# Complex Fenestration in *Radiance*

Greg Ward, Anyhere Software

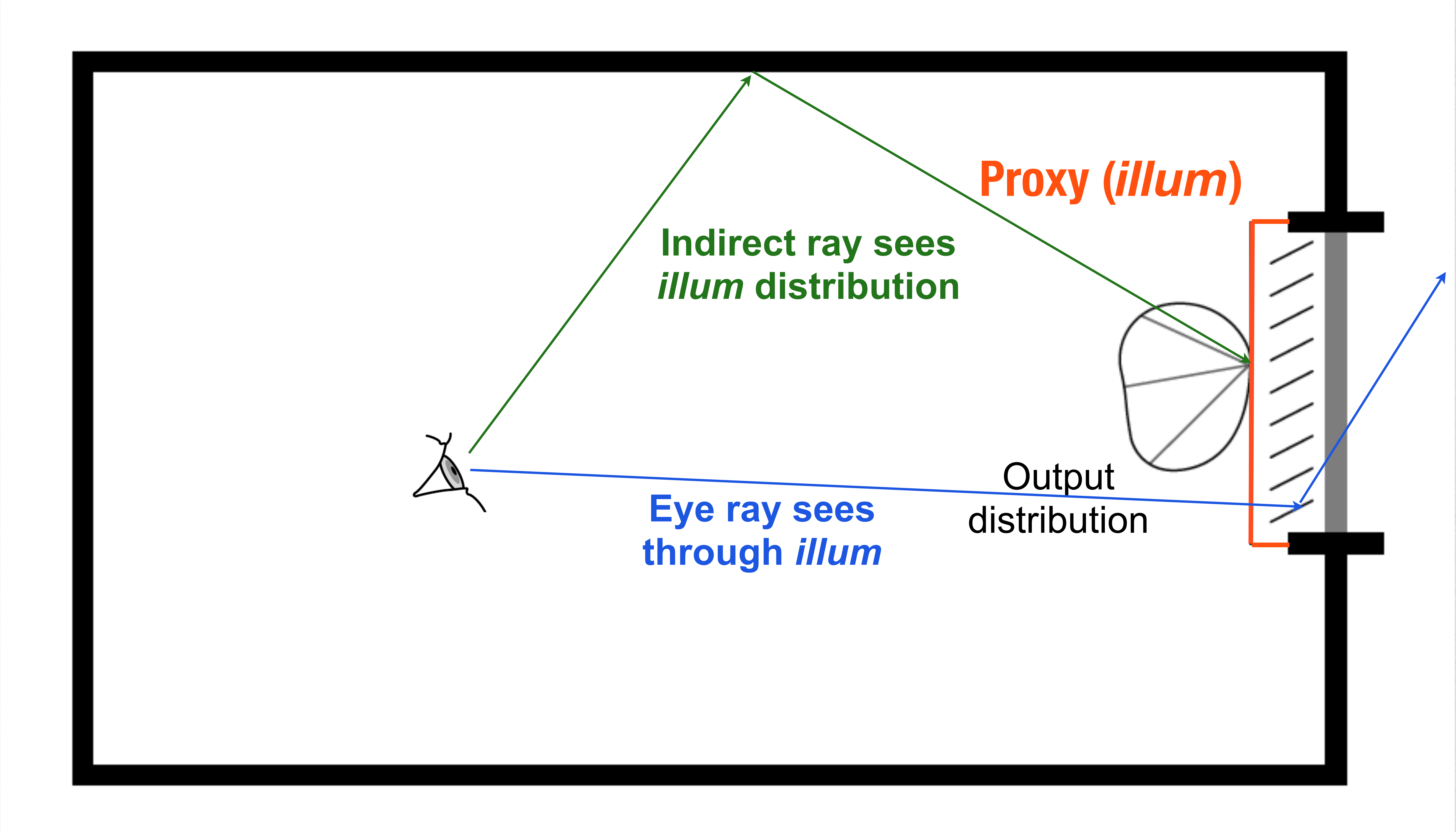
# Talk Overview

- History of complex fenestration in *Radiance*
- WINDOW 6 input to **mkillum**
- Using **genBSDF** to compute bidirectional scattering distribution function for new system
- Three-phase DC method for annual simulations
- New developments

