Full-Spectral Color Calculations in Realistic Image Synthesis

Garrett M. Johnson and Mark D. Fairchild
Rochester Institute of Technology

Realistic image synthesis has long served to create images nearly indistinguishable from real scenes. However, image synthesis isn't limited to creating attractive pictures. Often computer renderings provide an accurate simulation tool. Our work focuses on using computer graphics as a method for creating full-spectral synthetic images. These images can then be used for color-imaging systems design and analysis. We set out to find a graphics technique that could create noise-free images of arbitrary spatial and spectral resolution. In examining current rendering techniques, we realized that many suffer from some of the same problems that we face in designing color-imaging systems.

Color imaging and reproduction systems are currently undergoing a paradigm shift. Traditionally, color-image capture occurred in a trichromatic fashion, using photographic film with three distinct color sensitivity layers. Color reproduction was handled by printing the captured image with three dyes, as with photographic paper, or with four inks in a commercial printing press. These methods could reproduce color quite capably, but they often resulted in large color shifts. This becomes apparent in the image capture and reproduction of *metameric objects*, or objects that have different spectral properties but appear to match to a given observer and viewing conditions. Two metameric objects might look identical to you and me, but appear quite different to a camera. Traditional color printing inherently creates metameric matches to the original scene, since it's impossible to recreate all possible spectra with three or four inks. Thus while the reproduction might match the original scene under one illumination, that match might break down if the illumination changes. When designing new color-imaging systems, it's important to minimize the possible color shifts associated with metameric objects.

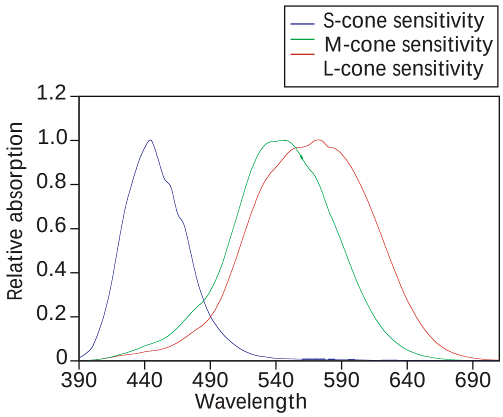
The only way to guarantee that the color of the original object and the reproduction appear the same across changes in observers and illumination is to ensure that the reproduction and the original match spectrally. This proves a formidable task, and many have researched the topic.[1] Computer simulations can help ease this difficult project and enhance a color-imaging system's design and analysis. Effective simulations need accurate input data. For color-imaging systems, this input data comes in the form of a full-spectral image. Unfortunately, obtaining full-spectral images remains a long and laborious process. Creating synthetic full-spectral images for imaging system design and analysis led us to the world of computer graphics.

Originally, we planned to use computer graphics techniques to create a synthetic input image that contained accurate radiometric data at each pixel. These synthetic images could then function as input data for color-imaging systems analysis. Furthermore, while in the process of creating these images, we also wanted to use computer graphics to interactively simulate various phenomena of color and color imaging. This requires maintaining accurate color representations throughout image synthesis.

However, standard graphics techniques fall prey to many of the same errors associated with color-imaging systems. Most commercially available rendering packages still use the standard RGB color model to perform color calculations.[2] Prone to large color shifts and color errors, this trichromatic technique also can't demonstrate metamerism. The graphics community has developed several alternate methods for more accurate color synthesis, which we'll review briefly here. All these methods attempt to reduce the dimensionality of a full-spectral problem to create a less computationally expensive solution. Unfortunately, much of the information contained in the spectral curves of light sources and materials gets lost when a reduced set of numbers represents the colors. This loss

The standard RGB color model manifests color errors when rendering or imaging metameric objects. Here we review some alternative color-representation methods for accurate color image synthesis.

0272-1716/99/$10.00 © 1999 IEEE



1 **Relative spectral sensitivities of cones.**

becomes especially apparent when rendering complicated metameric or fluorescent scenes. If you want truly accurate color calculations, rendering should use full-spectral information. We used a simple extension to OpenGL that can interactively simulate many important wavelength-based color phenomena, such as metamerism, by maintaining full-spectral information throughout color calculations.[3]

Metamerism and human vision

The "natural" world consists of tens of thousands of unique color spectra. Fortunately, we can create identical color pairs from objects that have very different spectral properties. This enables automotive manufacturers to match a cloth seat cover to a vinyl dashboard, or a metal panel to a plastic bumper. Metamerism also lets us display realistic scenes on a television screen or computer monitor. Many examples of these every day metamers exist. For that reason, it's important to understand the principles behind color in general, and the color of metameric objects specifically.

