Appendix C:
PXM File Format Specification

The PXM file format is designed to be a simple yet versatile raster file format. It is inspired by Jeff Poskanzer's popular PBM/PGM/PPM/PNM file formats.

A PXM file consists of a *header, palette data* (if the file is paletted) and *image data.*

paste_13.eps ¬
Paletted files

paste_14.eps ¬
Non-paletted files (Truecolor or grayscale)

Header

Unlike PNM files, PXM files are entirely binary and use Motorola (big-endian) byte ordering, which is the native byte ordering on most NeXT machines. The Intel 386/486 chips use opposite byte ordering (little-endian) from Motorola based machines. People wishing to read PXM files on i486 based NeXT's, PC's or other little-endian platforms will either have to swap bytes around as necessary or read

the header a byte at a time and shift accordingly.

Big-endian machines write the most significant byte (MSB) first and the least significant byte (LSB) last. Little-endian machines write the LSB first and the MSB last.

Writing the number 1234_{16} to disk:

paste_15.eps ¬
Big-endian

paste_16.eps ¬
Little-endian

The header consists of the following fields:

- 2 bytes for the ªmagic numberº

A magic number is one or more bytes at the start of a file which can be used to identify the format of the file. For example, the mach magic number which identifies NeXT binary executables is `0xfeedface`. The PXM magic number is a 'P' followed by a '+' or `0x502b`.

- 4 bytes for the width of the picture in pixels

- 4 bytes for the height of the picture in pixels

- 1 byte for the resolution of the picture in bits per color (max 8)

For example, a NeXTdimension system has 32 bits (8 bits each for red,

green, blue and alpha) so this byte would contain 8 for pictures produced by ªgrabbingº part of a NeXTdimension screen. A NeXTstation Color has 16 bits (4 bits each for the red, green, blue and alpha channels) so this byte would contain 4 for NeXTstation Color screen grabs. Monochrome NeXT's can show 4 shades of gray (2 bits) so this byte would be 2 for screen grabs on these systems.

For paletted files, this byte is based upon the resolution of the picture after it has been unpaletted for display, which is the same as the resolution of the palette. A standard IBM PC EGA display can show a picture with 16 colors simultaneously chosen from a palette of 262,144 possible colors (6 bits each for red, green and blue) so this byte would contain 6 for such pictures.

Note: Image and palette data are stored in such a way that this field is unnecessary for decoding. It is present for informational purposes only. See also the **Palette Data** and **Image Data** sections below.

- 1 byte for the version number (currently 1)
- 1 byte for the header size in bytes (currently 24)
- 2 bytes for the palette size in bytes (this is 0 if there is no palette)
- 1 byte for miscellaneous header flags:

`PXM_PALETTE`	`0x80`	file has a palette
`PXM_GRAY`	`0x40`	file is grayscale

PXM_ALPHA	0x20	file has an alpha channel
PXM_TOP	0x02	data is arranged top to bottom
PXM_RIGHT	0x01	data is arranged right to left

Example: A standard grayscale PXM file would not have a palette or an alpha channel, and its data would be arranged top to bottom, left to right. The byte to use for such files is 0x42.

Note 1: Currently, we do not support grayscale palettes, so flags PXM_PALETTE and PXM_GRAY are mutually exclusive.

Note 2: PXM files are normally produced with orientation top to bottom, left to right. Other orientations are produced only as intermediate files so PXM file readers/converters do not have to deal with this case. A PXM file with a non-standard orientation can be reoriented simply by opening and saving it with Pixel Magician.

· 4 bytes for the horizontal resolution in pixels per inch (2 bytes for the integer portion and 2 for the fraction, which should be rounded)

Example: Standard NeXT files are 72 pixels per inch. This field would contain 0x00480000.

Example: With a custom linescreen, the NeXT printer can print 400 pixels per inch. This field would contain 0x01900000.

Example: An image has 1 pixel per pica = 72 * 0.166 pixels per inch =

11.952 pixels per inch. This field would contain `0x000bf3b6`.

Note: This field is equivalent to a 4 byte unsigned quantity holding the integer part of (65536 * resolution + 0.5).

- 4 bytes for the vertical resolution in pixels per inch (2 bytes for the integer portion and 2 for the fraction, which should be rounded)

- header sizeÐ24 bytes for other information

Currently, no other information is used. Future versions of the PXM file format may use these bytes to hold information about how the file was created (*e.g.,* format from which the file was converted, compression originally employed, ¼). These fields will not be necessary for proper decoding of the PXM file so a PXM reader only needs to read the above fields and then can skip ahead to the next section.

Palette Data

Each entry in the palette contains 1 byte for the index number and 1 byte for each color in the palette. Alpha channel data is included in the image data, not in the palette, and grayscale palettes are not currently supported. Thus, palette entries currently are always 4 bytes in size. Palette entries are stored contig- uously (no padding).

