New Technical Notes
Macintosh
®
Developer Support

HyperCard And You: Economy Edition

Platforms & Tools

Revised by: jeremy j. bornstein, death dwarf of minraud February 1991
Written by: Chris Knepper November 1987

This Technical Note describes some HyperCard anomalies with which developers should be familiar when developing stackware, and it documents differences between HyperCard versions where appropriate.
Changes since November 1987: Updated material with regard to HyperCard 2.0 and condensing or obsoleting information on bugs which have been addressed.

The 15 Billion Horsemen of the Apocalypse

With the introduction of HyperCard 2.0, many of the old bugs were quashed, and absolutely no new bugs were created. In fact, the software was so bug-free that it immediately attained Nirvana and Apple has had problems getting it to do anything since. Just kidding.

HyperCard File Format

The HyperCard file format is available for licensing on a case-by-case basis. Since HyperCard has moved to Claris, developers should contact Claris for more information if you feel that your product is a substantial and important addition to HyperCard and the Macintosh.

Since HyperCard allows developers to control data input and output to files using HyperTalk, most "needs" for file formats may be easily met by using HyperTalk and writing scripts. In particular, the following HyperTalk commands are useful:

```
open file fileName
close file fileName
read from file fileName until <delimiter>
```

Versions of HyperCard after 1.0.1 come with examples of how to use these commands. For those with HyperCard 2.0 release stacks, look in the Power Tools stack for the "Export Stack Scripts" card, which provides frameworks for building custom file I/O routines.

Import(ant) Tip

Scripts which create many new cards, e.g., scripts which import data, should be sure to keep a counter of the number of cards created and call `doMenu "Compact Stack"` every 200 or so cards. Scripts which lack this feature may cause a variety of strange results, including stack corruption, which are revealed when Compact Stack is eventually selected.

Playing Sounds

In versions prior to 2.0, HyperCard could only play format 2 `'snd '` resources. In version 2.0 and later, HyperCard supports both format 1 and format 2 `'snd '` resources, which are documented in *Inside Macintosh*, Volume V, The Sound Manager (earlier versions did not include the format 2 description).

Visual Effects On Monitors With Multiple Bit Depths

With HyperCard 2.0, visual effects work with all monitor depths up to eight bits, although these effects are faster on a one-bit depth screen. In previous versions of HyperCard, the monitor had to be set to a one-bit depth for visual effects to be visible.

High Disk Space Requirement For BackGround Printing

Running HyperCard under MultiFinder with Background Printing enabled requires extensive disk space for printing large jobs, such as printing a stack (with Print Stack… in the File menu) of 100 cards. For example, a ten card stack may only be 20K in size, but the spool file might be greater than 1MB. This requirement is due to Print Monitor’s spooling the job to disk. Note that this requirement does not hold true for report printing, where no graphics need to be spooled.

Possible Printing Problems

LaserWriter Timeout Errors

The LaserWriter IINT and LaserWriter drivers prior to 6.0.1 had a bug which resulted in long jobs failing to print due to a timeout error after approximately one hour. Version 6.0.1 of the LaserWriter driver addressed this problem; however, you may still want to print large jobs in batches using a HyperTalk script.

Printing To Non-PostScript Printers

HyperCard 2.0 would not print to non-PostScript® printers. HyperCard 2.0v2 fixes this bug.

Word Wrap

HyperCard 2.0 uses TextEdit's word break routine with some optimizations. While it fixes bugs in the old HyperCard word break routine, it does things somewhat differently. Therefore, there may be significant changes in the way text fields are wrapped, and this may adversely affect the visual design of stacks intended for use with earlier versions of HyperCard. One of these changes is that HyperCard no longer considers quotation marks to be separate from their attached words.

Note that stacks designed under versions of HyperCard prior to 2.0 continue to use the old word break routine until converted to 2.0 format.

Beware The on idle Handler

An `on idle` handler in a HyperCard script executes while the user types into a field. A nasty side-effect is that the insertion point may disappear while the user types into a field, resulting in the Macintosh beeping since it cannot interpret the characters entered, or in the characters being displayed in the Message Box if it is open. A particularly insidious example is the following handler in a background script:

```
on idle
  put the time into bkgnd field "Time"
end idle
```

If the user types into a field with the example handler in a background script, the insertion point disappears from the field whenever the time changes, for example from "6:42 PM" to "6:43 PM," requiring HyperCard to change the contents of background field "Time."

