BSDFs in Radiance

Greg Ward, Anyhere Software

Bidirectional Scattering Distribution Function

$$L(\theta_r,\phi_r) = \iint L(\theta_i,\phi_i)\boxed{f(\theta_i,\phi_i;\theta_r,\phi_r)}\left|\cos(\theta_i,\phi_i)\right| d\theta_i\, d\phi_i$$

- "Bidirectional" because physics of light reversable
- "Scattering" = term for reflection + transmission
- "Distribution Function" because it is a PDF that sums to the total scattering probability
    - (PDF = "Probability Distribution Function")

Historical Perspective

- Since 1990, *Radiance* supports reflectance and transmittance sampling of built-in BSDF models
  - Latest version: WGMD & Ashikhmin-Shirley
- Other “general” materials do not sample specular:
  - *plasfunc*, *metfunc*, *transfunc*, *BRTDfunc*, *plasdata*, etc.
- BSDF measurements were largely unavailable...

Recent BSDF Impetus

- PAB-Opto pgII provides BSDF measurements
  - Existing parametric models usually fit poorly
- WINDOW 6.0 support of complex fenestration
  - Klems representation allows efficient sampling and facilitates daylight coefficient reformulation
  - Desire for efficient annual daylight simulations

BSDF Developments

1. Added support for WINDOW 6 input to **mkillum**
2. Wrote **rtcontrib** (replaced by better **rcontrib**)
3. **genBSDF** to compute BSDF matrix from geometry
4. Created 3-phase annual simulation method
5. Tensor tree representation for “peaky” BSDFs
6. Added BSDF type to *Radiance* scene language

Developments (cont'd)

7. Removed support for WINDOW 6 input to **mkillum**
8. Added Ashikhmin-Shirley parametric model
9. **dctimestep** upgrade for **gendaymtx**
10. 5-phase annual simulation method (A.McNeil)
11. Wrote **bsdf2klems** & **bsdf2ttree** utilities
12. Working on **pabopto2bsdf** interpolation tool

Some Likely Questions

- How can I get a BSDF into *Radiance*?
- When do I need to use Klems vs. Tensor Tree?
- Do I still need **mkillum** (if so, when)?
- How many phases are in an annual simulation, anyway?

BSDF Acquisition/ Import Methods

A. WINDOW 6 provides XML file

B. **genBSDF** calculation from scene description

C. **bsdf2klems** or **bsdf2ttree** from procedure

D. **pabopto2bsdf** from raw pgII measurements*

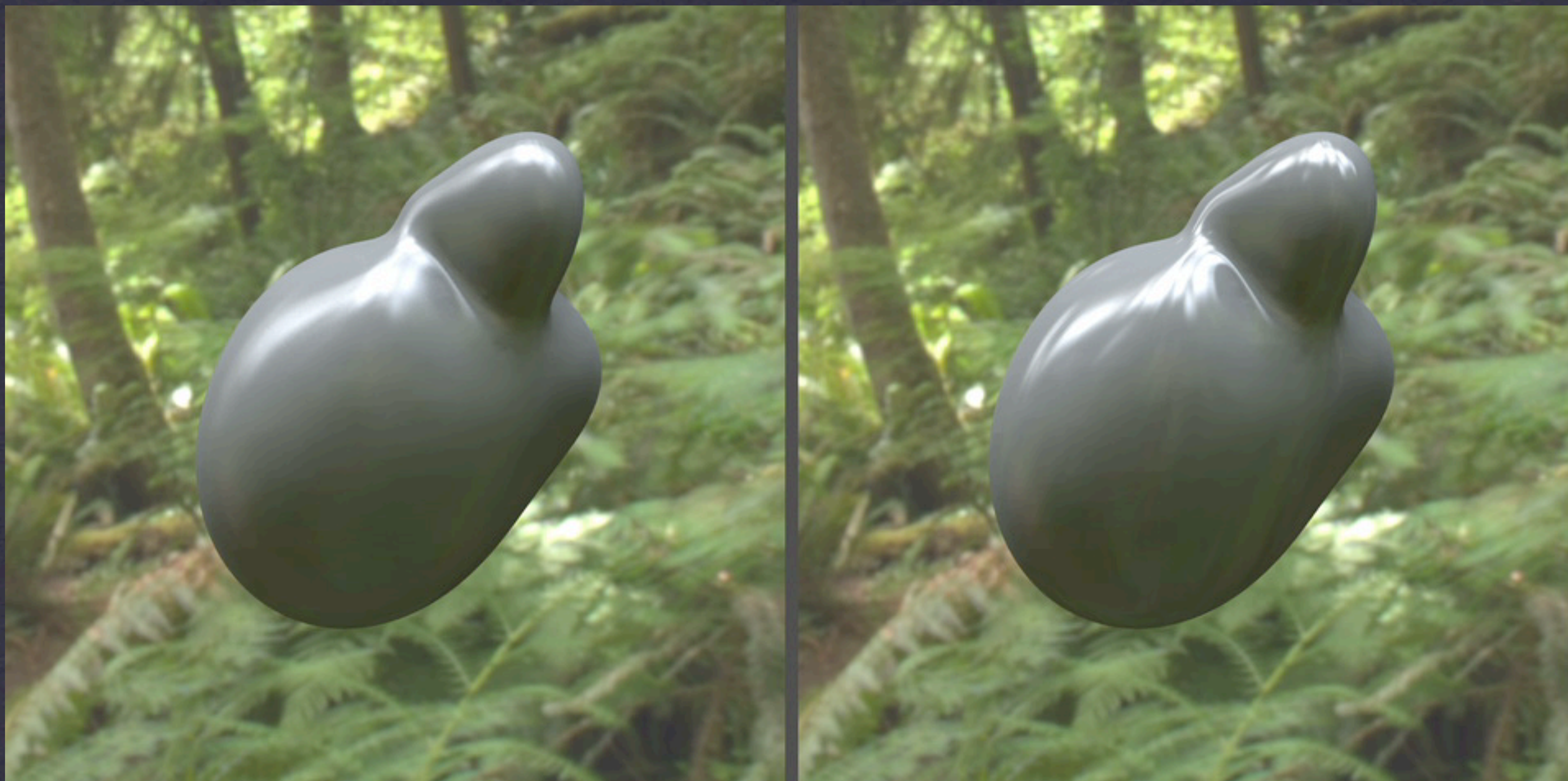
E. Measured BSDF(s) fed back into **genBSDF**

*We will revisit **pabopto2bsdf** later in this talk

Klems vs. Tensor Tree

- Each has its strengths and weaknesses
- Klems matrix BTDF enables 3-phase method
- Tensor Tree is more accurate where applicable
- Andy's 5-phase method can employ both types

BSDF VIEWER



Klems (left) vs. Tensor Tree (right)

Anisotropic Ward-Geisler-Moroder-Duer BRDF

Whither (Wither?) **mkillum**?

