An Object-Oriented Model of Color Devices

Richard A. Holub*

Color Device, Computational model, Object-oriented, Network

Abstract: A model of color devices is introduced which may be implemented in an object-oriented class hierarchy. The goals of the system are 1) to provide a general framework for representing color devices and associated profiles and 2) to enable color devices to be integrated into a network following simple Q&A between the system and an unsophisticated User or the device itself. Devices can be classified as reader or writer, as linear or non-linear and as employing three or more-than-three color channels. The model is a three-dimensional matrix based on the classification. Methods of computing and executing color transformations are known to subclasses and can be inherited along with other class properties. The ICC device types of 'scnr', 'mntr' and 'prtr' can be derived from the model. A video display is a physical device which can appear in different cells of the model, depending on the application; it can serve as a linear input device, linear output device and is often used to represent CMYK devices in proofing.

Introduction

This paper describes a framework for representing color devices in a system (Holub, 1999) for distributing and controlling color reproduction in a network. First, the context for the discussion will be established by reference to the Jini™ model of a distributed system (Sun Microsystems, Inc., 1999.) Next, concepts of Object-Oriented Programming (OOP) will be reviewed briefly. Then a classification of color devices will be developed with respect to attributes of linearity, I/O and numbers of channels. Lastly, the relationship of the model and of particular device objects to a class hierarchy will be considered.

---

* IMAGICOLOR

Practical Applications

The buzz about Java and Jini (both trademarks of Sun Microsystems, Inc.) has been fueled by litigation in which Sun charged that Microsoft has improperly expropriated aspects of Java language development. Behind all the extravagant hype are some useful concepts. By virtue of its architecture neutrality, robustness and network friendliness and security, Java offers important benefits for enterprise integration and distributed control in manufacturing operations (Atherton, 1998.)

Jini is a manifestation of the foregoing trend. Quoting from a technical white paper (Sun Microsystems, 1999):

> "A Jini system is a distributed system based on the idea of federating groups of users and the resources required by those users. The overall goal is to turn the network into a flexible, easily administered tool on which resources can be found by human and computational clients. Resources can be implemented as either hardware devices, software programs, or a combination of the two. The focus of the system is to make the network a more dynamic entity that better reflects the dynamic nature of the workgroup by *enabling the ability to add and delete devices flexibly*." (p. 1, italics added)

Jini's web page sports images of smart appliances which can be added to a home computer network and recognized readily. A user can send a message to a washing machine via the internet and direct it to perform its functions at given times. Nearer term, targeted network appliances are the printers and video monitors of office environments - these could include devices used in color publishing and printing. The "ability to add and delete devices flexibly" depends on how a device identifies itself to the network. This in turn depends on how well the device is modelled as a class in the software system.

The goal of this paper is to present a general and extensible model of color devices which maps naturally onto an object-oriented software system. By virtue of the model, color devices such as scanners, printers and video displays can be added to a system; either they identify themselves or an unsophisticated user can do so by way of simple Q&A.

Brief Review of OOP

Acceptance of Object-Oriented Programming is driven by the economics of software development. Re-usability is key. Software components that are wholly self-contained objects are readily recyclable or easily adapted for other uses.

An object consists of data and methods. Methods can be thought of as the verbs in a problem description; they enable the object to operate on data and get things done. Every object is an instance of a class which is descended from a more abstract class and has greater specificity and limitations than the parent - more on this presently. The ideal object is a black box which may have inputs and/or outputs. In Java's object-oriented practice, the inputs generally must not be modified by the accessed object.

