateManagedWidget("info_button",
                   xmPushButtonWidgetClass,
                   rowcol,
                   XmNlabelString, info_text,
                   NULL);
XmStringFree(info_text);

timer_button = XtVaCreateManagedWidget("timer_button",
                    xmPushButtonWidgetClass,
                    rowcol,
                    XmNlabelString, timer_text,
                    NULL);
XmStringFree(timer_text);

XSetLineAttributes(display_ptr, std_graphics_context, 1,
          LineSolid, CapButt, JoinMiter);


}
```

```
/*******************************************************
*  Redraws the drawing canvas.
*******************************************************/

void redraw(Widget, XtPointer,
        XtPointer cbs) {
 XmDrawingAreaCallbackStruct *temp_ptr;

 temp_ptr = (XmDrawingAreaCallbackStruct *) cbs;
 XCopyArea(temp_ptr->event->xexpose.display,
        drawing_area_pixmap, temp_ptr->window,
        std_graphics_context, 0, 0, width, height, 0, 0);
}

/*******************************************************
*  Draws a square black box on the canvas to aid in
*  graphic manipulation of objects.
*******************************************************/

void draw_handle(GC graphics_context, int x, int y) {

 x -= HANDLESIZE / 2;
 y -= HANDLESIZE / 2;
 if (x < 0)
   x = 0;
 if (y < 0)
   y = 0;


/*******************************************************
  * When the display function is set to GXxor, the pixel being
  * written is exclusive-or'ed with the target pixel to
  * determine color.  This means that writing the same pixel with
  * the same color twice restores the original color, simplifying
  * the process of erasing handles.

*******************************************************/

 XSetFunction(display_ptr, graphics_context, GXxor);
 XFillRectangle(display_ptr, draw_window, graphics_context,
          x, y, HANDLESIZE, HANDLESIZE);
 XFillRectangle(display_ptr, drawing_area_pixmap,
          graphics_context,
          x, y, HANDLESIZE, HANDLESIZE);
 XSetFunction(display_ptr, graphics_context, GXcopy);
}

/*******************************************************
*  This function erases the temporary guidelines used when
*  streams are drawn.
*  Dotted lines are erased first, then handles.  Since each
*  handle is overwritten with the following dotted line, an
*  erased handle makes an erased blotch in the beginning of the
*  next segment.  When the next segment is written in xor mode,
*  it makes a black mark where the erased handle overwrote the
*  beginning of its segment.
*******************************************************/

void erase_guides(int from_stream_id, SplineObject &temp_spline) {
 OperatorObject *temp_operator_ptr;
 XYPAIR line_start, line_end;

 temp_spline.reset_iter();
 if (from_stream_id != 0) {
   temp_operator_ptr = (OperatorObject *)
     graphic_list.target_object(OPERATOROBJECT, from_stream_id);
   line_start = temp_operator_ptr->center();
 }
 else
```

```
  line_start = temp_spline.next_pair();
 line_end = temp_spline.next_pair();
 while(line_end.x != -1) {
  XDrawLine(display_ptr, draw_window, dotted_context,
        line_start.x, line_start.y, line_end.x,
        line_end.y);
  XDrawLine(display_ptr,drawing_area_pixmap, dotted_context,
        line_start.x, line_start.y, line_end.x,
        line_end.y);
  line_start = line_end;
  line_end = temp_spline.next_pair();
 }
 temp_spline.reset_iter();
 line_end = temp_spline.next_pair();
 while(line_end.x != -1) {
  draw_handle(std_graphics_context, line_end.x, line_end.y);
  line_end = temp_spline.next_pair();
 }
}

/*******************************************************
*  This function is called when a stream is being drawn
*  and the mouse is clicked on either a clear spot on the
*  drawing canvas, or on top of another stream.  If a double-
*  click is registered, the user wants to terminate an external
*  stream.
*******************************************************/

void handle_null_point(int from_stream_id, int &last_point_x,
            int &last_point_y,
            int &x_state, int &y_state,
            XEvent in_event,
            SplineObject &temp_spline, BOOLEAN &done,
            GraphObject *&temp_object_ptr,
            StreamObject *&temp_stream_ptr) {

// Checks for two clicks in the same spot.

#ifdef GE_DEBUG
 cout << "handle_null_point" << endl;
#endif

 if ((from_stream_id != 0) &&
   ((last_point_x - (HANDLESIZE / 2) - HITFUDGE)
    < in_event.xbutton.x) &&
   ((last_point_x + (HANDLESIZE / 2) + HITFUDGE)
    > in_event.xbutton.x) &&
   ((last_point_y - (HANDLESIZE / 2) - HITFUDGE)
    < in_event.xbutton.y) &&
   ((last_point_y + (HANDLESIZE / 2) + HITFUDGE)
    > in_event.xbutton.y)) {
  erase_guides(from_stream_id, temp_spline);
  int new_id = graphic_list.request_id(STREAMOBJECT);
  temp_stream_ptr = new StreamObject("", new_id,
                    from_stream_id,
                    0, UNDEFINED_TIME, US, temp_spline,
TRUE,  //@15
                    FALSE);
  temp_stream_ptr->set_object_ptrs(&graphic_list);
  graphic_list.add(temp_stream_ptr);
  temp_stream_ptr->draw(SOLID);
  temp_stream_ptr = NULL;
  temp_object_ptr = NULL;
  done = TRUE;
  temp_spline.clear();
 }
 else {
  x_state = in_event.xbutton.x;
  y_state = in_event.xbutton.y;
  temp_spline.add(x_state, y_state);
```

```
   XDrawLine(display_ptr, draw_window, dotted_context,
         last_point_x, last_point_y, x_state, y_state);
   XDrawLine(display_ptr, drawing_area_pixmap, dotted_context,
         last_point_x, last_point_y, x_state, y_state);
   draw_handle(std_graphics_context, x_state, y_state);
#ifdef GE_DEBUG
   cout << "ge: " << x_state << " " << y_state << " " <<
     HANDLESIZE << endl;
#endif
   last_point_x = x_state;
   last_point_y = y_state;
  }
}

/*********************************************************
*  Once the user selects the Stream Tool and begins to draw,
*  the draw_stream() function handles all events to speed up
*  performance.
*********************************************************/

void draw_stream(int initial_x, int initial_y) {
 GraphObject *temp_object_ptr;
 OperatorObject *conv_ptr;
 XYPAIR temp_pair;
 int from_stream_id, x_state, y_state, last_point_x,
  last_point_y;
 unsigned long stream_event_mask =
              (ButtonPressMask | PointerMotionMask);
 unsigned long normal_mask = (ButtonPressMask | KeyPressMask |
                   ButtonMotionMask | ExposureMask |
                   ButtonReleaseMask | PointerMotionMask); //@11
 XEvent in_event;
 StreamObject *temp_stream_ptr;
 SplineObject temp_spline;
 BOOLEAN done = FALSE;
```



```
temp_object_ptr = graphic_list.hit(initial_x, initial_y);
if (temp_object_ptr == NULL) {  // External stream
 from_stream_id = 0;
 temp_spline.add(initial_x, initial_y);
 x_state = initial_x;
 y_state = initial_y;
 draw_handle(std_graphics_context, x_state, y_state);
}
else {
 if (temp_object_ptr->is_a() != OPERATOROBJECT) {
// External Stream
  from_stream_id = 0;
  temp_spline.add(initial_x, initial_y);
  x_state = initial_x;
  y_state = initial_y;
  draw_handle(std_graphics_context, x_state, y_state);
 }
 else {
  conv_ptr = (OperatorObject *) temp_object_ptr;
  from_stream_id = conv_ptr->id();
  temp_object_ptr = NULL;
  temp_pair = conv_ptr->center();
  x_state = temp_pair.x;
  y_state = temp_pair.y;
 }
}
last_point_x = x_state;
last_point_y = y_state;
XSelectInput(display_ptr, draw_window,
        stream_event_mask);
while(done == FALSE) {  // monitors the event loop
 XNextEvent(display_ptr, &in_event);
 if (in_event.xbutton.window == draw_window) {
  switch(in_event.type) {
```

```
    case MotionNotify:

#ifdef GE_DEBUG
      cout << "Motion" << endl;
#endif

      XDrawLine(display_ptr, draw_window, dotted_context,
           last_point_x, last_point_y, x_state, y_state);
      XDrawLine(display_ptr, drawing_area_pixmap,
           dotted_context, last_point_x, last_point_y,
           x_state, y_state);
      x_state = in_event.xbutton.x;
      y_state = in_event.xbutton.y;
      XDrawLine(display_ptr, draw_window, dotted_context,
           last_point_x, last_point_y, x_state, y_state);
      XDrawLine(display_ptr, drawing_area_pixmap,
           dotted_context, last_point_x, last_point_y,
           x_state, y_state);
      break;
    case ButtonPress:
#ifdef GE_DEBUG
      cout << "buttonpress" << endl;
#endif
      XDrawLine(display_ptr, draw_window, dotted_context,
           last_point_x, last_point_y, x_state, y_state);
      XDrawLine(display_ptr, drawing_area_pixmap,
           dotted_context, last_point_x, last_point_y,
           x_state, y_state);
      temp_object_ptr = graphic_list.hit(in_event.xbutton.x,
                        in_event.xbutton.y);
      if (temp_object_ptr == NULL) {
       handle_null_point(from_stream_id, last_point_x,
                last_point_y, x_state, y_state,
                in_event, temp_spline, done,
                temp_object_ptr, temp_stream_ptr);
      }
      else
       if (temp_object_ptr->is_a() == OPERATOROBJECT) {

#ifdef GE_DEBUG
 cout << "draw_stream:: object is a OPERATOROBJECT" << endl;
#endif
        erase_guides(from_stream_id, temp_spline);
        int new_id = graphic_list.request_id(STREAMOBJECT);

#ifdef GE_DEBUG
 cout << "draw_stream:: create a new stream" << endl;
#endif
        temp_stream_ptr =
          new StreamObject("", new_id, from_stream_id,
                temp_object_ptr->id(), UNDEFINED_TIME, US,
                temp_spline, TRUE, FALSE);
        temp_stream_ptr->set_object_ptrs(&graphic_list);

#ifdef GE_DEBUG
 cout << "draw_stream:: add to graphic_list" << endl;
#endif
        graphic_list.add(temp_stream_ptr);
        temp_stream_ptr->draw(SOLID);
        temp_stream_ptr = NULL;
        temp_object_ptr = NULL;
        done = TRUE;
        temp_spline.clear();
       }
       else
        if (temp_object_ptr->is_a() == STREAMOBJECT) {
         handle_null_point(from_stream_id, last_point_x,
                last_point_y, x_state, y_state,
                in_event, temp_spline, done,
                temp_object_ptr, temp_stream_ptr);
```

```
        }
      break;
    } //switch
  } //if right window
 } //while done == FALSE
 done = FALSE;
 XSelectInput(display_ptr, draw_window, normal_mask);
}

/*********************************************************
*  Draws the outline of the text being moved.
*********************************************************/

void draw_text_shadow(int x, int y, int width, int height) {

 XDrawRectangle(display_ptr, drawing_area_pixmap,
          dotted_context, x - width / 2, y - height / 2,
          width, height);
 XDrawRectangle(display_ptr, draw_window, dotted_context,
          x - width / 2, y - height / 2, width, height);
}

/*********************************************************
*  The main draw routine.  This function is called by the
*  window manager every time the mouse is moved, a mouse button
*  pressed, or a key pressed inside the draw window.  It is
*  called with a string token that indicates why it was called,
*  and processes the event accordingly.
*********************************************************/

void draw(Widget, XEvent *event, String *args, Cardinal *) {
 static OperatorObject *temp_operator_ptr = NULL;
 static StreamObject   *temp_stream_ptr   = NULL;
 static BOOLEAN first_draw = TRUE, handle_selected = FALSE,
          text_selected = FALSE, drawing_changed = FALSE;
 static int    x_state, y_state, shadow_height, shadow_width,
          from_stream_id;
 GraphObject *temp_object_ptr = NULL;
 int x = event->xbutton.x;
 int y = event->xbutton.y;
 OperatorObject *conv_op_ptr;
 // SplineObject temp_spline;   Can not find where this is needed
 StreamObject *conv_st_ptr;

#ifdef GE_DEBUG
 if (strcmp(args[0], "moved") != 0)
  cout << args[0] << endl;
#endif

 if (strcmp(args[0], "moved") == 0) {  // Mouse moved...no button
//@4
  temp_object_ptr = graphic_list.over(x, y);                    //@9
  if (temp_object_ptr != NULL)
   change_cursor(drawing_a, XC_arrow);
  else
   normal_cursor(drawing_a);
 }
 else
  if (strcmp(args[0], "down") == 0) {  // Button pressed
  XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
  x_state = x;
  y_state = y;
  if (tool_state == SELECT_TOOL) {
   if (selected_object_ptr != NULL) {
    if (selected_object_ptr->hit_handle(x, y)) {
     handle_selected = TRUE;
     if (selected_object_ptr->is_a() == OPERATOROBJECT) {
      delete temp_operator_ptr;
      conv_op_ptr = (OperatorObject *) selected_object_ptr;
      temp_operator_ptr =
```

```
          new OperatorObject("", 0, UNDEFINED_TIME, 0,
conv_op_ptr->x(),
                        conv_op_ptr->y(),
                        conv_op_ptr->radius(),
                        default_color, FALSE,
                        conv_op_ptr->is_composite(),
                        conv_op_ptr->is_terminator());
        temp_operator_ptr->set_handle_selected(
                        conv_op_ptr->handle_selected());
             graphic_list.set_psdl_modified();
       }
      }
      else {        // Unselects previously selected object
       handle_selected = FALSE;
       selected_object_ptr->unselect();
       selected_object_ptr = NULL;
       delete temp_operator_ptr;
       temp_operator_ptr = NULL;
      }
     }
     if (handle_selected == FALSE) {
      temp_object_ptr = graphic_list.hit(x, y);
      if (temp_object_ptr != NULL) {
       temp_object_ptr->select();
       selected_object_ptr = temp_object_ptr;
       text_selected = selected_object_ptr->text_selected();
       if (temp_object_ptr->is_a() == OPERATOROBJECT) {
         // Makes temporary operator to move around
        delete temp_operator_ptr;
        conv_op_ptr = (OperatorObject *) temp_object_ptr;
        temp_operator_ptr =
         new OperatorObject("", 0, UNDEFINED_TIME, 0,
conv_op_ptr->x(),
                        conv_op_ptr->y(),
                        conv_op_ptr->radius(),
                        default_color, FALSE,
                        conv_op_ptr->is_composite(),
                        conv_op_ptr->is_terminator());
             graphic_list.set_psdl_modified();
       }
      }
      else { // No object selected
            text_selected = FALSE;
       temp_object_ptr = NULL;
       selected_object_ptr = NULL;
       delete temp_operator_ptr;
       temp_operator_ptr = NULL;
      }
     }
    }
    else { //*** button down, operator tool selected? *******//
     text_selected = FALSE;
     if ((tool_state == OPERATOR_TOOL) ||
      (tool_state == TERMINATOR_TOOL)) {
      int new_id = graphic_list.request_id(OPERATOROBJECT);
      if (tool_state == OPERATOR_TOOL) {
       temp_operator_ptr =
// BROCKETT 1/22/93 default x and y values changed from 0 to 100
         new OperatorObject("", new_id, UNDEFINED_TIME, 0, 100,
100, 30,
                        default_color, TRUE, FALSE,
                        FALSE);
        temp_operator_ptr->set_location(x, y);
             graphic_list.set_psdl_modified();
       }
       else
        if (tool_state == TERMINATOR_TOOL) {
         temp_operator_ptr =
// BROCKETT 1/22/93 default x and y values changed from 0 to 100
```

```
        new OperatorObject("", new_id, UNDEFINED_TIME, 0, 100,
100, 30,
                        default_color, TRUE, FALSE,
                        TRUE);
       temp_operator_ptr->set_location(x, y);
            graphic_list.set_psdl_modified();
      }
     graphic_list.add((GraphObject *) temp_operator_ptr);
     temp_operator_ptr->draw(SOLID);
     temp_operator_ptr = NULL;
    }
    else // button down, stream tool selected?
     if (tool_state == STREAM_TOOL) {
      draw_stream(x, y);
           graphic_list.set_psdl_modified();
     }
   }
   tool_state = SELECT_TOOL;
   XtVaSetValues(tool_indicator, XmNvalue, "Select Tool", NULL);
  }
  else // button not down
   if (strcmp(args[0], "motion") == 0) {
    if (tool_state == SELECT_TOOL)
     if (selected_object_ptr != NULL) {
      drawing_changed = TRUE;
      if (text_selected) {
       if (first_draw == TRUE) {
        shadow_width = selected_object_ptr->text_width();
        shadow_height = selected_object_ptr->text_height();
        draw_text_shadow(x, y, shadow_width,
                  shadow_height);
        first_draw = FALSE;
       }
       else {
        draw_text_shadow(x_state, y_state, shadow_width,
                  shadow_height);
        draw_text_shadow(x, y, shadow_width,
                  shadow_height);
       }
      }
      else
       if (selected_object_ptr->is_a() == OPERATOROBJECT) {
        if (handle_selected == TRUE) {

         if (first_draw == TRUE) {
          selected_object_ptr->erase();
          selected_object_ptr->unselect();
          selected_object_ptr->draw(SOLID);
          temp_operator_ptr->draw(DOTTED);
          first_draw = FALSE;
         }
         else {
          temp_operator_ptr->move_handle(x - x_state,
                        y - y_state);
#ifdef GE_DEBUG
//          cout << "x: " << x << " y" << y << " xstate: " << x_state << "
y_state: " << y_state << endl;
#endif
         }
        }
        else {
         if (first_draw == TRUE)


/*****************************************************
                * Drawing the same thing twice in xor mode erases
it. When
             * moving an object, it is drawn once the first and last
time,
             * and twice afterwards
```

```
**********************************************************/

          first_draw = FALSE;
        else
          temp_operator_ptr->draw(DOTTED);
        temp_operator_ptr->move(x - x_state, y - y_state);
        temp_operator_ptr->draw(DOTTED);
      }
          graphic_list.set_psdl_modified();
    }
    else {
      if (selected_object_ptr->is_a() == STREAMOBJECT) {
        if (handle_selected) {
          if (first_draw == TRUE) {
            conv_st_ptr =
                 (StreamObject *) selected_object_ptr;
            conv_st_ptr->erase_handle();
            draw_handle(std_graphics_context, x, y);
            first_draw = FALSE;
          }
          else {
            draw_handle(std_graphics_context, x_state,
                   y_state);
            draw_handle(std_graphics_context, x, y);
            selected_object_ptr->move_handle(
                                 x - x_state,
                                 y - y_state);
          }
        }
      }
          graphic_list.set_psdl_modified();
    }
    x_state = x;
    y_state = y;
  }
}
else // button not down, mouse not moved
  if (strcmp(args[0], "up") == 0) {
    if (tool_state == SELECT_TOOL)
      if (selected_object_ptr != NULL) {
        if (text_selected) {
          if (first_draw == FALSE) {
            draw_text_shadow(x_state, y_state,
                        shadow_width, shadow_height);
            selected_object_ptr->text_locate(x, y);
          }
        }
        else
          if (selected_object_ptr->is_a() == OPERATOROBJECT) {
            if (first_draw == FALSE) {
              temp_operator_ptr->draw(DOTTED);
              XYPAIR temp_pair = temp_operator_ptr->center();
              conv_op_ptr =
                (OperatorObject *) selected_object_ptr;
              conv_op_ptr->set_radius(
                          temp_operator_ptr->radius());
              conv_op_ptr->set_location(temp_pair.x,
                                temp_pair.y);
              if (handle_selected)
                conv_op_ptr->set_default_text_location();
            }
          }
          else
            if ((selected_object_ptr->is_a() == STREAMOBJECT)
              && (handle_selected)) {
              draw_handle(std_graphics_context, x_state,
                     y_state);
            }
          if (drawing_changed == TRUE) {
```

```
        graphic_list.move_notify(
                selected_object_ptr->is_a(),
                selected_object_ptr->id());
        graphic_list.draw();
        drawing_changed = FALSE;
       }
       handle_selected = FALSE;
      }
     first_draw = TRUE;
    }
    else
         if (strcmp(args[0], "btn3dn") == 0) {              //@13
          temp_object_ptr = graphic_list.over(x, y);
          if (temp_object_ptr != NULL) {

            if (selected_object_ptr != NULL)
              selected_object_ptr->unselect();
            selected_object_ptr = temp_object_ptr;


/*********************************************************
        * WH 29 Aug 96 modified the if/else control structure below.
        * .if STREAMOBJECT is                                  @13
        * selected then copy all graphobjects to streamobjects..then call
        * the stream_property_dialog() method to display a dialog to
input
        * data else if a OPERATOROBJECT is selected then call the
operator
        * _property_menu() method                             @14
        * to display a dialog to update OPERATOROBJECT properties

*********************************************************/

        if (selected_object_ptr->is_a() == STREAMOBJECT) {
```

```
/*********************************************************
        * WH 29 Aug 96 added to support ECP #8, copy the data
members
        * from the graphobject to                            @13
        * to the StreamObject, the st_being_updated.copy() method
does
        * this copy

*********************************************************/

        st_being_updated.copy((StreamObject*) selected_object_ptr);
        stream_property_dialog(drawing_a, x, y);
            graphic_list.set_psdl_modified();

       } // end of if STREAMOBJECT selected

          else if (selected_object_ptr->is_a() == OPERATOROBJECT)
{


/*********************************************************
        * WH 29 Aug 96 added to support ECP #8, copy the data
members
        * from the graphobject to                            @13
        * to the StreamObject, the st_being_updated.copy() method
does
        * this copy  who

*********************************************************/

            op_being_updated.copy((OperatorObject*)
selected_object_ptr);
        operator_property_dialog(drawing_a, x, y);
            graphic_list.set_psdl_modified();
```

```
      } // end of elseif OPERATROBJECT selected

     } // if not = to null
     }
         else {

#ifdef GE_DEBUG
     cout << args[0] << endl;
     if (selected_object_ptr == NULL)
      cout << "selected_object_ptr == NULL" << endl;
#endif

     if (strcmp(args[0], "key") == 0) {

#ifdef GE_DEBUG
      cout << "key pressed: " << event->xkey.keycode << endl;
#endif

       if (alt_selected) {       // alt key pressed
        alt_selected = FALSE;
        switch(event->xkey.keycode) {
        case 85:
         handle_psdl_options(0);
         break;
        case 63:
         handle_psdl_options(1);
         break;
        case 70:
         handle_psdl_options(2);
         break;
        case 86:
         handle_psdl_options(3);
         break;
        case 68:
         handle_psdl_options(4);
         break;
        case 64:
         handle_psdl_options(5);
         break;
        case 61:
         handle_psdl_options(6);
         break;
        case 109:
         handle_edit_options(0);
         break;
        case 87:
         handle_edit_options(1);
         break;
        case 67:
         handle_edit_options(2);
         break;
        case 26:
         alt_selected = TRUE;
         break;
        default:
         break;
        }
       }
       else
        if (event->xkey.keycode == 26)
         alt_selected = TRUE;
        else
         if (selected_object_ptr != NULL) {
          if ((event->xkey.keycode == 50) || // backspace key
            (event->xkey.keycode == 73)) { // delete key
           selected_object_ptr->erase();
           int deleted_op_id = selected_object_ptr->id();
           graphic_list.delete_notify(selected_object_ptr->
                        is_a(), deleted_op_id);
```

```
            selected_object_ptr->set_deleted();
            selected_object_ptr = NULL;
            graphic_list.draw();
          }
        }
      }
    }
}

/*********************************************************
*  Checks to see whether the indicated string
*  represents a valid number.
*********************************************************/

BOOLEAN valid_num_string(char *num) {
 int index, num_length;

 if (num != NULL) {
  num_length = strlen(num);
  if (num_length > 0) {
   for (index = 0; index < num_length; index++) {
    if ((num[index] != '\n') &&
      (num[index] != '-') &&
      ((num[index] < '0') || (num[index] > '9')))
     return FALSE;
   }
   return TRUE;
  }
 }
 return FALSE;
}

/*********************************************************
*  Callback function.  Just destroys the widget.
*********************************************************/

void widget_killer(Widget widget, XtPointer, XtPointer) {

 XtDestroyWidget(widget);
}

/*********************************************************
*  Callback function called after user has entered data
*  in the property dialog box.  Function reads the data from
*  the dialog box and processes it accordingly.
*********************************************************/

void read_property(Widget widget, XtPointer clientdata,
            XtPointer cbs) {
 XmSelectionBoxCallbackStruct *temp_ptr;
 char *text;
 Widget* widget_ptr;
 StreamObject *conv_ptr;

 selected_object_ptr->erase(); // so it can be redrawn
 widget_ptr = (Widget *) clientdata;
 text = XmTextFieldGetString(*widget_ptr);
 if (text != NULL)
  selected_object_ptr->replace_name(text);
 temp_ptr = (XmSelectionBoxCallbackStruct *) cbs;
 XmStringGetLtoR(temp_ptr->value,
XmSTRING_DEFAULT_CHARSET,
            &text);
 // if (valid_num_string(text))
  //   selected_object_ptr->set_constraint(atoi(text));
  // else
  //   selected_object_ptr->set_constraint(NULL_VALUE);
 if (selected_object_ptr->is_a() == STREAMOBJECT) {
  conv_ptr = (StreamObject *) selected_object_ptr;
  conv_ptr->set_state_variable(state_stream);
```

```
 }
 selected_object_ptr->draw(SOLID);
 XtDestroyWidget(widget);
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
}

/*******************************************************
* Callback function. Called when Operator Tool button is
* pressed.
*******************************************************/

void op_button_cb(Widget, XtPointer, XtPointer) {

 tool_state = OPERATOR_TOOL;
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
 if (selected_object_ptr != NULL) {
  selected_object_ptr->unselect();
  selected_object_ptr = NULL;
 }
 XtVaSetValues(tool_indicator, XmNvalue, "Operator Tool", NULL);
}

/*******************************************************
*  Callback function. Called when Terminator Tool button is
* pressed.
*******************************************************/

void term_button_cb(Widget, XtPointer, XtPointer) {

 tool_state = TERMINATOR_TOOL;
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
 if (selected_object_ptr != NULL) {
  selected_object_ptr->unselect();
  selected_object_ptr = NULL;
 }
 XtVaSetValues(tool_indicator, XmNvalue, "Terminator Tool",
        NULL);
}

/*******************************************************
* Callback function. Called when Stream Tool button is
* pressed.
*******************************************************/

void stream_button_cb(Widget, XtPointer, XtPointer) {

 tool_state = STREAM_TOOL;
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
 if (selected_object_ptr != NULL) {
  selected_object_ptr->unselect();
  selected_object_ptr = NULL;
 }
 XtVaSetValues(tool_indicator, XmNvalue, "Stream Tool", NULL);
}

/*******************************************************
*  Callback function. Called when Select Tool button is
*  pressed.
*******************************************************/
void add_item_cb(Widget, XtPointer, XtPointer) {
  cout << "add item callback" << endl;
}

void timer_cancel_cb(Widget w, XtPointer client_data, XtPointer
call_data) {
  Widget shell = (Widget) client_data;
  XtDestroyWidget(shell);
}

void timer_button_cb(Widget w, XtPointer, XtPointer) {
```

```
 Arg args[5];
 int n=0;

  Widget dialog = XtVaCreatePopupShell("dialog",
xmDialogShellWidgetClass, XtParent (w),
  XmNtitle, "Timer Declarations",
  XmNdeleteResponse, XmDESTROY,
  NULL);
  Widget pane = XtVaCreateWidget("pane",
xmPanedWindowWidgetClass, dialog,
    XmNsashWidth, 1,
    XmNsashHeight, 1,
    NULL);

  Widget rc = XtVaCreateWidget("rc", xmRowColumnWidgetClass,
pane, NULL);

  XtSetArg (args[n], XmNvisibleItemCount, 5); n++;
  Widget list_w = XmCreateScrolledList (rc, "scrolled_list", args, n);
  XtManageChild(list_w);

 Widget text_w = XtVaCreateManagedWidget("text",
xmTextFieldWidgetClass, rc, XmNcolumns, 25,
  NULL);

  XtAddCallback(text_w, XmNactivateCallback, add_item_cb, list_w);
  Widget form = XtVaCreateWidget("form", xmFormWidgetClass, pane,
    XmNfractionBase, 5,
    NULL);
  Widget ok_button = XtVaCreateManagedWidget("OK",
    xmPushButtonGadgetClass, form,
    XmNtopAttachment, XmATTACH_FORM,
    XmNbottomAttachment, XmATTACH_FORM,
    XmNleftAttachment,  XmATTACH_POSITION,
    XmNleftPosition, 1,
    XmNrightPosition, XmATTACH_POSITION,
    XmNshowAsDefault,  True,
    XmNdefaultButtonShadowThickness, 1,
    NULL);

 Widget cancel_button = XtVaCreateManagedWidget("Cancel",
  xmPushButtonGadgetClass, form,
  XmNtopAttachment, XmATTACH_FORM,
  XmNbottomAttachment, XmATTACH_FORM,
  XmNleftAttachment, XmATTACH_POSITION,
  XmNleftPosition, 3,
  XmNrightAttachment, XmATTACH_POSITION,
  XmNrightPosition, 4,
  XmNshowAsDefault, False,
  XmNdefaultButtonShadowThickness, 1,
  NULL);
 XtAddCallback (cancel_button, XmNactivateCallback,
timer_cancel_cb, dialog) ;
  XtManageChild(form);
  XtManageChild(rc);
  XtManageChild(pane);
  XtPopup(dialog, XtGrabNone);

 //tool_state = SELECT_TOOL;
 // XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
 //  XtVaSetValues(tool_indicator, XmNvalue, "Select Tool", NULL);
  cout << "Timer Button Callback";

}
/*********************************************************
*  Null Callback.
*********************************************************/

void null_cb(Widget, XtPointer, XtPointer) {}
```

```
/*********************************************************
*   Callback function.  Called when the radio buttons in the
* properties dialog box are pushed.  Called twice: once to
* unselect old button, again to select the new one.
*********************************************************/

void radio_box_cb(Widget, XtPointer which,
            XtPointer cbs) {
 XmToggleButtonCallbackStruct *state =
  (XmToggleButtonCallbackStruct *) cbs;

 if (state->set) {
  if ((int) which == 0)
   state_stream = FALSE;
  else
   state_stream = TRUE;
 }
}

/*********************************************************
*  Called when Properties button is selected.
*********************************************************/

void info_button_cb(Widget widget, XtPointer, XtPointer) {

  cout << "Info Button Callback" << endl;
} // end of info_button_cb

/*********************************************************
* If graph_editor is invoked in viewer mode, this function
* handles ClientMessage events from the syntax-directed editor.
* Commented-out code handles data passed in a property, which
* this version of the editor doesn't take advantage of.
* Used during testing, and left in for future use, if necessary.
*********************************************************/
```

```
void event_handler(Widget widget, XtPointer,
            XEvent* in_event, Boolean*) {
 Display *display_ptr = XtDisplayOfObject(widget);
 Window window = DefaultRootWindow(display_ptr);
// char **data;
// int return_count;
// XTextProperty text_prop_return;
// Atom property_name;
 char message_in[30];

#ifdef GE_DEBUG
 cout << "\nReceiving:\n" << endl;
#endif

 strcpy(message_in, in_event->xclient.data.b);
 if (strcmp(message_in, "GEDATAIN") == 0) {
//   property_name = XInternAtom(display_ptr, "CAPS_DATA", False);
//   XGetTextProperty(display_ptr, window, &text_prop_return,
property_name);
//   XTextPropertyToStringList(&text_prop_return, &data,
&return_count);
//   if (strcmp(data[0], "Out!!!") == 0)
//     exit(0);
//   else {
//     display_string = XmStringCreateSimple(data[0]);
//     XtVaSetValues(*((Widget *) label), XmNlabelString,
display_string, NULL);
//     XmStringFree(display_string);
//   }

   graphic_list.from_psdl();                    //@3
   graph_desc_clear(current_graph_ptr);
   graphic_list.draw();
 }
```

```
 else
  if (strcmp(message_in, "GEQUIT") == 0) {

#ifdef GE_DEBUG
    cout << "Got viewer quit" << endl;
#endif

    exit(0);
  }
}

/*********************************************************
*  Displays information about event received by program.
* Not currently used, but left in for future use.
*********************************************************/

void event_list(XEvent *in_event) {

 cout << "type: " << in_event->xclient.type;
 cout << "\nserial: " << in_event->xclient.serial;
 cout << "\nsend_event: " << in_event->xclient.send_event;
 cout << "\ndisplay: " << in_event->xclient.display;
 cout << "\nwindow: " << in_event->xclient.window;
 cout << "\nformat: " << in_event->xclient.format;
 cout << "\ndata: " << in_event->xclient.data.b << endl;
}

/*********************************************************
*  Displays information about event received by program.
* Not currently used, but left in for future use.
*********************************************************/

int id_event(XEvent event_to_check) {

 switch(event_to_check.type) {
```

```
case KeyPress:
 cout << "KeyPress";
 break;
case KeyRelease:
 cout << "KeyRelease";
 break;
case ButtonPress:
 cout << "ButtonPress";
 break;
case ButtonRelease:
 cout << "ButtonRelease";
 break;
case MotionNotify:
 cout << "MotionNotify";
 break;
case EnterNotify:
 cout << "EnterNotify";
 break;
case LeaveNotify:
 cout << "LeaveNotify";
 break;
case FocusIn:
 cout << "FocusIn";
 break;
case FocusOut:
 cout << "FocusOut";
 break;
case KeymapNotify:
 cout << "KeymapNotify";
 break;
case Expose:
 cout << "Expose";
 break;
case GraphicsExpose:
 cout << "GraphicsExpose";
```

```
  break;
case NoExpose:
  cout << "NoExpose";
  break;
case VisibilityNotify:
  cout << "VisibilityNotify";
  break;
case CreateNotify:
  cout << "CreateNotify";
  break;
case DestroyNotify:
  cout << "DestroyNotify";
  break;
case UnmapNotify:
  cout << "UnmapNotify";
  break;
case MapNotify:
  cout << "MapNotify";
  break;
case MapRequest:
  cout << "MapRequest";
  break;
case ReparentNotify:
  cout << "ReparentNotify";
  break;
case ConfigureNotify:
  cout << "ConfigureNotify";
  break;
case ConfigureRequest:
  cout << "ConfigureRequest";
  break;
case GravityNotify:
  cout << "GravityNotify";
  break;
case ResizeRequest:
  cout << "ResizeRequest";
  break;
case CirculateNotify:
  cout << "CirculateNotify";
  break;
case CirculateRequest:
  cout << "CirculateRequest";
  break;
case PropertyNotify:
  cout << "PropertyNotify";
  break;
case SelectionClear:
  cout << "SelectionClear";
  break;
case SelectionRequest:
  cout << "SelectionRequest";
  break;
case SelectionNotify:
  cout << "SelectionNotify";
  break;
case ColormapNotify:
  cout << "ColormapNotify";
  break;
case ClientMessage:
  cout << "ClientMessage";
  break;
case MappingNotify:
  cout << "MappingNotify";
  break;
default:
  cout << event_to_check.type;
}
cout << " Event, send_event ";
if (event_to_check.xany.send_event)
  cout << "TRUE";
```

```
 else
  cout << "FALSE";
 cout << ", Target Window: " << hex << event_to_check.xany.window
<< endl;
 return event_to_check.type;
}

/*********************************************************
*  Changes the value displayed in the title box below the
*  draw window.
*********************************************************/

void set_title() {

 if (graphic_list.name() != NULL)
  XtVaSetValues(graph_title, XmNvalue, graphic_list.name(), NULL);
}

/*********************************************************
* translations provides the mappings for the keyboard
* mapping table that allow the drawing canvas to capture
* mouse and keyboard events.
*
*
* Initialize Motif Windows...This routine sets global variables so that
* the next time edit_graph is called, do not have to re-initialize the
* windows.
*********************************************************/

void init_motif() {                                    //@2
 motif_initialized = TRUE;

 // Simulated arguments
 char* args[] = {"edit_graph","-geometry","800x600",NULL};
//@18
int signed_argc = 3;                                   //@18
char** argv = args;

toplevel = XtVaAppInitialize(&app, "Graph_editor", options,
                             XtNumber(options), &signed_argc, argv,
                             NULL, NULL);

display_ptr = XtDisplay(toplevel);
XtGetApplicationResources(toplevel, (XtPointer) &Resrcs,
                          resources, XtNumber(resources),
                          NULL, 0);
screen_ptr = XtScreen(toplevel);
initialize_color_table(screen_ptr);
root_window = RootWindowOfScreen(screen_ptr);
gcv1.foreground = BlackPixelOfScreen(screen_ptr);
gcv1.background = WhitePixelOfScreen(screen_ptr);
gcv2.foreground = BlackPixelOfScreen(screen_ptr);
gcv2.background = WhitePixelOfScreen(screen_ptr);
gcv3.foreground = WhitePixelOfScreen(screen_ptr);
gcv3.background = WhitePixelOfScreen(screen_ptr);
gc_mask = GCForeground | GCBackground;
std_graphics_context = XCreateGC(display_ptr,
                                 root_window, gc_mask, &gcv1);
dotted_context = XCreateGC(display_ptr,
                           root_window, gc_mask, &gcv2);
erase_context = XCreateGC(display_ptr, root_window, gc_mask,
                          &gcv3);
XSetLineAttributes(display_ptr, dotted_context, 1,
                   LineOnOffDash, CapButt, JoinMiter);
XSetFunction(display_ptr, dotted_context, GXxor);

if (Resrcs.viewer == False) {  //  editor mode, not view mode
  main_w = XtVaCreateManagedWidget("main_w",
xmFormWidgetClass,
                                 toplevel, NULL);
```

```
  build_menu_bar(main_w, menubar);
  XtManageChild(menubar);
  rowcol =
   XtVaCreateManagedWidget("rowcol", xmRowColumnWidgetClass,
                           main_w,
                           XmNnumColumns, 1,
                           NULL);

  make_buttons(rowcol, op_button, term_button, stream_button,
               info_button, timer_button,
               op_button_pixmap, term_button_pixmap,
               stream_button_pixmap,
               display_ptr, screen_ptr);

  XtAddCallback(op_button, XmNactivateCallback, op_button_cb,
               NULL);
  XtAddCallback(term_button, XmNactivateCallback,
               term_button_cb, NULL);
  XtAddCallback(stream_button, XmNactivateCallback,
               stream_button_cb, NULL);
  XtAddCallback(timer_button, XmNactivateCallback,
               timer_button_cb, NULL);
  XtAddCallback(info_button, XmNactivateCallback,
               info_button_cb, NULL);
  scrolled_win =
   XtVaCreateManagedWidget("scrolled_win",
                           xmScrolledWindowWidgetClass,
                           main_w,
//                 XmNwidth, 1200,
//                 XmNheight, 750,
                           XmNscrollingPolicy, XmAUTOMATIC,
                           XmNscrollBarDisplayPolicy,
XmAS_NEEDED,
                           NULL);
   actions.string = "draw";
   actions.proc = draw;
   XtAppAddActions(app, &actions, 1);
   graph_title =
    XtVaCreateManagedWidget("graph_title", xmTextWidgetClass,
                            main_w,
                            XmNvalue, "",
                            NULL);
   tool_indicator = XtVaCreateManagedWidget("tool_indicator",
                                            xmTextWidgetClass,
                                            main_w,
                                            XmNvalue, "Select
Tool",
                                            NULL);
    XtVaSetValues(toplevel, XmNtitle, "Graph Editor", NULL);
   }
   else {
    scrolled_win =
     XtVaCreateManagedWidget("scrolled_win",
                             xmScrolledWindowWidgetClass,
toplevel,
//                 XmNwidth, 600,
//                 XmNheight, 400,
                             XmNscrollingPolicy, XmAUTOMATIC,
                             XmNscrollBarDisplayPolicy,
XmAS_NEEDED,
                             NULL);
    XtVaSetValues(toplevel, XmNtitle, "Graph Viewer", NULL);
   }

   drawing_a =
    XtVaCreateManagedWidget("drawing_a",
                            xmDrawingAreaWidgetClass,
scrolled_win,
                            XmNunitType, Xm1000TH_INCHES,
                            XmNwidth, 11000,
```

```
                        XmNheight, 8500,
                        XmNresizePolicy, XmNONE,
                        NULL);
XtAddCallback(drawing_a, XmNexposeCallback, redraw, NULL);

XtVaSetValues(drawing_a, XmNunitType, XmPIXELS, NULL);
XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
XtVaGetValues(drawing_a, XmNwidth, &width, XmNheight, &height,
              NULL);
drawing_area_pixmap = XCreatePixmap(display_ptr,
                                    root_window, width, height,

DefaultDepthOfScreen(screen_ptr));
XFillRectangle(display_ptr, drawing_area_pixmap,
               erase_context, 0, 0, width, height);

if (Resrcs.viewer == False) {
 XtVaSetValues(drawing_a,  XmNtranslations,
               XtParseTranslationTable(translations), NULL);
 XtVaSetValues(menubar, XmNtopAttachment, XmATTACH_FORM,
               XmNrightAttachment, XmATTACH_FORM,
               XmNleftAttachment, XmATTACH_WIDGET,
               XmNleftWidget, rowcol,
               XmNbottomAttachment, XmATTACH_NONE,
               NULL);

 XtVaSetValues(rowcol, XmNtopAttachment, XmATTACH_FORM,
               XmNrightAttachment, XmATTACH_NONE,
               XmNleftAttachment, XmATTACH_FORM,
               XmNbottomAttachment, XmATTACH_WIDGET,
               XmNbottomWidget, tool_indicator,
               NULL);
 XtVaSetValues(scrolled_win, XmNtopAttachment,
               XmATTACH_WIDGET,
               XmNtopWidget, menubar,
               XmNrightAttachment, XmATTACH_FORM,
               XmNleftAttachment, XmATTACH_WIDGET,
               XmNleftWidget, rowcol,
               XmNbottomAttachment, XmATTACH_WIDGET,
               XmNbottomWidget, graph_title,
               NULL);
 XtVaSetValues(graph_title, XmNtopAttachment,
XmATTACH_NONE,
               XmNrightAttachment, XmATTACH_FORM,
               XmNleftAttachment, XmATTACH_WIDGET,
               XmNleftWidget, tool_indicator,
               XmNbottomAttachment, XmATTACH_FORM,
               NULL);
 XtVaSetValues(tool_indicator, XmNtopAttachment,
               XmATTACH_NONE,
               XmNrightAttachment, XmATTACH_NONE,
               XmNleftAttachment, XmATTACH_FORM,
               XmNbottomAttachment, XmATTACH_FORM,
               NULL);
}
XtRealizeWidget(toplevel);
draw_window = XtWindow(drawing_a);
graphic_list.set_draw_environ(display_ptr,
                              std_graphics_context,
                              erase_context, dotted_context,
                              draw_window,
                              &drawing_area_pixmap,
                              color_table,
                              width, height);
graphic_list.set_error_tgt(drawing_a);
if (Resrcs.viewer == False)
 set_title();
XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);

if (Resrcs.viewer != False) {
```

```
  toplevel_window = XtWindowOfObject(toplevel);
  Atom display_id_atom = XInternAtom(display_ptr, "WINDOW_ID",
False);
  XChangeProperty(display_ptr, root_window, display_id_atom,
                  XA_WINDOW, 32, PropModeReplace,
                  (unsigned char *) &toplevel_window, 1);
  XtAddEventHandler(toplevel, NoEventMask, TRUE, event_handler,
NULL);
 }
}

/*********************************************************
* this method is added to support the edit graph and sde change over.
* now the edit graph module is not a standalone but a method called
* from the sde
*********************************************************/

int edit_graph(GRAPH_DESC current_graph, ACTION next_action) {
//@7

 XEvent event;  // added for custom main loop

 /*********************************************************
 *  7/11/96 MTS
 * add the following signed_argc variable
 * and replace the actual parameter &argc
 * by &signed_argc to get around type conflict
 *********************************************************/
 next_action_ptr  = next_action;
 current_graph_ptr = current_graph;
 return_sde_flag = FALSE;                          //@8
 // motif_initialized assumed to be false at start of procedure

 if (motif_initialized) {                          //@2
  graphic_list.from_psdl();                        //@3
  graph_desc_clear(current_graph_ptr);
  graphic_list.draw();
  normal_cursor(toplevel);  // change cursor back          //@2
 }
 else {
  init_motif();
  graphic_list.from_psdl();                        //@3
  if (Resrcs.viewer == False)
   set_title();
  graph_desc_clear(current_graph_ptr);
  graphic_list.draw();
 }

 printf("\n"); //flushes the event queue
 XFlush(display_ptr);

#ifdef GE_DEBUG
    cout << "Starting Motif event loop" << endl;
#endif

 selected_object_ptr = NULL;

 // Custom main loop to check for return to sde          //@8
 do {
  XtAppNextEvent(app, &event);
  XtDispatchEvent(&event);
 } while (return_sde_flag == FALSE);

#ifdef GE_DEBUG
 cout << "Leaving edit_graph..." << endl;
#endif

 // If we are not coming back, kill the window
 if (!next_action_ptr->reinvoke) {
  XtUnrealizeWidget(toplevel);
```

```
  XFlush(display_ptr); }
 else { // otherwise, will be returning, erase window and exit
  change_cursor(toplevel,XC_watch);                    //@5
  XFillRectangle(display_ptr, drawing_area_pixmap,
                    erase_context, 0, 0, width, height);
  XFillRectangle(display_ptr, draw_window,
                    erase_context, 0, 0, width, height); }

 return 0;
}

/*********************************************************
*              --- end of graph_editor.C
*********************************************************/
/* *******************************************************

Name:       graph_object.h
Author:     Capt Robert M. Dixon
Program:    graph_editor
Date Modified: 11 Sep 92
Remarks:  Specification for the GraphObject class.

      The GraphObject is the base class for the main graph
      objects displayed in the graph editor. It is not
      designed to be directly instantiated, so most of its
      methods are virtual.

      There are a number of static class variables used
      to store graphic information necessary for the
      descendents to draw themselves.

*********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

@1   KBM    15 Aug 96
   Changed include file to a .H from .h for C++.

@2   KBM    15 Aug 96
   Moved common items found in OperatorObject and StreamObject
      up to GraphObject along with the common methods.

@3   KBM    15 Aug 96
   Methods renamed:  write_to_disk   -> to_psdl
                     build_from_disk -> from_psdl

@4   KBM    15 Aug 96
   Added local new and delete operators for troubleshooting

@5   KBM    20 Aug 96
   Created a new function over() which is just like hit()
      except that it does not select objects

@6   KBM    20 Aug 96
   Created a new function over_handle() which is just like
      hit_handle() except that it does not select objects

@7   KBM    26 Aug 96
   Problem found with g++ where the _color_table[] was
      exceeded.  Not sure of correct solution.  Incremented
   the size of the table seems consistant.

@8   KBM    30 Aug 96
   Removed set_constraint.

