Haxial AppearanceEdit 1.200 Documentation

Haxial Software
http://www.haxial.com/

Description

Haxial recognizes that people have widely varying tastes, and what is delicious to one person is disgusting to another person, and what is beautiful to one person is hideous to another person. This is why Haxial wants to let you choose and customize the appearance of Haxial programs. Users should not be forced into one appearance.

Haxial programs render/display their graphical user interface (GUI) using the Haxial Appearance Engine, which supports a high degree of customization. In other words, if you are not happy with the appearance of Haxial programs, you can change it to suit yourself.

The easiest way to change the appearance is to download a Haxial Appearance file (a ".hap" file) from the Haxial website or another source, and load it into the Haxial program of your choice. The Appearance Engine then renders the GUI using this .hap file, and the appearance is completely changed.

Alternatively, if you are more adventurous, or you are the artistic type, then you might like to use Haxial AppearanceEdit to create your own appearance file, or modify an existing one. So in summary, AppearanceEdit is a program for creating and editing ".hap" files which change the appearance of Haxial programs.

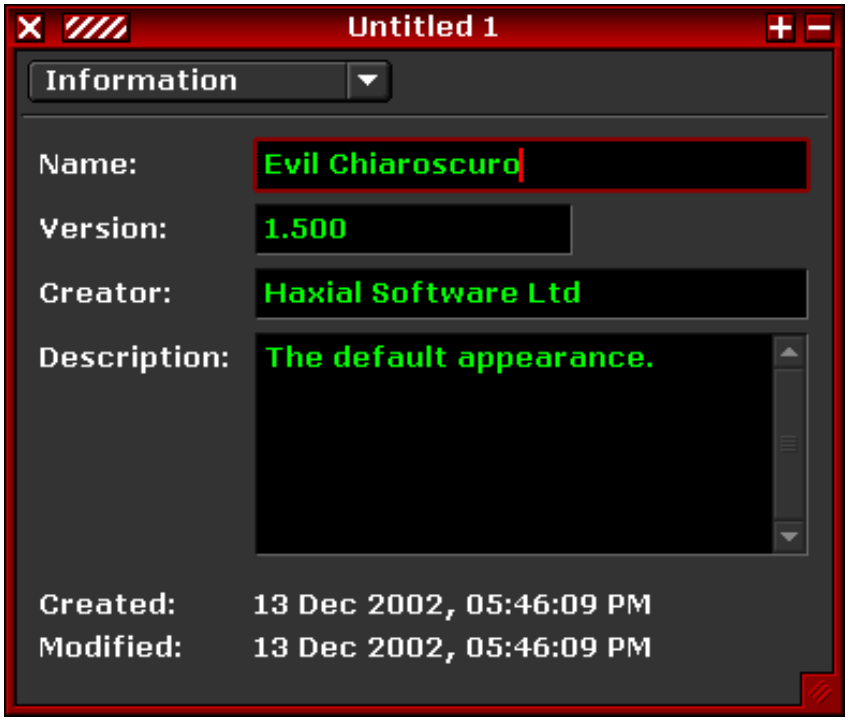
For examples of different appearances, please see:
http://www.haxial.com/appearance/

System Requirements

- MS Windows 95 or better, or
- MacOS 9 with CarbonLib 1.3.1 or better, or
- MacOS X (10) or better.

Step 1: Information Panel

Open AppearanceEdit, and it creates a new untitled window. It looks like this:

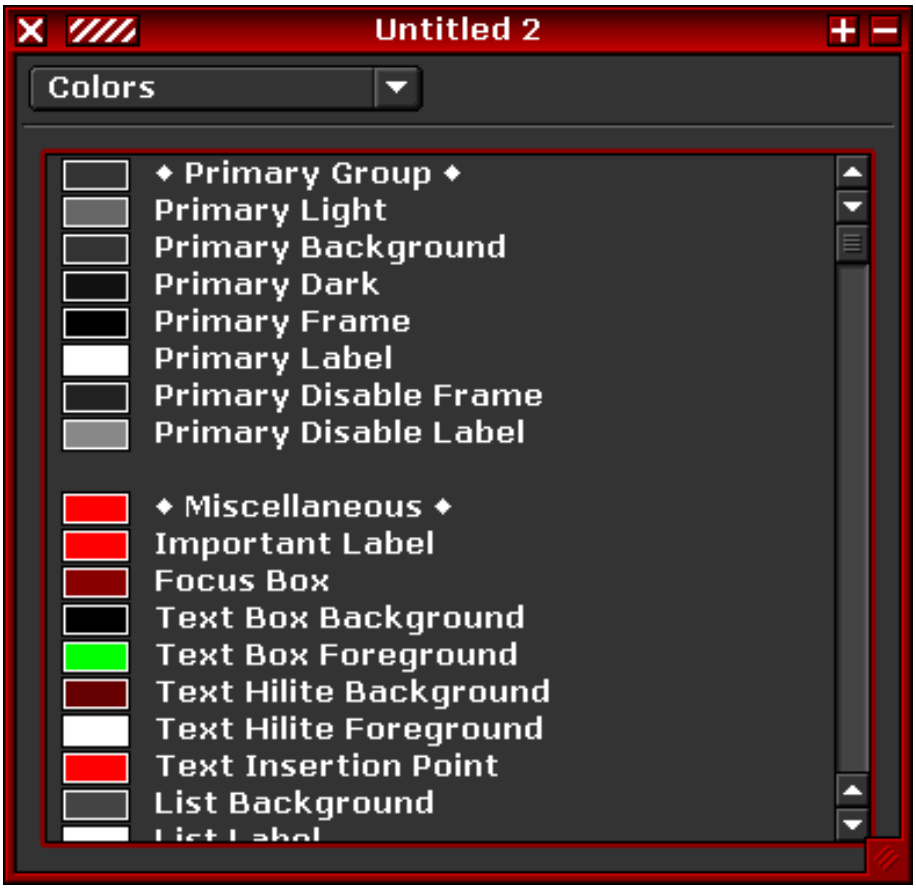


The "Information" panel is selected by default. In this panel, you can enter information about your new appearance, such as its name, its version number, who created it (you), and a description or comments.

