DYNAMIC RADIANCE – PREDICTING ANNUAL DAYLIGHTING WITH VARIABLE FENESTRATION OPTICS USING BSDFS

Mudit Saxena[1], Greg Ward[2], Timothy Perry[1], Lisa Heschong[1] and Randall Higa[3]
[1]Heschong Mahone Group, Gold River, CA
[2]Anyhere Software, Albany, CA
[3]Southern California Edison, Irwindale, CA

ABSTRACT

Existing annual daylight simulation software fall short with respect to variable fenestration optics that change interior daylight distribution with sun position and/or operating schedule, thus limiting the ability to compare the performance of advanced fenestration systems. Many of these window or skylight systems can be described efficiently as a bidirectional scattering distribution function (BSDF), which characterizes their flux output as a function of input for a particular configuration. In this paper, we describe a new method that employs measured or simulated BSDFs to permit fast, matrix-based annual daylighting calculations. The matrices themselves are precomputed by Monte Carlo ray-tracing in a modified daylight coefficient approach we call *Dynamic Radiance*. The inner time-step loop then consists of multiplying the desired sky luminance vectors against three matrices in the general case, where a separate BSDF matrix permits dynamic fenestration control strategies. In this paper, the authors describe their implementation of the Dynamic Radiance method and demonstrate its application to a set of 61 real spaces modeled for a research project to determine new daylight metrics. We present results from these simulations and discuss advantages and limitations of the new approach.

INTRODUCTION

It is well understood that energy savings and electric demand reduction potential of daylighting is substantial. However, accurately predicting daylighting at an hourly time-step, for an annual simulation, is not a simple task. This was the task at hand for the Daylight Metrics Project [Heschong et al. 2010, Saxena et al. 2010], a reseach project to develop a set of simulation-based metrics to describe daylighting in architectural spaces. The simulation task required annual daylighting simulations for 61 surveyed spaces in six cities across the United States.

Initially a research version of *DaySim DDS* version 2.4 [Bourgeois et al. 2008] was choosen to perform the annual simulations, as it would provided the most modeling accuracy and supported parametric studies. However, mainly due to limitations in *DaySim's* modeling assumptions for window shadings such as blinds and fabric shades (hereon called blinds for brevity), the project team decided to use an alternative progam. While *DaySim* had the ability to operate blinds according to a solar trigger, it was limited to one schedule for all blinds in a given space, irrespective of their orientation. Furthermore, simplified assumptions of blinds light transmittance (20% diffuse, 0% direct) were found to be too simplistic. While *DaySim 2.4* does support the simulation of blinds explicitly [Reinhart et al. 2001], that approach was not used due to additional demand on computation-time. Changes were made to the research vesion of *DaySim 2.4* to enable more than one blind schedules, ultimately, achieving full functionality for the new blinds operation and output functions in *DaySim* was found to be beyond the resources of the project team.

Considering many alternatives, the project team eventually decided to commission development of a new annual simulation approach using Radiance. This approach, which for the purposes of this paper we call *Dynamic Radiance*, would build on *DaySim's* achievements and use a similar daylight coefficient methodology. The Dynamic Radiance method provides the desired blinds-operation functionality, blinds light transmittance, and data ouput. It also adds an important new capability—the ability to model variable fenestration optics that change interior daylight distribution with sun position and/or operating schedule (dynamic fenestration performance).

BI-DIRECTIONAL SCATTER DISTRIBUTION FUNCTIONS (BSDF)

Central to this capability of modeling dynamic fenestration performance, is the use of Bi-directional Scatter Distribution Functions or BSDFs.

A full BSDF, as defined for WINDOW 6, consists of a full Klem sample, or a 145x145 matrix, defining light transmittance through a fenestration assembly. Incoming light striking the exterior surface of the assembly is represented through 145 exterior vectors.

