ornl

OAK RIDGE
NATIONAL
LABORATORY

MARTIN MARIETTA

ORNL/TM-12811

Experiences in Effective Use of Tcl/Tk

R. W. Lee

MANAGED BY
MARTIN MARIETTA ENERGY SYSTEMS, INC.
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

ORNL/TM-12811

Computer Science and Mathematics Division

EXPERIENCES IN EFFECTIVE USE OF TCL/TK

R. W. Lee

DATE PUBLISHED -- June 1995

Prepared by the
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831
managed by
MARTIN MARIETTA ENERGY SYSTEMS, INC.
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-84OR21400

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

Tcl and Tk were developed by John Ousterhout of the University of California at Berkeley. Tcl Distributed Programming (Tcl-DP) was developed by Lawrence A. Rowe of the University of California at Berkeley and Brian C. Smith of Cornell University.

ABSTRACT

Tcl/Tk (Tool Command Language and Tool Kit, pronounced "tickle tee-kay") is a scripting language supporting Motif™ style X Window interfaces. It is extendible, allowing developers to embed additional functionality as commands in the language. However, the power and flexibility of the system leads to many variations or possibilities in its usage. We describe effective methods for taking advantage of Tcl/Tk to increase productivity and enhance the flexibility and adaptability of applications: writing simple Tcl/Tk scripts, extending the Tcl/Tk widget set, wrapping Tcl commands around existing classes and functions, and building Tcl/Tk and 3GL coprocesses. Examples are presented from working applications.

1. INTRODUCTION

Imperative in application development today is a graphical user interface (GUI). For the most part, the UNIX world has standardized on the network-based X Window System. Sun's XView (Open Look) and OSF/Motif™ have battled to be the standard toolkit and widget set, with Motif™ emerging as the apparent victor. Regardless of the toolkit used (or even if programmed at the X Window level using Xlib), much code is necessary to construct windows/widgets, set resource values, establish desired behavior, and respond to input events in an application. Many "builder" tools are available to support interactive construction of interfaces, reducing the time and effort required for some development. However, event handlers and callback routines must be embedded in the generated code. Further, such tools do little to decrease the time spent in the *edit-compile-link* cycle necessary for each source code modification. Run-time resource managers, such as the Motif™ UIL (user interface language) offer the advantage of separation of form and function and dynamic binding of interfaces, but they too require compilation.

Often, the bottleneck in application development time is the edit-compile-link iteration necessary for even the most cosmetic of changes. This cost is particularly high in a rapid prototype environment [4]. Other factors adversely affecting development of GUI interfaces include the complexity of the toolkit libraries and disproportionately large amount of code dedicated to interface construction as opposed to behavior. For example, even a simple dialog window requires code to construct push button, radio button, text field, option menu, and other widgets. Then, callback routines must be written to respond to the push buttons at least and possibly other widgets as well.

In an effort to reduce the code dedicated to GUI, scripting tools have emerged. HyperCard™ has been available on the Macintosh for some time now. More recently, Microsoft Visual Basic has emerged as a similar tool for the DOS/Windows environment. In the X Window UNIX community, Tcl/Tk has become the most popular and most used GUI scripting environment, largely due to its extendibility. These scripting tools represent a new paradigm for development of GUI applications, for they attempt to separate an application's form from its function by isolating the construction of interfaces from 3GL source. The interface is maintained in an easy-to-modify, interpreted script.

To the application developer, Tcl/Tk is just one of many alternatives for constructing interfaces. In this report, experience developing working prototype systems with Tcl/Tk is described. Four aspects of Tcl/Tk development are explored:

- programming in the Tcl scripting language,
- extending the Tk widget set,
- constructing Tcl interfaces to 3GL classes and functions, and
- building Tcl and 3GL coprocesses.

Hopefully, this information will benefit someone considering Tcl/Tk as a potential development tool. The reader's familiarity with Tcl/Tk and C++ is assumed.

2. TCL/TK PRIMER

The best introduction to Tcl/Tk is found in Ousterhout's book [7]. The following excerpt is taken from page 1.

> *Together they [Tcl and Tk] provide a programming system for developing and using graphical user interface (GUI) applications. Tcl stands for "tool command language" and is pronounced "tickle"; it is a simple scripting language for controlling and extending applications. It provides generic programming facilities that are useful for a variety of applications, such as variables and loops and procedures. Furthermore, Tcl is embeddable; its interpreter is implemented as a library of C procedures that can easily be incorporated into applications, and each application can extend the core Tcl features with additional commands specific to that application.*
>
> *One of the most useful extensions to Tcl is Tk. It is a toolkit for the X Window System, and its name is pronounced "tee-kay". Tk extends the core Tcl facilities with additional commands for building user interfaces, so that you can construct Motif user interfaces by writing Tcl scripts instead of C code. Like Tcl, Tk is implemented as a library of C procedures so it too can be used in many different applications. Individual applications can also extend the base Tk features with new user-interface widgets and geometry managers written in C.*

Tk widgets provide Motif™ behavior but are built solely upon Xlib, thereby allowing Tcl/Tk to be ported to any UNIX-based X Window environment. That is, Tk widgets are not Motif™ widgets.[1]

Tcl/Tk is exhaustively described in Ousterhout's book, available in PostScript in four parts from the anonymous FTP site `sprite.berkeley.edu`. Here we provide a very brief and general description of Tcl/Tk and introduce the concepts discussed in the remainder of the paper.

2.1 THE SCRIPTING LANGUAGE

Essentially, Tcl and Tk are libraries for building shell interpreters. Tcl statements are read and processed by an interpreter program. The distributions provide the means for building default shells. Unextended, the Tcl shell is `tclsh`, which supports the basic language constructs (e.g., variable manipulation, iteration, control) but no GUI capabilities. The unextended Tcl/Tk shell is `wish` (windowing interactive shell), which supports many commands for creating widget objects and controlling windows. A great many extensions to Tcl/Tk have been developed and made available via the internet. These extensions provide the components (e.g., source code, object modules, and/or libraries) necessary to build new shell interpreters with the ability to process new commands and features. Additional sources

[1] The tclMotif package available from harbor.ecn.purdue.edu provides an interface to Motif widgets.

of information are the harbor.ecn.purdue.edu FTP site and the comp.lang.tcl news group.

The underlying paradigm for the Tcl shell interpreter is the UNIX command line. In fact, each statement in the language is a command line with options and arguments specified in the familiar UNIX shell format (i.e., options identified with a preceding dash). It is important for the developer to understand that even control statements such as if-elseif-else are single-line commands. Commands may be combined in lists to form procedures, which are defined with optional arguments and default values.

All variable values are stored as strings, making arithmetic computation slow, since conversions to and from numeric formats are necessary. Variable values are stored in a hash table indexed by variable name and can be scalar, lists, or arrays. Lists, stored as strings with delimiters between values, are the vehicle for most of the data manipulation as well as code interpretation in the language. Arrays are associative, implemented as individual hash tables with indexes always treated as strings. These mechanisms can be used in very powerful and effective ways from the viewpoint of the developer. However, one must always be mindful of the potential performance problems inherent in processing null-terminated C strings.

In and of itself, tclsh is as effective a scripting language as any of the UNIX shells, and in some ways awk or perl. Figure 2-1 shows an example script for computing factorials for a set of numbers and illustrates the compactness of the language.

```
% proc fact { n } \
{
for { set x 1 } { $n > 1 } { incr n -1 } \
  { set x [expr $x * $n] };
return $x;
}
% foreach n { 0 1 6 9 11 14 } \
{ puts stdout [format "%2d %d" $n [fact $n]] }
 0 1
 1 1
 6 720
 9 362880
11 39916800
14 1278945280
```

Figure 2-1 Tcl Script to Compute Factorials

Tk's utility becomes immediately clear in the simple example of Figure 2-2, which produces a window with two buttons. As in Motif™, a left mouse button release activates Tk buttons. In the example, activation of the buttons results in new xterm windows with green and blue backgrounds, respectively. Figure 2-3 shows further commands to change the appearance of the buttons.

```
% wm withdraw .
% button .one -text "Green" \
  -command { exec xterm -bg DarkGreen -fg white & }
% button .two -text "Blue" \
  -command { exec xterm -bg MidnightBlue -fg white & }
% pack .one .two -side left
% wm deiconify .
```

Figure 2-2 Simple Tcl/Tk Script For Two Buttons

```
% .one configure \
  -background DarkGreen -activebackground gray75 \
  -foreground yellow -activeforeground #005f00 \
  -font 9x15bold
% .two configure -background MidnightBlue -activebackground gray75
```

Figure 2-3 Additional Tcl/Tk Commands to Modify Button Appearance

```
int
TclMath::ProcessRandomCommand(
    ClientData          data,
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  char  buffer[ 64 ];

if ( ! _tclRandInitFlag )
  {
  srand48( time( NULL ) );
  _tclRandInitFlag = 1;
  }

sprintf( buffer, "%lg", drand48() );
Tcl_SetResult( interp, buffer, TCL_VOLATILE );
return  TCL_OK;
} // TclMath::ProcessRandomCommand

...

void
TclMath::RegisterCommands( Tcl_Interp * interp )
{
Tcl_CreateCommand(
    interp,
    "random",
    (Tcl_CmdProc *) ProcessRandomCommand,
    NULL,
    NULL
    );
...
} // TclMath::RegisterCommands
```

Figure 2-4 Example Tcl Functional Extension

2.2 TCL EXTENSIONS

The real power of Tcl is its extendibility. Developers extend it by compiling and linking a new shell interpreter with procedures "registered" to be called for new commands. Tk is such an extension. There are two modes for adding new functionality: functional, and object-based. In the functional mode, new commands correspond to individual functions. Refer to Figure 2-4 for code snippets illustrating a simple C++ implementation of a "random" command. Note that the methods in the example must be declared `static` in the class declaration.

```
int
Tcl_AppInit( Tcl_Interp * interp )
{
  int              status = TCL_OK;
  Tk_Window        main_window = Tk_MainWindow( interp );

        // Register Local C++ Classes and Packages
TkThing::RegisterCommand( main_window, interp );
TclMath::RegisterCommands( interp );

        // Initialize Other Packages
if (
    Tcl_Init( interp ) == TCL_ERROR ||
    Tk_Init( interp ) == TCL_ERROR
    )
  status = TCL_ERROR;

else
  tcl_RcFileName = "~/.wishrc";

return status;
}
```

Figure 2-5 User-Defined `Tcl_AppInit()`

Developers hook into Tcl/Tk and extension libraries by implementing a function named `Tcl_AppInit()` in which necessary initialization (e.g., registering commands) is performed. A template version is provided with the Tcl distribution. The `main()` routine is linked from the Tcl or Tk library. Figure 2-5 illustrates a user-defined `Tcl_AppInit()`. Note the standard initialization of the Tcl and Tk libraries via `Tcl_Init()` and `Tk_Init()`, respectively. Were we to produce an extension available to the internet, we would generate such a function.

Object-based extensions require at least two functions: one to implement a "factory" or object creation command, and another to implement the individual object command.[2] This is the paradigm of the Tk widgets. In Figure 2-2, the objects named `.one` and `.two` are created with the factory `button` command. Figure 2-3 shows use of the commands `.one` and `.two`. Within the factory command implementation, the command for the newly created object must be registered (via `Tcl_CreateCommand()`). Similarly, when the object is deleted, its command must be unregistered (via `Tcl_DeleteCommand()`).

[2] The concept of factory and instance methods is described by Cox [3].

2.3 EXTENDING THE TK WIDGET SET

As stated earlier, Tk extends Tcl with a collection of widgets and window management functions. Developers may add widgets using the object-based paradigm and meeting the requirements for widgets. A widget implementation must include procedures to:

- configure the widget,
- handle X Window events,
- display the widget,
- destroy the widget,
- respond to widget commands,
- create widgets.

The last two are the object and factory commands, respectively. Configuration involves setting and retrieving attribute values. Tk supplies constructs for automatic parsing as well as storage and retrieval of attributes as fields in a data structure. Example uses of these constructs are presented in Section 4.

One of the most powerful of the Tk widgets is the `canvas`. Within canvases, one creates and manipulates graphical canvas items of various types. Figure 2-6 shows a Tcl/Tk script which creates a rectangle within a canvas and defines an event handler for mouse motion with button one pressed. The requirements for additional canvas item types are more stringent. Functions for the item type must be provided to:

```
% canvas .can -width 200 -height 200;
% pack .can -side top;
% .can create rectangle 50 50 150 150 \
  -tags myrect \
  -fill red \
  -outline white \
  -width 2;
% .can bind myrect <B1-Motion> \
{ .can coords myrect %x %y [expr %x+100] [expr %y+100] }
```

Figure 2-6 Canvas Item Creation and Manipulation

- create an item instance,
- configure,
- store and retrieve coordinates,
- delete,
- draw,
- compute the distance to a point,
- determine overlap with a rectangular region,
- generate a PostScript representation,
- scale,
- translate.

Items processing text must supply text handling functions as well.

2.4 INTERPROCESS COMMUNICATION

There are two communication mechanisms available within Tcl/Tk. First, separate Tcl/Tk processes connected to the same X Window server may communicate using X events via the `send` command. Second, Tcl has built-in facilities for processing files and command pipes, allowing a Tcl/Tk application to operate as a coprocess. Both of these mechanisms allow distributed Tcl/Tk implementations. In addition, the Tcl-DP (distributed processing) extension, also from Berkeley, provides commands for socket communication and registration of procedure handlers. Of course, the developer can always build communication facilities in C/C++ and implement Tcl commands to manipulate them.

2.5 COMPARISON WITH OTHER TOOLS

Tcl/Tk is a unique user interface development tool. Not only is it interpreted, requiring no compilation, it offers powerful extensibility at the command or object level. Builder tools take a different approach. They generally produce source code to build interfaces defined with a very powerful point-and-click layout editor. Resource management systems like the Motif™ UIL offer a compromise by compiling into an intermediate form which is more or less interpreted at run time. Most builder tools will generate UIL as well.

For comparison, we evaluate the three approaches in terms of features desired in interface development: extensibility, time required to complete a change, and effort required to build the initial interface. Refer to Table 2-1.

	Builder Tools	**Motif™ UIL**	**Tcl/Tk**
Extensibilty	low	very low	very high
Time to complete change	high	moderate	low
Effort in constructing initial interface	low	high	very high

Table 2-1 Interface Development Comparison

SUMMARY

Tcl/Tk provides a set of libraries for building shell interpreters. Inherent in Tcl are language constructs for flow control, iteration, and modularization, and list and array as well as scalar variable storage. Language structure follows the UNIX command line paradigm. Tk provides commands for Motif™ style widgets implemented using Xlib. Tcl/Tk's real power lies in its extendibility, allowing the developer to add new commands and graphic widgets to the language.

In the following sections, we examine the four modes of development using Tcl/Tk: the scripting language, widget extensions, interfaces to 3GL code, and writing coprocesses.

3. PROGRAMMING IN THE SCRIPTING LANGUAGE

Tcl is to GUI interfaces as so-called "4GL" languages are to database management system (DBMS) interfaces. Thus, Tcl presents many of the same types of problems as do 4GLs:

- provides minimal support for programming structure,
- strains modular cohesion and coupling,
- forces run-time debugging,
- lacks debugging support tools,
- magnifies performance issues due to command interpretation.

Procedures, iteration, control, lists, arrays, and scalar variables are the only Tcl language constructs. Array variables must be global and cannot be passed as procedure parameters, forcing a dependence upon global variables. Variable declaration is implicit, so name misspelling and other errors are found at run time, often after much investigation. Except for matched braces, procedure syntax is not checked until run time, for the procedure is itself merely a string of commands to be executed. Commands are interpreted, making loops perform poorly.

However, simple operations can be scripted in Tcl for immediate testing, avoiding the compilation and linking of a C program. Changes are made quickly and tested immediately. Frequently, the developer is faced with an implementation choice:

- use only the scripting language,
- build functionality in C/C++,
- use a combination of scripting language and C/C++.

Three factors contribute to this decision:

- complexity,
- performance criticality,
- volatility.

The more complex the code, the more likely the scripting language will become a hindrance in the debugging effort. Similarly, there are situations, such as rapid real-time display, in which performance requires use of C/C++. On the other hand, highly volatile or changing objects or functions benefit from the flexibility and quick turnaround of the interpreted language. Figure 3-1 pictorially represents the interaction among these factors, assuming complexity and performance coincide.

In a prototyping environment, it is often desirable to use the scripting language until design decisions solidify, upon which critical objects and functions can be implemented in C/C++. Similarly, but less likely, it may be determined that operations should be moved from 3GL to the interpreted language to make changes easier.

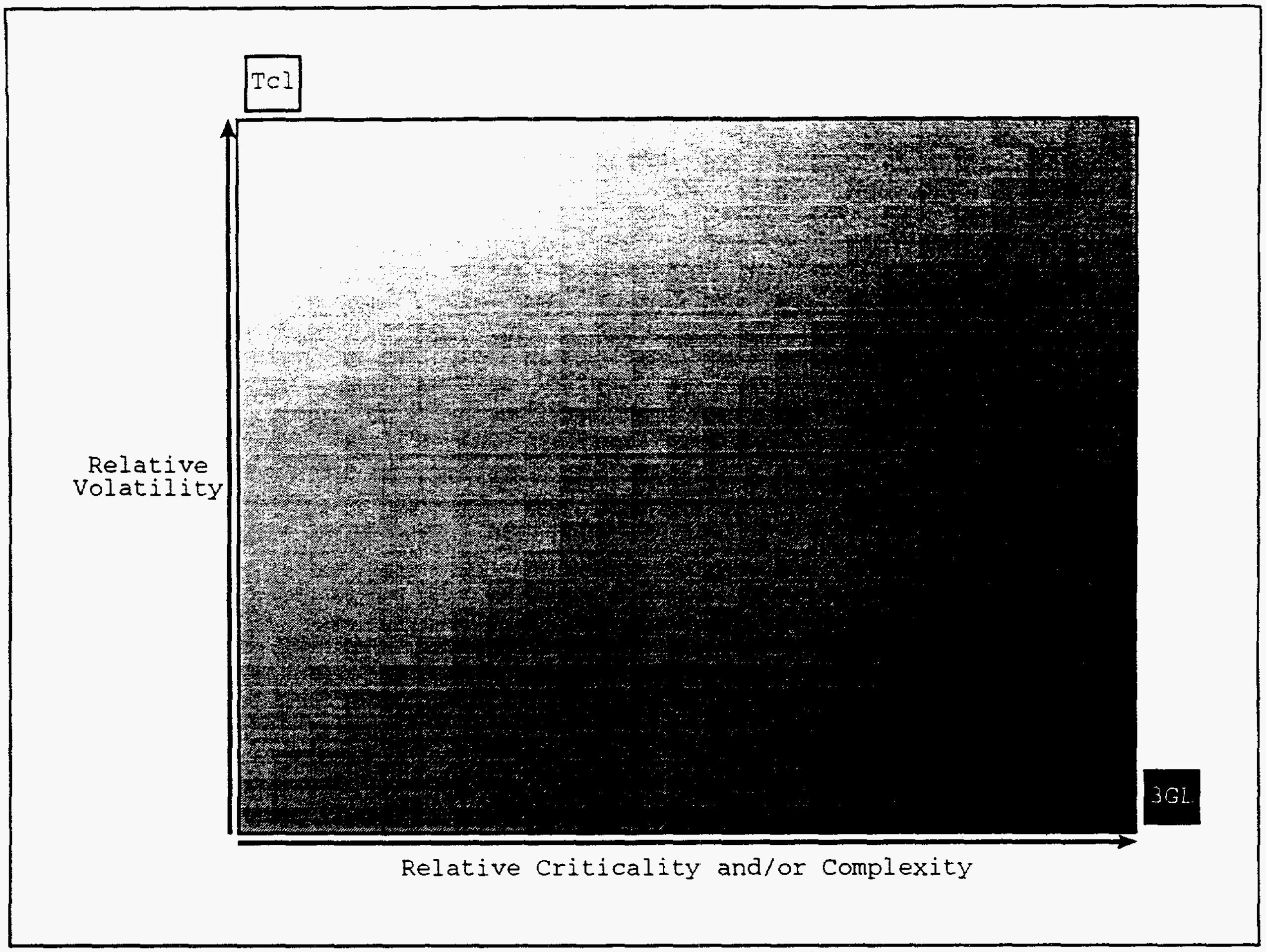

Figure 3-1 Scripting Language Versus 3GL Decision Metaphor

3.1 STRUCTURED DEVELOPMENT

In order for the Tcl scripting language to be used effectively, a structured framework for achieving software engineering goals, modifiability, efficiency, reliability, and understandability, must be adhered to. Software engineering concepts which make those goals achievable include: [2]

- abstraction,
- information hiding,
- modularity,
- localization,
- uniformity,
- completeness,
- confirmability.

Tcl language constructs do little to support these concepts. However, with the exception of the last two, the identified concepts can be realized by building upon the language constructs

Principle	Strategy
abstraction	Implement object-based packages
information hiding	Store attribute-value pairs for package instances in an array accessed only by the package methods
modularity	Implement package methods with procedures whose first parameter is the name of the object instance. Other than the package attribute array, all values manipulated by methods are passed as arguments or defined locally
localization	Implement all method functions within package framework
uniformity	Follow the above guidelines for all package implementations

Table 3-1 Tcl Programming Strategies to Achieve Software Engineering Concepts

and developing with discipline. For example, there are no "structures" or "records" to directly support data abstraction, but associative arrays and lists can be used for that purpose. Table 3-1 summarizes strategies to achieve the first five concepts.

3.2 OBJECT-BASED PACKAGING

Object-based extensions have been built for Tcl/Tk using only the scripting language. Tcl-DP provides such an environment. We have attempted to construct an object-oriented environment (supporting inheritance) adhering to Tcl-DP structure (refer to Appendix G) but found the performance hits for searching up class hierarchies too prohibitive. The interested reader should examine Tcl-DP as well as the scheme we used.

We've found a "class package" concept, modeled after the Ada package construct, to provide an effective object-based environment that fosters reuse. A package consists of a set of methods for creating and manipulating object instances of the class and a global array variable for storing class and object attributes. Class attribute values are referenced by attribute name, and object attribute values are referenced by object name and attribute name,

```
        class attribute:     BarWindow(count)
        object attribute:    BarWindow(barwin0,FileName)
```

Figure 3-2 Class and Object Attribute Reference

as shown in Figure 3-2.

Whereas the caller provides the name of the object when creating Tk widgets (necessary since parenting information is embedded in the name), package classes generate the name of the object and return it to the caller via a "create" method. Thereafter, the object name is passed to class methods. Methods related to a class are localized in source files so they can be loaded as needed using the Tcl `auto_load` feature. Use of the class attribute array limits the reusability of individual procedures to within its package, a weakness inherent in this approach.

3.2.1 Example

We used the class package concept for developing a graphical front end for SQL-based databases. Four data display constructs were implemented as packages:

- BarWindow
- SpreadSheet
- SpreadWindow
- TextWindow

A SpreadWindow contains a SpreadSheet, so resuse is already evident. All the packages are designed to be usable in other applications as well. Tcl/Tk source for the application is listed in Appendix A. We examine `BarWindow.tcl` below.

Specification

Since there are no explicit constructs for exporting or importing individual procedures or declaring variables. we must rely on documentation. Further, we cannot separate specification and implementation. Thus, the file header (shown in Figure 3-3) identifies which procedures are considered "exported." Next in the file, we set default resource values via the option command. These resources may be overridden in `.Xdefaults` files or by executing `xrdb`. Specification concludes with a comment block "declaring" object attributes. Of course, specifications of the procedure interfaces are found where the procedures are defined.

Implementation

Procedure/method bodies begin with the `global` command to identify the class array variable. Adhering to the software engineering principles of high intramodule cohesion and low intermodule coupling as much as possible, we reference only the class attribute array and procedure parameters.

Figure 3-4 lists `BarWindow_Create`, the package factory procedure. The new object's name is generated and returned to the caller and serves as the caller's handle for referencing the object via package procedures. Windows are constructed using other package methods, `BarWindow_CreateMenuBar` and `BarWindow_CreateGraph`.

Importing the Package

```
#------------------------------------------------------------------------
#-      PACKAGE:                                                        -
#-              BarWindow                                               -
#-      EXPORTED PROCEDURES:                                            -
#-              BarWindow_Create                                        -
#-              BarWindow_LoadFile                                      -
#-              BarWindow_LoadCommandResult                             -
#-      USAGE:                                                          -
#-              BarWindow_LoadFile [BarWindow_Create] filename          -
#-                                                                      -
#-              BarWindow_LoadCommandResult                             -
#-                      [BarWindow_Create] fdb-client-object            -
#------------------------------------------------------------------------
#------------------------------------------------------------------------
#-      Class Resource Values                                           -
#------------------------------------------------------------------------
option add *BarWindow*background gray75;
option add *BarWindow*foreground navy;
option add *BarWindow*barBackground gray75;
option add *BarWindow*plotBackground gray90;
option add *BarWindow*colors \
  { #ef0000 #00bf00 #dfdf00 #0000ff #00afdf orange #df00ff };
option add *BarWindow*plotBackground gray90;

global BarWindow;

#------------------------------------------------------------------------
#-      Class Attributes                                                -
#------------------------------------------------------------------------
set BarWindow(count) 0;

#------------------------------------------------------------------------
#-      Object Attributes                                               -
#-              Chart           name of the graph widget                -
#-              Data            list of observation lists, each with    -
#-                              the name of the observation and values  -
#-                              for each series                         -
#-              FileName        name of temp file containing data       -
#-              Series          list of series names                    -
#-              Stacked         if "true", each observation is a stack  -
#-                              of series value bars                    -
#-              Title           chart title                             -
#-              TopWindow       name of the toplevel widget which is    -
#-                              the window                              -
#-              XTitle          x-axis title                            -
#-              YTitle          y-axis title                            -
#------------------------------------------------------------------------
```

Figure 3-3 Tcl/Tk Package Specification

Driving the application is a module which builds the application menu bar and main window (`Fdb.tcl`). Since there are no package import language constructs, the driver must explicitly set a package's necessary resource values. However, when a procedure is referenced, its file location, if not in the current file, is searched automatically. Before importing one of these packages, its library resource value must be set, as shown in Figure 3-

```
        option add *BarWindow*library $Fdb_library;
        option add *SpreadWindow*library $Fdb_library;
        option add *TextWindow*library $Fdb_library;
```

Figure 3-5 Setting the Library Resource to Import Packages

```
proc  BarWindow_Create {} \
{
  global BarWindow;

#       -- Set Name of This Object
#       --
set this barwin$BarWindow(count);

#       -- Name of This Window
#       --
set name .bwtop$BarWindow(count);
set BarWindow($this,TopWindow) $name;

#       -- Build Toplevel
#       --
toplevel $name -class BarWindow;
wm iconbitmap $name @[option get $name library BarWindow]/barchart.xbm
wm iconmask $name @[option get $name library BarWindow]/barchart.xbm
wm iconname $name "BarChart - $BarWindow(count)";
wm minsize $name 300 200;
wm title $name "BarChart Window - $BarWindow(count)";

#       -- Build Frames
#       --
BarWindow_CreateMenuBar $name $this;
set chart_name [BarWindow_CreateGraph $this];
set BarWindow($this,Chart) $chart_name;

#       -- Pack Frames
#       --
pack $name.menubar -side top -fill x;
pack $chart_name -side top -expand yes -fill both;

#       -- Increment Window Count
#       --
incr BarWindow(count);

return  $this;
}
# BarWindow_Create
```

Figure 3-4 Package Factory Method

5. This option value points to the directory where run-time support files (e.g., icon bitmaps) are to be found. Each of the four packages can and hopefully will be used again in future applications. Although import is explicit, it is very simple.

Object-based packaging does provide some structure for development of reusable Tcl/Tk modules. However, the lack of support for inheritance hinders design and restricts reusability. Without some development framework, implementation in Tcl moves quickly toward an unmanageable heap of procedures that are very difficult to maintain. Development within the Tcl/Tk language would be greatly enhanced with extensions, implemented in C/C++, for declaration, definition, export, and import of class packages.

SUMMARY

As a language abstraction, Tcl offers compelling power through list processing. As a rapid prototyping tool, Tcl is unequaled in the speed with which a functional, working Motif™ style GUI application can be implemented.

However, as a software development environment, Tcl provides too little support for enforcing design structure. A means for declaring variables and ensuring they exist when referenced is necessary. Other semantic processing, such as type checking, would benefit the language greatly. Whereas compilers will often take care of syntax and semantic problems, Tcl code must be exhaustively exercised at run time in order to find these errors as well as logic problems. A reasonable heuristic is to limit implementation in Tcl language to GUI objects alone.

4. EXTENDING THE WIDGET SET

Extendibility is arguably Tcl/Tk's best feature for the developer. Functions and objects may be added to the language as available commands. In this section we focus on extending the widget set and drawing objects of Tk. Three types of enhancements are described: adding widgets, adding canvas items, and modifications affecting multiple widgets.

4.1 ADDING WIDGETS

Adding widgets is a refreshingly simple task in Tcl/Tk. Requirements for widget implementations are identified in Section 2.3. For each widget type, there exists a structure or class specifying the values/attributes stored for each object instance. Use of C++ allows the specification (and modular manipulation of) class attributes as well. For an example, refer to the source files `TkThing.h` and `TkThing.cc` listed in Appendix B. They contain a C++ class declaration/specification and implementation/body, respectively, for `thing` widgets. A `thing` widget is very simple--it merely draws a diagonal line from the top left corner to the bottom right corner. Figure 4-1 lists its object attributes.

```
Tk_Window           tkWin;
Tcl_Interp *        tkInterp;

unsigned short      tkFlags;
int                 tkWidth;
int                 tkHeight;

int                 tkBorderWidth;
Tk_3DBorder         tkBgBorder;
Tk_3DBorder         tkFgBorder;
int                 tkRelief;
```

Figure 4-1 Object Attributes for the `TkThing` Class

All widgets should have the first three attributes defined, for they store pointers to the widget's window and interpreter data structures and a mask of flags used to store the state of the widget, respectively. Most widgets will have a variable graphic size, making it necessary to store width and height. The final four attributes deal with background and border colors when drawing the widget.

The `Tk_3DBorder` type is used for three-dimensional shading of regions. When creating a `Tk_3DBorder` object, the caller specifies a single color, and the Tk library computes the corresponding illuminated and shaded colors (140% and 60% of base color red, green, and blue values). When drawing a three-dimensional region, such as via the supplied `Tk_Fill3DRectangle()` function, one of three relief values is specified: raised, sunken, or flat. Most of the Tk widgets are three-dimensional and have relief as a configuration option. Use of these features is illustrated in `TkThing::Display()`, shown in Figure 4-5. In the sections below we demonstrate the requirements for widgets as implemented for `things`.

