Short Note

The *Short Note* is intended for quick publication. A Short Note has been peer reviewed, but not as extensively as a feature article. Its purpose is to bring work to public attention as quickly as possible. Publication of a Short Note does not preclude later publication of a feature article describing the work in more detail.

How to Derive a Spectrum from an RGB Triplet

Andrew S. Glassner
Xerox Palo Alto Research Center*

The accurate display of synthetic images requires careful attention to color. Simply modeling color with an RGB triplet leads to color aliasing. Most color antialiasing techniques begin with a specification of color as a spectral energy distribution. There are two apparent hurdles to be overcome to support the spectral model in rendering systems. One is the difficulty of interactively designing a spectral description of a desired color. The other problem is the difficulty of converting existing and useful libraries of RGB triplets into equivalent spectral representations.This note presents a simple method that takes as input an RGB triplet viewed on a given monitor and computes a corresponding spectrum for use in a rendering system.

The authors of image-rendering software must try to avoid aliasing artifacts. Many of the aliasing phenomena well-known in computer graphics (e.g., jaggies, strobing, and jumping) arise when the geometry of samples in the sampling space forms a lattice or a repeating pattern. Much effort has been directed toward eliminating sampling artifacts in areas such as screen space,[1] texture space,[2] time,[3] and reflection space.[4] Other work has been published on spectral antialiasing in the domain of visible light.[5]

Many rendering systems simply evaluate color at three discrete wavelengths, typically one each for the red, green, and blue regions of the visible light spectrum. These wavelengths are implicitly based on the three phosphors in some monitor. This is a naive method for dealing with the complex phenomena that involve color, which include such critical image-generation topics as illumination, reflection, and refraction. Sampling all colors at exactly three fixed wavelengths leads directly to spectral aliasing, including severe undersampling artifacts.

I believe that this practice has remained popular to the present day only because other aspects of rendering were even more objectionable when handled incorrectly. Now that the rendering equation,[6] stochastic ray tracing,[3] and radiosity[7] have opened the doors to physically accurate photorealism, and indeed have even been merged,[4] I agree with many other researchers that it is time to acknowledge the inadequacy of the informal RGB triplet approach to color description, and seek to improve color fidelity in all rendering systems.

Spectral antialiasing is an important part of any rendering system that strives for realism. All colors in a scene description (coating colors, internal colors, light-source colors, fog colors, etc.) should be initially described as spectra to the rendering system, and all internal computations should be performed on a spectral representation of color (in this note I use the word "spectrum" to refer to a spectral energy distribution). Practical antialiasing techniques for computing with sampled spectra are still an area of active research.[8-10]

My goal in this note is to provide a tool that allows a user to design a color on a color monitor, and then automatically create a corresponding spectrum for use in a spectrally based antialiasing rendering system. This facility would allow a designer to mix a color interactively, and then use that color for objects,

* This research was done while the author was at the Technical University of Delft, The Netherlands.

lights, filters, and so on. It would also allow designers to use existing color libraries based on RGB triplets.

More formally, the algorithm's input is a color, described as a combination of red, green, and blue intensities on a given monitor. The output is a spectrum $S(\lambda)$, satisfying the following condition: A perfectly diffuse reflecting surface with a diffuse reflection curve given by $S(\lambda)$, directly illuminated by an equal energy spectrum (i.e., a white light), with no other shading conditions, would be rendered on the original monitor with the same RGB components that the user originally specified.

Hall has noted that systems which use both RGB and spectral representations of color may initially confuse a designer.[9] For example, a user may design a color called *monitor white*, with RGB values (1,1,1). When the user is psychophysically adapted to the monitor, this monitor white will appear "white," but because of color adaptation (1,1,1) is only the local white for that situation; the actual chromaticities of monitor white may reveal it to be reddish or bluish. Gregory describes this phenomenon nicely: "We see a car's headlamps as white while on a country drive, but in town where there are bright white lights for comparison, they look quite yellow."[11]

The user might expect a reflective material covered with a shiny layer of monitor white to be a perfect reflector, but since the color is not described by an equal-energy spectrum, the reflected light will be tinted. The correct interpretation of monitor-designed colors is difficult because of the psychophysics of color adaptation, another active research topic.[10] This does not diminish the utility or importance of a spectral representation, but emphasizes the need for users to be sensitive to these issues. Color designers must be aware of color adaptation, and should not expect a color viewed on a particular monitor under particular conditions to imply more than a suggestion of the coordinates of that color in some objective color space.

