# Picture Perfect *RGB* Rendering Using Spectral Prefiltering and Sharp Color Primaries

Greg Ward

Exponent - Failure Analysis Assoc.

Elena Eydelberg-Vileshin

Stanford University

Presented at the 13th Eurographics Workshop on Rendering in Pisa, Italy, June 2002

# Talk Overview

1. Color Rendering Techniques
2. Getting the Most Out of *RGB*
   a) Spectral prefiltering
   b) The von Kries white point transform
3. Three Tristimulus Spaces
4. Experimental Results
5. Conclusions

## 1. A Brief Comparison of Color Rendering Techniques

- Spectral Rendering
  - ✓N spectrally pure samples
- Component Rendering
  - ✓M vector basis functions
- *RGB* (Tristimulus) Rendering
  - ✓Tristimulus value calculations

## Spectral Rendering

1. Divide visible spectrum into N wavelength samples
2. Process spectral samples separately throughout rendering calculation
3. Compute final display color using CIE color matching functions and standard transformations

## Component Rendering

[Peercy '93 SIGGRAPH]

1. Divide visible spectrum into M vector bases using component analysis
2. Process colors using MxM matrix multiplication at each interaction
3. Compute final display color with 3xM matrix transform

## *RGB* (Tristimulus) Rendering

1. Precompute tristimulus values
2. Process 3 samples separately throughout rendering calculation
3. Compute final display color with 3x3 matrix transform (if necessary)

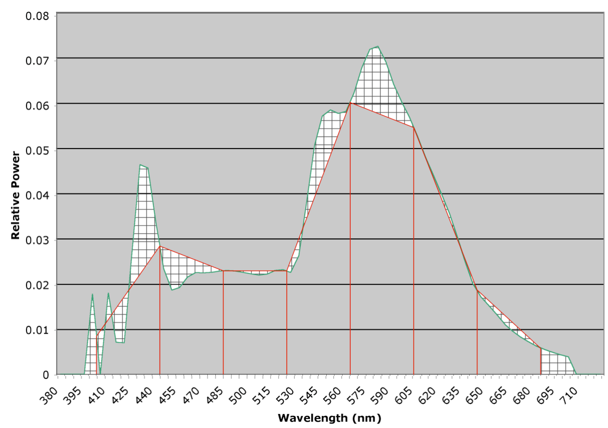
## Rendering Cost Comparison

| | Pre-processing | Multiplies / Interaction | Post-processing |
|---|---|---|---|
| Spectral | None | N ($N \geq 9$) | N multiplies per pixel |
| Component | Vector analysis | MxM ($M \geq 3$) | $3 \times M$ per pixel |
| *RGB* | Little or none | 3 | 0 to 9 per pixel |

## Strengths and Weaknesses

| | Strengths | Weaknesses |
|---|---|---|
| Spectral | Potential accuracy | Cost, aliasing, data mixing |
| Component | Optimizes cost/benefit | Preprocessing requirements |
| *RGB* | Fast, widely supported | Limited accuracy |

# Spectral Aliasing




[Meyer88] suffers worse with only 4 samples

# The Data Mixing Problem

- Typical situation:
  - Illuminants known to 5 nm resolution
  - Some reflectances known to 10 nm
  - Other reflectances given as tristimulus
- Two alternatives:

  A. Reduce all spectra to lowest resolution

  B. Interpolate/synthesize spectra [Smits '99]

## 2. Getting the Most Out of *RGB*

A. How Does *RGB* Rendering Work and When Does It Not?
B. Can *RGB* Accuracy Be Improved?
C. Useful Observations
D. Spectral Prefiltering
E. The von Kries White Point Transform

