Radiance Primer

INTRODUCTION TO *RADIANCE*

Based on material from ***Rendering with Radiance***

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

WHAT IS *RADIANCE*?

Radiance is a collection of fifty-odd programs that do everything from object modelling to point calculation, rendering, image processing, and display. The system was originally developed as a research tool to explore advanced rendering techniques for lighting design. It has evolved over the years into a highly sophisticated lighting visualization system, which is both challenging and rewarding to learn. *Radiance* is unique in its ability to accurately simulate light behaviour in complicated environments, which means two things: correct numerical results, and renderings that are indistinguishable from photographs. There is simply no other physically based rendering system, free or otherwise, with as much power and flexibility as *Radiance*.

Renderings that are indistinguishable from photographs?

Radiance?

Real?

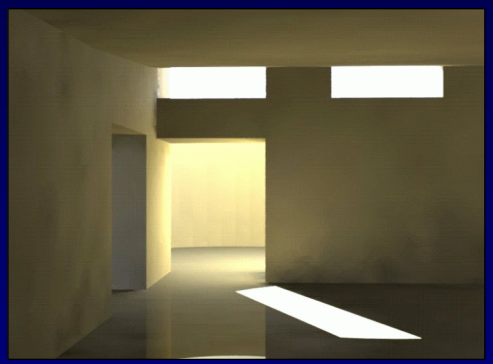
WHAT MAKES *RADIANCE* UNIQUE?

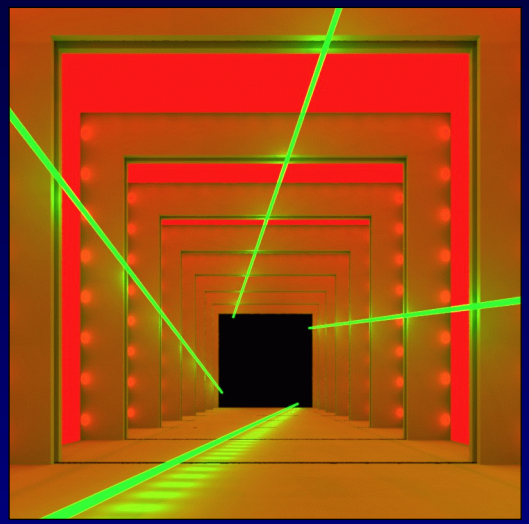
(a) Its singular flexibility. This is largely because the system is based on the UNIX toolbox model.

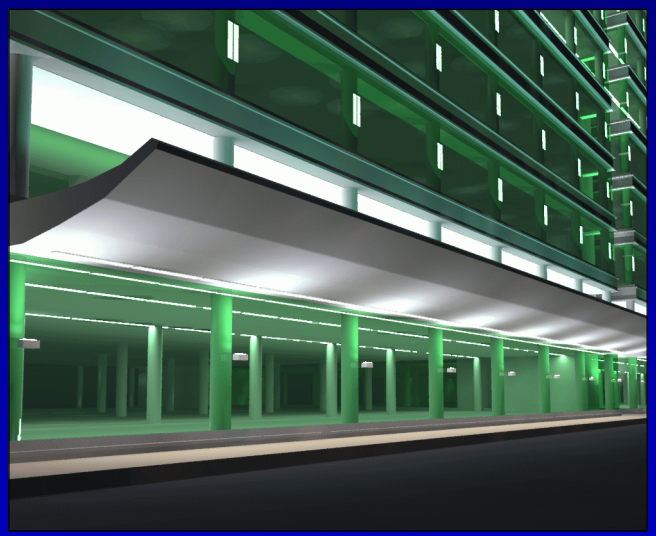
(b) The algorithms used to predict the transport of light are not found in any other lighting simulation system or package.

(c) The development history of the software. In the years since the first release, *Radiance* has benefited enormously from user feedback: most of the enhancements made to the system were outcome of real or perceived user requirements.

Example Applications I



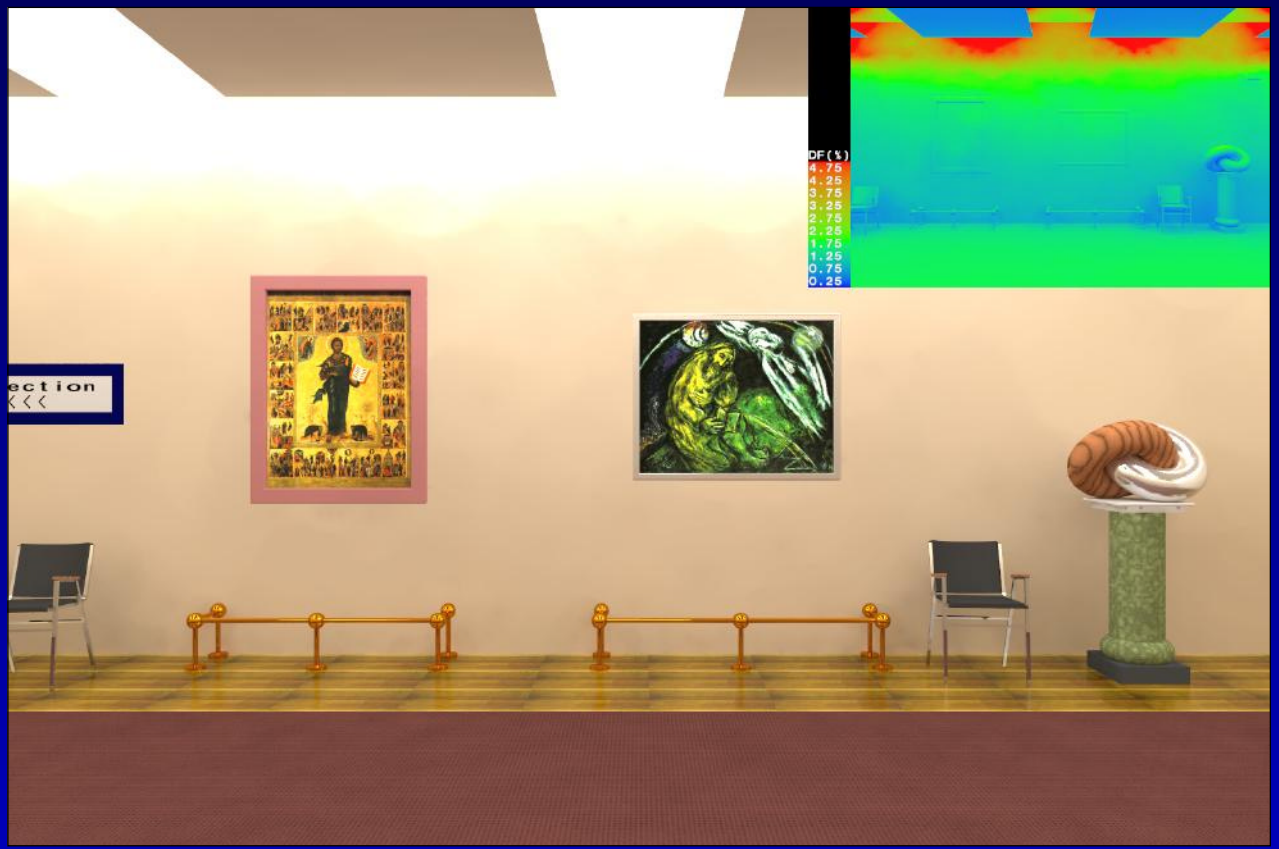




Example Applications II

Art Gallery - Illumination provided by rooflights

Shading devices below rooflights



Inset shows illumination expressed as a percentage of CIE overcast sky horizontal illuminance (daylight factor)

Example Applications III

Classroom wing of the Queen's Building DMU

Inset shows plan view of daylight factor at work plane height

Basic Concepts

Scene geometry: the model used to represent the shapes of objects in an environment, and the methods for entering and compiling this information.

