Hyperspectral Rendering in *Radiance*

2024 Workshop Tutorial

Greg Ward, Anyhere Software

Tutorial Outline

1. Introduction and context
2. New *Radiance* primitives
3. New rendering parameters
4. *Radiance* tools for hyperspectral
5. Example scene with comparisons
6. Spectral sky data + matrix methods

What Is Hyperspectral?

- Traditional rendering uses RGB to represent color values
  - 3x3 matrix to CIE XYZ colorimetry
- Hyperspectral rendering divides spectrum into >3 (typically ≥9) bands
- Spectral limits may exceed those of human visual response (IR, UV)
- More accurate with new applications

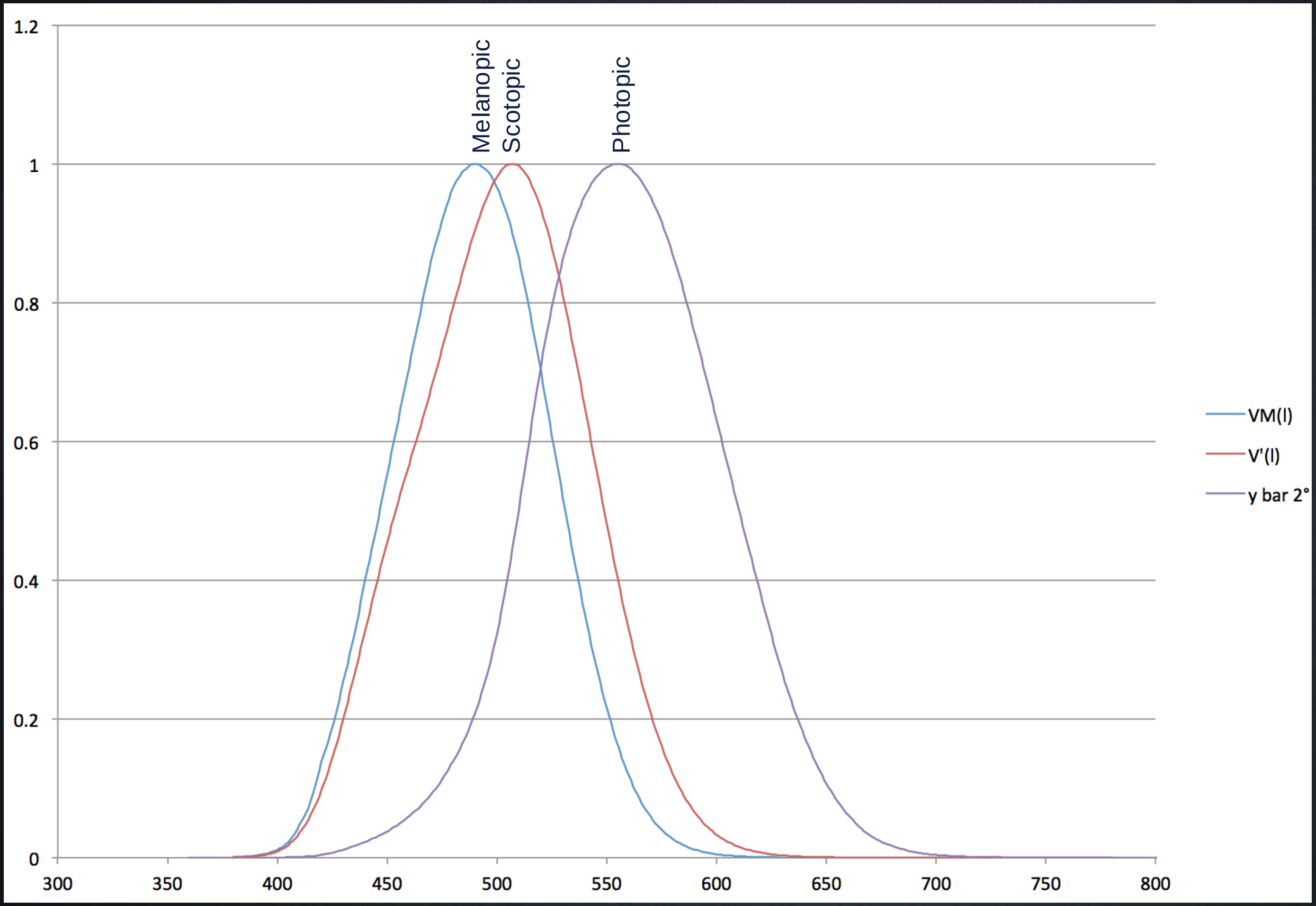
Current Impetus

- Many researchers and practitioners are concerned with melanopic response
  - melanopic ≠ photometric sensitivity
  - distrust of approximations from RGB
- Active research in terrestrial solar and sky spectra
- Current methods mostly retrace rays using *Radiance* with multiple passes

Advantages of Native Hyperspectral Rendering

- More efficient, since rays do not need to be stored or retraced
- Retracing rays also leads to chromatic noise due to non-repeating ray trees
- New HSR (hyperspectral radiance) image format introduced
  - new tools for HSR display and calcs

Sensitivity Curves



Radiance System Design

- Scene format has 3-component colors baked into material primitive types
- Initially contemplated redesign of scene description based on "Materials and Geometry Format" (MGF)
  - MGF colors cover visible spectrum only
  - Would have required full rewrite of existing rendering code

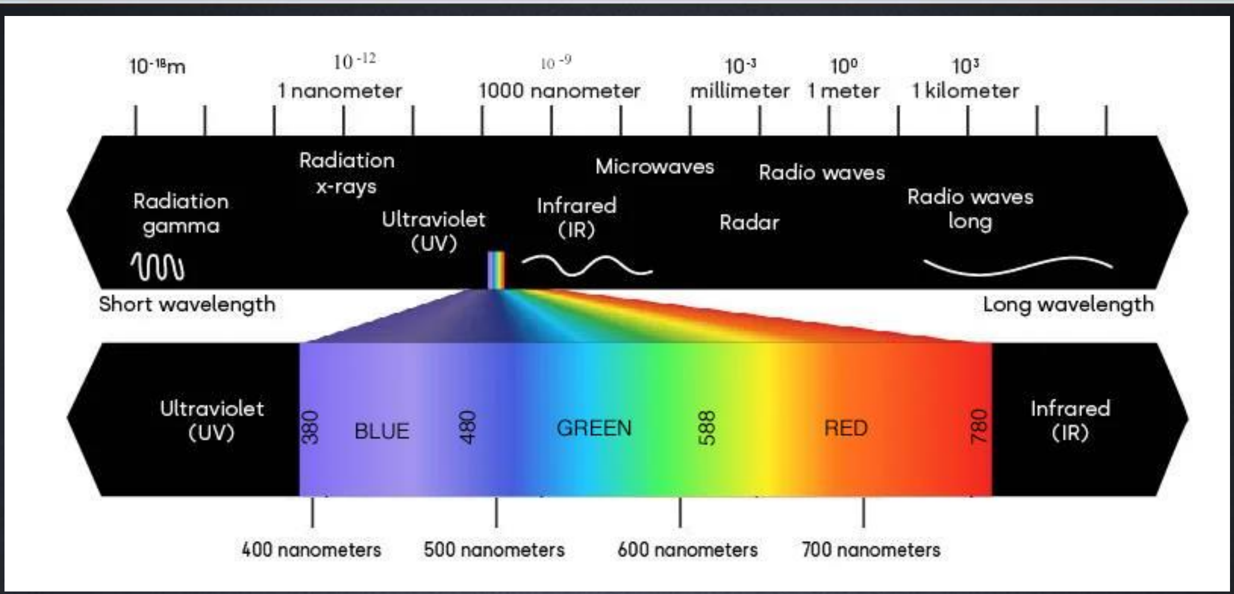
Hyperspectral Extensions

- While *Radiance* material types only support 3 spectral samples, new pattern primitives are multi-spectral:
  - *spectrum* is a basic spectral color
  - *specfile* takes static spectrum from file
  - *specfunc* is dynamic and procedural
  - *specdata* is dynamic and data-driven
  - *specpict* maps a hyperspectral image

How Spectral Patterns Work

- Depending on the primitive it modifies, a spectral pattern typically multiplies a material’s RGB color to determine reflectance or transmittance
- Multiplication rules are based on “partition” wavelengths that define RGB bands

Partitioning the Spectrum



- Spectrum is partitioned into wavelengths of 380-480nm for BLUE, 480-588nm for GREEN, and 588-780nm for RED for purposes of multiplication between material colors and spectral patterns
- Extrema may be adjusted to cover UV and/or IR if desired, which does not alter the range of GREEN, only BLUE and RED
- The number of spectral samples may be set from 3 to MAXCSAMP (24 in 6.0a prerelease)

Example Spectrum

```
void spectrum gold_spec
0
0
82 380 775 0.389133964 0.389133964 0.389255302 0.389980986 0.391051245
   0.392138273 0.392908482 0.393032534 0.392429968 0.391351324
   0.390030615 0.388650157 0.387407915 0.386320218 0.385530031
   0.385510124 0.386040307 0.386026057 0.388298949 0.394215681
   0.404140535 0.419966115 0.443246002 0.473861019 0.511014174
   0.552643396 0.596304788 0.639685049 0.682955923 0.723645597
   0.759200442 0.789126545 0.813469177 0.832955943 0.848265499
   0.860485356 0.870549939 0.878925721 0.885276016 0.891159039
   0.896546999 0.901393946 0.905636686 0.909327028 0.912475235
   0.91521538 0.917680109 0.919998384 0.922297675 0.924966158
   0.927728576 0.930571538 0.933477024 0.936421086 0.939377245
   0.942439249 0.945588396 0.948563802 0.951319598 0.953828111
   0.956075841 0.958059262 0.959789105 0.961306963 0.96261608
   0.963753124 0.964748535 0.965627808 0.966413149 0.96712339
   0.967778124 0.968398351 0.968976934 0.969517604 0.970024269
   0.970500409 0.970949021 0.971372989 0.971775043 0.972157635

gold_spec metal gold_smat
0
0
5 1 1 1 1 0
```

Starting λ

Ending λ

Multiplies spectrum

Using mgf2rad for MGF Input

```
c BlueFlower =
        cspec 380 775 21.5045 26.9856 34.0215 39.5109 43.4297 \
                44.9115 45.5818 45.9004 46.184 46.1863 45.7261 \
                45.0791 44.5095 44.2579 43.9035 43.278 42.2302 \
                40.9632 39.6658 38.9282 38.1228 37.0968 \
                35.9171 34.4883 32.7533 30.6119 28.3245 \
                26.1489 24.2761 22.8377 21.9115 21.5285 \
                21.5611 21.4244 20.7545 20.2763 20.2588 \
                20.6814 21.4462 22.3776 23.1821 23.7876 \
                24.1932 24.6781 25.1617 25.2637 24.9932 \
                24.9185 25.2695 26.0736 27.4431 29.666 32.6391 \
                36.2101 40.1034 43.9313 47.2898 49.8959 \
                52.0471 53.5721 54.6501 55.4622 56.2143 \
                56.6269 57.0836 57.6529 57.9171 58.3157 \
                58.5715 58.5715 58.5715 58.5715 58.5715 \
                58.5715 58.5715 58.5715 58.5715 58.5715 \
                58.5715 58.5715
m blueflower_mat =
        c BlueFlower
        rd 0.248407
```

```
void spectrum BlueFlower*
0
0
42 380 770      0.86346 1.22498 1.66552 1.81700 1.84899
        1.84546 1.79865 1.77019 1.71704 1.61895 1.54698
        1.46607 1.35011 1.18334 1.01235 0.89850 0.86534
        0.84676 0.81383 0.84582 0.91473 0.96342 1.00082
        1.00929 1.00788 1.07468 1.25085 1.53240 1.83158
        2.04704 2.17311 2.24250 2.28319 2.32035 2.34693
        2.35210 2.35210 2.35210 2.35210 0.11384

BlueFlower* plastic blueflower_mat
0
0
5 0.248407 0.248407 0.248407 0.000000 0.000000
```

New Rendering Parameters

Supported by all renderers (**rpict**, **rvu**, **rtrace**, & **rcontrib**):

-cs *N*	# number of spectral samples (default is 3)
-cw *w1 w2*	# spectrum equally divided from w1 to w2 (nm)

Supported by **rpict** and **rtrace**:

-pRGB	# output RGB color space (default)
-pXYZ	# output XYZ color space
-pc *xr yr xg yg xb yb xw yw*	# custom color space

Supported by **rtrace** only:

-pY	# output single-channel photopic luminance
-pS	# output single-channel scotopic luminance
-pM	# output single-channel melanopic luminance
-co+	# output N-channel spectra (also **rtpict**)

Note: **rcontrib** (& **rfluxmtx**) always produce N channels based on **-cs** setting

New HSR Image Format

- HSR for "Hyperspectral Radiance" uses similar common-exponent encoding to RGBE and XYZE (i.e., HDR format)
- Can be produced by **rtrace**, **rcontrib/rfluxmtx**, or **rtpict**
- Can be used by *specpict* primitive
- HSR files can be processed by **pfilt**, **rcode2bmp, rcomb**, **rcrop**, and **rmtxop** tools

Example with **rtpict** & **rcomb**

Render a hyperspectral image with **rtpict** and anti-alias with **pfilt**:

```
rtpict -vf myview.vf -x 2048 -y 2048 -cs 18 -co+ specscene.oct \
       | pfilt -1 -x /2 -y /2 -r .6 > specim.hsr
```

Convert result to RGB image and melanpopic pixel array using **rcomb**:

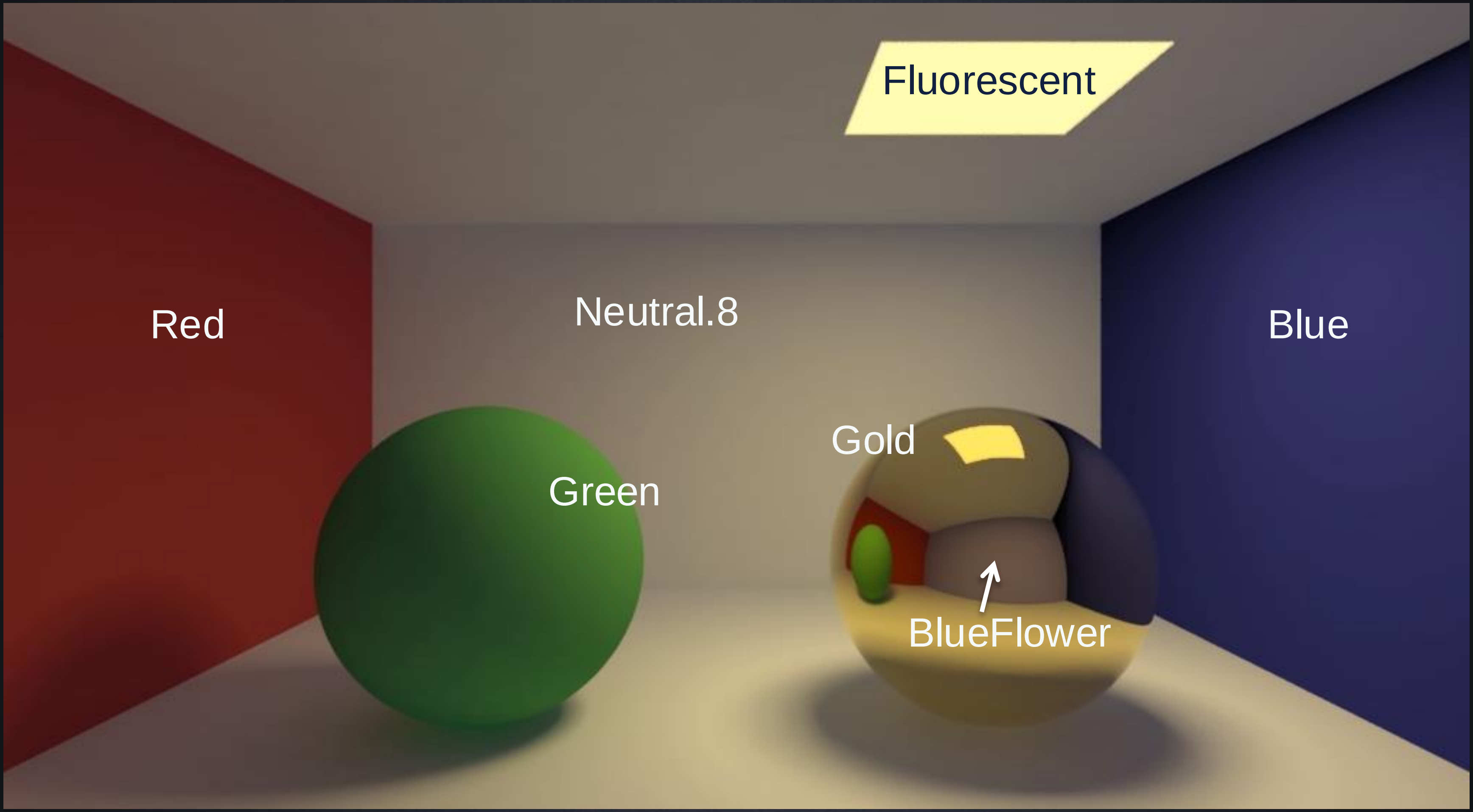
```
rcomb specim.hsr -c RGB -fc > rgbim.hdr
rcomb specim.hsr -c M -ff > melim.mtx
```

Tone-map result into 24-bit BMP image:

```
rcode2bmp specim.hsr
```

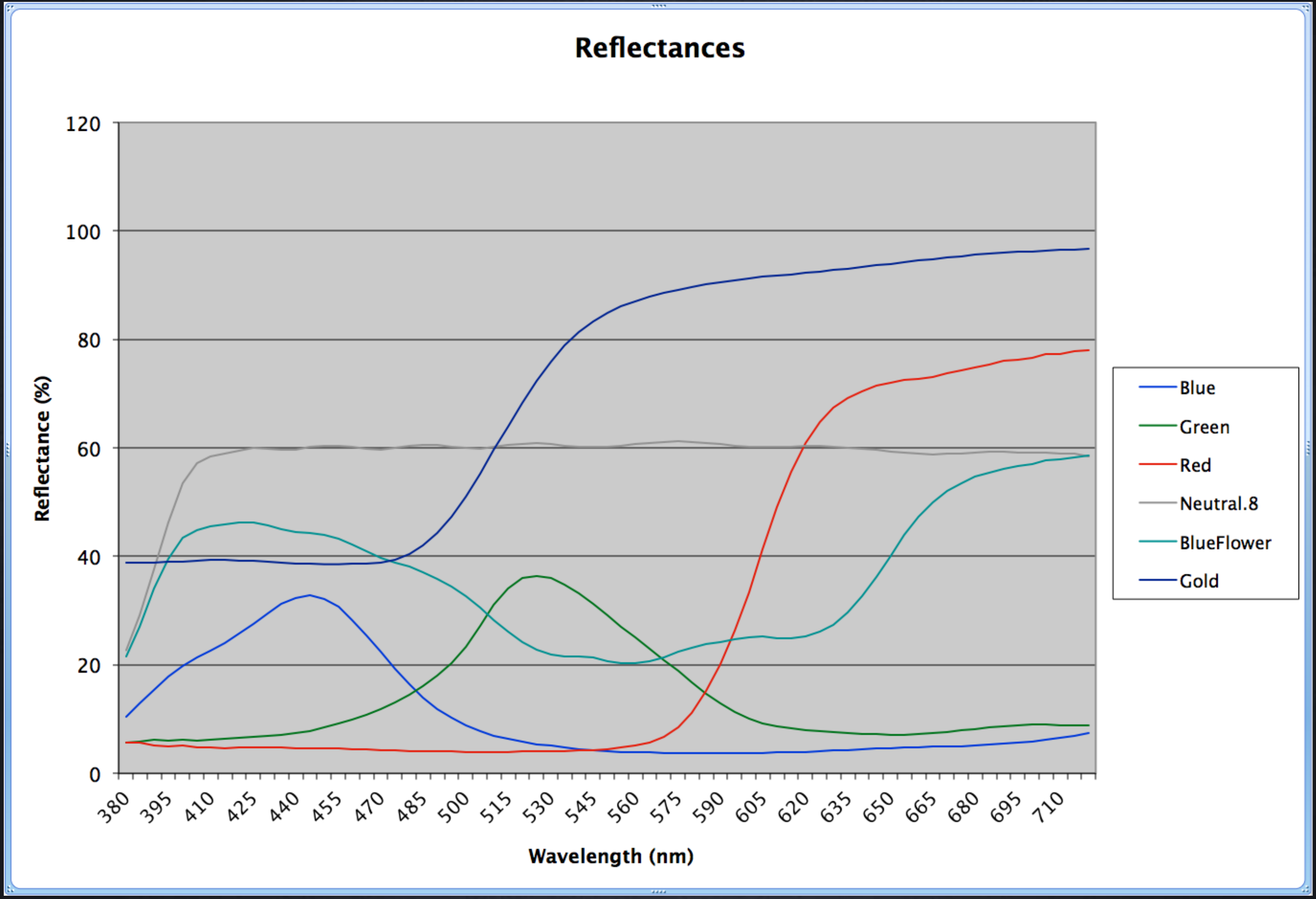
(open “specim.bmp”)