Example: 4 color cyan, magenta, yellow and black palette

color 0: `0x0000ffff` cyan
color 1: `0x01ff00ff` magenta
color 2: `0x02ffff00` yellow
color 3: `0x03000000` black

Indices are allowed to be permuted for convenience in hashing algorithms.

Example: equivalent 4 color cyan, magenta, yellow and black palette

color 2: `0x02ffff00` yellow
color 1: `0x01ff00ff` magenta
color 0: `0x0000ffff` cyan
color 3: `0x03000000` black

Unused entries must be set not to conflict with legal values. Conflicting values cause undefined behavior.

Example: invalid 4 color palette with 4 unused entries

color 0: `0x0000ffff` cyan
color 1: `0x01ff00ff` magenta
color 2: `0x02ffff00` yellow
color 3: `0x03000000` black
color 0: `0x00000000` unused
color 0: `0x00000000` unused

color 0: `0x00000000` unused
color 8: `0x08000000` unused

It is not clear whether palette entry 0 is cyan `0x00ffff` or black `0x000000`. Also illegal is color 8 since the palette only allows 8 colors. The color index must be between 0 and the number of entries in the palette -1 (inclusive).

Example: valid 4 color palette with 4 unused entries

color 0: `0x0000ffff` cyan
color 1: `0x01ff00ff` magenta
color 2: `0x02ffff00` yellow
color 3: `0x03000000` black
color 3: `0x03000000` black
color 4: `0x04000000` unused
color 7: `0x07000000` unused
color 7: `0x07010203` unused

The duplicate entry for color 3 does not matter since it does not conflict. The conflicting values for color 7 do not matter since it is not used. Colors 5 and 6 are undefined but unused so, again, they do not matter. As can be seen, there are many possibilities for dealing with unused entries.

Unused entries can be generated when converting from a file format which only supports a limited choice in the number of palette entries.

Example: A 10 color paletted PCX file must use a 16 color palette format since PCX does not have a 10 color palette format. Six entries will be unused.

Palette entries are stored *as if 256 shades were available for each color component*. This increases the speed with which PXM files can be loaded and displayed on the NeXT, since no bit depth conversion needs to be done. Thus, the resolution field in the header does not affect the palette data.

Example: black and white image paletted image

color 0: `0x00000000` black
color 1: `0x01ffffff` white

not

color 1: `0x01010101` very dark grey

Image Data

If there is palette data, each byte of image data will contain an index into the palette. The data is not compressed in any way. If there is an alpha channel and a palette, each palette index byte will be followed by a byte for the coverage component for that pixel. This byte is stored *as if 256 shades were available for the alpha channel,* regardless of how many bits are actually available for the alpha

channel.

If the image has no palette, one byte is used for each color component (1 for gray, 2 for gray images with an alpha channel, 3 for RGB pictures and 4 for RGBA pictures). Again, no compression is used and the data is stored *as if 256 shades were available for each color component*. No padding is used. Thus, data can be loaded directly into an *NXBitmapImageRep* with no translation necessary.

Example: 4 shade gray image with alpha

If we distribute 4 shades evenly between 0 and 255, we find that the shades are 0, 85, 170, and 255 (or `0x00`, `0x55`, `0xaa`, and `0xff`). The image data will only consist of the values `0x00`, `0x55`, `0xaa`, and `0xff`. The resolution field in the header should contain the value 2. Two bytes are used for each pixel. The image data has the format GAGAGAGA¼.

Example: 16 shade RGB image

If we distribute 16 shades evenly between 0 and 255, we find that the shades are `0x00`, `0x11`, `0x22`, `0x33`, ¼ `0xff`. The image data will only consist of the values `0x00`, `0x11`, `0x22`, ¼ `0xff`. The resolution field in the header should contain the value 4. Three bytes are used for each pixel. The image data has the format RGBRGBRGB¼.

Example: 5 bit palette with alpha

If we distribute 32 shades evenly between 0 and 255, we find that the shades are `0x00`, `0x08`, `0x10`, `0x19`, `0x21`, `0x29`, `0x31`, `0x3a`, `0x42`, `0x4a`, `0x52`, `0x5a`, `0x63`, `0x6b`, `0x73`, `0x7b`, `0x84`, `0x8c`, `0x94`, `0x9c`, `0xa5`, `0xad`, `0xb5`, `0xbd`, `0xc5`, `0xce`, `0xd6`, `0xde`, `0xe6`, `0xef`, `0xf7`, and `0xff`. The palette index byte must contain a value between 0 and 31, inclusive. The palette entries can only contain the above listed values for the color components. Similarly, the alpha channel byte can only contain the above listed values. The resolution field in the header should contain the value 5. Two bytes are used for each pixel. The image data has the format PAPAPA¼.