genBSDF Tutorial

Andy McNeil, LBNL

Revision History

29 September 2015	Version 1.0
23 October 2015	Version 1.0.1 fixed material name for example 1.

Abstract

This tutorial demonstrates how to use genBSDF to create a BSDF for a fenestration system component from a 3D model of the system. Three examples are provided, a venetian blind, an arbitrary shading system, and a prismatic daylight-redirecting device.

Contents:

1 - Introduction
2 – BSDF file format
3 – Using genBSDF (overview)
4 – Example 1 (venetian blind)
5 – Example 2 (arbitrary shading layer)
6 – Example 3 (prismatic daylight redirecting device)

1 Introduction

A Bidriectional Scattering Distribution Function (BSDF) describes the way light interacts with a material including the amount and direction of reflected and transmitted light. You may have heard of BTDF (transmission) and/or BRDF (reflection), well a BSDF is essentially the combination of a BRDF and BTDF.

The LBNL format BSDF file contains angularly resolved transmission and reflection data that simulation software can use to determine how to distribute light incident on a surface. It is particularly useful for simulating daylight transmitted by a complex fenestration system (CFS). A CFS, which transmits light in non-specular directions, is differentiated from a simple glazed façade which only transmits light in the specular direction. A BSDF file can describe an entire fenestration system, including several layers of specular and scattering components or a BSDF file can describe a single component of a fenestration system. A BSDF for a scattering fenestration component can be combined with specular glazing layers and other scattering layers to create a system BSDF using LBNL WINDOW.

An analogy that architectural lighting practitioners might find helpful: A BSDF file for a fenestration product is like an IES file for a luminaire (except it contains an outgoing light distribution for each discrete incident direction). Manufacturers can provide BSDF files for their daylighting & façade products that engineers and architects can use in simulation programs to accurately reproduce the amount and distribution of energy entering the building for various sun positions and sky conditions.

Radiance's genBSDF program can create a BSDF file from a model of the device via raytracing. The direction of tracing is unimportant for this application, though genBSDF acts like a forward ray tracer by switching the Radiance convention for sources and receivers.

There are two papers that offer a validation of genBSDF (McNeil et al. 2013 and Molina et al. 2015).

Tutorial Prerequisites

This tutorial assumes minimal knowledge of RADIANCE. However the tutorial assumes that a user knows how to use a shell prompt, basic bash scripting, and is able to use a text editor.

2 BSDF file format

The LBNL BSDF file uses XML format to encode information. One of the benefits of XML is that it is self-documenting. With little knowledge of the file (but with some knowledge of the subject matter) a user can open a BSDF file in a text editor and understand the data contained in the file.

The heart of the BSDF file is the data that describes intensity and direction of light leaving the system for each incident angle. In the LBNL format, this data is called "scattering data." Scattering data is contained in a "wavelength data" which also contains xml tags that describe the attributes or the scattering data block including wavelength (integral: visible or solar, discrete) direction (transmission front, reflection front, transmission back, reflection back) and the angle basis (Klems full, Klems half, Klems Quarter, or tensor tree).

The Klems full angle basis divides the incident and outgoing hemispheres into 145 discretized patches. For a Full Klems BSDF file, the scattering data blocks contain 145x145 values. For each of 145 incident angles there is an BSDF value for all 145 outgoing angles. Values in a column of the scattering data block are for a single incident patch and values in a row are for a single outgoing patch.

The three Klems angle bases are all fixed resolution resulting in square matrices (there are the same number of incident patches as outgoing patches). The resolution or the Klems angle basis may be too low for some uses. Increasing the number of patches in the angle basis is possible, but with a fixed resolution representation file size grows quickly, and much of the added data adds resolution in parts of the BSDF that aren't needed. The variable-resolution tensor tree BSDF format was created to allow increased resolution only where needed in the BSDF. This tutorial won't cover tensor tree BSDFs. For more information see Greg Ward's 2011 Radiance Workshop Presentation: http://radiance-online.org/community/workshops/2011-berkeley-ca/presentations/day1/GW3_Multires_BSDF.pdf

The values contained in the BSDF are the percent of incident flux arriving from within the discretized incident patch that leaves through the discretized outgoing patch divided by the solid angle of the patch and the integral of cosine theta over the patch area. This means that the range of physically possible BSDF values is from zero to 1/omega, where omega is the solid angle of the patch. For small patches, the BSDF value can be large, so don't be concerned if you see values much greater than one in a BSDF you generate.

3 Using genBSDF (overview)

There are three steps for creating a BSDF file using genBSDF:

1) Create model of system in RADIANCE or MGF format using the WINDOW coordinate system.
2) Run genBSDF with simulation parameters appropriately based on the properties of the system
3) Verify results

The importance of that third step can't be emphasized enough!

3.1 Step 1 - Creating a model for genBSDF

To create a BSDF file, genBSDF runs a raytracing simulation on a geometric model of the CFS. The model can be provided in RADIANCE or MGF format. Accurate material optical properties (reflectance/transmission) should be applied to system components in the model. We recommend measuring the material properties where possible. Reflectance and transmittance measurements are best made with flat samples in a spectrophotometer or similar.

genBSDF assumes the LBNL WINDOW coordinate system for orienting the system in the model. This coordinate system is different from the RADIANCE convention. In the WINDOW coordinate system +Y is up, and +Z is inside. The model provided to genBSDF should reside in the –Z half space, otherwise the user will receive a warning and in some cases geometry in the beyond Z=0 will not be raytraced by genBSDF.

3.2 Step 2 - Run genBSDF

To create a BSDF file, genBSDF runs a raytracing simulation on a geometric model of the CFS. The model can be provided in RADIANCE or MGF format. The BSDF usage template looks like this:

```
genBSDF  [  -c  Nsamp ][ -n Nproc ][ -r 'rcontrib opts...'  ][ -W ][ -s 'x=string;y=string' ]
        [ -t{3|4}  Nlog2  ][  {+|-}C  ][{+|-}forward  ][ {+|-}backward ][ {+|-}mgf ][ {+|-}geom unit ]
        [ -dim Xmin Xmax Ymin Ymax Zmin Zmax ] [ geom ..  ]
```

A genBSDF command might look something like this:

```
genBSDF -n 4 -c 2000 –t4 5 +backward +forward -geom millimeter -dim 45 55  40 60  -30 0 \
        –r '-ab 2' system.rad > system.xml
```

The following tables describe the command line options for genBSDF.

genBSDF Options	Description
-n *Nproc*	Sets the number of processes to run (for parallel processing). This option is not available on Windows.
-c *Nsamp*	Sets the number of samples per incident direction. The default is 2000 samples.

genBSDF Options	Description
-t{3|4} *Nlog2*	Tells genBSDF to create a tensor tree BSDF (rather than a Klems Basis BSDF). The 3 or 4 immediately follows the 't' (no space) and tells genBSDF what rank of tensor tree to create. A rank three tensor tree can be used for an isotropic system (has radial symmetry). For non-isotropic systems rank four tensor tree is required. The Nlog2 specifies the finest resolution of patch considered. The
+backward -backward +forward -forward [+-]b [+-]f	Tells genBSDF which directions to trace. Tracing the forward direction produces BSDF data for front transmittance and front reflectance (eg from the outside). Tracing the backward direction produces BSDF data for back transmittance and back transmittance (from the inside). Transmittance is only needed in one direction when the BSDF will be used in RADIANCE because RADIANCE will derive transmission from the opposite direction using reciprocity. The default is to trace backward direction only (+backward –forward). These options can be shortened as –b, +b –f, +f
+geom *unit* -geom *unit*	Includes or excludes model geometry in the BSDF file. Geometry is included in MGF format. If a RADIANCE model is provided, the geometry will be converted to MGF (using the rad2mgf converter). 'Unit' can be: meter, foot, inch, centimeter, or millimeter. The default is +geom meter
-dim *Xmin Xmax Ymin Ymax Zmin Zmax*	Sets the bounding dimensions for sampling. By default genBSDF uses the bounding box of the model provided. However there are cases where the user may wish to sample a small area of a larger model, particularly in situations where the CFS is comprised of transmitting materials.
-r *'rcontrib options'*	This option allows users to pass simulation parameters to rfluxmtx (the RADIANCE program that does the actual raytracing). These are options like -ab, -ad, -lw, -st, etc., which are familiar to Radiance users because they are used by RADIANCE raytracing programs (rtrace, rpict, rcontrib, rfluxmtx). For unix based shells the options are enclosed using single quotes. For a Windows dos shell, the options should be enclosed with double quotes.

Setting the rcontrib options appropriately can be critical to the accuracy of your BSDF. The following table contains the most useful parameters and their default settings. Since genBSDF uses rfluxmtx/rcontrib, there is no ambient caching, so parameters such as -aa, -ar, -as, and -aw have no effect. Additionally, genBSDF relies on the stochastic ambient calculation, there are no direct sources used in the simulation. Accordingly, direct simulation parameters such as -dt, -dc -dj, -ds, -dr, and -dp also have no effect on the output.

rfluxmtx parameters	Default setting	Description
-ab	5	The number of bounces traced. Once a ray tree gets to the specified number of bounces, no further rays are spawned.
-ad	700	The number of rays spawned at each surface intersection (contingent on the ray weight being above the lw setting). This is similar to 'ray splitting' parameters used in other raytracing software.
-lw	3e-6	The limit weight sets the minimum weight of a ray that is traced. Each traced ray is weighted according to its contribution to the result. At a ray intersection if 700 ambient division rays are spawned, the weight of each of those rays is 1/700th * the diffuse reflectance of the intersected material. If the weight of the spawned rays would be less than the limit weight, fewer than 700 are traced.
-lr	-10	The reflection limit determines how many specular reflections are traced. A negative value indicates that Russian roulette termination is used after the limit is reached.
-ss	1.0	The number of samples sent to sample highlights of rough specular materials.

