TkDND: a cross-platform drag'n'drop package

Georgios Petasis

**Software and Knowledge Engineering Laboratory,
Institute of Informatics and Telecommunications,
National Centre for Scientific Research "Demokritos",
Athens, Greece
petasis@iit.demokritos.gr**

Abstract

This paper is about TkDND, a Tcl/Tk extension that aims to add cross-application drag and drop support to Tk, for popular operating systems, such as Microsoft Windows, Apple OS X and GNU/Linux. Being in its second rewrite, TkDND 2.x has a stable implementation for Windows and OS X, while support for Linux and the XDND protocol is still under development.

1 Introduction

This paper is about TkDND, a cross-platform and multi-protocol Tcl/Tk extension, which aims in extending Tk with cross-application drag and drop capabilities. The term "drag and drop" refers to the action of clicking on a virtual object, and either dragging it to a different location, or onto a different virtual object. Being an important element of modern UI development, several attempts have been made in order to extend Tk with drag and drop (from now on denoted as "DnD") capabilities. Several projects are listed in the relevant Tcl Wiki page [1], which can be categorised into two main categories: approaches that target DnD within the same application, and approaches that target DnD among different applications. Implementing DnD within the context of a single application is a relatively easy task: a simple communication mechanism among virtual objects needs to be devised, which can be implemented even in Tcl. Thus, several intra-application DnD approaches are available, especially within packages that implement composite widgets or "megawidgets", such as BWidget [2]. On the other hand, supporting inter-application DnD is a more challenging task, but nevertheless a more interesting and useful task. Initial approaches in this category devised several inter-application communication schemes, which targeted other Tk applications, such as BLT [3] with its communication scheme based on Tk's "send" command. More recent approaches tried to solve an even more complex task, intra-application DnD without any restriction on the type of applications (i.e. even for non-Tk applications), by exploiting standardised DnD communication protocols available for the various operating systems. These approaches targeted mainly popular operating systems, such as Microsoft Windows. Olednd [4], developed by Gordon Chaffee, is amongst the first DnD approaches for Tk that can be characterised as complete: Olednd supported DnD with any application running on the Microsoft Windows operating system, by exploiting the platform's native DnD protocol, OLE DnD. Olednd has been the main inspiration of the extension described in this paper, TkDND, whose main motivation was to enhance the functionality offered by Olednd under Windows, and provide the same functionality under other popular operating systems, such as Gnu/Linux and Mac OS X.

TkDND is not a recent extension, as its development started in 2000. With an API for applications inspired by Olednd but targeting multiple operating systems, TkDND initially

concentrated on implementing DnD for the Windows operating system using OLE DnD, and having the Olednd implementation as a guide. Once the Windows implementation was in place, implementation shifted in supporting DnD under the Linux operating system. At that time, two communication protocols were available under Linux, all of which bind with a specific widget toolkit. The older of these protocol was the Motif [5] DnD protocol, supported by Motif and LessTif [6] applications, and partially by the GTK [7] and GNOME [8] applications. The second protocol was an open standard, in early stage of development, known as XDND [9]. XDND was mainly supported by Qt [10] and KDE [11] applications.

TkDND tried to compromise the two protocols: the 1.x series of TkDND tried to implement support for the XDND for both dragging and dropping actions, and support for dropping actions under the Motif DnD protocol. All implementation for both was done from scratch, as there were no general-purpose libraries implementing these protocols beside the widget toolkits implementing them. The end result was not satisfactory, as it exhibited several problems:

- The implementation was not stable enough. The primary author of TkDND tried to tackle the problem of implementing two protocols from scratch, without having adequate knowledge of Xlib [12], led a prolonged development period and a result of low quality.
- The XDND protocol was not mature enough and was under constant revisions. TkDND was forced to support several versions of the protocol, constituting XDND support a moving target.
- Several compatibility problems arose in practice. Despite both GTK and Motif applications sharing the same protocol (Motif), it was not uncommon drops from Motif applications or Firefox [13] to always succeed, while dropping from Gnome applications to exhibit random behaviour. The situation regarding XDND was no better: not only the protocol was changing, but also support from the applications was changing with each Qt release.

