

Cut 'n Paste; Drag 'n Drop

Transferring Data

- Cut and paste via the clipboard and drag and drop allows for (relatively) easy data transfer within and between applications
- Expected behaviour of any application

The Clipboard

- Ubiquitous data transfer method
 - Copy information (or pointer to information) to clipboard
 - Other applications can read data from the clipboard
- Any application can read this information
 - A potential security risk
 - Clipboard not accessible to Java applets running in web browser
- Requires common data formats to work seamlessly
 - Text is no problem
 - What about other formats?

The Clipboard: Formats

- Consider graphics
- How do we deal with:
 - Drawings in vector-based drawing programs?
 - Bitmap images?
 - Images from different file formats (JPEG, TIFF, GIF...)
 - 3D graphics?
 - PostScript drawings?
 - Charts?
 - Proprietary graphics formats?

The Clipboard

- When data is placed on clipboard, application indicates the formats in which it can provide the data
 - Example: “I can provide it as a vector image, bitmap image, or as text”
 - Simplest case: immediately place each supported data format on the clipboard, most preferred to least preferred
 - Mac Human Interface Guidelines specify that all applications must support either plaintext or an image; should always be able to cut/paste something.
- Data is not always copied to clipboard immediately
 - Why not?
 - What are implications?

Placing Data on Clipboard

- Data may be available in many formats
 - Wasteful to put all formats on clipboard at once
- Data may never be pasted
 - Again, wasteful to commit memory to a copy unless it is needed
 - Particularly after a “cut” operation, which can be used in place of delete ...
- If data is not immediately placed on clipboard:
 - Must create a copy if user changes data locally
 - Must put it on clipboard if application exits
 - Or at least prompt user
- Clipboard a function of the underlying windowing system, toolkit
 - Java will do it differently from Cocoa from Windows...

Java Clipboards

- Relevant packages:
 - java.awt.datatransfer (Clipboard, Drag and Drop)
- Relevant classes
 - Clipboard
 - DataFlavor
 - Transferable
 - Toolkit

Java Clipboards

- Local and system clipboards
- Local clipboards are named clipboards holding data only accessible by the application
 - `new Clipboard("My clipboard");`
- System clipboard is operating-system-wide clipboard
 - `Toolkit.getDefaultToolkit().getSystemClipboard()`
- System clipboard not available to applets

Copying Data to Clipboard

- Basic steps:
 - Get clipboard
 - To copy, create a Transferable object
 - Defines methods for responding to queries about what data formats (DataFlavors) are available
 - Defines method for getting data of specified type
 - Set clipboard contents to the new Transferable object
- Transferable object encapsulates all the data to handle the copy operation later
 - Similarities to what other object?

Transferable

- Encapsulates all data to copy in an object
- Similar in spirit to UndoableEdit
- Methods:
 - DataFlavor[] getTransferDataFlavors()
 - boolean isDataFlavorSupported(DataFlavor flavor)
 - Object getTransferData(DataFlavor flavor)

Pasting Data from Clipboard

- Basic steps:
 - Get clipboard
 - See if it supports the desired data format (DataFlavor)
 - Get the data, casting it to the proper Java object

Code Review: Cut 'n Paste

- DTPicture
 - first half: setting, painting image, focus highlighting
- PicturePanel
 - doCopy/doCut
 - Related
 - selectedPic
 - PicFocusListener
 - doPaste
- PictureTransferable

TransferHandler

- The TransferHandler class will be used for drag 'n drop. It can also be used for supporting cut 'n paste.
- The cut 'n paste support:
 - providing Action objects (actionListeners) for cut/copy/paste
 - exportToClipboard
- See Java Tutorial for more info

Drag and Drop

- Uses same Transferable, DataFlavor objects to pass information around
- Need to specify drag and drop sources

Supporting Drag

- “Dragging” refers to copying something *from* your control
- To support dragging:
 - Set a transfer handler for each component that supports D’nD
 - In the source of the drag, define a mouse listener that knows when a drag has started.
 - When a drag has started, get the component’s transfer handler and call its `exportAsDrag` function

Supporting Drop

- Drop support allows stuff to be dropped on component
- TransferHandler:
 - override `importData`

TransferHandler

- Methods:
 - boolean importData(JComponent c, Transferable t)
 - int getSourceActions(JComponent c)
 - returns one of COPY, MOVE, or COPY_OR_MOVE
 - Transferable createTransferable(JComponent c)
 - void exportAsDrag(JComponent c, InputEvent e, int action)
 - action is one of COPY, MOVE, or COPY_OR_MOVE
 - void exportDone(JComponent source, Transferable data, int action)

Code Review: Drag 'n Drop

- **PicturePanel**
 - Set handler in constructor
- **DTPicture**
 - DragGesture inner class
- **PictureTransferHandler**
 - dropping (canImport, importData)
 - dragging (getSourceActions, createTransferable, exportDone)

Politics of Data Formats

- Data formats can be instruments of control
 - The value of an application resides in its ability to create, manipulate, manage, and reference data
 - The more that an individual or company's worth is tied up in data, the more reliant they become on the tools that allow them to access and manipulate that data
 - This creates a market *disincentive* to create open data formats

Summary

- Make data available in a variety of data formats, ordered by preference.
- Target chooses the most applicable format to use.