- Is **mkillum** still useful, and if so, when?
  - Yes, to improve interior rendering time/results
  - Only difference now is window may have BSDF
  - Indirect lighting from BSDF too slow, noisy
- 3-phase and 5-phase results will never be as nice

3-phase? 5-phase? I’m Getting Aphasia

- Original DC method is a 2-phase calculation:
  1. compute daylight coefficients
  2. apply sky distribution for each time step
- 3-phase method separates DC into 2 components:
  A. sky-to-window daylight coefficients

  B. window-to-interior view coefficients

Three Phases Refer to Calculation Steps

I. Compute sky-to-window (daylight) coefficients

II. Compute window-to-interior (view) coefficients

III. Compute time steps (i.e., **dctimestep**)

The separation of coefficieints allows us to alternate window transmission matrices:
i = VTDs

5-phase Method

(This explanation intentionally left blank.)

BSDF Material Primitive Added in *Radiance* 4.1

- General, data-driven reflectance and transmittance distribution function
- Simple syntax relies on XML (eXtensible Markup Language) auxiliary file
- XML file may be imported from WINDOW 6 or created using **genBSDF**
- Proxy mode reveals detailed model underneath, similar to *illum* behavior

```
void BSDF m_bsdf110b
6 0 bsdf110b.xml 0 1 0 .
0
0
```

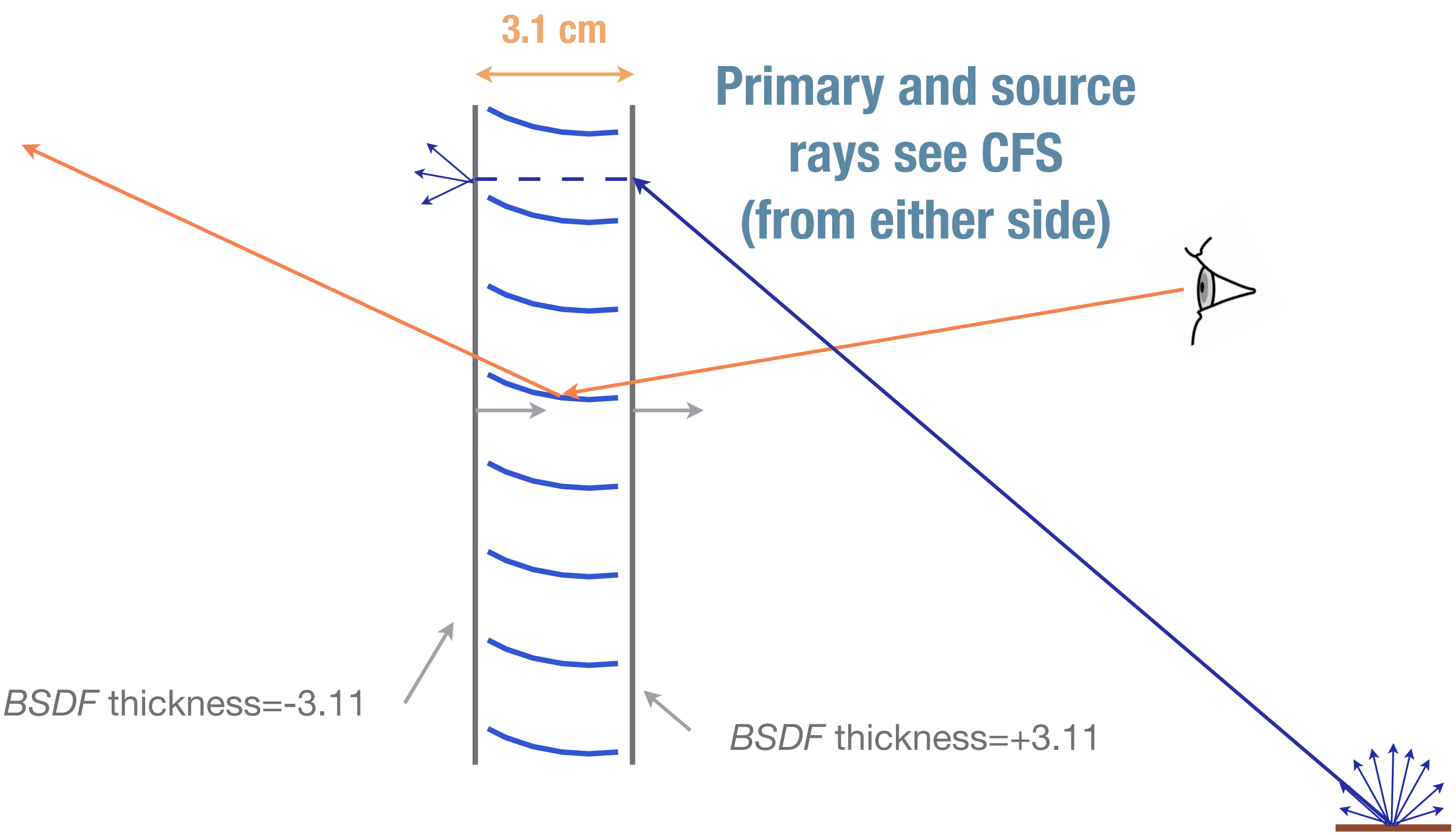
Placeholder for function file (if needed)

Thickness (non-zero for proxy)

Auxiliary XML with BSDF data

Up orientation vector

Proxy Example



Indirect rays use *BSDF* sampling methods

Using **genBSDF** to Create XML File

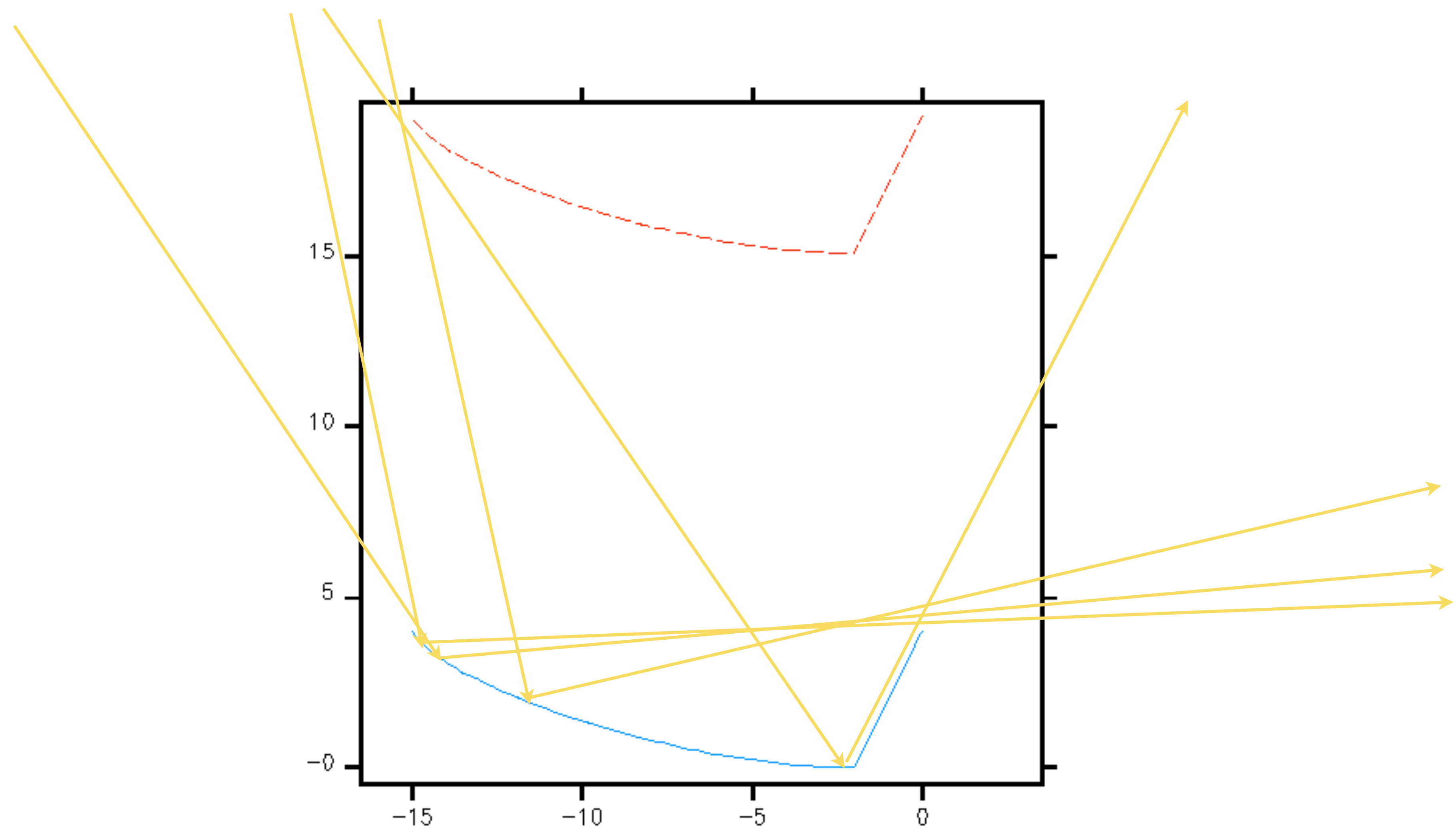
```
genBSDF +geom centimeter blinds.rad > blinds.xml
```