In addition to all these, TkDND 1.x was an ambitious attempt, targeting support for as much types as possible, including transferring of plain text, Unicode text, files, links, images, etc. All these problems led to the decision of abandoning the 1.x series of TkDND, and start a new, novel implementation of TkDND, the 2.x series, in 2006. TkDND brought yet another API for applications, although TkDND 1.x applications are still supported through an emulation layer written in Tcl. The main implementation guideline followed by TkDND 2.x is to re-implement as few elements as possible in C/C++, and implement as much as possible in Tcl. The current public release of TkDND is 2.2, which offers support for DnD under Windows and OS X in transferring text and files, but drops Linux support. For a while, TkDND 2.x offered support for the dropping operation under the XDND protocol, but this support has ceased working with recent versions of the Qt library. Regarding this issue, a TIP [14] has been proposed to enhance the “selection” Tk command, which hopefully will solve this incompatibility. In addition, TkDND 2.x introduces support for cross-platform types, by adding an abstraction layer over them: applications can use types such as DND_Text and DND_Files, with TkDND performing the necessary mappings to types depended on the operating system. Finally, TkDND would have not been possible without the work and ideas of several contributors: Gordon Chaffee, for providing the inspiration and OLE DnD prototype, Laurent Riesterer for helping in the implementation of various parts of TkDND 1.x, and Kevin Walzer, who with Daniel A. Steffen, was the motivation power behind the Mac OS X implementation, by driving the development, contributing a large part of the OS X implementation and testing TkDND under OS X.

The rest of the paper is organised as follows: chapter two briefly describes the motivation behind the implementation of TkDND 2.x, along with an overview of how TkDND can be used by Tcl/Tk applications. Chapter three provides some details about the protocols employed under the supported operating systems, while chapter four concludes this document.

2 Design and usage

One of the main aims of TkDND 2.x re-write was simplicity, through the implementation of a large part in Tcl/Tk, instead of C, which dominated the implementation of TkDND 1.x. Doing so, eases the development overhead, but also allows the easy adaptation of TkDND to specific needs, even by the end users. For example, TkDND 1.x offered a more complex API, where the various event callbacks had to be specified/configured through the TkDND API, whereas in TkDND 2.x callbacks are implemented through Tk virtual events: not only it is easier for users, as they are already familiar with Tk events, but it is also easier for TkDND maintainers, as the management/destruction/association with Tk widgets is done by Tk.

2.1 Using TkDND

Using TkDND is quite easy. Drag and Drop operations can be separated in two categories:

- Accepting a drop (either from the same or from another application). Widgets registered to accept drops (data) are named as “Drop Targets” in TkDND terminology.
- Initiating a drag operation, allowing the transfer of data from a Tk widget to another widget (either to the same or to another application). Widgets registered to initiate drag operations are named as “Drag Sources” in TkDND terminology.

2.1.1 Drop targets

A drop target can be registered as:

```
tkdnd::drop_target register window ?type-list?
```

where type-list can be a list of cross-platform types (such as DND_Text or DND_Files), platform dependent types (such as CF_TEXT – although usage of platform dependent types is discouraged), or the type “*”, interpreted by TkDND as “any type”.

When a drop operation occurs over a Tk widget registered as a drop target, TkDND will deliver the following virtual events:

- <<DropEnter>>: This event is triggered when the mouse enters the window during a drop action. The purpose of this event is to change the visual state of the window, so as to notify the user whether the drop will be accepted or not. The binding script is expected to return a single value that will define the drop action. This returned action can be one of **copy**, **move**, **link**, **ask**, **private**, and **refuse_drop**. This event is not mandatory, but if it is defined, it has to return an action. In case an action is not returned, the drop is refused for this window.
- <<DropPosition>>: This event is triggered when the mouse moves inside the window during a drop action. The purpose of this event is to let window decide if it will accept the drop and the action of the drop, if a drop is going to happen at *the specific mouse coordinates*. Thus, the script binding for such an event can get the mouse coordinates and the pressed modifier buttons (such as ctrl, shift or alt), and is expected to return the drop action, which again must be one of **copy**, **move**, **link**, **ask**, **private**, and **refuse_drop**. This event is not mandatory, but if it is defined, it has

to return an action. In case an action is not returned, the drop is refused for this window.

