VBXtasy, Vol. 1 -- Tech Note #1

This tech note describes how to use 256-color images with VBXtasy on a 256-color display adapter.

The problem

When you place any control that displays a 256-color image (such as a Picture box) on a form containing VBXtasy controls, the colors on those VBXtasy controls (including the background) become garbled on 256-color displays.

Background

When Windows runs on a monitor that is set to display 256 colors, all applications must share the system “palette ” — also called the color look up table (CLUT). The only colors that can be displayed at any one time are the 256 colors present in the current system palette (sometimes called the "hardware palette," because it defines what colors the monitor and adapter can display). For most Windows applications, this limitation is of no concern; they use only the twenty "static" colors that Windows automatically places in the system palette, so they're always guaranteed to have the colors they need. Applications that want to display more than those few predefined colors, however, must reserve the colors they need by participating in a negotiation process that is governed by the Windows palette manager. Simply put, Windows asks every application that desires access to the system palette to reserve the colors it needs (such applications are said to be "palette-aware"). This process starts with the active application (the one that's on top) and then steps through all other palette-aware applications in z-order, top to bottom.

In Visual Basic applications, individual controls can be palette-aware. When a Visual Basic application is asked to reserve the colors it needs in the system palette, it in turn queries all of the palette-aware controls on the currently active form in z-order, top to bottom, telling them to reserve the colors they need.

When VBXtasy controls are present on a 256-color display, they claim all 256 colors in order to enhance both their appearance and their drawing performance. Indeed, for performance reasons their display behaviors are based on the assumption that they have full control of the entire system palette on 256-color displays. If, however, you place a palette-aware, non-VBXtasy control in front of all VBXtasy controls on a form, that control may claim some or all of the colors on the system palette before any VBXtasy controls have the chance to stake their

claim. The result will be a serious distortion of the way the VBXtasy controls display.

Any solution to this problem must respect VBXtasy's need to control the system palette on 256-color displays. Fortunately, there are solutions which allow for the display of high-fidelity images, although not at the level of quality that would be possible in the absence of VBXtasy.

Each of VBXtasy's design sets has a single palette used by all of the design set's controls, including the Background control. We have made an effort to incorporate in all seven palettes a set of "rainbow colors" that will allow most high-fidelity images to be dithered gracefully. Dithering is a technique whereby a color that the hardware can't display is simulated by mixing adjacent pixels with available colors that approximate it. Gray is simulated on black and white displays, for example, by dithering together equal numbers of black and white pixels. As a result, using the techniques below will allow you to display 256-color images in the presence of VBXtasy controls in a way that will preserve the overall integrity of the image, although it may look more "dotty" or "rough" than it would look if you were to display the image without any VBXtasy controls present.

Solution #1 - Controlling the z-order

This technique assures that VBXtasy controls will be in a position to retain control of the system palette. Since Visual Basic bestows this power based on z-order, you just have to make sure that one or more VBXtasy controls lie in front of the control displaying the 256-color image in order for those controls to establish their authority over the system palette. When laying out your form, here's how you would accomplish this:

- Place the image-displaying control on the form.
- Place one or more of the VBXtasy controls you intend to use on the form.
- Optionally, you may now load the bitmap image you intend to use in the image-displaying control.

If you have already composed your form and you don't want to go through the trouble of laying it out again from scratch, then simply select the image-displaying control and execute the "Send to back" command in the "Edit" menu. If the display doesn't properly update right away, save your project, leave Visual Basic, and then re-run Visual Basic and open your project back up again.

Solution #2 - Using the design set palette

If, at design time, you already know what 256-color image you are going to need to display, you can change the image's own palette to match the palette for the design set you are using. You can obtain better results this way because many paint programs, such as JASC's excellent Paint Shop Pro (which is available as shareware), can load a new palette into an existing bitmap and do a better job of dithering the resulting colors than Windows normally does.

Included with this tech note are seven bitmap files, each containing a one-pixel-by-one-pixel image whose palette is the same as one of the design sets in VBXtasy, Volume 1. Their names correspond to the names of the VBX files for those design sets (e.g., XTCALUM.BMP, XTCMONDO.BMP, etc.). Any image manipulation program capable of loading a new palette for an existing image should also be able to extract the palette from an existing image. Simply extract the palette from the bitmap that corresponds to the design set you're using, then apply that palette to the image you want to display.

You will have to refer to the documentation for whatever image manipulation program you're using in order to find out how to load a new palette for an existing 256-color image. Once you've done so, you can load the resulting image into any image-displaying Visual Basic control and not worry about that control's z-order relationship with any VBXtasy controls that are present on the same form.