The following paper was originally published in the
Proceedings of the Fourth USENIX Tcl/Tk Workshop
Monterey, CA, July 10-13, 1996

A Tk Netscape Plugin

Jacob Levy
Sun Microsystems Labs

A Tk Netscape Plugin

Jacob Levy

Sun Microsystems Labs

2550 Garcia Ave., Mountain View, CA 94043

Abstract: I have built a plugin module for Tk, for Netscape Navigator. This module delivers Tk applications as elements of Web pages and it makes it possible to create web based applications that have a richer GUI than HTML.Tcl also makes building such applications easier due to its high level scripting nature, and Safe Tcl helps construct web applications that can safely perform interesting tasks such as communicating with local or remote resources. The demonstration will show how the plugin module works and some of its capabilities and limitations.

1.0 Introduction.

I believe that Tcl and Tk [1] are useful for scripting the Internet. Tcl provides an expressive set of features that are portably implemented on a wide variety of platforms. Tcl has recently been enhanced with network communication capabilities, significantly easing the task of creating network aware applications. Tk is a mature and portable GUI kit built on top of Tcl that is available for all of today's popular desktop systems.

To make Tcl a ubiquitous scripting language for the Internet, a delivery vehicle for Tcl applications over the network is needed. Netscape Navigator [2], with its ability to host plugin modules [3], offers such a delivery mechanism. A plugin module is an extension, supplied by a third party, to allow a browser to display information for which the browser does not have built in support. A plugin module displays its data in a window given to it by the browser. The plugin module can interact with the user through this window.

Plugin modules are not complete programs; they are activated inside the browser's address space and windowing hierarchy. While there is no standard in this area yet, generally the browser limits interaction between a plugin module and other displayed elements of a page. Plugin modules are written in C/C++ and have full access to the hosting system.

Embedding Tk programs in Web pages provides unique advantages:

- Tk allows the construction of much richer and reactive GUIs than possible with HTML [4]. For example, in HTML there is no way to display structured graphics, as is possible with Tk's canvas widget.
- Tcl, as a high level scripting language, facilitates the development of interactive web based applications. It is much easier to write a Tk program to perform a web based function than it is to write a complete plugin that delivers the same functionality.
- Safe Tcl [5] provides a solid base for flexible security policies that allow applications to perform functions that would otherwise be impossible in untrusted scripts. For example, Safe Tcl can provide security policies that allow varying degrees of access to local and remote resources. The Tcl plugin can make use of Navigator's authentication facilities, when Navigator will be extended with the required APIs.

- Tcl, due to its late binding, allows incremental delivery of applications over the Web. An application can be partitioned into small chunks of functionality that are loaded on demand, as the user requests access to each part. Additionally, Netscape Navigator's cache policy can be configured to ensure that the user always sees the latest version; this eliminates the need to maintain several versions of an application.

Two goals were paramount in designing the Tk plugin module:

- Tk applications should operate identically whether they are embedded inside a Web page or not, and whether they are delivered over a network or are locally available. This immediately makes all existing Tk applications deliverable via the plugin. The "." window should behave the same irrespective of whether the application runs inside the plugin module. Also, the full power of Tk, including event driven programming, should be available in the plugin module.
- Tk applications should execute in a safe environment, so that they cannot harm the user visiting a page. On the other hand, this environment should not be too restrictive, otherwise many interesting applications are ruled out.

The paper describes the implementation issues I faced when constructing the plugin module. It then describes some possible uses of the Tk plugin module, and discusses what the plugin can and cannot do. I then describe the current status of the development of the plugin, and briefly hint at possible future projects.

2.0 Implementation Issues.

To implement the Tk plugin module, I encountered three main issues:

- Embedding Tk windows inside another application's window hierarchy.
- Preserving Tk's event model as well as Netscape's event handling.
- Providing a safe yet functionally rich execution environment for untrusted scripts.

Netscape Navigator provides a native window in which an instance of a plugin can display itself. The window belongs to Netscape and can be modified at will by Navigator. Navigator provides notifications to the plugin to inform it of important events that affect the window such as resizing and destruction of the window. However, Tk expects that all windows that it uses to display content are created by it and are completely owned by it. Tk also believes that ".", the application's main window, is really a toplevel window and not the child of some other window; Tk wants to set the size and position of toplevel windows and interact directly with a window manager. To address this, Tk has been enhanced so that it is able to create a main window that is a child of another window. This also provides Tk with the illusion that it is interacting with a window manager to manage the position and size of its main window.

A related issue is event handling. Tcl has its own event loop that provides file and timer based events, in addition to UI events. Because I do not have access to the event loop in Navigator, I decided to provide a separate event loop for the plugin module, using the most appropriate mechanism for each platform. This also avoids extensive modifications to Tk. In Win32 [6] and Solaris 2.x, I use a timer that causes the Tcl event loop to be entered periodically. This allows the plugin module to react to events in a timely fashion without interacting with Navigator's event loop. I considered running Tk inside its own process and communicating with Navigator using sockets; however it is difficult to manage a window belonging to one process inside another, and the plugin would not have the same level of access to Navigator's API as is possible when it executes within the same address space.

Incoming scripts are not trusted and should be executed in an environment that prevents them from damaging the host system by e.g deleting files or stealing private information. Safe Tcl provides an environment in which an untrusted script cannot do such damage. Each script is executed inside its own interpreter which is made safe by removing "dangerous" commands. Safe Tcl also provides a mechanism for implementing security policies that allow untrusted scripts to safely perform many interesting tasks: interpreters are arranged in a master slave hierarchy; a slave interpreter can be extended by its master with safe access to unsafe functionality in the master. For example, a policy can enforce that a script has read only access to local files but cannot communicate with off-site processes.