- <<DropLeave>>: This event is triggered when the mouse leaves outside the area covered by the window, without a drop happening. The binding of such an event is expected to restore the visual state of the window to normal (i.e. the visual state the window was in before the <<DropEnter>> event was triggered). The binding for such an event is not expected to return a value.
- <<Drop>>: This event is triggered by a drop action, and it is expected to handle the dropped data and reset the visual state of the window. The binding script is expected to return a value, which will be the action that has been performed to the data, and must be one of **copy**, **move**, **link**, **ask**, **private**, and **refuse_drop**. This event is not mandatory, but if it is defined, it has to return an action. In case an action is not returned, the drop is refused for this window.
- <<Drop:***type***>>:This event is a specialisation of the generic <<Drop>> event, augmented with a type. If such a binding exists and the drop type matches *type*, this event binding will be executed, instead of the generic <<Drop>> event binding. These events allow for easy specialisation of drop bindings, according to the type of the dropped data. *type* can be either a platform independent or a platform specific type. The binding script of such an event is expected to return a value, which will be the action that has been performed to the data, and must be one of **copy**, **move**, **link**, **ask**, **private**, and **refuse_drop**. This event is not mandatory, but if it is defined, it has to return an action. In case an action is not returned, the drop is refused for this window.

2.1.2 Drag sources

A drag source can be registered as:

```
tkdnd::drag_source register window ?type-list? ?mouse-button?
```

where:

- type-list can be a list of cross-platform types (such as DND_Text or DND_Files), platform dependent types (such as CF_TEXT – although usage of platform dependent types is discouraged), or the type "*", interpreted by TkDND as "any type".
- mouse-button is the mouse button (i.e. "1", "2",. "3", etc.) that will be used in the binding for initiating a drag operation. If not specified, it defaults to "1", i.e. the left mouse button.

Once a widget is registered as a drag source, TkDND will add the proper bindings to initiate a drag operation. During the drag operation TkDND will not deliver any virtual events to the widget. The only event that will be delivered relates only to retrieving the data:

- <<DragInitCmd>>: This event is triggered when a drag action is about to start. This is a mandatory event (whose absence will cancel the dragging action), and is responsible for providing three things: the list of actions and format types supported by the drag source, and of course the data to be dropped. Thus, the binding script for such an event must return a list of three elements: the drop actions supported by the drag source (which can be any of **copy**, **move**, **link**, **ask**, and **private**), the format type list that the data can be dropped as (which can be any platform independent or platform specific type), and finally the data. A simple example of such a binding, is:

```
bind .drag_source <<DragInitCmd>> \
```

```
{list copy DND_Text {Hellow world!}}
```

- <<DragEndCmd>>: This event is triggered when the dragging action has finished (either when the drop was successful or not). Its main purpose is to process the dropped data according to the drop action returned by the drop target. Binding for such an event is not mandatory, and the binding is not expected to return a value.

3 Protocols and Operating Systems

TkDND implements support for different platforms under different operating systems. Currently, the supported protocols are OLE DnD (Windows), XDND (Linux) and Cocoa DnD (Mac OS X).

3.1 Microsoft Windows

The drag and drop under the Windows operating system is implemented through Object Linking and Embedding (OLE) [15]. The operating system provides a convenient public API for initialising OLE support in the application, registering a widget as a drop target, and performing drag operations.

3.1.1 Initialising OLE

The initialisation of OLE version 2 is very easy, through a call to OleInitialize().

3.1.2 Registering a widget as a drop target

The registration of a Tk widget as a drop target is performed through the RegisterDragDrop() function, which requires two parameters:

- The handle (HWND) of the widget.
- An object of type IDropTarget.

The IDropTarget object is used by the operating system to communicate the required events (DragEnter, DragOver, DragLeave, and Drop), to retrieve the types supported by the drop target, and to accept the dropped data. TkDND subclasses IDropTarget, in order to interface it with Tk.

3.1.3 Performing a drag operation

The operating system provides a function for performing drags from widgets, through the function DoDragDrop(), which requires four arguments:

- A data object of type IDataObject, which contains the data that will be transferred if the drop operation is successfully completed.
- A drag source object, of type IDropSource, which controls when the drop will happen, and what cursors will be used during the drag operation.
- The list of actions supported by the drag source (like copy, move, link, etc.).
- The action decided by the drop target, if the drag operation has not been cancelled.

3.1.4 Communicating events

OLE DnD supports four types of events that can occur during a drag and drop session, and most of them involve the drop target. These events are:

- DragEnter
- DragOver
- DragLeave
- Drop
- QueryContinueDrag

- GiveFeedback

All these events are communicated to the application by calling the corresponding methods on the provided by the application objects (IDropTarget and IDropSource).