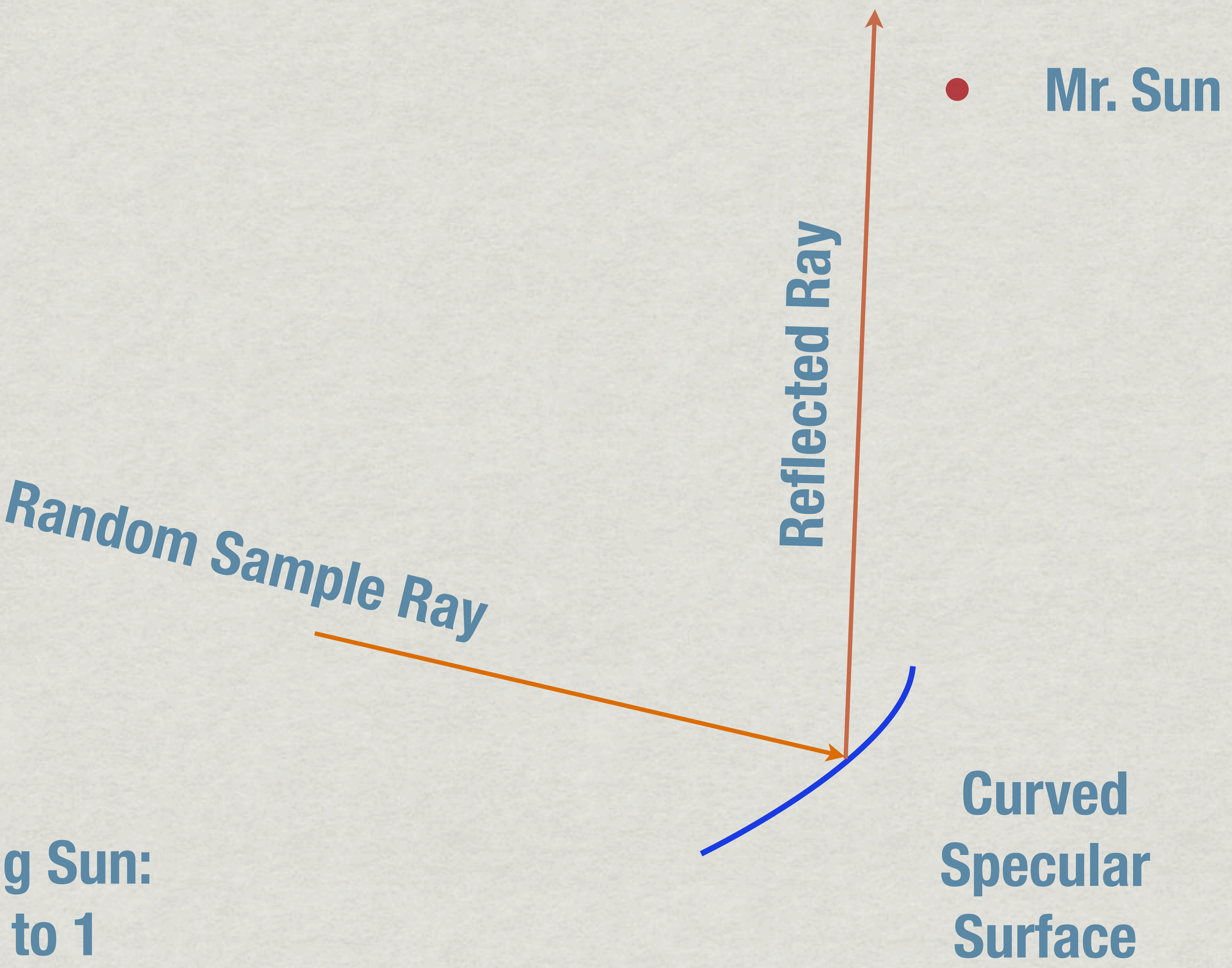
# The History of CFS in *Radiance*

- Use *illum* concept of proxied “secondary sources”
- The **mkillum** program has been around since 1991
  - Added during sabbatical at EPFL
  - Turns complex fenestration into proxy sources
  - Fails for sunlight on curved, specular systems

# Example Space



# Specular Sampling




**Chance of Hitting Sun:**
**100 thousand to 1**

# WINDOW 6 Input to **mkillum**

- WINDOW 6 supports 4-dimensional BSDF data
  - New XML format defined by LBNL
  - 145 input directions → 145 output directions
- **mkillum** samples exterior and uses BTDF to compute interior *illum* distribution
  - Overcomes limitations with specular systems

# WINDOW 6 XML File


```
<WavelengthData>
<Wavelength unit="Integral">NIR</Wavelength>
<SourceSpectrum>CIE Illuminant D65 1nm.ssp</SourceSpectrum>
<DetectorSpectrum>ASTM E308 1931 Y.dsp</DetectorSpectrum>
<WavelengthDataBlock>
    <WavelengthDataDirection>Transmission Front</WavelengthDataDirection>
    <ColumnAngleBasis>LBNL/Klems Full</ColumnAngleBasis>
    <RowAngleBasis>LBNL/Klems Full</RowAngleBasis>
    <ScatteringDataType>BTDF</ScatteringDataType>
    <ScatteringData>
 2.443881,   0.047337,   0.041435,   0.038990,   0.041435,
0.047337,   0.048413,   0.046964,   0.048413,   0.047337,
0.040883,   0.035154,   0.031478,   0.030108,   0.031363,
0.035154,   0.040605,   0.047337,   0.048086,   0.044691,
0.042586,   0.042007,   0.042537,   0.044691,   0.047921,
0.047337,   0.038892,   0.031273,   0.025227,   0.021345,
0.020007,   0.021345,   0.025227,   0.031273,   0.038892,
...
```