A solution to work around this problem involves checking the selection within the `on idle` handler, and reselecting the selection after the `on idle` handler has done its dirty work. The following handler fixes the problem with the preceding one:

```
on idle
  put the selectedChunk into oldSelectedChunk
  put the time into bkgnd field "Time"
  select oldSelectedChunk
end idle
```

Note that if the `on idle` handler changes field contents frequently, this technique produces unsightly blinking of the selection or of the insertion point. However, if the `on idle` handler is well-behaved, it avoids such behavior and uses this technique only as insurance.

find Command

Previous versions of HyperCard had various bugs with the `find` command. HyperCard 2.0 fixes these bugs, and `find` now works with all fields, except those for which the `dontSearch` property is set to `true`. If `dontSearch` has been set to `true` for a card or a background none of the fields within the card or background are searched.

Note that when `dontSearch` is set to `true` for a background, **none** of the fields on the background or on cards which belong to that background are searched.

Optimizing find

HyperTalk's `find` command works best when at least three characters per find criteria are specified, and the stack has recently been compacted with the Compact Stack item in the File menu. Please note that specifying less than three characters or using word break characters, such as the dash (—) or space ( ), for any of these three characters results in a slower find.

The more trigrams that are specified for the `find` (strings with at least three characters), the faster the find; thus, the following `find` in a company phone list:

```
      find "Thanasis Metagreek Appraisal"
```

would find "Thanasis Kehagias" in "Metagreek Appraisal Department" faster than would:

```
      find "Th"
      find "Thanasis"
      find "Thanasis K"
      find "Thanasis Kehagias"
```

Paint Tools and HyperTalk

HyperTalk can control all painting tools and commands except the polygon tool and the special effects tools (Rotate, Distort, Slant, and Perspective).

Selecting Contents Of A Field With HyperTalk

There are several ways of selecting text in a field with HyperTalk: by simulating a double click, a Shift-click, or a drag. The three button scripts presented here show how to do this.

The first shows a method of simulating a double click. This script performs a double click at the `loc` of a card field (approximately the center of the field's rectangle) then puts the selection into another card field:

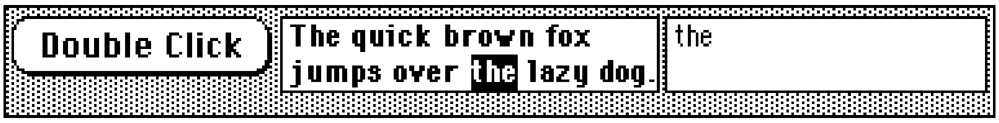

Figure 1–Simulating A Double Click

```
on mouseUp -- card button "Double Click"
  click at the loc of field "myField1"
  click at the loc of field "myField1"
  if the selection is empty
  then put "No selection" into card field "DoubleClickSelection"
  else put the selection into card field "DoubleClickSelection"
end mouseUp
```

This second script demonstrates a method of selecting text in a card field with the shift-click and puts the selection in another card field:

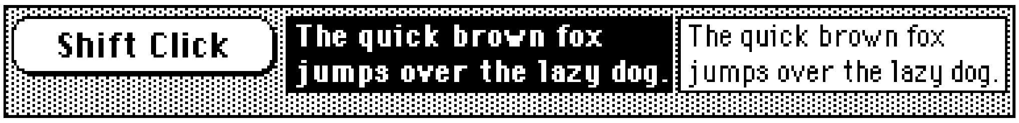
Shift Click
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

Figure 2–Simulating A Shift-Click

```
on mouseUp -- card button "Shift Click"
  get the rect of bkgnd field "myField2"
  click at item 3 to 4 of it
  click at item 1 to 2 of it with ShiftKey
  if the selection is empty
  then put "No selection" into card field "ShiftClickSelection"
  else put the selection into card field "ShiftClickSelection"
end mouseUp
```

The third script presents a method of selecting text by dragging over it. This method, however, requires `with shiftKey` as a parameter to `drag` in order to work correctly:

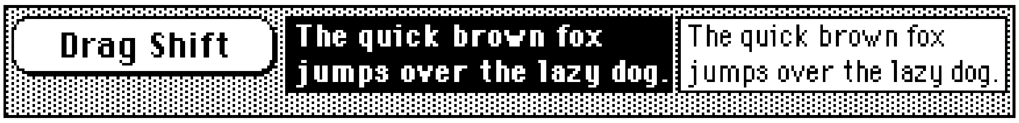
Drag Shift
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.