Similarly, light transmitted by and exiting the assembly is represented through 145 interior vectors, as shown in Figure 1. A BSDF file defines coefficients (c ≥0) to allocate light from each exterior vector to each interior vector. These coefficients are stored in a 145x145 table. Each columns represent a single exterior vector, while each row represents a single interior vector. The light transmitted into the space on any one interior vector is given by Eq. (1) below

$$I_j = \sum_{k=1}^{145} c_{jk} E_k \quad (1)$$

Where:

E_k = light along exterior vector k
I_j = light along interior vector j
c_{jk} = coefficient relating I_j to E_k which is stored in the cell located in column k, row j of the BSDF

Our implementation of the Dynamic Radiance method utilizes BSDF files to represent fenestration assemblies consisting of the glazing and window coverings. Previous research has shown BSDF data provides acceptable resolution for simulating complex fenestration assemblies [Konstantoglou et al, 2009]. Further discussion of the file format is available from LBNL [Jonsson, 2009; Fernandes, 2006].

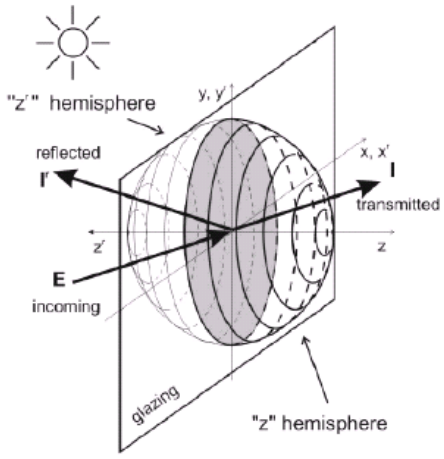

Figure 1: Schematic Diagram representing interior and exterior vectors of a BSDF [Fernandes, 2006]

DYNAMIC RADIANCE

Radiance is a lighting simulation and rendering system that was first released by the Lawrence Berkeley National Laboratory in 1989, and has undergone continuous modification and improvement since. Now in its 20th release, *Radiance* 4.0 includes the ability to predict the performance of complex window fenestration systems, defined as the BSDFs just described. To be clear, there is no identifiable program called "Dynamic Radiance." We have merely created a set of custom scripts and Makefile's that apply the tools already present in *Radiance* 4.0. The method we are calling Dynamic Radiance is not distributed separately, does not have a user interface, and would have to be substantially modified for a different set of building analyses. The basic tools we will introduce, **rtcontrib**, **genklemsamp**, **genskyvec**, and **dctimestep**, are all part of *Radiance* 4.0, and we are using them to illustrate this overall approach, which we call Dynamic Radiance.

Figure 2: A full simulation using a BSDF on a window with venetian blinds that took 17 hours to generate

In a more traditional mode, the BSDF is used in *Radiance* to represent a window as a "light source" in a backwards ray-tracing calculation of interior illumination. This requires the use of the *Radiance* **mkillum** program, which has been able to interpret BSDF files since the last release. Using this process, high-quality renderings may be obtained as shown in Figure 2, which took 17 hours to generate on a single processor. However, since daylight is a dynamic phenomenon, creating a view of a single point in time is of limited use, and we would prefer a collection of renderings or animations showing how our environment reacts to changing sky conditions. Ideally, we would plot this information over an entire year based on appropriate weather data. In the case of an operable shading system, we may even wish to compare different control algorithms as part of our analysis. If it takes hours to evaluate each time step, this type of annual daylight simulation would be impractical and forbidden to us.

In the past two decades, researchers have been exploring *daylight coefficients* as a means to faster annual calculations in complex spaces [Reinhart, Mardaljevic, etc.]. In this approach, the sky is subdivided and the connection or form factor between sky patches and interior illuminance values (typically) are computed. Since light is linear, it is then a simple matter to multiply the sky luminance values for a

particular condition by these coefficients and sum them together to obtain the desired, corresponding interior illuminances. This can be expressed as a matrix equation whose input is the sky vector corresponding to patch luminances at a particular time, and after passing through our daylight coefficient matrix, gives us a vector of predicted illuminance values:

$$\vec{\mathbf{i}} = \mathbf{C}\vec{\mathbf{s}} \quad (2)$$

Where:

i = resultant illuminance vector (N values)
C = daylight coefficient matrix (N rows by M columns)
s = sky luminance vector (M patch values)

The difficulty we face applying this technique to complex fenestration is two-fold. First, the calculation of the matrix **C** becomes intractable when the interactions at the window involve multiple reflections. Second, in the case of an operable shading system, we would like to be able to modify **C** as we adjust the system, calculating a different version of it for each shade position. This only exacerbates the first problem. What we need is a reformulation of the problem, which allows for the easy substitution of different shading conditions as BSDF's, and also factors the original **C** matrix into more easily calculated components. This is the revised formulation we use in our Dynamic Radiance method:

$$\vec{\mathbf{i}} = \mathbf{VTD}\vec{\mathbf{s}} \quad (3)$$

Where:

V = a "view matrix" that defines the relation between measurements and exiting window directions (N rows by K columns)
T = the transmission portion of the BSDF (K rows by L columns, usu. K = L)
D = the "daylight matrix" that defines the relation between incoming window directions and sky patches (L rows by M columns)

The **i** and **s** vectors are the same as above; we have simply factored the **C** matrix into three component matrices. The transmission matrix **T** is given as input, so all we really need to compute are the **V** and **D** matrices. For both problems, we employ the *Radiance* **rtcontrib** program.

The rtcontrib Program

Radiance performs its lighting calculations by following rays backwards from the point of measurement and into the scene in search of illumination sources, which are specified as input along with the scene's geometry and materials. The basic **rtrace** tool takes a ray origin and direction for example, and computes its radiance (the radiometric equivalent of luminance) by following the ray into the scene to see what it intersects. If the ray intersects a diffuse surface, for example, additional rays are spawned to the light sources to determine the surface illumination, whereby the outgoing radiance can be determined from reflectance. (The full calculation is a bit more complicated, involving multiple diffuse reflections and so on.)

What if we wanted to know how the outgoing radiance would change as a function of light source intensities? Say we have multiple, dimmable fixtures, which we wish to control continuously to optimize lighting in our space. Recomputing an entire image of radiance values, for each pixel corresponds to at least one ray, would be rather time consuming. It would be better and faster to compute one image for each light source, then add them together as components in our final result. Many people have taken advantage of the linearity of light to do exactly this, but with **rtcontrib**, we have an even more efficient route to such a solution.

Recomputing an image multiple times with different light sources involves many of the same ray intersections with surfaces, especially in the case of multiple diffuse interreflections. We can short-cut this process by computing our multiple images in a single step! We simply identify each light source in our scene that corresponds to a desired image, and **rtcontrib** does the rest. Moreover, we can subdivide exitant directions from our light sources, thereby allowing us to modify luminaire spatial output distributions. In applications such as directable theater lighting instruments, this would be an obvious advantage, but in our case, we want to know how different light output from our windows affects the interior illumination, which is the **V** matrix in the equation above.

Computing the View Matrix (V)

The view matrix **V** defines the relation between a particular set of sensors and a window group. The sensors may be a set of illuminance points on the workplane or ceiling, or an entire image of radiance directions from a particular viewpoint. The window group may be a single opening or a skylight, or a portion of a segmented window, or multiple windows all facing the same direction. The decision of how to group windows may be dictated by geometry, or the desire to control shading (such as blinds) independently, or other factors. At a minimum, we need a separate group for each window orientation, since we use the surface normal to anchor our directions. Figure 3 shows the contributions from the leftmost bay window of one of our test models, assigning a different random color to each of 145 output directions. Each bay window would require a different group, since they have independent orientations, but the two windows to the left of the bay could be placed into one group if desired.

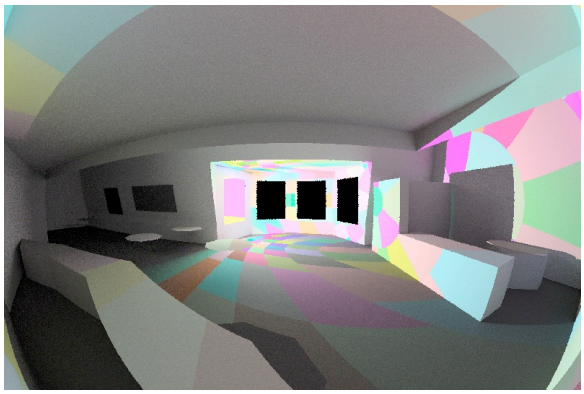

Figure 3: A rendering showing the different output directions for a single window group

Because **rtcontrib** permits output based on direction and material grouping, a single run can produce all the desired **V** matrices corresponding to every window group, and this calculation needs to be done only once per unique interior geometry. Depending on the length of the desired **i** sensor vector, scene complexity, and window groups, this step takes anywhere from under a minute to several hours. The scene above took about three hours to compute for 145 directions for each of 7 window groups feeding 426,400 sensors (pixels in an 800x533 image). That's about 426 million coefficients, which we packed into 1015 Radiance pictures (662 MBytes).