*********************************************************
*/
#ifndef graph_object_h
#define graph_object_h 1
```

```
#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include "ge_defs.h"
#include "font_table.H"                                    //@1

class GraphObjectList;

class GraphObject {
protected:
  static Display *_display_ptr;
  static  GC _graphics_context, _erase_context, _dotted_context;
  static Window _draw_window;
  static Pixmap *_drawing_area_pixmap;
  static unsigned long _color_table[MAXCOLORS+1];                  //@7
  static FontTable *_font_table;
  static int _default_font;
  static Widget _error_tgt;


  GraphObject *_next_ptr;

  BOOLEAN _is_selected;                                  //@2
  BOOLEAN _name_handles_drawn;                                 //@2
  BOOLEAN _name_selected;                                //@2
  int    _handle_selected;                             //@2
  int    _name_width;                                  //@2
  int    _name_height;                                 //@2

//-------------------------------------------------------------  //@2
// GraphObject

  int
      _id;              /* cannot be changed after creation */

  /* info for house keeping */
  BOOLEAN
      _is_deleted,
      _is_new,      /* _is_new is set by the GE and
                              cleared by the SDE' New_Operator procedure
*/
      _is_modified;

  /* info about the label */
  char
          *_name_ptr;
  int
      _name_font,
      _name_x,       /* actual x_position = _x + _name_x */
      _name_y;       /* actual y_position = _y + _name_y */

//-------------------------------------------------------------  //@2


public:

  static Dimension window_width, window_height;
  static BOOLEAN valid_num_string(char *num);
  static void set_draw_environ(Display *display_ptr,
                    GC graphics_context,
                    GC erase_context,
                    GC dotted_context,
                    Window draw_window,
                    Pixmap *drawing_area_pixmap,
                    unsigned long color_table[],
                    Dimension width,
                    Dimension height);
  static void font_init(Display *display_ptr);
  static void set_default_font(int font_id)
```

```
                    {_default_font = font_id;}
 static int font_text_width(int font_id, char *in_string);
 static int font_text_height(int font_id);
 static char *font_name(int font_id)
             {return _font_table->font_name(font_id);}
 static void set_font(GC graphics_context, int font_id);
 static void set_error_tgt(Widget widget) {_error_tgt = widget;}
 static void error_box(char *error_message);

 GraphObject();
 ~GraphObject() {delete _next_ptr;}

#ifdef GE_DEBUG
  void* operator new(size_t size) {                    //@4
   cout << "graph_object new" << endl;                  //@4
   return ::operator new(size); }                      //@4
  void  operator delete(void* s) {                     //@4
   cout << "graph_object delete" << endl;               //@4
   ::operator delete(s); }                             //@4
#endif

  void link(GraphObject &next_object) {_next_ptr = &next_object;}
  GraphObject* next() {return _next_ptr;}
  virtual GE_STATUS from_psdl(FILE *) {return FAILED;}
//@3
  virtual GE_STATUS to_psdl(FILE *) {return FAILED;}
//@3
  virtual GE_STATUS build_from_property();
  virtual GE_STATUS write_to_property() {return FAILED;}
  virtual void draw(DRAW_STYLE) {}
  virtual void select() {}
  virtual void unselect() {}
  void erase();                                        //@2
  void erase_text();                                   //@2
  virtual void clearObj() {}
  virtual BOOLEAN hit(int, int) {return FALSE;}
  virtual BOOLEAN over(int, int) {return FALSE;}                    //@5
  virtual CLASS_DEF is_a() {return GRAPHOBJECT;}
  virtual void set_object_ptrs(GraphObjectList *) {}
  int id() {return _id;}                                 //@2
  char *name() {return _name_ptr;}                       //@2
  virtual void set_location(int, int) {}
  void set_deleted() {_is_deleted = TRUE;}               //@2
  virtual void delete_notify(CLASS_DEF, int) {}
  virtual void replace_name(char *) {}
  // virtual void set_constraint(int) {}
  void set_modified() {_is_modified = TRUE;}             //@2
  virtual BOOLEAN text_selected() {return FALSE;}
  Display *display_ptr() {return _display_ptr;}
  GC graphics_context() {return _graphics_context;}
  Window draw_window() {return _draw_window;}
  Pixmap *drawing_area_pixmap() {return _drawing_area_pixmap;}
  virtual void move_notify(CLASS_DEF, int) {}
  virtual void move_handle(int, int) {}
  virtual BOOLEAN hit_handle(int, int) {return FALSE;}
  virtual BOOLEAN over_handle(int, int) {return FALSE;}
//@6
  virtual void set_color(COLOR) {}
  virtual void set_object_font(int) {};
  virtual void move_text(int, int) {}
  virtual void draw_text(DRAW_STYLE) {}
  virtual BOOLEAN is_deleted() {return FALSE;}
  virtual void undelete_notify(CLASS_DEF, int) {}
  virtual void reset_handles_drawn_state() {}
  virtual int text_width() {return 0;}
  virtual int text_height() {return 0;}
  virtual void text_locate(int, int) {}
  virtual int constraint() {return 0;}
};
```

```
#endif;
```

```
/* ********************************************************

Name:        graph_object.C
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 11 Sep 92
Remarks:  Implementation for the GraphObject class.

      The GraphObject is the base class for the main graph
     objects displayed in the graph editor. It is not
     designed to be directly instantiated, so most of its
     methods are virtual.

Credits:   Portions of code are adapted from the following:

      Barakati, Naba, X Window System Programming, SAMS,
       1991.

      Heller, Dan, Motif Programming Manual, O'Reilly and
       Associates, 1991.

      Johnson, Eric, and Reichard, Kevin, X Window
       Applications Programming, MIS Press, 1989.

      Young, Douglas, Object Oriented Programming With C++
       and OSF/Motif, Prentice-Hall, 1992.


********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
     Changed include file to a .H from .h for C++.

 @2   KBM    15 Aug 96
     Moved common methods up from OperatorObject and
     StreamObject.

 @3   KBM    26 Aug 96
     Corrected _color_table index size to match graph_editor.

 @4   WH     31 Aug 96
     Added standardized comments for methods

********************************************************
*/

#include <Xm/MessageB.h>
#include <stream.h>
#include "graph_object.H"                    //@1

// Initializers for the static class variables.

int GraphObject::_default_font = 0;
Widget GraphObject::_error_tgt = NULL;

// K. Moeller 8/2/96
// added definitions for the class static variables
Window GraphObject::_draw_window;
Display *GraphObject::_display_ptr;
GC GraphObject::_dotted_context;
GC GraphObject::_erase_context;
GC GraphObject::_graphics_context;
Pixmap *GraphObject::_drawing_area_pixmap;
FontTable *GraphObject::_font_table;
unsigned long GraphObject::_color_table[MAXCOLORS+1];
//@3
Dimension GraphObject::window_width;
```

```
Dimension GraphObject::window_height;

/*******************************************************
*  Determines whether the string represents a valid number.
*******************************************************/

BOOLEAN GraphObject::valid_num_string(char *num) {
 int index, num_length;

 if(num != NULL) {
  num_length = strlen(num);
  if(num_length > 0) {
   for(index = 0; index < num_length; index++) {
    if((num[index] != '\n') &&
      (num[index] != '-') &&
      ((num[index] < '0') || (num[index] > '9')))
     return FALSE;
   }
   return TRUE;
  }
 }
 return FALSE;
}

/*******************************************************
*  Displays an error message in a standard Motif error
* dialog box.
*******************************************************/

void GraphObject::error_box(char *error_message) {
 static Widget error_dialog = NULL;
 Arg arg[1];
 XmString t;

 if(_error_tgt != NULL) {
  if(error_dialog == NULL) {
   error_dialog = XmCreateMessageDialog(_error_tgt, "error",
                                        arg, 0);
   XtVaSetValues(XtParent(error_dialog),
                 XtNtitle, "Error",
                 NULL);
   XtUnmanageChild(XmMessageBoxGetChild(error_dialog,
                                        XmDIALOG_HELP_BUTTON));
  }
  t = XmStringCreateSimple(error_message);
  XtVaSetValues(error_dialog,
                XmNmessageString, t,
                NULL);
  XmStringFree(t);
  XtManageChild(error_dialog);
 }
 else
  cout <<
   "Error Target Widget must be set by calling module.\n"
    << endl;
}

/*******************************************************
* graphobject constructor
*******************************************************/

GraphObject::GraphObject() {
 _next_ptr = NULL;
}

/*******************************************************
*  Sets up the drawing environment for the graph objects.
*******************************************************/

void GraphObject::set_draw_environ(Display *display_ptr,
```

```
                    GC graphics_context,
                    GC erase_context,
                    GC dotted_context,
                    Window draw_window,
                    Pixmap *drawing_area_pixmap,
                    unsigned long color_table[],
                    Dimension width,
                    Dimension height) {
 int i;

 _display_ptr = display_ptr;
 _graphics_context = graphics_context;
 _erase_context = erase_context;
 _dotted_context = dotted_context;
 _draw_window = draw_window;
 _drawing_area_pixmap = drawing_area_pixmap;
 for(i = 0; i <= MAXCOLORS; i++)
   _color_table[i] = color_table[i];
 GraphObject::window_width = width;
 GraphObject::window_height = height;
}

/*********************************************************
*  Initializes the font table.
*********************************************************/

void GraphObject::font_init(Display *display_ptr) {

  _font_table = new FontTable();
  _font_table->init(display_ptr);
}

/*********************************************************
*  Returns the width in pixels of the requested font.
*********************************************************/

int GraphObject::font_text_width(int font_id, char *in_string) {

 return _font_table->width(font_id, in_string);
}

/*********************************************************
*  Returns the height in pixels of the requested font.
*********************************************************/

int GraphObject::font_text_height(int font_id) {

 return _font_table->height(font_id);
}

/*********************************************************
*  Sets the requested font as the drawing font.
*********************************************************/

void GraphObject::set_font(GC graphics_context, int font_id) {

  XSetFont(_display_ptr, graphics_context,
        _font_table->font_id(font_id));
}

/*********************************************************
*  build from disk property
*********************************************************/

GE_STATUS GraphObject::build_from_property() {return FAILED;}


/*********************************************************
*  erace object
*********************************************************/
```

```
void GraphObject::erase() {                    //@2

 draw(ERASE);
}

/*********************************************************
* erase text
*********************************************************/
void GraphObject::erase_text() {               //@2

 draw_text(ERASE);
}


/*********************************************************
*              end of code for graph_object.C
*********************************************************/
```

```
/* ********************************************************

Name:        graph_object_list.h
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 11 Sep 92
Remarks:  Specification for the GraphObjectList class.

      A GraphObjectList is a linked list of graphic
      objects, implemented as GraphObjects. Currently,
      a GraphObject may be either an OperatorObject or
      a StreamObject.

      Although not currently implemented this way, most
      of the functionality necessary to draw the graph title
      on the drawing canvas is present.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
    Changed include file to a .H from .h for C++.

 @2   KBM    15 Aug 96
    Added items out of GRAPH_DESC_NODE from ge_interface.h

 @3   KBM    15 Aug 96
    Renamed methods:     write_to_disk  -> to_psdl
                         build_from_disk -> from_psdl

 @4   KBM    15 Aug 96
    Added local new and delete operators for debugging

 @5   KBM    20 Aug 96
    Created a new function over() which is just like hit()
        except that it does not select objects

********************************************************
*/
#ifndef graph_object_list_h
#define graph_object_list_h 1

#include <X11/Intrinsic.h>
#include "ge_defs.h"
#include "ge_interface.h"                          //@2
#include "ge_utilities.H"
#include "graph_object.H"                          //@1
#include "llist.H"

class GraphObject;

class GraphObjectList {
 protected:
  GraphObject *_head_ptr;
  int _last_op_id, _last_stream_id, _name_font, _name_x,
   _name_y, _name_width, _name_height;
  Display *_display_ptr;
  Window _draw_window;
  GC _graphics_context, _erase_context, _dotted_context;
  Pixmap *_drawing_area_pixmap;
  Dimension _width, _height;
  BOOLEAN _title_selected;
  char *_name_ptr;
  Widget _error_tgt;
  BOOLEAN _psdl_modified;

  LList* _timer_list;                              //@2
```

```
  char* _edited_op_name;    /* name of the current operator being
edited */
  char* _parent_op_name;    /* name of the parent of the current
operator */
  char* _edited_op_spec;    /* PSDL specification of the current
operator */
  char* _graph_informal_desc;


  void error_box(char *error_message);
  static BOOLEAN valid_num_string(char *num);
  void set_text_dimensions();
  void set_name_location();
  void draw_handles(GC draw_context, int x1, int y1, int x2,
            int y2);

 public:
  GraphObjectList();
  ~GraphObjectList() {delete _head_ptr;delete _name_ptr;}

#ifdef GE_DEBUG
  void* operator new(size_t size) {                    //@4
   cout << "graph_object_list new" << endl;
   return ::operator new(size); }
  void  operator delete(void* s) {                     //@4
   cout << "graph_object_list delete" << endl;
   ::operator delete(s); }
#endif

  void set_psdl_modified()   { _psdl_modified = TRUE; }
  void reset_psdl_modified() { _psdl_modified = FALSE; }
  BOOLEAN psdl_modified()    { return _psdl_modified; }
  GE_STATUS from_psdl();                               //@3
  GE_STATUS to_psdl();                                 //@3
  GE_STATUS build_from_property();
  GE_STATUS write_to_property();
  void draw();
  void clearGraph();
  void draw_text(GC draw_context);
  void erase_text();
  void move_text(int x, int y);
  GraphObject* hit(int x, int y);
  GraphObject* over(int x, int y);                     //@5
  CLASS_DEF is_a() {return GRAPHOBJECTLIST;}
  GraphObject* target_object(CLASS_DEF object_type, int id);
  void delete_notify(CLASS_DEF class_type, int deleted_obj_id);
  int request_id(CLASS_DEF class_type);
  void add(GraphObject *new_object_ptr);
  void set_draw_environ(Display *display_ptr,
            GC graphics_context, GC erase_context,
            GC dotted_context, Window draw_window,
            Pixmap *drawing_area_pixmap,
            unsigned long color_table[],
            Dimension width, Dimension height);
  void get_del_op_list(char *del_op_str[], int del_op_id[],
            int &num_del_ops);
  void set_undeleted(CLASS_DEF class_type, int id);
  void set_default_font(int font_id);
  void move_notify(CLASS_DEF object_type, int object_id);
  char *font_name(int font_id)
          {return GraphObject::font_name(font_id);}
  BOOLEAN hit_text(int x, int y);
  void set_text_font(int font_id);
  void select_title();
  void unselect_title();
  char *name() {return _name_ptr;}
  void set_error_tgt(Widget widget);

  char *parent_op_name() { return _parent_op_name; }
  char *edited_op_name() { return _edited_op_name; }
```

```
    char *edited_op_spec() { return _edited_op_spec; }
    char *graph_informal_desc() { return _graph_informal_desc; }

    void parent_op_name(char *ptr);
    void edited_op_name(char *ptr);
    void edited_op_spec(char *ptr);
    void graph_informal_desc(char *ptr);
};

#endif
```

```
/* ********************************************************

Name:        graph_object_list.C
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 11 Sep 92
Remarks:  Implementation of a GraphObjectList.

      A GraphObjectList is a linked list of graphic
     objects, implemented as GraphObjects. Currently,
     a GraphObject may be either an OperatorObject or
     a StreamObject.

      Although not currently implemented this way, most
     of the functionality necessary to draw the graph title
     on the drawing canvas is present.


Credits:  Portions of code are adapted from the following:

      Barakati, Naba, X Window System Programming, SAMS,
       1991.

      Heller, Dan, Motif Programming Manual, O'Reilly and
       Associates, 1991.

      Johnson, Eric, and Reichard, Kevin, X Window
       Applications Programming, MIS Press, 1989.

      Young, Douglas, Object Oriented Programming With C++
       and OSF/Motif, Prentice-Hall, 1992.

Changes:
    9/94   add synchronization code to the from_psdl() routine
           so that it will creates the gedatatransfile.lock file before
           reading from the gedatatransfile.txt file and remove the
           gedatatransfile.lock file when the reading is completed.

           codes are also added to the store_data() routine in the
           ge_interface.c file so that it will only write to
           gedatatransfile.txt file if the gedatatransfile.lock
           does not exist in current directory.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
      Changed include file to a .H from .h for C++.

 @2   KBM    15 Aug 96
      gcc compiler did not like syntax, used NULL_VALUE not NULL

 @3   KBM    15 Aug 96
      to_psdl and from_psdl changes

 @4   KBM    20 Aug 96
      Created a new function over() which is just like hit()
           except that it does not select objects

 @5   KBM     2 Sep 96
      Convereted to_psdl and from_psdl to ge_interface.h.

********************************************************
*/
#include <stdio.h>
#include <string.h>
#include <stream.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <Xm/MessageB.h>
#include "graph_object_list.H"                    //@1
#include "operator_object.H"                      //@1
#include "stream_object.H"                        //@1
#include "ge_defs.h"          // grab NULL_VALUE definition  //@2
#include "ge_interface.h"                         //@5

extern GRAPH_DESC_NODE* current_graph_ptr;                //@5

//  Determines whether the input string represents a valid
// number.

BOOLEAN GraphObjectList::valid_num_string(char *num) {
 int index, num_length;

 if(num != NULL) {
  num_length = strlen(num);
  if(num_length > 0) {
   for(index = 0; index < num_length; index++) {
    if((num[index] != '\n') &&
      (num[index] != '-') &&
      ((num[index] < '0') || (num[index] > '9')))
     return FALSE;
   }
   return TRUE;
  }
 }
 return FALSE;
}

//  Displays a standard Motif error dialog box with the
// given error message.

void GraphObjectList::error_box(char *error_message) {
 static Widget error_dialog = NULL;
 Arg arg[1];
 XmString t;

 if(_error_tgt != NULL) {
  if(error_dialog == NULL) {
   error_dialog = XmCreateMessageDialog(_error_tgt, "error",
                                        arg, 0);
   XtVaSetValues(XtParent(error_dialog),
            XtNtitle, "Error",
            NULL);
   XtUnmanageChild(XmMessageBoxGetChild(error_dialog,
                                XmDIALOG_HELP_BUTTON));
  }
  t = XmStringCreateSimple(error_message);
  XtVaSetValues(error_dialog,
            XmNmessageString, t,
            NULL);
  XmStringFree(t);
  XtManageChild(error_dialog);
 }
 else
  cout <<
   "Error Target Widget must be set by calling module.\n"
    << endl;
}

GraphObjectList::GraphObjectList() {

 _head_ptr = NULL;
 _name_ptr = NULL;
 _last_op_id = 0;
 _last_stream_id = 0;
 _title_selected = FALSE;
```

```
 _error_tgt = NULL;
}

GE_STATUS GraphObjectList::build_from_property() {return
FAILED;}

//  Builds the GraphObjectList from disk.

GE_STATUS GraphObjectList::from_psdl() {                    //@3
 char buffer[INPUT_LINE_SIZE +1], *error_str_ptr;
 GraphObject *temp_object_ptr, *current_ptr;
 GE_STATUS status = SUCCEEDED;

#ifdef GE_DEBUG
 cout << "GraphObjectList::from_psdl" << endl;
#endif

 GRAPH_DESC_NODE* cgp = current_graph_ptr;        // Name
getting too long

 _last_op_id    = 0;
 _last_stream_id = 0;

 // Remove all old data
 clearGraph();

 if (cgp == NULL) {
  error_str_ptr = strdup("NULL value for GRAPH_DESC");
  status = FAILED;
 }
 else {
  // Make duplicates of char*
  _parent_op_name     = make_str(cgp->parent_op_name);
  _edited_op_name     = make_str(cgp->edited_op_name);
  _edited_op_spec     = make_str(cgp->edited_op_spec);
  _graph_informal_desc = make_str(cgp->graph_informal_desc);

  // Copy _edited_op_name over to GraphObjectList::_name_ptr
  if (_edited_op_name == NULL)
   _name_ptr = strdup("Noname");
  else
   _name_ptr = strdup(_edited_op_name);

   //?? _name_font
   //?? _name_x
   //?? _name_y
   //?? set_text_dimentions();
   //?? set_name_location();

  // Process Operators
  OP_LIST op = cgp->operator_list;
  while ((op != NULL) && (status == SUCCEEDED)) {
   temp_object_ptr = new OperatorObject(op->op);  // NULL indicates
failure
#ifdef GE_DEBUG
 cout << "from_psdl: operator  name: " << temp_object_ptr->name()
    << "  id: " << temp_object_ptr->id() << endl;
#endif
    op = op->next;

    if (temp_object_ptr == NULL) {
         status = FAILED;
         //?? Free up memory created by new
    }
    else {
         // Running count of last op number
         if (temp_object_ptr->id() > _last_op_id)
          _last_op_id = temp_object_ptr->id();

         // Now insert it into linked list
```

```
        if (_head_ptr == NULL) {              // At head
          _head_ptr   = temp_object_ptr;
          current_ptr = _head_ptr;
        }
        else {
          current_ptr->link(*temp_object_ptr);
          current_ptr = current_ptr->next();
        }
    }
  }

  // Process Streams
  ST_LIST st = cgp->stream_list;
  while ((st != NULL) && (status == SUCCEEDED)) {
    temp_object_ptr = new StreamObject(st->st);   // NULL indicates
failure
    st = st->next;

    if (temp_object_ptr == NULL) {
          status = FAILED;
          // Free up memory created by new
    }
    else {
          // Running count of last stream number
          if(temp_object_ptr->id() > _last_stream_id)
            _last_stream_id = temp_object_ptr->id();

          // Set up pointers to objects
          temp_object_ptr->set_object_ptrs(this);

          // Now insert it into linked list
          if(_head_ptr == NULL) {
            _head_ptr = temp_object_ptr;
            current_ptr = _head_ptr;
          }
          else {
            current_ptr->link(*temp_object_ptr);
            current_ptr = current_ptr->next();
          }
#ifdef GE_DEBUG
  StreamObject *temp_st_ptr = (StreamObject*) temp_object_ptr;
  cout << "from_psdl: stream  name: " << temp_object_ptr->name()
       << " id: "   << temp_object_ptr->id()
       << " from: " << temp_st_ptr->from()
       << " to: "   << temp_st_ptr->to() << endl;
#endif
    }
  }

  // Process timers
  if (cgp->timer_list == NULL)
    _timer_list = NULL;
  else {
    _timer_list = new LList;
    _timer_list->sort_readID_LIST(cgp->timer_list);
    }
  }

  // Process any errors
  if (status == FAILED) {
    error_box(error_str_ptr);
    free(error_str_ptr);
  }

  // psdl is no longer modified
  reset_psdl_modified();

  return status;
}
```

```
GE_STATUS GraphObjectList::write_to_property() {return FAILED;}

//  Writes the GraphObjectList to disk

GE_STATUS GraphObjectList::to_psdl() {                    //@3
 GraphObject *obj_ptr = _head_ptr;
 OperatorObject *temp_op;
 StreamObject   *temp_st;

#ifdef GE_DEBUG
 cout << "GraphObjectList::to_psdl" << endl;
#endif

 OP_LIST OP = NULL, prevOP = NULL;
 ST_LIST ST = NULL, prevST = NULL;

 GE_STATUS status = SUCCEEDED;
 char buffer[INPUT_LINE_SIZE + 1];

 GRAPH_DESC_NODE* cgp = current_graph_ptr;        // Name
getting too long

 // First clear out old information, just in case
 graph_desc_clear(cgp);

 // Make duplicates of char*
 cgp->parent_op_name     = make_str(_parent_op_name);
 cgp->edited_op_name     = make_str(_edited_op_name);
 cgp->edited_op_spec     = make_str(_edited_op_spec);
 cgp->graph_informal_desc = make_str(_graph_informal_desc);


 while (obj_ptr != NULL) {
  if (obj_ptr->is_a() == OPERATOROBJECT) {
   // Create a new OP_LIST entry plus a new OP_NODE
   OP     = (OP_LIST) malloc(sizeof(OP_LIST_NODE));
   OP->op  = (OPERATOR) malloc(sizeof(OP_NODE));
   initOP(OP);

   // Write the data
   temp_op = (OperatorObject*) obj_ptr;
   temp_op->copy_back(OP->op);

#ifdef GE_DEBUG
 cout << "to_psdl: operator  name: " << OP->op->label
    << "  id: " << OP->op->id << endl;
#endif

   // Link it into linked list
   if (prevOP == NULL) {
        cgp->operator_list = OP;
        prevOP = OP;
   }
   else {
        prevOP->next = OP;
        prevOP = OP;
   }
  } else {  // is_a() == STREAMOPERATOR
   ST     = (ST_LIST) malloc(sizeof(ST_LIST_NODE));
   ST->st  = (STREAM) malloc(sizeof(ST_NODE));
   initST(ST);

   // Write the data
   temp_st = (StreamObject*) obj_ptr;
   temp_st->copy_back(ST->st);
   ST->st->from = operator_id_ptr(cgp->operator_list, temp_st-
>from());
   ST->st->to   = operator_id_ptr(cgp->operator_list, temp_st->to());

#ifdef GE_DEBUG
```

```
 cout << "to_psdl: stream  name: " << ST->st->label
     << "  id: " << ST->st->id << endl;
 //      << "  from: " << ST->st->from->id        fails for external
 //      << "  to: " << ST->st->to->id << endl;
#endif

    // Link it into linked list
    if (prevST == NULL) {
          cgp->stream_list = ST;
          prevST = ST;
    }
    else {
          prevST->next = ST;
          prevST = ST;
    }
   }
   obj_ptr = obj_ptr->next();
 }

 // Process timers
 if (_timer_list == NULL)
   cgp->timer_list = NULL;
 else
   _timer_list->writeID_LIST(cgp->timer_list);

 return status;
}

//  Draws the GraphObjectList to the drawing canvas.

void GraphObjectList::draw() {
 GraphObject *temp_object_ptr = _head_ptr;

 XFillRectangle(_display_ptr, _draw_window, _erase_context, 0,
          0, _width, _height);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          _erase_context, 0, 0, _width, _height);

// Streams are drawn first to prevent the operators' handles
// from overwriting the arrowheads. Handles are drawn in xor
// mode so they will erase properly.

//  Since handles are drawn in xor mode, it's important to
// track whether they've been drawn or not. Redrawing the
// handles unintentionally erases them.

 while(temp_object_ptr != NULL) {
  if(temp_object_ptr->is_a() == STREAMOBJECT) {
   temp_object_ptr->reset_handles_drawn_state();
   temp_object_ptr->draw(SOLID);
  }
  temp_object_ptr = temp_object_ptr->next();
 }
 temp_object_ptr = _head_ptr;
 while(temp_object_ptr != NULL) {
  if(temp_object_ptr->is_a() == OPERATOROBJECT) {
   temp_object_ptr->reset_handles_drawn_state();
   temp_object_ptr->draw(SOLID);
  }
  temp_object_ptr = temp_object_ptr->next();
 }
}

void GraphObjectList::clearGraph() {
 GraphObject *obj_ptr = _head_ptr;
 GraphObject *nextObj;

 while (obj_ptr != NULL) {
  nextObj = obj_ptr->next();
  obj_ptr->clearObj();
```

```
   delete obj_ptr;
   obj_ptr = nextObj;
  }
  _head_ptr = NULL;
}

// If the given coordinates are located inside one of the
// graph objects or their text strings, the function returns
// a pointer to the object.

GraphObject* GraphObjectList::hit(int x, int y) {
 GraphObject *temp_object_ptr = _head_ptr;

 while(temp_object_ptr != NULL) {
  if(temp_object_ptr->hit(x, y))
   return temp_object_ptr;
  else
   temp_object_ptr = temp_object_ptr->next();
 }
 return (GraphObject *) NULL;
}

// If the given coordinates are located inside one of the
// graph objects or their text strings, the function returns
// a pointer to the object. But does not select anything.

GraphObject* GraphObjectList::over(int x, int y) {          //@4
 GraphObject *temp_object_ptr = _head_ptr;

 while(temp_object_ptr != NULL) {
  if(temp_object_ptr->over(x, y))
   return temp_object_ptr;
  else
   temp_object_ptr = temp_object_ptr->next();
 }
 return (GraphObject *) NULL;
}

// Returns a pointer to the requested object.

GraphObject* GraphObjectList::target_object(
                    CLASS_DEF object_type, int id) {
 GraphObject* temp_object_ptr = _head_ptr;
 while(temp_object_ptr != NULL) {
  if((temp_object_ptr->is_a() == object_type) &&
    (temp_object_ptr->id() == id)) {
   return temp_object_ptr;
  }
  temp_object_ptr = temp_object_ptr->next();
 }
 error_box("Requested operator id not found");
 return NULL;
}

// Notifies all the objects that one of them has been
// deleted, so that they may take appropriate actions.

void GraphObjectList::delete_notify(CLASS_DEF class_type,
                    int deleted_obj_id) {
 GraphObject *temp_object_ptr = _head_ptr;

 while(temp_object_ptr != NULL) {
  temp_object_ptr->delete_notify(class_type, deleted_obj_id);
  temp_object_ptr = temp_object_ptr->next();
 }
}

// When a new object is added to the list, it needs a
// unique identifier. The GraphObjectList tracks the
// highest stream and operator identifiers and issues the
```

```
// next higher on request.

int GraphObjectList::request_id(CLASS_DEF class_type) {

 if(class_type == OPERATOROBJECT) {
  _last_op_id++;
  return _last_op_id;
 }
 else {
  _last_stream_id++;
  return _last_stream_id;
 }
}

//  Adds a new GraphObject to the list.

void GraphObjectList::add(GraphObject *new_object_ptr) {
 GraphObject *temp_object_ptr = _head_ptr;

 if(_head_ptr == NULL)
  _head_ptr = new_object_ptr;
 else
  if(_head_ptr->next() == NULL)
   _head_ptr->link(*new_object_ptr);
  else {
   while(temp_object_ptr->next() != NULL)
    temp_object_ptr = temp_object_ptr->next();
   temp_object_ptr->link(*new_object_ptr);
  }
}

//  Sets the necessary drawing variables, and performs the
// same operation for the GraphObject class.

void GraphObjectList::set_draw_environ(Display *display_ptr,
        GC graphics_context, GC erase_context,
        GC dotted_context, Window draw_window,
        Pixmap *drawing_area_pixmap,
        unsigned long color_table[], Dimension width,
        Dimension height) {

 _display_ptr = display_ptr;
 _graphics_context = graphics_context;
 _erase_context = erase_context;
 _dotted_context = dotted_context;
 _draw_window = draw_window;
 _drawing_area_pixmap = drawing_area_pixmap;
 _width = width;
 _height = height;
 GraphObject::set_draw_environ(_display_ptr, _graphics_context,
        _erase_context, _dotted_context, _draw_window,
        _drawing_area_pixmap, color_table, width, height);
 GraphObject::font_init(_display_ptr);
}

//  Notifies the objects that one of them has been moved.

void GraphObjectList::move_notify(CLASS_DEF object_type,
                int object_id) {
 GraphObject *temp_object_ptr = _head_ptr;

 while(temp_object_ptr != NULL) {
  temp_object_ptr->move_notify(object_type, object_id);
  temp_object_ptr = temp_object_ptr->next();
 }
}

//  Sets the default font.

void GraphObjectList::set_default_font(int font_id) {
```

```
GraphObject::set_default_font(font_id);
}

// Sets the dimensions for the title string so it can be
// drawn on the canvas. Left in for future use.

void GraphObjectList::set_text_dimensions() {

 if(_name_ptr == NULL) {
  _name_width = 0;
  _name_height = 0;
 }
 else {
  _name_width = GraphObject::font_text_width(_name_font,
_name_ptr);
  _name_height = GraphObject::font_text_height(_name_font);
 }
}

// Sets a default name location. Left in for future use.

void GraphObjectList::set_name_location() {

 _name_y = 100;
 _name_x = (_width - _name_width) / 2;
}

// Draws the title string on the canvas. Left in for
// future use.

void GraphObjectList::draw_text(GC draw_context) {

 if(_name_ptr != NULL) {
  GraphObject::set_font(draw_context, _name_font);
  XDrawString(_display_ptr, _draw_window, draw_context, _name_x,
        _name_y, _name_ptr, strlen(_name_ptr));
  XDrawString(_display_ptr, *_drawing_area_pixmap, draw_context,
        _name_x, _name_y, _name_ptr, strlen(_name_ptr));
  if(_title_selected)
   draw_handles(draw_context, _name_x - HANDLESIZE,
        _name_y - _name_height - HANDLESIZE,
        _name_x + _name_width + HANDLESIZE,
        _name_y + HANDLESIZE);
 }
}

// Erases the title string. Left in for future use.

void GraphObjectList::erase_text() {

 draw_text(_erase_context);
}

// Moves the title string. Left in for future use.

void GraphObjectList::move_text(int x, int y) {

 if(_title_selected) {
  _name_x += x;
  _name_y += y;
 }
}

// Returns TRUE if the mouse clicked on the title string,
// FALSE otherwise.

BOOLEAN GraphObjectList::hit_text(int x, int y) {

 if(((x >= _name_x) && (x <= (_name_x + _name_width))) &&
```

```
   (y >= _name_y - _name_height) && (y <= _name_y)) {
  _title_selected = TRUE;
  return TRUE;
 }
 else
  return FALSE;
}

// Draws a handle on the graph.

void GraphObjectList::draw_handles(GC draw_context, int x1,
                        int y1, int x2, int y2) {

 XFillRectangle(_display_ptr, _draw_window, draw_context, x1,
          y1, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, _draw_window, draw_context,
          x2 - HANDLESIZE, y1, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, _draw_window, draw_context, x1,
          y2 - HANDLESIZE, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, _draw_window, draw_context,
          x2 - HANDLESIZE, y2 - HANDLESIZE, HANDLESIZE,
          HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x1, y1, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x2 - HANDLESIZE, y1,
          HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x1, y2 - HANDLESIZE, HANDLESIZE,
          HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x2 - HANDLESIZE, y2 - HANDLESIZE,
          HANDLESIZE, HANDLESIZE);
}

// Sets the name font to the desired value. Left in for
// future use.

void GraphObjectList::set_text_font(int font_id) {

 _name_font = font_id;
 set_text_dimensions();
}

// Unselects the title.

void GraphObjectList::unselect_title() {

 erase_text();
 _title_selected = FALSE;
 draw_text(_graphics_context);
}

// Selects the title.

void GraphObjectList::select_title() {

 erase_text();
 _title_selected = TRUE;
 draw_text(_graphics_context);
}

// Makes a list of deleted operators.

void GraphObjectList::get_del_op_list(char *del_op_str[],
                        int del_op_id[],
                        int &num_del_ops) {
 GraphObject *temp_obj_ptr = _head_ptr;
 int index = 0;
```

```
 while((temp_obj_ptr != NULL) && (index < MAXDELETEDOPS)) {
  if((temp_obj_ptr->is_a() == OPERATOROBJECT) &&
    (temp_obj_ptr->is_deleted())) {
     del_op_str[index] = strdup(temp_obj_ptr->name());
     del_op_id[index] = temp_obj_ptr->id();
     index++;
   }
   temp_obj_ptr = temp_obj_ptr->next();
 }
 num_del_ops = index;
}

// Notifies the objects that the given object has been
// undeleted.

void GraphObjectList::set_undeleted(CLASS_DEF class_type,
                        int id) {
 GraphObject *temp_obj_ptr = _head_ptr;

 while(temp_obj_ptr != NULL) {
  temp_obj_ptr->undelete_notify(class_type, id);
  temp_obj_ptr = temp_obj_ptr->next();
 }
}

// Sets the widget used to display the error message box.

void GraphObjectList::set_error_tgt(Widget widget) {

 _error_tgt = widget;
 GraphObject::set_error_tgt(widget);
 SplineObject::set_error_tgt(widget);
 FontTable::set_error_tgt(widget);
}
```

```
void GraphObjectList::parent_op_name(char *ptr) {

 free(_parent_op_name);
 _parent_op_name = make_str(ptr);
}



void GraphObjectList::edited_op_name(char *ptr) {

 free(_edited_op_name);
 _edited_op_name = make_str(ptr);
}



void GraphObjectList::edited_op_spec(char *ptr) {

 free(_edited_op_spec);
 _edited_op_spec = make_str(ptr);
}



void GraphObjectList::graph_informal_desc(char *ptr) {

 free(_graph_informal_desc);
 _graph_informal_desc = make_str(ptr);
}
```

```
void init_motif() {

 // Simulated arguments
 char* args[] = {"psdl_editor","-geometry","800x600",NULL};
 int signed_argc = 3;
 char** argv = args;

 glXmTopLevel = XtVaAppInitialize(&app, "psdl_editor", options,
                                  XtNumber(options),
&signed_argc, argv,
                                  NULL, NULL);

 display_ptr = XtDisplay(glXmTopLevel);
 XtGetApplicationResources(glXmTopLevel, (XtPointer) &Resrcs,
                           resources, XtNumber(resources),
                           NULL, 0);
 screen_ptr = XtScreen(glXmTopLevel);
 initialize_color_table(screen_ptr);
 root_window = RootWindowOfScreen(screen_ptr);
 gcv1.foreground = BlackPixelOfScreen(screen_ptr);
 gcv1.background = WhitePixelOfScreen(screen_ptr);
 gcv2.foreground = BlackPixelOfScreen(screen_ptr);
 gcv2.background = WhitePixelOfScreen(screen_ptr);
 gcv3.foreground = WhitePixelOfScreen(screen_ptr);
 gcv3.background = WhitePixelOfScreen(screen_ptr);
 gc_mask = GCForeground | GCBackground;
 std_graphics_context = XCreateGC(display_ptr,
                                  root_window, gc_mask, &gcv1);
 dotted_context = XCreateGC(display_ptr,
                            root_window, gc_mask, &gcv2);
 erase_context = XCreateGC(display_ptr, root_window, gc_mask,
                           &gcv3);
 XSetLineAttributes(display_ptr, dotted_context, 1,
                    LineOnOffDash, CapButt, JoinMiter);
 XSetFunction(display_ptr, dotted_context, GXxor);

 if (Resrcs.viewer == False) { // editor mode, not view mode
  main_w = XtVaCreateManagedWidget("main_w",
xmFormWidgetClass,
                                   glXmTopLevel, NULL);
  build_menu_bar(main_w, menubar);
  XtManageChild(menubar);
  rowcol =
   XtVaCreateManagedWidget("rowcol", xmRowColumnWidgetClass,
                           main_w,
                           XmNnumColumns, 1,
                           NULL);

  make_buttons(rowcol, op_button, term_button, stream_button,
               info_button, timer_button,
               op_button_pixmap, term_button_pixmap,
               stream_button_pixmap,
               display_ptr, screen_ptr);

  XtAddCallback(op_button, XmNactivateCallback, op_button_cb,
                NULL);
  XtAddCallback(term_button, XmNactivateCallback,
                term_button_cb, NULL);
  XtAddCallback(stream_button, XmNactivateCallback,
                stream_button_cb, NULL);
  XtAddCallback(timer_button, XmNactivateCallback,
                timer_button_cb, NULL);
  XtAddCallback(info_button, XmNactivateCallback,
                info_button_cb, NULL);
  scrolled_win =
   XtVaCreateManagedWidget("scrolled_win",
                           xmScrolledWindowWidgetClass,
                           main_w,
//                 XmNwidth, 1200,
//                 XmNheight, 750,
```

```
                                XmNscrollingPolicy, XmAUTOMATIC,
                                XmNscrollBarDisplayPolicy,
XmAS_NEEDED,
                                NULL);
   actions.string = "draw";
   actions.proc = draw;
   XtAppAddActions(app, &actions, 1);
   graph_title =
    XtVaCreateManagedWidget("graph_title", xmTextWidgetClass,
                                main_w,
                                XmNvalue, "",
                                NULL);
   tool_indicator = XtVaCreateManagedWidget("tool_indicator",
                                                xmTextWidgetClass,
                                                main_w,
                                                XmNvalue, "Select
Tool",
                                                NULL);
   XtVaSetValues(glXmTopLevel, XmNtitle, "Graph Editor", NULL);
  }
  else {
   scrolled_win =
    XtVaCreateManagedWidget("scrolled_win",
                                xmScrolledWindowWidgetClass,
glXmTopLevel,
//                  XmNwidth, 600,
//                  XmNheight, 400,
                                XmNscrollingPolicy, XmAUTOMATIC,
                                XmNscrollBarDisplayPolicy,
XmAS_NEEDED,
                                NULL);
   XtVaSetValues(glXmTopLevel, XmNtitle, "Graph Viewer", NULL);
  }

  drawing_a =
   XtVaCreateManagedWidget("drawing_a",
                                xmDrawingAreaWidgetClass,
scrolled_win,
                                XmNunitType, Xm1000TH_INCHES,
                                XmNwidth, 11000,
                                XmNheight, 8500,
                                XmNresizePolicy, XmNONE,
                                NULL);
  XtAddCallback(drawing_a, XmNexposeCallback, redraw, NULL);

  XtVaSetValues(drawing_a, XmNunitType, XmPIXELS, NULL);
  XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);
  XtVaGetValues(drawing_a, XmNwidth, &width, XmNheight, &height,
                NULL);
  drawing_area_pixmap = XCreatePixmap(display_ptr,
                                root_window, width, height,

DefaultDepthOfScreen(screen_ptr));
  XFillRectangle(display_ptr, drawing_area_pixmap,
                erase_context, 0, 0, width, height);

  if (Resrcs.viewer == False) {
   XtVaSetValues(drawing_a,  XmNtranslations,
                XtParseTranslationTable(translations), NULL);
   XtVaSetValues(menubar, XmNtopAttachment, XmATTACH_FORM,
                XmNrightAttachment, XmATTACH_FORM,
                XmNleftAttachment, XmATTACH_WIDGET,
                XmNleftWidget, rowcol,
                XmNbottomAttachment, XmATTACH_NONE,
                NULL);

   XtVaSetValues(rowcol, XmNtopAttachment, XmATTACH_FORM,
                XmNrightAttachment, XmATTACH_NONE,
                XmNleftAttachment, XmATTACH_FORM,
                XmNbottomAttachment, XmATTACH_WIDGET,
```

```
                    XmNbottomWidget, tool_indicator,
                    NULL);
  XtVaSetValues(scrolled_win, XmNtopAttachment,
                    XmATTACH_WIDGET,
                    XmNtopWidget, menubar,
                    XmNrightAttachment, XmATTACH_FORM,
                    XmNleftAttachment, XmATTACH_WIDGET,
                    XmNleftWidget, rowcol,
                    XmNbottomAttachment, XmATTACH_WIDGET,
                    XmNbottomWidget, graph_title,
                    NULL);
  XtVaSetValues(graph_title, XmNtopAttachment,
XmATTACH_NONE,
                    XmNrightAttachment, XmATTACH_FORM,
                    XmNleftAttachment, XmATTACH_WIDGET,
                    XmNleftWidget, tool_indicator,
                    XmNbottomAttachment, XmATTACH_FORM,
                    NULL);
  XtVaSetValues(tool_indicator, XmNtopAttachment,
                    XmATTACH_NONE,
                    XmNrightAttachment, XmATTACH_NONE,
                    XmNleftAttachment, XmATTACH_FORM,
                    XmNbottomAttachment, XmATTACH_FORM,
                    NULL);
 }
 XtRealizeWidget(glXmTopLevel);
 draw_window = XtWindow(drawing_a);
 graphic_list.set_draw_environ(display_ptr,
                                    std_graphics_context,
                                    erase_context, dotted_context,
                                    draw_window,
                                    &drawing_area_pixmap,
                                    color_table,
                                    width, height);
 graphic_list.set_error_tgt(drawing_a);
 if (Resrcs.viewer == False)
  set_title();
 XmProcessTraversal(drawing_a, XmTRAVERSE_CURRENT);

 if (Resrcs.viewer != False) {
  toplevel_window = XtWindowOfObject(glXmTopLevel);
  Atom display_id_atom = XInternAtom(display_ptr, "WINDOW_ID",
False);
  XChangeProperty(display_ptr, root_window, display_id_atom,
                    XA_WINDOW, 32, PropModeReplace,
                    (unsigned char *) &toplevel_window, 1);
  XtAddEventHandler(glXmTopLevel, NoEventMask, TRUE,
event_handler, NULL);
 }
}
```

```
/* ********************************************************

Name:        llist.H
Author:      Ken Moeller
Program:     graph_editor
Date Modified: 5 Sep 96
Remarks:     This is a linked list class which supports a
             single-linked list of pointers to char. It also
             provides support for the ID_LIST defined in
             ge_interface.h. Support routines for handling
             ID_LIST are provided in ge_utilities.H.

**********************************************************
Modification History:

 @1   KBM     5 Sep 96
      New file.

 @2   WH      11 Sept 96
      Added Print prototype

**********************************************************
*/


/*********************************************************
 * Implimentation Notes:
 *
 * LList*       LList          ListElm
 * +------+     +-------+      +----+next+----+   +----+
 * |      |------>| head |-------->|   |--->|   |--->|   |-X
 * +------+     | tail |      +----+   +----+   +----+
 *              | iter |      \ item   \        \
 *              +-------+       \ +----+\ +----+\ +----+
 *                              --|str | --|str | --|str |
 *                              +----+   +----+   +----+
 *
 *
 * ID_LIST      ID_NODE     ID_NODE     ID_NODE
 * +------+     +-------+NEXT+-------+   +-------+
 * |      |------>|      |--->|      |--->|      |-X
 * +------+     +-------+   +-------+   +-------+
 *              \ ID        \          \
 *               \ +----+   \ +----+   \ +----+
 *                --|str |    --|str |    --|str |
 *                  +----+     +----+     +----+
 *
 *
 *
 * Function        |return |iter
 *-----------------+-------+-----------------------------------
 * IsEmpty()       |boolean|n/a
 * item()          |char*  |returns item located at iter
 * reset()         |char*  |iter set to head
 * step()          |char*  |next or NULL if stepped off list
 * find()          |boolean|matching string or NULL if not found
 * find_next()     |boolean|matching string or NULL if not found
 * EOL()           |boolean|returns TRUE if iter is NULL
 * clear()         |void   |deletes all data, iter set to NULL
 * remove()        |void   |removes element at iter, iter points to next
 * insert()        |void   |inserts element in FRONT of iter
 * sorted_insert() |void   |inserts element in alphabetical order
 * append()        |void   |inserts element at tail
 * readID_LIST()   |void   |reads in an ID_LIST
 * sort_readID_LIST()|void  |reads in an ID_LIST and adds in
alphabetical order
 * writeID_LIST()  |void   |writes an LList to an ID_LIST
 *
 *
 * LList is a single linked-list.  Access to the list is through
 * the iter (private) symbol in LList.  iter is altered by several
```

```
* methods provided by LList.  reset() sets iter to the head of
* the list.  step() increments iter to the next item in the linked
* list.  find() is used to locate iter to the element of this list
* which item matches the provided char string.  Note that it will
* only find the first matching element.  Since find() always starts
* at the head of the list, there is no support to locate another
* matching element.  find_next() will locate the next matching
* element.  Also note that find() and find_next() does not return
* a pointer to the element but returns a boolean.  iter is left
* pointing to the element that was found.
*********************************************************/

#ifndef LLIST_H
#define LLIST_H

#include <stddef.h>   // NULL
#include <malloc.h>
#include "ge_defs.h"
#include "ge_interface.h"
#include "ge_utilities.H"

class ListElem {
public:
 ListElem(char *ptr);
 ~ListElem();
 ListElem *next;
 char    *item;

};

class LList {
public:
 LList();
 ~LList();
 BOOLEAN IsEmpty();
 char   *reset();
 char   *step();
 char   *item();
 BOOLEAN find(char *field);
 BOOLEAN find_next(char *field);
 BOOLEAN EOL();
 void   append(char *ptr);
 void   insert(char *ptr);
 void   sorted_insert(char *ptr);
 void   remove();
 void   clear();
 void   writeID_LIST(ID_LIST &ptr);
 void   readID_LIST(ID_LIST ptr);
 void   sort_readID_LIST(ID_LIST ptr);
 void   Print();                                    //@2
 int    length();

private:
 ListElem *head;
 ListElem *tail;
 ListElem *iter;
};

#endif
```

```
/* ********************************************************

Name:        llist.C
Author:      Ken Moeller
Program:     graph_editor
Date Modified: 5 Sep 96
Remarks:     This is a linked list class which supports a
          single-linked list of pointers to char. It also
              provides support for the ID_LIST defined in
              ge_interface.h. Support routines for handling
              ID_LIST are provided in ge_utilities.H.


********************************************************
Modification History:

 @1   KBM      5 Sep 96
     New file.

