The following paper was originally presented at the
Third Annual Tcl/Tk Workshop
sponsored by Unisys, Inc. and USENIX Association
Toronto, Ontario, Canada, July 1995.

Using Tcl/Tk to Program a Full Functional Geographic Information System

George C. Moon, Alex Lee, and Stephen Lindsey

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org

Using Tcl/Tk to Program a Full Functional Geographic Information System

George C. Moon

george@system9.unisys.com

Alex Lee

alex@system9.unisys.com

Stephen Lindsey

stephen@system9.unisys.com

Abstract

Tcl/tk is used to provide programming constructs and easy to build graphical user interfaces for a full functional Geographical Information System (GIS). This paper defines the term GIS, explains why tcl/tk is used, discusses which extensions to tcl/tk are used, and describes what extensions are added to support the GIS. Examples are provided. The GIS system is part of a commercial product family internally referred to as Harbour GIS (HG).

Tcl/tk and the GIS extensions are used to build common graphical user interface widgets for GIS program developers and to build GIS specific applications for GIS end users. This paper presents the extensions using simple examples that should be easy to follow without need for specific domain knowledge. By presenting simple examples of how the extensions are used, it is hoped that others may gain insights into how one commercial vendor has approached adding significant functionality to tcl/tk.

Geographical Information System

A Geographical Information System (GIS) can be thought of as a computer system that allows relationships between and among spatially located features to be analysed and viewed. The spatial data is usually two or three dimensional. Time may be considered by some systems. Most systems have non-spatial attributes associated with the spatial data. Relational database systems are often used to store and manipulate the non-spatial data. In some systems a database management system is also used to store the spatial data.

The concept of a feature in a GIS will mean different things in different GIS systems. For the purpose of this paper a feature is the representation by a set of two or three dimensional coordinate values of a geographic entity located in physical space. Examples may be a fire hydrant as represented by a single x,y coordinate pair, a cable as represented by two or more x,y coordinate pairs and a parcel as represented by three or more x,y coordinate pairs forming a closed surface. This paper will not attempt to discuss different ways space can be modeled or even the different ways the examples can be modeled

Early GIS systems were mainly concerned with the automating of the mapping process. Little in the way of actual analysis of relationships could be done. Today there are a number of spatial operations that are expected. Some of the most common operations are:

> Buffer: the process whereby a set of geometric data is expanded a specified distance. In the example of a single coordinate pair in space the result is a surface of circular shape the specified distance around the coordinate pair.
>
> Overlap: the process whereby two sets of geometric data are intersected. The result is a set of geometry's where all or parts of the two sets of geometry's overlap or intersect in coordinate space.
>
> Contain: the process whereby two sets of geometric data are intersected. The result is a set of geometry's where the first set is wholly contained within the second set.

A simple example using these operators starts with having a proposed zoning change for a land parcel (or land property). If all parcel owners within five hundred metres of the proposed change must receive written notification of the change, then the parcel can be buffered 500 metres. The resultant area can then be overlapped with the parcels in the area. The parcels that result in the operation are then used to obtained the names and addresses for the parcels and letters are written to the property owners.

Why Tcl/Tk

Harbour GIS is a sophisticated GIS evolving from the past ten years of GIS development. When looking for the next generation application development language a number of factors pointed to tcl/tk as the best choice. The main deciding factors were:

- comprehensive interpretive language
- easy to learn
- supports object oriented constructs through extensions
- simple to understand and use Motif like graphical user interface
- extensible
- large user community.

Overview

A standard approach was taken to using tcl/tk as a parser for the GIS extensions to generate hgtcl and hgwish. The prefix hg is used to uniquely identify the GIS extensions. Hgtcl allows non-graphic GIS operations. Hgwish provides non-graphic and graphic GIS operations, supports the building of tk interfaces and supports a modified canvas widget for displaying spatial geographic data.

Public Domain Extensions

Selected public domain extensions have been included because of their usefulness for building or supporting the development of GIS applications. These public domain extensions include:

Extended Tcl (TclX) (Karl Lehenbauer, NeoSoft Inc. and Mark Diekans, Santa Cruz Operation) - the ability to access POSIX system calls and functions is useful for advanced application development.