Let's begin with the definition of color:

> *Color* is an attribute of visual perception consisting of any combination of chromatic and achromatic content. This attribute can be described by chromatic color names such as yellow, orange, brown, red, pink, green, blue, purple, and so on, or by achromatic color names such as white, gray, black, and so on, and qualified by bright, dim, light, dark, and so on, or by combinations of such names.[4]

Although this definition seems circular, you need to remember that color serves as a visual perception attribute. That is, without the human eye, no color exists. While the human visual system consists of much more than just the eye, a quick look at the photoreceptors in the retina provides insight into this idea of metamerism. More information about the world of color vision and appearance appears elsewhere.[5,6]

The photosensitive cells in the retina separate into two categories, rods and cones. Rods primarily handle low-light vision, while cones handle color vision. Cones come in three distinct types, generally called L, M, and S, and each having different spectral sensitivities as shown in Figure 1.

These photoreceptors absorb light in their respective regions and send a signal to the rest of the visual system. This signal only indicates that the receptor absorbed light, it doesn't indicate the light's nature or wavelength. We call this *univariance*, because when the photoreceptor absorbs light, the effect stays the same no matter what the wavelength of the absorbed light might be.[5] This principle of univariance creates the phenomenon known as metamerism. Two different spectra can trigger the same photoreceptor response and thus appear to have the same color.

The implications for computer graphics are both positive and negative. Because of univariance and metamerism, CRT displays can create scenes that appear identical to real scenes, despite having only three distinct primaries. Unfortunately, metamerism also poses a large problem for color-image synthesis. It's almost impossible to accurately simulate metameric objects with the traditional RGB color model. Color-encoding methods that use more than three wavelengths work better, but can still result in large discrepancies.

Methods for color calculation

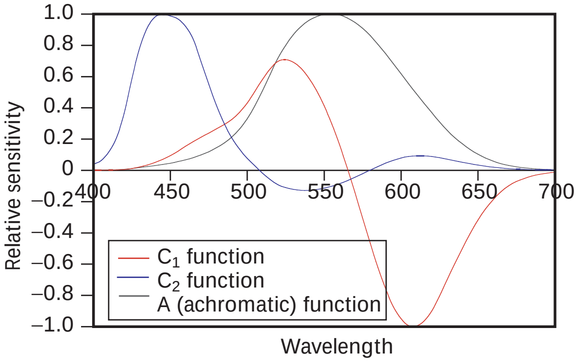
Hall[7] and Glassner[8] provide excellent insight into color calculations for computer graphics. We'll discuss briefly some of the more prevalent methods here.

RGB

The RGB color model remains the most common method for rendering systems today.[2] In this method, three wavelengths—corresponding to red, green, and blue—define light sources and material properties. Color calculations simply involve multiplying the light source and material RGB values. This model proves a logical choice for many computer graphics applications, since end users can easily select the material and light-source properties. The resulting color calculations can be quickly performed in hardware and displayed via a standard CRT monitor. Unfortunately, this method's problems far outweigh its convenience.[7] One obvious problem results from the extreme device dependence of the RGB primaries. The same RGB colors calculated and displayed on one device might not appear the same when displayed on a different device. It's also impossible to synthesize metameric objects accurately, since spectra are sampled using only three wavelengths. If two materials match (have identical RGB values) under one illumination, they must match for all illuminations. Likewise, if two objects don't match for one light source, this mismatch will occur for all other light sources. Clearly, the RGB model proves a poor choice when attempting to synthesize color images accurately.

Gaussian quadrature sampling

Meyer[9] suggested an alternate choice for color calculations that provides superior performance without the expense of full-spectral calculations. By realizing that most naturally occurring spectra vary smoothly, we can estimate most spectra with a small number of basis functions. Meyer chose these basis functions by first examining the human visual system, then deriving a set of color axes that passed through the most populated color regions. These axes formed a new opponent color space known as the AC_1C_2 space, where the A axis represents an achromatic channel, and the C_1C_2 axes represent opponent chromatic channels. This space is a simple linear transformation of the LMS sensitivities shown above. Figure 2 illustrates the relative sensitivities of the AC_1C_2 space.



