

Notes: Since HyperCard was introduced, a variety of magazines, or "stackzines" have popped up. Generally, we've only been able to find one or two issues of them and most of them have either been short-lived or their issues have not become widely available. What follows are comments, tips, an article, and so on from five of those "Stackzines" which we have found useful and hope you do too.

The first article contain some howtos which you may or may not want to try, but contains some excellent insight into the workings of HyperCard.

From "ScriptEase" in Stackazine, 1988. Author unknown.

With all the levels presented to the HyperCard programmer, it is sometimes difficult to tell where to put the script or graphics images that he/she is creating. It really doesn't matter where you put things if you have enough memory and the stack works the way you desire. If; however, you are going to share that stack with the rest of us, then you should make sure the stack is as efficient as possible, not all of us like having our hard disks filled with inefficient scripts and animations.

Creating efficient stacks requires thinking in terms of levels. Ask yourself, "What is common to the set of cards, a subset of cards, or to the entire stack?" If you can find something in common, put it at a level that takes full advantage of the commonality.

The background field is one of the first places we encounter this principle. For example, if you are creating a simple data base and desire each new card to have the same fields on it, place the fields at the background level. Each new card will take on that background and thus, have all the fields ready for use every time you type command-N.

Most of you probably figured out that much during your first day with HyperCard; however, there are other areas where thinking in terms of levels has a role to play. Let's take a look at various examples of putting things in the right places.

If you want all stacks to have access to a certain piece of script, it is best to enter that script in the home card. An example would be the Stack level script I have in my home stack which takes me from whatever card I am currently on to my address stack. The code I entered in the stack level of my Home stack looks like this:

```
on address name
  go to first card of stack "address" -- go to the address list
  if the paramcount > 0 then
    find name -- find the name in the address stack
    put "find " & quote & name & quote -- more finds
  end if
end address
```

Once this script has been entered in the home card, it can be executed from the

message window. The message window is for entering script that you want executed immediately.

To toggle the message window between visible and invisible, type command-m. Once the message window is visible, you can type any script that HyperCard will allow.

Because "address" is now a command defined in the Home Stack, you can type "address" followed by a carriage return in this command window, you will be transported to the address stack. No matter where you are, you will suddenly appear in the address stack. If you want a specific name, then type "address John". This will open the address stack and find the first John in the stack. Each time you hit return, you will find the next John in the address stack. Once you are through looking for John, or any other person in your address stack, click on the return arrow and you are taken whence you came.

The Home card is also a good place to put sounds. By putting the snd resources into the Home card, and removing them from the stacks that use them, you can reduce the redundancy. Just remember to put the snds in if you are giving a stack to someone else, they may not have the same home situation as you.

An Animation Note:

It helps to think of animations as actors working on a stage. This becomes very obvious if you decide to write an adventure type game with pictures. The stage is put in the background, this would be things like the picture of a forest road or a castle on a hill. The actors would be things like tools that the player could pick up such as knives, bottles, turkey basters, apples, etc. as well as the people and creatures you meet on your journeys. All of the actors could be stored on a set of cast cards from which they could be copied and pasted on the proper card for the adventure. The pasting would always be done at the card level, in front of the stage.

As you can see, putting things in the right place can improve the efficiency as well as expand the functionality of HyperCard. Think in terms of levels, look for similarities in script, and keep on script'n.

From the first edition of Mark Trent's MouseUp

If there is a word on the screen that you wish to 'find' in other parts of the stack you can do command/F and then, while holding down the command key, click on the word.

If you wish to page rapidly through a stack-yet not so rapidly as with the "scan" button then you can hold down either command key/ 3

or command key/4. As long as you continue holding down both keys you will continue your high speed browsing.

When you select a number for a 'phone button to dial it does not have to be in the field adjacent to the 'phone button. It can be anywhere on the card.

The find mechanism works only on words in fields. If the word is written with the text tool from the painting pallet it will not be 'found.'

Top Five Command Keys for Browsing

1. COMMAND KEY/H: GO HOME
2. COMMAND KEY/3: NEXT CARD
3. COMMAND KEY/R: A MAP OF RECENT CARDS*
4. Command TILDE: RECENT CARD-
"RECENT" REFERS THE CARD YOU HAVE JUST LEFT.
5. COMMAND KEY/? : GET HELP.
ANYWHERE. NO NEED TO GO HOME.*

IF YOU ARE IN A STACK WITHOUT A MENU BAR YOU CAN BRING UP THE BAR BY USING COMMAND KEY/SHIFT.