 @2   WH       11 Sep 96
     Added a Print method to list

********************************************************
*/
#include "llist.H"
#include <stdlib.h>
#include <stream.h>


/*******************************************************
 * ListElem(char *ptr)
 *
 * Constructor of a list element that makes a copy of the
 * supplied char pointer and insertes it into the list element.
 * The next pointer is initialized to NULL.
 *
 * Note:  this procedure does not insert a ListElem into a linked
 * list.  It only creates an element and a copy of the char string.
 *******************************************************/
ListElem::ListElem(char *ptr) {
  item = make_str(ptr);
  next = NULL;
}


/*******************************************************
 * ~ListElem()
 *
 * Destructor of a list element frees up the memory pointed to.
 *******************************************************/
ListElem::~ListElem() {
  free(item);
}


/*******************************************************
 * LList()
 *
 * Constructor of a Linked List.  Creates a linked list and initializes
 * it to an empty list.  Note that no memory is allocated in this
 * constructor.  But must be allocated by calling routine.
 *******************************************************/
LList::LList() {
  head = NULL;
  tail = NULL;
  iter = NULL;
}


/*******************************************************
```

```
* ~LList()
*
* Destructor for Linked List.  Removes all elements from the linked
* list.
*******************************************************/
LList::~LList() {
 clear();
}


/*******************************************************
 * reset()
 *
 * iter is the only means for traversing through the linked list.
 * reset() sets iter back to the head.  If the list is not empty, reset()
 * returns a pointer to the data item at the head of the linked list.
 *******************************************************/
char* LList::reset() {

 iter = head;
 if (iter == NULL)
   return NULL;
 else
   return iter->item;
}


/*******************************************************
 * step()
 *
 * iter is the only means for traversing through the linked list.
 * step() will go to the next element in the linked list or NULL if it
 * steps off the end of the linked list.  If the data item at this
 * new location exists, step() returns a pointer to this data item.
 *******************************************************/
char* LList::step() {

 if (iter != NULL) {
   iter = iter->next;
   if (iter != NULL)
     return iter->item;
   else
     return NULL;
 } else
   return NULL;
}


/*******************************************************
 * find(char *field)
 *
 * Starting at the head of the list, find() will look for an element
 * with a data item of field.  If a matching element is found, find()
 * will return TRUE, else it returns FALSE.  iter is left pointing to
 * the matching element if found or NULL if a matching element was not
 * found.
 *******************************************************/
BOOLEAN LList::find(char *field) {

 iter = head;          // Always start at head of list
 while (iter != NULL) {
   // First test that we do not have a null
   if (field == NULL && iter->item == NULL)
     return TRUE;
   else if (field != NULL && iter->item != NULL)
     if (strcmp(field,iter->item)==0)
           return TRUE;
   iter = iter->next;
 }
 return FALSE;
```

```
}


/*********************************************************
 * find_next(char *field)
 *
 * Starting with the element after the one pointed to by iter, find_next()
 * will look for the next element with a data item of field.  If a matching
 * element is found, find_next() will return TRUE, else it returns FALSE.
 * iter is left pointing to the matching element if found or NULL if a
 * matching element was not found.
 *********************************************************/
BOOLEAN LList::find_next(char *field) {

 step();  // Skip over this element, looking for next match
 while (iter != NULL) {
  // First test that we do not have a null
  if (field == NULL && iter->item == NULL)
   return TRUE;
  else if (field != NULL && iter->item != NULL)
   if (strcmp(field,iter->item)==0)
        return TRUE;
  iter = iter->next;
 }
 return FALSE;
}


/*********************************************************
 * IsEmpty()
 *
 * Returns a boolean.  TRUE if the linked list is empty, FALSE
otherwise.
 *********************************************************/
BOOLEAN LList::IsEmpty() {

 if (head == NULL)
  return TRUE;
 else
  return FALSE;
}


/*********************************************************
 * EOL()
 *
 * Returns a boolean.  TRUE if iter has stepped off the end of the
 * linked list or if the list is empty.  FALSE otherwise.
 *********************************************************/
BOOLEAN LList::EOL() {

 return (iter == NULL);
}


/*********************************************************
 * item()
 *
 * Returns a pointer to the data item which is located at the element
 * pointed to by iter.  If iter points to NULL, NULL is returned.
 *********************************************************/
char* LList::item() {

 if (iter == NULL)
  return NULL;
 else
  return (iter->item);
}
```

```
/*********************************************************
 * append(char *ptr)
 *
 * Adds an element with a data item of ptr to the end of the linked list.
 * A copy of the character string is made from ptr. If the list is empty,
 * head, tail and iter are initialized to the first element. Otherwise,
 * the element is inserted on the tail. iter is set to the new element.
 *********************************************************/
void LList::append(char *ptr) {
 ListElem *newElem = new ListElem(ptr);

 if (IsEmpty()) {
  head = newElem;
  tail = newElem;
 } else {
  tail->next = newElem;
  tail = newElem;
 }
 iter = newElem;
}


/*********************************************************
 * insert(char *ptr)
 *
 * Inserts a new element with a copy of the data item pointed to by ptr
 * into the linked list. The new element is inserted in FRONT of the
 * element pointed to by iter. If iter points to NULL (including an
 * empty list), insert() will append the new element. iter points to
 * new element upon exit.
 *********************************************************/
void LList::insert(char *ptr) {

 if (IsEmpty() || iter == NULL)
  append(ptr);
else {
 // Create a new element to add to linked list
 ListElem *newElem = new ListElem(ptr);

 // Now look for the element in front of iter to set the pointers
 ListElem *prev = head;
 if (prev == iter) {        // Add to head
  newElem->next = head;
  head = newElem;
 }
 else {
  while (prev->next != iter)   // Find the item right before iter
       prev = prev->next;

  newElem->next = iter;
  prev->next = newElem;
 }
 iter = newElem;
}
}


/*********************************************************
 * sorted_insert(char *ptr)
 *
 * Inserts a new element with a copy of the data item pointed to by ptr
 * into the linked list. The new element is inserted in alphabetical
 * order. iter points to the new element upon exit.
 *********************************************************/
void LList::sorted_insert(char *ptr) {
 char *dummy;

 ListElem *search = head;
 if (IsEmpty()) {            // Empty list, add to head
  append(ptr);
```

```
  } else {
   while ((!EOL()) && (strcmp(ptr,item()) < 0)) {
    dummy = step();
   }
   // Since we insert in front, go to next item
   dummy = step();  // no problem with NULL,
   insert(ptr);     // inserting at NULL goes to end of list
  }
}


/*********************************************************
 * remove()
 *
 * Remove the element pointed to by iter.  The memory used by the data
 * item for that element is recovered and the linked list is updated.
 * iter is left pointing to the next element after the removed element.
 *********************************************************/
void LList::remove() {
 ListElem *delPtr = iter;  // Keep track of the item that we are removing

 if (IsEmpty() || (iter == NULL))
  return;

 free(delPtr->item);   // Recover dynamic memory

 if (delPtr == head) {  // First item in list
  if (head == tail) {  // Only one item in list
   head = NULL;
   tail = NULL;
   iter = NULL;
  } else {
   head = delPtr->next;
   iter = head;
  }
 } else {
  ListElem *prev = head;
  while (prev->next != delPtr)   // Find the item right before delPtr
   prev = prev->next;

  if (delPtr == tail) {
   tail = prev;
   tail->next = NULL;
   iter = NULL;
  } else {
   prev->next = delPtr->next;
   iter = delPtr->next;
  }
 }
 delete delPtr;
}


/*********************************************************
 * clear()
 *
 * Removes all elements of the linked list, including the dynamic
 * memory allocated to the data items, and sets the linked list points
 * to NULL.
 *********************************************************/
void LList::clear() {
 ListElem *delPtr;

 while (head != NULL) {
  delPtr = head->next;
  free(head->item);
  delete head;
  head = delPtr;
 }
 head = NULL;
```

```
 tail = NULL;
 iter = NULL;
}


/*********************************************************
 * writeID_LIST(ID_LIST &ptr)
 *
 * This routine provides direct support of ge_interface.h by copying all
 * information from the LList linked list to the ID_LIST linked list.
 * The value of ptr will be changed in this routine.  First all memory
 * will be recovered from the ID_LIST.  Then a new ID_LIST will be
 * created.  iter is not changed by this routine.
 *********************************************************/
void LList::writeID_LIST(ID_LIST &ptr) {
 ID_LIST id, prev_id;
 char* ID_ptr;

 // Get rid of any present list
 clear_ID_LIST(ptr);

 prev_id = NULL;

 reset();
 while (!EOL()) {
  // create the space
  id = (ID_LIST) malloc(sizeof(ID_NODE));

  // insert a copy of char string into the ID_NODE
  ID_ptr  = item();
  id->ID  = make_str(ID_ptr);
  id->NEXT = NULL;

  // insert the ID_NODE into the ID_LIST
  if (prev_id == NULL) {
   ptr = id;
   prev_id = id;
  }
  else {
   prev_id->NEXT = id;
   prev_id = id;
  }
  step();     // next element in LList
 }
}


/*********************************************************
 * readID_LIST(ID_LIST ptr)
 *
 * This routine provides direct support of ge_interface.h by copying all
 * information from the ID_LIST linked list to the LList linked list.
 * First the LList is cleared of all data items.  Then a copy is made
 * of ID_LIST data items.  iter is not altered by this routine.
 *********************************************************/
void LList::readID_LIST(ID_LIST ptr) {

 // Get rid of any present list
 clear();

 while (ptr != NULL) {
  insert(ptr->ID);
  ptr = ptr->NEXT;
 }
}


/*********************************************************
 * sort_readID_LIST(ID_LIST ptr)
 *
```

```
 * This routine provides direct support of ge_interface.h by copying all
 * information from the ID_LIST linked list to the LList linked list.
 * First the LList is cleared of all data items.  Then each element of
 * ID_LIST is inserted into LList in alphabetical order.
 * iter is not altered by this routine.
 ********************************************************/
void LList::sort_readID_LIST(ID_LIST ptr) {

 // Get rid of any present list
 clear();

 while (ptr != NULL) {
  sorted_insert(ptr->ID);
  ptr = ptr->NEXT;
 }
}

/*********************************************************
*  List::Print
*    Output llist to screen
*********************************************************/

void LList::Print()

{ // beg of print

  if (IsEmpty()) {
    cout << "List is Empty!!!!!\a\n" << endl;
    return;
  } // if

  ListElem *ptr = head;

  while (ptr != NULL) {

    cout << (char *) ptr->item << ' '; //output to screen
    ptr = (ptr->next);              // move to next pointer

  } // while



} // end of Print


int LList::length() {

 int n = 0;
 ListElem *countPtr = head;

 while (countPtr != NULL) {
  n++;
  countPtr = countPtr->next;
 }

 return n;
}
```

```
/* ********************************************************

Name:        operator_object.h
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 17 Sep 92
Remarks:  This is the specification for the OperatorObject
      class.

       The OperatorObject class is the graphical
      representation of a PSDL operator.  It has three
      physical forms: circular for an atomic operator,
      two concentric circles for a non-atomic operator,
      and rectangular, for terminators.  A terminator
      simulates interaction with objects outside the system.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
     Changed include file to a .H from .h for C++.

 @2   KBM    15 Aug 96
     Renamed method:   write_to_disk   -> to_psdl
                       build_from_disk -> from_psdl

 @3   KBM    15 Aug 96
     Moved common items up to GraphObject

 @4   KBM    15 Aug 96
     Added new items and methods from ge_interface.h
     Still need to define methods for ID_LIST items

 @5   KBM    15 Aug 96
     Added local operator new and delete for debugging

 @6   KBM    20 Aug 96
     Added a new method over() which is just like hit() except
     that it does not select anything.  Same for over_handle().

 @7   KBM    26 Aug 96
     Steve Decato pointed out that the strcpy() procedure had the
          arguments reversed.

 @8   KBM    30 Aug 96
     Removed set_constraint.

********************************************************
*/

#ifndef operator_object_h
#define operator_object_h 1

#include <stdio.h>
#include <X11/Xlib.h>
#include "ge_defs.h"
#include "ge_interface.h"
#include "ge_utilities.H"
#include "graph_object.H"                          //@1
#include "llist.H"


class OperatorObject : public GraphObject {

protected:

  char    *_met_string_ptr;                        //@3
  int     _met_width;
```

```
int     _met_height;
BOOLEAN _met_selected;
BOOLEAN _op_handles_drawn;
BOOLEAN _met_handles_drawn;

//------------------------------------------------------------- //@4
// OperatorObject

/* info about location of operator icon */
int
    _x,             /* x_position of the circle's center point */
    _y,             /* y_position of the circle's center point */
    _radius,
    _color;

/* info about timing constraints */
int
    _timing_type;       /* NTC, PERIOD, SPORADIC     */

/* info about MET */
int
    _met,           /* UNDEFINED_TIME if no met */
    _met_unit,        /* US, MS, SECOND, MINUTE, HOUR */
    _met_font,
    _met_x,         /* actual x_position = _x + _met_x_offset */
    _met_y;         /* actual y_position = _y + _met_y_offset */

/* info about PERIOD */
int
    _period,        /* UNDEFINED_TIME if no period */
    _period_unit;         /* US, MS, SECOND, MINUTE, HOUR */

/* info about FINISH WITHIN */
int
    _fw,            /* UNDEFINED_TIME if no finish within */
    _fw_unit;           /* US, MS, SECOND, MINUTE, HOUR */

/* info about MAXIMUM RESPONSE TIME */
int
    _mrt,           /* UNDEFINED_TIME if no maximum response
time */
    _mrt_unit;          /* US, MS, SECOND, MINUTE, HOUR */

/* info about MINIMUM CALLING PERIOD */
int
    _mcp,           /* UNDEFINED_TIME if no minimum calling
period */
    _mcp_unit;          /* US, MS, SECOND, MINUTE, HOUR */

/* info about triggering */
int
    _trigger_type;      /* UNPROTECTED, BY_SOME, BY_ALL */
char*
    _if_condition;      /* triggered_if condition, set to TRUE
                       if not specified,
                       parameter to be forwarded to mini-sde
                       EDIT_IF_COND(char* _if_condition)    */

/* info about output guard */
char*
    _output_guard_list;   /* parameter to be forwarded to mini-sde
                       EDIT_OUTPUT_GUARD(char*
_output_guard_list) */

/* info about exception */
char*
    _exception_list;     /* parameter to be forwarded to mini-sde
                       EDIT_EXCEPTION(char* _exception_list) */

/* info about timer op */
```

```
char*
    _timer_op_list;       /* parameter to be forwarded to mini-sde
                          EDIT_TIMER_OP(char* _timer_op_list) */
/* info about key_words for operator spec */

/* info about informal description operator spec */
char*
    _operator_informal_desc;

/* info about formal description operator spec */
char*
    _operator_formal_desc;

/* operator properties */
BOOLEAN
    _is_composite,
    _is_terminator;

//------------------------------------------------------------------  //@4


 void set_default_met_location();
 void set_default_name_location();
 void set_text_dimensions();
 void set_object_font(int font_id);
 void draw_handles(GC draw_context, int x1, int y1, int x2, int y2);
 void reset_handles_drawn_state();


public:

 //** all LList members
 LList* _met_reqmts;       /* requirements trace */
 LList* _period_reqmts;    /* requirements trace */
 LList* _fw_reqmts;        /* requirements trace */
 LList* _mrt_reqmts;       /* requirements trace */
 LList* _mcp_reqmts;       /* requirements trace */
 LList* _key_word_list;    /* list of keywords   */
 LList* _trigger_reqmts;   /* requirements trace */
 LList* _trigger_streams;  /* requirements trace */




 void copy(OperatorObject *op_ptr);                    //@9
 void copy_back(OperatorObject *op_ptr);               //@9
 void copy_back(OPERATOR op_ptr);
 OperatorObject();
 OperatorObject(char *in_name_ptr, int in_id, int in_met, int
in_met_unit,
          int in_x, int in_y, int in_radius, int in_color,
          BOOLEAN in_is_new, BOOLEAN in_is_composite,
          BOOLEAN in_is_terminator);
 OperatorObject(OPERATOR op_ptr);
 ~OperatorObject();

#ifdef GE_DEBUG
 void* operator new(size_t size) {cout << "operator new" << endl;
//@5
                    return ::operator new(size); }
 void  operator delete(void* s) {cout << "operator delete" << endl;  //@5
                    ::operator delete(s); }
#endif

 GE_STATUS build_from_property();   //?? Delete this with pop up
properties
 GE_STATUS write_to_property();     //?? Delete this with pop up
properties
```

```
GE_STATUS to_psdl(FILE *outfile);                    //@2
GE_STATUS from_psdl(FILE *infile);                   //@2

void draw(DRAW_STYLE style);
void draw_text(DRAW_STYLE style);

void clearObj();
BOOLEAN hit(int x, int y);
BOOLEAN over(int x, int y);                          //@6
void select();
void unselect();
CLASS_DEF is_a() {return OPERATOROBJECT;}

XYPAIR center();
XYPAIR intercept(int x, int y);
void move(int x, int y);
void move_notify(CLASS_DEF , int ) {}
void undelete_notify(CLASS_DEF class_type, int deleted_obj_id);
BOOLEAN is_deleted() {return _is_deleted;}
void delete_notify(CLASS_DEF class_type, int deleted_obj_id);
void replace_name(char *new_name);
// void set_constraint(int constraint);              //@9

BOOLEAN text_selected() {return (_name_selected || _met_selected);}
int    text_width();
int    text_height();
void   move_text(int x, int y);
void   text_locate(int x, int y);
void   set_default_text_location();

void write_block(GC draw_context, int x, int y,
          char *instring, int block_height);

BOOLEAN hit_handle(int x, int y);
BOOLEAN over_handle(int x, int y);                   //@6
```



```
int    handle_selected() {return _handle_selected;}
void   move_handle(int x, int y);
void   set_handle_selected(int handle);

//------------------------------------------------------------  //@4
// OperatorObject
//
// /* info about location of operator icon */
// int
//      _x,              /* x_position of the circle's center point */
//      _y,              /* y_position of the circle's center point */
//      _radius,
//      _color;

int    x()                { return _x; }
int    y()                { return _y; }
void   set_x(int x)           { _x = x; }
void   set_y(int y)           { _y = y; }
void   set_location(int x, int y);

int    radius()           { return _radius; }
void   set_radius(int r)      { _radius = r; }

int    color()            { return _color; }
void   set_color(COLOR color)    {_color = color;}

//
// /* info about timing constraints */
// int
//      _timing_type;         /* NTC, PERIOD, SPORADIC     */
int    timing_type() { return _timing_type; };
void   set_timing_type(int tt)
       { _timing_type = tt;
            switch (tt) {
              case NTC:     { clear_period(); break; }
```

```
            case PERIODIC: { break; };
            case SPORADIC: { break; };
          }
        }
//
// /* info about MET */
// int
//     _met,              /* UNDEFINED_TIME if no met */
//     _met_unit,         /* US, MS, SECOND, MINUTE, HOUR */
//     _met_font,
//     _met_x,            /* actual x_position = _x + _met_x_offset */
//     _met_y;            /* actual y_position = _y + _met_y_offset */
// ID_LIST
//     _met_reqmts;       /* requirements trace        */

 int   constraint()       { return _met;}     //?? Consider changing to
met()
 int   met()              { return _met; }
 void  set_met(int t)     { _met = t; }
 void  clear_met()        { _met = UNDEFINED_TIME; }

 int   met_unit()         { return _met_unit; }
 void  set_met_unit(int units) { _met_unit = units; }
 void  default_met_unit()  { _met_unit = MS; }

 int   met_font()         { return _met_font; }
 void  set_met_font(int font) { _met_unit = font; }
//void  default_met_font()  { _met_unit = ; }

 int   met_x()            { return _met_x; }
 int   met_y()            { return _met_y; }
 void  set_met_x(int x)    { _met_x = x; }
 void  set_met_y(int y)    { _met_y = y; }

 // LList  met_req_list() { return *_met_reqmts; }
```

```
 // void  set_met_req_list(char* temp) {
_met_reqmts.sorted_insert(temp); }

 // NEED ID_LIST STUFF HERE



//
// /* info about PERIOD */
// int
//     _period,           /* UNDEFINED_TIME if no period */
//     _period_unit;      /* US, MS, SECOND, MINUTE, HOUR */
// ID_LIST
//     _period_reqmts;    /* requirements trace        */

 int   period()           { return _period; }
 void  set_period(int t)   { _period = t; }
 void  clear_period()      { _period = UNDEFINED_TIME; }

 int   period_unit()      { return _period_unit; }
 void  set_period_unit(int units) { _period_unit = units; }
 void  default_period_units()  { _period_unit = MS; }

 // NEED ID_LIST STUFF HERE



//
// /* info about FINISH WITHIN */
// int
//     _fw,               /* UNDEFINED_TIME if no finish within */
//     _fw_unit;          /* US, MS, SECOND, MINUTE, HOUR */
// ID_LIST
//     _fw_reqmts;        /* requirements trace        */
```

```
int    fw()              { return _fw; }
void   set_fw(int t)       { _fw = t; }
void   clear_fw()          { _fw = UNDEFINED_TIME; }

int    fw_unit()         { return _fw_unit; }
void   set_fw_unit(int units)  { _fw_unit = units; }
void   default_fw_units()     { _fw_unit = MS; }

 // NEED ID_LIST STUFF HERE



//
// /* info about MAXIMUM RESPONSE TIME */
// int
//      _mrt,              /* UNDEFINED_TIME if no maximum response
time */
//      _mrt_unit;          /* US, MS, SECOND, MINUTE, HOUR */
// ID_LIST
//      _mrt_reqmts;         /* requirements trace        */

 int    mrt()             { return _mrt; }
 void   set_mrt(int t)       { _mrt = t; }
 void   clear_mrt()          { _mrt = UNDEFINED_TIME; }

 int    mrt_unit()         { return _mrt_unit; }
 void   set_mrt_unit(int units) { _mrt_unit = units; }
 void   default_mrt_units()    { _mrt_unit = MS; }

  // NEED ID_LIST STUFF HERE



//
// /* info about MINIMUM CALLING PERIOD */
// int
//      _mcp,              /* UNDEFINED_TIME if no minimum calling
period */
//      _mcp_unit;          /* US, MS, SECOND, MINUTE, HOUR */
// ID_LIST
//      _mcp_reqmts;         /* requirements trace        */

 int    mcp()             { return _mcp; }
 void   set_mcp(int t)       { _mcp = t; }
 void   clear_mcp()          { _mcp = UNDEFINED_TIME; }

 int    mcp_unit()         { return _mcp_unit; }
 void   set_mcp_unit(int units) { _mcp_unit = units; }
 void   default_mcp_units()    { _mcp_unit = MS; }

  // NEED ID_LIST STUFF HERE



//
// /* info about triggering */
// int
//      _trigger_type;       /* UNPROTECTED, BY_SOME, BY_ALL */
// char*
//      _if_condition;       /* triggered_if condition, set to TRUE
//                           if not specified,
//                           parameter to be forwarded to mini-sde
//                           EDIT_IF_COND(char* _if_condition)    */

 int    trigger_type()        { return _trigger_type;}
 void   set_trigger_type(int tt) { _trigger_type = tt; }
 void   default_trigger_type()  { _trigger_type = UNPROTECTED; }

 char*  if_condition()        { return _if_condition; }
 void   set_if_condition(char* in_ptr)
```

```
    {
        _if_condition = make_str(in_ptr);
    }
void  delete_if_condition()
    {
        free(_if_condition);
        _if_condition = NULL;
    }
void  default_if_condition()
    {
        _if_condition = make_str("TRUE");
    }

//
// ID_LIST
//     _trigger_reqmts;      /* requirements trace        */


//
// /* info about output guard */
// char*
//     _output_guard_list;   /* parameter to be forwarded to mini-sde
//                       EDIT_OUTPUT_GUARD(char*
_output_guard_list) */

 char*  output_guard_list() { return _output_guard_list; }
 void   set_output_guard_list(char* in_ptr)
    {
        _output_guard_list = make_str(in_ptr);
    }
 void   delete_output_guard_list()
    {
        free(_output_guard_list);
        _output_guard_list = NULL;
    }
```

```
//
// /* info about exception */
// char*
//     _exception_list;      /* parameter to be forwarded to mini-sde
//                       EDIT_EXCEPTION(char* _exception_list) */
//

 char*  exception_list() { return _exception_list; }
 void   set_exception_list(char* in_ptr)
    {
        _exception_list = make_str(in_ptr);
    }
 void   delete_exception_list()
    {
        free(_exception_list);
        _exception_list = NULL;
    }

//
// /* info about timer op */
// char*
//     _timer_op_list;       /* parameter to be forwarded to mini-sde
//                       EDIT_TIMER_OP(char* _timer_op_list) */

 char*  timer_op_list() { return _timer_op_list; }
 void   set_timer_op_list(char* in_ptr)
    {
        _timer_op_list = make_str(in_ptr);
    }
 void   delete_timer_op_list()
    {
        free(_timer_op_list);
        _timer_op_list = NULL;
    }
```

```
//
// /* info about key_words for operator spec */
// ID_LIST
//      _key_word_list;       /* list of keywords */



//
// /* info about informal description operator spec */
// char*
//      _operator_informal_desc;

 char*   operator_informal_desc() { return _operator_informal_desc; }
 void    set_operator_informal_desc(char* in_ptr)
       {
            delete _operator_informal_desc;
            _operator_informal_desc = in_ptr;
          }
 void    delete_operator_informal_desc()
       {
            delete _operator_informal_desc;
            _operator_informal_desc = NULL;
          }

//
// /* info about formal description operator spec */
// char*
//      _operator_formal_desc;

 char*   operator_formal_desc() { return _operator_formal_desc; }
 void    set_operator_formal_desc(char* in_ptr)
       {
            delete _operator_formal_desc;
            _operator_formal_desc = in_ptr;
          }
 void    delete_operator_formal_desc()
       {
            delete _operator_formal_desc;
            _operator_formal_desc = NULL;
          }

//
// /* operator properties */
// BOOLEAN
//      _is_composite,
//      _is_terminator;

 BOOLEAN is_composite()       { return _is_composite; }
 void    set_composite()      { _is_composite = TRUE; }
 void    clear_composite()    { _is_composite = FALSE; }
 void    default_composite()  { clear_composite(); }

 BOOLEAN is_terminator()      { return _is_terminator; }
 void    set_terminator()     { _is_terminator = TRUE; _met = 0; }  // Req
7.7
 void    clear_terminator()   { _is_terminator = FALSE; }
 void    default_terminator() { clear_terminator(); }
//
//------------------------------------------------------------------  //@4

};

#endif;
```

```
/* ******************************************************

Name:        operator_object.C
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 17 Sep 92
Remarks:  This is the implementation of the OperatorObject
      class.

       The OperatorObject class is the graphical
      representation of a PSDL operator. It has three
      physical forms: circular for an atomic operator,
      two concentric circles for a non-atomic operator,
      and rectangular, for terminators. A terminator
      simulates interaction with objects outside the system.


Credits:  Portions of code are adapted from the following:

      Barakati, Naba, X Window System Programming, SAMS,
       1991.

      Heller, Dan, Motif Programming Manual, O'Reilly and
       Associates, 1991.

      Johnson, Eric, and Reichard, Kevin, X Window
       Applications Programming, MIS Press, 1989.

      Young, Douglas, Object Oriented Programming With C++
       and OSF/Motif, Prentice-Hall, 1992.

******************************************************
Modification History:

 Baseline taken from Robert M. Dixon

@1   KBM    15 Aug 96
   Changed include file to a .H from .h for C++.

@2   KBM    15 Aug 96
   Renamed methods:  write_to_disk  -> to_psdl
             build_from_disk -> from_psdl

@3   KBM    15 Aug 96
   Moved common stuff up to GraphObject

@4   KBM    20 Aug 96
   Created a new function over() which is just like hit()
       except that it does not select objects.  Same for over_handle().

@5   KBM    30 Aug 96
   Removed set_constraint.

@6   WH     31 Aug 96
   added include ge_utilities.H

@7   WH     1 Sept 96
   standardized comments added to each method

@8   WH     1 Sept 96
    Added the copy and copy_back method for the updating of the
       operator object...

******************************************************
*/

#include <string.h>
#include <stdlib.h>
#include <math.h>
#include <stream.h>
```

```
#include "operator_object.H"                    //@1
#include "graph_object_list.H"                  //@8
#include "ge_utilities.H"                       //@6

#define COMPOSITE_SPACE 5
#define NUM_STRING_LENGTH 15
#define NONE 0
#define UPPERLEFT 1
#define UPPERRIGHT 2
#define LOWERRIGHT 3
#define LOWERLEFT 4
#define MINRADIUS 5



/*******************************************************
* Create OperatorObject inheriting from GraphObject
*******************************************************/

OperatorObject::OperatorObject() : GraphObject() {

 _next_ptr                = NULL;

 _id                      = 0;

 _name_ptr                = NULL;
 _name_font               = _default_font;
 _name_x                  = NULL_VALUE;
 _name_y                  = NULL_VALUE;

 _is_new                  = FALSE;
 _is_modified             = FALSE;
 _is_deleted              = FALSE;

 _met_string_ptr          = NULL;
 _met_font                = _default_font;
 _met_x                   = NULL_VALUE;
 _met_y                   = NULL_VALUE;

 _is_selected             = FALSE;
 _handle_selected         = NONE;
 _name_selected           = FALSE;
 _met_selected            = FALSE;
 reset_handles_drawn_state();
 _name_width              = 0;
 _name_height             = 0;
 _met_width               = 0;
 _met_height              = 0;

 set_timing_type(NTC);
 clear_met();
 clear_period();
 clear_fw();
 clear_mrt();
 clear_mcp();
 default_trigger_type();
 default_if_condition();
 default_composite();
 default_terminator();

 _met_reqmts              = NULL;
 _period_reqmts           = NULL;
 _fw_reqmts               = NULL;
 _mrt_reqmts              = NULL;
 _mcp_reqmts              = NULL;
 _trigger_reqmts          = NULL;
 _output_guard_list       = NULL;
 _exception_list          = NULL;
 _timer_op_list           = NULL;
```

```
 _key_word_list                = NULL;

 _operator_informal_desc       = NULL;
 _operator_formal_desc         = NULL;

}

/*********************************************************
* MET values less than zero are considered to represent
* microsecond values.
*********************************************************/

OperatorObject::OperatorObject(char *in_name_ptr, int in_id,
                    int in_met, int in_met_unit,
                                int in_x, int in_y,
                    int in_radius, int in_color,
                    BOOLEAN in_is_new,
                    BOOLEAN in_is_composite,
                    BOOLEAN in_is_terminator) {
  char buffer[INPUT_LINE_SIZE];

  _name_ptr                = make_str(in_name_ptr);
  _id                      = in_id;
  _met                     = in_met;
  _met_unit                = in_met_unit;
  _met_string_ptr = time_unit_str(_met, _met_unit);

  _x                       = in_x;
  _y                       = in_y;
  _radius                  = in_radius;
  _color                   = in_color;
  _is_new                  = in_is_new;
  _is_deleted              = FALSE;
  _is_modified             = FALSE;
  _is_composite            = in_is_composite;
  _is_terminator           = in_is_terminator;
  _is_selected             = FALSE;
  _name_selected           = FALSE;
  _met_selected            = FALSE;
  _handle_selected         = NONE;
  _name_font               = _default_font;
  _met_font                = _default_font;
  set_text_dimensions();
  set_default_text_location();
  reset_handles_drawn_state();

  set_timing_type(NTC);
  clear_met();
  clear_period();
  clear_fw();
  clear_mrt();
  clear_mcp();
  default_trigger_type();
  default_if_condition();

  _met_reqmts              = NULL;
  _period_reqmts           = NULL;
  _fw_reqmts               = NULL;
  _mrt_reqmts              = NULL;
  _mcp_reqmts              = NULL;
  _trigger_reqmts          = NULL;
  _output_guard_list       = NULL;
  _exception_list          = NULL;
  _timer_op_list           = NULL;
  _key_word_list           = NULL;
  _operator_informal_desc  = NULL;
  _operator_formal_desc    = NULL;

}
```

```
OperatorObject::OperatorObject(OPERATOR op_ptr) {
 char    *temp_str;
 ID_LIST id_ptr;

 _next_ptr              = NULL;
 _is_selected           = FALSE;
 _is_composite          = op_ptr->is_composite;
 _is_terminator         = op_ptr->is_terminator;
 // _name_handles_drawn     = FALSE;
 // _name_selected          = FALSE;
 // _handle_selected        = FALSE;
 // _name_width             = op_ptr->name_width;
 // _name_height            = op_ptr->name_height;
 _id                = op_ptr->id;
 _is_deleted            = op_ptr->is_deleted;
 _is_new                = op_ptr->is_new;

 _is_modified           = op_ptr->is_modified;
 _name_ptr              = make_str(op_ptr->label);
 _name_font             = op_ptr->label_font;
 _name_x                = op_ptr->label_x_offset;
 _name_y                = op_ptr->label_y_offset;

 _x                 = op_ptr->x;
 _y                 = op_ptr->y;
 _radius            = op_ptr->radius;
 _color             = op_ptr->color;

 set_timing_type(op_ptr->timing_type);

 _met                = op_ptr->met;
 _met_unit           = op_ptr->met_unit;
 _met_string_ptr     = time_unit_str(_met, _met_unit);
 _met_font           = op_ptr->met_font;
 _met_x              = op_ptr->met_x_offset;
 _met_y              = op_ptr->met_y_offset;
 if (op_ptr->met_reqmts == NULL)
  _met_reqmts = NULL;
 else {
  _met_reqmts = new LList;
  _met_reqmts->sort_readID_LIST(op_ptr->met_reqmts);
 }

 _period             = op_ptr->period;
 _period_unit        = op_ptr->period_unit;
 if (op_ptr->period_reqmts == NULL)
  _period_reqmts = NULL;
 else {
  _period_reqmts = new LList;
  _period_reqmts->sort_readID_LIST(op_ptr->period_reqmts);
 }

 _fw                 = op_ptr->fw;
 _fw_unit            = op_ptr->fw_unit;
 if (op_ptr->fw_reqmts == NULL)
  _fw_reqmts = NULL;
 else {
  _fw_reqmts = new LList;
  _fw_reqmts->sort_readID_LIST(op_ptr->fw_reqmts);
 }

 _mrt                = op_ptr->mrt;
 _mrt_unit           = op_ptr->mrt_unit;
 if (op_ptr->mrt_reqmts == NULL)
  _mrt_reqmts = NULL;
 else {
  _mrt_reqmts = new LList;
  _mrt_reqmts->sort_readID_LIST(op_ptr->mrt_reqmts);
 }
```

```
_mcp                = op_ptr->mcp;
_mcp_unit           = op_ptr->mcp_unit;
if (op_ptr->mcp_reqmts == NULL)
  _mcp_reqmts = NULL;
else {
  _mcp_reqmts = new LList;
  _mcp_reqmts->sort_readID_LIST(op_ptr->mcp_reqmts);
}

_trigger_type       = op_ptr->trigger_type;
_if_condition       = make_str(op_ptr->if_condition);

if (op_ptr->trigger_reqmts == NULL)
  _trigger_reqmts = NULL;
else {
  _trigger_reqmts = new LList;
  _trigger_reqmts->sort_readID_LIST(op_ptr->trigger_reqmts);
}

_output_guard_list    = make_str(op_ptr->output_guard_list);
_exception_list       = make_str(op_ptr->exception_list);
_timer_op_list        = make_str(op_ptr->timer_op_list);

if (op_ptr->key_word_list == NULL)
  _key_word_list = NULL;
else {
  _key_word_list = new LList;
  _key_word_list->sort_readID_LIST(op_ptr->key_word_list);
}

_operator_informal_desc  = make_str(op_ptr-
>operator_informal_desc);
_operator_formal_desc    = make_str(op_ptr->operator_formal_desc);
_is_composite         = op_ptr->is_composite;
_is_terminator        = op_ptr->is_terminator;

set_text_dimensions();
reset_handles_drawn_state();
}

                                                    //@8
/*********************************************************
* copy OperatorObject members from GraphObject
*********************************************************/

void OperatorObject::copy(OperatorObject *op_ptr) {
 char buffer[INPUT_LINE_SIZE];

 //** copy graph  object members that are inheirted
 _next_ptr           = op_ptr->_next_ptr;
 _is_selected        = op_ptr->_is_selected;
 _is_composite       = op_ptr->_is_composite;
 _is_terminator      = op_ptr->_is_terminator;
 _name_handles_drawn    = op_ptr->_name_handles_drawn;
 _name_selected      = op_ptr->_name_selected;
 _handle_selected    = op_ptr->_handle_selected;
 _name_width         = op_ptr->_name_width;
 _name_height        = op_ptr->_name_height;
 _id                 = op_ptr->_id;
 _is_deleted         = op_ptr->_is_deleted;
 _is_new             = op_ptr->_is_new;
 _is_modified        = op_ptr->_is_modified;
 _name_ptr           = make_str(op_ptr->_name_ptr);
 _name_font          = op_ptr->_name_font;
 _name_x             = op_ptr->_name_x;
 _name_y             = op_ptr->_name_y;

 // copy operator object members

 _x                  = op_ptr->_x;
```

```
_y                  = op_ptr->_y;
_radius             = op_ptr->_radius;
_color              = op_ptr->_color;
_timing_type        = op_ptr->_timing_type;
_met                = op_ptr->_met;
_met_string_ptr     = time_unit_str(op_ptr->_met,op_ptr->_met_unit);
_met_unit           = op_ptr->_met_unit;
_met_font           = op_ptr->_met_font;
_met_x              = op_ptr->_met_x;
_met_y              = op_ptr->_met_y;
_met_reqmts         = op_ptr->_met_reqmts;
_period             = op_ptr->_period;
_period_unit        = op_ptr->_period_unit;
_period_reqmts      = op_ptr->_period_reqmts;
_fw                 = op_ptr->_fw;
_fw_unit            = op_ptr->_fw_unit;
_fw_reqmts          = op_ptr->_fw_reqmts;
_mrt                = op_ptr->_mrt;
_mrt_unit           = op_ptr->_mrt_unit;
_mrt_reqmts         = op_ptr->_mrt_reqmts;
_mcp                = op_ptr->_mcp;
_mcp_unit           = op_ptr->_mcp_unit;
_mcp_reqmts         = op_ptr->_mcp_reqmts;
_trigger_type       = op_ptr->_trigger_type;
_if_condition       = op_ptr->_if_condition;
_trigger_reqmts     = op_ptr->_trigger_reqmts;
_output_guard_list  = op_ptr->_output_guard_list;
_exception_list     = op_ptr->_exception_list;
_timer_op_list      = op_ptr->_timer_op_list;
_key_word_list      = op_ptr->_key_word_list;
_operator_informal_desc  = op_ptr->_operator_informal_desc;
_operator_formal_desc    = op_ptr->_operator_formal_desc;
_is_composite       = op_ptr->_is_composite;
_is_terminator      = op_ptr->_is_terminator;

 /* set_text_dimensions();*/

 op_ptr->set_default_text_location();
 op_ptr->reset_handles_drawn_state();
 op_ptr->clear_met();
 op_ptr->clear_period();
 op_ptr->clear_fw();
 op_ptr->clear_mrt();
 op_ptr->clear_mcp();
 op_ptr->default_trigger_type();
 op_ptr->default_if_condition();


 set_text_dimensions();
 reset_handles_drawn_state();
}
                                                    //@8
/*********************************************************
* copy OperatorObject members into GraphObject
*********************************************************/

void OperatorObject::copy_back(OperatorObject *op_ptr) {
 char buffer[INPUT_LINE_SIZE];

 //** copy graph  object members that are inheirted
 op_ptr->_next_ptr              = _next_ptr;
 op_ptr->_is_selected           = _is_selected;
 op_ptr->_is_composite          = _is_composite;
 op_ptr->_is_terminator         = _is_terminator;
 op_ptr->_name_handles_drawn    = _name_handles_drawn;
 op_ptr->_name_selected         = _name_selected;
 op_ptr->_handle_selected       = _handle_selected;
 op_ptr->_name_width            = _name_width;
```

```
op_ptr->_name_height          = _name_height;
op_ptr->_id                   = _id;
op_ptr->_is_deleted           = _is_deleted;
op_ptr->_is_new               = _is_new;
op_ptr->_is_modified          = _is_modified;
op_ptr->_name_ptr             = make_str(_name_ptr);
op_ptr->_name_font            = _name_font;
op_ptr->_name_x               = _name_x;
op_ptr->_name_y               = _name_y;

// copy operator object members

op_ptr->_x                    = _x;
op_ptr->_y                    = _y;
op_ptr->_radius               = _radius;
op_ptr->_color                = _color;
op_ptr->_timing_type          = _timing_type;
op_ptr->_met                  = _met;
op_ptr->_met_string_ptr       = time_unit_str(_met,_met_unit);
op_ptr->_met_unit             = _met_unit;
op_ptr->_met_font             = _met_font;
op_ptr->_met_x                = _met_x;
op_ptr->_met_y                = _met_y;
op_ptr->_met_reqmts           = _met_reqmts;
op_ptr->_period               = _period;
op_ptr->_period_unit          = _period_unit;
op_ptr->_period_reqmts        = _period_reqmts;
op_ptr->_fw                   = _fw;
op_ptr->_fw_unit              = _fw_unit;
op_ptr->_fw_reqmts            = _fw_reqmts;
op_ptr->_mrt                  = _mrt;
op_ptr->_mrt_unit             = _mrt_unit;
op_ptr->_mrt_reqmts           = _mrt_reqmts;
op_ptr->_mcp                  = _mcp;
op_ptr->_mcp_unit             = _mcp_unit;
op_ptr->_mcp_reqmts           = _mcp_reqmts;
op_ptr->_trigger_type         = _trigger_type;
op_ptr->_if_condition         = _if_condition;
op_ptr->_trigger_reqmts       = _trigger_reqmts;
op_ptr->_output_guard_list    = _output_guard_list;
op_ptr->_exception_list       = _exception_list;
op_ptr->_timer_op_list        = _timer_op_list;
op_ptr->_key_word_list        = _key_word_list;
op_ptr->_operator_informal_desc = _operator_informal_desc;
op_ptr->_operator_formal_desc = _operator_formal_desc;
op_ptr->_is_composite         = _is_composite;
op_ptr->_is_terminator        = _is_terminator;


/* set_text_dimensions();*/
set_default_text_location();
reset_handles_drawn_state();
clear_met();
clear_period();
clear_fw();
clear_mrt();
clear_mcp();
default_trigger_type();
default_if_condition();

set_text_dimensions();
reset_handles_drawn_state();

}

/*********************************************************
* copy OperatorObject members into OPERATOR
*********************************************************/

void OperatorObject::copy_back(OPERATOR op_ptr) {
```

```
op_ptr->id              = _id;
op_ptr->x               = _x;
op_ptr->y               = _y;
op_ptr->radius          = _radius;
op_ptr->color           = _color;

op_ptr->label           = make_str(_name_ptr);
op_ptr->label_font      = _name_font;
op_ptr->label_x_offset  = _name_x;
op_ptr->label_y_offset  = _name_y;

op_ptr->timing_type     = _timing_type;

op_ptr->met             = _met;
op_ptr->met_unit        = _met_unit;
op_ptr->met_font        = _met_font;
op_ptr->met_x_offset    = _met_x;
op_ptr->met_y_offset    = _met_y;
if (op_ptr->met_reqmts == NULL)
 _met_reqmts = NULL;
else
 _met_reqmts->writeID_LIST(op_ptr->met_reqmts);

op_ptr->period          = _period;
op_ptr->period_unit     = _period_unit;
if (op_ptr->period_reqmts == NULL)
 _period_reqmts = NULL;
else
 _period_reqmts->writeID_LIST(op_ptr->period_reqmts);

op_ptr->fw              = _fw;
op_ptr->fw_unit         = _fw_unit;
if (op_ptr->fw_reqmts == NULL)
 _fw_reqmts = NULL;
else
 _fw_reqmts->writeID_LIST(op_ptr->fw_reqmts);

op_ptr->mrt             = _mrt;
op_ptr->mrt_unit        = _mrt_unit;
if (op_ptr->mrt_reqmts == NULL)
 _mrt_reqmts = NULL;
else
 _mrt_reqmts->writeID_LIST(op_ptr->mrt_reqmts);

op_ptr->mcp             = _mcp;
op_ptr->mcp_unit        = _mcp_unit;
if (op_ptr->mcp_reqmts == NULL)
 _mcp_reqmts = NULL;
else
 _mcp_reqmts->writeID_LIST(op_ptr->mcp_reqmts);

op_ptr->trigger_type    = _trigger_type;

op_ptr->if_condition    = make_str(_if_condition);

if (op_ptr->trigger_reqmts == NULL)
 _trigger_reqmts = NULL;
else
 _trigger_reqmts->writeID_LIST(op_ptr->trigger_reqmts);

op_ptr->output_guard_list  = make_str(_output_guard_list);
op_ptr->exception_list     = make_str(_exception_list);
op_ptr->timer_op_list      = make_str(_timer_op_list);

if (op_ptr->key_word_list == NULL)
 _key_word_list = NULL;
else
 _key_word_list->writeID_LIST(op_ptr->key_word_list);
```

```
 op_ptr->operator_informal_desc   =
make_str(_operator_informal_desc);
 op_ptr->operator_formal_desc     = make_str(_operator_formal_desc);

 op_ptr->is_composite          = _is_composite;
 op_ptr->is_terminator         = _is_terminator;

 op_ptr->is_deleted            = _is_deleted;
 op_ptr->is_new                = _is_new;
 op_ptr->is_modified           = _is_modified;
}

/*******************************************************
* Draws handles around the given coordinates. Drawn in
* exclusive-or mode to simplify erasure.
*******************************************************/

void OperatorObject::draw_handles(GC draw_context, int x1,
                     int y1, int x2, int y2) {

 XSetFunction(_display_ptr, draw_context, GXxor);
 XFillRectangle(_display_ptr, _draw_window, draw_context, x1,
          y1, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, _draw_window, draw_context,
          x2 - HANDLESIZE, y1, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, _draw_window, draw_context, x1,
          y2 - HANDLESIZE, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, _draw_window, draw_context,
          x2 - HANDLESIZE, y2 - HANDLESIZE, HANDLESIZE,
          HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x1, y1, HANDLESIZE, HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x2 - HANDLESIZE, y1, HANDLESIZE,
          HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x1, y2 - HANDLESIZE, HANDLESIZE,
          HANDLESIZE);
 XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x2 - HANDLESIZE, y2 - HANDLESIZE,
          HANDLESIZE, HANDLESIZE);
 XSetFunction(_display_ptr, draw_context, GXcopy);
}

/*******************************************************
* Determines the dimensions of the text blocks. _name_font
* and _met_font must be set before calling. Text strings
* are automatically broken at underscores, except for
* underscores in the first and last character of the string.
*******************************************************/

void OperatorObject::set_text_dimensions() {
 int i, end_prev_index = -1, widest_string = 0, num_breaks = 0,
  widest_start_index, temp_width, temp_height, str_len;
 char temp_str[INPUT_LINE_SIZE];

 if (_name_ptr == NULL) {
  _name_width = 0;
  _name_height = 0;
 }
 else {
  str_len = strlen(_name_ptr);
  for (i = 0; i < str_len; i++)
   if (_name_ptr[i] == '_') {
    if ((i != 0) && (i != (str_len - 1))) {
     num_breaks++;
     temp_width = i - end_prev_index;
     if (temp_width > widest_string) {
      widest_string = temp_width;
      widest_start_index = end_prev_index + 1;
```

```
      }
      end_prev_index = i;
    }
   }
  if (end_prev_index < (str_len - 1)) {
   temp_width = str_len - 1 - end_prev_index;
   if (temp_width > widest_string) {
    widest_string = temp_width;
    widest_start_index = end_prev_index + 1;
   }
  }
  if (widest_string == 0) {
   widest_start_index = 0;
   widest_string = str_len;
  }
  strncpy(temp_str, &(_name_ptr[widest_start_index]),
        widest_string);
  temp_str[widest_string] = '\0';
  //   temp_str[widest_string] = NULL;
  _name_width = font_text_width(_name_font, temp_str);
  temp_height = font_text_height(_name_font);
  _name_height = temp_height * (num_breaks + 1);
 }
 if (_met != UNDEFINED_TIME) {
  _met_width = font_text_width(_met_font, _met_string_ptr);
  _met_height = font_text_height(_met_font);
 }
 else {
  _met_width = 0;
  _met_height = 0;
 }
}


/*******************************************************
```

```
* destuctor of Operator Object
*******************************************************/

OperatorObject::~OperatorObject() {

 delete _name_ptr;
 delete _met_string_ptr;
}

void OperatorObject::clearObj() {

 free(_name_ptr);          _name_ptr          = NULL;
 free(_met_string_ptr);    _met_string_ptr    = NULL;
 if (_met_reqmts != NULL) {
  _met_reqmts->clear();
  delete _met_reqmts;
 }
 if (_period_reqmts != NULL) {
  _period_reqmts->clear();
  delete _period_reqmts;
 }
 if (_fw_reqmts != NULL) {
  _fw_reqmts->clear();
  delete _fw_reqmts;
 }
 if (_mrt_reqmts != NULL) {
  _mrt_reqmts->clear();
  delete _mrt_reqmts;
 }
 if (_mcp_reqmts != NULL) {
  _mcp_reqmts->clear();
  delete _mcp_reqmts;
 }
 free(_if_condition);      _if_condition      = NULL;
 if (_trigger_reqmts != NULL) {
```