# Rendering Comparison 1

*Radiance* reference rendering

# Rendering Comparison 2

**mkillum** from geometry only

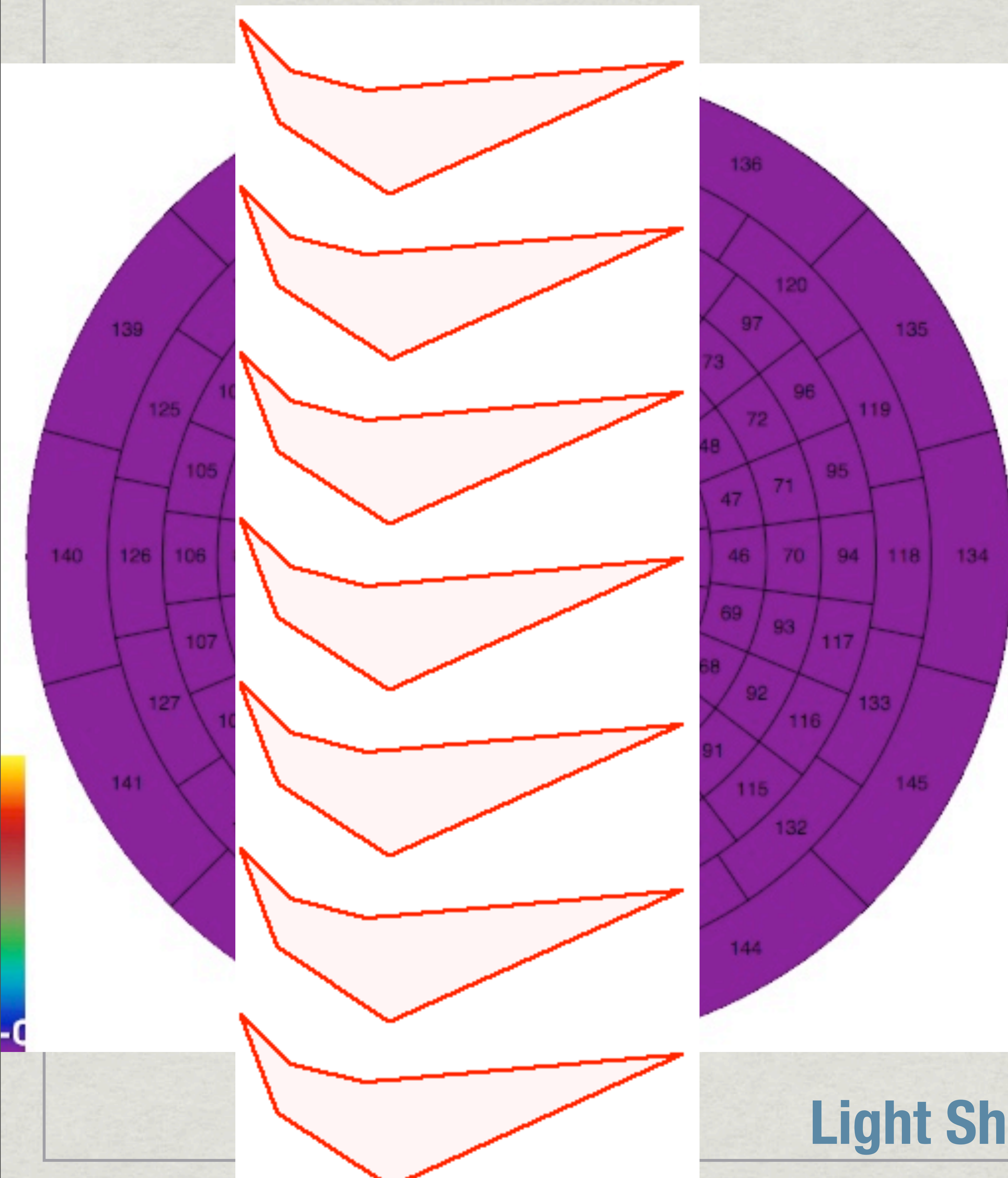
# Rendering Comparison 3

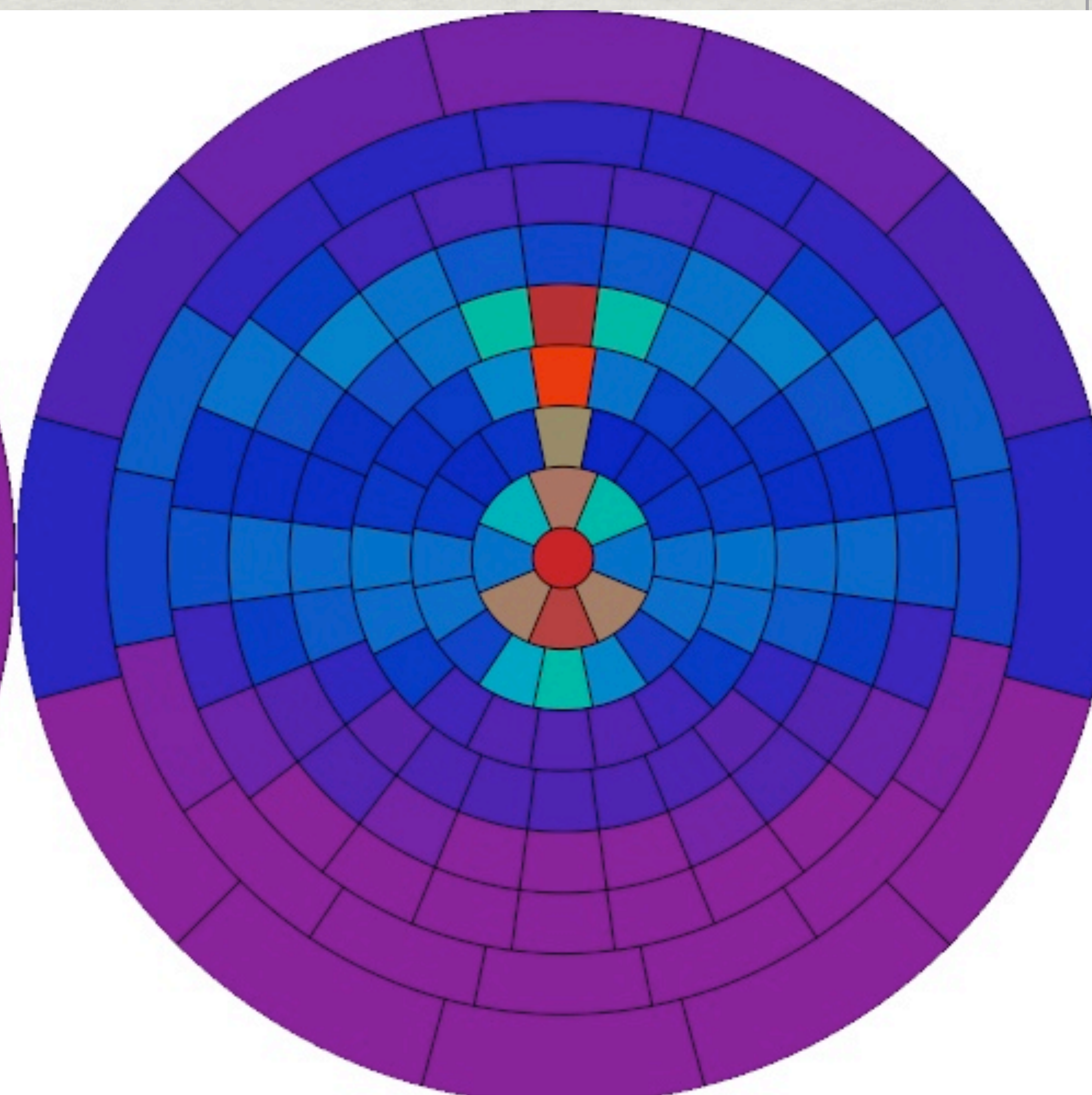
**mkillum** using BTDF data from WINDOW 6

# Computing BSDFs with **genBSDF**

- Uses **rtcontrib** to sample *Radiance* model of complex fenestration system
- Assembles results into WINDOW 6 format XML file
- Output usable in WINDOW 6 as well as *Radiance*
- Can include MGF description of CFS geometry

