Working with Color in an Object Oriented Environment

Jim Quarato, Bayles Holt, Jerry Harris
Taligent, Inc.

Abstract

Working with color on computers can be both a richly rewarding and a very frustrating experience for users. A good deal of this frustration stems from the inability to work effectively with colors in various spaces or to interchange colors between devices with indiscriminate ease. It is also difficult to reproduce color on different devices because the color producing capability of all color devices is not the same. When color rendering is performed, both the source and the destination color producing capabilities must be taken into account. Being able to understand and manage color devices is an insurmountable task for most clients. This paper describes a color reference model that allows colors to be handled polymorphically, independent of their color space, and which incorporates a framework for matching colors automatically between devices. The load on the user is greatly diminished thus allowing her to devote more attention to the use of color rather than to its constitution.

Taligent, which is a joint venture formed by Apple and IBM, will very shortly release a new fully object bas-ed operating system (TalOS) and an object based application environment system (TalAES). Both of these software products have color handling capability built in from the ground up. The TalAES can run on a variety of operating systems. These new OS and AES systems will be extensible from both the top and the bottom. Since color concerns were addressed from the outset, color from the user standpoint works in a much more seamless and intuitive manner. Because of the object nature of the software, code is reusable, product development is shortened, and usage is simplified. This paper discusses at a rather high level the basic color architecture that is in both software systems.

Graphic system on TaIOS

The Graphic System on the Taligent Operating System has very rich drawing primitives and extensible drawing capabilities, with a color architecture at its heart. The most prominent characteristic of the color architecture is that it is based on a calibrated color standard, the Commission Internationale de l'Eclairage (CIE). It is entirely object oriented (meaning that all colors and color objects can be treated polymorphically), it is extensible (meaning that you can easily incorporate your own favorite color space and color matching methods), and it is transmutable (meaning it is easy to get from one color environment to another). The basic color classes in the system are TGImage and TColor, abstract classes that represent the most primitive color objects of the graphic system. An associated color profile, TColorProfile, describes the color attributes of each object. All colors and images descend from one of these two classes. Because TColor and TGImage are primitive and abstract, the OS and clients can work with colors and images polymorphically.

Color Spaces, Profiles, and Gamuts on TaIOS

Many different color spaces are already defined in the TalOS. A subclass of TColor with a contained color profile is used to describe a color space. The color profile, TColorProfile, defines tonal reproduction curves for the color space and a conversion between RGB and XYZ color spaces. In the case where colors originate from a particular device, the color profile also contains a gamut which describes the range of colors reproducible by that device. Each color is defined in relation to a color space and every color space has an associated color profile. Even normally uncalibrated color spaces, such as RGB or HSV, are well defined using this approach. Every color can be converted to and from RGB space and to and from XYZ space. The color architecture is based on these simple properties.

When working with colors as TColors, it is not necessary to know in which color space its components are contained. It is easy to work in any particular color space, because every color space is a subclass of TColor. Colors that are not in the color space of choice may be treated as TColors and converted quickly to the color space desired.

Since every color is known in relation to a color space, every color, no matter what its origin, is well defined with respect to TColor and a transformation exists into and out of its space and any other space. Transforming between color spaces is accomplished conceptually by first transforming the original color into its equivalent in a well known color space and then from that standard space into the color space of interest. The "standard" interchange color space used by the Taligent color framework is the XYZ color space. All colors are defined to have an XYZ equivalent and a transformation to and from it. Colors already in XYZ simply have an identity transformation. XYZ was chosen as the standard interchange color space, a natural choice because it contains all of the known vis-ible colors.

From a theoretical standpoint, using XYZ as the standard interchange color space is all that is required for the color architecture to work. From a practical standpoint, however, the choice of XYZ is a rather poor one. A more common color space in widespread use is RGB; it is conceptually simple, is easily transformed to and from other color spaces (including calibrated color space such as XYZ), and maps into hardware devices quite naturally. Therefore, as a convenience, RGB is also included as a standard interchange space to allow transforms to and from RGB space as well as XYZ space. However, because RGB is not calibrated universally with all other RGB spaces, each RGB color and its associated transformation is defined with a color profile that characterizes the RGB space being used. Finally, since all colors have both an XYZ and RGB equivalent defined by the TColor base class, it is possible to

transform any color from one space to another without having to know its original color space, but also work easily in either XYZ or RGB space.