Object Oriented Tcl (incr Tcl) (Michael J. Mclennan, AT&T Bell Laboratories - several of the graphical user interface widgets developed for GIS application programs consist of incr Tcl classes. The ability to define classes and methods provide a simple to use set of pre-defined objects for developers.

Bell Labs Toolkit (BLT) (George Howlett) - often GIS application developers need to provide simple charts and graphics to illustrate the results of their spatial analysis. The BLT extensions provides an additional means of meeting some of the need.

Distributed Programming (Tcl-DP) (Lawrence A. Rowe, Brian Smith, Steve Yan, University of California at Berkeley) - the Harbour GIS product provides a client-server model for the GIS application developer and user. The Tcl-DP extensions provide a useful extension to compliment the GIS product.

Tcl Debugger (Tcl-Debugger) (Don Libes of National Institute of Standards and Technology) - occasionally application developers have need of a debugger.

Tcl Japanization (Tcl-JP) (Software Research Associates, Inc.) - the Asian market for GIS needs 16 bit extensions. Tcl-JP provides a minimum solution at this time.

Other extensions will be added as appropriate to the support of GIS applications. Support of runtime loading of extensions is a highly desired feature for the tcl/tk language. In the meantime, extensions that require the modification of the core of tcl/tk will be supplied with Harbour GIS only after qualification within the development group for Harbour. Extensions that do not modify the core of tcl/tk can be used without certification by the Harbour development team.

GIS Extensions

Harbour supports a client/server model with a number of different GIS servers from which application code requests services. Tcl/tk was extended by adding the client functions to tcl/tk. When a client request is made from within a tcl/tk written application, a message is sent to an appropriate server.

Only the client side was embedded in tcl/tk, partially to keep the size of the tcl/tk executable small. Keeping it small allows an application to launch or start up other applications that by themselves can be stand alone or part of a larger application without users knowing that they are not dealing with one tightly coupled application.

The extensions added consists of message management, session management for handling

client/server communications, GIS specific analysis tools, graphical construction tools for adding and modifying spatial data in the GIS and a graphical widget which is a modified tk canvas widget. The extensions are made available by loading the extensions using a command, hg_dload. All or a subset of the extensions can be made available. For example **hg_dload -extension {MSG SM ATB}** loads a useful (for GIS purposes) set of commands where the names correspond to:

NAME	DESCRIPTION OF EXTENSION
MSG	Message Management
SM	Session Management
ATB	GIS Persistent Processing Functions (application tool box)
VIEWER	Simple View Engine for viewing the results from the ATB extension
GA	Graphic Application Management
DCE	Data Capture and Edit
COGO	Coordinate Geometry

These extensions will be further discussed below.

Naming Conventions

It has already been mentioned that tcl and wish were renamed hgtcl and hgwish. To try and avoid conflicts with other tcl/tk extensions a naming convention has been adopted. The extensions added to support Harbour GIS all follow the convention **hg_extensionName_functionName**. An example using the GIS function buffer is hg_pf_buffer -i input_class -o output_class -W 500 which generates a 500 metre (assuming GIS database units are metric) around the feature instances represented within the input set of features, input_class.

Message Management (MSG)

Harbour GIS uses a consistent, language independent method of handling user messages from the tcl/tk extensions. All output messages are stored in message files. Different files can be used for different languages. The level of messaging can be set by the application or the user. Thus only fatal errors can be reported or all messages including general information, warning or non-fatal errors can be reported. The system can also be used to provide language independent menu interfaces. Available commands for message management are summarized as:

COMMAND	DESCRIPTION
hg_msg_load	Load messages for the calling application from an external message file.
hg_msg_add	Add a new message.
hg_msg_get	Return the message content string associated with a given message code.
hg_msg_type	Return the type of a message previously loaded by hg_msg_load or hg_msg_add. Types can be “information”, “warning”, “error” or “fatal”.
hg_msg_last	Return the last message issued by the internal commands.
hg_msg_check	Compare the given message code with that of the last message issued by the internal commands.
hg_msg_translate	Translate existing widget labels to the local language specified in a given message file.

An example of use can be given with **hg_msg_translate**.

