APPENDIX

Newton Toolkit Enhancements

Newton Toolkit version 1.6.4 provides support for pix families and gray icons. Two editors have been significantly changed for this purpose, the picture editor and the application icon editor. These new build-time functions have also been added to NTK: `GetSoundFrame`, `MakeBinaryFromHex`, `MakeDitheredPattern`, `MakeExtrasIcons`, `MakePixFamily`, and `UnpackRGB`.

Editors

The picture and application icon editors have been changed to support pix families.

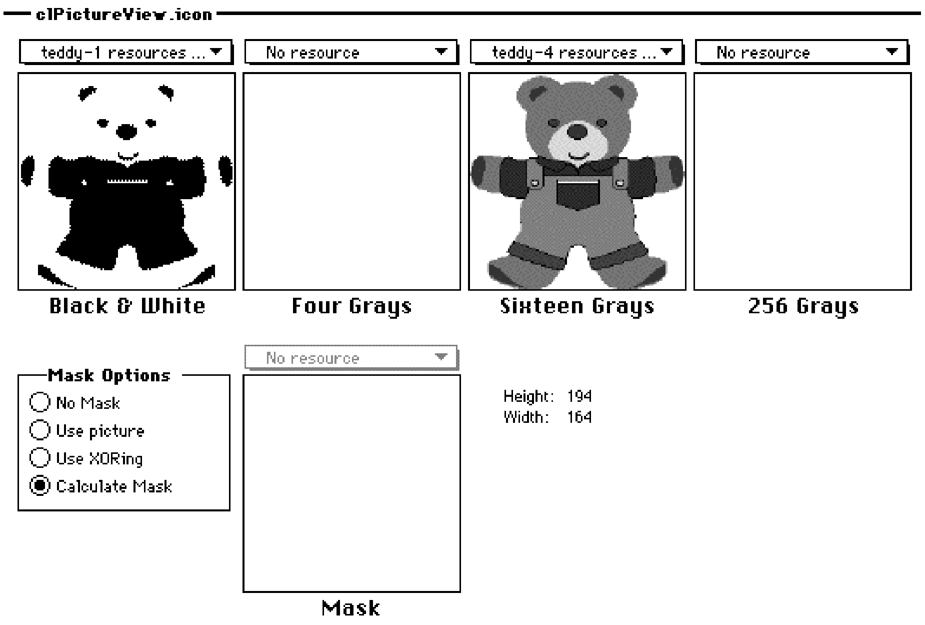
Picture Slot Editor

A picture slot editor, shown in Figure A-1, is used to create a pix family from a number of PICT resources at different bit depths to use for the `icon` slot of a `clPictureView`. The editor allows you to include different PICTs to display on black and white, 4 grays, 16 grays, and 256 grays screens. You may specify any number of these pictures; the system software determines the appropriate image to display at run time.

Note

The picture does not have to be the same bit-depth as the picture window it is placed in. For example, an 8-bit picture could be specified for the “Black and White” window. The picture would be properly displayed on a black and white screen. However, this would waste memory, and the picture would be drawn slower. You should reduce the bit depth of each PICT to the appropriate setting in NTK with a graphics utility on the desktop machine. ◆

Figure A-1 NTK's picture slot editor



Mask Options
- No Mask
- Use picture
- Use XORing
- [✓ Calculate Mask]

Height: 194
Width: 164

NTK displays the width and height in pixels. All included PICTs must be the same size. Each of the images is selected from a popup menu over the image. This popup menu contains all the PICT resources from resource files included in the current project.

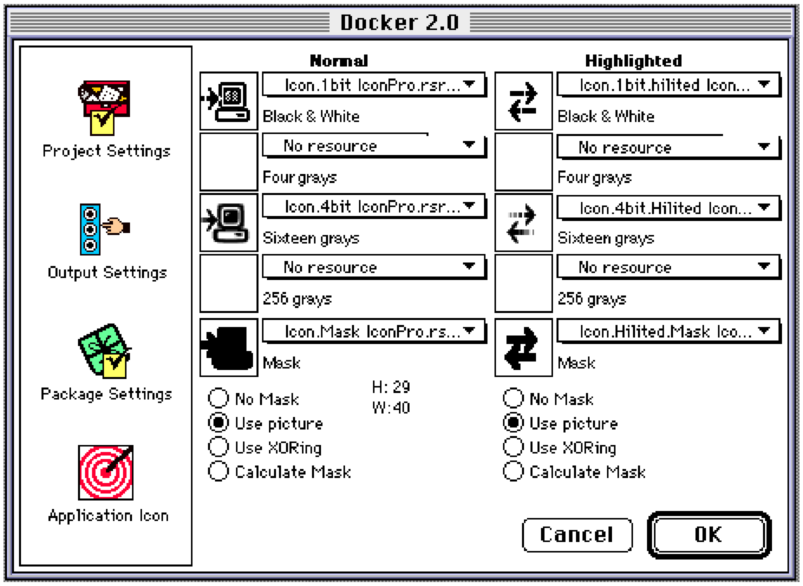
A number of options are provided for a picture’s mask:

No Mask	The bitmap does not contain a mask.
Use Picture	The mask is a PICT resource that is picked from the popup menu over the Mask window, just like any other image is selected.
Use XORing	A mask is generated such that when this mask is Xor’ed with the 1 bit image, the image selected in the mask field is displayed.
Calculate Mask	A mask is generated automatically from the black and white image. If no black and white image has been selected, one is created by NTK on the fly from one of the available images to generate the mask.

Application Icon Editor

The Application Icon pane of the Project Settings dialog allows you to select gray icons for your form part. There is a field for each of four screen resolutions, 1, 2, 4, and 8 bits, as well as a field for highlighted versions of the icon, and two fields for the normal and highlighted masks.

Figure A-2 NTK's Application Icon pane of the Project Setting dialog


Docker 2.0

Project Settings
Output Settings
Package Settings
Application Icon

Normal		Highlighted	
Black & White	Icon.1bit IconPro.rsr...	Black & White	Icon.1bit.hilited Icon...
Four grays	No resource	Four grays	No resource
Sixteen grays	Icon.4bit IconPro.rsr...	Sixteen grays	Icon.4bit.Hilited Icon...
256 grays	No resource	256 grays	No resource
Mask	Icon.Mask IconPro.rs...	Mask	Icon.Hilited.Mask Ico...

H: 29
W:40

Normal: No Mask [✓ Use picture] Use XORing Calculate Mask

Highlighted: No Mask [✓ Use picture] Use XORing Calculate Mask

A PICT is selected for each of these icons with the popup menu to the right of each of these fields. There are four choices available for the icon's mask:

No Mask	A mask is not used.
Use Picture	A black and white PICT is selected.
Use XORing	A mask is generated such that when this mask is Xor'ed with the 1 bit icon, the image selected in the mask field is displayed. The Extras Drawer in Newton 1.x and 2.0 OS Xor's an icon with its mask when the icon is selected.
Calculate Mask	A mask is generated automatically from the black and white image. If no black and white image has been selected, one is created by NTK on the fly from one of the available images, to generate the mask.

Functions

The following build-time functions are new in NTK version 1.6.4. Note that these functions are not available at run time.

GetSoundFrame

`GetSoundFrame(`*nameString*`)`

Retrieves a sound from an open Macintosh sound resource.

nameString	A string specifying the name of the sound resource to be retrieved.
return value	A sound frame containing a sound in whatever format is specified in the source sound resource. For details on the sound frame, see “Sound Frame” (page 1-29).

DISCUSSION

This function is similar to the older `GetSound` and `GetSound11` build-time functions. However, those functions require the sound to conform to a particular sampling rate, while `GetSoundFrame` is capable of loading sounds of any type. The Newton device, of course, will only play certain kinds of sounds.

MakeBinaryFromHex

`MakeBinaryFromHex(`*hexString*`, `*classSym*`)`

Returns a binary object of the specified class from the data in *hexString*.

hexString	A string consisting of an even number of hexdigits. Each set of two hexdigits makes for one byte of the binary object. This string is similar to the string returned by `StrHexDump`.
classSym	A symbol for the binary object’s class.
return value	A binary object of class *classSym*.

SEE ALSO

For example calls to this function, see “Black and White Patterns” (page 5-8), “Gray Patterns” (page 5-9), and “Dithered Patterns” (page 5-10).

MakeDitheredPatten

`MakeDitheredPattern(`*bwPattern*`,` *foregroundColor*`,` *backgroundColor*`)`

Creates a dithered pattern.

bwPattern	A one-bit pattern. A pattern is a binary object containing an 8x8 bitmap of class `'pattern`. The constants `vfWhite`, `vfLtGray`, `vfGray`, `vfDkGray`, and `vfBlack` specify patterns in the Newton OS ROM.
foregroundColor	A `kRGB_GrayXX` constant or a packed RGB integer returned by `PackRGB`.
backgroundColor	A `kRGB_GrayXX` constant or a packed RGB integer returned by `PackRGB`.
return value	A dithered pattern frame as defined in “Dithered Pattern” (page 5-26).