-st	0.15	The threshold for specular sampling. If the specular reflection or transmission of a material is below this value, specular sampling will not occur.

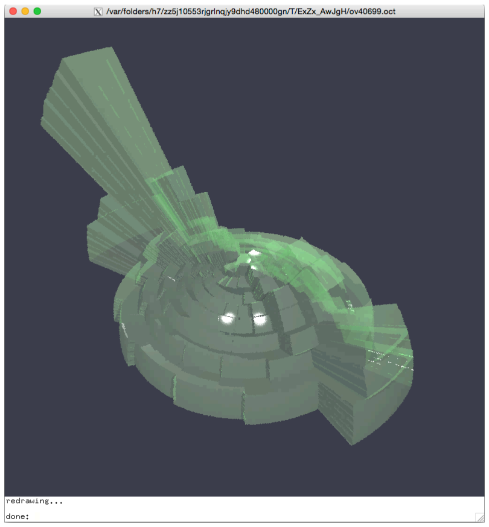
3.3 Step 3 - Verify Results

The final step is to verify the BSDF file is what you expect. We can't overstate the importance of this step. BSDF files contain a hodgepodge of numbers that makes it difficult for the user to recognize a minor or major flaw. Just like RADIANCE users are encouraged to render their space models so they can visually see that everything is as expected, we encourage users to "look" at the BSDF files closely to identify any irregularities or flaws.

There are two tools that are helpful for this verification step, bsdf2rad and BSDFviewer.

3.3.1 bsdf2rad

bsdf2rad is a RADIANCE program that will create a representational surface of the distribution for a single incidence angle. This is a debugging tool that is not compiled with the Radiance HEAD. To compile it go to the src/cv directory and run 'rmake bsdf2rad bsdfquery'. Then copy the binary files to the RADIANCE bin directory. **Error! Reference source not found.** shows an rendering of bsdf surface.



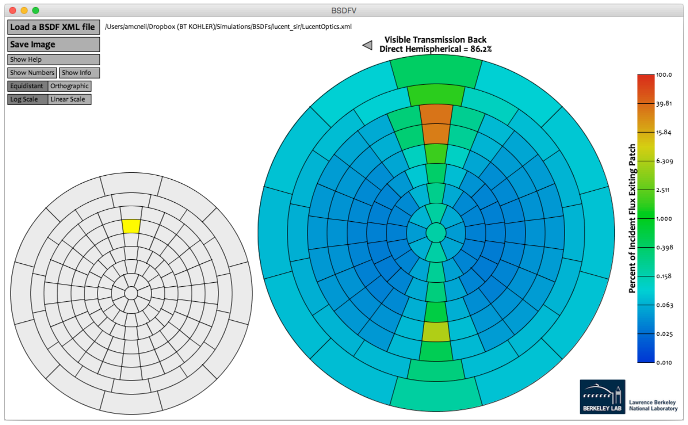
```
/var/folders/h7/zz5j10553rjgrlnqjy9dhd480000gn/T/ExZx_AwJgH/ov40699.oct
```

```
redrawing...
done:
```

Figure 1 - objview rendering of BSDF surface created by bsdf2rad.

3.3.2 BSDFviewer

BSDFviewer is a java applet that will load and display a falsecolor representation of a BSDF file. The color of each patch corresponds to the amount of flux leaving the patch. Users can click on the incident hemisphere to change the incident direction, or click on the arrows above the outgoing hemisphere to change between scattering data blocks in the BSDF file (visibile/solar, transmittance/reflectance, front/back). BSDFviewer also shows direct-hemispherical and hemispherical-hemispherical transmittance and reflectance data, which is useful for verifying against measured data if available. BSDFviewer supports tensor tree BSDFs on Mac OSX and Linux, (if it manages to compile the BSDF handling library correctly). BSDFviewer can be downloaded here: http://www.radiance-online.org/download-install/bsdf-viewer



/Users/amcneil/Dropbox (BT KOHLER)/Simulations/BSDFs/lucent_sir/LucentOptics.xml

Visible Transmission Back
Direct Hemispherical = 86.2%

Figure 2 - screenshot of BSDFviewer (this is the same BSDF file and incident angle as the bsdf2rad image above).

4 Example 1 – Venetian blind

We will create a BSDF file for a venetian blind as a quick first example. We'll use the RADIANCE program genblinds to create geometry for out blind system. We'll make a small model of blinds that is 1 inch in depth, 2 inches wide, and 4 inches tall. The slats will be spaced one inch apart and be tilted 20° downwards towards the outside. The following genblinds command creates just such a model (See the manual page for genblinds to understand syntax and all the options).

```
genblinds material_name blinds 1 2 4 4 20
```

This genblinds command creates a model where +Z is up and +Y is into the room, so we'll need to rotate the blinds using xform to orient the blinds the way genBSDF expects. We can pipe the output from genblinds directly to xform as shown in the following command:

```
genblinds material_name blinds 1 2 4 4 20 | xform -rz 90 -rx -90
```

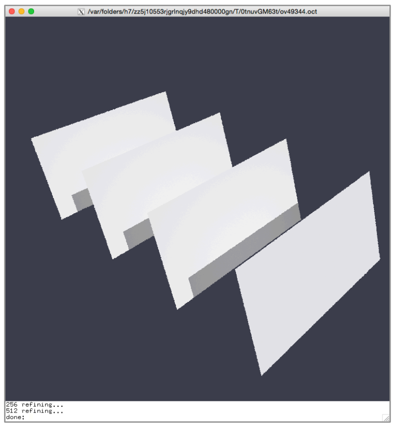
Next we can define a material, and create a text file to contain the material and geometry description. Instead of putting the output of the genblinds command into the file, we can put the command itself, preceded by an exclamation point, which tells RADIANCE to execute the command and include the output of the command in the place of the command. By including the command, we make a model file that is self-documenting and easy to modify if desired. We name the file blinds.rad.

```
### blinds.rad
void plastic white
0
0
5  0.7  0.7  0.7  0  0

!genblinds white blinds 1 2 4 4 20 | xform -rz 90 -rx -90
# this model contains a mistake, continue reading to find out more (so don't use this in your
simulation)
```

We created the plastic material 'white', which diffusely reflects 70% of incident light, and include it before the genblinds command in the text file. After saving the blind model file we can look at the geometry using the RADIANCE program objview. Objview creates a octree of the model with a few light sources, then it opens an rvu window with a view based on the size of the model. The following command will open an rvu window a render our model (Figure 3):

```
objview blinds.rad
```


/var/folders/h7/zz5j10553rjgrlnqjy9dhd480000gn/T/0tnuvGM63t/ov49344.oct

```
256 refining...
512 refining...
done:
```

Figure 3 - rvu interactive render window created by objview to display the blind model.

Next we'll want to verify that the model resides in the –Z half space. The RADIANCE program getbbox will tell you the bounding box or a radiance model.

```
getbbox blinds.rad
        xmin        xmax            ymin        ymax        zmin            zmax
        -2          5.75396e-17     0.5         3.84202     -0.939693       2.14313e-16
```

The output shows that the bottom side of the model is at Z= - 0.94 and the top of the model is just below Z=0 (by a very tiny amount). We are now satisfied that our model resides in the –Z halfspace.

Now that we have a satisfactory blind model, we can use genBSDF to create a BSDF file.

```
genBSDF +f +b -c 500 -geom inch  blinds.rad > blinds.xml
```

After genBSDF finishes we'll have an XML file that starts off like this:

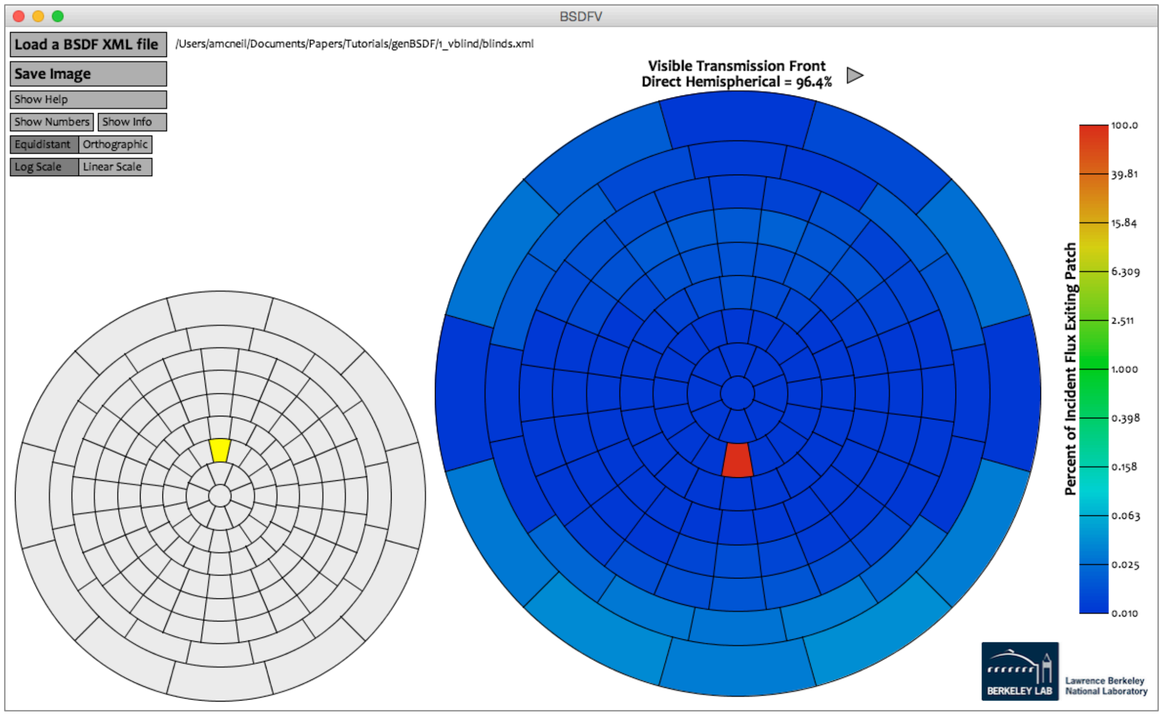
```
<!-- Created by: genBSDF +f +b -c 500 blinds.rad -->
<WindowElement xmlns="http://windows.lbl.gov" xmlns:xsi="http://www.w3.org/2001/XMLSch…
<WindowElementType>System</WindowElementType>
<FileType>BSDF</FileType>
<Optical>
<Layer>
        <Material>
                <Name>Name</Name>
                <Manufacturer>Manufacturer</Manufacturer>
                <DeviceType>Other</DeviceType>
                <Thickness unit="meter">0.939693</Thickness>
                <Width unit="meter">2</Width>
                <Height unit="meter">3.34202</Height>
        </Material>
        <DataDefinition>
                <IncidentDataStructure>Columns</IncidentDataStructure>
                <AngleBasis>
                        <AngleBasisName>LBNL/Klems Full</AngleBasisName>
                        <AngleBasisBlock>
                                <Theta>0</Theta>
                                <nPhis>1</nPhis>
```

And after scrolling we get to the angular data that interests us:

```
        <WavelengthData>
                <LayerNumber>System</LayerNumber>
                <Wavelength unit="Integral">Visible</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>
2.595e+01 4.777e-03 5.542e-03 6.963e-03 6.654e-03 5.512e-03 3.047e-03 2.465e-03 2.926e-03
5.310e-03 5.147e-03 6.617e-03 1.052e-02 9.937e-03 9.343e-03 8.523e-03 7.245e-03 5.247e-03
2.885e-03 1.704e-03 1.350e-03 2.166e-03 6.750e-04 1.246e-03 2.360e-03 4.258e-03 7.393e-03
8.945e-03 9.733e-03 1.101e-02 1.153e-02 1.074e-02 1.062e-02 6.995e-03 7.727e-03 4.744e-03
2.192e-03 1.316e-03 7.255e-03 1.217e-02 1.464e-02 1.058e-02 5.476e-03 2.131e-03 2.069e-03
4.014e-03 6.481e-03 9.313e-03 1.088e-02 1.219e-02 1.351e-02 1.438e-02 1.387e-02 1.159e-02
1.088e-02 8.987e-03 6.223e-03 4.077e-03 2.440e-03 2.999e-03 1.185e-02 1.806e-02 2.698e-02
2.744e-02 2.912e-02 2.151e-02 1.348e-02 4.780e-03 1.315e-03 3.626e-03 6.937e-03 …
```

There are four blocks (transmission front, reflection front, transmission back, reflection back) containing 21,025 values each. I don't know about you, but I find it difficult tell if this BSDF is any good. This is where a tool like BSDFviewer comes in handy. When we load the BSDF file into BSDF viewer we can select incident patches and look at the outgoing distribution. When we load the BSDF we just created we notice something unusual. For incident light coming from 20° below horizontal the total transmission is 95% and mostly contained in a single patch (**Error! Reference source not found.**). For incident light coming from 20° above horizontal the transmission is 45% and much less is contained in the specular patch


/Users/amcneil/Documents/Papers/Tutorials/genBSDF/1_vblind/blinds.xml

Visible Transmission Front
Direct Hemispherical = 96.4%

Figure 4 - BSDFviewer screenshot showing the front transmission distribution for incident light coming from 20° above horizontal.

(Figure 5).

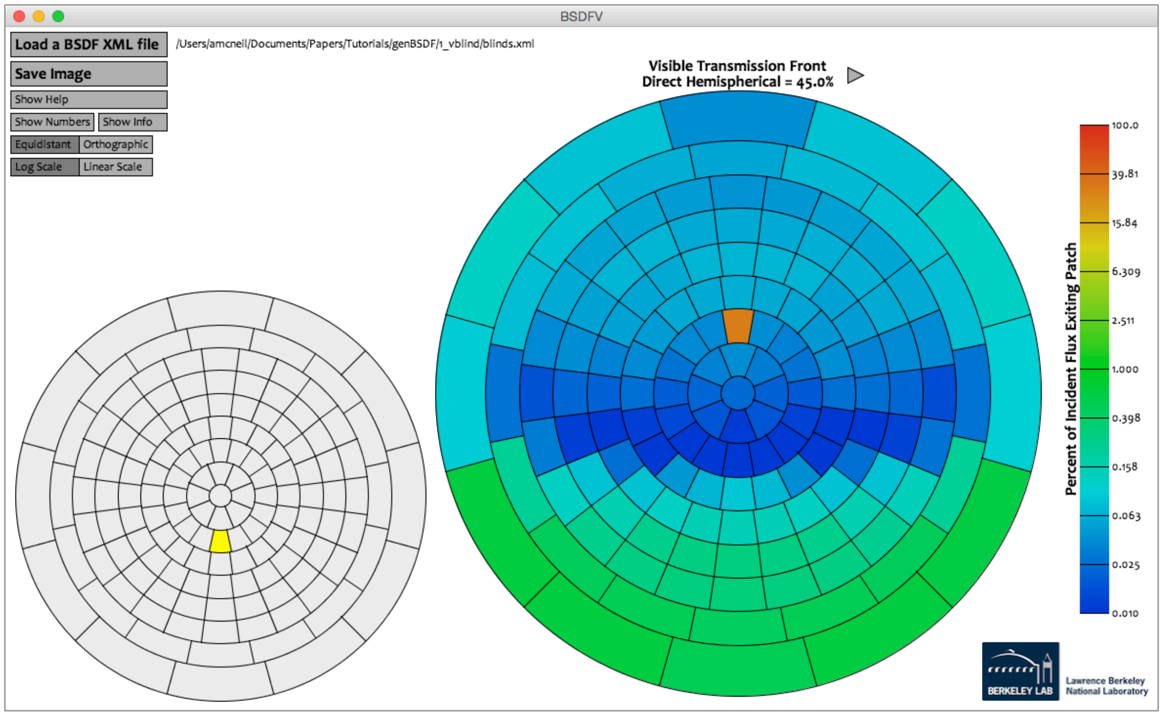

BSDFV
Load a BSDF XML file
/Users/amcneil/Documents/Papers/Tutorials/genBSDF/1_vblind/blinds.xml
Save Image
Show Help
Show Numbers
Show Info
Equidistant
Orthographic
Log Scale
Linear Scale
Visible Transmission Front
Direct Hemispherical = 45.0%
Percent of Incident Flux Exiting Patch
100.0
39.81
15.84
6.309
2.511
1.000
0.398
0.158
0.063
0.025
0.010
BERKELEY LAB
Lawrence Berkeley National Laboratory

Figure 5 - BSDFviewer screenshot showing the front transmission distribution for incident light coming from 20° below horizontal.

Our blinds were supposed to be tilted downwards to the outside – less light should be transmitted when incident from above horizontal compared to below horizontal. This indicates that either our tilt angle is wrong or our model is not oriented properly. Checking the genblinds man page we see that front or back isn't defined. I'm inclined to use a positive tilt angle to mean the outside facing edge is below the inside facing edge (a WINDOW convention). We'll change the xform command to fix the facing issue, which means changing the xform rotation about the x-axis from 90° to *negative* 90°.

```
### blinds.rad
void plastic white
0
0
5  0.7  0.7  0.7  0  0

!genblinds white blinds 1 2 4 4 20 | xform -rz -90 -rx -90
# this model contains mistakes so don't use this in your simulation
```

We run getbbox again:

```
getbbox blinds.rad
        xmin        xmax        ymin        ymax        zmin            zmax
        0           2           0.5         3.84202     -9.18485e-17    0.939693
```

And we discover that our blinds are in the +Z half space. We can add a translation in the xform command to move the whole model into the –Z half space.

```
### blinds.rad
void plastic white
0
0
5  0.7  0.7  0.7  0  0

!genblinds white_semispec blinds 1 3 4 4 20 | xform -rz -90 -rx -90 -t 0 0 -0.939693
# believe it or not, this model still has a mistake so don't use this in your simulation
```

We then use getbbox one more time to verify that we are in the negative Z halfspace:

```
getbbox blinds.rad
        xmin          xmax          ymin          ymax          zmin          zmax
        0             2             0.5           3.84202       -0.939693     -3.79214e-07
```

And we're good to run genBSDF again:

```
genBSDF +f +b -c 500 -geom inch  blinds.rad > blinds.xml
```

Now if we look at the outgoing distribution with BSDFviewer, we can see that the we have near 100% transmission with light incident from 20° below horizontal as expected (Figure 6).