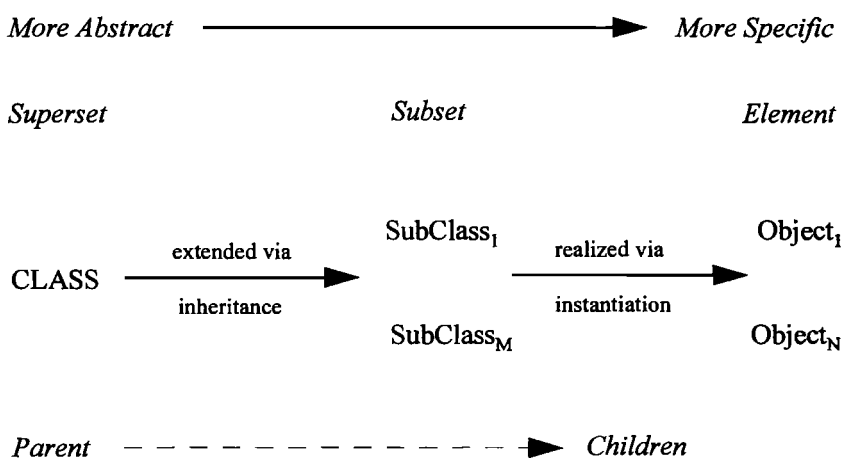

Figure One outlines the relations between classes and objects. Terms in italics suggest various ways of thinking about the relationships.

When one object requests services from another, the parameters of the message it sends should remain intact after access. If data hidden within the accessed object are to be changed, they should be changed by methods of the accessed object, rather than by unintended side effects of the caller. A rigorous discipline of "encapsulation" or "data hiding" is essential to re-usability and a foundation of OOP.

The object-oriented design process maps a problem onto a class structure in such a way as to make best use of existing, re-usable components. However, a good model of the problem facilitates the creation of classes of sufficient generality to be recyclable or readily extended to new class offspring and applications.

Figure one presents a simplified illustration of the relationships of classes and objects. Classes are related (Horstmann and Cornell, 1997) a) by use (a member of class "Node" in a distributed publishing system renders images on a member of class "Color Device,") b) by containment (the class of color devices is a superset of the class of color video displays) and c) by inheritance (the class of video displays may inherit methods of transforming color data from the class of color devices.) Objects consist of data and methods and are distinguished 1) by behavior (what messages do they accept,) 2) internal state (which is partly a function of history) and 3) identity (two or more objects may have the same state, e.g., "color-calibrated and verified," but may provide/request services to/from different classes of color devices in a network.)

Computational Model of Color Device

A framework is proposed in Figure Two in which color devices may be represented in terms of primitive attributes. In a subsequent section, I will show how it maps onto an object-oriented class hierarchy. With the possible exception of "3-channel" / ">3-channel" the attributes are binary. Either a device is linear or not; it either reads or writes. The number of channels is treated as binary here, but another approach could be rationalized.

Linearity of Color Mixture

Before describing the exemplars entered in the matrix, it is important to be clear about what is meant by the attributes. Linearity refers to color mixture or color modelling properties. If a linear model of color mixture can be applied to a device with adequate results, then the device is linear for our purposes. In fact, few, if any, devices are truly linear in the most general sense; Cathode Ray Tube (CRT-based) displays are not linear. However, it is often possible, for CRT displays, to isolate the non-linear characteristics and apply linear models of color mixture with very

acceptable results (assuming the non-linear components of the problem are also handled appropriately.)