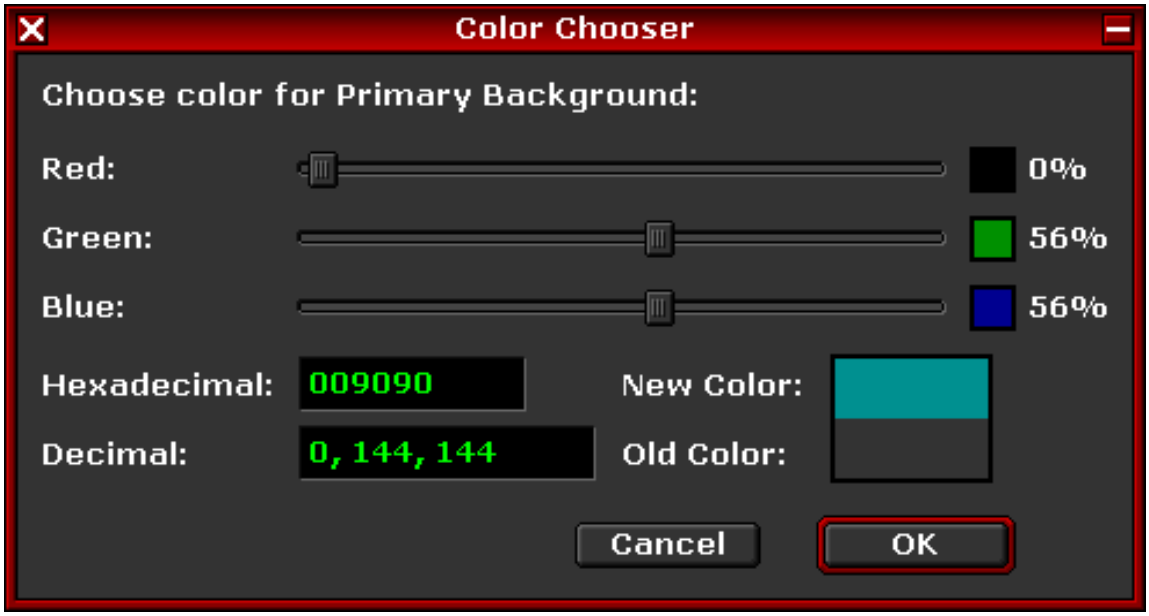
When you are done, click the "Information" button to switch to the next panel.

Step 2: Colors Panel

The "Colors" panel looks like this:



The purpose of the Colors panel is to change all the colors that the user interface is drawn in. You can see that there is a big list of named colors. The name describes the purpose of that color or where it is used. For example, "Primary Background" controls what color the background of most windows is drawn in. If you click on it, a window appears to allow you to change the color:



You can drag the sliders to change the proportions of red/green/blue (the 3 primary colors of the additive triangle of light). For example, if you set all 3 to 100%, then the result is white. If you set all 3 to 0%, the result is black. If you set all 3 to the same percentage, the result is a shade of gray. If you set red and green to 100% but blue to 0%, the result is bright yellow.

More advanced users can also edit the decimal representation of the color, which consists of 3 numbers (red, green, and blue) separated by commas. Each number can be in the range 0 to 255. So "255, 128, 0" means 100% red, 50% green, and 0% blue.

The hexadecimal representation of the color is similar to decimal, but there are no commas to separate the 3 primary colors. The first 2 digits are for red, the next 2 digits are for green, and the last 2 digits are for blue (RRGGBB). Each number can be in the range 00 to FF (FF is the hexadecimal representation of 255). So "FF8000" means 100% red, 50% green, and 0% blue. The hex number is useful for copying & pasting colors, and it is also the format used in web page design.

Returning back to the list of colors in the Colors panel, you can see that some colors are named with a diamond symbol (◆) either side of the name. This indicates a shortcut for setting multiple related colors simultaneously. For example, if you click "Primary Group", you can choose a single color, and it sets the colors for "Primary Light", "Primary Background", "Primary Dark", and "Primary Frame" by lightening or darkening the "base" color you selected.

Try changing the "Primary Group" color, and then click the Window Menu button (), then click "Preview". A window appears showing various samples of user interface items/widgets, displayed using the colors that you selected in the Colors panel. Try changing some more colors, and click Preview again.

Opening other appearance files in AppearanceEdit and looking at the colors is a good way to see how it works.

Do not worry, AppearanceEdit allows much more customization than only the colors, but for now, choose the colors you want, and then you can progress to more advanced customizations.