The advantage is that a final image for a particular shade and sky condition can be computed in about 10 seconds. Figure 4 shows one such time step.

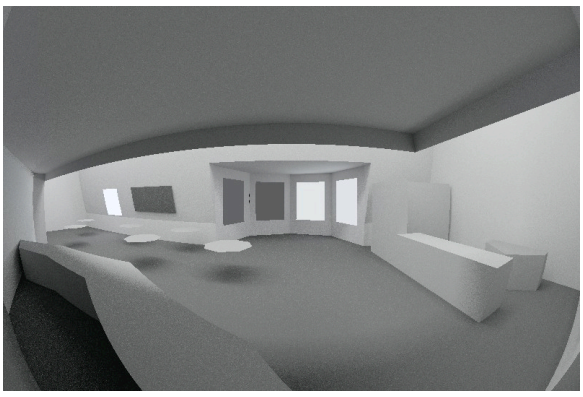

Figure 4: A combined result based on a particular time of day, year, and shading condition that took 10 seconds to generate

Computing the Daylight Matrix (D)

The calculations above rely on knowing how light is arriving at each window, which then passes through the BSDF matrix **T** for that group. These form factors are stored in the **D** matrix in Eq. (3), which relates sky patch luminances to incident window directions, accounting for external obstructions and interreflections. In fact, a separate **D** matrix is computed for each window group, since the set of directions is different for different orientations. The more general version of Eq. (3) is therefore

$$\vec{\mathbf{i}} = \sum_{g=1}^{n} \mathbf{V}_g \mathbf{T}_g \mathbf{D}_g \vec{\mathbf{s}} \qquad (4)$$

Where:

g = window group index
n = number of window groups

The actual calculation of **V** uses **rtcontrib** to sample outgoing ray directions for each window group, collecting results for each sky patch. To assist this process, we have written a Perl script **genklemsamp** that identifies windows with a given orientation in a given geometry file, then sends out rays with random origins distributed over their surface(s). We employed the full (145x145) Klems basis described in the WINDOW 6.1 / THERM 6.1 Research Version User Manual for our sample directions, since it corresponds to the BSDF data available to us from *WINDOW6* [Windows & Daylighting Group, 2006].

Figure 5 shows the exterior of the space we showed earlier. The circled bay window was used in the fisheye projection shown in Figure 6. We assigned a random color to the 145 Tregenza patches (plus one for the ground), and overlaid a grid corresponding to the 145 Klems patches on the window. The visible surfaces appear grayish because they see most of the sky, so the coloration averages out. Hence, the corresponding rows in our **D** matrix will have many non-zero terms. The rows that correspond to direct views of the sky will have only a few non-zero terms, since only a few Tregenza patches are visible from each. Of course, it would be unwise to generate the **D** matrix directly from such an image, as it samples only a single point on the window. Our Perl script therefore randomizes the sample ray origins over the window to obtain a good average for each matrix coefficient.

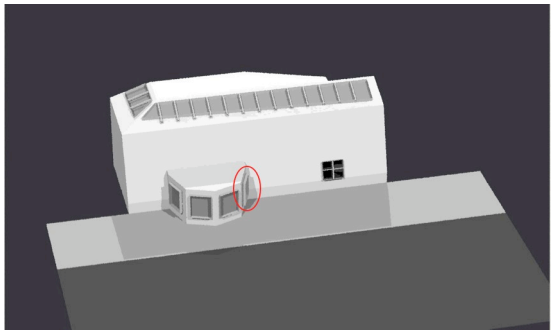

Figure 5: The exterior of our example space, indicating the window whose view is shown in Figure 6 below