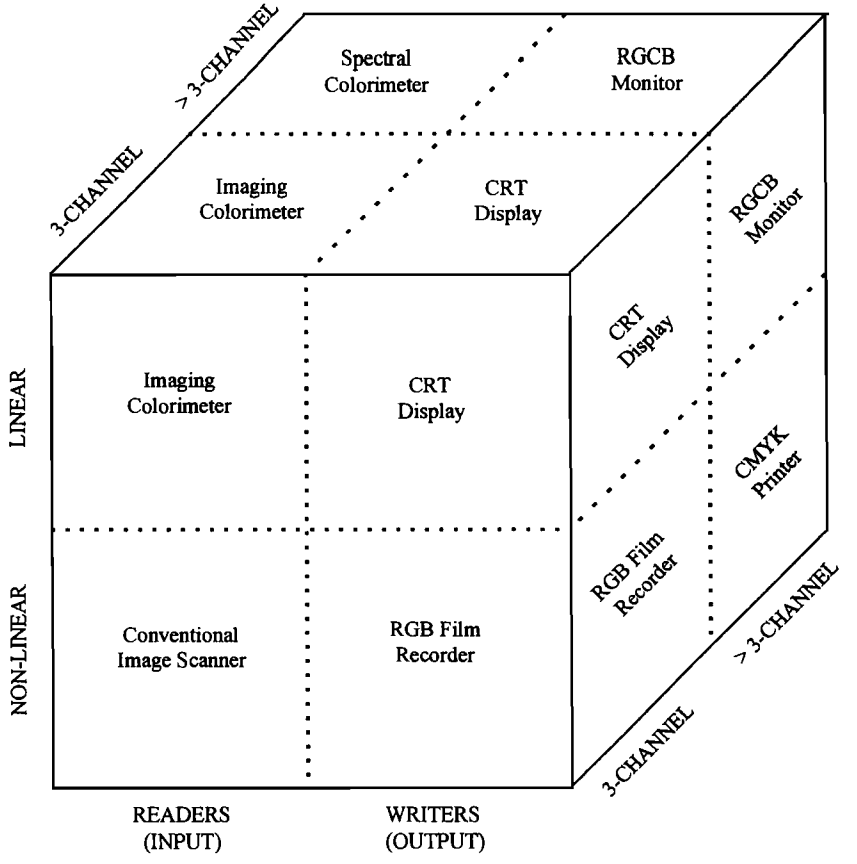

Figure Two illustrates a three-dimensional matrix in which color devices may be categorized for purposes of software control.

In the case of readers (input devices such as scanners and cameras) linearity implies conformity with what has variously been called the Maxwell-Ives and Luther-Ives criterion. Gordon and Holub (1993) and Holub (1995) provide succinct discussions of this point; Horn (1984) provides a more extended discussion which is both thorough and rigorous. An image capture device is linear, for the purposes of this

computational model, if the spectral sensitivities of the device's three channels are linear combinations of the human color matching functions.

When the Luther-Ives criterion is satisfied, the TriStimulus Variables (TSVs,) R, G and B, apprehended by a Standard Observer or a simulator of the Standard Observer, are a linear combination of the X, Y, Z TSVs. Furthermore, as shown rigorously by Gordon and Holub, the matrix elements comprising the calibration transformation which converts RGBs to XYZs are precisely the coefficients involved in defining the device's responses to spectral inputs in terms of the color matching functions of the Standard Observer.

Receptive Gamuts and Rendering Gamuts

When the criterion is ***not*** satisfied, it is necessarily the case that the reader device will make metameric matches that are not made by a Standard Observer. In other words, the reader in question will confuse colors that are distinguishable to a human who has normal color vision. How serious a problem this is depends on a number of factors. The problem is illustrated in Figure Three, where the gamuts of non-linear reader and human are represented by circles. Two spectrally different

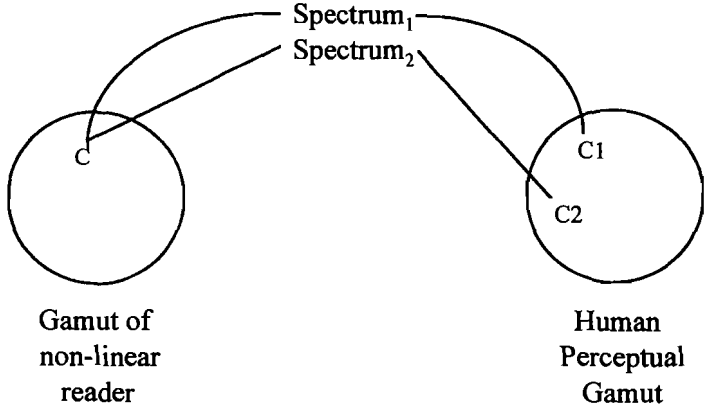

Figure Three uses simplified gamut diagrams to illustrate an essential property of a non-linear reader - there is no exact transformation from the reader's TSVs to the Standard Observer's, linear or otherwise.

stimuli give rise to a single color for the non-linear reader, but different colors for the human. Clearly, there is no way to map from the color encoded by the reader to those matched by the human.

In summary, linear input and output devices (three channels) have in common that a linear matrix transformation is adequate to describe the color mixing / color matching behavior of the device. For input devices, linearity insures conservation of gamut. A gamut is the set of all colors perceivable by a device; reasonably, the reference gamut is that of human vision. Conservation of gamut means that, in principle at least, a linear reader device can perceive exactly the gamut of colors that a human can.