. . .

hgwish> hg_msg_load -file my_file

hgwish> pack [button .stop_button -text Stop]

hgwish> hg_msg_translate -window .stop_button

If the file my_file contains:

Stop:i::Arrête

then the button originally created as Stop will appear to the user as Arrête. An entire interface can be translated from the root of the widget tree with one hg_msg_translate call at the beginning of the application. Supporting multiple languages within hgtcl and hgwish thus can be done without undo application development hardship. When supporting applications used for an international market, the language independence is important. One nice feature of tk is the ability of widgets to change size without re-programming the size of the fields for the widgets. The user does not have to write the interface for German in order to support multiple languages.

Extended tcl has support for X/Open Portability Guide (XPG/3) but the message file is binary. A message file for each platform and language combination would have to be supplied. Internationalization is a local activity so having an ASCII message file whereby local field offices can provide translations is important to our efforts. PC versions of Harbour GIS are also important requiring a messaging system we know can be supported. As such, the support for XPG/3 was not used.

Session Management (SM)

As mentioned, Harbour GIS uses a client/server model. The **hg_sm_*** extensions are used to control GIS sessions. Control consists of which and how many servers to start and to which servers client messages called from hgtcl or hgwish are directed. GIS sessions can start services on a local host or remote machines. A summary of the extensions are given as:

COMMAND	DESCRIPTION
hg_sm_start	**Start a new GIS session.**
hg_sm_end	**Shut down all servers in the current GIS session.**
hg_sm_list	**List information about the current GIS session.**
hg_sm_exist	**Check if a requested server exists in the current GIS session**
hg_sm_set	**Set the given GIS session to be the current session.**
hg_sm_get	**Return the identifier for the current GIS session.**
hg_sm_input	**Install or remove an input handler that supplies data to a GIS server.**
hg_sm_connect	**Creates a connection to a server within a currently active session.**
hg_sm_disconnect	**Close a connection created by hg_sm_connect.**
hg_sm_execute	**Request a server to execute an operation.**
hg_sm_describe	**Provide information on a connection.**
hg_sm_pause	**Interrupt operations being performed by a server.**

An example of how some of the functions can be used can be given by considering two different geographic areas. Normally in a GIS only one geographic area is used at one time but the following does illustrate the useful concept of being able to attach to different sessions which in the case of Harbour GIS are in different databases accessed from the same application.

The areas for the example are close to each other but managed separately. The following would allow data from one area to be used in an analysis of another area. In this case, rivers from Toronto are buffered (expanded) by 1000 metres and any schools in Scarborough that are within 1000 metres of any of the rivers in Toronto are identified. Both the Toronto and Scarborough sessions are ended.

. . .

hgtcl> set tor_sid [hg_sm_start -d "Ontario(Toronto)"]

hgtcl> set scarb_sid [hg_sm_start -d "Ontario(Scarborough)"]

hgtcl> hg_sm_set -session $tor_sid

hgtcl> set con1 [hg_sm_connect -server atb_pf]

hgtcl> hg_sm_execute -handle $con1 atb_pf_select -o river -s "select river;"

hgtcl> hg_sm_execute -handle $con1 atb_pf_buffer -i river -o river_buf -W 1000 -n +buf:geometry

hgtcl> hg_sm_disconnect -handle $con1

hgtcl> hg_sm_set -session $scarb_sid

hgtcl> set con2 hg_sm_connect -handle atb_pf

hgtcl> hg_sm_execute -handle $con2 atb_pf_select -o schools -s "select schools;"

hgtcl> hg_sm_execute -handle $con2 atb_pf_overlap -i river_buf -i schools -o schools_inside

hgtcl> hg_sm_end

hgtcl> hg_sm_set -session $tor_sid

hgtcl> hg_sm_end

A later example will show an easier way of executing parts of the above where the connect, execute and disconnect are combined. The hg_sm connect, execute, disconnect sequence is normally used only when the application wishes to attach to a service and execute against the service multiple times before relinquishing control. For most GIS operations this provides no advantage since the operation time is much greater than any time taken to connect and disconnect to a service. However if one were to add a tracking device where real time reading of values were needed, then executing against the service multiple times without having to connect and disconnect each time is beneficial.