## Status Quo Rendering

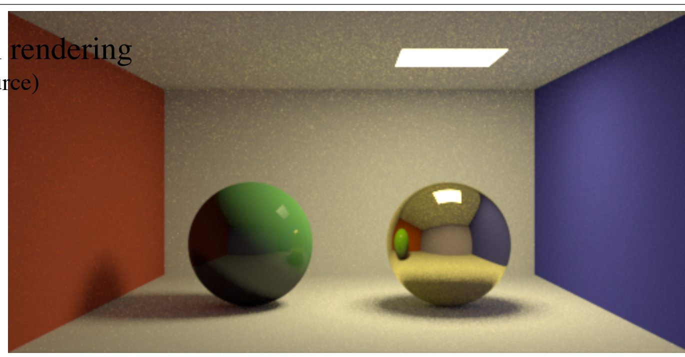
- White Light Sources
  - E.g., (R,G,B)=(1,1,1)
- *RGB* material colors obtained by dubious means
  - E.g., “That looks pretty good.”
    - ✓This actually works for fictional scenes!
- Color correction with ICC profile if at all

# When Does RGB Rendering Normally Fail?

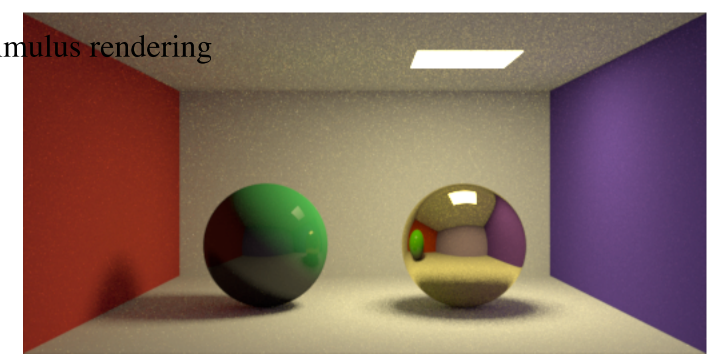
- When you start with measured colors
- When you want to simulate color appearance under another illuminant
- When your illuminant and surface spectra have sharp peaks and valleys

The Result: Wrong COLORS!

Full spectral rendering
(Fluorescent source)



Naïve tristimulus rendering
(CIE *XYZ*)



## Given Its Predominance, Can We Improve *RGB* Rendering?

- Identify and minimize sources of error
  - Source-surface interactions
  - Choice of rendering primaries
- Overcome ignorance and inertia
  - Many people render in *RGB* without really understanding what it means
  - White-balance problem scares casual users away from colored illuminants

## A Few Useful Observations

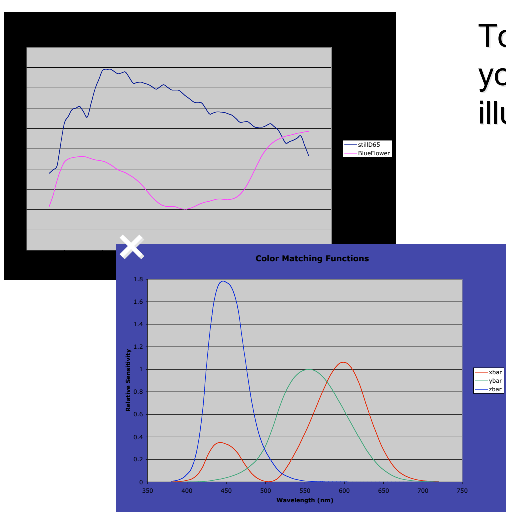
1. Direct illumination is the first order in any rendering calculation
2. Most scenes contain a single, dominant illuminant spectrum
3. Scenes with mixed illuminants will have a color cast regardless

Conclusion: Optimize for the Direct→Diffuse Case

# Picture Perfect *RGB* Rendering

1. Identify dominant illuminant spectrum
   a) Prefilter material spectra to obtain tristimulus colors for rendering
   b) Adjust source colors appropriately
2. Perform tristimulus (*RGB*) rendering
3. Apply white balance transform and convert pixels to display color space