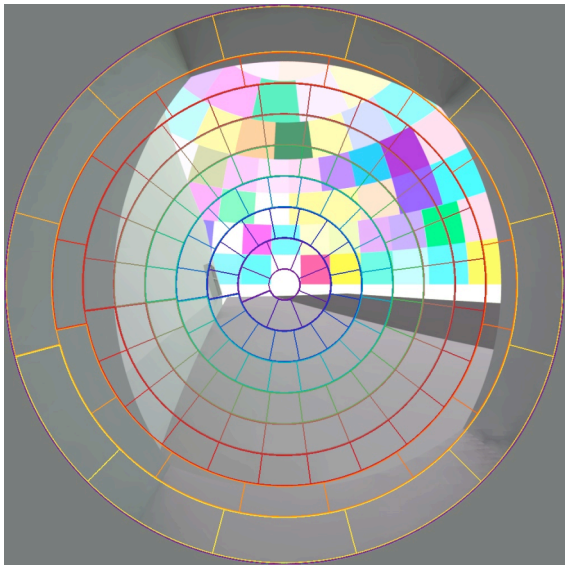

Figure 6: The grid lines divide our hemisphere into 145 patches using the full Klems basis. Randomly colored patches in the sky indicate the 145 Tregenza patches

It was discovered early on that, even if the shading system on the window is fairly diffusing and blocks any direct sunlight, 145 sky patches was not enough in cases where there was shadows cast by nearby geometry. The Tregenza resolution is roughly 12°, and we distribute the sun's energy into the three nearest patches, so the actual resolution is closer to 24°. If a neighboring building or structure is going to partially or fully block direct sun on the window, 24° is a pretty wide margin of error, and we noticed significant discrepancies in our early results. We found it necessary for many models to subdivide the sky further, and ended up using a 4x4 subdivision of the Tregenza patches developed by other researchers [Mardaljevic 2000, Bourgeois et al. 2008]. With 2305 patches (plus ground), we have an effective resolution of about 6°, corresponding roughly to a half hour in terms of solar position. Greater accuracy is of course possible with a finer subdivision, but we found this to be adequate to our needs

Sky Patch Vectors and Evaluation

Once we have our **Vg** and **Dg** matrices, and have selected the transmission matrices **Tg** for each window group, we can apply them directly in Eq. (4) or multiply and sum them together to arrive at a complete daylight coefficient matrix needed for our original Eq. (2):

$$\mathbf{C} = \sum_{g=1}^{n} \mathbf{V_g T_g D_g} \qquad (5)$$

In either case, we need a sky patch vector **s** corresponding to the current time step in order to compute a final result vector **i**. For this purpose, we have created another Perl script **genskyvec** that takes a sky model produced by the Radiance **gensky** program or **gendaylit** by the ISE in Freiburg, Germany. The advantage of the latter program is that it takes direct and indirect solar irradiance as input and computes the sky type from these data, which one can find in reference weather files for most climates.

The final evaluation involves multiplying the combined matrix by our sky vector, which is a very fast calculation. Even when different **T** matrices are being tried at each timestep to find an optimal result, the full matrix multiplication takes only a few seconds, and a convenient tool **dctimestep** is provided for this purpose.

USING DYNAMIC RADIANCE ON 61 MODELS

We applied the Dynamic Radiance approach to generate annual results for illuminance, sun penetration, and skyview for the Daylight Metrics project. A field survey provided detailed information to create detailed *Radiance* .rad scene files for 61 spaces in six different cities across the United States. The .rad files were created using *Ecotect v5.50*. Horizontal illuminance sensors were provide at 1 ft by 1 ft spacing, on the task level (31 inches), eye level (48 inches) and ceiling level (height varies by space)

After the models were exported from *EcoTect* the windows were grouped in each space by orientation. In addition, we further limited groups to windows which were co-planar, contained the same glass type, and the same window covering (blinds or shades where present). Lastly, BSDF files were assigned to each window group.

For the scope of this project, we limited blinds operation to only two conditions – blinds are either fully deployed or completely retracted, a deployed blind completely covers the window, while a retracted blind does not cover any portion of the window. Blinds were

triggered to deploy when 2% of the horizontal 'eye level' sensors had an illuminance of 4,000 lux (roughly equivalent to 50 Watt/m2 of solar radiation) or greater when considering only sunlight as an illumination source from any given window group. One of two BSDF files were assigned to window groups depending on the characteristics of the windows in that group. A BSDF representing an un-shaded, or open window was assigned to all window groups. This "open" BSDF accounted for the visible transmittance of the glazing in the windows. If the windows in the group had blinds or shades, an additional BSDF was assigned to the group to reflect the "closed" condition. This BSDF accounted for the visible transmittance of the glazing and the associated blinds or shades. Other details of blinds assumptions used in the project can be found here [Saxena et al, 2010].