```
  _trigger_reqmts->clear();
  delete _trigger_reqmts;
 }
 free(_output_guard_list);     _output_guard_list    = NULL;
 free(_exception_list);        _exception_list       = NULL;
 free(_timer_op_list);         _timer_op_list        = NULL;
 if (_key_word_list != NULL) {
   _key_word_list->clear();
   delete _key_word_list;
 }
 free(_operator_informal_desc); _operator_informal_desc = NULL;
 free(_operator_formal_desc);   _operator_formal_desc  = NULL;
}

/*********************************************************
*  Determines a default location for the MET string.
*********************************************************/

void OperatorObject::set_default_met_location() {

 if (_is_terminator)
   _met_x = _x + (int) ((float) _radius * 1.5)
     - (_met_width / 2);
 else
   _met_x = _x + _radius
     - (_met_width / 2);
 _met_y = _y;
}

/*********************************************************
*  Determines a default location for the name string.
*********************************************************/
void OperatorObject::set_default_name_location() {

 if (_is_terminator)
   _name_x = _x + (int) ((float) _radius * 1.5)
     - (_name_width / 2);
 else
   _name_x = _x + _radius
     - (_name_width / 2);
 _name_y = _y + _radius + (_name_height / 2);
}

/*********************************************************
*  Convenience routine for setting both locations at once.
*********************************************************/

void OperatorObject::set_default_text_location() {

 set_default_name_location();
 set_default_met_location();
}

GE_STATUS OperatorObject::build_from_property() {return FAILED;}

/*********************************************************
*  Builds an operator from a disk file.
*********************************************************/

GE_STATUS OperatorObject::from_psdl(FILE *infile) {          //@2
 char buffer[INPUT_LINE_SIZE + 1], *error_str_ptr = NULL;
 int last_char;
 GE_STATUS status = SUCCEEDED;

 fgets(buffer, INPUT_LINE_SIZE, infile);
 last_char = strlen(buffer) - 1;
 if (buffer[last_char] == '\n') // Strips off trailing newline
   buffer[last_char] = 0;
 if (strcmp(buffer, "STREAMS") == 0)
   return ENDED;
```

```
else {
 _name_ptr = strdup(buffer);

 fgets(buffer, INPUT_LINE_SIZE, infile);
 if (valid_num_string(buffer)) {
  _name_font = atoi(buffer);
  if (_name_font > MAXFONTS)
   error_str_ptr = strdup("Name Font Too Large");
 }
 else
  error_str_ptr = strdup("Corrupted Name Font");

 if (error_str_ptr == NULL) {
  fgets(buffer, INPUT_LINE_SIZE, infile);
  if (valid_num_string(buffer))
   _name_x = atoi(buffer);
  else
   error_str_ptr = strdup("Corrupted name_x");
 }

 if (error_str_ptr == NULL) {
  fgets(buffer, INPUT_LINE_SIZE, infile);
  if (valid_num_string(buffer))
   _name_y = atoi(buffer);
  else
   error_str_ptr = strdup("Corrupted name_y");
 }

 if (error_str_ptr == NULL) {
  fgets(buffer, INPUT_LINE_SIZE, infile);
  if (valid_num_string(buffer))
   _id = atoi(buffer);
  else
   error_str_ptr = strdup("Corrupted id");
 }

 if (error_str_ptr == NULL) {
  fgets(buffer, INPUT_LINE_SIZE, infile);
  if (valid_num_string(buffer))
   _met = atoi(buffer);
  else
   error_str_ptr = strdup("Corrupted met");
 }


 if (error_str_ptr == NULL) {
  if (_met == UNDEFINED_TIME)
   _met_string_ptr = NULL;
  else {
   if (_met < 0)
    sprintf(buffer, "%d us", -_met);
   else
    sprintf(buffer, "%d ms", _met);
   _met_string_ptr = strdup(buffer);
  }
 }

 if (error_str_ptr == NULL) {
  fgets(buffer, INPUT_LINE_SIZE, infile);
  if (valid_num_string(buffer)) {
   _met_font = atoi(buffer);
   if (_met_font > MAXFONTS)
    error_str_ptr = strdup("Met Font Too Large");
  }
  else
   error_str_ptr = strdup("Corrupted Met Font");
 }

 if (error_str_ptr == NULL)
  set_text_dimensions();
```

```
if (error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if (valid_num_string(buffer))
  _met_x = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted met_x");
}

if (error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if (valid_num_string(buffer))
  _met_y = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted met_y");
}

if (error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if (valid_num_string(buffer))
  _x = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted x");
}

if (error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if (valid_num_string(buffer))
  _y = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted y");
}

if (error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if (valid_num_string(buffer))
  _radius = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted radius");
}

if (error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if (valid_num_string(buffer)) {
  _color = atoi(buffer);
  if (_color > MAXCOLORS)
   error_str_ptr = strdup("Color Too Large");
 }
 else
  error_str_ptr = strdup("Corrupted Color");
}

if (error_str_ptr == NULL) {
 if (strcmp("TRUE\n",
       fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
  _is_deleted = TRUE;
 else
  if (strcmp(buffer, "FALSE\n") == 0)
   _is_deleted = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_deleted");
}

if (error_str_ptr == NULL) {
 if (strcmp("TRUE\n",
       fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
   //Change this if original _is_new should be saved.
  _is_new = FALSE;
  else
   if (strcmp(buffer, "FALSE\n") == 0)
```

```
      _is_new = FALSE;
     else
      error_str_ptr = strdup("Corrupted is_new");
}

if (error_str_ptr == NULL) {
 if (strcmp("TRUE\n",
        fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
   _is_modified = TRUE;
 else
  if (strcmp(buffer, "FALSE\n") == 0)
    _is_modified = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_modified");
}

if (error_str_ptr == NULL) {
 if (strcmp("TRUE\n",
        fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
   _is_composite = TRUE;
 else
  if (strcmp(buffer, "FALSE\n") == 0)
    _is_composite = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_composite");
}

if (error_str_ptr == NULL) {
 if (strcmp("TRUE\n",
        fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
   _is_terminator = TRUE;
 else
  if (strcmp(buffer, "FALSE\n") == 0)
    _is_terminator = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_terminator");
}

if (error_str_ptr == NULL) {
 if (_name_x == NULL_VALUE)
  set_default_name_location();
 if (_met_x == NULL_VALUE)
  set_default_met_location();
}

if (error_str_ptr != NULL) {
 sprintf(buffer, "operator %s: %s", _name_ptr,
       error_str_ptr);
 error_box(buffer);
 status = FAILED;
 free(error_str_ptr);
}

if (ferror(infile)) {
 sprintf(buffer,
  "Unix reported an error building operator %s",
   _name_ptr);
 error_box(buffer);
 clearerr(infile);
 status = FAILED;
}
}
return status;
}

/*********************************************************
* Write the object to disk
*********************************************************/

GE_STATUS OperatorObject::write_to_property() {return FAILED;}
```

```
/*********************************************************
* Writes the operator's attributes to disk.
*********************************************************/

GE_STATUS OperatorObject::to_psdl(FILE *outfile) {              //@2
 char buffer[INPUT_LINE_SIZE + 1];

 fprintf(outfile, "%s\n", _name_ptr);
 sprintf(buffer, "%d\n%d\n%d", _name_font, _name_x, _name_y);
 fprintf(outfile, "%s\n", buffer);
 sprintf(buffer, "%d", _id);
 fprintf(outfile, "%s\n", buffer);
 sprintf(buffer, "%d", _met);
 fprintf(outfile, "%s\n", buffer);
 sprintf(buffer, "%d\n%d\n%d", _met_font, _met_x, _met_y);
 fprintf(outfile, "%s\n", buffer);
 sprintf(buffer, "%d", _x);
 fprintf(outfile, "%s\n", buffer);
 sprintf(buffer, "%d", _y);
 fprintf(outfile, "%s\n", buffer);
 sprintf(buffer, "%d", _radius);
 fprintf(outfile, "%s\n", buffer);
 sprintf(buffer, "%d", _color);
 fprintf(outfile, "%s\n", buffer);
 if (_is_deleted)
  fprintf(outfile, "TRUE\n");
 else
  fprintf(outfile, "FALSE\n");
 if (_is_new)
  fprintf(outfile, "TRUE\n");
 else
  fprintf(outfile, "FALSE\n");
 if (_is_modified)
  fprintf(outfile, "TRUE\n");
 else
  fprintf(outfile, "FALSE\n");
 if (_is_composite)
  fprintf(outfile, "TRUE\n");
 else
  fprintf(outfile, "FALSE\n");
 if (_is_terminator)
  fprintf(outfile, "TRUE\n");
 else
  fprintf(outfile, "FALSE\n");
 if (ferror(outfile)) {
  clearerr(outfile);
  return FAILED;
 }
 else
  return SUCCEEDED;
}

/*********************************************************
* Writes a text block to the drawing canvas.  Text strings
* are broken at underscores, except when in the first and
* last character position.
*********************************************************/

void OperatorObject::write_block(GC draw_context, int x, int y,
   char *instring, int block_height) {
 int i, str_len, num_lines = 1, string_width, start_index = 0,
 yinc, block_index = 0;
 char *block_text[MAXTEXTLINES],
temp_string[INPUT_LINE_SIZE];

 if (instring != NULL) {
  str_len = strlen(instring);
  if (str_len != 0) {
   for (i = 0; i < str_len; i++)
```

```
  if (instring[i] == '_') {
   if ((i != 0) && (i != (str_len - 1))) {
    string_width = i - start_index + 1;
    strncpy(temp_string, &(instring[start_index]),
          string_width);
    temp_string[string_width] = '\0';
        //          temp_string[string_width] = NULL;
    block_text[num_lines - 1] = strdup(temp_string);
    start_index = i + 1;
    num_lines++;
   }
  }
 if (start_index < str_len) {
  string_width = i - start_index + 1;
  strncpy(temp_string, &(instring[start_index]),
        string_width);
       temp_string[string_width] = '\0';
       //          temp_string[string_width] = NULL;
  block_text[num_lines - 1] = strdup(temp_string);
 }
 else
  num_lines--;
 yinc = block_height / num_lines;
 for (i = 0; i < num_lines; i++) {
  str_len = strlen(block_text[i]);
  XDrawString(_display_ptr, _draw_window, draw_context, x,
          y - (yinc * (num_lines - i - 1)),
          block_text[i], str_len);
  XDrawString(_display_ptr, *_drawing_area_pixmap,
          draw_context, x,
          y - (yinc * (num_lines - i - 1)),
          block_text[i], str_len);
  free(block_text[i]);
 }
}
```



```
 }
}

/*********************************************************
* Writes text on the drawing canvas, including handles if
* appropriate.  Since handles are drawn in exclusive-or mode,
* it's important to make sure that they aren't redrawn
* unless they've been erased or they need to be erased.
*********************************************************/

void OperatorObject::draw_text(DRAW_STYLE style) {
 GC draw_context;

 if ((style == SOLID) || (style == DOTTED))
  draw_context = _graphics_context;
 else
  if (style == ERASE)
   draw_context = _erase_context;
 set_font(draw_context, _name_font);
 write_block(draw_context, _name_x, _name_y, _name_ptr,
         _name_height);
 if ((_name_selected) && (_name_handles_drawn == FALSE) &&
   (style == SOLID)) {
  draw_handles(_graphics_context, _name_x - HANDLESIZE,
          _name_y - _name_height - HANDLESIZE,
          _name_x + _name_width + HANDLESIZE,
          _name_y + HANDLESIZE);
  _name_handles_drawn = TRUE;
 }
 else
  if ((_name_selected) && (_name_handles_drawn == TRUE) &&
    (style == ERASE)) {
   draw_handles(_graphics_context, _name_x - HANDLESIZE,
          _name_y - _name_height - HANDLESIZE,
          _name_x + _name_width + HANDLESIZE,
```

```
            _name_y + HANDLESIZE);
    _name_handles_drawn = FALSE;
   }
  if (_met != UNDEFINED_TIME) {
   set_font(draw_context, _met_font);
   XDrawString(_display_ptr, _draw_window, draw_context,
          _met_x, _met_y, _met_string_ptr,
          strlen(_met_string_ptr));
   XDrawString(_display_ptr, *_drawing_area_pixmap,
          draw_context, _met_x, _met_y, _met_string_ptr,
          strlen(_met_string_ptr));
   if ((_met_selected) && (_met_handles_drawn == FALSE) &&
     (style == SOLID)) {
    draw_handles(_graphics_context, _met_x - HANDLESIZE,
            _met_y - _met_height - HANDLESIZE,
            _met_x + _met_width + HANDLESIZE,
            _met_y + HANDLESIZE);
    _met_handles_drawn = TRUE;
   }
   else
    if ((_met_selected) && (_met_handles_drawn == TRUE) &&
      (style == ERASE)) {
     draw_handles(_graphics_context, _met_x - HANDLESIZE,
             _met_y - _met_height - HANDLESIZE,
             _met_x + _met_width + HANDLESIZE,
             _met_y + HANDLESIZE);
     _met_handles_drawn = FALSE;
    }

  }
}

/*********************************************************
*  Draws the operator.
*********************************************************/
```

```
void OperatorObject::draw(DRAW_STYLE style) {
 GC draw_context;

 if (_is_deleted == FALSE) {
  if (style == SOLID)
   draw_context = _graphics_context;
  else
   if (style == DOTTED)
    draw_context = _dotted_context;
   else
    draw_context = _erase_context;
  if (_is_terminator) {
   if (style == SOLID) {
    XSetForeground(_display_ptr, draw_context,
            _color_table[_color]);
   }
   if (style != DOTTED) {
    XFillRectangle(_display_ptr, _draw_window, draw_context,
            _x, _y, _radius * 3, _radius * 2);
    XFillRectangle(_display_ptr, *_drawing_area_pixmap,
            draw_context, _x, _y, _radius * 3,
            _radius * 2);
    if (style == ERASE)
     _op_handles_drawn = FALSE;
   }
   if (style == SOLID)
    XSetForeground(_display_ptr, draw_context,
_color_table[BLACK]);
   XDrawRectangle(_display_ptr, _draw_window, draw_context,
           _x, _y, _radius * 3, _radius * 2);
   XDrawRectangle(_display_ptr, *_drawing_area_pixmap,
           draw_context, _x, _y, _radius * 3,
           _radius * 2);
```

```
  if ((_is_selected) && (_op_handles_drawn == FALSE) &&
    (style == SOLID)) {
   draw_handles(_graphics_context, _x, _y,
          _x + (3 * _radius), _y + (2 * _radius));
   _op_handles_drawn = TRUE;
  }
 }
 else {
  if (_is_composite) {
   if (style == SOLID) {
    XSetForeground(_display_ptr, draw_context,
          _color_table[_color]);
   }
   if (style != DOTTED) {
    XFillArc(_display_ptr, _draw_window, draw_context,
        _x + COMPOSITE_SPACE, _y + COMPOSITE_SPACE,
        (_radius - COMPOSITE_SPACE) * 2,
        (_radius - COMPOSITE_SPACE) * 2,
        CIRCLE_BEGIN, FULL_CIRCLE);
    XFillArc(_display_ptr, *_drawing_area_pixmap,
        draw_context, _x + COMPOSITE_SPACE,
        _y + COMPOSITE_SPACE,
        (_radius - COMPOSITE_SPACE) * 2,
        (_radius - COMPOSITE_SPACE) * 2,
        CIRCLE_BEGIN, FULL_CIRCLE);
   }
   if (style == SOLID) {
    XSetForeground(_display_ptr, draw_context,
_color_table[BLACK]);
   }
   XDrawArc(_display_ptr, _draw_window, draw_context, _x,
       _y, _radius * 2, _radius * 2, CIRCLE_BEGIN,
       FULL_CIRCLE);
   XDrawArc(_display_ptr, *_drawing_area_pixmap,
       draw_context, _x, _y, _radius * 2, _radius *2,
       CIRCLE_BEGIN, FULL_CIRCLE);
   XDrawArc(_display_ptr, _draw_window, draw_context,
       _x + COMPOSITE_SPACE, _y + COMPOSITE_SPACE,
       (_radius - COMPOSITE_SPACE) * 2,
       (_radius - COMPOSITE_SPACE) * 2, CIRCLE_BEGIN,
       FULL_CIRCLE);
   XDrawArc(_display_ptr, *_drawing_area_pixmap,
       draw_context, _x + COMPOSITE_SPACE,
       _y + COMPOSITE_SPACE,
       (_radius - COMPOSITE_SPACE) * 2,
       (_radius - COMPOSITE_SPACE) * 2, CIRCLE_BEGIN,
       FULL_CIRCLE);
  }
  else {
   if (style == SOLID) {
    XSetForeground(_display_ptr, draw_context,
    _color_table[_color]);
   }
   if (style != DOTTED) {
    XFillArc(_display_ptr, _draw_window, draw_context,
        _x, _y, _radius * 2, _radius * 2,
        CIRCLE_BEGIN, FULL_CIRCLE);
    XFillArc(_display_ptr, *_drawing_area_pixmap,
        draw_context, _x, _y, _radius * 2,
        _radius * 2, CIRCLE_BEGIN, FULL_CIRCLE);
   }
   if (style == SOLID) {
    XSetForeground(_display_ptr, draw_context,
_color_table[BLACK]);
   }
   XDrawArc(_display_ptr, _draw_window, draw_context,
       _x, _y, _radius * 2, _radius * 2, CIRCLE_BEGIN,
       FULL_CIRCLE);
   XDrawArc(_display_ptr, *_drawing_area_pixmap,
       draw_context, _x, _y, _radius * 2, _radius *2,
```

```
            CIRCLE_BEGIN, FULL_CIRCLE);
    }
    if ((_is_selected) && (_op_handles_drawn == FALSE) &&
      (style == SOLID)) {
     draw_handles(_graphics_context, _x, _y,
              _x + (2 * _radius), _y + (2 * _radius));
     _op_handles_drawn = TRUE;
    }
    else
     if ((_is_selected) && (_op_handles_drawn == TRUE) &&
       (style == ERASE)) {
      draw_handles(_graphics_context, _x, _y,
               _x + (2 * _radius), _y + (2 * _radius));
      _op_handles_drawn = FALSE;
     }
   }
   draw_text(style);
  }
}


/*********************************************************
* Returns true if the coordinates are located within either
* the operator or one of its text strings.
*********************************************************/

BOOLEAN OperatorObject::hit(int x, int y) {

 if (_is_deleted)
  return FALSE;
 else {
  if (_name_ptr != NULL) {
   if (strlen(_name_ptr) != 0)
    if (((x >= _name_x) && (x <= (_name_x + _name_width))) &&
      (y >= _name_y - _name_height) && (y <= _name_y)) {
     _name_selected = TRUE;
     return TRUE;
    }
  }
  if (_met_string_ptr != NULL) {
   if (strlen(_met_string_ptr) != 0)
    if (((x >= _met_x) && (x <= (_met_x + _met_width))) &&
      (y >= _met_y - _met_height) && (y <= _met_y)) {
     _met_selected = TRUE;
     return TRUE;
    }
  }
  if (_is_terminator) {
   if (((x >= _x) && (x <= (_x + (3 * _radius)))) &&
     (y >= _y) && (y <= (_y + (2 * _radius))))
    return TRUE;
   else
    return FALSE;
  }
  else {
   if (((x >= _x) && (x <= (_x + (2 * _radius)))) &&
     (y >= _y) && (y <= (_y + (2 * _radius))))
    return TRUE;
   else
    return FALSE;
  }
 }
}

/*********************************************************
* Returns true if the coordinates are located within either
* the operator or one of its text strings.
* Just like hit() except that it does not select anything
*********************************************************/
```

```
BOOLEAN OperatorObject::over(int x, int y) {                    //@4

 if (_is_deleted)
  return FALSE;
 else {
  if (_name_ptr != NULL) {
   if (strlen(_name_ptr) != 0)
    if (((x >= _name_x) && (x <= (_name_x + _name_width))) &&
      (y >= _name_y - _name_height) && (y <= _name_y)) {
     return TRUE;
    }
  }
  if (_met_string_ptr != NULL) {
   if (strlen(_met_string_ptr) != 0)
    if (((x >= _met_x) && (x <= (_met_x + _met_width))) &&
      (y >= _met_y - _met_height) && (y <= _met_y)) {
     return TRUE;
    }
  }
  if (_is_terminator) {
   if (((x >= _x) && (x <= (_x + (3 * _radius)))) &&
     (y >= _y) && (y <= (_y + (2 * _radius))))
    return TRUE;
   else
    return FALSE;
  }
  else {
   if (((x >= _x) && (x <= (_x + (2 * _radius)))) &&
     (y >= _y) && (y <= (_y + (2 * _radius))))
    return TRUE;
   else
    return FALSE;
  }
 }
}
```

```
/*********************************************************
* Returns the coordinates of the center of the given
* operator.
*********************************************************/

XYPAIR OperatorObject::center() {
 XYPAIR temp_pair;

 if (_is_terminator) {
  temp_pair.x = _x + (int) ((float) _radius * 1.5);
  temp_pair.y = _y + _radius;
 }
 else {
  temp_pair.x = _x + _radius;
  temp_pair.y = _y + _radius;
 }
 return temp_pair;
}

/*********************************************************
* Given the last coordinate, returns the point on the
* circumference of the operator where streams should begin
* or end.
*********************************************************/

XYPAIR OperatorObject::intercept(int x, int y) {
 int distance;
 float slope;
 XYPAIR temp_pair, obj_center;

 obj_center = center();
 if (_is_terminator) {
  slope = (float) (y - obj_center.y) /
                    (float) (x - obj_center.x);
```

```
 if (fabs(slope) >= (2.0 / 3.0)) {
  if (y <= _y)
   temp_pair.y = _y;
  else
   temp_pair.y = _y + (_radius * 2);
  temp_pair.x = (int) ((float) (temp_pair.y - obj_center.y) /
                        slope) +  obj_center.x;
 }
 else {
  if (x <= _x)
   temp_pair.x = _x;
  else
   temp_pair.x = _x + (_radius * 3);
  temp_pair.y = (int) ((float) (temp_pair.x - obj_center.x) *
                        slope) + obj_center.y;
 }
}
else {
 distance = (int) sqrt(((x - obj_center.x) * (x - obj_center.x)) +
            (y - obj_center.y) * (y - obj_center.y));
 temp_pair.x = obj_center.x + (int)
          (((float) _radius/ (float) distance) *
                    (float) (x - obj_center.x));
 temp_pair.y = obj_center.y + (int)
          (((float) _radius/(float) distance) *
           (float) (y - obj_center.y));
}
return temp_pair;
}

/*********************************************************
*  Moves the operator the given distance.
*********************************************************/

void OperatorObject::move(int x, int y) {
 _x += x;
 _y += y;
 if (_x < 0)
  _x = 0;
 if (_y < 0)
  _y = 0;
 _name_x += x;
 _name_y += y;
 _met_x += x;
 _met_y += y;
}

/*********************************************************
* Relocates the operator to the given position.  x and y
* represent the center of the operator.
*********************************************************/

void OperatorObject::set_location(int x, int y) {
 int old_x = _x, old_y = _y;

 if (_is_terminator)
  _x = x - (int) ((float) _radius * 1.5);
 else
  _x = x - _radius;
 _y = y - _radius;
 if (_x < 0)
  _x = 0;
 if (_y < 0)
  _y = 0;
 if (_name_x == NULL_VALUE)
  set_default_name_location();
 else {
  _name_x += _x - old_x;
  _name_y += _y - old_y;
```

```
 }
 if (_met_x == NULL_VALUE)
  set_default_met_location();
 else {
  _met_x += _x - old_x;
  _met_y += _y - old_y;
 }
 reset_handles_drawn_state();
}

/*******************************************************
* Included for symmetry.  Streams delete themselves when
* their operators are deleted, so this function is included
* for possible future use.
*******************************************************/

void OperatorObject::delete_notify(CLASS_DEF class_type,
                int deleted_object_id) {}

/*******************************************************
*  replace the name
*******************************************************/

void OperatorObject::replace_name(char *new_name) {

 delete _name_ptr;
 _name_ptr = strdup(new_name);
 set_text_dimensions();
 set_default_name_location();
}

/*******************************************************
* unselect the object
*******************************************************/

void OperatorObject::unselect() {

 erase();
 _is_selected = FALSE;
 _name_selected = FALSE;
 _met_selected = FALSE;
 draw(SOLID);
}

/*******************************************************
*  select the object
*******************************************************/
void OperatorObject::select() {

 _is_selected = TRUE;
 draw(SOLID);
}

/*******************************************************
* Returns TRUE if one of the handles is in the given
* location.
*******************************************************/

BOOLEAN OperatorObject::hit_handle(int x, int y) {

 if ((_x <= x) && (x <= (_x + HANDLESIZE)) &&
              (_y <= y) && (y <= (_y + HANDLESIZE))) {
  _handle_selected = UPPERLEFT;
  return TRUE;
 }
 if ((_x <= x) && (x <= (_x + HANDLESIZE)) &&
   ((_y + 2 * _radius - HANDLESIZE)<= y) &&
   (y <= (_y + 2 * _radius))) {
  _handle_selected = LOWERLEFT;
  return TRUE;
```

```
}
if (_is_terminator) {
 if (((_x + 3 * _radius - HANDLESIZE) <= x) &&
   (x <= (_x + 3 * _radius)) &&
   (_y <= y) && (y <= (_y + HANDLESIZE))) {
  _handle_selected = UPPERRIGHT;
  return TRUE;
 }
 if (((_x + 3 * _radius - HANDLESIZE) <= x) &&
   (x <= (_x + 3 * _radius)) &&
   ((_y + 2 * _radius - HANDLESIZE) <= y) &&
    (y <= (_y + 2 * _radius))) {
  _handle_selected = LOWERRIGHT;
  return TRUE;
 }
}
else {
 if (((_x + (_radius * 2) - HANDLESIZE) <= x) &&
   (x <= (_x +  2 * _radius)) &&
   (_y <= y) && (y <= (_y + HANDLESIZE))) {
  _handle_selected = UPPERRIGHT;
  return TRUE;
 }
 if (((_x + 2 * _radius - HANDLESIZE) <= x) &&
   (x <= (_x + 2 * _radius)) &&
   ((_y + 2 * _radius - HANDLESIZE) <= y) &&
   (y <= (_y + 2 * _radius))) {
  _handle_selected = LOWERRIGHT;
  return TRUE;
 }
}
_handle_selected = NONE;
return FALSE;
}
```

```
/*********************************************************
* Returns TRUE if one of the handles is in the given
* location.  Does not select anything.
*********************************************************/

BOOLEAN OperatorObject::over_handle(int x, int y) {          //@4

 if (!_op_handles_drawn) return FALSE;

 if ((_x <= x) && (x <= (_x + HANDLESIZE)) &&
              (_y <= y) && (y <= (_y + HANDLESIZE))) {
  return TRUE;
 }
 if ((_x <= x) && (x <= (_x + HANDLESIZE)) &&
   ((_y + 2 * _radius - HANDLESIZE)<= y) &&
   (y <= (_y + 2 * _radius))) {
  return TRUE;
 }
 if (_is_terminator) {
  if (((_x + 3 * _radius - HANDLESIZE) <= x) &&
    (x <= (_x + 3 * _radius)) &&
    (_y <= y) && (y <= (_y + HANDLESIZE))) {
   return TRUE;
  }
  if (((_x + 3 * _radius - HANDLESIZE) <= x) &&
    (x <= (_x + 3 * _radius)) &&
    ((_y + 2 * _radius - HANDLESIZE) <= y) &&
     (y <= (_y + 2 * _radius))) {
   return TRUE;
  }
 }
 else {
  if (((_x + (_radius * 2) - HANDLESIZE) <= x) &&
    (x <= (_x +  2 * _radius)) &&
    (_y <= y) && (y <= (_y + HANDLESIZE))) {
```

```
    return TRUE;
   }
   if (((_x + 2 * _radius - HANDLESIZE) <= x) &&
     (x <= (_x + 2 * _radius)) &&
     ((_y + 2 * _radius - HANDLESIZE) <= y) &&
     (y <= (_y + 2 * _radius))) {
    return TRUE;
   }
  }
  return FALSE;
}

/*********************************************************
*  When the handle is dragged, the operator is resized.
*********************************************************/

void OperatorObject::move_handle(int x, int y) {
 int radius_change = y / 2;

 //** used to eliminate "floating" due to roundoff

 draw(DOTTED);
 if (_handle_selected == UPPERLEFT) {
  _radius -= radius_change;
  if (_radius >= MINRADIUS) {
   if (_is_terminator)
    _x += 3 * radius_change;
   else
    _x += 2 * radius_change;
   _y += 2 * radius_change;
  }
  else
   _radius += radius_change;
 }
 else
 if (_handle_selected == UPPERRIGHT) {
  _radius -= radius_change;
  if (_radius >= MINRADIUS)
   _y += 2 * radius_change;
  else
   _radius += radius_change;
 }
 else
  if (_handle_selected == LOWERRIGHT) {
   _radius += radius_change;
   if (_radius < MINRADIUS)
    _radius -= radius_change;
  }
  else
   if (_handle_selected == LOWERLEFT) {
    _radius += radius_change;
    if (_radius >= MINRADIUS) {
     if (_is_terminator)
      _x -= 3 * radius_change;
     else
      _x -= 2 * radius_change;
    }
    else
     _radius -= radius_change;
   }
// Places the text in default location. May not be desired.
// set_default_text_location();
 draw(DOTTED);
}


/*********************************************************
// Changes the MET to that entered in the properties dialog
// box.
*********************************************************/
```

```
//void OperatorObject::set_constraint(int constraint) {
// char buffer[INPUT_LINE_SIZE];
// int old_met = _met;
//
// _met = constraint;
// delete _met_string_ptr;
// if (_met == UNDEFINED_TIME)
//   _met_string_ptr = NULL;
// else {
//   if (_met < 0)
//     sprintf(buffer, "%d us", -_met);
//   else
//     sprintf(buffer, "%d ms", _met);
//   _met_string_ptr = strdup(buffer);
// }
// set_text_dimensions();
// if (old_met == UNDEFINED_TIME)
//   set_default_met_location();
//}

/*********************************************************
* Changes the font of the selected text string.
*********************************************************/

void OperatorObject::set_object_font(int font_id) {

 if (_name_selected)
   _name_font = font_id;
 if (_met_selected)
   _met_font = font_id;
 set_text_dimensions();
}

/*********************************************************
* Moves the text.
*********************************************************/

void OperatorObject::move_text(int x, int y) {

 erase_text();
 if (_name_selected) {
  _name_x += x;
  _name_y += y;
 }
 else
  if (_met_selected) {
   _met_x += x;
   _met_y += y;
  }
 draw_text(SOLID);
}

/*********************************************************
  Handles undeleted operators.
*********************************************************/

void OperatorObject::undelete_notify(CLASS_DEF class_type,
                                     int deleted_obj_id) {

 if ((class_type == OPERATOROBJECT) &&
    (deleted_obj_id == _id)) {
   _is_deleted = FALSE;
   _is_modified = TRUE;
   _is_selected = FALSE;
   _name_selected = FALSE;
   _met_selected = FALSE;
   _handle_selected = NONE;
  }
}
```

```
/*********************************************************
* reset the handles of the object
*********************************************************/

void OperatorObject::reset_handles_drawn_state() {

 _op_handles_drawn = FALSE;
 _name_handles_drawn = FALSE;
 _met_handles_drawn = FALSE;
}

/*********************************************************
* get text height
*********************************************************/

int OperatorObject::text_height() {

 if (_name_selected)
  return _name_height;
 else
  if (_met_selected)
   return _met_height;
  else
   return 0;
}

/*********************************************************
* get text width
*********************************************************/

int OperatorObject::text_width() {

 if (_name_selected)
  return _name_width;
 else
  if (_met_selected)
   return _met_width;
  else
   return 0;
}

/*********************************************************
*   Moves the text to the desired location.
*********************************************************/

void OperatorObject::text_locate(int x, int y) {

 if (_met_selected) {
  _met_x = x - _met_width / 2;
  _met_y = y + _met_height / 2;
 }
 else
  if (_name_selected) {
   _name_x = x - _name_width / 2;
   _name_y = y + _name_height / 2;
  }
}

/*********************************************************
* set the handle to selected
*********************************************************/

void OperatorObject::set_handle_selected(int handle) {

 _handle_selected = handle;
 _is_selected = TRUE;
}


/*********************************************************
```

```
*                end of operator_object.C
*************************************************************/
```

```
/* ********************************************************

Name:        operator_property_menu.H
Author:      Wes Hester
Program:     graph_editor
Date Modified: 29 Aug 96
Remarks:

********************************************************
Modification History:

 @1    WH   29 Aug 96
     New header created so that operator_property_menu.C
          functions

********************************************************
*/

#ifndef operator_property_menu_h
#define operator_property_menu_h 1

extern OperatorObject op_being_updated;

extern BOOLEAN valid_num_string(char *num);

// operator  property menu


void operator_name_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void operator_type_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void op_state_query_cb(Widget, XtPointer which,
                XtPointer cbs);

void op_initial_state_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void op_latency_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void op_latency_unit_cb(Widget, XtPointer which,
                XtPointer cbs);

void operator_ok_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void operator_cancel_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void operator_property_dialog(Widget parent_widget, int x, int y);

#endif
```

```
/*********************************************************
*  mini_sde.C
*
*  Author: Mantak
*  Date: 7 Sept 96
*
*  Remarks: This file provides the mini-sde calls from the graph_editor
*           module.
*********************************************************/


#include <string.h>
#include <stdio.h>
#include <stdlib.h>

typedef char* STRING;


void write_to_file(STRING &text) {

 FILE *fp;

 /* write text string to file */
 fp = fopen("sdedatatransfer.txt", "w");
 fprintf(fp, text);
 fclose(fp);

}

void read_from_file(STRING &text) {

 FILE *fp;
 char buffer[1000];
 int i;

 /* read text string from file */
 fp = fopen("sdedatatransfer.txt", "r");
 i = 0;
 while((buffer[i] = getc(fp)) != EOF) i++;
 buffer[i] = '\0';

 fclose(fp);
 delete(text);
 text = strdup(buffer);

}


void EDIT_IF_COND(STRING &text) {

 write_to_file(text);

 /* issue system call */
 system("./exp_sde.exe -geom 500x500+100+100 sdedatatransfer.txt");

 read_from_file(text);

}


void EDIT_OUTPUT_GUARD(STRING &text) {

 write_to_file(text);

 /* issue system call */
 system("./og_sde.exe -geom 500x500+100+100 sdedatatransfer.txt");

 read_from_file(text);

}
```

```
void EDIT_EXCEPTION(STRING &text) {

 write_to_file(text);

 /* issue system call */
 system("./exception_sde.exe -geom 500x500+100+100
sdedatatransfer.txt");

 read_from_file(text);

}


void EDIT_TIMER_OP(STRING &text) {

 write_to_file(text);

 /* issue system call */
 system("./timer_op_sde.exe -geom 500x500+100+100
sdedatatransfer.txt");

 read_from_file(text);

}
```

```
/*********************************************************

Name:        operator_property_menu.C
Author:      Wes Hester
Program:     graph_editor
Date Modified: 29 Aug 96
Remarks:

*********************************************************
Modification History:

 Baseline taken from Man-Tak Shing

 @1   WH   31 Aug 96
    Converted original program from a ge_interface.h
        implementation to a GraphObject impementation.  Also
        added checks for NULL values when passing data to
        XtVaSetValues, since it would crash the program without
        it.

        Note that the op_being_updated is the object being updated
        similar to the st_being_updated in the stream_property_menu.C

        Note: Need to create button widgets..this is only a dummy
widget

 @2   WH   1 Sept 96
    Added the copy_back and redraw graph method calls

 @3   WH   2 Sept 96
    Added the Timing Info Button input methods ECP #7

 @4   WH   7 Sept 96
    The set the met value of a terminator to always 0 (zero) is
        done in the consturctor of an object in operator_object.H

*********************************************************
*/


#include <Xm/BulletinB.h>
#include <Xm/DrawingA.h>
#include <Xm/DrawnB.h>
#include <Xm/TextF.h>
#include <Xm/Form.h>
#include <Xm/Frame.h>
#include <Xm/LabelG.h>
#include <Xm/List.h>
#include <Xm/MainW.h>
#include <Xm/MessageB.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>
#include <Xm/ScrolledW.h>
#include <Xm/SelectioB.h>
#include <Xm/Separator.h>
#include <Xm/TextF.h>
#include <Xm/Text.h>
#include <Xm/ToggleB.h>
#include <Xm/ToggleBG.h>
#include <X11/Xatom.h>
#include <stdlib.h>
#include <string.h>
#include <stream.h>
#include "ge_defs.h"
#include "ge_interface.h"
#include "graph_object_list.H"
#include "graph_object.H"
#include "llist.H"
#include "operator_object.H"
```

```
#include "stream_object.H"

#include "mini_sde.C"

extern OperatorObject op_being_updated;
extern GraphObject* selected_object_ptr;
extern GraphObjectList graphic_list;

extern BOOLEAN valid_num_string(char *num);
char *text;


/*********************************************************
* get the operator name used with the ti_cb method getting the opeator
name
*********************************************************/

void operator_name_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

   Widget temp_w =  (Widget) client_data;
   char *text;

   text = XmTextFieldGetString(temp_w);
   if (text != NULL) {
      op_being_updated.replace_name(text);   XtFree(text);
   }
}


/*********************************************************
* add the met requriements_id called for
*********************************************************/

void add_reqid_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

   Widget list_w =  (Widget) client_data;
   char *text2;
   int n_str;

   text2 = XmTextFieldGetString(w);
   if (text2 != NULL) {
     XmString str = XmStringCreateSimple(text2);
     XtVaGetValues(list_w,
                   XmNitemCount, &n_str,
                   NULL);
     XmListAddItemUnselected(list_w,str,n_str+1);
     XtFree((char*)str);
     XtFree(text2);
   }

   XtUnmanageChild(XtParent(w));


}


/*********************************************************
* cancel_cb button
*********************************************************/

void cancel_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

  //* hey don't do anything...just unmanage the widget
  XtUnmanageChild((Widget)client_data);

}
```

```
/*******************************************************
* get the operator name used with the id_cb to set the infomal description
field
*******************************************************/

void informal_desc_cb(Widget w, XtPointer client_data,
                XtPointer call_data) {

   Widget temp_w =  (Widget) client_data;
   char *text;

   XmTextVerifyCallbackStruct *cbs = (XmTextVerifyCallbackStruct*)
call_data;

   text = XmTextGetString(temp_w);

   if (text != NULL) {
      op_being_updated.set_operator_informal_desc(strdup(text));
      XtFree(text);
   }

}

/*******************************************************
* get the operator name used with the m_cb to set the fomal description
field
*******************************************************/

void formal_desc_cb(Widget w, XtPointer client_data,
                XtPointer call_data) {

   Widget temp_w =  (Widget) client_data;
   char *text;

   XmTextVerifyCallbackStruct *cbs = (XmTextVerifyCallbackStruct*)
call_data;

   text = XmTextGetString(temp_w);

   if (text != NULL) {
      op_being_updated.set_operator_formal_desc(strdup(text));
      XtFree(text);
   }

}


/*******************************************************
* get the operator set met
*******************************************************/

void operator_met_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {


   Widget temp_w =  (Widget) client_data;
   char *text;

   text = XmTextFieldGetString(temp_w);

   if (valid_num_string(text))
      op_being_updated.set_met(atoi(text));

   XtFree(text);


}

/*******************************************************
```

```
* get the operator set period
*******************************************************/

void operator_period_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

   Widget temp_w =  (Widget) client_data;
   char *text;

   text = XmTextFieldGetString(temp_w);

   if (valid_num_string(text))
      op_being_updated.set_period(atoi(text));

   XtFree(text);
}

/*******************************************************
* get the operator set mrt
*******************************************************/

void operator_mrt_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {


   Widget temp_w =  (Widget) client_data;
   char *text;

   text = XmTextFieldGetString(temp_w);

   if (valid_num_string(text))
      op_being_updated.set_mrt(atoi(text));

   XtFree(text);
}

/*******************************************************
* get the operator set mcp
*******************************************************/

void operator_mcp_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {


   Widget temp_w =  (Widget) client_data;
   char *text;

   text = XmTextFieldGetString(temp_w);

   if (valid_num_string(text))
      op_being_updated.set_mcp(atoi(text));

   XtFree(text);


}

/*******************************************************
* get the operator finish within value
*******************************************************/

void operator_finish_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {


   Widget temp_w =  (Widget) client_data;
   char *text;
```

```
  text = XmTextFieldGetString(temp_w);

  if (valid_num_string(text))
    op_being_updated.set_fw(atoi(text));

  XtFree(text);
}


/*********************************************************
* get the operator finish within value
*********************************************************/

void tc_if_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

  Widget temp_w =  (Widget) client_data;
  char *text;

  // get the currentif_condition char*
  text =  op_being_updated.if_condition();

  // now call the mini_sde.C
  if (text == NULL)
    text = strdup("");

  //** call to mini-sde
  EDIT_IF_COND(text);

  //** now set the pointer after we return
  op_being_updated.set_if_condition(text);

  free(text);
}
```

```
/*********************************************************
* get the operator met  unit value
*********************************************************/

void set_met_unit_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

    op_being_updated.set_met_unit((int) client_data);

}

/*********************************************************
* get the operator period unit value
*********************************************************/

void set_period_unit_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

    op_being_updated.set_period_unit((int) client_data);

}

/*********************************************************
* get the operator finish unit value
*********************************************************/

void set_finish_unit_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

  op_being_updated.set_fw_unit((int) client_data);

}

/*********************************************************
```

```
* get the operator mrt unit value
*********************************************************/

void set_mrt_unit_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

     op_being_updated.set_mrt_unit((int) client_data);

}

/*********************************************************
* get the operator mcp  unit value
*********************************************************/

void set_mcp_unit_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

     op_being_updated.set_mcp_unit((int) client_data);

}

/*********************************************************
* get operator return selected button
*********************************************************/
void operator_r_o_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

 XtUnmanageChild((Widget)client_data);

}


/*********************************************************
* append to met list and return
*********************************************************/
```



```
void met_r_o_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

 Widget list_w = (Widget) client_data;

 XmString *str_list;
 char    *text;
 int     n_str;

 XtVaGetValues(list_w,
                XmNitemCount, &n_str,
                XmNitems,    &str_list,
                NULL);
 if (op_being_updated._met_reqmts == NULL)
  op_being_updated._met_reqmts = new LList;
 else
  op_being_updated._met_reqmts->clear();

 for (int i = 0; i < n_str; i++) {
  XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
  op_being_updated._met_reqmts->append(text);
 }

 XtUnmanageChild(XtParent(w));

}

/*********************************************************
* append to period list and return
*********************************************************/
void period_r_o_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

 Widget list_w = (Widget) client_data;
```

```
 XmString *str_list;
 char    *text;
 int     n_str;

 XtVaGetValues(list_w,
                  XmNitemCount, &n_str,
                  XmNitems,    &str_list,
                  NULL);
 if (op_being_updated._period_reqmts == NULL)
  op_being_updated._period_reqmts = new LList;
 else
  op_being_updated._period_reqmts->clear();

 for (int i = 0; i < n_str; i++) {
  XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
  op_being_updated._period_reqmts->append(text);
 }

 XtUnmanageChild(XtParent(w));

}

/*********************************************************
* append to finish list and return
*********************************************************/
void finish_r_o_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

 Widget list_w = (Widget) client_data;

 XmString *str_list;
 char    *text;
 int     n_str;

 XtVaGetValues(list_w,
                  XmNitemCount, &n_str,
                  XmNitems,    &str_list,
                  NULL);
 if (op_being_updated._fw_reqmts == NULL)
  op_being_updated._fw_reqmts = new LList;
 else
  op_being_updated._fw_reqmts->clear();

 for (int i = 0; i < n_str; i++) {
  XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
  op_being_updated._fw_reqmts->append(text);
 }

 XtUnmanageChild(XtParent(w));

}

/*********************************************************
* append to mrt list and return
*********************************************************/
void mrt_r_o_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

 Widget list_w = (Widget) client_data;

 XmString *str_list;
 char    *text;
 int     n_str;

 XtVaGetValues(list_w,
                  XmNitemCount, &n_str,
                  XmNitems,    &str_list,
```

```
                    NULL);
  if (op_being_updated._mrt_reqmts == NULL)
    op_being_updated._mrt_reqmts = new LList;
  else
    op_being_updated._mrt_reqmts->clear();

  for (int i = 0; i < n_str; i++) {
    XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
    op_being_updated._mrt_reqmts->append(text);
  }

  XtUnmanageChild(XtParent(w));

}

/*********************************************************
* append to mcp list and return
*********************************************************/
void mcp_r_o_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

  Widget list_w = (Widget) client_data;

  XmString *str_list;
  char     *text;
  int      n_str;

  XtVaGetValues(list_w,
                    XmNitemCount, &n_str,
                    XmNitems,     &str_list,
                    NULL);
  if (op_being_updated._mcp_reqmts == NULL)
    op_being_updated._mcp_reqmts = new LList;
  else
    op_being_updated._mcp_reqmts->clear();

  for (int i = 0; i < n_str; i++) {
    XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
    op_being_updated._mcp_reqmts->append(text);
  }

  XtUnmanageChild(XtParent(w));

}

/*********************************************************
* append to keywords list and return
*********************************************************/
void keywords_r_o_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

  Widget list_w = (Widget) client_data;

  XmString *str_list;
  char     *text;
  int      n_str;

  XtVaGetValues(list_w,
                    XmNitemCount, &n_str,
                    XmNitems,     &str_list,
                    NULL);
  if (op_being_updated._key_word_list == NULL)
    op_being_updated._key_word_list = new LList;
  else
    op_being_updated._key_word_list->clear();

  for (int i = 0; i < n_str; i++) {
```

```
  XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
  op_being_updated._key_word_list->append(text);
 }

 XtUnmanageChild(XtParent(w));

}

/*********************************************************
* append to triggering list and return
*********************************************************/
void tc_r_o_cb(Widget w, XtPointer client_data,
               XtPointer cb_struct_ptr) {

 Widget list_w = (Widget) client_data;

 XmString *str_list;
 char    *text;
 int     n_str;

 XtVaGetValues(list_w,
                  XmNitemCount, &n_str,
                  XmNitems,    &str_list,
                  NULL);
 if (op_being_updated._trigger_reqmts == NULL)
  op_being_updated._trigger_reqmts = new LList;
 else
  op_being_updated._trigger_reqmts->clear();

 for (int i = 0; i < n_str; i++) {
  XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
  op_being_updated._trigger_reqmts->append(text);
 }

 XtUnmanageChild(XtParent(w));

}


/*********************************************************
* append to triggering list and return
*********************************************************/
void tcstreams_r_o_cb(Widget w, XtPointer client_data,
               XtPointer cb_struct_ptr) {

 Widget list_w = (Widget) client_data;

 XmString *str_list;
 char    *text;
 int     n_str;

 XtVaGetValues(list_w,
                  XmNitemCount, &n_str,
                  XmNitems,    &str_list,
                  NULL);
 if (op_being_updated._trigger_streams == NULL)
  op_being_updated._trigger_streams = new LList;
 else
  op_being_updated._trigger_streams->clear();

 for (int i = 0; i < n_str; i++) {
  XmStringGetLtoR(str_list[i], XmSTRING_DEFAULT_CHARSET,
&text);
  op_being_updated._trigger_streams->append(text);
 }

 XtUnmanageChild(XtParent(w));
```

```
}


/*********************************************************
* set the ms
*********************************************************/
void set_the_ms(Widget w, XtPointer which,
                XtPointer cbs) {

 op_being_updated.set_met((int)which);

}


/*********************************************************
* add the met requirements id
*********************************************************/

void tcstreams_reqid_add_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {
   int x = 400;
   int y = 200;

   Widget dialog,
        cancel_widget,
        add_widget,
        req_id;

   XmString t,
         r_o_button;


   Arg args[10];

   int ac;

   char *prompt, buffer[INPUT_LINE_SIZE];

   Widget list_w = (Widget) client_data;

   /*****************************************************
   * build the formal description  property dialog box
   *****************************************************/
   ac = 0;
   XtSetArg(args[ac], XmNheight, 140); ac++;
   XtSetArg(args[ac], XmNwidth,  100); ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
   t = XmStringCreateSimple ("Add Requirements ID Dialog");
   XtSetArg(args[ac], XmNdialogTitle, t); ac++;

   dialog = XmCreateBulletinBoardDialog(w, "tcstreamsadd_reqid",
args, ac);