Surface materials: the mathematical models used to characterize light interaction with surfaces.

Lighting simulation and rendering: the technique used to calculate light propagation in an environment and the nature of the values computed.

Image manipulation and analysis: image processing and conversion capabilities.

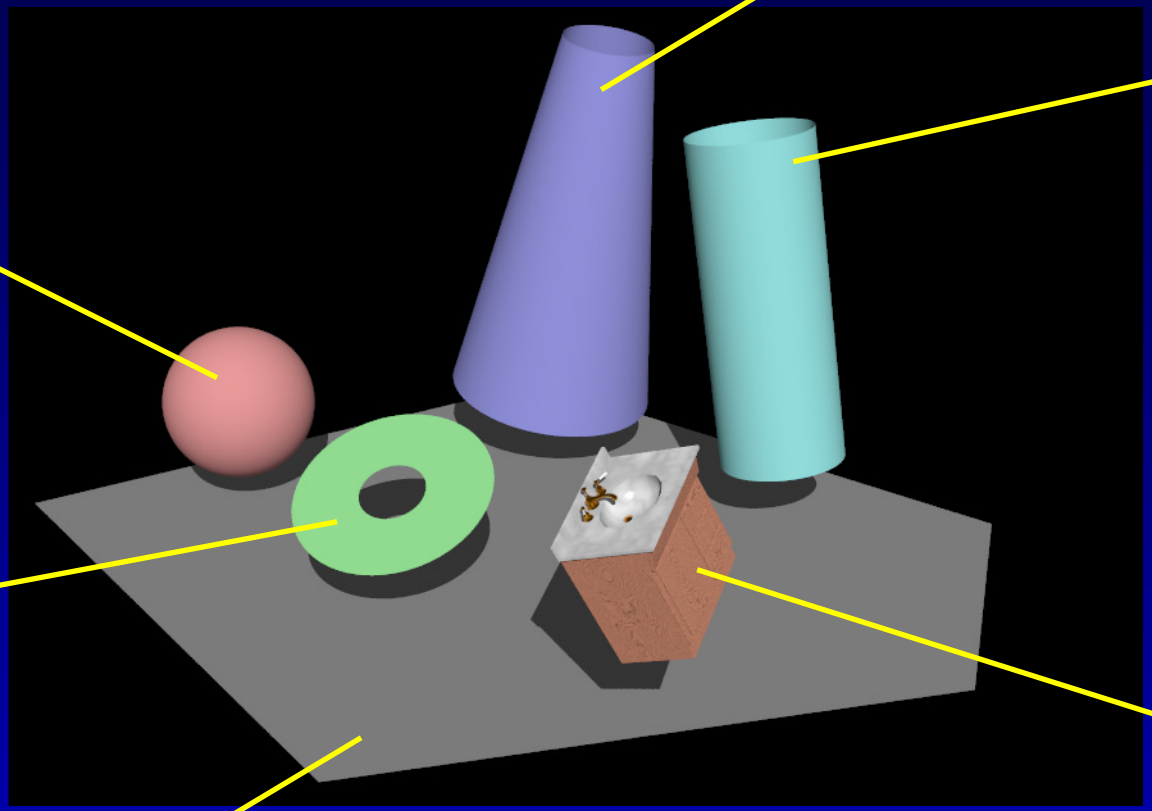
Integration: interconnection and automation of rendering and analysis processes, and links to other systems and computing environments.

SCENE GEOMETRY...

...is constructed from surface primitives:

- **`polygon`**
- **`sphere`**
- **`cone`** (**`cylinder`**, **`ring`**)
- **`source`** - a direction and subtended angle indicating a solid angle of light entering the environment, used to model the sun or the sky.
- **`instance`** - any number of surfaces confined to a region of space. Multiple occurrences of an instance use hardly any additional memory.
- **`antimatter`** - a pseudomaterial that can be used to subtract portions of a surface.

A Few Surface Primitives

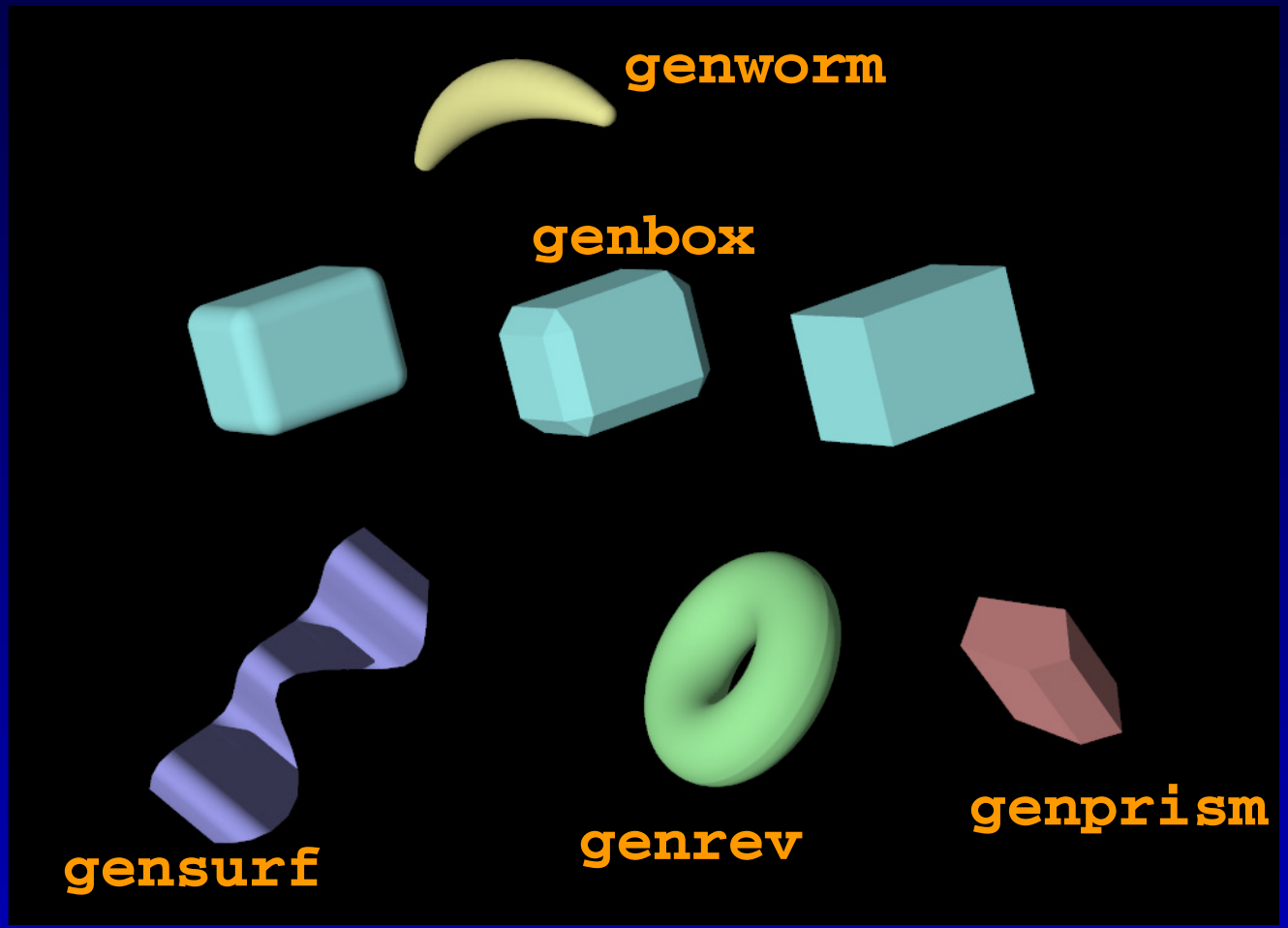


Scene Generators and Manipulators

There are a number of these in the *Radiance* toolbox. We can use them to create and manipulate quite complex geometries without having to resort to CAD. These include:

- **`xform`** - scales, rotates, and moves *Radiance* objects and scene descriptions.
- **`genbox`** - creates a parallelepiped with sharp, bevelled, or rounded corners.
- **`genprism`** - creates a truncated prism, extruded from a specified polygon along a given vector.
- **`gensky`** - generates a description of a clear, intermediate, overcast, or uniform sky, with or without a sun.

