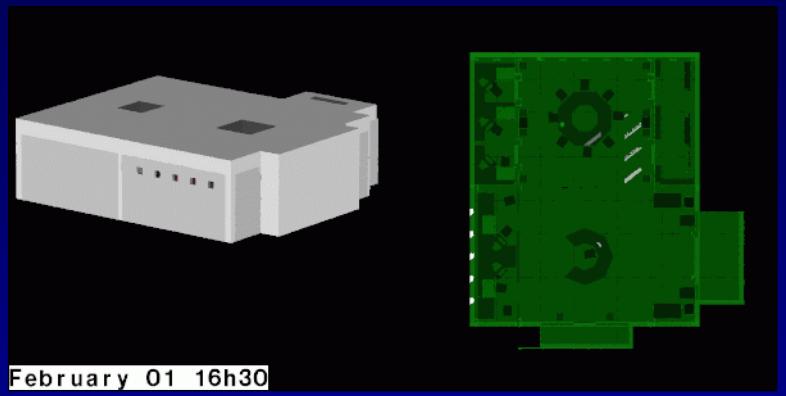
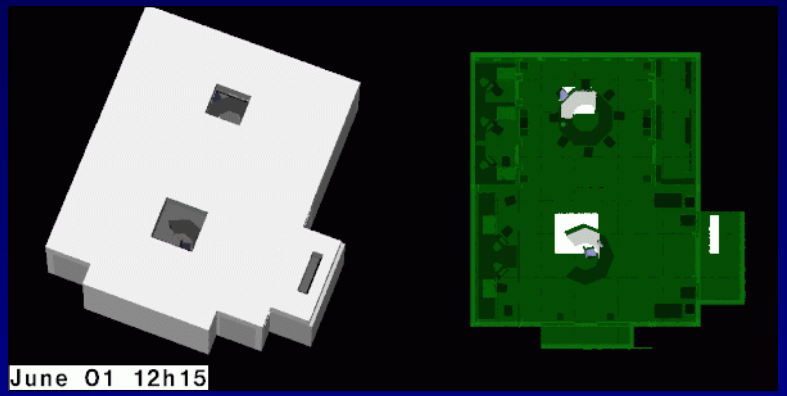
Radiance Course Day 2

MORE ADVANCED TECHNIQUES

Materials & Scripting





Dr. John Mardaljevic

jm@dmu.ac.uk

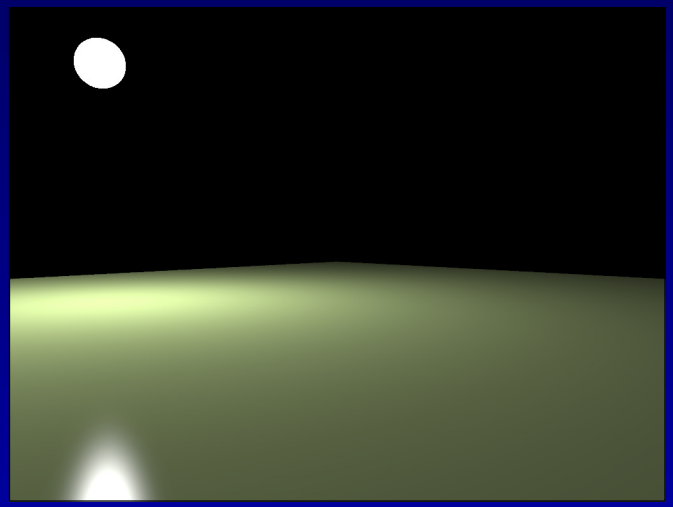
Institute of Energy and Sustainable Development
De Montfort University
Leicester, UK

http://www.iesd.dmu.ac.uk/~jm

Materials And Modifiers

Here's a simple scene:

Light source



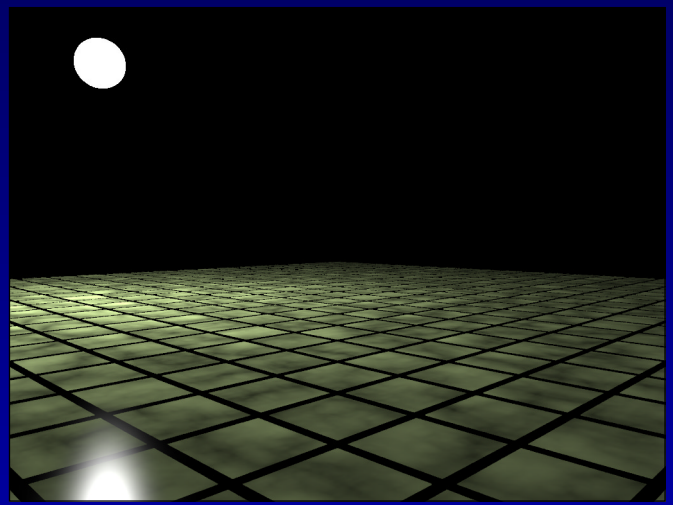
Polygon

```
void plastic plain
0
0
5 .33 .42 .18 .05 .05
```

Materials And Modifiers (cont.)

But now the polygon is “marble”:

Light source



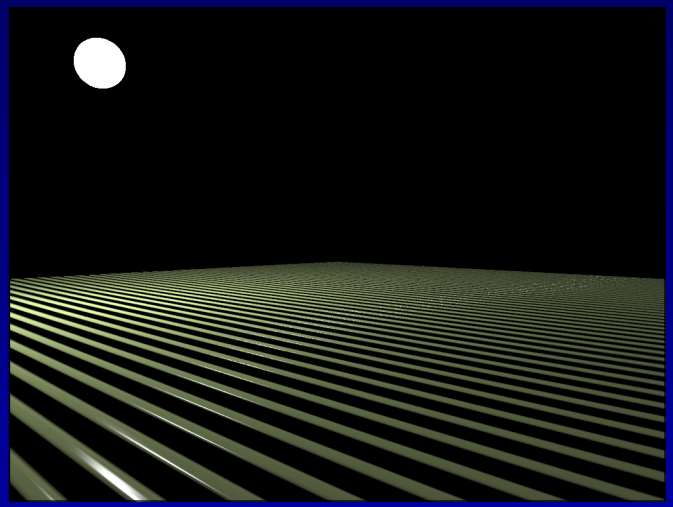
Polygon

```
void brightfunc testfcn
2 tilebright marbletile.cal
0
4 .1 .9 .9 .95
#
testfcn plastic marble
0
0
5 .33 .42 .18 .05 .05
```

MATERIALS AND MODIFIERS (CONT.)

And now it's "corrugated" (but still flat):

Light source



Polygon

```
void texfunc testfcn
4 xcor ycor zcor cor_iron.cal
0
1 .2
#
testfcn plastic corrug
0
0
5 .33 .42 .18 .05 .05
```

MARBLE

Here, **`brightfunc`** is applied to modify the diffuse reflectance of the 'base' material. The "rules" for the modification are taken from the function file **`marbletile.cal`** which is "called" in the scene file. Notice the specular component is not affected by applying this modifier.

The modifier works in two ways: a regular time pattern and a random marble-like pattern. The arguments **`.1 .9 .9 .95`** control the: mortar width, x and y dimension of tile and the degree of "marbleness".

