Ambient Calculation

Crash Course

John Mardaljevic

Professor of Building Daylight Modelling
School of Civil & Building Engineering
Loughborough University, UK
Lighting simulation is a hunt for light
There are several approaches we can use to hunt for light

- Ray tracing - forward or backwards
- Radiosity
- Others: Photon mapping, etc.
Direct sky

Indirect sky

Total illumination

Illuminance [lux]

100

1000

10000
Indirect sky
Indirect sun
Ray tracing: forwards or backwards?

Very little from there ends up in here.

*Radiance* uses *backwards* RT.
Sunlight (beam radiation) can be intense and comes (usually) from one direction.

Skylight and reflected light (from sun and sky) can come from all directions.
Radiance treats the components of light differently.
surfaces have the same diffuse reflectance, say for the floor and the walls as in Figure 6. With a constant ambient value it is impossible to distinguish between the floor and the wall, except for where they are illuminated by direct sun.

2.2 Computing the components of illumination

Here we introduce the Radiance notion of reflections, also known as 'ambient bounces'. Consider a simple scene comprised of a closed rectangular room with a single light source. Say we were to use Radiance to generate an image showing a view of one of the walls, Figure 7. The wall is illuminated directly by the light source (\text{a}) and also by light arriving at the wall after one (\text{b}) or two (\text{c}) reflections (i.e. bounces).

The ray tracing approach to rendering follows "view rays" from the virtual camera through pixels in an imaginary plane into the environment. The (virtual) pixel plane has the same dimension (in pixels) as the resulting image. The view parameters determine what is visible of the scene. A schematic showing how this process operates is given in Figure 8.

Conceptually, the direct component is the easiest. The view ray (\text{red}) intersects with the scene. A shadow ray (\text{orange}) is sent from the point of intersection towards the light source to determine the direct light source.

Figure 4. Components of illumination

Plate 23 Rendering with Radiance
We “hunt” using different tactics depending on the source of illumination

- A deterministic method for the direct contribution from “concentrated” (i.e. direct) sources of light, e.g. sun or luminaire.

- A random (or stochastic or Monte-Carlo) method to “hunt” for light that could arrive from any direction (e.g. skylight or any type of reflected light). In Radiance this is done using hemispherical sampling.
Deterministic and hemispherical sampling

**Deterministic** - we know *a priori* where the light is coming from, so we send rays to the source.

**Hemispherical** - we don’t know in advance where the illumination is coming from, so we search (i.e. sample) every direction where it might come from.

How we define an emitting material in *Radiance* determines how it will be sampled:

- Material type **light** -> deterministic sampling
- Material type **glow** -> hemispherical sampling
rpict

‘Camera’

‘Pixel plane’

rtrace
Lesson 2.2 Radiance basics Page 6 of 6 Copyright 2004 De Montfort University

Figure 8. Computation of direct, indirect and specular components

(a) View ray intersects with scene here. A “shadow ray” is then sent to determine if this point of the scene (i.e. pixel) is illuminated by the light.

Direct

Light 'Camera' 'Pixel plane'

Hemispherical sampling initiated here. Where a ray intersects with the scene, shadows rays may be sent out to determine if this point is illuminated by the light.

Specular reflection to light source.
Indirect

Hemispherical sampling initiated here. Where a ray intersects with the scene, shadows rays may be sent out to determine if this point is illuminated by the light source.
Figure 8. Computation of direct, indirect and specular components

- **Light**
  - Camera
  - Pixel plane
  - View ray intersects with scene here. A "shadow ray" is then sent to determine if this point of the scene (i.e. pixel) is illuminated by the light.

- **Hemispherical sampling initiated here.** Where a ray intersects with the scene, shadow rays may be sent out to determine if this point is illuminated by the light source.

- **Specular**
  - Specular reflection to (direct) light source.
  - Specular reflection to illuminated room surfaces.
When to use light and glow sources

We use the material light for important sources of illumination, e.g. electric luminaire, the sun.

These participate in the direct calculation of illumination.