2 **AC_1C_2 sampling curves.**

In constructing this space, Meyer used Gaussian quadrature rules to determine the minimum number of single wavelength samples necessary to get an accurate AC_1C_2 evaluation. Meyer found that as little as four spectral samples could provide an accurate estimate of the AC_1C_2 space, and thus of the human photoreceptor response.[9]

Far more accurate than the RGB color model, this method generally provides good color accuracy for many natural scenes because of its fundamental basis in both human perception and sampling theory. Unfortunately, representing a spectral curve with such a limited number of samples sacrifices large amounts of information. When color accuracy is crucial, such as simulating metameric objects, this method will often produce undesired results.

Linear-color representation

Another basis-function method for producing more accurate color calculations involves general linear models.[10,11] These methods generally use characteristic vector analysis, or principle component analysis, to determine the minimum number of basis functions necessary to accurately describe the light source and material spectra. This analysis can be simplified by adding in an "observer," or sensor function such as the LMS human sensitivities, or the International Commission on Illumination (Commission Internationale de L'Eclairage—CIE) 1931 Standard Colorimetric Observer. These methods can produce very accurate color calculations with a small number of basis functions.[10]

While highly accurate, these methods also have some limitations. The basis functions used to represent the materials and light sources must be calculated before the rendering stage, which often proves difficult. The basis functions also lose physical meaning and take on purely statistical meanings. These functions might be difficult for average users to interpret and understand if they're interested in examining the "spectra" of the material-lighting interactions.

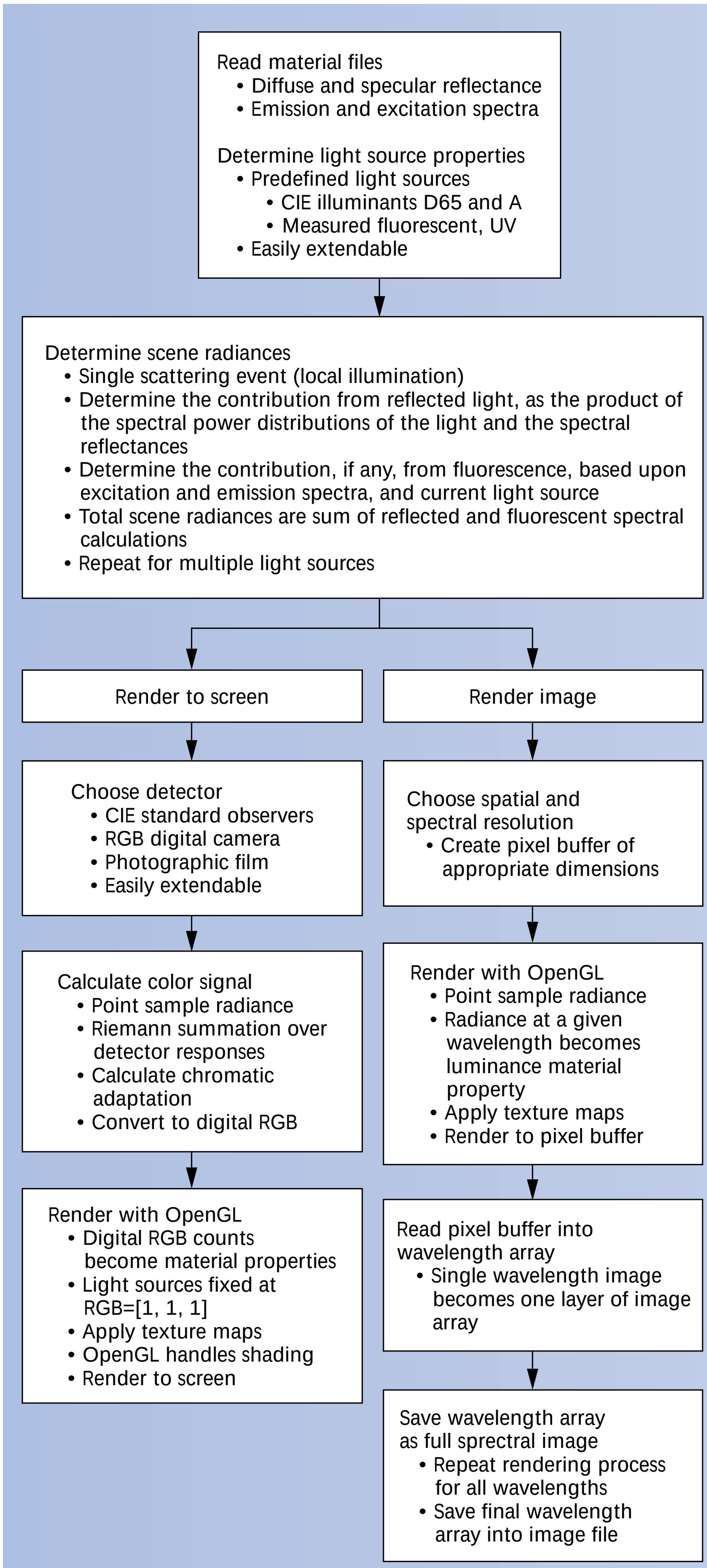
Linear-color representations synthesize most color images accurately. However, working with subtle spectral differences, such as metameric objects, might require a large number of basis functions to capture all the necessary color information accurately. In this case, representing materials in a statistical method yields little benefit.