CORRUGATED

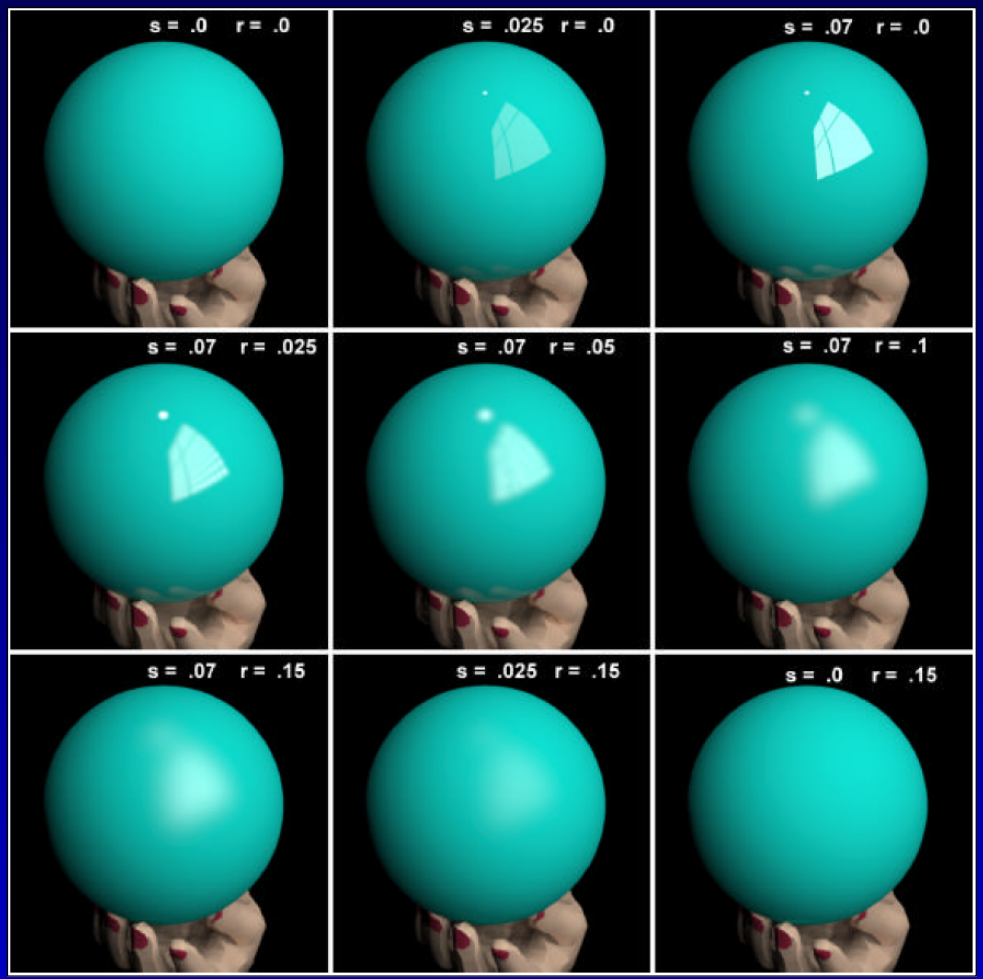
Here, **`texfunc`** is applied to modify the surface normal of the ‘base’ polygon. The polygon however remains (perfectly) flat. So, you cannot “hide” a small object in the corrugations. The “rules” for the modification are taken from the function file **`cor_iron.cal`** which is “called” in the scene file. Notice this modifier affects the diffuse and specular components.

There is only one argument which is the number of cycles per unit distance.

More Use Of These Modifiers

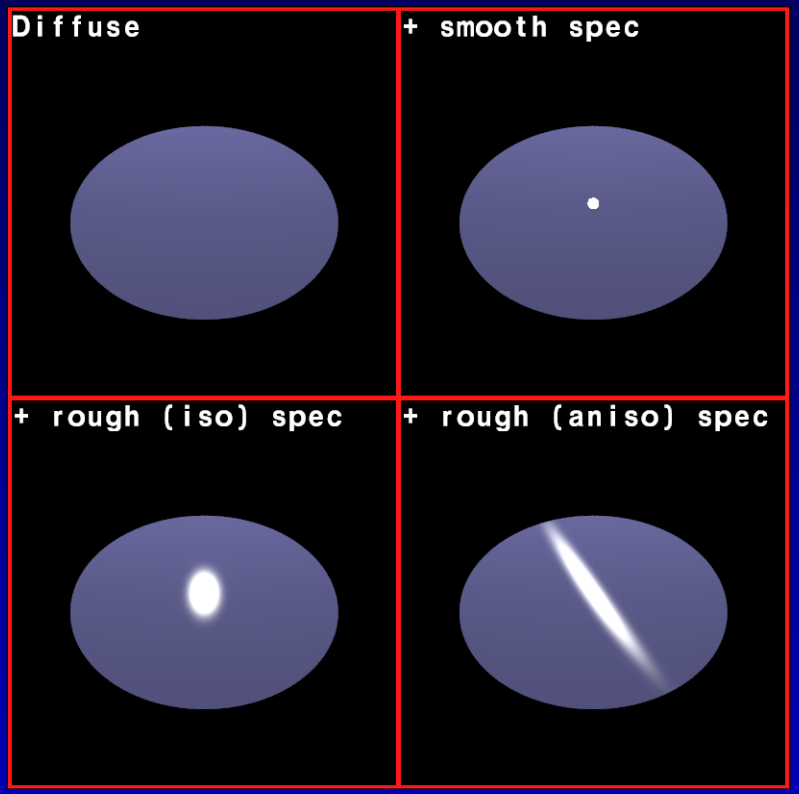
Roughness And Specularity

Many real materials have some specular component, and very few are perfectly “smooth”.



Isotropic And Anisotropic Roughness

Roughness where the orientation figures in how it appears (e.g. brushed metal) is called anisotropic.



SIMPLE SCRIPT TO AUTOMATE...

...the generation of renderings based on varying material properties taken from a list.

Here, we wish to produce a series of renderings to compare the effect of subtle (or gross) changes to one of the material properties.

We want to generate any number of renderings. And we're really lazy, so it must be as effortless as possible.

Here's how we do it...

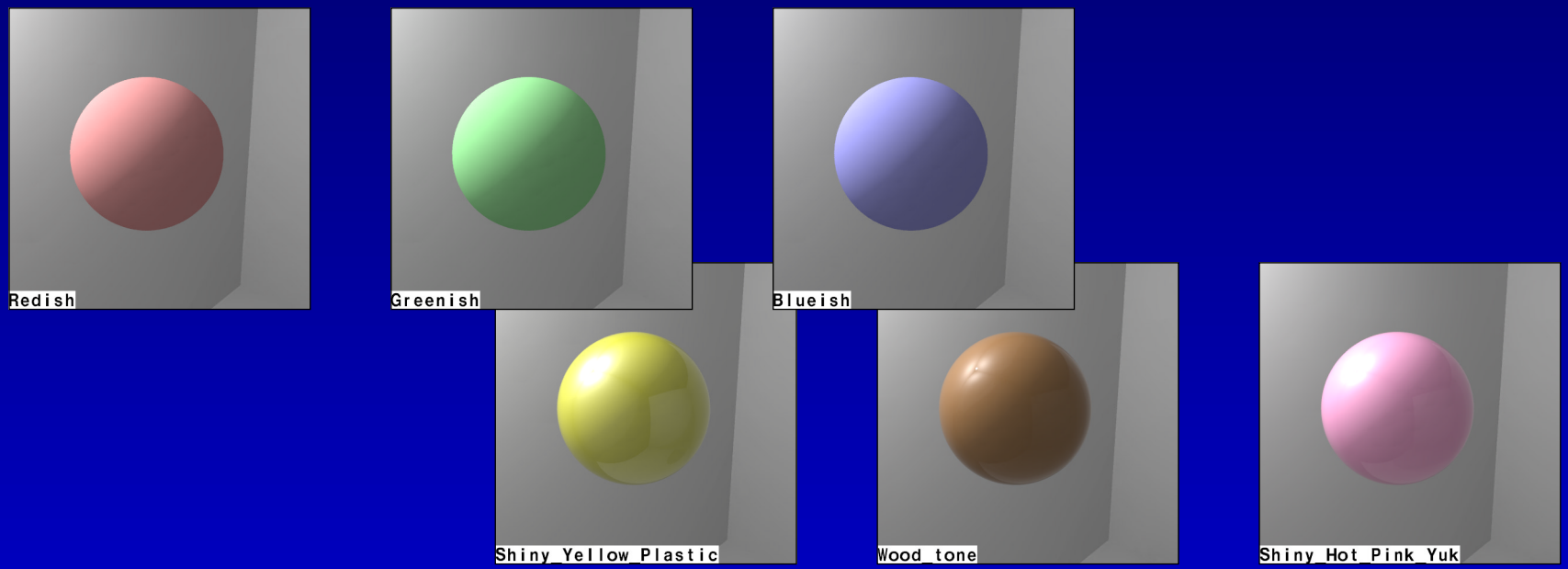
VARY MATERIAL PROPERTIES SCRIPT

```
#!/bin/csh -f
set n_mats = `cat mat_list | wc -l`
@ i = 1
while ($n_mats >= $i)
   set args = `sed -n "$i"p mat_list`
   set mat_name = $args[6]
   cp test_mat_part test_mat.rad
   echo $args[1-5] >> test_mat.rad
   echo "Rendering $mat_name ..."
        # Octree -> render -> filter -> composite with label
   oconv test_mat.rad test_obj.rad testroom \
   | rpict -vf v1.vf -ab 1 -ad 4096 -av .5 .5 .5 -x 1024 -y 1024 \
   | pfilt -1 -e -1.2 -x /2 -y /2 \
   | pcompos - 0 0 '\!psign -h 32 '"$mat_name" 0 0 >! $mat_name.pic
    @ i++
end
```