The material glow is used to describe extended sources of illumination (sky or ‘glowing’ ground) and also unimportant sources that may be visible to the ‘camera’ but do not contribute significantly to scene illumination. These participate in the indirect calculation of illumination.
Example scene: two polygons

The test scene comprises two polygons - one is an emitter of light which shines onto the other polygon. View parameters set to see source shining downwards and the resulting illumination on the upper-side of the polygon below.
Define the emitting material as **light**

A shadow ray is sent from the reflection polygon to the source at every point in the pixel plane where the reflection polygon is visible.

The reflecting polygon is evenly illuminated by the light source. This is clearly revealed in the false colour image.

Note: `-ab 0` setting used, i.e. inter-reflection calculation turned off.
Adaptive source subdivision

A light source will be subdivided until the width of each sample area divided by the distance to the illuminated point is below the ratio $ds$ [default value = 0.2].
Define the emitting material as **glow**

Now we have to switch on the inter-reflection to **hunt** for the light source, i.e. set `-ab 1`. We’ll hunt for the source using different numbers of hemispherical sampling rays (the `ad` parameter) to see the effect.

- `ad 32`
- `ad 64`
In scenes where some light-scattering surfaces are much brighter than others, it may be more efficient to sample those regions more heavily. The -as option specifies a number of additional samples to send into the regions of the hemisphere with the highest variance, estimated by the difference in sample radiances taken in each neighborhood. Note that this technique is effective only if the number of initial samples (set by -ad) is large enough to discover all of the problem areas [KA91].

Figure 12.8 shows a hemispherical fisheye view from the floor of a conference room model. Figure 12.9 shows the values returned by about 2000 sample rays distributed according to our stratified sampling technique. Note that this number of samples is not adequate to identify objects in the scene visually, though it is more than enough to compute a good average value for indirect irradiance. Note also that the light sources in the scene appear dark. This is because we are computing indirect contributions, so any source that is intercepted returns zero in this part of the calculation.

The sampling pattern is not evenly distributed across the hemisphere

Fig 12.7 Rendering with Radiance
Increasing the number of **ad** rays does produce smoother shading (at greater computational cost)
But even with \texttt{-ad 4096} the illumination from the \texttt{glow} material is not quite as smooth as with that from the \texttt{light} material.
Why are the **glow** renderings lumpy?

With a small **glow** source, sometimes the hemispherical sampling finds (i.e. “hits”) the source, and sometimes it doesn’t. Note also that there is a random (or stochastic) component to the ray direction.

Notice that the lumpiness occurs at scales much larger than the effective dimension of a pixel - what does that suggest about hemispherical sampling compared to deterministic?
What’s the significance of the big lumps?

These suggests to us that hemispherical sampling is not happening for every pixel.

If it was, then the “sometimes you find the source sometimes you don’t” effect would be happening from one pixel to the next - resulting in lumpiness at the pixel scale.

Usually in Radiance, hemispherical sampling is set to happen at points every now and then across a scene, and not at every pixel. Radiance then interpolates (i.e. estimates) values between these points.
Why use interpolation?

Simply, to be efficient. Consider, for the images used previously, the reflecting polygon comprised ~25,000 pixels. In the deterministic calculation (light), a shadow ray was sent to the source for each of the 25,000 pixels where a view ray intersected with the reflecting polygon.

If hemispherical sampling occurred at each of these pixels, then the number of rays sent would be 25,000 times the ad number:

\[ 25,000 \times 128 = 3,200,000 \text{ rays}; \text{ or,} \]
\[ 25,000 \times 4096 = 102,400,000 \text{ rays.} \]

Even for -ad 128 many times more hemispherical sampling rays are sent out than for the deterministic calculation, but most of those will “miss” the small source.
Where interpolation took place

The **genambpos** utility was used to place markers (red spheres •) in the scene where interpolation took place.

Hemispherical sampling took place at these points to generate this image.
Recap

For small, important sources of illumination, we describe the emitter using the material **light** so that it is sampled using the direct (deterministic) calculation.