Top Five Command Keys for Authoring/Painting

1. COMMAND KEY/N: New Card
2. COMMAND KEY/B: Background
3. COMMAND KEY/Z: Undo.
4. COMMAND KEY/A: Select All. NOTE: At present this only works for painting.
5. COMMAND KEY/Tab: Restores the Browse tool.

From LIMacStack & Kevin Moan, Aug & Nov 1989

Font Help

In order to make sure that the text in a field displays the font that the author intended, the stack must have access to that font.

A stack has access to a font when it resides in the user's System OR when it resides in the stack itself.

You may not have some of the fonts used in a stack in your System, but they can be installed into the stack with the application, Font DA Mover.

You can get the fonts in this stack for your own use by copying them into a new font Suitcase.

Add As Many Home Cards As You Want....

Even though the Home stack that you received with HyperCard had only one Home card, there is no reason to limit yourself to only one.

Actually the Home stack was designed to have as many different Home cards as you want. They can be organized into categories, types of stacks, or even an application launcher.

There are many custom Home cards available as Public Domain stacks that have many unique features to make your HyperCard environment a "gateway" to your total Mac environment.

The important thing is that you make the Home card do what you want it to do. Only then will you see that HyperCard can be the central switching point to the rest of your Mac activities.

We've also included two instructional stacks created by Kevin Moan for the HyperCard Sig of the Long Island Mac Users Group which may shed some light on various aspects of HyperCard.

The Sound of Music

by Harry Jones

Of all the features found in HyperCard, the ability to play music and sound effects is perhaps the most intriguing. Have you ever wondered where the music comes from?

In the Macintosh environment, every application program or data file can store graphics and digitized sounds in what's known as the resource fork. HyperCard has graphics, icons and sounds stored in the resource fork ready for you to access. The sounds include Harpsichord, Boing and Silence.

You can add your own sounds to a Hypercard stack by using ResEdit [Ed. Note - Resource Mover in HyperCard 2.0]. For detailed instructions, see the text file which accompanies ResEdit.

Once a sound has been installed in the resource fork of a stack, it can be accessed with a PLAY command. Consider this, however. If the sound is installed only in your home stack, HyperCard's hierarchy allows a play command to access it from any stack. But, if you run your stack on a Macintosh with a different home stack, all you will hear is silence. The sound must be installed in the resource fork of the current stack or some-where up the hierarchy, such as the home stack or HyperCard itself.

If you plan on distributing a stack with custom sounds, be sure to install the sounds in the resource fork of that stack. Be aware that digitized sounds eat up a lot of disk space, which translates to longer download times. You'll need to ask yourself if the extra fluff is really worth it?

There is an alternative that you really should consider, and that is tweeking with the existing sounds installed on all HyperCards, Boing and Harpsichord.

```
play <"voice">[tempo<tempo>]<"notes"><"octave">
```

Voice refers to the digitized sound stored in the resource fork of the current stack, home stack or HyperCard. Tempo is an option that controls the speed at which the notes will be played. If you use the command word "tempo" in a script you must include a tempo number, 200 is the tempo default. Notes are typed into the play command by letter, standard music notation is followed, A B C D E F G. The value of each note can be modified in three ways:

(1) Each note can be sharped (#) or flatted (b).

EXAMPLE: Play "Boing" "a a# f fb"

(2) The amount of time each note is played can be altered by entering a time value after the note. Whole Note (W), Half Note (H), Quarter Note (Q), Eighth Note (E), Sixteenth Note (S), Triplet (T), 64th (X). A note's time value can also be increased by one half of its value by placing a dot (.) after the duration parameter.

EXAMPLE: Play "Boing" "aw bh cq de fs dt"

(3) For our discussion, the third option is the most important because with it you can control the pitch or octave of any note. The normal octave range for HyperCard notes is 3 to 5. If you try 0, 1, 6, 7, 8, 9 or 79 for that matter, things start to get interesting.

EXAMPLE: Play "Boing" "b0 b1 b2 b6 b7 b8 b9"

By experimenting a bit, you'll create unique sounds without the overhead of adding sounds to the resource fork. Have fun with it!!

from HyperNews, Vol.1, No 1, Nov. 17, 1987