Example Spectral Scene

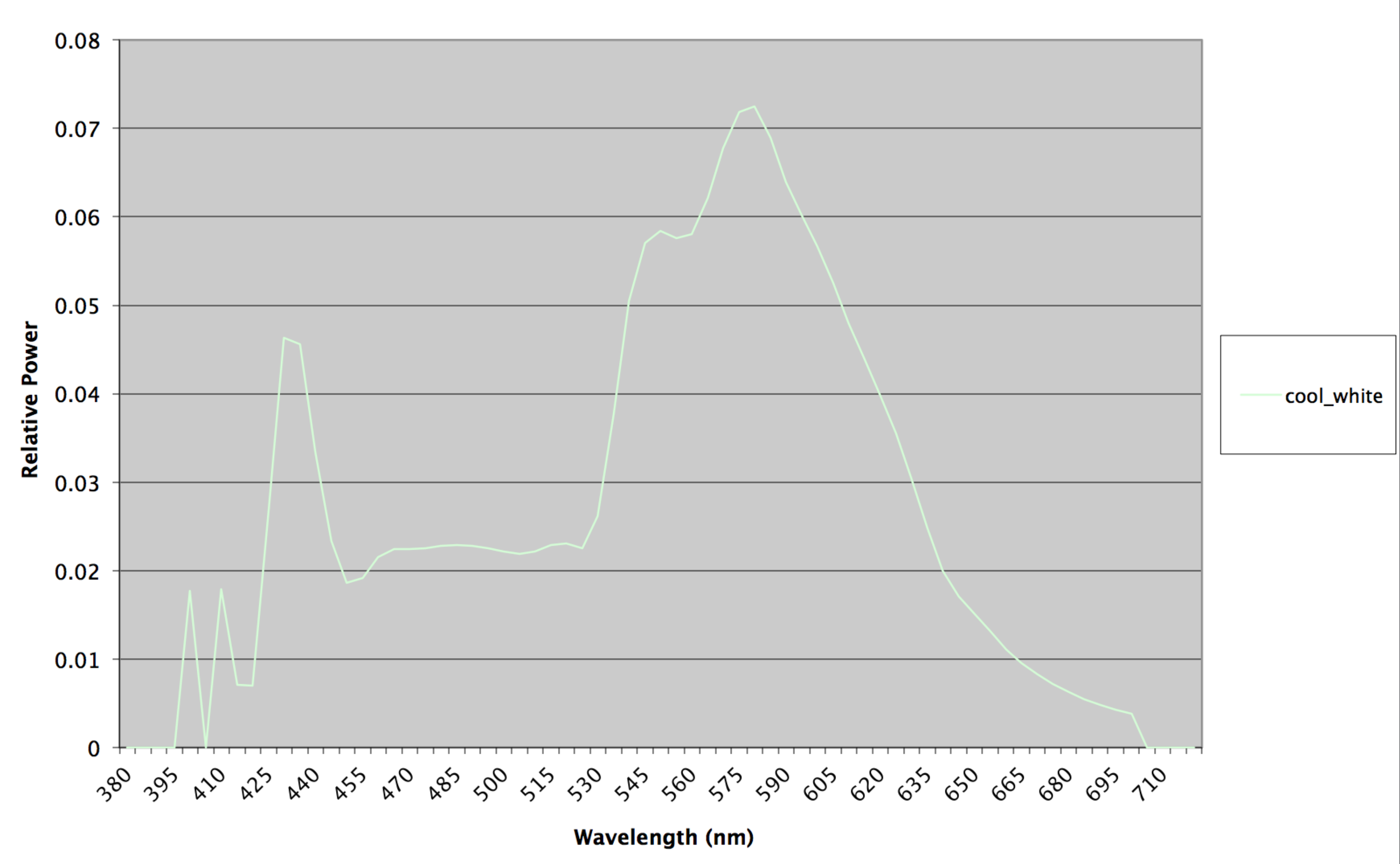


MacBeth Colors

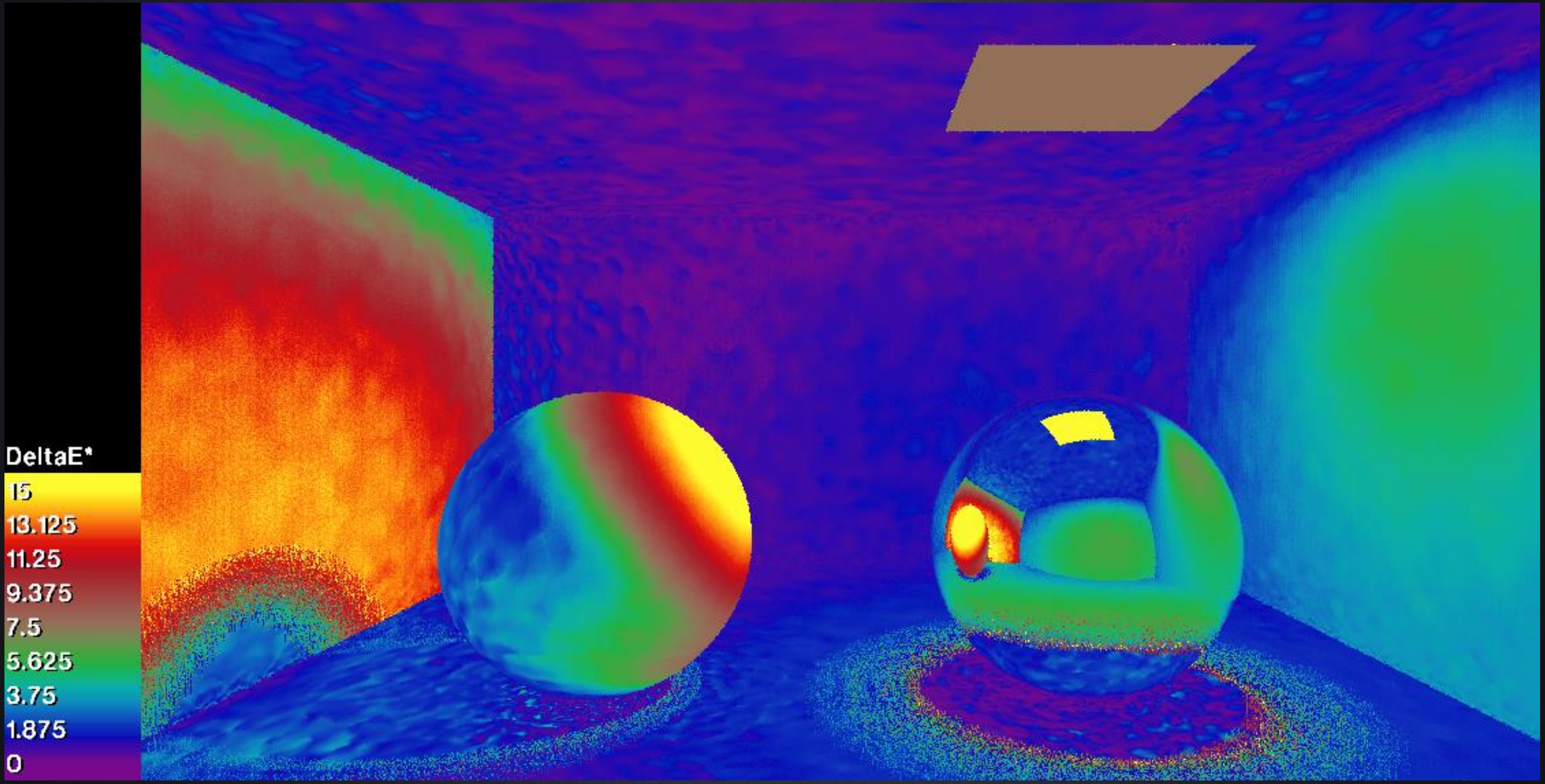
MacBeth Reflectance Spectra



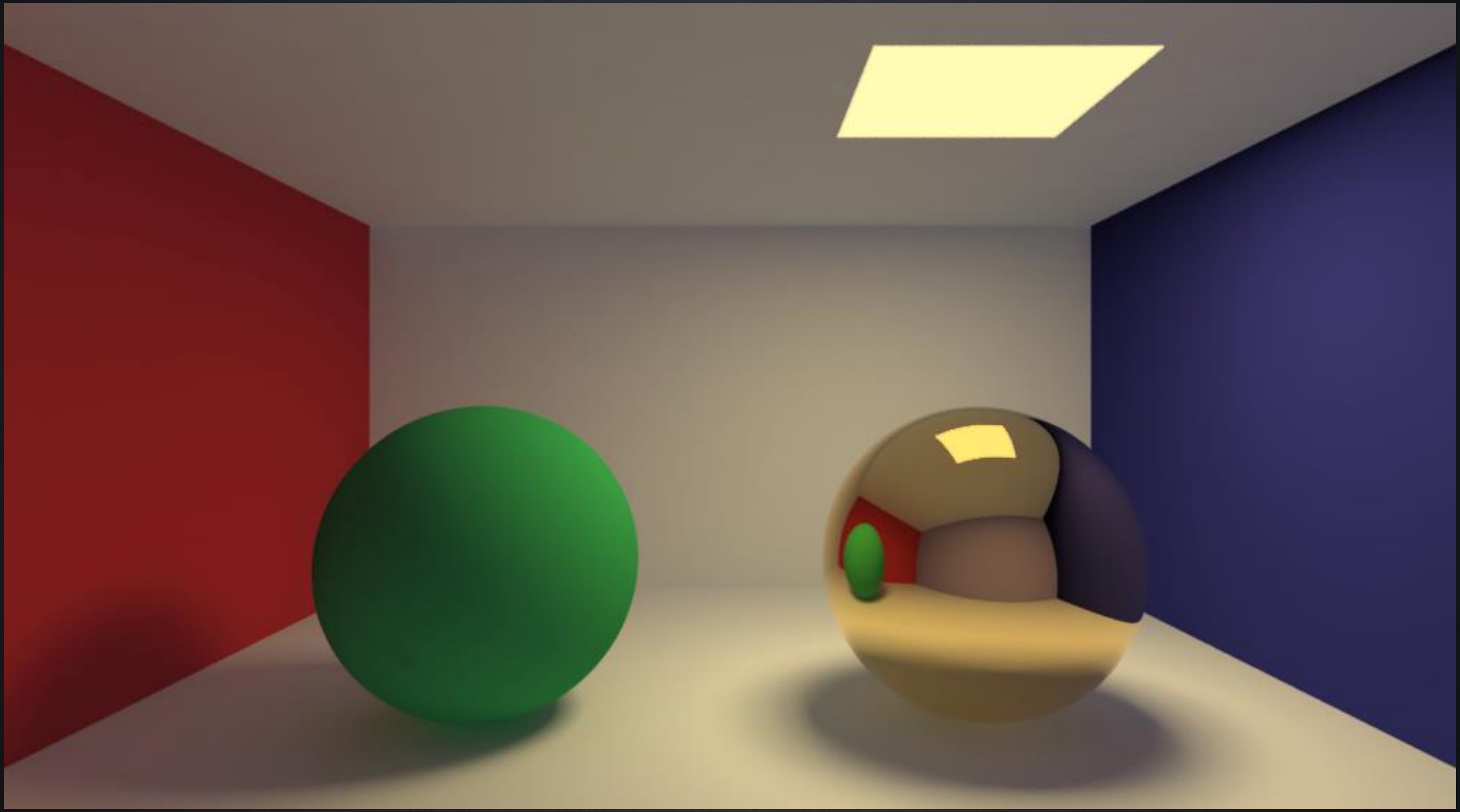
Fluorescent Lamp Spectrum



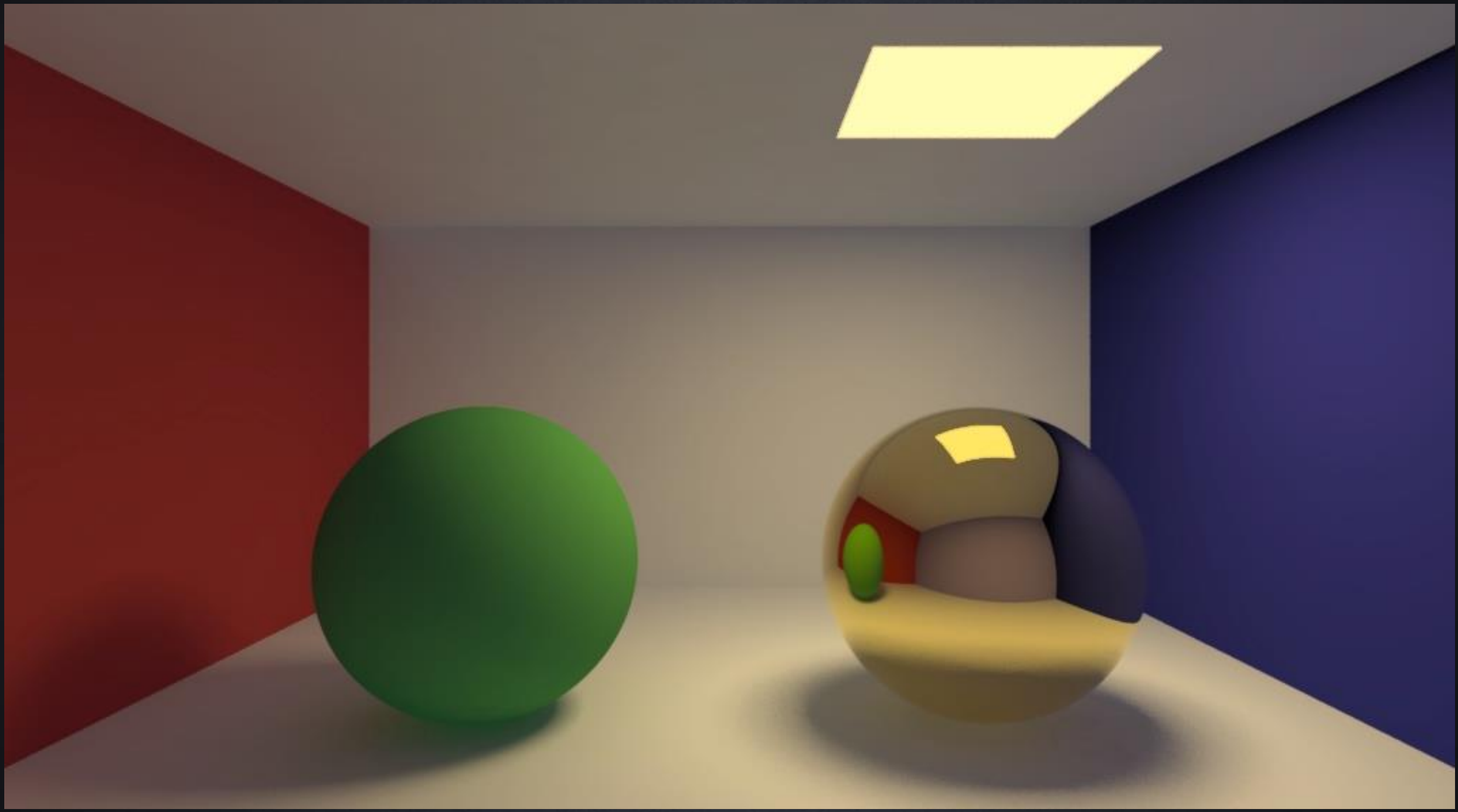
Error from Rendering in RGB



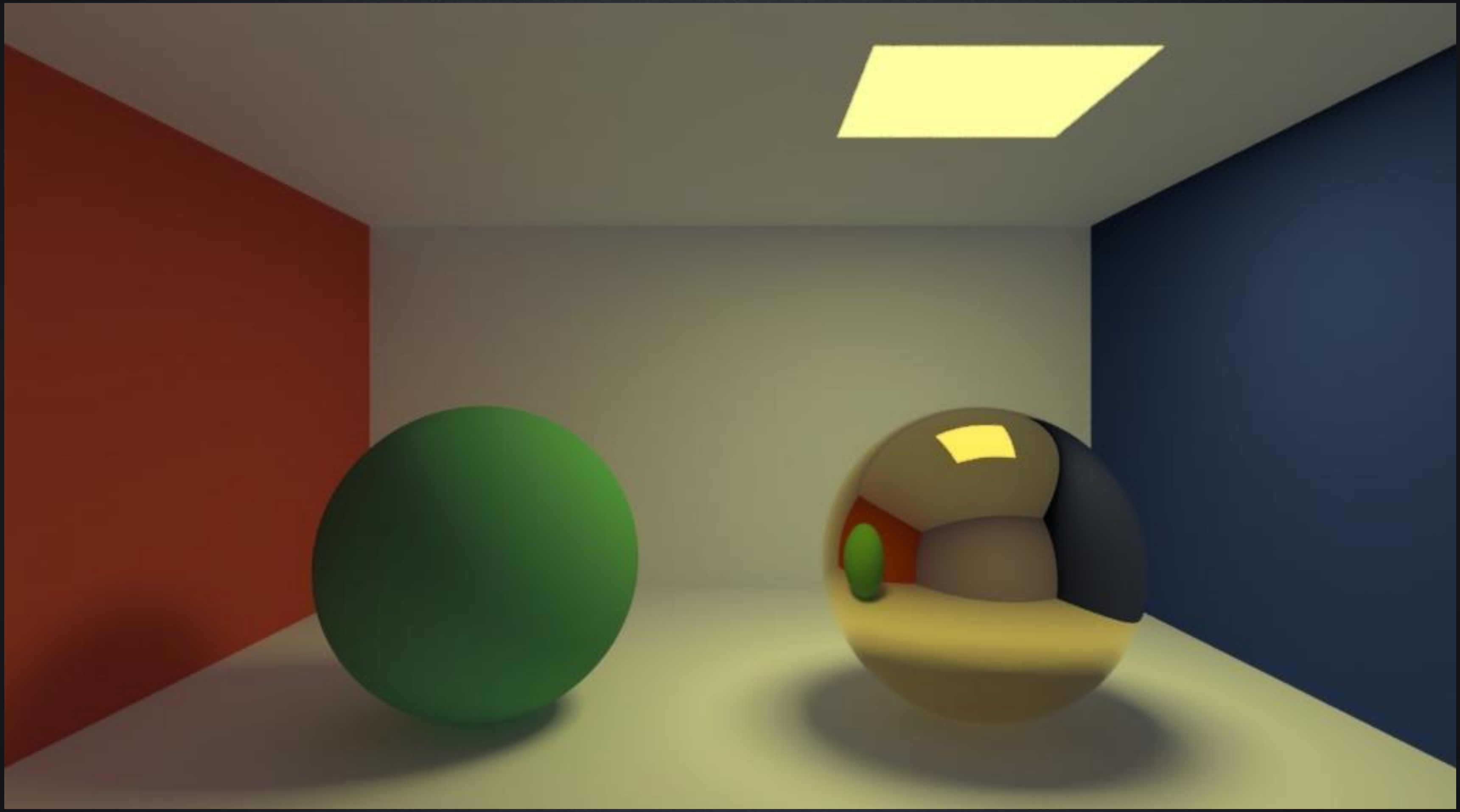
RGB Rendering



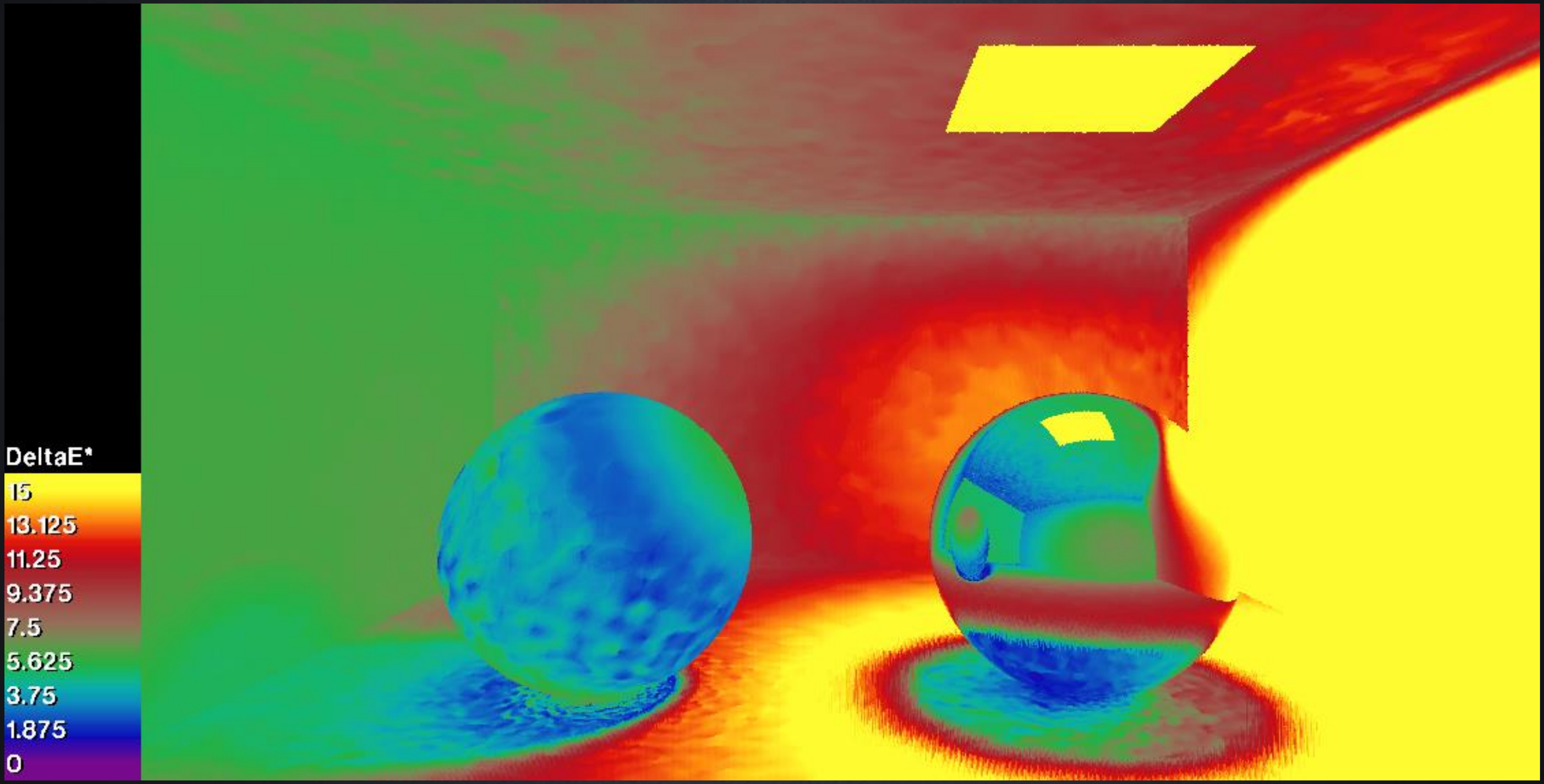
Reference Solution
(81 spectral samples)