   XmStringFree(t);

   prompt = strdup("Requirements ID to Add:");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

   req_id =  XtVaCreateManagedWidget("req_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
                NULL);

   XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);
```

```
  if (text != NULL)
    XtVaSetValues(req_id, XmNvalue, text, NULL);

  free(prompt);

  // create the cancel button
  ac = 0;
  XtSetArg(args[ac], XmNx, 60); ac++;
  XtSetArg(args[ac], XmNy, 95); ac++;
  cancel_widget = XmCreatePushButton(dialog, " Cancel ", args, ac);
  XtManageChild(cancel_widget);


  XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);


}


/*********************************************************
* add the met requirements id
*********************************************************/

void met_reqid_add_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {
   int x = 400;
   int y = 200;

   Widget dialog,
       cancel_widget,
       add_widget,
       req_id;

   XmString t,
        r_o_button;


   Arg args[10];

   int ac;

   char *prompt, buffer[INPUT_LINE_SIZE];

   Widget list_w = (Widget) client_data;

   /*************************************************
   * build the formal description  property dialog box
   *************************************************/
   ac = 0;
   XtSetArg(args[ac], XmNheight, 140); ac++;
   XtSetArg(args[ac], XmNwidth,  100); ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
   t = XmStringCreateSimple ("Add Requirements ID Dialog");
   XtSetArg(args[ac], XmNdialogTitle, t); ac++;

   dialog = XmCreateBulletinBoardDialog(w, "metadd_reqid", args, ac);

   XmStringFree(t);

   prompt = strdup("Requirements ID to Add:");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
```

```
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

    req_id =  XtVaCreateManagedWidget("req_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
                NULL);

    XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);

    if (text != NULL)
      XtVaSetValues(req_id, XmNvalue, text, NULL);

    free(prompt);

    // create the cancel button
    ac = 0;
    XtSetArg(args[ac], XmNx, 60); ac++;
    XtSetArg(args[ac], XmNy, 95); ac++;
    cancel_widget = XmCreatePushButton(dialog, " Cancel ", args, ac);
    XtManageChild(cancel_widget);


    XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

    XtManageChild(dialog);

    XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

    XtPopup(XtParent(dialog), XtGrabNone);
    XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);


}


/*********************************************************
* add the period requirements id
*********************************************************/

void period_reqid_add_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

    int x = 400;
    int y = 200;

    Widget dialog,
        cancel_widget,
        add_widget,
        req_id;

    XmString t,
         r_o_button;


    Arg args[10];

    int ac;

    char *prompt, buffer[INPUT_LINE_SIZE];

    Widget list_w = (Widget) client_data;

    /*****************************************************
    * build the formal description  property dialog box
```

```
*****************************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 140); ac++;
  XtSetArg(args[ac], XmNwidth,  100); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Add Requirements ID Dialog");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;

  dialog = XmCreateBulletinBoardDialog(w, "periodadd_reqid", args,
ac);

  XmStringFree(t);

  prompt = strdup("Requirements ID to Add:");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

  req_id =  XtVaCreateManagedWidget("req_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
                NULL);

  XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);

  if (text != NULL)
   XtVaSetValues(req_id, XmNvalue, text, NULL);

  free(prompt);

  // create the cancel button
  ac = 0;
  XtSetArg(args[ac], XmNx, 60); ac++;
  XtSetArg(args[ac], XmNy, 95); ac++;
  cancel_widget = XmCreatePushButton(dialog, "  Cancel  ", args, ac);
  XtManageChild(cancel_widget);


  XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);



}


/*********************************************************
* add the finish within  requirements id
*********************************************************/

void finish_reqid_add_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

  int x = 400;
  int y = 200;

  Widget dialog,
      cancel_widget,
      add_widget,
```

```
    req_id;

XmString t,
     r_o_button;


Arg args[10];

int ac;

char *prompt, buffer[INPUT_LINE_SIZE];

Widget list_w = (Widget) client_data;

/*****************************************************
* build the formal description  property dialog box
*****************************************************/
ac = 0;
XtSetArg(args[ac], XmNheight, 140); ac++;
XtSetArg(args[ac], XmNwidth,  100); ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
t = XmStringCreateSimple ("Add Requirements ID Dialog");
XtSetArg(args[ac], XmNdialogTitle, t); ac++;

dialog = XmCreateBulletinBoardDialog(w, "finishreqid", args, ac);

XmStringFree(t);

prompt = strdup("Requirements ID to Add:");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);
```

```
   req_id =  XtVaCreateManagedWidget("freq_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
                NULL);

   XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);

   if (text != NULL)
    XtVaSetValues(req_id, XmNvalue, text, NULL);

   free(prompt);

   // create the cancel button
   ac = 0;
   XtSetArg(args[ac], XmNx, 60); ac++;
   XtSetArg(args[ac], XmNy, 95); ac++;
   cancel_widget = XmCreatePushButton(dialog,  "  Cancel  ", args, ac);
   XtManageChild(cancel_widget);


   XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);



}
```

```
/*********************************************************
* add the triggering requirements id
*********************************************************/

void triggering_reqid_add_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {


   int x = 400;
   int y = 200;

   Widget dialog,
        cancel_widget,
        add_widget,
        req_id;

   XmString t,
         r_o_button;


   Arg args[10];

   int ac;

   char *prompt, buffer[INPUT_LINE_SIZE];

   Widget list_w = (Widget) client_data;

   /*************************************************
   * build the formal description  property dialog box
   *************************************************/
   ac = 0;
   XtSetArg(args[ac], XmNheight, 140); ac++;
   XtSetArg(args[ac], XmNwidth,  100); ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
   t = XmStringCreateSimple ("Add Requirements ID Dialog");
   XtSetArg(args[ac], XmNdialogTitle, t); ac++;

   dialog = XmCreateBulletinBoardDialog(w, "tgreqid", args, ac);

   XmStringFree(t);

   prompt = strdup("Requirements ID to Add:");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

   req_id =  XtVaCreateManagedWidget("req_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
                NULL);

   XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);

   if (text != NULL)
    XtVaSetValues(req_id, XmNvalue, text, NULL);

   free(prompt);

   // create the cancel button
   ac = 0;
   XtSetArg(args[ac], XmNx, 60); ac++;
   XtSetArg(args[ac], XmNy, 95); ac++;
   cancel_widget = XmCreatePushButton(dialog, "  Cancel  ", args, ac);
   XtManageChild(cancel_widget);
```

```
   XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);


}


/*******************************************************
* add the mcp requirements id
*******************************************************/

void mcp_reqid_add_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {


   int x = 400;
   int y = 200;

   Widget dialog,
        cancel_widget,
        add_widget,
        req_id;

   XmString t,
         r_o_button;
```



```
Arg args[10];

int ac;

char *prompt, buffer[INPUT_LINE_SIZE];

Widget list_w = (Widget) client_data;

/*******************************************************
* build the formal description  property dialog box
*******************************************************/
ac = 0;
XtSetArg(args[ac], XmNheight, 140); ac++;
XtSetArg(args[ac], XmNwidth,  100); ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
t = XmStringCreateSimple ("Add Requirements ID Dialog");
XtSetArg(args[ac], XmNdialogTitle, t); ac++;

dialog = XmCreateBulletinBoardDialog(w, "mcpreqid", args, ac);

XmStringFree(t);

prompt = strdup("Requirements ID to Add:");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

req_id =  XtVaCreateManagedWidget("req_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
```

```
                NULL);

  XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);

  if (text != NULL)
   XtVaSetValues(req_id, XmNvalue, text, NULL);

  free(prompt);

  // create the cancel button
  ac = 0;
  XtSetArg(args[ac], XmNx, 60); ac++;
  XtSetArg(args[ac], XmNy, 95); ac++;
  cancel_widget = XmCreatePushButton(dialog, " Cancel ", args, ac);
  XtManageChild(cancel_widget);


  XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);


}


/*******************************************************
* add the met requirements id
*******************************************************/
```

```
void mrt_reqid_add_cb(Widget w, XtPointer client_data,
            XtPointer cb_struct_ptr) {


  int x = 400;
  int y = 200;

  Widget dialog,
      cancel_widget,
      add_widget,
      req_id;

  XmString t,
       r_o_button;


  Arg args[10];

  int ac;

  char *prompt, buffer[INPUT_LINE_SIZE];

  Widget list_w = (Widget) client_data;

  /*****************************************************
  * build the formal description  property dialog box
  *****************************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 140); ac++;
  XtSetArg(args[ac], XmNwidth,  100); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Add Requirements ID Dialog");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;
```

```
dialog = XmCreateBulletinBoardDialog(w, "mrtreqid", args, ac);

XmStringFree(t);

prompt = strdup("Requirements ID to Add:");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

req_id =  XtVaCreateManagedWidget("req_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
                NULL);

XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);

if (text != NULL)
  XtVaSetValues(req_id, XmNvalue, text, NULL);

free(prompt);

// create the cancel button
ac = 0;
XtSetArg(args[ac], XmNx, 60); ac++;
XtSetArg(args[ac], XmNy, 95); ac++;
cancel_widget = XmCreatePushButton(dialog, " Cancel ", args, ac);
XtManageChild(cancel_widget);


XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

XtManageChild(dialog);

XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

XtPopup(XtParent(dialog), XtGrabNone);

XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);


}

/*********************************************************
* add the keywords requirements id
*********************************************************/

void keywords_reqid_add_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

  int x = 400;
  int y = 200;

  Widget dialog,
      cancel_widget,
      add_widget,
      req_id;

  XmString t,
       r_o_button;


  Arg args[10];

  int ac;

  char *prompt, buffer[INPUT_LINE_SIZE];
```

```
  Widget list_w = (Widget) client_data;

  /*****************************************************
  * build the formal description  property dialog box
  *****************************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 140); ac++;
  XtSetArg(args[ac], XmNwidth,  100); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Add Requirements ID Dialog");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;

  dialog = XmCreateBulletinBoardDialog(w, "keywordsaddreqid", args,
ac);

  XmStringFree(t);

  prompt = strdup("Requirements ID to Add:");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 40,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

  req_id =  XtVaCreateManagedWidget("req_id",
                xmTextFieldWidgetClass, dialog,
                XmNx, 35,
                XmNy, 35,
                NULL);

  XtAddCallback(req_id, XmNactivateCallback, add_reqid_cb, list_w);

  if (text != NULL)
    XtVaSetValues(req_id, XmNvalue, text, NULL);

  free(prompt);

  // create the cancel button
  ac = 0;
  XtSetArg(args[ac], XmNx, 60); ac++;
  XtSetArg(args[ac], XmNy, 95); ac++;
  cancel_widget = XmCreatePushButton(dialog, "  Cancel  ", args, ac);
  XtManageChild(cancel_widget);


  XtAddCallback(cancel_widget, XmNactivateCallback, cancel_cb,
(XtPointer) dialog);

  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(req_id, XmTRAVERSE_CURRENT);


}

/*****************************************************
* delete the requirements id     NOTE: Taken from example pg 918  Vol
6A
*****************************************************/

void delete_reqid_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

 int *sel, n;
 Widget list_w = (Widget) client_data;
```

```
 if (XmListGetSelectedPos(list_w, &sel, &n)) {
  while (n--)
   XmListDeletePos(list_w,sel[n]);
  XtFree( (char*) sel);
 }
}

/*******************************************************
* get the operator keyword form the More button
*******************************************************/

void operator_keyword_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {


  int x = 400;
  int y = 200;

  Widget dialog,
      frame, add_button, delete_button,
      r_o_widget,list_w,r_o_button;

  XmString t;

  Arg args[10];

  int ac, item_count;
  char *prompt, buffer[INPUT_LINE_SIZE];

  /*****************************************************
  * build the formal description  property dialog box
  *****************************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 140); ac++;
  XtSetArg(args[ac], XmNwidth,  100); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Keywords Requirements Dialog");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;
  dialog = XmCreateBulletinBoardDialog(w, "keywordreq", args, ac);

  XmStringFree(t);


  prompt = strdup("Enter/Delete Requirements IDs");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
              XmNx, 45,
              XmNy, 15,
              XmNalignment, XmALIGNMENT_BEGINNING,
              NULL);


  //** build a frame to put the list in ...
  ac = 0;
  XtSetArg(args[ac], XmNx,90);        ac++;
  XtSetArg(args[ac], XmNy, 50);        ac++;
  frame  = XmCreateFrame(dialog, "frame", args, ac);
  XtManageChild(frame);

  free(prompt);

  // Load initial list from LList
  if (op_being_updated._mrt_reqmts != NULL) {
   XmStringTable ListArray;

   int n = op_being_updated._key_word_list->length();
   ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

   op_being_updated._key_word_list->reset();
   for (int i = 0; i < n; i++) {
```

```
        ListArray[i] =
XmStringCreateSimple(op_being_updated._key_word_list->item());
        op_being_updated._key_word_list->step();
    }


    //* create the scrolled list of stream IDs //
    ac = 0;
    XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
    XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
    list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
    XtVaSetValues(list_w,
                  XmNitems,         ListArray,
                  XmNitemCount,     n,
                  XmNvisibleItemCount,  5,
                  NULL);
    XtManageChild(list_w);
   }
   else {
    ac = 0;
    XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
    XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
    list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
    XtVaSetValues(list_w,
                  XmNvisibleItemCount,  5,
                  NULL);
    XtManageChild(list_w);
   }

   //* create the ADD button
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 48); ac++;
   add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
   XtManageChild(add_button);

   XtAddCallback(add_button , XmNactivateCallback,
keywords_reqid_add_cb,  (XtPointer) list_w);


   //* create the DELETE button
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 72); ac++;
   delete_button = XmCreatePushButton(dialog, "  Delete ", args, ac);
   XtManageChild(delete_button);

   XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb,  (XtPointer) list_w);

   //* create the return to operator dialog button
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 105); ac++;
   r_o_button = XmCreatePushButton(dialog, "  Return ", args, ac);
   XtManageChild(r_o_button);



/*********************************************************
   * now we need the call back send the button and method linked to the
button

*********************************************************/

   XtAddCallback(r_o_button , XmNactivateCallback,
keywords_r_o_cb, (XtPointer) list_w);


   XtManageChild(dialog);
```

```
    XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

    XtPopup(XtParent(dialog), XtGrabNone);

    XmProcessTraversal(w, XmTRAVERSE_CURRENT);



}

/*********************************************************
* the mrt requirements by dialog
*********************************************************/
void ti_mrt_req_cb(Widget w, XtPointer client_data,
                   XtPointer cb_struct_ptr) {


    int x = 400;
    int y = 200;

    Widget dialog,
         frame, add_button, delete_button,
         r_o_widget,list_w,r_o_button;

    XmString t;

    Arg args[10];

    int ac, item_count;
    char *prompt, buffer[INPUT_LINE_SIZE];

    /*************************************************
    * build the formal description  property dialog box
```



```
    *************************************************/
    ac = 0;
    XtSetArg(args[ac], XmNheight, 140); ac++;
    XtSetArg(args[ac], XmNwidth,  100); ac++;
    XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
    t = XmStringCreateSimple ("MRT Requirements By Dialog");
    XtSetArg(args[ac], XmNdialogTitle, t); ac++;

    dialog = XmCreateBulletinBoardDialog(w, "mrtreq", args, ac);

    XmStringFree(t);


    prompt = strdup("Enter/Delete Requirements IDs");
    XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                    XmNx, 45,
                    XmNy, 15,
                    XmNalignment, XmALIGNMENT_BEGINNING,
                    NULL);


    //** build a frame to put the list in ...
    ac = 0;
    XtSetArg(args[ac], XmNx,90);          ac++;
    XtSetArg(args[ac], XmNy, 50);         ac++;
    frame  = XmCreateFrame(dialog, "frame", args, ac);
    XtManageChild(frame);

    free(prompt);

    // Load initial list from LList
    if (op_being_updated._mrt_reqmts != NULL) {
     XmStringTable ListArray;

     int n = op_being_updated._mrt_reqmts->length();
```

```
    ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

    op_being_updated._mrt_reqmts->reset();
    for (int i = 0; i < n; i++) {
          ListArray[i] =
XmStringCreateSimple(op_being_updated._mrt_reqmts->item());
          op_being_updated._mrt_reqmts->step();
    }


    //* create the scrolled list of stream IDs //
    ac = 0;
    XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
    XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
    list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
    XtVaSetValues(list_w,
                    XmNitems,          ListArray,
                    XmNitemCount,      n,
                    XmNvisibleItemCount, 5,
                    NULL);
    XtManageChild(list_w);
   }
   else {
    ac = 0;
    XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
    XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
    list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
    XtVaSetValues(list_w,
                    XmNvisibleItemCount, 5,
                    NULL);
    XtManageChild(list_w);
   }

    //* create the ADD button
    ac = 0;
    XtSetArg(args[ac], XmNx, 10); ac++;
    XtSetArg(args[ac], XmNy, 48); ac++;
    add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
    XtManageChild(add_button);

    XtAddCallback(add_button , XmNactivateCallback,
mrt_reqid_add_cb,  (XtPointer) list_w);


    //* create the DELETE button
    ac = 0;
    XtSetArg(args[ac], XmNx, 10); ac++;
    XtSetArg(args[ac], XmNy, 72); ac++;
    delete_button = XmCreatePushButton(dialog, "  Delete ", args, ac);
    XtManageChild(delete_button);

    XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb,  (XtPointer) list_w);

    //* create the return to operator dialog button
    ac = 0;
    XtSetArg(args[ac], XmNx, 10); ac++;
    XtSetArg(args[ac], XmNy, 105); ac++;
    r_o_button = XmCreatePushButton(dialog, "  Return ", args, ac);
    XtManageChild(r_o_button);



/*********************************************************
    * now we need the call back send the button and method linked to the
button

*********************************************************/
```

```
   XtAddCallback(r_o_button , XmNactivateCallback, mrt_r_o_cb,
(XtPointer) list_w);


   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(w, XmTRAVERSE_CURRENT);



}

/*********************************************************
* the mcp requirements by dialog
*********************************************************/
void ti_mcp_req_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {



   int x = 400;
   int y = 200;

   Widget dialog,
        frame, add_button, delete_button,
        r_o_widget,list_w,r_o_button;

   XmString t;

   Arg args[10];
```



```
int ac, item_count;
char *prompt, buffer[INPUT_LINE_SIZE];

/*************************************************
* build the formal description  property dialog box
*************************************************/
ac = 0;
XtSetArg(args[ac], XmNheight, 140); ac++;
XtSetArg(args[ac], XmNwidth,  100); ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
t = XmStringCreateSimple ("MCP Requirements By Dialog");
XtSetArg(args[ac], XmNdialogTitle, t); ac++;

dialog = XmCreateBulletinBoardDialog(w, "mcpreq", args, ac);

XmStringFree(t);


prompt = strdup("Enter/Delete Requirements IDs");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 45,
                XmNy, 15,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);


//** build a frame to put the list in ...
ac = 0;
XtSetArg(args[ac], XmNx,90);          ac++;
XtSetArg(args[ac], XmNy, 50);         ac++;
frame  = XmCreateFrame(dialog, "frame", args, ac);
XtManageChild(frame);

free(prompt);
```

```
// Load initial list from LList
if (op_being_updated._mcp_reqmts != NULL) {
 XmStringTable ListArray;

 int n = op_being_updated._mcp_reqmts->length();
 ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

 op_being_updated._mcp_reqmts->reset();
 for (int i = 0; i < n; i++) {
      ListArray[i] =
XmStringCreateSimple(op_being_updated._mcp_reqmts->item());
      op_being_updated._mcp_reqmts->step();
 }


 //* create the scrolled list of stream IDs //
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                XmNitems,          ListArray,
                XmNitemCount,      n,
                XmNvisibleItemCount, 5,
                NULL);
 XtManageChild(list_w);
}
else {
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                XmNvisibleItemCount, 5,
                NULL);
 XtManageChild(list_w);
}

//* create the ADD button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 48); ac++;
add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
XtManageChild(add_button);

XtAddCallback(add_button , XmNactivateCallback,
mcp_reqid_add_cb, (XtPointer) list_w);


//* create the DELETE button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 72); ac++;
delete_button = XmCreatePushButton(dialog, " Delete ", args, ac);
XtManageChild(delete_button);

XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb, (XtPointer) list_w);

//* create the return to operator dialog button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 105); ac++;
r_o_button = XmCreatePushButton(dialog, " Return ", args, ac);
XtManageChild(r_o_button);



/*********************************************************
```

```
   * now we need the call back send the button and method linked to the
button

*******************************************************/

   XtAddCallback(r_o_button , XmNactivateCallback, mcp_r_o_cb,
(XtPointer) list_w);


   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}

/*******************************************************
* the period requirements by dialog
*******************************************************/

void ti_period_req_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {



   int x = 400;
   int y = 200;

   Widget dialog,
        frame, add_button, delete_button,
        r_o_widget,list_w,r_o_button;
```



```
XmString t;

Arg args[10];

int ac, item_count;
char *prompt, buffer[INPUT_LINE_SIZE];

/*******************************************************
* build the formal description  property dialog box
*******************************************************/
ac = 0;
XtSetArg(args[ac], XmNheight, 140); ac++;
XtSetArg(args[ac], XmNwidth,  100); ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
t = XmStringCreateSimple ("Period Requirements By Dialog");
XtSetArg(args[ac], XmNdialogTitle, t); ac++;

dialog = XmCreateBulletinBoardDialog(w, "periodreq", args, ac);

XmStringFree(t);


prompt = strdup("Enter/Delete Requirements IDs");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 45,
                XmNy, 15,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);


//** build a frame to put the list in ...
ac = 0;
XtSetArg(args[ac], XmNx,90);          ac++;
XtSetArg(args[ac], XmNy, 50);         ac++;
```

```
  frame  = XmCreateFrame(dialog, "frame", args, ac);
  XtManageChild(frame);

  free(prompt);

  // Load initial list from LList
  if (op_being_updated._period_reqmts != NULL) {
   XmStringTable ListArray;

   int n = op_being_updated._period_reqmts->length();
   ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

   op_being_updated._period_reqmts->reset();
   for (int i = 0; i < n; i++) {
        ListArray[i] =
XmStringCreateSimple(op_being_updated._period_reqmts->item());
        op_being_updated._period_reqmts->step();
   }


   //* create the scrolled list of stream IDs //
   ac = 0;
   XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
   XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
   list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
   XtVaSetValues(list_w,
                 XmNitems,          ListArray,
                 XmNitemCount,      n,
                 XmNvisibleItemCount,  5,
                 NULL);
   XtManageChild(list_w);
  }
  else {
   ac = 0;
   XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
   XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
   list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
   XtVaSetValues(list_w,
                 XmNvisibleItemCount,  5,
                 NULL);
   XtManageChild(list_w);
  }

  //* create the ADD button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 48); ac++;
  add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
  XtManageChild(add_button);

  XtAddCallback(add_button , XmNactivateCallback,
period_reqid_add_cb,  (XtPointer) list_w);


  //* create the DELETE button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 72); ac++;
  delete_button = XmCreatePushButton(dialog, " Delete ", args, ac);
  XtManageChild(delete_button);

  XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb,  (XtPointer) list_w);

  //* create the return to operator dialog button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 105); ac++;
  r_o_button = XmCreatePushButton(dialog, " Return ", args, ac);
  XtManageChild(r_o_button);
```

```
/*******************************************************
  * now we need the call back send the button and method linked to the
button

*******************************************************/

  XtAddCallback(r_o_button , XmNactivateCallback, period_r_o_cb,
(XtPointer) list_w);


  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}

/*******************************************************
* the finish within requirements by dialog
*******************************************************/

void ti_finish_req_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

  int x = 400;
  int y = 200;

  Widget dialog,
      frame, add_button, delete_button,
      r_o_widget,list_w,r_o_button;

XmString t;


Arg args[10];

int ac, item_count;
char *prompt, buffer[INPUT_LINE_SIZE];

/*******************************************************
* build the formal description  property dialog box
*******************************************************/
ac = 0;
XtSetArg(args[ac], XmNheight, 140); ac++;
XtSetArg(args[ac], XmNwidth,  100); ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
t = XmStringCreateSimple ("Finish Requirements By Dialog");
XtSetArg(args[ac], XmNdialogTitle, t); ac++;

dialog = XmCreateBulletinBoardDialog(w, "finishreq", args, ac);

XmStringFree(t);


prompt = strdup("Enter/Delete Requirements IDs");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 45,
                XmNy, 15,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);


//** build a frame to put the list in ...
ac = 0;
```

```
XtSetArg(args[ac], XmNx,110);          ac++;
XtSetArg(args[ac], XmNy, 50);          ac++;
frame  = XmCreateFrame(dialog, "frame", args, ac);
XtManageChild(frame);

free(prompt);


// Load initial list from LList
if (op_being_updated._fw_reqmts != NULL) {
 XmStringTable ListArray;

 int n = op_being_updated._fw_reqmts->length();
 ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

 op_being_updated._fw_reqmts->reset();
 for (int i = 0; i < n; i++) {
      ListArray[i] =
XmStringCreateSimple(op_being_updated._fw_reqmts->item());
      op_being_updated._fw_reqmts->step();
 }


 //* create the scrolled list of stream IDs //
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                XmNitems,          ListArray,
                XmNitemCount,      n,
                XmNvisibleItemCount,  5,
                NULL);
 XtManageChild(list_w);
}
```



```
else {
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                XmNvisibleItemCount,  5,
                NULL);
 XtManageChild(list_w);
}

//* create the ADD button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 48); ac++;
add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
XtManageChild(add_button);

XtAddCallback(add_button , XmNactivateCallback,
finish_reqid_add_cb,  (XtPointer) list_w);


//* create the DELETE button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 72); ac++;
delete_button = XmCreatePushButton(dialog, " Delete ", args, ac);
XtManageChild(delete_button);

XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb,  (XtPointer) list_w);

//* create the return to operator dialog button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
```

```
   XtSetArg(args[ac], XmNy, 105); ac++;
   r_o_button = XmCreatePushButton(dialog,  "  Return ", args, ac);
   XtManageChild(r_o_button);



/*********************************************************
   * now we need the call back send the button and method linked to the
button

*********************************************************/

   XtAddCallback(r_o_button , XmNactivateCallback, finish_r_o_cb,
(XtPointer) list_w);


   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(w, XmTRAVERSE_CURRENT);


}

/*********************************************************
* the met  requirements by dialog
*********************************************************/

void ti_met_req_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

   int x = 400;
```



```
int y = 200;

Widget dialog,
    frame, add_button, delete_button,
    r_o_widget,list_w,r_o_button;

XmString t;

Arg args[10];

int ac, item_count;
char *prompt, buffer[INPUT_LINE_SIZE];

/*********************************************************
* build the formal description  property dialog box
*********************************************************/
ac = 0;
XtSetArg(args[ac], XmNheight, 140); ac++;
XtSetArg(args[ac], XmNwidth,  100); ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
t = XmStringCreateSimple ("Met Requirements By Dialog");
XtSetArg(args[ac], XmNdialogTitle, t); ac++;

dialog = XmCreateBulletinBoardDialog(w, "metreq", args, ac);

XmStringFree(t);


prompt = strdup("Enter/Delete Requirements IDs");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 45,
                XmNy, 15,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);
```

```
//** build a frame to put the list in ...
ac = 0;
XtSetArg(args[ac], XmNx,90);          ac++;
XtSetArg(args[ac], XmNy, 50);         ac++;
frame  = XmCreateFrame(dialog, "frame", args, ac);
XtManageChild(frame);

free(prompt);

// Load initial list from LList
if (op_being_updated._met_reqmts != NULL) {
 XmStringTable ListArray;

 int n = op_being_updated._met_reqmts->length();
 ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

 op_being_updated._met_reqmts->reset();
 for (int i = 0; i < n; i++) {
      ListArray[i] =
XmStringCreateSimple(op_being_updated._met_reqmts->item());
      op_being_updated._met_reqmts->step();
 }


 //* create the scrolled list of stream IDs //
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                  XmNitems,          ListArray,
                  XmNitemCount,      n,
                  XmNvisibleItemCount,  5,
                  NULL);
 XtManageChild(list_w);
}
else {
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                  XmNvisibleItemCount,  5,
                  NULL);
 XtManageChild(list_w);
}

//* create the ADD button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 48); ac++;
add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
XtManageChild(add_button);

 XtAddCallback(add_button , XmNactivateCallback,
met_reqid_add_cb,  (XtPointer) list_w);


//* create the DELETE button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 72); ac++;
delete_button = XmCreatePushButton(dialog, "  Delete ", args, ac);
XtManageChild(delete_button);

 XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb,  (XtPointer) list_w);

//* create the return to operator dialog button
```

```
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 105); ac++;
  r_o_button = XmCreatePushButton(dialog, " Return ", args, ac);
  XtManageChild(r_o_button);



/*********************************************************
  * now we need the call back send the button and method linked to the
button

*********************************************************/

  XtAddCallback(r_o_button , XmNactivateCallback, met_r_o_cb,
(XtPointer) list_w);


  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(w, XmTRAVERSE_CURRENT);


}


/*********************************************************
* the period requirements by dialog
*********************************************************/

void tc_reqid_cb(Widget w, XtPointer client_data,
                 XtPointer cb_struct_ptr) {



int x = 400;
int y = 200;

Widget dialog,
    frame, add_button, delete_button,
    r_o_widget,list_w,r_o_button;

XmString t;

Arg args[10];

int ac, item_count;
char *prompt, buffer[INPUT_LINE_SIZE];

/*****************************************************
* build the formal description  property dialog box
*****************************************************/
ac = 0;
XtSetArg(args[ac], XmNheight, 140); ac++;
XtSetArg(args[ac], XmNwidth,  100); ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
t = XmStringCreateSimple ("TC Requirements Dialog");
XtSetArg(args[ac], XmNdialogTitle, t); ac++;

dialog = XmCreateBulletinBoardDialog(w, "tcreq", args, ac);

XmStringFree(t);


prompt = strdup("Enter/Delete Requirements IDs");
```

```
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 45,
                XmNy, 15,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);


//** build a frame to put the list in ...
ac = 0;
XtSetArg(args[ac], XmNx,90);          ac++;
XtSetArg(args[ac], XmNy, 50);         ac++;
frame  = XmCreateFrame(dialog, "frame", args, ac);
XtManageChild(frame);

free(prompt);

// Load initial list from LList
if (op_being_updated._trigger_reqmts != NULL) {
  XmStringTable ListArray;

  int n = op_being_updated._trigger_reqmts->length();
  ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

  op_being_updated._trigger_reqmts->reset();
  for (int i = 0; i < n; i++) {
        ListArray[i] =
XmStringCreateSimple(op_being_updated._trigger_reqmts->item());
        op_being_updated._trigger_reqmts->step();
  }


  //* create the scrolled list of stream IDs //
  ac = 0;
  XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
  XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
  list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
  XtVaSetValues(list_w,
                XmNitems,          ListArray,
                XmNitemCount,      n,
                XmNvisibleItemCount,  5,
                NULL);
  XtManageChild(list_w);
 }
 else {
  ac = 0;
  XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
  XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
  list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
  XtVaSetValues(list_w,
                XmNvisibleItemCount,  5,
                NULL);
  XtManageChild(list_w);
 }

 //* create the ADD button
 ac = 0;
 XtSetArg(args[ac], XmNx, 10); ac++;
 XtSetArg(args[ac], XmNy, 48); ac++;
 add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
 XtManageChild(add_button);

 XtAddCallback(add_button , XmNactivateCallback,
triggering_reqid_add_cb,  (XtPointer) list_w);


 //* create the DELETE button
 ac = 0;
 XtSetArg(args[ac], XmNx, 10); ac++;
 XtSetArg(args[ac], XmNy, 72); ac++;
 delete_button = XmCreatePushButton(dialog, "  Delete ", args, ac);
```

```
  XtManageChild(delete_button);

  XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb,  (XtPointer) list_w);

  //* create the return to operator dialog button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 105); ac++;
  r_o_button = XmCreatePushButton(dialog,  "  Return ", args, ac);
  XtManageChild(r_o_button);



/*********************************************************
  * now we need the call back send the button and method linked to the
button

*********************************************************/

  XtAddCallback(r_o_button , XmNactivateCallback, tc_r_o_cb,
(XtPointer) list_w);


  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}
```

```
/*********************************************************
* the tc stream ids
*********************************************************/

void tc_strmid_cb(Widget w, XtPointer client_data,
            XtPointer cb_struct_ptr) {

  int x = 400;
  int y = 200;

  Widget dialog,
      frame, add_button, delete_button,
      r_o_widget,list_w,r_o_button;

  XmString t;

  Arg args[10];

  int ac, item_count;
  char *prompt, buffer[INPUT_LINE_SIZE];

  /*****************************************************
  * build the formal description  property dialog box
  *****************************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 140); ac++;
  XtSetArg(args[ac], XmNwidth,  100); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Stream ID Dialog");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;

  dialog = XmCreateBulletinBoardDialog(w, "strreq", args, ac);

  XmStringFree(t);
```

```
prompt = strdup("Enter/Delete Stream IDs");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 45,
                XmNy, 15,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);


//** build a frame to put the list in ...
ac = 0;
XtSetArg(args[ac], XmNx,90);          ac++;
XtSetArg(args[ac], XmNy, 50);         ac++;
frame  = XmCreateFrame(dialog, "frame", args, ac);
XtManageChild(frame);

free(prompt);

// Load initial list from LList
if (op_being_updated._trigger_streams != NULL) {
 XmStringTable ListArray;

 int n = op_being_updated._trigger_streams->length();
 ListArray = (XmStringTable) XtMalloc (n * sizeof (XmString *));

 op_being_updated._trigger_streams->reset();
 for (int i = 0; i < n; i++) {
      ListArray[i] =
XmStringCreateSimple(op_being_updated._trigger_streams->item());
      op_being_updated._trigger_streams->step();
 }


 //* create the scrolled list of stream IDs //
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                XmNitems,          ListArray,
                XmNitemCount,      n,
                XmNvisibleItemCount,  5,
                NULL);
 XtManageChild(list_w);
}
else {
 ac = 0;
 XtSetArg(args[ac], XmNmarginWidth,10);          ac++;
 XtSetArg(args[ac], XmNmarginHeight, 10);        ac++;
 list_w = XmCreateScrolledList (frame, "list_w", NULL, 0);
 XtVaSetValues(list_w,
                XmNvisibleItemCount,  5,
                NULL);
 XtManageChild(list_w);
}

//* create the ADD button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
XtSetArg(args[ac], XmNy, 48); ac++;
add_button = XmCreatePushButton(dialog, "  Add  ", args, ac);
XtManageChild(add_button);

XtAddCallback(add_button , XmNactivateCallback,
tcstreams_reqid_add_cb,  (XtPointer) list_w);


//* create the DELETE button
ac = 0;
XtSetArg(args[ac], XmNx, 10); ac++;
```

```
  XtSetArg(args[ac], XmNy, 72); ac++;
  delete_button = XmCreatePushButton(dialog, " Delete ", args, ac);
  XtManageChild(delete_button);

  XtAddCallback(delete_button , XmNactivateCallback,
delete_reqid_cb, (XtPointer) list_w);

  //* create the return to operator dialog button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 105); ac++;
  r_o_button = XmCreatePushButton(dialog, " Return ", args, ac);
  XtManageChild(r_o_button);



/*********************************************************
  * now we need the call back send the button and method linked to the
button

*********************************************************/

  XtAddCallback(r_o_button , XmNactivateCallback, tcstreams_r_o_cb,
(XtPointer) list_w);


  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}
```



```
/*********************************************************
* get the operator formal_des from the More Button
*********************************************************/

void operator_formal_desc_cb(Widget w, XtPointer client_data,
            XtPointer cb_struct_ptr) {

  int x = 400;
  int y = 200;

  Widget dialog,
      text_w,rc,
      r_o_widget;

  XmString t,
       r_o_button;


  Arg args[10];

  int ac;

  /*****************************************************
  * build the formal description  property dialog box
  *****************************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 140); ac++;
  XtSetArg(args[ac], XmNwidth,  100); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Formal Description Dialog");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;
```

```
  dialog = XmCreateBulletinBoardDialog(w, "formal_desc", args, ac);

  XmStringFree(t);

  rc = XtVaCreateWidget("work_area", xmRowColumnWidgetClass,
dialog, NULL);

  //***** create scrolled text area
  XtSetArg(args[ac], XmNrows, 5);                ac++;
  XtSetArg(args[ac], XmNcolumns, 5);              ac++;
  XtSetArg(args[ac], XmNheight, 80);              ac++;
  XtSetArg(args[ac], XmNwidth, 250);              ac++;
  XtSetArg(args[ac], XmNeditable, True);          ac++;
  XtSetArg(args[ac], XmNeditMode, XmMULTI_LINE_EDIT);  ac++;
  XtSetArg(args[ac], XmNcursorPositionVisible, True);  ac++;
  text_w = XmCreateScrolledText(rc, "text_w", args, ac);

  XtManageChild(text_w);

  // * if not null set text to display in widget on screen
  if (op_being_updated.operator_formal_desc() != NULL)
    XtVaSetValues(text_w, XmNvalue,
op_being_updated.operator_formal_desc(), NULL);

  XtAddCallback(text_w, XmNmodifyVerifyCallback, formal_desc_cb,
(XtPointer) text_w);

  XtManageChild(rc);

  // create the OK button
  ac = 0;
  XtSetArg(args[ac], XmNx, 50); ac++;
  XtSetArg(args[ac], XmNy, 115); ac++;
  r_o_widget = XmCreatePushButton(dialog, "  Return To More Dialog
", args, ac);
  XtManageChild(r_o_widget);



/*********************************************************
  * now we need the call back methods..send  button and method to the
button

*********************************************************/

  XtAddCallback(r_o_widget,XmNactivateCallback, operator_r_o_cb,
(XtPointer) dialog);

  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}



/*********************************************************
* used for property menu Timing Info Dialog box Periodic time selected
button
*********************************************************/
void operator_pb_cb(Widget w, XtPointer client_data,
                    XtPointer cb_struct_ptr) {

  int x = 450;
```

```
int y = 200;
int initial_button = 1;

Widget dialog,
     ms_widget,
     required_by_widget,
     r_ti_widget,
     unit_box,
     met_value,
     period_value,
     finish_value,frame,
     frame1, frame2, frame3, frame4, frame5,
     ms_option_menu, req_option_menu, req_button;

XmString t,
       pb_button,
       sb_button,
       ntc_button,
       ms_line,
       sc_line,
       us_line,
       ms_title,
       mn_line,
       hr_line;


Arg args[10];
char *prompt, buffer[INPUT_LINE_SIZE];
int ac;


//**********Set Timing_Type to Periodic **************
op_being_updated.set_timing_type(PERIODIC);

/*************************************************
   * build the Periodic Timing Info  property dialog box
   *************************************************/
   ac = 0;
   XtSetArg(args[ac], XmNheight, 200);                ac++;
   XtSetArg(args[ac], XmNwidth, 600);                 ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False);        ac++;
   t = XmStringCreateSimple ("Periodic Timing Info Dialog");
   XtSetArg(args[ac], XmNdialogTitle, t);             ac++;

   dialog  = XmCreateBulletinBoardDialog(w, "Periodic_Timing", args,
ac);

   XmStringFree(t);

   /***************************************
   ** Met row
   ***************************************/
   prompt = strdup("Met");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 15,
                XmNy, 12,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

   met_value =  XtVaCreateManagedWidget("met_value",
                xmTextFieldWidgetClass, dialog,
                XmNx, 80,
                XmNy, 10,
                NULL);

   XtAddCallback(met_value, XmNactivateCallback, operator_met_cb,
(XtPointer) met_value);


   //** check the current met value
```

```
  if (op_being_updated.met() != UNDEFINED_TIME) {
   sprintf(buffer, "%d", op_being_updated.met());
   //** put the actual met value to the widget
   XtVaSetValues(met_value, XmNvalue, buffer, NULL);
  }

  free(prompt);

  //** build a frame to put the pulldown widget for ms in...
  ac = 0;
  XtSetArg(args[ac], XmNheight, 0);      ac++;
  XtSetArg(args[ac], XmNwidth,  0);      ac++;
  XtSetArg(args[ac], XmNmarginWidth,  0); ac++;
  XtSetArg(args[ac], XmNmarginHeight, 0); ac++;
  XtSetArg(args[ac], XmNx, 222);         ac++;
  XtSetArg(args[ac], XmNy, 10);          ac++;
  //   XtSetArg(args[ac],
XmNalignment,XmALIGNMENT_WIDGET_TOP); ac++;
  frame  = XmCreateFrame(dialog, "frame", args, ac);
  XtManageChild(frame);



  //*** define met ms pulldown widget
  ms_line  =  XmStringCreateSimple("ms");
  sc_line  =  XmStringCreateSimple("sec");
  us_line  =  XmStringCreateSimple("us");
  mn_line  =  XmStringCreateSimple("min");
  hr_line  =  XmStringCreateSimple("hr");


  ms_option_menu = XmVaCreateSimpleOptionMenu(frame,
"met_ms_option_menu",
       NULL, 'M', op_being_updated.met_unit() /* init menu selection
*/,
       set_met_unit_cb, /* call back method */
     XmVaPUSHBUTTON, ms_line,  'M', NULL, NULL,
       XmVaPUSHBUTTON, sc_line,  'S', NULL, NULL,
       XmVaPUSHBUTTON, us_line,  'U', NULL, NULL,
     XmVaPUSHBUTTON, mn_line,  'N', NULL, NULL,
       XmVaPUSHBUTTON, hr_line,  'H', NULL, NULL,
       NULL);

  XmStringFree(ms_line);
  XmStringFree(sc_line);
  XmStringFree(us_line);
  XmStringFree(mn_line);
  XmStringFree(hr_line);
  XtManageChild(ms_option_menu);



 //* create the required by  button
  ac = 0;
  XtSetArg(args[ac], XmNx, 310); ac++;
  XtSetArg(args[ac], XmNy, 15); ac++;
  req_button = XmCreatePushButton(dialog,  " Required By  ", args, ac);
  XtManageChild(req_button);

  XtAddCallback(req_button , XmNactivateCallback, ti_met_req_cb,
(XtPointer) dialog);



  /****************************************
  ** period row
  ****************************************/
  prompt = strdup("Period");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
            XmNx, 15,
```

```
                XmNy, 50,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

  period_value = XtVaCreateManagedWidget("period_value",
                xmTextFieldWidgetClass, dialog,
                XmNx, 80,
                XmNy, 45,
                NULL);
  XtAddCallback(period_value, XmNactivateCallback,
operator_period_cb, (XtPointer) period_value);

  //** put the period value to the screen and into the period_value
  if (op_being_updated.period() != UNDEFINED_TIME) {
    sprintf(buffer, "%d", op_being_updated.period());
    XtVaSetValues(period_value, XmNvalue, buffer, NULL);
  }
  free(prompt);

  //** build a frame to put the period ms pulldown widget for ms in...
  ac = 0;
  XtSetArg(args[ac], XmNheight, 0);      ac++;
  XtSetArg(args[ac], XmNwidth,  0);      ac++;
  XtSetArg(args[ac], XmNmarginWidth,  0); ac++;
  XtSetArg(args[ac], XmNmarginHeight, 0); ac++;
  XtSetArg(args[ac], XmNx, 222);         ac++;
  XtSetArg(args[ac], XmNy, 45);          ac++;
  frame2 = XmCreateFrame(dialog, "frame2", args, ac);
  XtManageChild(frame2);


  ms_line = XmStringCreateSimple("ms");
  sc_line = XmStringCreateSimple("sec");
  us_line = XmStringCreateSimple("us");
  mn_line = XmStringCreateSimple("min");
  hr_line = XmStringCreateSimple("hr");


  ms_option_menu = XmVaCreateSimpleOptionMenu(frame2,
"period_ms_option_menu",
      NULL, 'M', op_being_updated.period_unit() /* init menu
selection */,
          set_period_unit_cb, /* call back method */
      XmVaPUSHBUTTON, ms_line,  'M', NULL, NULL,
          XmVaPUSHBUTTON, sc_line,  'S', NULL, NULL,
          XmVaPUSHBUTTON, us_line,  'U', NULL, NULL,
      XmVaPUSHBUTTON, mn_line,  'N', NULL, NULL,
          XmVaPUSHBUTTON, hr_line,  'H', NULL, NULL,
          NULL);

  XmStringFree(ms_line);
  XmStringFree(sc_line);
  XmStringFree(us_line);
  XmStringFree(mn_line);
  XmStringFree(hr_line);
  XtManageChild(ms_option_menu);


  //* create the required by  button
  ac = 0;
  XtSetArg(args[ac], XmNx, 310); ac++;
  XtSetArg(args[ac], XmNy, 50); ac++;
  req_button = XmCreatePushButton(dialog,  " Required By  ", args, ac);
  XtManageChild(req_button);

  XtAddCallback(req_button , XmNactivateCallback, ti_period_req_cb,
(XtPointer) dialog);

  /*****************************************
  ** finish row
```

```
*****************************************/
prompt = strdup("Finish");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 15,
                XmNy, 80,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

prompt = strdup("Within");
XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 15,
                XmNy, 92,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

finish_value = XtVaCreateManagedWidget("finish_value",
                xmTextFieldWidgetClass, dialog,
                XmNx, 80,
                XmNy, 78,
                NULL);

 // Callback for finish withing value
 XtAddCallback(finish_value, XmNactivateCallback,
operator_finish_cb, (XtPointer)finish_value);

 //****** get the finish withing time and display to screen
 if (op_being_updated.fw() != UNDEFINED_TIME) {
   sprintf(buffer, "%d", op_being_updated.fw());
   XtVaSetValues(finish_value, XmNvalue, buffer, NULL);
 }

 free(prompt);


 //** build a frame to put the finish ms pulldown widget for ms in...
 ac = 0;
 XtSetArg(args[ac], XmNheight, 0);      ac++;
 XtSetArg(args[ac], XmNwidth, 0);       ac++;
 XtSetArg(args[ac], XmNmarginWidth, 0); ac++;
 XtSetArg(args[ac], XmNmarginHeight, 0); ac++;
 XtSetArg(args[ac], XmNx, 222);         ac++;
 XtSetArg(args[ac], XmNy, 78);          ac++;
 frame4 = XmCreateFrame(dialog, "frame4", args, ac);
 XtManageChild(frame4);


 ms_line = XmStringCreateSimple("ms");
 sc_line = XmStringCreateSimple("sec");
 us_line = XmStringCreateSimple("us");
 mn_line = XmStringCreateSimple("min");
 hr_line = XmStringCreateSimple("hr");


 ms_option_menu = XmVaCreateSimpleOptionMenu(frame4,
"finish_ms_option_menu",
      NULL, 'M', op_being_updated.fw_unit() /* init menu selection */,
          set_finish_unit_cb, /* call back method */
      XmVaPUSHBUTTON, ms_line, 'M', NULL, NULL,
          XmVaPUSHBUTTON, sc_line, 'S', NULL, NULL,
          XmVaPUSHBUTTON, us_line, 'U', NULL, NULL,
      XmVaPUSHBUTTON, mn_line, 'N', NULL, NULL,
          XmVaPUSHBUTTON, hr_line, 'H', NULL, NULL,
          NULL);

 XmStringFree(ms_line);
 XmStringFree(sc_line);
 XmStringFree(us_line);
 XmStringFree(mn_line);
 XmStringFree(hr_line);
 XtManageChild(ms_option_menu);
```

```
  //* create the required by  button
  ac = 0;
  XtSetArg(args[ac], XmNx, 310); ac++;
  XtSetArg(args[ac], XmNy, 85); ac++;
  req_button = XmCreatePushButton(dialog, " Required By ", args, ac);
  XtManageChild(req_button);

  XtAddCallback(req_button , XmNactivateCallback, ti_finish_req_cb,
(XtPointer) dialog);


  // create the OK button
  ac = 0;
  XtSetArg(args[ac], XmNx, 100); ac++;
  XtSetArg(args[ac], XmNy, 140); ac++;
  r_ti_widget = XmCreatePushButton(dialog, " Return To Timing Info
Dialog ", args, ac);
  XtManageChild(r_ti_widget);



/*********************************************************
  * now we need the call back methods..send  button and method to the
button

*********************************************************/

  XtAddCallback(r_ti_widget,XmNactivateCallback, operator_r_o_cb,
(XtPointer) dialog);