Writers (output devices) generally have gamuts that are a subset (sometimes a small subset) of humans.' This is not necessarily so, however. The apparatus employed by Wright (1928-29) in conducting the color matching experiments, which served as a partial basis for the '31 Standard Observer, was an output device with the potential to render every humanly perceivable color. Presently, I will describe an RGCB monitor with a very large rendering gamut. Conservation of gamut is not really an issue with writers; they generally reproduce colors metamerically, just as did the original color matching apparatuses.

Example Objects and the Workings of Inheritance

Returning to Fig. 2, I will discuss the exemplar devices. Among non-linear readers (bottom left,) are conventional image scanners. "Conventional" means only that most scanning devices in the market have not been designed so as to meet the Luther-Ives criterion referred to above. Non-linear readers are often characterized by way of the procedures outlined in Appendix B of ANSI Standard IT8.7/1. The latter involve fitting polynomials to datasets consisting of device codes resulting from a scan of a standard calibration image and colorimetric measurements of the patches of the calibration image. The polynomials may be used to generate interpolation tables which mediate the conversion from a device coordinate system (that of the non-linear reader) to a device independent coordinate system.

What we just reviewed are generic methods of device characterization. Some of the techniques are applicable to non-linear writers as well. In

terms of an object-oriented implementation, we can visualize an ancestral "color-device-nonlinear," which knows about non-linear (including polynomial functions for interpolating measured datasets) characterization functions. It could also know about sparsely sampling the characterization function to make a table which serves as the basis for interpolation. It may know about interpolators (e.g., trilinear, tetrahedral-linear or pyramidal-linear,) but more likely, it would use the services of another class which knows about interpolators of different precisions, integer vs. float, etc.

Our ancestral *color-device-nonlinear* itself descends from the class *color-device*. (In an effort to improve clarity, complicated class and object names are italicized.) *Color-device-nonlinear* cannot be instantiated as an object; it is too abstract. Nevertheless, it can have grandchildren, such as *color-device-nonlinear-writer-3-channel* and *color-device-nonlinear-reader-3-channel* which draw on its knowledge and methods. The latter classes are instantiable as objects which correspond to specific network printing and scanning resources. Foregoing is a simple-minded example of the workings of inheritance and how it serves as an organizing principle for an object-oriented system.

By modelling the problem in a general way and defining appropriate classes, we can move the system in the direction of being self-organizing. In other words, we add a device to the system. On the basis of the answers to a few, simple questions, system knows what class the device instantiates and knows how to characterize the device and to make color transformation profiles for it. This isn't the case if a device merely identifies itself as 'scnr' in the ICC parlance.

Color-device-nonlinear-writer-3-channel has a sibling (to reduce this to absurdity!) *color-device-nonlinear-writer->3-channel*; they share many ancestral methods but are specialized in different ways. The latter object knows how to handle black ink or other extra colorants, for example. It is represented in Fig. 2 by a CMYK printer shown in the bottom-right-rear corner of the matrix. The exemplar of its 3-channel counterpart is a RGB Film Recorder in the bottom-right-front. If punch-through and interimage effects could be ignored, this device might observe linear color mixing rules in density space - at least for transparency photographic materials. However, print materials would surely be

subject to nonlinearities of reflection colorimetry and inherit appropriate color processing methods.

The upper-left-front cell of Fig. 2 is occupied by an imaging colorimeter. This usually will be a device whose channel response functions satisfy the Luther-Ives criterion - i.e., are composed of linear combinations of color matching functions. In some cases, four filters might be used to make such a device, but one of the filters represents the blue lobe of the x-bar Color Matching Function. Responses through this filter are added to those through the red lobe to produce an estimate of the X TSV.

In the upper-left-rear corner of Fig. 2 is a spectral colorimeter. It acquires data in fairly narrow bands across the visible spectrum. Practical devices may have from 16 to 160 or so data channels. Principles of linearity, in the common, general, engineering sense, are used to combine multi-channel data into TSVs or related CIE variables, at which point the device could be moved forward in the matrix to join the 3-channel devices. However, an important reason for using a spectral sensor is to be able to substitute one illumination source for another - to "see" the subject as it would look under lighting other than that used for image capture. Objects in this class (color-device-linear-reader->3-channel) should be endowed with specialized methods for exploiting this physical capability and be given a distinct cell in the model.