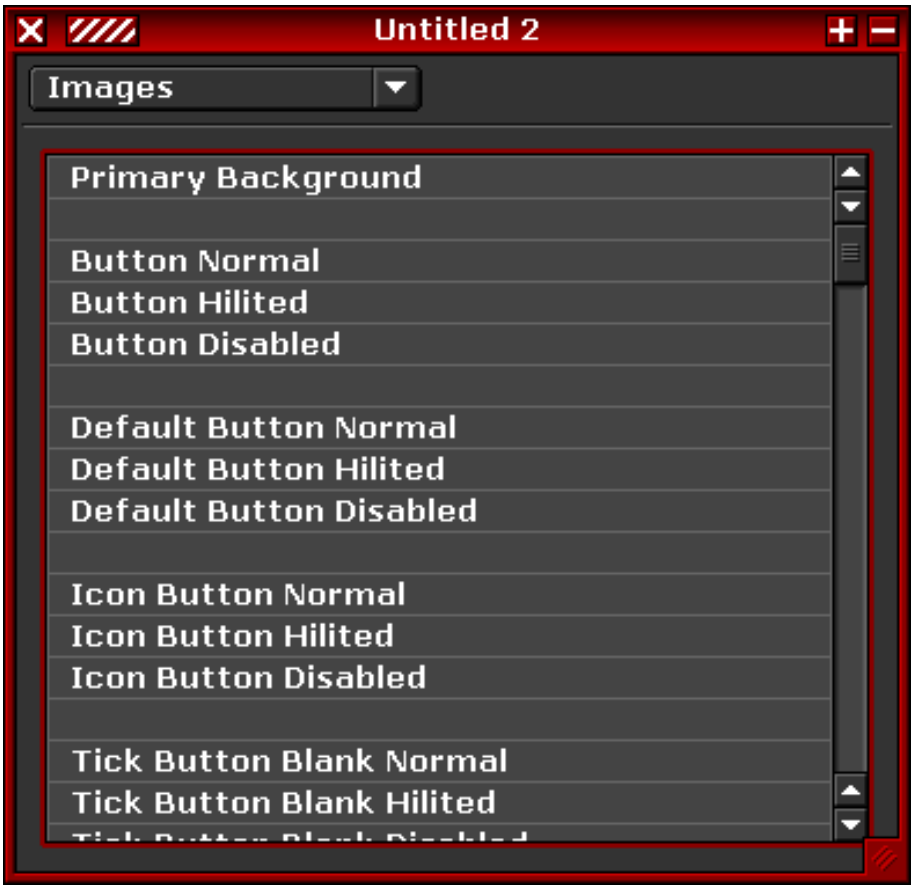
Preview: When you use the Preview command when the Colors panel is being displayed, only the colors are previewed (images and icons are not displayed). This allows you to see all the colors because when images are used, the images

may be overriding colors. If you wish to preview everything, switch to one of the other panels before clicking the Preview command. Alternatively, hold down the alt/option key while clicking the Preview command, and then it will preview everything regardless of whether or not the Colors panel is displayed.

Import Colors: In the Window Menu, you will find an "Import Colors" command. This allows you to grab the colors from a different appearance file, and load them into the current appearance file.

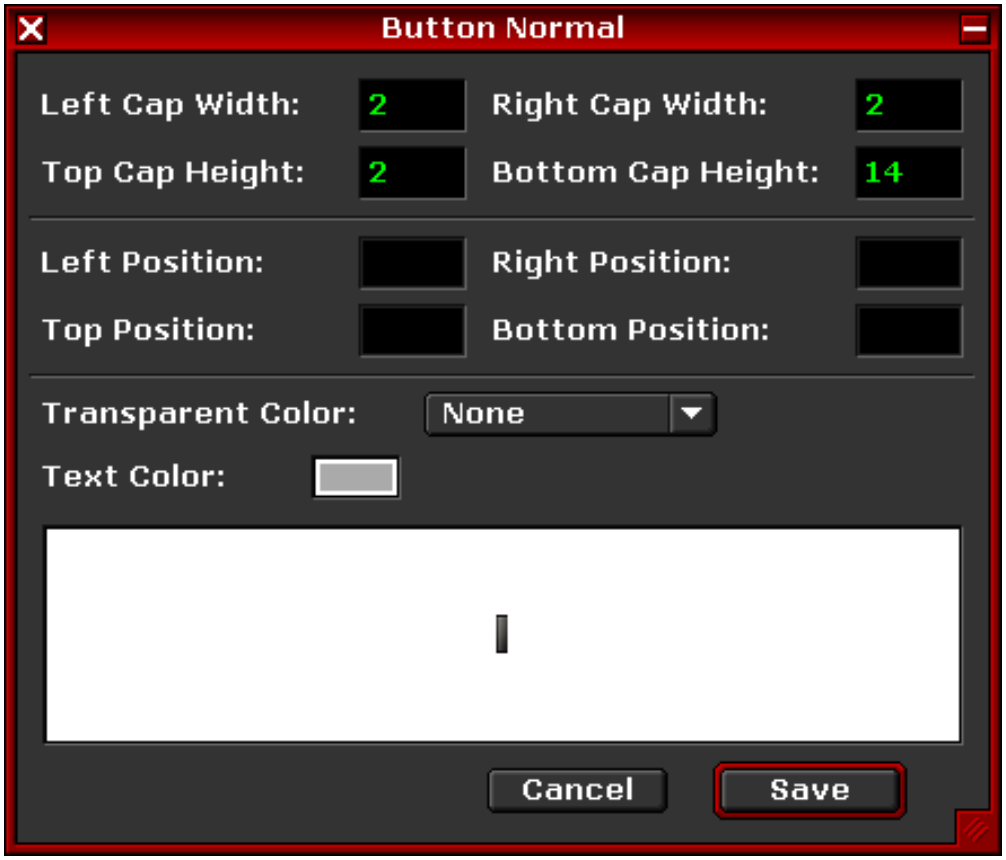
Step 3: Images Panel

The "Images" panel looks like this:



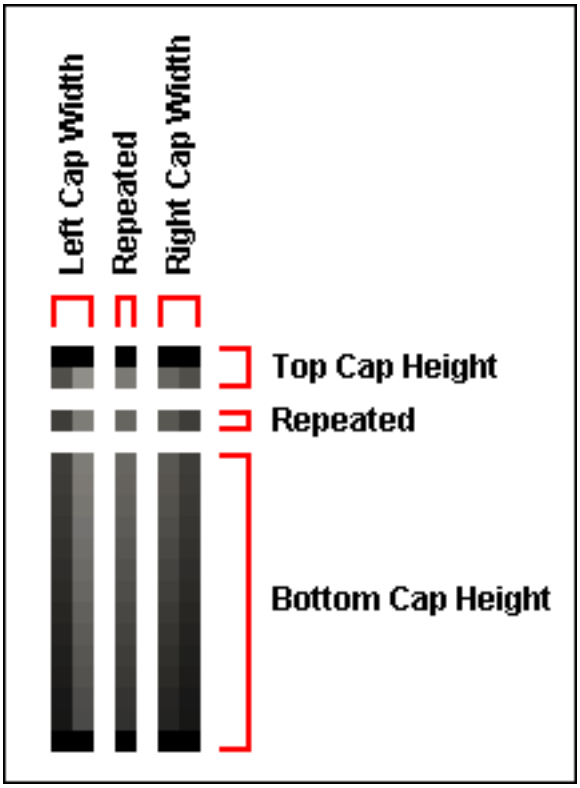
The names in the list represent images that you can supply. Initially they are all empty, but if you supply an image, the Haxial Appearance Engine will render that user interface item using your image. For example, if you supply an image for "Primary Background", then the background of most windows will be drawn using that image (as a repeating pattern). If you want to change the

appearance of buttons, then you would supply an image for “Button Normal” and the other Button images. When you click on an item in the list, a window for editing that item appears:



Now for a lesson about how these images work. Firstly, look at different buttons in a program. You can see that buttons come in all different sizes (mostly the width changes, but sometimes the height too). This poses a problem because, for example, say you drew a spunky looking image of a button that was 100 pixels wide and copied & pasted it into AppearanceEdit. Now it comes time for the Appearance Engine to display that image where a button is supposed to be. But the program is requesting a button that is 150 pixels wide! And then somewhere else it is requesting a button that is 85 pixels wide. How is the Appearance Engine supposed to draw buttons of different sizes using your single image of a button?

The solution is that the Appearance Engine supports “stretching” an image by repeating certain parts of the image. So you draw a very small image of a button, and then the Appearance Engine repeats part of it to stretch it to the desired size. Exactly how the stretching/repeating is performed is controlled by “Cap” parameters that you supply, and you can see in the window above that there are 4 “Cap” text boxes. Explaining these is best done with a diagram:

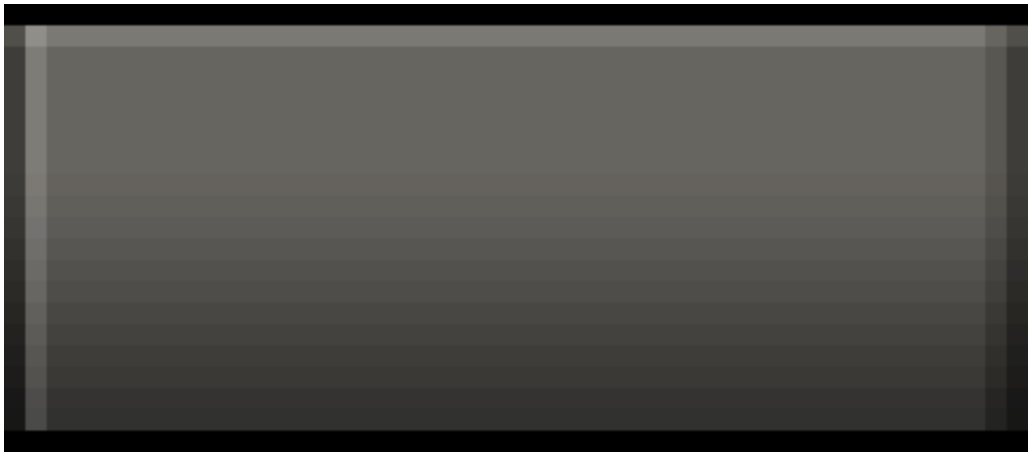


That is a button image enlarged to 800%, and separated into the components. As you can see, there are 9 components. The size of these components is controlled by you by firstly drawing your image at the appropriate size, and then entering the correct "Cap" values (in pixels) into the Cap text boxes in the window. (By the way, the maximum Cap size is 255 pixels.)

In other words, the 4 corners of your image are copied to the 4 corners of the button like this:

And then the space around the outside between the corners is filled by repeating the corresponding portions of the source image:

Then the remaining part in the middle is filled using that middle dot in the source image:



And finally the button title is drawn on top using the color specified by “Text Color” in the window. At 100%, the button looks like this:

The above example is for a button that can be stretched both horizontally and vertically. Some user interface items need only be stretched in one direction and thus the caps for the other direction are disabled. Otherwise they work in much the same way. For example, a Progress Bar is stretched only

horizontally, and so the Left Cap portion is drawn on the left side of the Progress Bar, and the Right Cap portion is drawn on the right side, and the middle is filled by repeating the middle portion of the source image.

Below the Cap text boxes, you can see "Position" text boxes. The exact meaning of these depends on the particular image you are editing. For example, for the "Window Close Button" images, the position text boxes are used to control where in the window title bar the close button is to be drawn. If the position text boxes are disabled, then they are not relevant for the current image. Position values can be in the range 0 to 255.

