The Five-Phase Method for Simulating Complex Fenestration with Radiance

Andy McNeil, LBNL

Revision History

26 August 2013	Version 2.0 - public release
9 September 2013	Version 2.1 - fixed BSDF thickness, made objects/* explicit in an oconv command and replaced tr/awk with the new rcollate utility.
11 September 2013	Version 2.2 - combined two rcollate commands into one (section 6.8)

1 Introduction

The "five-phase method" is an extension of the three-phase method that more closely follows the standard daylight coefficient model for dynamic daylight simulations proposed by Bourgeois et al (2008). More specifically, the five-phase method handles the direct solar component separately from the sky and inter-reflected solar component to achieve better accuracy of the distribution of direct solar light in a room for complex glazing systems (CFS).

The equation we used for the 3-phase method (equation 1) is adapted for the 5-phase method (equation 2)

$$I_{3ph} = VTDS \quad (1)$$

$$I_{5ph} = VTDS - V_dTD_dS_{ds} + C_{ds}S_{sun} \quad (2)$$

Where:

I	=	Result matrix containing time series of illuminance or luminance result
V	=	View matrix, relating outgoing directions on window to desired results at interior
V_d	=	Direct only view matrix
T	=	Transmission matrix, relating incident window directions to exiting directions (BSDF)
D	=	Daylight matrix, relating sky patches to incident directions on window
D_d	=	Direct only daylight matrix
C_{ds}	=	Coefficient matrix for direct sun relating radiance of many sun positions to direct illuminance at a sensor point using a BSDF with proxy geometry or a variable resolution BSDF material.
S	=	Sky matrix, a collection of sky vectors
S_{ds}	=	Sky matrix containing only the sun luminance (no sky luminance)
S_{sun}	=	Direct sun matrix containing the radiance and position of the sun

The basic approach of the 5-phase method is as follows:

1. Perform a three-phase simulation
2. Subtract the direct solar contribution (leaving the inter-reflected solar component)
3. Add direct solar contribution that is more accurately simulated.

The following figures contain rendered illustrations of the terms of the 5-phase equation for a specular glazing (Figure 1) window, and for a window with redirecting system above and venetian blind below (Figure 2).

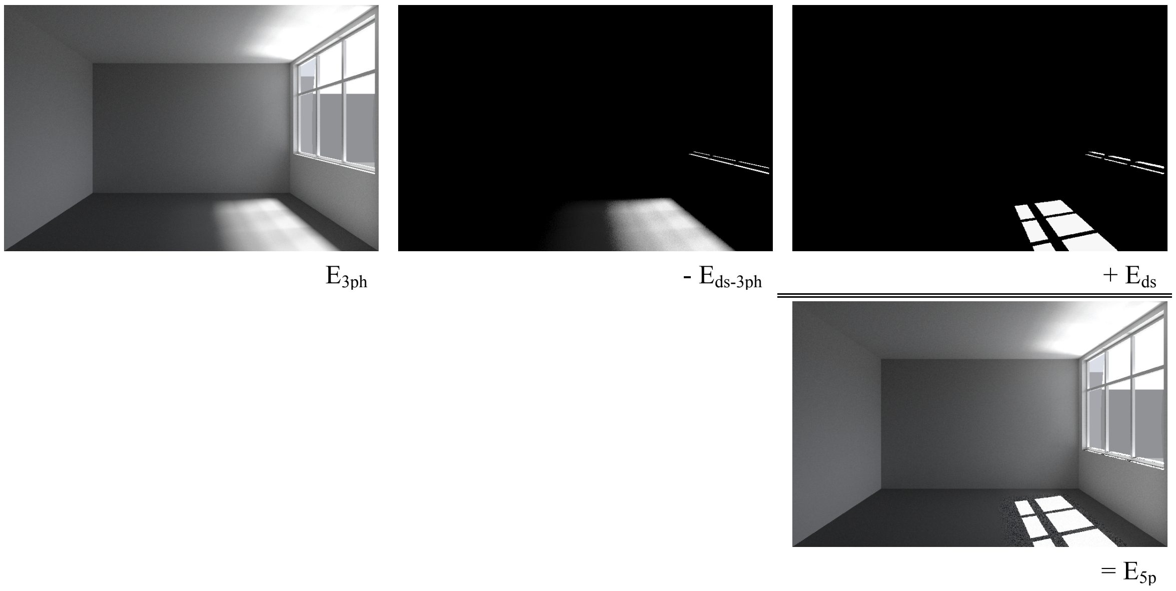


Figure 1 - Renderings of the luminance contributions that are used in the five-phase for a space with clear glazing.

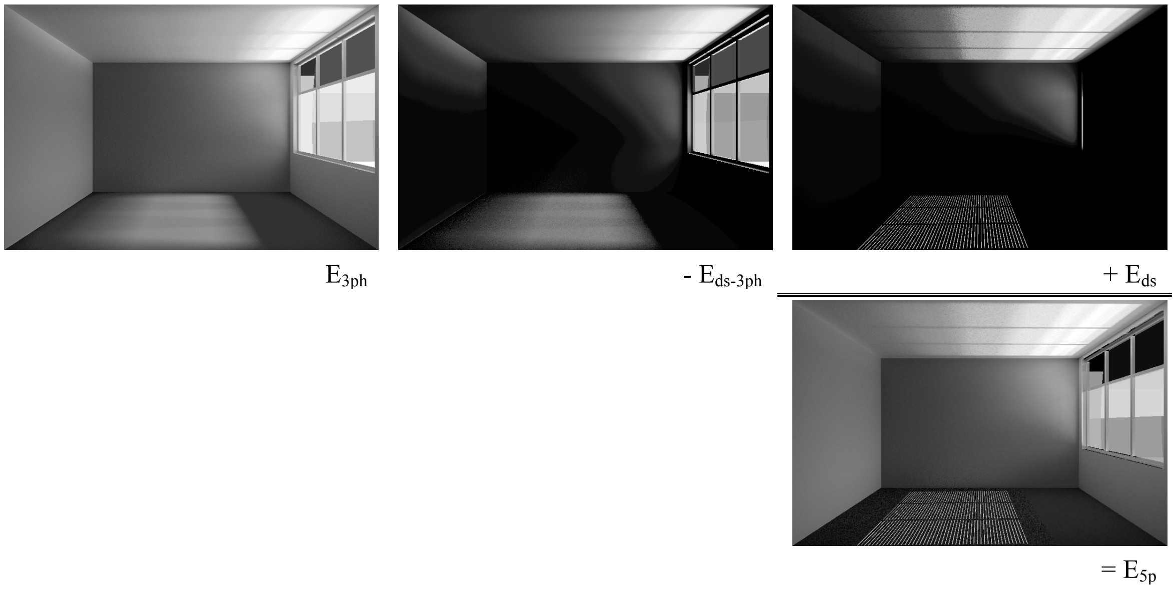


Figure 2 - Renderings of the luminance contributions that are used in the five-phase for a space with daylight redirecting system in the upper windows and venetian blinds in the lower windows.