In the previous example, the scene didn’t allow for inter-reflection. Here, we modify the scene by adding an occluding polygon to see how hemispherical sampling is used to compute indirect or (inter-reflected) light.
Scene with partially occluding polygon

Polygon B positioned to partially shade Polygon A from the light source (material light)

View shows the underside of Polygon B and the topside of Polygon A
Rendering for occluding scene -ab 0

Underside of polygon B not illuminated

Topside of polygon A half in shade

90,000 cd/m²

0 cd/m²

~450 cd/m²
Rendering for occluding scene -ab 1

Underside of polygon B now illuminated

Topside of polygon A still half in shade

90,000 cd/m²

~35 cd/m²

0 cd/m²

~450 cd/m²
Rendering for occluding scene -ab 2

Underside of polygon B illuminated

Shaded half of polygon A now illuminated by reflected light from polygon B
Hemispherical sampling (HS) took place at these locations for \texttt{-ab 1}

HS from here found the illuminated half of the lower polygon

But HS from here did not find any illuminated surfaces (the light source is excluded from the indirect calculation)
Hemispherical sampling took place at these locations for \(-ab\ 2\)

Level 1 HS from the lower polygon can now find the reflected light from the (underside) of the upper polygon.
Questions?
Some quantitative examples

- Predict the illuminance under a simple sky (without sun).
- First a uniform (i.e. constant brightness sky).
- Then a CIE standard overcast sky.
# sky_uni.rad
# uniform brightness sky (B=1)

void glow sky_glow
0
0
0
4 1 1 1 0

sky_glow source sky
0
0
0
4 0 0 1 180
% oconv sky_uni.rad > sky_uni.oct

% echo "0 0 0 0 0 1" \ |
| rtrace -h -I+ -w -ab 1 sky_uni.oct

3.141593e+00  3.141593e+00  3.141593e+00

\[ I = 0.265I_R + 0.670I_G + 0.065I_B \]
$$I = \int_{0}^{2\pi} \int_{0}^{\pi/2} B(\theta, \phi) \sin \theta \cos \theta \, d\theta \, d\phi$$

$$I = B \int_{0}^{2\pi} \int_{0}^{\pi/2} \sin \theta \cos \theta \, d\theta \, d\phi$$

$$I = \pi B$$

$$I = 3.1415926$$
Uniform sky

CIE standard overcast sky

\[ B_\zeta = B_z \]

\[ B_\zeta = \frac{B_z (1 + 2 \cos \zeta)}{3} \]
# sky_ovc.rad
# CIE overcast sky (Bz = 1)
!gensky -ang 45 0 -c -b 1
skyfunc glow sky_glow
0
0
4 1 1 1 0

sky_glow source sky
0
0
0
4 0 0 1 180
% oconv sky_ovc.rad > sky_ovc.oct

% rtrace -w -h -I+ -ab 1 \ 
  sky_ovc.oct < samp.inp \ 
  | rcalc -e '$1=$1*0.265+$2*0.670+$3*0.065'

2.434001 [default ad]

7\pi B_z/9 = 2.443451

2.443563 [higher ad]
$$y = \sqrt{x}$$

$$\text{area} = \int_{0}^{1} \sqrt{x} \, dx = 0.6667$$
Typical values commonly used to define the CIE overcast sky

- The CIE overcast sky is defined by its horizontal illuminance, usually given in lux.
- A convenient horizontal illuminance for a (brightish) overcast sky is 10,000lux, e.g. 500 lux corresponds to a 5% DF.
- In gensky we can specify either the zenith radiance (-b option) or the horizontal (diffuse) irradiance (-B option). The second option is perhaps the more direct, and we shall use that for the next rtrace example.
The irradiance that corresponds to this illuminance is $10,000/179 = 55.866 \text{ W/m}^2$.

This conversion factor is the *Radiance* system’s own internal value for luminous efficacy and is fixed at $k_R = 179 \text{ lumens/watt (lm/W)}$.

```sh
gensky -ang 45 0 -c -B 55.866
```

```sh
rtrace -w -h -I+ -ab 1 \sky_ovc.oct < samp.inp | rcalc -e \'
  '$1=($1*0.265+$2*0.670+$3*0.065)*179'
```

9977.17002 [near enough to 10,000 lux]
First, we need to modify the gensky command to produce a 10,000-lux sky. The irradiance that corresponds to this illuminance is 10,000/179 = 55.866 W/m². The line giving the gensky command should now look like this:

```
!gensky -ang 45 0 -c -B 55.866
```

The rest of the file remains as before. Let's now double-check that this sky is indeed what we specified. Run oconv as before, then execute a slightly modified rtrace command:

```
% rtrace -w -h -I+ -ab 1 sky_uni.oct < samp.inp | rcalc -e
'\$1=(\$1*0.265+\$2*0.670+\$3*0.065)*179'
```

The calculation returns the value 9977.17002 which is pretty close to our starting value of 10,000 lux, in fact within 0.3%.