4.1.1 Configuring the Widget

Configuration is the process of reading option values and setting the corresponding widget attributes accordingly. Fortunately, much of the grunt work involved in configuring a widget is handled by the `Tk_ConfigureWidget()` function, which is passed an array of `Tk_ConfigSpec` records. Each widget implementation should include a statically defined specification array. Whereas in C this array will probably have global scope (due to reference in more than one function), we can make it a class attribute in C++. Observe the declaration

```
class TkThing
  {
protected:

  static Tk_ConfigSpec  _tkConfigSpecs[];
.
.
.
Tk_ConfigSpec      TkThing::_tkConfigSpecs[] =
  {
    {TK_CONFIG_BORDER, "-background", "background", "Background",
    "#cdb79e", Tk_Offset(TkThing, tkBgBorder), TK_CONFIG_COLOR_ONLY},
    {TK_CONFIG_BORDER, "-background", "background", "Background",
    "white", Tk_Offset(TkThing, tkBgBorder), TK_CONFIG_MONO_ONLY},
    {TK_CONFIG_SYNONYM, "-bd", "borderWidth", (char *) NULL,
    (char *) NULL, 0, 0},
    {TK_CONFIG_SYNONYM, "-bg", "background", (char *) NULL,
    (char *) NULL, 0, 0},
    {TK_CONFIG_INT, "-borderwidth", "borderWidth", "BorderWidth",
    "2", Tk_Offset(TkThing, tkBorderWidth), 0},
    {TK_CONFIG_SYNONYM, "-fg", "foreground", (char *) NULL,
    (char *) NULL, 0, 0},
    {TK_CONFIG_BORDER, "-foreground", "foreground", "Foreground",
    "#b03060", Tk_Offset(TkThing, tkFgBorder), TK_CONFIG_COLOR_ONLY},
    {TK_CONFIG_BORDER, "-foreground", "foreground", "Foreground",
    "black", Tk_Offset(TkThing, tkFgBorder), TK_CONFIG_MONO_ONLY},
    {TK_CONFIG_INT, "-height", "height", "Height",
    "64", Tk_Offset(TkThing, tkHeight), 0},
    {TK_CONFIG_RELIEF, "-relief", "relief", "Relief",
    "raised", Tk_Offset(TkThing, tkRelief), 0},
    {TK_CONFIG_INT, "-width", "width", "Width",
    "64", Tk_Offset(TkThing, tkWidth), 0},
    {TK_CONFIG_END, (char *) NULL, (char *) NULL, (char *) NULL,
    (char *) NULL, 0, 0}
  };
```

Figure 4-2 Widget Configuration C++ Code

and definition of `TkThing::_tkConfigSpecs` shown in Figure 4-2.

A `Tk_ConfigSpec` record describes the characteristics of the configuration option:

- the option data type
- switch name (beginning with a dash and used in scripts)
- option database name
- option database class name
- a default string value

- the offset within the widget data structure of the corresponding attribute
- flag indicating the type of option
- optional function for storing and retrieving values for custom option types.

Offsets are computed using the Tk_Offset macro, and flag macros are defined for most of the data types one can use (e.g., TK_CONFIG_BORDER for Tk_3DBorder attributes and TK_CONFIG_INT for int attributes).

Implementations for widgets with configurable options should support a configure subcommand for widget object commands.[3] Configuration is also performed when the object is created. Figure 4-3 lists the TkThing::Configure() method, which is modeled after widget configuration routines in the Tk distribution. Note that configuration may result in the need to redraw the widget object, in which case the RedrawPending flag bit is set and Tk_DoWhenIdle() is called to register a class display method for invocation at a convenient time.

```
int
TkThing::Configure( int argc, char * argv[], int flags )
{
  int      result;


result = Tk_ConfigureWidget(
    tkInterp,
    tkWin,
    _tkConfigSpecs,
    argc,
    argv,
    (char *) this,
    flags
    );

if ( result == TCL_OK )
  {
  Tk_SetBackgroundFromBorder( tkWin, tkBgBorder );

  Tk_GeometryRequest( tkWin, tkWidth, tkHeight );
  Tk_SetInternalBorder( tkWin, tkBorderWidth );

  if ( !( tkFlags & RedrawPending ) )
    {
    Tk_DoWhenIdle( (Tk_IdleProc *) HandleDisplay, (ClientData) this );
    tkFlags |= RedrawPending;
    }
  }

return  result;
} // TkThing::Configure
```

Figure 4-3 Widget Configuration C++ Method

[3] A subcommand is usually the first argument of an object command.

4.1.2 Handling X Events

```
void
TkThing::HandleEvent( XEvent & event )
{
if (
    event.type == ConfigureNotify ||
    ( event.type == Expose &&
      event.xexpose.count == 0 )
    )
  {
  tkWidth = Tk_Width( tkWin );
  tkHeight = Tk_Height( tkWin );

  if ( !( tkFlags & RedrawPending ) )
    {
    Tk_DoWhenIdle( HandleDisplay, (ClientData) this );
    tkFlags |= RedrawPending;
    }
  }

else if ( event.type == DestroyNotify )
  {
  Tcl_DeleteCommand( tkInterp, Tk_PathName( tkWin ) );
  tkWin = NULL;

  if ( tkFlags & RedrawPending )
    Tk_CancelIdleCall( HandleDisplay, (ClientData) this );

  Tk_EventuallyFree( (ClientData) this, (Tk_FreeProc *) HandleDestroy );
  }
} // TkThing::HandleEvent
```

Figure 4-4 C++ Class Method for Handling X Events

The types of X Window events to be handled for a widget are dependent upon the nature of the widget, of course. Highly interactive widgets must respond to appropriate events (e.g., `ButtonPress, MotionNotify`). Resizable widgets must handle `ConfigureNotify`, but all widgets should process `Expose` and `DestroyNotify`. `Thing` widgets are not interactive, but the class method `TkThing::HandleEvent()`, listed in Figure 4-4, shows the algorithm for drawing a widget. Since `things` are resizable, we may need to draw on `ConfigureNotify` as well as `Expose` events. (We draw the entire widget rather than trying to reconstruct occluded regions, so we wait for an expose count of 0.)

Tk provides a mechanism to improve overall performance with Expose events. A call to `Tk_DoWhenIdle()` requests that a function (or C++ static method) be invoked when little other processing is ongoing. The `RedrawPending` mask bit is cleared when the widget is drawn in `TkThing::Display()`.

4.1.3 Displaying the Widget

Since display of a window is requested by the X server via an `Expose` event, the event handler method initiates most drawing of the widget. Some configuration changes require update of a widget's appearance. The static class method `TkThing::HandleDisplay()`, which merely invokes `Display()` for the object, is necessary so that a function address may

```
class  TkThing
  {
...
  static void           HandleDisplay( TkThing & thing )
                          {
                          tkthing.Display();
                          );
...
void
TkThing::Display()
{
  int          offset = tkBorderWidth << 1;
  GC           gc;


tkFlags &= ~RedrawPending;

if ( tkWin != NULL && Tk_IsMapped( tkWin ) )
  {
  gc = DefaultGC(
      Tk_Display( tkWin ),
      DefaultScreen( Tk_Display(tkWin) )
      );

  Tk_Fill3DRectangle(
      Tk_Display( tkWin ),
      Tk_WindowId( tkWin ),
      tkBgBorder,
      0, 0,
      Tk_Width( tkWin ), Tk_Height( tkWin ),
      tkBorderWidth,
      tkRelief
      );

  XSetLineAttributes(
      Tk_Display( tkWin ),
      gc,
      5,
      LineSolid,
      CapRound,
      JoinRound
      );

  XSetForeground(
      Tk_Display( tkWin ),
      gc,
      (Tk_3DBorderColor( tkFgBorder ))->pixel
      );

  XDrawLine(
      Tk_Display( tkWin ),
      Tk_WindowId( tkWin ),
      gc,
      offset,
      offset,
      tkWidth - offset,
      tkHeight - offset
      );
  } // if
} // TkThing::Display
```

Figure 4-5 Widget Display Code

be passed to `Tk_DoWhenIdle()`. Figure 4-5 lists the two methods.[4]

[4] The client data value passed in the Tk_DoWhenIdle() call is the C++ object address, which is passed by Tcl/Tk to the invoked routine. This address can be declared as a reference in the called method.

Widget display is accomplished using Xlib and Tk drawing functions. X context information (display, screen, drawable, graphics context) can be obtained from the `tkWin` object attribute (class member).

4.1.4 Destroying the Widget

A `DestroyNotify` event instigates widget destruction. The Tcl command for the widget must be unregistered, any pending display calls must be canceled, any memory allocated for the widget must be freed, and the widget data structure itself must be freed. This presents some potential problems. In the parent-child widget tree, it is possible for a child widget data structure to be deallocated prior to reference within a parent. Tk provides the `Tk_EventuallyFree()` function to circumvent this problem. It delays a widget's deallocation until all its ancestors have destroyed themselves.

Note also that our use of C++ requires another static class method, `TkThing::HandleDestroy()`, which is passed to `Tk_EventuallyFree()`. The widget object is deallocated with the C++ `delete` operator.

4.1.5 Responding to Widget Commands

As with options, the nature of the widget determines what commands it will support. A good deal of design effort should go into the subcommands a widget will process. In addition to `configure`, which all widgets should provide, widgets representing a value should provide subcommands for storing and retrieving values (e.g., `set` and `get`).

`TkThing::ProcessCommand()`, the object method for handling the `thing` commands, is called from the static class method `TkThing::ProcessWidgetCommand()` (refer to Figure 4-6). Here the only subcommand is `configure`, but it must be handled specially due to its three forms:

- no additional arguments -- request the values of all options
- one additional argument -- request the value of an option
- two or more additional arguments -- set the values of one or more options

`Tk_ConfigureInfo()` builds return value strings for the first two forms. Refer to Figure 4-6 for a widget object command handler.

4.1.6 Creating Widget Objects

Widget objects are created via a factory command. In our example, the command is `thing`, the implementation for which is the class method `TkThing::ProcessFactoryCommand()`, listed in Figure 4-7. `Tk_CreateWindowFromPath()` handles creation of the widget's X window from the window path name. A new C++ object is allocated with the `new` operator, which calls the appropriate object constructor for initialization. The widget's class name (for the option database operations) is set, the X event handler for the window is registered, and a command for the new widget object is registered.

```
int
TkThing::ProcessCommand( int argc, char * argv[] )
{
  int      result = TCL_OK;


if ( argc < 2 )
  {
  Tcl_AppendResult( tkInterp, "Usage: ", argv[0], " option ?arg ...?", NULL );
  result = TCL_ERROR;
  }

else if ( strcmp( argv[1], "configure" ) == 0 )
  {
  if ( argc == 2 )
    {
    result = Tk_ConfigureInfo(
    tkInterp,
    tkWin,
    _tkConfigSpecs,
    (char *) this,
    NULL,
    0
    );
    }

  else if ( argc == 3 )
    {
    result = Tk_ConfigureInfo(
    tkInterp,
    tkWin,
    _tkConfigSpecs,
    (char *) this,
    argv[2],
    0
    );
    }

  else
    result = Configure( argc - 2, argv + 2, TK_CONFIG_ARGV_ONLY );
  }

return  result;
} // TkThing::ProcessCommand
```

Figure 4-6 Widget Object Command Handler

4.2 ADDING CANVAS ITEMS

Unfortunately, the ease of adding widgets to Tcl/Tk is not quite duplicated for canvas items, though there is arguably less need for extension of canvas item types. Canvas items are not windows, but they are otherwise very similar to widgets in that both must support configuration (using the same facilities), deletion, and both must draw themselves. Section 2.3 enumerates the requirements for canvas item type implementations. Of particular note are the distance computation and PostScript generation routines.

Appendix C contains the source code for a canvas item extension, the `bar` item, implemented as the C++ `TkBarItem` class. `TkBarItem` contains two class attributes: a `Tk_ConfigSpec` object describing configuration options, and a `Tk_ItemType` object, necessary for canvas item implementations. It gives the canvas widget information necessary for processing the item type, including:

```
int
TkThing::ProcessFactoryCommand(
    Tk_Window       main_win,
    Tcl_Interp *    interp,
    int             argc,
    char *          argv[]
    )
{
  int               result = TCL_OK;
  Tk_Window         new_win;
  TkThing *         new_thing;


     // Check For Minimal Arguments
     //
if ( argc < 2 )
  {
  Tcl_SetResult( interp, "Usage: thing pathname ?options?", TCL_VOLATILE );
  result = TCL_ERROR;
  }

     // Create New Tk Window
     //
else if (
    (new_win = Tk_CreateWindowFromPath( interp, main_win, argv[1], NULL )) ==
    NULL
    )
  {
  Tcl_SetResult( interp, "Error creating window", TCL_VOLATILE );
  result = TCL_ERROR;
  }

     // Create New Object
     //
else if (
    (new_thing = new TkThing( new_win, interp, argc - 2, argv + 2 )) == NULL
    )
  {
  Tcl_SetResult( interp, "Error allocating data structure", TCL_VOLATILE );
  result = TCL_ERROR;
  }

     // Check For Configure Error
     //
else if ( new_thing->tkFlags & ConfigError )
  {
  Tk_DestroyWindow( new_win );
  delete new_thing;
  result = TCL_ERROR;
  }

else
  {
  Tk_SetClass( new_win, "Thing" );

  Tk_CreateEventHandler(
      new_win,
      ExposureMask | StructureNotifyMask,
      (Tk_EventProc *) HandleEvents,
      (ClientData) new_thing
      );

  Tcl_CreateCommand(
      interp,
      Tk_PathName( new_win ),
      (Tcl_CmdProc *) ProcessWidgetCommand,
      (ClientData) new_thing,
      NULL
      );

  Tcl_SetResult( interp, Tk_PathName( new_win ), TCL_VOLATILE );
  } // else everything allocated

return  result;
} // TkThing::ProcessFactoryCommand
```

Figure 4-7 Widget Factory Command Handler

- the item type name,
- the size of the item's data structure, and
- the addresses of required and optional routines

Item types are registered by passing the `Tk_ItemType` object to the `Tk_CreateItemType()` function. Figure 4-8 shows the definition of the two class attributes.

```
Tk_ConfigSpec            TkBarItem::_biConfigSpecs[] =
  {
    { TK_CONFIG_BORDER, "-fill", (char *) NULL, (char *) NULL,
        "gray50", Tk_Offset(TkBarItem, biFillBorder), TK_CONFIG_NULL_OK },
    { TK_CONFIG_CUSTOM, "-tags", (char *) NULL, (char *) NULL,
        (char *) NULL, 0, TK_CONFIG_NULL_OK, &tkCanvasTagsOption },
    { TK_CONFIG_PIXELS, "-width", (char *) NULL, (char *) NULL,
        "1", Tk_Offset(TkBarItem, biWidth), TK_CONFIG_DONT_SET_DEFAULT },
    { TK_CONFIG_RELIEF, "-relief", NULL, NULL,
        "raised", Tk_Offset(TkBarItem, biRelief), TK_CONFIG_DONT_SET_DEFAULT },
    { TK_CONFIG_END, (char *) NULL, (char *) NULL, (char *) NULL,
        (char *) NULL, 0, 0 }
  };

Tk_ItemType              TkBarItem::_biItemType =
  {
  "bar",                                /* name */
  sizeof(TkBarItem),                    /* itemSize */
  TkBarItem::Create,                    /* createProc */
  TkBarItem::_biConfigSpecs,            /* configSpecs */
  TkBarItem::Configure,                 /* configureProc */
  RectOvalCoords,                       /* coordProc */
  TkBarItem::Delete,                    /* deleteProc */
  TkBarItem::Draw,                      /* displayProc */
  0,                                    /* alwaysRedraw */
  RectToPoint,                          /* pointProc */
  RectToArea,                           /* areaProc */
  TkBarItem::GenPostScript,             /* postscriptProc */
  ScaleRectOval,                        /* scaleProc */
  TranslateRectOval,                    /* translateProc */
  (Tk_ItemIndexProc *) NULL,            /* indexProc */
  (Tk_ItemCursorProc *) NULL,           /* icursorProc */
  (Tk_ItemSelectionProc *) NULL,        /* selectionProc */
  (Tk_ItemInsertProc *) NULL,           /* insertProc */
  (Tk_ItemDCharsProc *) NULL,           /* dTextProc */
  (Tk_ItemType *) NULL                  /* nextPtr */
  };
```

Figure 4-8 Canvas Item Configuration and Type Definition

4.2.1 Creating an Item Instance

Canvas item types are created via the canvas `create` subcommand, so no factory command is necessary. In our example, the creation is implemented with the class method `TkBarItem::Create()`, listed in Figure 4-9. The canvas allocates the item object's memory prior to calling the create function. As with widgets, initial configuration must be performed, and the initial coordinates passed as command arguments are set using the `TkGetCanvasCoord()` function.

4.2.2 Item Configuration

Canvas item configuration mirrors widget configuration with one exception, the need to update any coordinate changes. After calling `Tk_ConfigureWidget()`, the class method `TkBarItem::Configure()` calls `ComputeRectOvalBox()` to update bounding box information, as illustrated in Figure 4-10.

4.2.3 Storing and Retrieving Coordinates

Extent or boundary information must be maintained for a canvas item for drawing,

```
int
TkBarItem::Create(
    Tk_Canvas *  canvas_ptr,
    Tk_Item *    item_ptr,
    int          argc,
    char *       argv[]
    )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;
  int                   status = TCL_OK;

        // Check Argument List
        //
if ( argc < 4 )
  {
  Tcl_AppendResult(
      canvas_ptr->interp,
      "wrong # args:  should be \"",
      Tk_PathName( canvas_ptr->tkwin ),
      "\" create ",
      item_ptr->typePtr->name,
      " x1 y1 x2 y2 ?options?",
      NULL
      );

  status = TCL_ERROR;
  }

else
  {
  bar_ptr->biWidth = 1;
  bar_ptr->biFillBorder = NULL;
  bar_ptr->biRelief = TK_RELIEF_RAISED;

                 // Process Args
                 //
  if (
    (TkGetCanvasCoord( canvas_ptr, argv[0], &bar_ptr->biBbox[0] ) != TCL_OK) ||
    (TkGetCanvasCoord( canvas_ptr, argv[1], &bar_ptr->biBbox[1] ) != TCL_OK) ||
    (TkGetCanvasCoord( canvas_ptr, argv[2], &bar_ptr->biBbox[2] ) != TCL_OK) ||
    (TkGetCanvasCoord( canvas_ptr, argv[3], &bar_ptr->biBbox[3] ) != TCL_OK) ||
    (Configure( canvas_ptr, item_ptr, argc - 4, argv + 4, 0 ) != TCL_OK)
      )
    {
    Delete( canvas_ptr, item_ptr );
    status = TCL_ERROR;
    }
  }

return  status;
} // TkBarItem::Create
```

Figure 4-9 Canvas Item Create Method

computing distances to points, and determining overlap with regions. Canvas item types native to Tk include rectangles, ovals, polygons, and text, so most geometric situations are already handled within Tk source code. However, these C routines, internal to the Tk library, are defined as `static`, and in order to make them exportable, and thus usable, modification of the Tk source is necessary. The `bar` item type is rectangular in shape, so we removed `static` tags and used the routines from `tkRectOval.c`. `RectOvalCoords()` is registered to handle maintenance of the item's bounding box.

4.2.4 Deletion

Obviously, when a canvas item is deleted, any memory or objects allocated during the

```
int
TkBarItem::Configure(
    Tk_Canvas *        canvas_ptr,
    Tk_Item *          item_ptr,
    int                argc,
    char *             argv[],
    int                flags
    )
{
  register TkBarItem * bar_ptr = (TkBarItem *) item_ptr;
  int                  status;

status = Tk_ConfigureWidget(
    canvas_ptr->interp,
    canvas_ptr->tkwin,
    _biConfigSpecs,
    argc,
    argv,
    (char *) bar_ptr,
    flags
    );

if ( status == TCL_OK )
  ComputeRectOvalBbox( canvas_ptr, (char *) bar_ptr );

return  status;
} // TkBarItem::Configure
```

Figure 4-10 Canvas Item Configuration

item's life must be freed. The canvas takes care of deallocating the memory for the item itself. Refer to the class method `TkBarItem::Delete()`.

4.2.5 Drawing the Item

Unlike widgets, canvas items are not themselves windows. Instead, they are drawn by the canvas in its widget window. Thus, item coordinates must be translated to the coordinate space of the canvas's current scroll region. `TkBarItem::Draw()` is listed in Figure 4-11. Note that the canvas passes the X Drawable for use with X rendering commands. This is often a pixmap used for buffering canvas display.

4.2.6 Distance Computation

As with bounding box maintenance, the distance computation routines are taken from `tkRectOval.c`. `RectToPoint()` computes the distance from the item's extent to a point, and `RectToArea()` determines whether the item lies within, outside, or overlaps a rectangular region.

4.2.7 Generating PostScript

Canvases provide the `postscript` subcommand to generate a PostScript representation of the canvas. Thus, each canvas item type must be capable of rendering an item in PostScript. `TkBarItem::GenPostScript()`, listed in Figure 4-12, demonstrates the PostScript generation for a canvas item. Again, Tk provides support routines. `TkCanvPsY()` projects the canvas coordinate into the vertical dimension of the PostScript page, and

```
void
TkBarItem::Draw(
    Tk_Canvas * canvas_ptr,
    Tk_Item *   item_ptr,
    Drawable    drawable
    )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;

  int                   x1 = SCREEN_X( canvas_ptr, bar_ptr->biBbox[0] );
  int                   y1 = SCREEN_Y( canvas_ptr, bar_ptr->biBbox[1] );
  int                   x2 = SCREEN_X( canvas_ptr, bar_ptr->biBbox[2] );
  int                   y2 = SCREEN_Y( canvas_ptr, bar_ptr->biBbox[3] );


        // Must Be At Least 1 Pixel Big in Each Dimension
        //
if ( x2 <= x1 )
  x2 = x1 + 1;

if ( y2 <= y1 )
  y2 = y1 + 1;

Tk_Fill3DRectangle(
    canvas_ptr->display,
    drawable,
    bar_ptr->biFillBorder,
    x1, y1,
    x2 - x1 - 1, y2 - y1 - 1,
    bar_ptr->biWidth,
    bar_ptr->biRelief
    );
} // TkBarItem::Draw
```

Figure 4-11 Drawing the Canvas Item

`TkCanvPsColor()` generates PostScript commands to set an appropriate color. PostScript commands are set as the interpreter result in the generator routine.

4.2.8 Scaling and Translation

In response to the `scale` and `move` subcommands, a canvas widget calls routines for referenced items to scale or translate, respectively. Both operations require update of an item's extent or bounding box. For the `bar` item we use `ScaleRectOval()` and `TranslateRectOval()`, respectively, from `tkRectOval.c`.

4.3 OTHER ENHANCEMENTS

Enhancements other than new widgets or canvas items will likely require modification of source in the Tk distribution. We describe two examples here: a simple enhancement of the `Tk_3DBorder` facilities, and extension of bitmap processing to include XPM2 pixmap files.

4.3.1 Border Color Functions

There are many data structures and functions defined for internal use within Tcl/Tk. The discussions above identified the need to export needed functions. Adhering to the software engineering principle of information hiding, many data structures have external or exported versions which are merely opaque pointers. Examples include:

```
int
TkBarItem::GenPostScript(
    Tk_Canvas *         canvas_ptr,
    Tk_Item *           item_ptr,
    Tk_PostscriptInfo * ps_info_ptr
    )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;
  int                   status = TCL_OK;

  char                  path_cmd[ 500 ];
  char                  string[ 100 ];

  double                y1 = TkCanvPsY( ps_info_ptr, bar_ptr->biBbox[1] );
  double                y2 = TkCanvPsY( ps_info_ptr, bar_ptr->biBbox[3] );
  double                deltax = bar_ptr->biBbox[2] - bar_ptr->biBbox[0];
  double                deltay = y2 - y1;

  XColor *              bg = Tk_3DBorderColor( bar_ptr->biFillBorder );
  XColor *              light = Tk_3DBorderLightColor( bar_ptr->biFillBorder );
  XColor *              dark = Tk_3DBorderDarkColor( bar_ptr->biFillBorder );


        // Check Colors
        //
if ( bg == NULL || light == NULL || dark == NULL )
  status = TCL_ERROR;

else
  {
                // Draw Filled Rectangle
                //
  sprintf(
      path_cmd,
      "newpath\n"
      "%g %g moveto %g 0 rlineto 0 %g rlineto %g neg 0 rlineto closepath\n",
      bar_ptr->biBbox[0], y1,
      deltax,
      deltay,
      deltax
      );

  Tcl_AppendResult( canvas_ptr->interp, path_cmd, NULL );
  TkCanvPsColor( canvas_ptr, ps_info_ptr, bg );
  Tcl_AppendResult( canvas_ptr->interp, "fill\n", NULL );

                // Draw Bottom Shade Lines
                //
  sprintf(
      path_cmd,
      "%g %g moveto %g 0 rlineto 0 %g rlineto %d setlinewidth\n",
      bar_ptr->biBbox[0], y2,
      deltax,
      -deltay,
      bar_ptr->biWidth
      );

  Tcl_AppendResult( canvas_ptr->interp, path_cmd, NULL );
  TkCanvPsColor(
      canvas_ptr,
      ps_info_ptr,
      bar_ptr->biRelief == TK_RELIEF_RAISED ? dark :
      bar_ptr->biRelief == TK_RELIEF_FLAT ? bg : light
      );
  Tcl_AppendResult( canvas_ptr->interp, "stroke\n", NULL );

                // Draw Top Shade Lines
                //
  sprintf(
      path_cmd,
      "%g %g moveto 0 %g rlineto %g 0 rlineto %d setlinewidth\n",
      bar_ptr->biBbox[0], y2,
      -deltay,
      deltax,
      bar_ptr->biWidth
      );

  Tcl_AppendResult( canvas_ptr->interp, path_cmd, NULL );
  TkCanvPsColor(
      canvas_ptr,
      ps_info_ptr,
      bar_ptr->biRelief == TK_RELIEF_RAISED ? light :
      bar_ptr->biRelief == TK_RELIEF_FLAT ? bg : dark
      );
  Tcl_AppendResult( canvas_ptr->interp, "stroke\n", NULL );
  }

return  status;
} // TkBarItem::GenPostScript
```

Figure 4-12 Canvas Item PostScript Generation

- private `TkWindow` and public `Tk_FakeWin`
- private `Border` and public `Tk_3DBorder`

```
/************************************************************************
*       TYPE:           Border                                          *
************************************************************************/
struct Border
  {
  Display •             display;
  int                   refCount;
  XColor *              bgColorPtr;
  XColor *              lightColorPtr;
  XColor •              darkColorPtr;
  Pixmap                shadow;
  GC                    lightGC;
  GC                    darkGC;
  GC                    bgGC;
  Tcl_HashEntry *       hashPtr;
  };

/************************************************************************
*       NAME:           Tk_3DBorderLightColor()                         *
*       HISTORY:                                                        *
************************************************************************/
XColor *
Tk_3DBorderLightColor( Tk_3DBorder border )
{
return  ((Border •)border)->lightColorPtr;
} // Tk_3DBorderLightColor

/************************************************************************
•       NAME:           Tk_3DBorderDarkColor()                          •
*       HISTORY:                                                        *
************************************************************************/
XColor •
Tk_3DBorderDarkColor( Tk_3DBorder border )
{
return  ((Border *)border)->darkColorPtr;
} // Tk_3DBorderDarkColor
```

Figure 4-13 Retrieval of Shade Colors From a Tk_3DBorder

Border objects store XColor values for background, light shade, and dark shade colors. However, only the opaque pointer type Tk_3DBorder is exported. A function is provided to retrieve the background color from a border, Tk_3DBorderColor(), but no facilities are provided for retrieving the shade colors. Figure 4-13 lists the trivial code to support retrieval of the shade colors. The Border structure definition is repeated (not good from the maintenance point of view) to allow access of the necessary fields.

4.3.2 Pixmap Processing

Bitmap facilities are not effectively localized in an object package in Tk. Instead, they are spread among the various widget packages which process them:

- buttons,
- canvases,
- menus, and
- menubuttons.

Unfortunately, this requires modification of several Tk source files:

- tkBitmap.c,
- tkButton.c,
- tkCanvBmap.c,
- tkMenu.c, and
- tkMenubutton.c

Tk extensions for handling XPM files exist, but at the time of our efforts there were no facilities for handling XPM2 pixmap files in the a format generated by iconedit under OpenWindows. We added this capability with two additional source files, tkPixmap.h and tkPixmap.c, as well as slight modifications to the files listed above. Appendix D contains the source listings. Tk supports reference of X Window bitmap files (most commonly used as the value of the -bitmap option) by name with a prefix of '@'. We extend this notation to include the '%' prefix for XPM2 format files and the '^' prefix for X Window dump files. Examples of such references follow:

- @/usr/include/X11/bitmaps/xlogo32
- %/usr/openwin/include/pixmaps/folder.xpm
- ^/usr/local/xwd/picture.xwd

Two basic modifications were necessary to existing Tk code: changes within tkBitmap.c to enhance bitmap processing, and changes to the widget source for rendering "bitmaps." Within tkBitmap.c, the unexported TkBitmap structure definition was modified to include a depth field, a Tk_DepthOfBitmap() function was added for retrieval of a bitmap object's depth value, and the Tk_GetBitmap() function was enhanced to support the new prefixes in file name references.

As delivered, Tk widget sources render bitmaps using XCopyPlane(). Each such call had to be modified with a call to Tk_DepthOfBitmap() to determine bitmap versus pixmap, and a conditional statement for rendering. Whereas a depth of one indicates a bitmap and results in an XCopyPlane() call, a depth greater than one is a pixmap and is rendered using XCopyPlane(). This is illustrated in Figure 4-14, which lists a code snippet from the DisplayButton() function in tkButton.c.

```
static void
DisplayButton(clientData)
    ClientData;    /* Information about widget. */
{
    .
    .
    .
        if ( depth > 1 )
      {
      XCopyArea(
          butPtr->display, butPtr->bitmap,
          pixmap, gc,
          0, 0, width, height,
          x, y
          );
      }
    else
      {
      XCopyPlane(
          butPtr->display, butPtr->bitmap,
          pixmap, gc,
          0, 0, width, height,
          x, y, 1
          );
      }
```

Figure 4-14 Code to Render Pixmaps and Bitmaps

5. EMBEDDING CLASSES AND FUNCTIONS

One of the most practical ways to exploit the power of Tcl is to build commands for classes, data structures, and functions developed in a 3GL, thereby embedding their functionality within an interactive scripting language. This allows test harnesses to be scripted, permitting more exhaustive tests to be developed quickly. Further, one should work to improve the exportability of modules and classes in order to increase the likelihood of their reuse. Two examples of Tcl interfaces to C++ classes are used to discuss the mechanics of such an interface and techniques for data exchange. The first, `TclFdbClient`, is built on an application-specific socket communication facility. Second, `TclSampler` fronts a statistical sampling facility. Source code for the examples is listed in Appendix E.

5.1 TCL INTERFACES TO C++ CLASSES

C++ supports inheritance, the single characteristic distinguishing object-oriented from object-based environments [1]. Thus, we can construct a derived class which provides the boilerplate and processing methods for achieving a Tcl interface to the class from which it inherits. Figure 5-1 shows the declaration of such a class (with comments removed). Common to all such classes are object attributes, class methods for Tcl callbacks and registration, and object methods to implement subcommands. Clearly we want to use the object-based extension mode.

5.1.1 Object Attributes

A necessary object attribute for any Tcl class is a pointer to the Tcl interpreter, for it is passed to most Tcl/Tk functions. For object-based extensions, it is necessary to store the name of object instance as referenced in the Tcl script in order to unregister the object's command when it is destroyed. Hence we have the `tfcInterp` and `tfcName` attributes, respectively. The first two parameters in the constructor are for setting their values. Thus, they must be provided when an object is instantiated.

5.1.2 Class Methods for Tcl Callbacks and Registration

C++ class examples in Section 4 demonstrated the need for class methods implementing the factory command, object command, and object deletion. In addition, the inline class method `Register()` calls `Tcl_CreateCommand()` to register the factory command name, `FdbClient`, and the factory method, `ProcessClassCommand()`.

Figure 5-2 lists `TclFdbClient::ProcessClassCommand()`. If the necessary arguments are provided, a new object is allocated and initialized via the C++ `new` operator, which calls the constructor. If object allocation succeeds, `Tcl_CreateCommand()` is invoked to register a command with the same name as the object itself. The callback for all object commands is `ProcessObjectCommand()`, and the object command deletion handler is `DeleteObject()`.