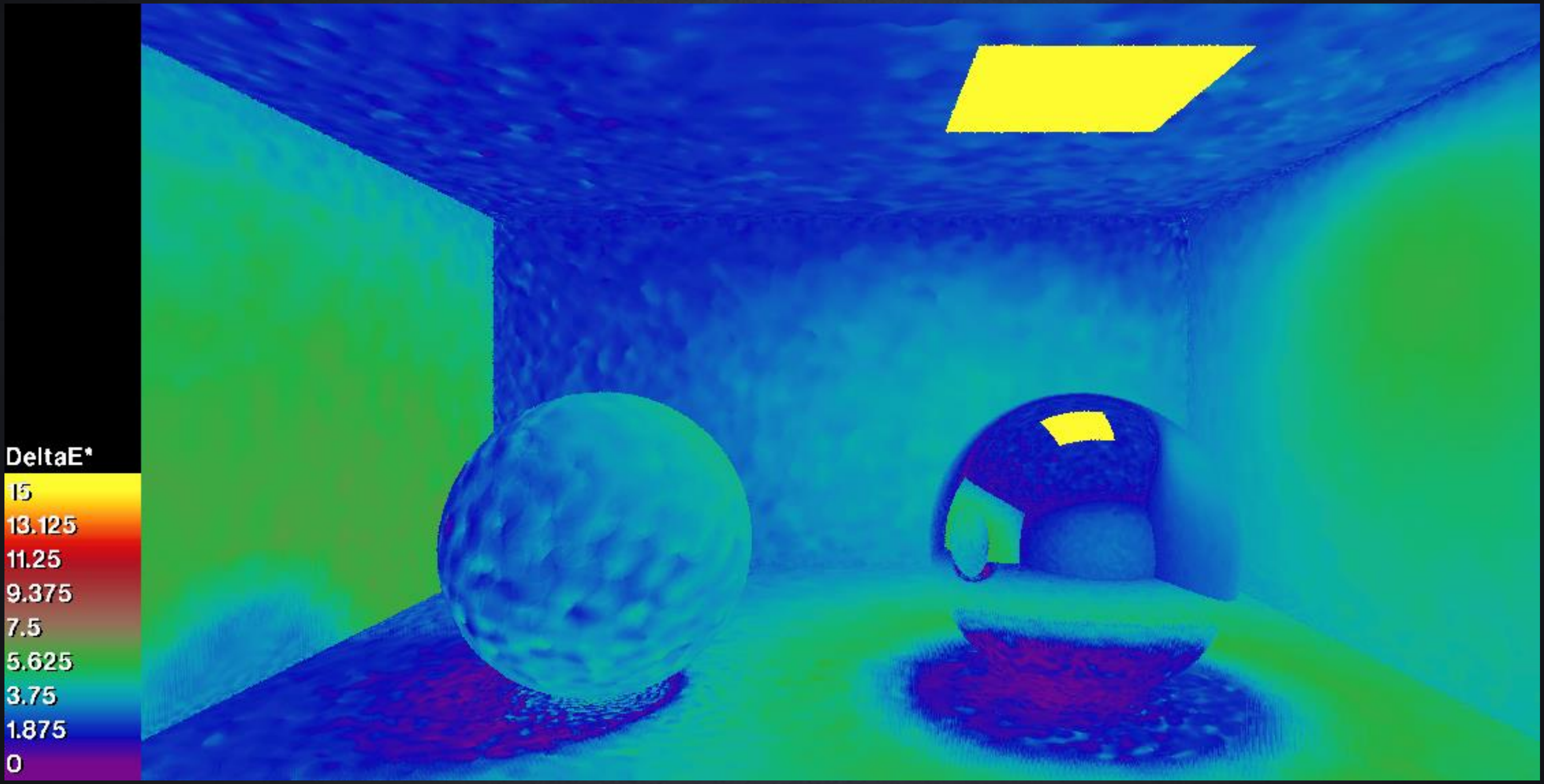
Rendering with 6 Spectral Samples



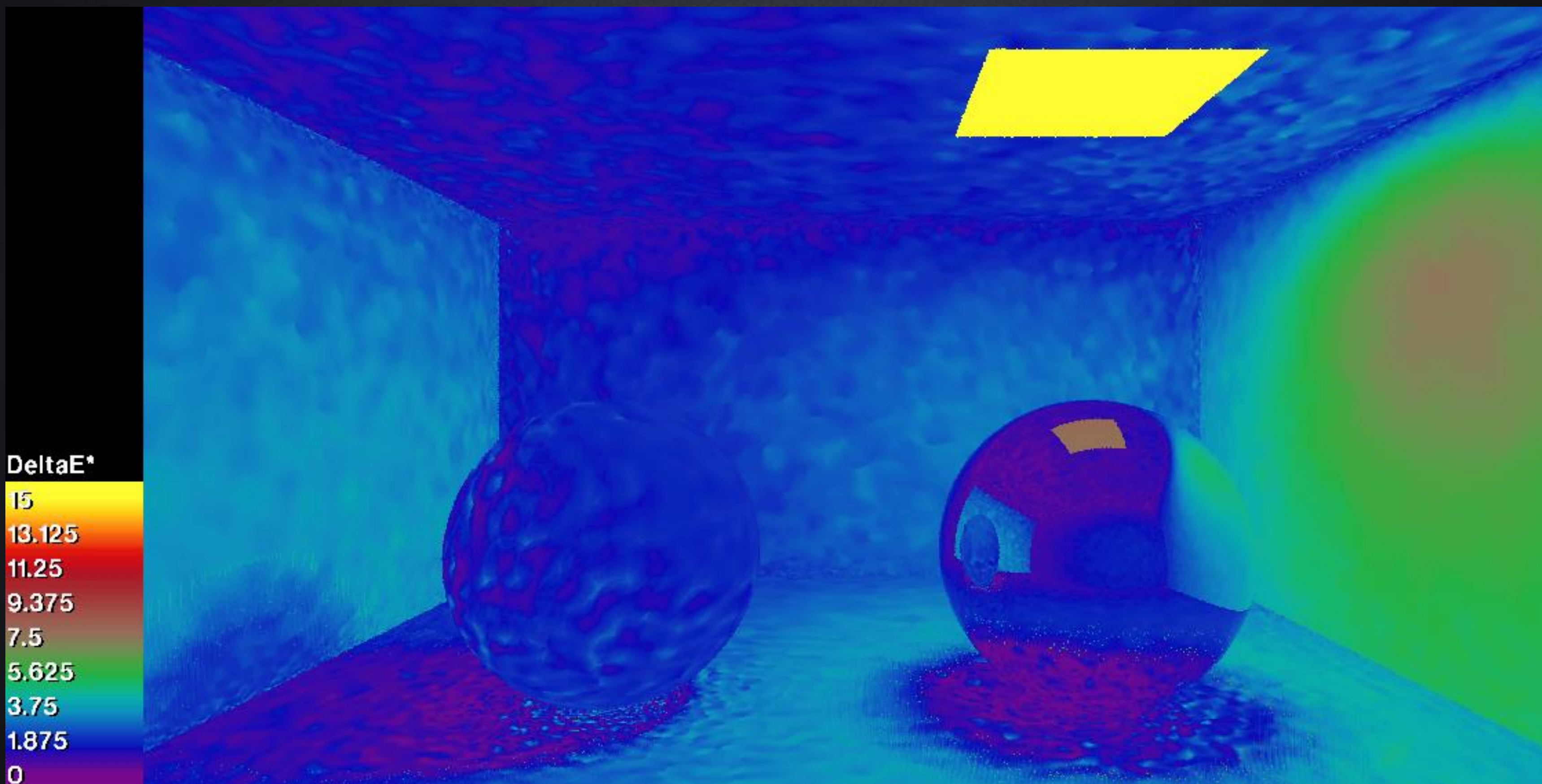
Error with 6 Spectral Samples



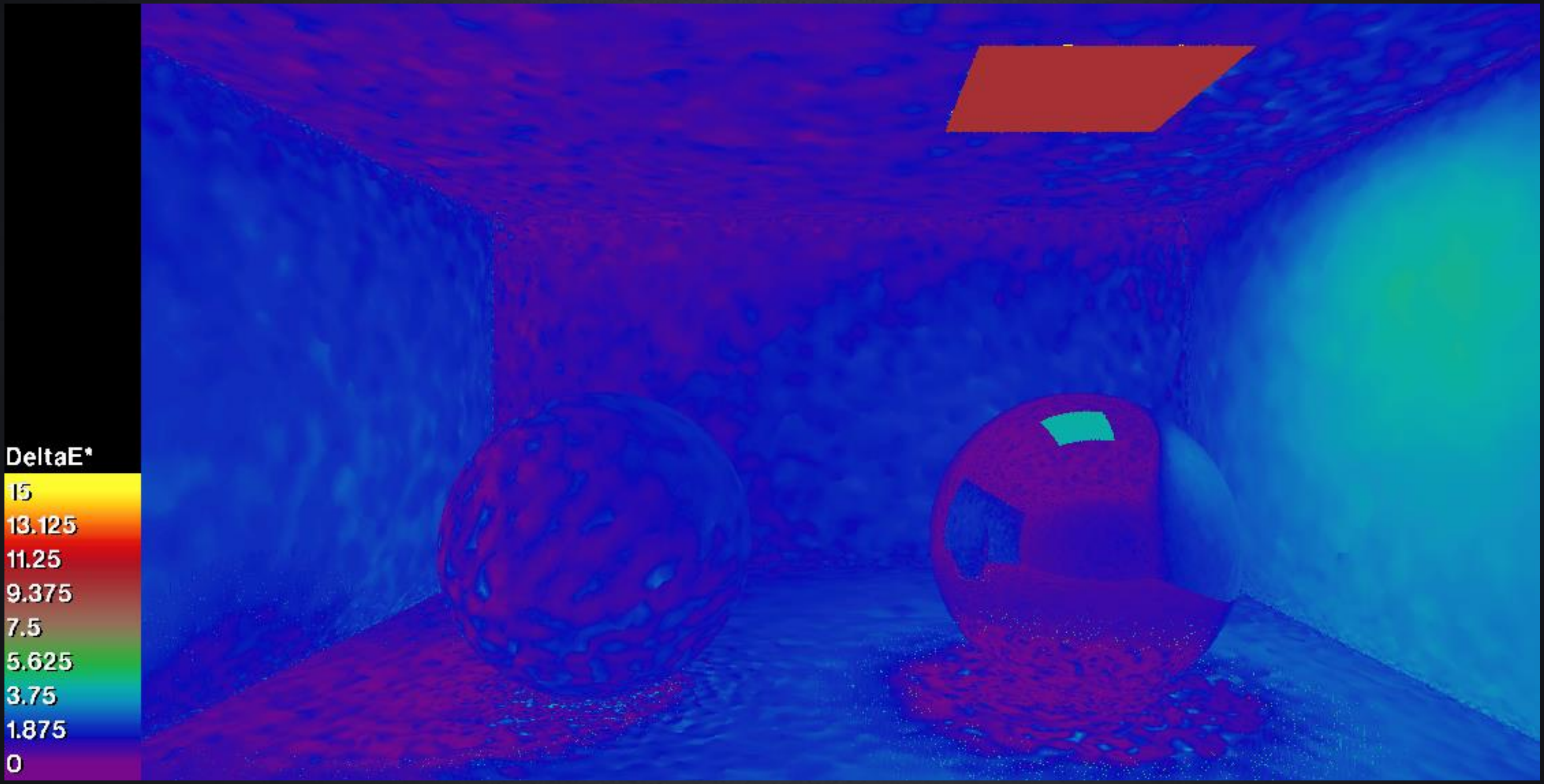
9 Spectral Sample Error



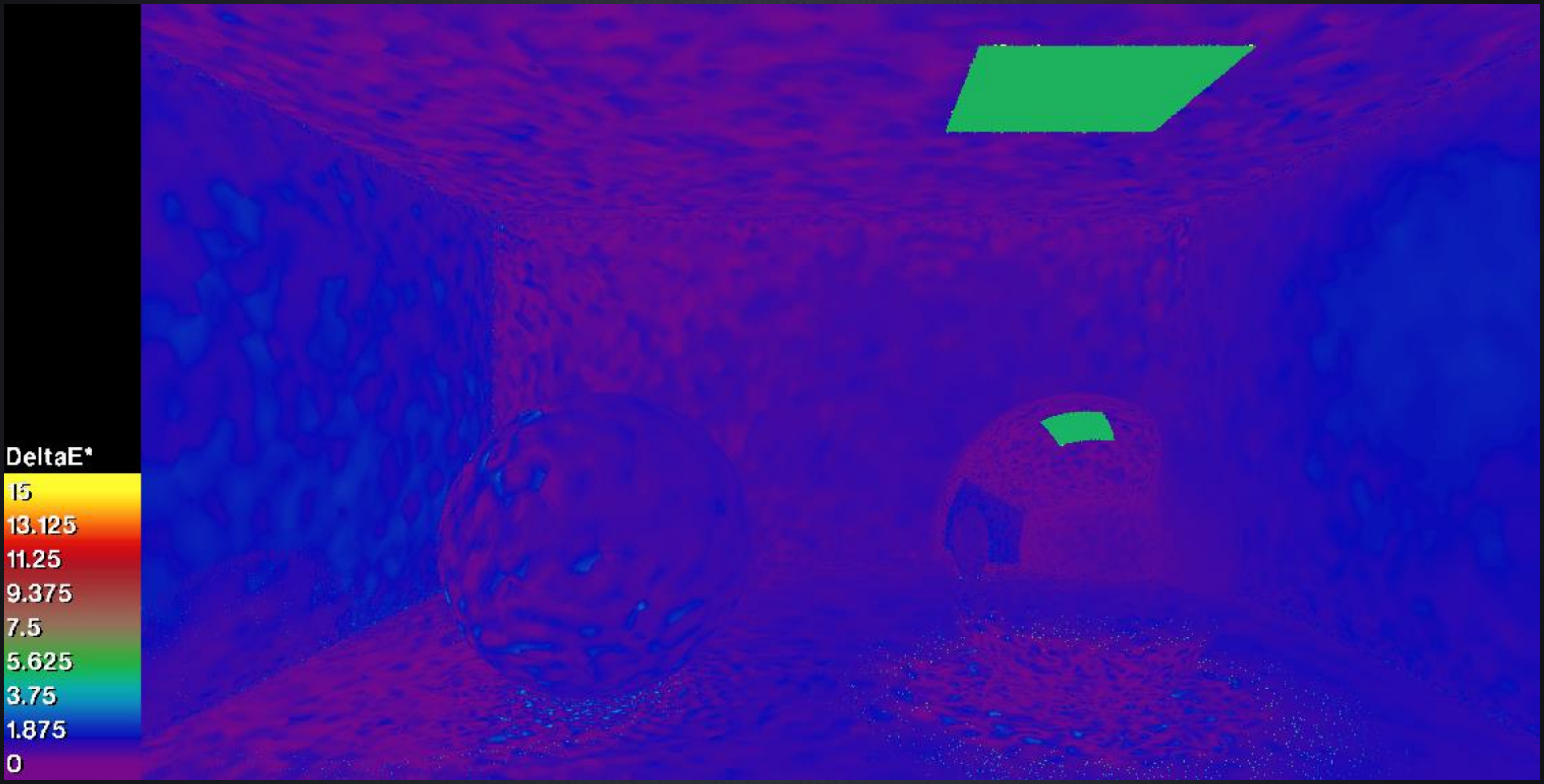
12 Spectral Sample Error



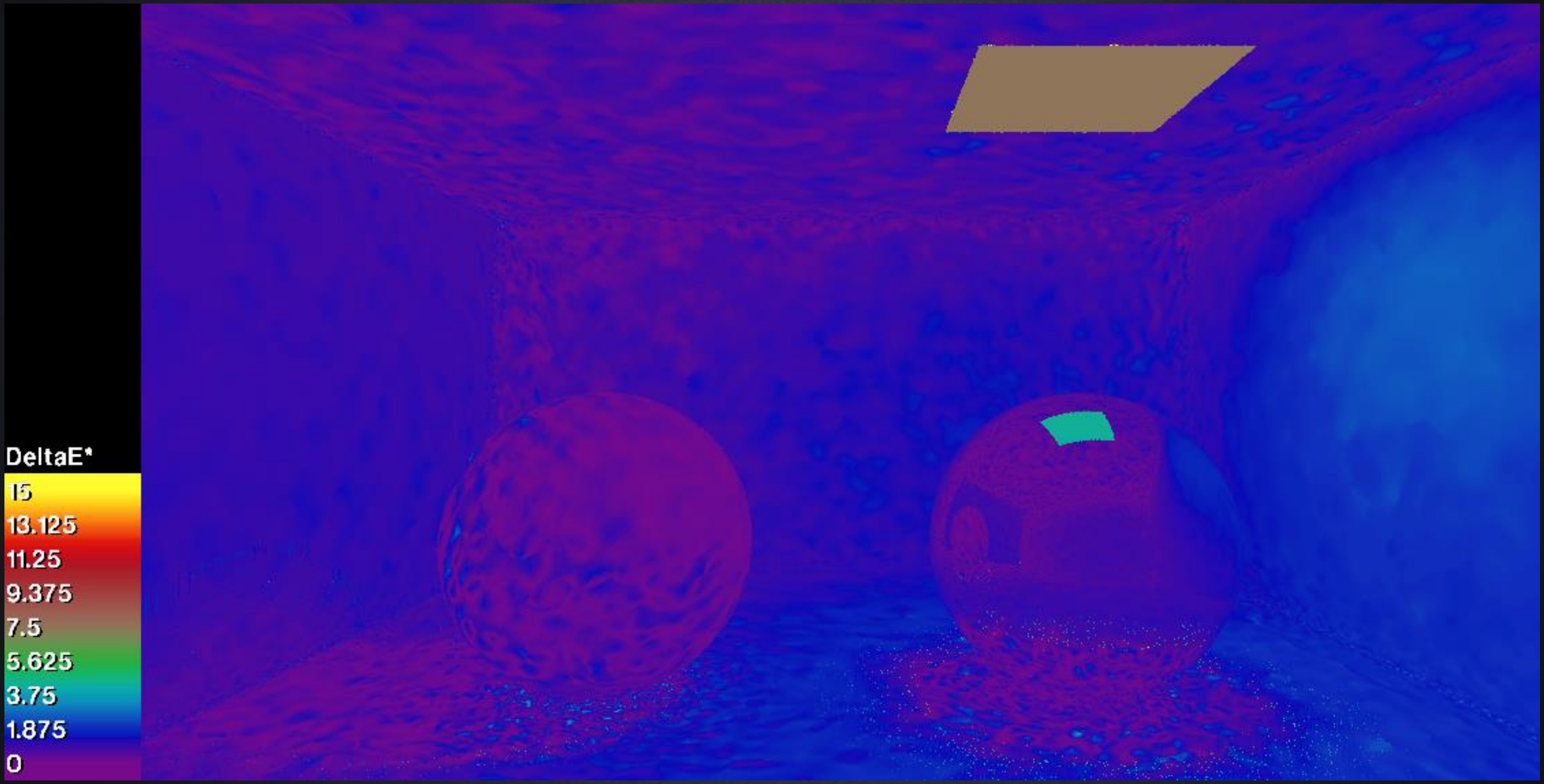
15 Spectral Sample Error



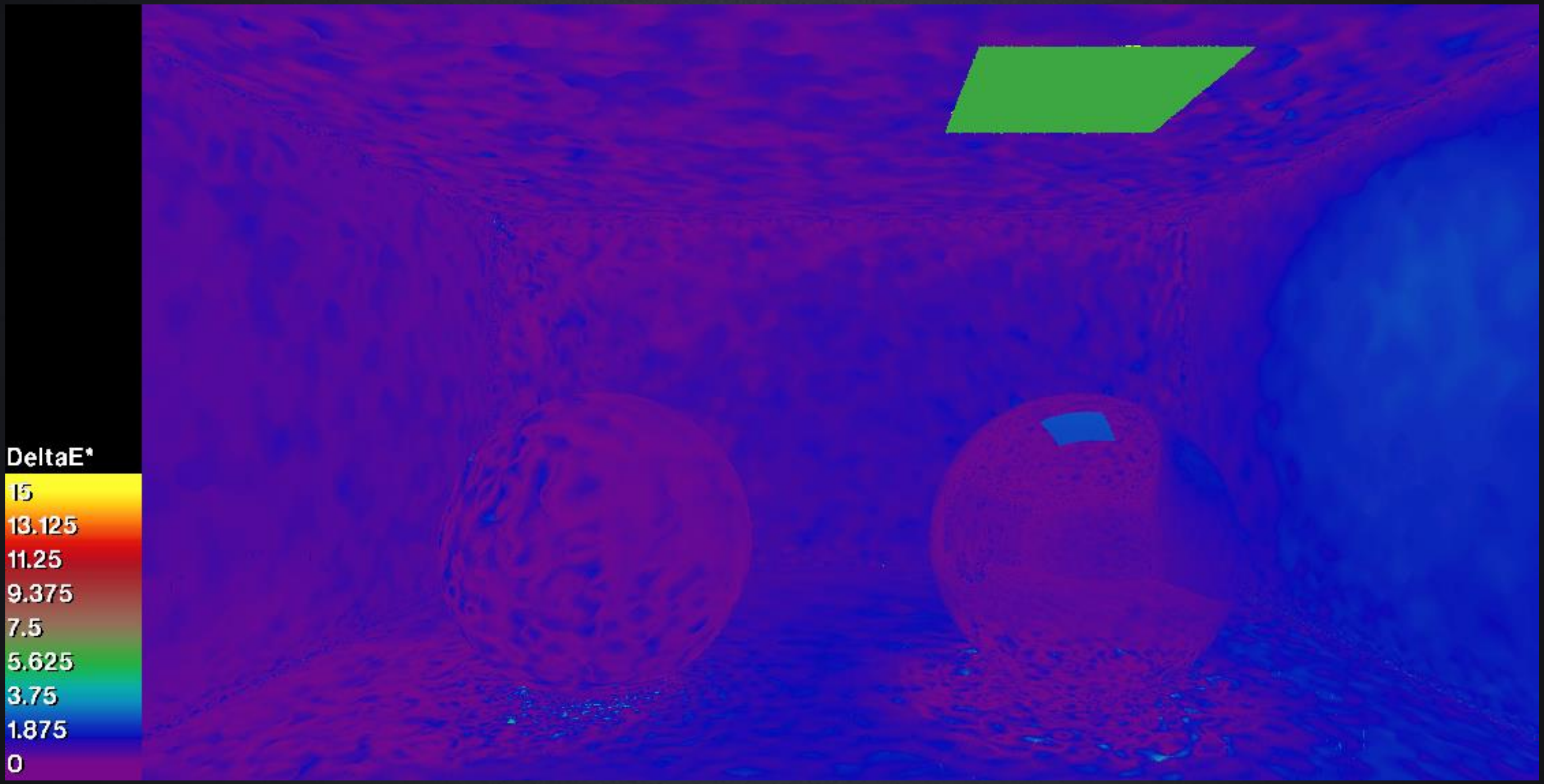
18 Spectral Sample Error