Color Matching

As mentioned, RGB is not a single color space; any given set of colors in one RGB space cannot be arbitrarily exchanged for those in another RGB space without first knowing something about the device from which the RGB colors derived. The characteristics of an RGB device are defined by a TColorProfile, which contains a color gamut, specifying the range of colors that can be represented by the device, and the tonal reproduction curves for the device. The same thing applies to CMYK devices.

When the color profile for an RGB color is known, the color can be transformed to another RGB color space provided its color profile is known. The transformation can be made through XYZ space or directly between RGB spaces, particularly if the color profiles are the same. The fact that a transformation exists between any two devices does not imply that the destination device can reproduce the source device color. If the source device color is out of range of the destination gamut, then color matching must be employed. The intent of color matching is to produce a color that is near or perceptually equivalent to the desired color. It is important to note that when colors are matched between some devices that inverse transformations may not always produce the same color as the original. This is particularly true when the destination color gamut is not equal to or contained within the source color gamut.

Transporting colors in and out of the Taligent framework is also possible. A color gamut and profile are provided to the constructor when the color object is instantiated. If the gamut and profile are unknown, a default color profile can be assigned, in which case the resulting color will be no less definitive than it was outside the Taligent framework.

Even so, because of the consistent model for handling colors, it is possible to work with all kinds of colors from many different sources, from colors within a specific color space, or even within an abstract color space, with equal aplomb.

Images

Sometimes it is desirable to work with groups of color samples instead of individual color samples, as in the case of images. The format and organization of such images are very diverse, even though most images are usually represented by some form of RGB samples. In the Taligent color framework, images have certain common properties that allow them to be handled precisely with respect to their color content.

1. All images have a reference to a color profile.
2. Every image provides its own transformation into an RGB, Gray, or Luv color space.

In the case of images, color conversion between spaces is not normally performed until device rendering time. However, image processing, filtering, and so forth, can be and often are performed before rendering. In the Graphics System architecture, images are treated as devices as well as graphic objects and can be drawn into much like any other display device. When another color or color object is rendered into an image, it must first be converted to the color space or format used by the image. During the rendering, color matching is performed in exactly the same manner as described above for individual colors.

When an image is being rendered as a graphic, the whole image is color matched to the destination color space (an output device or another image) as it is being rendered. Image classes provide methods for copying or rendering through data streaming operators. Streams of data can be formed for XYZ, device dependent RGB, or CMYK, which are fully color matched sources. The only difference that image color matching provides as opposed to individual color matching is that color samples are handled all at the same time, as continuous streams, rather than as individual samples.

In effect, images act just like color classes because they have similar operators with respect to color conversion. An entire image may be converted to a different color matched color space (device dependent or not) by converting each color pixel of the image to XYZ and then to the desired space as described above. Because of the obvi- ous overhead in double converting in this way, some ad-ditional liberties are taken with images when converting. For example, in addition to providing streams of XYZ pixels, images provide streams in pre-defined RGB and CMYK formats. Conversion is made directly to these spaces without first going through XYZ.

Extending the System

Though Taligent provides default color matching between devices, some clients may want to provide a form of their own. In addition, clients may wish to provide additional color spaces beyond those supplied by the system. This is all permissible.

To extend the number of color spaces, clients subclass from TColor and create components in the color space desired. A transformation to and from XYZ, RGB and the target color space must be provided as a matter of course. In addition, clients can add any other transformations, filters, or functionality as deemed expedient by the application. Instances of this subclass can then be treated polymorphically by the rest of the system without modification.

In the case of color matching, a preferred color matching method can be added by clients as desired. To do so, both a color matcher (Color Matching Module) and a color profile type must be provided by the method. An ISV who wants to supply their own color profiles may do so by subclassing from the class TColorProfile and TColor-Matcher. TColorMatcher provides the color match for colors and images while TColorProfile defines the device descriptions needed by the color matching process. Subclassing from TColorProfile ensures an acceptable default behavior should a subclassed TColorMatcher encounter a color profile which it does not understand. Clients are also free to provide color matching between their own color matching method and that of any other client's method.