For our particular task, the inability to create a full spectral-radiance image proved another limitation of linear-color models. Reducing color interactions to statistical interactions makes it difficult to return to the physical world. Whereas we wanted to know the actual spectral data of the scene at each pixel, a linear-color representation resulted in a statistical weighting at each pixel. These weighting values would then combine with the precalculated basis functions to approximate the correct spectra. This necessary extra step for using the images as input into an imaging system simulation provided only an approximation of the original spectra. For purposes where the output image displays immediately, either as hardcopy or on a CRT, linear-color models offer an excellent choice for accurate color synthesis.

Full-spectral local illumination

For our particular purposes, we created a simple extension to OpenGL to allow for full-spectral color calculations.[3] This initial method used finely point-sampled spectral representations for light sources and material properties. We also included absorption, emission, and specular spectra in the material definitions. This let us simulate many wavelength-based color phenomena such as metamerism and fluorescence. By using OpenGL-based hardware acceleration, we could interactively simulate full-spectral-based color calculations. By maintaining full-spectral representations in our local illumination model (93 wavelength bands), we could also make full-spectral images quickly for our color-imaging systems analysis. Effective image analysis requires rendering the full-spectral images in as high a bit depth as possible. Using high-resolution graphics hardware permits creating these images with up to 16 bits per pixel for each wavelength.

This technique, while adequate for most spectral-based color calculations, remains limited in many ways. The standard OpenGL lighting and shading model is a simple local illumination-type model. This means that our full-spectral technique applies only to single interactions between objects and light sources. Objects don't interact with each other, thus greatly simplifying the rendering equations. The resulting color images aren't designed to be perceptually indistinguishable from a real scene, but rather to have accurate color. Consequently, we weren't as concerned with the rendering efficiency, and we could still achieve real-time interactivity. See the sidebar "OpenGL-based Spectral Color Rendering" (next page) for more details of our technique.


Read material files
• Diffuse and specular reflectance
• Emission and excitation spectra
Determine light source properties
• Predefined light sources
• CIE illuminants D65 and A
• Measured fluorescent, UV
• Easily extendable
Determine scene radiances
• Single scattering event (local illumination)
• Determine the contribution from reflected light, as the product of the spectral power distributions of the light and the spectral reflectances
• Determine the contribution, if any, from fluorescence, based upon excitation and emission spectra, and current light source
• Total scene radiances are sum of reflected and fluorescent spectral calculations
• Repeat for multiple light sources
Render to screen
Render image
Choose detector
• CIE standard observers
• RGB digital camera
• Photographic film
• Easily extendable
Choose spatial and spectral resolution
• Create pixel buffer of appropriate dimensions
Calculate color signal
• Point sample radiance
• Riemann summation over detector responses
• Calculate chromatic adaptation
• Convert to digital RGB
Render with OpenGL
• Point sample radiance
• Radiance at a given wavelength becomes luminance material property
• Apply texture maps
• Render to pixel buffer
Render with OpenGL
• Digital RGB counts become material properties
• Light sources fixed at RGB=[1, 1, 1]
• Apply texture maps
• OpenGL handles shading
• Render to screen
Read pixel buffer into wavelength array
• Single wavelength image becomes one layer of image array
Save wavelength array as full sprectral image
• Repeat rendering process for all wavelengths
• Save final wavelength array into image file

A **Flowchart of OpenGL-based spectral color rendering.**

OpenGL-based Spectral Color Rendering

Figure A shows the technique we used for our full spectral extension to OpenGL. This code is freely available at http://www.cis.rit.edu/research/mcsl/online/spectral.html.