Finding a spectrum from an RGB triplet

In this note, the term *RGB space* refers to the 3D color space available on a particular monitor; this space is determined by the phosphors used in the monitor. The term *XYZ space* refers to the 1931 2° observer CIE *XYZ* color space.[12]

Rather than find a spectrum that matches a point in a particular monitor's RGB space, it is easier to match the corresponding point in *XYZ* space. Consider the standard conversion of a color in *XYZ* space to a particular monitor's RGB space. This is accomplished via a linear transformation M_i, built for monitor i. It is useful to write the *XYZ* color as a row vector $\mathbf{X} = [x\ y\ z]$, and the RGB color as a row vector $\mathbf{R} = [r\ g\ b]$ (where $0 \le x,y,z,r,g,b \le 1$). Then M_i is a 3×3 matrix built from the chromaticities of the phosphors in monitor i:

$$\mathbf{R} = \mathbf{X}M_i \tag{1}$$

To convert a spectrum $A(\lambda)$ into a point in *XYZ* space we integrate with the three CIE standard matching functions $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$ to produce the chromaticity values x, y, and z. Wyszecki provides tables for the matching functions for the CIE 1931 2° standard observer, from 380 to 780 nm:[12]

$$\begin{aligned} x &= \int_{380}^{780} A(\lambda)\, \bar{x}(\lambda)\, d\lambda \\ y &= \int_{380}^{780} A(\lambda)\, \bar{y}(\lambda)\, d\lambda \\ z &= \int_{380}^{780} A(\lambda)\, \bar{z}(\lambda)\, d\lambda \end{aligned} \tag{2}$$

Consider that an infinite number of spectra $A(\lambda)$ can give the same *XYZ* coordinates; these spectra are all metamers. Any of these spectra would be a solution to Equation 2 for a given vector $\mathbf{X}$.[13,14]

A monochromatic line spectrum may be represented as a delta function, $\delta_x(\lambda)$:

$$\delta_x(\lambda) = \delta(x-\lambda) = \begin{cases} 0 \text{ when } x \ne \lambda \\ 1 \text{ when } x = \lambda \end{cases} \tag{3}$$

Define $\boldsymbol{T}$ to be a 3D space of line spectra with an orthonormal base of axes given by $(\delta_u(\lambda), \delta_v(\lambda), \delta_w(\lambda))$, $u \ne v \ne w$. Thus a point $\mathbf{S} = [a\ b\ c]$ corresponds to a spectrum $S(\lambda) = a\delta_u(\lambda) + b\delta_v(\lambda) + c\delta_w(\lambda)$. To convert $S(\lambda)$ to *XYZ* space, write the vector $\mathbf{X}$ as a product of the point $\mathbf{S}$ and a matrix C, where C holds the nine entries corresponding to the amplitudes of $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$ at the three wavelengths λ_u, λ_v, and λ_w:

$$\mathbf{X} = \mathbf{S}C \tag{4}$$

where

$$\mathbf{S} = [S(\lambda_u)\ \ S(\lambda_v)\ \ S(\lambda_w)]$$

$$C = \begin{bmatrix} \bar{x}(\lambda_u) & \bar{y}(\lambda_u) & \bar{z}(\lambda_u) \\ \bar{x}(\lambda_v) & \bar{y}(\lambda_v) & \bar{z}(\lambda_v) \\ \bar{x}(\lambda_w) & \bar{y}(\lambda_w) & \bar{z}(\lambda_w) \end{bmatrix}$$

Combining Equations 1 and 4 yields $\mathbf{R} = (\mathbf{S}C)M_i$, and solving for $\mathbf{S}$ gives the solution:

$$\mathbf{S} = \mathbf{R}(CM_i)^{-1} \tag{5}$$

Discussion

Matrix inversion is a costly process. It is useful to precompute a matrix $J_i = (CM_i)^{-1}$ for each monitor i that will be used for color designing; of course J_i may be computed at runtime for new monitors.

The spectra of the axes of the spectrum space $\boldsymbol{T}$ are nonzero only at three wavelengths: λ_u, λ_v, and λ_w. We must exercise a bit of caution when picking these wavelengths, since they give rise to the matrix C, which must be nonsingular. I associate λ_u, λ_v, and λ_w with the wavelength of the largest entry in my sampled versions of the $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$ matching functions respectively; the result is nonsingular and numerically stable. My matrix C is built from samples of the matching functions at $\lambda_u = 590$ nm, $\lambda_v = 560$ nm, and $\lambda_w = 440$ nm. The corresponding entries of the matching functions then give the C below:

$$C = \begin{bmatrix} 1.026 & 0.757 & 0.001 \\ 0.594 & 0.995 & 0.004 \\ 0.348 & 0.023 & 1.747 \end{bmatrix}$$

Lasers may be used to create monochromatic light.[15] Thus, a line spectrum such as the one generated by the above method can be created with a combination of lasers tuned to the appropriate frequencies.