GENERATED ENTITIES

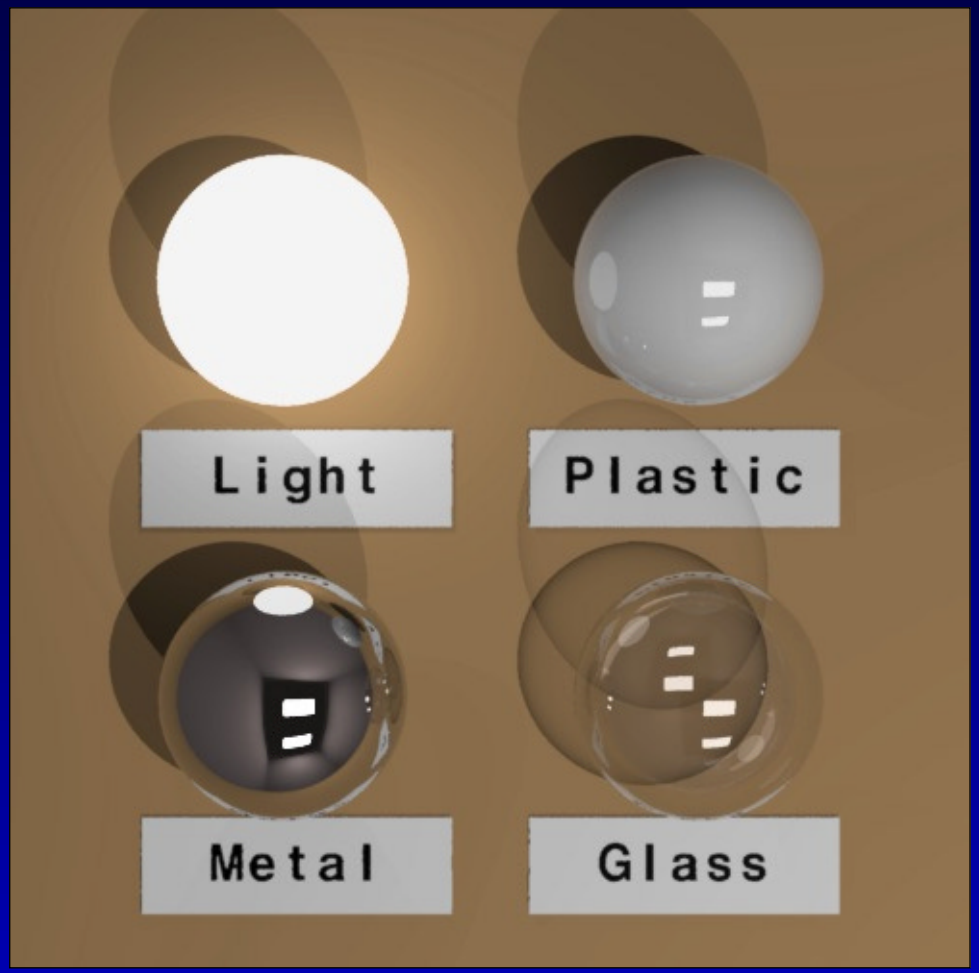


SURFACE MATERIALS

A non-exhaustive list of *Radiance* materials:

- **`light`** is used for an emitting surface, and it is by material type that *Radiance* determines which surfaces act as light sources.
- **`plastic`** is used for most (opaque) building materials. Any specular component does not take on the colour of the base material.
- **`metal`** is used for, well..., metals. Here, the specular component takes on the hue of the metal.
- **`trans`** is used to model thin translucent materials.
- **`illum`** is a special material for “secondary” light sources.

RENDERING SHOWING SOME MATERIALS



Lighting Simulation And Rendering

Radiance employs a light-backwards ray-tracing method.

Light is followed along geometric rays from the point of measurement (the view point or virtual photometer) into the scene and back to the light sources.

The result is mathematically equivalent to following light forward, but the process is generally more efficient because most of the light leaving a source never reaches the point of interest.

The Key Rendering Programs

The most used rendering programs in the *Radiance* toolkit are:

- **`rview`** - The interactive program for scene viewing. Use to preview a scene, debug the scene geometry and select views for final, high-quality rendering with **`rpict`**. See also **`objview`**.
- **`rpict`** - This program produces the highest-quality raw (unfiltered) pictures. The picture is not generally viewed until the rendering calculation is complete and the output has been filtered.
- **`rtrace`** - This program computes individual radiance or irradiance values for lighting analysis or other custom applications.

Other Rendering Programs

These are used for specific tasks and generally make calls (i.e. use) to the **`rtrace`** program:

- **`dayfact`** - An interactive script to compute illuminance values and daylight factors on a specified work plane.
- **`findglare`** - Locates bright sources that would cause discomfort glare in a human observer.
- **`glare`** - An interactive script that simplifies the generation and interpretation of **`findglare`** results.
- **`mkillum`** - Converts specified scene surfaces into illum secondary sources for more efficient rendering.

IMAGE MANIPULATION AND ANALYSIS

A *Radiance* picture is unlike any other computer graphics image you are likely to encounter. The pixel values are real numbers corresponding to the physical quantity of radiance (recorded in watts/ steradian/m^2).