1.1 Tutorial Prerequisites

This tutorial builds on two previous tutorials for annual simulation with Radiance. Users are encouraged to complete the following more basic tutorials before attempting the five-phase method tutorial:

Tutorial	**Author**	**Link**
Understanding rtcontrib (rtcontrib lesson)	Axel Jacobs	http://www.jaloxa.eu/resources/radiance/documentation/
The three-phase method for simulating complex fenestration	Andy McNeil	http://www.radiance-online.org/learning/tutorials/Tutorial-ThreePhaseMethod.pdf

1.2 Contents of this document

Each of the next three sections of the tutorial contains a detailed discussion of a term of the five-phase equation. Then there are two example simulations with step-by-step details.

1. Introduction (this!)
2. **VTDS**: The normal three-phase calculation
3. **$V_dTD_dS_{ds}$**: Isolating the three-phase direct sun contribution
4. **$C_{ds}S_{sun}$**: Calculating the direct sun contribution
5. Combining the components
6. Example 1: Illuminance simulation of a space with venetian blinds
7. Example 2: Rendering a space with venetian blinds
8. Simulation Duration

2 VTDS - The typical three-phase calculation

The VTDS term is the same as a typical three-phase calculation, which is detailed in the three-phase tutorial: http://www.radiance-online.org/learning/tutorials/Tutorial-ThreePhaseMethod.pdf.

However, subsequent to the creation of the three-phase tutorial the following parameters have been determined to provide reliably accurate results for perimeter office spaces (McNeil & Lee 2013).

View matrix simulation parameters: -ab 10 -ad 65536 -lw 1.52e-5
Daylight matrix simulation parameters: -ab 2 -ad 1024 -c 1000

3 $V_dTD_dS_{ds}$ - Isolating the three-phase direct sun contribution

The $V_dTD_dS_{ds}$ term isolates the direct sun contribution of the three-phase calculation. We need to isolate the direct sun so that it can be subtracted from the result prior to adding a more accurate direct sun contribution.

3.1 S_{ds} – Direct sun matrix

S_{ds} is a sky matrix that contains only the direct solar radiance. It is created using the -d parameter of gendaymtx or genskyvec. Figure 3 illustrates the difference between a typical sky vector and the direct sun only sky vector.

```
$ epw2wea Chicago.epw Chicago.wea
$ gendaymtx -m 1 -d Chicago.wea > Chicago_direct_m1.smx
```

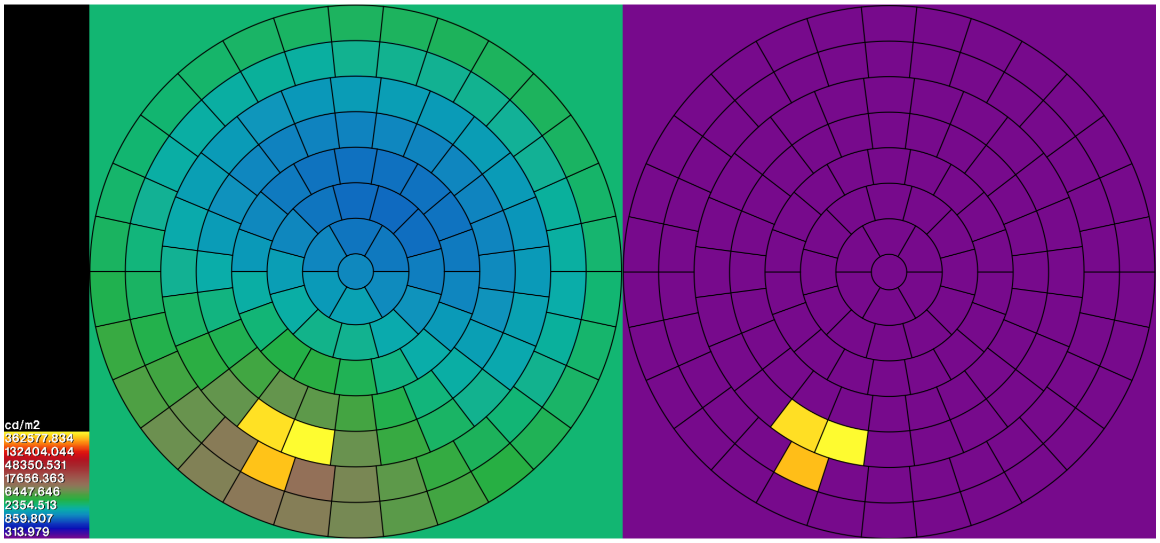


Figure 3 - Renderings of sky patch luminance for sky+sun (left) and sun only (right).

3.2 D_d – Direct daylight matrix

The direct daylight contribution matrix contains characterizes the amount of light from the sky that reaches the window without reflection. We can use -ab 0 even thought we have a sky with glow material because we are using genklemsamp to send sample rays from the window into the scene.

While setting ambient bounces to zero avoids most inter-reflections, if a ray hits a surface with specularity greater than zero and roughness less than 0.002 a mirror reflection ray will be sent (even with -ab 0 and -st 1) that might then strike the sky.

In order to avoid mirror specular reflections, we can create a new material file with specularity of all materials to be set to zero, however later we will need to change all materials to black for similar reasons (see section 4.1) so for simplicity we will change materials to black here too. The -m option of xform is the simplest way to change all materials in a model. First we add a new material to the material file (that won't normally be used):

```
### materials.rad
void plastic black
0
0
5   0  0  0  0  0
```

Then we use xform to change all objects except the sky and window glow materials to this new material (and use oconv to create an octree):

```
$ xform -m black model.rad | oconv materials.rad - win_glow.rad sky.rad > model_allblack.oct
```

Then the daylight matrix is generated using the all black model in the typical manner:

```
$ genklemsamp -vd 0 -1 0 win_glow.rad | \
            rcontrib -c 1000 –ab 0 -e MF:1 -f reinhart.cal -b rbin -bn Nrbins \
                        -m skymat -faf model_allblack.oct > south_direct.dmx
```

3.3 V_d – Direct view matrix

The direct view matrix contains light that goes directly from the window to a sensor point or, in the case of a rendering, a surface. We can use the same octree created for the three-phase view matrix.

Direct view matrix for sensor points:

```
$ rcontrib < test.pts -f klems_int.cal -b kbinS -bn Nkbins -m window_mat \
            -I+ -ab 1 -ad 50000 -lw 2e-5 model.oct > south_direct.vmx
```

Direct view matrix for a rendering:

```
$ vwrays -ff -vf view.vf -x 500 -y 500 \
            | rcontrib `vwrays -vf view.vf -x 500 -y 500 -d` -ffc -fo -o V_direct_%03d.hdr \
                        -f klems_int.cal -b kbinS -bn Nkbins -m window_mat \
                        -ab 1 -ad 65536 -lw 1.52e-5 model.oct
```