For consistency with other Tcl object-based extensions (e.g., Tk widgets), object

```
class  TclFdbClient : public FdbClient
    {
protected:

  Tcl_Interp *          tfcInterp;
  char *                tfcName;                // Tcl variable name

public:

                        TclFdbClient(
                            Tcl_Interp *,
                            char *              var_name,
                            char *              auth_id,
                            char *              auth_key
                            );

  virtual               ~TclFdbClient();

protected:

  int                   ProcessGetStatus(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

  int                   ProcessReadNextLine(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

  int                   ProcessSendCommand(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

public:

  static void           DeleteObject( TclFdbClient * );

  static int            ProcessObjectCommand(
                            TclFdbClient *      ds_ptr,
                            Tcl_Interp *        interp,
                            int                 argc,
                            char *              argv[]
                            );

  static int            ProcessClassCommand(
                            ClientData          data,
                            Tcl_Interp *        interp,
                            int                 argc,
                            char *              argv[]
                            );

  static inline
  void                  Register( Tcl_Interp * interp )
                          {
                          Tcl_CreateCommand(
                              interp,
                              "FdbClient",
                              (Tcl_CmdProc *) ProcessClassCommand,
                              (ClientData) NULL,
                              (Tcl_CmdDeleteProc *) NULL
                              );
                          };
  };
```

Figure 5-1 Tcl Derived Class Declaration

commands should take a subcommand as the first argument and subcommand-specific arguments and options on the remainder of the command line. In designing the command structure, we need only refer to the class from which we inherit. Methods to be interfaced should have a corresponding subcommand. In the case of `TclFdbClient`, we implement three such subcommands: `GetStatus`, `ReadNextLine`, and `SendCommand`.[5] For each subcommand there is a corresponding object method. `TclFdbClient::ProcessObjectCommand()` is listed in Figure 5-3. When the subcommand is recognized as one of those supported, the

[5] The use of capitalization in commands and subcommands is a departure from standard Tcl practice, but it identifies our local C++ implementations and distinguishes them from other extensions.

```
int
TclFdbClient::ProcessClassCommand(
    ClientData,
    Tcl_Interp *            interp,
    int                     argc,
    char •                  argv[]
    )
{
  int                       status = TCL_OK;
  TclFdbClient •            new_item;
  char                      address_value[ 32 ];

        // Check For Necessary Parameters
        //
if ( argc < 4 )
  {
  Tcl_SetResult(
      interp,
      •Usage: FdbClient varname auth-id auth-key•,
      TCL_STATIC
      );
  status = TCL_ERROR;
  }
        // Process Arguments
        //
else if (
    (new_item = new TclFdbClient( interp, argv[1], argv[2], argv[3] )) == NULL
    )
  {
  Tcl_SetResult( interp, •error allocating new client object•, TCL_STATIC );
  status = TCL_ERROR;
  }

else
  {
                // Register Object Command
                //
  Tcl_CreateCommand(
      interp,
      argv[1],
      (Tcl_CmdProc *) ProcessObjectCommand,
      (ClientData) new_item,
      (Tcl_CmdDeleteProc *) DeleteObject
      );
  }

return  status;
} // TclFdbClient::ProcessClassCommand
```

Figure 5-2 Tcl Class Factory Command Method

corresponding object method is invoked for the object. The remainder of the command line after the subcommand itself is passed to the subcommand method. Note that the `Delete` subcommand is common to all object-based commands and is handled with a call to `Tcl_DeleteCommand()`.

For uniformity, one of our software engineering principles, the method naming scheme should be repeated for all Tcl interface classes:

- `ProcessClassCommand()` -- class factory command method
- `ProcessObjectCommand()` -- class object command method
- `DeleteObject()` -- class object deletion method
- `Process`*???*`()` -- object subcommand methods

```
int
TclFdbClient::ProcessObjectCommand(
    TclFdbClient *      client_ptr,
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status;

        // Default Command is GetStatus
        //
if ( argc < 2 )
  status = client_ptr->ProcessGetStatus( interp, 0, NULL );

else if ( strcmp( argv[1], "GetStatus" ) == 0 )
  status = client_ptr->ProcessGetStatus( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "ReadNextLine" ) == 0 )
  status = client_ptr->ProcessReadNextLine( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "SendCommand" ) == 0 )
  status = client_ptr->ProcessSendCommand( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "Delete" ) == 0 )
  {
  status = TCL_OK;
  Tcl_DeleteCommand( interp, client_ptr->tfcName );
  }

else
  {
  status = TCL_ERROR;

  Tcl_AppendResult(
      interp,
      "Usage:\n",
      "\tFdbClient Delete\n",
      "\tFdbClient GetStatus (returns status code)\n",
      "\tFdbClient ReadNextLine varname [ wait-secs ] (returns eof on EOF)\n",
      "\tFdbClient SendCommand command-string\n",
      NULL );
  }

return  status;
} // TclFdbClient::ProcessObjectCommand
```

Figure 5-3 Tcl Class Object Command Method

5.1.3 Object Subcommand Methods

The specification for object subcommand methods contains three parameters: a pointer to the Tcl interpreter, and argc and argv values describing the subcommand arguments. For an example, refer to Figure 5-4 which lists TclFdbClient::ProcessReadNextLine(). Recall that all data are represented within Tcl as strings. As with any UNIX command, numeric arguments must be converted from string format. Within Tcl, numeric results must be converted back to strings. Command results are returned by setting the result field of the Tcl interpreter. Tcl provides five functions to assist in setting results:

- Tcl_SetResult()
- Tcl_AppendResult()
- Tcl_AppendElement()
- Tcl_ResetResult()

```
int
TclFdbClient::ProcessReadNextLine(
    Tcl_Interp •        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status = TCL_OK;
  char          buffer[ 32 ];
  char •        next_line;
  Timeval       wait_time;
  Timeval •     time_ptr = NULL;

        // Check For VarName Parameter
        //
if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult(
      interp,
      •Usage: client ReadNextLine varname [ wait-secs ]•,
      TCL_STATIC
      );
  }

else
  {
  if ( argc > 1 )
    {
    wait_time = Timeval( atoi( argv[1] ), 0 );
    time_ptr = & wait_time;
    }

  if ( (next_line = (char *) ReadNextLine( time_ptr )) == NULL )
    {
    Tcl_SetVar( interp, argv[0], ••, TCL_LEAVE_ERR_MSG );
    strcpy( buffer, •eof• );
    }

  else
    {
    Tcl_SetVar( interp, argv[0], next_line, TCL_LEAVE_ERR_MSG );
    sprintf( buffer, •%d•, strlen( next_line ) );
    } // else read succeeded

  Tcl_SetResult( interp, buffer, TCL_VOLATILE );
  } // else parameters supplied

return  status;
} // TclFdbClient::ProcessReadNextLine
```

Figure 5-4 Tcl Object Subcommand Method

- `Tcl_FreeResult()`

`ReadNextLine` takes the name of a Tcl variable to set as an argument. Variable values may be set and retrieved via `Tcl_SetVar()` and `Tcl_GetVar()`, respectively. Other functions are provided for setting and retrieving array values and removing a variable.

5.2 TECHNIQUES FOR DATA EXCHANGE

Inevitably, one will encounter a need to exchange a large amount of data, such as an array or structure, between the Tcl language and 3GL code. Tcl lists are an effective mechanism for this exchange. Further, addresses and handles may be passed as well.

5.2.1 Arrays and Lists

Lists are the primary construct for storing collections of data in Tcl and are implemented as strings. Elements are delimited within the string. Tcl provides several functions for manipulating lists, including `Tcl_SplitList()` and `Tcl_Merge()` for dividing a list into individual element strings and combining element strings into a list, respectively. If special handling is not required (e.g., processing backslash characters or nested quotes), lists can be created using string functions.

Figure 5-5 lists `TclSampler::ProcessGetLastSample()`. The `GetLastSample` subcommand of a `Sampler` object returns an array of tuples. We accomplish this by building a list of lists, which is merely a large string with the element lists inside braces. Within the `for` loop of Figure 5-5, `sprintf()` is called to append the tuple lists to the outer list string. Figure 5-7 shows a sample Tcl subcommand call. Data arrays are passed in a similar way. Elements are extracted from the argument containing the list via `Tcl_SplitList()`.

5.2.2 Addresses and Handles

The need may arise to exchange a memory address or data handle between 3GL modules through the Tcl language. Very simply, these values may be exchanged as numbers in string form. An example of such an operation is in `TclSampler::ProcessGetImageAddr()`, listed in Figure 5-6. The value of the `sImage` attribute, an `XImage` address, is set as the command result.

```
int
TclSampler::ProcessGetImageAddr( Tcl_Interp • interp, int argc, char * argv[] )
{
  int           status = TCL_OK;
  char          address[ 32 ];

        // Must Have Gotten Name of Canvas
        //
sprintf( address, "%01d", (long) sImage );
Tcl_SetResult( interp, address, TCL_VOLATILE );

return  status;
} // TclSampler::ProcessGetImageAddr
```

Figure 5-6 Returning a Memory Address in a Tcl Object Method

Figure 5-7 contains a Tcl script snippet invoking the `GetImageAddr` subcommand. The resulting string address is processed as any other value resulting from a Tcl command. Figure 5-8 shows configuration data structure and method for the `TkImageItem` which receives the image address as the `-address` option value.

```
int
TclSampler::ProcessGetLastSample(
    Tcl_Interp •        interp,
    int                 argc,
    char •              argv[]
    )
{
  int                             status = TCL_OK;
  char •                          buffer;
  int                             i;
  SampleList::SampleBlock *       bptr;

        // Retrieve Last Sample From List
        //
if ( (bptr = (SampleList::SampleBlock *) sSamplesList.LastItem()) == NULL )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error retrieving last sample block", TCL_STATIC );
  }
        // Determine Number of Samples, Allow 128 Bytes For Each
        //
else if ( (buffer = (char *) malloc( bptr->sbSize * 128 )) == NULL )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error allocating return array", TCL_STATIC );
  }

else
  {
  buffer[0] = 0;

  for ( i = 0; i < bptr->sbSize; i++ )
    {
    sprintf(
        buffer,
        "%s { %d %d %d %d %hd %g }",
        buffer,
        bptr->sbPtr[i].sIndexes[0],
        bptr->sbPtr[i].sIndexes[1],
        bptr->sbPtr[i].sIndexes[2],
        bptr->sbPtr[i].sIndexes[3],
        bptr->sbPtr[i].sTag,
        bptr->sbPtr[i].sValue
        );
    }

  Tcl_SetResult( interp, buffer, TCL_DYNAMIC );
  }

return  status;
} // TclSampler::ProcessGetLastSample
```

Figure 5-5 Return of Array Data in Tcl Object Method

```
if { $res(showBins) } \
  {
  DisplayPoints $res(canvas) [$res(sampler) GetLastSample];
  }
    .
    .
$res(canvas) create image 0 0 \
    -tags sampleimage \
    -address [$res(sampler) GetImageAddr];
```

Figure 5-7 Tcl Code Snippets

```
Tk_ConfigSpec           TkImageItem::_iiConfigSpecs[] =
  {
    { TK_CONFIG_CUSTOM, •-tags•, (char *) NULL, (char *) NULL,
        (char *) NULL, 0, TK_CONFIG_NULL_OK, &tkCanvasTagsOption },
    { TK_CONFIG_STRING, •-address•, (char *) NULL, (char *) NULL,
        (char *) NULL, Tk_Offset(TkImageItem, iiImageAddr), 0, 0 },
    { TK_CONFIG_END, (char *) NULL, (char *) NULL, (char *) NULL,
        (char *) NULL, 0, 0 }
  };
.
.
.
int
TkImageItem::Configure(
    Tk_Canvas *         canvas_ptr,
    Tk_Item *           item_ptr,
    int                 argc,
    char •              argv[],
    int                 flags
    )
{
  register TkImageItem *        image_ptr = (TkImageItem *) item_ptr;
  int                           status;


status = Tk_ConfigureWidget(
    canvas_ptr->interp,
    canvas_ptr->tkwin,
    _iiConfigSpecs,
    argc,
    argv,
    (char *) image_ptr,
    flags
    );

if ( status == TCL_OK )
  {
  image_ptr->iiImage = (
      image_ptr->iiImageAddr != NULL ?
        (XImage *) atol( image_ptr->iiImageAddr ) : NULL
      );

  image_ptr->ComputeBoundingBox();
  }

return  status;
} // TkImageItem::Configure
```

Figure 5-8 Configuration for Processing a Memory Address

6. COPROCESSES

Tcl/Tk provides a very effective means for quickly generating an X Window interface with interactive widgets such as push buttons, text fields, and radio buttons. Graphics can be generated rather quickly as well with `canvas` widgets. However, graphics requiring a lot of operations (e.g., drawing thousands of dots) can suffer from poor performance with command interpretation in Tcl. Intensive graphic applications, such as image manipulations, using Xlib calls will all but require implementation in C/C++. There are two development options in such a case: implement the capabilities as Tcl commands, or implement the functionality in a separate process. Even if the latter option is chosen, it is desirable to build Tcl/Tk interfaces to control such applications. Further, there are utilities and applications which use standard input and output to receive textual commands and return results. Tcl/Tk interfaces to such processes prove useful as well.

The second alternative for such interfaces uses coprocesses, two cooperating processes in a master-slave relationship. The slave process reads commands from standard input, performs required operations (e.g., drawing graphics to an X window) and returns information to standard output. The master directs the slave by writing its input and reading its output.

In this section we present a simple example of a Tcl/Tk master process and a slave written in C++ and describe the technique of embedding windows. Source listings are given in Appendix F.

6.1 SLAVE PROCESS

By design, slave processes read commands from standard input. Clearly, the command language must be developed to meet the needs of the master and slave. Our example slave reads an X Window dump file, permutes the points, and renders a specified number of points from the permuted image. Each new line of standard input is parsed as a command, in this case merely the number of points to display and a scale factor.

Slaves controlling windows must also respond to X events. Figure 6-1 shows a portion of `permute_view.cc` which handles both input sources. A call to `select()` with read mask bits set for standard input (0) and the X display connection file descriptor blocks until either becomes active on input. X events are given priority.

6.2 MASTER SCRIPT

The Tcl/Tk script is the master and spawns the slave with a pipe for communication between them. Figure 6-2 shows the script fragments for spawning the slave process. Tcl opens pipes for standard input, output, and error. Thereafter, input from the file object reads from the slave's output, and output to the file object writes to the slave's input. Note that information flows in both directions. Figure 6-3 shows the `WriteValue` procedure which outputs to the slave. The `flush` command is necessary in order for the slave to receive the output immediately.

```
for ( finished = False; !finished; )
  {
    fd_set        read_fds;

  FD_ZERO( &read_fds );
  FD_SET( display->fd, &read_fds );
  FD_SET( 0, &read_fds );

  switch(
      select( getdtablesize(), &read_fds, NULL, NULL, NULL )
      )
    {
    case -1:
      perror( "select() error" );
      break;

    case 0:
      break;

    default:
      if ( FD_ISSET( display->fd, &read_fds ) )
        {
          XEvent  event;

        XNextEvent( display, &event );

        if (
            (event.type == Expose || event.type == ConfigureNotify ||
             event.type == MapNotify) &&
            event.xexpose.window == xwin.GetWindow()
            )
          sample_image.Put( xwin.GetWindow() );
        }

      else if (FD_ISSET( 0, &read_fds ) )
        {
          char    line[ 80 ];
          char *  token;
          int     new_size;

        if ( fgets( line, sizeof(line), stdin ) )
          {
          if ( (token = strchr( line, '/' )) != NULL )
            *token++ = 0;

          if ( sscanf( line, "%d", &new_size ) > 0 )
            sample_size = new_size;

          if ( sscanf( token, "%d", &new_size ) > 0 )
            pt_size_factor = new_size;

          if ( sample_size == 0 )
            finished = True;

          else
            {
            ReadSample( points, sample_image, total_points, sample_size, pt_size_factor );

            sample_image.Put( xwin.GetWindow() );
            }
          }
        }
    } // switch
  } // for
```

Figure 6-1 Handling X Events and Standard Input

```
set cmd "|/usr2/src/XObjects/permute_view [lindex $argv 0]";

if { [catch { set xpipe [open $cmd "r+"] }] } \
  {
  puts stderr "Error spawning permute_view"
  } \
else \
  {
  set value_list [ReadSizes $xpipe];
  puts stdout "... $value_list";
  }
```

Figure 6-2 Spawning the Slave from Tcl

```
proc  WriteValue { fid { one 0 } { two 0 } } \
{
if { "$one" == "0" } \
  {
  set factor [set size 0]
  } \
else \
  {
  set size [ $one get ];
  set factor [ $two get ];
  }

puts stdout "$size/$factor";
puts $fid "$size/$factor";
flush $fid;
}
```

Figure 6-3 Writing Output to the Slave

```
if ( (parent_str = getenv("PARENT_WINDOW")) == NULL )
  parent = RootWindow( theDisplay, theScreen );
else
  sscanf( parent_str, "%lx", &parent );

theWindow = XCreateWindow(
    theDisplay,
    parent,
    0, 0,
    640, 480,
    0, 0,
    BlackPixel( thedisplay, theScreen )
    );
```

Figure 6-4 Slave Process Checking for Parent Window Handle

```
frame .frame -width 642 -height 482 -borderwidth 2 -relief sunken;
pack .frame -side top;
update;

set env(PARENT_WINDOW) [winfo id .frame];
set fid [open "|/home/re7/X/xwin" "r"];

bind .frame <Destroy> "exec kill [pid $fid]";
```

Figure 6-5 Tcl/Tk Commands to Embed Slave Window

6.3 EMBEDDING WINDOWS

The `permute_view` slave creates its own window as a child of the root window of the X display. Thus, the Tcl/Tk window and slave process window are separate on the screen. One may embed the slave window within a Tcl/Tk window very simply. Tk provides the `winfo` command to provide information on the X windows it controls. The `id` subcommand returns a string with the hexadecimal value of a window's handle. This can be passed to an X window slave to be used as the parent parameter when creating a window. Figure 6-4 lists a code fragment of a slave process which checks the environment for the value of parent window handle. Figure 6-5 gives sample Tcl/Tk commands to embed the slave window within a frame widget and ensure proper termination of the slave process when the widget is destroyed.

7. SUMMARY

Tcl/Tk is a powerful and effective development environment for building Motif™ X Window interfaces. Prototype efforts are especially aided by the ease with which Tcl/Tk scripts can be generated.

Extensibility allows the developer to embed new functionality within the language. Developers may program in the scripting language, extend the Tk widget set and drawing constructs, wrap Tcl interfaces around 3GL code, and build Tcl/Tk and 3GL coprocesses. However, the scripting language is limited in its support of structured development, and its use should probably be limited to interface objects and constructs. C++'s inheritance feature is especially useful in building Tcl classes on top of existing classes.

REFERENCES

1. Booch, G., Object Oriented Design With Applications, Benjamin/Cummings, California, 1991.

2. Booch, G., and Bryan D., Software Engineering With Ada, 3rd Edition, Benjamin/Cummings, California, 1994.

3. Cox, B.J., Object Oriented Programming: An Evolutionary Approach, Addison-Wesley, Massachusetts, 1987.

4. Lee, R.W., and R. Hume, Geoserver: "A Dynamic Geographic Presentation System," *Proceedings of the 1992 ACM Symposium on Applied Computing*, Vol. 3, pp. 1265-1273.

5. Nye A. (editor), Xlib Reference Manual, O'Reilly & Associates, 1992.

6. Open Software Foundation, OSF/Motif™ Release 1.2 Style Guide, Prentice Hall, Englewood Cliffs, NJ, 1993.

7. Ousterhout, J.K., Tcl and the Tk Toolkit, Addison-Wesley, 1993.

APPENDIX A. TCL/TK SOURCE LISTINGS

A.1 BARWINDOW.TCL

```
#-      %Z% %M%  %I% %G%
#------------------------------------------------------------------------
#-      NAME:           BarWindow.tcl                                   -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PACKAGE:                                                        -
#-              BarWindow                                               -
#-      EXPORTED PROCEDURES:                                            -
#-              BarWindow_Create                                        -
#-              BarWindow_LoadFile                                      -
#-              BarWindow_LoadCommandResult                             -
#-      USAGE:                                                          -
#-              BarWindow_LoadFile [BarWindow_Create] filename          -
#-                                                                      -
#-              BarWindow_LoadCommandResult                             -
#-                      [BarWindow_Create] fdb-client-object            -
#------------------------------------------------------------------------


#------------------------------------------------------------------------
#-      Class Resource Values                                           -
#------------------------------------------------------------------------
option add *BarWindow*background gray75;
option add *BarWindow*foreground navy;
option add *BarWindow*barBackground gray75;
option add *BarWindow*plotBackground gray90;
option add *BarWindow*colors \
  { #ef0000 #00bf00 #dfdf00 #0000ff #00afdf orange #df00ff };
option add *BarWindow*plotBackground gray90;


global BarWindow;

#------------------------------------------------------------------------
#-      Class Attributes                                                -
#------------------------------------------------------------------------
set BarWindow(count) 0;


#------------------------------------------------------------------------
#-      Object Attributes                                               -
#-              Chart           name of the graph widget                -
#-              Data            list of observation lists, each with    -
#-                              the name of the observation and values  -
#-                              for each series                         -
#-              FileName        name of temp file containing data       -
#-              Series          list of series names                    -
#-              Stacked         if "true", each observation is a stack  -
#-                              of series value bars                    -
#-              Title           chart title                             -
#-              TopWindow       name of the toplevel widget which is    -
#-                              the window                              -
#-              XTitle          x-axis title                            -
#-              YTitle          y-axis title                            -
#------------------------------------------------------------------------


#------------------------------------------------------------------------
#-      NAME:           BarWindow_CreateSpreadWindow()                  -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Creates a SpreadWindow object from the data file        -
```

```
#------------------------------------------------------------------------
proc  BarWindow_CreateSpreadWindow { this } \
{
  global BarWindow;

SpreadWindow_LoadFile [SpreadWindow_Create] $BarWindow($this,FileName);
}
# BarWindow_CreateSpreadWindow


#------------------------------------------------------------------------
#-      NAME:           BarWindow_CreateMenuBar()                       -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#------------------------------------------------------------------------
proc  BarWindow_CreateMenuBar { parent this } \
{
  global BarWindow;

set bg [option get . menuBackground Tk];
set active_bg [option get . menuActiveBackground Tk];
set fg [option get . menuForeground Tk];
set font [option get . menuFont Tk];
set button_font [option get . menuButtonFont Tk];

frame $parent.menubar \
    -relief raised \
    -background $bg \
    -borderwidth 2;

#       -- File Button
menubutton $parent.menubar.file \
    -text "File" \
    -underline 0 \
    -padx 10 \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -font $button_font \
    -foreground $fg \
    -menu $parent.menubar.file.menu;

menu $parent.menubar.file.menu \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -borderwidth 2 \
    -foreground $fg \
    -font $font;

#               -- File Menu
$parent.menubar.file.menu add command \
    -state disabled \
    -label "Save" \
    -underline 0 \
    -command { puts stdout "Not yet implemented" };

$parent.menubar.file.menu add separator;

$parent.menubar.file.menu add command \
    -label "Exit" \
    -underline 1 \
    -background #7f0000 \
    -activebackground #af0000 \
    -command "destroy $parent";

#       -- Display Button
menubutton $parent.menubar.display \
    -text "Display" \
    -underline 0 \
    -padx 10 \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -font $button_font \
    -foreground $fg \
    -menu $parent.menubar.display.menu;

menu $parent.menubar.display.menu \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -borderwidth 2 \
    -foreground $fg \
    -font $font;

#               -- Display Menu
$parent.menubar.display.menu add command \
    -label " Spread Sheet " \
    -underline 1 \
    -command "BarWindow_CreateSpreadWindow $this";

#       -- Set Up Menu
pack \
    $parent.menubar.file \
    $parent.menubar.display \
    -side left \
    -ipadx 5 \
    -ipady 2;

bind $parent <Any-FocusIn> \
  "
  if { (\"%d\" == \"NotifyVirtual\") && (\"%m\" == \"NotifyNormal\") } \
    \"focus $parent.menubar\"
  "

tk_menuBar $parent.menubar \
    $parent.menubar.file;

return  $parent.menubar;
}
# BarWindow_CreateMenuBar


#------------------------------------------------------------------------
#-      NAME:           BarWindow_CreateGraph()                         -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Creates the graph widget                                -
#------------------------------------------------------------------------
proc  BarWindow_CreateGraph ( this ) \
{
  global BarWindow;

set top $BarWindow($this,TopWindow);
set chart $top.chart;

blt_barchart $chart \
    -background [option get $top background Tk] \
    -foreground [option get $top foreground Tk] \
    -plotbackground [option get $top plotBackground Tk] \
    -width 800 -height 600;

$chart legend configure -background [option get $top plotBackground Tk];

return  $chart;
```

```
}
# BarWindow_CreateGraph


#------------------------------------------------------------------------
#-      NAME:           BarWindow_Create()                              -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Creates a BarWindow object                              -
#------------------------------------------------------------------------
proc  BarWindow_Create () \
{
  global BarWindow;

#       -- Set Name of This Object
#       --
set this barwin$BarWindow(count);

#       -- Name of This Window
#       --
set name .bwtop$BarWindow(count);
set BarWindow($this,TopWindow) $name;

#       -- Build Toplevel
#       --
toplevel $name -class BarWindow;
wm iconbitmap $name @[option get $name library BarWindow]/barchart.xbm
wm iconmask $name @[option get $name library BarWindow]/barchart.xbm
wm iconname $name "BarChart - $BarWindow(count)";
wm minsize $name 300 200;
wm title $name "BarChart Window - $BarWindow(count)";

#       -- Build Frames
#       --
BarWindow_CreateMenuBar $name $this;
set chart_name [BarWindow_CreateGraph $this];
set BarWindow($this,Chart) $chart_name;

#       -- Pack Frames
#       --
pack $name.menubar -side top -fill x;
pack $chart_name -side top -expand yes -fill both;

#       -- Increment Window Count
#       --
incr BarWindow(count);

return  $this;
}
# BarWindow_Create


#------------------------------------------------------------------------
#-      NAME:           BarWindow_DrawXLabel()                          -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Procedure called to label the x-axis                    -
#------------------------------------------------------------------------
proc  BarWindow_DrawXLabel ( this w ndex ) \
{
  global BarWindow;

set ndex [lindex [split $ndex "."] 0];
return  [lindex [lindex $BarWindow($this,Data) [expr $ndex - 1]] 0];
}
# BarWindow_DrawXLabel


#------------------------------------------------------------------------
#-      NAME:           BarWindow_BuildElements()                       -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Builds the observation-series bars                      -
#-              Creates the graph widget                                -
#-----------------------------------------------------------------------
proc  BarWindow_BuildElements { this } \
{
  global BarWindow;

set BarWindow($this,Series) [split $BarWindow($this,Series) ":"];

set top $BarWindow($this,TopWindow);
set chart $BarWindow($this,Chart);

$chart xaxis configure \
    -command "BarWindow_DrawXLabel $this" \
    -font 9x15 \
    -title $BarWindow($this,XTitle);

$chart yaxis configure \
    -font 9x15 \
    -title $BarWindow($this,YTitle);

$chart configure \
    -title $BarWindow($this,Title);

#       -- Build Dummy Legend Elements
set sindex 0;
foreach ser $BarWindow($this,Series) \
  {
  $chart element create dummy_$(ser) \
      -data "1 0" \
      -foreground [lindex [option get $top colors Tk] $sindex] \
      -label $ser;
  incr sindex;
  }

#       -- Build Elements for Each Observation
set bg [option get $top barBackground Tk];
set oindex 0;
foreach obs $BarWindow($this,Data) \
  {
  $chart element create el_$(oindex) \
      -xdata [expr $oindex + 1] \
      -ydata [lrange $obs 1 end] \
      -background $bg \
      -stacked $BarWindow($this,Stacked) \
      -bd 1 \
      -relief raised \
      -fg [option get $top colors Tk] \
      -label "";
  incr oindex;
  }
}
# BarWindow_BuildElements


#-----------------------------------------------------------------------
#-      NAME:           BarWindow_ParseAttrs()                          -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Parse the attribute line in the data stream             -
#-----------------------------------------------------------------------
proc  BarWindow_ParseAttrs { this line } \
{
  global BarWindow;

foreach piece [split $line "\t"] \
  {
  set parts [split $piece "="];
  set BarWindow($this,[lindex $parts 0]) [lindex $parts 1];
  }
}
# BarWindow_ParseAttrs


#-----------------------------------------------------------------------
#-      NAME:           BarWindow_LoadFile()                            -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Populate the data for the chart from the data file      -
#-----------------------------------------------------------------------
proc  BarWindow_LoadFile { this filename } \
{
  global BarWindow;

#       -- Initialize Attributes
#       --
set BarWindow($this,Title) "Could not open file $filename";
set BarWindow($this,XTitle) "";
set BarWindow($this,YTitle) "";
set BarWindow($this,Stacked) "true";
set BarWindow($this,Series) "";
set BarWindow($this,Data) {};
set BarWindow($this,FileName) "";

#       -- Open File
#       --
if { [catch { set fid [open $filename "r"] }] == 0 } \
  {
  set BarWindow($this,FileName) $filename;

#               -- Read Attribute Line
#               --
  gets $fid line;
  BarWindow_ParseAttrs $this $line;

  set BarWindow($this,Data) {};

#               -- Read Observation Lines
#               --
  while { [gets $fid line] > -1 } \
    {
    lappend BarWindow($this,Data) [split $line "\t"];
    }
  }

BarWindow_BuildElements $this;
update;
}
# BarWindow_LoadFile


#-----------------------------------------------------------------------
#-      NAME:           BarWindow_LoadCommandResult()                   -
#-      HISTORY:                                                        -
#-              14 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Populate the data for the chart from the network stream -
#-              resulting from a command to the server                  -
#-----------------------------------------------------------------------
proc  BarWindow_LoadCommandResult { this fdb } \
{
  global Fdb BarWindow;

#       -- Set Data File Name
#       --
set filename "/tmp/[option get . filePrefix ""]$this";
if { [catch { set fid [open $filename "w"] }] } \
  {
  set BarWindow($this,FileName) "";
  puts stderr "Error creating $filename";
  } \
else \
  {
  set BarWindow($this,FileName) $filename;
  }

#       -- Initialize Attributes
#       --
set BarWindow($this,Title) "Error reading results";
set BarWindow($this,XTitle) "";
set BarWindow($this,YTitle) "";
set BarWindow($this,Stacked) "true";
set BarWindow($this,Series) "";
set BarWindow($this,Data) {};

#       -- Read Attribute Line
#       --
if { "[$fdb ReadNextLine line]" != "eof" } \
  {
  catch { puts $fid $line };
  BarWindow_ParseAttrs $this $line;

  set BarWindow($this,Data) {};

#               -- Read Observation Lines
#               --
  while { "[$fdb ReadNextLine line]" != "eof" } \
    {
    catch { puts $fid $line };
    lappend BarWindow($this,Data) [split $line "\t"];
    }
  }

catch { close $fid };

BarWindow_BuildElements $this;
update;
}
# BarWindow_LoadCommandResult
```