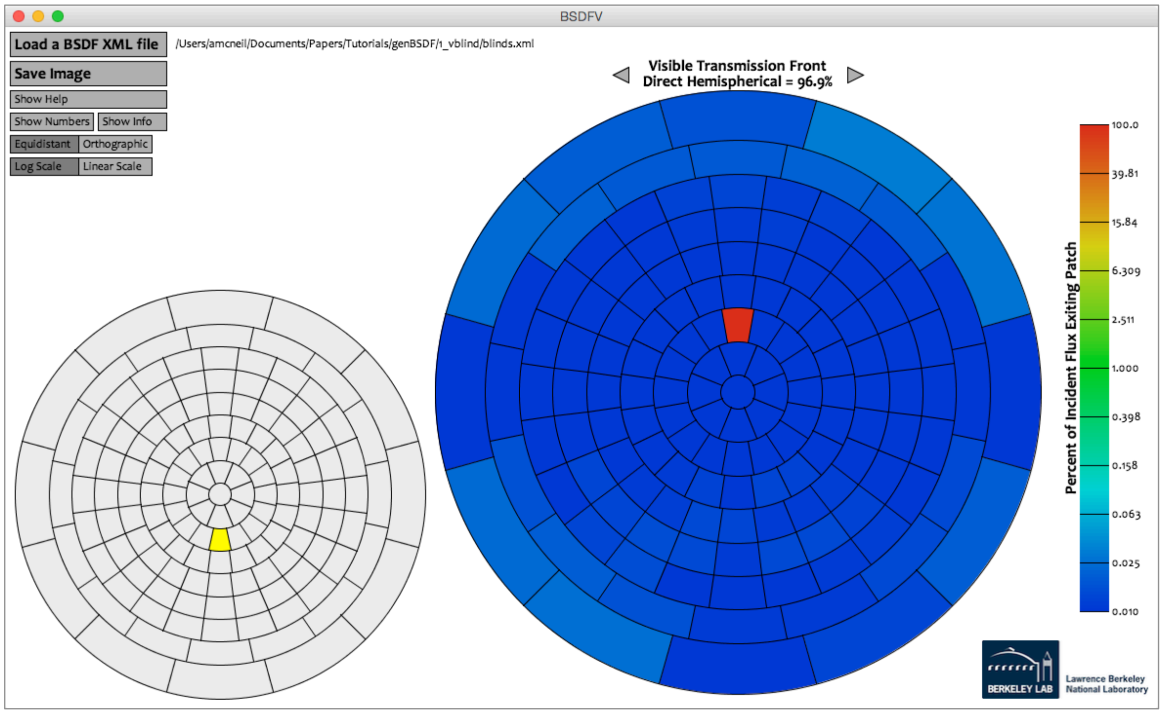

BSDFV
Load a BSDF XML file
/Users/amcneil/Documents/Papers/Tutorials/genBSDF/1_vblind/blinds.xml
Save Image
Show Help
Show Numbers
Show Info
Equidistant
Orthographic
Log Scale
Linear Scale
Visible Transmission Front
Direct Hemispherical = 96.9%
100.0
39.81
15.84
6.309
2.511
1.000
0.398
0.158
0.063
0.025
0.010
Percent of Incident Flux Exiting Patch
BERKELEY LAB
Lawrence Berkeley National Laboratory

Figure 6 - Front transmission of BSDF with light incident from 20° below horizontal.

However, if we look at light incident from an angle above the shade we can see an alarmingly high specular transmission (Figure 7). It turns out that this light is passing through the blind untouched at angles that they would normally intersect a blind slat. This occurs because genBSDF samples the system by randomly distributing ray origins over a polygonal face that bounds the geometry. Rays with origins near an edge of the system can possibly exit the system boundary without intersecting geometry that it otherwise would (Figure 8).

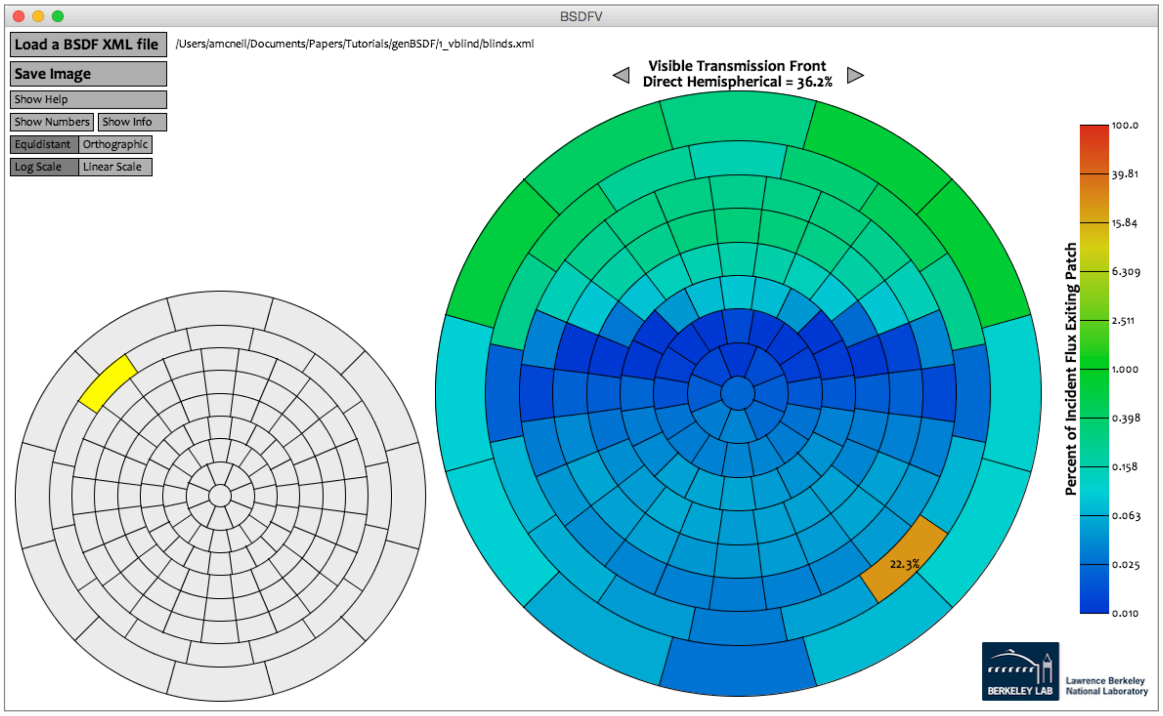
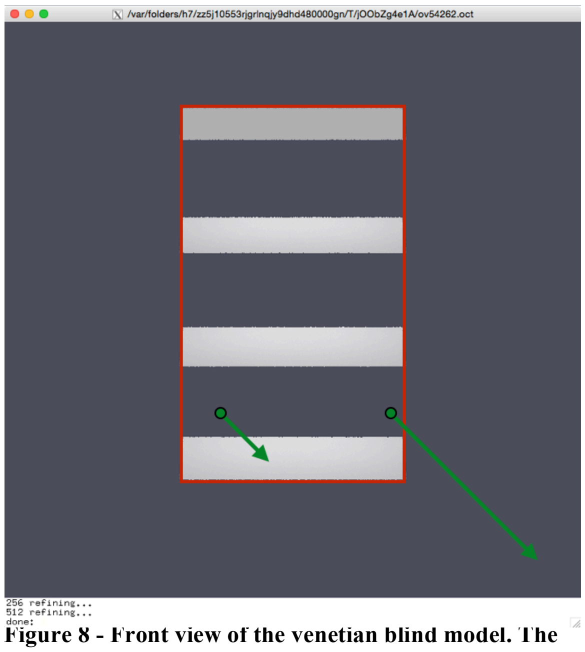

BSDFV
Load a BSDF XML file
/Users/amcneil/Documents/Papers/Tutorials/genBSDF/1_vblind/blinds.xml
Save Image
Show Help
Show Numbers
Show Info
Equidistant
Orthographic
Log Scale
Linear Scale
Visible Transmission Front
Direct Hemispherical = 36.2%
22.3%
Percent of Incident Flux Exiting Patch
100.0
39.81
15.84
6.309
2.511
1.000
0.398
0.158
0.063
0.025
0.010
BERKELEY LAB
Lawrence Berkeley National Laboratory

Figure 7 - Front transmission for lighting incident from above left of the blind.


/var/folders/h7/zz5j10553rjgrlnqjy9dhd480000gn/T/jOObZg4e1A/ov54262.oct

```
256 refining...
512 refining...
done:
```

Figure 8 - Front view of the venetian blind model. The front bounding face is show as a red rectangle. Of the two rays shown in green with the same angle but different origins, the one on the left intersects with the blind slat, while the one on the right leaves the bounding box before intersecting a blind slat.

There are two ways to solve the boundary issue, either including a boundary condition in the model or by making the geometry large enough that the probability of a ray leaving through the edge of the system becomes insignificant. Modeling the boundary condition makes sense when the boundary condition is known. For example if the blinds are for a window of known size and the framing geometry and material properties are known. Otherwise it's advisable to create an 'infinite layer' BSDF where there are no boundary conditions assumed. The later method is the approach used by LBNL WINDOW software.

We'll adjust our model to approximate an infinite layer. Since the slats are opaque we don't need to worry about increasing the height of the system and repeating more slats. We want to set the width of the blind to be very wide, we'll use 400 inches.

```
### blinds.rad
void plastic white
0
0
5  0.7  0.7  0.7  0  0

!genblinds white blinds 1 400 4 4 20 | xform -rz -90 -rx -90 -t 0 0 -0.939693
```

We don't want samples to originate near the edge of the model, because we're trying to omit the boundary condition. We can control how the system is sampled using genBSDF's -dim option. Since the geometry is consistent in the y-axis we can simply sample along a line in the center of the model. Additionally, we want the height of our sampled area to be an interger multiple of the spacing between slats (the boundary as set in Figure 8 introduces bias – see if you can figure out why). We'll set the dim command so that it samples along a line, roughly between the middle two slats. We'll center the sample area in the width of the model (at 200 inches). The Y boundaries will be 1 inch apart (the slat spacing) and will be coordinated with the edge of the slat. Since we have an extra slat in either direct in the Y-axis, we don't have to be too precise with the edges of the boundaries. The following command shows the dim option used.