Notice that the irradiance output is now multiplied by 179 to convert it to illuminance (lux).

So far, the only ambient parameter that we've set for the simulation has been -ab; all the other parameters will use the default settings.

Since this scene comprises only a glow source, the parameters that relate directly to the density of the irradiance gradient calculation (i.e., -aa and -ar) will have no effect. Before we go on to more complex (i.e., realistic scenes), we will first have a look at the sky we have generated. To view the sky, start the rview program:

```
% rview -vta -vp 0 0 0 -vd 0 0 1 -vu 0 1 0 -vh 180 -vv 180
```

```
sky_ovc.oct
```

to give an angular fish-eye view of the entire sky. The viewpoint will be useful later on, so save it in a file called `ang180.vf` using the rview command. A false-color image of the sky will show more clearly the CIE overcast sky luminance distribution:

```
% rpict -vf ang180.vf sky_ovc.oct | falsecolor -s 4000 -l cd/m² > ovc_lum.pic
```

The luminance scale in the `falsecolor -s` option was set too close to the approximate zenith luminance of the sky, found either from Eq 5 or by using the trace command in rview. The default label `nits` has been changed to the more familiar cd/m², which means the same thing. The false-color image shows what we expect to see from Eq 4: a brightness distribution depending only on altitude where the zenith luminance is three times that of the horizon.

### 1.4 The Ground “Glow”: An “Upside-Down” Sky

Although it might seem too self-evident to point out, we should remind ourselves that at the horizon the sky “meets” the ground. An actual ground plane of finite extent, say, a disc of radius $r$, will always fall short of an “infinite” horizon, Figure 1. For any given view toward the horizon, we can

2. This conversion factor is the Radiance system's own value for luminous efficacy and is fixed at $K_R = 179$ lumens/watt (lm/w). This should not be confused with the more usual daylighting value, which can be anywhere between 50 and 150 lm/w depending on the type of sky or light considered.

---

This is what we can see if we add a ground plane

![Diagram](image.png)
Ground glow - an upside down sky

```
skyfunc glow ground_glow  
0
0
4 1 1 1 0 

ground_glow source ground 
0
0
4 0 0 -1 180
```
This creates a ‘seamless luminous envelope’ around our scene.

Fig 6.5 Rendering with Radiance
Predicting internal illuminance

[No ground plane in this example]
/bin/csh -f
# loop through ab

foreach ab (1 2 3 4 5)
  echo "Ambient bounces" $ab

# Calculate DF
  rtrace -w -h -I+ -ab $ab -aa 0.2 -ad 512 \ 
    -as 0 -ar 128 scene.oct \ 
    < samp1.inp | rcalc -e\ 
    '$1=($1*0.265+$2*0.670+$3*0.065*179/10000*100'

end
Fig 6.7 Rendering with Radiance
Fig 6.7 Rendering with Radiance
Questions?
Adding complexity

• Now we add a ground plane and a nearby building to our simple scene. We model the ground plane as a disc of, say, radius 20 meters, centered on the origin.