A.2 FDB.TCL

```
#!fdb_wish -f
#-      %Z% %M%  %I% %G%
#-----------------------------------------------------------------------
#-      NAME:           Fdb.tcl                                        -
#-      HISTORY:                                                       -
#-              22 Apr 94       re7@ornl.gov                           -
#-              Many, many changes and additions                       -
#-              23 Mar 94       re7@ornl.gov                           -
#-      PURPOSE:                                                       -
#-              Tcl driver for FDB X11 interface                       -
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
#-      Add FDB_DIR to the library search path                         -
#-----------------------------------------------------------------------
if { [info exists env(FDB_DIR)] } \
  {
  set Fdb_library $env(FDB_DIR);
  } \
else \
  {
  set Fdb_library /usr/local/lib/fdb;
  }

set tcl_library [info library];
source $tcl_library/init.tcl
set auto_path "$tcl_library $tk_library $blt_library $Fdb_library";
```

```
set tcl_precision 10;


#-----------------------------------------------------------------------
#-      Set Resources For Imported Libraries                           -
#-----------------------------------------------------------------------
option add *BarWindow*library $Fdb_library;
option add *SpreadWindow*library $Fdb_library;
option add *TextWindow*library $Fdb_library;


#-----------------------------------------------------------------------
#-      Set Resources                                                  -
#-----------------------------------------------------------------------
option add Tk*background                gray50                  startupFile;
option add Tk*cancelForeground          #00007f                 startupFile;
option add Tk*filePrefix                fdbdata                 startupFile;
option add Tk*foreground                #cfcf00                 startupFile;
option add Tk*menuActiveBackground      #0030a0                 startupFile;
option add Tk*menuBackground            #0040b0                 startupFile;
option add Tk*menuForeground            white                   startupFile;
option add Tk*menuButtonFont            *courier-bold-r*14*     startupFile;
option add Tk*menuFont                  *courier-medium-r*14*   startupFile;
option add Tk*messageBackground         gray50                  startupFile;
option add Tk*messageFont               *times-medium-i*18*     startupFile;
option add Tk*messageForeground         #ffcf00                 startupFile;
option add Tk*okayForeground            #7f0000                 startupFile;
option add Tk*queryActiveBackground     #a2491e                 startupFile;
option add Tk*queryBackground          #b2591e               startupFile;
option add Tk*queryForeground          yellow                startupFile;
option add Tk*reportActiveBackground   #006f00               startupFile;
option add Tk*reportBackground         #008f00               startupFile;
option add Tk*reportForeground         yellow                startupFile;
option add Tk*statusFont               *helvetica-medium-r*14* startupFile;
option add Tk*statusBackground         DarkGreen             startupFile;
option add Tk*statusForeground         yellow                startupFile;
option add Tk*toolActiveBackground     gray65                startupFile;
option add Tk*toolBackground           gray75                startupFile;
option add Tk*toolFont                 *courier-bold-r*14*   startupFile;
option add Tk*toolForeground           #0000af               startupFile;
option add Tk*toolbarBackground        gray50                startupFile;
option add Tk*Font                     *courier-bold-r*14*   startupFile;
option add Tk*Frame*background         gray25                startupFile;


#-------------------------------------------------------------------------
#-      Set global resource values                                       -
#-------------------------------------------------------------------------
global Fdb;
set Fdb(menuBar) .menubar;
set Fdb(toolBar) .toolbar;
set Fdb(message) .message;
set Fdb(statusBar) .statusbar;
set Fdb(fdbClient) fdb;

set Fdb(Queries) \
  {
    { Budget { "Cost Center Account" } "Cost Center Budget" spread }
    { CCSummary { "Cost Center Account" } "Cost Center Summary" bar }
  };

set Fdb(Reports) \
  {
    { Test {"First Parameter" "Second Parameter"} "Parameter Test" text }
    { CCSummary { "Cost Center Account Substring" }
      "Cost Center Summary" text }
    { CCDummy {} "Current Assumptions Example" text }
  };

set Fdb(Tools) \
  {
    { budget.xbm "FindQuery CCSummary" "" }
    { exit.xbm  "DoExit" "Exit the application" }
  };


#-------------------------------------------------------------------------
#-      NAME:           DoExit()                                         -
#-------------------------------------------------------------------------
proc  DoExit {} \
{
  global Fdb Fdb_library;

catch { exec $Fdb_library/Clean [option get . filePrefix ""] };
exit;
}
# DoExit


#-------------------------------------------------------------------------
#-      NAME:           FindQuery()                                      -
#-------------------------------------------------------------------------
proc  FindQuery { qname } \
{
  global Fdb;

foreach x $Fdb(Queries) \
  {
  if { "[lindex $x 0]" == "$qname" } \
    "
    DoCommand query [list $x];
```

```
    break;
    "
  }
}
# FindQuery


#-------------------------------------------------------------------------
#-      NAME:           Popup()                                          -
#-------------------------------------------------------------------------
proc  Popup { msg } \
{
  global Fdb;

#       -- Top Level Window
toplevel .poptop -class Popup;

wm title .poptop "Dialog";
wm geometry .poptop +300+300;

#       -- Message
message .poptop.message \
    -borderwidth 1 \
    -relief sunken \
    -justify center \
    -width 500 \
    -text $msg;

#       -- Buttons
frame .poptop.button_frame;

button .poptop.button_frame.ok \
    -text "OK" \
    -background [option get . toolBackground Tk] \
    -foreground [option get . okayForeground Tk] \
    -activeforeground [option get . okayForeground Tk] \
    -activebackground [option get . toolActiveBackground Tk] \
    -font [option get . toolFont Tk] \
    -command { destroy .poptop };

pack \
    .poptop.button_frame.ok \
    -side left \
    -padx 20 \
    -ipadx 2 \
    -ipady 2;

pack \
    .poptop.message \
    .poptop.button_frame \
    -side top \
    -pady 5;

#       -- Let Dialog Window Come Up
tkwait visibility .poptop;
grab .poptop;
tkwait window .poptop;
}
# Popup


#-------------------------------------------------------------------------
#-      NAME:           RunDialog()                                      -
#-------------------------------------------------------------------------
proc  RunDialog { dlist } \
{
  global Fdb dialog_values run_flag;

#       -- Top Level Window
toplevel .top \
      -background [option get . toolBackground Tk];

wm title .top "Parameter Dialog";
wm geometry .top +300+300;

#       -- Entry Frames
set pindex 0;
foreach param $dlist \
  {
  frame .top.frame_$pindex \
      -background [option get . toolBackground Tk];

  label .top.frame_$pindex.label \
      -text [concat $param] \
      -background [option get . toolBackground Tk] \
      -foreground [option get . toolForeground Tk] \
      -borderwidth 1 \
      -relief flat;

  entry .top.frame_$pindex.entry \
      -borderwidth 1 \
      -relief sunken \
      -background [option get . toolBackground Tk] \
      -foreground [option get . toolForeground Tk] \
      -text dialog_values($pindex) \
      -width 20;

  pack \
      .top.frame_$pindex.label \
      -side left \
      -ipadx 2 \
      -ipady 2;

  pack \
      .top.frame_$pindex.entry \
      -side right \
      -ipadx 2 \
      -ipady 2;

  pack .top.frame_$pindex -side top -fill x -pady 5;

  incr pindex;
  }

#               -- Set Key Bindings
for { set i 0 } { $i < $pindex } { incr i } \
  {
  set next_guy [expr [expr $i + 1] % $pindex];
  bind .top.frame_$i.entry <Tab> \
    "
    focus .top.frame_$next_guy.entry;
    ";
  bind .top.frame_$i.entry <Return> \
    {
    set run_flag 1;
    destroy .top;
    };
  bind .top.frame_$i.entry <Escape> \
    {
    set run_flag 0;
    destroy .top;
    };
  }

#       -- Buttons
frame .top.button_frame \
      -background [option get . toolBackground Tk];

button .top.button_frame.ok \
    -text "OK" \
    -background [option get . toolBackground Tk] \
    -foreground [option get . okayForeground Tk] \
    -activeforeground [option get . okayForeground Tk] \
    -activebackground [option get . toolActiveBackground Tk] \
    -font [option get . toolFont Tk] \
    -command { set run_flag 1; destroy .top };
```

```
button .top.button_frame.cancel \
    -text "Cancel" \
    -background [option get . toolBackground Tk] \
    -foreground [option get . cancelForeground Tk] \
    -activeforeground [option get . cancelForeground Tk] \
    -activebackground [option get . toolActiveBackground Tk] \
    -font [option get . toolFont Tk] \
    -command { set run_flag 0; destroy .top };

pack \
    .top.button_frame.ok \
    .top.button_frame.cancel \
    -side left \
    -padx 20 \
    -ipadx 2 \
    -ipady 2;

pack .top.button_frame -side top -pady 5;

#       -- Let Dialog Window Come Up
tkwait visibility .top;
focus .top.frame_0.entry;
grab .top;
tkwait window .top;

set result "";
if { $run_flag } \
  {
  for { set x 0 } { $x < $pindex } { incr x } \
    {
    lappend result $dialog_values($x);
    }
  }

return  $result;
}
# RunDialog


#------------------------------------------------------------------------
#-      NAME:           DoCommand()                                     -
#------------------------------------------------------------------------
proc  DoCommand { type qlist } \
{
  global Fdb;

#       -- Initialize Command String
#       --
set cmd_string "[lindex $qlist 0].$type";

#       -- Check For Parameters
#       --
if { [llength [set param_names [lindex $qlist 1]]] > 0 } \
  {
  set param_list [RunDialog $param_names];

  if { "$param_list" != "" } \
    {
    foreach p $param_list \
      {
      set cmd_string "$cmd_string \"$p\"";
      }
    } \
  else \
    {
    set cmd_string "";
    }
  }

if { "$cmd_string" != "" } \
  {
  puts stdout "Running command: $cmd_string";
  WriteStatus "Sending command to FDB server ...";
  $Fdb(fdbClient) SendCommand $cmd_string;

  if { [$Fdb(fdbClient) GetStatus] != 0 } \
    {
    WriteStatus "Send failed !!";
    } \
  else \
    {
    WriteStatus "Reading query results ...";

    set wtype [lindex $qlist 3];

    if { "$wtype" == "spread" } \
      {
      SpreadWindow_LoadCommandResult [SpreadWindow_Create] $Fdb(fdbClient);
      } \
    elseif { "$wtype" == "bar" } \
      {
      BarWindow_LoadCommandResult [BarWindow_Create] $Fdb(fdbClient);
      } \
    else \
      {
      TextWindow_LoadCommandResult [TextWindow_Create] $Fdb(fdbClient);
      }

    WriteStatus "Query complete";
    }
  }
}
# DoCommand


#---------------------------------------------------------------------
#-      NAME:           CreateMenuBar()                             -
#---------------------------------------------------------------------
proc  CreateMenuBar {} \
{
  global       Fdb;

set bg [option get . menuBackground Tk];
set active_bg [option get . menuActiveBackground Tk];
set fg [option get . menuForeground Tk];
set font [option get . menuFont Tk];
set button_font [option get . menuButtonFont Tk];

frame .menubar \
    -relief raised \
    -background $bg \
    -borderwidth 2;

#               -- Session button
menubutton .menubar.session \
    -text "Session" \
    -underline 0 \
    -padx 10 \
    -font $button_font \
    -background $bg \
    -activebackground $active_bg \
    -foreground $fg \
    -activeforeground $fg \
    -menu .menubar.session.menu;

menu .menubar.session.menu \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -borderwidth 2 \
    -foreground $fg \
    -font $font;

#                      -- Session menu
.menubar.session.menu add command \
```

```
    -label "Text Window" \
    -command { TextWindow_LoadFile [TextWindow_Create] "|last -100" };

.menubar.session.menu add command \
    -label "Message of the Day" \
    -command \
  {
    global Fdb;

  WriteStatus "Sending command to FDB server ...";
#  $Fdb(fdbClient) SendCommand "cat /etc/motd";
  $Fdb(fdbClient) SendCommand "cat.cmd /etc/motd"

  WriteStatus "Reading server response ...";
  TextWindow_LoadCommandResult [TextWindow_Create] $Fdb(fdbClient);
  WriteStatus "Dialog complete";
  }

.menubar.session.menu add separator;

.menubar.session.menu add command \
    -label "Exit" \
    -underline 1 \
    -background #7f0000 \
    -activebackground #af0000 \
    -command { DoExit };

#               -- Queries button
menubutton .menubar.queries \
    -text "Queries" \
    -underline 0 \
    -padx 10 \
    -font $button_font \
    -background $bg \
    -activebackground $active_bg \
    -foreground $fg \
    -activeforeground $fg \
    -menu .menubar.queries.menu;

menu .menubar.queries.menu \
    -font $font \
    -borderwidth 2 \
    -background [option get . queryBackground Tk] \
    -foreground [option get . queryForeground Tk] \
    -activeforeground [option get . queryForeground Tk];

#                      -- Queries menu
foreach rep $Fdb(Queries) \
  {
  .menubar.queries.menu add command \
      -label [lindex $rep 2] \
      -background [option get . queryBackground Tk] \
      -activebackground [option get . queryActiveBackground Tk] \
      -command "DoCommand query [list $rep]";
  }

#               -- Reports button
menubutton .menubar.reports \
    -text "Reports" \
    -underline 0 \
    -padx 10 \
    -font $button_font \
    -background $bg \
    -activebackground $active_bg \
    -foreground $fg \
    -activeforeground $fg \
    -menu .menubar.reports.menu;

menu .menubar.reports.menu \
    -font $font \
    -borderwidth 2 \
    -background [option get . reportBackground Tk] \
    -foreground [option get . reportForeground Tk] \
    -activeforeground [option get . reportForeground Tk];

#                       -- Reports menu
foreach rep $Fdb(Reports) \
  {
  .menubar.reports.menu add command \
      -label [lindex $rep 2] \
      -background [option get . reportBackground Tk] \
      -activebackground [option get . reportActiveBackground Tk] \
      -command "DoCommand report [list $rep]";
  }

#               -- Set Up Menu
pack \
    .menubar.session \
    .menubar.queries \
    .menubar.reports \
    -side left \
    -ipadx 5 \
    -ipady 2;

bind . <Any-FocusIn> \
  {
  if ( ("%d" == "NotifyVirtual") && ("%m" == "NotifyNormal") ) \
    ( focus .menubar )
  }

tk_menuBar .menubar \
    .menubar.session \
    .menubar.queries \
    .menubar.reports;
}
# CreateMenuBar


#---------------------------------------------------------------------
#-      NAME:          CreateToolBar()                               -
#---------------------------------------------------------------------
proc  CreateToolBar () \
{
  global        Fdb Fdb_library;

frame .toolbar \
    -borderwidth 2 \
    -relief sunken \
    -background [option get . toolbarBackground Tk];

set count 0;
foreach tool $Fdb(Tools) \
  {
  button .toolbar.tool$count \
      -bitmap @$Fdb_library/[lindex $tool 0] \
      -background [option get . toolBackground Tk] \
      -activebackground [option get . toolActiveBackground Tk] \
      -foreground [option get . toolForeground Tk] \
      -activeforeground [option get . toolForeground Tk] \
      -command [lindex $tool 1];

#  bind .toolbar.tool$count <Enter> +"WriteStatus \"[lindex $tool 2]\"";

  pack .toolbar.tool$count \
      -side left \
      -padx 2 \
      -pady 2;

  incr count;
  }
}
# CreateToolBar


#---------------------------------------------------------------------
#-      NAME:          CreateMessage()                               -
#---------------------------------------------------------------------
proc  CreateMessage () \
{
  global        Fdb;

message .message \
    -background [option get . messageBackground Tk] \
    -font [option get . messageFont Tk] \
    -foreground [option get . messageForeground Tk] \
    -borderwidth 1 \
    -relief sunken \
    -justify center \
    -width 500 \
    -text "Finance Database Client Application";
}
# CreateMessage


#---------------------------------------------------------------------
#-      NAME:          CreateStatusBar()                             -
#---------------------------------------------------------------------
proc  CreateStatusBar () \
{
  global        Fdb;

message .statusbar \
    -font [option get . statusFont Tk] \
    -background [option get . statusBackground Tk] \
    -foreground [option get . statusForeground Tk] \
    -relief raised \
    -borderwidth 1 \
    -width 500 \
    -justify center \
    -text "";
}
# CreateStatusBar


#---------------------------------------------------------------------
#-      NAME:          WriteStatus()                                 -
#---------------------------------------------------------------------
proc  WriteStatus ( msg ) \
{
  global        Fdb;

$Fdb(statusBar) configure -text $msg;
update;
}
# WriteStatus


#---------------------------------------------------------------------
#-      NAME:          PackFrames()                                  -
#---------------------------------------------------------------------
proc  PackFrames () \
{
  global        Fdb;

pack \
    $Fdb(menuBar) \
    $Fdb(toolBar) \
    -side top \
    -fill x;

pack \
    $Fdb(message) \
    -side top \
    -fill x \
    -ipady 2 \
    -pady 2;

pack \
    $Fdb(statusBar) \
    -side top \
    -fill x \
    -pady 2;
}
# PackFrames


#-----------------------------------------------------------------------
#-      NAME:           main()                                         -
#-----------------------------------------------------------------------
wm withdraw .;

set auth_list [GetAuth];
FdbClient $Fdb(fdbClient) [lindex $auth_list 0] [lindex $auth_list 1];

if { "[fdb GetStatus]" != "0" } \
  {
  Popup "User Authentication Failed";
  exit;
  } \
else \
  {
  wm title . "FDB V%R%.%L%";
  wm iconbitmap . @$Fdb_library/fdb.xbm;
  wm iconmask . @$Fdb_library/fdb.xbm;
  wm geometry . +300+50;

  CreateMenuBar;
  CreateToolBar;
  CreateMessage;
  CreateStatusBar;
  PackFrames;

  WriteStatus "User Authentication Successful";

  wm deiconify .;
  }
```

A.3 SPREADSHEET.TCL

```
#-      %Z% %M%  %I% %G%
#-----------------------------------------------------------------------
#-      NAME:           SpreadSheet.tcl                                -
#-      HISTORY:                                                       -
#-              21 Apr 94       re7@ornl.gov                           -
#-      PACKAGE:                                                       -
#-              SpreadSheet                                            -
#-      USAGE:                                                         -
#-              set sheet [SpreadSheet_Create]                         -
#-              SpreadSheet_SetRowValues $sheet observation-list       -
#-----------------------------------------------------------------------

global SpreadSheet;

#-----------------------------------------------------------------------
#-      Class Attributes                                               -
#-----------------------------------------------------------------------
set SpreadSheet(count) 0;
set SpreadSheet(Bg) gray65;
set SpreadSheet(Fg) white;
set SpreadSheet(CellBg) gray80;
set SpreadSheet(CellFg) #0000ff;
set SpreadSheet(CellOutline) gray30;
set SpreadSheet(CellFont) 9x15;
set SpreadSheet(CellHalfWidth) 64;
set SpreadSheet(CellHalfHeight) 16;
set SpreadSheet(CellWidth) 128;
set SpreadSheet(CellHeight) 32;
set SpreadSheet(LabelBg) gray75;
set SpreadSheet(LabelFg) #af0000;
set SpreadSheet(LabelFont) 9x15bold;
set SpreadSheet(ScrollWidth) 16;
set SpreadSheet(ScrollHeight) 16;
set SpreadSheet(Font) 9x15;


#-----------------------------------------------------------------------
#-      Object Attributes                                              -
#-              Frame           frame containing all the canvases      -
#-              LeftCan         canvas containing row labels           -
#-              Matrix          canvas containing matrix cells         -
#-              TopCan          canvas containing column labels        -
#-----------------------------------------------------------------------


#-----------------------------------------------------------------------
#-      NAME:           SpreadSheet_HandleColBar()                     -
#-      HISTORY:                                                       -
#-              21 Apr 94       re7@ornl.gov                           -
#-----------------------------------------------------------------------
proc  SpreadSheet_HandleColBar { this col } \
{
  global SpreadSheet;

set can $SpreadSheet($this,TopCan);
set matrix $SpreadSheet($this,Matrix);

if { "[lindex [$can itemconfigure colbar_$col -relief] 4]" == "raised" } \
  {
  $can itemconfigure colbars -relief raised;
  $can itemconfigure colbar_$col -relief sunken;

#  $matrix itemconfigure cell_col_$col -outline $SpreadSheet(CellOutline);
  } \
else \
  {
  $can itemconfigure colbar_$col -relief raised;
  }
}
# SpreadSheet_HandleColBar


#---------------------------------------------------------------------
#-      NAME:           SpreadSheet_HandleRowBar()                  -
#-      HISTORY:                                                    -
#-              21 Apr 94       re7@ornl.gov                        -
#---------------------------------------------------------------------
proc  SpreadSheet_HandleRowBar { this row } \
{
  global SpreadSheet;

set can $SpreadSheet($this,LeftCan);
set matrix $SpreadSheet($this,Matrix);

if { "[lindex [$can itemconfigure rowbar_$row -relief] 4]" == "raised" } \
  {
  $can itemconfigure rowbars -relief raised;
  $can itemconfigure rowbar_$row -relief sunken;

#  $matrix itemconfigure cell_row_$row -outline $SpreadSheet(CellOutline);
  } \
else \
  {
  $can itemconfigure rowbar_$row -relief raised;
  }
}
# SpreadSheet_HandleRowBar


#---------------------------------------------------------------------
#-      NAME:           SpreadSheet_SetCellValues()                 -
#-      HISTORY:                                                    -
#-              21 Apr 94       re7@ornl.gov                        -
#-      PURPOSE:                                                    -
#-              Given a list of lists (one per row) containing cell -
#-              values, populates accordingly                       -
#---------------------------------------------------------------------
proc  SpreadSheet_SetCellValues { this value_list } \
{
  global SpreadSheet;

$SpreadSheet($this,LeftCan) delete celltexts;

set rindex 0;
set ypos $SpreadSheet(CellHalfHeight);
foreach row_list $value_list \
  {
  set cindex 0;
  set xpos $SpreadSheet(CellHalfWidth);
  foreach value $row_list \
    {
    $SpreadSheet($this,Matrix) create text \
        $xpos $ypos \
        -tags "celltexts celltext_$(rindex)_$(cindex)" \
        -anchor c \
        -font $SpreadSheet(CellFont) \
        -text $value \
        -fill $SpreadSheet(CellFg);
```

```
    incr xpos $SpreadSheet(CellWidth);
    incr cindex;
    }

  incr ypos $SpreadSheet(CellHeight);
  incr rindex;
  }
}
# SpreadSheet_SetCellValues


#---------------------------------------------------------------------
#-      NAME:           SpreadSheet_SetColNames()                   -
#-      HISTORY:                                                    -
#-              21 Apr 94       re7@ornl.gov                        -
#-      PURPOSE:                                                    -
#-              Given a list containing row label names, populates the -
#-              left canvas bars with the label strings as text     -
#---------------------------------------------------------------------
proc  SpreadSheet_SetColNames { this name_list } \
{
  global SpreadSheet;

$SpreadSheet($this,LeftCan) delete coltexts;

set cindex 0;
set xpos $SpreadSheet(CellHalfWidth);
foreach name $name_list \
  {
  $SpreadSheet($this,TopCan) create text \
      $xpos $SpreadSheet(CellHalfHeight) \
      -tags "coltexts coltext_$cindex" \
      -anchor c \
      -font $SpreadSheet(LabelFont) \
      -text $name \
      -fill $SpreadSheet(LabelFg);

  $SpreadSheet($this,TopCan) bind coltext_$cindex <ButtonRelease-1> \
    "SpreadSheet_HandleColBar $this $cindex";

  incr cindex;
  incr xpos $SpreadSheet(CellWidth);
  }
}
# SpreadSheet_SetColNames


#---------------------------------------------------------------------
#-      NAME:           SpreadSheet_SetRowNames()                   -
#-      HISTORY:                                                    -
#-              21 Apr 94       re7@ornl.gov                        -
#-      PURPOSE:                                                    -
#-              Given a list containing row label names, populates the -
#-              left canvas bars with the label strings as text     -
#---------------------------------------------------------------------
proc  SpreadSheet_SetRowNames { this name_list } \
{
  global SpreadSheet;

#       -- Delete Existing Text Items
#       --
$SpreadSheet($this,LeftCan) delete rowtexts;

set rindex 0;
set ypos $SpreadSheet(CellHalfHeight);
foreach name $name_list \
  {
  $SpreadSheet($this,LeftCan) create text \
      $SpreadSheet(CellHalfWidth) $ypos \
      -tags "rowtexts rowtext_$rindex" \
      -anchor c \
      -font $SpreadSheet(LabelFont) \
      -text $name \
      -fill $SpreadSheet(LabelFg);

  $SpreadSheet($this,LeftCan) bind rowtext_$rindex <ButtonRelease-1> \
    "SpreadSheet_HandleRowBar $this $rindex";

  incr rindex;
  incr ypos $SpreadSheet(CellHeight);
  }
}
# SpreadSheet_SetRowNames


#------------------------------------------------------------------------
#-      NAME:           SpreadSheet_SetRowValues()                      -
#-      HISTORY:                                                        -
#-              21 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Given a list containing row values (label followed by   -
#-              column values), populates the spreadsheet               -
#------------------------------------------------------------------------
proc  SpreadSheet_SetRowValues { this rv_list } \
{
  global SpreadSheet;

#       -- Delete Existing Text Items
#       --
$SpreadSheet($this,LeftCan) delete rowtexts;
$SpreadSheet($this,Matrix) delete celltexts;

#       -- Process Each List Item as a Row
#       --
set rindex 0;
set ypos $SpreadSheet(CellHalfHeight);

foreach row_list $rv_list \
  {
#               -- Create Label Text
#               --
  $SpreadSheet($this,LeftCan) create text \
      $SpreadSheet(CellHalfWidth) $ypos \
      -tags "rowtexts rowtext_$rindex" \
      -anchor c \
      -font $SpreadSheet(LabelFont) \
      -text [lindex $row_list 0] \
      -fill $SpreadSheet(LabelFg);

  $SpreadSheet($this,LeftCan) bind rowtext_$rindex <ButtonRelease-1> \
    "SpreadSheet_HandleRowBar $this $rindex";

#               -- Process Remaining Items in Sublist as Column Values
#               --
  set cindex 0;
  set xpos $SpreadSheet(CellHalfWidth);

  foreach value [lrange $row_list 1 end] \
    {
    $SpreadSheet($this,Matrix) create text \
        $xpos $ypos \
        -tags "celltexts celltext_${rindex}_${cindex}" \
        -anchor c \
        -font $SpreadSheet(CellFont) \
        -text $value \
        -fill $SpreadSheet(CellFg);

    incr xpos $SpreadSheet(CellWidth);
    incr cindex;
    }

  incr ypos $SpreadSheet(CellHeight);
  incr rindex;
  }
}
# SpreadSheet_SetRowValues
```

```
#------------------------------------------------------------------------
#-      NAME:           SpreadSheet_XScroll()                           -
#-      HISTORY:                                                        -
#-              21 Apr 94       re7@ornl.gov                            -
#------------------------------------------------------------------------
proc  SpreadSheet_XScroll { this x } \
{
  global SpreadSheet;

$SpreadSheet($this,Matrix) xview $x;
$SpreadSheet($this,TopCan) xview $x;
}
# SpreadSheet_XScroll


#------------------------------------------------------------------------
#-      NAME:           SpreadSheet_YScroll()                           -
#-      HISTORY:                                                        -
#-              21 Apr 94       re7@ornl.gov                            -
#------------------------------------------------------------------------
proc  SpreadSheet_YScroll { this y } \
{
  global SpreadSheet;

$SpreadSheet($this,Matrix) yview $y;
$SpreadSheet($this,LeftCan) yview $y;
}
# SpreadSheet_YScroll


#------------------------------------------------------------------------
#-      NAME:           SpreadSheet_Create()                            -
#-      HISTORY:                                                        -
#-              21 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Creates a SpreadSheet object                            -
#------------------------------------------------------------------------
proc  SpreadSheet_Create { \
  parent \
  { nrows 25 } \
  { ncols 10 } \
  { display_rows 10 } \
  { display_cols 5 } \
  } \
{
  global SpreadSheet;

#       -- Get Name of New Object
#       --
set name spread$SpreadSheet(count);
incr SpreadSheet(count);

#       -- Compute Canvas Display Values
#       --
if { $display_cols > $ncols } { set display_cols $ncols };
if { $display_rows > $nrows } { set display_rows $nrows };
set display_width [expr $display_cols * $SpreadSheet(CellWidth)];
set display_height [expr $display_rows * $SpreadSheet(CellHeight)];
set region_width [expr $ncols * $SpreadSheet(CellWidth)];
set region_height [expr $nrows * $SpreadSheet(CellHeight)];

#       -- Container Frame
#       --
frame $parent.${name}_frame \
    -background $SpreadSheet(Bg) \
    -borderwidth 1 \
    -relief sunken;

set SpreadSheet($name,Frame) $parent.${name}_frame;

#       -- Packing Dummies
#       --
frame $parent.${name}_frame.right \
    -background $SpreadSheet(Bg) \
    -width $SpreadSheet(ScrollWidth) \
    -borderwidth 1;
pack $parent.${name}_frame.right -side right -fill y;

frame $parent.${name}_frame.left \
    -background $SpreadSheet(Bg) \
    -width $SpreadSheet(CellWidth) \
    -borderwidth 1;
pack $parent.${name}_frame.left -side left -fill y;

frame $parent.${name}_frame.right.bottom \
    -background $SpreadSheet(Bg) \
    -height $SpreadSheet(ScrollHeight) \
    -borderwidth 1;
pack $parent.${name}_frame.right.bottom -side bottom;

frame $parent.${name}_frame.right.top \
    -background $SpreadSheet(Bg) \
    -height $SpreadSheet(CellHeight) \
    -borderwidth 1;
pack $parent.${name}_frame.right.top -side top;

frame $parent.${name}_frame.left.bottom \
    -background $SpreadSheet(Bg) \
    -height $SpreadSheet(ScrollHeight) \
    -borderwidth 1;
pack $parent.${name}_frame.left.bottom -side bottom;

frame $parent.${name}_frame.left.top \
    -background $SpreadSheet(Bg) \
    -height $SpreadSheet(CellHeight) \
    -borderwidth 1;
pack $parent.${name}_frame.left.top -side top;

#       -- Top Canvas
#       --
canvas $parent.${name}_frame.topcan \
    -width $display_width \
    -height $SpreadSheet(CellHeight) \
    -scrollregion "0 0 $region_width $SpreadSheet(CellHeight)" \
    -background $SpreadSheet(Bg) \
    -borderwidth 1 \
    -scrollincrement $SpreadSheet(CellHeight);
pack $parent.${name}_frame.topcan -side top -fill x;
set SpreadSheet($name,TopCan) $parent.${name}_frame.topcan;

#       -- Left Canvas
#       --
canvas $parent.${name}_frame.left.leftcan \
    -width $SpreadSheet(CellWidth) \
    -height $display_height \
    -scrollregion "0 0 $SpreadSheet(CellWidth) $region_height" \
    -background $SpreadSheet(Bg) \
    -borderwidth 1 \
    -scrollincrement $SpreadSheet(CellHeight);
pack $parent.${name}_frame.left.leftcan -side left -fill y;
set SpreadSheet($name,LeftCan) $parent.${name}_frame.left.leftcan;

#       -- Bottom Scrollbar
#       --
scrollbar $parent.${name}_frame.xscroll \
    -relief sunken \
    -orient hor \
    -width $SpreadSheet(ScrollHeight) \
    -command "SpreadSheet_XScroll $name";
pack $parent.${name}_frame.xscroll -side bottom -fill x;

#       -- Right Scrollbar
#       --
scrollbar $parent.${name}_frame.right.yscroll \
    -relief sunken \
    -orient ver \
    -width $SpreadSheet(ScrollWidth) \
    -command "SpreadSheet_YScroll $name";
pack $parent.${name}_frame.right.yscroll -side right -fill y;

#       -- Matrix Canvas
#       --
canvas $parent.${name}_frame.matrix \
    -width $display_width \
    -height $display_height \
    -scrollregion "0 0 $region_width $region_height" \
    -background $SpreadSheet(Bg) \
    -scrollincrement $SpreadSheet(CellHeight) \
    -xscroll "$parent.${name}_frame.xscroll set" \
    -yscroll "$parent.${name}_frame.right.yscroll set";
pack $parent.${name}_frame.matrix -side top -expand yes -fill both;
set SpreadSheet($name,Matrix) $parent.${name}_frame.matrix;

#       -- Create Buttons and Cells
#       --
for \
    { set row 0; set ypos 0 } \
    { $row < $nrows } \
    { incr row; incr ypos $SpreadSheet(CellHeight) } \
  {
  $parent.${name}_frame.left.leftcan create bar \
      0 $ypos \
      $SpreadSheet(CellWidth) [expr $ypos + $SpreadSheet(CellHeight)] \
      -tags "rowbars rowbar_$row" \
      -fill $SpreadSheet(LabelBg) \
      -width 1;
  $parent.${name}_frame.left.leftcan bind rowbar_$row <ButtonRelease-1> \
    "SpreadSheet_HandleRowBar $name $row";

  for \
      { set col 0; set xpos 0 } \
      { $col < $ncols } \
      { incr col; incr xpos $SpreadSheet(CellWidth) } \
    {
    if { $row == 0 } \
      {
      $parent.${name}_frame.topcan create bar \
          $xpos 0 \
          [expr $xpos + $SpreadSheet(CellWidth)] $SpreadSheet(CellHeight) \
          -tags "colbars colbar_$col" \
          -fill $SpreadSheet(LabelBg) \
          -width 1;
      $parent.${name}_frame.topcan bind colbar_$col <ButtonRelease-1> \
        "SpreadSheet_HandleColBar $name $col";
      }

    $parent.${name}_frame.matrix create rectangle \
        $xpos $ypos \
        [expr $xpos + $SpreadSheet(CellWidth) - 1] \
        [expr $ypos + $SpreadSheet(CellHeight)] \
        -tags "cells cell_row_$row cell_col_$col cell_${row}_${col}" \
        -fill $SpreadSheet(CellBg) \
        -outline $SpreadSheet(CellOutline) \
        -width 1;
    }
  }

return $name;
}
# SpreadSheet_Create
```