Using **pkgBSDF** to extract geometry:

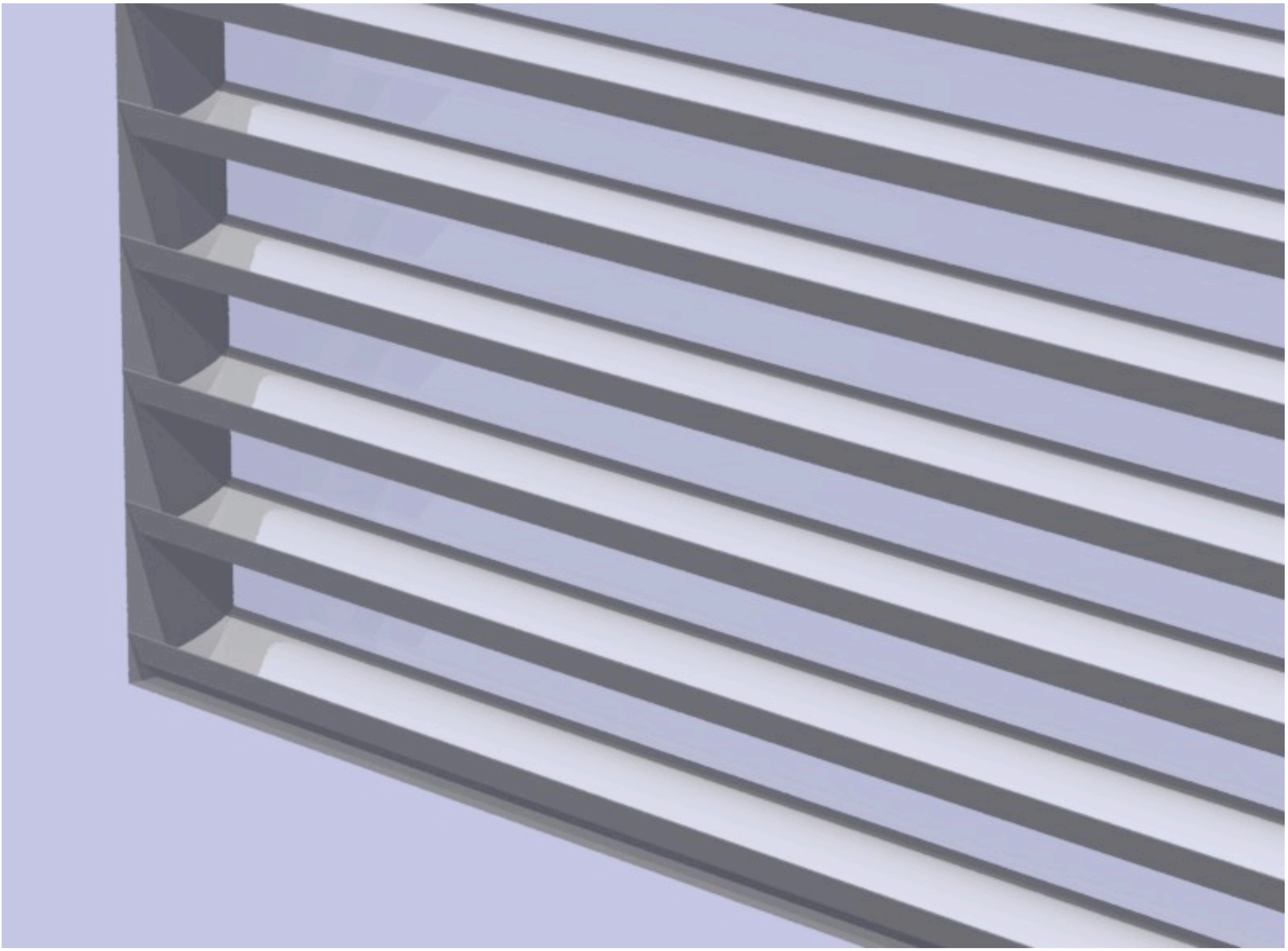
```
!pkgBSDF -s blinds.xml
```

Converts MGF to *Radiance* and places proxy surfaces in front and behind (if appropriate)

Example Glazing System with Embedded Slats (Top specular, Bottom matte black)