  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}

/*********************************************************
* used for property menu Timing Info Dialog box Sporadic time selected
button
*********************************************************/
void operator_sb_cb(Widget w, XtPointer client_data,
            XtPointer cb_struct_ptr) {


  int x = 450;
  int y = 200;
  int initial_button = 1;

  Widget dialog,
      ms_widget,
      required_by_widget,
      r_ti_widget,
      unit_box,
      met_value,
      mrt_value,
      mcp_value,
      frame,
      frame2, frame3, frame4, frame5,
      ms_option_menu,
      req_option_menu, req_button;

  XmString t,
       pb_button,
       sb_button,
```

```
        ntc_button,
        r_o_button,
        ms_line,
        sc_line,
        us_line,
        ms_title,
        mn_line,
        hr_line;


  Arg args[10];
  char *prompt, buffer[INPUT_LINE_SIZE];
  int ac;

  //**********Set Timing_Type to Sporadic **************
  op_being_updated.set_timing_type(SPORADIC);


  /*****************************************************
  * build the Sporadic Timing Info  property dialog box
  *****************************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 200); ac++;
  XtSetArg(args[ac], XmNwidth, 600); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Sporadic Timing Info Dialog");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;

  dialog = XmCreateBulletinBoardDialog(w, "Sporadic_Timing", args,
ac);

  XmStringFree(t);

  /****************************************
  *    Met row
  ****************************************/
  prompt = strdup("Met");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 15,
                XmNy, 12,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

  met_value =  XtVaCreateManagedWidget("met_value",
                xmTextFieldWidgetClass, dialog,
                XmNx,80,
                XmNy, 10,
                NULL);

  XtAddCallback(met_value, XmNactivateCallback, operator_met_cb,
(XtPointer)met_value);

  //** check the current met value
  if (op_being_updated.met() != UNDEFINED_TIME) {
   sprintf(buffer, "%d", op_being_updated.met());
   //** put the actual met value to the widget
   XtVaSetValues(met_value, XmNvalue, buffer, NULL);
  }

  free(prompt);

  //** build a frame to put the pulldown widget for ms in...
  ac = 0;
  XtSetArg(args[ac], XmNheight, 0);      ac++;
  XtSetArg(args[ac], XmNwidth,  0);      ac++;
  XtSetArg(args[ac], XmNmarginWidth,  0); ac++;
  XtSetArg(args[ac], XmNmarginHeight, 0); ac++;
  XtSetArg(args[ac], XmNx, 222);         ac++;
  XtSetArg(args[ac], XmNy, 10);          ac++;
```

```
  //   XtSetArg(args[ac],
XmNalignment,XmALIGNMENT_WIDGET_TOP); ac++;
  frame = XmCreateFrame(dialog, "frame", args, ac);
  XtManageChild(frame);

  //*** define met ms pulldown widget
  ms_line = XmStringCreateSimple("ms");
  sc_line = XmStringCreateSimple("sec");
  us_line = XmStringCreateSimple("us");
  mn_line = XmStringCreateSimple("min");
  hr_line = XmStringCreateSimple("hr");


  ms_option_menu = XmVaCreateSimpleOptionMenu(frame,
"met_ms_option_menu",
      NULL, 'M', op_being_updated.met_unit() /* init menu selection
*/,
          set_met_unit_cb, /* call back method */
      XmVaPUSHBUTTON, ms_line, 'M', NULL, NULL,
          XmVaPUSHBUTTON, sc_line, 'S', NULL, NULL,
          XmVaPUSHBUTTON, us_line, 'U', NULL, NULL,
      XmVaPUSHBUTTON, mn_line, 'N', NULL, NULL,
          XmVaPUSHBUTTON, hr_line, 'H', NULL, NULL,
          NULL);

  XmStringFree(ms_line);
  XmStringFree(sc_line);
  XmStringFree(us_line);
  XmStringFree(mn_line);
  XmStringFree(hr_line);
  XtManageChild(ms_option_menu);


  //* create the required by  button
  ac = 0;
  XtSetArg(args[ac], XmNx, 310); ac++;
  XtSetArg(args[ac], XmNy, 15); ac++;
  req_button = XmCreatePushButton(dialog,  " Required By  ", args, ac);
  XtManageChild(req_button);

  XtAddCallback(req_button , XmNactivateCallback, ti_met_req_cb,
(XtPointer) dialog);


  /****************************************
  ** MRT row
  ****************************************/
  prompt = strdup("MRT");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
              XmNx, 15,
              XmNy, 50,
              XmNalignment, XmALIGNMENT_BEGINNING,
              NULL);

   mrt_value = XtVaCreateManagedWidget("mrt_value",
              xmTextFieldWidgetClass, dialog,
              XmNx, 80,
              XmNy, 45,
              NULL);
   XtAddCallback(mrt_value, XmNactivateCallback, operator_mrt_cb,
(XtPointer)mrt_value);
  //** check the current mrt value
  if (op_being_updated.mrt() != UNDEFINED_TIME) {
   sprintf(buffer, "%d", op_being_updated.mrt());
   //** put the actual mrt value to the widget
   XtVaSetValues(mrt_value, XmNvalue, buffer, NULL);
  }

  free(prompt);
```

```
//** build a frame to put the mrt ms pulldown widget for ms in...
ac = 0;
XtSetArg(args[ac], XmNheight, 0);      ac++;
XtSetArg(args[ac], XmNwidth,  0);      ac++;
XtSetArg(args[ac], XmNmarginWidth,  0); ac++;
XtSetArg(args[ac], XmNmarginHeight, 0); ac++;
XtSetArg(args[ac], XmNx, 222);         ac++;
XtSetArg(args[ac], XmNy, 45);          ac++;
frame2  = XmCreateFrame(dialog, "frame2", args, ac);
XtManageChild(frame2);


ms_line  =  XmStringCreateSimple("ms");
sc_line  =  XmStringCreateSimple("sec");
us_line  =  XmStringCreateSimple("us");
mn_line  =  XmStringCreateSimple("min");
hr_line  =  XmStringCreateSimple("hr");


 ms_option_menu = XmVaCreateSimpleOptionMenu(frame2,
"mrt_ms_option_menu",
       NULL, 'M', op_being_updated.mrt_unit() /* init menu selection
*/,
           set_mrt_unit_cb, /* call back method */
       XmVaPUSHBUTTON, ms_line,  'M', NULL, NULL,
           XmVaPUSHBUTTON, sc_line,  'S', NULL, NULL,
           XmVaPUSHBUTTON, us_line,  'U', NULL, NULL,
       XmVaPUSHBUTTON, mn_line,  'N', NULL, NULL,
           XmVaPUSHBUTTON, hr_line,  'H', NULL, NULL,
           NULL);

 XmStringFree(ms_line);
 XmStringFree(sc_line);
 XmStringFree(us_line);
 XmStringFree(mn_line);
 XmStringFree(hr_line);
 XtManageChild(ms_option_menu);


 //* create the required by  button
 ac = 0;
 XtSetArg(args[ac], XmNx, 310); ac++;
 XtSetArg(args[ac], XmNy, 50); ac++;
 req_button = XmCreatePushButton(dialog,  " Required By  ", args, ac);
 XtManageChild(req_button);

 XtAddCallback(req_button , XmNactivateCallback, ti_mrt_req_cb,
(XtPointer) dialog);


 /****************************************
 * mcp row
 ****************************************/
 prompt = strdup("MCP");
 XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
             XmNx, 15,
             XmNy, 92,
             XmNalignment, XmALIGNMENT_BEGINNING,
             NULL);

 mcp_value  =  XtVaCreateManagedWidget("mcp_value",
             xmTextFieldWidgetClass, dialog,
             XmNx, 80,
             XmNy, 78,
             NULL);
 XtAddCallback(mcp_value, XmNactivateCallback, operator_mcp_cb,
(XtPointer)mcp_value);
 //** check the current mcp value
 if (op_being_updated.mcp() != UNDEFINED_TIME) {
  sprintf(buffer, "%d", op_being_updated.mcp());
```

```
  //** put the actual mcp value to the widget
  XtVaSetValues(mcp_value, XmNvalue, buffer, NULL);
 }

 free(prompt);

 //** build a frame to put the mcp ms pulldown widget for ms in...
 ac = 0;
 XtSetArg(args[ac], XmNheight, 0);      ac++;
 XtSetArg(args[ac], XmNwidth,  0);      ac++;
 XtSetArg(args[ac], XmNmarginWidth,  0); ac++;
 XtSetArg(args[ac], XmNmarginHeight, 0); ac++;
 XtSetArg(args[ac], XmNx, 222);         ac++;
 XtSetArg(args[ac], XmNy, 78);          ac++;
 frame4 = XmCreateFrame(dialog, "frame4", args, ac);
 XtManageChild(frame4);

 ms_line = XmStringCreateSimple("ms");
 sc_line = XmStringCreateSimple("sec");
 us_line = XmStringCreateSimple("us");
 mn_line = XmStringCreateSimple("min");
 hr_line = XmStringCreateSimple("hr");

 ms_option_menu = XmVaCreateSimpleOptionMenu(frame4,
"mcp_ms_option_menu",
      NULL, 'M', op_being_updated.mcp_unit() /* init menu selection
*/,
          set_mcp_unit_cb, /* call back method */
      XmVaPUSHBUTTON, ms_line,  'M', NULL, NULL,
          XmVaPUSHBUTTON, sc_line,  'S', NULL, NULL,
          XmVaPUSHBUTTON, us_line,  'U', NULL, NULL,
      XmVaPUSHBUTTON, mn_line,  'N', NULL, NULL,
          XmVaPUSHBUTTON, hr_line,  'H', NULL, NULL,
          NULL);

 XmStringFree(ms_line);
 XmStringFree(sc_line);
 XmStringFree(us_line);
 XmStringFree(mn_line);
 XmStringFree(hr_line);
 XtManageChild(ms_option_menu);



 //* create the required by  button
 ac = 0;
 XtSetArg(args[ac], XmNx, 310); ac++;
 XtSetArg(args[ac], XmNy, 85); ac++;
 req_button = XmCreatePushButton(dialog,  " Required By  ", args, ac);
 XtManageChild(req_button);

 XtAddCallback(req_button , XmNactivateCallback, ti_mcp_req_cb,
(XtPointer) dialog);


 // create the OK button
 ac = 0;
 XtSetArg(args[ac], XmNx, 100); ac++;
 XtSetArg(args[ac], XmNy, 140); ac++;
 r_ti_widget = XmCreatePushButton(dialog,  "  Return To Timing Info
Dialog  ", args, ac);
 XtManageChild(r_ti_widget);



/*********************************************************
 * now we need the call back methods..send  button and method to the
button

*********************************************************/
```

```
   XtAddCallback(r_ti_widget,XmNactivateCallback, operator_r_o_cb,
(XtPointer) dialog);


   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}

/*********************************************************
* used for property menu Timing Info Dialog box Non Time Critical
time button
*********************************************************/
void operator_ntc_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

   int x = 450;
   int y = 245;
   int initial_button = 1;

   Widget dialog,
        ms_widget,
        required_by_widget,
        r_ti_widget,
        unit_box;

   XmString t,
         pb_button,
         sb_button,
         ntc_button,
         r_o_button;


   Arg args[10];
   char *prompt, buffer[INPUT_LINE_SIZE];
   int ac;

   //**********Set Timing_Type to NTC **************
   op_being_updated.set_timing_type(NTC);

   /*********************************************************
   * build the Mcpic Timing Info  property dialog box
   *********************************************************/
   ac = 0;
   XtSetArg(args[ac], XmNheight, 80); ac++;
   XtSetArg(args[ac], XmNwidth,  80); ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
   t = XmStringCreateSimple ("NTC Timing Info Dialog");
   XtSetArg(args[ac], XmNdialogTitle, t); ac++;

   dialog = XmCreateBulletinBoardDialog(w, "NTC_Timing", args, ac);

   XmStringFree(t);

   prompt = strdup(" The Timing Sytle Has Been Set to Non Critical
Time");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 10,
                XmNy, 30,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

   free(prompt);
   // create the OK button
```

```
   ac = 0;
   XtSetArg(args[ac], XmNx, 55); ac++;
   XtSetArg(args[ac], XmNy, 70); ac++;
   r_ti_widget = XmCreatePushButton(dialog, "  Return To Timing Info
Dialog  ", args, ac);
   XtManageChild(r_ti_widget);



/*********************************************************
   * now we need the call back methods..send  button and method to the
button

*********************************************************/

   XtAddCallback(r_ti_widget,XmNactivateCallback, operator_r_o_cb,
(XtPointer) dialog);

   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}

/*********************************************************
* get operator timing info  selected button from the Operator Property
Dialog menu
*********************************************************/


void operator_ti_cb(Widget w, XtPointer client_data,
                    XtPointer cb_struct_ptr) {

int x = 500;
int y = 200;
int ac;

Widget dialog,
     radio_box,
     r_o_button,
     tbutton1,
     tbutton2,
     tbutton3,
     tbutton4,
     rowcol,
     frame,
     met_value,
     timing_type;

XmString t,
      tblabel1,
      tblabel2,
      tblabel3,
      tblabel4;

Arg args[10];
char *prompt, buffer[INPUT_LINE_SIZE];

/*****************************************
* build the Timing Info  property dialog box
*****************************************/

ac = 0;
XtSetArg(args[ac], XmNheight, 500);        ac++;
XtSetArg(args[ac], XmNwidth, 100);         ac++;
XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
```

```
t = XmStringCreateSimple ("Timing Info");
XtSetArg(args[ac], XmNdialogTitle, t);      ac++;

dialog = XmCreateBulletinBoardDialog(w, "Timing_Info", args, ac);

//** create a row col manager

ac = 0;
XtSetArg(args[ac], XmNpacking, XmPACK_COLUMN); ac++;
XtSetArg(args[ac], XmNspacing, 5);          ac++;
rowcol = XmCreateRadioBox(dialog,"rowcol",args, ac);
XtManageChild(rowcol);


//** create compound string for first toggle button label

  switch (op_being_updated.timing_type()) {
  case PERIODIC: {

        tblabel1 = XmStringCreateLtoR(" Periodic     <current setting>
", XmSTRING_DEFAULT_CHARSET);
        tblabel2 = XmStringCreateLtoR(" Sporadic ",
XmSTRING_DEFAULT_CHARSET);
            tblabel3 = XmStringCreateLtoR(" NTC ",
XmSTRING_DEFAULT_CHARSET);
            tblabel4 = XmStringCreateLtoR(" Return To Operator
Property Dialog ", XmSTRING_DEFAULT_CHARSET);
            break;
    }
    case SPORADIC: {

        tblabel1 = XmStringCreateLtoR(" Periodic  ",
XmSTRING_DEFAULT_CHARSET);
        tblabel2 = XmStringCreateLtoR(" Sporadic     <current setting>
", XmSTRING_DEFAULT_CHARSET);
            tblabel3 = XmStringCreateLtoR(" NTC ",
XmSTRING_DEFAULT_CHARSET);
            tblabel4 = XmStringCreateLtoR(" Return To Operator
Property Dialog ", XmSTRING_DEFAULT_CHARSET);
            break;
    }
    case NTC: {

        tblabel1 = XmStringCreateLtoR(" Periodic  ",
XmSTRING_DEFAULT_CHARSET);
        tblabel2 = XmStringCreateLtoR(" Sporadic ",
XmSTRING_DEFAULT_CHARSET);
            tblabel3 = XmStringCreateLtoR(" NTC      <current
setting> ", XmSTRING_DEFAULT_CHARSET);
            tblabel4 = XmStringCreateLtoR(" Return To Operator
Property Dialog ", XmSTRING_DEFAULT_CHARSET);
            break;
    }
    default: {

        tblabel1 = XmStringCreateLtoR(" Periodic ",
XmSTRING_DEFAULT_CHARSET);
        tblabel2 = XmStringCreateLtoR(" Sporadic ",
XmSTRING_DEFAULT_CHARSET);
            tblabel3 = XmStringCreateLtoR(" NTC ",
XmSTRING_DEFAULT_CHARSET);
            tblabel4 = XmStringCreateLtoR(" Return To Operator
Property Dialog ", XmSTRING_DEFAULT_CHARSET);
            break;
    }
  } // end of case


  //** create toggle button
  ac = 0;
```

```
  XtSetArg(args[ac], XmNlabelString, tblabel1);  ac++;
  XtSetArg(args[ac], XmNshadowThickness, 3);     ac++;
  tbutton1 = XmCreateToggleButton(rowcol, "tbutton1", args, ac);
  XtManageChild(tbutton1);
  XmStringFree(tblabel1);

  //** Add callback
  XtAddCallback(tbutton1, XmNarmCallback, operator_pb_cb, NULL);

  //** create toggle button
  ac = 0;
  XtSetArg(args[ac], XmNlabelString, tblabel2);          ac++;
  XtSetArg(args[ac], XmNshadowThickness, 3);             ac++;
  tbutton2 = XmCreateToggleButton(rowcol, "tbutton1", args, ac);
  XtManageChild(tbutton2);
  XmStringFree(tblabel2);

  //**Add callback
  XtAddCallback(tbutton2, XmNarmCallback, operator_sb_cb, NULL);

  //** create toggle button
  ac = 0;
  XtSetArg(args[ac], XmNlabelString, tblabel3); ac++;
  XtSetArg(args[ac], XmNshadowThickness, 3); ac++;
  tbutton3 = XmCreateToggleButton(rowcol, "tbutton3", args, ac);
  XtManageChild(tbutton3);
  XmStringFree(tblabel3);

  //** Add callback
  XtAddCallback(tbutton3, XmNarmCallback, operator_ntc_cb,
NULL);

  //** create toggle button
  ac = 0;
  XtSetArg(args[ac], XmNlabelString, tblabel4); ac++;
  XtSetArg(args[ac], XmNshadowThickness, 3); ac++;
  tbutton4 = XmCreateToggleButton(rowcol, "tbutton4", args, ac);
  XtManageChild(tbutton4);
  XmStringFree(tblabel4);

  //**Add callback
  XtAddCallback(tbutton4, XmNarmCallback, operator_r_o_cb,
(XtPointer) dialog);

  //** now manage dialog
  XtManageChild(dialog);

  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);

  XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}


/*********************************************************
* used for property menu Trigger Condition  Dialog box Radio Button
retrival
*********************************************************/
void triggering_radio_cb(Widget w, XtPointer which,
                XtPointer cbs)
{


  XmToggleButtonCallbackStruct *state =
    (XmToggleButtonCallbackStruct *) cbs;

  if (state->set) {
```

```
   op_being_updated.set_trigger_type((int)which); // put the new
trigger_type
  }

  else op_being_updated.default_trigger_type(); //default

}


/*********************************************************
* get operator triggering condition selected button from Operator
Property Dialog menu
*********************************************************/
void operator_tc_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

   int x = 400;
   int y = 200;
   int ac;
   int initial_button;
   int item_count;

   Widget dialog,
        radio_box,
        r_o_button,
        list_w,list_r,
        if_button,
        required_button, if_option_menu,
        req_option_menu,frame, frame1,frame2,
        add_button, delete_button,frameq;

   XmString t,
         byall_button,
         bysome_button,
         unprot_button, ms_line, hr_line, us_line, mn_line, sc_line;

   Arg args[10];
   char *prompt, buffer[INPUT_LINE_SIZE];

   /*************************************************
   * build the Triggering Condition property dialog box
   *************************************************/
   ac = 0;
   XtSetArg(args[ac], XmNheight, 300); ac++;
   XtSetArg(args[ac], XmNwidth, 100); ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
   t = XmStringCreateSimple ("Triggering Condition");
   XtSetArg(args[ac], XmNdialogTitle, t); ac++;

   dialog = XmCreateBulletinBoardDialog(w, "Triggering_Info", args,
ac);

   XmStringFree(t);

   prompt = strdup("Triggering Condition");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 9,
                XmNy, 5,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

   //* get the current operator value for trigger type
    initial_button = op_being_updated.trigger_type();

   //* create titles for radio buttons and XmStrings
   byall_button  = XmStringCreateSimple("By All");
   bysome_button  = XmStringCreateSimple("By Some");
   unprot_button  = XmStringCreateSimple("Unprotected");

   free(prompt);
```

```
  //* display radio box button layout
  radio_box = XmVaCreateSimpleRadioBox(dialog, "radio2_box",
                initial_button, triggering_radio_cb,
                XmVaRADIOBUTTON, byall_button, NULL, NULL,
NULL,
                XmVaRADIOBUTTON, bysome_button, NULL,
NULL, NULL,
                XmVaRADIOBUTTON, unprot_button, NULL,
NULL, NULL,
                XmNx, 30,
                XmNy, 25,
                XmNorientation, XmVERTICAL,
                NULL);

  //* set the XmString free...memory hog
  XmStringFree(byall_button);
  XmStringFree(bysome_button);
  XmStringFree(unprot_button);

  //* now display buttons
  XtManageChild(radio_box);

  prompt = strdup("Enter Mini-SDE Booleans");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 5,
                XmNy, 175,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);



 //* create the mini-sde If button
  ac = 0;
  XtSetArg(args[ac], XmNx, 45); ac++;
  XtSetArg(args[ac], XmNy, 192); ac++;
  if_button = XmCreatePushButton(dialog, " If ", args, ac);
  XtManageChild(if_button);

  XtAddCallback(if_button , XmNactivateCallback, tc_if_cb,
(XtPointer) dialog);

  free(prompt);

  prompt = strdup("Enter/Delete Stream IDs");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 160,
                XmNy, 5,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);
  //* create the requirements button
  ac = 0;
  XtSetArg(args[ac], XmNx, 190); ac++;
  XtSetArg(args[ac], XmNy, 160); ac++;
  delete_button = XmCreatePushButton(dialog, " Requirement IDs ",
args, ac);
  XtManageChild(delete_button);

  XtAddCallback(delete_button , XmNactivateCallback, tc_reqid_cb,
(XtPointer) dialog);


  prompt = strdup("Enter/Delete Requirement IDs");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 158,
                XmNy, 133,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

  //* create the stream id requirements  button
```

```
   ac = 0;
   XtSetArg(args[ac], XmNx, 190); ac++;
   XtSetArg(args[ac], XmNy, 45); ac++;
   add_button = XmCreatePushButton(dialog, "  Stream IDs  ", args,
ac);
   XtManageChild(add_button);

   XtAddCallback(add_button , XmNactivateCallback, tc_strmid_cb,
(XtPointer) dialog);




   //* create the return to operator dialog button
   ac = 0;
   XtSetArg(args[ac], XmNx, 140); ac++;
   XtSetArg(args[ac], XmNy, 250); ac++;
   r_o_button = XmCreatePushButton(dialog, "  Return ", args, ac);
   XtManageChild(r_o_button);



/*********************************************************
   * now we need the call back send the button and method linked to the
button

*********************************************************/

   XtAddCallback(r_o_button , XmNactivateCallback, operator_r_o_cb,
(XtPointer) dialog);

   XtManageChild(dialog);
```

```
   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(w, XmTRAVERSE_CURRENT);

}


/*********************************************************
* get operator Output Guard selected button
*********************************************************/

void operator_og_cb(Widget w, XtPointer client_data,
            XtPointer cb_struct_ptr) {

   Widget temp_w =  (Widget) client_data;
   char *text;

   // get the current output guard list  char*
   text =  op_being_updated.output_guard_list();

   // now call the mini_sde.C
   if (text == NULL)
    text = strdup("");

   EDIT_OUTPUT_GUARD(text);

   //** now set the pointer after we return
   op_being_updated.set_output_guard_list(text);

   free(text);

 }
```

```
/*********************************************************
* get operator exception selected button
*********************************************************/
void operator_e_cb(Widget w, XtPointer client_data,
                   XtPointer cb_struct_ptr) {

   Widget temp_w =  (Widget) client_data;
   char *text;

   // get the current exception list
   text =  op_being_updated.exception_list();


   // now call the mini_sde.C
   if (text == NULL)
     text = strdup("");

   EDIT_TIMER_OP(text);

   //** now set the pointer after we return
   op_being_updated.set_exception_list(text);

   free(text);

}

/*********************************************************
* get operator timer op  selected button
*********************************************************/
void operator_to_cb(Widget w, XtPointer client_data,
                   XtPointer cb_struct_ptr) {


   Widget temp_w =  (Widget) client_data;
   char *text;

   // get the current timer op list
   text =  op_being_updated.timer_op_list();

   // now call the mini_sde.C
   if (text == NULL)
     text = strdup("");

    EDIT_TIMER_OP(text);

   //** now set the pointer after we return
   op_being_updated.set_timer_op_list(text);

   free(text);

}


/*********************************************************
* get operator informal desc selected button
*********************************************************/
void operator_id_cb(Widget w, XtPointer client_data,
                   XtPointer cb_struct_ptr) {
   int x = 400;
   int y = 200;
   int ac;

   Widget dialog,
        text_w,rc,
        r_o_widget;

   XmString t,
         r_o_button;

   Arg args[10];
```

```
/*****************************************************
 * build the formal description  property dialog box
 *****************************************************/
 ac = 0;
 XtSetArg(args[ac], XmNheight, 140); ac++;
 XtSetArg(args[ac], XmNwidth,  100); ac++;
 XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
 t = XmStringCreateSimple ("Informal Description Dialog");
 XtSetArg(args[ac], XmNdialogTitle, t); ac++;

 dialog = XmCreateBulletinBoardDialog(w, "informal_desc", args, ac);

 XmStringFree(t);

 rc = XtVaCreateWidget("work_area", xmFormWidgetClass, dialog,
NULL);

 ac = 0;
 //***** create scrolled text area
 XtSetArg(args[ac], XmNrows, 5);                ac++;
 XtSetArg(args[ac], XmNcolumns, 5);             ac++;
 XtSetArg(args[ac], XmNheight, 80);             ac++;
 XtSetArg(args[ac], XmNwidth,  250);            ac++;
 XtSetArg(args[ac], XmNeditable, True);         ac++;
 XtSetArg(args[ac], XmNeditMode, XmMULTI_LINE_EDIT);  ac++;
 XtSetArg(args[ac], XmNcursorPositionVisible, True);  ac++;
 text_w = XmCreateScrolledText(rc, "text_w", args, ac);

 XtManageChild(text_w);


 // * if not null set text to display in widget on screen
 if (op_being_updated.operator_informal_desc() != NULL)
   XtVaSetValues(text_w, XmNvalue,
op_being_updated.operator_informal_desc(), NULL);

 XtAddCallback(text_w, XmNmodifyVerifyCallback,
informal_desc_cb, (XtPointer) text_w);

 XtManageChild(rc);

 // create the OK button
 ac = 0;
 XtSetArg(args[ac], XmNx, 50); ac++;
 XtSetArg(args[ac], XmNy, 100); ac++;
 r_o_widget = XmCreatePushButton(dialog, "  Return To Timing
Dialog  ", args, ac);
 XtManageChild(r_o_widget);



/*****************************************************
 * now we need the call back methods..send  button and method to the
button

*****************************************************/

 XtAddCallback(r_o_widget,XmNactivateCallback, operator_r_o_cb,
(XtPointer) dialog);

 XtManageChild(dialog);

 XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

 XtPopup(XtParent(dialog), XtGrabNone);

 XmProcessTraversal(w, XmTRAVERSE_CURRENT);
```

```
}


/*******************************************************
* get operator more selected button
*******************************************************/
void operator_m_cb(Widget w, XtPointer client_data,
                   XtPointer cb_struct_ptr) {

int x = 200;
   int y = 150;
   int ac;
   int initial_button;

   Widget dialog,
        radio_box,
        r_o_button,
        list_w,
        keyword_button,
        formal_des_button,
        text_w;

   XmString t,
          byall_button,
          bysome_button,
          unprot_button;

   Arg args[10];
   char *prompt, buffer[INPUT_LINE_SIZE];


   /*****************************************************
   * build the more property dialog box
   *****************************************************/
   ac = 0;
   XtSetArg(args[ac], XmNheight, 200); ac++;
   XtSetArg(args[ac], XmNwidth, 85); ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
   t = XmStringCreateSimple ("More Dialog");
   XtSetArg(args[ac], XmNdialogTitle, t); ac++;

   dialog = XmCreateBulletinBoardDialog(w, "more_info", args, ac);

   XmStringFree(t);

   prompt = strdup("Select Keywords or Formal Description");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                   XmNx, 35,
                   XmNy, 25,
                   XmNalignment, XmALIGNMENT_BEGINNING,
                   NULL);


   //* create the Keywords button
   ac = 0;
   XtSetArg(args[ac], XmNx, 19); ac++;
   XtSetArg(args[ac], XmNy, 100); ac++;
   keyword_button = XmCreatePushButton(dialog,          "      Keywords
", args, ac);
   XtManageChild(keyword_button);
   free(prompt);

  XtAddCallback(keyword_button , XmNactivateCallback,
operator_keyword_cb,  (XtPointer) dialog);



   //* create the formal description button
```

```
  ac = 0;
  XtSetArg(args[ac], XmNx, 160); ac++;
  XtSetArg(args[ac], XmNy, 100); ac++;
  formal_des_button = XmCreatePushButton(dialog,     "   Formal
Description ", args, ac);
  XtManageChild(formal_des_button);

 XtAddCallback(formal_des_button , XmNactivateCallback,
operator_formal_desc_cb,  (XtPointer) dialog);

  //* create the return to operator dialog button
  ac = 0;
  XtSetArg(args[ac], XmNx, 31); ac++;
  XtSetArg(args[ac], XmNy, 156); ac++;
  r_o_button = XmCreatePushButton(dialog,  "  Return To Operator
Property Dialog ", args, ac);
  XtManageChild(r_o_button);


/*********************************************************
  * now we need the call back method.. button and method linked to the
button

*********************************************************/

  XtAddCallback(r_o_button , XmNactivateCallback, operator_r_o_cb,
(XtPointer) dialog);

  XtManageChild(dialog);


  XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

  XtPopup(XtParent(dialog), XtGrabNone);
  XmProcessTraversal(w, XmTRAVERSE_CURRENT);


}



/*********************************************************
* get operator return to graph editor selected button
*********************************************************/
void operator_r_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr) {

 op_being_updated.copy_back((OperatorObject*) selected_object_ptr);
//@2
 graphic_list.draw();
 XtUnmanageChild((Widget)client_data);
}


/*********************************************************
* main driver for this package that displays the operator widget and gets
the
* needed inputs
*********************************************************/

void operator_property_dialog(Widget parent_widget, int x, int y) {

  Widget dialog,
       ti_button,
       tc_button,
       og_button,
       e_button,
       to_button,
       id_button,
```

```
    m_button,
    r_button,
    operator_name;

  XmString t;

  Arg args[10];
  char *prompt, buffer[INPUT_LINE_SIZE];
  int ac;

  /*************************************
  * build the property dialog box
  *************************************/
  ac = 0;
  XtSetArg(args[ac], XmNheight, 310); ac++;
  XtSetArg(args[ac], XmNwidth, 120); ac++;
  XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
  t = XmStringCreateSimple ("Operator Property");
  XtSetArg(args[ac], XmNdialogTitle, t); ac++;

  dialog = XmCreateBulletinBoardDialog(parent_widget,
"Operator_Property", args, ac);

  // dialog = XtVaCreateManagedWidget(
  XmStringFree(t);

  prompt = strdup("Operator Name:");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 43,
                XmNy, 10,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);

  operator_name = XtVaCreateManagedWidget("operator_name",
                xmTextFieldWidgetClass, dialog,
                XmNx, 22,
                XmNy, 27,
                NULL);
  XtAddCallback(operator_name, XmNactivateCallback,
operator_name_cb, (XtPointer)operator_name);
  if (op_being_updated.name() != NULL)
    XtVaSetValues(operator_name, XmNvalue,
op_being_updated.name(), NULL);
  free(prompt);

  // create the Timing Info  button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 60); ac++;
  ti_button = XmCreatePushButton(dialog, "     Timing Info       ", args,
ac);
  XtManageChild(ti_button);

  // create the triggering condition  button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 90); ac++;
  tc_button = XmCreatePushButton(dialog, "  Triggering Condition  ",
args, ac);
  XtManageChild(tc_button);

  // create output guard button
  ac = 0;
  XtSetArg(args[ac], XmNx, 10); ac++;
  XtSetArg(args[ac], XmNy, 120); ac++;
  og_button = XmCreatePushButton(dialog, "     Output Guard     ",
args, ac);
  XtManageChild(og_button);

  // create the Exception button
```

```
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 150); ac++;
   e_button = XmCreatePushButton(dialog, "      Exception      ", args,
ac);
   XtManageChild(e_button);

   // create the Timer Op  button
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 180); ac++;
   to_button = XmCreatePushButton(dialog, "      Timer Op      ", args,
ac);
   XtManageChild(to_button);

   // create the Informal Desc  button
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 210); ac++;
   id_button = XmCreatePushButton(dialog, "    Informal Desc    ",
args, ac);
   XtManageChild(id_button);

   // create the More  button
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 240); ac++;
   m_button = XmCreatePushButton(dialog, "       More       ", args,
ac);
   XtManageChild(m_button);


   // create the OK button
   ac = 0;
   XtSetArg(args[ac], XmNx, 10); ac++;
   XtSetArg(args[ac], XmNy, 268); ac++;
   r_button = XmCreatePushButton(dialog, "  Return To Graph Editor ",
args, ac);
   XtManageChild(r_button);



/*********************************************************
   * now we need the call back methods..send the button and method
linked to the button
   *********************************************************/

   XtAddCallback(ti_button, XmNactivateCallback, operator_ti_cb,
(XtPointer) dialog);
   XtAddCallback(tc_button, XmNactivateCallback, operator_tc_cb,
(XtPointer) dialog);
   XtAddCallback(og_button, XmNactivateCallback, operator_og_cb,
(XtPointer) dialog);
   XtAddCallback(e_button,  XmNactivateCallback, operator_e_cb,
(XtPointer) dialog);
   XtAddCallback(to_button, XmNactivateCallback, operator_to_cb,
(XtPointer) dialog);
   XtAddCallback(id_button, XmNactivateCallback, operator_id_cb,
(XtPointer) dialog);
   XtAddCallback(m_button,  XmNactivateCallback, operator_m_cb,
(XtPointer) dialog);
   XtAddCallback(r_button,  XmNactivateCallback, operator_r_cb,
(XtPointer) dialog);

   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);
```

```
  XmProcessTraversal(parent_widget, XmTRAVERSE_CURRENT);
}


/*********************************************************
*                end of operator_property_menu.C
*********************************************************/
```

```
/* ********************************************************

Name:        spline_object.h
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 11 Sep 92
Remarks:  Specification for the SplineObject class.

      The SplineObject is used to create the curved lines
     used in the graph_editor's splines.  It stores a
     linked list of control points which it uses to draw
     itself, point by point.

      In order to correctly terminate in the appropriate
     locations, it adds special control points I call
     'shadow points' at the ends of the lines.  Since
     b-splines don't normally end at the first and last
     control points, the extra shadow points make the
     curve stop in the right place.

**********************************************************
Modification History:

 Baseline taken from Robert M. Dixon
 Changed include file to a .H from .h for C++.

 @1   KBM    15 Aug 96
        Changes made for infinate loop investigation...to be removed
        later.  Removed the delete operator from the destructor and
        added a cout comment.  Local versions of new and delete added
        for investigation.

 @2   KBM    15 Aug 96
     Renamed write_to_disk to to_psdl and build_from_disk to
        from_psdl to be more reflective of new calling mechanism.

 @3   KBM    20 Aug 96
     Added over_handle() to detect being over a handle without
     selecting the handle.

**********************************************************
*/

#ifndef spline_object_h
#define spline_object_h 1

#include <stdio.h>
#include <X11/Xlib.h>
#include <X11/Intrinsic.h>
#include <Xm/MessageB.h>
#include "ge_defs.h"
#include "ge_interface.h"

class OperatorObject;
class StreamObject;

// S. Decato 8/1/96
// added the following class forward declaration
// to avoid complaints from the friend definition
class SplineObject;

class _spline_node {
  friend SplineObject;
protected:
  int _x, _y;
  _spline_node *_next_ptr;
public:
  _spline_node() {_next_ptr = NULL;}
  _spline_node(int x, int y) {_x = x; _y = y; _next_ptr = NULL;}
```

```
 _spline_node(XYPAIR inpair){_x = inpair.x; _y = inpair.y; _next_ptr =
NULL;}
 ~_spline_node() {
#ifdef GE_DEBUG
            cout << "delete spline_node\n";
#endif
          }

#ifdef GE_DEBUG
 void* operator new(size_t size) {                    //@1
  cout << "spline_node new" << endl;                     //@1
  return ::operator new(size);                      //@1
 }                                                  //@1
 void  operator delete(void* s) {                     //@1
  cout << "spline_node delete" << endl;                  //@1
  ::operator delete(s);                             //@1
 }                                                  //@1
#endif

};

class SplineObject {
protected:
 static Widget _error_tgt;

 _spline_node *_head_ptr;
 BOOLEAN _shadow_pts_set, _handles_drawn;
 _spline_node *_iter;
 int _handle_selected;

 static valid_num_string(char *num);

 void draw_arrowhead(GC graphics_context, StreamObject *parent,
             _spline_node *endpoint);
 static void error_box(char *error_message);
 int num_points(_spline_node *p1, _spline_node *p2,
         _spline_node *p3, _spline_node *p4);
public:
 static void set_error_tgt(Widget widget) {_error_tgt = widget;}

 SplineObject();
 ~SplineObject() {
#ifdef GE_DEBUG
            cout << "delete _head" << endl;
#endif
          } //delete _head_ptr;} //@1

#ifdef GE_DEBUG
 void* operator new(size_t size) {                    //@1
  cout << "spline new" << endl;                      //@1
  return ::operator new(size);                      //@1
 }                                                  //@1
 void  operator delete(void* s) {                     //@1
  cout << "spline delete" << endl;                   //@1
  ::operator delete(s);                             //@1
 }                                                  //@1
#endif

 GE_STATUS from_psdl(FILE *infile);                          //@2
 GE_STATUS to_psdl(FILE *outfile, EXTERN_STATUS status);
//@2
 void copy(SPLINE_PTR ptr);
 void copy_back(SPLINE_PTR &ptr, EXTERN_STATUS status);
 void clearList();
 CLASS_DEF is_a() {return SPLINEOBJECT;}
 SplineObject& operator=(SplineObject&);
 void set_object_ptrs(OperatorObject *_from_ptr,
             OperatorObject *_to_ptr);
 void draw(StreamObject* parent, GC graphics_context,
       GC handle_context, DRAW_STYLE style,
```

```
        EXTERN_STATUS status);
 XYPAIR first_point();
 XYPAIR last_point();
 void draw_handles(GC draw_context, StreamObject *parent,
             int x1, int y1);
 void clear();
 void add(int x, int y);
 void reset_iter();
 XYPAIR next_pair();
 BOOLEAN hit(int x, int y);
 void set_text_location(int &name_x, int &name_y,
                 int &latency_x, int &latency_y);
 void set_name_location(int &name_x, int &name_y);
 void set_latency_location(int name_x, int name_y,
                  int &latency_x, int &latency_y);
 void set_extern_location(XYPAIR &extern_location,
                 EXTERN_STATUS status);
 BOOLEAN hit_handle(int x, int y, EXTERN_STATUS status);
 BOOLEAN over_handle(int x, int y, EXTERN_STATUS status);
//@3
 void move_handle(int x, int y);
 BOOLEAN empty() {return _head_ptr == NULL;}
 void erase_handle(GC graphics_context, StreamObject *parent);
 void reset_handles_drawn() {_handles_drawn = FALSE;}
};

#endif
```

```
/* ********************************************************

Name:        spline_object.C
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 17 Sep 92
Remarks:  Implementation of the SplineObject class.

      The SplineObject is used to create the curved lines
     used in the graph_editor's splines.  It stores a
     linked list of control points which it uses to draw
     itself, point by point.

      In order to correctly terminate in the appropriate
     locations, it adds special control points I call
     'shadow points' at the ends of the lines.  Since
     b-splines don't normally end at the first and last
     control points, the extra shadow points make the
     curve stop in the right place.


Credits:  Portions of code are adapted from the following:

      Barakati, Naba, X Window System Programming, SAMS,
        1991.

      Heller, Dan, Motif Programming Manual, O'Reilly and
        Associates, 1991.

      Johnson, Eric, and Reichard, Kevin, X Window
        Applications Programming, MIS Press, 1989.

      Young, Douglas, Object Oriented Programming With C++
        and OSF/Motif, Prentice-Hall, 1992.

      Formula for b-spline curves comes from:

      Zyda, Michael, Book Number 5, Graphics and Video
        Laboratory, Naval Postgraduate School, 1990.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
     Changed include file to a .H from .h for C++.

 @2   KBM    15 Aug 96
     Changed method names: write_to_disk   -> to_psdl
                           build_from_disk -> from_psdl

 @3   KBM    15 Aug 96
     Added DEBUG statements for troubleshooting, use GE_DEBUG
flag

     JWH    19 Aug 96
     Entered into RCS

 @4   KBM    20 Aug 96
     Added over_handle() to detect being over a handle without
     selecting the handle.

********************************************************
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stream.h>
```

```
#include "ge_defs.h"
#include "spline_object.H"                    //@1
#include "stream_object.H"                    //@1
#include "operator_object.H"                  //@1

#define ARROWANGLE 45.0
#define ARROWSIDELENGTH 10.0
#define HITDISTANCE 10
#define SKIPFACTOR 4
#define NONE 0
#define EXTERN_OFFSET 30

// Initializes the static error target widget.

Widget SplineObject::_error_tgt = NULL;

// Determines whether the string represents a valid number.

BOOLEAN SplineObject::valid_num_string(char *num) {
 int index, num_length;

 if(num != NULL) {
  num_length = strlen(num);
  if(num_length > 0) {
   for(index = 0; index < num_length; index++) {
    if((num[index] != '\n') &&
     ((num[index] < '0') || (num[index] > '9')))
      return FALSE;
   }
   return TRUE;
  }
 }
 return FALSE;
}
```



```
// Displays the error message in a Motif error dialog box.

void SplineObject::error_box(char *error_message) {
 static Widget error_dialog = NULL;
 Arg arg[1];
 XmString t;

 if(_error_tgt != NULL) {
  if(error_dialog == NULL) {
   error_dialog = XmCreateMessageDialog(_error_tgt, "error",
                                        arg, 0);
   XtVaSetValues(XtParent(error_dialog),
            XtNtitle, "Error",
            NULL);
   XtUnmanageChild(XmMessageBoxGetChild(error_dialog,
                                XmDIALOG_HELP_BUTTON));
  }
  t = XmStringCreateSimple(error_message);
  XtVaSetValues(error_dialog,
           XmNmessageString, t,
           NULL);
  XmStringFree(t);
  XtManageChild(error_dialog);
 }
 else
  cout <<
   "Error Target Widget must be set by calling module.\n"
    << endl;
}

SplineObject::SplineObject() {

 _head_ptr = NULL;
 _iter = NULL;
 _shadow_pts_set = FALSE;
```

```
 _handle_selected = NONE;
 _handles_drawn = FALSE;
}

// Draws a handle at the given coordinates.

void SplineObject::draw_handles(GC draw_context,
                    StreamObject *parent, int x1,
                    int y1) {
#ifdef GEDEBUG
     cout << "Spline: " << (x1 - (HANDLESIZE / 2)) << " " <<
      (y1 - (HANDLESIZE / 2)) << " " << HANDLESIZE << endl;
#endif
     XSetFunction(parent->display_ptr(), draw_context, GXxor);
     XFillRectangle(parent->display_ptr(),
               parent->draw_window(),
               draw_context, (x1 - (HANDLESIZE / 2)),
               (y1 - (HANDLESIZE / 2)), HANDLESIZE,
               HANDLESIZE);
     XFillRectangle(parent->display_ptr(),
               *(parent->drawing_area_pixmap()),
               draw_context, (x1 - (HANDLESIZE / 2)),
               (y1 - (HANDLESIZE / 2)), HANDLESIZE,
               HANDLESIZE);
     XSetFunction(parent->display_ptr(), draw_context, GXcopy);
}

// Builds the spline from disk.

GE_STATUS SplineObject::from_psdl(FILE *infile) {          //@2
 char buffer[INPUT_LINE_SIZE + 1], *error_str_ptr = NULL;
 _spline_node *temp_node_ptr;
 int x, y;
 BOOLEAN done = FALSE;
 GE_STATUS status = SUCCEEDED;

fgets(buffer, INPUT_LINE_SIZE, infile); // clear "SPLINE"
while((!done) && (status == SUCCEEDED)) {
 if(strcmp("SPLINEEND\n",
       fgets(buffer, INPUT_LINE_SIZE, infile)) != 0) {
  if(valid_num_string(buffer))
   x = atoi(buffer);
  else
   error_str_ptr = strdup("Corrupted spline x");

  if(error_str_ptr == NULL) {
   fgets(buffer, INPUT_LINE_SIZE, infile);
   if(valid_num_string(buffer))
    y = atoi(buffer);
   else
    error_str_ptr = strdup("Corrupted spline y");
  }

  if(error_str_ptr == NULL) {
   if(_head_ptr == NULL) {
    _head_ptr = new _spline_node(x, y);
    temp_node_ptr = _head_ptr;
   }
   else {
    temp_node_ptr->_next_ptr = new _spline_node(x, y);
    temp_node_ptr = temp_node_ptr->_next_ptr;
   }
  }
 }
 else
  done = TRUE;
 if(error_str_ptr != NULL) {
  error_box(error_str_ptr);
  status = FAILED;
  free(error_str_ptr);
```

```
  }
 }
 if(ferror(infile)) {
  error_box("Unix reports an error building spline.");
  clearerr(infile);
  status = FAILED;
 }
 return status;
}

//  Writes the graphic attributes of the spline to disk.

GE_STATUS SplineObject::to_psdl(FILE *outfile,                    //@2
                                EXTERN_STATUS status) {
 char buffer[INPUT_LINE_SIZE + 1];
 _spline_node *temp_node_ptr = _head_ptr;

 if(status == FROM_EXTERNAL)
  temp_node_ptr = temp_node_ptr->_next_ptr;
 else
  temp_node_ptr = temp_node_ptr->_next_ptr->_next_ptr;
 fprintf(outfile, "SPLINE\n");
 while(temp_node_ptr->_next_ptr->_next_ptr != NULL) {
  sprintf(buffer, "%d\n", temp_node_ptr->_x);
  fprintf(outfile, buffer);
  sprintf(buffer, "%d\n", temp_node_ptr->_y);
  fprintf(outfile, buffer);
  temp_node_ptr = temp_node_ptr->_next_ptr;
 }
 if(status == TO_EXTERNAL) {
  sprintf(buffer, "%d\n", temp_node_ptr->_x);
  fprintf(outfile, buffer);
  sprintf(buffer, "%d\n", temp_node_ptr->_y);
  fprintf(outfile, buffer);
 }
 fprintf(outfile, "SPLINEEND\n");
 if(ferror(outfile)) {
  clearerr(outfile);
  return FAILED;
 }
 else
  return SUCCEEDED;
}

//  Assignment operator.

SplineObject& SplineObject::operator=(SplineObject& spline) {
 _spline_node *source_temp_ptr, *target_temp_ptr;

#ifdef GE_DEBUG
 cout << "operator =" << endl;                                   //@3
#endif

 if(spline._head_ptr != NULL) {
  _head_ptr = new _spline_node;
  _head_ptr->_x = spline._head_ptr->_x;
  _head_ptr->_y = spline._head_ptr->_y;
  source_temp_ptr = spline._head_ptr->_next_ptr;
  target_temp_ptr = _head_ptr;
  while(source_temp_ptr != NULL) {
   target_temp_ptr->_next_ptr = new _spline_node;
   target_temp_ptr = target_temp_ptr->_next_ptr;
   target_temp_ptr->_x = source_temp_ptr->_x;
   target_temp_ptr->_y = source_temp_ptr->_y;
   source_temp_ptr = source_temp_ptr->_next_ptr;
  }
 }
 return *this;
}
```

```
// Returns the coordinates of the first point in the spline.

XYPAIR SplineObject::first_point() {
 XYPAIR temp_pair;

 if(_head_ptr != NULL) {
  temp_pair.x = _head_ptr->_x;
  temp_pair.y = _head_ptr->_y;
 }
 else {
  temp_pair.x = 0;
  temp_pair.y = 0;
 }
 return temp_pair;
}

// Returns the coordinates of the last point in the spline.