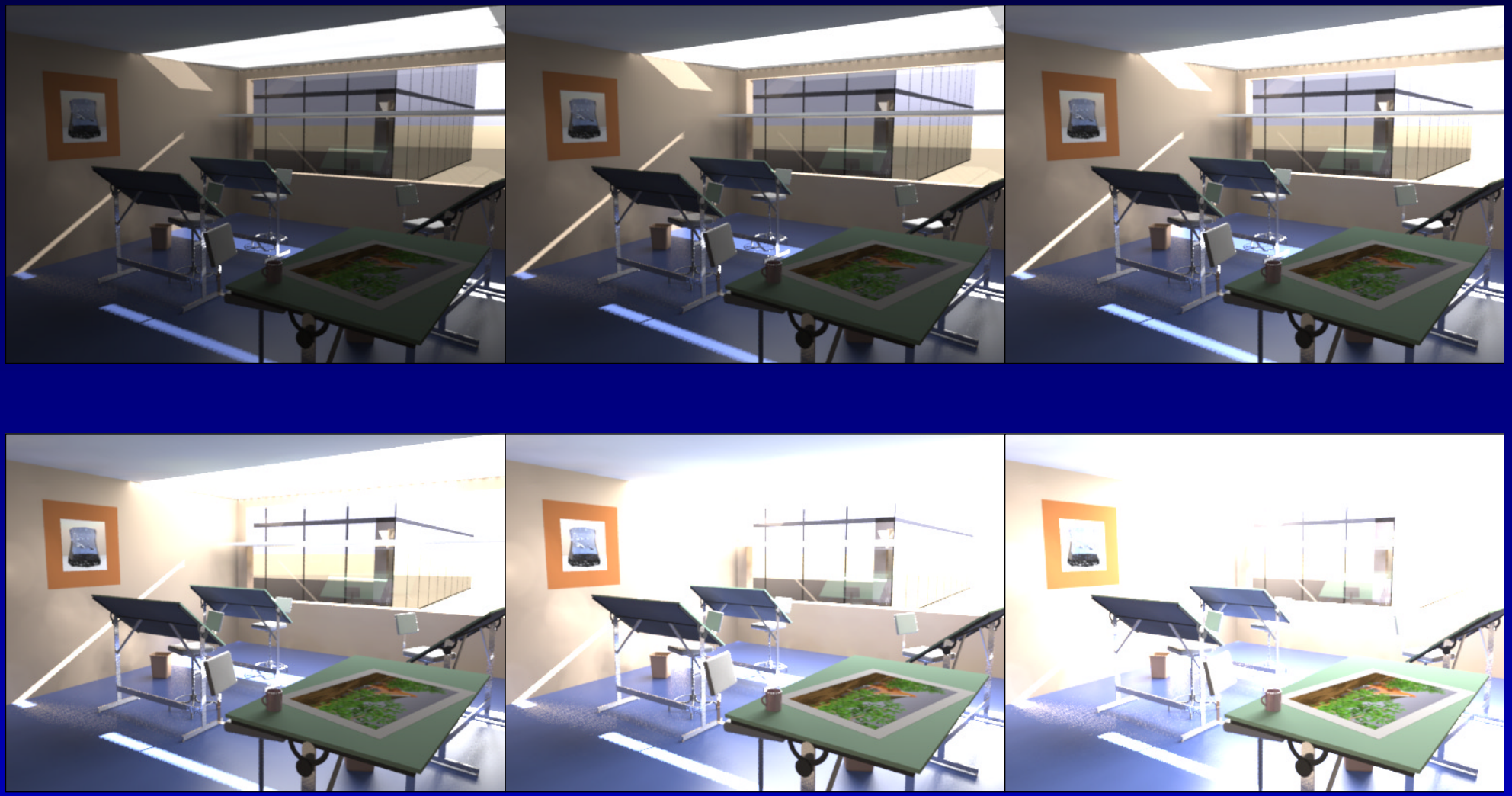
The ASCII header contains pertinent information on the generating commands, view options, exposure adjustments, and colour values that can be used to recover pixel ray parameters and other information needed for various types of image processing.

High-Dynamic Range Images

Radiance predicts real-world luminances and illuminances. These (luminances) are often much greater than can be reproduced by any display device, e.g. CRT monitor.

Thus the user must decide what exposure to apply to the image when it is displayed. This is similar to setting an exposure value when using a camera for daylit scenes: you choose to either expose for the brightest parts (leaving the darkest under-exposed) or vice-versa.

VARYING EXPOSURE BY 1-STOP INCREMENTS



A Few Of The Available Programs:

- **`pfilt`** - Performs antialiasing and exposure adjustment. A picture is not really finished until it has passed through this filter.
- **`falsecolor`** - Converts a picture to a false-colour representation of luminance values with a corresponding legend for easy interpretation.
- **`pcompos`** - Composites pictures together in any desired montage.
- **`pcond`** - Conditions pictures for output to specific devices, compressing the dynamic range as necessary to fit within display capabilities.
- **`ximage`** - Displays one or more *Radiance* pictures on an X11 windows server.

INTEGRATION

Executive programs to simplify the rendering process. The two most important of these tools are listed below.

- **`rad`** - This is probably the single most useful program in the entire *Radiance* system, since it controls scene compilation, rendering, and filtering from a single interface.
- **`ranimate`**: This control program handles many of the administrative tasks associated with creating an animation. It coordinates one or more processes on one or more host machines, juggles files within limited disk space, and interpolates frames, even adding motion blur if desired.

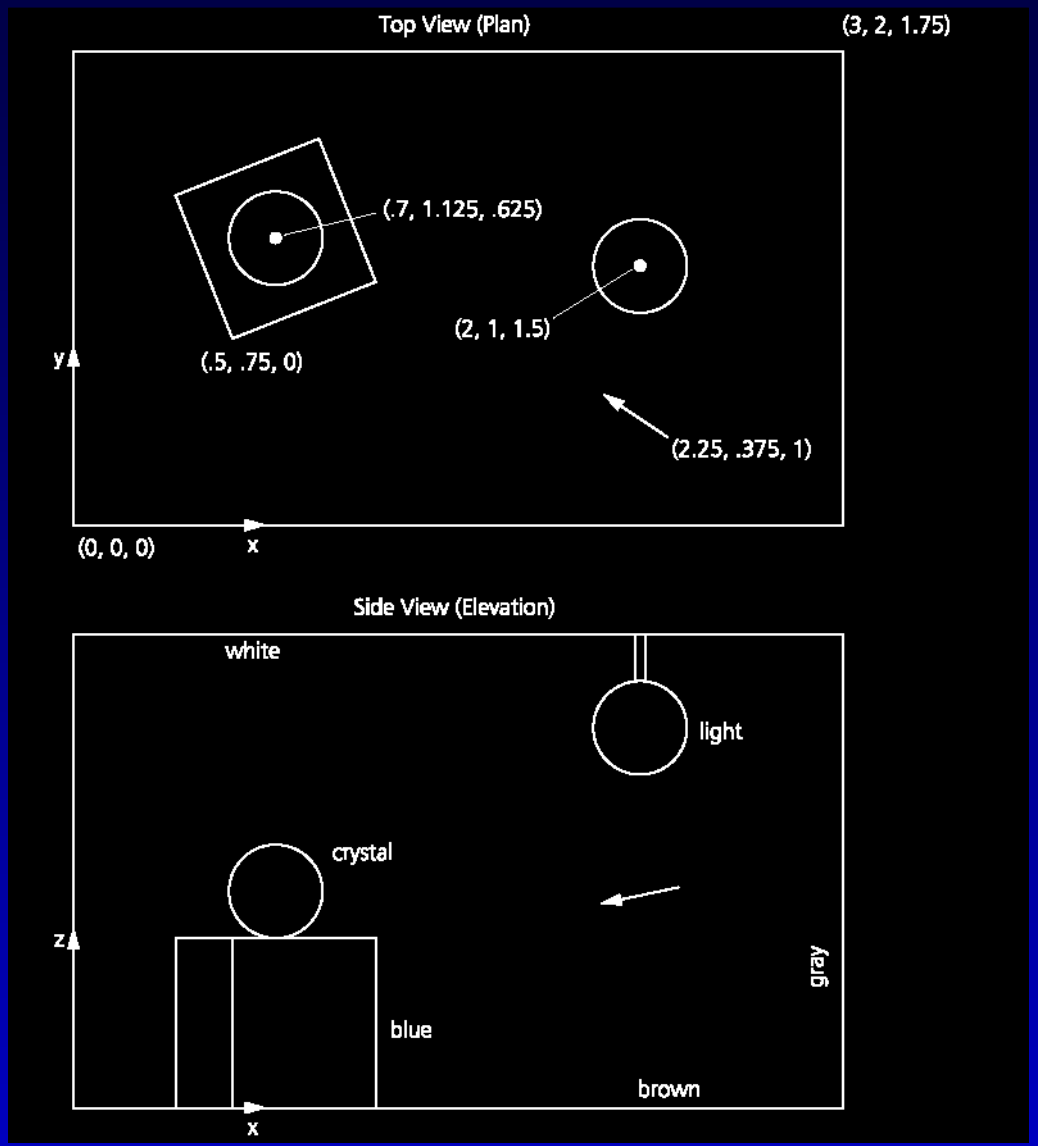
In Addition To The Tools...

