The Three-Phase Method for Simulating Complex Fenestration with Radiance

A McNeil, LBNL

Revision History

19 October 2010	created
18 March 2013	updated
2 April 2013	minor edits
3 April 2013	added gendaymtx to sections 2.1.2 & 3.4 and added annual dctimestep to section 2.5 & 3.4
23 July 2014	fixed error in command in section 2.4.1

---

1 Introduction

The "three-phase method" is a means to perform annual simulation of complex and/or dynamic fenestration systems. Flux transfer is broken into the following three phases for independent simulation:

1. Sky to exterior of fenestration
2. Transmission through fenestration
3. Interior of fenestration into the simulated space

Rather than simulate a specific daylight condition, the three-phase method calculates normalized coefficients that relate flux input to output for each phase. A result for a specific daylight condition is computed by multiplying the coefficient matrices by the input values (sky luminance values). Matrix calculation can be performed very quickly enabling the user to simulate many sky conditions and fenestration transmission properties.

This document starts with a brief overview of the three-phase method. Following is a detailed discussion of each phase of flux transfer including an explanation of new tools that were developed for the three-phase method, explained in detail. Two examples follow the detailed discussion, the first a simple space with one south facing window, the second a space with south and east facing windows.

1.1 Method overview

A matrix is used to characterize each phase of light transport. The input condition, sky luminance, is a vector. The result, illuminance values or a rendering, is also vector. The result is achieved by multiplying the sun vector by each matrix representing each phase of flux transfer. This process is described by the following equation:

i = VTDs or (1)
I = VTDS (2)

where:
i = point in time illuminance or luminance result
I = matrix containing time series of illuminance or luminance result
V= view matrix, relating outgoing directions on window to desired results at interior
T = transmission matrix, relating incident window directions to exiting directions (BSDF)
D = daylight matrix, relating sky patches to incident directions on window
s = sky vector, assigning luminance values to patches representing sky directions.
S = sky matrix, a collection of sky vectors

The **V** and **D** matrices are created with a Radiance simulation. The **T** matrix can be created using LBNL WINDOW software, by simulation (ie TracePro or Radiance genBSDF) or can be measured with a goniophotometer. The **s** vector is generated from a Radiance sky description.

1.2 File extension conventions

Radiance, like most Unix software, does not pay attention to file extensions. Any input file provided will be used regardless of the extension. If the file doesn't contain the information in a format that is expected Radiance will quit with an error. It is in the user's best interest to use consistent file extensions. This tutorial uses the following conventions for file extensions:

*.vmx - view matrix (data)
*.hdr - view matrix (renderings)
*.dmx - daylight matrix
*.skv - sky vector
*.smx – sky matrix
*.xml - transmission matrix (the file is xml format)

Other, generally accepted conventions for Radiance file extensions are:

*.rad - radiance scene data
*.oct - octree file
*.vf - view file
*.dat - data

Users who are not familiar with the radiance program rcontrib are urged to review the "Understanding rcontrib" Tutorial created by Axel Jacobs prior to reading this tutorial. Understaning rcontrib can be found here (linked as "rcontrib lesson") : http://www.jaloxa.eu/resources/radiance/documentation/

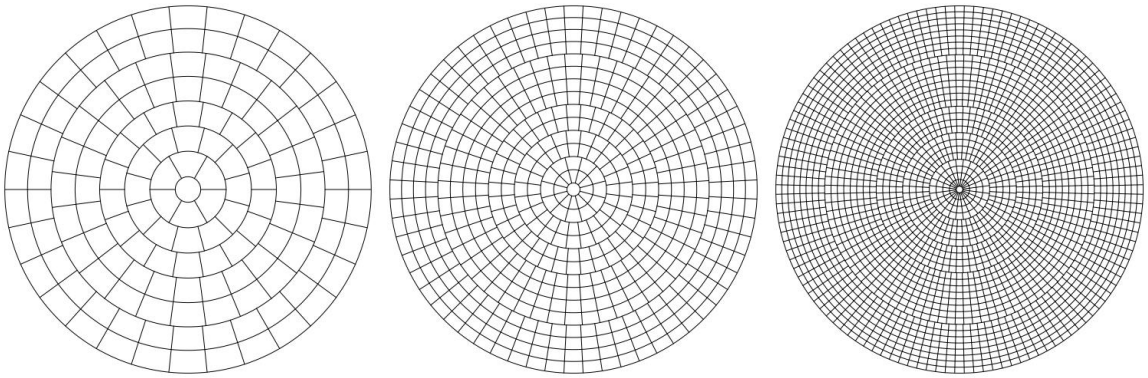
2 Discussion of Flux Transfer Phases

2.1 Sky

The sky vector/matrix isn't a phase of flux transfer; instead it's an input. It describes the starting condition (i.e. quantity and origin of flux). Generating a sky vector is covered in Axel's understanding rcontrib tutorial linked above. In this section we provide an example and illustration for completeness and recommend that users review sky vectors in Axel's tutorial. Additionally, this section describes how to use gendaymtx to generate a sky matrix file.

2.1.1 Sky Vector

The Radiance program ***genskyvec*** accepts a sky description and converts it to a sky vector. To generate a sky vector the sky is discretized using either the Tregenza or Reinhart division schemes. A sky vector is a list of average RGB radiance values for each descretized patch of the sky. The length of the vector is equal to the total number of sky divisions.

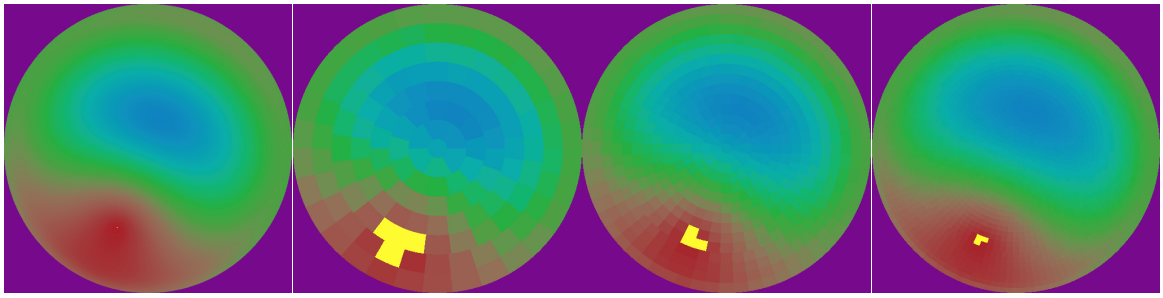


(a) (b) (c)

Figure 1. Orthographic projections of sky division schemes (a) Tregenza - 145 divisions, (b) Reinhart MF:2 - 580 divisions, and (c) Reinhart MF:4 - 2305 divisions.

A sky description from ***gensky*** or ***gendaylit*** can be piped directly to ***genskyvec*** to generate a sky vector.

```
$ gensky 1 21 11 | genskyvec -m 1 > skyvec_1-21-11_1.skv
$ gensky 1 21 11 | genskyvec -m 2 > skyvec_1-21-11_2.skv
$ gensky 1 21 11 | genskyvec -m 4 > skyvec_1-21-11_4.skv
```



(a) (b) (c) (d)

Figure 2. Visualizations of a continuous sky model (a) and discretized versions of the sky (b) Tregenza, (c) Reinhart MF:3 and (d) Reinhart MF:4.

2.1.2 Sky Matrix

A sky matrix is a time series set of sky vectors. An annual hourly sky matrix contains 8760 sky vectors, one for each hour of the year. There aren't restrictions on the number of timesteps or the size of the timestep in a sky matrix file.

The program ***gendaymtx*** generates a sky matrix from a **.**wea format data file. The wea format was created by DAYSIM developers, a converter from energyplus weather data (*.epw format) to *.wea format is distributed with DAYSIM. Users should download epw2wea from http://daysim.ning.com/

A sky matrix file can be generated as follows:

```
$ epw2wea USA_IL_Chicago-Midway.AP.725340_TMY.epw  Chicago.wea
$ gendaymtx Chicago.wea > Chicago.smx
```

The resulting file contains the Radiance associated with each patch for each hour of the year. The file is formatted so that RGB Radiance values for patch 0 are given in order. Following the 8760 lines for patch 0 is an empty line and then 8760 lines for patch 1, continuing through all patches).

gendaymtx uses Tregenza sky discretization by default, however this can be changed using the –m option:

```
$ gendaymtx -m 4 Chicago.wea > Chicago.smx
```

2.2 Transmission Matrix (BTDF)

The transmission matrix relates incident flux directions to an outgoing flux distributions for a fenestration system. The transmission matrix is a Bi-directional Transmission Distribution Function (BTDF) which contains outgoing flux coefficients in all directions for each incident direction

Radiance currently uses the WINDOW 6 xml format for a Bi-directional Scattering Distribution Function (BSDF) file. A This BSDF file contains a BTDF and BRDF (bi-directional reflection distribution function) for the front and back of a glazing system. Currently, Radiance only uses the front transmission data, thus front and back reflection and back transmission are ignored by Radiance.

This WINDOW 6 format uses a Klems angle basis to describe incoming and outgoing angles. The Klems angle basis is named after Joe Klems. The hemispherical divisions are arranged to so that each patch has relatively equal cosine-weighted solid angle. Figure 3 illustrates the 145 klems patches in an angular fisheye projection.

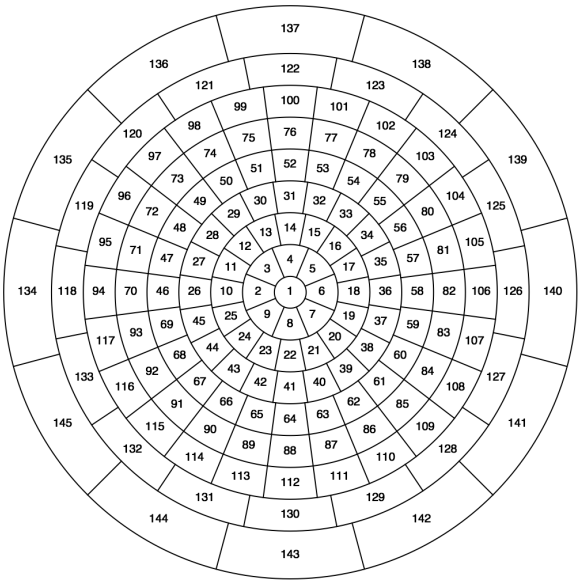


Figure 3. Klems 145-patch hemispherical basis with numbered subdivisions

LBNL's WINDOW software can generate a transmission matrix for a fenestration system. WINDOW (LBNL software) runs only in Windows (Microsoft operating system). The pre-release research version of WINDOW 7 can be downloaded from: http://windows.lbl.gov/software/window/window.html

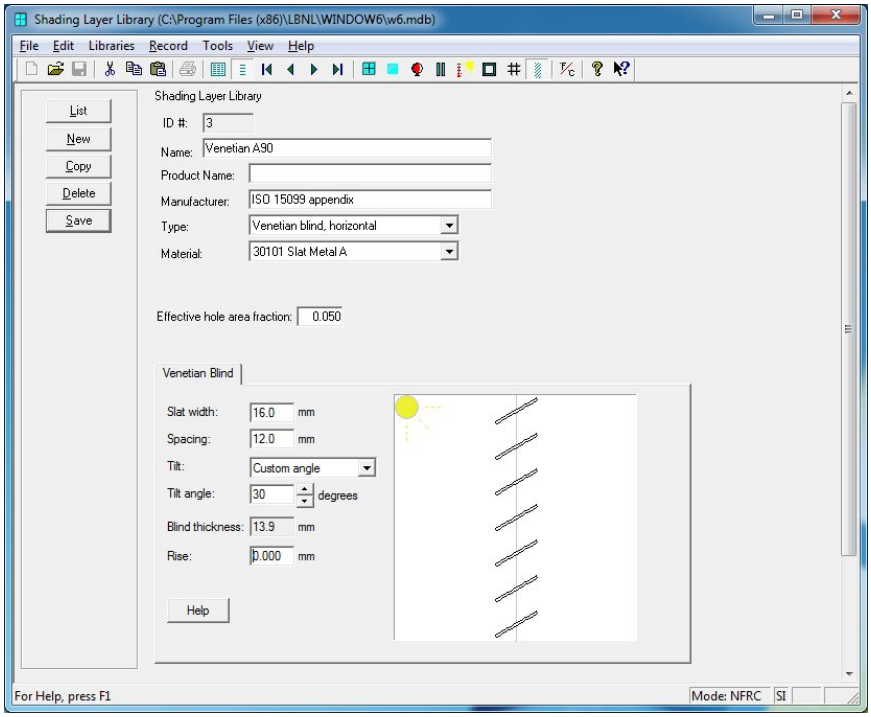


Figure 4. Venetian blind configuration input in WINDOW 7.