A.4 SPREADWINDOW.TCL

```
#-      %Z% %M%  %I% %G%
#----------------------------------------------------------------------
#-      NAME:           SpreadWindow.tcl                              -
#-      HISTORY:                                                      -
#-              22 Apr 94       re7@ornl.gov                          -
#-      PACKAGE:                                                      -
#-              SpreadWindow                                          -
#-      USAGE:                                                        -
#-              SpreadWindow_LoadFile [SpreadWindow_Create] filename  -
#-                                                                    -
#-              SpreadWindow_LoadCommandResult                        -
#-                      [SpreadWindow_Create] fdb-client-object       -
#----------------------------------------------------------------------

global SpreadWindow;

#----------------------------------------------------------------------
#-      Class Attributes                                              -
#----------------------------------------------------------------------
set SpreadWindow(Bg) gray75;
set SpreadWindow(TitleFg) navy;
set SpreadWindow(TitleFont) 10x20;
set SpreadWindow(count) 0;


#----------------------------------------------------------------------
#-      Object Attributes                                             -
#-              Data            list of observation lists, each with  -
#-                              the name of the observation and values -
#-                              for each series                       -
#-              FileName        name of temp file containing data     -
#-              Series          list of series names                  -
#-              Sheet           SpreadSheet object name               -
#-              Stacked         if "true", each observation is a stack -
#-                              (unused)                              -
#-              Title           chart title                           -
#-              TopWindow       name of the toplevel widget which is  -
#-                              the window                            -
#-              XTitle          x-axis title                          -
#-              YTitle          y-axis title                          -
#----------------------------------------------------------------------


#----------------------------------------------------------------------
#-      NAME:           SpreadWindow_CreateBarWindow()                -
#-      HISTORY:                                                      -
#-              22 Apr 94       re7@ornl.gov                          -
#-      PURPOSE:                                                      -
#-              Creates a BarWidnow object from the data file         -
#----------------------------------------------------------------------
proc  SpreadWindow_CreateBarWindow { this } \
{
  global SpreadWindow;

BarWindow_LoadFile [BarWindow_Create] $SpreadWindow($this,FileName);
}
# SpreadWindow_CreateBarWindow


#----------------------------------------------------------------------
#-      NAME:           SpreadWindow_CreateMenuBar()                  -
#-      HISTORY:                                                      -
#-              22 Apr 94       re7@ornl.gov                          -
#----------------------------------------------------------------------
proc  SpreadWindow_CreateMenuBar { parent this } \
```

```
{
  global Fdb SpreadWindow;

set bg [option get . menuBackground Tk];
set active_bg [option get . menuActiveBackground Tk];
set fg [option get . menuForeground Tk];
set font [option get . menuFont Tk];
set button_font [option get . menuButtonFont Tk];

frame $parent.menubar \
    -relief raised \
    -background $bg \
    -borderwidth 2;

#       -- File Button
menubutton $parent.menubar.file \
    -text "File" \
    -underline 0 \
    -padx 10 \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -font $button_font \
    -foreground $fg \
    -menu $parent.menubar.file.menu;

menu $parent.menubar.file.menu \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -borderwidth 2 \
    -foreground $fg \
    -font $font;

#               -- File Menu
$parent.menubar.file.menu add command \
    -state disabled \
    -label "Save" \
    -underline 0 \
    -command { puts stdout "Not yet implemented" };

$parent.menubar.file.menu add separator;

$parent.menubar.file.menu add command \
    -label "Exit" \
    -underline 1 \
    -background #7f0000 \
    -activebackground #af0000 \
    -command "destroy $parent";

#       -- Display Button
menubutton $parent.menubar.display \
    -text "Display" \
    -underline 0 \
    -padx 10 \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -font $button_font \
    -foreground $fg \
    -menu $parent.menubar.display.menu;

menu $parent.menubar.display.menu \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -borderwidth 2 \
    -foreground $fg \
    -font $font;

#               -- Display Menu
$parent.menubar.display.menu add command \
    -label " Bar Graph " \
    -underline 1 \
    -command "SpreadWindow_CreateBarWindow $this";

$parent.menubar.display.menu add command \
    -state disabled \
    -label " Line Graph " \
    -underline 1 \
    -command { puts stdout "Not yet implemented" };

#       -- Set Up Menu
pack \
    $parent.menubar.file \
    $parent.menubar.display \
    -side left \
    -ipadx 5 \
    -ipady 2;

bind $parent <Any-FocusIn> \
  "
  if { (\"%d\" == \"NotifyVirtual\") && (\"%m\" == \"NotifyNormal\") } \
    \"focus $parent.menubar\"
  "

tk_menuBar $parent.menubar \
    $parent.menubar.file;

return  $parent.menubar;
}
# SpreadWindow_CreateMenuBar


#------------------------------------------------------------------------
#-      NAME:           SpreadWindow_Create()                           -
#-      HISTORY:                                                        -
#-              22 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Creates a SpreadWindow object                           -
#------------------------------------------------------------------------
proc  SpreadWindow_Create {} \
{
  global SpreadWindow Fdb_library;

#       -- Name of This Object
#       --
set this spreadwin$SpreadWindow(count);

#       -- Name of Toplevel Window
#       --
set name .swtop$SpreadWindow(count);
set SpreadWindow($this,TopWindow) $name;

#       -- Build Toplevel
toplevel $name;
wm iconbitmap $name @$Fdb_library/spread.xbm
wm iconmask $name @$Fdb_library/spread.xbm
wm iconname $name "SpreadSheet - $SpreadWindow(count)";
wm minsize $name 300 200;
wm title $name "SpreadSheet Window - $SpreadWindow(count)";

#       -- Build MenuBar
SpreadWindow_CreateMenuBar $name $this;

#       -- Pack Frames
pack $name.menubar -side top -fill x;

#       -- Increment Window Count
incr SpreadWindow(count);

return  $this;
}
# SpreadWindow_Create


#------------------------------------------------------------------------
#-      NAME:           SpreadWindow_CreateSheet()                      -
#-      HISTORY:                                                        -
#-              22 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Creates the SpreadSheet object contained in this object -
#------------------------------------------------------------------------
proc  SpreadWindow_CreateSheet { this } \
{
  global SpreadWindow SpreadSheet;

#       -- Create Title Label
#       --
label $SpreadWindow($this,TopWindow).title \
    -background $SpreadWindow(Bg) \
    -foreground $SpreadWindow(TitleFg) \
    -font $SpreadWindow(TitleFont) \
    -text $SpreadWindow($this,Title);

#       -- Create SpreadSheet Widget
#       --
set sheet \
    [SpreadSheet_Create \
        $SpreadWindow($this,TopWindow) \
        [llength $SpreadWindow($this,Data)] \
        [llength $SpreadWindow($this,Series)] \
        ];
set SpreadWindow($this,Sheet) $sheet;

SpreadSheet_SetColNames $sheet $SpreadWindow($this,Series);

SpreadSheet_SetRowValues $sheet $SpreadWindow($this,Data);

pack $SpreadWindow($this,TopWindow).title \
    -side top \
    -ipady 5 \
    -fill x;

pack $SpreadSheet($sheet,Frame) \
    -side top -expand yes -fill both;
}
# SpreadWindow_CreateSheet


#------------------------------------------------------------------------
#-      NAME:           SpreadWindow_ParseAttrs()                       -
#-      HISTORY:                                                        -
#-              22 Apr 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Parse the attribute line in the data stream             -
#------------------------------------------------------------------------
proc  SpreadWindow_ParseAttrs { this line } \
{
  global SpreadWindow;

foreach piece [split $line "\t"] \
  {
  set parts [split $piece "="];
  set SpreadWindow($this,[lindex $parts 0]) [lindex $parts 1];
  }
}
# SpreadWindow_ParseAttrs


#------------------------------------------------------------------------
#-      NAME:           SpreadWindow_LoadFile()                         -
#-      HISTORY:                                                     -
#-              22 Apr 94       re7@ornl.gov                         -
#-      PURPOSE:                                                     -
#-              Populate the data for the chart from the data file   -
#--------------------------------------------------------------------
proc  SpreadWindow_LoadFile { this filename } \
{
  global SpreadWindow Fdb_library;

#       -- Initialize Attributes
#       --
set SpreadWindow($this,Title) "Could not open file $filename";
set SpreadWindow($this,XTitle) "";
set SpreadWindow($this,YTitle) "";
set SpreadWindow($this,Stacked) "true";
set SpreadWindow($this,Series) "";
set SpreadWindow($this,Data) {};
set SpreadWindow($this,FileName) "";
set SpreadWindow($this,Sheet) "";

#       -- Open File
#       --
if { [catch { set fid [open $filename "r"] }] == 0 } \
  {
#               -- Save File Name
#               --
  set SpreadWindow($this,FileName) $filename;

#               -- Read Attribute Line
#               --
  gets $fid line;
  SpreadWindow_ParseAttrs $this $line;
  set SpreadWindow($this,Series) \
      [split $SpreadWindow($this,Series) ":"];

  set SpreadWindow($this,Data) {};

#               -- Read Observation Lines
#               --
  while { [gets $fid line] > -1 } \
    {
    lappend SpreadWindow($this,Data) [split $line "\t"];
    }

#               -- Create SpreadSheet
#               --
  SpreadWindow_CreateSheet $this;
  }

update;
}
# SpreadWindow_LoadFile


#--------------------------------------------------------------------
#-      NAME:           SpreadWindow_LoadCommandResult()             -
#-      HISTORY:                                                     -
#-              22 Apr 94       re7@ornl.gov                         -
#-      PURPOSE:                                                     -
#-              Populate the data for the chart from the network stream -
#-              resulting from a command to the server               -
#--------------------------------------------------------------------
proc  SpreadWindow_LoadCommandResult { this fdb } \
{
  global Fdb SpreadWindow Fdb_library;

#       -- Set Data File Name
#       --
set filename "/tmp/[option get . filePrefix ""]$this";
if { [catch { set fid [open $filename "w"] }] } \
  {
  set SpreadWindow($this,FileName) "";
  puts stderr "Error creating $filename";
  } \
else \
  {
  set SpreadWindow($this,FileName) $filename;
  }

#       -- Initialize Attributes
#       --
set SpreadWindow($this,Title) "Error reading results";
set SpreadWindow($this,XTitle) "";
set SpreadWindow($this,YTitle) "";
set SpreadWindow($this,Stacked) "true";
set SpreadWindow($this,Series) "";
set SpreadWindow($this,Data) {};
set SpreadWindow($this,Sheet) "";

#       -- Read Attribute Line
#       --
if { "[$fdb ReadNextLine line]" != "eof" } \
  {
  catch { puts $fid $line };
  SpreadWindow_ParseAttrs $this $line;
  set SpreadWindow($this,Series) \
      [split $SpreadWindow($this,Series) ":"];

  set SpreadWindow($this,Data) {};

#               -- Read Observation Lines
#               --
  while { "[$fdb ReadNextLine line]" != "eof" } \
    {
    catch { puts $fid $line };
    lappend SpreadWindow($this,Data) [split $line "\t"];
    }

#               -- Create SpreadSheet
#               --
  SpreadWindow_CreateSheet $this;
  }

catch { close $fid };

update;
}
# SpreadWindow_LoadCommandResult
```

A.5 TEXTWINDOW.TCL

```
#-      %Z% %M%  %I% %G%
#-------------------------------------------------------------------------
#-      NAME:           TextWindow.tcl                                   -
#-      HISTORY:                                                         -
#-               4 Mar 94       re7@ornl.gov                             -
#-      PACKAGE:                                                         -
#-              TextWindow                                               -
#-      USAGE:                                                           -
#-              TextWindow_LoadFile [TextWindow_Create] filename         -
#-                                                                       -
#-              TextWindow_LoadCommandResult                             -
#-                      [TextWindow_Create] fdb-client-object            -
#-------------------------------------------------------------------------

global TextWindow;

#-------------------------------------------------------------------------
#-      Class Attributes                                                 -
#-------------------------------------------------------------------------
set TextWindow(Bg) gray30;
set TextWindow(Fg) white;
set TextWindow(Font) 9x15;
#set TextWindow(Font) 8x13;

set TextWindow(count) 0;


#-------------------------------------------------------------------------
#-      Object Attributes                                                -
#-              Text            name of text widget                      -
#-              TopWindow       name of the toplevel widget which is     -
#-                              the window                               -
#-------------------------------------------------------------------------


#-------------------------------------------------------------------------
#-      NAME:           TextWindow_CreateMenuBar()                       -
#-      HISTORY:                                                         -
#-               4 Mar 94       re7@ornl.gov                             -
#-------------------------------------------------------------------------
proc  TextWindow_CreateMenuBar ( parent this ) \
(
  global Fdb TextWindow;

set bg [option get . menuBackground Tk];
set active_bg [option get . menuActiveBackground Tk];
set fg [option get . menuForeground Tk];
set font [option get . menuFont Tk];
set button_font [option get . menuButtonFont Tk];

frame $parent.menubar \
    -relief raised \
    -background $bg \
    -borderwidth 2;

#       -- File Button
menubutton $parent.menubar.file \
    -text "File" \
    -underline 0 \
    -padx 10 \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -font $button_font \
    -foreground $fg \
    -menu $parent.menubar.file.menu;
```

```
menu $parent.menubar.file.menu \
    -activebackground $active_bg \
    -activeforeground $fg \
    -background $bg \
    -borderwidth 2 \
    -foreground $fg \
    -font $font;

#               -- File Menu
$parent.menubar.file.menu add command \
    -state disabled \
    -label "Save" \
    -underline 0 \
    -command ( puts stdout "Not yet implemented" );

$parent.menubar.file.menu add separator;

$parent.menubar.file.menu add command \
    -label "Exit" \
    -underline 1 \
    -background #7f0000 \
    -activebackground #af0000 \
    -command "destroy $parent";

#       -- Set Up Menu
pack \
    $parent.menubar.file \
    -side left \
    -ipadx 5 \
    -ipady 2;

bind $parent <Any-FocusIn> \
  "
  if ( (\"%d\" == \"NotifyVirtual\") && (\"%m\" == \"NotifyNormal\") ) \
    \"focus $parent.menubar\"
  "

tk_menuBar $parent.menubar \
    $parent.menubar.file;

return  $parent.menubar;
)
# TextWindow_CreateMenuBar


#-------------------------------------------------------------------------
#-      NAME:           TextWindow_CreateText()                          -
#-      HISTORY:                                                         -
#-               4 Mar 94       re7@ornl.gov                             -
#-      PURPOSE:                                                         -
#-------------------------------------------------------------------------
proc  TextWindow_CreateText ( this ) \
(
  global Fdb TextWindow;

set parent $TextWindow($this,TopWindow);

#       -- Frame For Text and Scrollbar
#       --
frame $parent.text_frame \
    -background $TextWindow(Bg);

set text_widget $parent.text_frame.text;

#       -- Build Text Widget
#       --
text $text_widget \
    -background $TextWindow(Bg) \
    -foreground $TextWindow(Fg) \
    -borderwidth 2 \
    -relief sunken \
    -font $TextWindow(Font) \
    -height 20 \
    -width 80 \
    -wrap word \
    -yscrollcommand "$parent.text_frame.yscroll set";

#       -- Build Scrollbar
#       --
scrollbar $parent.text_frame.yscroll \
    -orient ver \
    -relief sunken \
    -command "$parent.text_frame.text yview";
#       -- Pack
#       --
pack \
    $parent.text_frame.yscroll \
    -side right \
    -pady 2 \
    -fill y;

pack \
    $parent.text_frame.text \
    -side top \
    -padx 5 \
    -pady 2 \
    -expand yes \
    -fill both;

return  $text_widget;
}
# TextWindow_CreateText

#------------------------------------------------------------------------
#-      NAME:           TextWindow_Create()                             -
#-      HISTORY:                                                        -
#-               4 Mar 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Creates a TextWindow object                             -
#------------------------------------------------------------------------
proc  TextWindow_Create {} \
{
  global TextWindow Fdb_library;

#       -- Set Name of This Object
#       --
set this textwin$TextWindow(count);

#       -- Name of This Window
#       --
set name .twtop$TextWindow(count);
set TextWindow($this,TopWindow) $name;

#       -- Build Toplevel
#       --
toplevel $name;
wm iconbitmap $name @$Fdb_library/terminal.xbm
wm iconmask $name @$Fdb_library/terminal.xbm
wm iconname $name "Text - $TextWindow(count)";
wm minsize $name 10 10;
wm title $name "Text Window - $TextWindow(count)";

#       -- Build Frames
#       --
TextWindow_CreateMenuBar $name $this;
set TextWindow($this,Text) [TextWindow_CreateText $this];

#       -- Pack Frames
#       --
pack $name.menubar -side top -fill x;
pack $name.text_frame -side top -ipadx 5 -ipady 2 -expand yes -fill both;
```

```
#       -- Increment Window Count
#       --
incr TextWindow(count);

return  $this;
}
# TextWindow_Create

#------------------------------------------------------------------------
#-      NAME:           TextWindow_LoadFile()                           -
#-      HISTORY:                                                        -
#-               4 Mar 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Populate the data for the text from the data file       -
#------------------------------------------------------------------------
proc  TextWindow_LoadFile { this filename } \
{
  global TextWindow Fdb_library;

set text_widget $TextWindow($this,Text);

#       -- Open File
#       --
if { [catch { set fid [open $filename "r"] }] } \
  {
  $text_widget insert end "Could not open file $filename";
  } \
else \
  {
  while { [set width [gets $fid line]] > -1 } \
    {
    if { $width > 80 } { $text_widget configure -width 132 };

    $text_widget insert end [format "%s\n" $line];
#    update;
    }
  }
update;
}
# TextWindow_LoadFile

#------------------------------------------------------------------------
#-      NAME:           TextWindow_LoadCommandResult()                  -
#-      HISTORY:                                                        -
#-               4 Mar 94       re7@ornl.gov                            -
#-      PURPOSE:                                                        -
#-              Populate the data for the text from the network stream -
#-              resulting from a command to the server                  -
#------------------------------------------------------------------------
proc  TextWindow_LoadCommandResult { this fdb } \
{
  global TextWindow Fdb_library;

set text_widget $TextWindow($this,Text);
#       -- Open File
#       --
while { "[set width [$fdb ReadNextLine line]]" != "eof" } \
  {
  if { $width > 80 } { $text_widget configure -width 132 };

  $text_widget insert end [format "%s\n" $line];
  }
update;
}
# TextWindow_LoadCommandResult
```

APPENDIX B. WIDGET EXTENSION SOURCE LISTINGS

B.1 TKTHING.H

```
        /* %Z% %M%  %I% %G% */
#ifndef __TkThing__
#define __TkThing__
/**********************************************************************
*       NAME:           TkThing.h                                     *
*       HISTORY:                                                      *
*               27 Sep 93       Ron Lee                               *
*       CLASS:                                                        *
*               TkThing                                               *
**********************************************************************/
extern "C"
(
#include <tk.h>
)


/**********************************************************************
*       Flag Mask Values                                              *
**********************************************************************/
#define ConfigError     0x01
#define RedrawPending   0x02


/***********************************************************************
*)      CLASS:          TkThing                                       (*
*)      PURPOSE:                                                      (*
*)              Example C++ class implementing a TkTcl command        (*
***********************************************************************/
class  TkThing
  (
                // Class Attributes
                //
protected:

  static Tk_ConfigSpec  _tkConfigSpecs[];


                // Object Attributes
                //
protected:

  Tk_Window             tkWin;
  Tcl_Interp *          tkInterp;

  unsigned short        tkFlags;
  int                   tkWidth;
  int                   tkHeight;

  int                   tkBorderWidth;
  Tk_3DBorder           tkBgBorder;
  Tk_3DBorder           tkFgBorder;
  int                   tkRelief;


                // Constructors and Destructors
                //
public:

  /*********************************************************************
  *     NAME:           TkThing()                                     *
  *     PURPOSE:                                                      *
  *             Construct a particular TkThing given the path name    *
  *********************************************************************/
                      TkThing(
                          Tk_Window,
                          Tcl_Interp *,
                          int,
                          char * []
                          );

  /*********************************************************************
  *     NAME:           ~TkThing()                                    *
  *     PURPOSE:                                                      *
  *             Destructor                                            *
  *********************************************************************/
  virtual             ~TkThing();


                // Class Methods
                //
public:

  /*********************************************************************
  *     NAME:           HandleDestroy()                               *
  *     PURPOSE:                                                      *
  *             Routine registered to destroy an object when all      *
  *             references have exhausted.                            *
  *     NOTE:                                                         *
  *             This is not truly an inline                           *
  *********************************************************************/
  static void         HandleDestroy( TkThing * thing )
                        (
                        delete  thing;
                        );

  /*********************************************************************
  *     NAME:           HandleDisplay()                               *
  *     PURPOSE:                                                      *
  *             Routine registered as an idle procedure to display the *
  *             object.                                               *
  *     NOTE:                                                         *
  *             This is not truly an inline                           *
  *********************************************************************/
  static void         HandleDisplay( TkThing & thing )
                        (
                        thing.Display();
                        );

  /*********************************************************************
  *     NAME:           HandleEvents()                                *
  *     PURPOSE:                                                      *
  *             Routine registered to handle events                   *
  *     NOTE:                                                         *
  *             This is not truly an inline                           *
  *********************************************************************/
  static void         HandleEvents( TkThing & thing, XEvent & event )
                        (
                        thing.HandleEvent( event );
                        );
/**********************************************************************
*       NAME:           ProcessFactoryCommand()                       *
*       PURPOSE:                                                      *
*               Processes the "thing" command by creating a new       *
*               TkThing object.                                       *
*       OUTPUT:                                                       *
*               Tcl status                                            *
**********************************************************************/
static int              ProcessFactoryCommand(
                            Tk_Window           main_window,
                            Tcl_Interp *        interp,
                            int                 argc,
                            char *              argv[]
                            );


/**********************************************************************
*       NAME:           ProcessWidgetCommand()                        *
*       PURPOSE:                                                      *
*               Processes the "thing" widget command                  *
*       OUTPUT:                                                       *
*               Tcl status                                            *
**********************************************************************/
static int              ProcessWidgetCommand(
                            TkThing &           thing,
                            Tcl_Interp *        interp,
                            int                 argc,
                            char *              argv[]
                            )
                          (
                          return  thing.ProcessCommand( argc, argv );
                          );


/**********************************************************************
*       NAME:           RegisterCommand()                             *
*       PURPOSE:                                                      *
*               Registers the "thing" command with the interpreter    *
**********************************************************************/
static void             RegisterCommand(
                            Tk_Window           main_win,
                            Tcl_Interp *        interp
                            )
                          (
                          Tcl_CreateCommand(
                              interp,
                              "thing",
                              (Tcl_CmdProc *) ProcessFactoryCommand,
                              (ClientData) main_win,
                              (Tcl_CmdDeleteProc *) NULL
                              );
                          );


                // Object Methods
                //
protected:

/**********************************************************************
*       NAME:           Configure()                                   *
*       PURPOSE:                                                      *
*               Processes the configure options                       *
*       OUTPUT:                                                       *
*               Tcl status                                            *
**********************************************************************/
int                     Configure( int argc, char * argv[], int flags = 0 );

/**********************************************************************
*       NAME:           Display()                                     *
*       PURPOSE:                                                      *
**********************************************************************/
void                    Display();


/**********************************************************************
*       NAME:           HandleEvent()                                 *
*       PURPOSE:                                                      *
**********************************************************************/
void                    HandleEvent( XEvent & );


/**********************************************************************
*       NAME:           ProcessCommand()                              *
*       PURPOSE:                                                      *
*       OUTPUT:                                                       *
*               Tcl status                                            *
**********************************************************************/
int                     ProcessCommand( int argc, char * argv[] );
);


#endif
```

B.2 TKTHING.CC

```
static char *   sccs_id = "%Z% %M%  %I% %G%";
/*************************************************************************
*       NAME:           TkThing.cc                                      *
*       HISTORY:                                                        *
*               27 Sep 93       Ron Lee                                 *
*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "TkThing.h"


/*************************************************************************
*       NAME:           TkThing::_tkConfigSpecs                         *
*************************************************************************/
Tk_ConfigSpec           TkThing::_tkConfigSpecs[] =
  {
    {TK_CONFIG_BORDER, "-background", "background", "Background",
        "#cdb79e", Tk_Offset(TkThing, tkBgBorder), TK_CONFIG_COLOR_ONLY},
    {TK_CONFIG_BORDER, "-background", "background", "Background",
        "white", Tk_Offset(TkThing, tkBgBorder), TK_CONFIG_MONO_ONLY},
    {TK_CONFIG_SYNONYM, "-bd", "borderWidth", (char *) NULL,
        (char *) NULL, 0, 0},
    {TK_CONFIG_SYNONYM, "-bg", "background", (char *) NULL,
        (char *) NULL, 0, 0},
    {TK_CONFIG_INT, "-borderwidth", "borderWidth", "BorderWidth",
        "2", Tk_Offset(TkThing, tkBorderWidth), 0},
    {TK_CONFIG_SYNONYM, "-fg", "foreground", (char *) NULL,
        (char *) NULL, 0, 0},
    {TK_CONFIG_BORDER, "-foreground", "foreground", "Foreground",
        "#b03060", Tk_Offset(TkThing, tkFgBorder), TK_CONFIG_COLOR_ONLY},
    {TK_CONFIG_BORDER, "-foreground", "foreground", "Foreground",
        "black", Tk_Offset(TkThing, tkFgBorder), TK_CONFIG_MONO_ONLY},
    {TK_CONFIG_INT, "-height", "height", "Height",
        "64", Tk_Offset(TkThing, tkHeight), 0},
    {TK_CONFIG_RELIEF, "-relief", "relief", "Relief",
        "raised", Tk_Offset(TkThing, tkRelief), 0},
    {TK_CONFIG_INT, "-width", "width", "Width",
        "64", Tk_Offset(TkThing, tkWidth), 0},
    {TK_CONFIG_END, (char *) NULL, (char *) NULL, (char *) NULL,
        (char *) NULL, 0, 0}
  };


/*************************************************************************
*       NAME:           TkThing::TkThing()                              *
*       HISTORY:                                                        *
*               27 Sep 93       Ron Lee                                 *
*************************************************************************/
TkThing::TkThing( Tk_Window win, Tcl_Interp * interp, int argc, char * argv[] )
{
tkWin = win;
tkInterp = interp;
tkFlags = 0;
tkBgBorder = NULL;
tkFgBorder = NULL;

if ( Configure( argc, argv ) != TCL_OK )
  tkFlags |= ConfigError;
} // TkThing::TkThing


/*************************************************************************
*       NAME:           TkThing::~TkThing()                             *
*       HISTORY:                                                        *
*               27 Sep 93       Ron Lee                                 *
```

```
*************************************************************************/
TkThing::~TkThing()
{
if ( tkBgBorder != NULL )
  Tk_Free3DBorder( tkBgBorder );

if ( tkFgBorder != NULL )
  Tk_Free3DBorder( tkFgBorder );
} // TkThing::~TkThing


/*************************************************************************
*       NAME:           TkThing::Configure()                            *
*       HISTORY:                                                        *
*               27 Sep 93       Ron Lee                                 *
*************************************************************************/
int
TkThing::Configure( int argc, char * argv[], int flags )
{
  int           result;


result = Tk_ConfigureWidget(
    tkInterp,
    tkWin,
    _tkConfigSpecs,
    argc,
    argv,
    (char *) this,
    flags
    );

if ( result == TCL_OK )
  {
  Tk_SetBackgroundFromBorder( tkWin, tkBgBorder );

  Tk_GeometryRequest( tkWin, tkWidth, tkHeight );
  Tk_SetInternalBorder( tkWin, tkBorderWidth );

  if ( !( tkFlags & RedrawPending ) )
    {
    Tk_DoWhenIdle( (Tk_IdleProc *) HandleDisplay, (ClientData) this );
    tkFlags |= RedrawPending;
    }
  }

return  result;
} // TkThing::Configure


/*************************************************************************
*       NAME:           TkThing::Display()                              *
*       HISTORY:                                                        *
*               27 Sep 93       Ron Lee                                 *
*************************************************************************/
void
TkThing::Display()
{
  int                   offset = tkBorderWidth << 1;
  GC                    gc;


tkFlags &= ~RedrawPending;

if ( tkWin != NULL && Tk_IsMapped( tkWin ) )
  {
  gc = DefaultGC(
      Tk_Display( tkWin ),
      DefaultScreen( Tk_Display(tkWin) )
      );

  Tk_Fill3DRectangle(
      Tk_Display( tkWin ),
      Tk_WindowId( tkWin ),
      tkBgBorder,
      0, 0,
      Tk_Width( tkWin ), Tk_Height( tkWin ),
      tkBorderWidth,
      tkRelief
      );

  XSetLineAttributes(
      Tk_Display( tkWin ),
      gc,
      5,
      LineSolid,
      CapRound,
      JoinRound
      );

  XSetForeground(
      Tk_Display( tkWin ),
      gc,
      (Tk_3DBorderColor( tkFgBorder ))->pixel
      );

  XDrawLine(
      Tk_Display( tkWin ),
      Tk_WindowId( tkWin ),
      gc,
      offset,
      offset,
      tkWidth - offset,
      tkHeight - offset
      );
  } // if
} // TkThing::Display

/***********************************************************************
*       NAME:           TkThing::HandleEvent()                         *
*       HISTORY:                                                       *
*               27 Sep 93       Ron Lee                                *
***********************************************************************/
void
TkThing::HandleEvent( XEvent & event )
{
if (
    event.type == ConfigureNotify ||
    ( event.type == Expose &&
      event.xexpose.count == 0 )
    )
  {
  tkWidth = Tk_Width( tkWin );
  tkHeight = Tk_Height( tkWin );

  if ( !( tkFlags & RedrawPending ) )
    {
    Tk_DoWhenIdle( HandleDisplay, (ClientData) this );
    tkFlags |= RedrawPending;
    }
  }

else if ( event.type == DestroyNotify )
  {
  Tcl_DeleteCommand( tkInterp, Tk_PathName( tkWin ) );
  tkWin = NULL;

  if ( tkFlags & RedrawPending )
    Tk_CancelIdleCall( HandleDisplay, (ClientData) this );
```

```
  Tk_EventuallyFree( (ClientData) this, (Tk_FreeProc *) HandleDestroy );
  }
} // TkThing::HandleEvent

/***********************************************************************
*       NAME:           TkThing::ProcessCommand()                      *
*       HISTORY:                                                       *
*               27 Sep 93       Ron Lee                                *
***********************************************************************/
int
TkThing::ProcessCommand( int argc, char * argv[] )
{
  int           result = TCL_OK;


if ( argc < 2 )
  {
  Tcl_AppendResult( tkInterp, "Usage: ", argv[0], " option ?arg ...?", NULL );
  result = TCL_ERROR;
  }

else if ( strcmp( argv[1], "configure" ) == 0 )
  {
  if ( argc == 2 )
    {
    result = Tk_ConfigureInfo(
        tkInterp,
        tkWin,
        _tkConfigSpecs,
        (char *) this,
        NULL,
        0
        );
    }

  else if ( argc == 3 )
    {
    result = Tk_ConfigureInfo(
        tkInterp,
        tkWin,
        _tkConfigSpecs,
        (char *) this,
        argv[2],
        0
        );
    }

  else
    result = Configure( argc - 2, argv + 2, TK_CONFIG_ARGV_ONLY );
  }

return  result;
} // TkThing::ProcessCommand

/***********************************************************************
*       NAME:           TkThing::ProcessFactoryCommand()               *
*       HISTORY:                                                       *
*               27 Sep 93       Ron Lee                                *
***********************************************************************/
int
TkThing::ProcessFactoryCommand(
    Tk_Window           main_win,
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  Tk_Window                 result = TCL_OK;
  Tk_Window               new_win;
  TkThing *               new_thing;
```

```
        // Check For Minimal Arguments
        //
if ( argc < 2 )
  {
  Tcl_SetResult( interp, "Usage: thing pathname ?options?", TCL_VOLATILE );
  result = TCL_ERROR;
  }

        // Create New Tk Window
        //
else if (
    (new_win = Tk_CreateWindowFromPath( interp, main_win, argv[1], NULL )) ==
    NULL
    )
  {
  Tcl_SetResult( interp, "Error creating window", TCL_VOLATILE );
  result = TCL_ERROR;
  }

        // Create New Object
        //
else if (
    (new_thing = new TkThing( new_win, interp, argc - 2, argv + 2 )) == NULL
    )
  {
  Tcl_SetResult( interp, "Error allocating data structure", TCL_VOLATILE );
  result = TCL_ERROR;
  }

        // Check For Configure Error
        //
else if ( new_thing->tkFlags & ConfigError )
  {
  Tk_DestroyWindow( new_win );
  delete new_thing;
  result = TCL_ERROR;
  }

else
  {
  Tk_SetClass( new_win, "Thing" );

  Tk_CreateEventHandler(
      new_win,
      ExposureMask | StructureNotifyMask,
      (Tk_EventProc *) HandleEvents,
      (ClientData) new_thing
      );

  Tcl_CreateCommand(
      interp,
      Tk_PathName( new_win ),
      (Tcl_CmdProc *) ProcessWidgetCommand,
      (ClientData) new_thing,
      NULL
      );

  Tcl_SetResult( interp, Tk_PathName( new_win ), TCL_VOLATILE );
  } // else everything allocated

return  result;
} // TkThing::ProcessFactoryCommand
```