Building the client side only into tcl keeps the size of hgwish small allowing many independent components to be started and stopped as part of an application rather than starting a large monolithic application with everything built into the executable. The server side provides the heavy computation required of a GIS leaving hgwish to provide a very good user interface presentation and control mechanism for end users.

GIS Processing Functions (ATB)

Harbour GIS consists of full functional GIS capabilities. That is, the functionality that users and application developers need from a GIS to meet any GIS user request can be accessed from within hgtcl and hgwish with the added extensions and formed into an application quickly. There are more than one hundred different functions available. These range from as simple as calculating the area of a spatial feature to a complex set of functions that can provide functionality called dynamic segmentation (useful especially for managing highway and railway spatial data). The functions can be broken into the following categories:

COMMAND	DESCRIPTION
hg_pf_*	**Set of ATB client functions that send messages to GIS servers for GIS operations.**
hg_viewer_*	**Simple graphics display mechanism to view graphical results of GIS operations.**
hg_client	**Handles interface with a custom server.**
hg_client_create	**Used to create custom server functions.**

A variation of the above example is given to illustrate the use of the ATB and VIEWER extensions. Hgwish is used since the **atb_viewer_*** commands are graphic commands

. . .

hgwish> hg_sm_set -session $tor_sid

hgwish> hg_pf_ select -o river -s "select river;"

hgwish> hg_pf_buffer -i river -o river_buf -W 1000 -n +buf:geometry

hgwish> hg_sm_set -session $scarb_sid

hgwish> hg_pf_select -o schools -s "select schools;"

hgwish> hg_pf_overlap -i river_buf -i schools -o schools_inside

hgwish> hg_viewer_display_data_flow -i schools_inside

. . .

As alluded to in the previous section the **hg_pf_*** commands are essentially equivalent to **hg_sm_connect, hg_sm_execute** an_atb_command**, hg_sm_disconnect**.

An user can use the hg_client_create command to provide a custom interface to their own server functions. For example if the user wanted to use a custom buffer command then either:

hg_client +my_buffer -i river -o river_buf -W 1000 -n +buf:geometry or

hg_client_create -command my_buffer; my_buffer -i river -o river_buf -W 1000 -n +buf:geometry

allows a user (or third party) buffer to be used. The client commands establishes the correct client/server protocol from within hgtcl or hgwish to allow the connection. The ease of third party addition is one of the big advantages of tcl/tk and one of the main reasons for choosing the language.

Construction Functions (DCE/COGO)

A set of functions are provided to help construct and build spatial databases. The commands take the form:

COMMAND	DESCRIPTION
hg_dce_*	**Functions designed to build and edit geometries that are part of a spatial database.**
hg_cogo_*	**Specialized functions using coordinate geometry to manipulate coordinate data.**

Conceptually their use is the same as the functions in the previous section. Those familiar with CAD packages would see similarities in functionality. One of the most significant construction differences between CAD and GIS is the generation of topology from the coordinate data. Topology being the mathematical concept of connectivity means that software is used to build connected graphics from the geometries (coordinates) being edited. Thus a lake ends up being represented as a set of surfaces representing shorelines (lake edges) and islands. When graphically displayed of course, the lake looks like a lake; probably coloured blue. Topologic construction tools as well as geometry tools are callable from **hg_dce**.

Visual Display (GA)

The graphical widget is a variation of the tk canvas with the ability to display large amounts of graphical data on the canvas. There are two sides to GA. One side is a server side which knows about displaying large amounts of GIS data and the other side is the client side that knows how to sketch data either in screen space or in GIS coordinate space. Separating the visual display into client side and server side allows a lot of customization.

Client side graphics can be used to support user defined graphic cursors, limited animation, real time visual update and sketching independent of display operations at the server. The client side graphics thus provides capabilities to the GIS community not usually available in conjunction with spatial data. A simple example is the real time tracking of ground vehicles using global positioning input (a satellite positioning system that provides ground coordinates for any global positioning receiver) on a computer map display while at the same time interacting with the data in the map display doing some type of GIS analysis. The real time tracking could also trigger actions depending on the real world spatial location of the vehicle.