The Material Properties List

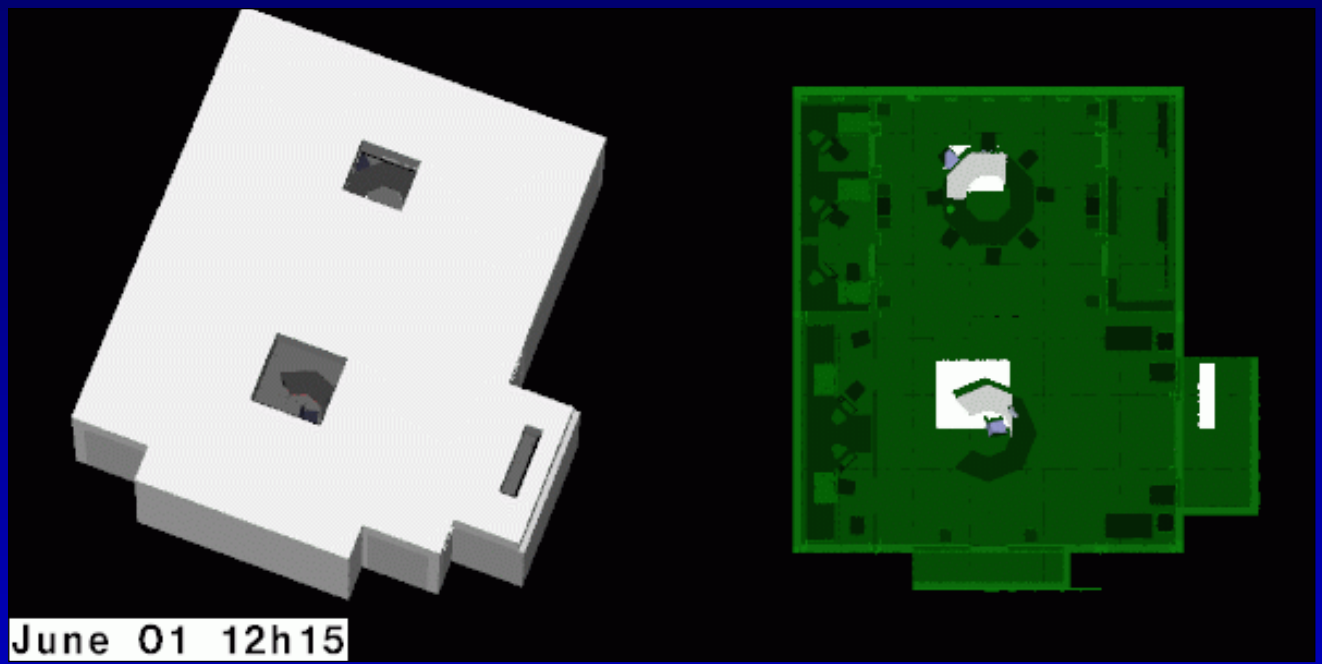
```
0.7 0.3 0.3 0 0 Redish
0.3 0.7 0.3 0 0 Greenish
0.3 0.3 0.7 0 0 Blueish
0.68 0.68 0.002 0.2 0 Shiny_Yellow_Plastic
.3 .15 .05 .04 0.0 Wood_tone
1.0 0.41 0.70 0.1 0 Shiny_Hot_Pink_Yuk
```

Executing the script using the list above produces:



Script To Generate "Movie" Sequence

This script evolved from the example in the *Radiance* book. As well as a "static" view from above, it also generates a view from the direction of the sun.



PREPARATION

For the view from the sun, it helps matters if the scene is centred on the origin. In a movie sequence, the building will appear to pivot about the origin.

Determine where you want the pivot point to be, then use `xform` to transform that location of the scene to the origin.

De-construction of the script - see hardcopy.

Show movie.

Further Investigation Of The Ambient Calculation And Use Of `mkillum`

This section will make use of ‘live’ simulations to understand the effect of changing key parameters.

First, a recap of the ambient calculation (taken from the Primer).

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.

When to Use `light` and `glow` Sources

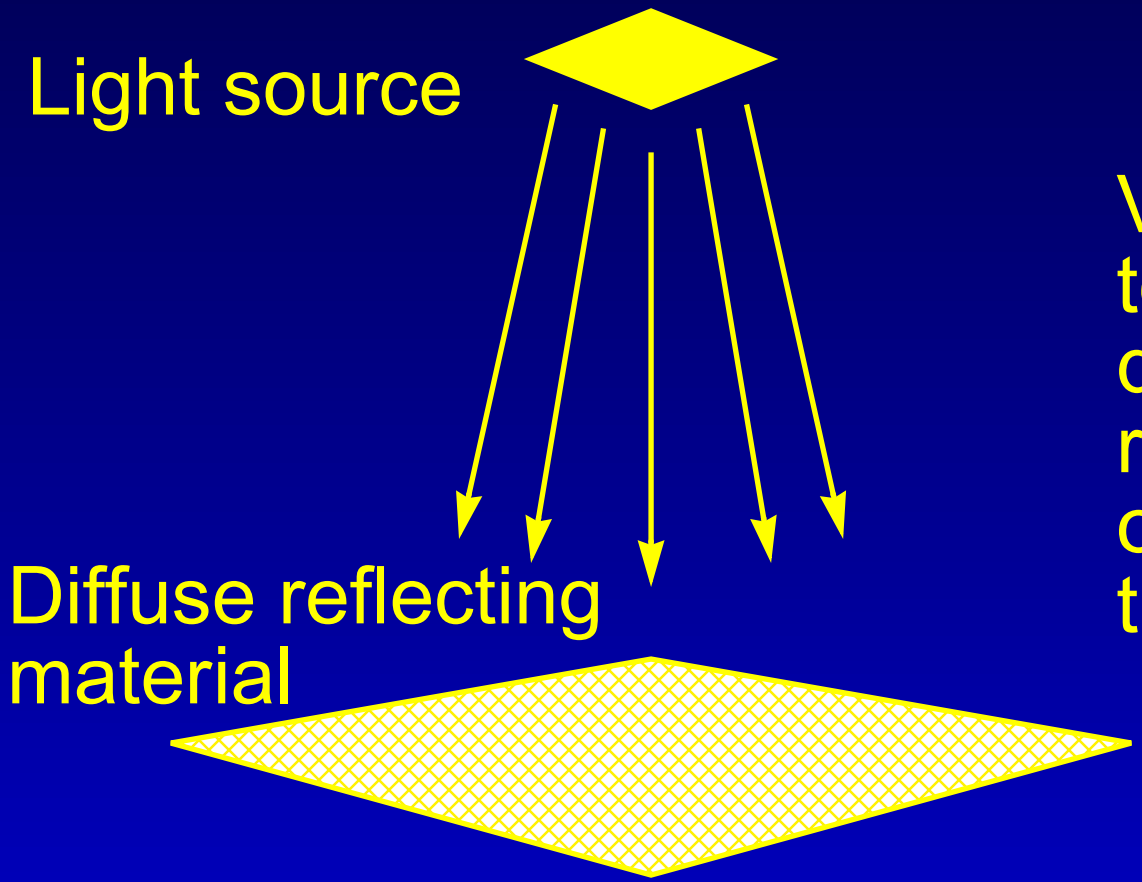
We use the material `light` for important sources of illumination, e.g. electric luminaire, the sun. These participate in the direct calculation of illumination.