Tk prevents a script from damaging the host site or stealing its private information; however, currently it allows scripts to mount denial of service attacks, e.g. by globally grabbing the mouse and never releasing the grab. A subset of Tk that prevents such attacks should defined and be made available in safe interpreters. This is an area for further research.

3.0 Uses for a Tk Plugin Module.

Three main uses are envisioned for the Tk plugin module:

- Provide richer GUI elements in a page than is possible with HTML alone. Tk can be used, e.g., to decorate pages with interactive structured graphics using canvas widgets that provide bindings for events to interact with the user in interesting ways.
- Implement interactive forms that provide client side validation and assistance; for example, a tax form application can perform internal consistency checks and suggest allowable deductions that the user did not take. An application can be structured as a series of linked forms; for example, if the user selects a home mortgage interest deduction in the main form, she can be presented with an itemized deduction schedule to fill out in another form. The script can also automatically fill out parts of the form for the user if it is given access to local data such as the user's email address.
- Safely host applications that require communication with a variety of backends such as databases, or access to local resources. Tcl provides its own communication mechanisms such as sockets that can be safely used by untrusted applications, and it allows a range of interesting security policies to govern access to local and remote resources. For example, a script can connect to a remote multi-user forum and provide a way for the user to interact through its window with other users in a shared application. As another example, a user can safely give a script access to her private financial data using a security policy that prevents the script from communicating remotely.

4.0 Limitations of the Plugin.

There are some limitations to what a plugin module can do; these mostly stem from the very narrow interface provided by the hosting browser. For example, a plugin module currently does not have access to other elements of the displayed page such as embedded forms. Some limited access to JavaScript [7] is possible, but it is currently not possible to control Java [8] applets or other plugins from inside a plugin. While access to forms written in HTML is desirable, Tk is rich enough to allow a form to be constructed entirely with Tk UI elements, but this does require programming.

5.0 Status and Future Plans.

I intend to release the plugin for Windows NT, Windows 95 and Solaris 2.x at the workshop, for experimentation by the Tcl community. A MacOS version of the plugin will be distributed later. The plugin is in active development at the time of writing and its capabilities are improving rapidly. It is currently able to display Tk applets inside windows embedded in Navigator's window hierarchy, and it is able to load scripts from remote or local URLs.

ActiveX [9, 10] is being promoted by Microsoft as an alternative to plugins. I have created a version of the plugin packaged as an OLE control (OCX) [11], and I will also release this package at the workshop. My intent is to continue developing the OCX to make it usable as a component in the upcoming Windows 95 desktop which will be an OCX container.

Plugin modules are currently not usable in a variety of browsers, because there is no standard for the interface between the browser and the plugin module. I expect that increasing interest in plugin modules will spur standardization efforts in this area, perhaps under the auspices of the W3 consortium. Therefore I will target my efforts to provide a Tk plugin module at those plugin enabled browsers that capture a significant market share. I expect that more browsers will shortly support plugin modules. Microsoft Internet Explorer [12] , Spyglass Mosaic [13] and Oracle's PowerBrowser [14] also support plugins; Oracle PowerBrowser uses Netscape's plugin API.

I am planning to add the following functionality to the plugin, to enable it to host more powerful applets:

- The ability to send data to Navigator for display. This will allow the plugin to behave as a filter or HTML generator.
- The ability to initiate requests for new remote or local URLs to be fetched and presented to the applets as Tcl channels. This will enable the plugin to host applets that are composed of several parts, and not require that they all be downloaded before any display can occur. The main problem in realizing this is likely to be how to allow this while still preserving the privacy of information in the user's environment.
- The plugin should allow each applet to manage more than a single toplevel window within the page being displayed by Navigator. This will make possible the construction of powerful form-like interfaces with interactive behavior, e.g. for data verification and consistency checking. Of course, because the full power of Tk is available in each of these toplevel windows, the interface is not limited to only form elements.

6.0 Acknowledgments.

Early versions of the plugin module were developed by Jeff Hobbs. Colin Stevens provided Win32 expertise and helped develop the plugin module on Windows NT and Windows 95. John Ousterhout made the enhancements to Tk that allow it to be embedded in other window hierarchies.

7.0 References.

1. "Tcl and the Tk Toolkit", John K. Ousterhout, Addison-Wesley, (1994).
2. Netscape Navigator 3.0, URL: http://home.netscape.com/.
3. Netscape Navigator SDK, URL: http://home.netscape.com/
4. W3 Consortium Web Site, URL: http://www.w3.org/pub/WWW/.

5. “A Safe Tcl Toolkit For Electronic Meeting Places”, Jacob Levy, in Proc. 1st USENIX Workshop on Electronic Commerce, pp. 133-137, NY, NY (1995).

6. “Win32 Programmer’s Reference, Vol. 1-5”, Microsoft Press, (1994).

7. JavasScript URL: http://www.netscape.com/

8. Java URL: http://www.javasoft.com.

9. ActiveX URL: http://www.microsoft.com/intdev/sdk/.

10. “Inside OLE” Kraig Brockschmidt, Microsoft Press (1995).

11. “OLE Controls Inside Out”, Adam Denning, Microsoft Press (1995).

12. Microsoft Internet Explorer, URL: http://www.microsoft.com/windows/ie/msie.htm.

13. SpyGlass Mosaic, URL: http://www.spyglass.com/.

14. Oracle PowerBrowser, URL: http://www.oracle.com/products/websystem/powerbrowser/html/index.html.