APPENDIX C. SOURCE LISTINGS FOR CANVAS ITEM EXTENSION

C.1 TKBARITEM.H

```
        /* %Z% %M%  %I% %G% */
#ifndef __TkBarItem__
#define __TkBarItem__
/*************************************************************************
*       NAME:           TkBarItem.h                                      *
*       HISTORY:                                                         *
*               18 Oct 93       Ron Lee                                  *
*       CLASS:                                                           *
*               TkBar                                                    *
*************************************************************************/
#include "tkext.h"


/*************************************************************************
*)      CLASS:          TkBarItem                                       (*
*)      PURPOSE:                                                        (*
*************************************************************************/
class TkBarItem
  (
                // Attributes
                //
protected:

  static Tk_ConfigSpec  _biConfigSpecs[];
  static Tk_ItemType    _biItemType;

public:

  Tk_Item               biHeader;
  double                biBbox[ 4 ];
  int                   biWidth;
  Tk_3DBorder           biFillBorder;
  int                   biRelief;


                // Methods
                //
protected:

  /**********************************************************************
  *     NAME:           Configure()                                      *
  *     PURPOSE:                                                         *
  ***********************************************************************/
  static int            Configure(
                            Tk_Canvas *,
                            Tk_Item *,
                            int                 argc,
                            char *              argv[],
                            int                 flags
                            );

  /**********************************************************************
  *     NAME:           Create()                                         *
  *     PURPOSE:                                                         *
  ***********************************************************************/
  static int            Create(
                            Tk_Canvas *,
                            Tk_Item *,
                            int                 argc,
                            char *              argv[]
                            );

  /**********************************************************************
  *     NAME:           Delete()                                         *
  *     PURPOSE:                                                         *
  ***********************************************************************/
  static void           Delete( Tk_Canvas *, Tk_Item * );

  /**********************************************************************
  *     NAME:           Draw()                                           *
  *     PURPOSE:                                                         *
  ***********************************************************************/
  static void           Draw( Tk_Canvas *, Tk_Item *, Drawable );

  /**********************************************************************
  *     NAME:           GenPostScript()                                  *
  *     PURPOSE:                                                         *
  ***********************************************************************/
  static int            GenPostScript(
                            Tk_Canvas *,
                            Tk_Item *,
                            Tk_PostscriptInfo *
                            );

public:

  /**********************************************************************
  *     NAME:           Register()                                       *
  *     PURPOSE:                                                         *
  *                                                                      *
  ***********************************************************************/
  static void           Register();
  );

#endif
```

C.2 TKBARITEM.CC

```
static char *   sccs_id = "%Z% %M%  %I% %G%";
/**********************************************************************
*       NAME:           TkBarItem.cc                                  *
*       HISTORY:                                                      *
*               18 Oct 93       Ron Lee                               *
**********************************************************************/
#include <stdio.h>
#include <math.h>
#include "TkBarItem.h"


/**********************************************************************
*       NAME:           TkBarItem::_biConfigSpecs                     *
**********************************************************************/
Tk_ConfigSpec           TkBarItem::_biConfigSpecs[] =
  {
    { TK_CONFIG_BORDER, "-fill", (char *) NULL, (char *) NULL,
        "gray50", Tk_Offset(TkBarItem, biFillBorder), TK_CONFIG_NULL_OK },
    { TK_CONFIG_CUSTOM, "-tags", (char *) NULL, (char *) NULL,
        (char *) NULL, 0, TK_CONFIG_NULL_OK, &tkCanvasTagsOption },
    { TK_CONFIG_PIXELS, "-width", (char *) NULL, (char *) NULL,
        "1", Tk_Offset(TkBarItem, biWidth), TK_CONFIG_DONT_SET_DEFAULT },
    { TK_CONFIG_RELIEF, "-relief", NULL, NULL,
        "raised", Tk_Offset(TkBarItem, biRelief), TK_CONFIG_DONT_SET_DEFAULT },
    { TK_CONFIG_END, (char *) NULL, (char *) NULL, (char *) NULL,
        (char *) NULL, 0, 0 }
  };


/**********************************************************************
*       NAME:           TkBarItem::_biItemType                        *
**********************************************************************/
Tk_ItemType             TkBarItem::_biItemType =
  {
  "bar",                                /* name */
  sizeof(TkBarItem),                    /* itemSize */
  TkBarItem::Create,                    /* createProc */
  TkBarItem::_biConfigSpecs,            /* configSpecs */
  TkBarItem::Configure,                 /* configureProc */
  RectOvalCoords,                       /* coordProc */
  TkBarItem::Delete,                    /* deleteProc */
  TkBarItem::Draw,                      /* displayProc */
  0,                                    /* alwaysRedraw */
  RectToPoint,                          /* pointProc */
  RectToArea,                           /* areaProc */
  TkBarItem::GenPostScript,             /* postscriptProc */
  ScaleRectOval,                        /* scaleProc */
  TranslateRectOval,                    /* translateProc */
  (Tk_ItemIndexProc *) NULL,            /* indexProc */
  (Tk_ItemCursorProc *) NULL,           /* icursorProc */
  (Tk_ItemSelectionProc *) NULL,        /* selectionProc */
  (Tk_ItemInsertProc *) NULL,           /* insertProc */
  (Tk_ItemDCharsProc *) NULL,           /* dTextProc */
  (Tk_ItemType *) NULL                  /* nextPtr */
  };


/**********************************************************************
*       NAME:           TkBarItem::Create()                           *
*       HISTORY:                                                      *
*               18 Oct 93       Ron Lee                               *
**********************************************************************/
int
TkBarItem::Create(
    Tk_Canvas * canvas_ptr,
    Tk_Item *   item_ptr,
    int         argc,
    char *      argv[]
    )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;
  int                   status = TCL_OK;

        // Check Argument List
        //
if ( argc < 4 )
  {
  Tcl_AppendResult(
      canvas_ptr->interp,
      "wrong # args:  should be \"",
      Tk_PathName( canvas_ptr->tkwin ),
      "\" create ",
      item_ptr->typePtr->name,
      " x1 y1 x2 y2 ?options?",
      NULL
      );

  status = TCL_ERROR;
  }

else
  {
  bar_ptr->biWidth = 1;
  bar_ptr->biFillBorder = NULL;
  bar_ptr->biRelief = TK_RELIEF_RAISED;

                // Process Args
                //
  if (
    (TkGetCanvasCoord( canvas_ptr, argv[0], &bar_ptr->biBbox[0] ) != TCL_OK) ||
    (TkGetCanvasCoord( canvas_ptr, argv[1], &bar_ptr->biBbox[1] ) != TCL_OK) ||
    (TkGetCanvasCoord( canvas_ptr, argv[2], &bar_ptr->biBbox[2] ) != TCL_OK) ||
    (TkGetCanvasCoord( canvas_ptr, argv[3], &bar_ptr->biBbox[3] ) != TCL_OK) ||
    (Configure( canvas_ptr, item_ptr, argc - 4, argv + 4, 0 ) != TCL_OK)
      )
    {
    Delete( canvas_ptr, item_ptr );
    status = TCL_ERROR;
    }
  }

return  status;
} // TkBarItem::Create


/**********************************************************************
*       NAME:           TkBarItem::Configure()                        *
*       HISTORY:                                                      *
*               18 Oct 93       Ron Lee                               *
**********************************************************************/
int
TkBarItem::Configure(
    Tk_Canvas *         canvas_ptr,
    Tk_Item *           item_ptr,
    int                 argc,
    char *              argv[],
    int                 flags
    )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;
  int                   status;


status = Tk_ConfigureWidget(
    canvas_ptr->interp,
    canvas_ptr->tkwin,
    _biConfigSpecs,
    argc,
    argv,
    (char *) bar_ptr,
    flags
    );

if ( status == TCL_OK )
  ComputeRectOvalBbox( canvas_ptr, (char *) bar_ptr );

return  status;
} // TkBarItem::Configure


/**********************************************************************
*       NAME:           TkBarItem::Delete()                          *
*       HISTORY:                                                     *
*               18 Oct 93       Ron Lee                              *
**********************************************************************/
void
TkBarItem::Delete( Tk_Canvas * canvas_ptr, Tk_Item * item_ptr )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;

if ( bar_ptr->biFillBorder != NULL )
  Tk_Free3DBorder( bar_ptr->biFillBorder );
} // TkBarItem::Delete


/**********************************************************************
*       NAME:           TkBarItem::Draw()                            *
*       HISTORY:                                                     *
*               18 Oct 93       Ron Lee                              *
**********************************************************************/
void
TkBarItem::Draw(
    Tk_Canvas * canvas_ptr,
    Tk_Item *   item_ptr,
    Drawable    drawable
    )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;

  int                   x1 = SCREEN_X( canvas_ptr, bar_ptr->biBbox[0] );
  int                   y1 = SCREEN_Y( canvas_ptr, bar_ptr->biBbox[1] );
  int                   x2 = SCREEN_X( canvas_ptr, bar_ptr->biBbox[2] );
  int                   y2 = SCREEN_Y( canvas_ptr, bar_ptr->biBbox[3] );


        // Must Be At Least 1 Pixel Big in Each Dimension
        //
if ( x2 <= x1 )
  x2 = x1 + 1;

if ( y2 <= y1 )
  y2 = y1 + 1;

Tk_Fill3DRectangle(
    canvas_ptr->display,
    drawable,
    bar_ptr->biFillBorder,
    x1, y1,
    x2 - x1 - 1, y2 - y1 - 1,
    bar_ptr->biWidth,
    bar_ptr->biRelief
    );
} // TkBarItem::Draw


/**********************************************************************
*       NAME:           TkBarItem::GenPostScript()                   *
*       HISTORY:                                                     *
*               18 Oct 93       Ron Lee                              *
**********************************************************************/
int
TkBarItem::GenPostScript(
    Tk_Canvas *           canvas_ptr,
    Tk_Item *             item_ptr,
    Tk_PostscriptInfo * ps_info_ptr
    )
{
  register TkBarItem *  bar_ptr = (TkBarItem *) item_ptr;
  int                   status = TCL_OK;

  char                  path_cmd[ 500 ];
  char                  string[ 100 ];

  double                y1 = TkCanvPsY( ps_info_ptr, bar_ptr->biBbox[1] );
  double                y2 = TkCanvPsY( ps_info_ptr, bar_ptr->biBbox[3] );
  double                deltax = bar_ptr->biBbox[2] - bar_ptr->biBbox[0];
  double                deltay = y2 - y1;

  XColor *              bg = Tk_3DBorderColor( bar_ptr->biFillBorder );
  XColor *              light = Tk_3DBorderLightColor( bar_ptr->biFillBorder );
  XColor *              dark = Tk_3DBorderDarkColor( bar_ptr->biFillBorder );


        // Check Colors
        //
if ( bg == NULL || light == NULL || dark == NULL )
  status = TCL_ERROR;

else
  {
                // Draw Filled Rectangle
                //
  sprintf(
      path_cmd,
      "newpath\n"
      "%g %g moveto %g 0 rlineto 0 %g rlineto %g neg 0 rlineto closepath\n",
      bar_ptr->biBbox[0], y1,
      deltax,
      deltay,
      deltax
      );

  Tcl_AppendResult( canvas_ptr->interp, path_cmd, NULL );
  TkCanvPsColor( canvas_ptr, ps_info_ptr, bg );
  Tcl_AppendResult( canvas_ptr->interp, "fill\n", NULL );

                // Draw Bottom Shade Lines
                //
  sprintf(
      path_cmd,
      "%g %g moveto %g 0 rlineto 0 %g rlineto %d setlinewidth\n",
      bar_ptr->biBbox[0], y2,
      deltax,
      -deltay,
      bar_ptr->biWidth
      );

  Tcl_AppendResult( canvas_ptr->interp, path_cmd, NULL );
  TkCanvPsColor(
      canvas_ptr,
      ps_info_ptr,
      bar_ptr->biRelief == TK_RELIEF_RAISED ? dark :
      bar_ptr->biRelief == TK_RELIEF_FLAT ? bg : light
      );
  Tcl_AppendResult( canvas_ptr->interp, "stroke\n", NULL );

                // Draw Top Shade Lines
                //
  sprintf(
      path_cmd,
      "%g %g moveto 0 %g rlineto %g 0 rlineto %d setlinewidth\n",
      bar_ptr->biBbox[0], y2,
      -deltay,
      deltax,
      bar_ptr->biWidth
      );

  Tcl_AppendResult( canvas_ptr->interp, path_cmd, NULL );
  TkCanvPsColor(
      canvas_ptr,
      ps_info_ptr,
      bar_ptr->biRelief == TK_RELIEF_RAISED ? light :
      bar_ptr->biRelief == TK_RELIEF_FLAT ? bg : dark
      );
  Tcl_AppendResult( canvas_ptr->interp, "stroke\n", NULL );
  }

return  status;
} // TkBarItem::GenPostScript


/************************************************************************
*       NAME:           TkBarItem::Register()                           *
*       HISTORY:                                                        *
*               18 Oct 93       Ron Lee                                 *
************************************************************************/
void
TkBarItem::Register()
{
Tk_CreateItemType( &_biItemType );
} // TkBarItem::Register
```

APPENDIX D. SOURCE LISTINGS FOR PIXMAP EXTENSIONS

D.1 TKPIXMAP.H

```
        /* "%Z% %M%  %I% %G%" */
#ifndef __tkPixmap_h__
#define __tkPixmap_h__
/*************************************************************************
*       NAME:           tkPixmap.h                                       *
*       HISTORY:                                                         *
*               14 Mar 94       re7@ornl.gov                             *
*               XWD File                                                 *
*               23 Dec 93       re7@ornl.gov                             *
*************************************************************************/
#include <X11/Xlib.h>


/*************************************************************************
*       NAME:           Tk_ReadXPM2File()                                *
*       PURPOSE:                                                         *
*               Read an XPM2 file and build the corresponding pixmap     *
*************************************************************************/
int                     Tk_ReadXPM2File(
                            Display *,
                            Drawable,
                            char *              filename,
                            unsigned int *      width_return,
                            unsigned int *      height_return,
                            Pixmap *
                            );


/*************************************************************************
*       NAME:           Tk_ReadXwdFile()                                 *
*       PURPOSE:                                                         *
*               Reads an XWD file into a pixmap                          *
*************************************************************************/
int                     Tk_ReadXwdFile(
                            Display *,
                            Drawable,
                            char *              filename,
                            unsigned int *      width_return,
                            unsigned int *      height_return,
                            Pixmap *
                            );


#endif
```

D.2 TKPIXMAP.C

```
static char *   sccs_id = "%Z% %M%  %I% %G%";
/*************************************************************************
*       NAME:           tkPixmap.c                                       *
*       HISTORY:                                                         *
*               14 Mar 94       re7@ornl.gov                             *
*               23 Dec 93       re7@ornl.gov                             *
*       PURPOSE:                                                         *
*               Facilities for processing XPM2 pixmap and XWD file       *
*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <memory.h>
#include <fcntl.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <X11/Xlib.h>
#include <X11/XWDFile.h>
#include "ByteOrder.h"
```

```
/**********************************************************************
*       NAME:           FindColor()                                   *
*       HISTORY:                                                      *
*               23 Dec 93       re7@ornl.gov                          *
**********************************************************************/
static u_long
FindColor( XColor * xcolors, int count, char ch )
{
  int           i;

for ( i = 0; i < count; i++ )
  {
  if ( ch == (char) xcolors[i].pad )
    break;
  }

return  ( i < count ? xcolors[i].pixel : 0 );
} /* FindColor */


/**********************************************************************
*       NAME:           Tk_ReadXPM2File()                             *
*       HISTORY:                                                      *
*               23 Dec 93       re7@ornl.gov                          *
**********************************************************************/
int
Tk_ReadXPM2File(
    Display *           display,
    Drawable            drawable,
    char *              filename,
    unsigned int *      width_return,
    unsigned int *      height_return,
    Pixmap *            pixmap_return
    )
{
  int           status = Success;

  FILE *        fp;
  Colormap      cmap;
  GC            gc;
  XGCValues     gcvalues;
  XColor *      xcolors;
  int           color_count, chars_per_pixel;
  int           row, col;
  char          line[ 256 ];


if ( display == NULL )
  status = BadAccess;

else if ( drawable == None )
  status = BadDrawable;

else if ( (fp = fopen( filename, "r" )) == NULL )
  status = BadName;

else
  {
  cmap = DefaultColormap( display, DefaultScreen( display ) );
  gc = DefaultGC( display, DefaultScreen( display ) );

                /*
                * Look For Signature
                */
  if ( fgets( line, sizeof(line), fp ) == NULL )
    status = BadName;

  else if ( memcmp( line, "! XPM2", 6 ) )
    status = BadValue;

  else
    {
                /*
```

```
                * Skip Comments, Read Parameters Line
                */
while ( fgets( line, sizeof(line), fp ) )
  {
  if ( line[0] != '!' && line[0] != '#' )  break;
  }

row = sscanf(
    line,
    "%d %d %d %d",
    width_return,
    height_return,
    &color_count,
    &chars_per_pixel
    );

                    /*
                    * Check Parameters
                    */
if ( row < 4 || chars_per_pixel > 1 )
  status = BadValue;

                        /*
                        * Allocate XColors Buffer
                        */
else if (
    (xcolors = (XColor *) calloc( color_count, sizeof(*xcolors))) == NULL
    )
  status ■ BadAlloc;

else
  {
                        /*
                        * Read Color Descriptions
                        */
  for ( row = 0; row < color_count; row++ )
    {
    if ( fgets( line, sizeof(line), fp ) )
      {
      xcolors[ row ].pad ■ line[0];

      sscanf(
          line + 5,
          "%4hx%4hx%4hx",
          &xcolors[ row ].red,
          &xcolors[ row ].green,
          &xcolors[ row ].blue
          );

      XAllocColor( display, cmap, xcolors + row );
      } /* if line successfully read */
    } /* for each color line */

                        /*
                        * Create Pixmap
                        */
  *pixmap_return = XCreatePixmap(
      display,
      drawable,
      *width_return,
      *height_return,
      DefaultDepth( display, DefaultScreen(display) )
      );

  if ( *pixmap_return <= LastExtensionError )
    status = BadAlloc;

  else
    {
                            /*
                            * Read Pixmap Values
                            */
    XSetFunction( display, gc, GXcopy );
```

```
        for ( row = 0; row < *height_return; row++ )
          {
          if ( fgets( line, sizeof(line), fp ) )
            {
            for ( col = 0; col < *width_return; col++ )
              {
              XSetForeground(
                  display,
                  gc,
                  FindColor( xcolors, color_count, line[ col ] )
                  );

              XDrawPoint( display, *pixmap_return, gc, col, row );
              } /* for each pixmap column */
            } /* if line read */
          } /* for each pixmap row */
        } /* else pixmap allocated */

      free( xcolors );
      } /* else parameters read, color buffer allocated */
    } /* else file signature found */

  fclose( fp );
  } /* else file opened */

return  status;
} /* Tk_ReadXPM2File */


/***********************************************************************
*       NAME:           ComputeImageSize()                             *
*       HISTORY:                                                       *
*               14 Mar 94       re7@ornl.gov                           *
***********************************************************************/
unsigned int
ComputeImageSize( XImage * image )
{
return  (
    image->format != ZPixmap ?
      image->bytes_per_line * image->height * image->depth
    :
      image->bytes_per_line * image->height
    );
} /* ComputeImageSize */


/***********************************************************************
*       NAME:           CopyImageToPixmap()                            *
*       HISTORY:                                                       *
*               14 Mar 94       re7@ornl.gov                           *
***********************************************************************/
void
CopyImageToPixmap(
    Display *           display,
    XImage *            image,
    int                 ncolors,
    XColor *            colors,
    Pixmap              pixmap
    )
{
  register int          x, y;
  register XColor *     color;
  GC                    gc;
  int                   screen = DefaultScreen( display );
  Colormap              cmap = DefaultColormap( display, screen );

gc = DefaultGC( display, screen );
XSetFunction( display, gc, GXcopy );

for ( x = 0; x < ncolors; x++ )
  colors[ x ].flags = 0;

for ( y = 0; y < image->height; y++ )
  for ( x = 0; x < image->width; x++ )
    {
    color = colors + XGetPixel( image, x, y );

    if ( ! color->flags )
      {
      color->flags = DoRed | DoGreen | DoBlue;
      XAllocColor( display, cmap, color );
      }

    XSetForeground( display, gc, color->pixel );
    XDrawPoint( display, pixmap, gc, x, y );
    }
} /* CopyImageToPixmap */


/***********************************************************************
*       NAME:           Tk_ReadXwdFile()                               *
*       HISTORY:                                                       *
*               14 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
Tk_ReadXwdFile(
    Display *           display,
    Drawable            drawable,
    char *              filename,
    unsigned int *      width_return,
    unsigned int *      height_return,
    Pixmap *            pixmap_return
    )
{
  int                   status = Success;

  FILE *                fp;
  GC                    gc;
  XGCValues             gcvalues;
  XWDColor *            xwd_colors;
  XWDFileHeader         xwd_header;
  XImage *              ximage;
  int                   swap_flag = MustSwap();
  unsigned int          size;
  int                   i;
  char *                buffer;

  Display               fake_display;
  ScreenFormat          fake_format;


if ( display == NULL )
  status = BadAccess;

else if ( drawable == None )
  status = BadDrawable;

else if ( (fp = fopen( filename, "r" )) == NULL )
  status = BadName;

                /*
                * Read Header
                */
else if ( (fread( (char *)&xwd_header, sizeof(xwd_header), 1, fp )) < 1 )
  status = BadAccess;

else
  {
  if ( swap_flag )
    Swap32( (char *)&xwd_header, sizeof(xwd_header) );

  if ( xwd_header.file_version != XWD_FILE_VERSION )
    status = BadMatch;

  else if ( xwd_header.header_size < sizeof(xwd_header) )
    status = BadAccess;
```

```
                          /*
                          * Create Image
                          */
    else
      {
                          /* Skip Name */
      fseek( fp, xwd_header.header_size - sizeof(xwd_header), SEEK_CUR );

                          /* Initialize For Image */
      fake_display.byte_order = (int) xwd_header.byte_order;
      fake_display.bitmap_unit = (int) xwd_header.bitmap_unit;
      fake_display.bitmap_bit_order = (int) xwd_header.bitmap_bit_order ;
      fake_display.pixmap_format = &fake_format;
      fake_display.nformats = 1;
      fake_format.depth = (int) xwd_header.pixmap_depth;
      fake_format.bits_per_pixel = (int) xwd_header.bits_per_pixel;

      ximage = XCreateImage(
          &fake_display, NULL,
          (int) xwd_header.pixmap_depth,
          (int) xwd_header.pixmap_format,
          (int) xwd_header.xoffset, NULL,
          *width_return = (unsigned int) xwd_header.pixmap_width,
          *height_return = (unsigned int) xwd_header.pixmap_height,
          (int) xwd_header.bitmap_pad,
          (int) xwd_header.bytes_per_line
          );

      size = ComputeImageSize( ximage );

                          /*
                          * Read Colormap Entries
                          */
      if (
          ximage == NULL  ||

          (xwd_colors = malloc( xwd_header.ncolors * sizeof(*xwd_colors) )) ==
              NULL  ||

          (buffer = malloc( size )) == NULL  ||

          fread(
              (char *)xwd_colors,
              sizeof(*xwd_colors),
              xwd_header.ncolors,
              fp
              ) < xwd_header.ncolors   ||

          fread( buffer, size, 1, fp ) < 1
          )
        status = BadAccess;

      else
        {
        ximage->red_mask = xwd_header.red_mask;
        ximage->green_mask = xwd_header.green_mask;
        ximage->blue_mask = xwd_header.blue_mask;

        if ( swap_flag )
          {
          for ( i = 0; i < xwd_header.ncolors; i++ )
            {
            xwd_colors[i].flags = DoRed | DoGreen | DoBlue;
            Swap32( (char *) &xwd_colors[i].pixel, sizeof(xwd_colors[i].pixel) );
            Swap16( (char *) &xwd_colors[i].red, 3 * sizeof(xwd_colors[i].red) );
            }
          }
                          /* Assume Visual Okay and Go On */
        ximage->data = buffer;

                          /*
                          * Create Pixmap
                          */
      *pixmap_return = XCreatePixmap(
          display,
          drawable,
          xwd_header.pixmap_width,
          xwd_header.pixmap_height,
          DefaultDepth( display, DefaultScreen(display) )
          );

      if ( *pixmap_return <= LastExtensionError )
        status = BadAlloc;

      else
        {
        CopyImageToPixmap(
            display,
            ximage,
            xwd_header.ncolors,
            (XColor *) xwd_colors,
            *pixmap_return
            );
        }

      if ( xwd_colors != NULL )
        free( xwd_colors );

      if ( buffer != NULL )
        free( buffer );

      if ( ximage != NULL )
        free( ximage );
      } /* colors and buffer allocated and read */
    } /* else versions match, all is well */

  fclose( fp );
  } /* else header read */

return  status;
} /* Tk_ReadXwdFile */
```

D.3 TKBITMAP.C MODIFICATIONS

```
        .
        .
        .
#include "tkPixmap.h"
        .
        .
        .
typedef struct {
    Pixmap bitmap;              /* X identifier for bitmap.  None means this
                                 * bitmap was created by Tk_DefineBitmap
                                 * and it isn't currently in use. */
    unsigned int width, height; /* Dimensions of bitmap. */
/*@@@@@*/
    unsigned int        depth;
    Display *display;           /* Display for which bitmap is valid. */
    int refCount;               /* Number of active uses of bitmap. */
    Tcl_HashEntry *hashPtr;     /* Entry in nameTable for this structure
                                 * (needed when deleting). */
} TkBitmap;
        .
        .
        .
Pixmap
Tk_GetBitmap(interp, tkwin, string)
    Tcl_Interp *interp;         /* Interpreter to use for error reporting. */
    Tk_Window tkwin;            /* Window in which bitmap will be used. */
    Tk_Uid string;              /* Description of bitmap.  See manual entry
                                 * for details on legal syntax. */
{
  int           depth = 1;
        .
        .
        .

    if ( *string == '%' )
      {
        Tcl_DString     buffer;
        int             result;

      if ( (string = Tcl_TildeSubst( interp, string + 1, &buffer )) == NULL )
        goto error;

      else
        {
        depth = Tk_Depth( tkwin );

        result = Tk_ReadXPM2File(
            nameKey.display,
            RootWindowOfScreen( Tk_Screen(tkwin) ),
            string,
            &width,
            &height,
            &bitmap
            );

        Tcl_DStringFree( &buffer );

        if ( result != Success )
          {
          Tcl_AppendResult(
              interp,
              "error reading XPM2 file \"",
              string,
              "\"",
              NULL
              );
          goto error;
          }
        }
      } /* if string begins with '%' */

    else if ( *string == '^' )
      {
        Tcl_DString     buffer;
        int             result;

      if ( (string = Tcl_TildeSubst( interp, string + 1, &buffer )) == NULL )
        goto error;

      else
        {
        depth = Tk_Depth( tkwin );

        result = Tk_ReadXwdFile(
            nameKey.display,
            RootWindowOfScreen( Tk_Screen(tkwin) ),
            string,
            &width,
            &height,
            &bitmap
            );

        Tcl_DStringFree( &buffer );

        if ( result != Success )
          {
          Tcl_AppendResult(
              interp,
              "error reading XWD file \"",
              string,
              "\"",
              NULL
              );

          goto error;
          }
        }
      } /* if string begins with '%' */

    else if (*string == '@') {
        Tcl_DString buffer;
        int result;

        string = Tcl_TildeSubst(interp, string + 1, &buffer);
        if (string == NULL) {
            goto error;
        }
        result = XReadBitmapFile(nameKey.display,
                RootWindowOfScreen(Tk_Screen(tkwin)), string, &width,
                &height, &bitmap, &dummy2, &dummy2);
        Tcl_DStringFree(&buffer);
        if (result != BitmapSuccess) {
            Tcl_AppendResult(interp, "error reading bitmap file \"", string,
                    "\"", (char *) NULL);
            goto error;
        }
    } else {
        predefHashPtr = Tcl_FindHashEntry(&predefTable, string);
        if (predefHashPtr == NULL) {
            Tcl_AppendResult(interp, "bitmap \"", string,
                    "\" not defined", (char *) NULL);
            goto error;
        }
        predefPtr = (PredefBitmap *) Tcl_GetHashValue(predefHashPtr);
        width = predefPtr->width;
        height = predefPtr->height;
        bitmap = XCreateBitmapFromData(nameKey.display,
                DefaultRootWindow(nameKey.display), predefPtr->source,
                width, height);
    }

    /*
     * Add information about this bitmap to our database.
     */

    bitmapPtr = (TkBitmap *) ckalloc(sizeof(TkBitmap));
    bitmapPtr->bitmap = bitmap;
    bitmapPtr->width = width;
    bitmapPtr->height = height;

    bitmapPtr->depth = depth;

    bitmapPtr->display = nameKey.display;
    bitmapPtr->refCount = 1;
    bitmapPtr->hashPtr = nameHashPtr;
    idKey.display = nameKey.display;
    idKey.pixmap = bitmap;
    idHashPtr = Tcl_CreateHashEntry(&idTable, (char *) &idKey,
            &new);
    if (!new) {
        panic("bitmap already registered in Tk_GetBitmap");
    }
    Tcl_SetHashValue(nameHashPtr, bitmapPtr);
    Tcl_SetHashValue(idHashPtr, bitmapPtr);
    return bitmapPtr->bitmap;

    error:
    Tcl_DeleteHashEntry(nameHashPtr);
    return None;
}
        .
        .
        .
```