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.

> Examples to show how they behave in *Radiance*.

Test Scene - Two Polygons

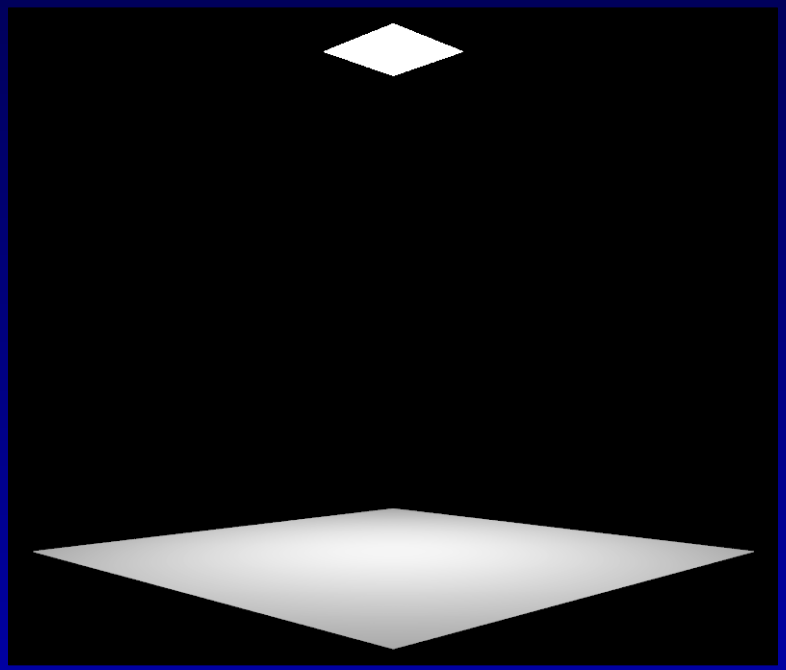
Our test scene comprises two polygons. One is an emitter of light which shines onto the other.

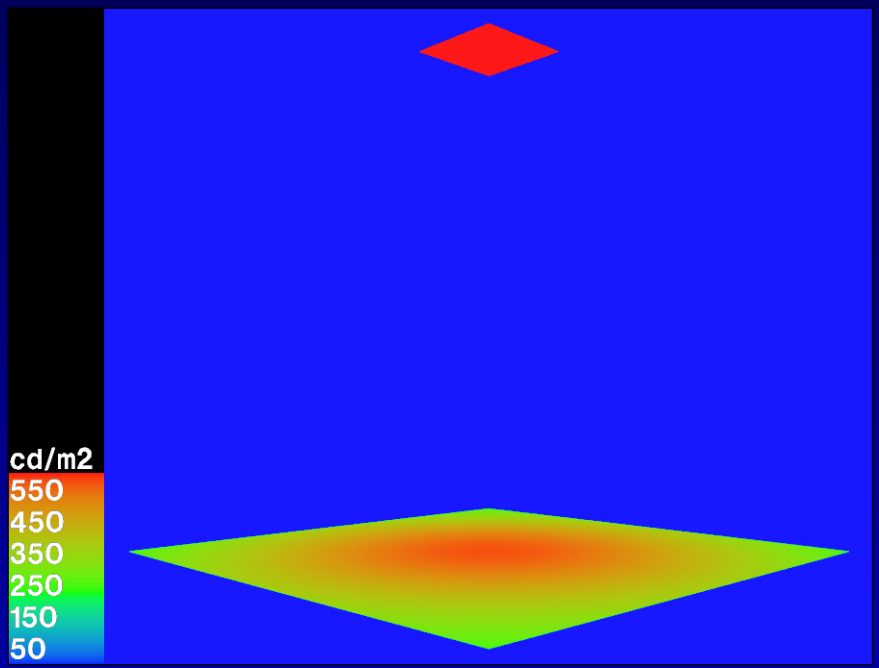


View parameters set to see source shining downwards and the resulting illumination on the upper-side of the polygon below.

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.

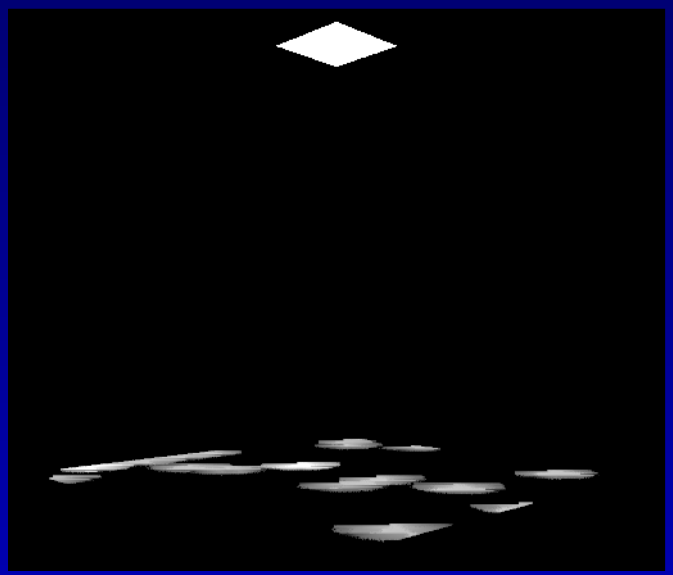




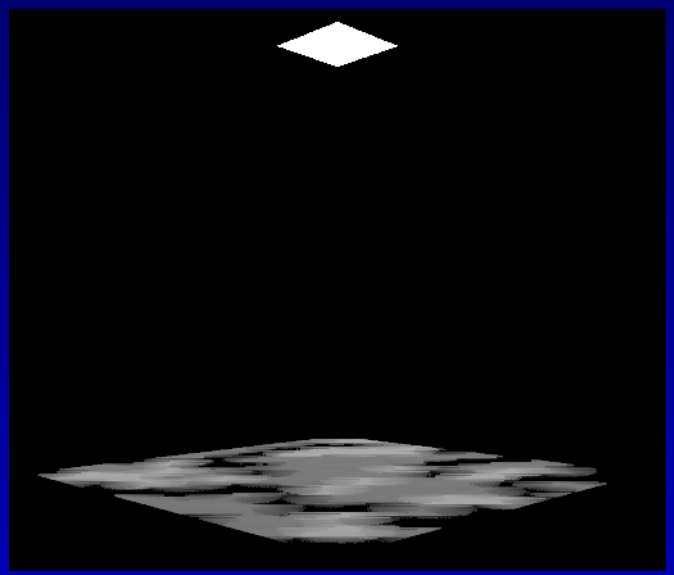
The reflecting polygon is evenly illuminated by the light source.
This is clearly revealed in the false colour image.
Note, "`-ab 0`" setting used, i.e. inter-reflection calculation turned off.

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 rays (the `ad` parameter) to see the effect.