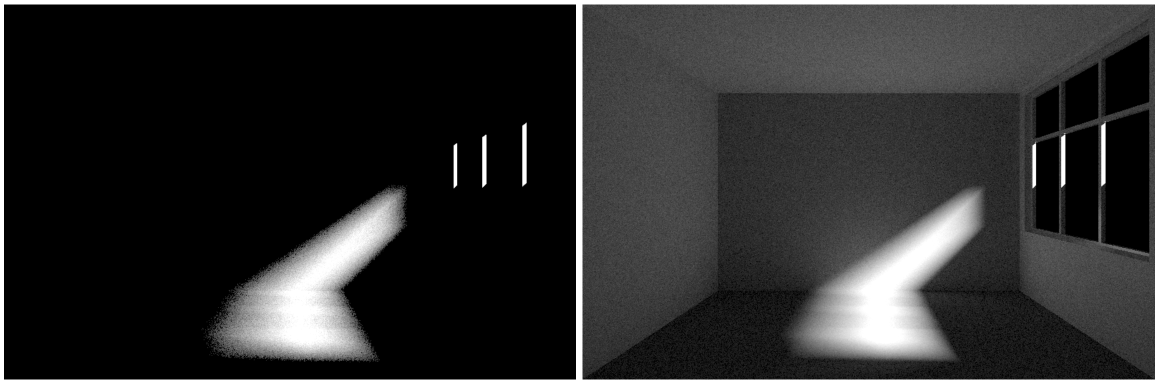


Figure 4 - Images from rendered view matrices. Direct view matrix (left) compared to the usual view matrix (right).

3.4 *Multiplying $V_dTD_dS_{ds}$ with dctimestep*

We can generate the $V_dTD_dS_{ds}$ term for sensor point view matrices using dctimestep as follows:

```
$ dctimestep -n 8760 -if  south_direct.vmx  window.xml  south_direct.dmx  Chicago.smx \
            > i_ds3ph.dat
```

or for rendered view matrices as follows:

```
$ dctimestep -n 8760 -o I_ds3ph_%04d.hdr \
            V_direct_%03d.hdr  window.xml  south_direct.dmx  Chicago.smx
```

4 $C_{ds}S_{sun}$ - Calculating the direct sun contribution

4.1 *Creating C_{ds}*

This run uses a sky with many suns in the sky. For now, the following bash commands generate suns in the center of each reinhart MF:6 sky patches (a simple radiance utility could be created to simplify this step).

```
$ echo void light solar 0 0 3 1e6 1e6 1e6 > suns.rad
$ cnt 5185 | rcalc -e MF:6 -f reinsrc.cal \
            -e Rbin=recno -o 'solar source sun 0 0 4 ${ Dx } ${ Dy } ${ Dz } 0.533' >> suns.rad
```

Figure 5 is an angular rendering of the suns looking straight up. There are 5185 suns in this rendering, some near the zenith overlap, but Radiance's direct source sampling can accommodate overlapping light sources.

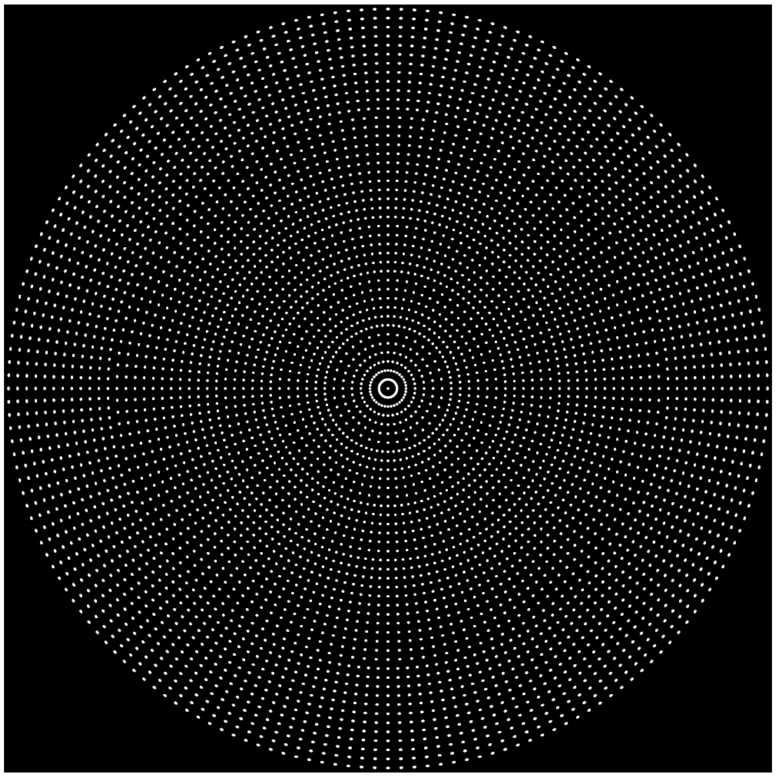


Figure 5 - Angular fisheye rendering (looking up) of the sun positions for a reinhart MF:6 sky.

We can't simply use –ab 0 because in cases with a BSDF material we need to sample the BSDF material from all directions, which can only be done with ambient sample rays. We use –ab 1 to generate ambient samples, however we don't want ambient sampling to include reflected sunlight, so we need to use an all black model. We can use the same xform -m method used previously.

The direct sun contribution is calculated using a BSDF (high-resolution tensor tree BSDF is preferred) with or without proxied geometry. All other materials should be black, and we'll again use xform to change the material of the non-window geometry.

We then create an octree using the black geometry, window geometry and suns. The file winBSDF.rad contains a radiance description of a window with a tensor tree BSDF proxy.

```
$ xform -m black model.rad | oconv materials.rad - winBSDF.rad suns_m6.rad > model_suns.oct
```

Then the direct sunlight matrix is calculated using rcontrib and reinhart.cal to bin the coefficients as follows:

```
$ rcontrib < test.pts -I -ab 1 -ad 50000 -lw 2e-5 -dc 1 -dt 0 -dj 0 -st 1 -ss 0 -faf  \
          -e MF:6 -f reinhart.cal -b rbin -bn Nrbins -m solar model_suns.oct > directsun.dsmx
```

Calculating the direct view matrix for a rendering is a bit more complex. We need all the surfaces to be black to prevent indirect contributions from direct sources visible through proxied geometry, however a typical luminance rendering would be all black. Instead we create an illuminance rendering with the all black model and multiply the illuminance result by the reflectance divided by pi for each pixel. This approach ignores specular reflection from a surface to the viewpoint.

First the illuminance rendering is created as follows:

```
$ vwrays -ff -vf view.vf -x 500 -y 500 \
        | rcontrib `vwrays -vf view.vf -x 500 -y 500 -d` -i -ffc -fo -o Vi_directsun_%04d.hdr \
                -ab 1 -ad 1000 -lw 1e-3 -dc 1 -dt 0 \
                -e MF:6 -f reinhart.cal -b rbin -bn Nrbins -m solar model_suns.oct
```

Then we create a rendering with each pixel value equal to the material reflectance divided by pi:

```
$ rpict -x 500 -y 500 -vf view.vf -av 0.31831 0.31831 0.31831 -aa 0 model.oct > materialmap.hdr
```

4.2 Creating S_{sun}

The sun matrix file is created using gendaymtx. Gendaymtx has the currently undocumented option ‘-5’ for the 5-phase method direct sun matrix generation. This option creates a direct only sky with all the sun’s energy in a single patch. Additionally, the energy is adjusted to suit the solid angle of the sun instead of a full sky patch (since suns are used in the simulation). Figure 6 shows the position of the sun and the position used for the five-phase calculation. For this example the actual sun position is 1.153° from the sun position used in the five-phase calculation.