# Spectral Prefiltering




To obtain a tristimulus color, you *must* know the illuminant spectrum

$$X = \int I(\lambda)\,\rho(\lambda)\,\bar{x}(\lambda)\,d\lambda$$

$$Y = \int I(\lambda)\,\rho(\lambda)\,\bar{y}(\lambda)\,d\lambda$$

$$Z = \int I(\lambda)\,\rho(\lambda)\,\bar{z}(\lambda)\,d\lambda$$

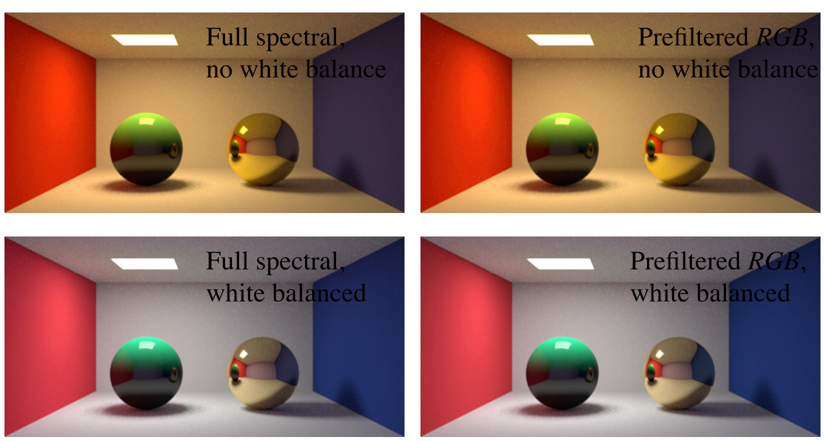
*XYZ* may then be transformed by 3×3 matrix to any linear tristimulus space (e.g., *sRGB*)

# Prefiltering vs. Full Spectral Rendering

+ Prefiltering performed once per material vs. every rendering interaction
+ Spectral aliasing and data mixing problems disappear with prefiltering
- However, mixed illuminants and interreflections not computed exactly

Regardless which technique you use, remember to apply white balance to result!

# Quick Comparison



## The von Kries Transform for Chromatic Adaptation

The von Kries transform takes colors from absolute $XYZ$ to adapted equiv. $XYZ'$

Where:

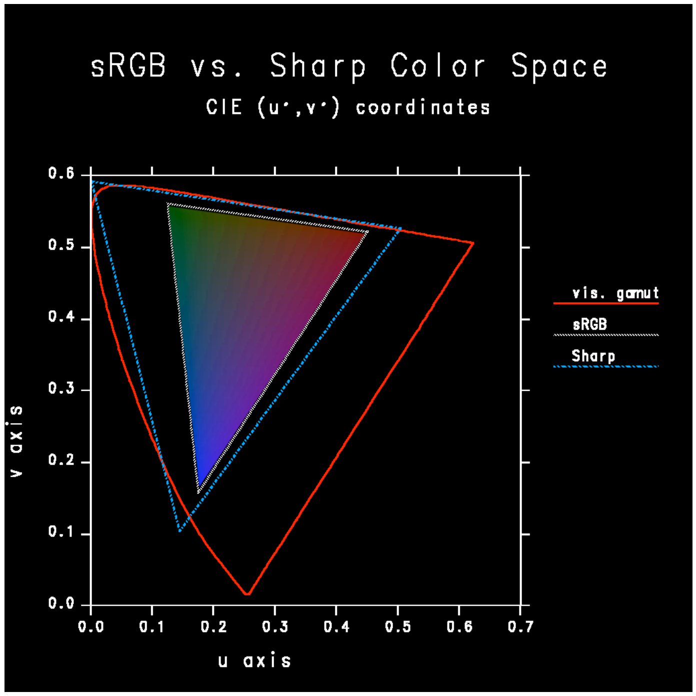
$$\begin{bmatrix} R_w' \\ G_w' \\ B_w' \end{bmatrix} = \mathbf{M}_c \begin{bmatrix} X_w' \\ Y_w' \\ Z_w' \end{bmatrix} \qquad \begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix} = \mathbf{M}_c \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix}$$

Display white point    Scene white point

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \mathbf{M}_C^{-1} \begin{bmatrix} \frac{R_w'}{R_w} & 0 & 0 \\ 0 & \frac{G_w'}{G_w} & 0 \\ 0 & 0 & \frac{B_w'}{B_w} \end{bmatrix} \mathbf{M}_C \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