...within the **UNIX** *Radiance* distribution, there are a few other systems that integrate *Radiance* in CAD or other environments, these include:

- **ADELINE**: A collection of CAD, simulation, and visualization tools for MSDOS systems, which includes a DOS version of *Radiance*.
- **Desktop Radiance** - A Windows 95/98/NT software package that integrates *Radiance* with AutoCAD Release 14. Desktop Radiance includes libraries of materials, glazings, luminaires and furnishings so you can quickly create realistic lighting models.

THE SCENE0 TUTORIAL

A simple room with a block, a ball, and a light source.



UNIX TUTORIALS

There are a great number of UNIX tutorials available on-line.

This one does a good job of covering the basics:

http://www.ee.surrey.ac.uk/Teaching/Unix/index.html

Work through this teaching package if you are new to UNIX.

Environment Variables

The **PATH** variable lists the directories the shell should search to find a command. In other words, the *Radiance* executable programs (binaries and scripts) need to be listed in the **PATH** variable so that the shell can find them when you type a command. To list your path, enter:

```
% echo $PATH
```

Which may give something like:

```
.:/home/jm/bin:/home/jm/RADIANCE/ray/
bin:/home/jm/RADIANCE/rayuser/bin
```

THE `raypath` ENVIRONMENT VARIABLE

This variable lists the directories where Radiance will search for various library files. For example:

`skybright.cal` - This file contains the sky brightness function for sunny and cloudy skies

`woodtex.cal` - Functions to produce a wood grain texture.

Enter: **`% echo $RAYPATH`** to list directories.

As with the **`PATH`** variable, you can add directories to **`RAYPATH`** so *Radiance* can find additional cal files that are not in the standard release.

The Components of Illumination

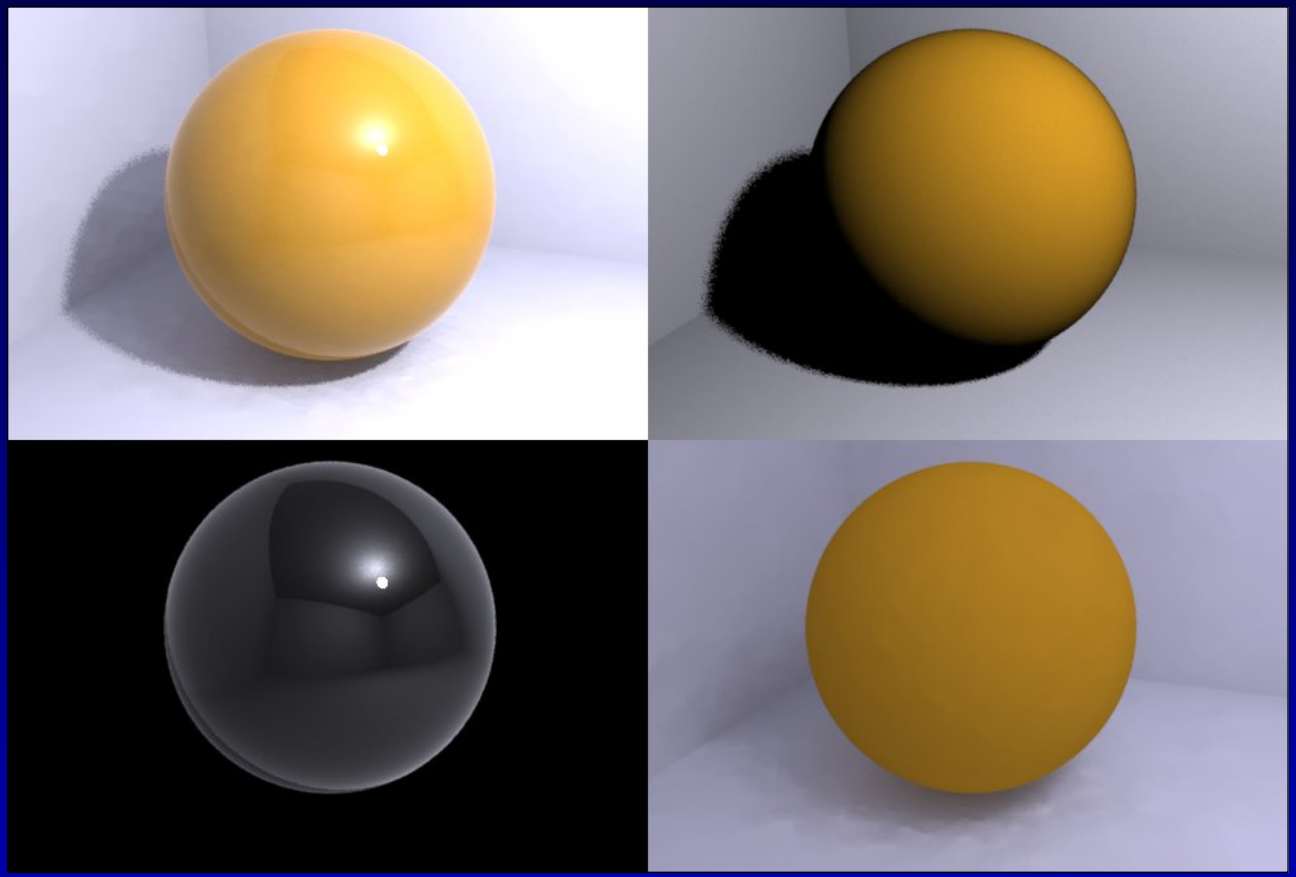
The visual appearance of a scene (e.g. shading, highlights, shadows etc.) is largely determined by the quality of the incoming light and the nature of reflections from the objects in the field of view.

In *Radiance*, we compute various components of illumination separately. These are called:

- Direct - 'important' contributions, e.g. electric lights, sun.
- Indirect - light that arrives following one or more reflections, or extended sources i.e. sky.
- Specular - shiny reflections.

Here They Are Together and Separate

Together



Direct

Specular

Indirect

The "Ambient" Calculation

Also known variously as the "diffuse", "indirect" or "inter-reflection" calculation, it is one of the keystone features of *Radiance*. It is also considered one of the hardest features to understand.

First, we need to distinguish between it and the "direct" calculation.

Simply put, (in *Radiance*) we know in advance where the direct light is coming from, but we have to hunt around to determine where the indirect light is coming from.

Why is Indirect Light Important?

Non-physical renderers (e.g. 3DStudio) make no attempt to model indirect light:

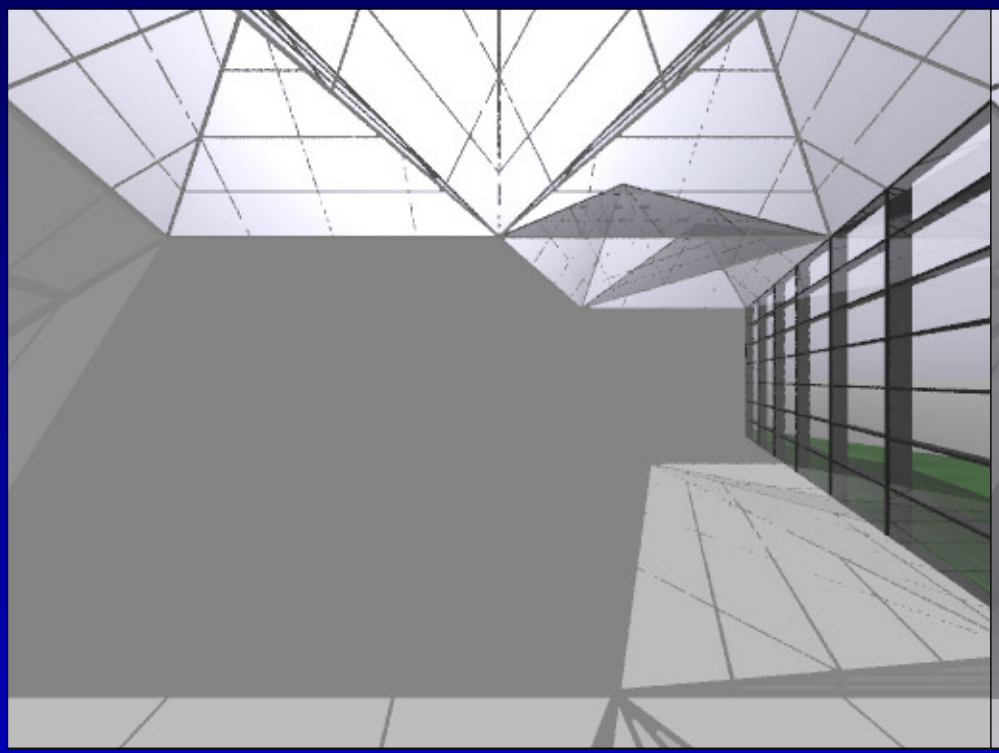
Flat, unreal appearance.

Some ‘shiny’ reflections.

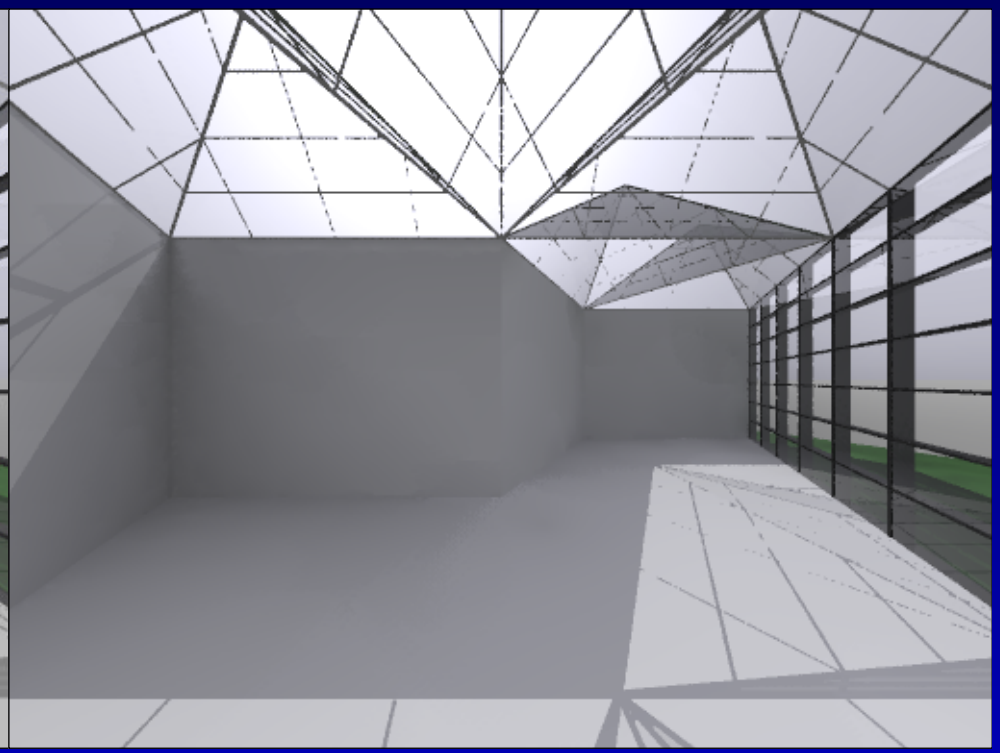
Could not be taken for the ‘real thing’, or used for lighting design.

INDIRECT (AMBIENT, WHATEVER...) LIGHT

Instead of predicting indirect light, non-physical renders apply a constant (ambient) value. With *Radiance*, the indirect light can be predicted.



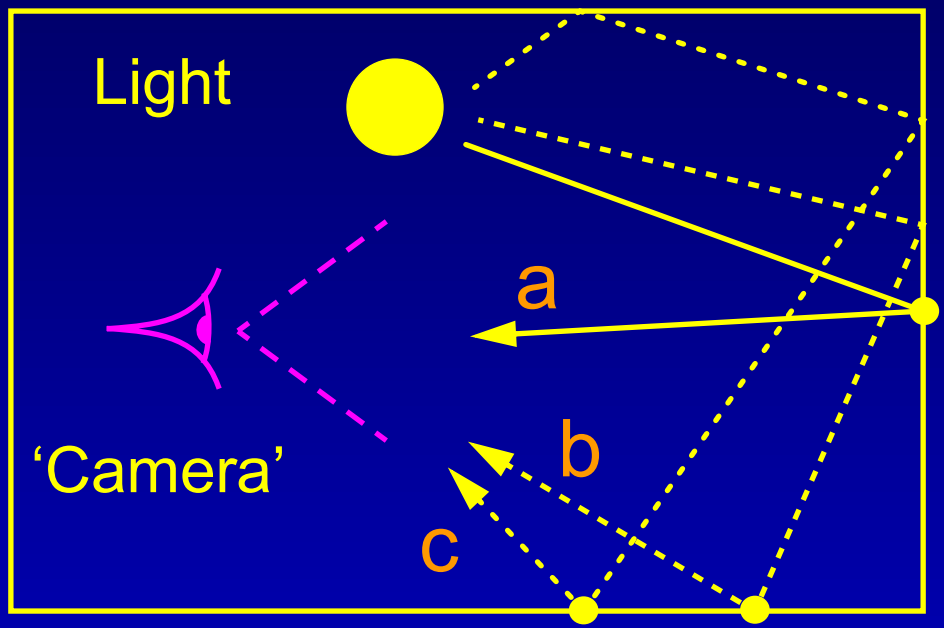
Constant ambient value



Indirect light predicted

Direct and Indirect Light

Introducing the *Radiance* notion of reflections, also known as ‘bounces’.



Direct illumination

Indirect illumination

[a] Light from surface illuminated by source viewed directly

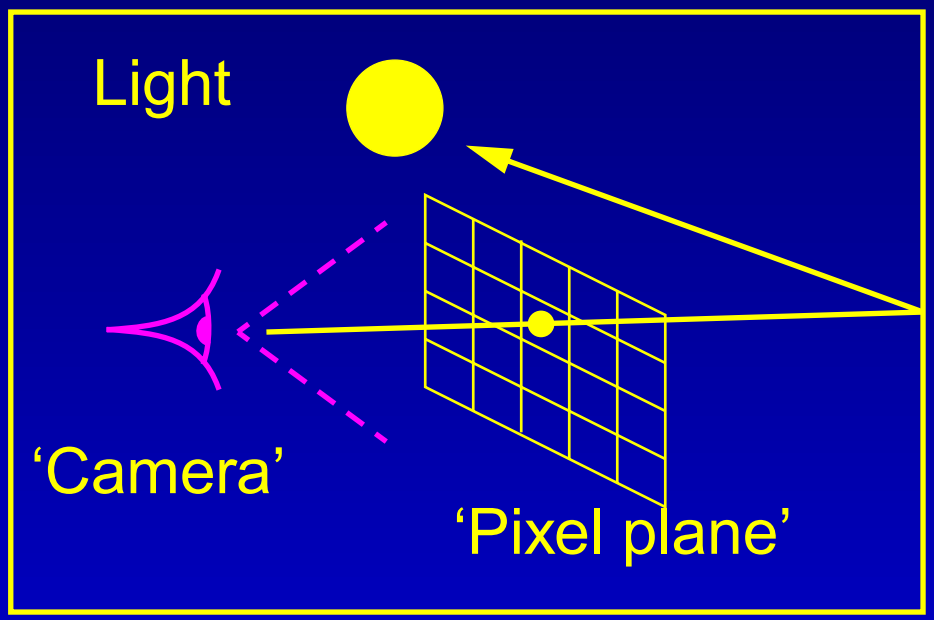
[b] Light from surface illuminated by source viewed after 1 bounce

[c] Light from surface illuminated by source viewed after 2 bounces

RADIANCE COMPUTES THESE BY...