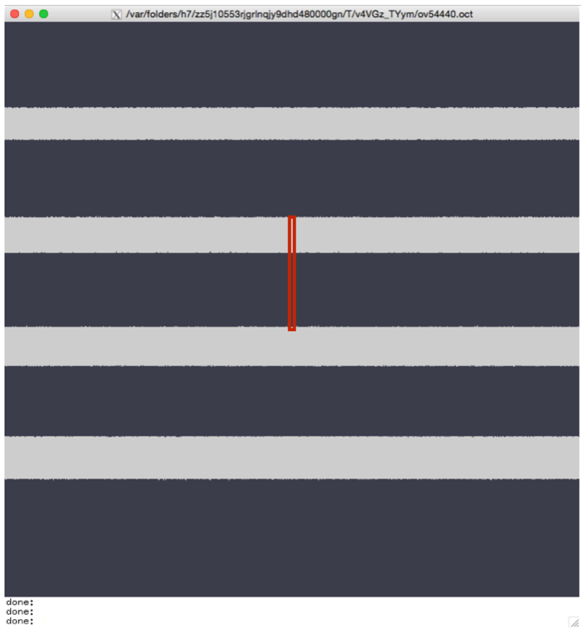

```
/var/folders/h7/zz5j10553rjgrinqjy9dhd480000gn/T/v4VGz_TYym/ov54440.oct
done:
done:
done:
```

Figure 9 - A view of the center of the 400 inch wide model showing the thin sample boundary.

```
genBSDF +f +b -c 500 -geom inch -dim 199 201 1.5 2.5 -0.939693 0 blinds.rad > blinds.xml
```

Opening this new BSDF in BSDFviewer we see that there is no longer a high value in the specular patch indicating that light is no longer transmitting directly by exiting the model at the edge (Figure 10).

[Author's Note: While I purposely made the model small to illustrate the edge effect, the error with orientation was a complete accident. This occurred despite my thinking through the xform rotations to achieve what I though was the desired orientation. I decided to include my error as an example and to include this note so that users know that orienting the model is tricky and to demonstrate that it is of vital importance to verify that your BSDF behaves as you would expect.]

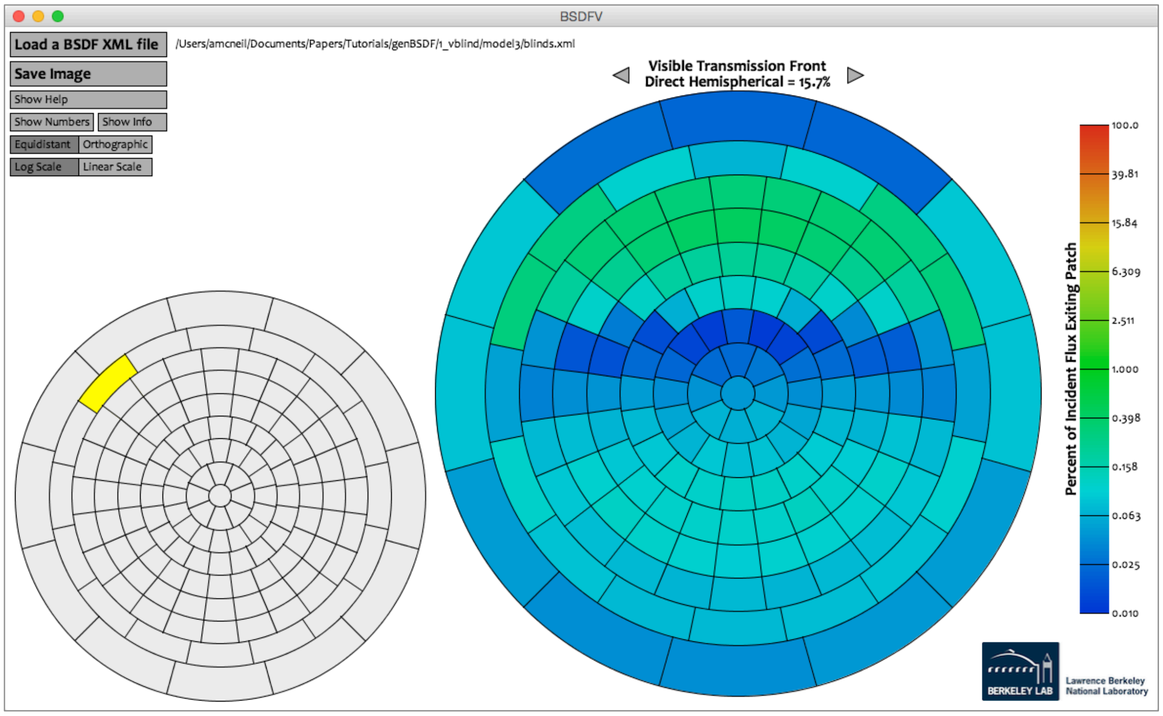

BSDFV
Load a BSDF XML file
/Users/amcneil/Documents/Papers/Tutorials/genBSDF/1_vblind/model3/blinds.xml
Save Image
Show Help
Show Numbers
Show Info
Equidistant
Orthographic
Log Scale
Linear Scale
Visible Transmission Front
Direct Hemispherical = 15.7%
Percent of Incident Flux Exiting Patch
100.0
39.81
15.84
6.309
2.511
1.000
0.398
0.158
0.063
0.025
0.010
BERKELEY LAB
Lawrence Berkeley National Laboratory

Figure 10 - Screen shot of BSDFviewer showing that there is no longer direct transmission for light incident from the left above horizontal.

5 Example 2 – Arbitrary Shading Device

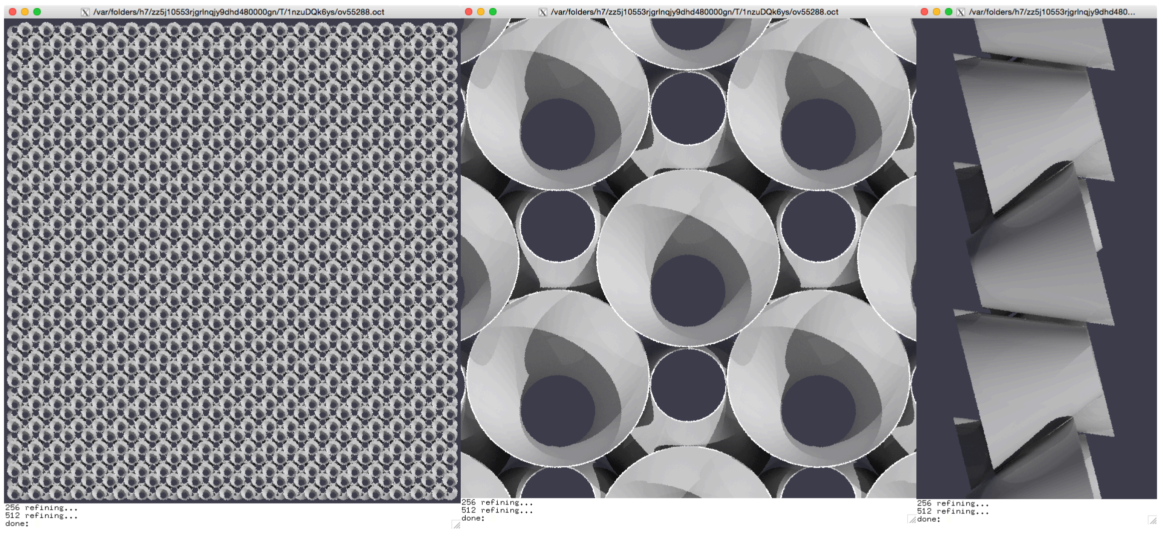
In this example we'll create our model using MGF. Repetitive models can be represented fairly compactly using MGF, and since the XML spec allows geometry to be included in MGF format, the model can be included in the BSDF file using the +geom option.

In the 'arbitraryshade.mgf' model file, we start by defining the material 'white83' using diffuse transmission (td), specular transmission (ts), diffuse reflection (rd), and specular reflection (rs). Then we define the eight vertices of our geometry, followed by four cones contained in a transformation context (xf). The transformation arrays the four cones several times to create a larger model. This website contains documentation of the MGF format: http://radsite.lbl.gov/mgf/

```
# arbitraryshade.mgf
m white83 =
        td 0.3
        ts 0 0
        rd 0.5
        rs 0.03 0

o arbitraryshade
        v v1 =
                    p 0 -0.5 -2
        v v2 =
                    p 0 0 0
        v v3 =
                    p 2 0 -2
        v v4 =
                    p 2 0.5 0
        v v5 =
                    p 0 1.8 -2
        v v6 =
                    p 0 2.3 0
        v v7 =
                    p 2 1.9 -2
        v v8 =
                    p 2 2.4 0
        m white83
        xf -t 0 0 -0.341975 -a 21 -t 0 4.25 0 -a 21 -t 4 0 0
                    cone v1 0.59 v2 1.41
                    cone v3 1.41 v4 0.59
                    cone v5 1.41 v6 0.59
                    cone v7 0.59 v8 1.41
        xf
o
```

Figure 11 shows rendered views of the model. These renderings are created by converting the MGF model to RADIANCE using mgf2rad, then by running objview with the converted model file.