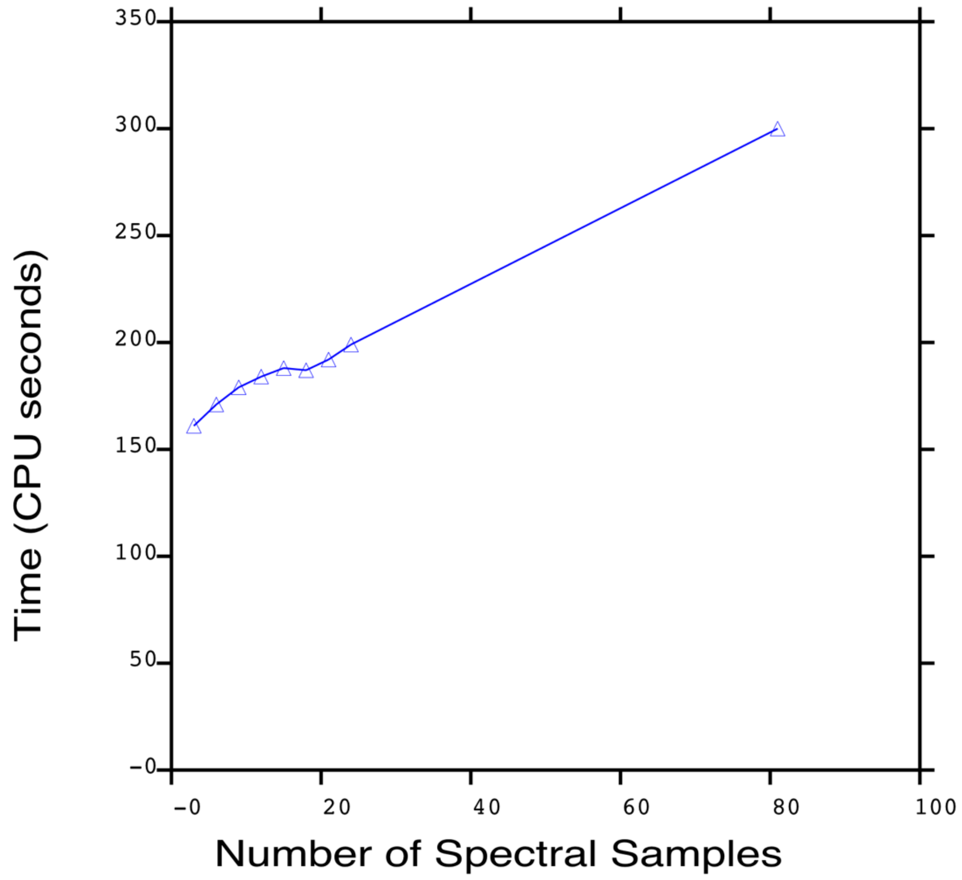
21 Spectral Sample Error



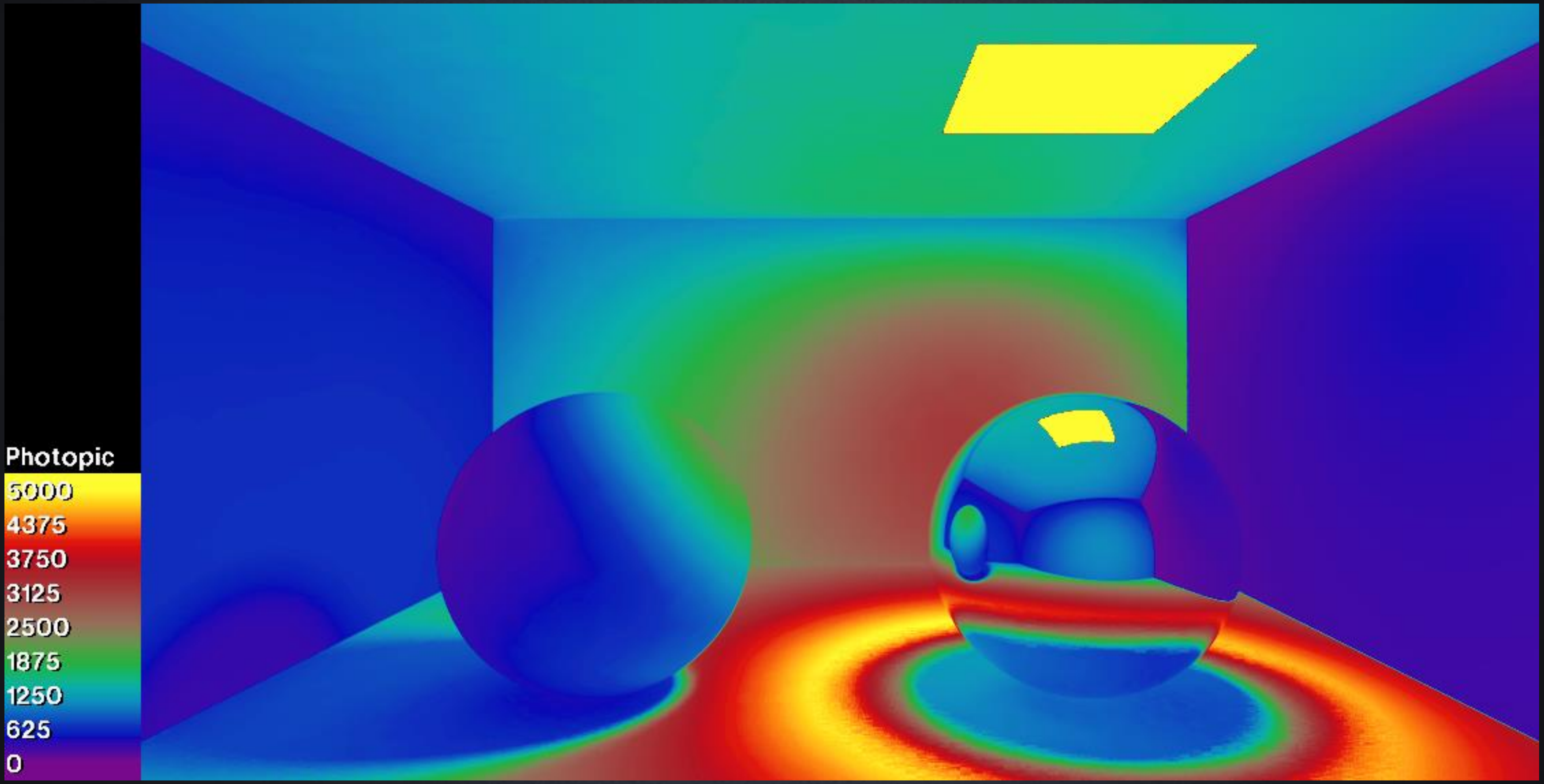
24 Spectral Sample Error



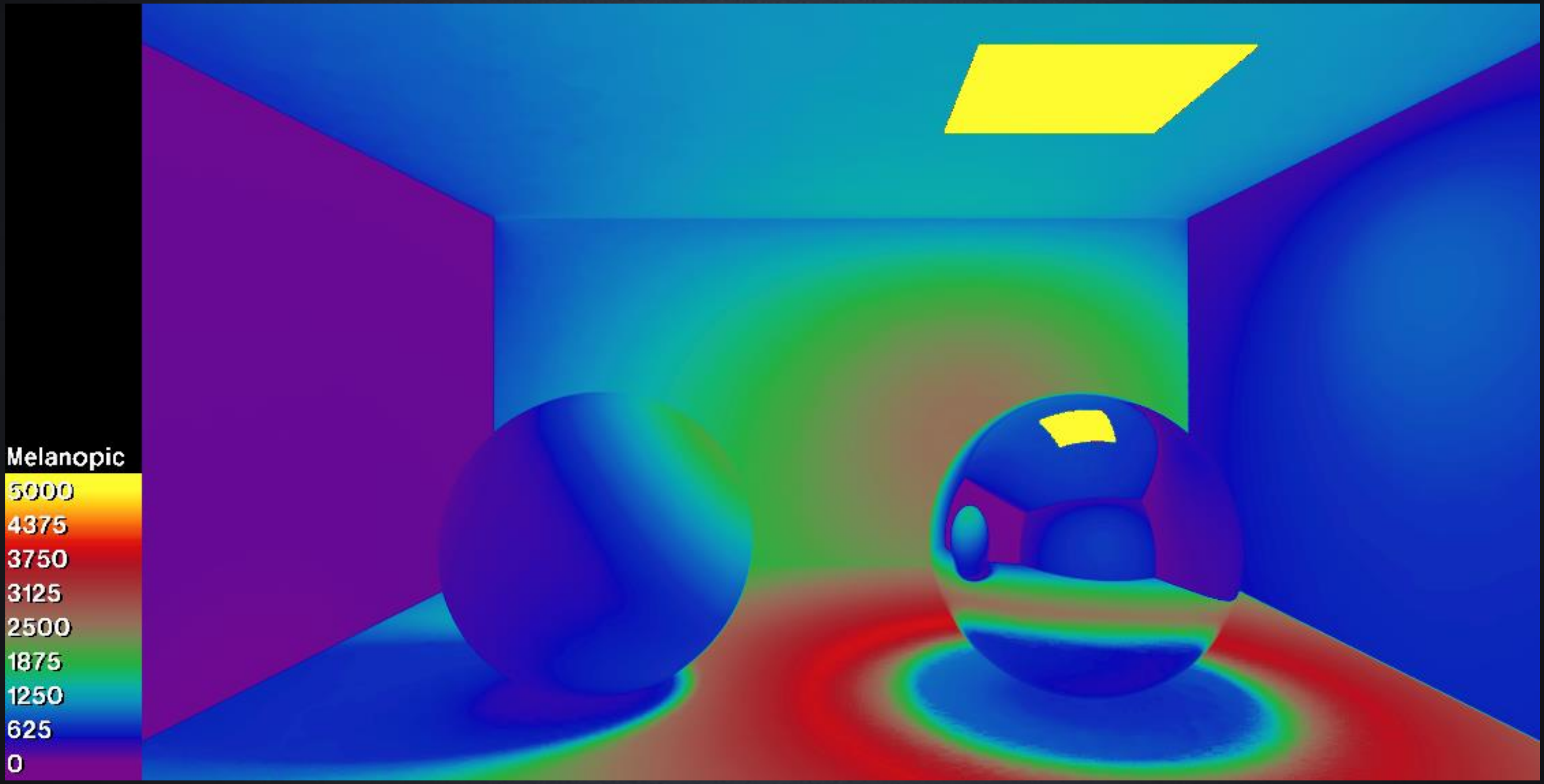
Rendering Time for Different Spectral Sample Densities



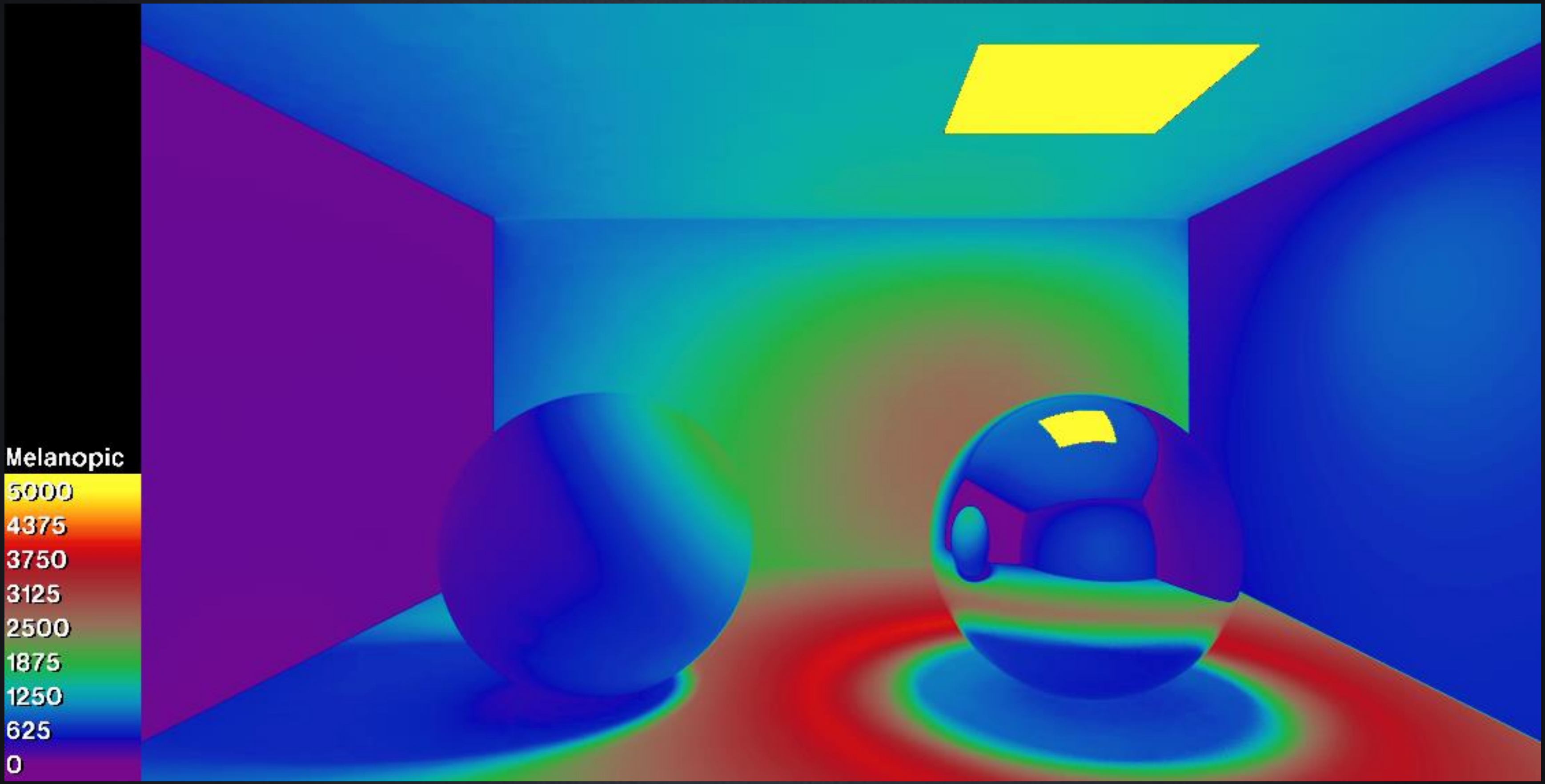
Photopic Luminance



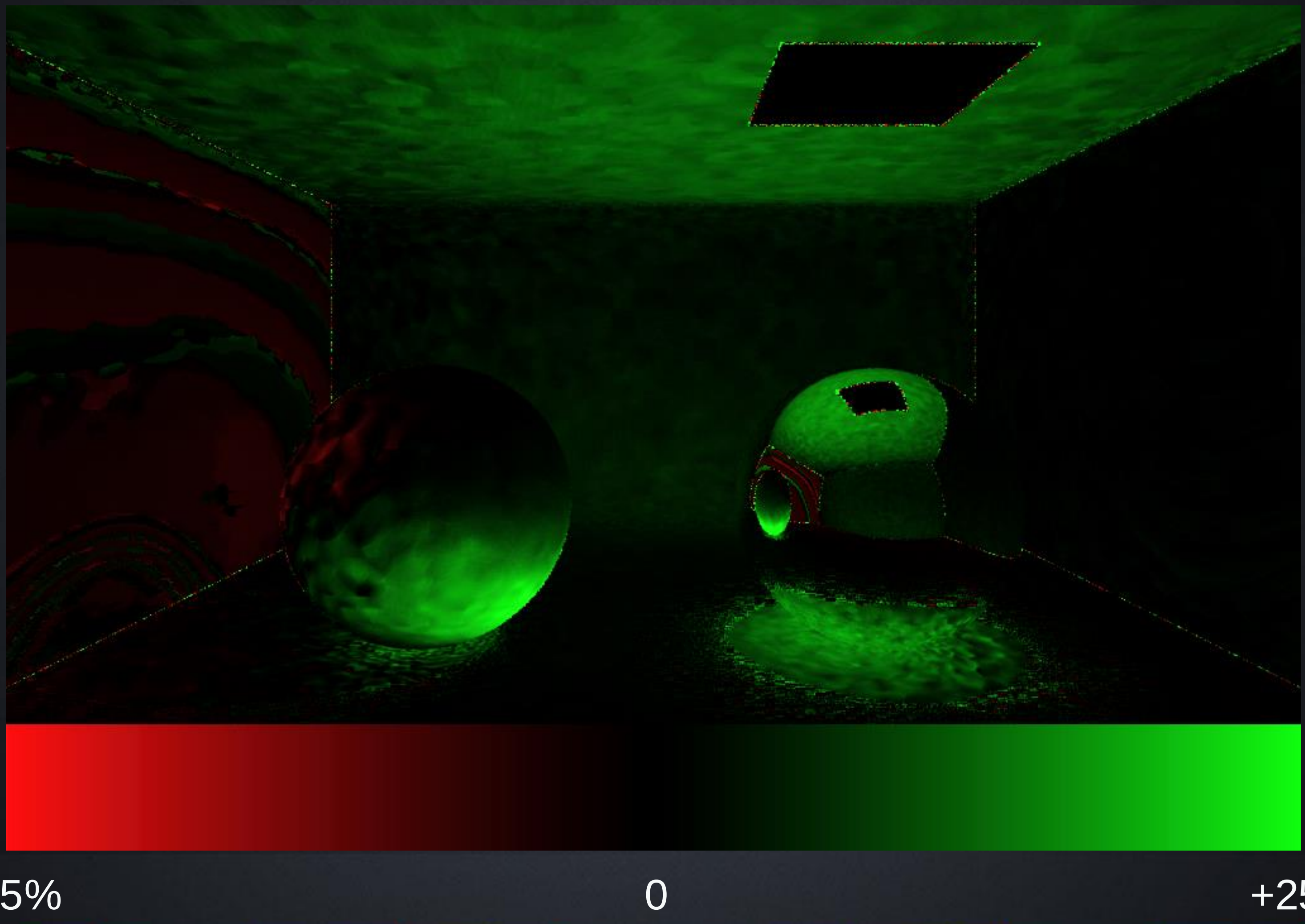
Melanopic Luminance
(from 24 spectral samples)



Melanopic Luminance
(estimated from RGB)