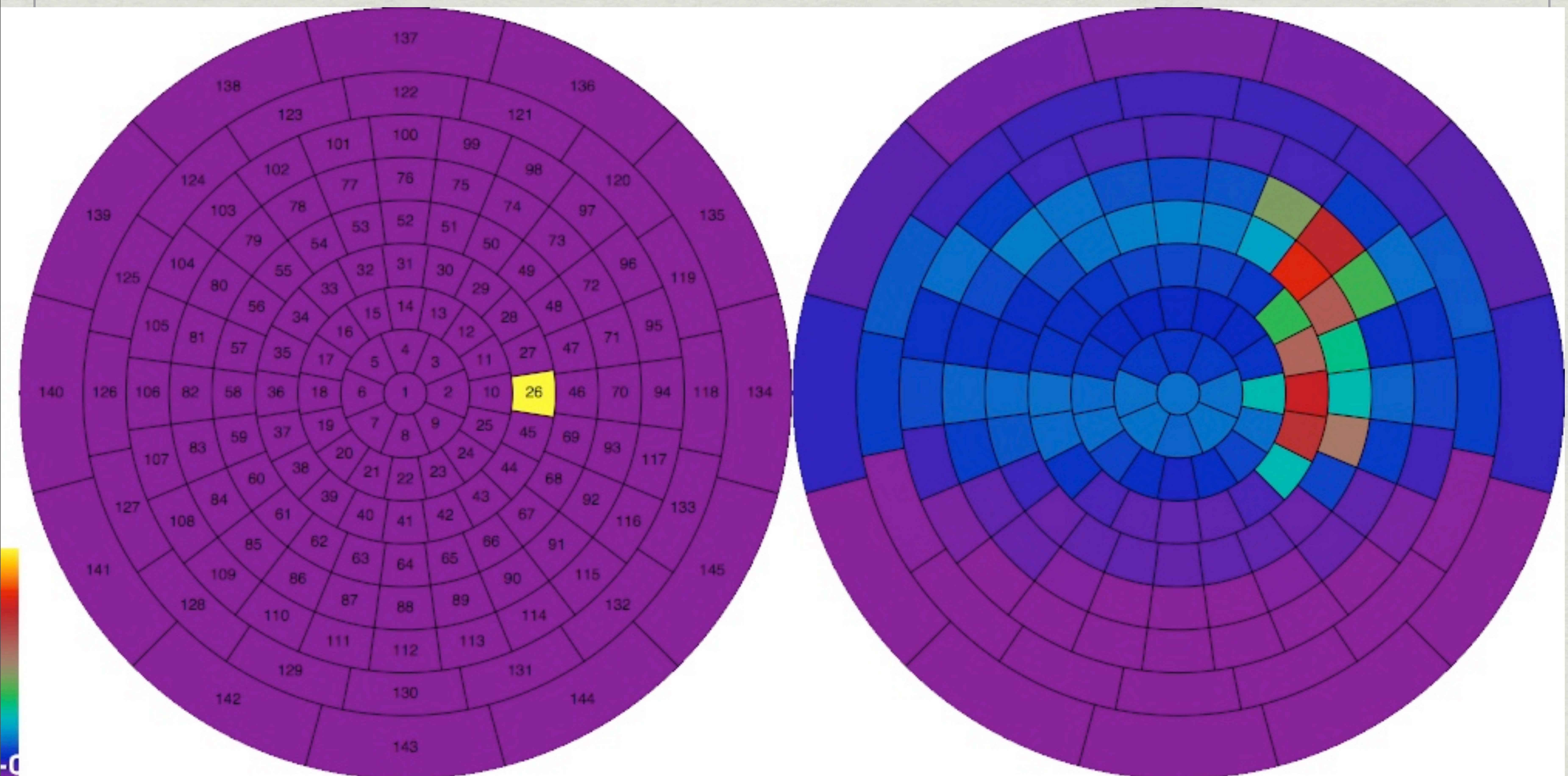
# Sample BTDF Data (1)



**Light Shelf**

# Sample BTDF Data (2)




**Light Shelf**

**Visualization by Andrew McNeil**

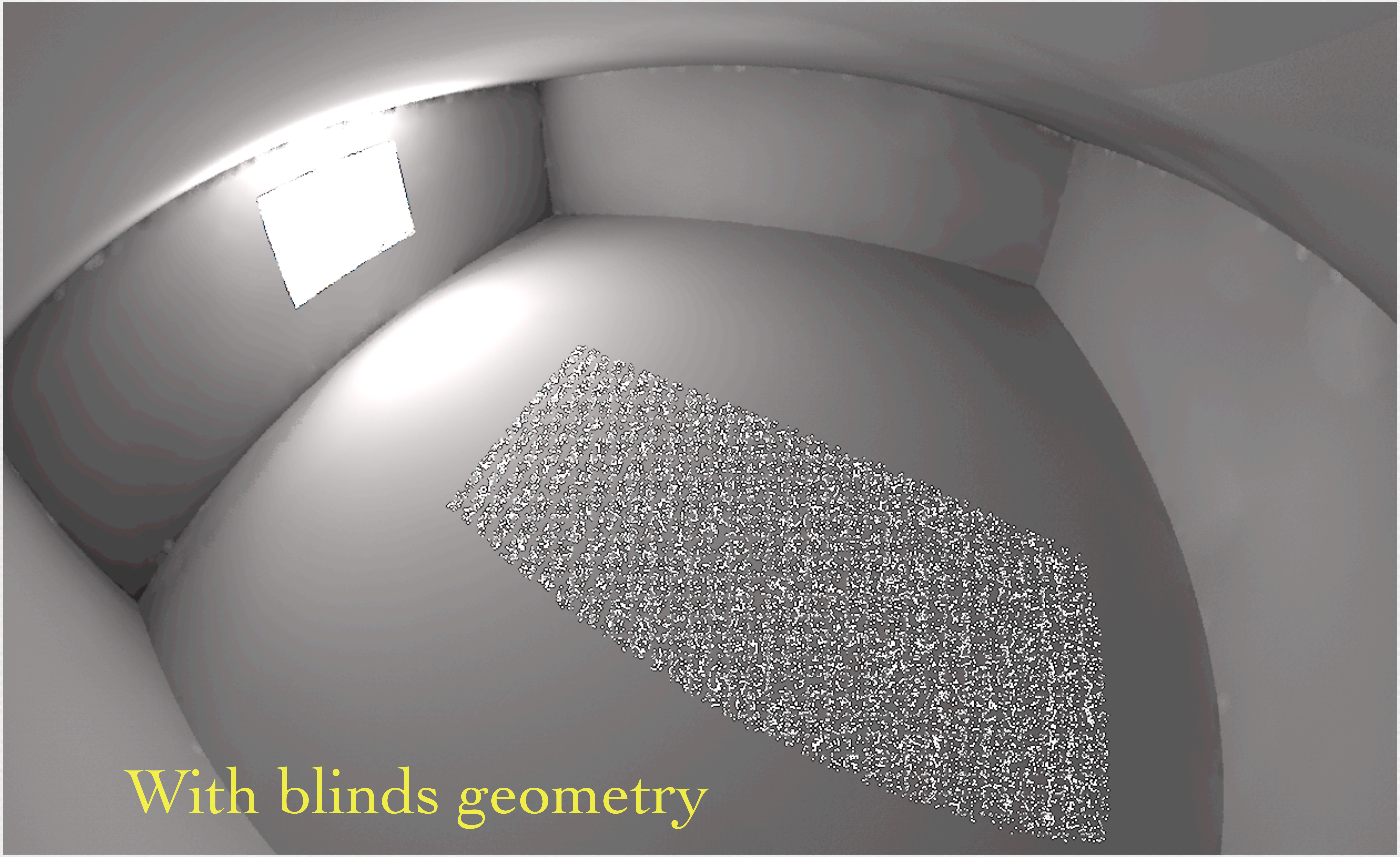
# Sample MGF

```
<Geometry format="MGF" unit="Meter">
# Y-axis points "up", Z-axis into room, right-handed coordinates
m WhitePlastic =
     rd .7
     rs .02 0
     sides 2
o VenetianBlinds
xf -rx -60 -a 67 -t 0 .03 0
    o Slat
     v v1 =
         p -2 0 0
     v v2 =
         p 2 0 0
     v v3 =
         p 2 0 .04
     v v4 =
         p -2 0 .04
     f v1 v2 v3 v4
    o
xf
o
</Geometry>
```