Note that the above technique may generate spectra with some negative values, producing a physically unrealizable situation. This problem is not particular to the technique presented here; even traditional RGB triplet sampling may lead to negative intensities. The correct theoretical treatment of such situations has not yet been resolved, but some practical measures have been suggested.[9,16]

Observe that the technique in this note does not select *the* spectrum corresponding to a given RGB triplet; an infinite number of such spectra exist. In fact, the spectrum built from just a single application of this method is perhaps the worst for the job, since it is zero everywhere but at three specific wavelengths. In a situation where this spectrum describes a transparent filter, most of the light energy will be blocked; a more spread-out spectrum would probably pass more energy.[17] This could lead to unexpected results when rendering.

One way to get a fuller spectrum is to build several spectra by the above process. Each spectrum is built using a different set of wavelengths as the basis for the spectrum space $\boldsymbol{T}$. For each set n, $(\lambda_u, \lambda_v, \lambda_w)_n$ defines a different space $\boldsymbol{T_n}$, and thus requires its own transformation matrix $J_{i,n}$. Once computed, these spectra may be averaged together (any weights may be used on the spectra when averaging; equal weighting works fine). The composite averaged spectrum will be a metamer for any of its components.

Where in the visible region should the additional sample points be placed? I think we should be guided by two principles: computational stability and visual perception. Computational stability is achieved with a judicious choice of sampling frequencies, guided by the standard matching functions. The goal is a matrix that is both nonsingular and stable under inversion. For visual perception we should capitalize on the fact that the hue sensitivity of the human visual system varies across the spectrum.[11] I prefer sampling at those points in the hue discrimination curve where the eye is most sensitive; these are the regions of the spectrum conveying the most information to the viewer.

Another approach to building a more robust spectrum is to interpret the line spectra produced by the technique in this note as samples of a continuous spectrum. Appropriate interpolation would then provide a fuller spectrum.

Maloney has argued that most naturally occurring spectra are fairly smooth.[18] This assumption suggests that an upper limit can be placed a priori on the highest frequency in a naturally occurring spectrum (this simplification is generally true, but there are exceptions; for example, the spectrum of a fluorescent lamp can be very spiky). Within the limits of this assumption, a spectrum can be approximated by a linear combination of a small number of fixed functions (called fitting functions).[8] Wandell has shown that the first three functions from a Fourier basis suffice to match a collection of 462 Munsell color chips to within about 0.02 normalized root-mean-squared error.[19]

Specifically, choose a flat spectrum and a single cycle each of sine and cosine. To use these curves as fitting functions, find their RGB coordinates in the color space of the monitor holding the color to be matched. Since these functions form a basis in spectral space, the same linear combination of RGB values that matches the desired RGB triplet can be applied to the fitting functions to create the desired spectrum. We write the fitting functions (from 380 to 780 nm) as follows:

$$F_1(\lambda) = 1.0$$

$$F_2(\lambda) = \sin\left(2\pi \frac{\lambda - 380}{400}\right)$$

$$F_3(\lambda) = \cos\left(2\pi \frac{\lambda - 380}{400}\right)$$

We can then find the correct weights **W** by matching the fitting functions' RGB counterparts with the goal RGB triplet **R**; this relationship is given by $\mathbf{W} = \mathbf{R}D^{-1}$. Matrix D contains the RGB points produced by the fitting functions, found by applying Equation 2 to each function to produce an XYZ point, and then converting to RGB space with Equation 1. Row r of D is thus given by

$$D_{r,} = \left[\int_{380}^{780} F_r(\lambda)\, \bar{x}(\lambda)\, d\lambda \;\int_{380}^{780} F_r(\lambda)\, \bar{y}(\lambda)\, d\lambda \;\int_{380}^{780} F_r(\lambda)\, \bar{z}(\lambda)\, d\lambda \right] M_i$$

The new spectrum is then formed by $S(\lambda) = \mathbf{W}_1F_1(\lambda) + \mathbf{W}_2F_2(\lambda) + \mathbf{W}_3F_3(\lambda)$.

