Drag-and-Drop in GTK+ and GNOME

Owen Taylor
Red Hat, Inc.

otaylor@redhat.com

GTK+ provides a drag-and-drop implementation that allows for sophisticated negotiation of the drag operation (move, copy, link, or ask), and the format of the data that is transferred while providing interactive feedback to the user. It supports the Xdnd and Motif protocols, allowing interoperability with most common applications. By writing to the GTK+'s interfaces for drag-and-drop, the application developer transparently gains support for both protocols. Drag-and-drop is used throughout GNOME to provide tight integration between applications and the desktop.

Overview

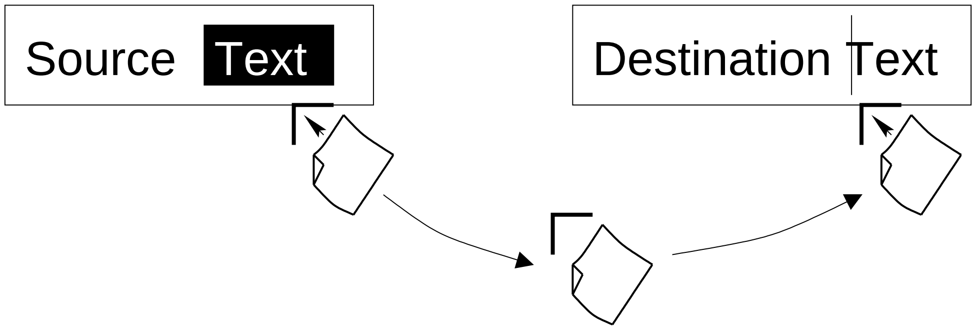

Figure 1. Moving text with drag-and-drop. Icon changes during drag to indicate whether a drop can occur is valid. (*drag over animation*) Cursor is displayed in target to indicate where text will be inserted. (*drag under animation*).

Drag-and-drop is a user interface idiom commonly used in most modern user interfaces. The user clicks on a user interface object, drags the mouse elsewhere and releases the mouse. Usually, this is used to trigger moving or copying information from one location to another. The operation that is performed is known as the *action*. Along with move and copy, another fairly frequent action is that of linking. In linking, a *hot link* is established, where the data at the target location is dynamically updated to reflect future changes to data at the source location. The action is typically selected by holding down a modifier key during the drag; for instance, the Control key indicates the copy action.

While the user is dragging, several forms of feedback can be provided. An icon is displayed that represents the object being dragged. Additional symbols may be attached to the icon to show the action being performed and whether a drop can be be performed at the current location. Feedback done in this fashion is known as *drag-over animation*. In addition, the appearance of the drop target may be modified to indicate when the cursor is over a valid drop target. Animation where the destination widget is modified is known as *drag-under animation*.

The source and target need to cooperate to determine what the result of the drag will be so that appropriate feedback is shown to the user. This can be done in a limited fashion if all possible destination windows attach information to their windows that the source application can read to determine what drop types will be accepted. However this is cumbersome and prevents having the feedback depend on the data of the drag. For that reason, most implementations of drag-and-drop negotiate the drag status by exchanging messages between the source and target applications.

Protocols

A number of different protocols for drag-and-drop have been used under Unix and the X Window System. Historically, the commonly used protocols have been the Motif and OfficeX protocols, along with a number of proprietary vendor-specific protocols. The Motif protocol is the native drag-and-drop protocol of the Motif toolkit. [1] The protocol provides a comprehensive set of capabilities, but is very complex to implement and not documented. A fairly complete job reverse-engineering the protocol was done by the LessTif project [2] but in general, the Motif protocol is not a satisfactory solution for non-Motif applications.

The OfficeX protocol is much a simpler protocol that has been used in a number of free software projects. It covers the basics of drag-and-drop, but has no provision for dynamically communicating between the source and target to provide the correct feedback. A modified version of OfficeX protocol is used in KDE. [3]

Clearly, there was a need for a new protocol with similar capabilities to the Motif protocol, but simpler to implement and available as an open, public, standard. This need was filled by the Xdnd protocol. [4] The Xdnd protocol was developed by a group of interested developers, including representives from Red Hat Software [5] , Troll Tech [6] (the makers of the Qt Toolkit), and a number of different free software projects. It is a lightweight protocol that allows for feedback during the drag, and flexible negotiation of drag action and the format of transferred data. It also provides the necessary mechanism to for an additional type of action: "ask". For this action, when the drop occurs, a menu pops up allowing the user to select an action. This mode of operation is often easier on the user than requiring them to remember an obscure set of modifier keys for each drag action.

The Xdnd protocol has rapidly gained support. Not only is it supported by two of the most commonly used toolkits, GTK+ and Qt, but support for it is also being added, or has been added to a number of applications not using these toolkits, such as XEmacs [7] and StarOffice. [8] However, there is still a large installed base of programs using the Motif toolkit that support only Motif drag-and-drop.

Data Types

An important part of the information that must be negotiated during the drag is the type of the data. There must be a standard way of representing data types for the drag. The traditional way of representing data types for interprocess communication under X is by a standard set of names defined in the X Inter-Client Communication Conventions Manual (ICCCM). However, these names are out-of-date with respect to currently used data types, and since this registry is not widely used in the computing industry, there is little incentive to use it. For this reason, the Xdnd protocol specifies that data types are represented using standard MIME types. [9]

Although the choice of which MIME types to use depends on the applications needs, there are a few standard MIME types that are somewhat independent of application. For example, for simple strings, the MIME type `txt/plain` is used.