D.4 TKBUTTON.C MODIFICATIONS

```
        .
        .
        .
static void
DisplayButton(clientData)
    ClientData clientData;      /* Information about widget. */
{
        .
        .
        .
        if ( depth > 1 )
          {
          XCopyArea(
              butPtr->display, butPtr->bitmap,
              pixmap, gc,
              0, 0, width, height,
              x, y
              );
          }
        else
          {
          XCopyPlane(
              butPtr->display, butPtr->bitmap,
              pixmap, gc,
              0, 0, width, height,
```

```
              x, y, 1
              );
          }
        .
        .
        .
```

D.5 TKCANVBMAP.C MODIFICATIONS

```
        .
        .
        .
static void
DisplayBitmap(canvasPtr, itemPtr, drawable)
    register Tk_Canvas *canvasPtr;      /* Canvas that contains item. */
    Tk_Item *itemPtr;                   /* Item to be displayed. */
    Drawable drawable;                  /* Pixmap or window in which to draw
                                         * item. */
{
    register BitmapItem *bmapPtr = (BitmapItem *) itemPtr;

    unsigned int        depth;

    if (bmapPtr->bitmap != None) {
        Tk_DepthOfBitmap(
            Tk_Display( canvasPtr->tkwin ),
            bmapPtr->bitmap,
            &depth
            );

        if ( depth > 1 )
          {
          XCopyArea(
              Tk_Display( canvasPtr->tkwin ), bmapPtr->bitmap, drawable,
              bmapPtr->gc, 0, 0,
              (u_int) bmapPtr->header.x2 - bmapPtr->header.x1,
              (u_int) bmapPtr->header.y2 - bmapPtr->header.y1,
              bmapPtr->header.x1 - canvasPtr->drawableXOrigin,
              bmapPtr->header.y1 - canvasPtr->drawableYOrigin
              );
          }

        else
          {
          XCopyPlane(Tk_Display(canvasPtr->tkwin), bmapPtr->bitmap, drawable,
                  bmapPtr->gc, 0, 0,
                  (unsigned int) bmapPtr->header.x2 - bmapPtr->header.x1,
                  (unsigned int) bmapPtr->header.y2 - bmapPtr->header.y1,
                  bmapPtr->header.x1 - canvasPtr->drawableXOrigin,
                  bmapPtr->header.y1 - canvasPtr->drawableYOrigin, 1);
          }
    }
}
        .
        .
        .
```

D.6 TKMENU.C MODIFICATIONS

```
        .
        .
        .
static void
DisplayMenu(clientData)
    ClientData clientData;      /* Information about widget. */
{
        .
        .
        .
            if ( depth > 1 )
              {
              XCopyArea(
              \   menuPtr->display, mePtr->bitmap, Tk_WindowId( tkwin ),
                  gc, 0, 0, width, height,
                  menuPtr->borderWidth + menuPtr->selectorSpace,
                  (int) (mePtr->y + (mePtr->height - height) / 2)
                  );
              }
            else
              {
              XCopyPlane(menuPtr->display, mePtr->bitmap, Tk_WindowId(tkwin),
                      gc, 0, 0, width, height,
                      menuPtr->borderWidth + menuPtr->selectorSpace,
                      (int) (mePtr->y + (mePtr->height - height)/2), 1);
              }
        .
        .
        .
```

D.7 TKMENUBUTTON.C MODIFICATIONS

```
        .
        .
        .
static void
DisplayMenuButton(clientData)
    ClientData clientData;       /* Information about widget. */
{
        .
        .
        .
        if ( depth > 1 )
          {
          XCopyArea(
              mbPtr->display, mbPtr->bitmap, pixmap,
              gc, 0, 0, width, height, x, y
              );
          }
        else
          {
          XCopyPlane(mbPtr->display, mbPtr->bitmap, pixmap,
                  gc, 0, 0, width, height, x, y, 1);
          }
        .
        .
        .
```

APPENDIX E. SOURCE LISTINGS FOR C++ TCL CLASSES

E.1 TCLFDBCLIENT.H

```
        /* %Z% %M%  %I% %G% */
#ifndef __TclFdbClient__
#define __TclFdbClient__
/*************************************************************************
*       NAME:           TclFdbClient.h                                   *
*       HISTORY:                                                         *
*               31 Mar 94       re7@ornl.gov                             *
*       CLASS:                                                           *
*               TclFdbClient                                             *
*************************************************************************/
extern "C"
{
#include <tcl.h>
}

#include <FdbClient.h>


/**************************************************************************
*)      CLASS:          TclFdbClient                                     (*
*)      PURPOSE:                                                         (*
*)              Tcl commands on top of an FdbClient object               (*
*************************************************************************/
class  TclFdbClient : public FdbClient
  {
                // Object Attributes
                //
protected:

  Tcl_Interp *          tfcInterp;
  char *                tfcName;                // Tcl variable name


                // Constructors and Destructor
                //
public:

  /***********************************************************************
  *     NAME:           constructor                                      *
  *     PURPOSE:                                                         *
  ***********************************************************************/
                        TclFdbClient(
                            Tcl_Interp *,
                            char *                  var_name,
                            char *                  auth_id,
                            char *                  auth_key
                            );


  /***********************************************************************
  *     NAME:           destructor                                       *
  *     PURPOSE:                                                         *
  ***********************************************************************/
  virtual               ~TclFdbClient();


                // Object Methods
                //
protected:
```

```
  /***********************************************************************
  *     NAME:           ProcessGetStatus()                               *
  *     PURPOSE:                                                         *
  *             Processes the "GetStatus" object command                 *
  ***********************************************************************/
  int                   ProcessGetStatus(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );


  /***********************************************************************
  *     NAME:           ProcessReadNextLine()                            *
  *     PURPOSE:                                                         *
  *             Processes the "ReadNextLine" object command              *
  ***********************************************************************/
  int                   ProcessReadNextLine(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );


  /***********************************************************************
  *     NAME:           ProcessSendCommand()                             *
  *     PURPOSE:                                                         *
  *             Processes the "SendCommand" object command               *
  ***********************************************************************/
  int                   ProcessSendCommand(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );


public:

  /***********************************************************************
  *     NAME:           DeleteObject()                                   *
  *     PURPOSE:                                                         *
  *             Processes the object command                             *
  ***********************************************************************/
  static void           DeleteObject( TclFdbClient * );


  /***********************************************************************
  *     NAME:           ProcessObjectCommand()                           *
  *     PURPOSE:                                                         *
  *             Processes the object command                             *
  *     NOTES:                                                           *
  *             'data' should be pointer/reference to the object         *
  *     OUTPUT:                                                          *
  *             Tcl status                                               *
  ***********************************************************************/
  static int            ProcessObjectCommand(
                            TclFdbClient *      ds_ptr,
                            Tcl_Interp *        interp,
                            int                 argc,
                            char *              argv[]
                            );
  /**********************************************************************
  *     NAME:           ProcessClassCommand()                           *
  *     PURPOSE:                                                        *
  *             Processes the class command to create objects          *
  *     OUTPUT:                                                         *
  *             Tcl status                                              *
  **********************************************************************/
  static int            ProcessClassCommand(
                            ClientData              data,
                            Tcl_Interp *            interp,
                            int                     argc,
                            char *                  argv[]
                            );

  /**********************************************************************
  *     NAME:           Register()                                      *
  *     PURPOSE:                                                        *
  *             Registers the class command in the interpreter         *
  **********************************************************************/
  static inline
  void                  Register( Tcl_Interp * interp )
                          {
                          Tcl_CreateCommand(
                              interp,
                              "FdbClient",
                              (Tcl_CmdProc *) ProcessClassCommand,
                              (ClientData) NULL,
                              (Tcl_CmdDeleteProc *) NULL
                              );
                          );
  };


#endif
```

E.2 TCLFDBCLIENT.CC

```
static char *   sccs_id = "%Z% %M%  %I% %G%";
/************************************************************************
*       NAME:           TclFdbClient.cc                                 *
*       HISTORY:                                                        *
*               31 Mar 94       re7@ornl.gov                            *
************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "TclFdbClient.h"


/************************************************************************
*       NAME:           TclFdbClient::TclFdbClient()                    *
*       HISTORY:                                                        *
*               31 Mar 94       re7@ornl.gov                            *
************************************************************************/
TclFdbClient::TclFdbClient(
    Tcl_Interp *        interp,
    char *              var_name,
    char *              auth_id,
    char *              auth_key
    ) :
    FdbClient( auth_id, auth_key ),
    tfcInterp( interp ),
    tfcName( strdup( var_name ) )
{
} // TclFdbClient::TclFdbClient


/************************************************************************
*       NAME:           TclFdbClient::~TclFdbClient()                   *
*       HISTORY:                                                        *
*               31 Mar 94       re7@ornl.gov                            *
************************************************************************/
TclFdbClient::~TclFdbClient()
{
if ( tfcName != NULL )
  free( tfcName );
} // TclFdbClient::~TclFdbClient


/************************************************************************
*       NAME:           TclFdbClient::ProcessGetStatus()                *
*       HISTORY:                                                        *
*               31 Mar 94       re7@ornl.gov                            *
************************************************************************/
int
TclFdbClient::ProcessGetStatus( Tcl_Interp * interp, int argc, char * argv[] )
{
  char          buffer[ 32 ];

sprintf( buffer, "%d", GetStatus() );
Tcl_SetResult( interp, buffer, TCL_VOLATILE );

return  TCL_OK;
```

```
} // TclFdbClient::ProcessGetStatus


/************************************************************************
*       NAME:           TclFdbClient::ProcessReadNextLine()             *
*       HISTORY:                                                        *
*               31 Mar 94       re7@ornl.gov                            *
************************************************************************/
int
TclFdbClient::ProcessReadNextLine(
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status = TCL_OK;
  char          buffer[ 32 ];
  char *        next_line;
  Timeval       wait_time;
  Timeval *     time_ptr = NULL;

        // Check For VarName Parameter
        //
if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult(
      interp,
      "Usage: client ReadNextLine varname [ wait-secs ]",
      TCL_STATIC
      );
  }

else
  {
  if ( argc > 1 )
    {
    wait_time = Timeval( atoi( argv[1] ), 0 );
    time_ptr = & wait_time;
    }

  if ( (next_line = (char *) ReadNextLine( time_ptr )) == NULL )
    {
    Tcl_SetVar( interp, argv[0], "", TCL_LEAVE_ERR_MSG );
    strcpy( buffer, "eof" );
    }

  else
    {
    Tcl_SetVar( interp, argv[0], next_line, TCL_LEAVE_ERR_MSG );
    sprintf( buffer, "%d", strlen( next_line ) );
    } // else read succeeded

  Tcl_SetResult( interp, buffer, TCL_VOLATILE );
  } // else parameters supplied

return  status;
} // TclFdbClient::ProcessReadNextLine


/************************************************************************
*       NAME:           TclFdbClient::ProcessSendCommand()              *
*       HISTORY:                                                        *
*               31 Mar 94       re7@ornl.gov                            *
************************************************************************/
int
TclFdbClient::ProcessSendCommand(
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status = TCL_OK;
```

```
  char          buffer[ 32 ];
  char *        args[ FdbMaxArgs ];
  int           i;
  int           send_status;

        // Check For CommandString Parameter
        //
if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult(
      interp,
      "Usage: client SendCommand command-string",
      TCL_STATIC
      );
  }

else
  {
                // Parse Arguments
                //
  strtok( argv[0], " \t" );
  for (
      i = 0, args[ i ] = strtok( NULL, " \t" );
      i < FdbMaxArgs && args[ i ] != NULL;
      args[ ++i ] = strtok( NULL, " \t" )
      )
    ;

  if ( (send_status = SendCommand( argv[0], args )) < 0 )
    status = TCL_ERROR;

  sprintf( buffer, "%d", send_status );
  Tcl_SetResult( interp, buffer, TCL_VOLATILE );
  } // else parameters supplied

return  status;
} // TclFdbClient::ProcessSendCommand


/************************************************************************
*       NAME:           TclFdbClient::ProcessObjectCommand()            *
*       HISTORY:                                                        *
*               31 Mar 94       re7@ornl.gov                            *
************************************************************************/
int
TclFdbClient::ProcessObjectCommand(
    TclFdbClient *      client_ptr,
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status;

        // Default Command is GetStatus
        //
if ( argc < 2 )
  status = client_ptr->ProcessGetStatus( interp, 0, NULL );

else if ( strcmp( argv[1], "GetStatus" ) == 0 )
  status = client_ptr->ProcessGetStatus( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "ReadNextLine" ) == 0 )
  status = client_ptr->ProcessReadNextLine( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "SendCommand" ) == 0 )
  status = client_ptr->ProcessSendCommand( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "Delete" ) == 0 )
  {
  status = TCL_OK;
  Tcl_DeleteCommand( interp, client_ptr->tfcName );
  }

else
  {
  status = TCL_ERROR;

  Tcl_AppendResult(
      interp,
      "Usage:\n",
      "\tFdbClient Delete\n",
      "\tFdbClient GetStatus (returns status code)\n",
      "\tFdbClient ReadNextLine varname [ wait-secs ] (returns eof on EOF)\n",
      "\tFdbClient SendCommand command-string\n",
      NULL );
  }

return  status;
} // TclFdbClient::ProcessObjectCommand


/***********************************************************************
*       NAME:           TclFdbClient::DeleteObject()                   *
*       HISTORY:                                                       *
*               31 Mar 94       re7@ornl.gov                           *
***********************************************************************/
void
TclFdbClient::DeleteObject( TclFdbClient * client_ptr )
{
if ( client_ptr != NULL )
  delete client_ptr;
} // TclFdbClient::DeleteObject


/***********************************************************************
*       NAME:           TclFdbClient::ProcessClassCommand()            *
*       HISTORY:                                                       *
*               31 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclFdbClient::ProcessClassCommand(
    ClientData,
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int                   status = TCL_OK;
  TclFdbClient *        new_item;
  char                  address_value[ 32 ];

        // Check For Necessary Parameters
        //
if ( argc < 4 )
  {
  Tcl_SetResult(
      interp,
      "Usage: FdbClient varname auth-id auth-key",
      TCL_STATIC
      );
  status = TCL_ERROR;
  }

        // Process Arguments
        //
else if (
    (new_item = new TclFdbClient( interp, argv[1], argv[2], argv[3] )) == NULL
    )
  {
  Tcl_SetResult( interp, "error allocating new client object", TCL_STATIC );
  status = TCL_ERROR;
  }

else
```

```
  {
                // Register Object Command
                //
  Tcl_CreateCommand(
      interp,
      argv[1],
      (Tcl_CmdProc *) ProcessObjectCommand,
      (ClientData) new_item,
      (Tcl_CmdDeleteProc *) DeleteObject
      );
  }

return  status;
} // TclFdbClient::ProcessClassCommand
```

E.3 TCLSAMPLER.H

```
        /* %Z% %M%  %I% %G% */
#ifndef __TclSampler__
#define __TclSampler__
/**********************************************************************
*       NAME:           TclSampler.h                                 *
*       HISTORY:                                                     *
*               17 Mar 94       re7@ornl.gov                         *
*       CLASS:                                                       *
*               TclSampler                                           *
*********************************************************************/
extern "C"
{
#include <tcl.h>
}

#include "Sampler.h"


/**********************************************************************
*)      CLASS:          TclSampler                                   (*
*)      PURPOSE:                                                     (*
*)              Tcl wrapper around Sampler                           (*
*********************************************************************/
class  TclSampler : public Sampler
  {
                // Object Attributes
                //
protected:

  Tcl_Interp *          tsInterp;
  char *                tsName;         // Tcl variable name


                // Constructors and Destructor
                //
public:

  /*******************************************************************
  *     NAME:           constructor                                  *
  *     PURPOSE:                                                     *
  ******************************************************************/
                        TclSampler(
                            Tcl_Interp *,
                            char *                name,
                            char *                filename
                            );

  /*******************************************************************
  *     NAME:           destructor                                   *
  *     PURPOSE:                                                     *
  ******************************************************************/
  virtual               ~TclSampler();


                // Object Methods
                //
protected:

  /*******************************************************************
  *     NAME:           ProcessComputeColorValue()                   *
  *     PURPOSE:                                                     *
  *             Processes the "ComputeColorValue" object command     *
  ******************************************************************/
```

```
int                     ProcessComputeColorValue(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

/**********************************************************************
*       NAME:           ProcessCreateBins()                          *
*       PURPOSE:                                                     *
*               Processes the "CreateBins" object command            *
*********************************************************************/
int                     ProcessCreateBins(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

/**********************************************************************
*       NAME:           ProcessDrawImage()                           *
*       PURPOSE:                                                     *
*               Processes the "DrawImage" object command             *
*********************************************************************/
int                     ProcessDrawImage(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

/**********************************************************************
*       NAME:           ProcessDrawSamples()                         *
*       PURPOSE:                                                     *
*               Processes the "DrawSamples" object command           *
*********************************************************************/
int                     ProcessDrawSamples(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

/**********************************************************************
*       NAME:           ProcessGetBins()                             *
*       PURPOSE:                                                     *
*               Processes the "GetBins" object command               *
*********************************************************************/
int                     ProcessGetBins(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );

/**********************************************************************
*       NAME:           ProcessGetImageAddr()                        *
*       PURPOSE:                                                     *
*               Processes the "GetImageAddr" object command          *
*********************************************************************/
int                     ProcessGetImageAddr(
                            Tcl_Interp *,
                            int         argc,
                            char *      argv[]
                            );


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

```
*     NAME:           ProcessGetLastSample()                               *
*     PURPOSE:                                                             *
*             Processes the "GetLastSample" object command                 *
***************************************************************************/
int                   ProcessGetLastSample(
                          Tcl_Interp *,
                          int           argc,
                          char *        argv[]
                          );

/***************************************************************************
*     NAME:           ProcessGetNormValue()                                *
*     PURPOSE:                                                             *
*             Processes the "GetNormValue" object command                  *
***************************************************************************/
int                   ProcessGetNormValue(
                          Tcl_Interp *,
                          int           argc,
                          char *        argv[]
                          );

/***************************************************************************
*     NAME:           ProcessGetSamples()                                  *
*     PURPOSE:                                                             *
*             Processes the "GetSamples" object command                    *
***************************************************************************/
int                   ProcessGetSamples(
                          Tcl_Interp *,
                          int           argc,
                          char *        argv[]
                          );

/***************************************************************************
*     NAME:           ProcessGetSampleRange()                              *
*     PURPOSE:                                                             *
*             Processes the "GetSampleRange" object command                *
***************************************************************************/
int                   ProcessGetSampleRange(
                          Tcl_Interp *,
                          int           argc,
                          char *        argv[]
                          );

/***************************************************************************
*     NAME:           ProcessRunSet()                                      *
*     PURPOSE:                                                             *
*             Processes the "RunSet" object command                        *
***************************************************************************/
int                   ProcessRunSet(
                          Tcl_Interp *,
                          int           argc,
                          char *        argv[]
                          );

/***************************************************************************
*     NAME:           ProcessSetSampleSize()                               *
*     PURPOSE:                                                             *
*             Processes the "SetSampleSize" object command                 *
***************************************************************************/
int                   ProcessSetSampleSize(
                          Tcl_Interp *,
                          int           argc,
                          char *        argv[]
                          );

public:
```

```
/***************************************************************************
*     NAME:           DeleteObject()                                       *
*     PURPOSE:                                                             *
*             Processes the object command                                 *
***************************************************************************/
static void           DeleteObject( TclSampler * );

/***************************************************************************
*     NAME:           ProcessObjectCommand()                               *
*     PURPOSE:                                                             *
*             Processes the object command                                 *
*     NOTES:                                                               *
*             'data' should be pointer/reference to the object             *
*     OUTPUT:                                                              *
*             Tcl status                                                   *
***************************************************************************/
static int            ProcessObjectCommand(
                          TclSampler *        ds_ptr,
                          Tcl_Interp *        interp,
                          int                 argc,
                          char *              argv[]
                          );

/***************************************************************************
*     NAME:           ProcessClassCommand()                                *
*     PURPOSE:                                                             *
*             Processes the class command to create objects                *
*     OUTPUT:                                                              *
*             Tcl status                                                   *
***************************************************************************/
static int            ProcessClassCommand(
                          ClientData          data,
                          Tcl_Interp *        interp,
                          int                 argc,
                          char *              argv[]
                          );

/***************************************************************************
*     NAME:           Register()                                           *
*     PURPOSE:                                                             *
*             Registers the class command in the interpreter               *
***************************************************************************/
static inline
void                  Register( Tcl_Interp * interp )
                        {
                        Tcl_CreateCommand(
                            interp,
                            "Sampler",
                            (Tcl_CmdProc *) ProcessClassCommand,
                            (ClientData) NULL,
                            (Tcl_CmdDeleteProc *) NULL
                            );
                        };
};

#endif
```

E.4 TCLSAMPLER.CC

```
static char *   sccs_id = "%Z% %M%  %I% %G%";
/************************************************************************
*       NAME:           TclSampler.cc                                   *
*       HISTORY:                                                        *
*               17 Mar 94       re7@ornl.gov                            *
************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <memory.h>
#include <ctype.h>
#include <errno.h>
#include "TclSampler.h"

extern "C"
{
#include <tclInt.h>
#include <tkInt.h>
#include <tkCanvas.h>
}


/************************************************************************
*       NAME:           TclSampler::TclSampler()                        *
*       HISTORY:                                                        *
*               17 Mar 94       re7@ornl.gov                            *
************************************************************************/
TclSampler::TclSampler( Tcl_Interp * interp, char * name, char * filename ) :
    Sampler( filename, Tk_Display( Tk_MainWindow( interp ) ) ),
    tsInterp( interp ),
    tsName( strdup( name ) )
{
} // TclSampler::TclSampler


/************************************************************************
*       NAME:           TclSampler::~TclSampler()                       *
*       HISTORY:                                                        *
*               17 Mar 94       re7@ornl.gov                            *
************************************************************************/
TclSampler::~TclSampler()
{
if ( tsName != NULL )
  free( tsName );
} // TclSampler::~TclSampler


/************************************************************************
*       NAME:           TclSampler::ProcessComputeColorValue()          *
*       HISTORY:                                                        *
*               28 Mar 94       re7@ornl.gov                            *
************************************************************************/
int
TclSampler::ProcessComputeColorValue(
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status = TCL_OK;
  XColor        xcolor;
  char          buffer[ 64 ];


if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Usage: Sampler ComputeColorValue value", TCL_STATIC );
  }

else
  {
  xcolor = ComputeColorValue( strtod( argv[0], NULL ) );

  sprintf(
      buffer,
      "#%.04hx%.04hx%.04hx",
      xcolor.red, xcolor.green, xcolor.blue
      );

  Tcl_SetResult( interp, buffer, TCL_VOLATILE );
  }

return  status;
} // TclSampler::ProcessComputeColorValue


/************************************************************************
*       NAME:           TclSampler::ProcessCreateBins()                 *
*       HISTORY:                                                        *
*               17 Mar 94       re7@ornl.gov                            *
************************************************************************/
int
TclSampler::ProcessCreateBins( Tcl_Interp * interp, int argc, char * argv[] )
{
  int           status = TCL_OK;
  int           sizes[4];
  int           i;


if ( argc < 4 )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Usage: Sampler CreateBins x y z t", TCL_STATIC );
  }

else
  {
  for ( i = 0; i < 4; i++ )
    sizes[i] = atoi( argv[i] );

  if ( CreateBins( sizes[0], sizes[1], sizes[2], sizes[3] ) < 0 )
    {
    status = TCL_ERROR;
    Tcl_SetResult( interp, "Error creating bins", TCL_STATIC );
    }
  }

Tcl_SetResult( interp, "", TCL_STATIC );

return  status;
} // TclSampler::ProcessCreateBins

/************************************************************************
*       NAME:           TclSampler::ProcessDrawImage()                  *
*       HISTORY:                                                        *
*               14 Apr 94       re7@ornl.gov                            *
************************************************************************/
int
TclSampler::ProcessDrawImage( Tcl_Interp * interp, int argc, char * argv[] )
```

```
{
  int                          status = TCL_OK;

  Tcl_HashEntry *              hash_ptr;
  Command *                    cmd_ptr;


        // Must Have Gotten Name of Canvas
        //
if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult(
      interp,
      "Usage: sampler DrawImage canvas",
      TCL_STATIC
      );
  }

        // Retrieve Canvas Command
        //
else if (
    (hash_ptr =
        Tcl_FindHashEntry( &((Interp *) interp)->commandTable, argv[0] ))
    == NULL  ||

    (cmd_ptr = (Command *) Tcl_GetHashValue( hash_ptr )) == NULL
    )
  {
  status = TCL_ERROR;
  Tcl_AppendResult(
      interp,
      "Error:  canvas ",
      argv[0],
      " unknown",
      NULL
      );
  }

        // Get Window and Draw
        //
else
  {
  BuildImage();

  DrawImage(
      Tk_Display( ((Tk_Canvas *)cmd_ptr->clientData)->tkwin ),
      Tk_WindowId( ((Tk_Canvas *)cmd_ptr->clientData)->tkwin ),
      0, 0
      );
  }

return  status;
} // TclSampler::ProcessDrawImage


/*************************************************************************
*       NAME:           TclSampler::ProcessDrawSamples()                 *
*       HISTORY:                                                         *
*               28 Mar 94       re7@ornl.gov                             *
*************************************************************************/
int
TclSampler::ProcessDrawSamples( Tcl_Interp * interp, int argc, char * argv[] )
{
  int                          status = TCL_OK;
  char *                       buffer;
  int                          i;
  SampleList::SampleBlock *    bptr;

  Tcl_HashEntry *              hash_ptr;
  Command *                    cmd_ptr;
  XColor                       xcolor;
  int                          block_size = 5;
```

```
  static char *                canvas_argv[] =
                                 {
                                 NULL,
                                 "create",
                                 "rectangle",
                                 NULL, NULL,           // 3, 4, 5, 6
                                 NULL, NULL,
                                 "-tags", "points",
                                 "-fill", NULL,        // 10
                                 "-outline", NULL,     // 12
                                 "-width", "0"
                                 );


        // Must Have Gotten Name of Canvas
        //
if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult(
      interp,
      "Usage: sampler DrawSamples canvas [block-size]",
      TCL_STATIC
      );
  }

        // Retrieve Canvas Command
        //
else if (
    (hash_ptr =
        Tcl_FindHashEntry( &((Interp *) interp)->commandTable, argv[0] ))
    == NULL  ||

    (cmd_ptr = (Command *) Tcl_GetHashValue( hash_ptr )) == NULL
    )
  {
  status = TCL_ERROR;
  Tcl_AppendResult(
      interp,
      "Error:  canvas ",
      argv[0],
      " unknown",
      NULL
      );
  }

        // Allocate Strings For Parameters
        //
else if (
    (canvas_argv[3] = new char[ 16 ]) == NULL  ||
    (canvas_argv[4] = new char[ 16 ]) == NULL  ||
    (canvas_argv[5] = new char[ 16 ]) == NULL  ||
    (canvas_argv[6] = new char[ 16 ]) == NULL  ||
    (canvas_argv[12] = canvas_argv[10] = new char[ 64 ]) == NULL
    )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error allocating command buffers", TCL_STATIC );
  }

        // Execute Rectangle Create Command For Each Sample
        //
else
  {
  if ( argc > 1 && (block_size = atoi( argv[1] )) == 0 )
    block_size = 2;

  canvas_argv[0] = argv[0];

                // Traverse Each Sample Block
                //
  for (
      bptr = (SampleList::SampleBlock *) sSamplesList.FirstItem();
      bptr != NULL;
      bptr ■ (SampleList::SampleBlock *) sSamplesList.NextItem()
      )
    {
    for ( i = 0; i < bptr->sbSize; i++ )
      {
      xcolor = ComputeColorValue( bptr->sbPtr[i].sValue );

      sprintf( canvas_argv[3], "%d", bptr->sbPtr[i].sIndexes[0] - block_size );
      sprintf( canvas_argv[4], "%d", bptr->sbPtr[i].sIndexes[1] - block_size );
      sprintf( canvas_argv[5], "%d", bptr->sbPtr[i].sIndexes[0] + block_size );
      sprintf( canvas_argv[6], "%d", bptr->sbPtr[i].sIndexes[1] + block_size );
      sprintf(
          canvas_argv[10],
          "%c%.04hx%.04hx%.04hx",
          '#',
          xcolor.red, xcolor.green, xcolor.blue
          );

/*
      cout <<
          canvas_argv[3] << "," <<
          canvas_argv[4] << "," <<
          canvas_argv[10] << endl;
*/

      cmd_ptr->proc( cmd_ptr->clientData, interp, 15, canvas_argv );
      //Tk_UpdateCmd( Tk_MainWindow( interp ), interp, 0, canvas_argv );
      }
    } // for each sample block

  delete [] canvas_argv[10];
  delete [] canvas_argv[6];
  delete [] canvas_argv[5];
  delete [] canvas_argv[4];
  delete [] canvas_argv[3];
  }

return  status;
} // TclSampler::ProcessDrawSamples


/***********************************************************************
*       NAME:           TclSampler::ProcessGetBins()                   *
*       HISTORY:                                                       *
*               17 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessGetBins( Tcl_Interp * interp, int argc, char * argv[] )
{
  int                    status = TCL_OK;
  char *                 buffer;
  int                    x, y, z, t;
  unsigned long          total_bins;
  Bin *                  bin_ptr;