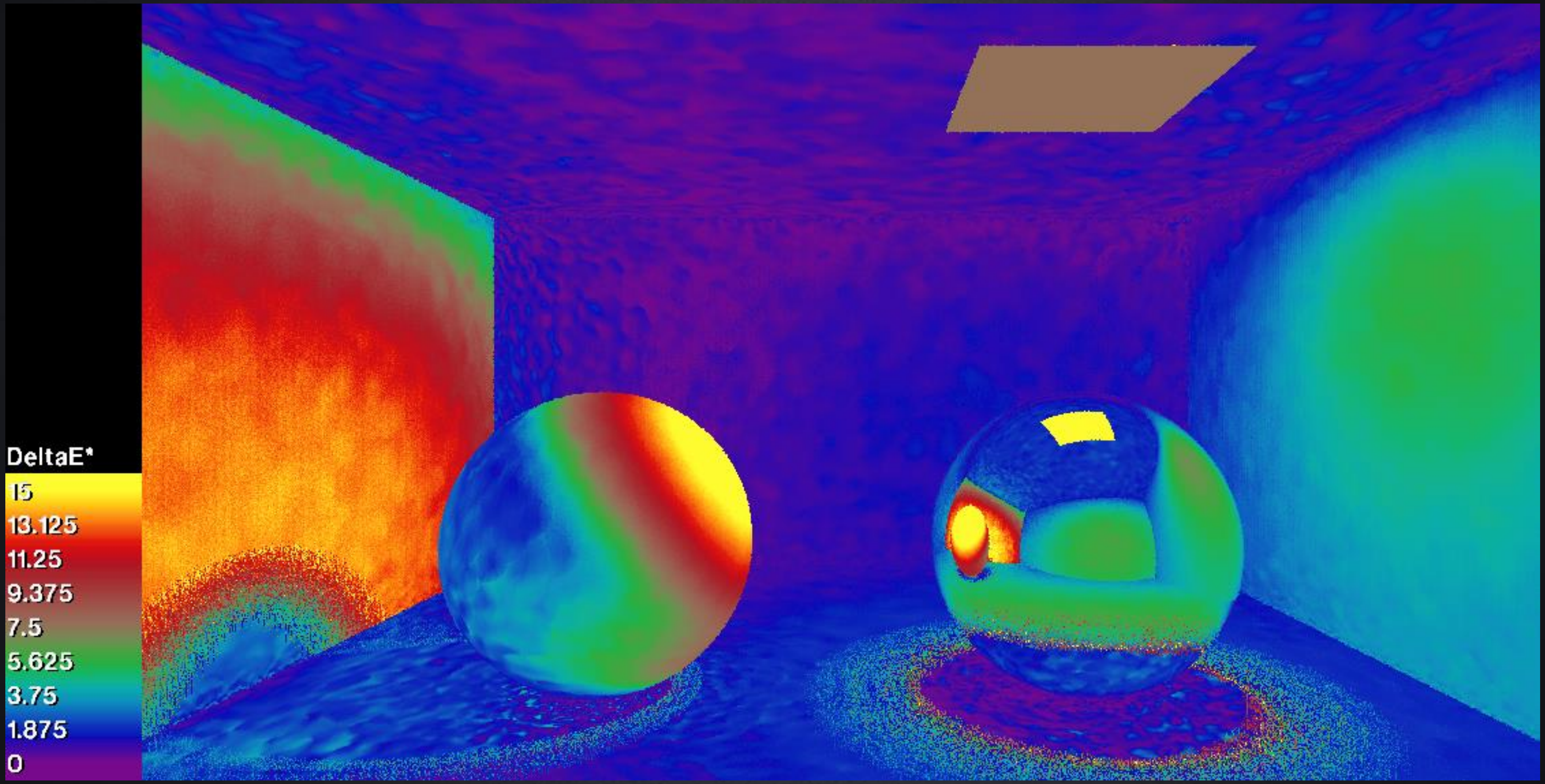
Relative Error from Using RGB Estimate



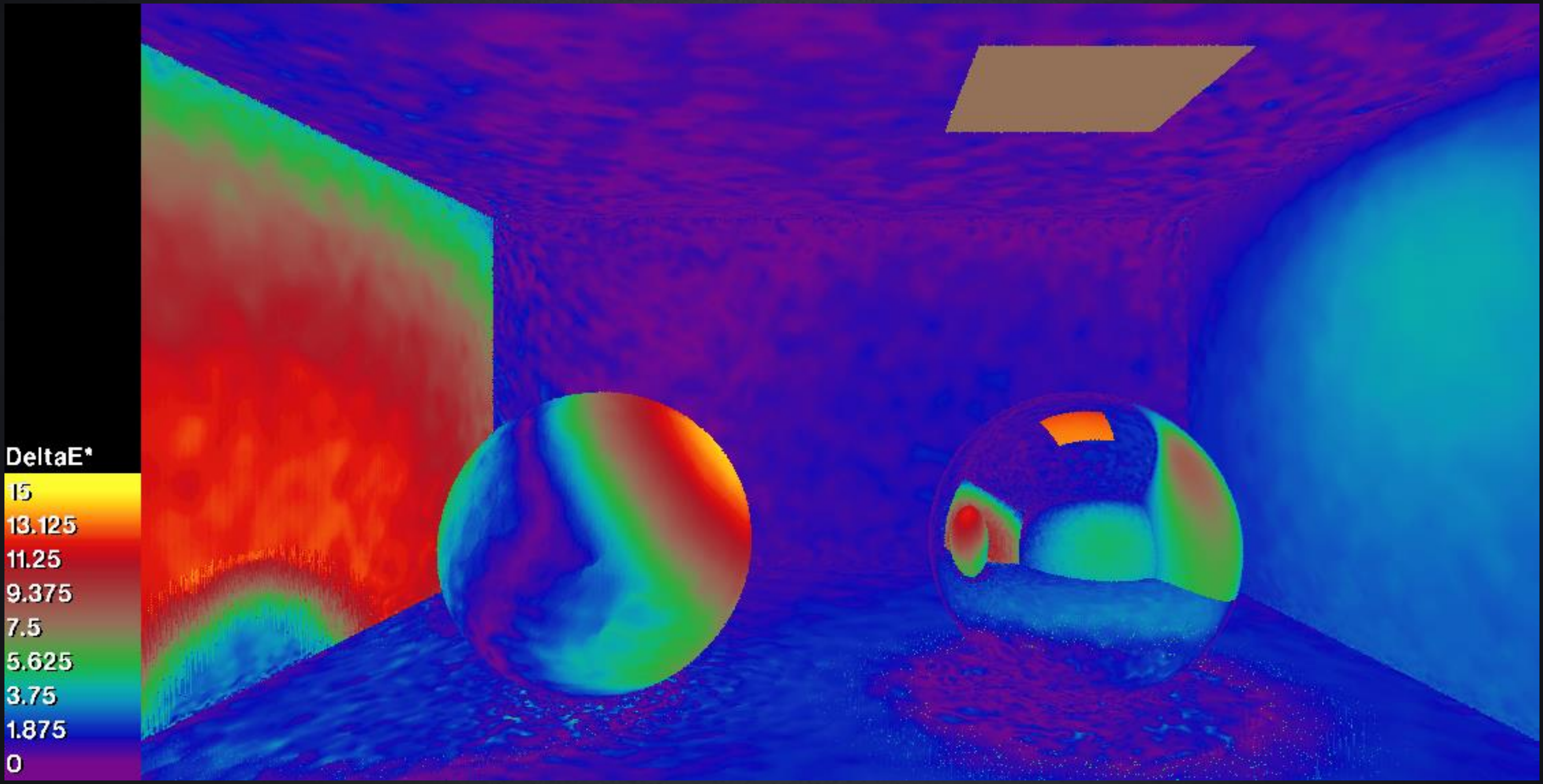
Points for Discussion

- Improvement levels off around 18 samples, at least for this scene
- Benefits of RGB color space over straight spectral rendering means we don’t see advantage until 9 samples
- Better selection of RGB primaries leads to more accurate results (EGWR ‘02)
- Extending beyond visible limits?

Error from Rendering in RGB



Using Sharpened RGB Space

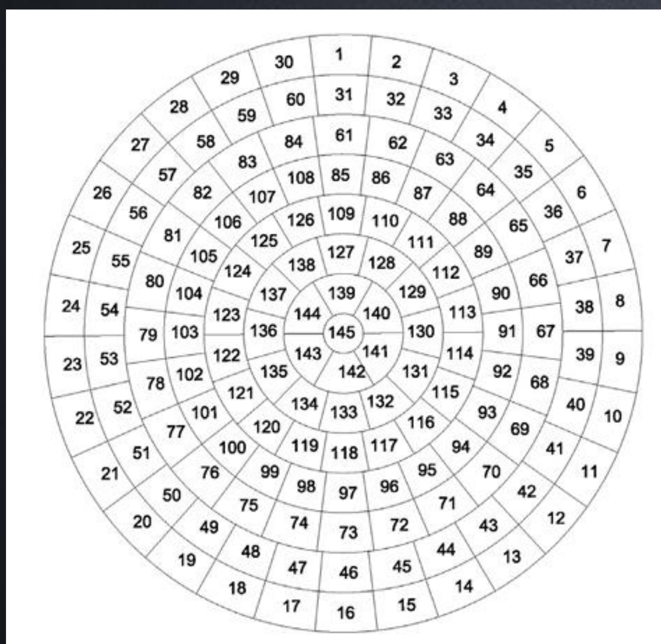


Using Measured Spectral Skies

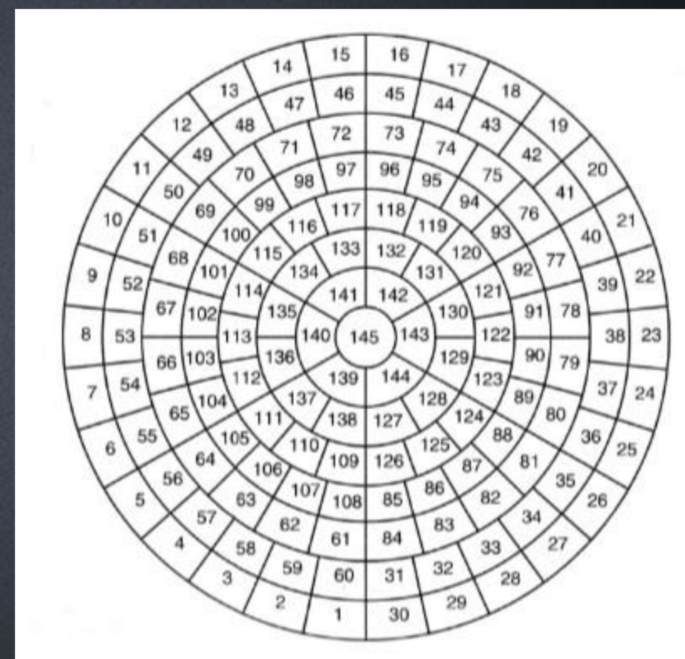
- Measured spectral sky data
  - CIE sky patches
  - Solar spectrum
- Method 1: drive from RGB sky capture(s)
  - Spectral component approximation
- Method 2: simulate atmospheric scattering (**genssky** & **gensdaymtx**)

Method 1: Spectral Sky Scan

Luo, T., Da Yan, R. Lin, J. Zhao, (2014). "Sky-luminance distribution in Beijing." *Lighting Research and Technology*. 47. 10.1177/1477153514532466.

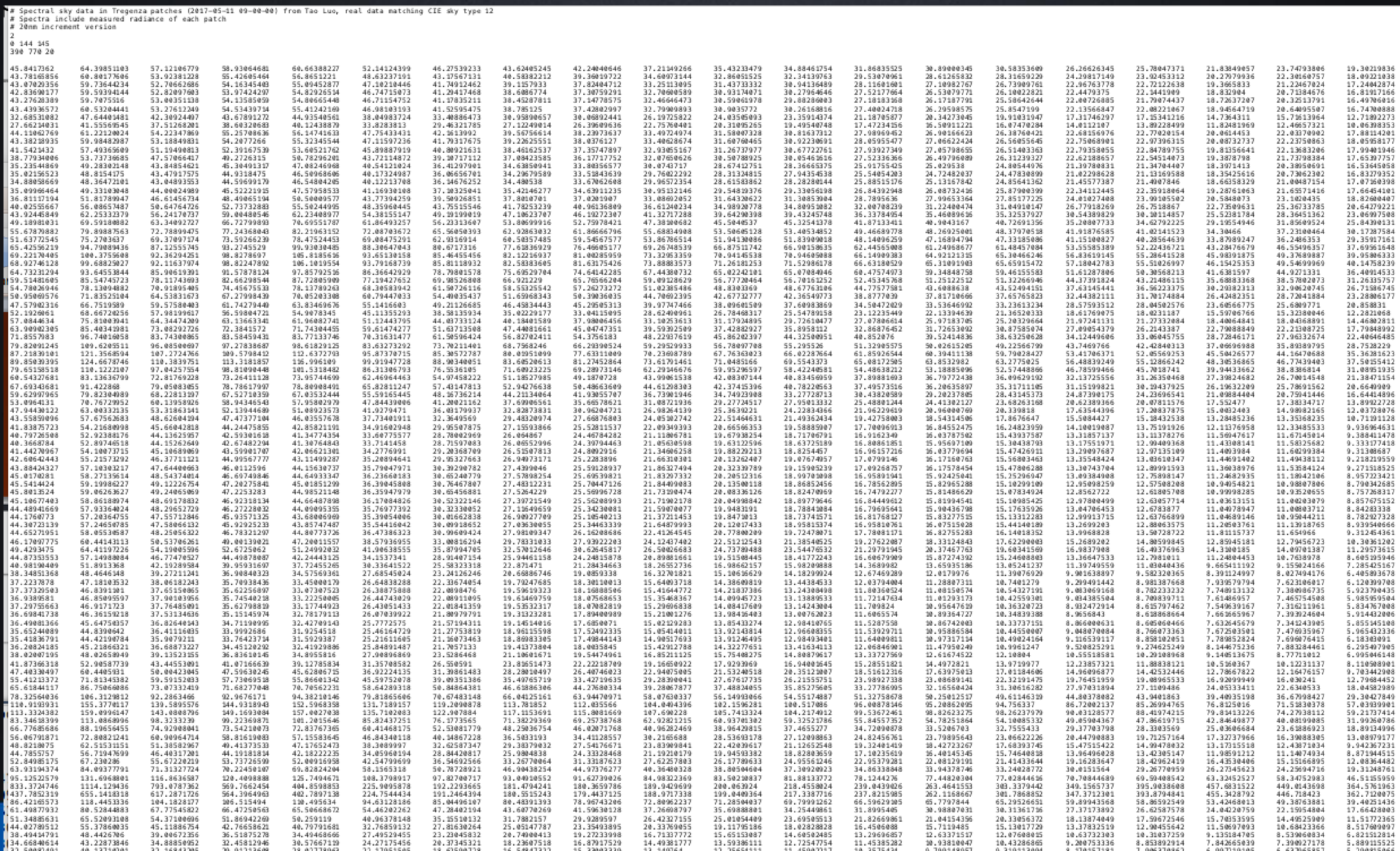


Tregenza Sky Patches



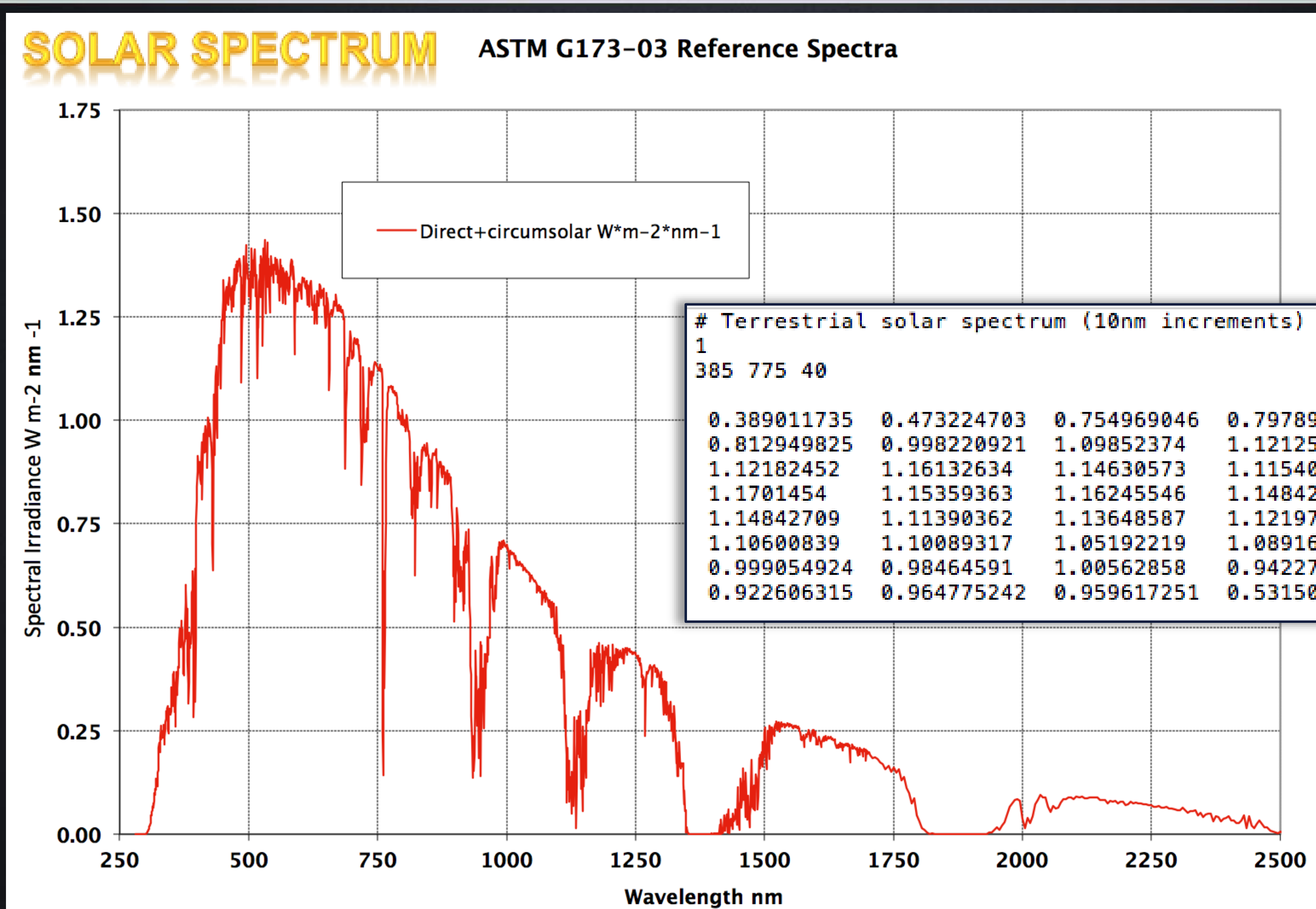
CIE Measurement Positions

Method 1: Spectral Sky



```
# Spectral sky data in Tregenza patches (2017-05-11 09-00-00) from Tao Luo, real data matching CIE sky type 12
# Spectra include measured radiance of each patch
# 20nm increment version
2
0 144 145
390 770 20
```

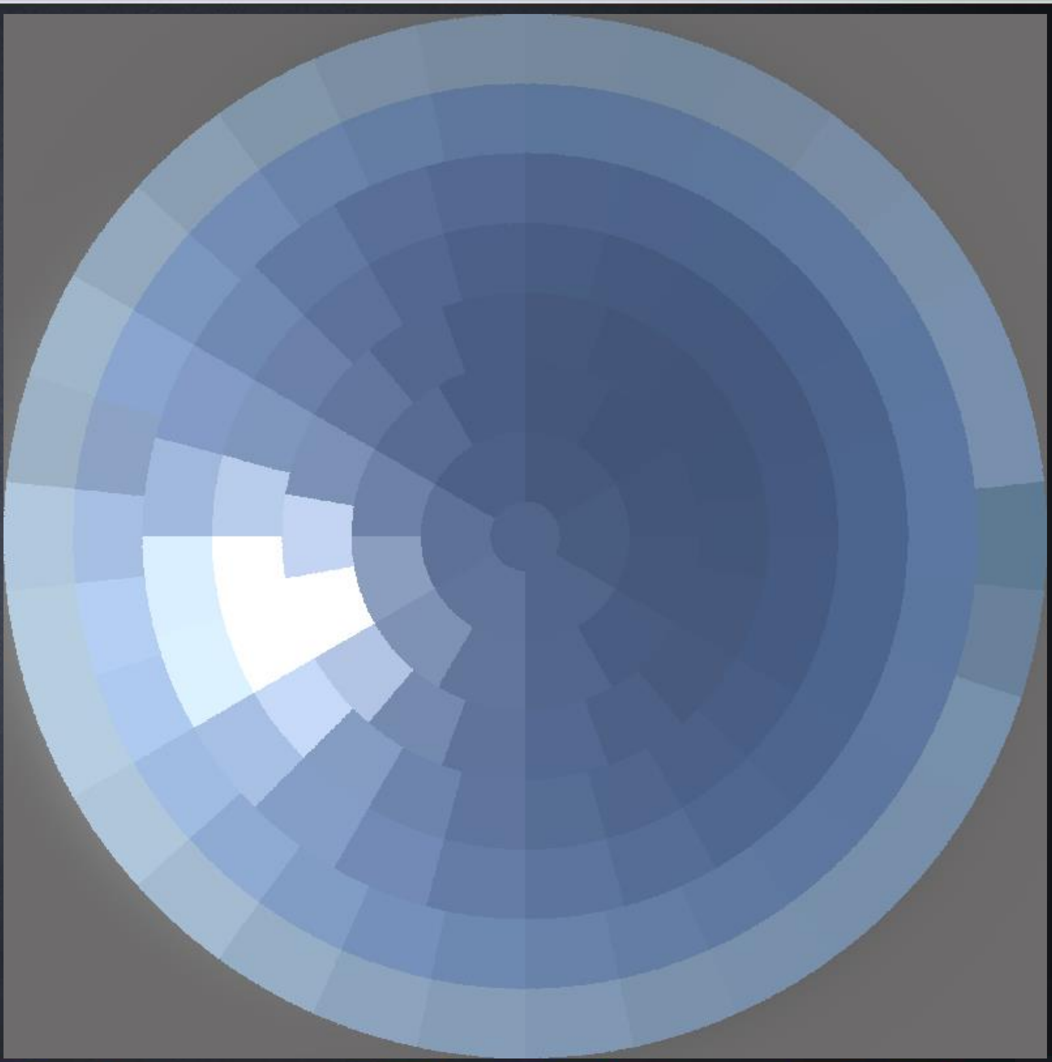
Method 1: Spectral Sky Scan



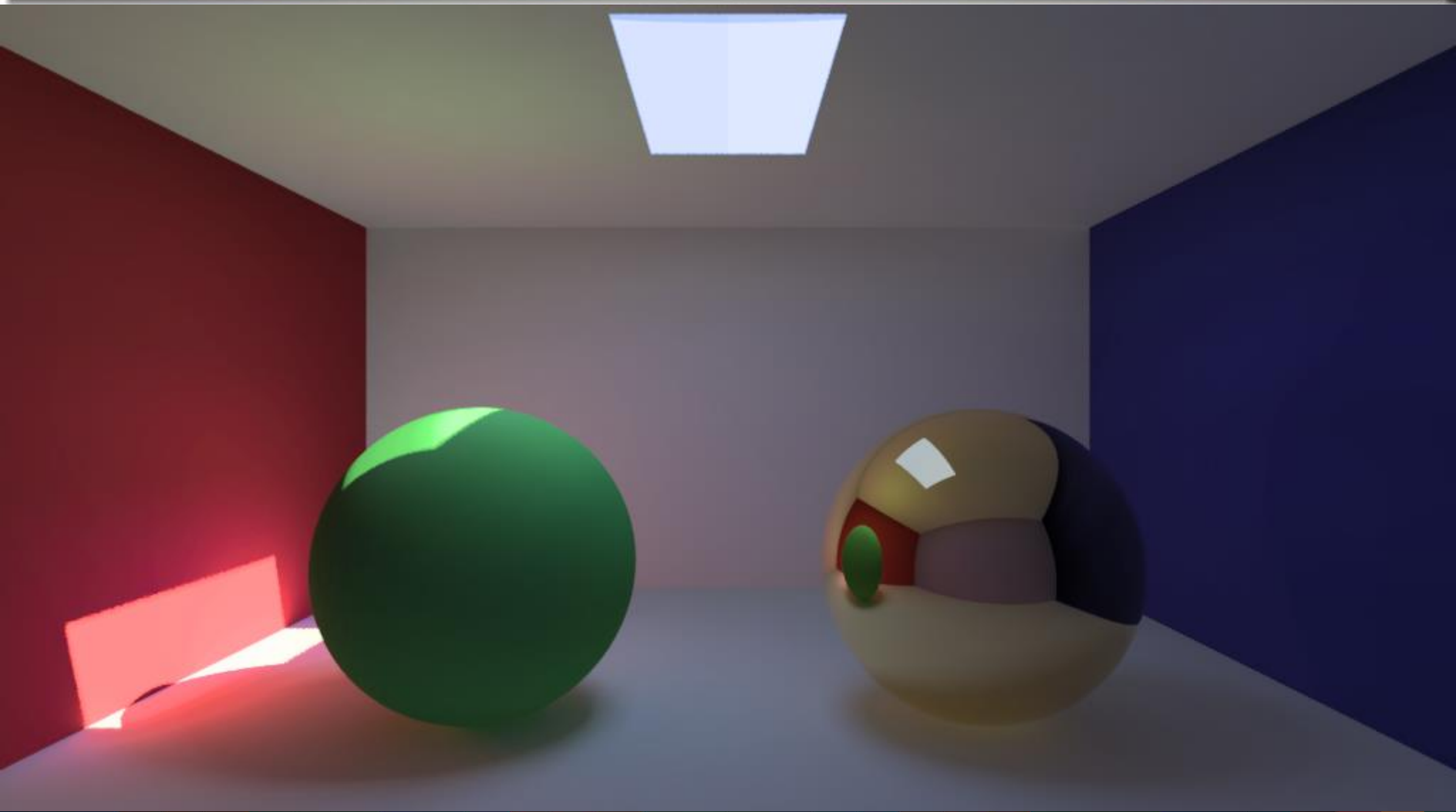

```
# Terrestrial solar spectrum (10nm increments)
1
385 775 40

 0.389011735  0.473224703  0.754969046  0.797894137  0.815415053
 0.812949825  0.998220921  1.09852374   1.12125141   1.14368824
 1.12182452   1.16132634   1.14630573   1.1154091    1.14863238
 1.1701454    1.15359363   1.16245546   1.14842709   1.13570747
 1.14842709   1.11390362   1.13648587   1.12197849   1.09929359
 1.10600839   1.10089317   1.05192219   1.08916579   1.08886641
 0.999054924  0.98464591   1.00562858   0.942275966  0.842634361
 0.922606315  0.964775242  0.959617251  0.531502301  0.921338629
```