**XML embedding**

**Supports arrays**

# Example Results



# Annual Simulation

* Using mkillum with BTDFs is fairly quick, but...
* Re-rendering a scene 2000+ times for each hour?
* We need something faster...
* Can we use daylight coefficients with BTDF data?

# Three Phase Method

- **Phase I**:
  Light transport from sky patches to window exteriors

- **Phase II**:
  Light transport from window interiors to measurements (images, illuminance values, etc.)

- **Phase III** (time-step calculation):
  sky * exterior * BTDF * interior

# Our Matrix Equation

**i = VTDs**

where:

- **i** is the desired result vector (radiances, irradiances, etc.)
- **V** is the "View" matrix defining the lighting connection between results and exiting directions for a window group
- **T** is the "Transmission" matrix defining the BTDF of the window group
- **D** is the "Daylight" matrix defining the coefficients between incoming directions for the window group and sky patches
- **s** is a vector of sky patch luminances for a particular time and date

In a more explicit form, this would be:

$$\begin{bmatrix} sens1 \\ \dots \\ sensM \end{bmatrix} = \begin{bmatrix} sens1edir1 & \dots & sens1edirN \\ \dots & \dots & \\ sensMedir1 & & sensMedirN \end{bmatrix} \begin{bmatrix} edir1idir1 & \dots & edir1idirN \\ \dots & \dots & \\ edirNidir1 & & edirNidirN \end{bmatrix} \begin{bmatrix} idir1dc1 & \dots & idir1dcK \\ \dots & \dots & \\ idirNdc1 & & idirNdcK \end{bmatrix} \begin{bmatrix} sky1 \\ \dots \\ skyK \end{bmatrix}$$

# With Generality, There Come Challenges

- **rtcontrib** is used both in characterizing the exterior **D** in ***Phase I***, and in computing the interior **V** in ***Phase II***.

- It was not specifically designed to do either

- Rather, **rtcontrib** is a general tool for tracking light contributions

- Some scripts have been written to simplify the process, but much is still manual at this stage

# Phase I: Compute **D**

- Apply **rtcontrib** to relate sky patches to incident directions on window exterior
  - Need separate calculation for each orientation and major geometric feature
  - **genklemsamp** utility generates samples over a given window group

# Phase I Example

```
genklemsamp -vd -0.416041763 -0.909345507 0 -c 20000 \
  material_detailed.rad bg5wind.rad \
| rtcontrib -c 20000 -faf -f reinhart.cal -b rbin -bn Nrbins -m skyglow \
  @rtc_dmx.opt model_dumbsky.oct > SouthGroup.dmx
```

# Phase I Example

View defines window group orientation

```
genklemsamp -vd -0.416041763 -0.909345507 0 -c 20000 \
 material_detailed.rad bg5wind.rad \
| rtcontrib -c 20000 -faf -f reinhart.cal -b rbin -bn Nrbins -m skyglow \
 @rtc_dmx.opt model_dumbsky.oct > SouthGroup.dmx
```