It was noted earlier that self-emissive video displays typically observe linear rules of color mixture - quality devices are designed to do so. Intrinsically, these devices are writers. It was mentioned in the abstract, that they may appear to belong to other classes, *depending on the application*. For example, a software image synthesis or creation application works in intimate association with a video display to input images to other phases of production. The gamut of the display in effect becomes the palette of the input medium and its linear additive color mixture properties dictate the coordinate system of the image data. This underscores the fact that software application can qualify as a color-device in this model. The combination of application and display in the present example appears to belong to class color-device-linear-reader-3-channel. Obviously, objects of this sort rely on objects which can take responsibility for accurate rendering to the display in order to effect their purposes.

Earlier I noted that writers of large gamut are possible. Figure Four expresses the motivation for such a device. It is represented in the model of Fig. 2 as an RGCB monitor in the upper-right-rear corner. The familiar horseshoe of the spectrum locus of the CIE Standard 2° Observer is indicated by the solid line in Fig. 4. When the open ends are joined by the line of the extraspectral hues, the diagram shows the gamut in two dimensions, to the extent that is possible.

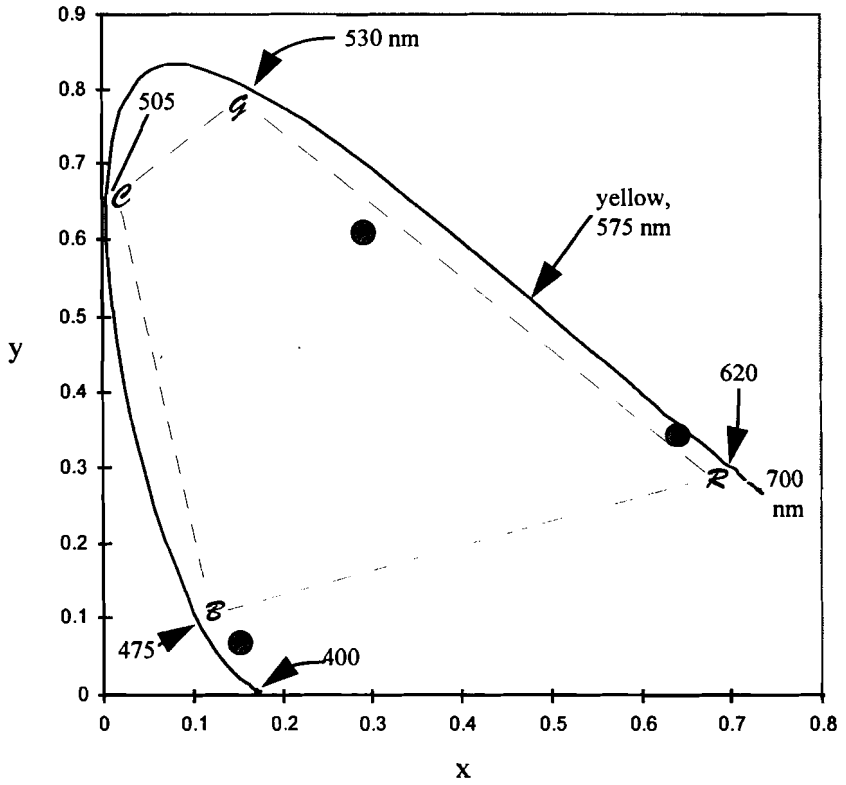

Figure Four illustrates an extended RGCB gamut, referred to the chromaticity diagram. The value of x chromaticity appears on the abscissa and of y on the ordinate.

Each point on the solid line represents the chromaticity coordinates derived from the X, Y and Z TSVs of the color matched to a narrow spectral band of illumination by the Standard Observer. The approximate disposition of spectral points of 400, 475 ("B"), 505 ("C," or Cyan,) 530 ("G"), 575, 620 ("R") and 700 nanometers are indicated

on the diagram. For comparison, the chromaticity coordinates of the Red, Green and Blue channels of an EBU CRT are expressed by large, filled circles. Connecting those circles defines the boundary of a fairly restricted gamut.