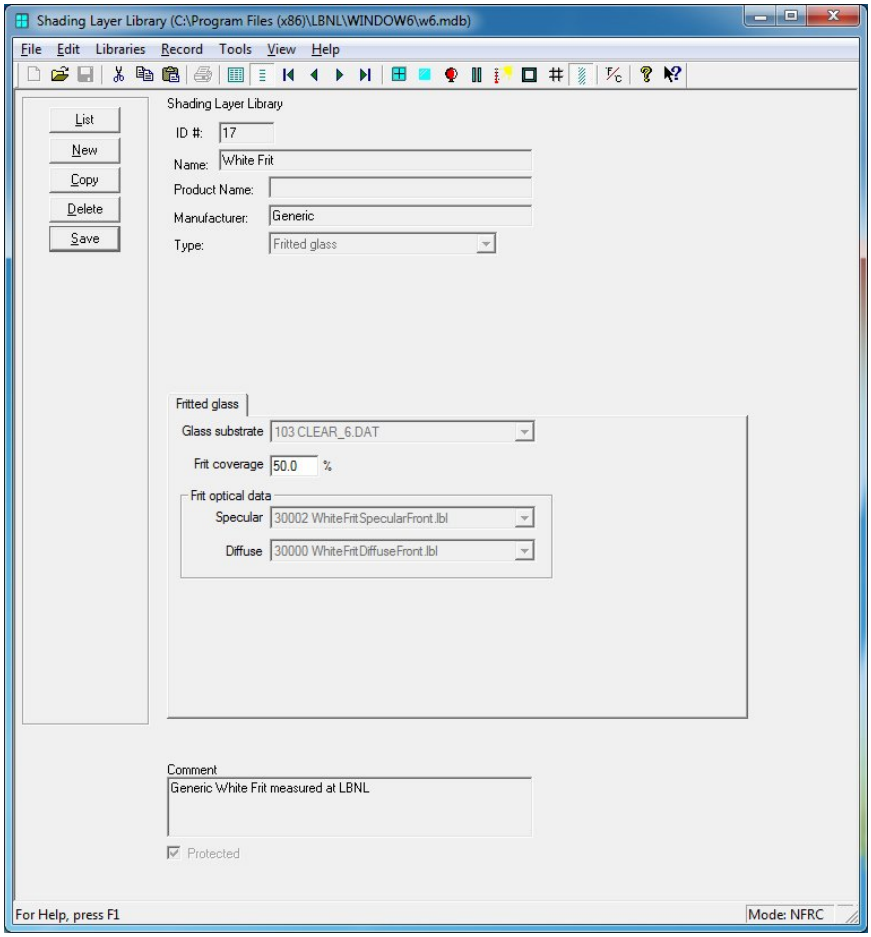


Figure 5. Frit configuration input in WINDOW 7.

2.3 Daylight Matrix

The daylight matrix contains luminous flux transfer coefficients from the sky divisions to the window's incident Klems divisions. Figure 6 is a hemispherical view from a window looking out to the surroundings. The random colored patches are the sky divisions. A diagram of the Klems divisions is overlaid onto the view. The scene geometry includes a ground plane and a nearby building.

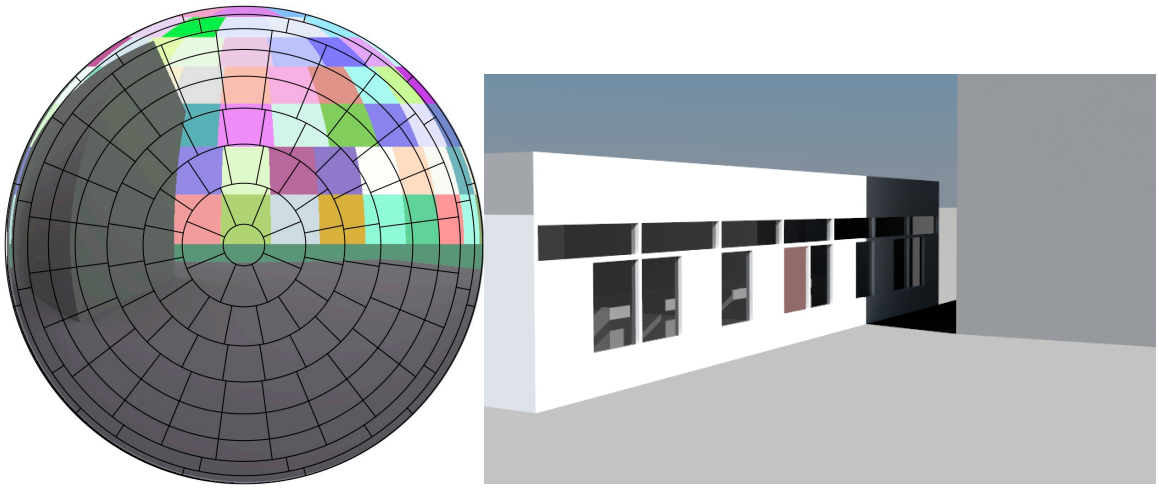


Figure 6. left: A hemispherical view looking out from a window overlayed with Klems divisions. Colored patches each colored patch is a Tregenza sky division. **right:** an exterior view of the model with the window used in the left image shaded pink.

The coefficients in the daylight matrix describe the amount of luminous flux from a sky patch that is incident on window in the direction of each Klems division. Figure 7 illustrates coefficients for a sky division. The coefficient is zero for Klems divisions that are filled with black sky. Coefficients for Klems divisions that have a direct view of the sky division has a coefficient that is relative to the amount of the division that is subtended by the division. The coefficient for a Klems divisions that contains nearby obstructions (gound or building) includes the contribution of reflected light.

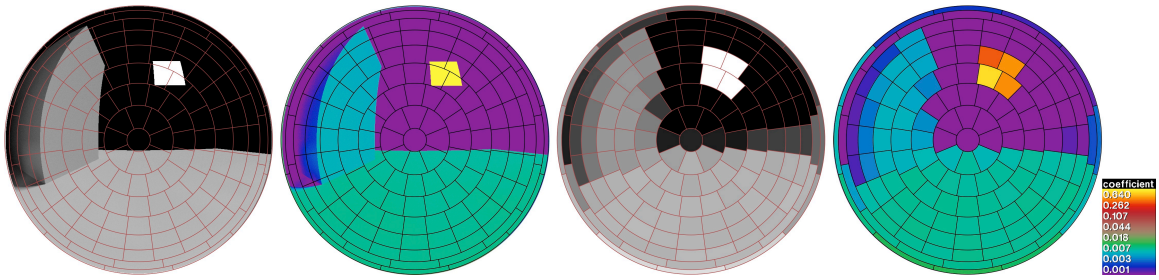


Figure 7. Renderings of contributions from Tregenza patch 74 (left two images) and visualizations of the D matrix coefficients for the same Tregenza patch (right two images). Reflections from model geometry (ground polygon and adjacent building) are included in the D matrix.

The daylight matrix includes a coefficient for the ground contribution. In cases where a ground polygon exists in the 3D model, the ground coefficient only includes light that originates between the modeled ground plane and the horizon. Contribution from the near ground is included as reflected light from the sky divisions. Using a ground plane allows the for the inclusion of shadow effects caused by nearby obstructions and the simulated building itself.

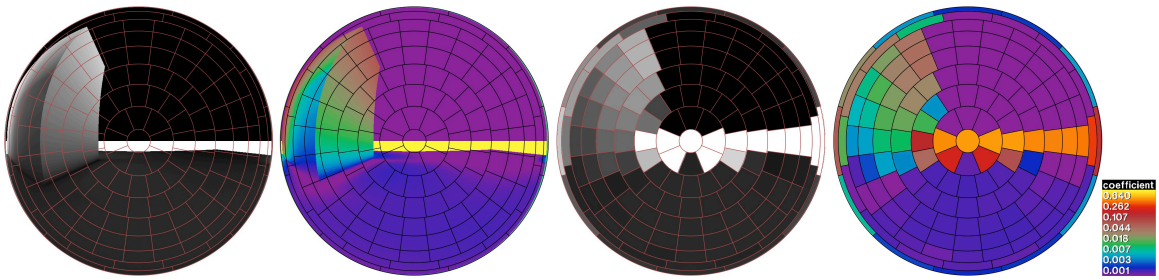


Figure 8. Renderings of contributions from the ground patch (left two images) and visualizations of the D matrix coefficients for the ground patch (right two images). Because this model includes a ground polygon, the ground patch fills the space between the edge of the ground polygon and the horizon.

The above are visual representations of parts of a daylight matrix. An actual daylight matrix is a matrix of RGB coefficients. Klems divisions are contained in rows and sky divisions are contained in columns. The above images depict a column from a daylight matrix.

2.3.1 genklemsamp

To generate a daylight matrix sampling rays generated by ***genklemsamp*** are passed to ***rcontrib*** as follows:

```
$ genklemsamp -vd 0 -1 0 window.rad | \
        rcontrib -c 1000 -e MF:4 -f reinhart.cal -b rbin -bn Nrbins -m uniform -faf \
                octrees/uniform.oct > output.dmx
```

The program ***genklemsamp*** generates sampling rays for klems divisions. The only required input for ***genklemsamp*** is the normal of the exterior surface of the fenestration, specified using the -vd (view direction) option.

```
$ genklemsamp -vd 0 -1 0
21.96971   -1e-05   -22.08820   0.02254421   -0.9990516   -0.03725006
20.91647   -1e-05   -21.52361  -0.009145640  -0.9989603   -0.04466038
19.07640   -1e-05   -22.10972   0.06378483   -0.9979531    0.003981985
16.97956   -1e-05   -21.42568   0.007601188  -0.9993630   -0.03486689
...
```

Each output line defines a sample ray. The first three fields are the x, y and z coordinates of the ray origin. The following three fields are the x, y and z components of the direction vector.

The ***-c*** option is used to set the number of samples produced per Klems division. The following command will produce one sample ray per Klems division, a total of 145 lines. To verify this we can pipe the output to ***wc*** (word count). The first output field from wc is the number of lines counted.

```
$ genklemsamp -c 1 -vd 0 -1 0 | wc
  145     870    9675
```

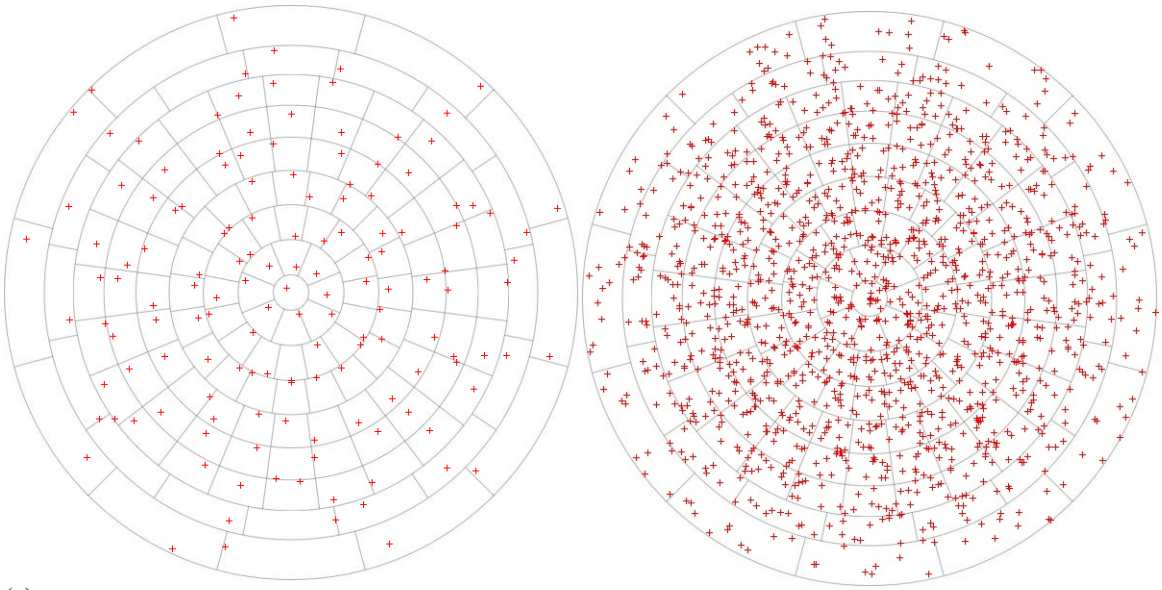
Changing to -c 10 produces ten sample rays per Klems divisions.

```
$ genklemsamp -c 10 -vd 0 -1 0 | wc
  1450    8700   96809
```

The default value for the -c option is 1000, producing 1000 sample rays per Klems division for a total of 145000 sample rays.

```
$ genklemsamp -vd 0 -1 0 | wc
  145000  870000  9681117
```

Figure 9 plots the output directions generated by ***genklemsamp*** on an equiangular projection overlaid with the Klems divisions illustrates ray directions with respect to Klems divisions.



(a) (b)

Figure 9. Plot of sample ray directions generated by genklemsamp for (a) one sample per Klems division and (b) ten samples per Klems division.

Ray origins produced by genklemsamp are randomly distributed over the window. The area for sampling can be specified either as a parallel view or by providing a file containing window polygons.