... sending rays out from the viewpoint (or virtual camera). This is backwards ray tracing.

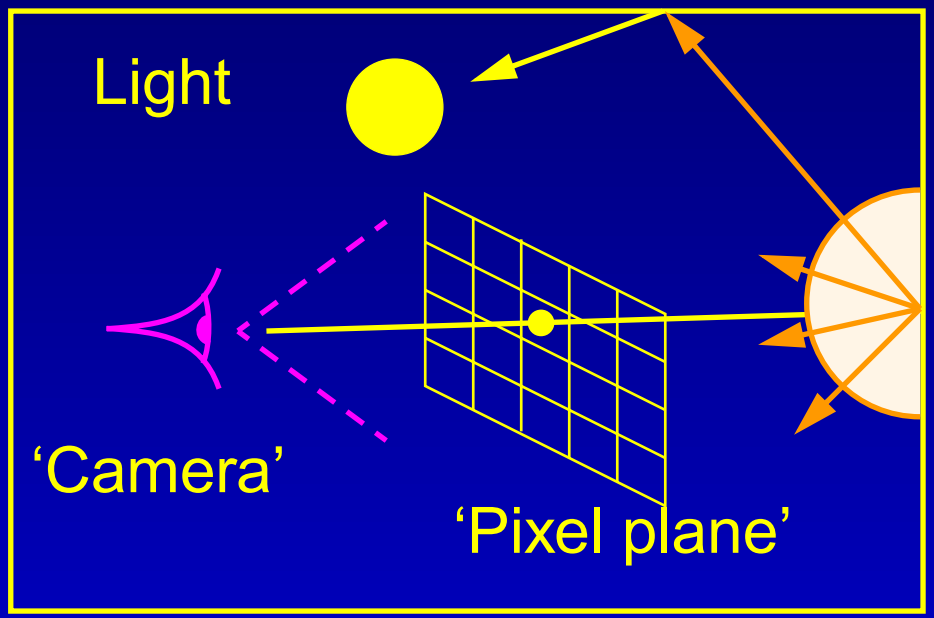
The ray tracing approach to rendering follows “view rays” from the virtual camera through pixels in an imaginary plane into the environment.



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.

SEEMS EASY FOR DIRECT LIGHT...

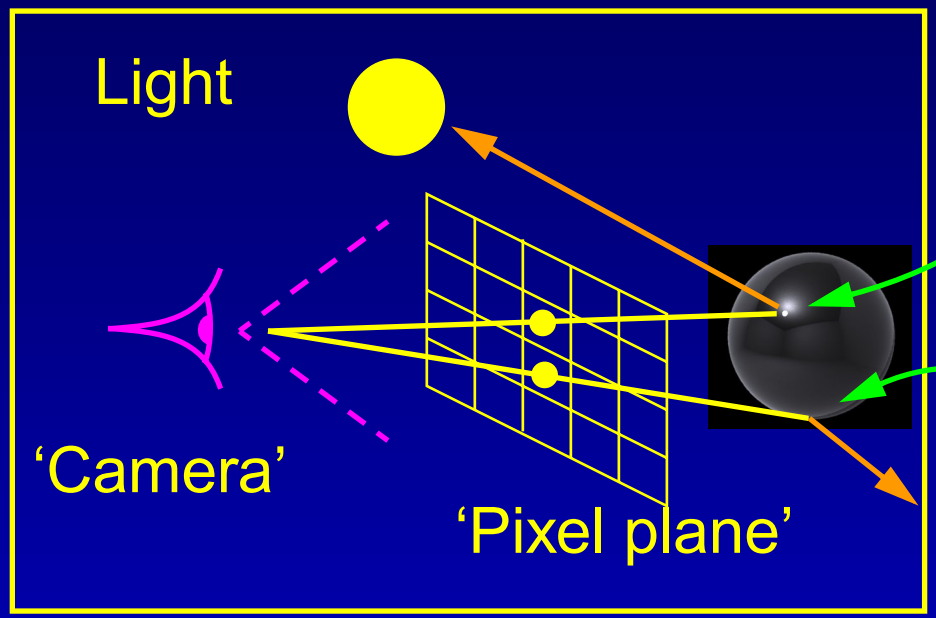
...but indirect light can arrive from any direction. *Radiance* computes indirect light by sending out many rays to find illumination that arrives from the light source by one or more reflections. This is called “hemispherical sampling”.



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.

For Specular Reflections...

...we compute the ‘mirror-like’ reflections off shiny surfaces (backwards from the camera).



Specular reflection to (direct) light source.

Specular reflection to illuminated room surfaces.

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

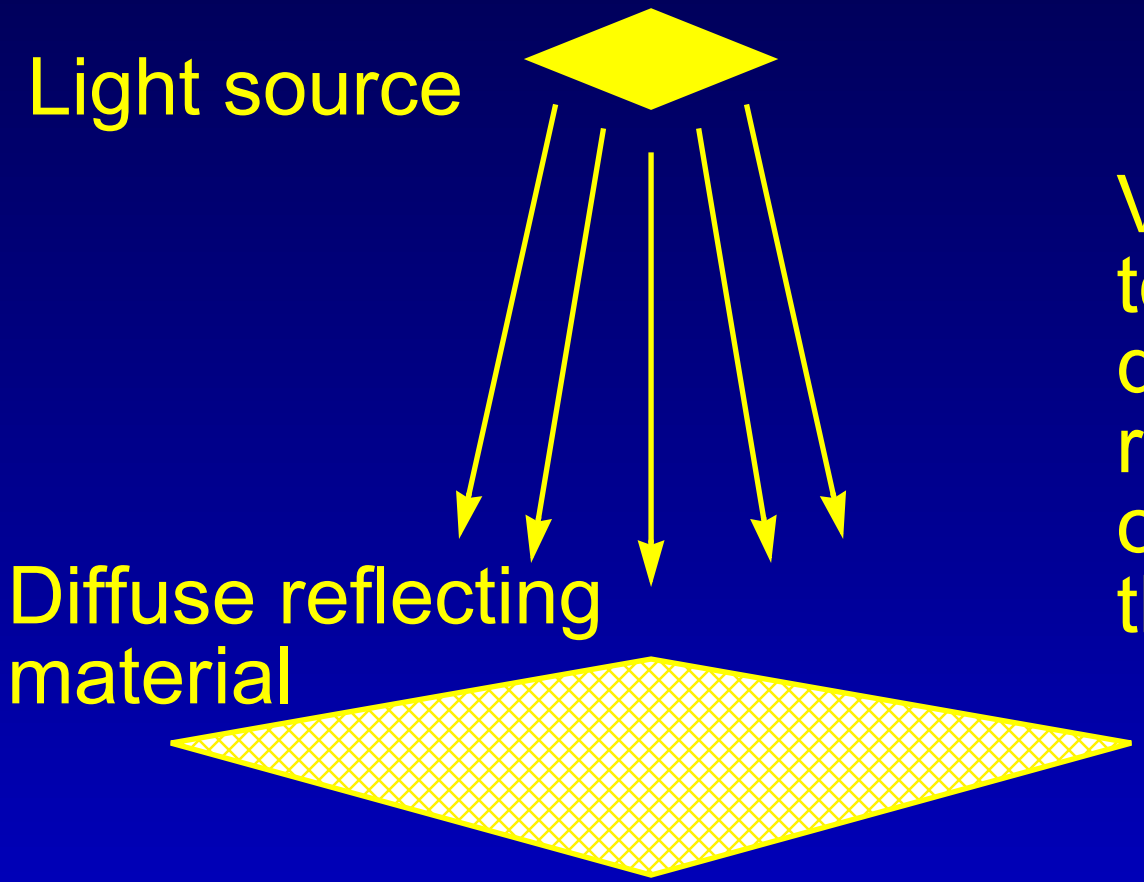
We use the material `light` for important sources of illumination, e.g. electric luminaire, the sun. These participate in the direct calculation of illumination.