The big white area in the window is where you copy & paste your image. You can second-click (or control-click) on it to show a Context Menu, or you can click in it to select it, and then use the hotkeys, for example control-V to paste (command if you are using a Mac).

Note that opening a different .hap file and looking at the images is a good way to learn how they work.

The Transparent Color

The "Transparent Color" menu allows you to select a color which will become transparent. This allows you to have non-rectangular images or images with "holes". For example, if you select "100% Green", then wherever 100% green appears in your image, whatever is below will show through as if the green were non-existent. If you have ever used transparent GIFs, or "green screens" in movies, this is the same idea -- the green in your image is replaced by the background.

If you select 100% green but it does not work, check that your image is using EXACTLY 100% green with 0% red and 0% blue (hex 00FF00). For example, 99% green will show as green, not transparent. You must use exactly the color specified.

The Transparent Color cannot be used to make a non-rectangular Window Frame. Due to problems with cross-platform portability, non-rectangular Window Frames are not supported at this time.

Maximum of 256 Colors

Note that AppearanceEdit images are limited to 256 different colors. Don't complain, 256 colors is more than enough provided that you (or your graphics program) use them properly, and especially considering how small most of the images in AppearanceEdit are. When done properly, it is very difficult if not impossible for a human to pick the difference between 256 colors and full color. AppearanceEdit will reduce an image to 256 colors when you save it, but you may (or may not) get better results by reducing it to 256 colors in your image editing program.

For example, if you are using Photoshop, you can change the image mode to "Indexed Color" (Image -> Mode -> Indexed Color). Only do this as the FINAL step just before copying your image into AppearanceEdit. In other words, you should do all your editing in "RGB Color" mode, then as the final step, you can reduce it to Indexed Color.

If you would like to optimize your image to make it display faster and use less memory, then in the Indexed Color window in Photoshop, you can select "Adaptive" palette (or "Perceptual" etc), and reduce the number of colors. The aim is to make the number of colors as low as possible without making the image look bad. You do not have to take this to an extreme because saving a couple or a few colors is not going to make any difference, but for example, if the image looks equally as good with 50 colors as with 200 colors, then you made a saving of 150 colors.

Note: You can put all of your images into a single Photoshop document, and then change the whole thing to Indexed Color mode, and this might work well, or it might look bad. If it looks bad, then you need to used Indexed Color on each individual image separately (in separate documents).

It might help if you understand what Indexed Color actually does. Firstly, a "full color" image is one that can display practically any color anywhere, out of a palette of approximately 16.7 million colors. However, most images do not need this many colors, and thus they are wasting memory and processing power. Indexed Color is a way of limiting an image to only the colors that it actually needs, up to a maximum of 256 different colors. In Indexed Color, you still have that palette of 16.7 million colors available to you, except that the number of DIFFERENT colors that the image ACTUALLY USES is limited to 256, which is sufficient for practically any image that AppearanceEdit needs.

Now, if your image already uses less than (or equal to) 256 different colors, then it can be exactly copied & pasted into AppearanceEdit with no changes. If

it uses more than 256, then AppearanceEdit will modify the image to make it use no more than 256 different colors. You probably will not even notice when this happens, but if you do notice and it bothers you, then use the Adaptive Indexed Color mode in Photoshop.

Generally speaking, smaller images convert to Indexed Color better than large images because often smaller images contain a lesser number of different colors than bigger images. This is why you may see better image quality by converting your user interface widgets to Indexed Color separately in separate documents (each widget gets its own set of 256 colors), rather than all at once in the same document (which would limit your entire appearance to 256 colors!).

User Interface States

You may have noticed that, for example, the list of images contains "Button Normal", "Button Hilited", and "Button Disabled". These are called "states". What is the difference between them?

Normal: This is how the button looks normally, when it is NOT being clicked.

Hilited: This is how the button looks when the user is clicking on the button (the hilited image is displayed while the mouse button is being held down).

Disabled: The disabled image is displayed when the button is disabled, meaning that if the user clicks on it, it will do nothing.

Focus: This is when the user interface item has the "keyboard focus", meaning that if the user types on the keyboard, it will be directed into this item.

Each Image Explained

Primary Background:
Controls the background of most windows. You can paste a pattern into this to have a patterned background. If you want a solid color, do not use this, instead use Primary Background in the Colors panel.

Button Normal/Hilited/Disabled:
For the regular type of buttons -- the most common type of button that you see.

These are usually 20 pixels high, but the caps allow them to be drawn at other sizes.

Default Button Normal/Hilited/Disabled:
The same as the regular type of button, except with a slightly different appearance to indicate that this button is the default button in the window -- the button that is "clicked" if the user presses the return/enter key, such as an "OK" or "Save" button. Haxial programs create Default buttons 3 pixels bigger on all 4 sides than regular buttons. This allows your appearance to make Default buttons which have a border around them. If you do not want any border, you can put 3 pixels of transparent color on all 4 sides.

Icon Button Normal/Hilited/Disabled:
Buttons which display an icon on/inside the button, and optionally the title.

Tick Button Blank/Ticked/Tristated Normal/Hilited/Disabled:
A user clicks a Tick Button to place a tick in it, indicating that the option is now on. Some Tick Buttons can also display a third state (in addition to blank and ticked), known as tristate. Tristate is usually used to indicate "no change", or when displaying the status of multiple items some of which are ticked and others not. The title of a Tick Button is drawn outside of the button. The height of the button cannot exceed 18 pixels.