Practical rendering systems can be designed and built to accept as input the results of the techniques presented in this note. Internally, such a system would use finely sampled spectra for all computation. I present here some relevant details about such a system, which produced images published earlier.[20] All colors in the rendering system were read from the scene file as spectra, and all color computations were performed on spectra. Colors were reduced to RGB only in the last step in the rendering process, when the final spectrum at each pixel was integrated to find the corresponding point in XYZ space and then converted to the RGB space for the appropriate monitor. Spectra were stored as 14 samples, spaced in 30-nm increments from 380 to 780 nm. Input spectra were computed from RGB triplets using the techniques in this note repeated with five different sets of unique frequencies; the extra sample was discarded. The representation was interpreted by the system as a sampled version of a linearly continuous spectrum.

Internal color computations for illumination, reflection, refraction, atmospheric dispersion, etc., were performed by stochastically sampling the involved spectra, performing the operation, and then reconstructing the result for storage. Interpolation was performed by linearly interpolating stored spectral values; reconstruction used non-overlapping triangular filters. Better interpolation and reconstruction techniques may be worthwhile.

I plan to examine alternative basis functions for the spectrum space, built from continuous, nonzero polynomials. I hope that these functions will provide more paths to finding computationally robust and visually predictable spectra. ■

Acknowledgments

Thanks to Wim Bronsvoort and Frits Post of the Faculty of Mathematics and Informatics at the Technical University of Delft, my hosts in The Netherlands while I was first learning about the subtleties of color; Rob Tow of Xerox PARC for discussions about color and comments on a draft of this note; Ken Shoemake of Xerox PARC for bringing two of the references to my attention; and the reviewers for constructive comments that helped improve the presentation. This note is a revised version of Technical Memo #87-2, Faculty of Mathematics and Informatics, Technical University of Delft, The Netherlands, written in November 1987.

References

1. F. Crow, "The Aliasing Problem in Computer-Generated Shaded Images," *CACM*, Vol. 20, No. 11, Nov. 1977, pp. 799-805.
2. P.S. Heckbert, "Survey of Texture Mapping," *CG&A*, Vol. 6, No. 11, Nov. 1986, pp. 56-67.
3. R. Cook, "Stochastic Sampling in Computer Graphics," *ACM Trans. Graphics*, Vol. 5, No. 1, Jan. 1986, pp. 51-72.
4. J. Wallace, M. Cohen, and D.P. Greenberg, "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods," *Computer Graphics* (Proc. SIGGRAPH), Vol. 21, No. 4, July 1987, pp. 311-320.
5. R.A. Hall and D.P. Greenberg, "A Testbed for Realistic Image Synthesis," *CG&A*, Vol. 3, No. 8, Nov. 1983, pp. 10-20.
6. J.T. Kajiya, "The Rendering Equation," *Computer Graphics* (Proc. SIGGRAPH), Vol. 20, No. 4, Aug. 1986, pp. 143-150.
7. M.F. Cohen et al., "An Efficient Radiosity Approach for Realistic Image Synthesis," *CG&A*, Vol. 6, No. 2, Mar. 1986, pp. 26-35.
8. G.W. Meyer, "Wavelength Selection for Synthetic Image Generation," *Computer Graphics, Vision, and Image Processing*, Vol. 41, No. 1, Jan. 1988, pp. 57-79.
9. R. Hall, *Illumination and Color in Computer Generated Imagery*, Springer-Verlag, New York, 1988.
10. M.C. Stone et al., "Color Gamut Mapping and the Printing of Digital Color Images," Tech. Report EDL-88-1, Xerox PARC, Palo Alto, Calif., 1988.

11. R.L. Gregory, *Eye and Brain: The Psychology of Seeing,* World University Library, London, 1966.
12. G. Wyszecki and W.S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae,* Wiley & Sons, New York, 1982.
13. H. Grassman, "On the Theory of Compound Colors," *Phil. Mag.,* Apr. 1854.
14. R.W. Hunt, *The Reproduction of Color,* Wiley & Sons, New York, 1975.
15. E. Hecht and A. Zajac, *Optics,* Addison-Wesley, Reading, Mass., 1974, pp. 481-506.
16. A.S. Glassner, "Exposure Keys for the Digital Darkroom," *The Ray Tracing News,* Vol. 2, No. 2, June 1988, pp. 6-7, privately circulated—contact author to obtain.
17. R. Evans, *An Introduction to Color,* Wiley & Sons, New York, 1948.
18. L.T. Maloney, "Computational Approaches to Color Constancy," Tech. Report 1985-01, Applied Psychology Lab., Stanford Univ., Stanford, Calif., 1985.
19. B.A. Wandell, "The Synthesis and Analysis of Color Images," Tech. Memo 86844, NASA Ames Research Center, Moffett Field, Calif., 1985.
20. A.S. Glassner, "Spacetime Ray Tracing for Animation," *CG&A,* Vol. 8, No. 2, Mar. 1988, pp. 60-70.