XYPAIR SplineObject::last_point() {
 XYPAIR temp_pair;
 _spline_node *temp_node_ptr = _head_ptr;

 if(_head_ptr != NULL) {
  while(temp_node_ptr->_next_ptr != NULL)
   temp_node_ptr = temp_node_ptr->_next_ptr;
  temp_pair.x = temp_node_ptr->_x;
  temp_pair.y = temp_node_ptr->_y;
 }
 else {
  temp_pair.x = 0;
  temp_pair.y = 0;
 }
 return temp_pair;
}
```

```
// Sets _from_ptr and _to_ptr to the operators at either
// end of the spline. Pointers for external streams equal
// NULL. Also adds the necessary shadow and intercept points,
// and adds a control point for streams defined without one.
// Splines with externals must always have at least one point.

// Streams with external endpoints use the last control
// point as the physical endpoint of the spline. Otherwise,
// the attached operator provides the coordinates for the
// intercept point, i.e. the spline endpoint.

// In a spline with shadow points, the points are stored
// in the linked list in the following order:
//
//      'to' shadow point - 'to' intercept point -
//            (defined control points) -
//      'from' intercept point - 'from' shadow point

void SplineObject::set_object_ptrs(OperatorObject *_from_ptr,
                                   OperatorObject *_to_ptr) {
 XYPAIR from_intercept, to_intercept, temp_pair, first_point,
  last_point;
 _spline_node *temp_node_ptr, *temp_head_ptr, *temp_node_ptr2;

#ifdef GE_DEBUG
 cout << "set_object_ptrs" << endl;                    //@3
#endif

 if(_shadow_pts_set == TRUE) {

// strips off existing intercept and shadow points. For
// externals, the intercept points are saved.

  temp_node_ptr = _head_ptr;
  while(temp_node_ptr->_next_ptr->_next_ptr->_next_ptr != NULL)
```

```
  temp_node_ptr = temp_node_ptr->_next_ptr;
 if(_to_ptr == NULL) {
  temp_pair.x = temp_node_ptr->_next_ptr->_x;
  temp_pair.y = temp_node_ptr->_next_ptr->_y;
  to_intercept = temp_pair;
 }
 delete temp_node_ptr->_next_ptr;
 temp_node_ptr->_next_ptr = NULL;
 temp_node_ptr2 = _head_ptr;
 temp_node_ptr = _head_ptr->_next_ptr->_next_ptr;
 temp_node_ptr2->_next_ptr->_next_ptr = NULL;
 if(_from_ptr == NULL) {
  temp_pair.x = _head_ptr->_next_ptr->_x;
  temp_pair.y = _head_ptr->_next_ptr->_y;
  from_intercept = temp_pair;
 }
 delete temp_node_ptr2;
 _head_ptr = temp_node_ptr;
}
else {
 _shadow_pts_set = TRUE;
 if(_from_ptr == NULL) {
  if(_head_ptr == NULL) {
   error_box("External streams must have at least one control point");
   return;
  }
  else {
   temp_node_ptr = _head_ptr;
   _head_ptr = _head_ptr->_next_ptr;
   from_intercept.x = temp_node_ptr->_x;
   from_intercept.y = temp_node_ptr->_y;
   temp_node_ptr->_next_ptr = NULL;
   delete temp_node_ptr;
  }
 }
 if(_to_ptr == NULL) {
  if(_head_ptr == NULL) {
   error_box("External streams must have at least one control point");
        return;
  }
  else {
   temp_node_ptr = _head_ptr;
   if(temp_node_ptr->_next_ptr == NULL) {
    to_intercept.x = temp_node_ptr->_x;
    to_intercept.y = temp_node_ptr->_y;
    delete temp_node_ptr;
    _head_ptr = NULL;
   }
   else {
    while(temp_node_ptr->_next_ptr->_next_ptr != NULL)
     temp_node_ptr = temp_node_ptr->_next_ptr;
    to_intercept.x = temp_node_ptr->_next_ptr->_x;
    to_intercept.y = temp_node_ptr->_next_ptr->_y;
    delete temp_node_ptr->_next_ptr;
    temp_node_ptr->_next_ptr = NULL;
   }
  }
 }
}
if(_head_ptr != NULL) {
 temp_pair = this->first_point();
 if(_from_ptr != NULL)
  from_intercept = _from_ptr->intercept(temp_pair.x,
                                        temp_pair.y);
 first_point.x = from_intercept.x -
  (temp_pair.x - from_intercept.x);
 first_point.y = from_intercept.y -
  (temp_pair.y - from_intercept.y);
 temp_pair = this->last_point();
 if(_to_ptr != NULL)
```

```
  to_intercept = _to_ptr->intercept(temp_pair.x,
                                    temp_pair.y);
 last_point.x = to_intercept.x - (temp_pair.x -
                                  to_intercept.x);
 last_point.y = to_intercept.y - (temp_pair.y -
                                  to_intercept.y);
 temp_head_ptr = new _spline_node(first_point);
 temp_head_ptr->_next_ptr = new _spline_node(from_intercept);
 temp_node_ptr = temp_head_ptr->_next_ptr;
 temp_node_ptr->_next_ptr = _head_ptr;
 while(temp_node_ptr->_next_ptr != NULL)
  temp_node_ptr = temp_node_ptr->_next_ptr;
 temp_node_ptr->_next_ptr = new _spline_node(to_intercept);
 temp_node_ptr->_next_ptr->_next_ptr =
  new _spline_node(last_point);
 _head_ptr = temp_head_ptr;
}
else {
 if(_to_ptr == NULL)
  from_intercept = _from_ptr->intercept(to_intercept.x,
                                        to_intercept.y);
 else
  if(_from_ptr == NULL)
   to_intercept =
    _to_ptr->intercept(from_intercept.x,
                       from_intercept.y);
  else {
   temp_pair = _to_ptr->center();
   from_intercept = _from_ptr->intercept(temp_pair.x,
                                         temp_pair.y);
   to_intercept = _to_ptr->intercept(from_intercept.x,
                                     from_intercept.y);
  }
 temp_pair.x = (from_intercept.x + to_intercept.x) / 2;
 temp_pair.y = (from_intercept.y + to_intercept.y) / 2;
 first_point.x = from_intercept.x - (temp_pair.x -
                                     from_intercept.x);
 first_point.y = from_intercept.y - (temp_pair.y -
                                     from_intercept.y);
 last_point.x = to_intercept.x - (temp_pair.x -
                                  to_intercept.x);
 last_point.y = to_intercept.y - (temp_pair.y -
                                  to_intercept.y);
 temp_head_ptr = new _spline_node(first_point);
 temp_head_ptr->_next_ptr = new _spline_node(from_intercept);
 temp_node_ptr = temp_head_ptr->_next_ptr;
 temp_node_ptr->_next_ptr = new _spline_node(temp_pair);
 temp_node_ptr = temp_node_ptr->_next_ptr;
 temp_node_ptr->_next_ptr = new _spline_node(to_intercept);
 temp_node_ptr->_next_ptr->_next_ptr =
  new _spline_node(last_point);
 _head_ptr = temp_head_ptr;
 }
}

// Determines the number of points to be calculated for the
// spline.

int SplineObject::num_points(_spline_node *p1, _spline_node *p2,
                             _spline_node *p3,
                             _spline_node *p4) {
 int distance, number_points = 0;

 if(p4 != NULL) {
  distance = abs(p1->_x - p2->_x);
  if(distance > number_points)
   number_points = distance;
  distance = abs(p1->_x - p3->_x);
  if(distance > number_points)
   number_points = distance;
```

```
  distance = abs(p1->_x - p4->_x);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p2->_x - p3->_x);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p2->_x - p4->_x);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p3->_x - p4->_x);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p1->_y - p2->_y);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p1->_y - p3->_y);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p1->_y - p4->_y);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p2->_y - p3->_y);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p2->_y - p4->_y);
  if(distance > number_points)
    number_points = distance;
  distance = abs(p3->_y - p4->_y);
  if(distance > number_points)
    number_points = distance;
 }
 return number_points;
}

// Draws the arrowheads for the line.
```

```
void SplineObject::draw_arrowhead(GC graphics_context,
                    StreamObject *parent,
                    _spline_node *endpoint) {
 _spline_node *last_point = endpoint->_next_ptr;
 double angle, temp_angle;
 double half_arrow_angle = ARROWANGLE / 2.0 * M_PI / 180.0;
 XPoint vertices[3];

 if(last_point->_x == endpoint->_x) {
  if(last_point->_y > endpoint->_y)
    angle = M_PI / 2.0;
  else
    angle = 3.0 * M_PI / 2.0;
 }
 else {
  angle = atan((double) (last_point->_y - endpoint->_y) /
          (double) (last_point->_x - endpoint->_x));
  if((last_point->_x < endpoint->_x))
    angle = M_PI + angle;
 }
 vertices[0].x = endpoint->_x;
 vertices[0].y = endpoint->_y;
 temp_angle = angle - half_arrow_angle;
 vertices[1].x = (short) (endpoint->_x - (cos(temp_angle)
                    * ARROWSIDELENGTH));
 vertices[1].y = (short) (endpoint->_y - (sin(temp_angle) *
                    ARROWSIDELENGTH));
 temp_angle = angle + half_arrow_angle;
 vertices[2].x = (short) (endpoint->_x - (cos(temp_angle) *
                    ARROWSIDELENGTH));
 vertices[2].y = (short) (endpoint->_y - (sin(temp_angle) *
                    ARROWSIDELENGTH));
 XFillPolygon(parent->display_ptr(), parent->draw_window(),
         graphics_context, vertices, 3, Convex,
```

```
        CoordModeOrigin);
 XFillPolygon(parent->display_ptr(),
        *(parent->drawing_area_pixmap()),
        graphics_context, vertices, 3, Convex,
        CoordModeOrigin);
}

// Draws the spline point by point, and if the coordinates
// are near a spline point, TRUE is returned.

BOOLEAN SplineObject::hit(int in_x, int in_y) {
 int points;
 float inc;
 float t, t2, t3, term1, term2, term3, term4, x, y;
 _spline_node *p1, *p2, *p3, *p4;

 p1 = _head_ptr;
 p2 = p1->_next_ptr;
 p3 = p2->_next_ptr;
 p4 = p3->_next_ptr;
 points = num_points(p1, p2, p3, p4) / SKIPFACTOR;
 while(p4 != NULL) {
  inc = 1.0 / (float) points;
  for(t = 0.0; t <= 1.0; t += inc) {
   t2 = t * t;
   t3 = t2 * t;
   term1 = (-t3 + (3 * t2) - (3 * t) +1);
   term2 = ((3 * t3) - (t2 * 6) + 4);
   term3 = ((-3 * t3) + (3 * t2) + (3 * t) + 1);
   term4 = t3;
   x =  term1 * p1->_x;
   x += term2 * p2->_x;
   x += term3 * p3->_x;
   x += term4 * p4->_x;
   x /= 6.0;
   y =  term1 * p1->_y;
   y += term2 * p2->_y;
   y += term3 * p3->_y;
   y += term4 * p4->_y;
   y /= 6.0;
   if((abs((int) x - in_x) < HITDISTANCE) &&
     (abs((int) y - in_y) < HITDISTANCE))
    return TRUE;
  }
  p1 = p2;
  p2 = p3;
  p3 = p4;
  p4 = p4->_next_ptr;
  points = num_points(p1, p2, p3, p4) / SKIPFACTOR;
 }
 return FALSE;
}

// Draws the spline using the control points.

void SplineObject::draw(StreamObject *parent,
            GC graphics_context, GC handle_context,
            DRAW_STYLE style,
            EXTERN_STATUS status) {
 int points, i, j;
 float inc, t, t2, t3, term1, term2, term3, term4, x, y;
 _spline_node *p1, *p2, *p3, *p4, *temp_node_ptr;
 BOOLEAN need_handles;

 p1 = _head_ptr;
 p2 = p1->_next_ptr;
 p3 = p2->_next_ptr;
 p4 = p3->_next_ptr;
 points = num_points(p1, p2, p3, p4);
```

```
 while(p4 != NULL) {
/* Debug code
  if((p1->_x > 1000) || (p1->_x < 0))
   printf("Bogus draw p1x: %d id: %d", p1->_x, parent->id());
  if((p1->_y > 1000) || (p1->_y < 0))
   printf("Bogus draw p1y: %d id: %d", p1->_y, parent->id());
  if((p2->_x > 1000) || (p2->_x < 0))
   printf("Bogus draw p2x: %d id: %d", p2->_x, parent->id());
  if((p2->_y > 1000) || (p2->_y < 0))
   printf("Bogus draw p2y: %d id: %d", p2->_y, parent->id());
  if((p3->_x > 1000) || (p3->_x < 0))
   printf("Bogus draw p3x: %d id: %d", p3->_x, parent->id());
  if((p3->_y > 1000) || (p3->_y < 0))
   printf("Bogus draw p3y: %d id: %d", p3->_y, parent->id());
  if((p4->_x > 1000) || (p4->_x < 0))
   printf("Bogus draw p4x: %d id: %d", p4->_x, parent->id());
  if((p4->_y > 1000) || (p4->_y < 0))
   printf("Bogus draw p4y: %d id: %d", p4->_y, parent->id());
*/
  inc = 1.0 / (float) points;
  for(t = 0.0; t <= 1.0; t += inc) {
   t2 = t * t;
   t3 = t2 * t;
   term1 = (-t3 + (3 * t2) - (3 * t) +1);
   term2 = ((3 * t3) - (t2 * 6) + 4);
   term3 = ((-3 * t3) + (3 * t2) + (3 * t) + 1);
   term4 = t3;
   x =  term1 * p1->_x;
   x += term2 * p2->_x;
   x += term3 * p3->_x;
   x += term4 * p4->_x;
   x /= 6.0;

   y =  term1 * p1->_y;
   y += term2 * p2->_y;
   y += term3 * p3->_y;
   y += term4 * p4->_y;
   y /= 6.0;
   if(parent->is_state_variable()) { // draw thicker line
    for(i = (int) x - 1; i < (int) x + 1; i++)
     for(j = (int) y - 1; j < (int) y + 1; j++) {
      XDrawPoint(parent->display_ptr(),
            parent->draw_window(),
            graphics_context, i, j);
      XDrawPoint(parent->display_ptr(),
            *(parent->drawing_area_pixmap()),
            graphics_context, i, j);
     }
   }
   else {
    XDrawPoint(parent->display_ptr(), parent->draw_window(),
          graphics_context, (int) x, (int) y);
    XDrawPoint(parent->display_ptr(),
          *(parent->drawing_area_pixmap()),
          graphics_context, (int) x, (int) y);
   }
  }
  p1 = p2;
  p2 = p3;
  p3 = p4;
  p4 = p4->_next_ptr;
  points = num_points(p1, p2, p3, p4);
 }
 temp_node_ptr = _head_ptr->_next_ptr;
 if(((_handles_drawn == FALSE) && (style == SOLID)) ||
   ((_handles_drawn == TRUE) && (style == ERASE)))
  need_handles = TRUE;
 else
  need_handles = FALSE;
```

```
 if(parent->is_selected() && (status == FROM_EXTERNAL) &&
   need_handles)
    draw_handles(handle_context, parent, temp_node_ptr->_x,
          temp_node_ptr->_y);
 temp_node_ptr = temp_node_ptr->_next_ptr;
 while(temp_node_ptr->_next_ptr->_next_ptr != NULL) {
  if(parent->is_selected() && need_handles)
   draw_handles(handle_context, parent, temp_node_ptr->_x,
          temp_node_ptr->_y);
  temp_node_ptr = temp_node_ptr->_next_ptr;
 }
 draw_arrowhead(graphics_context, parent, temp_node_ptr);
 if(parent->is_selected() && (status == TO_EXTERNAL) &&
   need_handles)
  draw_handles(handle_context, parent, temp_node_ptr->_x,
          temp_node_ptr->_y);
 if(need_handles) {
  if(_handles_drawn == TRUE)
   _handles_drawn = FALSE;
  else
   _handles_drawn = TRUE;
 }
}

// Erases the control points.

void SplineObject::clear() {

 delete _head_ptr;
 _head_ptr = NULL;
}

// Adds a new control point to the spline.

void SplineObject::add(int x, int y) {
 _spline_node *temp_node_ptr;

/* Debug code
 if((x > 900) || (x < 0))
  printf("Bogus spline x: %d", x);
 if((y > 900) || (y < 0))
  printf("Bogus spline y: %d", x);
*/
 if(_head_ptr == NULL)
  _head_ptr = new _spline_node(x, y);
 else {
  temp_node_ptr = _head_ptr;
  while(temp_node_ptr->_next_ptr != NULL)
   temp_node_ptr = temp_node_ptr->_next_ptr;
  temp_node_ptr->_next_ptr = new _spline_node(x, y);
 }
}

// Returns the control point pointed to by the iterator
// pointer, then advances the iterator pointer.

XYPAIR SplineObject::next_pair() {
 XYPAIR temp_pair;

 if(_iter == NULL) {
  temp_pair.x = -1;
  temp_pair.y = -1;
 }
 else {
  temp_pair.x = _iter->_x;
  temp_pair.y = _iter->_y;
  _iter = _iter->_next_ptr;
 }
 return temp_pair;
}
```

```
// Resets the iterator pointer to the beginning of the spline.

void SplineObject::reset_iter() {

 _iter = _head_ptr;
}

// Places the name between the middle two control points.

void SplineObject::set_name_location(int &name_x, int &name_y) {

 _spline_node *temp_node_ptr = _head_ptr;
 int temp_x, temp_y, i, num_nodes = 0;

 while(temp_node_ptr != NULL) {
  num_nodes++;
  temp_node_ptr = temp_node_ptr->_next_ptr;
 }

 temp_node_ptr = _head_ptr;
 if(_head_ptr != NULL) {
  for(i = 1; i < num_nodes / 2; i++)
   temp_node_ptr = temp_node_ptr->_next_ptr;
  temp_x =
   (temp_node_ptr->_x + temp_node_ptr->_next_ptr->_x) / 2;
  temp_y =
   (temp_node_ptr->_y + temp_node_ptr->_next_ptr->_y) / 2;

  name_x = temp_x;
  name_y = temp_y;
 }
}

// Places the latency location underneath the name.

void SplineObject::set_latency_location(int temp_x, int temp_y,
                                        int &latency_x,
                                        int &latency_y) {

 latency_x = temp_x;
 latency_y = temp_y + 15;
}

// Convenience function for setting the text location.

void SplineObject::set_text_location(int &name_x, int &name_y,
                                     int &latency_x,
                                     int &latency_y) {

 set_name_location(name_x, name_y);
 set_latency_location(name_x, name_y, latency_x, latency_y);
}

// Checks to see if the coordinates are within any of the
// handles, or within the ends of external streams.

BOOLEAN SplineObject::hit_handle(int x, int y,
                                 EXTERN_STATUS status) {
 int num_handle = 2;
 _spline_node *temp_node_ptr = _head_ptr->_next_ptr;

 if(status != FROM_EXTERNAL) {
  temp_node_ptr = temp_node_ptr->_next_ptr;
  num_handle++;
 }
 while(temp_node_ptr->_next_ptr->_next_ptr != NULL) {
  if((((temp_node_ptr->_x) - (HANDLESIZE / 2) - HITFUDGE)
                                        <= x) &&
     (x <=
```

```
 ((temp_node_ptr->_x) + (HANDLESIZE / 2) + HITFUDGE)) &&
 (((temp_node_ptr->_y) - (HANDLESIZE / 2) - HITFUDGE) <= y) &&
 (y <= ((temp_node_ptr->_y) + (HANDLESIZE / 2) + HITFUDGE))) {
     _handle_selected = num_handle;
     return TRUE;
  }
  temp_node_ptr = temp_node_ptr->_next_ptr;
  num_handle++;
 }
 if(status == TO_EXTERNAL) {
  if((((temp_node_ptr->_x) - (HANDLESIZE / 2) - HITFUDGE)
                                  <= x) &&
    (x <=
      ((temp_node_ptr->_x) + (HANDLESIZE / 2) + HITFUDGE)) &&
       (((temp_node_ptr->_y) - (HANDLESIZE / 2) - HITFUDGE)
        <= y) &&
      (y <= ((temp_node_ptr->_y) + (HANDLESIZE / 2) +
          HITFUDGE))) {
     _handle_selected = num_handle;
     return TRUE;
  }
 }
 _handle_selected = NONE;
 return FALSE;
}

// Checks to see if the coordinates are within any of the
// handles, or within the ends of external streams.

BOOLEAN SplineObject::over_handle(int x, int y,                    //@4
                    EXTERN_STATUS status) {
 int num_handle = 2;
 _spline_node *temp_node_ptr = _head_ptr->_next_ptr;

 if(status != FROM_EXTERNAL) {
  temp_node_ptr = temp_node_ptr->_next_ptr;
  num_handle++;
 }
 while(temp_node_ptr->_next_ptr->_next_ptr != NULL) {
  if((((temp_node_ptr->_x) - (HANDLESIZE / 2) - HITFUDGE)
                                  <= x) &&
    (x <=
 ((temp_node_ptr->_x) + (HANDLESIZE / 2) + HITFUDGE)) &&
 (((temp_node_ptr->_y) - (HANDLESIZE / 2) - HITFUDGE) <= y) &&
 (y <= ((temp_node_ptr->_y) + (HANDLESIZE / 2) + HITFUDGE))) {
     return TRUE;
  }
  temp_node_ptr = temp_node_ptr->_next_ptr;
  num_handle++;
 }
 if(status == TO_EXTERNAL) {
  if((((temp_node_ptr->_x) - (HANDLESIZE / 2) - HITFUDGE)
                                  <= x) &&
    (x <=
      ((temp_node_ptr->_x) + (HANDLESIZE / 2) + HITFUDGE)) &&
       (((temp_node_ptr->_y) - (HANDLESIZE / 2) - HITFUDGE)
        <= y) &&
      (y <= ((temp_node_ptr->_y) + (HANDLESIZE / 2) +
          HITFUDGE))) {
     return TRUE;
  }
 }
 return FALSE;
}

// Moves a spline handle.

void SplineObject::move_handle(int x, int y) {
 _spline_node *temp_node_ptr = _head_ptr;
 int i;
```

```
 for(i = 1; i < _handle_selected; i++)
  temp_node_ptr = temp_node_ptr->_next_ptr;
 temp_node_ptr->_x += x;
 temp_node_ptr->_y += y;
 if(temp_node_ptr->_x < 0)
  temp_node_ptr->_x = 0;
 if(temp_node_ptr->_y < 0)
  temp_node_ptr->_y = 0;
}

// Erases the given handle

void SplineObject::erase_handle(GC graphics_context,
                    StreamObject *parent) {
 _spline_node *temp_node_ptr = _head_ptr;
 int i;

 for(i = 1; i < _handle_selected; i++)
  temp_node_ptr = temp_node_ptr->_next_ptr;
 draw_handles(graphics_context, parent, temp_node_ptr->_x,
        temp_node_ptr->_y);
}

// Determines the location for the 'EXTERNAL' label on
// the drawing.

void SplineObject::set_extern_location(XYPAIR &extern_location,
                EXTERN_STATUS status) {
 _spline_node *temp_node_ptr, *intercept_ptr, *shadow_ptr;
 int temp_x, temp_y;

 if(status == FROM_EXTERNAL) {
  shadow_ptr = _head_ptr;
  intercept_ptr = _head_ptr->_next_ptr;
 }
 else
  if(status == TO_EXTERNAL) {
   temp_node_ptr = _head_ptr;
   while(temp_node_ptr->_next_ptr->_next_ptr != NULL)
    temp_node_ptr = temp_node_ptr->_next_ptr;
   intercept_ptr = temp_node_ptr;
   shadow_ptr = temp_node_ptr->_next_ptr;
  }
 if(shadow_ptr->_y > intercept_ptr->_y)
  temp_y = intercept_ptr->_y + 7;
 else
  temp_y = intercept_ptr->_y - 7;
 if(shadow_ptr->_y == intercept_ptr->_y)
  if(shadow_ptr->_x > intercept_ptr->_x)
   temp_x = EXTERN_OFFSET;
  else
   temp_x = -EXTERN_OFFSET;
 else
  temp_x = (temp_y - intercept_ptr->_y) /
   ((shadow_ptr->_y) - (intercept_ptr->_y)) *
    ((shadow_ptr->_x) - (intercept_ptr->_x));
 if(abs(temp_x) > EXTERN_OFFSET)
  if(shadow_ptr->_x > intercept_ptr->_x)
   temp_x = EXTERN_OFFSET;
  else
   temp_x = -EXTERN_OFFSET;
 temp_x += intercept_ptr->_x;
 temp_y += 10;
 extern_location.x = temp_x;
 extern_location.y = temp_y;
}

void SplineObject::copy(SPLINE_PTR ptr) {
```

```
 SPLINE_PTR   SP;
 _spline_node *SN, *prevSN;

 // Get rid of original data
 clearList();

 if (ptr == NULL)
  _head_ptr = NULL;
 else {
  prevSN = NULL;
  SP    = ptr;

  while (SP != NULL) {
   // Create the space, initialized
   SN = new _spline_node(SP->x,SP->y);
   SP = SP->next;

#ifdef GE_DEBUG
 cout << "spline copy: x" << SN->_x << "  y" << SN->_y << endl;
#endif

   // Add new node to linked list
   if (prevSN == NULL) {
        _head_ptr = SN;
        prevSN   = SN;
   } else {
        prevSN->_next_ptr = SN;
        prevSN = SN;
   }
  }
 }
}

void SplineObject::copy_back(SPLINE_PTR &ptr, EXTERN_STATUS
status) {

 SPLINE_PTR   SP = NULL, prevSP = NULL;
 _spline_node* sn;

 // Get rid of original data
 spline_delete(ptr);

 sn = _head_ptr;

 if (status == FROM_EXTERNAL) {
#ifdef GE_DEBUG
 cout << "spline copy_back: FROM_EXTERN" << endl;
#endif
  sn = sn->_next_ptr;
 }
 else
  sn = sn->_next_ptr->_next_ptr;

 while (sn->_next_ptr->_next_ptr != NULL) {
  // Create the space
  SP = (SPLINE_PTR) malloc(sizeof(SPLINE_NODE));

  // Copy over the data
  SP->x   = sn->_x;
  SP->y   = sn->_y;
  SP->next = NULL;

#ifdef GE_DEBUG
 cout << "spline copy_back: x" << SP->x << "  y" << SP->y << endl;
#endif

  // Add new node to linked list
  if (prevSP == NULL) {
   ptr   = SP;
   prevSP = SP;
```

```
  } else {
   prevSP->next = SP;
   prevSP      = SP;
  }
  sn = sn->_next_ptr;
 }

 if (status == TO_EXTERNAL) {
  SP = (SPLINE_PTR) malloc(sizeof(SPLINE_NODE));

  // Copy over the data
  SP->x    = sn->_x;
  SP->y    = sn->_y;
  SP->next = NULL;

#ifdef GE_DEBUG
 cout << "spline copy_back: x" << SP->x << "  y" << SP->y
     << " TO_EXTERN " << endl;
#endif

  // Add new node to linked list
  if (prevSP == NULL) {
   ptr    = SP;
   prevSP = SP;
  } else {
   prevSP->next = SP;
   prevSP      = SP;
  }
 }
}

void SplineObject::clearList() {

 _spline_node *sn, *snNext;

 sn = _head_ptr;

 while (sn != NULL) {
  snNext = sn->_next_ptr;
  delete sn;
  sn = snNext;
 }
 delete _head_ptr;
}
```

```
/* ********************************************************

Name:        stream.h
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 11 Sep 92
Remarks:  Specification for the StreamObject class.

      The StreamObject is a graphical representation of
      a curved PSDL stream.  The stream is drawn as a
      b-spline specified by a series of control points.
      Streams automatically connect to their attached
      operators, and always have arrowheads at their
      terminating points.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

 @1   KBM    15 Aug 96
     Changed include file to a .H from .h for C++.

 @2   KBM    15 Aug 96
     Added new data items from ge_interface.h

 @3   KBM    15 Aug 96
     Moved common items up to GraphObject.

 @4   KBM    15 Aug 96
     Renamed methods:  write_to_disk  -> to_psdl
                       build_from_disk -> from_psdl

 @5   KBM    15 Aug 96
     Added local operator new and delete for debugging

 @6   KBM    20 Aug 96
     Created a new function over() which is just like hit()
          except that it does not select objects

 @7   KBM    20 Aug 96
     Created a new function over_handle() which is just like
          hit_handle() except that it does not select objects

 @8   KBM    30 Aug 96
     Modifications for latency units.

 @9   KBM    30 Aug 96
     Removed set_constraint.

********************************************************
*/

#ifndef stream_object_h
#define stream_object_h 1

#include <stdio.h>
#include <X11/Xlib.h>
#include "ge_defs.h"
#include "ge_interface.h"                    //@2
#include "graph_object.H"                    //@1
#include "spline_object.H"                   //@1
#include "operator_object.H"                 //@1

class GraphObjectList;

class StreamObject : public GraphObject {

protected:
```

```
 char    *_latency_string_ptr;                    //@3
 int     _latency_width;
 int     _latency_height;
 BOOLEAN _latency_selected;
 BOOLEAN _st_handles_drawn;
 BOOLEAN _latency_handles_drawn;
 XYPAIR  name_location;
 XYPAIR  lat_location;
 XYPAIR  extern_location;

//------------------------------------------------------------- //@2
// StreamObject

 /* the two endpoints of the stream */
 int
     _from,
     _to;

 OperatorObject
     *_from_ptr,
     *_to_ptr;

 /* linked list of SPLINE_NODE defining the shape of the edge on the
display */
 SplineObject
     _arc;          /* null pointer if the _arc is a
                    straight line */

 /* info about LATENCY */
 int
     _latency,              /* UNDEFINED_TIME if not specified */
     _latency_unit,         /* US, MS, SECOND, MINUTE, HOUR */
     _latency_font,
     _latency_x,  /* actual x_position = x_mid_point + _latency_x */
     _latency_y;  /* actual Y_position = y_mid_point + _latency_y */

 /* stream type */
 char*
     _stream_type_name;

 /* initial value if it is a state stream */
 char*
     _state_initial_value;

 /* stream visible properties */
 BOOLEAN
     _is_state_variable;

//------------------------------------------------------------- //@2


public:

 StreamObject();
 StreamObject(char *in_name_ptr, int in_id, int in_from,
         int in_to, int in_latency, int in_latency_unit,    //@8
             SplineObject in_arc,
         BOOLEAN in_is_new, BOOLEAN in_is_state_variable);
 StreamObject(STREAM st_op);
 ~StreamObject() {
  delete _name_ptr; delete _latency_string_ptr;
 }

 void set_default_text_location();
 void set_default_name_location();
 void set_default_latency_location();
 void StreamObject::draw_handles(GC draw_context, int x1,
                         int y1, int x2, int y2);

#ifdef GE_DEBUG
```

```
 void* operator new(size_t size) {                    //@5
  cout << "stream new" << endl;
  return ::operator new(size);
 }
 void  operator delete(void* s) {                     //@5
  cout << "stream delete" << endl;
  ::operator delete(s);
 }
#endif

 GE_STATUS to_psdl(FILE *outfile);                    //@4
 GE_STATUS from_psdl(FILE *infile);                   //@4
 GE_STATUS build_from_property();
 GE_STATUS write_to_property();

 void draw_spline(DRAW_STYLE style);
 void draw(DRAW_STYLE style);
 void draw_text(DRAW_STYLE style);
 void move_text(int x, int y);
 void select();
 void unselect();
 BOOLEAN hit(int x, int y);
 BOOLEAN over(int x, int y);                          //@6
 CLASS_DEF is_a() {return STREAMOBJECT;}
 void clearObj();
 void copy(StreamObject *st_ptr);
 void copy_back(StreamObject *st_ptr);
 void copy_back(STREAM st_ptr);
 void set_object_ptrs(GraphObjectList *parent);
 void delete_notify(CLASS_DEF class_type, int deleted_obj_id);
 void replace_name(char *new_name);
 // void set_constraint(int constraint, int units);   //@9
 void move_notify(CLASS_DEF object_type, int object_id);
 BOOLEAN is_selected() {return _is_selected;}
 BOOLEAN hit_handle(int x, int y);
 BOOLEAN over_handle(int x, int y);                   //@7
 void move_handle(int x, int y) {_arc.move_handle(x, y);}
 BOOLEAN text_selected()
          {return (_name_selected || _latency_selected);}
 void set_text_dimensions();
 void set_object_font(int font_id);
 void undelete_notify(CLASS_DEF class_type, int deleted_obj_id);
 void reset_handles_drawn_state();
 int text_width();
 int text_height();
 void text_locate(int x, int y);
 void erase_handle() {_arc.erase_handle(_graphics_context, this);}
 int constraint() {return _latency;}

//------------------------------------------------------------ //@2
// StreamObject
//
// /* the two endpoints of the stream */
// int
//      _from,
//      _to;

 int from()       {return _from;}
 void set_from(int f) {_from = f;}
 int to()         {return _to;}
 void set_to(int t)   {_to = t;}

//
// OperatorObject
//      *_from_ptr,
//      *_to_ptr;

 OperatorObject*  from_ptr()                  {return _from_ptr;}
 void         set_from_ptr(OperatorObject *O) {_from_ptr = O;}
 OperatorObject*  to_ptr()                    {return _to_ptr;}
```

```
 void          set_to_ptr(OperatorObject *O)   {_to_ptr = O;}

//
// /* linked list of SPLINE_NODE defining the shape of the edge on the
display */
// SplineObject
//      _arc;           /* null pointer if the _arc is a
//                      straight line */

 SplineObject arc()                    {return _arc;}
 void       set_arc(SplineObject arc_ptr) {_arc = arc_ptr;}
 // void       delete_arc();  ?? This function needs to be defined to do a
deep deletion

//
// /* info about LATENCY */
// int
//      _latency,               /* UNDEFINED_TIME if not specified */
//      _latency_unit,          /* US, MS, SECOND, MINUTE, HOUR */
//      _latency_font,
//      _latency_x, /* actual x_position = x_mid_point + _latency_x */
//      _latency_y; /* actual Y_position = y_mid_point + _latency_y */

  int    latency()               { return _latency; }
  void   set_latency(int t)        { _latency = t; }
  void   clear_latency()          { _latency = UNDEFINED_TIME; }

  int    latency_unit()           { return _latency_unit; }
  void   set_latency_unit(int units)  { _latency_unit = units; }
  void   default_latency_unit()      { _latency_unit = MS; }

  int    latency_font()           { return _latency_font; }
  void   set_latency_font(int font)   { _latency_unit = font; }
//void   default_latency_font()      { _latency_unit = ; }

  int    latency_x()              { return _latency_x; }
  int    latency_y()              { return _latency_y; }
  void   set_latency_x(int x)       { _latency_x = x; }
  void   set_latency_y(int y)       { _latency_y = y; }

//
// /* stream type */
// char*
//      _stream_type_name;

 char*   stream_type_name()
        {return _stream_type_name;}
 void   set_stream_type_name(char* in_ptr) {      // NOTE: Uses
existing memory
         delete _stream_type_name;
              _stream_type_name = in_ptr;
            }

//
// /* initial value if it is a state stream */
// char*
//      _state_initial_value;

 char*   state_initial_value()
        {return _state_initial_value;}
 void   set_state_initial_value(char* in_ptr) {      // NOTE: Uses existing
memory
         delete _state_initial_value;
              _state_initial_value = in_ptr;
            }

//
// /* stream visible properties */
// BOOLEAN
//      _is_state_variable;
```

```
 BOOLEAN is_state_variable()
        {return _is_state_variable;}
 void   set_state_variable(BOOLEAN state)
        {_is_state_variable = state;}

//
//------------------------------------------------------------------ //@2

};

#endif;
```

```
/* ********************************************************

Name:        stream_object.C
Author:      Capt Robert M. Dixon
Program:     graph_editor
Date Modified: 11 Sep 92
Remarks:  Implementation of the StreamObject class.

      The StreamObject is a graphical representation of
      a curved PSDL stream.  The stream is drawn as a
      b-spline specified by a series of control points.
      Streams automatically connect to their attached
      operators, and always have arrowheads at their
      terminating points.


Credits:  Portions of code are adapted from the following:

      Barakati, Naba, X Window System Programming, SAMS,
        1991.

      Heller, Dan, Motif Programming Manual, O'Reilly and
        Associates, 1991.

      Johnson, Eric, and Reichard, Kevin, X Window
        Applications Programming, MIS Press, 1989.

      Young, Douglas, Object Oriented Programming With C++
        and OSF/Motif, Prentice-Hall, 1992.

********************************************************
Modification History:

 Baseline taken from Robert M. Dixon

@1   KBM    15 Aug 96
   Changed include file to a .H from .h for C++.

@2   KBM    15 Aug 96
   Renamed methods:   write_to_disk  -> to_psdl
                      build_from_disk -> from_psdl

@3   KBM    15 Aug 96
   Moved common stuff up to GraphObject

@4   KBM    20 Aug 96
   Created a new function over() which is just like hit()
      except that it does not select objects

@5   KBM    20 Aug 96
   Created a new function over_handle() which is just like
      hit_handle() except that it does not select objects

@6   KBM    30 Aug 96
   Handle new time units.  time_unit_str function created
      in ge_utilities and used.  No longer use negative values
      to flag units.

@7   WH   31 Aug 96
   Added standardized comments to each method

********************************************************
*/

#include <stdlib.h>
#include <string.h>
#include <stream.h>
#include "stream_object.H"                    //@1
#include "graph_object_list.H"                //@1
#include "ge_utilities.H"                     //@6
```

```
#define MAX_CONTROL_POINTS 100

/*********************************************************
* StreamObject inherits from GraphObject
*********************************************************/

StreamObject::StreamObject() : GraphObject() {

 _next_ptr         = NULL;
 _is_selected      = FALSE;
 _is_new           = FALSE;
 _is_deleted       = FALSE;
 _is_modified      = FALSE;

 _id               = 0;

 _name_ptr         = NULL;
 _name_font        = _default_font;
 _name_x           = NULL_VALUE;
 _name_y           = NULL_VALUE;
 _name_selected    = FALSE;

 _to_ptr           = NULL;
 _from_ptr         = NULL;

 // _arc           = NULL; //??? ge_interface.h says this is NULL for
straight line

 _latency          = UNDEFINED_TIME;
 // default_latency_unit();
 _latency_string_ptr = NULL;
 _latency_font     = _default_font;
 _latency_x        = NULL_VALUE;
 _latency_y        = NULL_VALUE;
 _latency_selected = FALSE;

 _stream_type_name = NULL;

 _state_initial_value = NULL;

 _is_state_variable = FALSE;

 set_text_dimensions();
 reset_handles_drawn_state();
}

/*********************************************************
* StreamObject constructor
*********************************************************/

StreamObject::StreamObject(char *in_name_ptr, int in_id,
   int in_from, int in_to, int in_latency, int in_latency_unit,
   SplineObject in_arc, BOOLEAN in_is_new,
   BOOLEAN in_is_state_variable) : GraphObject() {

 _name_ptr         = strdup(in_name_ptr);
 _name_font        = _default_font;
 _id               = in_id;
 _from             = in_from;
 _to               = in_to;
 _latency          = in_latency;
 _latency_unit     = in_latency_unit;
 _latency_string_ptr = time_unit_str(_latency,_latency_unit);
 _latency_font     = _default_font;
 _arc              = in_arc;
 _is_state_variable = in_is_state_variable;

 // the following 4 initializations were added by M.T. Shing
 _name_x           = NULL_VALUE;
```

```
 _name_y          = NULL_VALUE;
 _latency_x       = NULL_VALUE;
 _latency_y       = NULL_VALUE;

 _is_deleted      = FALSE;
 _is_new          = in_is_new;
 _is_modified     = FALSE;
 _is_selected     = FALSE;
 _name_selected   = FALSE;
 _latency_selected = FALSE;
 _from_ptr        = NULL;
 _to_ptr          = NULL;
 _stream_type_name  = NULL;
 _state_initial_value = NULL;
 set_text_dimensions();
 reset_handles_drawn_state();
}

StreamObject::StreamObject(STREAM st_ptr) {
 char  *temp_str;
 ID_LIST id_ptr;

 _next_ptr        = NULL;
 // _is_selected      = st_ptr->is_selected;
 // _name_handles_drawn  = st_ptr->_name_handles_drawn;
 // _name_selected     = st_ptr->_name_selected;
 // _handle_selected   = st_ptr->_handle_selected;
 // _name_width        = st_ptr->_name_width;
 // _name_height       = st_ptr->_name_height;
 _id              = st_ptr->id;
 _is_deleted      = st_ptr->is_deleted;
 _is_new          = st_ptr->is_new;
 _is_modified     = st_ptr->is_modified;
 _name_ptr        = make_str(st_ptr->label);
 _name_font       = st_ptr->label_font;
 _name_x          = st_ptr->label_x_offset;
 _name_y          = st_ptr->label_y_offset;

 // _latency_width      = st_ptr->_latency_width;
 // _latency_height     = st_ptr->_latency_height;
 // _latency_selected   = st_ptr->_latency_selected;
 // _st_handles_drawn   = st_ptr->_st_handles_drawn;
 // _latency_handles_drawn = st_ptr->_latency_handles_drawn;
 // name_location       = st_ptr->name_location;
 // lat_location        = st_ptr->lat_location;
 // extern_location     = st_ptr->extern_location;

 if (st_ptr->from == NULL)
  _from           = 0;
 else
  _from           = st_ptr->from->id;
 if (st_ptr->to  == NULL)
  _to             = 0;
 else
  _to             = st_ptr->to->id;
 _arc.copy(st_ptr->arc);
 _latency         = st_ptr->latency;
 _latency_unit    = st_ptr->latency_unit;
 delete _latency_string_ptr;
 _latency_string_ptr  = time_unit_str(_latency,_latency_unit);  //@6
 _latency_font    = st_ptr->latency_font;
 _latency_x       = st_ptr->latency_x_offset;
 _latency_y       = st_ptr->latency_y_offset;

 _stream_type_name   = make_str(st_ptr->stream_type_name);
 _state_initial_value  = make_str(st_ptr->state_initial_value);

 _is_state_variable   = st_ptr->is_state_variable;

 set_text_dimensions();
```

```
 reset_handles_drawn_state();
}


/*********************************************************
* copy StreamObject members from GraphObject
*********************************************************/

void StreamObject::copy(StreamObject *st_ptr) {
 char buffer[INPUT_LINE_SIZE];

 _next_ptr = st_ptr->_next_ptr;
 _is_selected = st_ptr->_is_selected;
 _name_handles_drawn = st_ptr->_name_handles_drawn;
 _name_selected = st_ptr->_name_selected;
 _handle_selected = st_ptr->_handle_selected;
 _name_width = st_ptr->_name_width;
 _name_height = st_ptr->_name_height;
 _id = st_ptr->_id;
 _is_deleted = st_ptr->_is_deleted;
 _is_new = st_ptr->_is_new;
 _is_modified = st_ptr->_is_modified;
 _name_ptr = make_str(st_ptr->_name_ptr);
 _name_font = st_ptr->_name_font;
 _name_x = st_ptr->_name_x;
 _name_y = st_ptr->_name_y;

 _latency_width = st_ptr->_latency_width;
 _latency_height = st_ptr->_latency_height;
 _latency_selected = st_ptr->_latency_selected;
 _st_handles_drawn = st_ptr->_st_handles_drawn;
 _latency_handles_drawn = st_ptr->_latency_handles_drawn;
 name_location = st_ptr->name_location;
 lat_location = st_ptr->lat_location;
 extern_location = st_ptr->extern_location;
 _from = st_ptr->_from;
 _to = st_ptr->_to;
 _from_ptr = st_ptr->_from_ptr;
 _to_ptr = st_ptr->_to_ptr;
 _arc = st_ptr->_arc;              //??? Do we want to duplicate arc?
 _latency = st_ptr->_latency;
 _latency_unit = st_ptr->_latency_unit;
 delete _latency_string_ptr;
 _latency_string_ptr = time_unit_str(_latency,_latency_unit);  //@6
 _latency_font = st_ptr->_latency_font;
 _latency_x = st_ptr->_latency_x;
 _latency_y = st_ptr->_latency_y;
 _stream_type_name = make_str(st_ptr->_stream_type_name);
 _state_initial_value = make_str(st_ptr->_state_initial_value);
 _is_state_variable = st_ptr->_is_state_variable;

 set_text_dimensions();
 reset_handles_drawn_state();
}

/*********************************************************
 * copy StreamObject members into GraphObject
 *********************************************************/

void StreamObject::copy_back(StreamObject *st_ptr) {
 char buffer[INPUT_LINE_SIZE];

 st_ptr->_next_ptr          = _next_ptr;
 st_ptr->_is_selected       = _is_selected;
 st_ptr->_name_handles_drawn  = _name_handles_drawn;
 st_ptr->_name_selected     = _name_selected;
 st_ptr->_handle_selected   = _handle_selected;
 st_ptr->_name_width        = _name_width;
 st_ptr->_name_height       = _name_height;
```

```
 st_ptr->_id              = _id;
 st_ptr->_is_deleted      = _is_deleted;
 st_ptr->_is_new          = _is_new;
 st_ptr->_is_modified     = _is_modified;
 st_ptr->_name_ptr        = make_str(_name_ptr);
 st_ptr->_name_font       = _name_font;
 st_ptr->_name_x          = _name_x;
 st_ptr->_name_y          = _name_y;

 st_ptr->_latency_width       = _latency_width;
 st_ptr->_latency_height      = _latency_height;
 st_ptr->_latency_selected    = _latency_selected;
 st_ptr->_st_handles_drawn    = _st_handles_drawn;
 st_ptr->_latency_handles_drawn = _latency_handles_drawn;
 st_ptr->name_location        = name_location;
 st_ptr->lat_location         = lat_location;
 st_ptr->extern_location      = extern_location;

 st_ptr->_from            = _from;
 st_ptr->_to              = _to;
 st_ptr->_from_ptr        = _from_ptr;
 st_ptr->_to_ptr          = _to_ptr;
 st_ptr->_arc             = _arc;          //??? Do we want to duplicate
arc
 st_ptr->_latency         = _latency;
 st_ptr->_latency_unit    = _latency_unit;
 st_ptr->_latency_font    = _latency_font;
 delete st_ptr->_latency_string_ptr;
 st_ptr->_latency_string_ptr  = time_unit_str(_latency,_latency_unit);
//@6
 st_ptr->_latency_x       = _latency_x;
 st_ptr->_latency_y       = _latency_y;
 st_ptr->_stream_type_name    = make_str(_stream_type_name);
 st_ptr->_state_initial_value = make_str(_state_initial_value);
 st_ptr->_is_state_variable   = _is_state_variable;

 set_text_dimensions();
 reset_handles_drawn_state();
}


/*********************************************************
 * Copy StreamObject members into STREAM.
 * NOTE:  from and to are not assigned in this routine.  But need to be
 * updated by calling program.
 *********************************************************/

void StreamObject::copy_back(STREAM st_ptr) {

 st_ptr->id               = _id;

 st_ptr->label            = make_str(_name_ptr);
 st_ptr->label_font       = _name_font;
 st_ptr->label_x_offset   = _name_x;
 st_ptr->label_y_offset   = _name_y;

 st_ptr->from             = NULL;  // Must be updated by calling
 st_ptr->to               = NULL;  // program.
 if (_from == 0)
  _arc.copy_back(st_ptr->arc, FROM_EXTERNAL);
 else if (_to == 0)
  _arc.copy_back(st_ptr->arc, TO_EXTERNAL);
 else
  _arc.copy_back(st_ptr->arc, NO_EXTERNAL);

 st_ptr->latency          = _latency;
 st_ptr->latency_unit     = _latency_unit;
 st_ptr->latency_font     = _latency_font;
 st_ptr->latency_x_offset = _latency_x;
 st_ptr->latency_y_offset = _latency_y;
```

```
 st_ptr->stream_type_name     = make_str(_stream_type_name);
 st_ptr->state_initial_value  = make_str(_state_initial_value);

 st_ptr->is_state_variable    = _is_state_variable;

 st_ptr->is_new               = _is_new;
 st_ptr->is_modified          = _is_modified;
 st_ptr->is_deleted           = _is_deleted;

}

void StreamObject::clearObj() {

 free(_name_ptr);            _name_ptr         = NULL;
 free(_latency_string_ptr);  _latency_string_ptr = NULL;
                    _from_ptr         = NULL;
                    _to_ptr           = NULL;
 _arc.clearList();
 free(_stream_type_name);    _stream_type_name  = NULL;
 free(_state_initial_value); _state_initial_value = NULL;
}