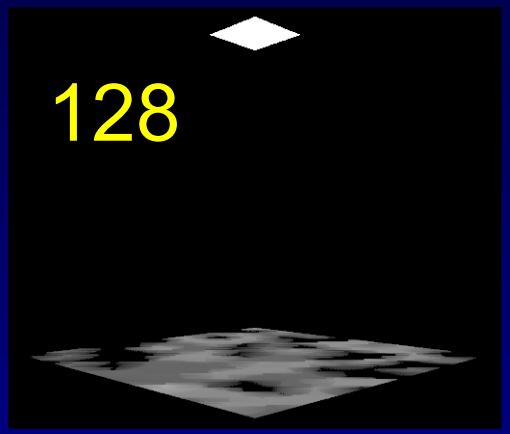
`-ad 32`

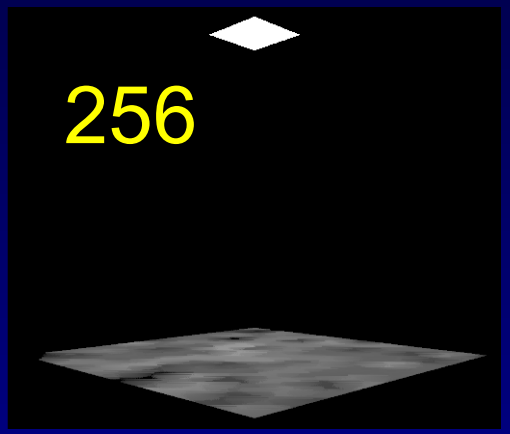


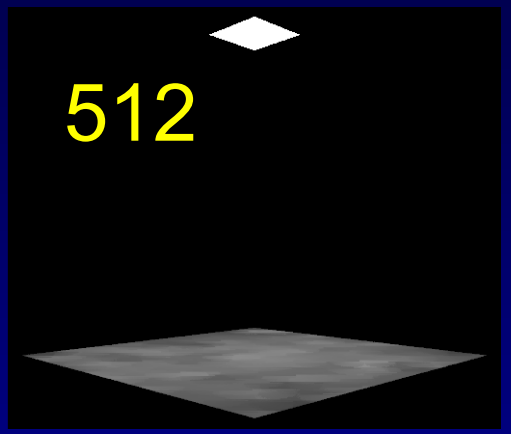
`-ad 64`

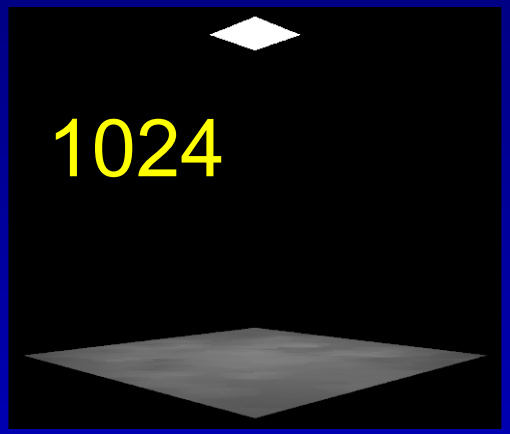
INCREASING THE NUMBER OF ad RAYS...

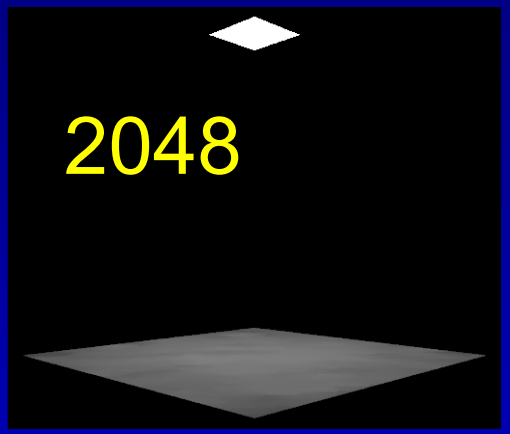
...does produce smoother renderings.

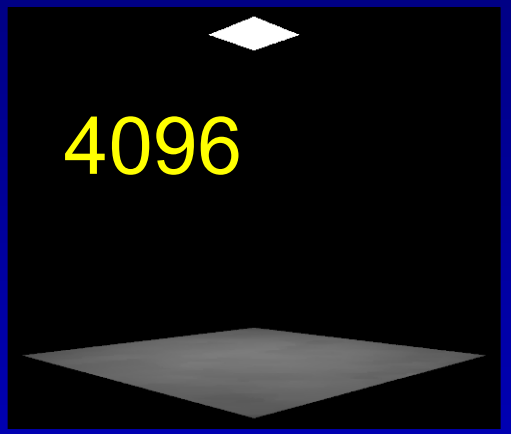








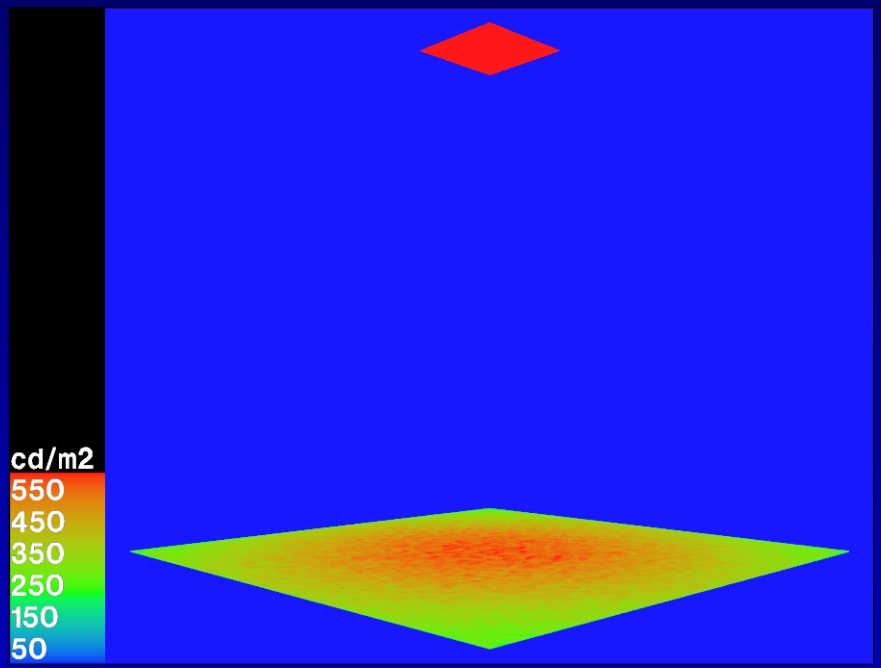




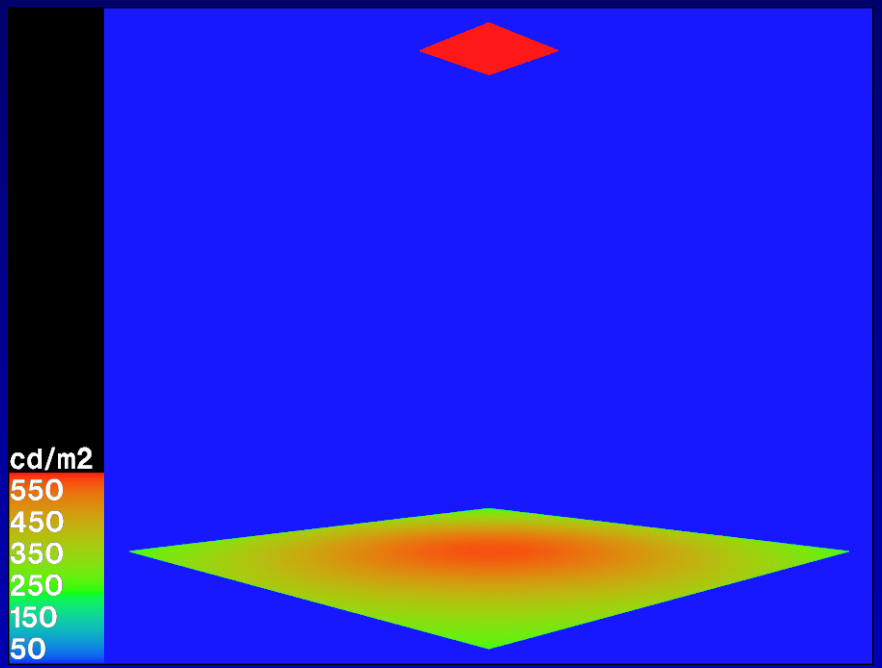
But Even With `-ad 4096`...