• External obstruction is a nearby building positioned so that it faces the room window and obscures much of the view of the sky from inside the room. The DF predictions are repeated as before, only now we increase the maximum -ab to 7.
Ground plane

Ground glow

Fig 6.8 Rendering with Radiance
For -ab 3 ray samples ground plane radiance calculated from sky brightness

For -ab 2 ray samples ground glow radiance

Fig 6.9 Rendering with Radiance
2.2 Views from the DF Plane

It often helps to visualize the scene from one or more viewpoints along the DF plane. Choose a point in the DF plane, say, near the window, and generate a view looking directly upward - use the interactive previewer `rview`. Set the view type to hemispherical (`h`) and the view angles to 180 degrees. As the image resolution gradually improves, you will see a hemispherical projection view of the sky through the window. Set `-av` to some value to reveal the other surfaces. This makes it easier to understand the image, but what we are really interested in is the view of the sky. Compare the views with and without the external obstruction Figure 5. The impact of the nearby building on internal light levels can be roughly estimated just from these images. Since the building obscures about half the view of the sky, the DF values will be approximately halved. This is a worst-case guess—it will, of course, depend on the facade reflectance. Examining a scene in this way will help you to appreciate the luminous environment "from a light meter's point of view."

2.3 The Ambient Exclude/Include Options

It is possible to limit the number of surfaces that participate directly in the indirect irradiance calculation. By limiting the scope of the ambient calculation, we can make significant savings in simulation time. This is achieved by telling `rtrace` not to include certain named material modifiers in the indirect calculation. Instead, the named materials will receive the constant ambient-value approximation. There is a complementary option called ambient include. With this option, only the named materials participate in the indirect calculation; the rest receive constant ambient-value approximation. We should take care to exclude only those materials that play no major part in the illumination of the space. The `rtrace` manual page explains how the options are enabled.

---

**Photocell’s ‘view’ from the front near the window**

![Previous](image1.png) ![With obstruction](image2.png)

---

Fig 6.12 Rendering with Radiance
Fig 6.10 Rendering with Radiance
Fig 6.10 Rendering with Radiance
The ambient resolution parameter [ar]

The art of sfumato in Radiance
Example Scenario

The scenario used for these examples is a precariously positioned box intersecting with a horizontal plane, Figure 1. Illumination is provided by a CIE standard overcast sky (i.e. a glow source). There is no ground glow in this model and the view parameters are set so that the dark space above the plane is in actuality below the horizon, i.e. the sky is present but not seen. Because the only illumination is a glow source, we need at least \(-ab\) 1 to be able to compute some luminance for the visible surfaces in this scene (the constant ambient value is set to zero). The arrangement of the box above the plane makes for some, potentially, subtle shading on the horizontal plane. The renderings A and B in Figure 1 were generated using slightly different ambient parameter settings. The parameters common to both renderings were:

- \(-ad\) 2048 -\(as\) 128 -\(ab\) 1 -\(aa\) 0.15 -\(av\) 0 0 0

The \(-ar\) parameter was set to 4 and 64 for renderings A and B respectively. Examine closely the shading under the box in both renderings (zoom in on the PDF). Notice how rendering B (\(-ar\) 64) better achieves the shading pattern we might expect from this arrangement. Renderings A' and B' were created with the makers generated by \textit{genambpos} in the scene description.

Now we clearly see what we previously suspected - the density of hemispherical sampling points was greater for B. Also evident is the way the density increases closer in to the 'crease' between the box and the plane. Whilst \(-ar\) 4 prevents the ambient sampling from
Example Scenario

The scenario used for these examples is a precariously positioned box intersecting with a horizontal plane, Figure 1. Illumination is provided by a CIE standard overcast sky (i.e. a \textit{glow} source). There is no ground glow in this model and the view parameters are set so that the dark space above the plane is in actuality below the horizon, i.e. the sky is present but not seen. Because the only illumination is a \textit{glow} source, we need at least \(-\text{ab} 1\) to be able to compute some luminance for the visible surfaces in this scene (the constant ambient value is set to zero). The arrangement of the box above the plane makes for some, potentially, subtle shading on the horizontal plane. The renderings A and B in Figure 1 were generated using slightly different ambient parameter settings. The parameters common to both renderings were:

\[-\text{ad} 2048 \quad \text{as} 128 \quad \text{ab} 1 \quad \text{aa} 0.15 \quad \text{av} 0 \ 0 \ 0\]

The \(-\text{ar}\) parameter was set to 4 and 64 for renderings A and B respectively. Examine closely the shading under the box in both renderings (zoom in on the PDF). Notice how rendering B (\(-\text{ar} 64\)) better achieves the shading pattern we might expect from this arrangement. Renderings A' and B' were created with the makers generated by \textit{genambpos} in the scene description.