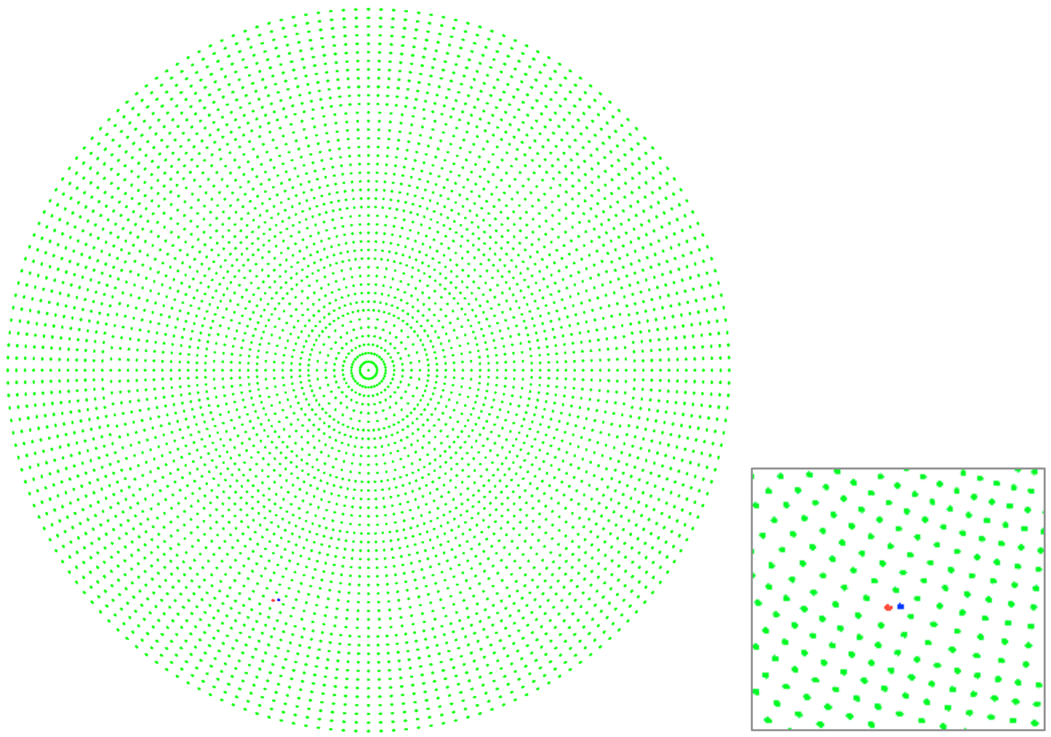


Figure 6 - A rendering of the sun positions (green) with the actual sun position for a timestep (blue) and the sun position used in the 5-phase calculation (red). A zoomed view of sun position is shown to the right.

The direct sun matrix is created as follows:

```
$ gendaymtx -m 6 -5 -of Chicago.wea > Chicago_direct_m6.smx
```

4.3 Multiplying $C_{ds}S_{sun}$ with dctimestep

To generate the $C_{ds}S_{sun}$ component, we use dctimestep.

```
$ dctimestep -n 8760 -if directsun.dsmx Chicago_direct_m6.smx > i_ds5ph.dat
```

Or for a rendering:

```
$ dctimestep -n 8760 -o I_ds5ph_%04d.hdr  V_directsun_%04d.hdr Chicago_direct_m6.smx
$ for i in {1..8760} ; do  ts=`printf "%04d" $i`
        pcomb –e 'lo=li(1)*li(2)' Vi_directsun_${ts}.hdr materialmap.hdr > V_directsun_${ts}.hdr ;
done
```

5 Combining the components

First I'll remind you of the five-phase equation (which is presented in the introduction):

$$I_{5ph} = VTDS - V_dTD_dS_{ds} + C_{ds}S_{sun}$$

The first term is the usual three-phase result, the second term is the direct only component calculated using the three-phase method (so that it can be sub-tracted) and the third term is the five-phase direct sun result (using high-resolution BSDF and proxied geometry). The method of combining these two depends on the type of result. For an illuminance sensor result we use rcalc and for a rendered result we use pcomb.

5.1 Illuminance sensor simulation

You might have noticed previously that we were using binary output (-of from dctimestep). The reason for that is to make this step easier. With binary data, we can use rcalc in a to add the data easily.

```
$ rlam -if3 i_3ph.dat i_ds3ph.dat i_ds5ph.dat | \
        rcalc -if9 -e 'r=$1-$4+$7;g=$2-$5+$8;b=$3-$6+$9' \
        -e '$1=179*(.265*r+.670*g+.065*b)' | \
        awk '{printf("%f\t",$1);if(NR%8760==0) printf("\n")}' > illum.txt
```

The resulting file has a row for each sensor point and a column for each hour.

5.2 Renderings

For five-phase simulation with rendered images, we need to use pcomb:

```
$ for j in {0..8760} ;  do  $k=`printf %04d $j`
            pcomb -s 1 V_3ph_${k}.hdr -s -1 V_direct_${k}.hdr  -s 1 V_directsun_${k}.hdr > image_${k}.hdr
done
```

6 Example 1 – Illuminance Calculations

We're going to use the model that Axel Jacobs created as part of rtcontrib lesson, which can be obtained here: http://www.jaloxa.eu/resources/radiance/documentation/.

6.1 Model adjustments for five-phase simulation

Axel's model has no fenestration system so we'll create a files called 'glazing.rad' and 'venetianblind.rad' in the objects directory. The following is added to the glazing.rad file:

```
#objects/glazing.rad
void glass glazing
0
0
3 0.73 0.73 0.73

glazing polygon glass
0
0
12    0.5  -0.25   1
      0.5  -0.25   2
      3.5  -0.25   2
      3.5  -0.25   1
```

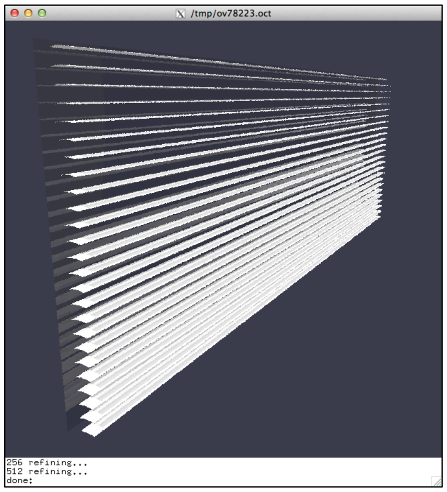
And a venetian blind:

```
# objects/venetianblind.rad
void plastic blindslat
0
0
5 .65 .65 .65 .01 0

!genblinds blindslat blinds .05 3 1 25 0 +r .15 | xform -rz -90 -t 0.5 -.15 1
```

This contains a glass surface and set of venetian blinds that fit into the window of the model. We can look at the geometry using objview:

```
$ objview objects/venetianblind.rad objects/glazing.rad
```



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

Figure 7 - objview of fenestration.rad (after adjusting the view parameters)

Another consideration is whether or not to include the window reveal in the BSDF or in a combination of the view and daylight matrix. For a three-phase simulation, either is suitable, and may depend on the source of the BSDF. If the BSDF is generated using LBNL Window, then the BSDF properties are for an infinite layer, and the edge conditions should be included in the combination view and daylight matrix. However if the BSDF is generated using genBSDF, then the properties are for a finite system, and to prevent leaking light at the edges, we want to include edge conditions in the BSDF model. Further with a five-phase calculation we are using the BSDF in the direct sun calculation, so if we have the proxied geometry, then we are using a BSDF with thickness so we should include the edge condition in the BSDF. But if we are using the BSDF without proxied geometry and zero thickness then it is preferable to have the edge condition in the view and daylight matrices.