DISCUSSION

Using this function, as opposed to creating your own frame ensures that the frame shares a frame map with other dithered pattern frames.

SEE ALSO

For an example use of this function, see “Dithered Patterns” (page 5-11).

MakeExtrasIcons

`MakeExtrasIcons(`*iconRsrcSpecs*`,`*unhilitedMaskRsrcSpec*`,`*hilitedMaskRsrcSpec*`)`

Creates a frame with an `iconPro`, and optionally an `icon`, slot; these slots can be copied to a part frame.

iconRsrcSpecs An array of frames with the following format:

`unhilitedRsrcSpec`
String for the name of a PICT resource to use as the normal icon.

`hilitedRsrcSpec`
Optional. String for the name of a PICT resource for highlighted icon.

`bitDepth` Optional. Integer indicating resource's bit depth. The allowable values are 1, 2, 4, and 8.

If you do not specify a bit depth for a particular PICT, the bit depth is determined automatically from the PICT resource. If you want the icon to be included in your project at a particular bit depth, you should specify it explicitly.

Note

All the PICTs provided in this array must be of the same size. ◆

unhilitedMaskRsrcSpec
String for the name of a black and white PICT to be the mask for normal icon.

hilitedMaskRsrcSpec String for the name of the black and white PICT to be a mask for the highlighted icon, or `nil` if no highlighted icon is provided.

return value A frame with an `iconPro` slot, and if 1-bit information is provided in *iconRsrcSpecs*, an `icon` slot. These slots can be copied to a part frame.

DISCUSSION

If the *iconRsrcSpecs* array contains more than one icon, the system determines the appropriate one for the current hardware.

The resource names are for named PICT resources within any resource file included in the current project. If more than one PICT is used, then all the PICTs must have the same size bounds, or this function will throw. This includes all the PICTs referred to in the *iconRsrcSpecs*, *unhilitedMaskRsrcSpec*, and *hilitedMaskRsrcSpec* parameters.

SEE ALSO

The Project Settings dialog provides an editor to use for an application's part's icon; see "Application Icon Editor" (page A-3). You must use `MakeExtrasIcons` to create icons for other types of parts.

For an example of using this function, see Listing 5-1 (page 5-13).

MakePixFamily

`MakePixFamily(`*bwRsrcSpec*`,` *maskRsrcSpec*`,` *colorSpecs*`)`

Creates pix family from a set of PICTs.

bwRsrcSpec	String for the name of a black and white PICT resource to use in 2.0 and 1.x systems, or `nil` if backward compatibility is not desired.
maskRsrcSpec	String for the name of a black and white PICT resource to use as a mask or `nil` if there is no mask.
colorSpecs	A color spec or an array of color specs. A color spec is either a string for the PICT resource name or a frame with the following slots:

`rsrcSpec`	Required. A string for the PICT resource name.
`bitDepth`	Optional. An integer for the bit depth of the PICT. The following values are allowed: 1, 2, 4, and 8. If you do not specify a bit depth for a particular PICT, the bit depth is determined automatically from the PICT

	resource. If you want the image to be included in your project at a particular bit depth, you should specify it explicitly.
return value	A pix family frame, it can be passed to `CopyBits`, used in the icon slot of a `clPictureView`, or passed to `MakeShape` to create a bitmap shape.

DISCUSSION

If *colorSpecs* contains an array, the system displays the most appropriate image for the current hardware.

The resource names are for named PICT resources within any resource file included in the current project. If more than one PICT is used, then all the PICTs must have the same size bounds, or this function will throw. This includes all the PICTs referred to in the *bwRsrcSpec*, *maskRsrcSpec*, and *colorSpecs* parameters.

SEE ALSO

NTK's picture slot editor provides a simple way to create a pix family. See "Picture Slot Editor" (page A-1).

UnPackRGB

`UnPackRGB(`*packedRGB*`)`

Returns a frame with information about the red, green, and blue components of a packed RGB integer.

packedRGB	A packed RGB integer, as returned by the function `PackRGB`.
return value	A frame with `red`, `green`, and `blue` slots. Each slot contains an integer in the range [0, 65535] for that color component's value.

SPECIAL CONSIDERATIONS

`UnPack(PackRGB(r,g,b))` returns a frame `{red: redInt, green:greenInt, blue: blueInt}`. Note that `r` might not equal `redInt`, `g` might not equal `greenInt`, and `b` might not equal `blueInt`. It is only guaranteed that these values are similar, not identical.