ICONstructor

General Info

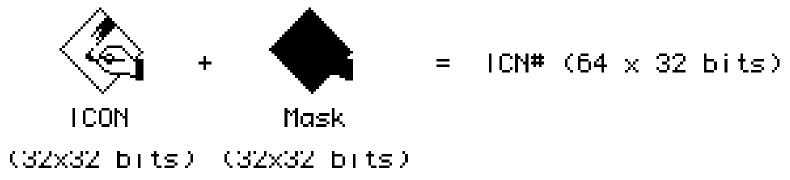
This program will allow you to open, create, modify, animate, and generally hack-up icons. Before we get into the "how-to's" of using this program, though, you'll need to know a little bit about icon related resources.

ICON & ICN#

There are really two different objects that we generally refer to as "icons." The icons you see in menus and dialogs usually have the resource type 'ICON', whereas the more familiar desktop variety can be identified by its resource type, 'ICN#'.

ICON - a 32 X 32 pixel object (128 total bytes). Turn some pixels on or off (1's and 0's in memory, respectively) and you get your little picture.

ICN# - called an *icon list* , it consists of an ICON plus a *mask*. This mask is a 32 X 32 bit entity, like an ICON, that tells the Finder how to draw the ICON when it gets selected (clicked-on). You put the ICON and the mask together into a 64 X 32 bit object that the Finder can use. A "normal" mask is usually just the ICON's outline filled in completely as below:



ANIMATION

You've probably seen Desktop icons that change when you click on (select) them. Scott Watson's Red Ryder has an icon like this. How is this done? Well, it's all in the mask. If you take an ICON the way it appears on the Desktop and another ICON that is the image you want when selected, take the BitXor (Exclusive OR) of these two to create a special mask, and add this mask to the original ICON, you get an ICN# that will produce the desired effect. (Whew!).
That is:

BitXor(Main ICON, ICON as Selected) => Mask
Main ICON + Mask => ICN# {creates the desired effect}

Note: The Exclusive OR is as follows:

0 Xor 0 => 0
1 Xor 0 => 1
0 Xor 1 => 1
1 Xor 1 => 0

Representing Icons

It is convenient to represent ICONs and ICN#s as lists of hexadecimal (base 16) numbers. A 32 bit integer (LongInt) can be represented by a string of eight hexadecimal characters. 32 of these numbers define an ICON. 64 define an ICN# (32 for the ICON + 32 for the mask).

If you're not a programmer you won't care squat about this, but this is what the funny numbers that ICONstructor uses are all about.

File Menu

Open...

There are four ways to get icons into the program. That is, get them on the screen.

1) Open TEXT - presents the standard SFGetFile dialog that will allow you to choose a text file to open. Open TEXT will "look at" any text file but, of course, it's looking for 32 lines of hexadecimal numbers. From the beginning to the end of the text file, Open TEXT reads a line of text and, if the line contains exactly eight valid hex-characters and at most one invalid one, it loads the number or else the line is ignored. If the text file contains less than 32 valid lines, the remaining lines will be filled with zeros. If more than 32 valid lines are present, the extras are ignored.

2) Open ICON rsrc - will load an ICON resource stored in a resource file that you will choose from the SFGetFile dialog. (However, you can't get at the ICON resources of the System file because ICONstructor won't find them. I have no clue why. Any ideas, folks?) You will then be presented with a dialog that will show the first ICON in the resource file and the question, "Is this the resource you wish to Open/Change?" (This same dialog is used in all the Open & Change procedures.) If it is the one you want to open, choose 'YES'. IF not, you can search through the ICONs in the file by choosing 'NO/NEXT' until you find the one you want, or choose 'CANCEL' to dismiss the dialog. If you have chosen an ICON, the ICON will be displayed in the program window.

3) Open ICN# rsrc - is similar to 'Open ICON rsrc' except it lets you browse through ICN# resources. When you find the one you want and choose 'YES', it will load the ICON portion of the list. It does NOT load the mask.

4) Use the on-screen grid, your mouse, and your talent to create your own ICON image.

Save ...
ICONstructor allows you to save your ICONs and ICN#s as hexadecimal text files or as resource files that the program will create for you. (In addition you may add resources to existing resource files - see Add ...) However, it will NOT ask you if you want to save changes before you quit the application. So ... If you've got something worth keeping, use one of the Save, Add, or Change options before you quit.

1) Save Screen -> TEXT - creates a text file of the hex characters showing on the screen headed by an RMaker defintion. You get:

```
 Type ICON = GNRL
   ,128              {change this ID with EDIT or whatever}
 .H
 XXXX....
 ........etc.
```

2) Save List -> TEXT - same as 'Save Screen -> TEXT' but the list is defined by the ICONs you have locally stored as "MAIN" and "MASK" (see Save_Local Menu). You also get the RMaker header as above except the first line will read, "Type ICN# = GNRL".