The parameters used to specify a parallel view are as follows:

View Option	Description
-vp x y z	center coordinate of the window (view point)
-vd xd yd zd	window normal vector (view direction)
-vu xd yd zd	window up vector for bsdf alignment (view up)
-vh val	window width in model units (view horizontal)
-vv val	window height in model units (view vertical)

The following example generates Klems sampling rays for a south facing window (-vd 0 -1 0) that is 80units x 40units (-vh 80 -vv 40) that is centered at (50,0,25). The sample ray origins are plotted in figure 10. The grey area represents the area of the window polygon. The red crosses indicate the position of a ray origin.

```
$ genklemsamp -vp 50 0 25 -vh 80 -vv 40 -vd 0 -1 0  >  samplerays_1.dat
```

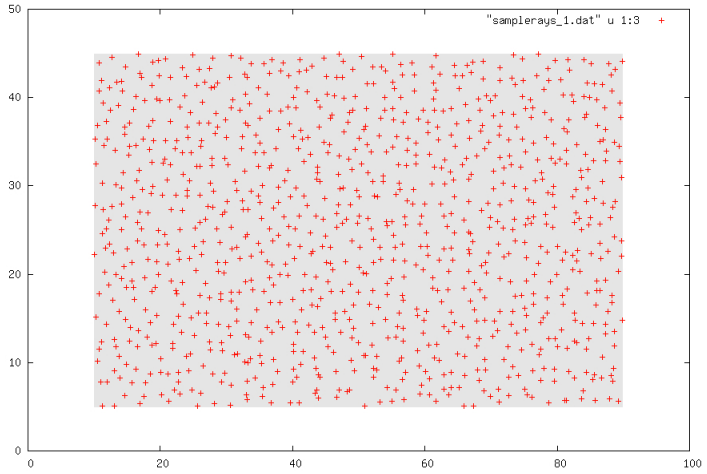


Figure 10. A plot of the ray origins produced for one outgoing Klems direction by genklemsamp for a parallel view specification. The gray area represents the area of the window.

Alternatively ***genklemsamp*** can be provided with a radiance geometry file containing polygons for sampling. For example the file 'window.rad' contains the following polygon:

```
void glass window
0
0
3 .5 .5 .5

window polygon window1
0
0
12    10 0 5
        90 0 5
        90 0 45
        10 0 45
```

The file windows.rad can be given to genklemsamp as follows:

```
$ genklemsamp -vd 0 -1 0 window.rad  >  samplerays_2.dat
```

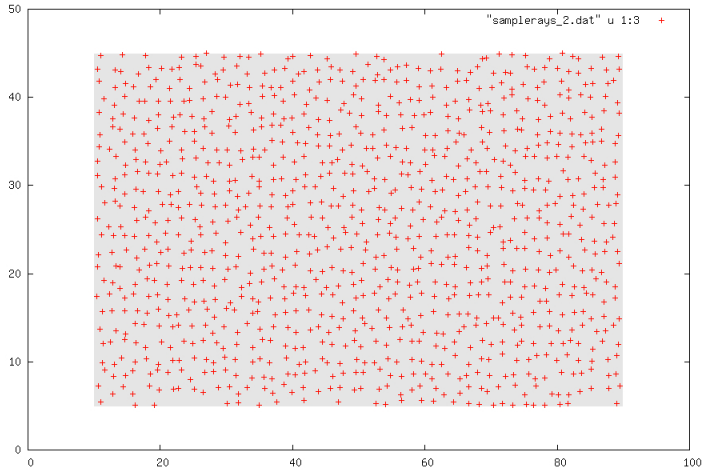


Figure 11. A plot of the ray origins produced for one outgoing Klems direction by genklemsamp for a radiance geometry file. The gray areas represent the polygon in the geometry file.

2.3.2 rcontrib (for daylight matrix)

Returning to our original example, sample rays produced by ***genklemsamp*** are passed directly to ***rcontrib*** to compute contribution coefficients from sky divisions.

```
$ genklemsamp -vd 0 -1 0 window.rad | \
        rcontrib -c 1000 -e MF:4 -f reinhart.cal -b rbin -bn Nrbins \
                -m skymat -faf octrees/model.oct > south.dmx
```

This example uses ***rcontrib*** in the same way as in Axel's Understanding rcontrib tutorial. The notable exceptions are the ***-c 1000*** option. The -c option tells rcontrib how many sample rays to accumulate. Since (in this example) genklemsamp produces 1000 sample rays per Klems division we want rcontrib to accumulate and average the results in blocks of 1000. The number of rays accumulated by ***rcontrib*** should always match the number produced by ***genklemsamp***.

2.4 View Matrix

The view matrix characterizes the relationship between light leaving a window and arriving at a point. The program ***rcontrib*** is used again to generate the view matrix. To characterize this relationship we first change the fenestration material to the Radiance glow type. The fenestration emits light in all directions into the space. Make sure the surface normals of the windows face into the space otherwise we would emit light out of the space.

2.4.1 rcontrib for view matrix

To calculate illuminace at several points we would create a text file (test.pts in this case) with our points and pass it to rcontrib.

```
$ rcontrib < test.pts -f klems_int.cal -b kbinS -bn Nkbins -m window_mat \
        -I+ -ab 12 -ad 50000 -lw 2e-5 interior.oct > south.vmx
```

The file klems_int.cal contains equations that allow rcontrib to bin rays hitting the window based on incident direction. klems_int.cal is distributed with the Radiance library files. Nkbins is defined in klems_int.cal and is the number of bins in which rays will be collected. kbinS is also defined in klems_int.cal, this sets the window orientation for the Klems bins. kbinS is for a window with a normal in the +y direction for the interior of the window. There is also kbinN kbinE and kbinW for other vertical window orientations and kbinD for flat skylights. Users can specify other window directions by adding them to the klems_int.cal file, or using the more general specification that takes normalized window normal and up vectors:

```
kbin(nx, ny, nz, ux, uy, uz)
```

2.4.2 vwrays for renderings

If we want to generate renderings instead of illuminance values, we can use ***vwrays*** to pass view rays to rcontrib. This was covered in Axel's tutorial so we'll only briefly mention it here.

```
$ vwrays -ff -vf view/fishout.vf -x 500 -y 500 \
        | rcontrib `vwrays -vf view/fishout.vf -x 500 -y 500 -d` -ffc -fo -o V_%03d.hdr \
                -f klems_int.cal -b kbinS -bn Nkbins -m window_mat \
                -ab 12 -ad 50000 -lw 2e-5 interior.oct
```

The first vwrays generates a sample ray for each pixel of the final image. The second vwrays is in backquotes in the rcontrib command line. The backquotes tell the shell to run the command and replace it with the output. With the -d option vwrays outputs the view dimensions, so this is just giving the -x and -y of the actual rays produced (the -x and -y is only really a guide, Radiance maintains the view aspect specified by the other view options, so the actual pixel dimensions may vary).

The -f option specifies input/output format. The first vwrays outputs floating point values (-ff) so rcontrib needs to expect floating point values. Then we want the output to be an image so we use 'c' for the output specification to get 4-byte radiance color specification use in hdr images. To accomplish we use -ffc for our input/output option.

The -o option specifies output format. Since we want one image per Klems bin we use a string format with %03d to separate into files by bin number (with 3 digits for the bin number). The -fo option forces output by overwriting existing files if there are any.

The result of this command is 145 image files in the format V_000.hdr, V_001.hdr ... V_145.hdr. Each image is a rendering of the window's contribution to the space for a single Klems bin.

2.5 Final result with dctimestep

We use dctimestep to multiply our sky vector and three matrices.

```
$ dctimestep south.vmx window_wBlinds.xml south.dmx skyvec_1-21-11_4.skv  \
        > illuminance_1-21-11.dat
```

The output file contains illuminance values at our test points for 11:00 on January 21.

We can generate a rendering using our bin renderings as follows:

```
$ dctimestep V_%03d.hdr window_wBlinds.xml south.dmx skyvec_1-21-11_4.skv > V_1-21-11.hdr
```

Instead of generating the skyvector in advance, we can generate a sky vector at the same time as the final result and pipe it directly to dctimestep as follows:

```
$ gensky 1 21 11 | genskyvec -m 4  | \
        dctimestep south.vmx window_wBlinds.xml south.dmx skyvec_1-21-11_4.skv \
        > illuminance_1-21-11.dat
```

Additionally we can generate an annual result with a sky matrix using the –n option with the number of timesteps and the –o option for image output:

```
$ dctimestep –n 8760 south.vmx window_wBlinds.xml south.dmx Chicago.smx \
        > illuminance_Chicago.dat
$ dctimestep –n 8760 –o Chicago_%04d.hdr \
        V_%03d.hdr window_wBlinds.xml south.dmx Chicago.smx
```

The resulting illuminance file contains one line for each simulation point containing RGB irradiance values for each of the 8760 simulated timesteps. In the second command, one rendered hdr files is produced for each of the 8760 timesteps.

3 Example 1

This example uses Axel's model from his "Understanding rtcontrib" lesson. To follow along you can download model files here: http://www.jaloxa.eu/resources/radiance/documentation/ (download the files for the rtcontrib lesson). The scene is a simple rectangular space with a large south-facing window.

3.1.1 Viewmatrix

We'll create both an illumiance sensor view matrix and a rendered view matrix. First we need to create a glow source that replaces the window in the model. Create the file window.rad in the objects directory:

```
# objects/window.rad
void glow windowglow
0
0
4  1   1   1   0

windowglow polygon window
0
0
12    0.5   -.15   1
      0.5   -.15   2
      3.5   -.15   2
      3.5   -.15   1
```

Then we generate a octree with the window light material:

```
$ oconv materials/testroom.mat objects/window.rad objects/testroom.rad > model_vmx.oct
```

Let's quickly check to make sure our surface normal faces the correct direction:

```
$ rvu -ab 2 -vf views/back.vf model_vmx.oct
```

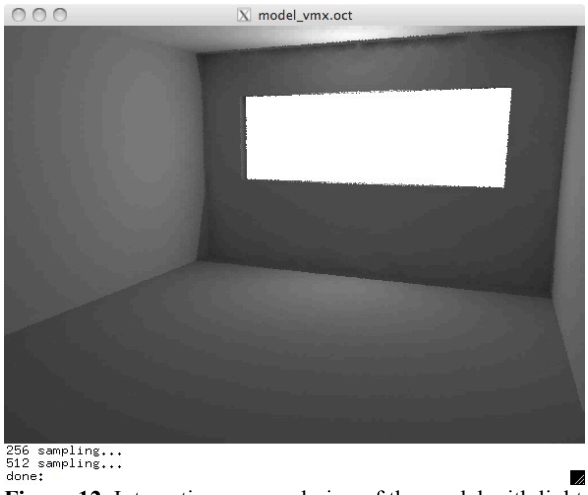


Figure 12. Interactive **rvu** rendering of the model with light emitting window polygon.

Yes, the surface normal is facing into the room. The light material only emits light in the direction of the surface normal, so if it were not facing into the room we'd be looking at a black room.

Next well generate our two view matrices. First create a directory in the images directory called vmx. Then we'll create our view matrix renderings and data file rcontrib:

```
$ mkdir images/vmx

$ vwrays -ff -vf views/back.vf -x 600 -y 600 | \
        rcontrib `vwrays -vf views/back.vf -x 600 -y 600 -d` -ffc  -fo \
                -o images/vmx/window_%03d.hdr  -f klems_int.cal -b kbinS -bn Nkbins \
                -m windowglow  -ab 12 -ad 50000 -lw 2e-5 model_vmx.oct

$ rcontrib  -f klems_int.cal -b kbinS -bn Nkbins -m windowglow -I+ -ab 12 -ad 50000 -lw 2e-5 \
        model_vmx.oct  < data/photocells.pts > results/photocells.vmx
```

We can combine all of our view matrix renderings into a single contact sheet using pcompos:

```
$ pcompos -a 10 images/vmx/window_*.hdr | \
        pfilt -x /6 -y /6 | \
        ra_tiff -z - images/vmxcontact.tif
```