Drag-and-Drop in GTK+

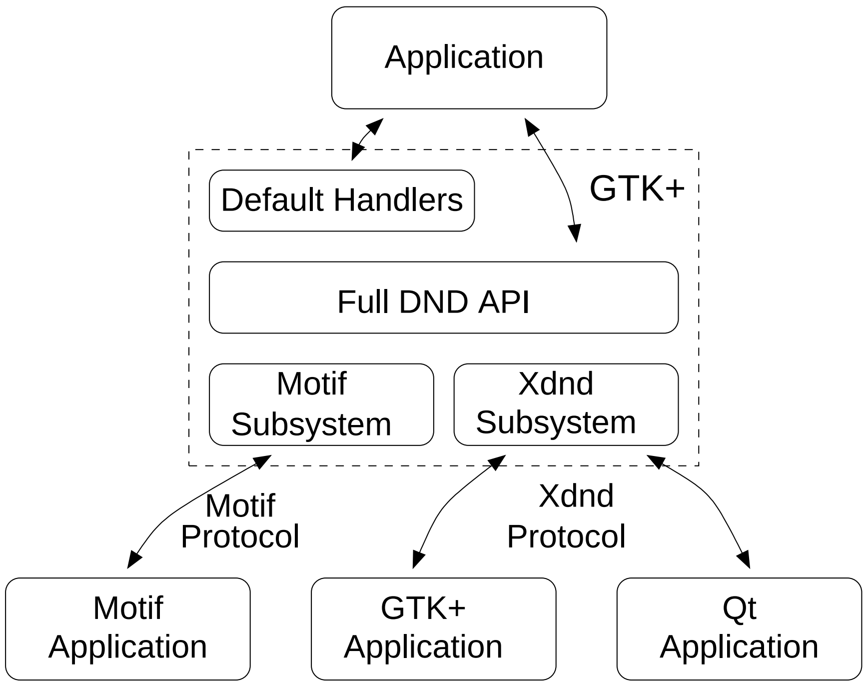

Figure 2. Architecture of drag-and-drop in GTK+

The GTK+ toolkit [10] internally supports both Xdnd and Motif drag-and-drop. On top of the protocol implementations, GTK+ provides a comprehensive set of interfaces for the application developer who wants to use drag-and-drop. By writing to these interfaces, the developer tranparently gains support for both protocols, along with any protocols which are added to GTK+ in the future.

The basic principle of GTK+ drag-and-drop is that there is an extremely flexible interface underneath that allows applications fine-grained control over the details of drag-and-drop, and then, on top of this, there are a set of default behaviors that the application can request. By requesting the default behaviors, the amount of work that the application needs to do is greatly reduced. This gives the developer the best of both worlds - it is very easy to get simple instances of drag and drop going, but the application can also customize the behavior when more sophisticated actions are needed.

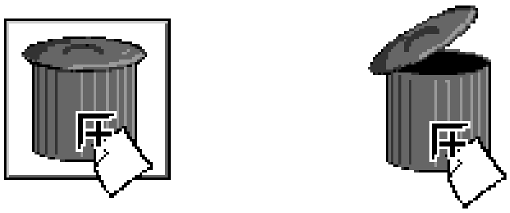

Figure 3. Default and custom drag-under highlighting

As an example of this, consider the drag-under animation mentioned above. If the application requests the `GTK_DRAG_DEFAULT_HIGHLIGHT` behavior for a widget, then when the user is dragging over that widget and it is a valid drop target, GTK+ will automatically draw a highlight border around the widget to indicate that the drop will be received by that widget. However, if this default behavior is not requested by the application, then it can directly handle the signals when the drag enters and leaves the widget and display a custom form of highlighting. For instance, a trashcan drop target might open when the drag enters the widget. (See Figure 3)

Drag and Drop in GNOME

GNOME [11] applications use the GTK+ interfaces for drag and drop, however GNOME also adds some more conventions about the types of data that are drag-and-dropped, and perhaps more importantly, provides an environment to the user in which drag-and-drop is available thoughout. The most prominent place source and target for drag-and-drop in GNOME is `gmc`, the GNOME file manager. `gmc` not only allows the user to access the file system, it also manages the desktop. The desktop provides a central place for the user to drop files, URL's, program launchers, and other types of data.

The drag-and-drop implementation in GTK+ is quite complete and should provide for the needs of GNOME for the forseeable future. However, what will develop is the type of objects that are tranferred via drag-and-drop. Currently, most drag-and-drop in GNOME is transferring file names; as the Bonobo document model becomes used in more parts of GNOME, it will become more common to transfer the object references for Bonobo objects via drag-and-drop. This will result in considerable more power and flexibility since it will be possible to do things like change the form of transferred data *after* it has been transferred, and also to do true hot-link embedding of one document into another.

Notes

1. Motif (http://www.opengroup.org/desktop/)
2. The LessTif Project (http://www.lesstif.org/)
3. The K Desktop Environment (http://www.kde.org/)
4. The Xdnd Protocol (http://www.cco.caltech.edu/~jafl/xdnd/)
5. Red Hat Software (http://www.redhat.com/)

6. Troll Tech (http://www.troll.no/)
7. XEmacs (http://www.xemacs.org/)
8. StarOffice (http://www.stardivision.com/)
9. MIME is defined in RFC's 2045-2049. A registry (ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/) of MIME types is maintained by the Internet Assigned Numbers Authority (IANA).
10. GTK+ (http://www.gtk.org)
11. GNOME (http://www.gnome.org)