        // Determine Number of Samples, Allow 32 Bytes For Each
        //
for ( total_bins = 1, x = 0; x < 4; x++ )
  total_bins *= sNumSlots[x];

if ( sBins == NULL )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error:  bins not allocated", TCL_STATIC );
  }

else if ( (buffer = (char *) malloc( total_bins * 32 )) == NULL )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error allocating return array", TCL_STATIC );
  }
```

```
else
  {
  buffer[0] = 0;
  for ( bin_ptr = sBins, t = 0; t < sNumSlots[3]; t++ )
    for ( z = 0; z < sNumSlots[2]; z++ )
      for ( y = 0; y < sNumSlots[1]; y++ )
        for ( x = 0; x < sNumSlots[0]; x++, bin_ptr++ )
          {
          sprintf(
              buffer,
              "%s ( %d %d %d %d %hd )",
              buffer,
              x, y, z, t,
              bin_ptr->GetTag()
              );
          }

  Tcl_SetResult( interp, buffer, TCL_DYNAMIC );
  }

return  status;
} // TclSampler::ProcessGetBins


/***********************************************************************
*       NAME:           TclSampler::ProcessGetImageAddr()              *
*       HISTORY:                                                       *
*               12 May 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessGetImageAddr( Tcl_Interp * interp, int argc, char * argv[] )
{
  int           status = TCL_OK;
  char          address[ 32 ];

        // Must Have Gotten Name of Canvas
        //
sprintf( address, "%01d", (long) sImage );
Tcl_SetResult( interp, address, TCL_VOLATILE );

return  status;
} // TclSampler::ProcessGetImageAddr


/***********************************************************************
*       NAME:           TclSampler::ProcessGetLastSample()             *
*       HISTORY:                                                       *
*               17 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessGetLastSample(
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int                           status = TCL_OK;
  char *                        buffer;
  int                           i;
  SampleList::SampleBlock *     bptr;


        // Retrieve Last Sample From List
        //
if ( (bptr = (SampleList::SampleBlock *) sSamplesList.LastItem()) == NULL )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error retrieving last sample block", TCL_STATIC );
  }

        // Determine Number of Samples, Allow 128 Bytes For Each
        //
else if ( (buffer = (char *) malloc( bptr->sbSize * 128 )) == NULL )
```

```
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error allocating return array", TCL_STATIC );
  }

else
  {
  buffer[0] = 0;

  for ( i = 0; i < bptr->sbSize; i++ )
    {
    sprintf(
        buffer,
        "%s ( %d %d %d %d %hd %g )",
        buffer,
        bptr->sbPtr[i].sIndexes[0],
        bptr->sbPtr[i].sIndexes[1],
        bptr->sbPtr[i].sIndexes[2],
        bptr->sbPtr[i].sIndexes[3],
        bptr->sbPtr[i].sTag,
        bptr->sbPtr[i].sValue
        );
    }

  Tcl_SetResult( interp, buffer, TCL_DYNAMIC );
  }

return  status;
} // TclSampler::ProcessGetLastSample

/***********************************************************************
*       NAME:           TclSampler::ProcessGetNormValue()             *
*       HISTORY:                                                       *
*               20 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessGetNormValue( Tcl_Interp * interp, int argc, char * argv[] )
{
  int           status = TCL_OK;
  char          buffer[ 32 ];
  double        value;


if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Usage: sampler GetNormValue value", TCL_STATIC );
  }

else
  {
  value = strtod( argv[0], NULL );
  sprintf(
      buffer,
      "%g",
      (value - sSampleRange.drMin) / (sSampleRange.drMax - sSampleRange.drMin)
      );

  Tcl_SetResult( interp, buffer, TCL_VOLATILE );
  }

return  status;
} // TclSampler::ProcessGetNormValue

/***********************************************************************
*       NAME:           TclSampler::ProcessGetSamples()               *
*       HISTORY:                                                       *
*               17 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessGetSamples( Tcl_Interp * interp, int argc, char * argv[] )
```

```
{
  int                             status = TCL_OK;
  char *                          buffer;
  int                             i;
  SampleList::SampleBlock *       bptr;


        // Determine Number of Samples, Allow 128 Bytes For Each
        //
if ( (buffer = (char *) malloc( sTotalSamples * 128 )) == NULL )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Error allocating return array", TCL_STATIC );
  }

else
  {
  buffer[0] = 0;

  for (
      bptr = (SampleList::SampleBlock *) sSamplesList.FirstItem();
      bptr != NULL;
      bptr = (SampleList::SampleBlock *) sSamplesList.NextItem()
      )
    {
    for ( i = 0; i < bptr->sbSize; i++ )
      {
      sprintf(
          buffer,
          "%s ( %d %d %d %d %hd %g )",
          buffer,
          bptr->sbPtr[i].sIndexes[0],
          bptr->sbPtr[i].sIndexes[1],
          bptr->sbPtr[i].sIndexes[2],
          bptr->sbPtr[i].sIndexes[3],
          bptr->sbPtr[i].sTag,
          bptr->sbPtr[i].sValue
          );
      }
    } // for each sample block

  Tcl_SetResult( interp, buffer, TCL_DYNAMIC );
  }

return  status;
} // TclSampler::ProcessGetSamples

/***********************************************************************
*       NAME:           TclSampler::ProcessGetSampleRange()           *
*       HISTORY:                                                       *
*               17 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessGetSampleRange(
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status = TCL_OK;
  char          buffer[ 64 ];


sprintf( buffer, "%g %g", sSampleRange.drMin, sSampleRange.drMax );

Tcl_SetResult( interp, buffer, TCL_VOLATILE );

return  status;
} // TclSampler::ProcessGetSampleRange

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

```
*       NAME:           TclSampler::ProcessRunSet()                    *
*       HISTORY:                                                       *
*               17 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessRunSet( Tcl_Interp * interp, int argc, char * argv[] )
{
  int           status = TCL_OK;
  Boolean       check_flag, learn_flag;
  char          buffer[ 32 ];

if ( argc < 2 )
  {
  status = TCL_ERROR;
  Tcl_SetResult(
      interp,
      "Usage:  object RunSet check-flag learn-flag",
      TCL_STATIC
      );
  }

else
  {
  check_flag = atoi( argv[0] );
  learn_flag = atoi( argv[1] );
  RunSet( check_flag, learn_flag );

  sprintf( buffer, "%d %d", sInCount, sOutCount );
  Tcl_SetResult( interp, buffer, TCL_VOLATILE );
  }

return  status;
} // TclSampler::ProcessRunSet


/**********************************************************************
*       NAME:           TclSampler::ProcessSetSampleSize()             *
*       HISTORY:                                                       *
*               17 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessSetSampleSize( Tcl_Interp * interp, int argc, char * argv[] )
{
  int           status = TCL_OK;
  int           size;

if ( argc < 1 )
  {
  status = TCL_ERROR;
  Tcl_SetResult( interp, "Usage: Sampler SetSampleSize size", TCL_STATIC );
  }

else
  {
  size = atoi( argv[0] );
  SetSampleSize( size );
  Tcl_SetResult( interp, "", TCL_STATIC );
  }

return  status;
} // TclSampler::ProcessSetSampleSize


/**********************************************************************
*       NAME:           TclSampler::ProcessObjectCommand()             *
*       HISTORY:                                                       *
*               17 Mar 94       re7@ornl.gov                           *
***********************************************************************/
int
TclSampler::ProcessObjectCommand(
    TclSampler *        ptr,
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int           status;

if ( argc < 2 )
  status = TCL_OK;

else if ( strcmp( argv[1], "ComputeColorValue" ) == 0 )
  status = ptr->ProcessComputeColorValue( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "CreateBins" ) == 0 )
  status = ptr->ProcessCreateBins( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "DrawSamples" ) == 0 )
  status = ptr->ProcessDrawSamples( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "DrawImage" ) == 0 )
  status = ptr->ProcessDrawImage( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "GetBins" ) == 0 )
  status = ptr->ProcessGetBins( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "GetImageAddr" ) == 0 )
  status = ptr->ProcessGetImageAddr( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "GetLastSample" ) == 0 )
  status = ptr->ProcessGetLastSample( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "GetNormValue" ) == 0 )
  status = ptr->ProcessGetNormValue( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "GetSamples" ) == 0 )
  status = ptr->ProcessGetSamples( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "GetSampleRange" ) == 0 )
  status = ptr->ProcessGetSampleRange( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "RunSet" ) == 0 )
  status = ptr->ProcessRunSet( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "SetSampleSize" ) == 0 )
  status = ptr->ProcessSetSampleSize( interp, argc - 2, argv + 2 );

else if ( strcmp( argv[1], "Delete" ) == 0 )
  {
  status = TCL_OK;
  Tcl_DeleteCommand( interp, ptr->tsName );
  }

else
  {
  status = TCL_ERROR;
  Tcl_AppendResult(
      interp,
      "Usage:\n",
      "\tSampler CreateBins x y z t\n",
      "\tSampler DrawSamples canvas [ block-size ]\n",
      "\tSampler DrawImage canvas\n",
      "\tSampler GetNormValue\n",
      "\tSampler GetSamples\n",
      "\tSampler GetSampleRange\n",
      "\tSampler RunSet flag\n",
      "\tSampler SetSampleSize size\n",
      NULL
      );
  }

return  status;
} // TclSampler::ProcessObjectCommand


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

```
*       NAME:           TclSampler::DeleteObject()                        *
*       HISTORY:                                                          *
*               17 Mar 94       re7@ornl.gov                              *
*************************************************************************/
void
TclSampler::DeleteObject( TclSampler * ptr )
{
if ( ptr != NULL )
  {
  Tcl_UnsetVar(
      ptr->tsInterp,
      ptr->tsName,
      TCL_GLOBAL_ONLY | TCL_LEAVE_ERR_MSG
      );

  delete  ptr;
  }
} // TclSampler::DeleteObject


/*************************************************************************
*       NAME:           TclSampler::ProcessClassCommand()                 *
*       HISTORY:                                                          *
*               17 Mar 94       re7@ornl.gov                              *
*************************************************************************/
int
TclSampler::ProcessClassCommand(
    ClientData,
    Tcl_Interp *        interp,
    int                 argc,
    char *              argv[]
    )
{
  int                   status = TCL_OK;
  TclSampler *          new_item;
  int                   i;
  char                  address_value[ 32 ];


        // Check For Minimal Arguments
        //
if ( argc < 3 )
  {
  Tcl_SetResult(
      interp,
      "Usage:  Sampler var-name file-name\n",
      TCL_STATIC
      );
  status = TCL_ERROR;
  }

        // Process Arguments
        //
else if ( (new_item = new TclSampler( interp, argv[1], argv[2] )) == NULL )
  {
  Tcl_SetResult( interp, "error allocating", TCL_STATIC );
  status = TCL_ERROR;
  }

else
  {
                // Store Address as Variable Value
                //
  sprintf( address_value, "%0x", new_item );
  Tcl_SetVar(
      interp,
      argv[1],
      address_value,
      TCL_GLOBAL_ONLY | TCL_LEAVE_ERR_MSG
      );

                // Register object command
                //
```

```
  Tcl_CreateCommand(
      interp,
      argv[1],
      (Tcl_CmdProc *) ProcessObjectCommand,
      (ClientData) new_item,
      (Tcl_CmdDeleteProc *) DeleteObject
      );
  } // else we have arguments

return  status;
} // TclSampler::ProcessClassCommand
```

APPENDIX F. SOURCE LISTINGS FOR COPROCESSES

F.1 PERMUTE_VIEW.CC

```
static char *   sccs_id = "%Z% %M%  %I% %G%";
/************************************************************************
*       NAME:           permute_view.cc                                 *
*       HISTORY:                                                        *
*               26 May 94       re7@ornl.gov                            *
*       PURPOSE:                                                        *
*               Randomly permute image for display sampling             *
************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <math.h>
#include <sys/time.h>

#ifndef __osf__
#include <sys/types.h>
#include <sys/filio.h>
#else
#include <sys/ioctl.h>
#endif

#include <time.h>
#include <iostream.h>
#include <X11/bitmaps/xlogo32>
#include <X11/cursorfont.h>
#include <Time.h>
#include <UnixPort.h>
#include "XwdFile.h"
#include "X_Window.h"


#define Max( A, B )     ((A) > (B) ? (A) : (B))


/************************************************************************
*       TYPE:           ImagePoint                                      *
************************************************************************/
struct  ImagePoint
  {
  int           x;
  int           y;
  int           pixel;

  inline                ImagePoint( int xin = 0, int yin = 0, int pixelin ■ 0 )
                          {
                          x = xin;
                          y = yin;
                          pixel = pixelin;
                          };
  };


/************************************************************************
*       NAME:           ReadSample()                                    *
*       PURPOSE:                                                        *
************************************************************************/
void
ReadSample(
    register ImagePoint *       pts,
    X_Image &                   sample,
    int                         total_points,
    int                         sample_size,
    int                         pt_size_factor
    )
{
  register int  i, x, y;
  int           pt_size;


if ( pts != NULL )
  {
  sample.Clear();

  pt_size_factor = Max( pt_size_factor, 1 );

  pt_size = (int)sqrt( (double)total_points / sample_size ) / pt_size_factor;
  pt_size ■ Max( pt_size, 1 );

  for ( i = 0; i < sample_size; i++ )
    {
    for ( y = 0; y < pt_size; y++ )
      for ( x = 0; x < pt_size; x++ )
        {
        sample.PutPixel( pts[i].x + x, pts[i].y + y, pts[i].pixel );
        }
    }
  }
} // ReadSample


/************************************************************************
*       NAME:           main()                                          *
*       NOTES:                                                          *
*               We store the image in memory, assuming we'll have       *
*               the swap space to do so, though performance will        *
*               degrade.                                                *
************************************************************************/
int
main( int argc, char * argv[] )
{
  int                   x, y;

  X_TopWindow           xwin;

  Display *             display;
  int                   screen;

  XwdFile *             xwd_file = NULL;
  X_Image               sample_image;
  int                   counter;

  u_char *              data_ptr;
  int                   total_points;
  int                   sample_size;
  int                   pt_size_factor;
  ImagePoint *          points;
  Bool                  finished;


        // Process Command Line
        //
if ( argc < 2 )
  cerr << "Usage: permute_view xwd-file\n";
```

```
        // Open Display
        //
else if ( (display = XOpenDisplay( NULL )) == NULL )
  cerr << "Error:  could not open display\n";

else
  {
  screen = DefaultScreen( display );

                // Create Window
                //
  xwin.Create(
      display,
      RootWindow( display, screen ),
      100, 100,
      400, 400
      );

  xwin.DefineCursor( XCreateFontCursor( display, XC_iron_cross ) );

  xwin.SetWMProps(
      argv[0], argv[0],
      argv, argc,
      XCreateBitmapFromData(
          display,
          xwin.GetWindow(),
          xlogo32_bits,
          xlogo32_width,
          xlogo32_height
          )
      );

                // Select Input and Map
                //
  xwin.SelectInput(
      ButtonPressMask | ButtonReleaseMask | ExposureMask | StructureNotifyMask
      );
  xwin.Map();

                // Open Xwd File
                //
  if (
      (xwd_file = new XwdFile( display, argv[1] )) == NULL ||
      ! xwd_file->IsOpen()
      )
    cerr << "Error:  could not read xwd file '" << argv[1] << "'\n";

                // Allocate Point Array
                //
  else if (
      (points = new ImagePoint[
          total_points =
          xwd_file->GetImage().GetWidth() * xwd_file->GetImage().GetHeight()
          ]) == NULL
      )
    cerr << "Error:  could not allocate point array\n";

  else
    {
      ImagePoint *      ptr;
      ImagePoint        temp;

                        // Read Source Image Values
                        //
    cout << "reading source image " << argv[1] << " ...\n";
    for (
        y = 0, counter = 0, data_ptr = xwd_file->GetImage().GetData();
        y < xwd_file->GetImage().GetHeight();
        y++, data_ptr += xwd_file->GetImage().GetBytesPerLine()
        )
      {
      for ( x = 0; x < xwd_file->GetImage().GetWidth(); x++ )
        points[ counter++ ] = ImagePoint( x, y, data_ptr[x] );
```

```
  }
cout << "read complete\n";

                    // Permute Source Values
                    //
cout << "permuting source image ...\n";
srand48( time( NULL ) );
for (
    counter = total_points, ptr = points;
    counter > 0;
    counter--, ptr++
    )
  {
  x = (int) (drand48() * counter);
  temp = *ptr;
  *ptr = *(ptr + x);
  *(ptr + x) = temp;
  }
cout << "permutation complete\n";

                    // Create Sample Image
                    //
sample_image.Create(
    display,
    xwd_file->GetImage().GetWidth(),
    xwd_file->GetImage().GetHeight(),
    xwd_file->GetImage().GetBytesPerLine()
    );

if ( ! sample_image.IsCreated() )
  cerr << "Error:  could not create sample image\n";

else
  {
                            // Output TotalPoints
                            //
  sample_size = total_points / 10;
  pt_size_factor = 1;
  cout.form( "TotalPoints/%d\nSampleSize/%d\n", total_points, sample_size );
  cout << flush;

  ReadSample(
      points,
      sample_image,
      total_points,
      sample_size,
      pt_size_factor
      );

  xwin.Resize(
      xwd_file->GetImage().GetWidth(),
      xwd_file->GetImage().GetHeight()
      );
  XFlush( display );

                            // Event Loop
                            //
  for ( finished = False; !finished; )
    {
      fd_set        read_fds;

    FD_ZERO( &read_fds );
    FD_SET( display->fd, &read_fds );
    FD_SET( 0, &read_fds );

    switch(
        select( getdtablesize(), &read_fds, NULL, NULL, NULL )
        )
      {
      case -1:
        perror( "select() error" );
        break;
```

```
          case 0:
            break;

          default:
            if ( FD_ISSET( display->fd, &read_fds ) )
              {
                XEvent  event;

              XNextEvent( display, &event );

              if (
                  (event.type == Expose || event.type == ConfigureNotify ||
                   event.type == MapNotify) &&
                  event.xexpose.window == xwin.GetWindow()
                  )
                sample_image.Put( xwin.GetWindow() );
              }

            else if (FD_ISSET( 0, &read_fds ) )
              {
                char    line[ 80 ];
                char *  token;
                int     new_size;

              if ( fgets( line, sizeof(line), stdin ) )
                {
                if ( (token = strchr( line, '/' )) != NULL )
                  *token++ = 0;

                if ( sscanf( line, "%d", &new_size ) > 0 )
                  sample_size = new_size;

                if ( sscanf( token, "%d", &new_size ) > 0 )
                  pt_size_factor = new_size;

                if ( sample_size == 0 )
                  finished = True;

                else
                  {
                  ReadSample(
                      points,
                      sample_image,
                      total_points,
                      sample_size,
                      pt_size_factor
                      );

                  sample_image.Put( xwin.GetWindow() );
                  }
                }
              }
          } // switch
        } // for
      } // else sample image created
    } // else XwdFile object and ImagePoints allocated
  } // else display opened

delete xwd_file;
} // main
```

F.2 PERMUTEVIEW.TCL

```
#!/usr/local/bin/wish -f


#-----------------------------------------------------------------------
#-      NAME:           WriteValue()                                    -
#-----------------------------------------------------------------------
proc  WriteValue { fid { one 0 } { two 0 } } \
{
if { "$one" == "0" } \
  {
  set factor [set size 0]
  } \
else \
  {
  set size [ $one get ];
  set factor [ $two get ];
  }

puts stdout "$size/$factor";
puts $fid "$size/$factor";
flush $fid;
}
# WriteValue


#-----------------------------------------------------------------------
#-      NAME:           BuildWindow()                                   -
#-----------------------------------------------------------------------
proc  BuildWindow { fid total size } \
{
label .l \
    -text "$total Total Points";

scale .sample_size \
    -activeforeground gray60 \
    -command "set dummy" \
    -from 1 \
    -label "Sample Size" \
    -length 500 \
    -orient hor \
    -showvalue true \
    -tickinterval 0 \
    -to $total;

.sample_size set $size;

scale .pt_factor \
    -activeforeground gray60 \
    -command "set dummy" \
    -from 1 \
    -to 10 \
    -label "Pt Size Quotient" \
    -length 200 \
    -orient hor \
    -showvalue true \
    -tickinterval 1;


button .draw_button \
    -activebackground gray60 \
    -command "WriteValue $fid .sample_size .pt_factor" \
    -text "Re-Draw";

button .quit_button \
    -activebackground gray60 \
    -command "WriteValue $fid; exit" \
    -text "Quit";
```

```
pack .l \
    -side top;

pack .sample_size \
    -side top \
    -ipadx 20 \
    -expand yes \
    -fill both;

pack .pt_factor \
    -side top \
    -ipadx 20;

pack .draw_button \
    -side top \
    -padx 5 \
    -fill x;

pack .quit_button \
    -side top \
    -padx 5 \
    -fill x;

wm deiconify .;
}
# BuildWindow


#-----------------------------------------------------------------------
#-      NAME:           ReadSizes()                                     -
#-----------------------------------------------------------------------
proc  ReadSizes { fid } \
{
set total_points 0;
set sample_size 0;
set count 0;

while { ($total_points == 0 || $sample_size == 0) && $count > -1 } \
  {
  set count [gets $fid line];
  set linelist [split $line "/"];

  if { "[lindex $linelist 0]" == "TotalPoints" } \
    {
    set total_points [lindex $linelist 1];
    } \
  elseif { "[lindex $linelist 0]" == "SampleSize" } \
    {
    set sample_size [lindex $linelist 1];
    }
  }

return  [list $total_points $sample_size];
}
# ReadSizes


#-----------------------------------------------------------------------
#-      NAME:           main()                                          -
#-----------------------------------------------------------------------

wm withdraw .
wm title . "Permute View";

if { $argc < 1 } \
  {
  puts stderr "Usage: PermuteView xwd-file";
  } \
else \
  {
  set cmd "|/usr2/src/XObjects/permute_view [lindex $argv 0]";

  if { [catch { set xpipe [open $cmd "r+"] }] } \
    {
    puts stderr "Error spawning permute_view"
    } \
  else \
    {
    set value_list [ReadSizes $xpipe];
    puts stdout "... $value_list";
    }

  set total_points [lindex $value_list 0];
  set sample_size [lindex $value_list 1];

  if { $total_points == 0 || $sample_size == 0 } \
    {
    puts stderr "Error reading from permute_view";
    } \
  else \
    {
    BuildWindow $xpipe $total_points $sample_size;
    }
  }
```

APPENDIX G. SOURCE LISTINGS FOR OBJECT-ORIENTED TCL CONSTRUCTS

```
#-      @(#) dp_class.tcl  1.2 12/15/93
#----------------------------------------------------------------------
#-      NAME:           dp_class.tcl                                  -
#-      HISTORY:                                                      -
#-              15 Dec 93       re7@ornl.gov                          -
#-        Migration to dp_object style                                -
#-              14 Dec 93       re7@ornl.gov                          -
#-               8 Dec 93       re7@ornl.gov                          -
#-      PURPOSE:                                                      -
#-              Object-oriented style environment                     -
#----------------------------------------------------------------------
#       -- Globals for the Class Manager
global  _ClassMgr_
set _ClassMgr_(debug) 0


#----------------------------------------------------------------------
#-      NAME:           class()                                       -
#-      PURPOSE:                                                      -
#----------------------------------------------------------------------
proc  class { class_name { super_class none } attrs } \
{
dp_objectCreateClass $class_name $super_class $attrs
}
# class


#----------------------------------------------------------------------
#-      NAME:           dp_objectCreateClass()                        -
#-      PURPOSE:                                                      -
#-              Defines the class by creating global array with parent -
#-              class name (isa) and attribute list (attributes)      -
#----------------------------------------------------------------------
proc dp_objectCreateClass { class_name { super_class none } attrs } \
{
  global  _ClassMgr_
  global  $class_name


if { $_ClassMgr_(debug) } \
  {
  puts stdout \
"\
\[dp_objectCreateClass\] called with $class_name, $super_class, $attrs\
"
  }

set result ""

if { [info exists _ClassMgr_(classes,$class_name)] } \
  {
  puts stderr \
      "Error:  a class with name $class_name has already been declared"
  } \

elseif { \
    "$super_class" != "none" && \
    ! [info exists _ClassMgr_(classes,$super_class)] \
    } \
  {
  puts stderr \
      "Error:  superclass $super_class has not been declared"
  } \

else \
  {
```

```
#       -- Build Class Array
  set _ClassMgr_(classes,$class_name) $class_name

  set ${class_name}(isa) $super_class
  set ${class_name}(attributes) $attrs

#       -- Build the Class Command For Object Declarations
  proc $class_name { args } \
    "
    foreach item \$args \{ dp_objectCreateObject $class_name \$item \}
    "

  set result $class_name
  }

return  $result
}
# dp_objectCreateClass


#----------------------------------------------------------------------
#-      NAME:           dp_objectCreateObject()                       -
#----------------------------------------------------------------------
proc dp_objectCreateObject { class_name object_name } \
{
  global  _ClassMgr_
  global  $class_name
  global  $object_name


if { $_ClassMgr_(debug) } \
  {
  puts stdout \
      "\[dp_objectCreateObject\] called with $class_name, $object_name"
  }

set result ""

#       -- Check For Class
if { ! [info exists _ClassMgr_(classes,$class_name)] } \
  {
  puts stderr "Error:  class $class_name has not been defined"
  } \

else \
  {
#               -- Set Isa
  set ${object_name}(isa) $class_name

#               -- Build Class Hierarchy List
  set class_list $class_name
  for \
      { set parent [set ${class_name}(isa)]; global $parent } \
      { "$parent" != "none" } \
      { set parent [set ${parent}(isa)]; global $parent } \
    {
    lappend class_list $parent
    }

#               -- Set Object Attributes
  for \
      { set index [expr [llength $class_list]-1] } \
      { $index >= 0 } \
      { incr index -1 } \
    {
    set parent [lindex $class_list $index]
    foreach item [set $(parent)(attributes)] \
      {
      if ( [set len [llength $item]] > 1 ) \
        ( set $(object_name)([lindex $item 0]) [lindex $item 1] ) \

      elseif ( $len == 1 ) \
        ( set $(object_name)([lindex $item 0]) "" )
      )
    )

#                 -- Build Object Command
  proc $object_name ( op args ) \
    "
    dp_objectRunMethod $object_name \$op \$args
    "

  set result $object_name
  )

return  $result
)
# dp_objectCreateObject


#-----------------------------------------------------------------------
#-      NAME:           dp_objectExists()                              -
#-----------------------------------------------------------------------
proc  dp_objectExists ( object_name ) \
(
if ( "[info proc $object_name]" == "" ) ( return 0 ) \
else ( return 1 )
)
# dp_objectExists


#-----------------------------------------------------------------------
#-      NAME:           dp_objectFree()                                -
#-----------------------------------------------------------------------
proc  dp_objectFree ( object_name ) \
(
  global $object_name

rename $object_name "";
unset $object_name;
)
# dp_objectFree


#-----------------------------------------------------------------------
#-      NAME:           dp_objectSlotFree()                            -
#-----------------------------------------------------------------------
proc  dp_objectSlotFree ( object_name slot ) \
(
  global $object_name

catch ( unset $(object_name)($slot) );
)
# dp_objectSlotFree


#-----------------------------------------------------------------------
#-      NAME:           dp_objectSlot()                                -
#-----------------------------------------------------------------------
proc  dp_objectSlot ( object_name slot ) \
(
  global $object_name

return  [set $(object_name)($slot)];
)
# dp_objectSlot


#-----------------------------------------------------------------------
```

```
#-      NAME:           dp_objectSlotSet()                             -
#-----------------------------------------------------------------------
proc  dp_objectSlotSet ( object_name slot value ) \
(
  global $object_name

return  [set $(object_name)($slot) $value];
)
# dp_objectSlotSet


#-----------------------------------------------------------------------
#-      NAME:           dp_objectSlotAppend()                          -
#-----------------------------------------------------------------------
proc  dp_objectSlotAppend ( object_name slot value ) \
(
  global $object_name

return  [lappend $(object_name)($slot) $value];
)
# dp_objectSlotAppend


#-----------------------------------------------------------------------
#-      NAME:           dp_objectSlots()                               -
#-----------------------------------------------------------------------
proc  dp_objectSlots ( object_name ) \
(
  global $object_name

return  [array names $object_name];
)
# dp_objectSlots


#-----------------------------------------------------------------------
#-      NAME:           dp_objectConfigure()                           -
#-----------------------------------------------------------------------
proc  dp_objectConfigure ( object_name args ) \
(
  global $object_name

set result ""
set argc [llength $args]

if ( $argc < 1 ) \
  (
  foreach attr [array names $object_name] \
    (
    lappend result [list -$attr () [set $(object_name)($attr)]]
    )
  ) \

elseif ( $argc == 1 ) \
  (
  set attr [string trimleft [lindex $args 0] \-]
  if ( [catch ( set value [set $(object_name)($attr)] ) ] != 0 ) \
    (
    puts stderr "Error:  slot $attr not defined for object $object_name"
    ) \

  else \
    (
    set result [list -$attr () $value]
    )
  ) \

else \
  (
  incr argc -1

  for \
      ( set aindex 0 ) \
      { $aindex < $argc } \
      { incr aindex } \
    {
    set attr [string trimleft [lindex $args $aindex] \-]
    incr aindex
    set value [lindex $args $aindex]
    set ${object_name}($attr) $value
    }
  }

return  $result
}
# dp_objectConfigure


#-----------------------------------------------------------------------
#-      NAME:           dp_objectRunMethod()                           -
#-----------------------------------------------------------------------
proc dp_objectRunMethod { object_name op arg_list } \
{
  global  _ClassMgr_
  global  $object_name

if { $_ClassMgr_(debug) } \
  {
  puts stdout \
      "\[dp_objectRunMethod\] called with $object_name, $op, $arg_list"
  }

set result ""

#               -- Find Method
set proc_name ""
for \
    { set parent [set ${object_name}(isa)] } \
    { "$parent" != "none" } \
    { set parent [set ${parent}(isa)] } \
  {
    global  $parent

  if { "[info procs $parent.$op]" != "" } \
    {
    set proc_name $parent.$op
    break
    }
  }

if { "$proc_name" == "" } \
  {
  puts stderr "Error:  method $op not defined for $object_name"
  } \

else \
  {
  if { $_ClassMgr_(debug) } \
    {
    puts stdout \
        "\[dp_objectRunMethod\] found method $op in class $parent"
    }

  set result \
        [eval [concat $proc_name $object_name $arg_list]]
#       [$proc_name $object_name [concat $arg_list]]
  }

return  $result
}
# dp_objectRunMethod
```

ORNL/TM-12811

INTERNAL DISTRIBUTION

1. R. K. Browning
2. C. W. Glover
3. R. Hume
4. W. R. Lee
5. D. M. Lopez
6. R. C. Mann
7. J. P. McNeely
8. E. M. Oblow
9. S. Petrov
10. J. A. Rome
11. M. B. Shah
12. R. A. Tannert, Jr.
13. E. C. Uberbacher
14. EPMD Reports Office
15-16. Laboratory Records, ORNL-RC
17. Document Reference Section
18. Central Research Library
19. ORNL Patent Office

EXTERNAL DISTRIBUTION

20-21. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831.

20. Office of Assistant Manager for Energy Research and Development, Oak Ridge Operations, U.S. Department of Energy, P.O. Box 2008, Oak Ridge, TN 37831

21. Jim Decker, Director, Office of Energy Research, Dept. of Energy, Washington, DC 20585

22. Prof. Donald J. Dudziak, Dept. of Nuclear Engineering, 110B Burlington Engineering Labs, North Carolina State University, Raleigh, NC 27695-7909.

23. Prof. Dr. James E. Leiss, Rt. 2, Box 142C, Broadway, VA 22815