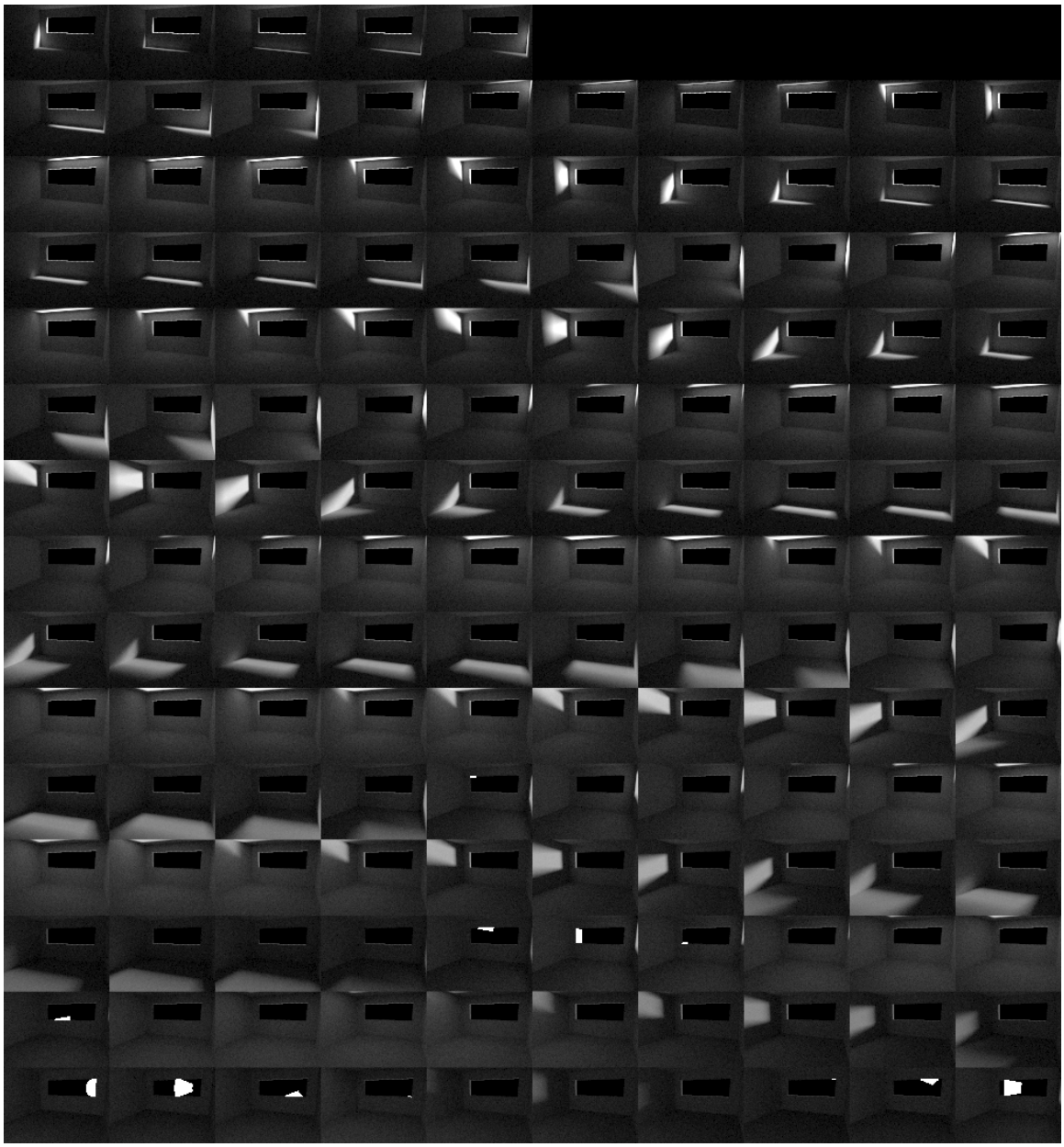


Figure 13. Contact sheet of Klems bin renderings. The lower left frame is bin 0, bin numbers increase to the right.

3.2 Daylight Matrix

To generate our daylight matrix we use genklemsamp with rcontrib. First we need to create our octree file. We'll use an external ground plane as well.

```
$ oconv materials/testroom.mat objects/ground.rad objects/testroom.rad \
        skies/sky_white1.rad > model_dmx.oct
```

Then we create a daylight matrix. We'll use the reinhart MF:4 sky subdivision for our run. Using the smaller subdivisions helps to keep the direct solar radiance in fewer of the klems window divisions.

```
$ genklemsamp -vd 0 -1 0 objects/window.rad | \
        rcontrib -c 1000 -e MF:4 -f reinhart.cal -b rbin -bn Nrbins -m sky_glow -faf \
                model_dmx.oct > results/south.dmx
```

3.2.1 Transmission Matrix (BSDF .xml file)

We'll generate three bsdf files using WINDOW 7. WINDOW 7 can be downloaded from here: http://windows.lbl.gov/software/window/window.html

When started the first thing to do is adjust the preferences for optical calcs by doing the following:

- check "use matrix method for specular systems"
- un-check "Solar band"
- check "Visible Band"
- change Angular basis to "W6 standard basis".

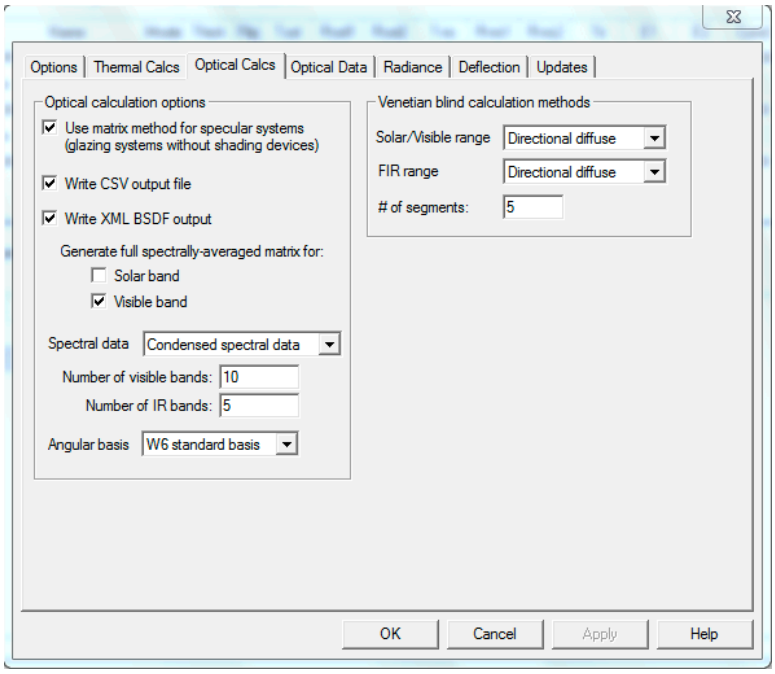


Figure 14. Preferences settings for WINDOW 7.

Then we'll create a glazing system that has a single pane of clear glass and an internal venetian blind. To add a blind, select "shade or frit" from the drop down menu for the second layer (where it says Glass 2).

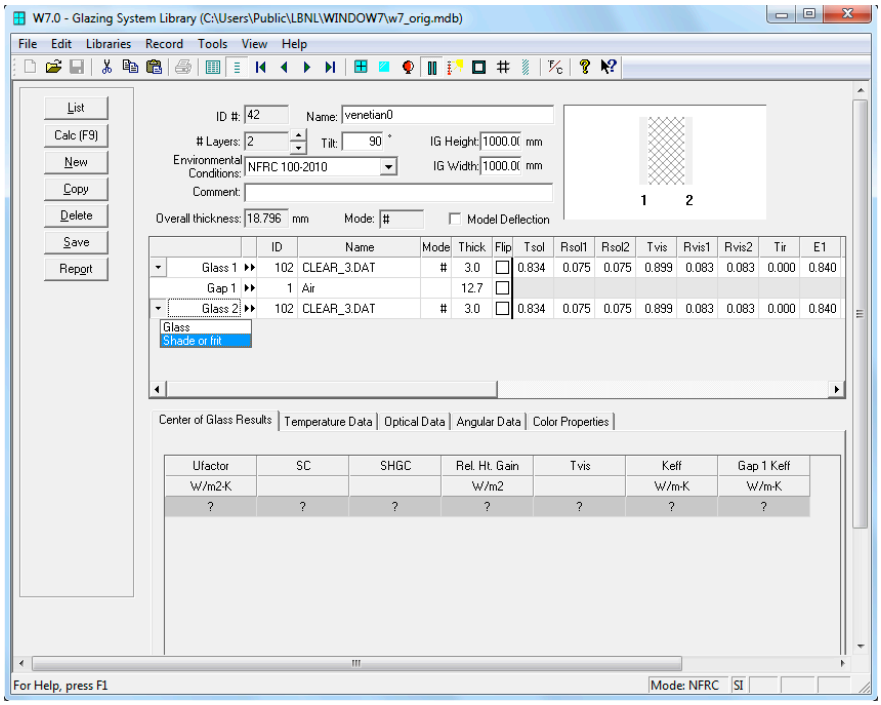


Figure 15. Selecting "Shade or frit" for layer 2.

The shade properties can be adjusted or a new shade can be created from within the shading layer library.

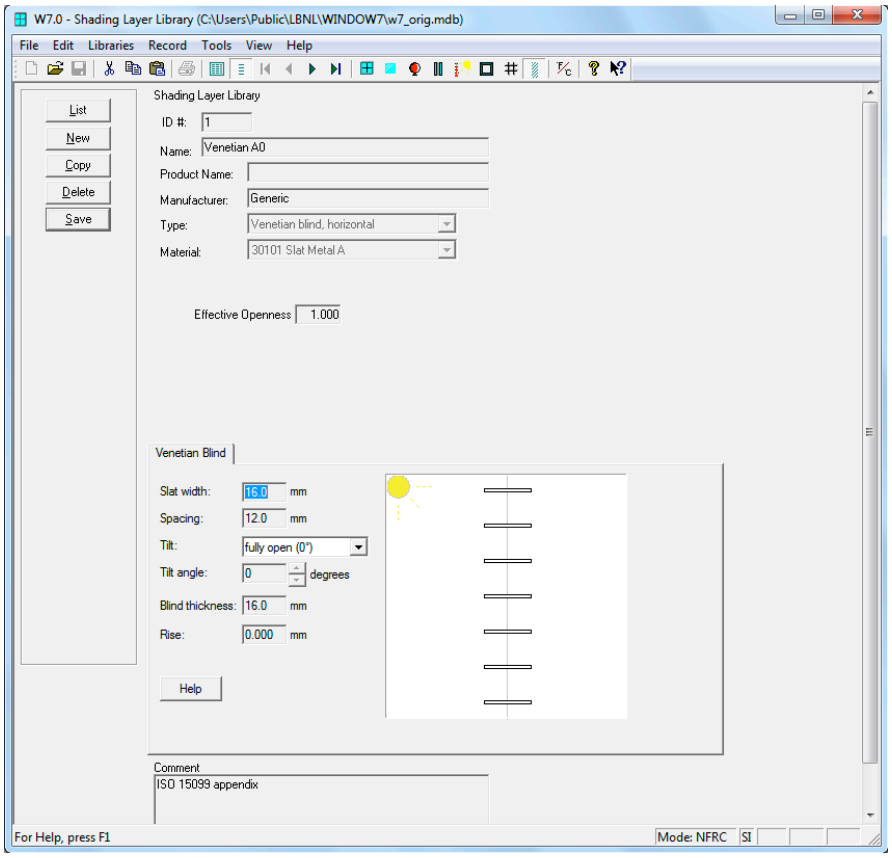


Figure 16. Venetian blind configuration for VenetianA0.

We'll use the Venetian A0 shade from the shading library.

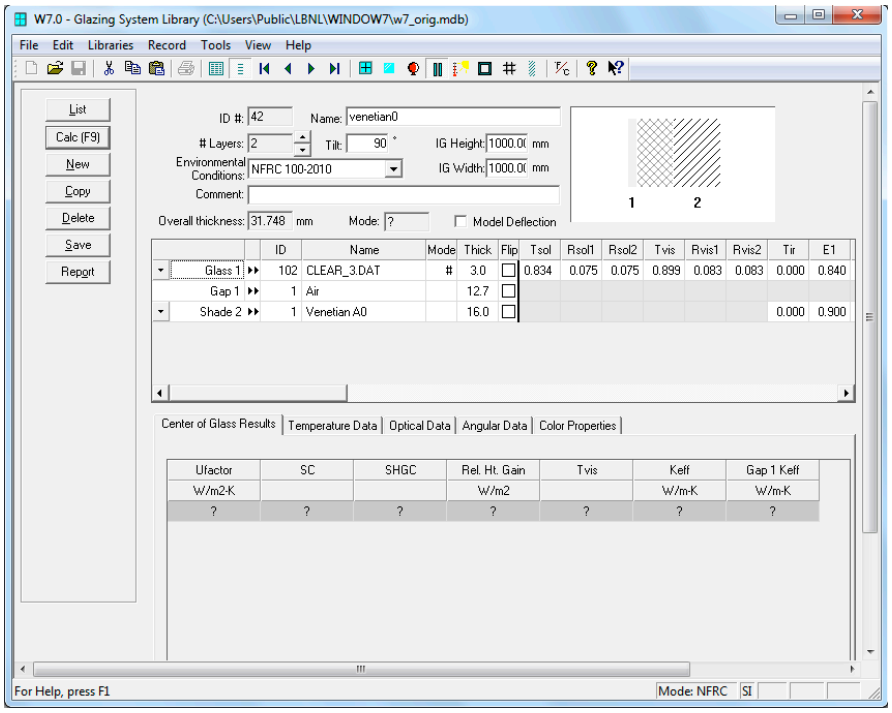


Figure 17. Complete glazing system with single pane glass and Venetian blind shading layer.

Then click the Calc [F9] button. An xml file with the name of the system will be created in the following directories depending on operating system:

```
Windows XP:   C:\Program Files\LBNL\WINDOW7
Windows 7:  C:\Users\Public\LBNL\WINDOW7\BSDFs
```

Then we follow the same process to generate a 45° venetian blind and a single glazed window with no blind. For convenience we should copy the xml files to the data directory.