/var/folders/h7/zz5j10553rjgrlnqjy9dhd480000gn/T/1nzuDQk6ys/ov55288.oct
/var/folders/h7/zz5j10553rjgrlnqjy9dhd480000gn/T/1nzuDQk6ys/ov55288.oct
/var/folders/h7/zz5j10553rjgrlnqjy9dhd480...

```
256 refining...
512 refining...
done:
```

```
256 refining...
512 refining...
done:
```

```
256 refining...
512 refining...
done:
```

Figure 11 - Objview rendered views of the arbitrary shading system including a view from inside the shading system (left), a zoomed in view from the inside (center), and a side view (right). In the side view the left is outside and right is inside.

We discussed boundary conditions in the previous example. For this system we are again going to create a BSDF that represents an infinite layer. Because the cone material transmits light, we need to replicate the model geometry several times to ensure that a ray doesn't transmit through just a few cones and then out of the system. In this model we've replicated the four original cones 21 times in each direction using the transform context. We'll then use the –dim option to sample one repetition of the geometry in the center of the model. We don't have to worry so much about getting the exact center of the model, or about aligning the sampling face with the geometry as long as we make sure that we sample the exact repetition distance, which we conveniently have in the transform context of the MGF file. The geometry repeats every 4 inches in the X direction and every 4.25 inches in the Y direction. We can again use getbbox to get the overall dimensions of the model.

```
mgf2rad cones.mgf | getbbox
   xmin     xmax     ymin      ymax      zmin      zmax
  -1.41     83.41   -1.3679   88.7679  -2.68395 2.31301e-07
```

Then we can set our sample area to be roughly in the middle of the model and 4 inches wide by 4.25 inches tall. For the Z dimensions we simply take the model bounds. Our genBSDF command looks as follows:

```
genbsdf +f +b +geom inch -c 4000 -r '-ab 20 -ad 100 -lw 1e-2 -st 9e-4' \
        -dim 39 43 41.5 45.75 -2.68395 0 +mgf cones.mgf > cones.xml
```

This is the first time that we've used the -r command to change the ray tracing simulation parameters. The best way to determine these parameters for your simulation is to perform a parametric convergence testing. Where you parametrically adjust parameters to find the lowest settings where the result has converged. However, for this example I used experience and intuition to set the parameters, which admittedly is dangerous but since this is just a tutorial, whatevs.

-ab 20 : Each time a ray hits the cone, 83% of energy is reflected and 17% absorbed. If we take the reflectance is r and the number of ambient bounces used in the simulation is n, then we know that the most possible energy left after n ambient bounces is r^n. So for a reflectance of 0.83, the default 5 ambient bounces means we could have at most 39% of energy left unaccounted for at the end of the simulation (since RADIANCE just stops tracing). That seems too high, no? So instead using 20 ambient bounces leaves us with at most 2.4% of energy left unaccounted for (but probably less since many rays will exit the system before reaching 20 ambient bounces).

-ad 100 : We want many rays to be spawned after the ray intersection, but do we need 700? In this case I think no. Were sending 4000 samples per Klems patch so 50*4000 gives us 200,000 ambient divisions after one intersection in each klems patch. Seems like enough to me.

-lw 1e-2 : I've set this parameter high because while we want a lot of rays to be spawned after the first ray intersection, we don't really need the number of rays to grow after subsequent ambient bounces. I've set this one to be 1/ad.

-st 9e-4 : I've decreased the specular threshold because we want to consider specular reflections, so it needs to be at least less then the specular reflectance of our cone material. I've set it to rs^2 so that we could include secondary reflections, though I'm not really sure secondary specular reflections are important.

-c 4000 : I've also increased the number of samples per Klems patch. In my first run, I found that there was some discrepancy between the specular transmission for mirrored patches, so I increased the sampling density. Also, since this model varies in two dimensions, higher sampling density makes sense.

With these settings, genBSDF took 7 minutes, 14 seconds using a single core of a 2014 macbook pro with i7 processor.

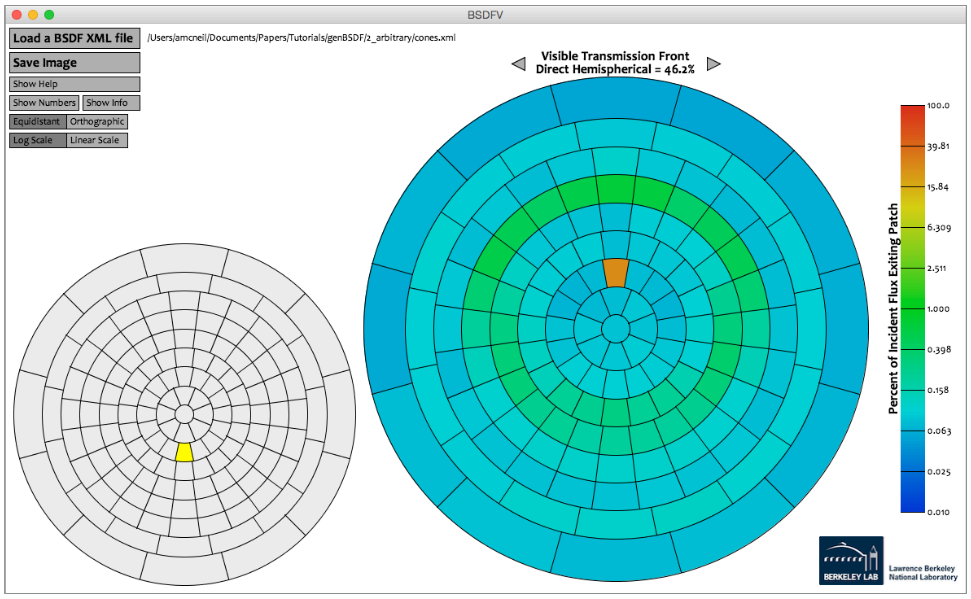

BSDFV
Load a BSDF XML file
/Users/amcneil/Documents/Papers/Tutorials/genBSDF/2_arbitrary/cones.xml
Save Image
Show Help
Show Numbers
Show Info
Equidistant
Orthographic
Log Scale
Linear Scale
Visible Transmission Front
Direct Hemispherical = 46.2%
Percent of Incident Flux Exiting Patch
100.0
39.81
15.84
6.309
2.511
1.000
0.398
0.158
0.063
0.025
0.010
BERKELEY LAB
Lawrence Berkeley National Laboratory

Figure 12 - Screenshot of BSDFview showing the front transmission distribution of the arbitrary shading system for light incident at 20° below horizontal.

Using BSDFviewer to look at the result for an incident angle from below horizontal (Figure 12) we can see a ring of green patches, which results from specular reflection off of the conical shapes. Since this shading system has bilateral (left-right) symmetry we can check the quality of our BSDF by looking at mirror patches. The direct-hemispherical transmission should be the same for patches mirrored about the center of the BSDF, and the distribution should be a mirror image. In Figure 13 two mirrored patches have direct hemispherical transmission values of 39.5% and 39.3%, less than 1% discrepancy.

