NIST Special Publication 500-253
May 2003

GLUT/Tk:[*] OpenGL[†] with Tcl/Tk

Joseph C. Kończal <joe.konczal@nist.gov>
Information Technology Laboratory
National Institute of Standards and Technology

Abstract

GLUT/Tk provides a mechanism for control of 3D graphics applications by a full-featured GUI system. It connects control windows written in Tk to OpenGL graphics windows managed by GLUT, with minimal disruption of each package's default program structure and small changes to the GLUT source code. GLUT/Tk is available for both Unix/X and Windows platforms, and our experience indicates that using GLUT/Tk, instead of platform specific interprocess communication (IPC) methods, makes programs much easier to port.

1 Introduction

The Visualization and Usability Group has produced several graphical application prototypes with separate OpenGL[1] and Tcl/Tk[2] processes running on Unix, using X Window System[3] messages for ad hoc interprocess communication, e.g., NIRVE[4] and VisVIP[5]. These applications support spaceball[6] input for control of the graphical portion, and have complicated user interfaces written in Tcl/Tk. They are also very difficult to port to Windows, which is important in order to reach a larger number of potential users. GLUT/Tk[7] allows us to continue to develop programs in about the same way, using our robust and powerful Unix workstations, but makes them much easier to port to Windows.

GLUT/Tk supports the production of multi-platform high-performance graphical programs with sophisticated user interfaces by combining the strengths of OpenGL and Tcl/Tk. OpenGL provides high performance device independent 2D and 3D graphics. Tcl/Tk is excellent for creating sophisticated graphical user interfaces. GLUT[8] adds support for input devices, fonts, and window management to OpenGL, but GLUT and Tcl/Tk are difficult to combine within one process, since they both were designed to work within their own complete application frameworks. GLUT/Tk provides a lightweight IPC mechanism to allow native GLUT and Tcl/Tk processes to communicate and work together with minimal changes to their source code or programming style. To the GLUT program, after the initial configuration dialog, the Tcl/Tk process looks like just another input device, albeit more complicated than a typical mouse or spaceball.

2 Alternatives

Before writing GLUT/Tk we looked for existing tools that would meet our requirements for high performance graphics, complex GUIs, spaceball support,

[*] Contribution of the National Institute of Standards and Technology. Not subject to copyright.

[†] Certain products and trade names are identified in the paper, and some were used to create GLUT/Tk. This does not imply any kind of recommendation or endorsement by the National Institute of Standards and Technology.

and low cost, on both Unix/X and Windows platforms. We also wanted to continue using our skills in OpenGL, and Tcl/Tk programming, and our existing C code libraries as much as possible. We did not find anything that fulfilled all our requirements.

Complete graphical development environments like Java3D[9] and the Visualization Toolkit[10] provide many desirable and powerful features on multiple platforms. However, since they are based on construction of scene graphs, instead of direct graphics commands, they require a complete change of paradigm and the learning of complex new APIs. Concerns about the performance of these large complex systems made it undesirable to invest the effort required to start using them.

Tcl/Tk extensions supporting OpenGL, like Togl[11] and TkSM[12], appeared to be similar to what we needed. However, they seem to be designed for writing Tcl/Tk scripts that do some graphics, but we wanted interactive graphics programs with Tcl/Tk control panels. We feared that the Tcl/Tk application framework might not provide adequate interactive graphics performance.

3 Implementation

A GLUT/Tk program consists of an OpenGL graphics process written in C using the GLUT/Tk extended version of GLUT, and a control process that it launches, which is a Tcl/Tk script using the GLUT/Tk dynamic module. It is possible to create one Tcl/Tk control window for each GLUT graphics window. The graphics process passes its window identifier as the first parameter when it starts the control process. When the control process loads the GLUT/Tk module, it removes the first argument and uses it to send its own window identifier back to the graphics process. (The application programmer never actually handles the script argument containing the window id, since it is inserted and removed automatically.) Both processes now know each other's window identifiers, which they use to exchange window system events.

3.1 Changes to GLUT

The extended version of GLUT is created by applying a 2500 line patch[13] to the GLUT sources. It adds four new functions, and also allows GLUT to compile with the Cygwin tools[14] on Windows. **glutCreateWindow4Tcl()** is an enhanced version of **glutCreateWindow()**, which, in addition to creating a new graphics window, also launches a specified Tcl/Tk script that communicates with it. **glutTclFunc()** registers a callback to handle input events received from the Tcl/Tk process. **glutPressTkButton()** sends an event to the Tcl/Tk process simulating the effect of a user pressing a Tk button. **glutButton2Tk()** is a lower level function providing more control over the details of events sent to the Tcl/Tk process. **glutPressTkButton()** actually invokes **glutButton2Tk()** using some default values for the extra parameters.

3.2 Tcl/Tk module

The GLUT/Tk dynamic Tcl module consists of about 500 lines of C code, with a TEA[15] compliant build system, and a separate makefile for Cygwin GCC on Windows. It does some initialization, consuming the first script argument which contains the window identifier of the associated GLUT window, and provides three new Tcl functions. **send_to_glut** sends a string or integer message to the GLUT window. **enable_glut_to_tk** sets up a list of Tk buttons that can be virtually clicked by the GLUT process. This could be used to implement keyboard shortcuts in the graphics window for control functions. **tcl_target** is a lower level command called by **enable_glut_to_tk**, which adds a single entry to the list of Tk windows that the GLUT process can activate.