To digress briefly, the script R, G, C and B points in the diagram indicate the approximate locations of the primaries of the RGCB display having the dominant wavelengths noted above and called out on the graph. The dashed lines connecting these primaries enclose a substantial gamut; however, it is probably a conservative estimate of what is available. The latter point is supported by Figure Five, which consists of plots of the luminous output of EBU phosphors as a function of wavelength for the three channels. Although the functions are fairly broad and "sloppy," their chromaticities lie surprisingly close to the spectrum locus. From this it appears, in principle at least, that one can chose R, G, C, and B so that they will lie even closer to the spectrum locus while providing enough luminous intensity to be useful.

Why care about a supergamut video display? First, it will be useful in portraying some of the vivid dyes used in textiles and other products. Second, it will provide a means of compensating for gamut sacrificed in a projection medium. In other words, when a video display is used to project images onto a screen or onto paper stock, there are bound to be effects which desaturate colors, reduce dynamic range, etc. End of digression.

Returning to Fig. 4, we know from Grassmann's Laws that one of the four, linear, additive primaries can be matched by a combination of the other three, even if it is necessary to add one of the three to the one-to-be-matched. If C were the primary to be matched, then an amount of R (represented in color matching by a negative value) would be added to C in order to move its influence toward the line connecting B and G on the chromaticity diagram. Although the system is linear, we have a rendering problem reminiscent of that occasioned by adding a black ink. Objects in this class, such as our hypothetical RGCB monitor, would need specialized methods for dealing with extra colorant(s).

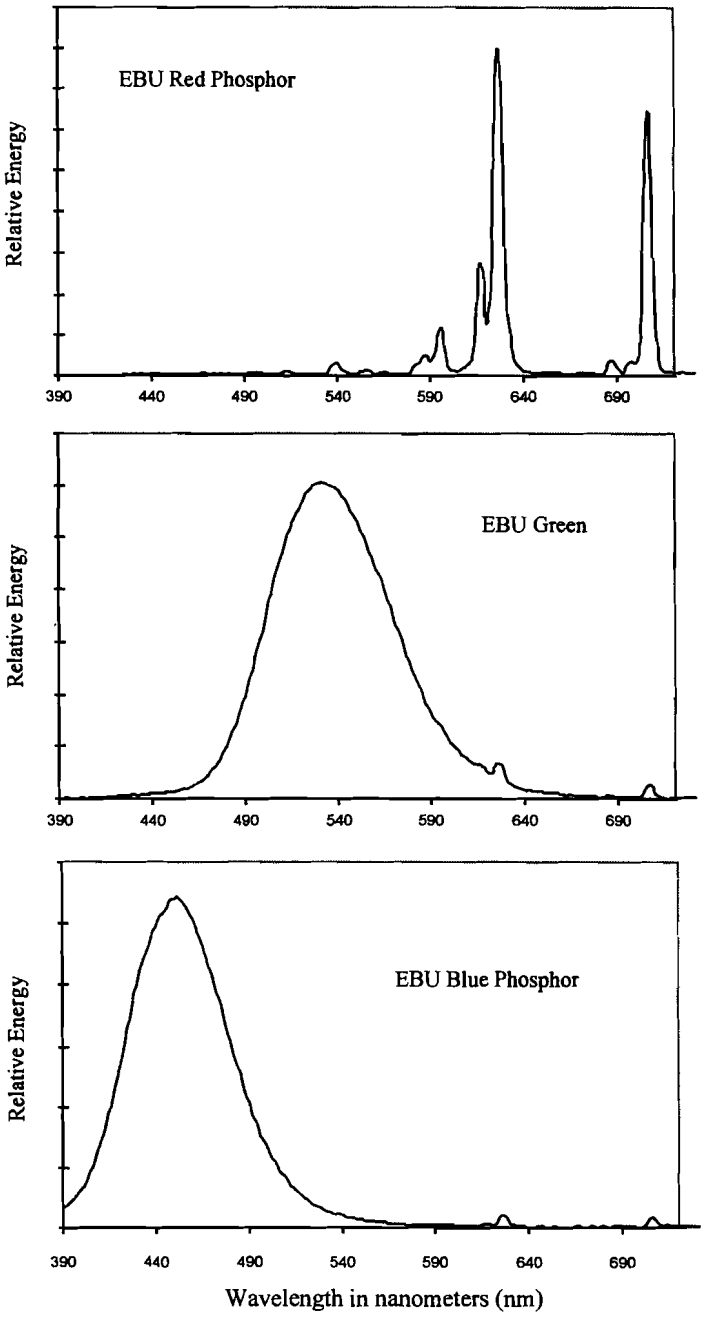
EBU Red Phosphor
Relative Energy
390
440
490
540
590
640
690
EBU Green
Relative Energy
390
440
490
540
590
640
690
EBU Blue Phosphor
Relative Energy
390
440
490
540
590
640
690
Wavelength in nanometers (nm)