Mutex Button Blank/Ticked/Tristated Normal/Hilited/Disabled:
Mutex Buttons are like Tick Buttons, except that Mutex Buttons are usually displayed in a group, and only 1 of the buttons in the group may be ticked at any one time -- i.e. they are mutually exclusive, or mutex for short.

Small/Medium Plus/Minus Button Normal:
Plus/Minus Buttons are used for disclosure purposes, for example in a hierarchical list. Click the plus '+' button to show more information, and then it changes into a minus '-' button which you can click to go back to how it was before. The medium size is 16x16 pixels. The small size is 12x12 pixels.

Popup Button Normal/Hilited/Disabled:
A Popup Button displays a popup window or popup menu when clicked. The user can select a menu item, which is then displayed in the button as the users choice (or less frequently, the title inside the button is fixed, and the menu items are commands). These are usually 20 pixels high, but the caps allow them to be drawn at other sizes.

Popup Button No Title Normal/Hilited/Disabled:
The same as Popup Buttons, except that no title is displayed inside the button.

These are usually used next to a text box. The user can type their own value/text into the text box, or they can choose from a menu. These are usually 20x20 pixels in size, but the caps allow them to be drawn at other sizes. The "Popup Button Symbol" image is drawn centered over this button.

Popup Button Symbol Normal/Hilited/Disabled:
This symbol is drawn over the top of the Popup Button. It is usually a downward-pointing arrow indicating that a popup window/menu will be displayed below the button when clicked. The symbol is also drawn centered over "No Title" popup button, and in this case the Position text boxes are ignored. When drawn over regular Popup Buttons:

Left Position:	Horizontally, the symbol is drawn this many pixels from the left side of the button. Or if 0, use Right Position.
Right Position:	Horizontally, the symbol is drawn this many pixels from the right side of the button.
Top Position:	Vertically, the symbol is drawn this many pixels from the top side of the button. Or if 0, use Bottom Position.
Bottom Position:	Vertically, the symbol is drawn this many pixels from the bottom side of the button. Or if 0, vertically center the symbol.

Focus Box Normal/Hilited/Disabled:
A Focus Box is a type of box that is drawn around text boxes and lists. It also indicates to the user where their typing will go (which text box has the keyboard focus). The box should be 3 pixels thick on all sides. The middle/inside of the box should be transparent (use the transparent color).

Horiz Separator Line:
A horizontal line drawn between unrelated items to visually separate them (Horizontal is like the horizon, but straight :). The height cannot exceed 4 pixels. The line is drawn vertically centered.

Vert Separator Line:
A vertical line drawn between unrelated items to visually separate them (vertical is perpendicular to horizontal). The width cannot exceed 4 pixels. The line is drawn horizontally centered.

Box:
A box drawn around a group of items to visually group them and separate them from other unrelated items.

Framed Raised Box:
Usually drawn as a sort of "placard" at the top or bottom of a window that consists of a scrolling area or list which consumes the whole window, except for the "placard" area which may have buttons above it. A scrolling area of this type has no focus box because it goes all the way to the edges of the window, and thus the window itself becomes the indicator of focus.

Progress Bar:
A Progress Bar is used to visually indicate how much of a task has been completed. For example, how much of a file has been downloaded or copied. The "Progress Bar" image is drawn first, and this image should be an "empty" progress bar, i.e. as if the task were 0% complete (or the "background" of the progress bar). Then the "Progress Bar Fill" image is drawn above to a width that reflects the percentage complete. The progress bar is drawn vertically centered. The height of the progress bar cannot exceed 16 pixels.

Progress Bar Fill:
Drawn above the "Progress Bar" image to a width that reflects the percentage complete.

Left Position:	Horizontally, the Fill begins this many pixels from the left side of the progress bar.
Right Position:	Horizontally, the Fill ends this many pixels from the right side of the progress bar. (If the progress bar were 100%, the fill would stop this many pixels before the right side.)
Top Position:	Vertically, the Fill is drawn this many pixels from the top side of the progress bar.
Bottom Position:	Vertically, the Fill cannot extend below this many pixels from the bottom side of the progress bar.

H Slider Bar Normal/Hilited/Disabled:
A horizontal slider consists of an indicator that the user can drag along a horizontal bar or "track" to visually select a value within a certain range, for example the volume of sound. This image is the bar/track, and it is drawn vertically centered. The total height of the slider (including indicator) cannot exceed 30 pixels.

Left Position:	The indicator can be dragged no further left than this many pixels from the left side of the slider.
Right Position:	The indicator can be dragged no further right than this many pixels from the right side of the slider.

H Slider Indicator Normal/Hilited/Disabled:
The horizontal indicator that the user drags along the bar/track.

Top Position: Vertically, the top of the indicator is drawn this many pixels ABOVE the top of the bar.

H Slider Pointed Indicator Normal/Hilited/Disabled:
Same as the regular indicator, but this indicator points downwards. It is intended to be used with a scale of some sort that is drawn below the slider, so the indicator points to a number that indicates the current setting.

V Slider Bar Normal/Hilited/Disabled:
A vertical slider consists of an indicator that the user can drag along a vertical bar or "track" to visually select a value within a certain range, for example the volume of sound. This image is the bar/track, and it is drawn horizontally centered. The total width of the slider (including indicator) cannot exceed 30 pixels.

Top Position: The indicator can be dragged no further upwards than this many pixels from the top side of the slider.

Bottom Position: The indicator can be dragged no further downwards than this many pixels from the bottom side of the slider.

V Slider Indicator Normal/Hilited/Disabled:
The vertical indicator that the user drags along the bar/track.

Left Position: Vertically, the left of the indicator is drawn this many pixels left of the left of the bar.