/*********************************************************
* build from the disk property method
*********************************************************/


GE_STATUS StreamObject::build_from_property() {

 return FAILED;
}


/*********************************************************
*  Builds the stream from disk.
*********************************************************/

GE_STATUS StreamObject::from_psdl(FILE *infile) {          //@2
 char buffer[INPUT_LINE_SIZE + 1], *error_str_ptr = NULL;
 int last_char;
 GE_STATUS status = SUCCEEDED;

 fgets(buffer, INPUT_LINE_SIZE, infile);
 last_char = strlen(buffer) - 1;
 if(buffer[last_char] == '\n')
  buffer[last_char] = 0;
 if(strcmp(buffer, "ENDDATA") == 0)
  return ENDED;
 else {
  _name_ptr = strdup(buffer);

  fgets(buffer, INPUT_LINE_SIZE, infile);
  if(valid_num_string(buffer)) {
   _name_font = atoi(buffer);
   if(_name_font > MAXFONTS)
    error_str_ptr = strdup("Name Font Too Large");
  }
  else
   error_str_ptr = strdup("Corrupted Name Font");

  if(error_str_ptr == NULL) {
   fgets(buffer, INPUT_LINE_SIZE, infile);
   if(valid_num_string(buffer))
    _name_x = atoi(buffer);
   else
    error_str_ptr = strdup("Corrupted name_x");
  }

  if(error_str_ptr == NULL) {
```

```
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer))
  _name_y = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted name_y");
}

if(error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer))
  _id = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted id");
}

if(error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer))
  _from = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted \"from\"");
}

if(error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer)) {
  _to = atoi(buffer);
  if((_from == 0) && (_to == 0))
   error_str_ptr =
    strdup("From & To pointers both NULL.\n");
 }
 else
  error_str_ptr = strdup("Corrupted \"to\"");
}
```



```
if(error_str_ptr == NULL) {
 status = _arc.from_psdl(infile);                    //@2
 if(status == FAILED)
  error_str_ptr = strdup("Unable to build spline");
}

if(error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer))
  _latency = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted latency");
}

if(error_str_ptr == NULL) {
 if(_latency == UNDEFINED_TIME)
  _latency_string_ptr = NULL;
 else {
  if(_latency < 0)
   sprintf(buffer, "%d us", -_latency);
  else
   sprintf(buffer, "%d ms", _latency);
  _latency_string_ptr = strdup(buffer);
 }
}

if(error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer)) {
  _latency_font = atoi(buffer);
  if(_latency_font > MAXFONTS)
   error_str_ptr = strdup("Latency Font Too Large");
 }
 else
  error_str_ptr = strdup("Corrupted Latency Font");
```

```
}

if(error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer))
  _latency_x = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted latency_x");
}

if(error_str_ptr == NULL) {
 fgets(buffer, INPUT_LINE_SIZE, infile);
 if(valid_num_string(buffer))
  _latency_y = atoi(buffer);
 else
  error_str_ptr = strdup("Corrupted latency_y");
}

if(error_str_ptr == NULL) {
 if(strcmp("TRUE\n",
   fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
  _is_deleted = TRUE;
 else
  if(strcmp("FALSE\n", buffer) == 0)
   _is_deleted = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_deleted");
}

if(error_str_ptr == NULL) {
 if(strcmp("TRUE\n",
   fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)

//Change this if original _is_new should be saved.
```



```
  _is_new = FALSE;
 else
  if(strcmp("FALSE\n", buffer) == 0)
   _is_new = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_new");
}

if(error_str_ptr == NULL) {
 if(strcmp("TRUE\n",
   fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
  _is_modified = TRUE;
 else
  if(strcmp("FALSE\n", buffer) == 0)
   _is_modified = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_modified");
}

if(error_str_ptr == NULL) {
 if(strcmp("TRUE\n",
   fgets(buffer, INPUT_LINE_SIZE, infile)) == 0)
  _is_state_variable = TRUE;
 else
  if(strcmp("FALSE\n", buffer) == 0)
   _is_state_variable = FALSE;
  else
   error_str_ptr = strdup("Corrupted is_state_variable");
}

_is_selected = FALSE;
_name_selected = FALSE;
_latency_selected = FALSE;

if(error_str_ptr == NULL) {
```

```
    set_text_dimensions();
   }

   if(error_str_ptr != NULL) {
    status = FAILED;
    sprintf(buffer, "Stream %s: %s", _name_ptr, error_str_ptr);
    error_box(buffer);
    free(error_str_ptr);
   }

   if(ferror(infile)) {
    sprintf(buffer,
      "Unix reported an error building stream %s", _name_ptr);
    error_box(buffer);
    clearerr(infile);
    status = FAILED;
   }
  }
  return status;
}


/*******************************************************
* Draws handles around the specified location.  Handles are
* drawn in exclusive-or mode to simplify erasing them without
* disturbing the underlying text.
*******************************************************/

void StreamObject::draw_handles(GC draw_context, int x1, int y1,
                    int x2, int y2) {

  XSetFunction(_display_ptr, draw_context, GXxor);
  XFillRectangle(_display_ptr, _draw_window, draw_context, x1,
          y1, HANDLESIZE, HANDLESIZE);
  XFillRectangle(_display_ptr, _draw_window, draw_context,
          x2 - HANDLESIZE, y1, HANDLESIZE, HANDLESIZE);
  XFillRectangle(_display_ptr, _draw_window, draw_context, x1,
          y2 - HANDLESIZE, HANDLESIZE, HANDLESIZE);
  XFillRectangle(_display_ptr, _draw_window, draw_context,
          x2 - HANDLESIZE, y2 - HANDLESIZE, HANDLESIZE,
          HANDLESIZE);
  XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x1, y1, HANDLESIZE, HANDLESIZE);
  XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x2 - HANDLESIZE, y1, HANDLESIZE,
          HANDLESIZE);
  XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x1, y2 - HANDLESIZE, HANDLESIZE,
          HANDLESIZE);
  XFillRectangle(_display_ptr, *_drawing_area_pixmap,
          draw_context, x2 - HANDLESIZE, y2 - HANDLESIZE,
          HANDLESIZE, HANDLESIZE);
  XSetFunction(_display_ptr, draw_context, GXcopy);
}

/*******************************************************
//  Draws text strings.
*******************************************************/

void StreamObject::draw_text(DRAW_STYLE style) {
  GC draw_context;

  if(style == SOLID)
   draw_context = _graphics_context;
  else
   if(style == ERASE)
    draw_context = _erase_context;
  set_font(draw_context, _name_font);
  XDrawString(_display_ptr, _draw_window, draw_context,
         _name_x, _name_y, _name_ptr, strlen(_name_ptr));
```

```
XDrawString(_display_ptr, *_drawing_area_pixmap, draw_context,
        _name_x, _name_y, _name_ptr, strlen(_name_ptr));
if(_name_selected) {
 if((_name_handles_drawn == FALSE) && (style == SOLID)) {
  draw_handles(_graphics_context, _name_x - HANDLESIZE,
            _name_y - _name_height - HANDLESIZE,
            _name_x + _name_width + HANDLESIZE,
            _name_y + HANDLESIZE);
  _name_handles_drawn = TRUE;
 }
 else
  if((_name_handles_drawn == TRUE) && (style == ERASE)) {
   draw_handles(_graphics_context, _name_x - HANDLESIZE,
             _name_y - _name_height - HANDLESIZE,
             _name_x + _name_width + HANDLESIZE,
             _name_y + HANDLESIZE);
   _name_handles_drawn = FALSE;
  }
}

if(_latency != UNDEFINED_TIME) {
 set_font(draw_context, _latency_font);
 XDrawString(_display_ptr, _draw_window, draw_context,
         _latency_x, _latency_y, _latency_string_ptr,
         strlen(_latency_string_ptr));
 XDrawString(_display_ptr, *_drawing_area_pixmap,
         draw_context, _latency_x, _latency_y,
         _latency_string_ptr,
         strlen(_latency_string_ptr));
 if(_latency_selected) {
  if((_latency_handles_drawn == FALSE) && (style == SOLID)) {
   draw_handles(_graphics_context, _latency_x - HANDLESIZE,
             _latency_y - _latency_height - HANDLESIZE,
             _latency_x + _latency_width + HANDLESIZE,
             _latency_y + HANDLESIZE);
   _latency_handles_drawn = TRUE;
  }
  else
   if((_latency_handles_drawn == TRUE) &&
     (style == ERASE)) {
    draw_handles(_graphics_context,
             _latency_x - HANDLESIZE,
             _latency_y - _latency_height - HANDLESIZE,
             _latency_x + _latency_width + HANDLESIZE,
             _latency_y + HANDLESIZE);
    _latency_handles_drawn = FALSE;
   }
  }
 }
}


/*********************************************************
* write to disk property
*********************************************************/

GE_STATUS StreamObject::write_to_property() {
 return FAILED;
}

/*********************************************************
*  Writes the graphic attributes of the stream to disk.
*********************************************************/

GE_STATUS StreamObject::to_psdl(FILE *outfile) {          //@2
 char buffer[INPUT_LINE_SIZE + 1];

 fprintf(outfile, "%s\n", _name_ptr);
 sprintf(buffer, "%d\n%d\n%d", _name_font, _name_x, _name_y);
 fprintf(outfile, "%s\n", buffer);
```

```
sprintf(buffer, "%d", _id);
fprintf(outfile, "%s\n", buffer);
sprintf(buffer, "%d", _from);
fprintf(outfile, "%s\n", buffer);
sprintf(buffer, "%d", _to);
fprintf(outfile, "%s\n", buffer);
if(_from == 0)
  _arc.to_psdl(outfile, FROM_EXTERNAL);                    //@2
else
 if(_to == 0)
   _arc.to_psdl(outfile, TO_EXTERNAL);                     //@2
 else
   _arc.to_psdl(outfile, NO_EXTERNAL);                     //@2
sprintf(buffer, "%d", _latency);
fprintf(outfile, "%s\n", buffer);
sprintf(buffer, "%d\n%d\n%d", _latency_font, _latency_x,
      _latency_y);
fprintf(outfile, "%s\n", buffer);
if(_is_deleted)
 fprintf(outfile, "TRUE\n");
else
 fprintf(outfile, "FALSE\n");
if(_is_new)
 fprintf(outfile, "TRUE\n");
else
 fprintf(outfile, "FALSE\n");
if(_is_modified)
 fprintf(outfile, "TRUE\n");
else
 fprintf(outfile, "FALSE\n");
if(_is_state_variable)
 fprintf(outfile, "TRUE\n");
else
 fprintf(outfile, "FALSE\n");
if(ferror(outfile)) {
  clearerr(outfile);
  return FAILED;
 }
 else
  return SUCCEEDED;
}

/*********************************************************
* Draws the stream.  The actual curved line is contained
* in a spline, which draws itself.
*********************************************************/

void StreamObject::draw(DRAW_STYLE style) {
 GC draw_context;
 EXTERN_STATUS status;

 if(_is_deleted == FALSE) {
  if((_from_ptr != NULL) || (_to_ptr != NULL)) {
   if(style == SOLID)
    draw_context = _graphics_context;
   else
    if(style == ERASE)
     draw_context = _erase_context;
    else
     draw_context = _dotted_context;
   if(_from_ptr == NULL)
    status = FROM_EXTERNAL;
   else
    if(_to_ptr == NULL)
     status = TO_EXTERNAL;
    else
     status = NO_EXTERNAL;
   _arc.draw(this, draw_context, _graphics_context, style, status);
   draw_text(style);
   if((_from_ptr == NULL) || (_to_ptr == NULL)) {
```

```
    if(_from_ptr == NULL)
     _arc.set_extern_location(extern_location,
                      FROM_EXTERNAL);
    else
     if(_to_ptr == NULL)
      _arc.set_extern_location(extern_location,
                       TO_EXTERNAL);
    set_font(draw_context, _name_font);
    XDrawString(_display_ptr, _draw_window, draw_context,
            extern_location.x, extern_location.y,
            "External", strlen("External"));
    XDrawString(_display_ptr, *_drawing_area_pixmap,
            draw_context, extern_location.x,
            extern_location.y, "External",
            strlen("External"));
   }
  }
 }
}


/*******************************************************
* Returns TRUE if the given coordinates are close to the
* line, or within the boundaries of the text strings.
*******************************************************/

BOOLEAN StreamObject::hit(int x, int y) {

 if(_is_deleted)
  return FALSE;
 else {
  if(_name_ptr != NULL) {
   if(strlen(_name_ptr) != 0)
    if(((x >= _name_x) && (x <= (_name_x + _name_width))) &&
     (y >= _name_y - _name_height) && (y <= _name_y)) {
      _name_selected = TRUE;
      return TRUE;
     }
  }
  if(_latency_string_ptr != NULL) {
   if(strlen(_latency_string_ptr) != 0)
    if(((x >= _latency_x) &&
      (x <= (_latency_x + _latency_width))) &&
      (y >= _latency_y - _latency_height) &&
      (y <= _latency_y)) {
      _latency_selected = TRUE;
      return TRUE;
     }
  }
  return _arc.hit(x, y);
 }
}

/*******************************************************
 *  Returns TRUE if the given coordinates are close to the
 * line, or within the boundaries of the text strings.
 * Just like hit() except that it does not select anything
 *******************************************************/

BOOLEAN StreamObject::over(int x, int y) {                    //@4

 if(_is_deleted)
  return FALSE;
 else {
  if(_name_ptr != NULL) {
   if(strlen(_name_ptr) != 0)
    if(((x >= _name_x) && (x <= (_name_x + _name_width))) &&
     (y >= _name_y - _name_height) && (y <= _name_y)) {
      return TRUE;
     }
```

```
  }
  if(_latency_string_ptr != NULL) {
   if(strlen(_latency_string_ptr) != 0)
    if(((x >= _latency_x) &&
       (x <= (_latency_x + _latency_width))) &&
      (y >= _latency_y - _latency_height) &&
      (y <= _latency_y)) {
     return TRUE;
    }
  }
  return _arc.hit(x, y);
 }
}

/*********************************************************
*  Sets the _from_ptr and _to_ptr equal to the locations of
* the from and to operators.  Simplifies getting properties
* of the operators when needed.
*********************************************************/

void StreamObject::set_object_ptrs(GraphObjectList *parent) {

 if(_from != 0)
  _from_ptr = (OperatorObject *) parent->
              target_object(OPERATOROBJECT, _from);
 if(_to != 0)
  _to_ptr = (OperatorObject *) parent->
              target_object(OPERATOROBJECT, _to);
 _arc.set_object_ptrs(_from_ptr, _to_ptr);
 /* modified by M.T.Shing, 3/7/94 */
 /* set_default_text_location is only called if _name_x ==
NULL_VALUE */
 if (_name_x == NULL_VALUE)
        set_default_text_location();
}

/*********************************************************
*  Notifies the stream that the given object has been deleted.
* If it's an operator connected to the stream, the stream
* deletes itself.
*********************************************************/

void StreamObject::delete_notify(CLASS_DEF class_type,
                    int deleted_obj_id) {

 if((class_type == OPERATOROBJECT) &&
   ((_from == deleted_obj_id) || (_to == deleted_obj_id))) {
  erase();
 _is_deleted = TRUE;
 }
}

/*********************************************************
* replace the name method
*
*********************************************************/

void StreamObject::replace_name(char *new_name) {

 delete _name_ptr;
 _name_ptr = strdup(new_name);
 set_text_dimensions();
}

/*********************************************************
*  Notifies the stream that the given object has moved.  If
*  it's a connected object, the endpoints of the stream must
*  move.
*********************************************************/
```

```
void StreamObject::move_notify(CLASS_DEF object_type, int
object_id) {

 if(object_type == OPERATOROBJECT) {
  if((_from == object_id) || (_to == object_id))
    _arc.set_object_ptrs(_from_ptr, _to_ptr);
 }
 else
  if((object_type == STREAMOBJECT) && (object_id == _id))
    _arc.set_object_ptrs(_from_ptr, _to_ptr);
}

/*********************************************************
* select method
*********************************************************/

void StreamObject::select() {

 erase();
 _is_selected = TRUE;
 draw(SOLID);
}

/*********************************************************
* unselect the stream method
*********************************************************/

void StreamObject::unselect() {

 erase();
 _is_selected = FALSE;
 _name_selected = FALSE;
 _latency_selected = FALSE;
 draw(SOLID);
}
```

```
/*********************************************************
* Returns true if any of the stream's handles enclose the
* given coordinates.
*********************************************************/

BOOLEAN StreamObject::hit_handle(int x, int y) {
 EXTERN_STATUS status;

 if(_from_ptr == NULL)
  status = FROM_EXTERNAL;
 else
  if(_to_ptr == NULL)
   status = TO_EXTERNAL;
  else
   status = NO_EXTERNAL;
 return _arc.hit_handle(x, y, status);
}

/*********************************************************
* Returns true if any of the stream's handles enclose the
* given coordinates.
*********************************************************/

BOOLEAN StreamObject::over_handle(int x, int y) {          //@5
 EXTERN_STATUS status;

 if(_from_ptr == NULL)
  status = FROM_EXTERNAL;
 else
  if(_to_ptr == NULL)
   status = TO_EXTERNAL;
  else
   status = NO_EXTERNAL;
 return _arc.over_handle(x, y, status);
```

```
}

/*********************************************************
* set the default text location method
*********************************************************/

void StreamObject::set_default_text_location() {

 _arc.set_text_location(_name_x, _name_y, _latency_x,
                _latency_y);
}

/*********************************************************
* set the default name location
*********************************************************/

void StreamObject::set_default_name_location() {

 _arc.set_name_location(_name_x, _name_y);
}

/*********************************************************
* set the default latency location
*********************************************************/

void StreamObject::set_default_latency_location() {

 _arc.set_latency_location(_name_x, _name_y, _latency_x,
                _latency_y);
}

// Sets latency from the value entered in the properties
// dialog box.

//void StreamObject::set_constraint(int constraint, int units) {
//
// _latency = constraint;
// free(_latency_string_ptr);              // created with strdup (malloc)
// if(_latency == NULL_VALUE)
//  _latency_string_ptr = NULL;
// else
//  _latency_string_ptr = time_unit_str(_latency,units);          //@6
//
// set_text_dimensions();
//}

/*********************************************************
*  Sets values for the dimensions of the text strings.
*  _name_font and _met_font must be set before calling!
*********************************************************/

void StreamObject::set_text_dimensions() {

 if(_name_ptr == NULL) {
  _name_width = 0;
  _name_height = 0;
 }
 else {
  _name_width = font_text_width(_name_font, _name_ptr);
  _name_height = font_text_height(_name_font);
 }
 if(_latency != UNDEFINED_TIME) {
  _latency_width = font_text_width(_latency_font,
                _latency_string_ptr);
  _latency_height = font_text_height(_latency_font);
 }
 else {
  _latency_width = 0;
  _latency_height = 0;
 }
```

```
}

/*********************************************************
* set teh object font
*********************************************************/

void StreamObject::set_object_font(int font_id) {

 if(_name_selected)
   _name_font = font_id;
 else
  if(_latency_selected)
    _latency_font = font_id;
 set_text_dimensions();
}

/*********************************************************
*  Moves appropriate text strings the given amount.
*********************************************************/

void StreamObject::move_text(int x, int y) {

 erase_text();
 if(_name_selected) {
   _name_x += x;
   _name_y += y;
 }
 else
  if(_latency_selected) {
    _latency_x += x;
    _latency_y += y;
  }
 draw_text(SOLID);
}

/*********************************************************
* Notifies the stream that the given object has been
* undeleted.  If it's an object connected to the stream,
* the stream may need to undelete itself.  The undeleted object
* could be the stream itself.
*********************************************************/

void StreamObject::undelete_notify(CLASS_DEF class_type,
                    int deleted_obj_id) {

 if((class_type == STREAMOBJECT) &&
   (deleted_obj_id == _id)) {
  _is_deleted = FALSE;
  _is_modified = TRUE;
  _is_selected = FALSE;
  _name_selected = FALSE;
  _latency_selected = FALSE;
 }
 else
  if((class_type == OPERATOROBJECT) &&
    (_is_deleted == TRUE)) {
   if(_from == deleted_obj_id) {
    if(_to_ptr != NULL) {
     if(_to_ptr->is_deleted() == FALSE) {
      _is_deleted = FALSE;
      _is_modified = TRUE;
      _is_selected = FALSE;
      _name_selected = FALSE;
      _latency_selected = FALSE;
     }
    }
    else {
     _is_deleted = FALSE;
     _is_modified = TRUE;
     _is_selected = FALSE;
```

```
      _name_selected = FALSE;
      _latency_selected = FALSE;
     }
    }
    else
     if(_to == deleted_obj_id) {
      if(_from_ptr != NULL) {
       if(_from_ptr->is_deleted() == FALSE) {
        _is_deleted = FALSE;
        _is_modified = TRUE;
        _is_selected = FALSE;
        _name_selected = FALSE;
        _latency_selected = FALSE;
       }
      }
      else {
       _is_deleted = FALSE;
       _is_modified = TRUE;
       _is_selected = FALSE;
       _name_selected = FALSE;
       _latency_selected = FALSE;
      }
     }
   }
}

/*********************************************************
* The handles_drawn states are used to prevent the handles
* from accidentally redrawing and erasing a handle.
*********************************************************/

void StreamObject::reset_handles_drawn_state() {

 _st_handles_drawn = FALSE;
 _name_handles_drawn = FALSE;
 _latency_handles_drawn = FALSE;
 _arc.reset_handles_drawn();
}

/*********************************************************
* set the text height
*********************************************************/

int StreamObject::text_height() {

 if(_name_selected)
  return _name_height;
 else
  if(_latency_selected)
   return _latency_height;
  else
   return 0;
}

/*********************************************************
* set the text width
*********************************************************/
int StreamObject::text_width() {

 if(_name_selected)
  return _name_width;
 else
  if(_latency_selected)
   return _latency_width;
  else
   return 0;
}

/*********************************************************
*  Relocates the appropriate text strings at the given
```

```
* locations.
*******************************************************/

void StreamObject::text_locate(int x, int y) {

 if(_latency_selected) {
  _latency_x = x - _latency_width / 2;
  _latency_y = y + _latency_height / 2;
 }
 else
  if(_name_selected) {
   _name_x = x - _name_width / 2;
   _name_y = y + _name_height / 2;
  }
}
```

```
/* ********************************************************

Name:        stream_property_menu.H
Author:      Ken Moeller
Program:     graph_editor
Date Modified: 29 Aug 96
Remarks:

********************************************************
Modification History:

 @1   KBM    29 Aug 96
      New header created so that stream_property_menu.C
          functions could be split from graph_editor.C, since
          this file is getting way to big.

********************************************************
*/

#ifndef stream_property_menu_h
#define stream_property_menu_h 1

extern StreamObject st_being_updated;

extern BOOLEAN valid_num_string(char *num);

// stream property menu

void stream_name_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void stream_type_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void state_query_cb(Widget, XtPointer which,
                XtPointer cbs);

void initial_state_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void latency_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void latency_unit_cb(Widget, XtPointer which,
                XtPointer cbs);

void stream_ok_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void stream_cancel_cb(Widget w, XtPointer client_data,
                XtPointer cb_struct_ptr);

void stream_property_dialog(Widget parent_widget, int x, int y);

#endif
```

```
/* ******************************************************

Name:        stream_property_menu.C
Author:      Man-Tak Shing
Program:     graph_editor
Date Modified: 29 Aug 96
Remarks:

******************************************************
Modification History:

 Baseline taken from Man-Tak Shing

 @1   KBM    29 Aug 96
     Converted original program from a ge_interface.h
          implementation to a GraphObject impementation.  Also
          added checks for NULL values when passing data to
          XtVaSetValues, since it would crash the program without
          it.

          NOTE:  There are still problems with this code that
          are of the cut-and-paste type that were in the original.
          I still need to remove these.

******************************************************
*/

#include <stdlib.h>
#include <Xm/MainW.h>
#include <Xm/DrawingA.h>
#include <Xm/RowColumn.h>
#include <Xm/ScrolledW.h>
#include <Xm/DrawnB.h>
#include <Xm/Form.h>
#include <Xm/TextF.h>
#include <Xm/Text.h>
#include <stream.h>
#include <Xm/SelectioB.h>
#include <Xm/LabelG.h>
#include <Xm/ToggleBG.h>
#include <X11/Xatom.h>
#include <Xm/List.h>
#include <Xm/MessageB.h>

//#include "graph_editor.h"
#include "ge_defs.h"
#include "graph_object_list.H"
#include "graph_object.H"
//#include "operator_object.H"
//#include "spline_object.H"
#include "stream_object.H"

#include "ge_interface.h"
#include <Xm/Separator.h>
#include <Xm/PushB.h>

extern StreamObject st_being_updated;
extern GraphObject* selected_object_ptr;
extern GraphObjectList graphic_list;

extern BOOLEAN valid_num_string(char *num);

// stream property menu

void stream_name_cb(Widget w, XtPointer client_data,
                    XtPointer cb_struct_ptr) {

   Widget temp_w =  (Widget) client_data;
   char *text;
```

```
  text = XmTextFieldGetString(temp_w);
  if (text != NULL) {
    st_being_updated.replace_name(text);
        XtFree(text);
  }
}



void stream_type_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

  Widget temp_w =  (Widget) client_data;
  char *text;

  text = XmTextFieldGetString(temp_w);
  if (text != NULL) {
    st_being_updated.set_stream_type_name(strdup(text));
        XtFree(text);
  }
}


void state_query_cb(Widget, XtPointer which,
              XtPointer cbs) {
 XmToggleButtonCallbackStruct *state =
  (XmToggleButtonCallbackStruct *) cbs;

 if(state->set)
  st_being_updated.set_state_variable((int) which);
}


void initial_state_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {
```

```
  Widget temp_w =  (Widget) client_data;
  char *text;

  text = XmTextFieldGetString(temp_w);
  if (text != NULL) {
    st_being_updated.set_state_initial_value(strdup(text));
    XtFree(text);
  }
}

void latency_cb(Widget w, XtPointer client_data,
              XtPointer cb_struct_ptr) {

  Widget temp_w =  (Widget) client_data;
  char *text;

  text = XmTextFieldGetString(temp_w);

  if (valid_num_string(text))
    st_being_updated.set_latency(atoi(text));

  XtFree(text);
}

void latency_unit_cb(Widget, XtPointer which,
              XtPointer cbs) {
 XmToggleButtonCallbackStruct *state =
  (XmToggleButtonCallbackStruct *) cbs;

 if(state->set)
   st_being_updated.set_latency_unit((int) which);
}
```

```
void stream_ok_cb(Widget w, XtPointer client_data,
                  XtPointer cb_struct_ptr) {

 st_being_updated.copy_back((StreamObject*) selected_object_ptr);
 graphic_list.draw();
 XtUnmanageChild((Widget)client_data);
}


void stream_cancel_cb(Widget w, XtPointer client_data,
                  XtPointer cb_struct_ptr) {

 XtUnmanageChild((Widget)client_data);
}



void stream_property_dialog(Widget parent_widget, int x, int y) {

  Widget dialog, stream_name, radio_box, unit_box,
       stream_type, latency, initial_state,
       ok_button, cancel_button;
  XmString button1, button2, button3, button4, button5,
        t, init_value;
  Arg args[10];
  char *prompt, buffer[INPUT_LINE_SIZE];
  int initial_button, ac;


   ac = 0;
   XtSetArg(args[ac], XmNheight, 450); ac++;
   XtSetArg(args[ac], XmNwidth, 600); ac++;
   XtSetArg(args[ac], XmNautoUnmanage, False); ac++;
   t = XmStringCreateSimple ("Stream Property");
   XtSetArg(args[ac], XmNdialogTitle, t); ac++;

   dialog = XmCreateBulletinBoardDialog(parent_widget,
"Stream_Property", args, ac);

   XmStringFree(t);


   // create the stream label & text field
   prompt = strdup("Stream Name:");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                  XmNx, 10,
                  XmNy, 10,
                  XmNalignment, XmALIGNMENT_BEGINNING,
                  NULL);

   stream_name =  XtVaCreateManagedWidget("stream_name",
                  xmTextFieldWidgetClass, dialog,
                  XmNx, 150,
                  XmNy, 10,
                  NULL);
   XtAddCallback(stream_name, XmNactivateCallback,
stream_name_cb, (XtPointer)stream_name);
   if (st_being_updated.name() != NULL)
    XtVaSetValues(stream_name, XmNvalue, st_being_updated.name(),
NULL);
   free(prompt);

   // create the stream type label and text field
   prompt = strdup("Stream Type:");
   XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                  XmNx, 10,
                  XmNy, 50,
                  XmNalignment, XmALIGNMENT_BEGINNING,
                  NULL);
```

```
  stream_type =  XtVaCreateManagedWidget("stream_type",
                 xmTextFieldWidgetClass, dialog,
                 XmNx, 150,
                 XmNy, 50,
                 NULL);
  XtAddCallback(stream_type, XmNactivateCallback, stream_type_cb,
(XtPointer)stream_type);
  if (st_being_updated.stream_type_name() != NULL)
   XtVaSetValues(stream_type, XmNvalue,
st_being_updated.stream_type_name(), NULL);
  free(prompt);

  // create a state_stream query label
  prompt = strdup("Is a state stream?");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                 XmNx, 10,
                 XmNy, 90,
                 XmNalignment, XmALIGNMENT_BEGINNING,
                 NULL);
  free(prompt);


  // create radio-box for state stream
  button1 = XmStringCreateSimple("No");
  button2 = XmStringCreateSimple("Yes");
  if (st_being_updated.is_state_variable())
        initial_button = 1;
  else
        initial_button = 0;
  radio_box = XmVaCreateSimpleRadioBox(dialog, "radio_box",
                 initial_button, state_query_cb,
                 XmVaRADIOBUTTON, button1, NULL, NULL,
NULL,
                 XmVaRADIOBUTTON, button2, NULL, NULL,
NULL,
                 XmNx, 150,
                 XmNy, 90,
                 XmNorientation, XmHORIZONTAL,
                 NULL);
  XmStringFree(button1);
  XmStringFree(button2);
  XtManageChild(radio_box);

  // create the initial state value label and text field
  prompt = strdup("Initial State Value:");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                 XmNx, 10,
                 XmNy, 130,
                 XmNalignment, XmALIGNMENT_BEGINNING,
                 NULL);
  initial_state =  XtVaCreateManagedWidget("initial_state",
                 xmTextFieldWidgetClass, dialog,
                 XmNx, 150,
                 XmNy, 130,
                 NULL);
  XtAddCallback(initial_state, XmNactivateCallback, initial_state_cb,
(XtPointer)initial_state);
  if (st_being_updated.is_state_variable() &&
        st_being_updated.state_initial_value() != NULL)
     XtVaSetValues(initial_state,
                 XmNvalue, st_being_updated.state_initial_value(),
NULL);
  free(prompt);

  // create the latency label and text field
  prompt = strdup("Latency:");
  XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                 XmNx, 10,
                 XmNy, 170,
                 XmNalignment, XmALIGNMENT_BEGINNING,
```

```
                NULL);
    latency =  XtVaCreateManagedWidget("latency",
                xmTextFieldWidgetClass, dialog,
                XmNx, 150,
                XmNy, 170,
                NULL);
    XtAddCallback(latency, XmNactivateCallback, latency_cb,
(XtPointer)latency);
    if (st_being_updated.latency() != UNDEFINED_TIME) {
        sprintf(buffer, "%d", st_being_updated.latency());
        XtVaSetValues(latency, XmNvalue, buffer, NULL);
    }
    free(prompt);


    // create a latency_unit label
    prompt = strdup("Latency unit:");
    XtVaCreateManagedWidget(prompt, xmLabelGadgetClass, dialog,
                XmNx, 10,
                XmNy, 210,
                XmNalignment, XmALIGNMENT_BEGINNING,
                NULL);
    free(prompt);


    // create the radio-box for the latency_units
    button1 = XmStringCreateSimple("micro");
    button2 = XmStringCreateSimple("ms");
    button3 = XmStringCreateSimple("sec");
    button4 = XmStringCreateSimple("min");
    button5 = XmStringCreateSimple("hour");
    initial_button = 1;
    if (st_being_updated.latency() != UNDEFINED_TIME) {
        initial_button = st_being_updated.latency_unit();
    }
```

```
    unit_box = XmVaCreateSimpleRadioBox(dialog, "unit_box",
                initial_button, latency_unit_cb,
                XmVaRADIOBUTTON, button1, NULL, NULL,
NULL,
                XmVaRADIOBUTTON, button2, NULL, NULL,
NULL,
                XmVaRADIOBUTTON, button3, NULL, NULL,
NULL,
                XmVaRADIOBUTTON, button4, NULL, NULL,
NULL,
                XmVaRADIOBUTTON, button5, NULL, NULL,
NULL,
                XmNx, 10,
                XmNy, 230,
                XmNorientation, XmHORIZONTAL,
                NULL);
    XmStringFree(button1);
    XmStringFree(button2);
    XmStringFree(button3);
    XmStringFree(button4);
    XmStringFree(button5);
    XtManageChild(unit_box);

    // create the OK and CANCEL buttons
    ac = 0;
    XtSetArg(args[ac], XmNx, 20); ac++;
    XtSetArg(args[ac], XmNy, 270); ac++;
    ok_button = XmCreatePushButton(dialog, "   OK   ", args, ac);
    XtManageChild(ok_button);

    ac = 0;
    XtSetArg(args[ac], XmNx, 230); ac++;
    XtSetArg(args[ac], XmNy, 270); ac++;
    cancel_button = XmCreatePushButton(dialog, " CANCEL ", args, ac);
    XtManageChild(cancel_button);
```

```
   XtAddCallback(ok_button, XmNactivateCallback, stream_ok_cb,
(XtPointer) dialog);

   XtAddCallback(cancel_button, XmNactivateCallback,
stream_cancel_cb, (XtPointer) dialog);

   XtManageChild(dialog);

   XtVaSetValues(dialog, XmNx, x, XmNy, y, NULL);

   XtPopup(XtParent(dialog), XtGrabNone);

   XmProcessTraversal(parent_widget, XmTRAVERSE_CURRENT);
}
```

```
/* Written by Dan Heller and Paula Ferguson.
 * Copyright 1994, O'Reilly & Associates, Inc.
 * Permission to use, copy, and modify this program without
 * restriction is hereby granted, as long as this copyright
 * notice appears in each copy of the program source code.
 * This program is freely distributable without licensing fees and
 * is provided without guarantee or warrantee expressed or implied.
 * This program is -not- in the public domain.
 */

/* ask_user.c -- the user is presented with two pushbuttons.
 * The first creates a file (/tmp/foo) and the second removes it.
 * In each case, a dialog pops up asking for verification of the action.
 *
 * This program is intended to demonstrate an advanced implementation
 * of the AskUser() function.  This time, the function is passed the
 * strings to use for the OK button and the Cancel button as well as
 * the button to use as the default value.
 */

#ifndef windows_h
#define windows_h 1

#include <Xm/DialogS.h>
#include <Xm/SelectioB.h>
#include <Xm/RowColumn.h>
#include <Xm/MessageB.h>
#include <Xm/PushB.h>
#include "ge_utilities.H"

#define YES    1
#define NO     2
#define CANCEL 3
#define OK     4

#define BTN1   1
#define BTN2   2
#define BTN3   3

/* Generalize the question/answer process by creating a data structure
 * that has the necessary labels, questions and everything needed to
 * execute a command.
 */
typedef struct {
  char *label;    // label for pushbutton used to invoke cmd
  char *question; // question for dialog box to confirm cmd
  char *btn1;     // label for first button
  char *btn2;     // label for second button
  char *btn3;     // label for third button
  int  dflt;      // which should be the default answer
} Quest_Script;

typedef struct {
  char *printer;    // Name of printer
  int  answer;      // Command that was requested (OK, Cancel)
} PrintCmd;


int AskUser(XtAppContext app, Widget parent, Quest_Script scr);
void response(Widget widget, XtPointer client_data, XtPointer
call_data);

void AskPrint(XtAppContext app, Widget parent, PrintCmd *prn);
void PrintResponse(Widget widget, XtPointer client_data, XtPointer
call_data);

#endif
```

```
#include "windows.H"
#include "graph_editor.h"

#ifdef GE_DEBUG
#include <stream.h>
#endif

/* Written by Dan Heller and Paula Ferguson.
 * Copyright 1994, O'Reilly & Associates, Inc.
 * Permission to use, copy, and modify this program without
 * restriction is hereby granted, as long as this copyright
 * notice appears in each copy of the program source code.
 * This program is freely distributable without licensing fees and
 * is provided without guarantee or warrantee expressed or implied.
 * This program is -not- in the public domain.
 */


/*********************************************************
* AskUser() -- a generalized routine that asks the user a question
* and returns a response.
*********************************************************/
int AskUser(XtAppContext app, Widget parent, Quest_Script scr)
{
   static Widget dialog;          // static to avoid multiple creation
   XmString text, LblBTN1, LblBTN2, LblBTN3;
   static int answer;
   int default_ans;

   if (!dialog) {
      dialog = XmCreateQuestionDialog(parent, "dialog", NULL, 0);
      XtVaSetValues(dialog,
         XmNdialogStyle,
XmDIALOG_FULL_APPLICATION_MODAL,
         NULL);
      XtAddCallback(dialog, XmNokCallback,     response, &answer);
      XtAddCallback(dialog, XmNcancelCallback, response, &answer);
      XtAddCallback(dialog, XmNhelpCallback,   response, &answer);
   }

   answer = 0;
   text   = XmStringCreateSimple(scr.question);
   LblBTN1 = XmStringCreateSimple(scr.btn1);
   LblBTN2 = XmStringCreateSimple(scr.btn2);
   LblBTN3 = XmStringCreateSimple(scr.btn3);
   if (scr.dflt == BTN1)
    default_ans = XmDIALOG_OK_BUTTON;
   else if (scr.dflt == BTN2)
    default_ans = XmDIALOG_CANCEL_BUTTON;
   else
    default_ans = XmDIALOG_HELP_BUTTON;

   XtVaSetValues(dialog,
      XmNmessageString,     text,
      XmNokLabelString,     LblBTN1,
      XmNcancelLabelString, LblBTN2,
          XmNhelpLabelString,   LblBTN3,
      XmNdefaultButtonType, default_ans,
      NULL);

   XmStringFree(text);
   XmStringFree(LblBTN1);
   XmStringFree(LblBTN2);
   XmStringFree(LblBTN3);

   XtManageChild(dialog);
   XtPopup(XtParent(dialog), XtGrabNone);

   while (answer == 0)
```

```
    XtAppProcessEvent(app, XtIMAll);

  XtPopdown(XtParent(dialog));
  /* make sure the dialog goes away before returning. Sync with server
   * and update the display.
   */
  XSync(XtDisplay(dialog), 0);
  XmUpdateDisplay(parent);

  return answer;
}



/*******************************************************
* response() --The user made some sort of response to the
* question posed in AskUser(). Set the answer (client_data)
* accordingly.
*******************************************************/
void response(Widget widget, XtPointer client_data, XtPointer call_data)
{
  int *answer = (int *) client_data;
  XmAnyCallbackStruct *cbs = (XmAnyCallbackStruct *) call_data;

  if (cbs->reason == XmCR_OK)
    *answer = BTN1;
  else if (cbs->reason == XmCR_CANCEL)
    *answer = BTN2;
  else
    *answer = BTN3;
}



/*******************************************************
* AskPrint() -- Provide the user with a printer popup with the option
* to enter a printer name. Returns both answer and printer in PrintCmd.
*******************************************************/
void AskPrint(XtAppContext app, Widget parent, PrintCmd *prn)
{
  static Widget dialog;          // static to avoid multiple creation
  static PrintCmd print_cmd;

  print_cmd.printer = make_str(prn->printer);

  XmString label   = XmStringCreateSimple("Printer Name:");
  XmString prn_str = XmStringCreateSimple(print_cmd.printer);

  if (!dialog) {
    dialog = XmCreatePromptDialog(parent, "Printer", NULL, 0);
    XtVaSetValues(dialog,
      XmNdialogStyle,
XmDIALOG_FULL_APPLICATION_MODAL,
      NULL);

    XtAddCallback(dialog, XmNokCallback,    PrintResponse,
&print_cmd);
    XtAddCallback(dialog, XmNcancelCallback, PrintResponse,
&print_cmd);
  }

  XtVaSetValues(dialog,
    XmNselectionLabelString,    label,
    XmNdefaultButtonType,       XmDIALOG_OK_BUTTON,
      XmNtextString,            prn_str,
    NULL);

  // No help is available
  XtSetSensitive(XmSelectionBoxGetChild(dialog,
XmDIALOG_HELP_BUTTON), False);
```

```
    XmStringFree(label);
    XmStringFree(prn_str);

    XtManageChild(dialog);
    XtPopup(XtParent(dialog), XtGrabNone);

    print_cmd.answer = 0;
    while (print_cmd.answer == 0)
      XtAppProcessEvent(app, XtIMAll);

    XtPopdown(XtParent(dialog));
    /* make sure the dialog goes away before returning. Sync with server
     * and update the display.
     */
    XSync(XtDisplay(dialog), 0);
    XmUpdateDisplay(parent);

    prn->printer = print_cmd.printer;
    prn->answer  = print_cmd.answer;
}

/*********************************************************
* PrintResponse() --The user made some sort of response to the
* question posed in AskUser().  Set the answer (client_data)
* accordingly.
*********************************************************/
void PrintResponse(Widget widget, XtPointer client_data, XtPointer
call_data)
{
    PrintCmd *print_cmd = (PrintCmd *) client_data;
    char *prn_name;

    XmSelectionBoxCallbackStruct *cbs =
(XmSelectionBoxCallbackStruct *) call_data;

    if (cbs->reason == XmCR_OK) {
      XmStringGetLtoR(cbs->value,
XmSTRING_DEFAULT_CHARSET, &prn_name);
      print_cmd->printer = make_str(prn_name);
      print_cmd->answer = OK;
    }
    else if (cbs->reason == XmCR_CANCEL)
      print_cmd->answer = CANCEL;
    else
      print_cmd->answer = CANCEL;

    XtFree(prn_name);
}
```

```
/* This is a simple driver program to simulate the SDE.  All code is
   writen in C and not C++ since the SDE will be writen in C.      */


/* Note:  This program will have to be modified to handle some basic
        file operations with GRAPH_DESC_NODE for testing of to_psdl
        and from_psdl operations.  However, until these two methods
        are updated, I have not yet implemented any file operations. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "ge_interface.h"
#include "graph_editor.h"

int main(unsigned int argc, char ** argv) {

  GRAPH_DESC_NODE* current_graph;
  ACTION_NODE*     next_action;
  char             c;

  current_graph = (GRAPH_DESC)
malloc(sizeof(GRAPH_DESC_NODE));
  next_action   = (ACTION) malloc(sizeof(ACTION_NODE));

  /* Fill data for call.  This should be modified when to_psdl and
     from_psdl is implemented.  However, for now this puts something
     in the data structure.  These operators are taken from the C3I
     CAPS program.                                          */
  current_graph->edited_op_name = strdup("EDITED_OP_NAME");
  current_graph->operator_list  = NULL;
  current_graph->stream_list    = NULL;

  /* Fill in some static data to start with */
  /*  printf("Use static data: "); */
  /*  c = getchar(); */
  c = 'Y';
  if (c == 'y' || c == 'Y') {
    current_graph->parent_op_name = strdup("PARENT_OP_NAME");
    current_graph->edited_op_spec = strdup("EDITED_OP_SPEC");
    current_graph->graph_informal_desc =
strdup("GRAPH_INFORMAL_DESC");
    current_graph->timer_list = (ID_LIST) malloc(sizeof(ID_NODE));
    current_graph->timer_list->ID = strdup("timer A");
    current_graph->timer_list->NEXT = (ID_LIST)
malloc(sizeof(ID_NODE));
    current_graph->timer_list->NEXT->ID = strdup("timer B");
    current_graph->timer_list->NEXT->NEXT = NULL;
  }
  else {
    current_graph->parent_op_name     = NULL;
    current_graph->edited_op_spec     = NULL;
    current_graph->graph_informal_desc = NULL;
    current_graph->timer_list         = NULL;
  }

  /* Dummy data to verify that next_action was updated upon exit.    */
  next_action->save = -99;
  next_action->next_op = strdup("NEXT_OP");

  /* Loop until no longer reinvoked.                                 */
  do {
    edit_graph(current_graph, next_action);  /* This is the actual call */

    if (next_action->save == TRUE)          /* Print what edit_graph   */
      printf("SDE: Save psdl");             /* returned upon exit.     */
    else
      printf("SDE: Abandon psdl");
    if (next_action->reinvoke == TRUE) {
```

```
    printf(", reinvoke with operator %s", next_action->next_op);
    current_graph->edited_op_name = next_action->next_op; }
   printf("\n");
  } while (next_action->reinvoke);

  printf("SDE: Exit\n");

  /* Pause so that you can see if window was killed */
  /*  printf("Type any character: "); */   /* Working, not needed any more
*/
  /*  c = getchar();              */

  free((char *) current_graph);
  free((char *) next_action);

  return 0;
}
```

LIST OF REFERENCES

1. Pressman, R. S., *Software Engineering: A Practitioner's Approach*, San Francisco, CA: Mc Graw-Hill, 1987.

2. Luqi, "Computer-Aided Software Prototyping," *IEEE Computer*, Vol 24, No.9, September 1991.

3. Stankovic, J. A., Ramamritham, K., *Hard Real-Time Systems*, Los Alamitos, CA, IEEE Computer Society Press, 1988.

4. Goldsack, S., Finkelstein, A., "Requirements Engineering For Real-time Systems," *Software Engineering Journal*, May, 1991.

5. Berzins, V., Luqi, Shing, M., Chmura, L., *Managing Real-time Software Projects: Problems and issues*, Technical report NPSCS-92-016, Monterey, CA., December 1992.

6. Berzins, V., Luqi, *Software Engineering with Abstractions,* Menlo Park, CA: Addison-Wesley Publishing Company, 1991.

7. Luqi, Berzins, V., Yeh, R., "A Prototyping Language for Real-time Software," *IEEE Transactions on Software Engineering*, October 1988.

8. Cooke, R. P., "Technology Transfer of the Computer-Aided Prototyping System," M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1996.

9. Luqi, Ketabchi, M., *A Computer-Aided Prototyping System*, IEEE Software, 1988.

10. Brockett, *A CAPS Tutorial,* Naval Postgraduate School, Monterey, CA, 1995.

11. Bokhan, S. H., "The Linux Operating System," IEEE August 1995.

12. Silberschatz A., Galvin, P. B., *Operating System Concepts*, Menlo Park, CA: Addison-Wesley Publishing Company, 1994.

13. Welsh, M., Kaufman, L., *Running Linux*, Sebastopol, CA: O'Reilly & Associates, Inc., 1995.

14. Infomagic, Linux 3.0 Developers Resource, *http://www.infomagic.com.*

15. Comer, D. E., *Internetworking with TCP/IP*, Englewoods Cliffs, NJ : Prentice Hall,1995.

16. A Linux Homepage, *Common Problems,* http://www.cica.es/Linux_doc/Ethernet-HOWTO.html.

17. Tackett, J. Jr., Gunter, D., *Using Linux*, Indianapolis, IN: Que Corporation, 1996.

18. Gildea, S., *Xwindows System, Version 11, Release 6, Release Notes*, Cambridge, MA: X-Consortium, 1994.

19. Irwin, D., "Re-engineering of the Computer-Aided Prototyping System for Portability," M.S. Thesis, Naval Postgraduate School, Monterey, CA, September 1996.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, California 93943-5101	2
3.	Professor Luqi Code CS/Lq Naval Postgraduate School Monterey, California 93943-5101	2
4.	Professor Valdis Berzins Code CS/Be Naval Postgraduate School Monterey, California 93943-5101	2
5.	Ltjg. Recep Erdinc Yetkin Íl Jandarma Alay Komutanliĝi Kayseri Turkey	2
6.	Lt. Sotero Enriquez 16949 Wing Lane Valinda, Ca 93940	2
7.	Deniz Harp Okulu KomutanliĝI Kutuphanesi Tuzla/Istanbul Turkey	2

8. Deniz Kuvvetleri KomutanliğI 1
Personel Tedarik ve Yetistirme Daire Bsk.
Bakanliklar/Ankara 06100
Turkey