3.3 Sky Vector

To generate the sky vector we use gensky and genskyvec:

```
$ gensky 12 21 15 | genskyvec -m 4 -c 1 1 1 > skies/12_21_15.skv
```

3.4 Putting it all Together

We use dctimestep to calculate illuminance for our three shading conditions:

```
$ dctimestep results/photocells.vmx data/singleclear.xml \
      results/south.dmx skies/12_21_15.skv | \
      rcalc -e '$1=179*($1*0.265+$2*0.670+$3*0.065)' > results/illum_122115_clear.dat
```

```
$ dctimestep results/photocells.vmx data/venetian0.xml \
      results/south.dmx skies/12_21_15.skv | \
      rcalc -e '$1=179*($1*0.265+$2*0.670+$3*0.065)'  > results/illum_122115_vb0.dat

$ dctimestep results/photocells.vmx data/venetian45.xml \
      results/south.dmx skies/12_21_15.skv | \
      rcalc -e '$1=179*($1*0.265+$2*0.670+$3*0.065)'  > results/illum_122115_vb45.dat
```

The dctimestep result is RGB irradiance values the rcalc command above converts to lux. Plotting the result yields the following graph:

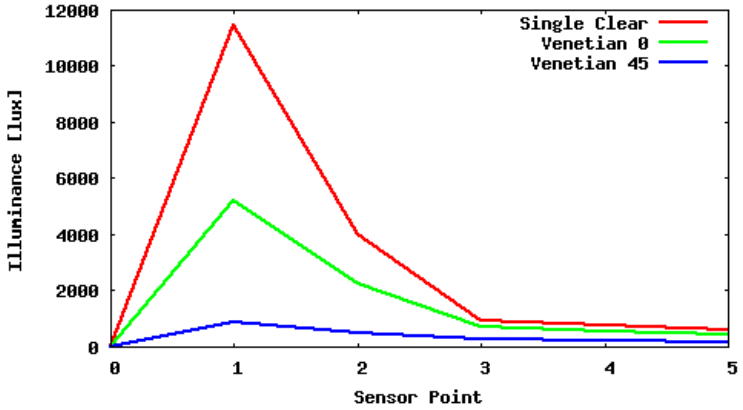


Figure 18: Plot of illuminance on Dec, 12 at 15:00 with no blind, venetian blind with 0° tilt, and venetian blind with 45° tilt.

The first sensor point (with x position of 0) is a rooftop horizontal illuminance sensor. Since the view matrix is generated with light emitted at the window, the result will always be zero for points outside of the space. The other points are horizontal work plane sensors 1m, 2m, 3m, 4m and 5m from the window.

```
epw2wea USA_IL_Chicago-Midway.AP.725340_TMY.epw Chicago.wea
gendaymtx -m 4 Chicago.wea > Chicago.smx
dctimestep -n 8760 results/photocells.vmx data/singleclear.xml results/south.dmx Chicago.smx \
        > results/illum_clear.dat
```

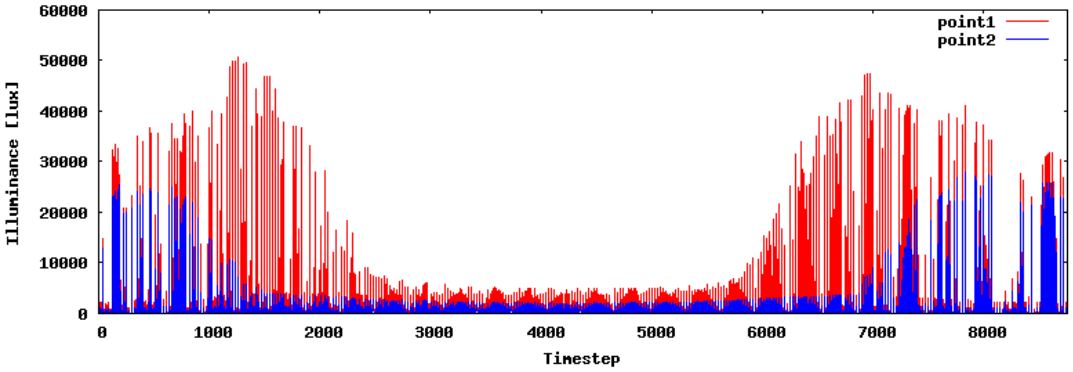


Figure 19: Plot of illuminance with no blind, using Chicago weather data.

We also use dctimestep to generate renderings:

```
$ dctimestep images/vmx/window_%03d.hdr data/singleclear.xml \
      results/south.dmx skies/12_21_15.skv  > images/122115_clear.hdr
$ pcond images/122115_clear.hdr | \
            pcompos -a 2 - '!falsecolor -s 2000 -log 2 -i images/122115_clear.hdr' | \
            ra_tiff -z - images/122115_clear.tif

$ dctimestep images/vmx/window_%03d.hdr data/venetian0.xml \
      results/south.dmx skies/12_21_15.skv  > images/122115_vb0.hdr
$ pcond images/122115_vb0.hdr | \
            pcompos -a 2 - '!falsecolor -s 2000 -log 2 -i images/122115_vb0.hdr' | \
            ra_tiff -z - images/122115_vb0.tif

$ dctimestep images/vmx/window_%03d.hdr data/venetian45.xml \
      results/south.dmx skies/12_21_15.skv > images/122115_vb45.hdr
$ pcond images/122115_vb45.hdr | \
            pcompos -a 2 - '!falsecolor -s 2000 -log 2 -i images/122115_vb45.hdr' | \
            ra_tiff -z - images/122115_vb45.tif
```

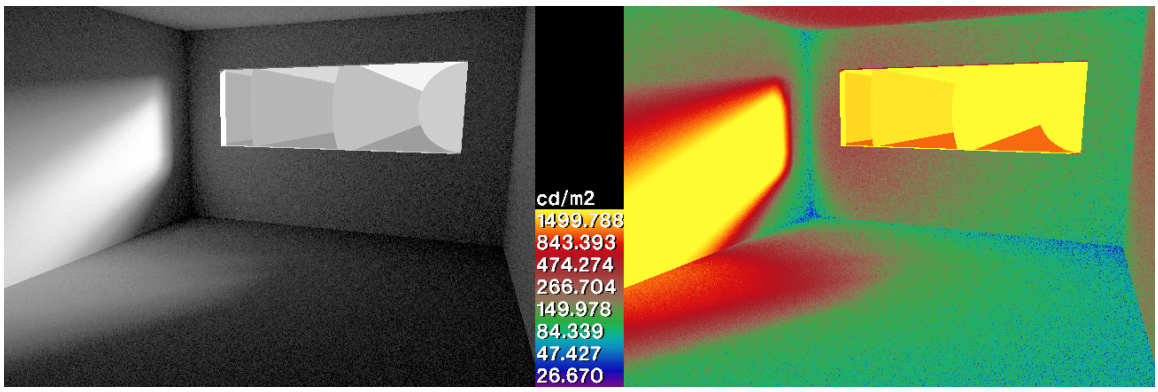


Figure 20: Renderings using singleclear.xml BSDF file (no shading layer).

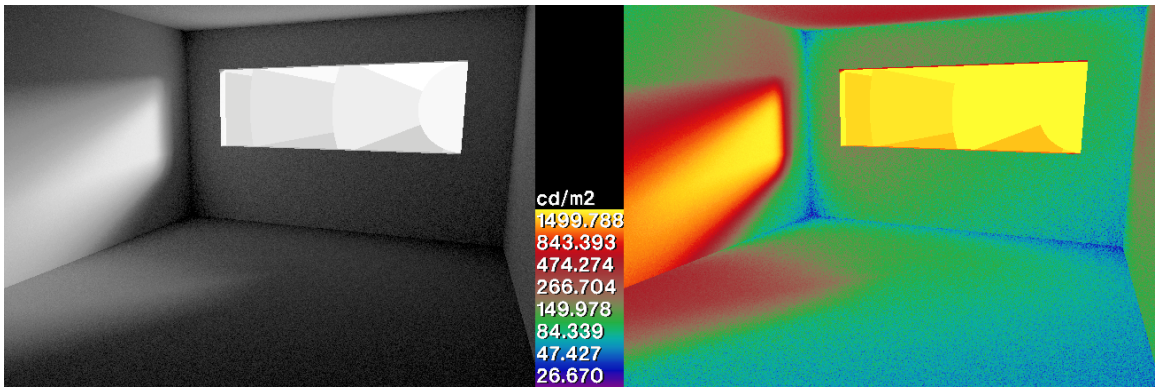


Figure 21: Renderings using venetian0.xml BSDF file (single glazing with Venetian blind at 0° tilt angle)

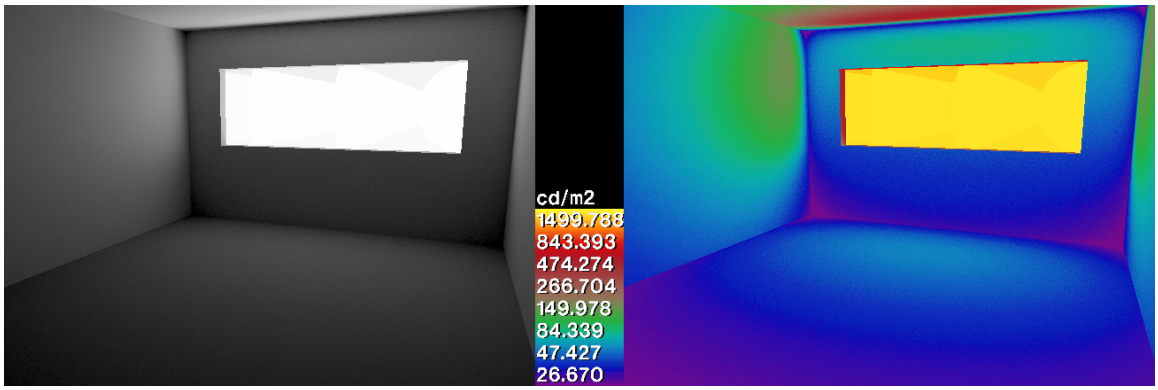


Figure 22: Renderings with venetian45.xml BSDF file (single glazing with Venetian blind at 45° tilt angle)

As is evident in the renderings, there is no physical representation of the blinds however the blind is included in the distribution and intensity of light entering the space. Also, the Klems divisions are visible in the window much like Tregenza divisions would be visible in the sky for daylight coefficient rendering.

We can also generate annual renderings using dctimestep

```
dctimestep -n 8760 -o annualimages/images_%04d.hdr images/vmx/window_%03d.hdr \
        data/singleclear.xml results/south.dmx Chicago.smx
```

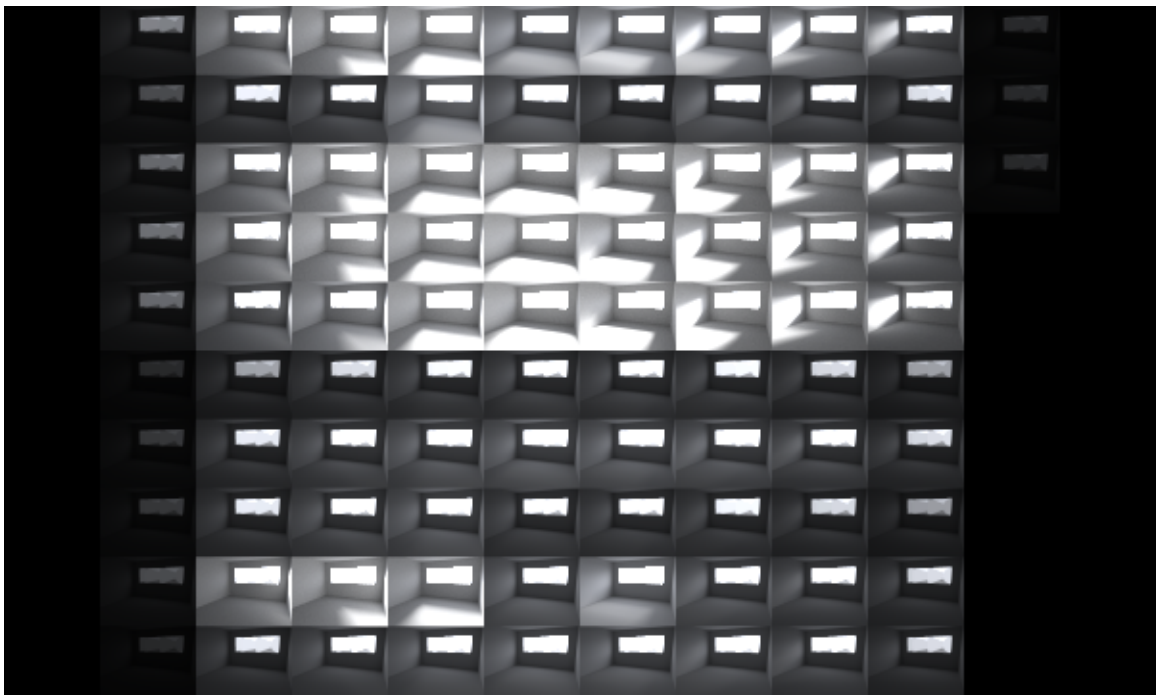


Figure 23: Contact sheet of first 10 days of January using Chicago weather data.

4 Example 2

This second example is more complex. It uses the same rectangular space with three windows, two south facing and one east facing.