Figure 3–Simulating A Drag

```
on mouseUp -- card button "Drag Shift"
  get the rect of bkgnd field "myField3"
  drag from item 1 to 2 of it to item 3 to 4 of it with shiftKey
  if the selection is empty
  then put "No selection" into card field "DragShiftSelection"
  else put the selection into card field "DragShiftSelection"
end mouseUp
```

Dialing The Telephone

dial Command

System Software 6.0.7 has a bug on the Macintosh IIsi which causes HyperCard 2.0 to run the tones together when executing a `dial` command. Version 2.0v2 fixes this bug; however, it contains another bug in which a comma (,) in the `dial` command is ignored if it is preceded by more than three characters which produce tones.

HyperCard 2.0v2 fixes all known bugs with the dial command other than the one just documented. In addition, the dial command now supports the A, B, C, and D buttons found on military telephones.

.TouchTone 'DRVR'

Note that the .TouchTone `'DRVR'` was removed in HyperCard 1.1.

Launching Applications Under MultiFinder With open

HyperCard 2.0 returns a value in `the result` if an application launch with the `open` command fails; this message is either “out of memory” or “couldn’t open that application,” depending upon the reason it could not launch the given application.

mouseDown and mouseUp Messages in Scrolling Fields

HyperCard does not send the `mouseDown` and `mouseUp` messages to scrolling fields when it detects a mouse click in the field’s scroll bar.

Passing it As A Parameter To Handlers

Passing `it` as a parameter to a handler is no different than passing any other parameter to a handler. This means that: it is always a local variable to the handler in which it is used, and it is passed to a handler by value, not by reference.

Disappearing Resource Forks

If an `'XCMD'` adds a resource fork to a currently open stack, HyperCard does not know about it. When a stack is compacted, and HyperCard does not think that the stack has a resource fork, then the resource fork is not preserved after the compaction. The solution to work around this is to `go home` before compacting the stack. Then, when the stack is opened again, HyperCard recognizes that the stack contains a resource fork and behaves appropriately.

Using Globals From 'XCMD' Resources

It used to be the case that one could not create a global from an `'XCMD'`. HyperCard 2.0 fixed this bug; however, there is an issue in that messages an `'XCMD'` passes back to HyperTalk are executed in the scope of the handler which called the `'XCMD'`. This situation means that the calling handler must declare `global someGlobal` for the following to work:

```
SendCardMessage(paramPtr,”\p put someGlobal into cd fld 1”);
```

Some people may notice that there is still the problem of how to make sure the handler declares the global if you do not want the `'XCMD'` to be dependent upon its calling script. Simply replace the example with the following code:

```
SendCardMessage(paramPtr,”\p global someGlobal \n put someGlobal into cd fld 1”);
```

Note the return character (\n) in the second example above—the message passed to HyperCard becomes a two-line message (yes, it is possible). The first line makes the handler’s scope aware of the global `someGlobal`, while the second line does what you want with the global. It is now safe to pass a global variable back to HyperTalk since the `'XCMD'` has added it to the local scope of the handler.

La Bomba (Ya No Soy Marinero)

It is imperative that scripts test the global property `the heapspace` in a variety of scripting situations which reduce the amount of available heap space. HyperTalk needs to have about 32K of heap space available at all times when a script is running. If `the heapspace` drops below this value, HyperCard aborts the script with an “out of memory” error. Furthermore, in its attempts to clean up, HyperCard may empty those global variables upon which it was working when the error occurred.

Unfortunately, there is no way for a script to trap for out of memory errors when they occur, although it is sometimes possible to enter the debugger to determine exactly what was happening when memory ran out like so many gamma particles from Einsteinium. However, scripts can check `the heapspace` whenever they are about to do something which might require much memory, then either fall back to a less memory-intensive way of performing the same operation or simply report the error and abort safely.

For example, in scripts which import large amounts of data into a variable, test `the heapspace` frequently, and when it gets small, dump the contents of the variable into a field.

Another good use for `the heapspace` is in testing whether or not `'XCMD'` or `'XFCN'` resources are properly disposing of memory. A test script can call the external command repeatedly, displaying `the heapspace` in the Message Box each time. For example, the following script repeatedly calls the `'XCMD'` flash, putting `the heapspace` into the Message Box after each call, until the Shift key is pressed. If the value displayed in the Message Box decreases consistently, it should be clear that the `'XCMD'` is eating memory, perhaps by allocating blocks with calls to `_NewHandle` which are not disposed of with a corresponding call to `_DisposHandle`.

```
on mouseUp
  repeat while the shiftKey is up -- depress the Shift Key to stop
    flash 2 -- call your XCMD or XFCN
    put the heapspace -- monitor this value to see if it's decreasing
  end repeat
end mouseUp
```

Further Reference:

- HyperCard Script Language Guide
- Getting Started With HyperCard 2.0
- HyperCard 2.0 Release Notes
- The Age of Reason