V Slider Pointed Indicator Normal/Hilited/Disabled:
Same as the regular indicator, but this indicator points to the right. It is intended to be used with a scale of some sort that is drawn to the right of the slider, so the indicator points to a number that indicates the current setting.

Column Header Normal/Hilited/Disabled:
In a scrolling list with columns, the Column Header is drawn at the top of the column and its title explains to the user the meaning of the values in that column in the list. A Column Header can normally be clicked to sort the list by that column.

H Scroll Bar Double Arrows:
A horizontal scroll bar, used for scrolling left and right in an area that is bigger than the portion displayed. This is an image of the scroll bar with double arrows at each end (at each end of the scroll bar, there is an arrow button for scrolling left and right). The height of the scroll bar must be exactly 16 pixels.

Left Position: The indicator can be dragged no further left than this many pixels from the left side of the scroll bar.

Right Position: The indicator can be dragged no further right than this many pixels from the right side of the scroll bar.

H Scroll Bar Single Arrows:
Same as Double Arrows, but with only 1 arrow at each end.

H Scroll Bar Disabled:
The scroll bar is disabled, the user cannot use it for scrolling. (Probably because the scrollable area is smaller than the area for displaying it.) There is not a separate disabled image for double and single arrows, the same image is displayed in either case (it does not matter because it is disabled anyway).

H Scroll Bar Too Small:
If the scroll bar is in a window that is resized to cause the scroll bar to be very small, too small to display the Single Arrows image (according to the caps of the Single Arrows image), then the scroll bar is disabled and this image is displayed instead.

H Scroll Bar Indicator Normal/Hilited:
The horizontal indicator that the user drags along the scroll bar to scroll the area.

H Scroll Bar Indicator Grips Normal/Hilited:
These are optional. If supplied, this image is drawn centered over the indicator. It can be used to show "grips".

H Scroll Bar First Left Arrow Hilited:
When the user clicks on the left-pointing left-most arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Left Position: The image is drawn this many pixels from the left side of the scroll bar.

H Scroll Bar First Right Arrow Hilited:
When the user clicks on the right-pointing right-most arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Right Position: The image is drawn this many pixels from the right side of the scroll bar.

H Scroll Bar Second Left Arrow Hilited:
When the user clicks on the left-pointing second-from-right arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Right Position: The image is drawn this many pixels from the right side of the scroll bar.

H Scroll Bar Second Right Arrow Hilited:
When the user clicks on the right-pointing second-from-left arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Left Position: The image is drawn this many pixels from the left side of the scroll bar.

H Scroll Bar Single Left Arrow Hilited:
When the user clicks on the left arrow button in the Single Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Left Position: The image is drawn this many pixels from the left side of the scroll bar.

H Scroll Bar Single Right Arrow Hilited:
When the user clicks on the right arrow button in the Single Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Right Position: The image is drawn this many pixels from the right side of the scroll bar.

V Scroll Bar Double Arrows:
A vertical scroll bar, used for scrolling up and down in an area that is bigger than the portion displayed. This is an image of the scroll bar with double arrows at each end (at each end of the scroll bar, there is an arrow button for scrolling up and down). The width of the scroll bar must be exactly 16 pixels.

Top Position: The indicator can be dragged no further up than this many pixels from the top side of the scroll bar.

Bottom Position: The indicator can be dragged no further down than this many pixels from the bottom side of the scroll bar.

V Scroll Bar Single Arrows:
Same as Double Arrows, but with only 1 arrow at each end.

V Scroll Bar Disabled:
The scroll bar is disabled, the user cannot use it for scrolling. (Probably

because the scrollable area is smaller than the area for displaying it.) There is not a separate disabled image for double and single arrows, the same image is displayed in either case (it does not matter because it is disabled anyway).

V Scroll Bar Too Small:
If the scroll bar is in a window that is resized to cause the scroll bar to be very small, too small to display the Single Arrows image (according to the caps of the Single Arrows image), then the scroll bar is disabled and this image is displayed instead.

V Scroll Bar Indicator Normal/Hilited:
The vertical indicator that the user drags along the scroll bar to scroll the area.

V Scroll Bar Indicator Grips Normal/Hilited:
These are optional. If supplied, this image is drawn centered over the indicator. It can be used to show "grips".

V Scroll Bar First Up Arrow Hilited:
When the user clicks on the up-pointing top-most arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Top Position: The image is drawn this many pixels from the top side of the scroll bar.

V Scroll Bar First Down Arrow Hilited:
When the user clicks on the down-pointing bottom-most arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Bottom Position: The image is drawn this many pixels from the bottom side of the scroll bar.

V Scroll Bar Second Up Arrow Hilited:
When the user clicks on the up-pointing second-from-bottom arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Bottom Position: The image is drawn this many pixels from the bottom side of the scroll bar.

V Scroll Bar Second Down Arrow Hilited:
When the user clicks on the down-pointing second-from-top arrow button in the Double Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Top Position: The image is drawn this many pixels from the top side

of the scroll bar.

V Scroll Bar Single Up Arrow Hilited:
When the user clicks on the up arrow button in the Single Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Top Position: The image is drawn this many pixels from the top side of the scroll bar.

V Scroll Bar Single Down Arrow Hilited:
When the user clicks on the down arrow button in the Single Arrows scroll bar, this image is displayed above the scroll bar to indicate that the arrow button is being pressed.

Bottom Position: The image is drawn this many pixels from the bottom side of the scroll bar.

Menu Bar Pattern:
Optional. This pattern is drawn as the background pattern of menu bars.