Method 1: Spectral Sky

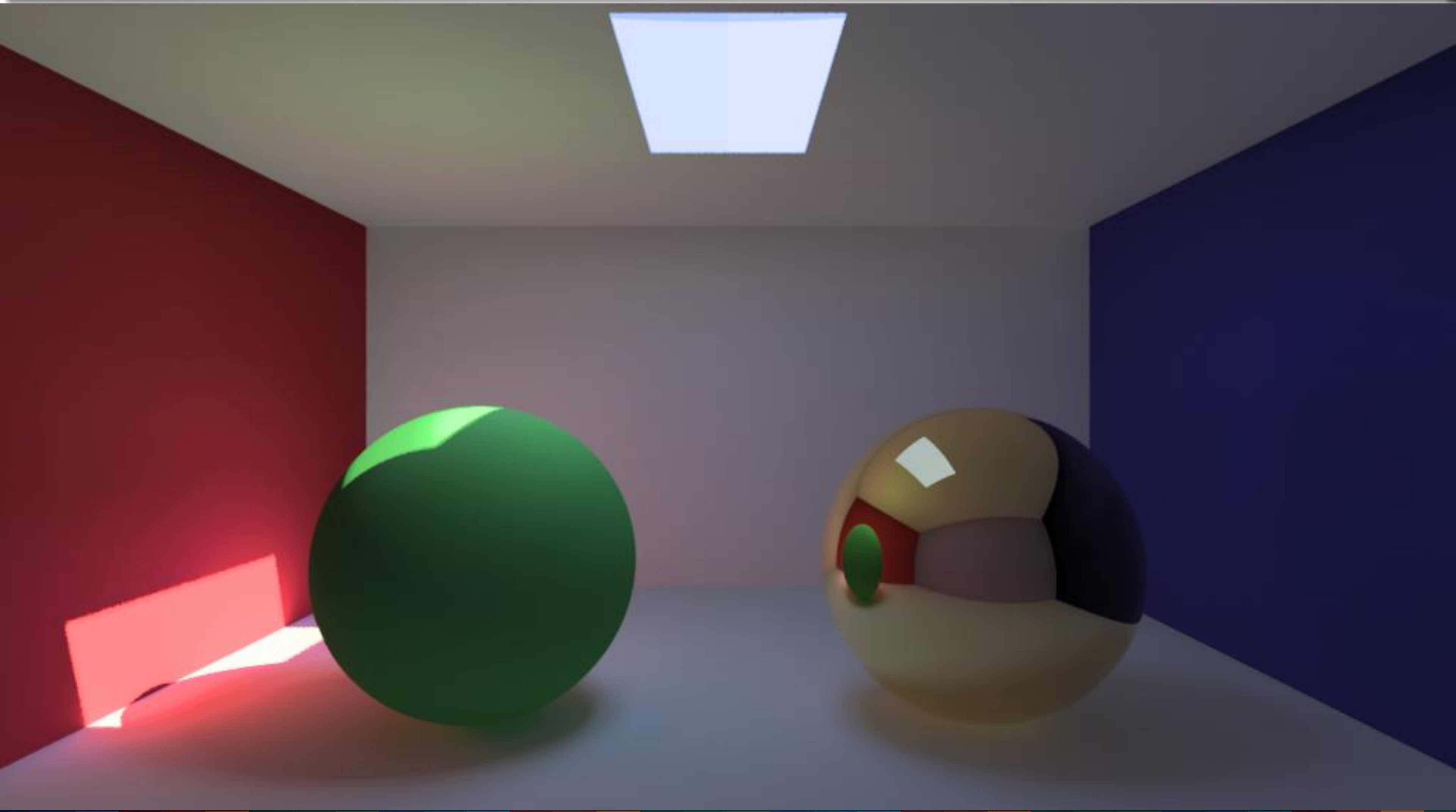
```
# Measured sky from 2017-05-11 9am in Beijing, CN

# Terrestrial direct solar spectrum (with circumsolar)
void specfile solarA1.5D
1 SpecSun10.dat
0
0

# Corresponding solar position:
# gensky 5 11 9 -y 2017 -m -120 -o -116.4074 -a 39.9042 +s
# Local solar time: 8.82
# Solar altitude and azimuth: 43.5 -76.0
# Ground ambient level: 14.6

solarA1.5D light solar
0
0
3 6.732e+06 6.732e+06 6.732e+06

solar source sun
0
0
4 0.703765 -0.175375 0.688446 0.5

void brightfunc skyfunc
2 skybr skybright.cal
0
7 1 9.172e+00 2.063e+01 5.480e-01 0.703765 -0.175375 0.688446

void specdata sky12
4 noop SpecSky12b.dat cieskyscan.cal cbin
0
0

sky12 glow sky12glow
0
0
4 1 1 1 0

sky12glow source sky12
0
0
4 0 0 1 180

skyfunc glow groundglow
0
0
4 1 1 1 0

groundglow source ground
0
0
4 0 0 -1 180
```



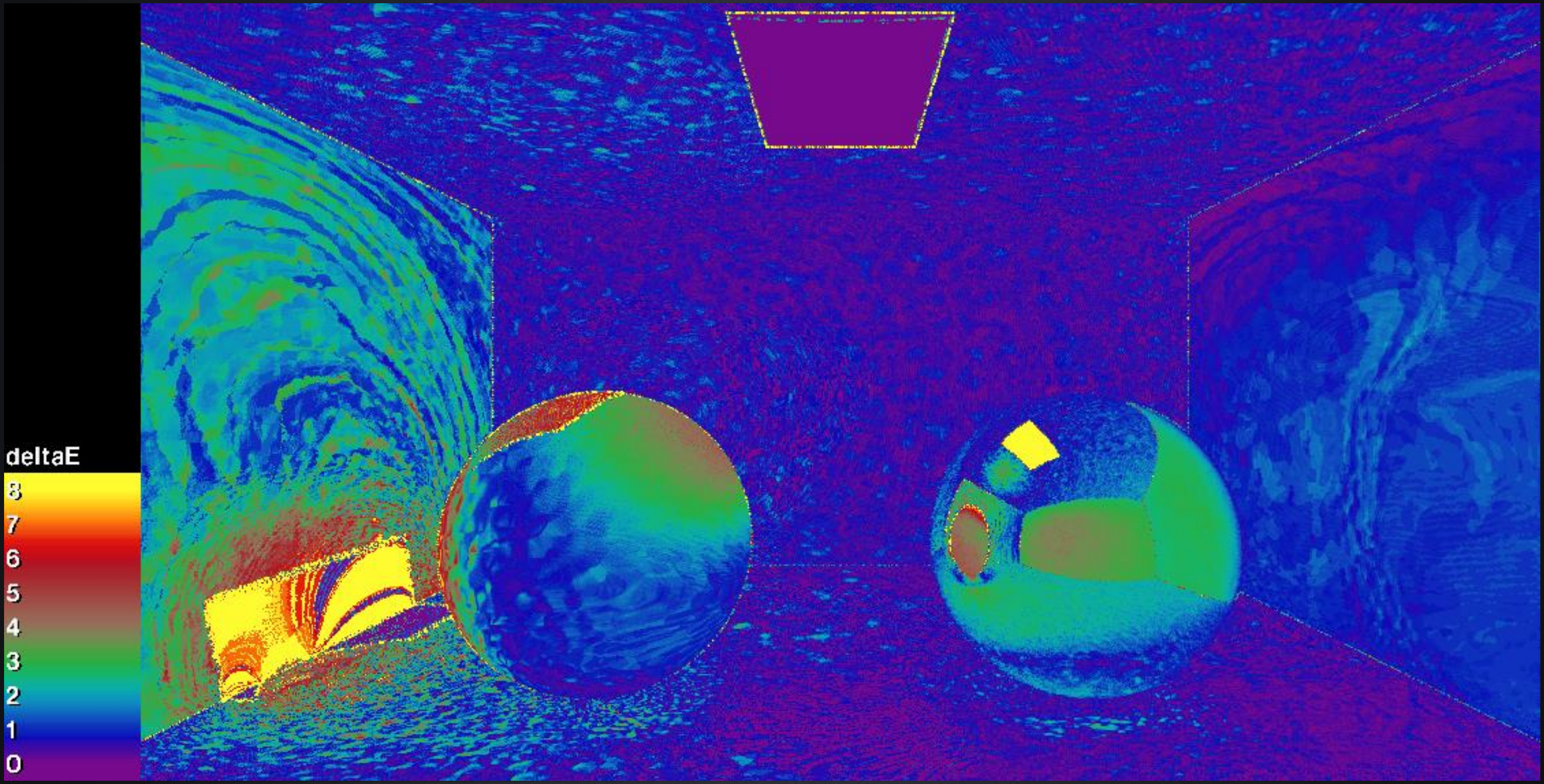
Method 1 Results (Spectral)



Method 1 Results (RGB)



Method 1 Difference (ΔE*)



Deep Dive w/ **rcomb** Tool (1)

- **rcomb** sits halfway between **rmtxop**, which performs matrix operations, and **pcomb**, which combines HDR images
- Less memory for supported operations
  - cannot transpose matrices
  - concatenates at most two matrices
- Programmable like **pcomb**, but for matrix and HSR input and output

Deep Dive w/ **rcomb** Tool (2)

- Useful options
  - shared with **rmtxop**:

-c {CS|file|coefficient list}	Convert input or output color space
-C {CS|file}	Default input color conversion
-s {sf|coefficients}	Apply scale factor(s) to components
-f{a|f|d|c}	Output format (always to stdout)

  - similar to **pcomb**:

-f {file.cal}	.cal file input
-e {expr}	.cal expression
-w	turn off warnings
-h	trim header

  - unique to **rcomb**:

-m {matrix}	concatenate with final matrix
-n {nproc}	multiprocessing row calculations

Deep Dive w/ **rcomb** Tool (3)

- Predefined variables and functions:

nfiles, ncols, ncomp	#inputs, #columns, #components
nrows	#rows, which may be 0 if unknown
r, c	current row and column (r=0 @ top)
ci(i,p)	component *p* of input *i* at this *r*, *c*
wl(p)	wavelength (nm) for component *p*

- User-defined variables and functions:

co	current component output, or:
co(p)	component *p* output, or:
ro, go, bo	RGB component outputs

- If co undefined, output is straight sum

Typical **rcomb** Usage

- Simple example #1 - compute ratio of melanopic values between pictures:

rcomb -C M view1.hsr view1.hdr -e “co=ci(2)/ci(1)” **-ff** > Mratio.mtx

- Simple example #2 - apply spectral weights and convert to HSR to HDR:

rcomb -s `cat sfactors.txt` original.hsr **-c** RGB **-fc** > weighted.hdr

Annual Holodeck Simulation

1. Take a previously rendered holodeck (point-in-time) and extract sample origins and directions for **rcontrib** or **rfluxmtx**:

```
rhcopy orig.hdk -ff -ood \
| rfluxmtx -ffc [options] -o daycoef.mtx \
      - tregsky.rad -i scene.oct
```

1. Run **rcomb** for particular sky and put into empty holodeck (coupling origins & hit points):

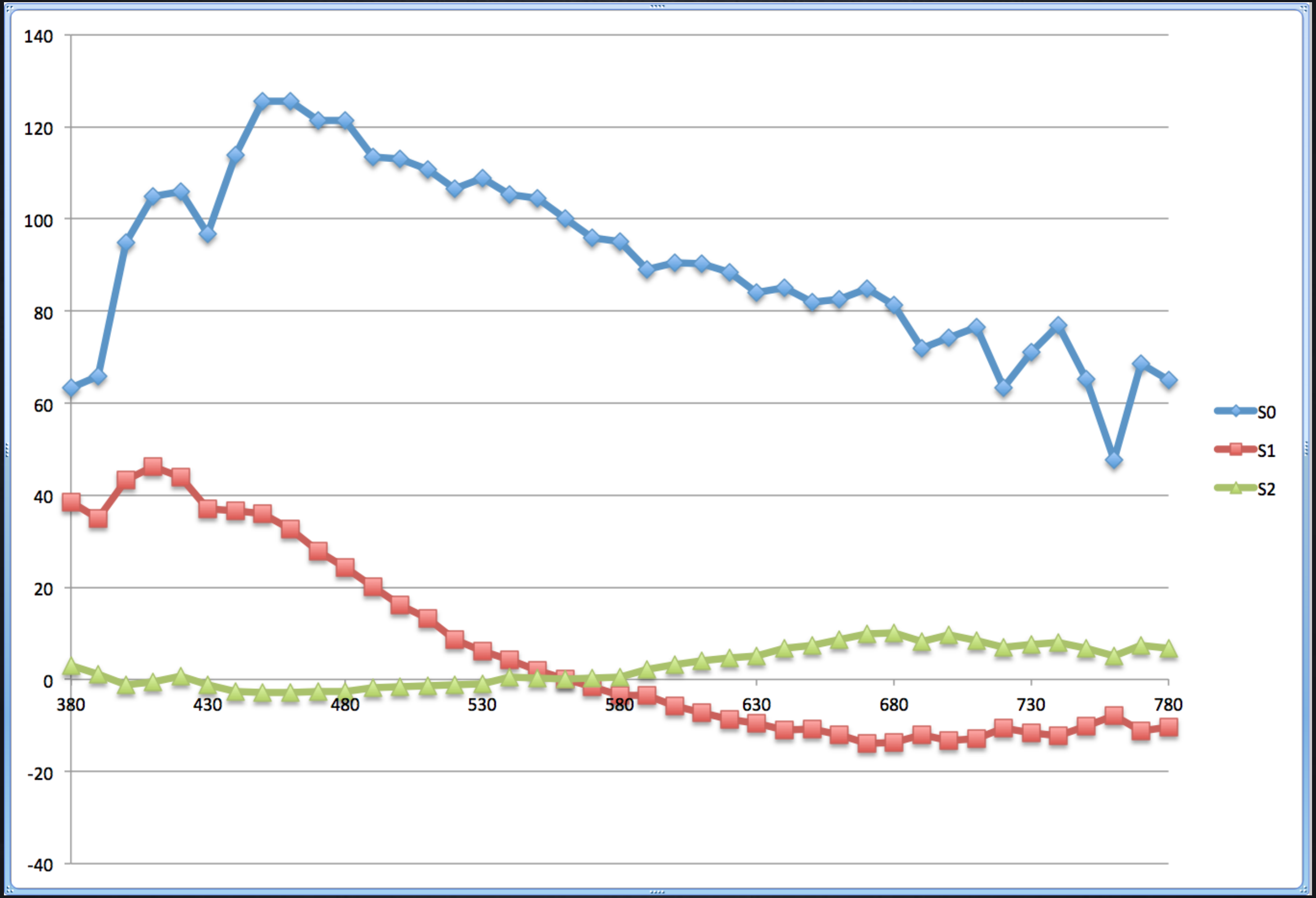
```
rholo new08-10@14.hdk template.hif + OCTREE=sc08-10@14.oct
gensky 8 10 14 [location] | genskyvec -m 1 \
| rcomb -ff daycoef.mtx -m - | getinfo - \
| rlam -if3 - -if6 “!rhcopy orig.hdk -ff -oop” \
| rhcopy new08-10@14.hdk -ivop -ff -u -d
```

Method 2: RGB→Spectral Sky

- Use method due to Preetham et al., “A Practical Analytic Model for Daylight,” *SIGGRAPH 1999*:

- convert from RGB to XYZ and then to xyY
- convert xyY color M0, M1, M2 coefficients
- use M coefficients in a linear combination of the CIE D65 standard illuminant components

Method 2: RGB→Spectral Sky



Method 2: RGB➔Spectral Sky

```
rcomb -fc -c XYZ input_sky.hdr -f skyfact.cal -f sky2spectra.cal \
        -c `cnt 20 | rcalc -f skyfact.cal -e 'wl=780-20*($1+.5)' \
                -e '$1=S0(wl);$2=S1(wl);$3=S2(wl)'` \
        | getinfo -r 'WAVELENGTH_SPLITS= 780 588 480 380' \
        > specSky.hsr
```

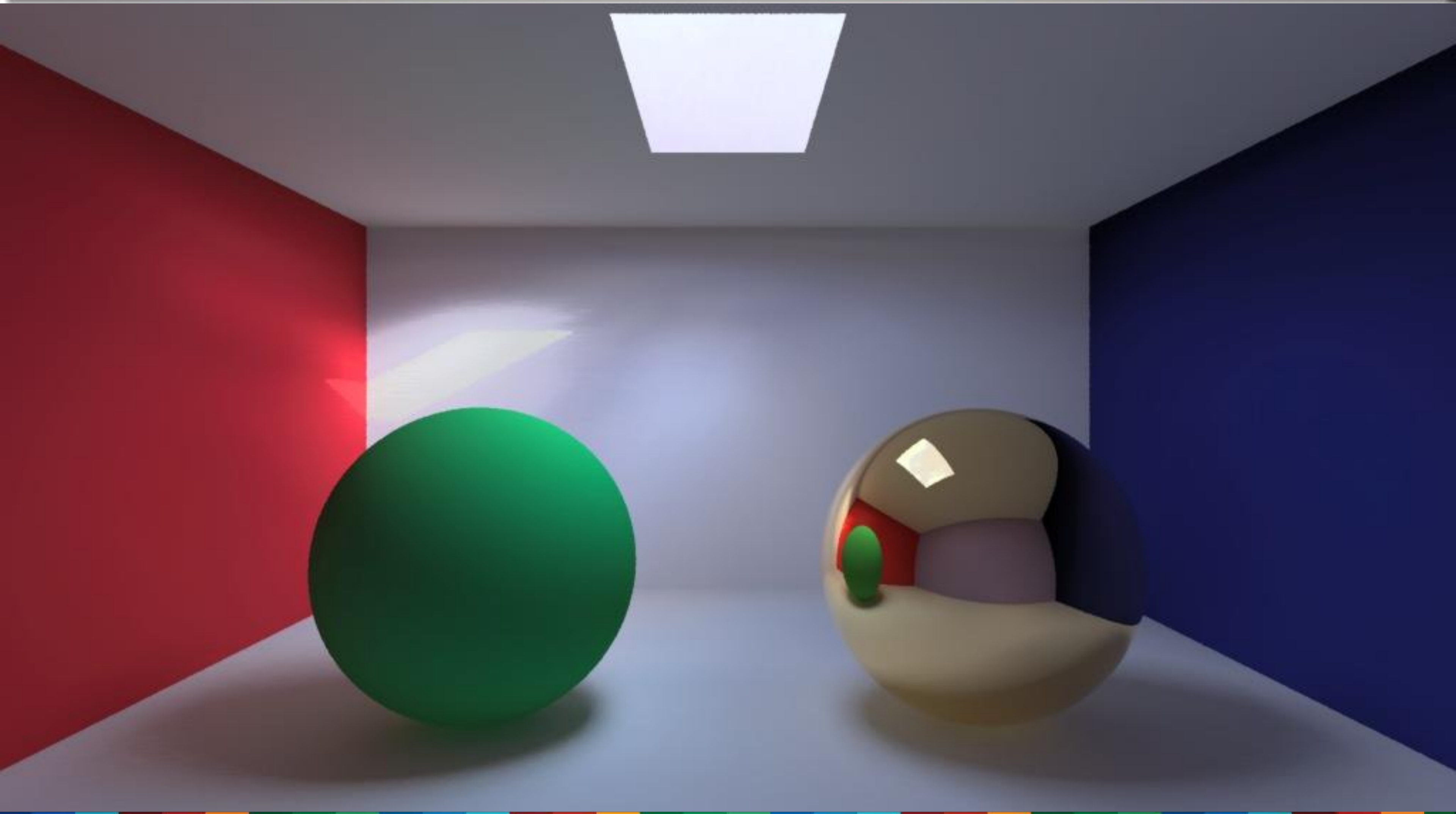
```
{--- sky2spectra.cal ---}

in_Y = ci(1,2);
xyzM = 1/(ci(1,1) + in_Y + ci(1,3));
in_x = ci(1,1)*xyzM;
in_y = in_Y*xyzM;

in_M1 = M1(in_x,in_y);   {(M1,M2) from CIE (x,y) using Preetham et al.}
in_M2 = M2(in_x,in_y);

co_Multiplier = in_Y / (S0normf + in_M1*S1normf + in_M2*S2normf) / 179;

co(n) = if(in_Y - 1e-4, co_Multiplier*select(n, 1, in_M1, in_M2), 0);
```