We will generate the BSDF with genBSDF in this example, so we will include the edge condition in the BSDF file. To do this we will exclude the south wall from the testroom.rad file and include it in a file called testroom_Swall.rad like so:

```
# objects/testroom_Swall.rad
!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_west .5 .3 1 |xform -t 3.5 -.3 1
```

The file testroom.rad is reduced to the following:

```
# objects/testroom.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_e .3 6.6 2.5 |xform -t 4 -.3 0
!genbox wall_mat wall_n 4 .3 2.5 |xform -t 0 6 0
```

Finally, we need to create the surfaces used to generate the view and daylight matrices. Since the window reveal will be included in the BSDF file, we want to put the glow surface for the view matrix on the inside face of the window opening. For the daylight matrix we'll put the surface passed to genklemsamp on the outside of the window opening.

```
#objects/viewmtxsurf.rad
void glow viewsurf
0
0
4 1 1 1 0

viewsurf polygon inside
0
0
12    0.5   0   1
      0.5   0   2
      3.5   0   2
      3.5   0   1
```

```
#objects/daymtxsurf.rad
void plastic daymtxsurf
0
0
5 0 0 0 0 0

daymtxsurf polygon outside
0
0
12    0.5   -0.3   1
      0.5   -0.3   2
      3.5   -0.3   2
      3.5   -0.3  1
```

And I'll need to make a BSDF proxy surface in the (which is the same polygon as the view matrix surface):

```
#objects/glazing_bsdf.rad
void BSDF BSDFproxy
6 0.30 bsdf/fullwindow_t45.xml 0 0 1 .
0
0

BSDFproxy polygon inside
0
0
12    0.5   0   1
      0.5   0   2
      3.5   0   2
      3.5   0   1
```

Finally add the material ‘black’ to the materials file (materials/testroom.mat):

```
void plastic black
0
0
5 0 0 0 0 0
```

6.2 Creating BSDFs

We’ll use two BSDF files, a Klems basis BSDF for the 3-phase part of the simulation and a tensor tree BSDF for the direct sun simulation.

First we need to orient the geometry properly for genBSDF using xform. The plane of the window should be parallel to the xy plane, so we rotate the geometry about the y axis. The geometry should lie in the negative z half space, so we translate the geometry -0.3 units in the z-direction. The xform command looks like this:

```
$ xform -rz 180 -rx -90 -t 4 0 0 objects/glazing.rad objects/venetianblind.rad materials/testroom.mat \
        objects/testroom_Swall.rad > bsdf/fullwindow.rad
```

Then we use genBSDR to generate a tensor tree BSDF. We’re not going to include geometry in the BSDF since in this example it is easy to handle the fenestration geometry separately. Excluding the geometry is achieved with ‘-geom meters’ (you neeed to include model units with –geom or +geom). We’ll use the –dim command to specify the extents of the window for BSDF sampling.

```
$ genBSDF +f +b -geom meter -dim 0.5 3.5 1 2 -.3 0 -t4 5 \
        bsdf/fullwindow.rad > bsdf/fullwindow_t45.xml
```

We can use bsdf2klems to resample the tensor tree BSDF created by genBSDF converting it to a Klems basis BSDF. However this method doesn’t work well when re-sampling BSDFs with a specular component, so instead we’ll run genBSDF again to create a klems resolution BSDF.

```
$ bsdf2klems bsdf/fullwindow_t45.xml > bsdf/fullwindow_klems.xml
$ genBSDF +f +b -geom meter -dim 0.5 3.5 1 2 -.3 0 \
        bsdf/fullwindow.rad > bsdf/fullwindow_klems.xml
```

6.3 View matrix (V)

We generate the view matrix as described in the three-phase simulation tutorial. First generating the octree, then running rcontrib:

```
$ oconv materials/testroom.mat objects/daymtxsurf.rad objects/ground.rad objects/testroom.rad \
        objects/testroom_Swall.rad objects/viewmtxsurf.rad skies/sky_white.rad \
        objects/viewmtxsurf.rad objects/daymtxsurf.rad > octs/model_3ph.oct
$ rcontrib < data/photocells.pts -f klems_int.cal -b kbinS -bn Nkbins -m viewsurf \
        -I+ -ab 10 -ad 65536 -lw 1.52e-5 octs/model_3ph.oct > matrices/viewmatrix.vmx
```

6.4 Direct view matrix (V_d)

The direct view matrix is generated the same way as the view matrix except with only one ambient bounce.

```
$ rcontrib < data/photocells.pts -f klems_int.cal -b kbinS -bn Nkbins -m viewsurf \
        -I+ -ab 1 -ad 65536 -lw 1.52e-5 octs/model_3ph.oct > matrices/viewmatrix_direct.vmx
```

6.5 Daylight matrix (D)

The daylight matrix is generated as described in the three-phase simulation tutorial:

```
$ genklemsamp -c 1000 -vd 0 -1 0 objects/daymtxsurf.rad | \
        rcontrib -c 1000 -ab 2 -ad 1024 -e MF:1 -f reinhart.cal -b rbin -bn Nrbins \
                -m sky_glow octs/model_3ph.oct > matrices/daylightmatrix.dmx
```

6.6 Direct daylight matrix (D_d)

For the direct daylight matrix we'll need an all-black model:

```
$ xform -m black objects/testroom_Swall.rad objects/testroom.rad objects/ground.rad \
        objects/venetianblind.rad  |  oconv materials/testroom.mat - objects/viewmtxsurf.rad \
        objects/daymtxsurf.rad skies/sky_white.rad > octs/model_allblack.oct
```

Then the direct daylight matrix is created using the all black model and zero ambient bounces:

```
$ genklemsamp -c 1000 -vd 0 -1 0 objects/daymtxsurf.rad | \
        rcontrib -c 1000 -ab 0 -e MF:1 -f reinhart.cal -b rbin -bn Nrbins \
                -m sky_glow octs/model_allblack.oct > matrices/daylightmatrix_direct.dmx
```

6.7 Direct sun coefficient matrix (C_{ds})

First we'll create the radiance scene description for 9217 suns:

```
$ echo void light solar 0 0 3 1e6 1e6 1e6 > skies/suns.rad
$ cnt 5185 | rcalc -e MF:6 -f reinsrc.cal -e Rbin=recno \
        -o 'solar source sun 0 0 4 ${ Dx } ${ Dy } ${ Dz } 0.533' >> skies/suns.rad
```