The simulations using the Dynamic Radiance approach took between 2 and 14 hours for 80% of the models with a median time of 5.2 hours. The quickest model finished in just under an hour. The space had only one window, and had little exterior context modeled. The longest model took just over 28 hours. It was a relatively large model, had 18 window groups, and was surrounded by multiple high-rise skyscrapers. Processor run times were not recorded for all models, however, of the 41 timed runs, only 3 took longer than 14 hours.

Simulation Results

The color contour plots in Figure 7 represent average monthly illuminance for each sensor on the task level illumination grid for January for a space facing south.

The plot on the left shows illuminance without blinds, while that on the right is with blinds operated as per the blinds trigger assumption. The data were averaged separately for each hourly time step from 8:00-17:00, for the months January.

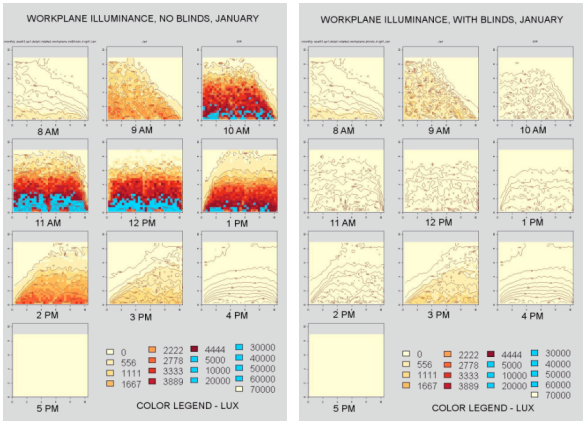

Figure 7: Average workplane illuminance in January – No blinds case (left), blinds operated case (right)

The plots clearly show that during the hours when blinds are deployed, the average illumination at the workplane is much lower with the blinds closed, as expected, but the directional nature of the light through the blinds is preserved due to the use of BSDFs.

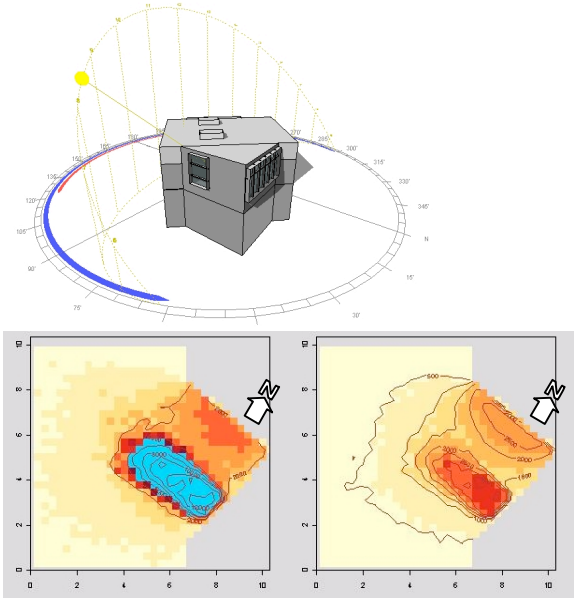

Figure 8: Illuminance distributions at 8:00 AM on July 11th - No blinds case (left), blinds operated case (right)

Figure 8 show the 3D model and illuminance plots for a space in San Francisco at 8:00 AM on July 11th. The plot on the left is without blinds, with that on the right is with blinds operated as per the blinds trigger assumptions. The space has two windows, one facing north and another facing east. As per the rule-set for defining window groups, since each window has a different orientation, two window groups were assigned, one for each window.

The illuminance plot without blinds (left) shows that at 8:00 AM, the east-facing window is receiving direct sun (shown by blue >5000 lux), while the north-facing window receives only diffuse or reflected light.

The illuminance plot with blinds operated (right) shows that, illuminance next to the east window reduces to show that blinds have been deployed, while that next to the north-facing window reamins more or less unchanged. This result is in-line with what can be expected with two window groups. Only the blinds on the east-facing window are getting deployed, while blinds on the noth-facing window reamin open.