Following the flowchart from top to bottom shows the steps taken in our full-spectral-based rendering technique. The first block indicates the material spectral definitions. End users must specify, as a minimum, the diffuse spectral reflectance as a function of wavelength, from 300 to 740 nm in 5-nm steps. If desired, they can also specify the specular reflectance function, useful for defining metallic objects as well as emission and excitation spectra. The latter two material properties are necessary for fluorescence. Users can also specify a gloss coefficient or a texture-mapping file. Once they define the material properties, users must select the lighting properties. Up to three lights are currently implemented, one ambient and two local light sources. Users can select from the predefined light sources, defined by the spectral power distributions of the same wavelength range as the material properties. If users want other light sources, they can easily add them to the existing database.

Next, users must determine the scene radiance values by using a single scattering event. Multiplying the material reflectance functions by the light source spectral power distributions results in the diffuse and spectral reflectance contributions. The calculation for the fluorescent contribution also builds on the excitation and emission spectra and the current light source properties. Summing the reflectance and fluorescent contributions results in the total radiance. This process repeats for multiple light sources. Now users must choose the rendering path.

If the desired output is an interactive simulation on the monitor, users must choose a detector or "observer" function. The detector functions essentially sample the spectra into tristimulus values, following the color calculations. This proves necessary, since no current display device can recreate an entire wavelength spectra. Users can choose from a set of predefined observer functions, including the CIE 1931 Standard Colorimetric Observer, a digital camera, and photographic film. The resulting color signals are then calculated by a Riemann summation over the chosen detector function as a method for numerical integration. If users selected a human observer, they could also select a chromatic adaptation transform. Once calculated, the color

A word on global-illumination techniques

In general, the global-illumination community recognizes the need for accurate color rendering and for physically accurate image synthesis. Consequently, many global-illumination algorithms use spectral information, where accuracy ranks higher than efficiency.

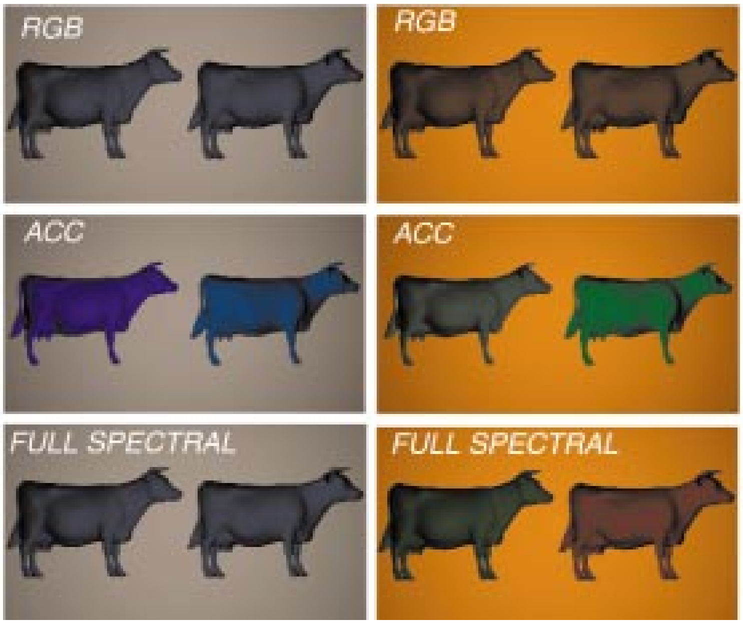
signal values are transformed into monitor digital counts, using a monitor characterization model.[13] The system feeds these counts into OpenGL as material properties. OpenGL then handles the rendering and rasterization. This process can be repeated interactively, as users change the light source or detector properties.

If users want a full-spectral image as output, they must first choose the dimensions of the output image. They can select the spatial resolution, currently limited only by the frame buffer's size. Users also choose the spectral resolution, including starting and ending wavelengths, as well as wavelength steps. The system then feeds the scene radiance values into OpenGL as luminance material properties, one wavelength at a time. Once the scene is rendered, the image is read from the frame buffer into one layer of an image array. This process repeats for all chosen wavelengths. The image array is then saved as an *n*-dimensional Hierarchical Data Format (HDF) scientific data file.[14]