We'll modify the example scene from Axel's tutorial to split the south facing window into two and add an east-facing window. Our new scene geometry is as follows. We'll name it testroom2.rad.

```
# objects/testroom2.rad
!genbox floor_mat floor 4.6 6.6 .3 |xform -t -.3 -.3 -.3
!genbox ceiling_mat ceiling 4.6 6.6 .3 |xform -t -.3 -.3 2.5

!genbox wall_mat wall_w .3 6.6 2.5 |xform -t -.3 -.3 0
!genbox wall_mat wall_n 4 .3 2.5 |xform -t 0 6 0

!genbox wall_mat wall_s_bottom 4 .3 1 |xform -t 0 -.3 0
!genbox wall  mat wall  s  top 4 .3 .5 |xform -t 0 -.3 2
!genbox wall_mat wall_s_west .5 .3 1 |xform -t 0 -.3 1
!genbox wall  mat wall  s  center .5 .3 1 |xform -t 1.75 -.3 1
!genbox wall_mat wall_s_east .5 .3 1 |xform -t 3.5 -.3 1

!genbox wall_mat wall_e_bottom .3 6.6 1 |xform -t 4 -.3 0
!genbox wall_mat wall_e_top .3 6.6 .5 |xform -t 4 -.3 2
!genbox wall_mat wall_e_south .3 1.3 1 |xform -t 4 -.3 1
!genbox wall_mat wall_e_south .3 3.75 1 |xform -t 4 2.55 1

#EOF
```

A quick rvu rendering of our new scene:

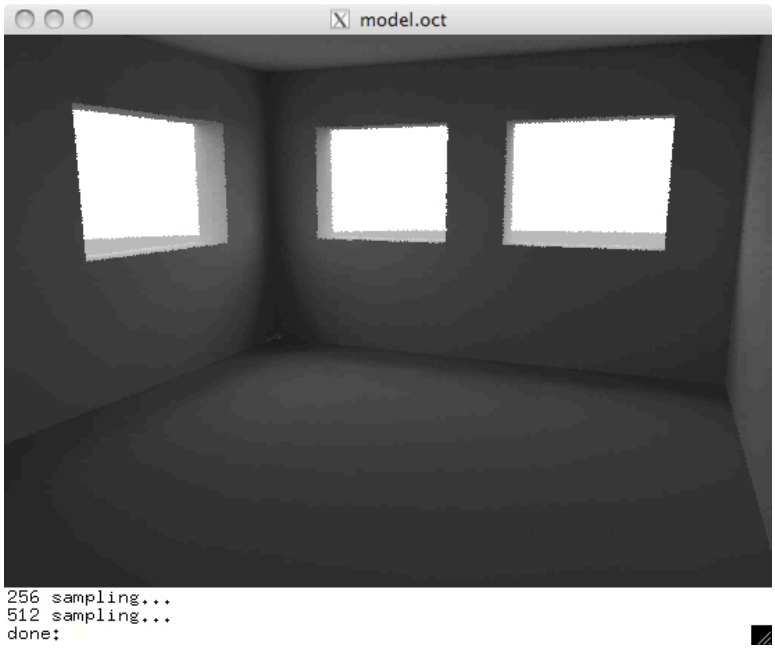


Figure 24. An interactive rvu rendering of the new scene with three windows.

Next we need to make our window polygons for the windows. We'll make two seperate files, one for the south facing windows and one for the east facing window.

```
# objects/south_windows.rad
void glow window_south
0
0
4   1   1   1   0

window  south polygon window
0
0
12    0.5   -.15   1
      0.5   -.15   2
      1.75   -.15   2
      1.75   -.15   1

window_south polygon window
0
0
12    2.25   -.15   1
      2.25   -.15   2
      3.5   -.15   2
      3.5   -.15   1

# objects/east_windows.rad
void glow window_east
0
0
4   1   1   1   0

window_east polygon window
0
0
12    4.15    1       1
      4.15   1       2
      4.15   2.25   2
      4.15   2.25   1
```

We can group the two south facing windows because the have the same orientation and there are no external obstructions that cause discrepancies in incident daylight on the windows.

4.1 View matrices

We'll create view matrices in the same way as before, except now we'll create one view matrix for the south facing windows and one view matrix for the east facing window. We can create view matrices for both window groups at the same time using one rcontrib command with two material type definitions (-m east and -m south). We need to use a different -b option to specify window orientation. Additionally we need to add a %s to the output specification so that the two matrices are put in separate output files.

First we create our new octree:

```
$ oconv materials/testroom.mat objects/testroom2.rad objects/south_window.rad \
        objects/east_window.rad objects/ground.rad > model2_vmx.oct
```

Then we'll create the photosensor view matrix:

```
$ rcontrib  -f klems_int.cal -bn Nkbins -fo -o results/photocells_%s.vmx \
            -b kbinS -m window_south  \
            -b kbinE -m window_east \
            -I+ -ab 12 -ad 50000 -lw 2e-5 model2_vmx.oct < data/photocells.pts
```

Before creating the rendered view matrix, unix type operating systems have a limit on the number of files that can be opened simultaneously by the shell. To ascertain the limit we can use ulimit:

```
$ ulimit –n
256
```

With Mac OSX the command returns 256, meaning we can open 256 files simetaneously. However when we create the rendered view matrix we will create 145 image files for each window group (south and eas) for a total of 290 image files. In order to do this we need to increase the limit on open files.

```
$ ulimit –n 512
$ ulimit –n
512
```

Now we are ready to generate the rendered view matrix (this may take a while):

```
$ vwrays -ff -vf views/back.vf -x 600 -y 600 | \
        rcontrib `vwrays -vf views/back.vf -x 600 -y 600 -d` -ffc -fo -o images/vmx/%s_%03d.hdr  \
                -f klems_int.cal -bn Nkbins \
                -b kbinS  -m window_south \
                -b kbinE -m window_east  \
                -ab 12 -ad 50000 -lw 2e-5 model2_vmx.oct
```

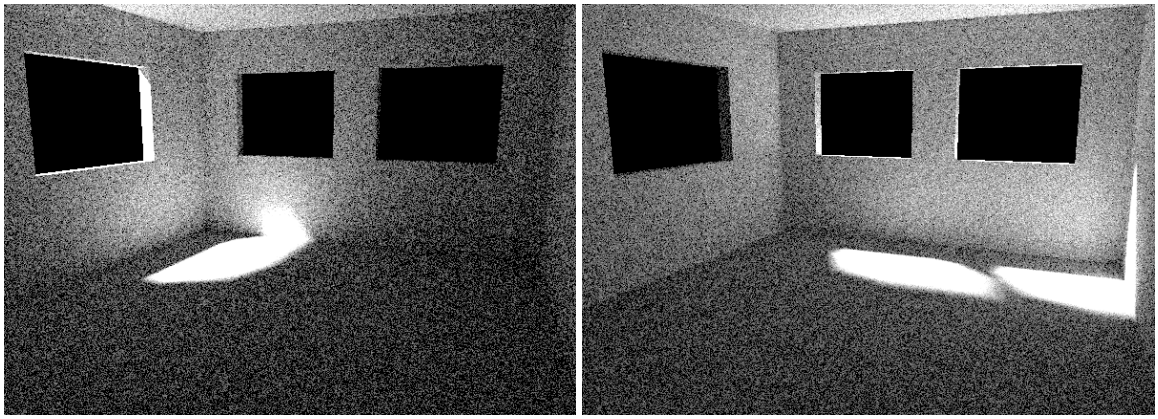


Figure 25. Renderings of outgoing window contributions for Klems patch 109 for east (left) and south (right) window groups.

4.2 Daylight Matrix

We also need to create two daylight matrices, one for the east facing windows and one for the south facing windows. For the south facing windows we can give ***genklemsamp*** the rad file containing both window polygons and it will distribute samples over both windows.

```
$ genklemsamp -vd 0 -1 0 objects/south_window.rad > southsamples.dat
```

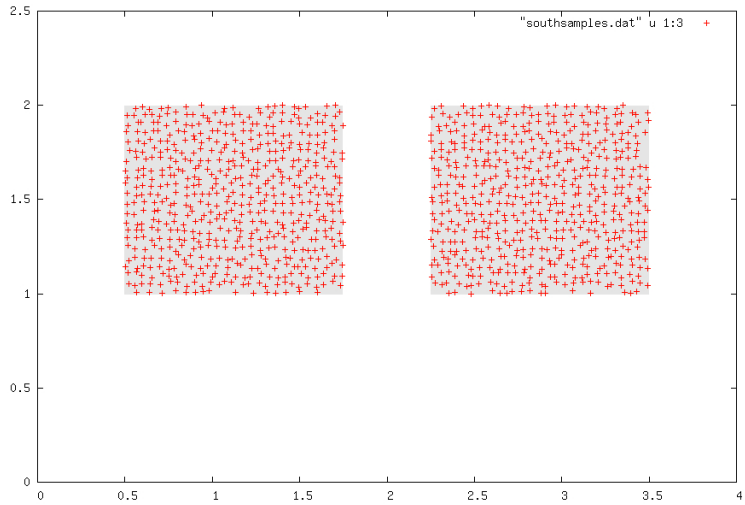


Figure 26. sample ray origin distribution generated by **genklemsamp** for two polygons contained in the input file.

First we create or daylight matrix octree file:

```
$ oconv materials/testroom.mat objects/testroom2.rad objects/ground.rad \
      skies/sky_white1.rad > model2_dmx.oct
```

Then we create a daylight matrix file for each window group:

```
$ genklemsamp -vd 0 -1 0 objects/south_window.rad | \
      rcontrib -c 1000 -e MF:4 -f reinhart.cal -b rbin -bn Nrbins -m sky_glow \
      -faf model2_dmx.oct > results/south.dmx

$ genklemsamp -vd 1 0 0 objects/east_window.rad | \
      rcontrib -c 1000 -e MF:4 -f reinhart.cal -b rbin -bn Nrbins -m sky_glow \
      -faf model2_dmx.oct > results/east.dmx
```

4.3 Putting it all together

We'll use the BSDF files created in the first example (singleclear.xml, venetian_0.xml and venetian_45.xml). We'll generate two skyvectors for this example. One early morning sky with the sun shining in the east window and one afternoon sky with the sun shining in the south windows.

```
$ gensky 5 21 13 | genskyvec -m 4 -c 1 1 1 > skies/5_21_13.skv
$ gensky 5 21 7 | genskyvec -m 4 -c 1 1 1 > skies/5_21_07.skv
```

Then we'll use pcomb with inline dctimestep commands to generate a single image with both window group contributions.

```
$ pcomb '!dctimestep images/vmx/window_east_%03d.hdr data/singleclear.xml \
                results/east.dmx skies/5_21_13.skv' \
        '!dctimestep images/vmx/window_south_%03d.hdr data/singleclear.xml \
                results/south.dmx skies/5_21_13.skv' \
        > images/5_21_13_clear-clear.hdr

$ pcond images/5_21_13_clear-clear.hdr | \
        pcompos -a 2 - '!falsecolor -s 2000 -log 2 -i images/5_21_13_clear-clear.hdr' | \
        ra_tiff -z - images/5_21_13_clear-clear.tif

$ pcomb '!dctimestep images/vmx/window_east_%03d.hdr data/singleclear.xml \
                results/east.dmx skies/5_21_13.skv' \
        '!dctimestep images/vmx/window_south_%03d.hdr data/venetian0.xml \
                results/south.dmx skies/5_21_13.skv' \
        > images/5_21_13_clear-vb0.hdr

$ pcond images/5_21_13_clear-vb0.hdr | \
        pcompos -a 2 - '!falsecolor -s 2000 -log 2 -i images/5_21_13_clear-vb0.hdr' | \
        ra_tiff -z - images/5_21_13_clear-vb0.tif
```

You can see how the shading affects the sun patch from the south windows.

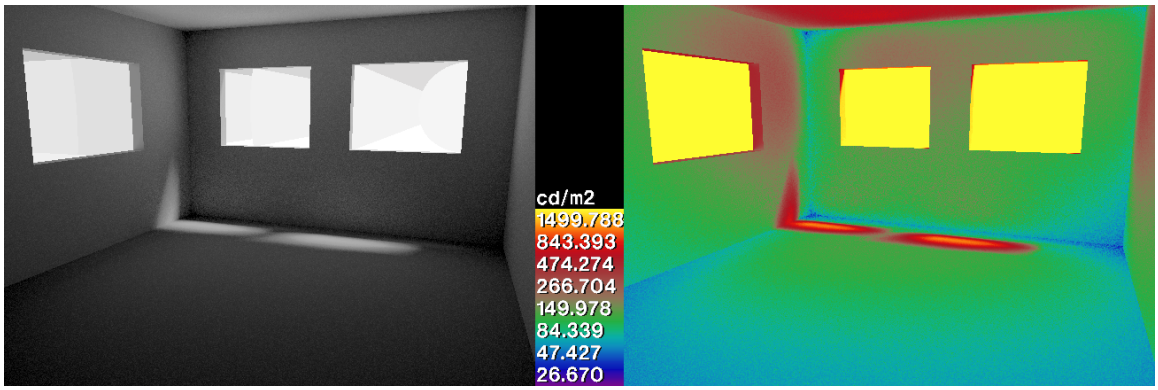


Figure 27. Rendering for 1 PM May 21st, no blinds.