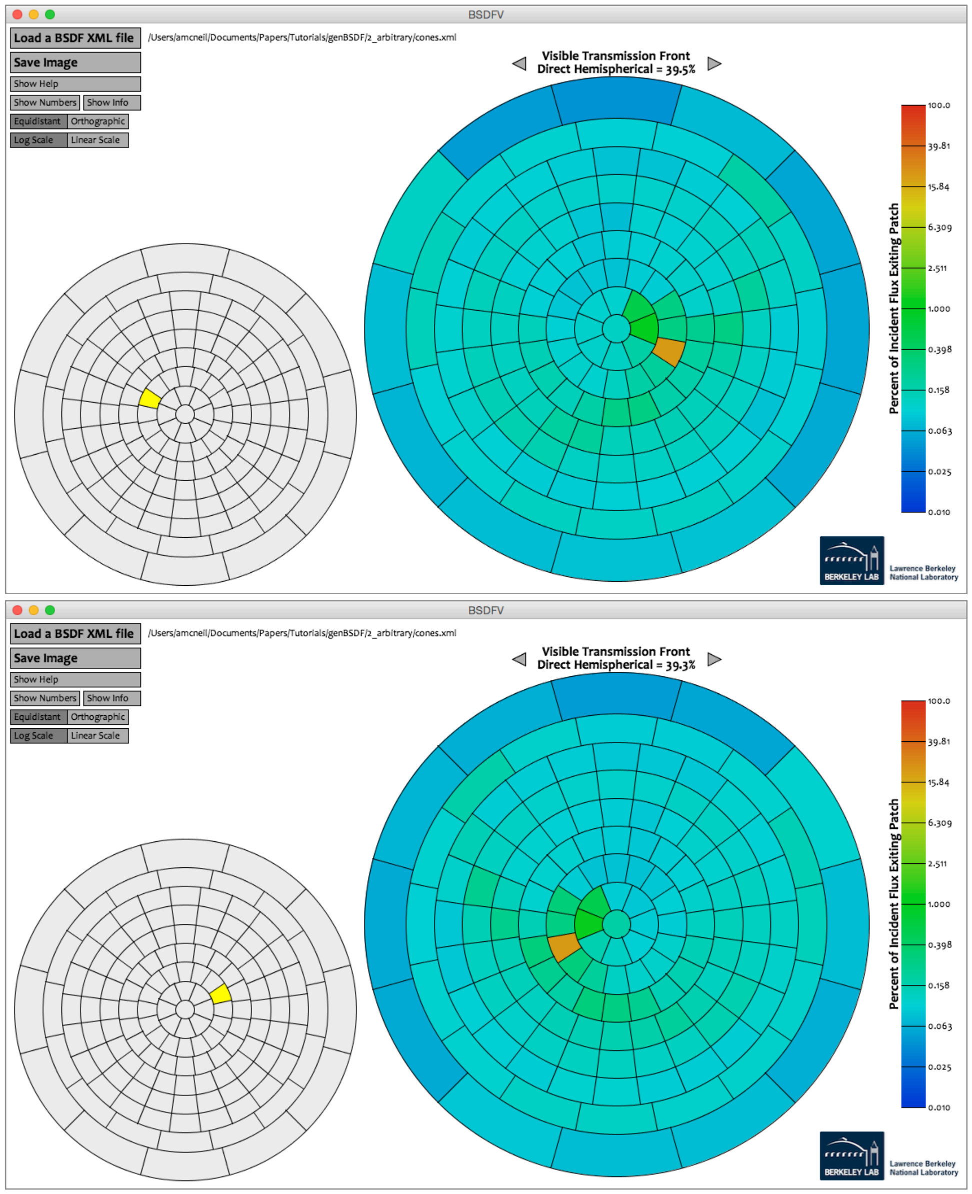

BSDFV
Load a BSDF XML file
/Users/amcnell/Documents/Papers/Tutorials/genBSDF/2_arbitrary/cones.xml
Save Image
Show Help
Show Numbers
Show Info
Equidistant
Orthographic
Log Scale
Linear Scale
Visible Transmission Front
Direct Hemispherical = 39.5%
Percent of Incident Flux Exiting Patch
100.0
39.81
15.84
6.309
2.511
1.000
0.398
0.158
0.063
0.025
0.010
BERKELEY LAB
Lawrence Berkeley National Laboratory
BSDFV
Load a BSDF XML file
/Users/amcnell/Documents/Papers/Tutorials/genBSDF/2_arbitrary/cones.xml
Save Image
Show Help
Show Numbers
Show Info
Equidistant
Orthographic
Log Scale
Linear Scale
Visible Transmission Front
Direct Hemispherical = 39.3%
Percent of Incident Flux Exiting Patch
100.0
39.81
15.84
6.309
2.511
1.000
0.398
0.158
0.063
0.025
0.010
BERKELEY LAB
Lawrence Berkeley National Laboratory

Figure 13 a,b - Front transmission distributions for mirrored patches. In the top image light is incident from the left just above horizontal. In the bottom image light is incident from the right just above horizontal.

6 Example 3 – Prismatic redirection device

In the third example, we're going to create a BSDF for a prismatic daylight-redirecting device. A section of the prism we'll be creating is shown in Figure 14 (don't read too much into this shape, it's not optimized in any way).

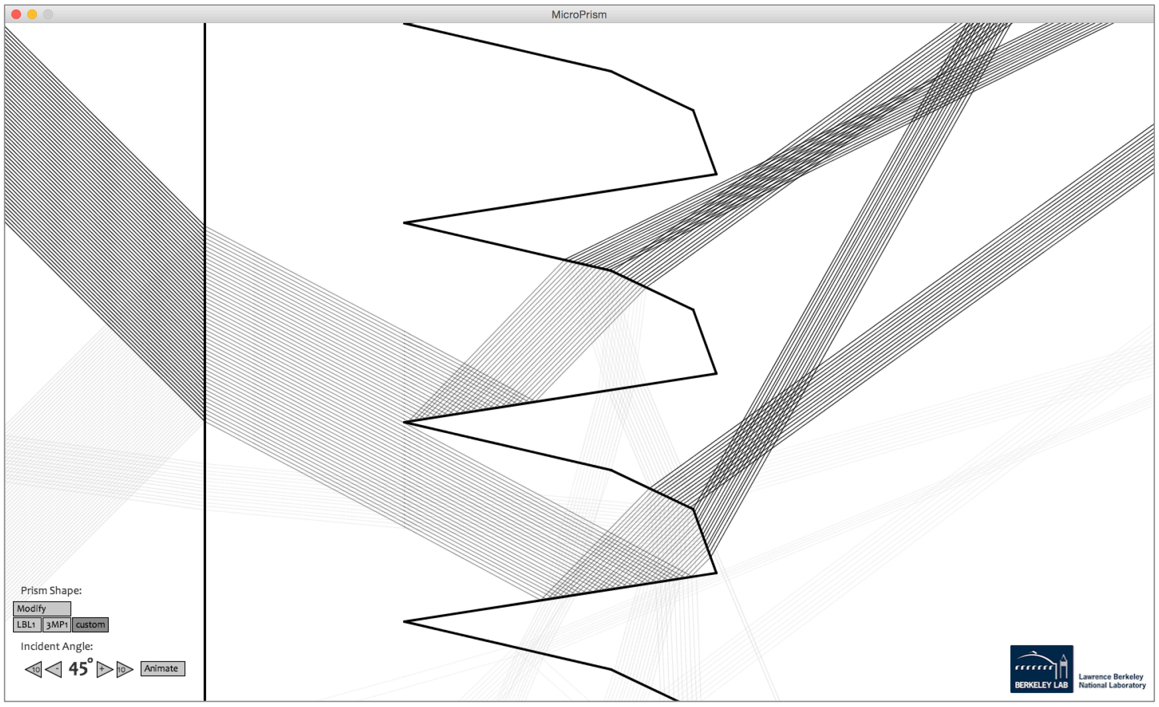


Figure 14 - Section profile of a prismatic daylight redirecting device with diagrammatic rays shown for an incidence angle of 45° above horizontal.

We'll start by defining a dielectric material using an index of refraction of 1.5 for our prismatic material. Each time a ray intersects with the dielectric surface the rays refract & reflect according to Snell's law. It's important that the surface normals of the model all face the correct direction (out). In this case the normal should always face out from the prism solid. I like to change the material temporarily to glow (which only glows on the side that the surface normal faces) then inspect the model with objvew to verify that all normal face the expected direction. Since this model is in millimeters, we'll use transmission coefficients of 0.997 (equivalent to 0.92 per inch).

Then we'll use the genprism program to create a prism shape (manual page: http://www.radiance-online.org/learning/documentation/manual-pages/pdfs/genprism.pdf). An xform command translates to the position we want (prisms facing into the room) and arrays the prism 50 times. Finally, we create a front face and thin edges to close our dielectric volume.

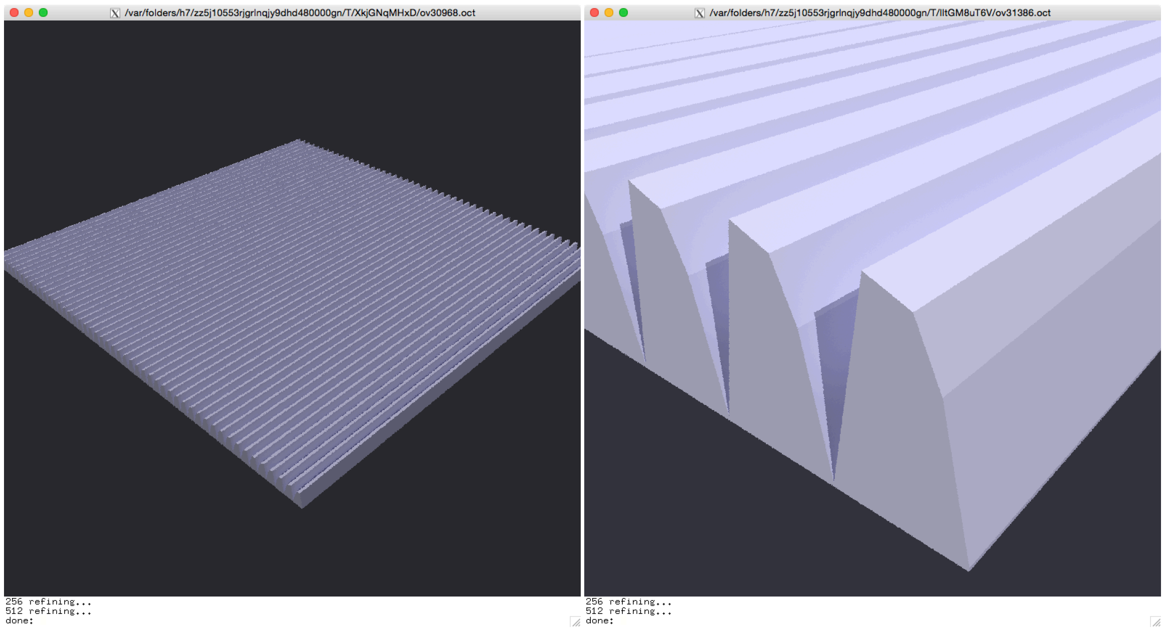
Our model file (prisms.rad) contains the following information:

```
# prims.rad
void dielectric prismface
0
0
5 .997 .997 .997 1.5 0

!genprism prismface prism \
5           0.0 0.0 \
            78.3 12.199997 \
            72.4 28.199997 \
            51.9 38.0 \
            0.0 50.0 \
            -c -l 0 0 -2500 | xform -ry -90 -t 0 -1250 -78.3 -a 50 -t 0 50 0

prismface polygon front
0
0
12          0               -1250           -80
            0               1250            -80
            2500            1250            -80
            2500            -1250           -80
prismface polygon edge1
0
0
12          0               -1250           -78.3
            0               1250            -78.3
            0               1250            -80
            0               -1250           -80
prismface polygon edge2
0
0
12          2500            -1250           -80
            2500            1250            -80
            2500            1250            -78.3
            2500            -1250           -78.3
prismface polygon edge3
0
0
12          0               -1250           -78.3
            0               -1250           -80
            2500            -1250           -80
            2500            -1250           -78.3
prismface polygon edge4
0
0
12          0               1250            -80
            0               1250            -78.3
            2500            1250            -78.3
            2500            1250            -80
```

Figure 15 shows screenshots of an rvu window containing the model. The prisms material was changed to plastic so that the shape of the prisms is visible and can be inspected for proper orientation.