4 Development

John Cugini, also from the Visualization and Usability Group at NIST, wrote the original GLUT/Tk on IRIX. It used a slightly modified GLUT 3.6 library, and a custom Tcl 7.6/Tk 4.2 interpreter with the GLUT/Tk communication functions added. Joseph Konczal modified GLUT/Tk to work on other Unix platforms with GLUT 3.7 and Tcl/Tk 8.2 or higher, and ported it to Windows. The Tcl/Tk support was moved out of the interpreter into a dynamic module linked with the stubs library, which should also work with newer versions of Tcl/Tk. The spaceball support in GLUT was updated to use the Xdriver program[16] on Unix, because the X Input Extension used on IRIX was not available for other platforms. Spaceball support for GLUT on Windows was added using the 3DxWare driver[17].

Porting GLUT/Tk to Windows also required major changes to the IPC code, since the Unix/X version uses X ClientMessage events. The Windows version uses four kinds of Windows messages, two built-in types: **WM_COPYDATA** and **BN_CLICKED**, and two application defined types: **GLUTTK_WINDOW_ID**, and **GLUTTK_DATA**. **GLUTTK_WINDOW_ID** is used to send window identifiers from Tcl/Tk to GLUT. **WM_COPYDATA** is used to send larger messages, like strings. **BN_CLICKED** is used to send virtual button clicks from GLUT to Tcl/Tk. **GLUTTK_DATA** is used internally by GLUT to queue the processing of the data received in **WM_COPYDATA** messages, which must be deferred to avoid deadlock.

5 Open Issues

5.1 Maintenance

No major changes were required, besides the repackaging of the Tcl/Tk code into a module, to move GLUT/Tk to newer versions of GLUT and Tcl/Tk, and the Windows port works on versions 95 through XP, so we expect it won't be difficult to adapt GLUT/Tk to future releases of these components. The source code[7] is available, so somebody else could do it if necessary.

5.2 Application Porting

The GLUT/Tk demonstration program Platonics, written on IRIX with the X Window System, ported to Linux, Solaris, and Windows with minimal effort. However, our first real GLUT/Tk application, FLUDViz[18], took longer to port to Windows because of other operating system dependencies that are not addressed by GLUT/Tk. We have begun compiling a library of generic functions to address these other porting problems.

5.3 Application Termination

Coordinating the termination of both processes is tricky. One might make each window notify the other if the user initiates termination of the program. This could lead to looping instead of actual termination, as each process repeatedly tells the other one to quit. We have written our programs under the assumption that the user will quit by pressing the "Quit" button in the Tcl/Tk control window, which sends a message to the GLUT window telling it to quit. If the user terminates the graphics process directly, the control window will remain until it is also explicitly terminated.

5.4 Security

GLUT/Tk passes window system messages between different processes on the same desktop. It would not work if a particularly strict security configuration were to block these messages. We have not investigated whether the X Window System or Windows 2000 or XP could be configured in such a way that GLUT/Tk would not work.

References

[1] OpenGL—High Performance 2D/3D Graphics. http://www.opengl.org.

[2] Tcl SourceForge Project. http://tcl.sourceforge.net.

[3] The X11 Window System. http://www.x.org.

[4] NIST. NIRVE Web Site. http://www.itl.nist.gov/iaui/vug/cugini/uicd/nirve-home.html.

[5] NIST. VisVIP Web Site. http://www.itl.nist.gov/iaui/vug/cugini/webmet/visvip/vv-home.html.

[6] 3Dconnexion. Product Overview—Spaceball 4000. http://www.3dconnexion.com/products/4000.

[7] NIST. GLUT/Tk Web Site. http://www.nist.gov/gluttk.

[8] GLUT—OpenGL Utility Toolkit. http://www.opengl.org/developers/documentation/glut.html.

[9] Sun Microsystems. Java 3D. http://java.sun.com/products/java-media/3D.

[10] The Visualization Toolkit. http://www.vtk.org.

[11] Brian Paul and Ben Bederson. A Tk OpenGL Widget. http://togl.sourceforge.net.

[12] SAL—Computer Graphics, Image & Signals—Misc.—TkSM. http://ceu.fi.udc.es/SAL/E/0/TKSM.html.

[13] Patch. http://www.gnu.org/software/patch/patch.html.

[14] Cygwin—a Unix environment for Windows. http://cygwin.com.

[15] Brent Welsh and Michael Thomas. The Tcl Extension Architecture. In *Proceedings of the 7th USENIX Tcl/Tk Conference*, February 2000. http://www.usenix.org/events/tcl2k/full_papers/welchextension/welchextension.pdf.

[16] 3Dconnexion. Xdriver—Spaceball Driver for Unix. http://www.3dconnexion.com/software/browse.html.

[17] 3Dconnexion. 3DxWare—Spaceball Driver for MS Windows. http://www.3dconnexion.com/software/win/driver.

[18] NIST. FLUDViz Web Site. http://zing.ncsl.nist.gov/WebTools/FLUDViz/overview.html.