3) Save Screen -> ICON - creates a new resource file and puts the ICON currently displayed on the screen into it.

4) Save List -> ICN# - creates a new resource file and puts the "MAIN"+"MASK" ICN# into it. (See Save_Local Menu)

Add ...
The following options allow you to add an ICON or ICN# resource to the resource fork of an existing file. (except the System File)

1) Add Screen as ICON - adds the ICON currently displayed on the screen to the resource fork of an existing file that you will choose with the SFGetFile dialog.

2) Add List as ICN# - adds the "MAIN"+"MASK" ICN# (See Save_Local Menu) to the resource fork of an existing file.

Change ...

The Change options will allow you to actually alter ICON and ICN# resources. When dealing with ICN#s, you'll probably want your
changes to be displayed on the Desktop. To do this, you'll need to change the appropriate ICN# in the file named "Desktop" (for application and document icon lists). A change to the Desktop file only will produce the cosmetic effect and allow you to undo all changes by rebuilding the Desktop file (re-booting with the <command> & <option> keys down). However, if you want permanent changes, change the resource in the application AND the Desktop.

1) Change an ICON - displays the standard SFGetFile dialog to allow you to choose the file that contains the ICON resource you want to change. Browse through the ICONs in the 'OPEN/CHANGE' dialog you'll get until you locate the one that you want to alter. When you find it, choose the 'YES' button and the ICON will be replaced with the one currently showing in ICONstructor's window.

2) Change an ICN# - same as 'Change an ICON' except you will be looking at ICN# resources. When you choose the 'YES' button in the dialog, the "MAIN"+"MASK" (see Save_Local Menu) ICN# will be substituted for the list you have chosen.

Save_Local Menu

MAIN - the icon as it appears (unselected) on the desktop.
SELECTED - as you want the icon to appear when you click on it - if appropriate.
MASK - mask for the MAIN ICON

To work with these three 32 X 32 bit objects that you've either opened or created, you'll need to save them within the environment of the program. This will allow you to switch easily between them using the Show Menu options.

Note: The options in this menu do NOT save information to disk. They merely provide local storage for the objects that you will be manipulating. For permanent saves, use one of the Save options in the File Menu.

1) As Main - locally saves the icon showing on the screen as the "MAIN" ICON, which, of course, can be the ICON portion of an ICN#.

2) As Selected - locally saves the screen icon as the "SELECTED" ICON. When you have ICONs saved locally as "MAIN" and as "SELECTED" you can use the 'ANIMATE' option in the AutoMask Menu to generate a BitXor mask for the "MAIN" ICON.

3) As Mask - If you don't want the BitXor mask you can create your own and use this option to save it within the program.

Show Menu

The Save, Add, and Change ICON options in the File Menu require that the ICON be showing on the screen. The Show Menu options will display the ICONs that you have saved locally.

1) Main - displays the "Saved Locally as Main" ICON.
2) Selected - displays the "Saved Locally as Selected" ICON.
3) Mask - displays the "Saved Locally as Mask" ICON.

Shift & Erase Menus

1) Shift - shifts the on-screen image one pixel in the specified direction (left,right,up,down). Pixels shifted off the grid are lost.

2) Erase - simply clears the screen. Does not effect anything that has been saved.

AutoMask Menu

When you have saved ICONs locally as "MAIN" and as "SELECTED", 'Animate' in the AutoMask Menu will create a BitXor mask and automatically save it locally as "MASK".

An Example

You've got a program with a hideously ugly icon. You'd like to draw your own (and animate it!). Start up ICONstructor.

1. Draw your ICON in the grid provided or open an icon to start with using one of the Open options in the File Menu. When you get it the way you want it, choose "as Main" from the Save_Local Menu.

2. Draw, open, modify or otherwise get another ICON on the screen. This will be the image you'll get when you click on your new icon. Choose "as Selected" from the Save_Local Menu.

3. Choose "Animate" from the AutoMask Menu. It'll probably look rather strange - don't worry about it.

4. Choose "Change an ICN#" from the File Menu. It'll present you with the standard GetFile dialog - choose "Desktop".

5. You'll see a dialog that asks, "Is this the resource you wish to Open/Change?" and displays an icon. If it's displaying the ugly one, choose the 'YES' button. If not, choose the 'NO/NEXT' button until you see the nasty picture that you want to modify. "Uuugh! There it is!" you cry in horror. Choose the 'YES' button.

6. Quit the program and sigh contentedly as you see that the little bugger has been changed.

Note: When you copy the program to another disk the changes will go with it. If you do want to get the old one back you can re-boot with the <option> & <command> keys down.