Now we clearly see what we previously suspected - the density of hemispherical sampling points was greater for B. Also evident is the way the density increases closer in to the 'crease' between the box and the plane. Whilst \(-\text{ar} 4\) prevents the ambient sampling from

2. Direct lighting from three sources was used to illuminate the scene with makers.

Figure 1. Shading and sampling locations under a glow sky

B
A
A'
B'

99 locations 563 locations
Example Scenario

The scenario used for these examples is a precariously positioned box intersecting with a horizontal plane, Figure 1. Illumination is provided by a CIE standard overcast sky (i.e. a glow source). There is no ground glow in this model and the view parameters are set so that the dark space above the plane is in actuality below the horizon, i.e. the sky is present but not seen. Because the only illumination is a glow source, we need at least $-ab 1$ to be able to compute some luminance for the visible surfaces in this scene (the constant ambient value is set to zero). The arrangement of the box above the plane makes for some, potentially, subtle shading on the horizontal plane. The renderings A and B in Figure 1 were generated using slightly different ambient parameter settings. The parameters common to both renderings were:

$-ad 2048$  $-as 128$  $-ab 1$  $-aa 0.15$  $-av 0 0 0$

The $-ar$ parameter was set to 4 and 64 for renderings A and B respectively. Examine closely the shading under the box in both renderings (zoom in on the PDF). Notice how rendering B ($-ar 64$) better achieves the shading pattern we might expect from this arrangement. Renderings A' and B' were created with the makers generated by genambpos in the scene description.

Now we clearly see what we previously suspected - the density of hemispherical sampling points was greater for B. Also evident is the way the density increases closer in to the 'crease' between the box and the plane. Whilst $-ar 4$ prevents the ambient sampling from...
Example Scenario

The scenario used for these examples is a precariously positioned box intersecting with a horizontal plane, Figure 1. Illumination is provided by a CIE standard overcast sky (i.e. a glow source). There is no ground glow in this model and the view parameters are set so that the dark space above the plane is in actuality below the horizon, i.e. the sky is present but not seen. Because the only illumination is a glow source, we need at least \(-ab \) to be able to compute some luminance for the visible surfaces in this scene (the constant ambient value is set to zero). The arrangement of the box above the plane makes for some, potentially, subtle shading on the horizontal plane. The renderings A and B in Figure 1 were generated using slightly different ambient parameter settings. The parameters common to both renderings were:

\[-ad 2048 -as 128 -ab 1 -aa 0.15 -av 0 0 0\]

The \(-ar\) parameter was set to 4 and 64 for renderings A and B respectively. Examine closely the shading under the box in both renderings (zoom in on the PDF). Notice how rendering B (\(-ar 64\)) better achieves the shading pattern we might expect from this arrangement. Renderings A' and B' were created with the makers generated by \texttt{genambpos} in the scene description.

Now we clearly see what we previously suspected - the density of hemispherical sampling points was greater for B. Also evident is the way the density increases closer in to the 'crease' between the box and the plane. Whilst \(-ar 4\) prevents the ambient sampling from

\[S_{min} = \frac{D_{max} \times aa}{ar}\]
The overture calculation

• Execute the simulation as normal, however save the ambient file (i.e. values determined from hemispherical sampling), but don’t keep the image.

• Then, redo the simulation using the saved ambient file and the same ambient parameters.
without overture
with overture
without overture
with overture
Estimate

Irradiance

Pixel position along ‘scanline’
Irradiance

Pixel position along ‘scanline’
-af keep.af

Irradiance

Pixel position along 'scanline'

> /dev/null
-af keep.af

> image.hdr

Irradiance

Pixel position along 'scanline'
Why overture?

• In a ‘one-off’ simulation, *Radiance* has to sometimes use **extrapolation** to estimate values between sampling locations as it progresses from one sampling point to the next.