Number of samples per direction must match

# Phase I Example

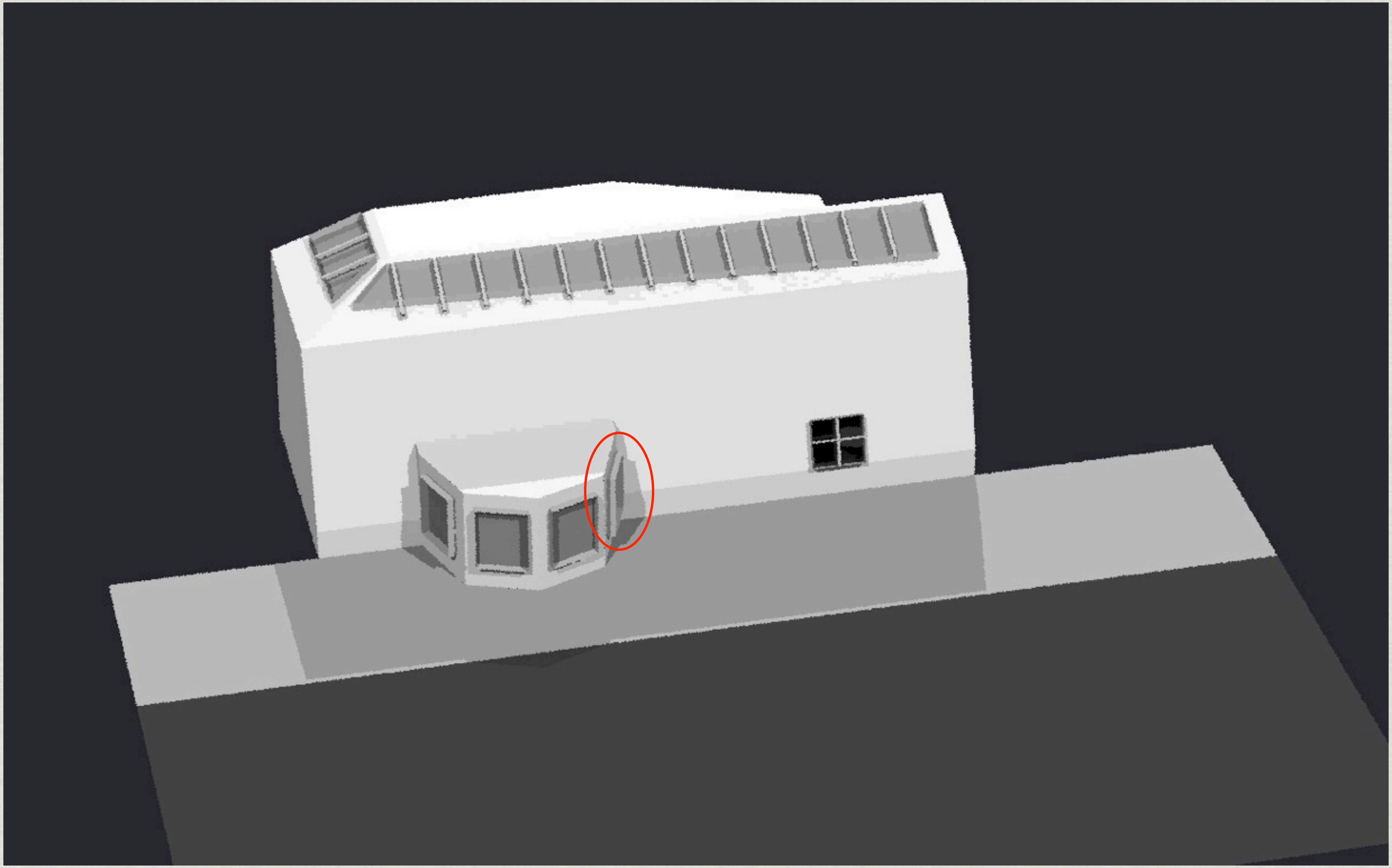
Window description may contain multiple surfaces, subset of octree

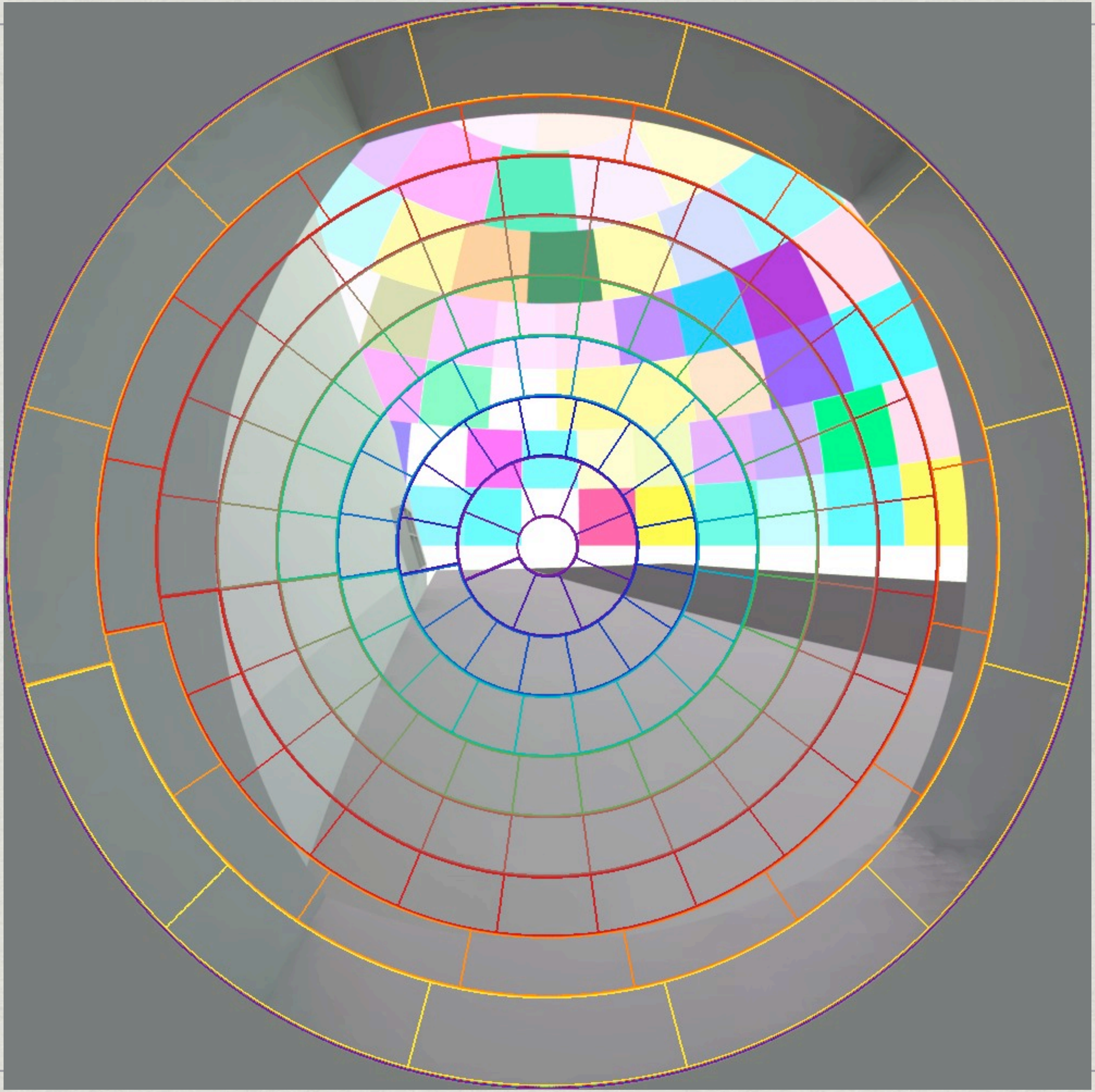
```
genklemsamp -vd -0.416041763 -0.909345507 0 -c 20000 \
 material_detailed.rad bg5wind.rad \
| rtcontrib -c 20000 -faf -f reinhart.cal -b rbin -bn Nrbins -m skyglow \
 @rtc_dmx.opt model_dumbsky.oct > SouthGroup.dmx
```

Sky uses Reinhart's subdivision of Tregenza sky patches for better accuracy

Different window orientations require separate **rtcontrib** runs

# Example Space





# Phase II: Compute **V**

- Use **rtcontrib** to relate sensor locations to exiting directions on window interiors
  - a single run can cover all window groups
  - **`klems_int.cal`** file maps to BTDF coord.