COMMAND	DESCRIPTION
hg_ga_start	**Start a graphics server.**
hg_ga_exit	**Shutdown the current graphics server.**
hg_ga_connect	**Attach to the given graphics server.**
hg_ga_view_*	**Functions to control what is displayed on the graphics server.**
hg_ga_callback*	**Functions that allow the client side response to graphical events.**
hg_ga_db*	**Functions for determining spatial characteristics of the current GIS database.**
hg_viewto*	**Functions to convert between view and GIS coordinate systems.**
hg_wcto*	**Same as hg_viewto* except from GIS coordinates (world coordinate) to view coordinate.**
hg_wc*	**Graphic operations in GIS coordinates including operations such as create, move, scale**
hg_changeto*	**Functions to change client side graphics to GIS coordinates or visa versa.**

The following creates a graphics application view and then displays some GIS data in the view:

...

hg_ga_view_create -view .main.f3.view1 -height 600 -width 700; .main.f3.view1 config -relief ridge

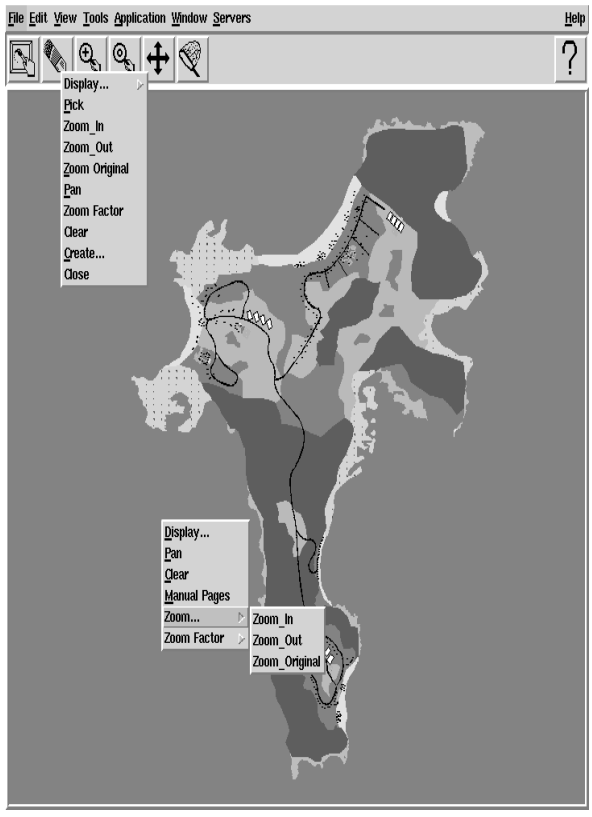
pack .main.f3 .main.f3.view1 -side bottom -fill both -expand yes

hg_ga_view_display -fn {county_line township_line borough_line major_highway minor_highway township_route borough_street private_road_street utility_line GPS_point electoral_disrtict}

Typically a GIS display within Harbour GIS will consist of ten's of thousands of graphical objects. Tests with the canvas widget standard in tk showed that performance was not adequate for GIS applications once more than ten thousand objects were displayed. Fast picking from a large set of objects in a canvas display list is also needed. The display list of the graphical widget uses a spatial index to quickly retrieve picked data from the display list rather than sequentially traversing each object in the display list. With the amount of data in a typical GIS application the difference is 'real time', less than two seconds to unacceptable, more than five seconds.

Common Widgets

A set of specific classes built with itcl and an application building guide are provided to give a consistent look and feel to the GIS applications. The following illustration shows a typical view when the widgets are used and the extensions mentioned above are used to build an application:



Common Graphical Widget

Conclusion

Tcl/tk with the extensions described above has proved itself to be an excellent environment for building comprehensive GIS applications. Application developers are able to build end user applications quickly that can be used as part of a rapid prototyping exercise or as final product.

Early concerns about performance when very large numbers of graphical objects are displayed and interacted with during a GIS session were overcome by using a client server model where the server handles the large number of graphical objects.

Availability

Harbour GIS is currently in beta and will be commercially available in Q3 of 1995. Additional information can be obtained by accessing the Unisys web page http://www.unisys.com/ or contacting one of the above authors.