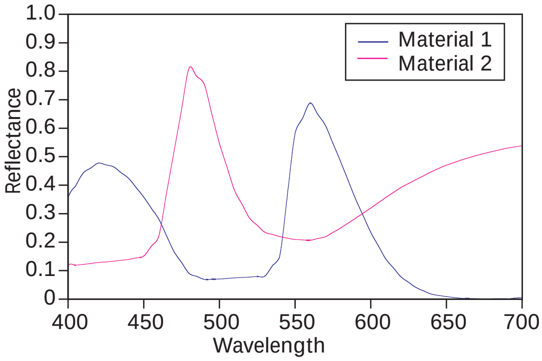
On the other hand, most publicly available global-illumination packages—more concerned with rendering time—sacrifice spectral accuracy. You can use any of the techniques discussed above in conjunction with a global-illumination algorithm to increase color accuracy. For our purposes, we needed noise-free spectroradiometric images that we could make quickly. Physically accurate renderings, often computationally expensive, resulted in rendering times far longer than we desired. The random nature of many spectral-based global-illumination techniques (such as the Monte Carlo-based technique) often create moderately noisy images, depending on the amount of time dedicated to the rendering process. Although not noticeable to the human eye, this noise becomes apparent when the image serves as input to a color-imaging simulation.

The need for accurate color synthesis

Because many industries now work with rendered images, accurate color synthesis has increasingly become more important. For designers, the ability to render different materials under various lighting conditions is very important. Consider, for example, an automotive manufacturer who wants to select different materials for the interior of a car. Perhaps the dashboard uses a certain plastic, while the seat covers consist of a specific cloth. Ideally, the colors should match each other in any given lighting (and for any given customer). The only way to guarantee this would be to create a perfect spectral match between the seats and the dashboard. Unfortunately, this is probably not possible, because the colorant materials differ greatly. Wouldn't it be nice to simulate the materials under various lighting conditions, such as the showroom fluorescent lights and daylight? If the materials and light sources aren't rendered accurately, then the designer might make poor choices. For instance, if the materials and lights specify the RGB color model or the AC$_1$C$_2$ model, the materials might always appear to match for all light sources or might never appear to match. Figure 3 gives a simple illustration of this problem. We rendered two materials using the RGB, AC$_1$C$_2$, and full spectral methods. These highly metameric materials, as shown in Figure 4, are designed to match for an average human observer under CIE Illuminant D65 (an average daylight). The material spectra used in this example, known as "metameric blacks," are mathematically generated metamers commonly used in industry and color research laboratories.[12] While not always attainable from real colorants, these spectra provide excellent insight into color reproduction systems.



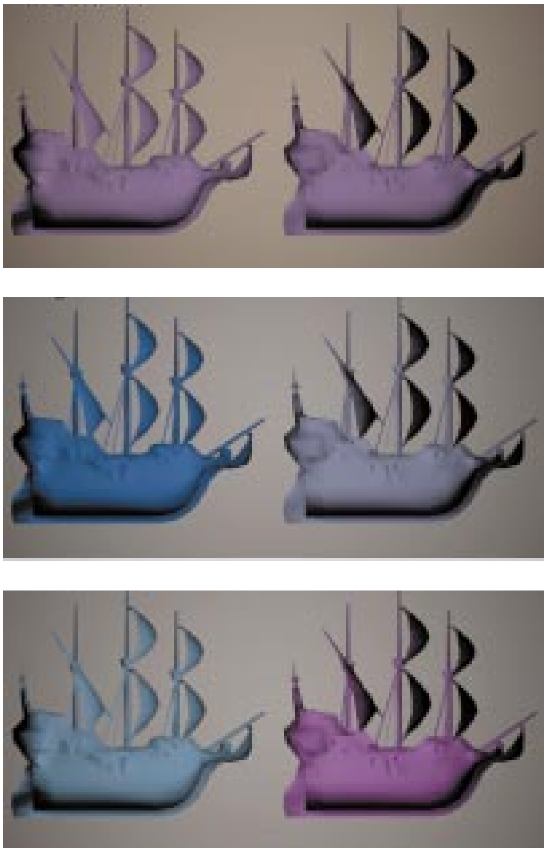
3 **Illustrating the need for spectral color calculations. Rendered materials using RGB, AC$_1$C$_2$, and full-spectral methods under CIE Illuminant D65 (left) and A (right).**



4 **An example of two highly metameric objects.**

The left column of Figure 3 shows the two objects rendered using the RGB, AC_1C_2, and full-spectral models while illuminated by D65. You can see that the RGB model and the full-spectral model accurately predict a color match. The AC_1C_2 method predicts that the colors don't match. When we change the illuminating light source to CIE Illuminant A, as shown in the right column, the RGB model again predicts a match. The AC_1C_2 model continues to predict a mismatch. The full-spectral model now accurately predicts a mismatch, whereas before it predicted a match. You can see from this dramatic example the importance of full-spectral-based color calculations.