...the illumination from the `glow` material is not quite as smooth as that from the `light` material.

`glow`

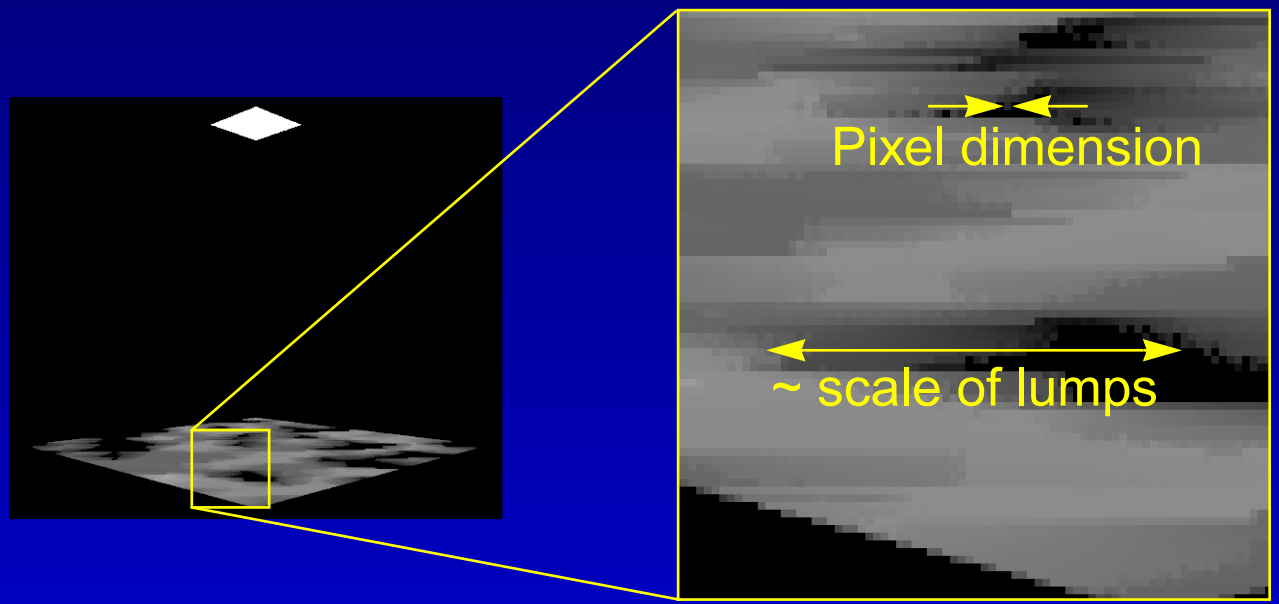


`light`



WHY ARE THE `glow` RENDERINGS LUMPY?

With a small `glow` source, sometimes the hemispherical sampling ‘finds’ the source, and sometimes it doesn’t. There is a random (or stochastic) component to the ray direction. Notice, the lumpiness occurs at scales much larger than the effective dimension of a pixel.



What's the Significance of Big Lumps?

Well, it 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 x 128 = 3,200,000 rays; or,

25,000 x 4096 = 102,400,000 rays.

Where Interpolation Took Place

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

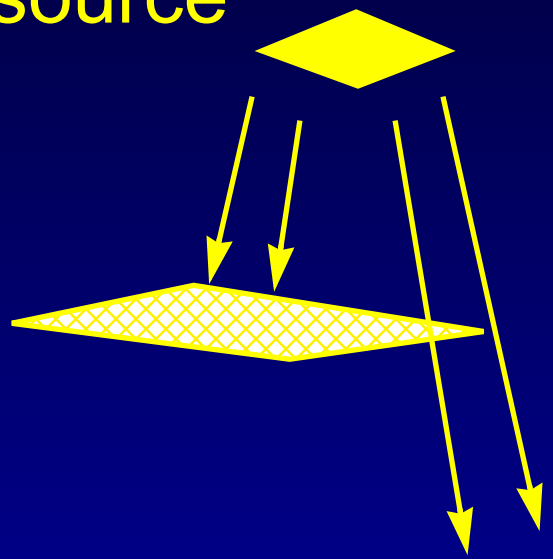
-ad 4096

What Have We Discovered So Far?

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

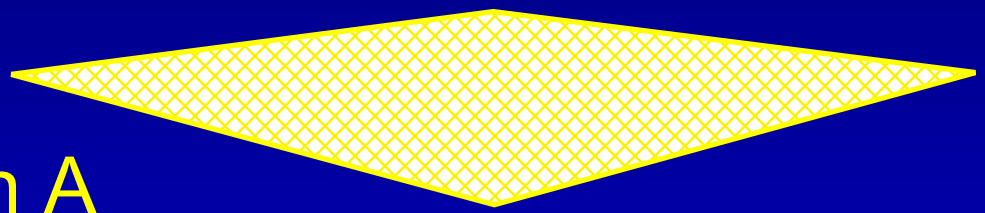
In the previous example, the scene didn't allow for inter-reflection. Here, we modify the scene by adding an occluding polygon to see how hemispherical sampling is used to compute indirect or (inter-reflected) light.

Scene With Occluding Polygon



Polygon B
(occluding)

Occluding polygon shades lower polygon from light source.