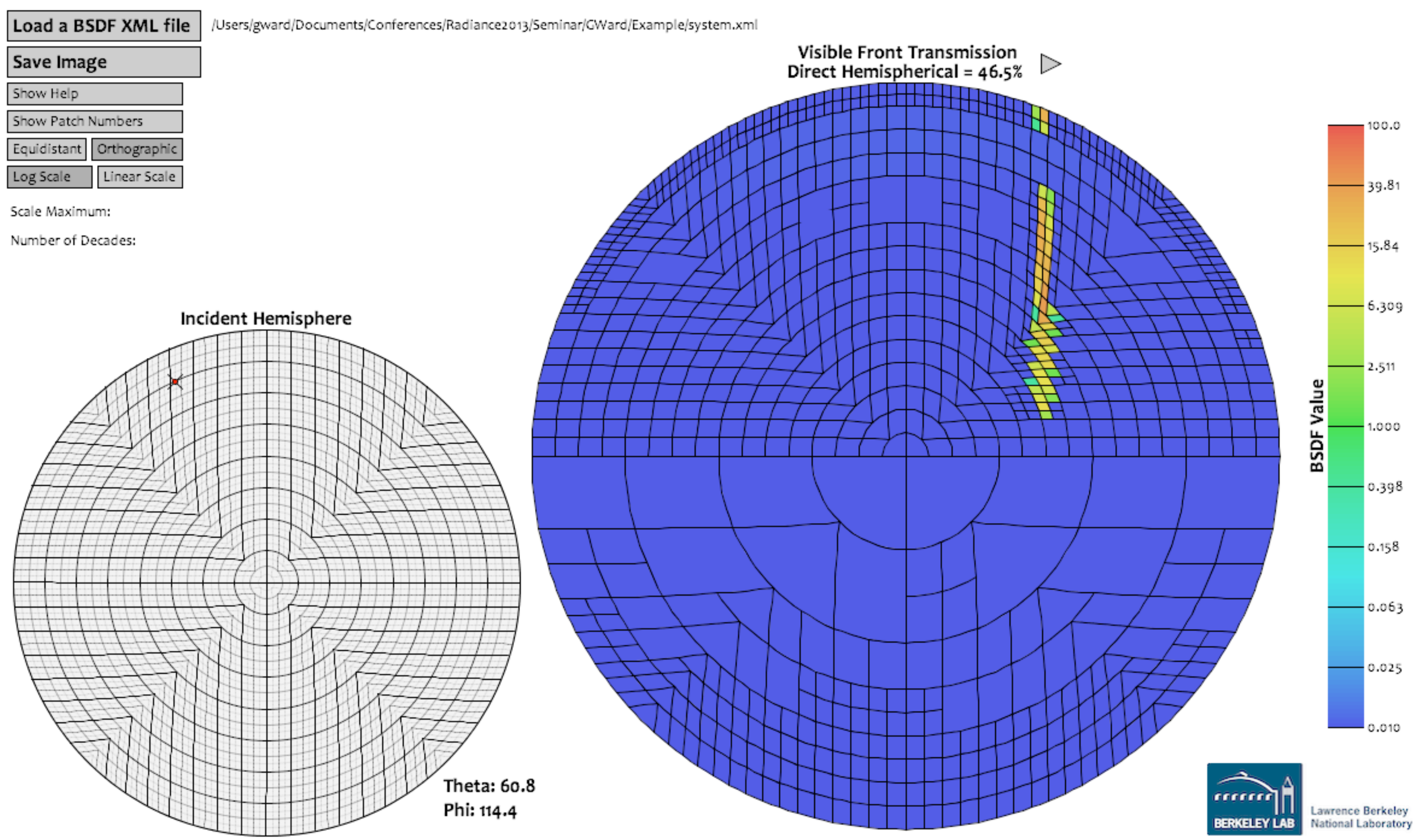


Complete System (Includes Front & Rear Glazing)



Compute Tensor Tree BSDF

```
genBSDF -t4 6 -n 4 +mgf +geom millimeter \
        -r '-ab 1' system.mgf > cfsystem.xml
```



Creating a BSDF-only Window Model

```
#
# CFS (BSDF-only) without proxy geometry
#

#@mkillum i=m_cfsystem_f c=d m=cfs_noprox

void BSDF m_cfsystem_f
14 0 cfsystem.xml 0 1 0 . -rz 180 -rx -90 -t 2.5 0 2.25
0
0

m_cfsystem_f polygon cfs.cfsystem_f
0
0
12
     4  0  1.25
     1  0  1.25
     1  0  3.25
     4  0  3.25
```

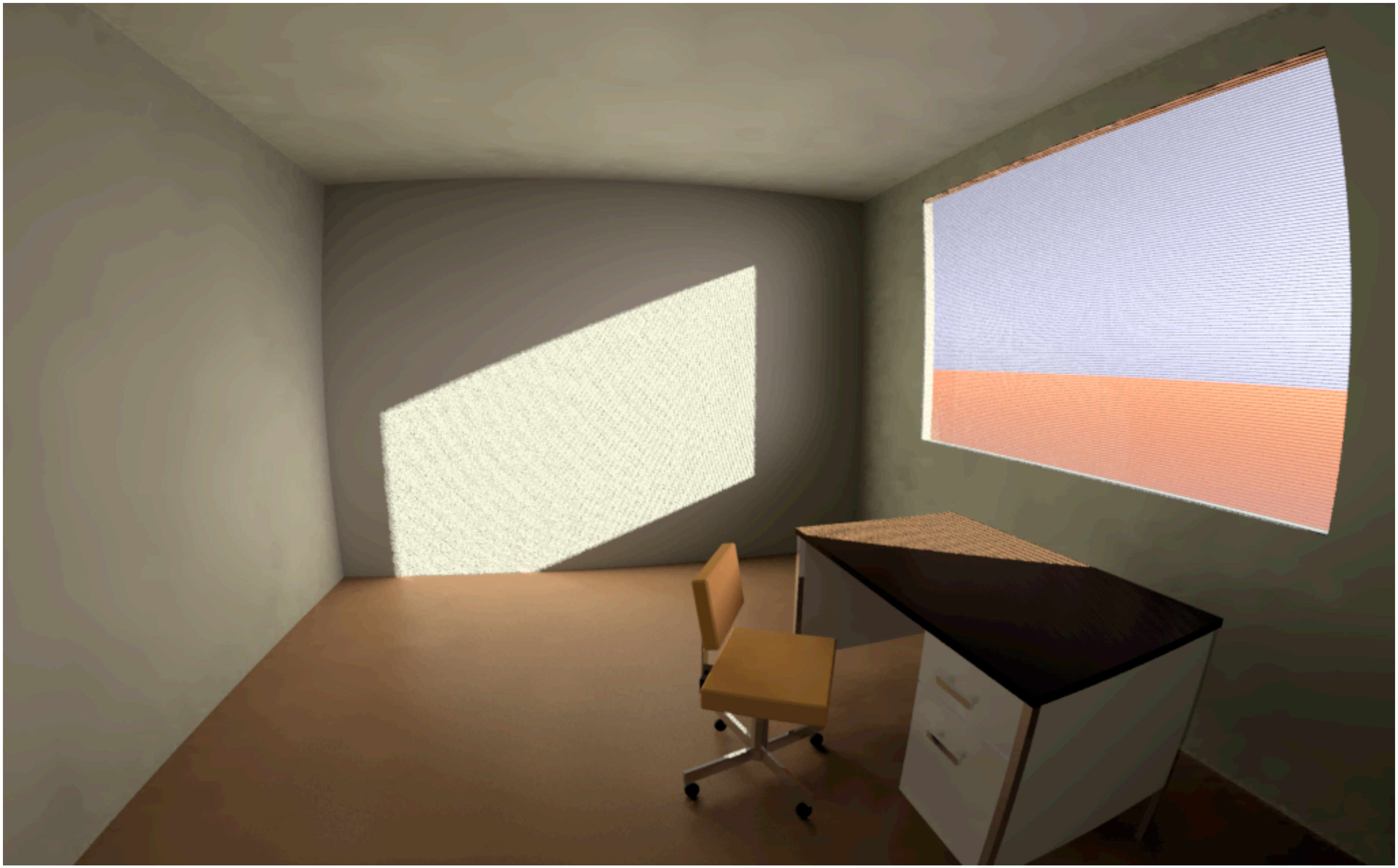
Setting up a **rad** Input File

```
#
# Render scene using BSDF without proxy and mkillum
# Equinox
#
ZONE = Interior  0 5  0 5  0 3.5
EXPOS = -1
scene = "!gensky 9 21 13 -g .15"
scene = exterior.rad office.rad furniture.rad
objects = desk.rad orange_chair.rad cfsystem.xml
illum = cfs_noprox.rad
AMBF = eqbsdfmo.amb
QUA = High
IND = 1
VAR = Med
DET = Low
PEN = True
view = int -vf int.vf
RES = 1024
mkillum = -as 0 -ad 1024 -aa 0 -lr -8 -u+
```