Performing wavelength-based color calculations outside of any given "observer" also permits simulating different imaging systems. Most realistic image-synthesis systems aim to create images that the human eye can't distinguish from a real scene. Computer graphics isn't just limited to a human observer, however. Often we want to render an image that might "appear" identical to a real scene as viewed by another type of imaging system, such as motion-picture film or a digital camera. If materials and light sources are defined by their respective spectral properties, it's very easy to interactively change the scene's ultimate detector and see the effects on the resulting colors.

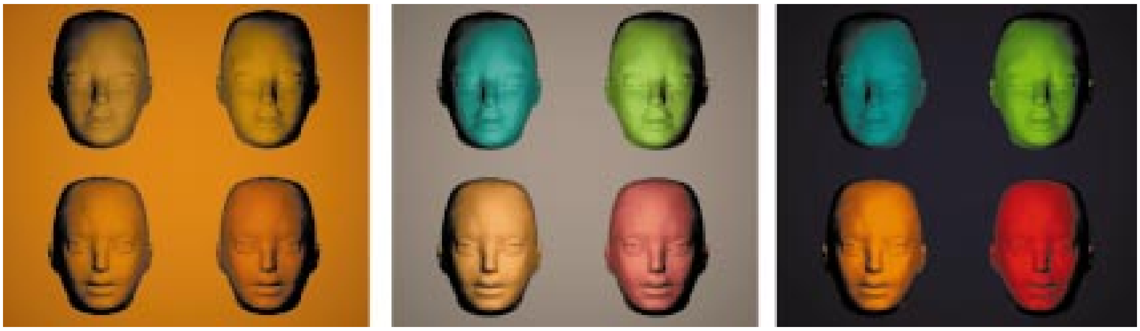


5 The simulation of various imaging systems: a human (top), a digital camera (middle), and a photographic transparency film (bottom).

Such simulations remain impossible if a small number of samples, such as the RGB model, represents colors. If one function samples a spectrum, it's impossible to determine how that spectrum might have appeared to different sampling functions. The AC_1C_2 color space exemplifies a sampling function based on the human visual system. Once the AC_1C_2 function samples a spectrum, however, the system can't recover all the information in the original spectrum. Therefore, we can't accurately determine how a different detector would "view" the original scene. The linear-color representation methods for image synthesis can simulate various imaging systems, provided we choose the original basis functions wisely. With those methods, it's often a good idea to have the desired detectors in mind when determining the basis functions.

One of the original goals of our work was to interactively simulate various imaging systems. By maintaining full-spectral information throughout all the color calculations, we simulated the colors that several different imaging systems determined. Figure 5 shows how a scene might appear to a human, a digital camera, and a photographic transparency film.

Full-spectral color calculations can also synthesize the color phenomenon called fluorescence—the absorption of light at one wavelength and emission of that light in a different (usually longer) wavelength. Simulating fluorescence requires tracking reflection, excitation, and emission spectra as well as wavelength interactions, since they depend on each other. It's impossible to render fluorescent materials accurately with simple color models such as the RGB and AC_1C_2 methods. Such materials require a more complicated color representation scheme such as a linear-basis function model or our full-spectral model.[3,10] Figure 6 shows an example of fluorescence rendered using our extension to OpenGL. The excitation spectrum for all objects occurs mainly in the short wavelength region, between 300 nm and 400 nm. The left panel shows the scene illuminated under CIE Illuminant A (a typical incandescent light), which has little energy in the short wavelength region and thus doesn't produce fluorescence. The middle panel shows the same scene illuminated by D65, which has some



6 Rendering of fluorescent objects. The scene illuminated under CIE Illuminant A (left), D65 (middle), and black light (right).

energy in the short wavelength region. The right panel shows the objects illuminated by a black light, which has all of its energy in the short wavelength region.

Conclusion

We set out to use computer graphics as a tool for creating spectroradiometric synthetic images for color-imaging-systems analysis. In the process, we realized that computer graphics often suffer from the same problems as color imaging systems. These problems occur when a small number of "primaries" samples and represents the spectral nature of light. Large color errors often show up during the imaging and reproduction of metameric objects, or objects that have different spectral properties, but appear to match for a given viewing condition.