3.2 Apple Mac OS X

The implementation of TkDND for the OS X operating system is written in Objective C, and uses the DnD facilities offered by the operating system for the Cocoa framework [16]. The Cocoa DnD protocol is very similar to the XDND protocol supported by GNU/Linux, as there exists great resemblance on how the two protocols operate, how the types are named, and the communication events exchange. However, there is significant difference between the two protocols: The Cocoa DnD offers a convenient API any application can use to support DnD: applications can either accept drops or initiate dragging actions without having to re-implement the protocol from scratch (as is the case for XDND). From a programmer's point of view, Cocoa DnD can be regarded as "XDND done right". Of course, TkDND took advantage of the similarities between the two toolkits, in re-using all the support code (written in Tcl, in the library section of TkDND) initially written for XDND under the Linux operating system.

3.2.1 Initialising Cocoa DnD

No specific initialisation is required.

3.2.2 Registering a widget as a drop target

The registration of a Tk widget as a drop target is performed through the registerForDraggedTypes() method of the NSView, associated to the widget. This method requires a single parameter:

- An array of the supported types of the drop target.

3.2.3 Performing a drag operation

The operating system provides a method for performing drags from widgets, through NSView, associated with the drag source widget. The data must be copied to the clipboard, prior to invoking the dragging action. In order to initiate a dragging action, the following parameters are required:

- An icon that will be used during the dragging operation.
- The location of the icon (screen coordinates) when the dragging operation is initiated. These coordinates are used to provide visual feedback of a failed dragging operation.
- The mouse event that initiated the dragging operation.
- The clipboard containing the data to be dropped.
- The NSView of the window initiating the dragging operation.
- A Boolean flag, denoting whether the icon will slide back in case the drag action fails.

Under the Cocoa DND there is no support in retrieving the action applied to the data during a successful drop action. TkDND assumes that this action is always **copy**.

3.2.4 Communicating events

Cocoa DnD supports five types of events that can occur during a drag and drop session, and most of them involve the drop target. These events are:

- draggingEntered
- draggingUpdated
- draggingExited

- prepareForDragOperation
- performDragOperation

3.3 GNU/Linux

Gnu/Linux is currently the least supported operating system by TkDND 2.x. The main obstacle of this platform is the complete absence of any public API that can be used for DnD operations, along with various compatibility issues observed among the various widget toolkits available for the platform. The dominant DnD protocol under Linux is XDND, details of which can be found at [9]. TkDND offers limited support for this protocol, by supporting only dropping actions. However, currently dropping from applications on Tk applications does not work: while the dropping operation is performed, TkDND is unable to retrieve the dropped data from the clipboard due to timestamp mismatches. Regarding this issue, a TIP [14] has been proposed to enhance the "selection" Tk command, which hopefully will solve this incompatibility.

4 Conclusions

This paper presents TkDND, a Tcl/Tk extension that aims to add cross-application drag and drop support to Tk, for popular operating systems, such as Microsoft Windows, Apple OS X and GNU/Linux. TkDND is an open source project, implementing a Tcl extension written in C, C++ and Objective C, distributed under the BSD license. The source code is hosted at the SVN repositories of SourceForge [17]. The current public release is TkDND 2.2, offering stable support for the Microsoft Windows and Apple OS X (Cocoa framework) operating systems. Support for the Linux operation system is currently missing, with stable support under Linux being a desirable, but difficult to achieve, target.

5 References

[1] Drag and Drop in Tk: http://wiki.tcl.tk/571
[2] The BWidget toolkit: http://wiki.tcl.tk/2251
[3]BLT: http://blt.sourceforge.net/
[4] Olednd: http://wiki.tcl.tk/571
[5] Motif: http://www.opengroup.org/motif/
[6] LessTif: http://lesstif.sourceforge.net/
[7] The GTK+ Project: http://www.gtk.org/
[8] GNOME: The Free Software Desktop Project: http://www.gnome.org/
[9] XDND – Drag-and-Drop Protocol for the X Window System: http://www.newplanetsoftware.com/xdnd/
[10] Qt – A cross-platform application and UI framework: http://qt.nokia.com/
[11] The KDE Community: http://www.kde.org/
[12] Xlib: http://en.wikipedia.org/wiki/Xlib
[13] Mozilla Firefox: http://www.mozilla-europe.org/
[14] TIP 370: http://www.tcl.tk/cgi-bin/tct/tip/370.html
[15] OLE – Object Linking and Embedding: http://en.wikipedia.org/wiki/Object_Linking_and_Embedding
[16] Drag and Drop Programming Topics for Cocoa: http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/DragandDrop/DragandDrop.html
[17] TkDND at SourceForge: https://sourceforge.net/projects/tkdnd/