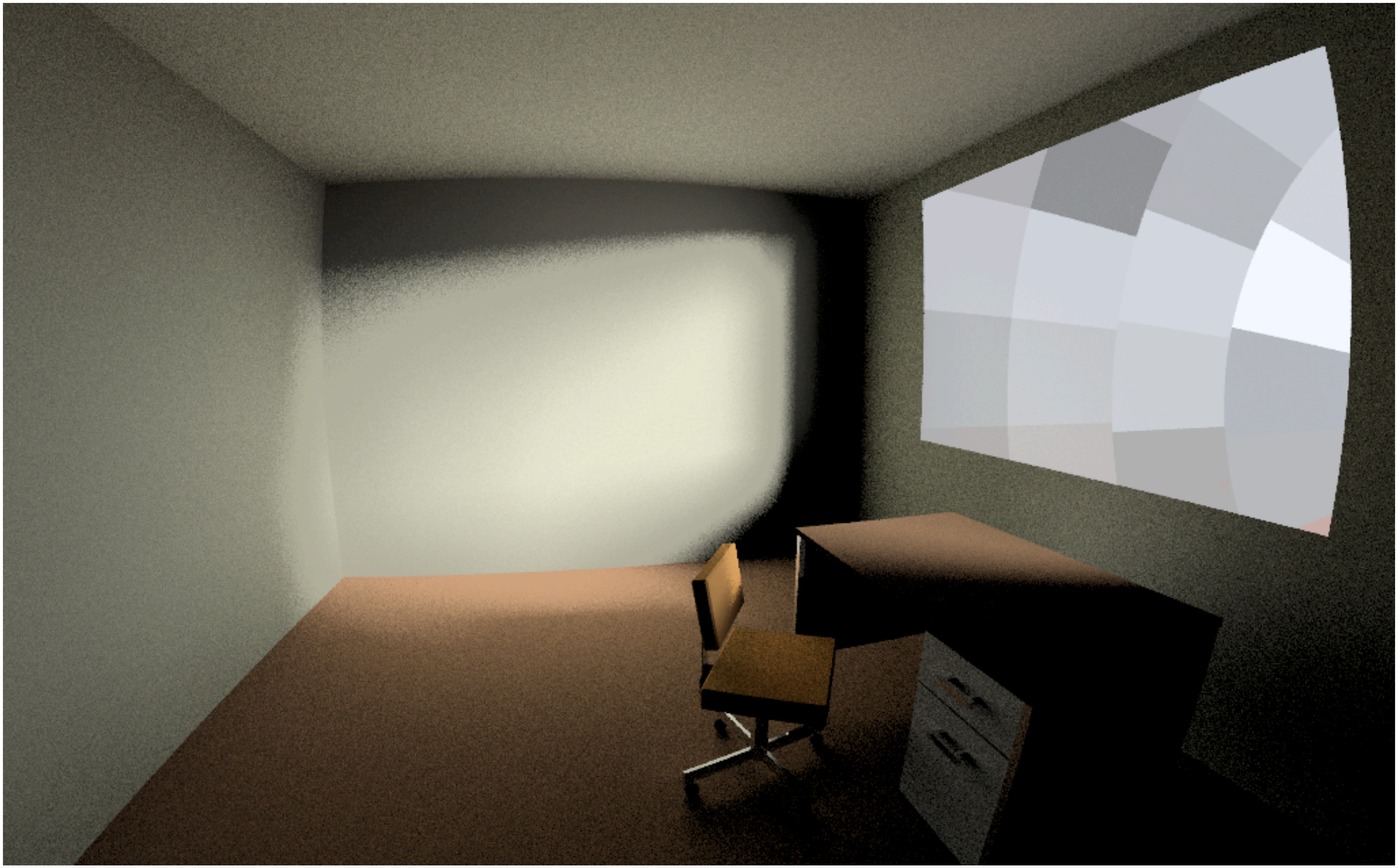
Rendering BSDF-only Using **mkillum** (**rad** does this)

Same Model & Method
Midwinter at 3pm

Using Proxy Geometry (**pkgBSDF** -s cfsystem.xml)



Using 3-phase Annual Simulation Method



Phase I: Render Exterior Paths Sky➞Window

```
oconv dummy_exterior.rad dummy_window.rad office.rad \
  > exterior.oct

genklemsamp -c 2000 -ff -vd 0 -1 0 dummy_window.rad \
  | rcontrib -ff -c 2000 -e MF:2 -f ~/cal/reinhart.cal \
    -bn Nrbins -b rbin -n 4 -m skyglow exterior.oct \
    > reinhart2.dmx
```

Phase II: Render Interior Paths Window→View

```
oconv dummy_window.rad office.rad furniture.rad \
     > dummy_office.oct

vwrays -ff -c 9 -pj .9 -vf int.vf -x 1024 -y 635 -pa 0 \
 | rcontrib -ffc -ab 2 -ad 1000 -lw 1e-4 -x 1024 -y 635 \
   -c 9 -f ~/cal/klems_int.cal -bn Nkbins -b kbinS \
   -o comp/comp%03d.hdr -m black_glass -n 4 dummy_office.oct
```

Phase III: Time-step Calculation Using BSDF

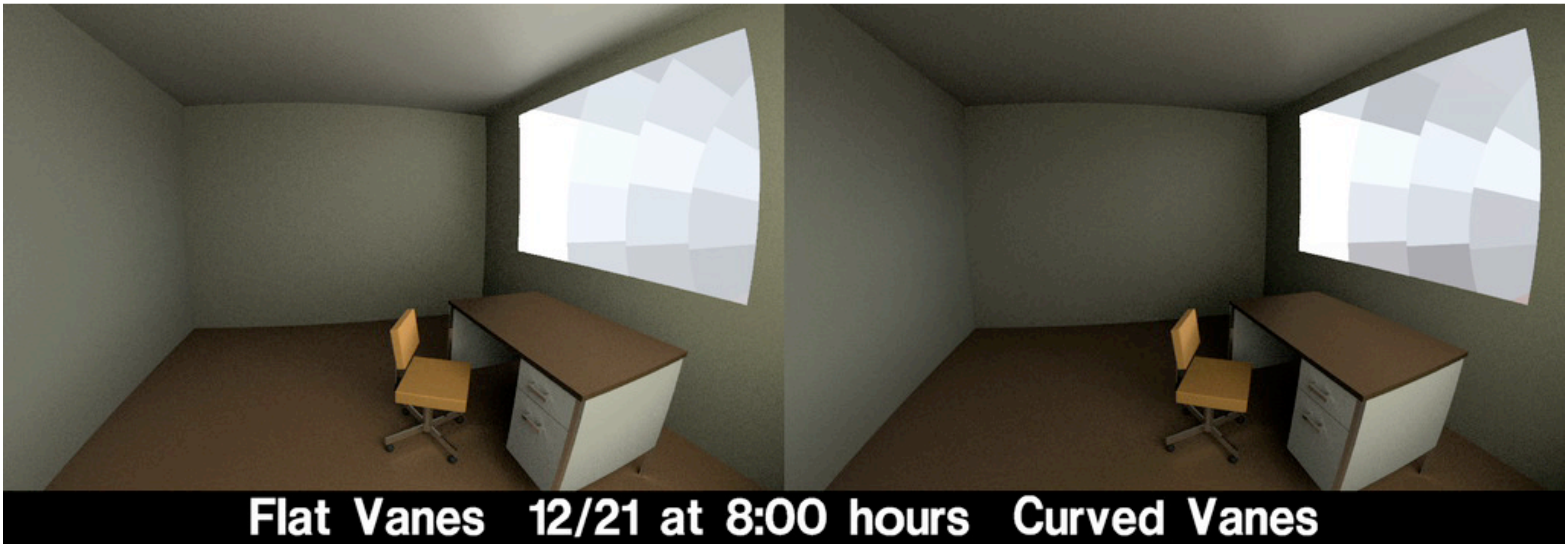
First, we need to convert our Tensor Tree to a Klems matrix representation using our new tool:

```
bsdf2klems cfsystem.xml > cfsystem_k.xml
```