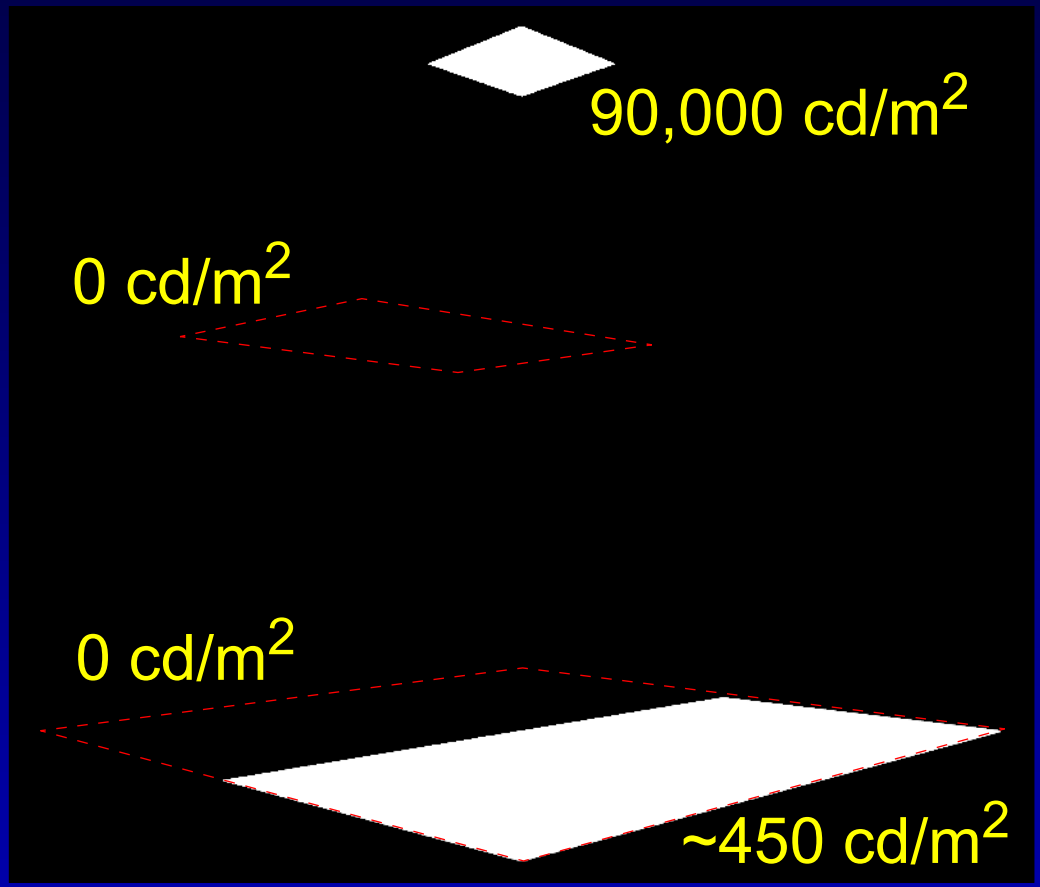


Polygon A

RENDERING FOR OCCLUDING SCENE `-ab 0`

Underside of polygon B not illuminated

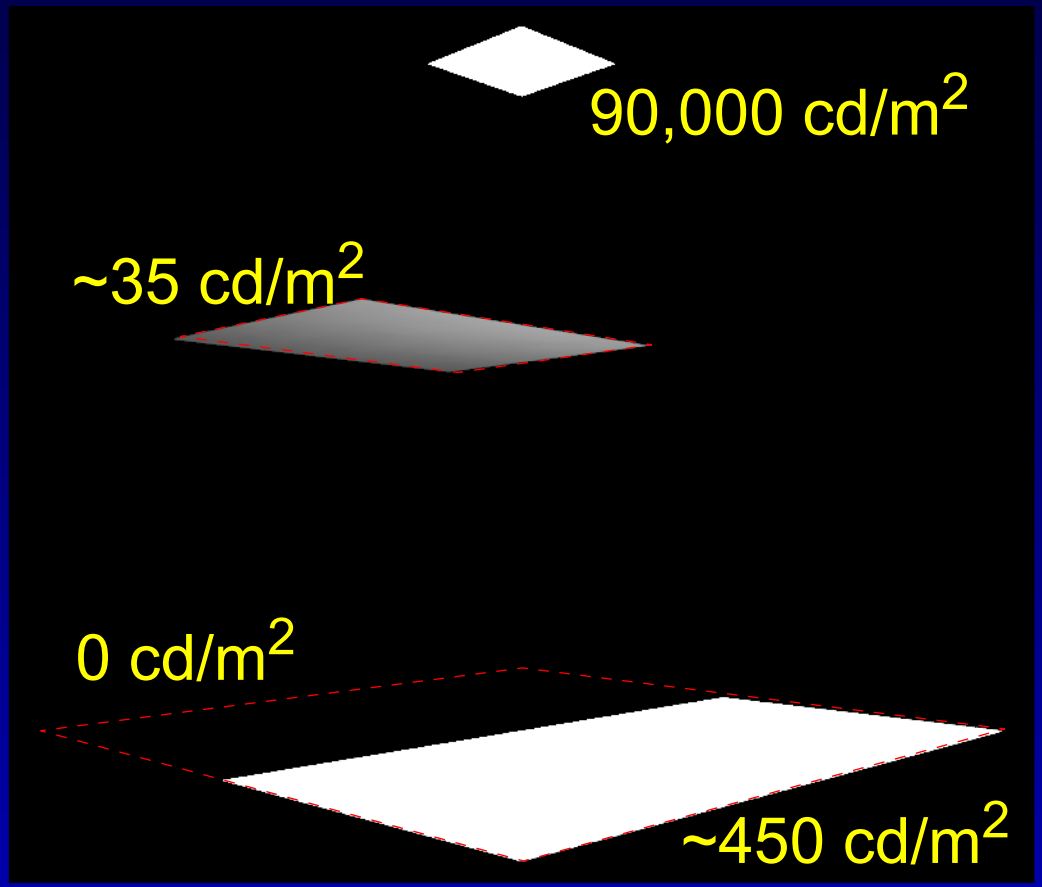
Topside of polygon A half in shade



FOR -ab 1

Underside of polygon B now illuminated

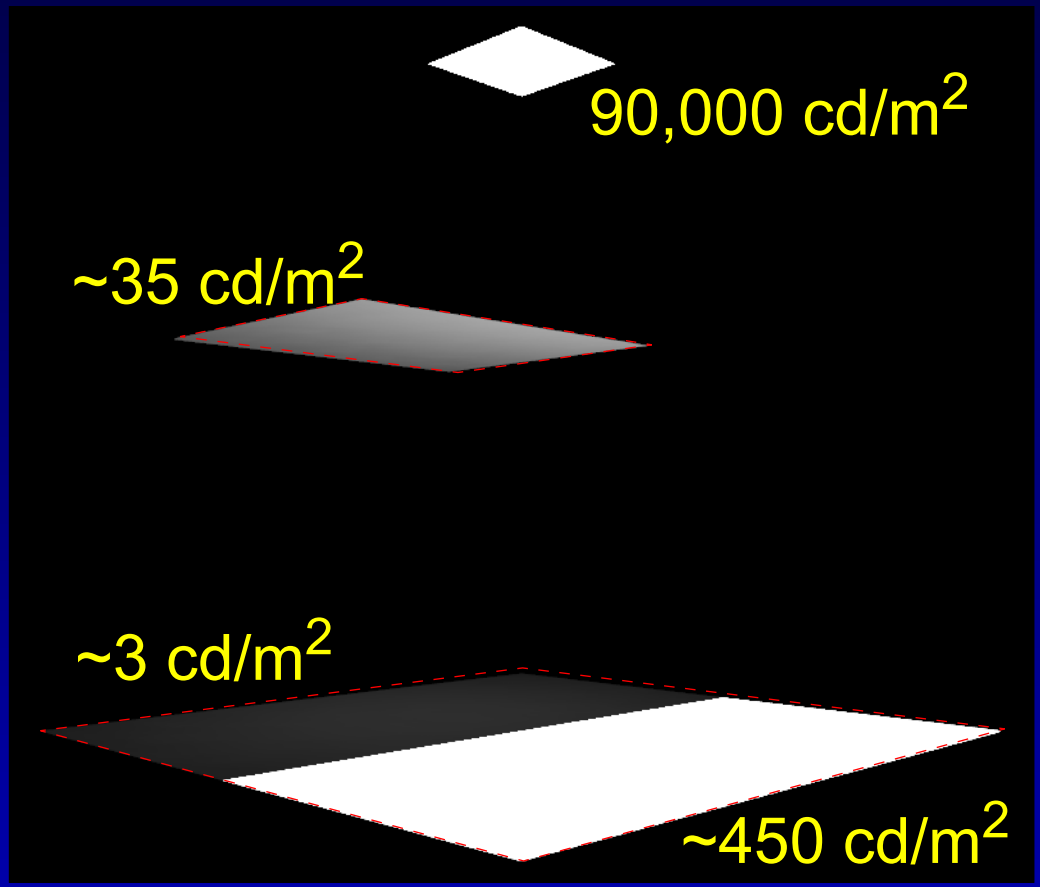
Topside of polygon A still half in shade