## Chromatic Adaptation Matrix

- The matrix $\mathbf{M}_C$ transforms *XYZ* into an “adaptation color space”
- Finding the optimal CAM is an under-constrained problem -- many candidates have been suggested
- “Sharper” color spaces tend to perform better, and seem to be more “plausible” [Susstrunk2001] [Finlayson2001]




# 3. Three Tristimulus Spaces for Color Rendering

- CIE *XYZ*
  - Covers visible gamut with positive values
  - Well-tested standard for color-matching
- *sRGB*
  - Common standard for image encoding
  - Matches typical CRT display primaries
- Sharp *RGB*
  - Developed for chromatic adaptation

# *XYZ* Rendering Process

1. Apply prefiltering equation to get absolute *XYZ* colors for each material
   a) Divide materials by illuminant:
   $$X_m{}^* = \frac{X_m}{X_w}, \quad Y_m{}^* = \frac{Y_m}{Y_w}, \quad Z_m{}^* = \frac{Z_m}{Z_w}$$
   b) Use absolute *XYZ* colors for sources
2. Render using tristimulus method
3. Finish w/ CAM and display conversion

# *sRGB* Rendering Process

1. Perform prefiltering and von Kries transform on material colors
   a) Model dominant light sources as neutral
   b) For spectrally distinct light sources use:
   $$R_s{}^* = \frac{R_s}{R_w}, \quad G_s{}^* = \frac{G_s}{G_w}, \quad B_s{}^* = \frac{B_s}{B_w}$$
2. Render using tristimulus method
3. Resultant image is *sRGB*

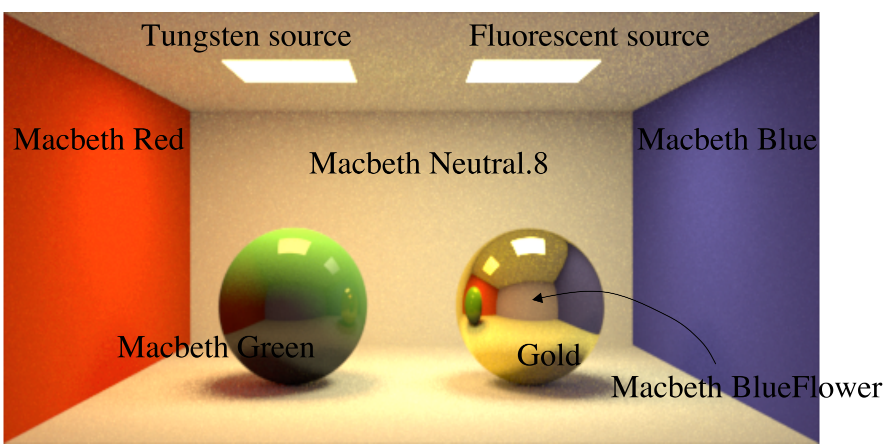
# Sharp *RGB* Rendering Process

1. Prefilter material colors and apply von Kries transform to Sharp *RGB* space:

$$\begin{bmatrix} R_m{}^* \\ G_m{}^* \\ B_m{}^* \end{bmatrix} = \begin{bmatrix} \frac{1}{R_w} & 0 & 0 \\ 0 & \frac{1}{G_w} & 0 \\ 0 & 0 & \frac{1}{B_w} \end{bmatrix} \mathbf{M}_{Sharp} \begin{bmatrix} X_m \\ Y_m \\ Z_m \end{bmatrix}$$