/var/folders/h7/zz5j10553rjgrlnqjy9dhd480000gn/T/XkjGNqMHxD/ov30968.oct
/var/folders/h7/zz5j10553rjgrlnqjy9dhd480000gn/T/lltGM8uT6V/ov31386.oct

```
256 refining...
512 refining...
done:
```

```
256 refining...
512 refining...
done:
```

Figure 15 - screenshots of rvu window displaying the prism model (with dielectric material replaced by blue plastic so prisms are visible).

We'll use genBSDF's dim option to specify the location and size of the sampling face. The radiance program getbbox helps us by providing the bounding box for the prism model, so we can make sure we're sampling in the center.

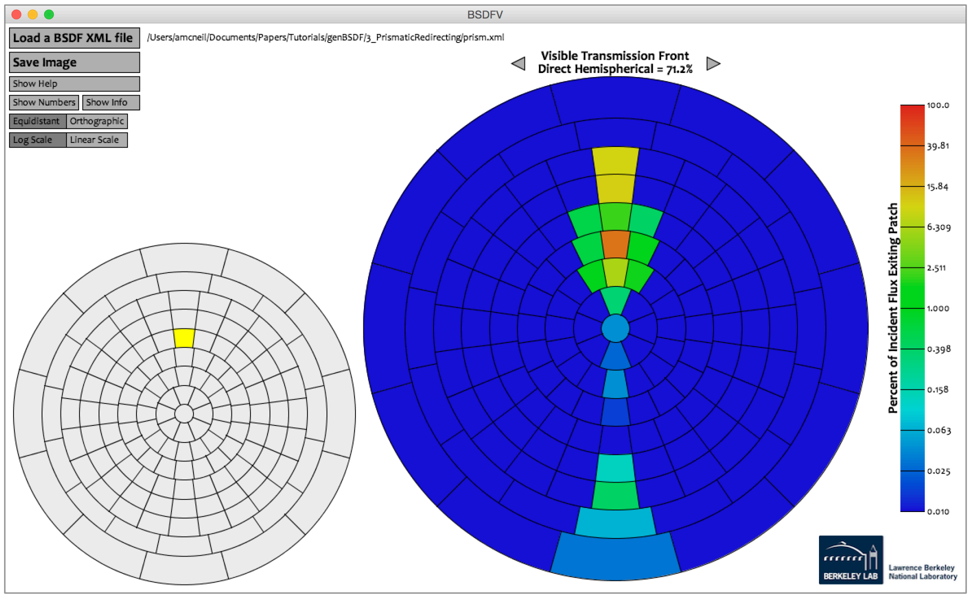
```
getbbox prism.rad
    xmin     xmax     ymin     ymax     zmin     zmax
      0      2500    -1250     1250      -80        0
```

Since a single prism spans 50 mm in height (y-axis) we set the y-dimension to that distance in the center of the model. The model is consistent across the x direction, so we can sample a small width in the center, say 2mm. Finally, we set the z bounds to fully encompass our model.

I've set the number of samples per Klems direction to 500, which should be plenty since our model only varies in one dimension. The number of ambient divisions was set to 1 because we don't expect any scattering with an ideal dielectric material.

```
genBSDF -c 500 +f +b -r '-ab 10 -ad 1’ -dim 1249 1251 -25 25 -80 0 -geom millimeter \
          prism.rad > prism.xml
```

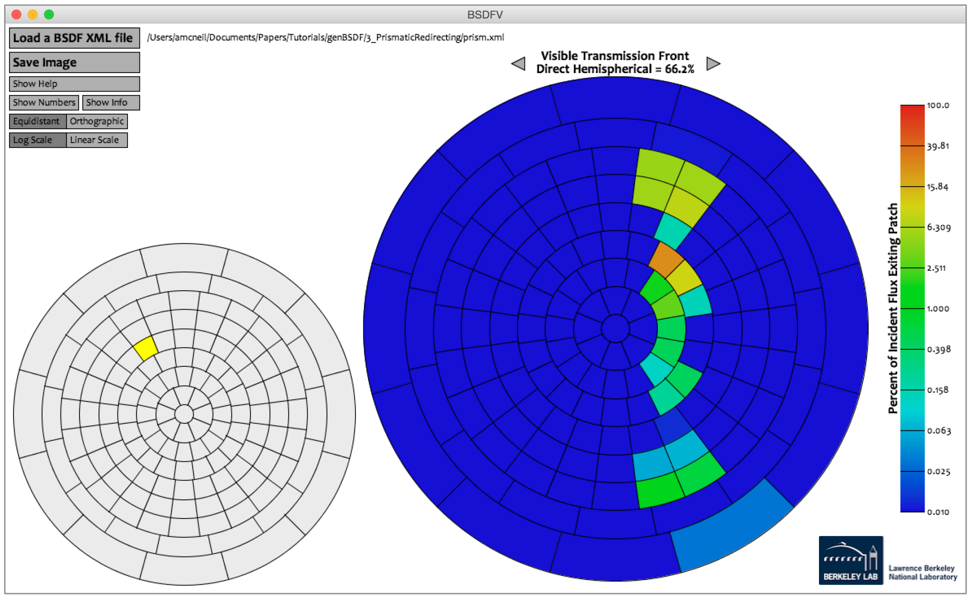
Figure 17 shows a screenshot of BSDFviewer. The BSDF redirects light as expected. Addtionally, mirrored patches exhibit similar (mirrored) distributions and total transmission as symmetry would dictate (Figure 17).



/Users/amcneil/Documents/Papers/Tutorials/genBSDF/3_PrismaticRedirecting/prism.xml

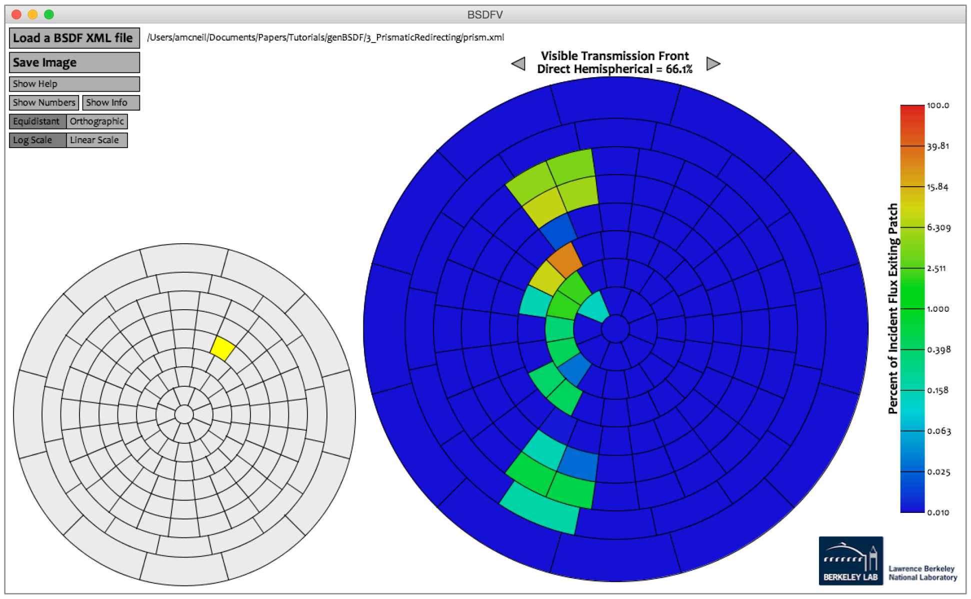
Visible Transmission Front
Direct Hemispherical = 71.2%

Figure 17 - BSDFviewer screenshot showing light incident from above horizontal redirected upward into the space.



/Users/amcneil/Documents/Papers/Tutorials/genBSDF/3_PrismaticRedirecting/prism.xml

Visible Transmission Front
Direct Hemispherical = 66.2%



/Users/amcneil/Documents/Papers/Tutorials/genBSDF/3_PrismaticRedirecting/prism.xml

Visible Transmission Front
Direct Hemispherical = 66.1%

Figure 16 - BSDFviewer screenshots showing mirror symmetry for two patches.

7 Acknowledgements

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Building Technologies Program, of the U.S. Department of Energy, under Contract No. DE-AC02-05CH11231

8 References

McNeil, A., Jonsson, C.J., Appelfeld, D., Ward, G., Lee, E.S. A validation of a ray-tracing tool used to generate bi-directional scattering distribution functions for complex fenestration systems, Solar Energy, Volume 98, Part C, December 2013, Pages 404-414. http://dx.doi.org/10.1016/j.solener.2013.09.032.

Molina, G., Bustamante, W., Rao, J., Fazio, P., Vera, S. Evaluation of radiance's genBSDF capability to assess solar bidirectional properties of complex fenestration systems (2015) Journal of Building Performance Simulation, 8 (4), pp. 216-225. http://dx.doi.org/10.1080/19401493.2014.912355