AND LASTLY FOR `-ab 2`

Underside of polygon B now illuminated

Shaded half of polygon A now illuminated by reflected light from polygon B



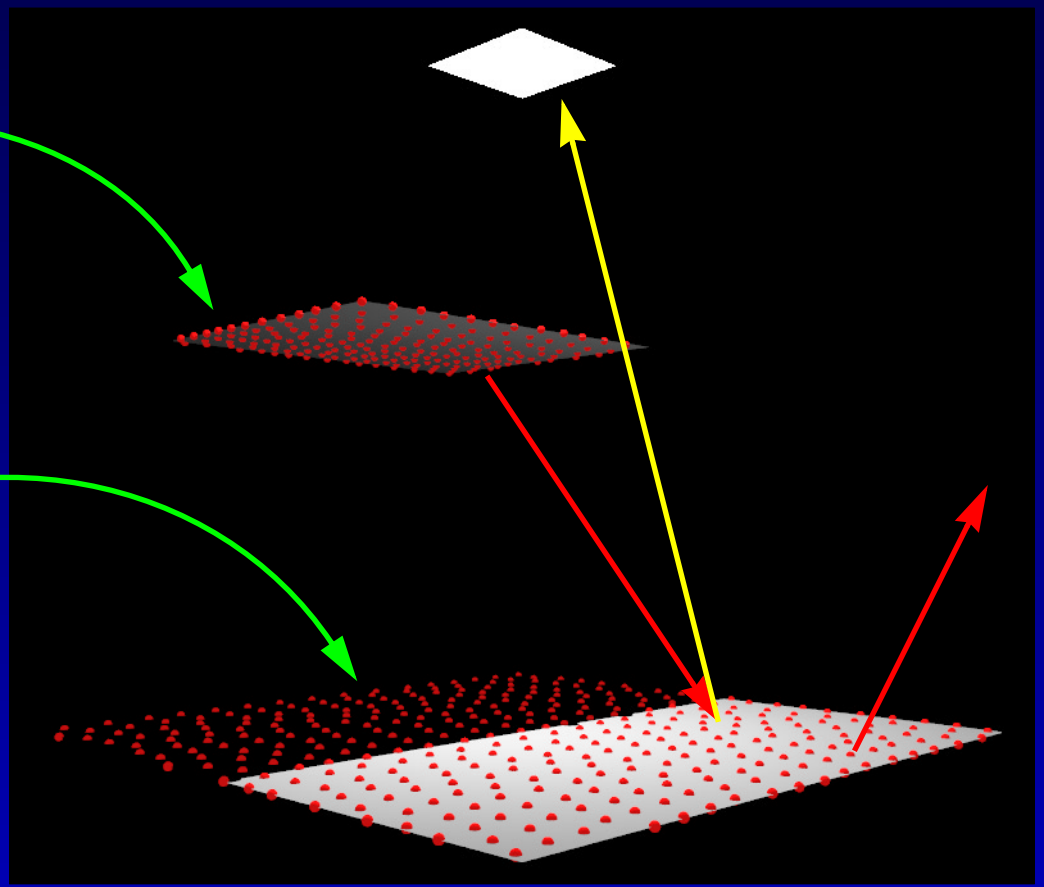
HEMISPHERICAL SAMPLING TOOK PLACE...

...at these locations for **`-ab 1`**

Level 0

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)

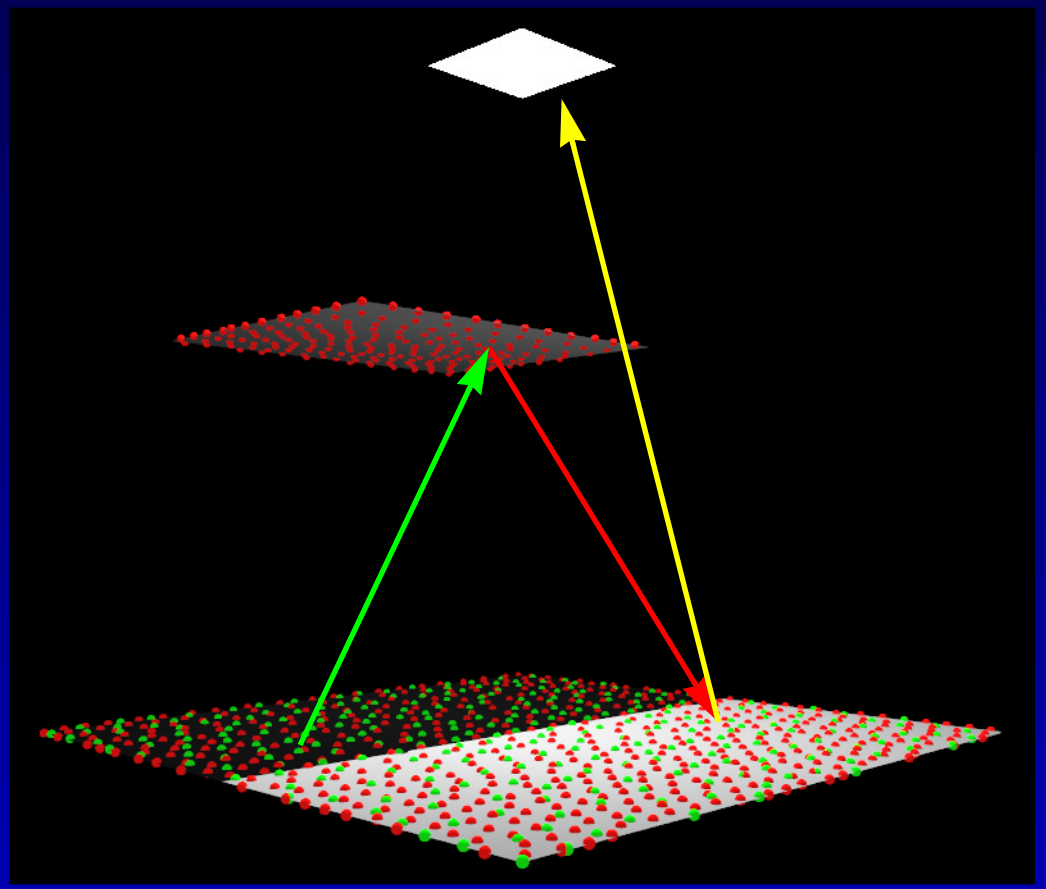


HEMISPHERICAL SAMPLING TOOK PLACE...

...at these locations for **`-ab 2`**

Level 0
Level 1

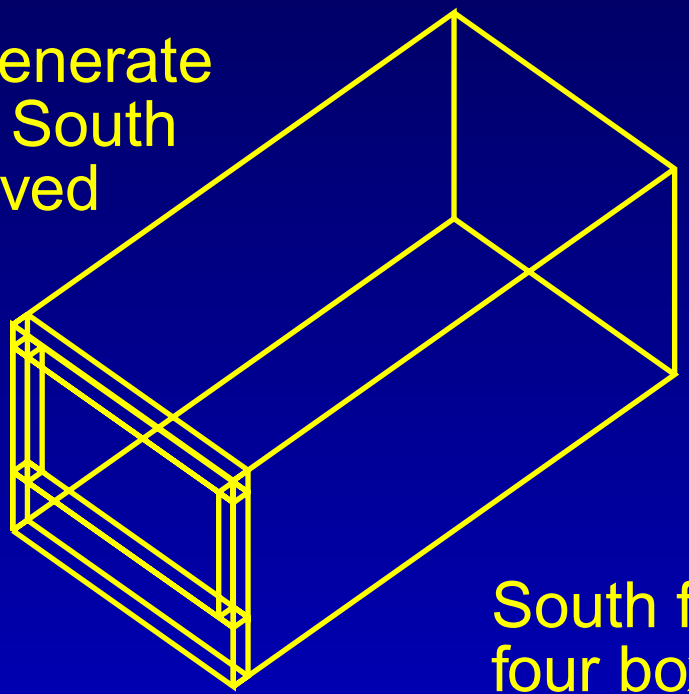
Level 1 HS from the lower polygon can now find the reflected light from the (underside) of the upper polygon



A More Realistic Scene: Daylit Office

The office model is a closed ‘box’ with a window on the South-facing wall. It was created using **`genbox`** and **`xform`**.

`genbox` used to generate (thin) outer walls - South wall polygon removed



Material properties set using text editor on scene file

South facade comprised of four boxes and glazing

Now We Begin The 'Live' Simulations...