# Phase II Example

```
vwrays -ff -vf back.vf -x 1024 -y 1024 \
| rtcontrib `vwrays -vf back.vf -x 1024 -y 1024 -d` -ffc \
  -o comp/back_%s%03d.hdr -f klems_int.cal -bn Nkbins \
  -b kbinE -m EastGroup -b kbinS -m SouthGroup \
  -b kbinN -m NorthGroup -b kbinW -m WestGroup \
  @render.opt model.oct
```

# Phase II Example

Generating a set of image components

```
vwrays -ff -vf back.vf -x 1024 -y 1024 \
| rtcontrib `vwrays -vf back.vf -x 1024 -y 1024 -d` -ffc \
  -o comp/back_%s%03d.hdr -f klems_int.cal -bn Nkbins \
  -b kbinE -m EastGroup -b kbinS -m SouthGroup \
  -b kbinN -m NorthGroup -b kbinW -m WestGroup \
  @render.opt model.oct
```

The klems_int.cal file defines Klems patches over specific hemispheres

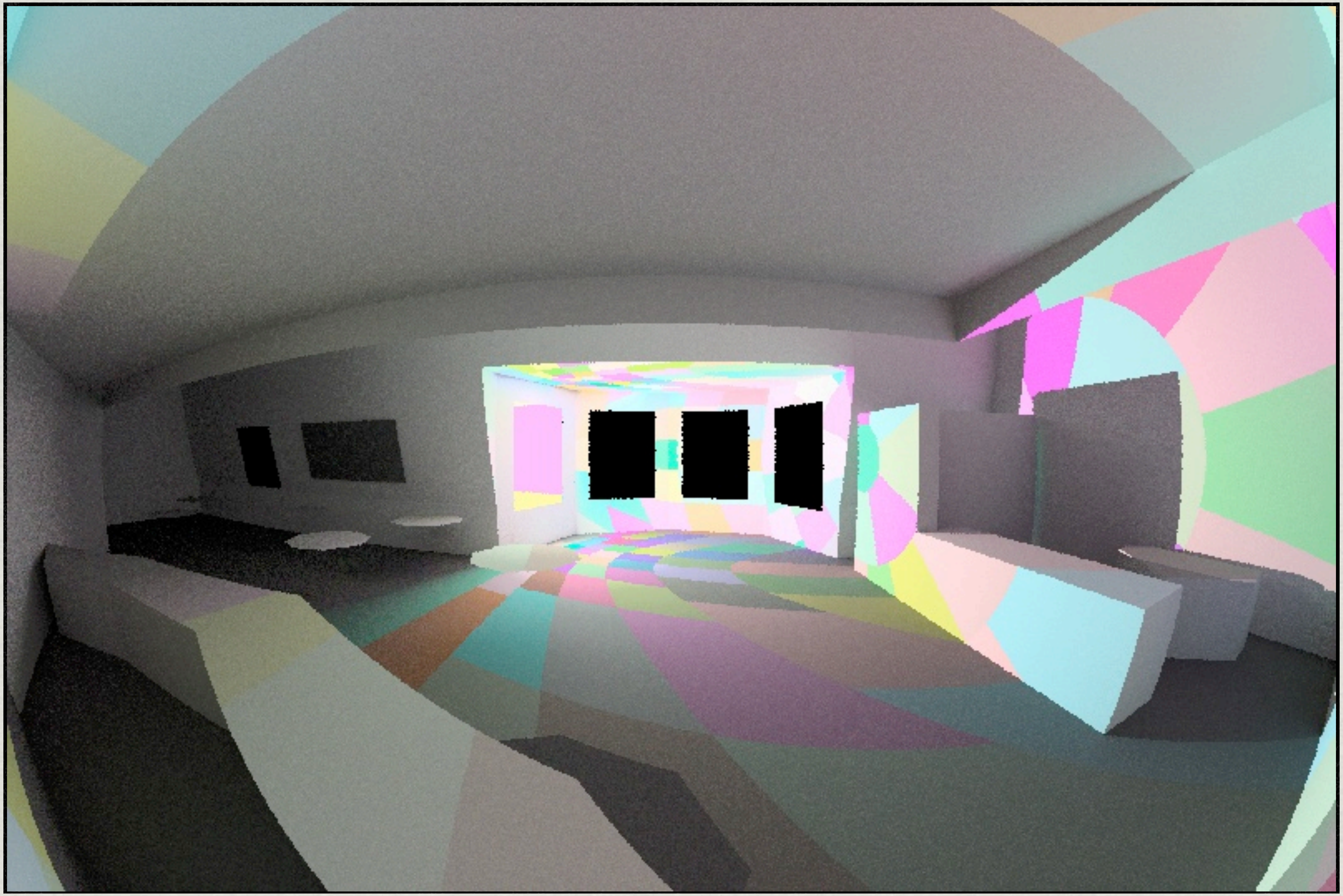
# Phase II Example

What is a reasonable set of rendering parameters?

```
vwrays -ff -vf back.vf -x 1024 -y 1024 \
| rtcontrib `vwrays -vf back.vf -x 1024 -y 1024 -d` -ffc \
  -o comp/back_%s_%03d.hdr -f klems_int.cal -bn Nkbins \
  -b kbinE -m EastGroup -b kbinS -m SouthGroup \
  -b kbinN -m NorthGroup -b kbinW -m WestGroup \
  @render.opt model.oct
```

```
-ab 4 -ds .05 -dj .7 -ad 2000 -lw 2e-4
```

- Windows *may be* sources
- No indirect caching
- One **rtcontrib** run captures everything