2. Render using tristimulus method
3. Finish up CAM and convert to display

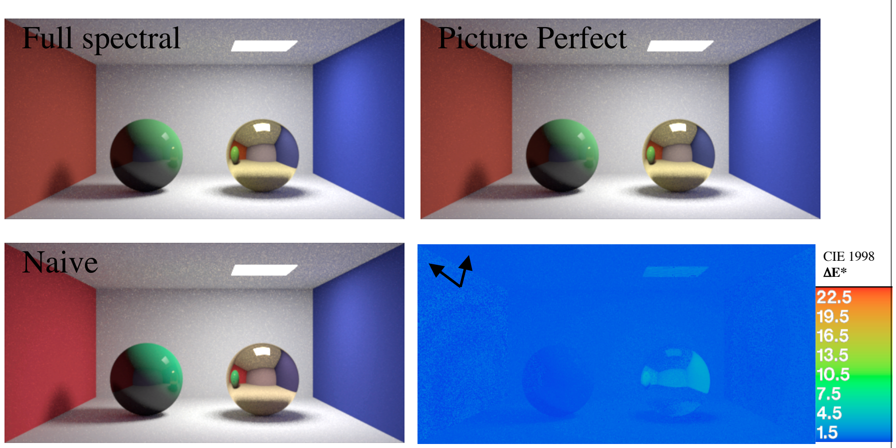
# Our Experimental Test Scene



# 4. Experimental Results

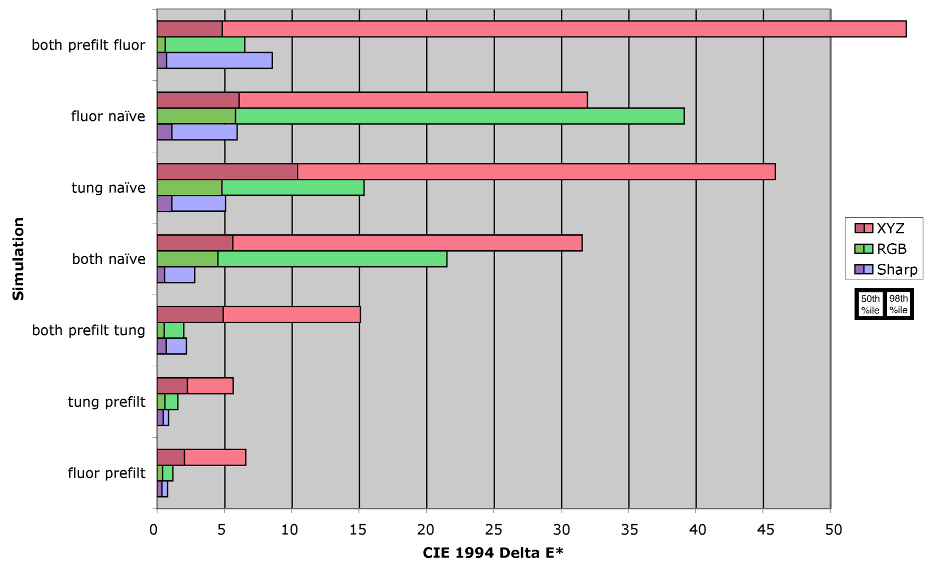
- Three lighting conditions
  - Single 2856°K tungsten light source
  - Single cool white fluorescent light source
  - Both light sources (tungsten & fluorescent)
- Three rendering methods
  - Naïve *RGB* (assumes equal-energy white)
  - Picture Perfect *RGB*
  - Full spectral rendering (380 to 720 nm / 69 samp.)
- Three color spaces (*XYZ*, *sRGB*, Sharp *RGB*)

# Example Comparison (*sRGB*)




CIE 1998 ΔE* of 5 or above is visible in side-by-side comparisons

## ΔE* Error Percentiles for All Experiments




# Results Summary

- Prefiltering has ~1/6 the error of naïve rendering for single dominant illuminant
- Prefiltering errors similar to naïve in scenes with strongly mixed illuminants
- CIE *XYZ* color space has 3 times the rendering errors of *sRGB* on average
- Sharp *RGB* rendering space reduces errors to 1/3 that of *sRGB* on average

## 5. Conclusions

- Prefiltering is simple and practically free
- Avoids aliasing and data mixing problems of full spectral rendering
- Error comparable to 3 component rendering [Peercy93] at 1/3 the cost
- Mixed illuminants and specular reflections no worse than naïve *RGB*

## *Radiance* Details

- Be sure to note different color space used in *Radiance* materials file
- Use “vinfo” to edit picture color space:

```
PRIMARIES= .6898 .3206 .0736 .9003 .1166 .0374 .3333 .3333
```

- The **ra_xyze** or **pcond** program may then be used to convert color space