3wish: Distributed [incr Tcl] Extensions for Physical-World Interfaces

Brygg Ullmer
MIT Media Lab
Cambridge, MA

3wish: Distributed [incr Tcl] Extensions for Physical-World Interfaces

Brygg Ullmer
MIT Media Lab
20 Ames St., E15-445
Cambridge, MA 02139 USA
ullmer@media.mit.edu / http://www.media.mit.edu/~ullmer

Abstract

The creation of physical-world interfaces seamlessly integrated with the physical environment poses new implementation and interface challenges for the Tcl language. 3wish is a suite of [incr Tcl] class libraries and C/C++ extensions which supports user interfaces integrating distributed physical sensors, displays, and 3D graphics. The poster presents an overview of 3wish, its application to physical-world interfaces, and its implementation of distributed sensors, displays, object proxies, and platform-independent 3D graphics.

1. Introduction

Traditional 2D graphical user interfaces have been well-supported by the Tcl/Tk language. A new kind of user interface, "tangible user interfaces," uses physical objects and surfaces as physical interfaces to digital information. [1] These interfaces require sensing and augmentation by multiple physical-world sensors and displays, often distributed across several computers.

Tcl and the [incr Tcl] object/namespace extensions are well-suited for this style of user interface. Tcl's platform-independent, high-level operation, coupled with [incr Tcl]'s OOP class and namespace mechanisms, jointly support the rapid integration of access methods for distributed platform-specific sensors and displays. At the same time, new features are required to cleanly support physical-world interaction and distributed operation. 3wish is a suite of [incr Tcl] class libraries and C/C++ extensions that addresses these needs.

2. 3wish

3wish is designed to support user interfaces driven by physical objects. These physical objects must sense and process their physical state (position, orientation, etc.), coordinate among each other, and display various outputs (graphics, sound, etc.) to the user. Some of these sensors and displays are physically attached to interface objects; examples include position-trackers and flat-panel displays. Other sensing and display is performed on behalf of passive physical objects. For example, computer vision is used to track objects, while transparent objects are illuminated by back-projected displays. A sample interface is pictured in Figure 1, and discussed in detail in [3].

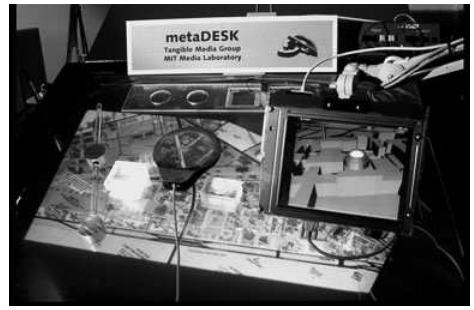

Figure 1: Image of metaDESK-based 3wish application

3wish hides these implementational details from the user interface. 3wish provides an [incr Tcl]-based "proxy" class for each physical object. [2, 3] These classes provide methods for querying object state and activating supported displays. However, underlying sensor/display coordination is hidden by the API. This abstraction supports a high degree of flexibility in constructing complex interfaces without becoming immersed in implementational minutia.

3. Distributed sensors and displays

At its lowest level, 3wish communicates with drivers for physical-world sensors and displays, currently operating on both SGI and Wintel platforms. Tcl 7.6's `load` command is used to load drivers for devices with C and C++ API's, where Tcl-DP 4.0 is used to communicate with serial devices.

Sensor APIs are implemented using an [incr Tcl] sensor class which provides caching and optional interpolation for sensor values. The same class is shared on both client and server machines; peer server and client classes work to synchronize sensor client and server states. Client sensor value requests are non-blocking, using the most recent cached/interpolated value available.

New sensor implementations usually require no additional networking code, as client/server code is abstracted from individual sensor fields and access methods. Similar display support is underway, though is currently at an earlier stage of development.

4. Object proxies and namespaces

Applications do not query sensor and display clients directly. Instead, [incr Tcl]-based "proxies" are provided as API's for each physical object. The resulting object-centric access methods are independent of the underlying sensing/display technologies employed. An illustration of this "proxy-distributed" or "proxdist" architecture is illustrated in Figure 2, and discussed further in [2] and [3]. Distributed namespaces are used to provide clean distributed coordination and avoid host/port/protocol dependencies.

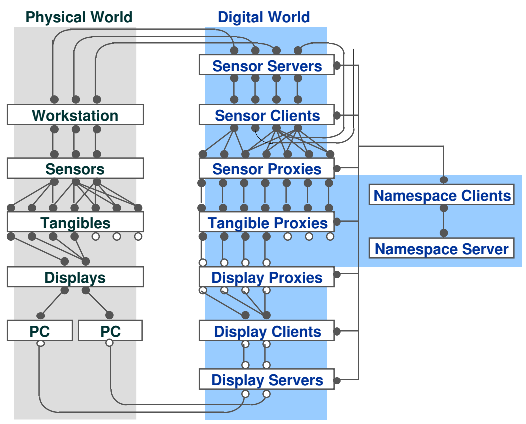

Figure 2: Diagram of 3wish sensor/display architecture

5. 3D Graphics

A first iteration of 3wish developed a set of platform-independent 3D graphics extensions, inspired by the 2D Tk toolkit. 3wish's 3D graphics core draws on the Open Inventor (OIV) toolkit, using TGS's OIV port on non-SGI platforms. 3wish provides commands for asserting named OIV/VRML geometries, binding events involving these geometries to Tcl callbacks, etc. 3wish also supports registering scene graphs across multiple 2D and 3D displays, binding graphical geometries to physical objects, as well as providing other distributed graphics capabilities.

References

[1] Ishii, H., and Ullmer, B. "Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms." In *Proc. Of CHI'97*, pp. 234-241.

[2] Ullmer, B. *Behavioral Realizations of Proxy-Distributed Computation.* http://www.media.mit.edu/~ullmer/-courses/agents/paper1.html March 1996.

[3] Ullmer, B., and Ishii, H. "The metaDESK: Models and Prototypes for Tangible User Interfaces." Submitted to UIST'97