Appendix: Computing the transformation matrices

For completeness, I give a summary of how to derive the XYZ-to-RGB transformation matrix M_i for a given monitor. Hall gives a more detailed discussion.[9]

From the monitor specifications, find the xy chromaticities for the CRT white spot (w_x, w_y) and red, green, and blue phosphors (r_x, r_y), (g_x, g_y), and (b_x, b_y). Each xy pair is then augmented with the corresponding $z = 1 - x - y$. From the phosphor triplets build the matrix K below. From the white-spot triplet build the XYZ color vector **W**, which is the color corresponding to the chromaticities (w_x, w_y), scaled so that the luminance Y has the value 1.0. It will be useful in the derivation to define the RGB white vector $\mathbf{F} = [1\ 1\ 1]$, and the XYZ-to-RGB matrix $N_i = (M_i)^{-1}$. Also define the vector **V** and matrix G, which are related by $\mathbf{V} = \mathbf{F}G$:

$$\mathbf{W} = \begin{bmatrix} \frac{w_x}{w_y} & 1 & \frac{w_z}{w_y} \end{bmatrix}, \qquad K = \begin{bmatrix} r_x & r_y & r_z \\ g_x & g_y & g_z \\ b_x & b_y & b_z \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} G_r & G_g & G_b \end{bmatrix}, \qquad G = \begin{bmatrix} G_r & 0 & 0 \\ 0 & G_g & 0 \\ 0 & 0 & G_b \end{bmatrix}$$

First, observe $N_i = GK$ (the RGB-to-XYZ matrix is the phosphor matrix, differentially scaled so that the white spot Y value is set to 1.0). Since N_i relates the XYZ white **W** to the RGB white **F**, write $\mathbf{W} = \mathbf{F}N_i$. Substituting for N_i, $\mathbf{W} = \mathbf{F}(GK)$, so $\mathbf{W}K^{-1} = \mathbf{F}G$. Recalling $\mathbf{V} = \mathbf{F}G$, rewrite this as $\mathbf{V} = \mathbf{W}K^{-1}$ and thereby compute **V** and build the corresponding matrix G. Then compute $N_i = GK$, and from that compute $M_i = (N_i)^{-1}$. In summary, the steps are as follows:

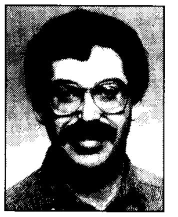
1. Build **W** and K from the monitor white spot and color phosphors.
2. Compute $\mathbf{V} = \mathbf{W}K^{-1}$.
3. From the vector **V** build the matrix G.
4. Compute $N_i = GK$.
5. Compute $M_i = (N_i)^{-1}$.

The XYZ values for the standard NTSC monitor xy chromaticities are given in Hall as $r = (0.670, 0.330)$, $g = (0.210, 0.710)$, and $b = (0.140, 0.080)$.[9] The white spot for illuminant D_{6500} is $w = (0.313, 0.329)$. Following the above procedure and then combining with the matrix C given earlier will produce the matrices M_i, J_i, and D with values:

$$M_i = \begin{bmatrix} 1.967 & -0.955 & 0.064 \\ -0.548 & 1.938 & -0.130 \\ -0.297 & -0.027 & 0.982 \end{bmatrix}$$

$$J_i = (CM_i)^{-1} = \begin{bmatrix} 0.724 & -0.260 & .0002 \\ -0.340 & 0.866 & 0.037 \\ -0.130 & 0.190 & 0.583 \end{bmatrix}$$

$$D = \begin{bmatrix} 0.032 & -0.042 & 0.010 \\ -0.018 & 0.045 & -0.046 \\ 0.031 & 0.004 & 0.035 \end{bmatrix}$$



Andrew S. Glassner is a member of the research staff at the Xerox Palo Alto Research Center, where he develops algorithms for image synthesis, modeling, and animation. He also lectures and writes on computer graphics for both technical and popular audiences. His current research interests include realistic image synthesis and modeling, computer-assisted animation, repeating geometric patterns, and graphical programming techniques. He is also interested in other uses of computers for enhancing and supporting creativity, including music, multimedia displays, and interactive fiction, both verbal and visual.

Glassner received the BS in computer science from Case Western Reserve University in 1984, and the MS and PhD in computer science from the University of North Carolina at Chapel Hill in 1987 and 1988.

Glassner can be reached at Xerox PARC, 3333 Coyote Hill Road, Palo Alto, CA 94304.