• With an overture calculation, the ambient file (aka ambient cache) is first populated with values. Thereby ensuring that - when reused to create an image - *Radiance* uses **interpolation** between already calculated values rather than less reliable extrapolation. Negligible overhead in overall computation time.
Parameter settings and CPU costs
% rtrace -defaults

-av 0.0 0.0 0.0 # ambient value
-aw 0 # ambient value weight
-ab 0 # ambient bounces
-aa 0.100000 # ambient accuracy
-ar 256 # ambient resolution
-ad 1024 # ambient divisions
-as 512 # ambient super-samples
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Change</th>
<th>Potential CPU overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ad</strong> ambient divisions</td>
<td>512 to 1024 i.e. doubling</td>
<td>x 2</td>
</tr>
<tr>
<td><strong>aa</strong> ambient accuracy</td>
<td>0.2 to 0.1 i.e. halving</td>
<td>x 4</td>
</tr>
<tr>
<td></td>
<td>no interpolation 0</td>
<td>x a lot?</td>
</tr>
<tr>
<td><strong>ar</strong> ambient resolution</td>
<td>32 to 64 i.e. doubling</td>
<td>x 4</td>
</tr>
<tr>
<td></td>
<td>unlimited resolution 0</td>
<td>x a lot?</td>
</tr>
<tr>
<td>Parameter</td>
<td>Change</td>
<td>Potential CPU overhead</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td><strong>ad</strong> ambient divisions</td>
<td>512 to 1024 i.e. doubling</td>
<td>x 2</td>
</tr>
<tr>
<td><strong>aa</strong> ambient accuracy</td>
<td>0.2 to 0.1 i.e. halving</td>
<td>x 4</td>
</tr>
<tr>
<td></td>
<td>no interpolation 0</td>
<td>x a lot?</td>
</tr>
<tr>
<td><strong>ar</strong> ambient resolution</td>
<td>32 to 64 i.e. doubling</td>
<td>x 4</td>
</tr>
<tr>
<td></td>
<td>unlimited resolution 0</td>
<td>x a lot?</td>
</tr>
<tr>
<td>Parameter</td>
<td>Change</td>
<td>Potential CPU overhead</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>ad ambient divisions</td>
<td>512 to 1024 i.e. doubling</td>
<td>x 2</td>
</tr>
<tr>
<td>aa ambient accuracy</td>
<td>0.2 to 0.1 i.e. halving</td>
<td>x 4</td>
</tr>
<tr>
<td></td>
<td>no interpolation 0</td>
<td>x a lot?</td>
</tr>
<tr>
<td>ar ambient resolution</td>
<td>32 to 64 i.e. doubling</td>
<td>x 4</td>
</tr>
<tr>
<td></td>
<td>unlimited resolution 0</td>
<td>x a lot?</td>
</tr>
</tbody>
</table>
mkillum - hunt twice to avoid having to search wide only to find small openings that lead to the light.
Illumination from outside

“window”

Look ‘everywhere’ to find the light from the sky through window

Mkillum “window” accounts for illumination from outside

Look first at the window because we know \textit{a priori} that it is the ‘source’ of illumination
Step 1

• Create the octree as normal.

  • It is important for the **mkillum** process that follows to be able to identify the windows that need to be treated.

• Use **mkillum** to compute the *window output distribution* i.e. a similar specification to that used to characterise the light output distribution of a luminaire. Ambient settings as required.

  • A new window is created using the **illum** material.
Step 2

- Recreate the octree replacing the window with the new description created by `mkillum`.
  - Replace `window.rad` with `mkiwin.rad`.
- Run `rpict` or `rtrace` on the new octree with ambient settings as required.
oconv room.rad window.rad sky.rad \ out.rad > scene.oct

mkillum [options] scene.oct < window.rad > \ mkiwin.rad

oconv office.rad mkiwin.rad sky.rad \ out.rad > mkiscene.oct

rpict / rtrace [options] mkiscene.oct
ab 1

MKI ab 1 ; ab 0
ab 2

MKI ab 1 ; ab 1
Issues with mkillum

• Many windows can results in too many light sources.
• Nearby external obstructions - subdivide window.
• CAD input - rectangles, surface normals.
Modelling venetian blinds using mkillum

A five-sided illum box encloses the blinds on the inside.
Cases where the **mkillum** approach doesn’t work

![Fig 13.8 Rendering with Radiance](image)

- Curved mirror louvres
- Light pipes
Questions?
Thank you