Now, we can render a particular day and time:

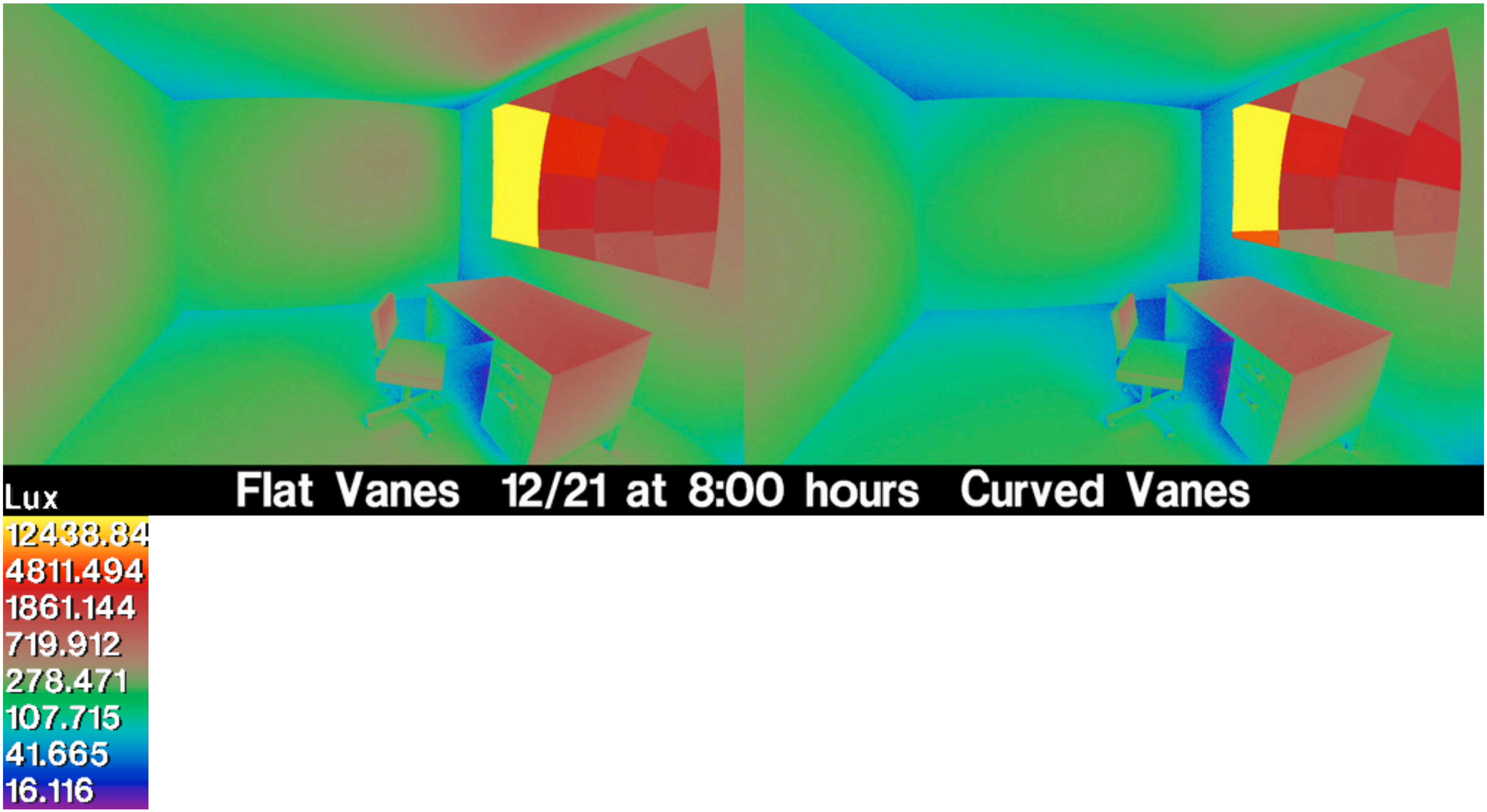
```
gensky 9 21 13 -g .15 \
 | genskyvec -m 2 \
 | dctimestep comp/comp%03d.hdr cfsystem_k.xml \
   reinhart2.dmx > eqbsdf3phase.hdr
```

We Can Repeat This Calculation Quickly



This also explains why we used 3-phase method over simpler daylight coefficients: comparing CFSs

