XPM
The X PixMap Format

Arnaud Le Hors & Colas Nahaboo
lehors@sophia.inria.fr
colas@sophia.inria.fr

BULL Research FRANCE – Sophia Antipolis

November 1991

1 Introduction: Why another image format?

As many X programmers, when we began to try to include color images in our X programs, we were faced with a plethora of image formats to choose from. We thought that each of these formats had been designed to handle big (typically more than 600x400 pixels), colorful (with at least 8 bit colormaps) images. However, we soon found that we had different needs, and thus these image formats were somewhat inadequate. We thus designed the XPM (X PixMap) icon format, and refined it thanks to the international X community. We feel that it is mature enough now to be proposed to the X consortium to be included in the standard MIT tape. This paper describes this third version of the XPM format.

2 Why is there no standard image format?

At first glance, one may think that a single image format could encompass all the needs of computer graphics professionals. The problem is that in most cases, images must be handled very efficiently, and thus the image format chosen often closely reflected the underlying hardware, as is the case for most formats coming from the PC world (e.g. GIF, IFF's ILBM, Mac resources). In the academic world, there is nearly one format per research team. The situation has stabilized, the formats can be grouped into three classes:

1. the working format: the one that is most adapted to the application or graphics hardware and software. In our X world, this can be thought of as the XWD format (X Window Dump, which is derived from the memory structure XImage). Rasterfiles on Sun workstations and GIF files on PCs are example of formats reflecting the underlying hardware.

2. the storage format: one that is often chosen for its built-in picture compression facility. Gif (and perhaps Tiff in the future) seems to play this role nowadays.

3. the interchange format: the one chosen to ease the writing of conversion programs between this format and any other. One good example of this pivotal format is Jeff Posanker's **pmbplus** package.

3 The characteristics of icons

What we really need is an **icon** format rather than an **image** format. In a typical application one often want a lot a small images, which can be either elements of the decoration (for a 3D look), buttons to press on, or "road signs" to convey information. These images have different characteristics such as being small, numerous, stylized, and customizable.

Also an icon is composed of several objects associated with single colors. If you consider for instance a 3D button you may define five objects: front, top, bottom, left, and right faces (figure 1.1). We need to be able to specify the way the colors associated with the different objects composing the icon are changed depending on the type of visual it is displayed on. For instance, on a color visual we may have a 3D button with white left and top faces, a gray front face, and black bottom and right faces. Displayed with the same colors on a black & white visual it would be: white left and top faces, a white front face, and black bottom and right faces (figure 1.2). This is not satisfactory because we would rather have: a white front face, and black side faces (figure 1.3).

We also need to be able to override built-in colors in order to let the user change them. For instance if we define an application interface with 3D buttons, we need to let the user redefine the colors used to keep the 3D look if the application is customised to be within green or blue shades.

In addition we want to have transparent colors to be able to define non-rectangular icons. One application of this is to define cursors. Indeed so far cursors are two colors only and are defined with two bitmaps: the source specifying the foreground and the background colors, and the mask specifying the shape (these are stored in two different files). This could be handled by a single icon in which the outer pixels are defined as transparent. Furthermore having cursors defined with such icons would provide support for future multicolor cursors.

With cursors comes the need to have hotspot information associated with the icon, but this is not the only case for which such an information can be useful. In particular, we may want to make animation sequences by putting several icons, one after the other, at the same place. To do so we need to have a hotspot related to the icons to know how to place them. For example to animate a warning sign by growing and reducing it, one needs to have a hotspot to easily center the icons whatever their size.