Then we create an all-black octree containing the suns and high-resolution BSDF material with proxied geometry.

```
$ xform -m black objects/testroom_Swall.rad objects/testroom.rad objects/ground.rad \
        | oconv materials/testroom.mat - objects/glazing.rad objects/venetianblind.rad \
                objects/glazing_bsdf.rad skies/suns.rad > octs/model_suns.oct
```

Finally we create the direct sun coefficient matrix.

```
$ rcontrib < data/photocells.pts -I -ab 1 -ad 65536 -lw 1.52e-5  -dc 1 -dt 0 -dj 0 -st 1 -ss 0 -faf  \
        -e MF:6 -f reinhart.cal -b rbin -bn Nrbins -m solar \
        octs/model_suns.oct > matrices/directsun.dsmx
```

6.8 Putting it all together

For this simulation we'll download a weather data file for Oakland, California from the energy plus website:
http://apps1.eere.energy.gov/buildings/energyplus/weatherdata/4_north_and_central_america_wmo_region_4/1_usa/USA_CA_Oakland.Intl.AP.724930_TMY.zip

Then we convert to a wea file with epw2wea and then create a sky matrix file using gendaymtx.

```
$ epw2wea skies/USA_CA_Oakland.Intl.AP.724930_TMY3.epw skies/OakLand.wea
$ gendaymtx -of skies/OakLand.wea > matrices/OakLand.smx
$ gendaymtx -of -d skies/OakLand.wea > matrices/OakLand_direct.smx
$ gendaymtx -5 -d -m 6 -of skies/OakLand.wea > matrices/OakLand_direct_m6.smx
```

The first term of the 5-phase equation is the 3-phase simulation result, so the command to create the first term is familiar:

```
$ dctimestep -n 8760 -if matrices/viewmatrix.vmx bsdf/fullwindow.xml  \
          matrices/daylightmatrix.dmx matrices/OakLand.smx | \
          rcollate -h -oc 1 > i_3ph.txt
```

Next we'll create the direct sun contribution of the 3-phase method (the second term). Here we use dctimestep with the direct view matrix, direct daylight matrix and annual direct sky matrix:

```
$ dctimestep -n 8760 -if  matrices/viewmatrix_direct.vmx bsdf/fullwindow.xml  \
          matrices/daylightmatrix_direct.dmx matrices/OakLand_direct.smx  \
          rcollate -h -oc 1 > i_ds3ph.txt
```

And then comes the third term, (the direct sun contribution). Here we'll use dctimestep with the direct sun coefficient matrix and annual sun matrix:

```
$ dctimestep -n 8760 -if matrices/directsun.dsmx matrices/OakLand_direct_m6.smx \
          rcollate -h -oc 1 > i_ds5ph.txt
```

Finally, there's some tricky business required to get the final result. We use rlam to combine the three terms (each with R G and B values) onto a single line. With rcalc we combine the three terms of the five-phase equation for each of the R, G and B channels, then convert from RGB to luminance. The rcollate command reshapes the data back to the original 8760 columns and 6 rows and then it transposes the data so that each line is a timestep and each column is a row.

```
$ rlam i_3ph.txt i_ds3ph.txt i_ds5ph.txt | \
          rcalc -e 'r=$1-$4+$7;g=$2-$5+$8;b=$3-$6+$9' -e '$1=179*(.265*r+.670*g+.065*b)' | \
          rcollate -h -fa1 -ic 8760 -t > illum.txt
```

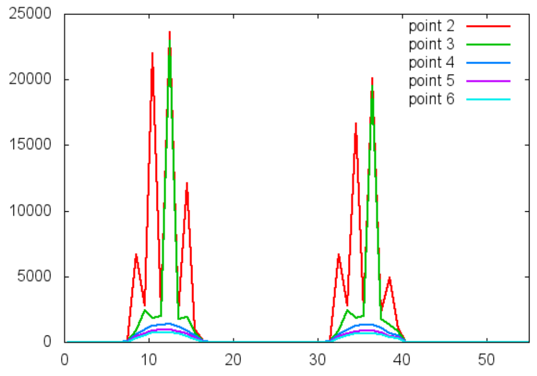


Figure 8 – Sensor point illuminance plotted for the first 50 hours of the year.

Note that the illuminance of point 2 and point 3 bounce up and down from one hour to the next, this is because they venetian blind shadow pattern moves across the sensor points. Also, values for sensor point 1 are invalid since it was located outside of the space.

7 Example 2 – Rendering

For example 2 we'll use the same fenestration system created in section 6.1 and BSDF generated in section 6.2. For this example we'll use the same octrees generated for example one in section 0.

7.1 View matrix (V)

The view matrix is generated in the same manner as the three-phase method:

```
$ vwrays -vf views/back.vf  -ff -x 500 -y 500 \
        | rcontrib `vwrays -vf views/back.vf  -x 500 -y 500 -d` -ffc -fo -o viewpics/back_%03d.hdr \
                -f klems_int.cal -b kbinS -bn Nkbins -m viewsurf \
                -ab 10 -ad 65536 -lw 1.52e-5 octs/model_3ph.oct
```

7.2 Direct view matrix (V_d)

The direct view matrix is generated in the same manner, except using -ab 1 instead of -ab 10.

```
$ vwrays -vf views/back.vf  -ff -x 500 -y 500 \
        | rcontrib `vwrays -vf views/back.vf  -x 500 -y 500 -d` -ffc -fo -o viewpics_dir/back_%03d.hdr \
                -f klems_int.cal -b kbinS -bn Nkbins -m viewsurf \
                -ab 1 -ad 65536 -lw 1.52e-5 octs/model_3ph.oct
```

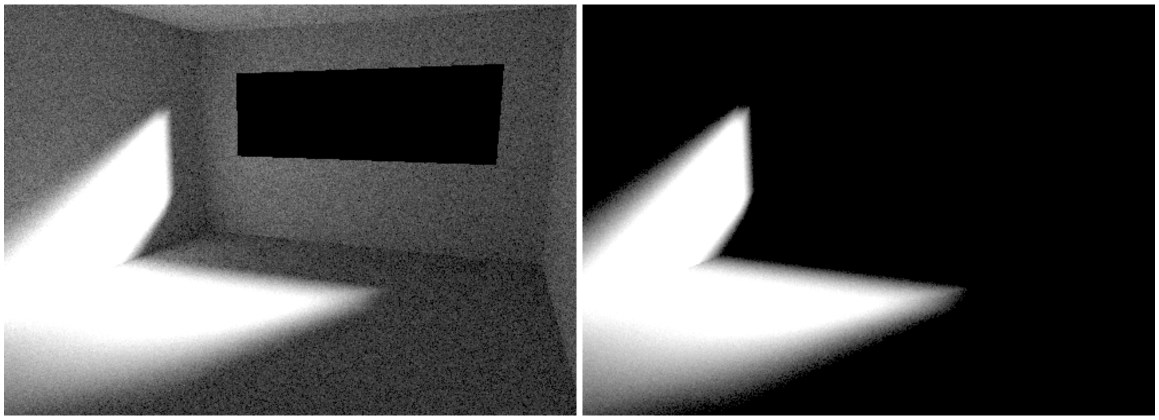


Figure 9 - Renderings for patch 60 of the view marix (left) and direct view matrix (right)