Rerun **rcontrib** to Compute Illuminance on Surfaces



What We Learned from Our Example

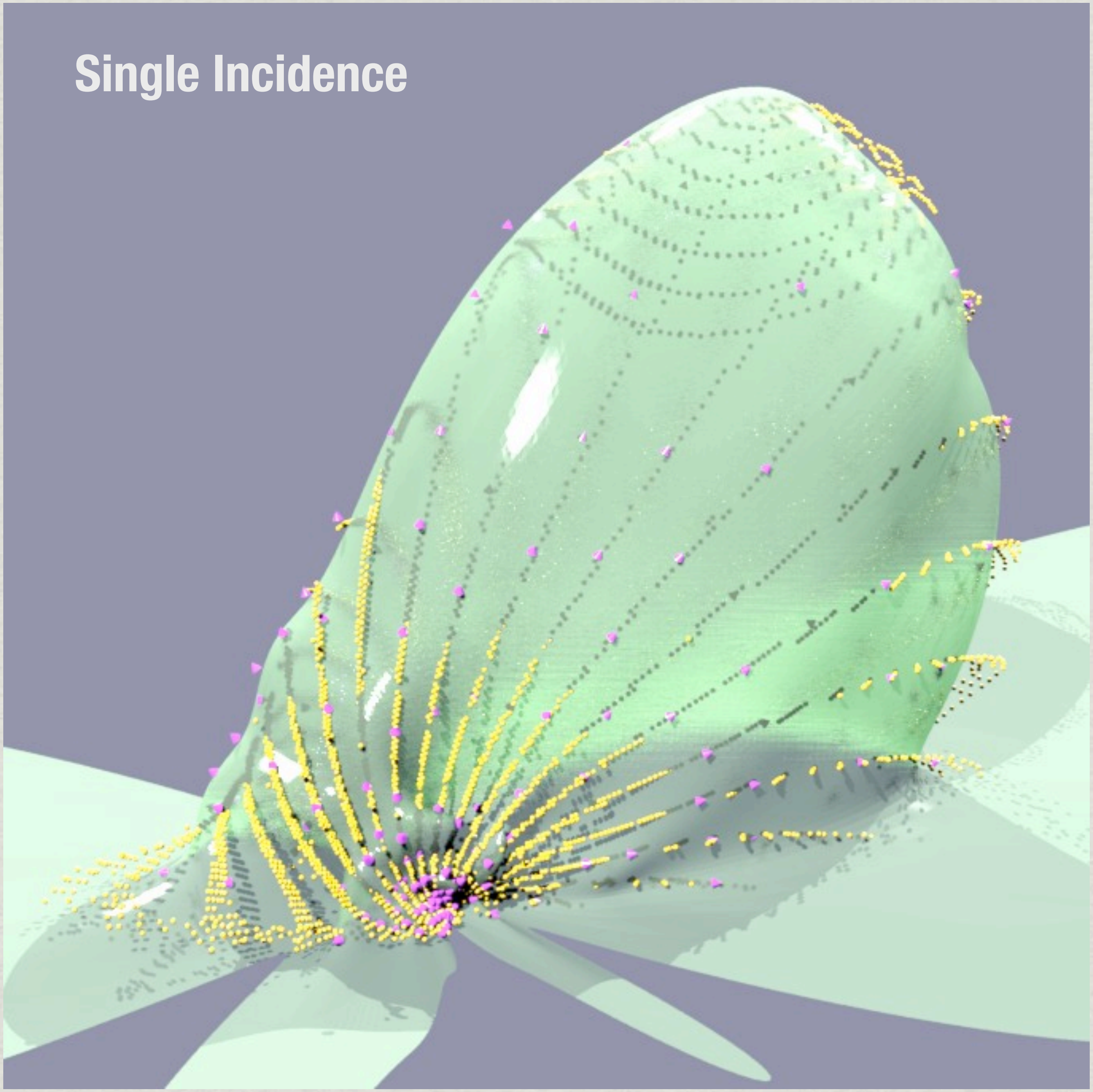
- **mkillum** is still very useful for testing and rendering
  - could not compute a single view in a reasonable time without it
  - **rad** is also useful for keeping track of everything
- Tensor Tree representation is good, put proxy geometry is best for direct component
- Time-lapse animation is a nice way to learn about CFS behavior
  - analyzing a new CFS design is actually kind of fun

Coming soon: **pabopto2bsdf**

- Tackles difficult problem of interpolating measured BSDF into continuous distribution for resampling
- Input is a set of pgII scattering measurements, one data file per incident direction
- Output is an "interpolant" file for **bsdf2klems** and **bsdf2ttree**, which can produce final XML files

BSDF Interpolation Method

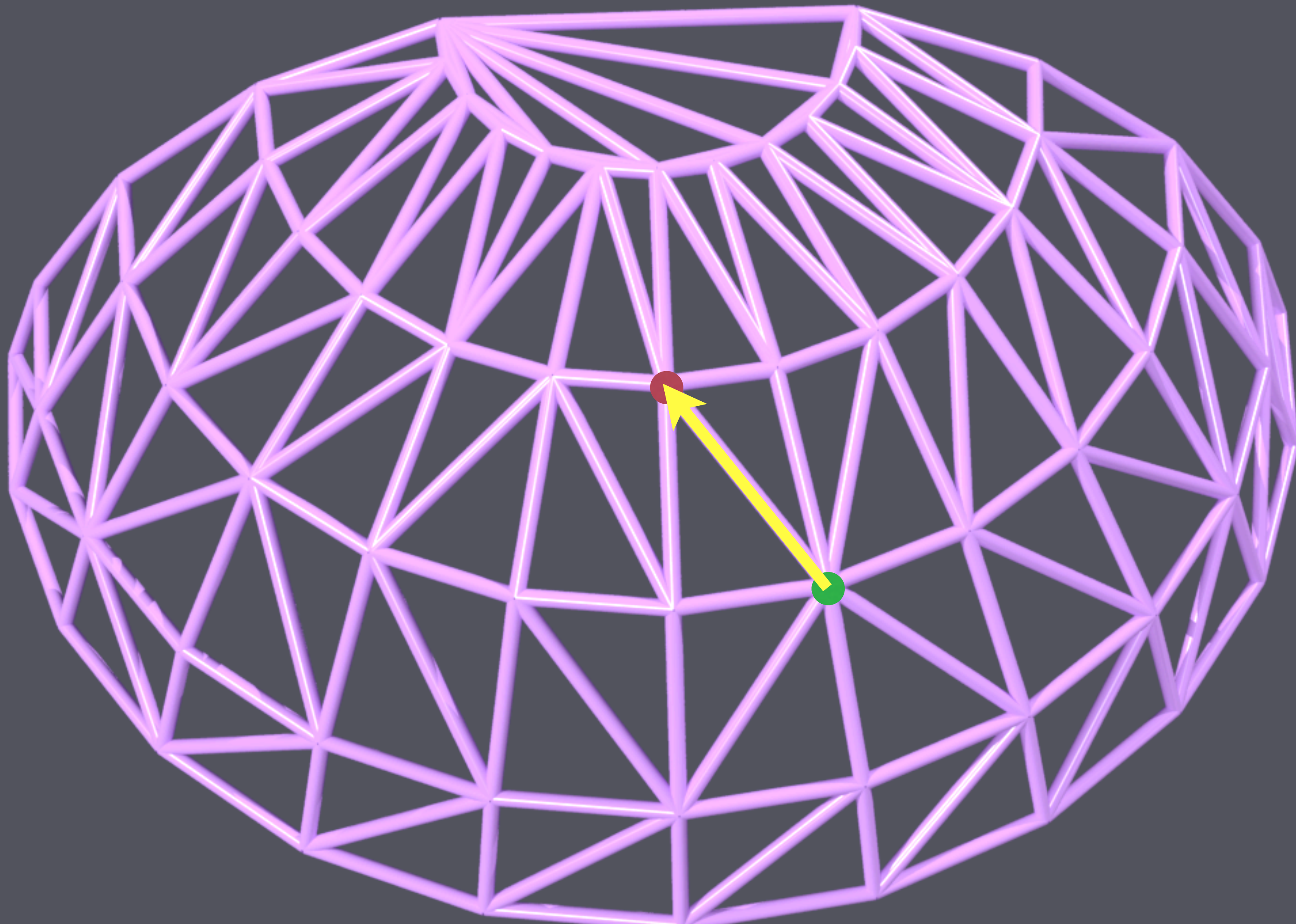
- Fit a sum of Gaussian lobes to culled output distribution samples (i.e., Radial Basis Function)
- Detect/reproduce measurement symmetry
- Compute “migration coefficient matrix” between each pair of adjacent incident directions
- RBFs (Gaussians) with migration matrices comprise *Scattering Interpolant Representation*