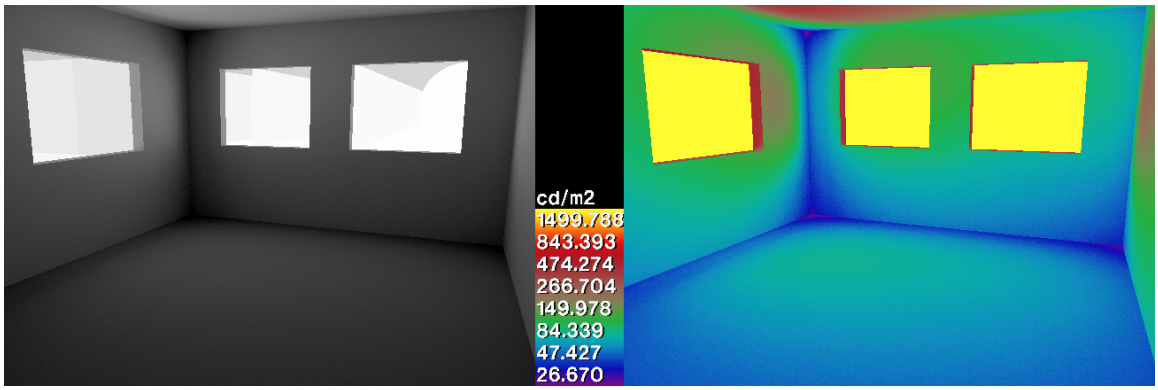


Figure 28. Rendering for 1 PM May 21st, Venetian blinds on the south windows with 0° tilt.

Then the same again for the early morning:

```
$ pcomb '!dctimestep images/vmx/window_east_%03d.hdr data/singleclear.xml \
            results/east.dmx skies/5_21_07.skv' \
      '!dctimestep images/vmx/window_south_%03d.hdr data/singleclear.xml \
            results/south.dmx skies/5_21_07.skv' \
      > images/5_21_07_clear-clear.hdr

$ pcond images/5_21_07_clear-clear.hdr | \
           pcompos -a 2 - '!falsecolor -s 2000 -log 2 -i images/5_21_07_clear-clear.hdr ' | \
           ra_tiff -z - images/5_21_07_clear-clear.tif

$ pcomb '!dctimestep images/vmx/window_east_%03d.hdr data/venetian45.xml \
            results/east.dmx skies/5_21_07.skv' \
      '!dctimestep images/vmx/window_south_%03d.hdr data/singleclear.xml \
            results/south.dmx skies/5_21_07.skv' \
      > images/5_21_07_vb45-clear.hdr

$ pcond images/5_21_07_vb45-clear.hdr | \
           pcompos -a 2 - '!falsecolor -s 2000 -log 2 -i images/5_21_07_vb45-clear.hdr ' | \
           ra_tiff -z - images/5_21_07_vb45-clear.tif
```

We can see how the shading affects the sunpatch from the east window but doesn't affect the south windows.

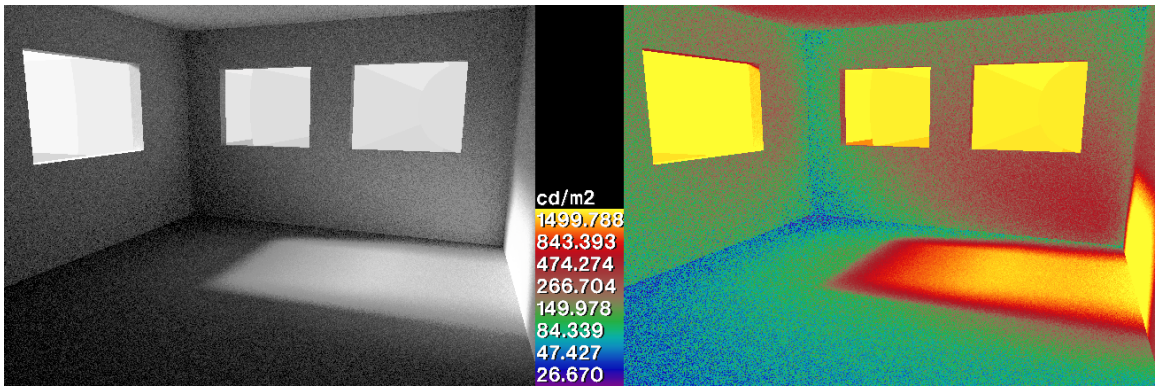


Figure 29. Rendering for 7:00 AM May 21st, no blinds.

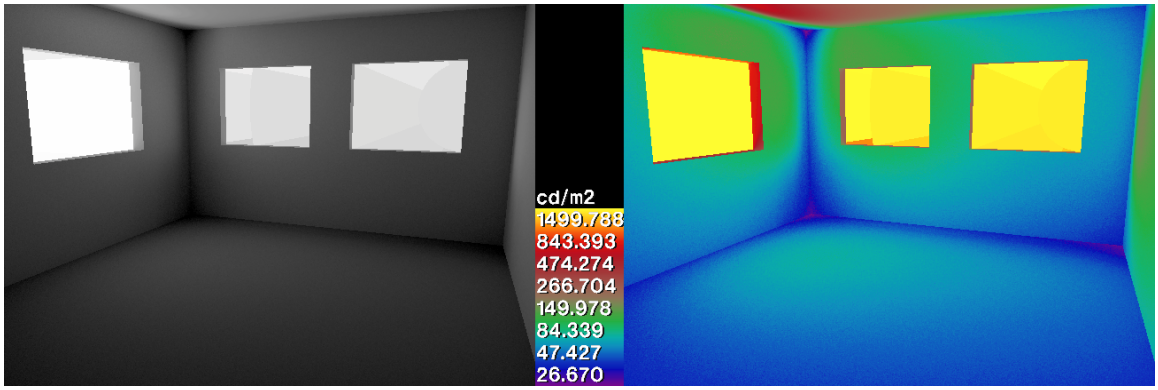


Figure 30. Rendering for 7:00 AM May 21st, Venetian blinds on the east windows with 45° tilt.

We'll calculate illuminance at the sensor points using rlam and rcalc with dctimestep commands in line. First without shading:

```
$ rlam '!dctimestep results/photocells_window_east.vmx data/singleclear.xml \
            results/east.dmx skies/5_21_13.skv' \
      '!dctimestep results/photocells_window_south.vmx data/singleclear.xml \
            results/south.dmx skies/5_21_13.skv' | \
      rcalc -e '$1=179*(($1+$4)*0.265+($2+$5)*0.670+($3+$6)*0.065)'  \
      >  results/illum_052113_clear-clear.dat
```

And then with blinds on the south window:

```
$ rlam '!dctimestep results/photocells_window_east.vmx data/singleclear.xml \
            results/east.dmx skies/5_21_13.skv' \
      '!dctimestep results/photocells_window_south.vmx data/venetian0.xml \
            results/south.dmx skies/5_21_13.skv' | \
      rcalc -e '$1=179*(($1+$4)*0.265+($2+$5)*0.670+($3+$6)*0.065)'  \
      >  results/illum_052113_clear-vb0.dat
```

And we can plot the illuminance in the space for the two conditions:

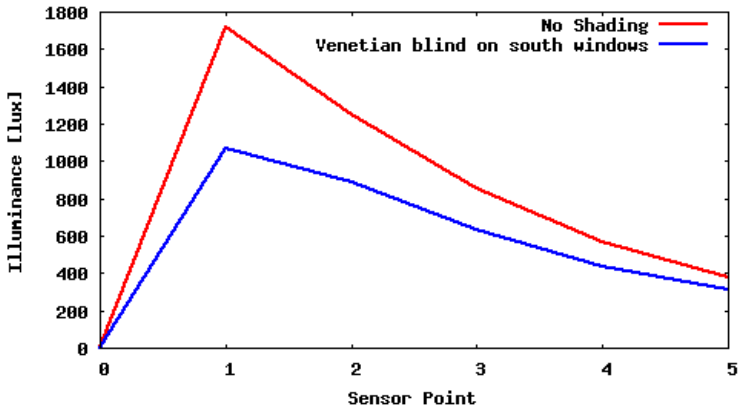


Figure 31. Plot of illuminance at sensor points for two shading conditions.

5 Acknowledgements

Thanks to Axel Jacobs and Greg Ward for their contributions to this tutorial.

This work was supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Building Technology, State and Community Programs, Office of Building Research and Standards of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

6 Appendix - Additional Considerations

This Appendix describes additional considerations for determining when to group windows and when to subdivide windows.

6.1 Window grouping and subdivision for exterior obstructions

The **D** matrix contains daylight coefficient contributions either for a single point on the facade or integrated over an area of the facade. A simulation may need to be split into many parts with more than one **D** matrix to reproduce non-uniform shading effects caused by external obstructions (Fig. 30). The number of **D** matrices required depends on the proximity and size of external obstructions.

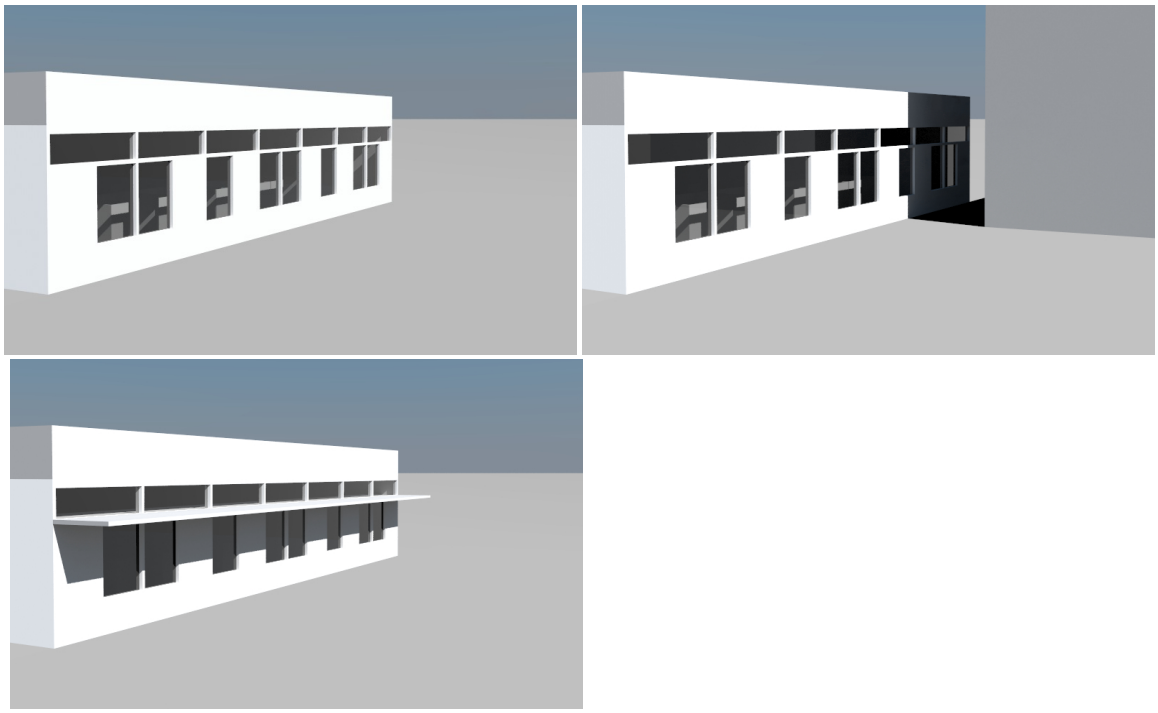


Figure 30. Renderings of a facade containing daylight and view glazing with (a) no external obstructions, (b) a large local obstruction and (c) a building attached obstruction (overhang).

Computing separate exterior daylight matrix for window sub-groups allows for localization of exterior obstruction effects providing a more accurate value for incident flux on the window. If there are no external obstructions affecting daylight in a building (Fig. 30a), then a single daylight matrix can be computed for all windows of the same orientation (Fig. 31).

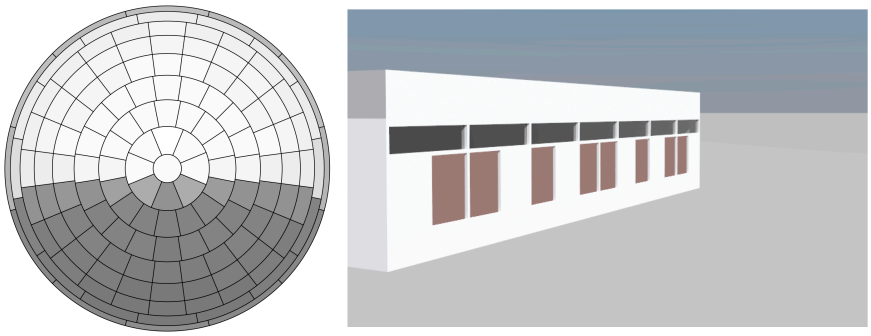


Figure 31. A visualization of an incident daylight matrix (cumulative coefficients) for all of the view windows (shaded pink)

If a nearby obstruction provides non-uniform shading on a façade, then computing a single daylight matrix for the windows on the façade averages the shading over all the windows (Figure 24). Instead, daylight matrices should be computed separately for windows with drastically different shading characteristics in order to reproduce localized interior effects of external obstructions (Figure 32).