7.3 Daylight matrix (D)

We can use the same daylight matrix produced in example 1 (section 6.5)

7.4 Direct daylight matrix (D_d)

We can use the same direct daylight matrix produced in example 1 (section 6.6)

7.5 Direct sun coefficient matrix (C_{ds})

First we need to increase the number of files that the shell can open simultaneously using ulimit (since we are generating 5185 renderings). For the rendering we'll use the black model with suns created in the previous example. We're using the -c option with vwrays, which sends multiple sample rays for each pixel, to improve the quality of this image (specifically with regard to the venetian blinds). A matching -c option is used in the rcontrib command. The -pj option is also included in vwrays to jitter the rays in the pixel sampling – without this option the multiple rays for each pixel would be identical. We have an rcalc command between the vwrays and rcontrib command that returns the intersection and surface normal for each view ray. Then rcontrib calculates the illuminance at the intersection point using the -I (capital I) command. We are using the rtrace command instead of using -i (lower case i) with rcontrib in this case because the rcontrib method would calculate the illuminance on the BSDF surface instead of the on the proxied geometry. The middle rtrace command allows us to see the venetian blind in the sun coefficient renderings.

We're using a model with no suns for the middle rtrace to save time tracing shadow rays (which rtrace does even though we only want intersection and surface normal). We'll convert from the illuminance rendering to a luminance rendering using the material map in the next step.

```
$ ulimit -n 9999
$ xform -m black objects/testroom_Swall.rad objects/testroom.rad objects/ground.rad | \
        oconv materials/testroom.mat - objects/venetianblind.rad > octs/model_nosuns.oct
$ vwrays -c 4 -pj 1 -fa -vf views/back.vf -x 500 -y 500 | \
        rtrace -h- -opn -faa -ab 0 octs/model_nosuns.oct | \
        rcontrib `vwrays -vf views/back.vf -x 500 -y 500 -d` -n $procs -fac -fo \
        -o viewpics_ds/back_%04d.hdr -e MF:6 -f reinhart.cal -b rbin -bn Nrbins -m solar \
        -c 4 -I -ab 1 -ad 100 -dt 0 -dc 1 -lw 1e-2 octs/model_suns.oct
```

Figure 10 shows a one of the 5185 renders that comprise the direct sun coefficient matrix. The figure 10 rendering isn't equivalent to the direct view matrix rendering (Figure 9), but instead is equivalent to the second term of the five-phase equation (Figure 12). Additionally note that the venetian blinds are visible in this rendering, but because the resolution of the rendering is low they appear as spotty lines in the window. The stripe pattern on the wall and floor is fuzzy, since rcontrib uses jittering by default. If you prefer to see sharp shadows you can use -dj 0 in the rcontrib command.

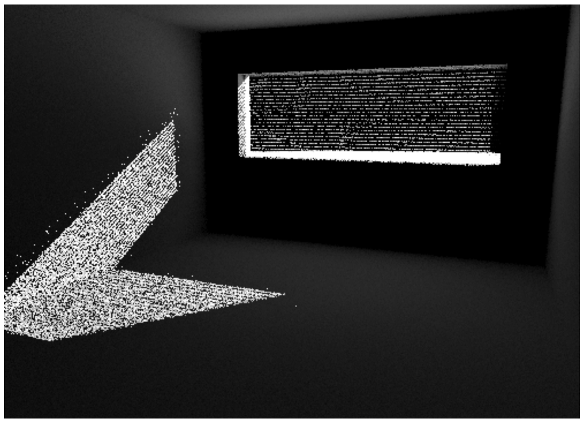


Figure 10 - Rendering from the direct sun contribution matrix.

Then we create a rendering with each pixel value equal to the material reflectance divided by pi. This is achieved by rendering an image with no ambient bounces, no light sources and setting -av to 0.31831. We're also using -x 1000 -y 1000 -ps 1 -pj 1 to mimic the oversampling used in the previous step.

```
$ oconv materials/testroom.mat objects/testroom_Swall.rad objects/testroom.rad \
        objects/ground.rad objects/glazing.rad objects/venetianblind.rad \
        objects/daymtxsurf.rad > octs/model_material.oct
$ rpict -x 1000 -y 1000 -vf views/back.vf -ps 1 -pj 1 -av 0.31831 0.31831 0.31831 -aa 0 \
        octs/model_material.oct | pfilt -x /2 -y /2 -1 -e 1 > materialmap.hdr
```

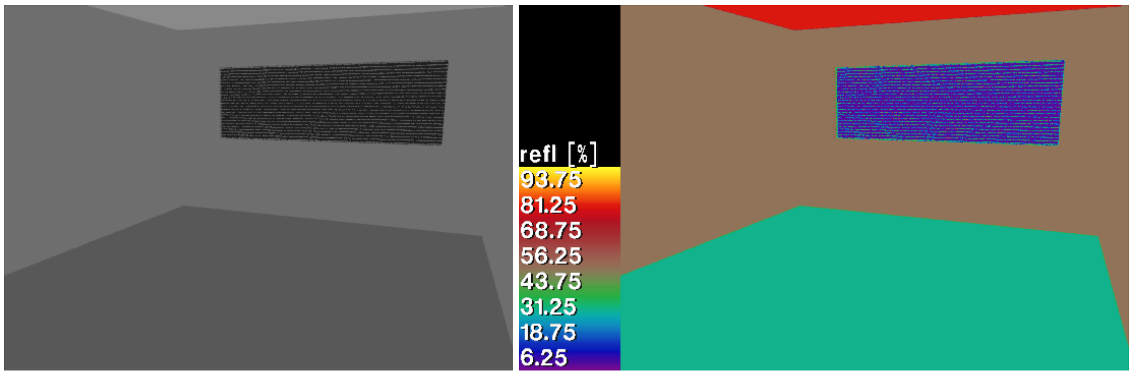


Figure 11 - Material map rendering and falsecolor representation.

7.6 Putting it all together

We will again use weather data for Oakland, CA and can use the sky matrix files generated in section 6.8. However to save time in this step you can cut the .wea file down to a few days and generate new sky matrix files.