Yellow dots are original dense measurements

Pink dots are culled samples

Green sheet is sum of Gaussian lobes

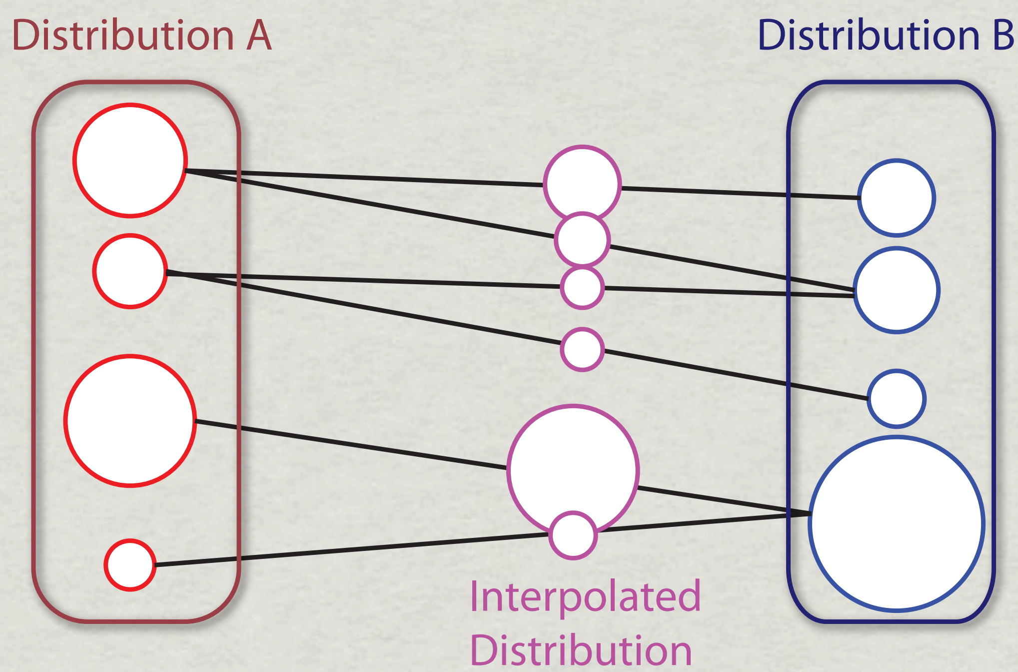


Migration matrix describes how distribution changes from one set of Gaussian lobes to another

Example with Bilateral Symmetry

Delaunay triangulation on hemisphere — each vertex is a fit of Gaussian lobes

Advection and EMD



- Earth Mover's Distance minimizes cost of migration matrix **M**
- Source (A) and destination (B) distributions typically have a different number of lobes
- Interpolated distribution usually has more than either A or B

Advection Math

Advection Between 2 Nearest entry Points

$$\vec{D}_s M = \vec{D}_d$$

$$\vec{D}_s = \left[\frac{2\pi\sigma_{s1}^2}{E_S} w_{s1},\ \frac{2\pi\sigma_{s2}^2}{E_S} w_{s2} \dots \right] \qquad E_S = 2\pi \sum w_{si}\cdot\sigma_{si}^2$$

$$\vec{D}_d = \frac{2\pi}{E_d}\cdot\left[\sigma_{d1}^2 w_{d1},\ \sigma_{d2}^2 w_{d2} \dots \right]$$

M = rectangular migration matrix from optimizer

Each row in M must sum to 1.0, with all values non-negative

Let t = distance along migration, $0 \le t \le 1.0$

For each non-zero entry in M, M_{ij}, create an advection particle (Gaussian lobe) with peak value:

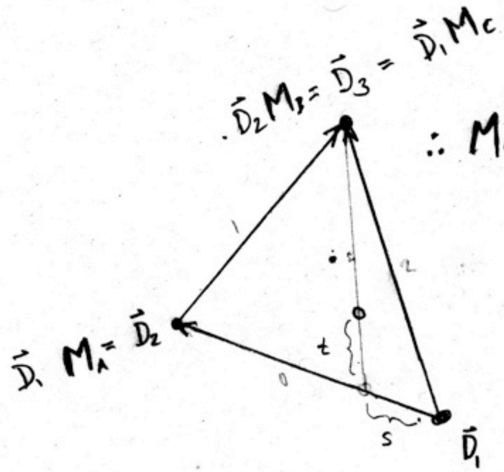
$$W_{mij}(t) = \frac{w_{si}}{E_S}\cdot M_{ij}\cdot\left[(1-t)E_S + t\cdot E_d\right]$$

radius: $\sigma_{mij}(t) = \sqrt{(1-t)\sigma_{si}^2 + t\,\sigma_{dj}^2}$

and position: $\hat{P}_{mij}(t) = \hat{P}_{si}$ rotated $t\cdot\operatorname{acos}(\hat{P}_{si}\cdot\hat{P}_{dj})$ towards $\hat{P}_{dj}$

Advection in Triangle

→ Each column of M_{ij} adds to 1.0



$\therefore M_C = M_A M_B$ is one valid matrix → underconstrained so may not be optimal

Compute s & t based on where target incidence lands relative to nearest incident meas.

Particles

peak: $w_{ijk}(s,t) = w_{1i} \cdot M_{Aij} \cdot \left[(1-s)\cdot M_{Cik}\left(1-t+t\cdot\frac{E_3}{E_1}\right) + s\cdot\frac{E_2}{E_1}\cdot M_{Bjk}\cdot\left(1-t+t\cdot\frac{E_3}{E_2}\right)\right]$ splits & adjusts particle energies

radius: $\sigma_{ijk}(s,t) = \sqrt{(1-s)(1-t)\sigma_{1i}^2 + s\cdot(1-t)\cdot\sigma_{2j}^2 + t\cdot\sigma_{3k}^2}$ adjusts particle sizes

direction: $\hat{P}_{ijk}(s,t) = \hat{R}\left(\hat{R}(\hat{P}_{1i}, s, \hat{P}_{2j}), t, \hat{P}_{3k}\right)$

where: $\hat{R}(\hat{p}_1, x, \hat{p}_2)$ is unit vector resulting from rotating $\hat{p}_1$ by $x\cdot\cos^{-1}(\hat{p}_1\cdot\hat{p}_2)$ towards $\hat{p}_2$ alters particle outgoing directions

Generate particle (lobe) for every $\{i,j,k\}$ combination of $\{M_{Aij}>0,\ M_{Cik}>0$ or $M_{Bjk}>0\}$

Evaluating BSDF

- Actual advection performed by **bsdf2klems** or **bsdf2ttree** to produce XML representation
  - Takes output from multiple **pabopto2bsdf** runs for multiple components (front/back/refl./trans.)
  - Takes minutes to hours to run, depending
- Klems matrix or Tensor Tree needed for efficient BSDF evaluation and ray sampling

Current Measurement Interpolation Status

- Mechanically, everything is working
- Accuracy of results is disappointing
- Issues with RBF fit appear early on
- Anisotropic Gaussians or better fitting method?
- Ongoing work this year

Things to Bear in Mind with BSDFs in *Radiance*

- Not a cure-all — BSDF sampling can be noisy
  - increasing **-ss** parameter often helps
- Direct component problem without proxy mode
  - even tensor tree won't model pure specular
- Think carefully about thick systems (e.g., façades)