Method 2: RGB➔Spectral Sky

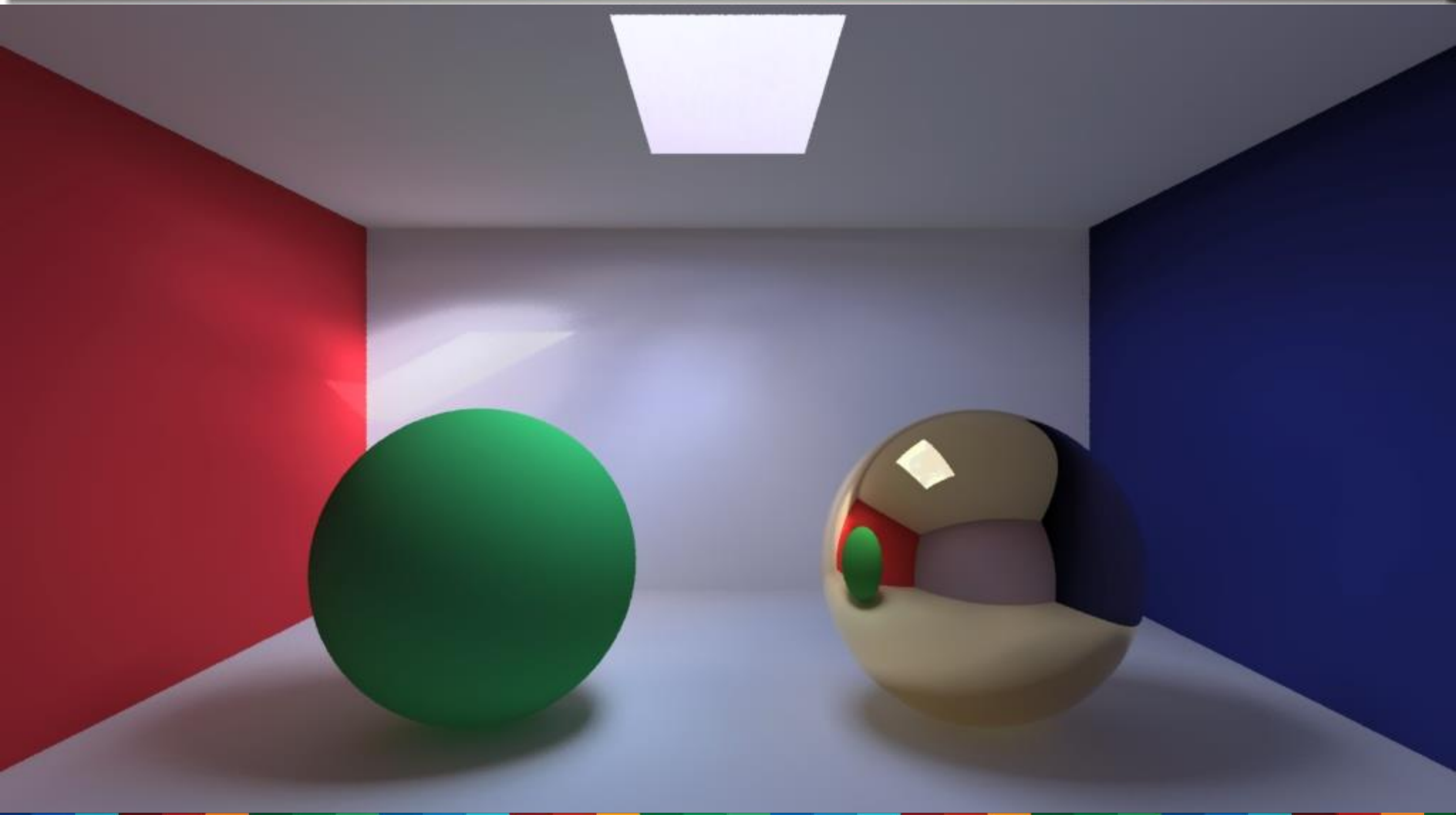
```
# Spectral sky generated by mkSPECskies.csh

void specfile solarSpectrum
1 ../SpecSun10.dat
0
0

solarSpectrum light solar
0
0
3 250174.581 250174.581 250174.581

solar source sun
0
0
4 0.600657421 -0.739446978 0.304021101 0.53
```

```
void glow ground_glow
0
0
4 5.68421884 5.68421884 5.68421884 0

ground_glow source ground
0
0
4 0 0 -1 180

void specpict skyfunc
9 noop 14-30-00/specSky.hsr fisheye.cal fish_u fish_v
            -rx 90 -rz 180
0
0

skyfunc glow sky_glow
0
0
4 1 1 1 0

sky_glow source sky
0
0
4 0 0 1 180
```

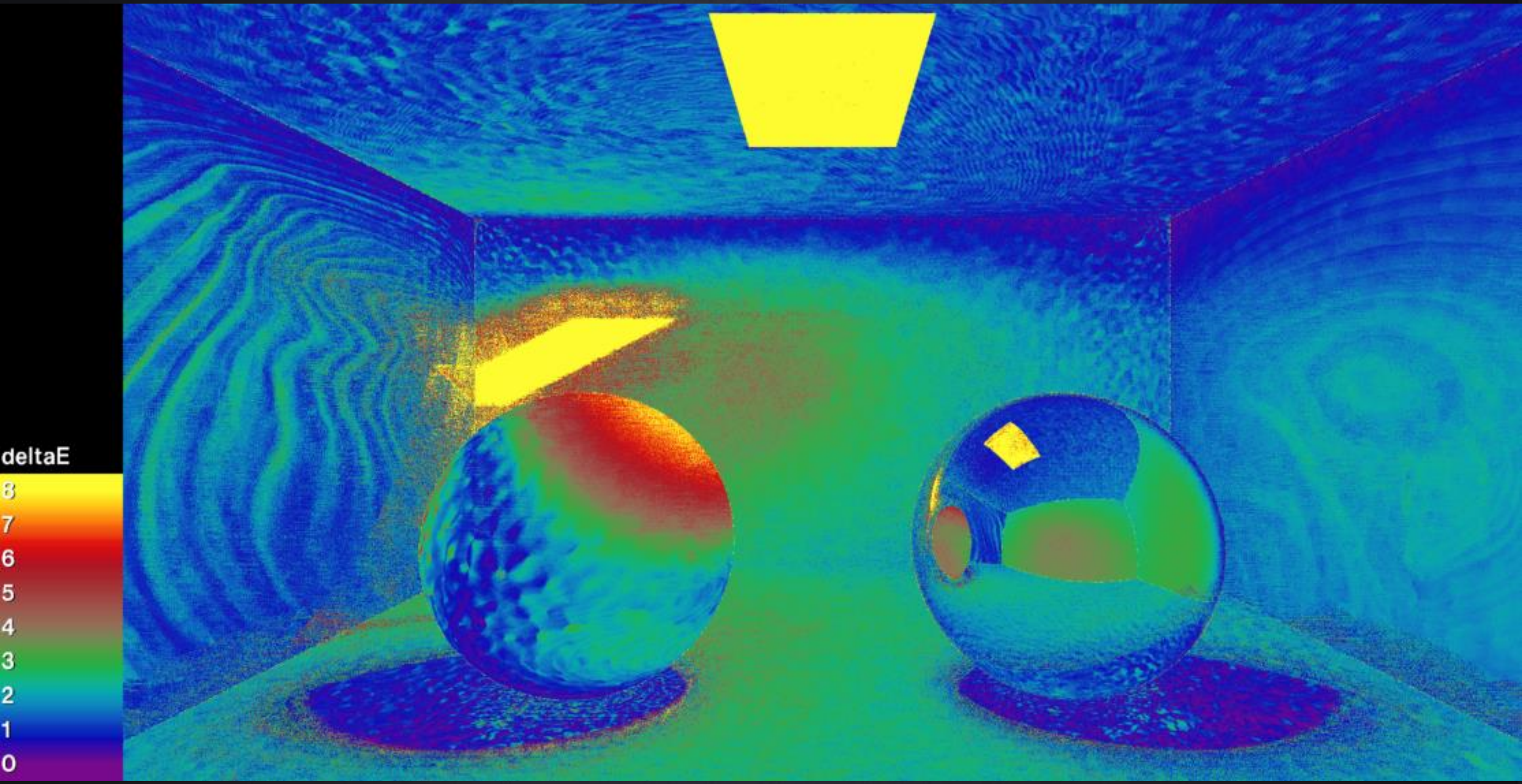
Method 2 Results (Spectral)



Method 2 Results (RGB)



Method 2 Difference (ΔE*)



DC Spectral Matrix Method

1. Convert RGB sky captures to column vectors in time-stepped daylight matrix
2. Run **rfluxmtx** to create spectral **V** matrix
3. Use **rcomb** and **rsplit** to compute time-step result HSR pictures
4. Apply post-processing for electrochromics

Sky Capture➔Matrix

- No real automation for this step
- Spectral sky w/o sun could be computed with **rfluxmtx**, but
- Spectral sky w/ sun needs **genskyvec** or similar
  - **genskyvec** currently RGB only
- Taoning has developed a new tool for weather tape ➔spectra: **gensdaymtx**

Computing Spectral **V** Matrix

- Use **rfluxmtx** to convert samples to spectral matrix
  - **dctimestep** not equipped for HSR, yet

```
vwrays -vf def.vf -ff -x 2048 -y 1138 -pa 0 \
| rfluxmtx -ff -t 300 -x 2048 -y 1138 -o vw_spec_r4.mtx \
      -ab 2 -ad 4096 -lr -12 -lw 1e-4 -cs 20 -n 12 \
      - sky_src_rh4.rad room_spec.rad
```

```
#@rfluxmtx h=r4 u=Y

void light skyLighter
0
0
3 1 1 1

skyLighter polygon skylight
0
0
12
        8.5     7       10
        8.5     13      10
        11.5    13      10
        11.5    7       10
```

Computing Spectral **V** Matrix

- Output is matrix, not image set, since file specification had no %d format

```
#?RADIANCE
oconv -f room_spec.rad sky_src_rh4.rad
rcontrib -fo+ -ab 2 -ad 4096 -lr -12 -lw 1e-4 -cs 20 -n 12 -x 2048 -y 1138 -fff -c 1 -f
reinhartb.cal -p MF=4,rNx=0,rNy=0,rNz=-1,Ux=0,Uy=1,Uz=0,RHS=+1 -bn Nrbins -b rbin -m
skyLighter
SOFTWARE= RADIANCE 6.0a lastmod Tue Apr 30 16:41:39 PDT 2024 by gward on behemouth
CAPDATE= 2024:05:13 15:55:43
GMT= 2024:05:13 22:55:43
WAVELENGTH_SPLITS= 780 588 480 380
NCOMP=20
NROWS=2330624
NCOLS=2305
BigEndian=0
FORMAT=float
```

-x 2048 -y 1138

NROWS = 2048x1138

NCOLS = Reinhart patches (no ground)

Multiply **V** Matrix by **S** Matrix

- Sky matrix **S** has one row per sky patch and one column per time step
- **rmtxop** tool would need to load 2336204x2305 matrix (x20 channels) into memory = 800 GBytes
- **rcomb** tool does a row at a time, instead:

```
rcomb -n 8 -fc vw_spec_r4.mtx -m skySPEC4.mtx > sky_frames.hsr
```

Extract HSR Time Steps

- Each row is a new pixel position
- Each column within a row is a different time step

pixel0,step0 pixel0,step1 pixel0,step2 … pixel0,step2304
pixel1,step0 pixel1,step1 pixel1,step2 … pixel1,step2304
pixel2,step0 pixel2,step1 pixel2,step2 … pixel2,step2304
…
pixel2330623,step0 pixel2330623,step1 pixel2330623,step2 … pixel2330623,step2304

To assemble first time step view, we need to gather first column pixels into a single image; second time step column into second image, etc.

Split the HSR Time Step Pixels into Pictures

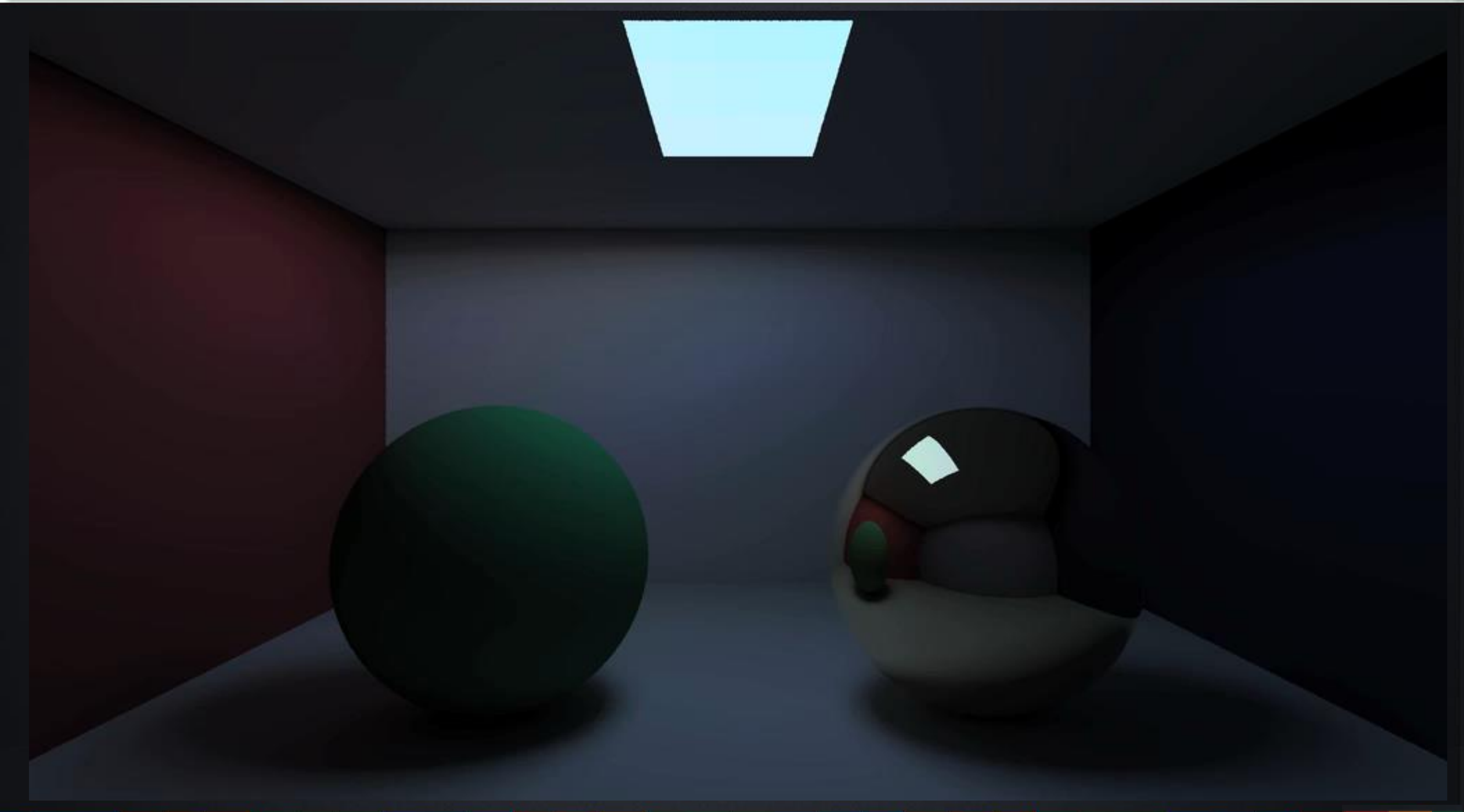
```
#?RADIANCE
VIEW= -vtv -vp 10 -4 5 -vd 0 20 0 -vu 0 0 1 -vh 80 -vv 50 -vo 5
-va 0 -vs 0 -vl 0
WAVELENGTH_SPLITS= 780 588 480 380
NCOMP=20
FORMAT=Radiance_spectra

-Y 1138 +X 2048
```

Include dimension string!

```
for d from 0 to 1134
        duplicate header as step%04d.hsr

rsplit -ih -iH < sky_frames.hsr -ob21 -a step????.hsr
```

21 bytes per spectral pixel for NCOMP=20

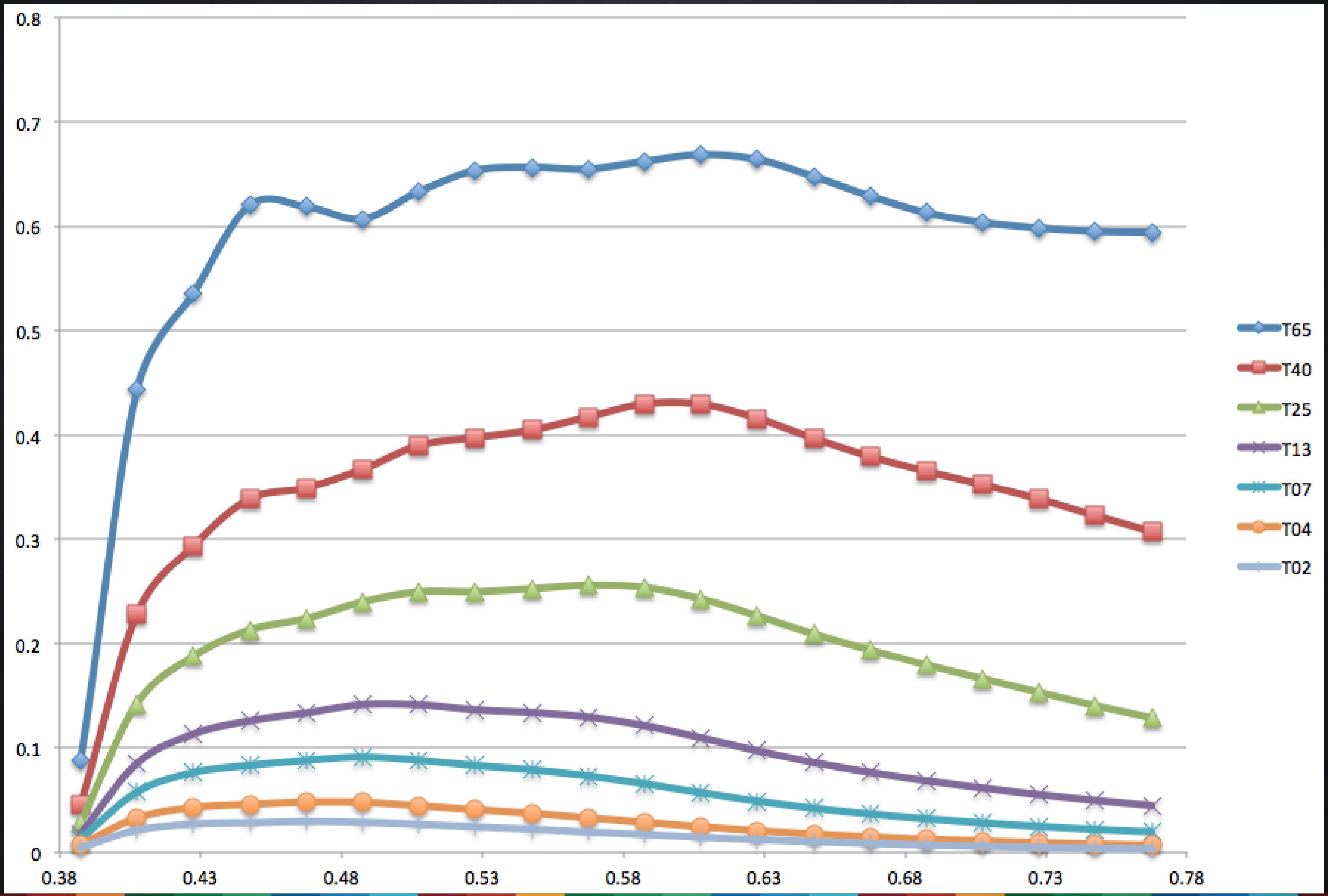
900x Time-lapse



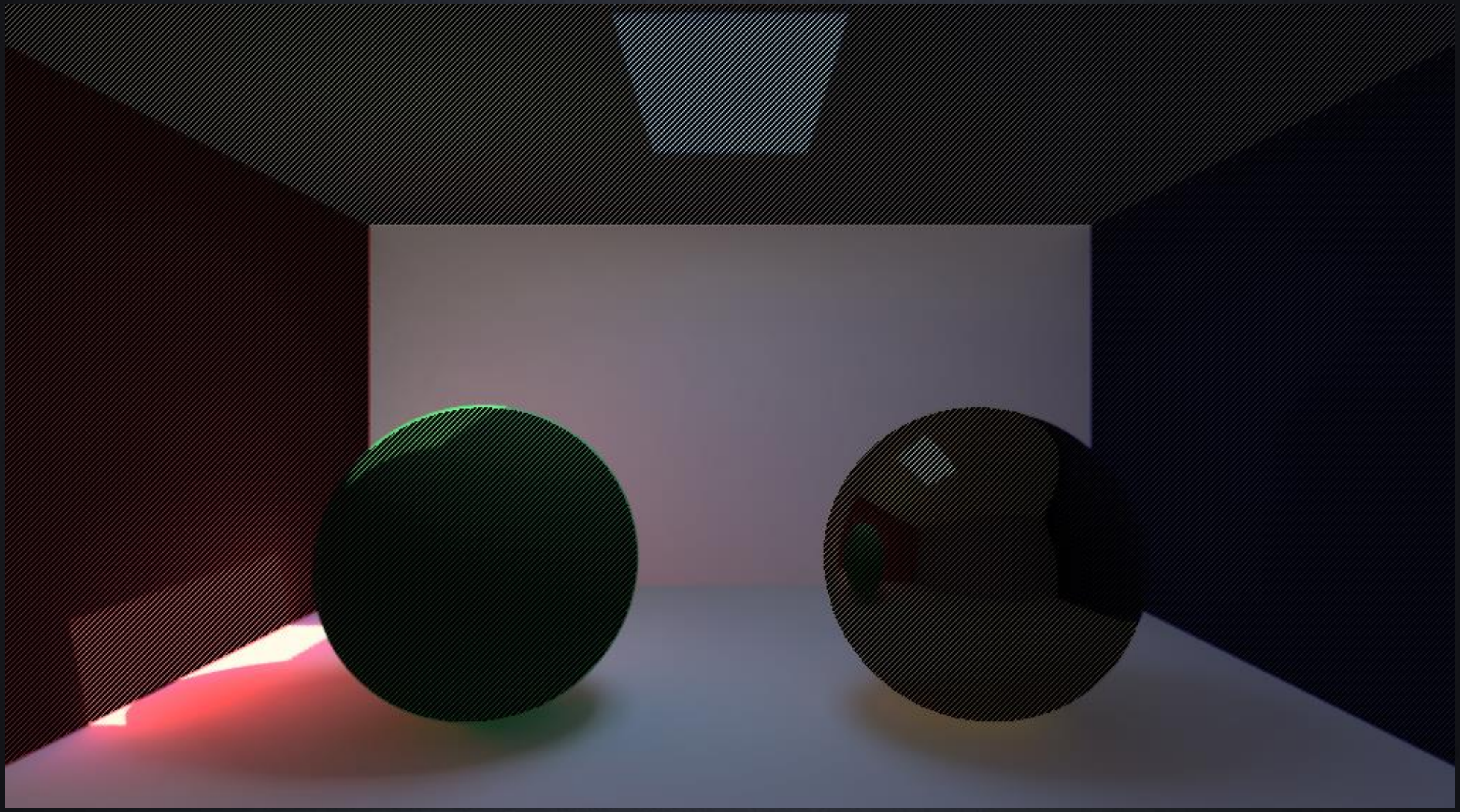
Adding Electrochromics

1. Get spectral transmittance for different EC control voltages
2. Create mask to average luminance
3. Determine in-range control voltage to achieve desired average luminance per time step
4. Compute transmission spectrum for each time step and apply to frames