The first term of the 5-phase equation is the three-phase simulation result:

```
$ dctimestep -n 8760 -if -o hourlypics/back_%04d.hdr viewpics/back_%03d.hdr \
        bsdf/fullwindow.xml matrices/daylightmatrix.dmx matrices/OakLand.smx
```

The second term is the direct sun component of the three-phase method:

```
$ dctimestep -n 8760 -if -o hourlypics_dir/back_%04d.hdr viewpics_dir/back_%03d.hdr \
        bsdf/fullwindow.xml matrices/daylightmatrix_direct.dmx matrices/OakLand_direct.smx
```

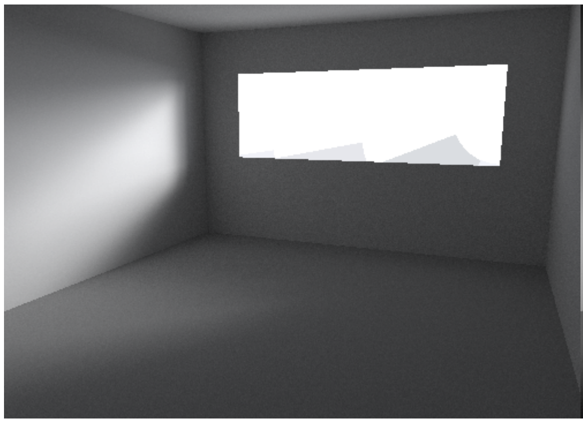
The third term is the direct sun component using the model with many suns:

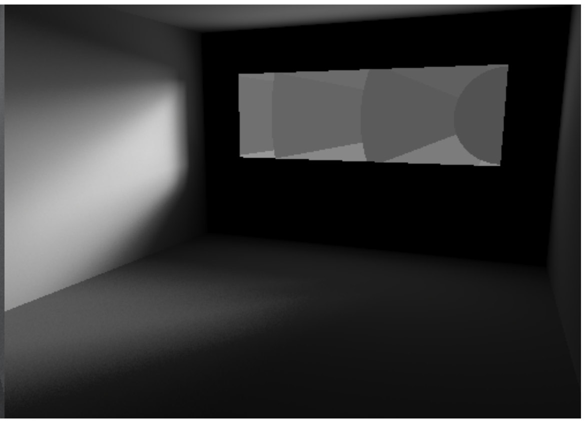
```
$ dctimestep -n 8760 -if -o hourlypics_ds/back_%04d.hdr viewpics_ds/back_%04d.hdr \
        matrices/OakLand_direct_m6.smx
```

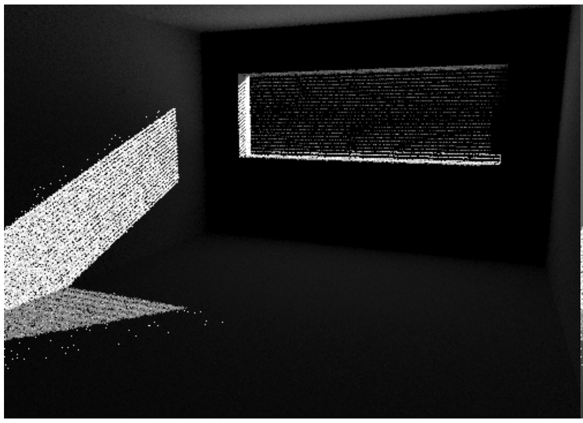
Then we can use pcomb to combine the terms, including the material map multiplication for the third term. The following bash command loops over the 8760 hourly timesteps, generating a complete five-phase rendering for each timestep:

```
for t in {1..8760}
do ts=`printf %04d $t `
pcomb -e 'lo=li(1)-li(2)+li(3)*li(4)' -o hourlypics/back_${ts}.hdr -o hourlypics_dir/back_${ts}.hdr \
        -o hourlypics_ds/back_${ts}.hdr -o materialmap.hdr > hourlyresult/back_${ts}.hdr
done
```

Renderings in Figure 12 represent all three terms of the three-phase method and the five-phase result for one timestep (Jan 1, 15:00 for Oakland, CA). Figure 13 shows the final rendering with tone mapping to mimic human visual response (using pcond). The venetian blinds are visible in the image, are difficult to make out due to their small size and the low resolution of these images.


First term (three-phase result)


Second term


Third term

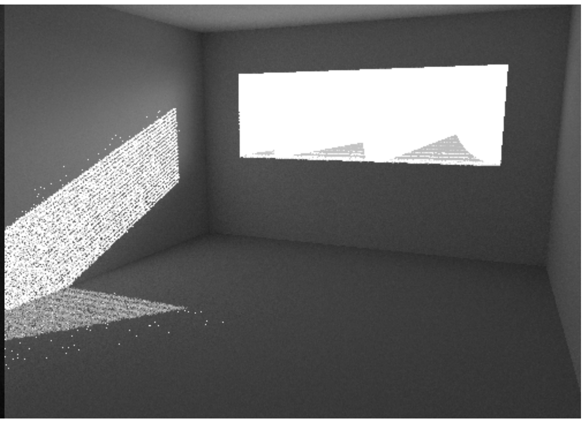

Five-phase result

Figure 12 - Rendered components and result for the 5-phase method

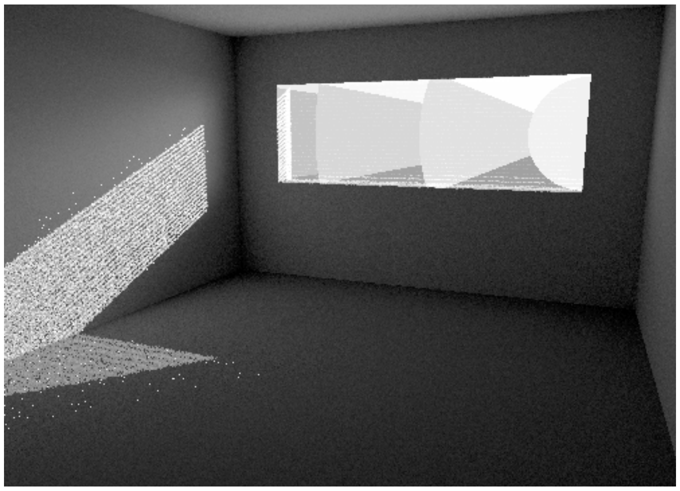


Figure 13 - Five-phase result with tone mapped using pcond

8 Simulation Duration

The following table contains simulation times for each stage of the five-phase simulation method. The simulation was run on a four-core (2.3 GHz intel Core i7) Apple Macbook Pro purchased in 2011. Rcontrib runs used the -n option (-n 4) to spread processes across the multiple cores.

	Illuminance Sensors (example 1)	**Renderings (example 2)**
View Matrix	0m 1.0s	0h 55m 50.6s
Direct View Matrix	0m 0.3s	0h 18m 7.5s
Daylight Matrix	0m 6.7s	0h 0m 6.7s
Direct Daylight Matrix	0m 1.5s	0h 0m 1.5s
Sun Coefficient Matrix	4m 36.6s	12h 38m 25.2s
Combining result	0m 4.6s	1h 1m 48.9s
Total Time	**4m 50.7s**	**14h 54m 20.4s**

I'm providing a test script that runs and times each stage of the simulations of the two examples. The script requires only a recent HEAD installation of Radiance, all scene files are downloaded automatically by the script. The script is written in bash, so should run on either mac or linux. The script can be obtained from the following website:

http://www.radiance-online.org/learning/tutorials/fivephasetutorialfiles/test_5phase.bsh/at_download/file

Prior to running, change the following line to the number of CPU cores on the machine for testing:

```
procs=4
```

9 References

Bourgeois, D., Reinhart, C.F., Ward, G. 2008 Standard daylight coefficient model for dynamic daylighting simulations, Building Research & Information, Vol. 36 p 68-82. http://www.tandfonline.com/doi/abs/10.1080/09613210701446325

McNeil, A., Lee, E.S. 2013 Further validation of the three-phase method, un-submitted manuscript.