Outgoing Directions for One Window Group

# Phase III: Time Step

- Use **genskyvec** to create sky patch vector **s**
- Use **dctimestep** to multiply it all together

**BTDF**

**Result Vector** **Sky Patch Vector**

$$\mathbf{i} = \mathbf{VTDs}$$

**View Matrix** **Daylight Matrix**

# Phase III Example

```
gensky 9 21 12:00 -a 37.71 -o 122.21 -m 120 | genskyvec > eq.skv
pcomb '!dctimestep comp/back_SouthGroup%03d.hdr blinds1.xml SouthGroup.dmx eq.skv' \
   '!dctimestep comp/back_WestGroup%03d.hdr blinds2.xml WestGroup.dmx eq.skv' \
   '!dctimestep comp/back_NorthGroup%03d.hdr blinds2.xml NorthGroup.dmx eq.skv' \
   '!dctimestep comp/back_EastGroup_%03d.hdr blinds1.xml EastGroup.dmx eq.skv' \
  > back_9-21_1200.hdr
rm eq.skv
```

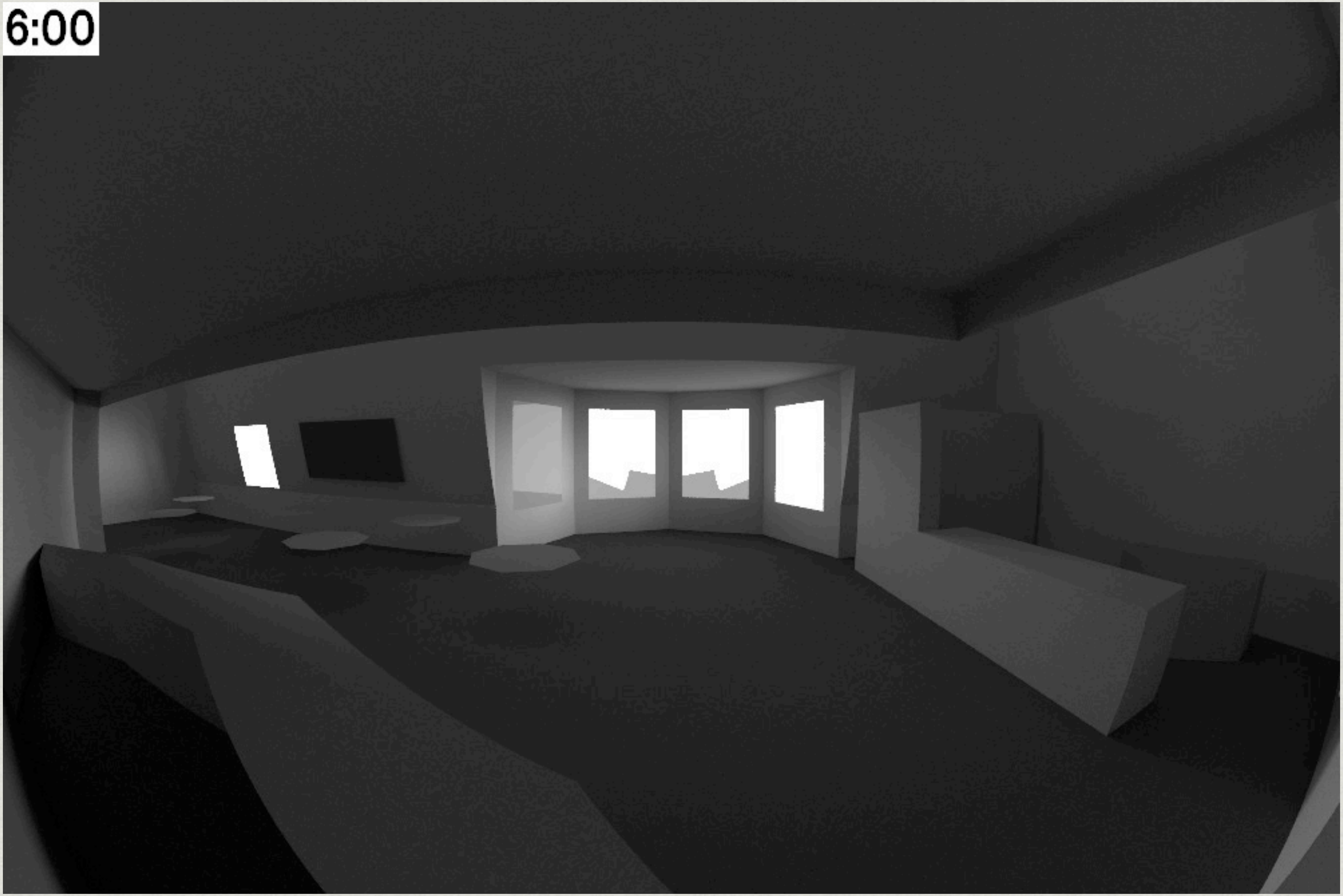
# Phase III Example

Generate sky vector for noon at the Autumn equinox

```
gensky 9 21 12:00 -a 37.71 -o 122.21 -m 120 | genskyvec > eq.skv
pcomb '!dctimestep comp/back_SouthGroup%03d.hdr blinds1.xml SouthGroup.dmx eq.skv' \
   '!dctimestep comp/back_WestGroup%03d.hdr blinds2.xml WestGroup.dmx eq.skv' \
   '!dctimestep comp/back_NorthGroup%03d.hdr blinds2.xml NorthGroup.dmx eq.skv' \
   '!dctimestep comp/back_EastGroup_%03d.hdr blinds1.xml EastGroup.dmx eq.skv' \
  > back_9-21_1200.hdr
rm eq.skv
```

Each call to **dctimestep** computes contributions of one window group

**Time to run the above is less than 4 seconds on my laptop**




Equinox Simulation

# Caveats

- Nearby interior and exterior geometry may require additional window subdivisions & ***Phase I*** runs
- Sky resolution trades off runtime with solar shading accuracy
- Interior and exterior reflections from windows will not be associated with different BSDFs

# New Developments

- Until now, BTDF data (but not BRDF) could be used in specific *Radiance* settings:
  - **mkillum** (neglecting interior window reflections)
  - Annual simulations using 3-phase DC method
- *Radiance* 4.1 supports BSDF data directly
  - Including new variable-resolution specification

# Example Rendering

## Measured and Modelled Materials