ADVANTAGES AND DISADVANTAGES OF DYNAMIC RADIANCE

Speed and the ability to incorporate arbitrary BTDFs on windows and skylights are the principal advantages of the Dynamic Radiance method. Combining daylight coefficients with window-specific BSDF data allows us to generate annual simulations in an operationally-acceptable time-span. By splitting the daylight coefficient matrix into two matrixes, an interior- and an exterior-matrix, we are able trace light paths inside and outside the building only once and reuse the results. Then, simple matrix math gives us the resultant illumination for each point with a given window matrix (BSDF) substituted in between the interior- and an exterior-matrices. This timestep calculation can be inserted into an annual simulation system without requiring direct links to Radiance, simplifying the process as well. Any controllable shading system that can be discretized into a finite number of BSDFs may be evaluated, and the control algorithm can be simple or complex, since the calculation is so quick. This opens up the possibilites to evaluate the daylighting performance of dynamic blinds and shading systems that use moroized controls and change postion based on climtic inputs.

The Dynamic Radiance approach utilizes top-level *Radiance* component programs. These programs have an established interface and years of testing. In the event that bug fixes or enhancements are added to *Radiance*, the suite of scripts and Makefile's used to implement the Dynamic Radiance approach can be updated simply by installing the current version of *Radiance*. No compilation is necessary due to the loose coupling and standard interfaces between the programs that constitute the Dynamic Radiance approach and *Radiance* 4.0 component programs.

Despite its benefits for annual simulation and complex fenestration, the Dynamic Radiance method comes with some limitations. Firstly, it does not project exterior shadows into the space, so a partially obscured window group will pass the average light reaching its exterior, evenly distributed over the area of the window group. The window may be subdivided to compensate, but doing so increases the computation time, and determining the optimal subdivision in advance is difficult. Secondly, window-assembly light-distribution patterns are limited by the BSDF format, so any direct or redirected component is smeared over about 15° with the current standard basis. This is illustrated by Figure 9, which can be compared to Figure 2 calculated by **mkillum**. We have lost the details of the blinds, and even the shadows due to the window edges have been blurred significantly. However, this took only an hour to compute, including precalculation, and the next time step can be computed in a matter of seconds.

Figure 9: The same scene as Figure 2, computed using the Dynamic Radiance approach

ACKNOWLEDGMENT

The Dynamic Radiance approach was developed with funding from Southern California Edison, and additional funds from Lawrence Berkeley National Lab (LBNL).

The Daylight Metrics Project is funded by the California Energy Commission (CEC) Pubilc Interest Energy Research (PIER) Project, with additonal funding support from Sothern California Edison (SCE), Northwest Energy Efficiency Alliance (NEEA), and New York State Energy Research and Development Authority (NYSERDA).

REFERENCES

Bourgeois, D., C.F. Reinhart, and G. Ward 2008. A Standard Daylight Coefficient Model for Dynamic Daylighting Simulations. Building Research & Information, 2008. 36(1): p. 68 - 82

Fernandes, L. 2006. Coordinate Systems for Representation of Bidirectional Optical Properties of Fenestration. Lawrence Berkeley National Lab, Berkeley, California, USA

Heschong, L., M. Saxena, and R. Higa 2010. Improving Prediction of Daylighting Performance. Proceedings of the ACEEE 2010 Summer Study on Energy Efficiency in Buildings.

Jonsson, J. C. 2009. How to Interpret the Window6 BSDF Matrix. Lawrence Berkeley National Lab, Berkeley, California, USA

Konstantoglou, M., J. C. Jonsson, and E. Lee 2009. Simulating Complex Window Systems using BSDF Data. PLEA 2009 - 26th Conference on Passive and Low Energy Architecture, Quebec City, Canada

Mardaljevic J 2000. Daylight Simulation: Validation, Sky Models and Daylight Coefficients. PhD Thesis, De Montfort University, Leicester, UK

Reinhart, C. F. and O. Walkenhorst 2001. Dynamic RADIANCE-based Daylight Simulations for a full-scale Test Office with outer Venetian Blinds. Energy & Buildings 33(7): 683-697

Saxena, M., L. Heschong, K. V. D. Wymelenberg, S. Wayland, 2010. 61 Flavors of Daylight. Proceedings of the ACEEE 2010 Summer Study on Energy Efficiency in Buildings

Windows & Daylighting Group 2006. WINDOW 6.1 / THERM 6.1 Research Version User Manual. Lawrence Berkeley National Lab, Berkeley, California, USA