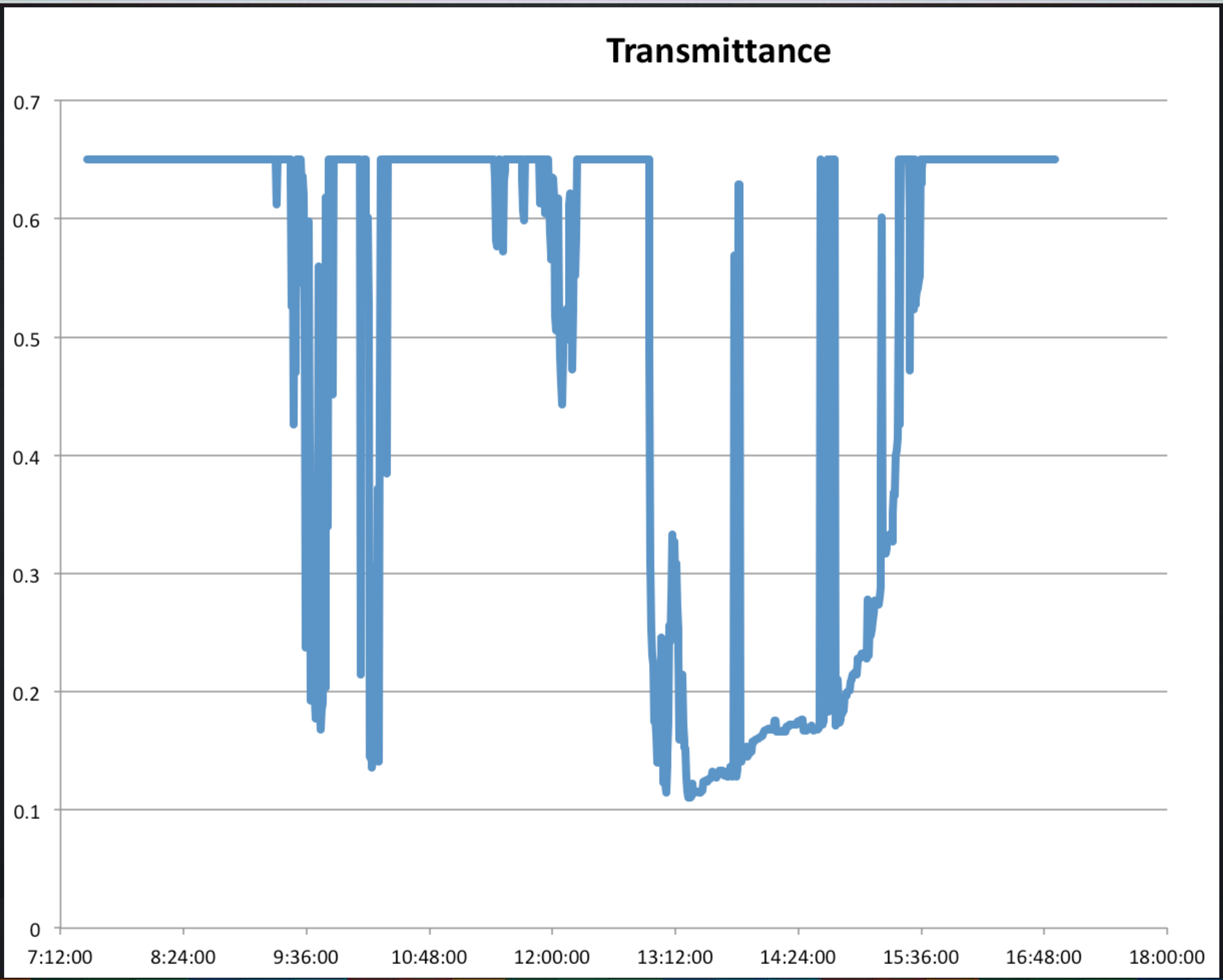
1. HALIO Spectra versus Transmittance



2. Mask for Averaging



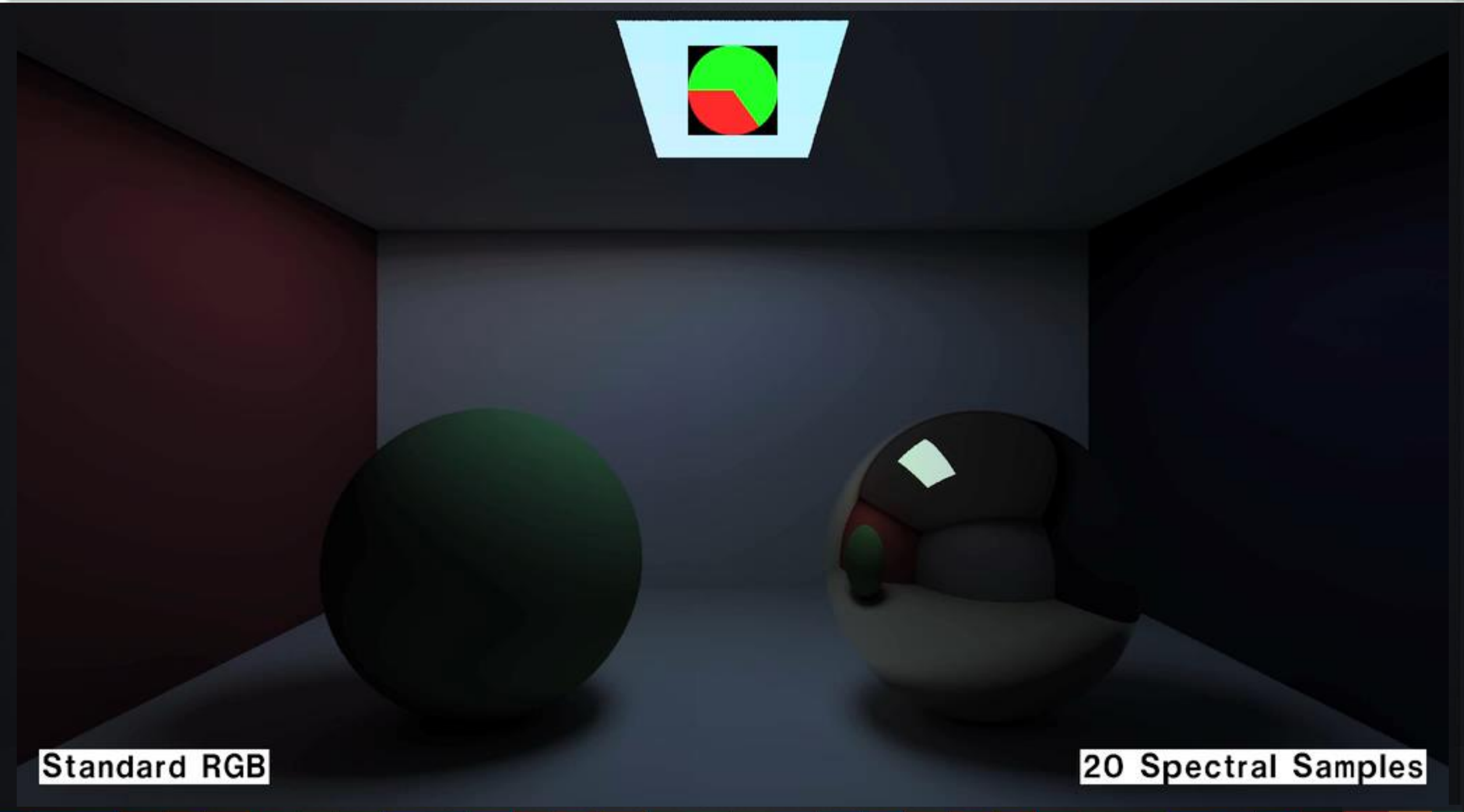
3. Control Settings over Day



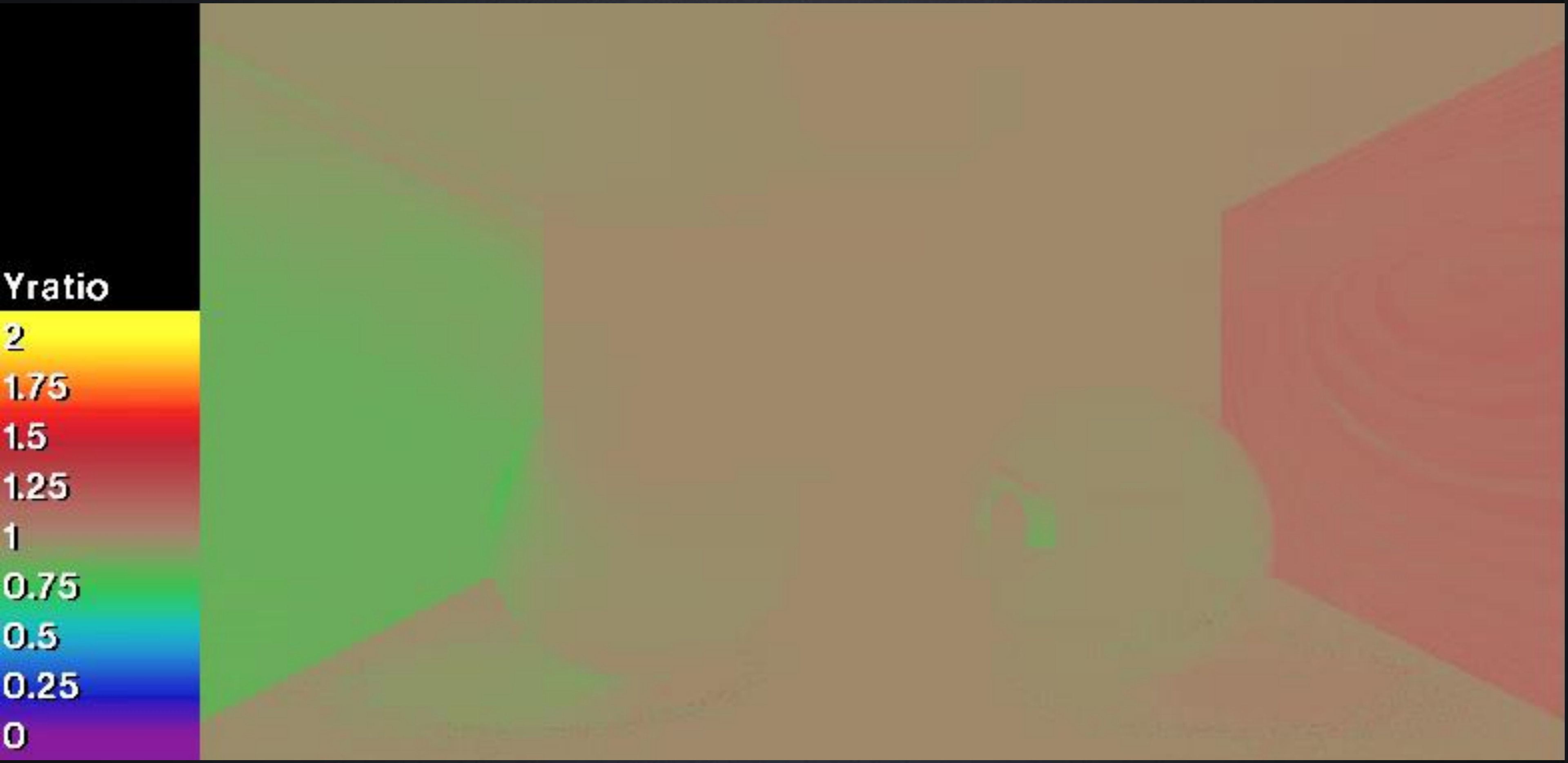
4. Apply Spectra to Frames

```
rcalc -f transmap.cal -o spec_sf.fmt spec_select.txt \
        > electrochromic_sf.txt
mkdir frames
set i=0
while ($i < 1135)
        set ii=$i
        if ($i < 1000) set ii=0$i
        if ($i < 100) set ii=00$i
        if ($i < 10) set ii=000$i
        @ i++
        set f=SPEChalfmin/slow$ii.hsr
                rcomb -fc -s `sed -n ${i}p electrochromic_sf.txt` $f \
                -c RGB > /tmp/t$$.hdr
        pcond -I <../phisto.txt /tmp/t$$.hdr \
                | ra_tiff - frames/out$ii.tif
end
```

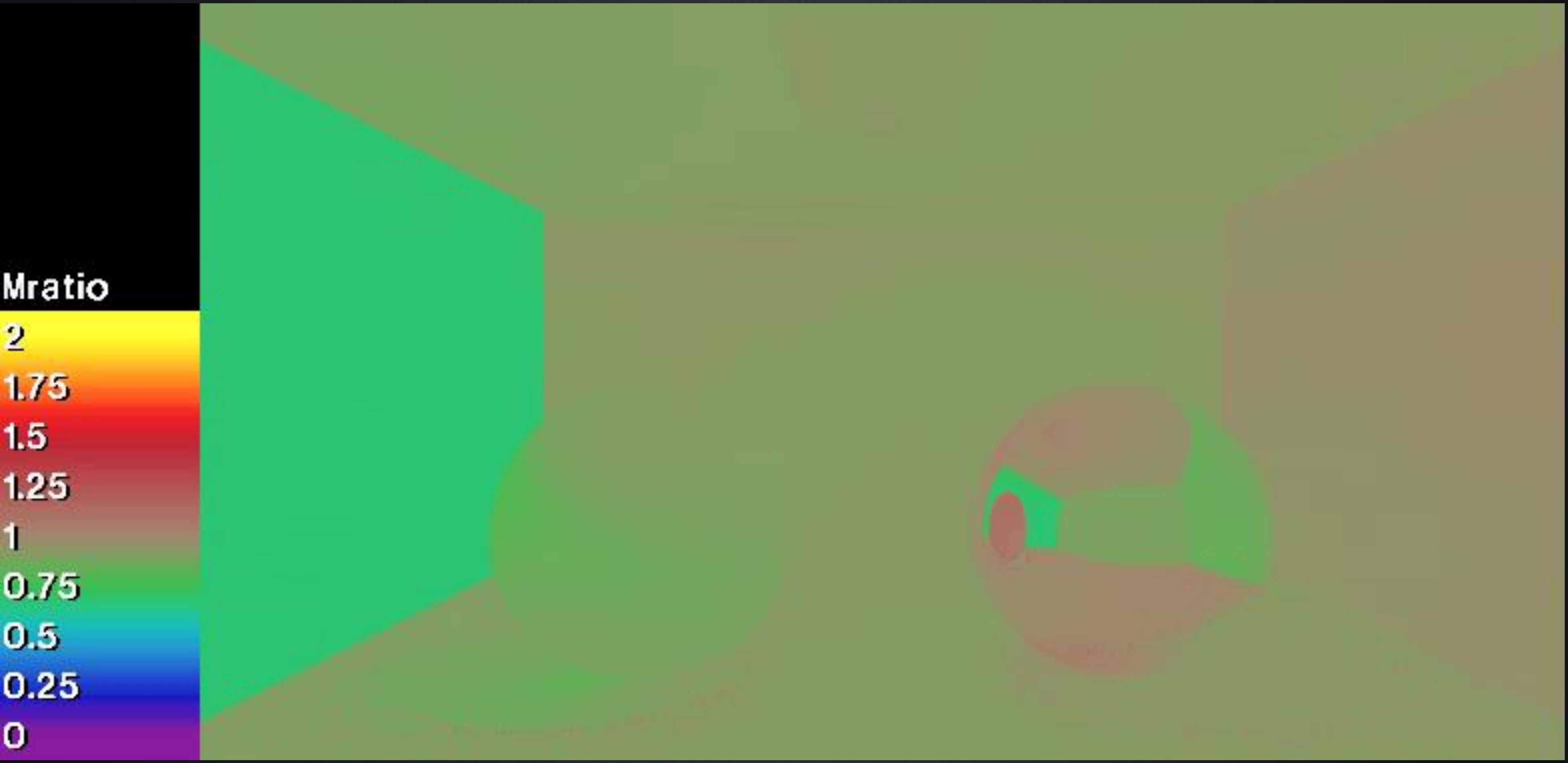
Results RGB vs. Spectral



Relative Luminance Error



Relative Melanopic Error



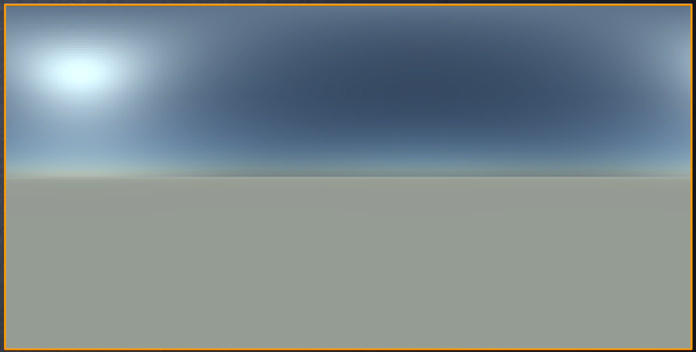
Taoning's New **genssky** and **gensdaymtx** Tools

- **genssky** creates an HSR image of sky based on atmospheric scattering simulation
- **gensdaymtx** does the same, but uses Reinhart sky subdivision to generate matrix from weather tape
  - New **epw2wea -a** option adds EPW atmospheric information needed by scattering model

genssky Example

gensky 8 16 14 -y 2024 -f Aug16@14 -n 4 -r 256 \
-a 40.7447625 -o 111.8901784 -m 90 > Aug16@14.rad

```
void spectrum sunrad
0
0
22 380 780 0.457 0.798 0.824 1.035 1.113 1.109 1.104 1.134 1.139 1.139 1.134 1.110
1.098 1.058 1.051 1.010 0.971 0.941 0.906 0.868

sunrad light solar
0
0
3 7363797.3 7363797.3 7363797.3

solar source sun
0
0
4 -0.438911 -0.348285 0.828284 0.533

void specpict skyfunc
5 noop ./Aug16@14_sky.hsr . 'Atan2(Dy,Dx)/PI+1' '1-Acos(Dz)/PI'
0
0
```

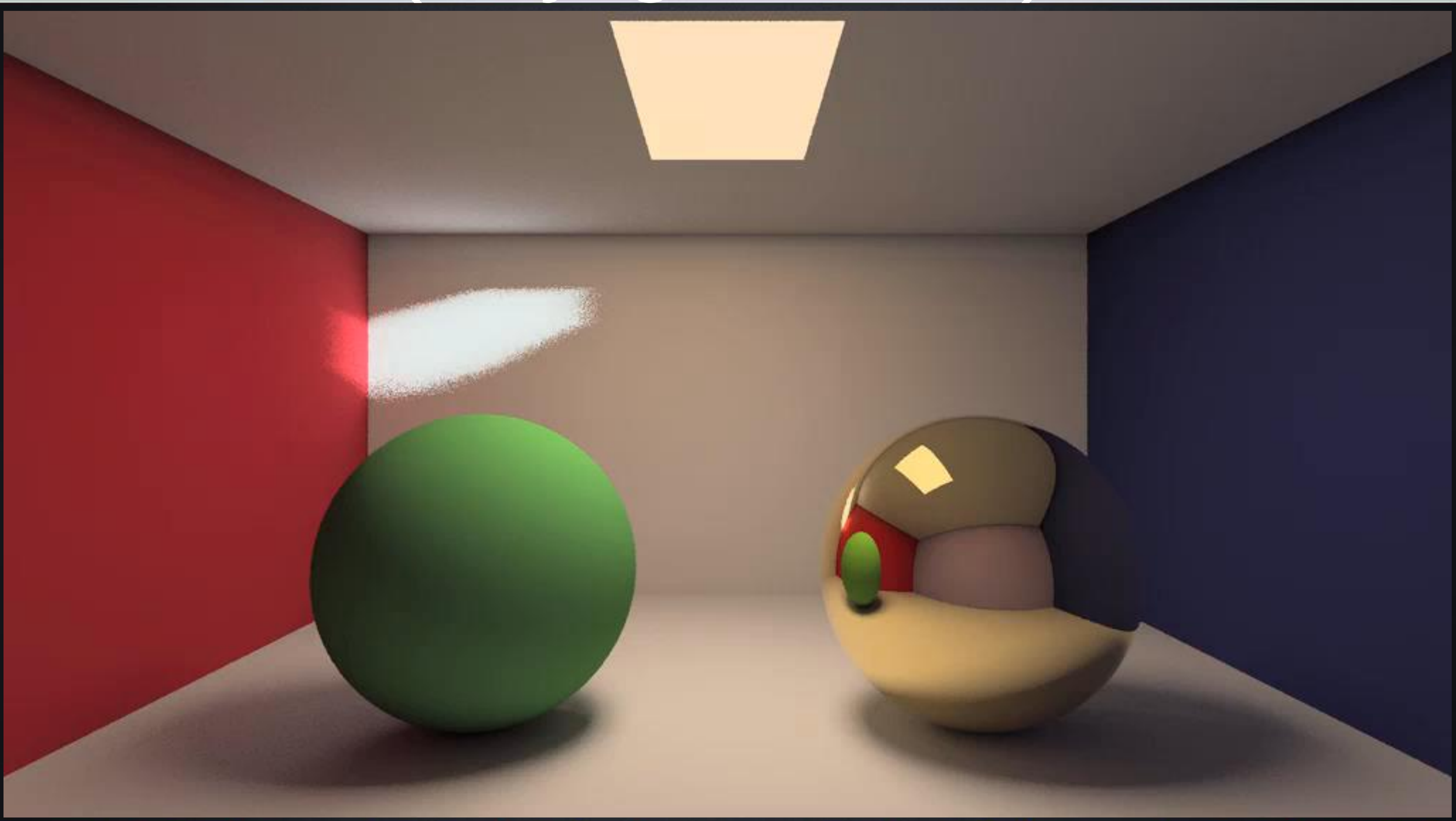


gensdaymtx Example

```
epw2wea -a CAN_ON_Ottawa_CWEC.epw \
        | gensdaymtx -n 8 -u -m 4 -of \
        | rcrop 1 0 0 0 > CAN_ON_Ottawa_SPEC4day.mtx

rcomb -n 8 -fc sky_spec_r4.mtx -m CAN_ON_Ottawa_SPEC4day.mtx \
        | rsplit -ih -iH -ob21 -a SPECyear/*.hsr
```

- Similar to RGB➔Spectral Sky method
- **rcomb** concatenates spectral rcontrib **V** matrix with **gensdaymtx** output
- **rsplit** divvies results to HSR frames

Year-long Spectral Simulation (Daylight Hours)



Extending Spectral Bounds

- 380-780 nm is used for most situations
- Extending to NIR works OK, but thermal IR ~3000-100000 nm is issue for uniform wavelength steps
- Uniform steps in frequency would make more sense, but conflicts with current measurement conventions

QUESTIONS?