FIGURE FIVE

SUMMARY

We have considered object-oriented concepts and practices with a view to appreciating the benefits of a general model of color devices. The model lends itself to Jini-like implementations which facilitate the addition (and deletion) of devices to (from) dynamic network configurations, while at the same time encompassing ICC device types. Figure Six illustrates and summarizes the manner of developing a hierarchical class structure explored in this paper. Classes at higher levels of abstraction may not be instantiated (realized) as objects corresponding to specific devices, but they may embody methods of processing color data which are available to real objects through inheritance.

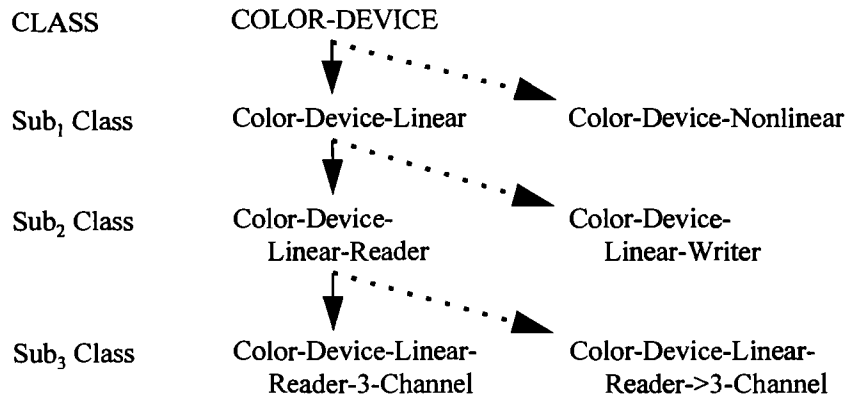

Figure Six illustrates one way to develop a class hierarchy for color devices, consistent with the computational model presented. Categories pointed to by dotted lines fan out into successive sub-categories which are not spelled out by the diagram, but correspond to cells in Fig. 2.

Literature Cited

ANSI, The American National Standards Institute (1993) Graphic technology - Color transmission target for input scanner calibration, Standard IT8.7/1. Secretariat is NPES, 1899 Preston White Dr., Reston, VA 22091.

Atherton, R. W. (1998) Moving Java to the factory. *IEEE Spectrum*, **35**(12): 19-23.

Gordon, J. J. and R. A. Holub (1993) On the use of linear transformations for scanner calibration. *Color Research & Application*, **18**(3): 218-219.

Holub, R. A. (1999) Accountable color in network applications. These *TAGA Proceedings*.

Holub, R. A. (1995) Colorimetric aspects of image capture. 48th Annual Conference Proceedings, IS&T, the Society for Imaging Science and Technology, Arlington, VA., pp. 449-451.

Horn, B. K. P. (1984) Exact reproduction of colored images. *Comput. Vision Graphics Image Process.*, **26:** 135-167.

Horstmann, C. S. and G. Cornell (1997) Core Java. Vol. 1: Fundamentals. Upper Saddle River, New Jersey, USA: Sun Microsystems Press / Prentice Hall, pp. 630.

Sun Microsystems, Inc. (1999) Jini™ architectural overview: Technical white paper. Available at web site www.sun.com/jini, pp. 23.

Wright, W. D. (1928-29) A re-determination of the trichromatic coefficients of the spectral colours. *Trans. Opt. Soc.*, **30:** 141-164.