Creating synthetic images using the traditional RGB model visibly produces these same errors. Despite these errors, many commercial graphics programs still exclusively use the RGB color model. We've discussed a few alternatives for color representation capable of far more accurate color calculations. This proves especially important for applications where computer graphics simulates real-world lighting and material interactions. If accurate color rendering is important, we strongly recommend using full-spectral color calculations.

These images are currently being used as input data for color-imaging-systems analysis. We're examining ways to create more realistic scenes, while maintaining interactivity. This includes the addition of full-spectral texture mapping, as well as other texturing algorithms such as bump-mapping. By adding textured "detail," we can analyze when imaging systems begin to lose color information. We're also extending this work to accurately simulate more complicated color phenomena, such as absorption and scattering for transparent and translucent objects. ■

Acknowledgments

This work was supported by the Eastman Kodak Company and Munsell Color Science Laboratory, and was partially completed while we were on sabbatical leave at Cornell University's Program of Computer Graphics.

References

1. P.D. Burns and R.S. Berns, "Image Noise and Colorimetric Precision in Multispectral Image Capture," *Proc. Sixth Color Imaging Conf.*, The Society for Imaging Science and Technology/Society for Information Display (IS&T/SID), Springfield, Va., 1998, pp. 83-85.
2. M.S. Peercy, B.M. Zhu, and D.R. Baum, "Interactive Full-Spectral Rendering," *Proc. 1995 Symp. on Interactive 3D Graphics*, ACM Press, New York, 1995, pp. 67-68.
3. G.M. Johnson and M.D. Fairchild, "Computer Synthesis of Spectroradiometric Images for Color Imaging Systems Analysis," *Sixth Color Imaging Conf.*, The Society for Imaging Science and Technology/Society for Information Display (IS&T/SID), Springfield, Va. 1998, pp. 150-153./
4. International Commission on Illumination (Commission Internationale de L'Eclairage—CIE), *International Lighting Vocabulary*, CIE Publ. No. 17.4, Vienna, 1987.
5. B.A. Wandell, *Foundations of Vision*, Sinauer, Sunderland, Mass., 1995.
6. M.D. Fairchild, *Color Appearance Models*, Addison-Wesley, Reading, Mass., 1998.
7. R. Hall, *Illumination and Color in Computer-Generated Imagery*, Springer, Berlin, 1989.
8. A.S. Glassner, *Principles of Digital Image Synthesis*, Morgan-Kaufmann, San Francisco, 1995.
9. G.W. Meyer, "Wavelength Selection for Synthetic Image Generation," *Computer Vision, Graphics, and Image Processing*, Vol. 41, No. 1, 1988, pp. 57-79.
10. M.S. Peercy, "Linear Color Representations for Full-Spectral Rendering," *Computer Graphics* (Proc. Siggraph 93), ACM Press, New York, Vol. 27, 1993, pp. 191-198.
11. B.A. Wandell, "The Synthesis and Analysis of Color Images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 9, No. 1, 1987, pp. 2-13.
12. G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*, John Wiley and Sons, New York, 1982.
13. R.S. Berns, "Methods for Characterizing CRT Displays," *Displays*, Vol. 16, No. 4, 1996, pp. 173-182.
14. National Center for Supercomputing Applications, *Hierarchical Data Format (HDF)*, Univ. of Illinois at Urbana-Champaign, 1997, http://hdf.ncsa.uiuc.edu.

Garrett M. Johnson *is a PhD student in the Center for Imaging Science at the Rochester Institute of Technology, Rochester, New York. He received his MS in color science and BS in imaging science from RIT. His research interests include color-appearance modeling, color-image synthesis, and computer graphics.*

Mark D. Fairchild *is Director of the Munsell Color Science Laboratory and Professor of Color Science and Imaging Science at Rochester Institute of Technology, Rochester, New York. He received his BS and MS degrees in imaging science from RIT and his PhD in vision science from the University of Rochester. He served as chair of CIE Technical Committee 1-34, which formulated the CIE 1997 Interim Color Appearance Model (CIECAM 97). He received the 1995 Bartleson Award from the Color Group (Great Britain) for his research in color appearance and other aspects of color science.*

Readers may contact Johnson at the Munsell Color Science Laboratory, Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology, 54 Lomb Memorial Dr., Rochester, NY 14623, e-mail gmj8204@cis.rit.edu.