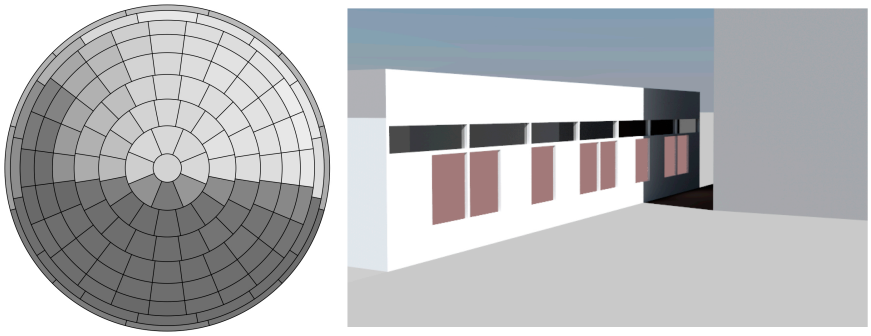


Figure 32. A visualization of an incident daylight matrix (cumulative coefficients) for all of view windows (shaded pink)

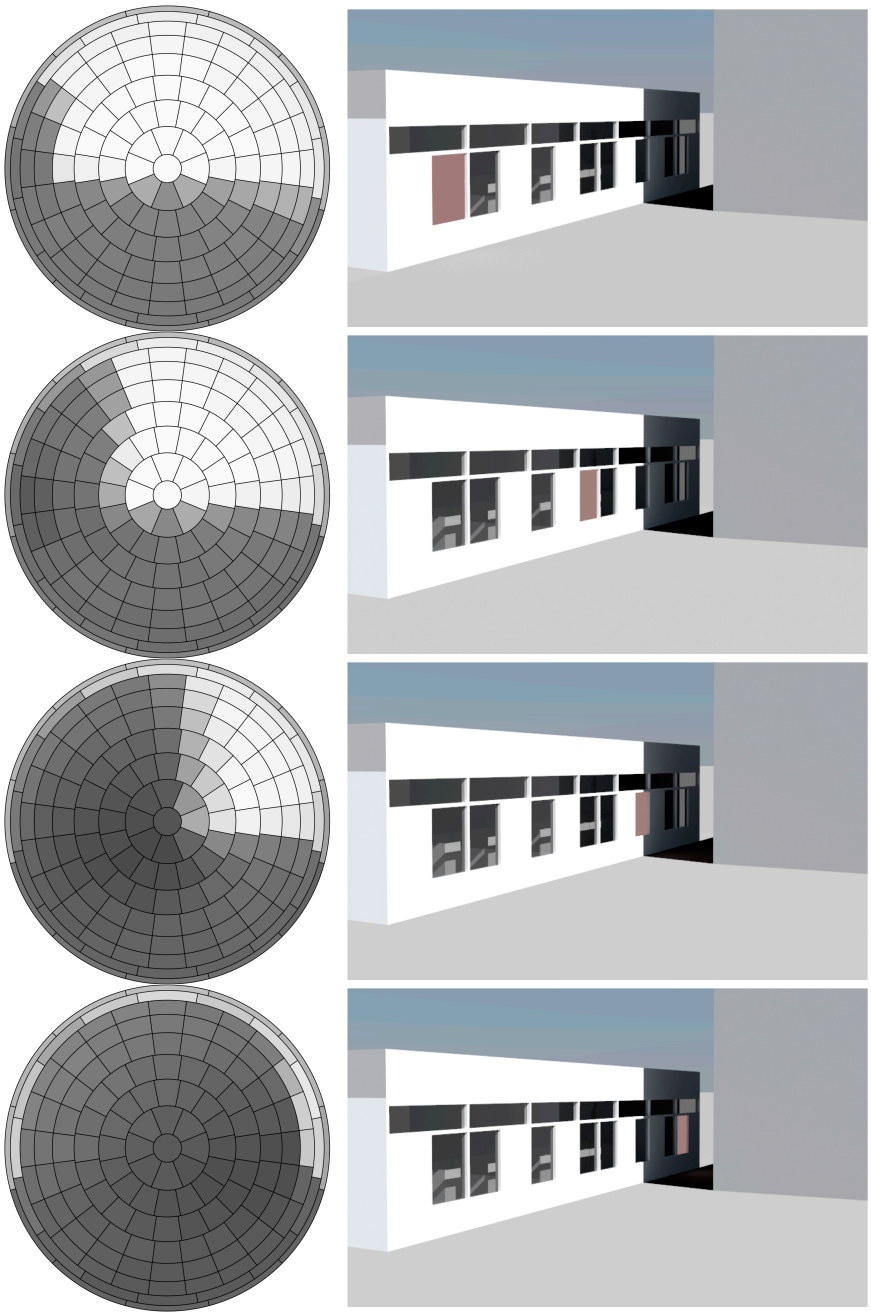


Figure 33. Visualizations of an incident daylight matrix (cumulative coefficients) for individual windows (shaded pink)

Building attached obstructions may require subdivision of individual windows. For example, an overhang above a window provides more shading for the top of the window than the bottom. Computing the daylight matrix for the whole window averages the effect of the overhang over the entire window (Figure 33). Subdividing the window into horizontal bands reproduces the variation in shading over the height of the window provided by the overhang (Figure 34).

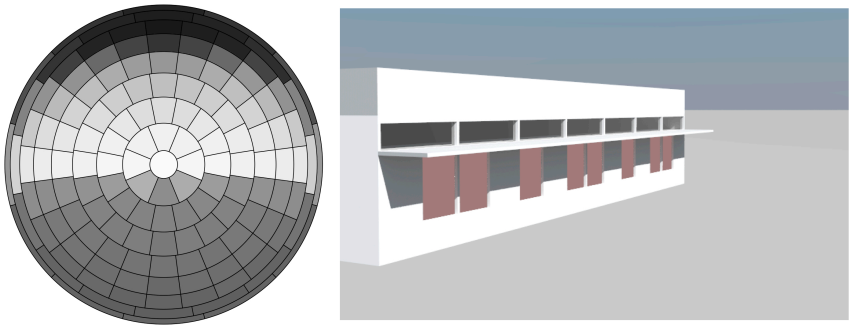


Figure 33. Visualization of an incident daylight matrix (cumulative coefficients) for all of view windows (shaded pink)

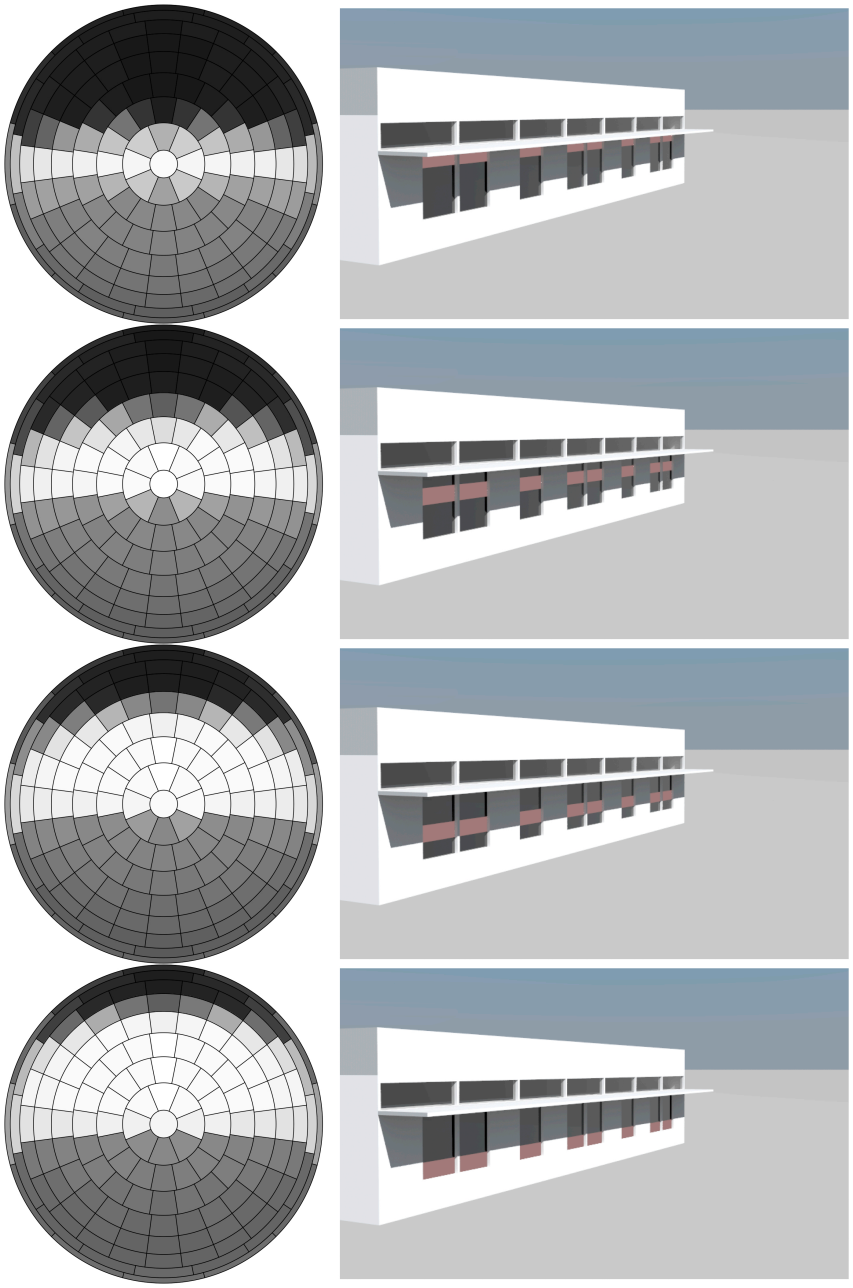


Figure 34. Visualizations of an incident daylight matrix (cumulative coefficients) for one-quarter subdivisions of the view windows

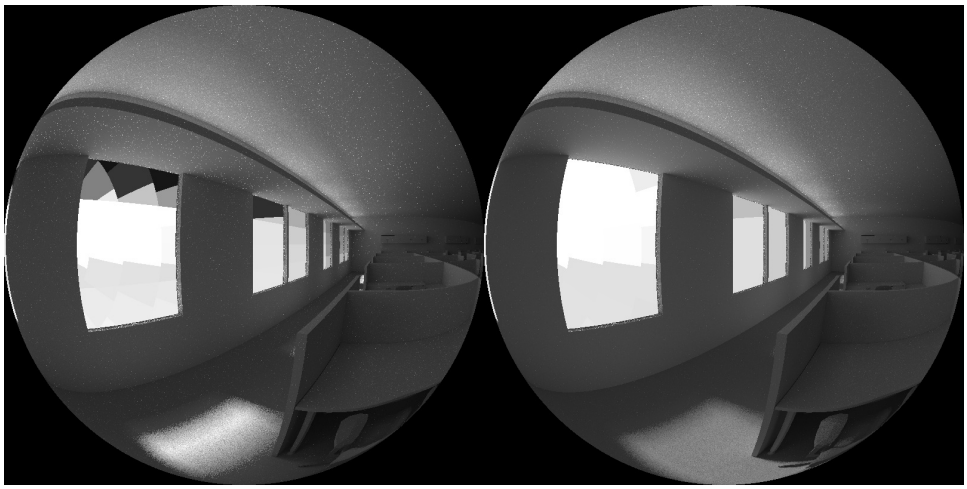


Figure 35. Two interior renderings with an overhang. Windows in the left image are subdivided into four bands. Windows in the right image are not subdivided. The effect of the overhang is averaged over the entire window on the right. The effect of the overhang is considered for the four window subdivisions in the left image. Note the size and brightness of sun patch on the floor and also the shaded appearance of the window on the left.

6.2 Window subdivision for variable height shading systems (blinds & shades)

Window groupings and subdivisions used for the **V** matrix should mirror those used for the **D** matrix. For some cases (i.e., variable height shading systems), it is desirable to divide the windows further for the **V** matrix computation. Further division for V matrix computation is permissible as long as **D** matrix divisions/groups are reflected in **V** matrix subdivisions/subgroups.

Variable-height shading systems require two BSDFs to represent optical properties above and below the bottom of the shade. Examples of variable-height shading systems include venetian blinds and roller shades. The use of two separate BTDFs for the upper and lower portion of the window requires dividing the window geometry into two polygons.

Commonly, the blind position is typically unknown before simulation. For example, stochastic models used to model user behavior will define shade position for each time step. To account for variable position shades, the window should be subdivided into many horizontal bands. These subdivided bands provide discrete potential shade positions for simulation. For example, a window divided into 4 equal bands allows the simulation of discrete shade height positions in 25% increments (Fig. 36).

Figure 36. rcontrib three-phase renderings with venetian blinds covering 0%, 25%, 50%, 75% and 100% of the windows. Other than the darkening of the parts of the window covered by blinds window there is no visual representation of the blinds in these renderings. Despite the lack of physical geometry, the effect of the blinds on daylight in the space is present. Most notably the reduction of the sun patch on the floor as the blind is lowered.