Menu Bar:
Optional. This is drawn over the menu bar, stretched using the caps settings. You can use this without using Menu Bar Pattern, or alternatively you may use this in conjunction with Menu Bar Pattern by making the middle portion of this image transparent (the pattern will replace the transparent color in this image).

Menu Bar Title Pattern Normal/Hilited/Disabled:
Optional. This pattern is drawn as the background pattern of each menu bar title, obliterating Menu Bar Pattern and Menu Bar in that space, if supplied.

Menu Bar Title Normal/Hilited/Disabled:
Optional. This is drawn over the menu bar title, stretched using the caps settings. This is drawn after the previously mentioned menu bar images, so it will appear "above" them, obliterating them in that space, except where any transparent color used.

Menu Background Pattern:
Optional. Controls the background of menus. You can paste a pattern into this to have a patterned background. If you want a solid color, do not use this, instead use Menu Background in the Colors panel.

Menu Background:
Optional. This is drawn over the whole menu, stretched using the caps settings. You can use this without using Menu Background Pattern, or alternatively you

may use this in conjunction with Menu Background Pattern by making the middle portion of this image transparent (the pattern will replace the transparent color in this image).

Menu Item Pattern Normal/Hilited/Disabled:
Optional. This pattern is drawn as the background pattern of each menu item, obliterating Menu Background Pattern and Menu Background in that space, if supplied.

Menu Item Normal/Hilited/Disabled:
Optional. This is drawn over the menu item, stretched using the caps settings. This is drawn after the previously mentioned backgrounds, so it will appear "above" them, obliterating them in that space, except where any transparent color used.

Menu Separator:
Optional. This is drawn as a separator line between menu items, stretched using the left/right cap settings. It is vertically centered in its space.

Popup Window Frame Normal/Focus:
This is the window frame/border drawn around popup windows or popup menus. The middle portion should be transparent. The "Position" text boxes specify the thickness of the frame on each of the 4 sides. The caps are permitted to be bigger than the frame thickness. The thicknesses must be even numbers (2, 4, 6, 8, 10 etc) for performance and compatibility reasons. The Transparent Color cannot be used to make a non-rectangular window frame at this time due to problems with cross-platform portability.

Window Frame Normal/Focus:
This is the regular window frame/border drawn around windows. The middle portion should be transparent. The "Position" text boxes specify the thickness of the frame on each of the 4 sides. The caps are permitted to be bigger than the frame thickness. The thicknesses must be even numbers (2, 4, 6, 8, 10 etc) for performance and compatibility reasons. The Transparent Color cannot be used to make a non-rectangular window frame at this time due to problems with cross-platform portability. "Focus" refers to the window that is front-most, where keyboard typing will be directed to.

Window Close/Minimize/Maximize Button Normal/Focus/Hilited/Disabled:
This is the Close/Minimize/Maximize button drawn above the window titlebar (the top of the window frame), such as an "X" / "-" / "+" button. The Disabled image is optional -- if supplied, it will be drawn when that button has been disabled by the program, or if not supplied, the button is simply hidden when it

has been disabled by the program.

Right Position:	Horizontally, the button is drawn this many pixels from the right side of the window frame. Or if 0, use Left Position.
Left Position:	Horizontally, the button is drawn this many pixels from the left side of the window frame.
Top Position:	Vertically, the button is drawn this many pixels from the top side of the window frame.

Window Menu Button Normal/Focus/Hilited/Disabled:
As above, but this button shows the "Window Menu", a menu containing commands relevant to that particular window.

Window Resize Button Normal/Focus:
This button is drawn in the right-bottom corner of a window that is resizable. The user clicks and drags this button to resize the window. NOTE: A Resize button with a transparent color is not supported at this time.

Right Position:	Horizontally, the button is drawn this many pixels from the right side of the window frame.
Bottom Position:	Vertically, the button is drawn this many pixels from the bottom side of the window frame.

WonderLight Off/Pause/Ready/Go/Finished:
The WonderLight is a "light" drawn to indicate various states of activity, or a lack of activity. For example, KDX uses it to visually indicate the state of file transfers. The WonderLight must be 16x16 pixels in size.

WonderLight Flash Off/On 1/On 2:
A WonderLight that flashes in order to get the users attention. For example, KDX uses it to visually indicate when a message from another user arrives. When on, the WonderLight is animated, flashing/alternating between 2 images, Flash On 1 and Flash On 2. The WonderLight must be 16x16 pixels in size.

Step 4: Icons Panel

The “Icons” panel looks like this:

On the left, you can see the standard icon, and you can click on it to replace the icon with your own icon.

Icons come in 2 sizes, 16x16 pixels and 32x32 pixels. Programs select which size icon they want to display. You can provide icons in both sizes.

Please see the sections “Transparent Color” and “Maximum of 256 Colors” in the documentation above, because this information also applies to Icons.

Opening other appearance files in AppearanceEdit and looking at the icons is a good way to see how it works.

Using Appearance Files In Other Haxial Programs

Create a folder named “Appearances” in the same folder as the Haxial program (for example, KDX). Put your .hap files into that folder. Then go to Settings, and now the appearance menu shows the contents of that folder. Select the appearance file you want, and you are done.

If you want multiple Haxial programs to all use the same Appearances folder, you can replace the Appearances folder with an alias/shortcut of the same name.

If the Haxial program is one that does not have a Settings window (such as a server program with very little user interface), place an appearance file (or alias/shortcut) into the same folder as the program, and name it “Appearance.hap”. Then open the program and it should load the appearance file.

Any questions/suggestions/feedback?

Your feedback and suggestions are welcomed. Feel free to send a message to Haxial using the form on this webpage:

http://www.haxialsoftware.com/contact/

--