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

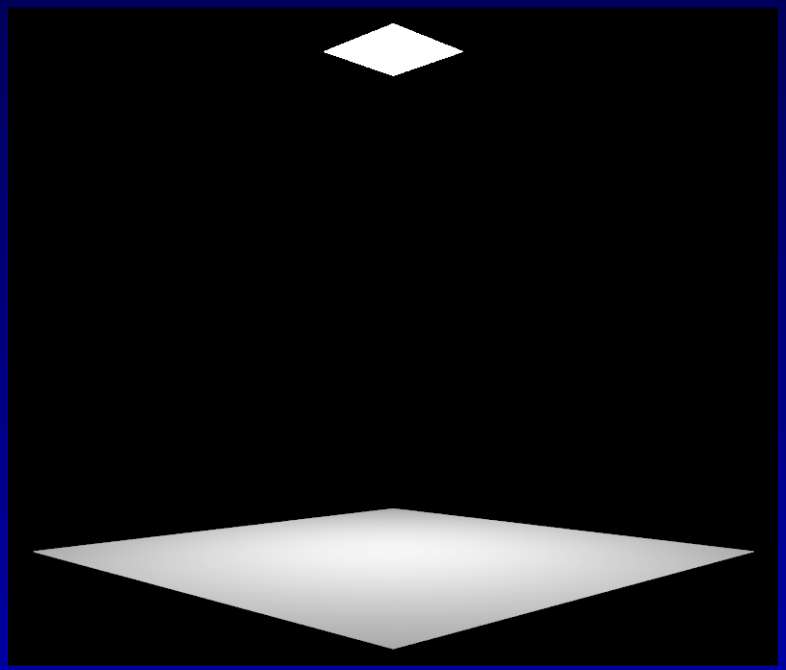
Our test scene comprises two polygons. One is an emitter of light which shines onto the other.

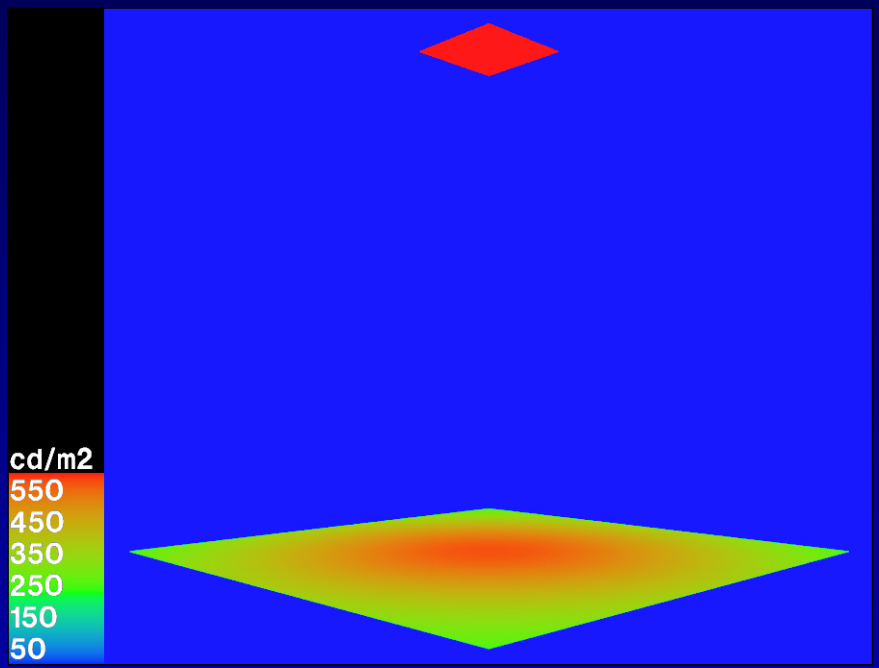


View parameters set to see source shining downwards and the resulting illumination on the upper-side of the polygon below.

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.

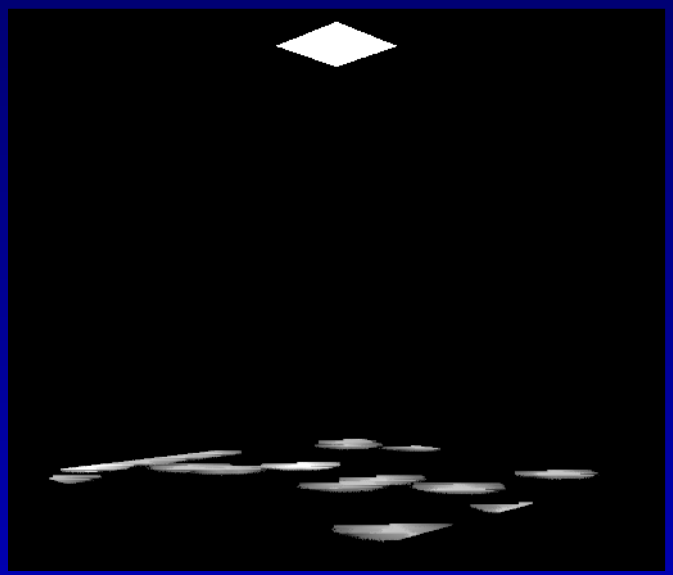




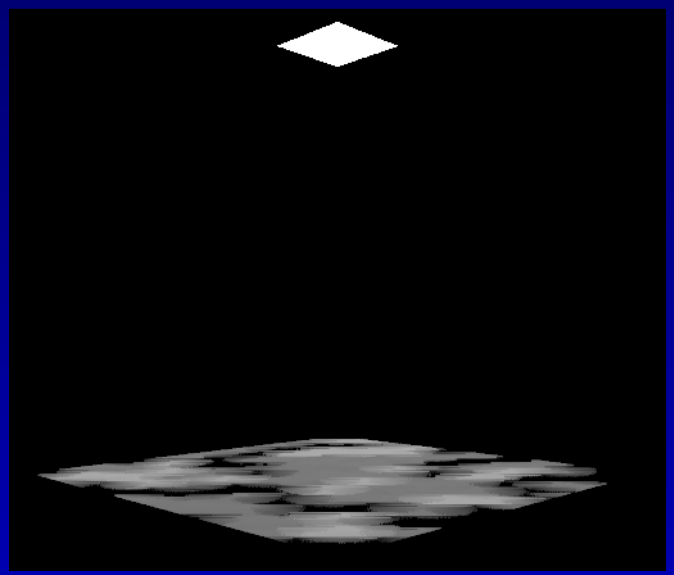
The reflecting polygon is evenly illuminated by the light source.
This is clearly revealed in the false colour image.
Note, "`-ab 0`" setting used, i.e. inter-reflection calculation turned off.

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.



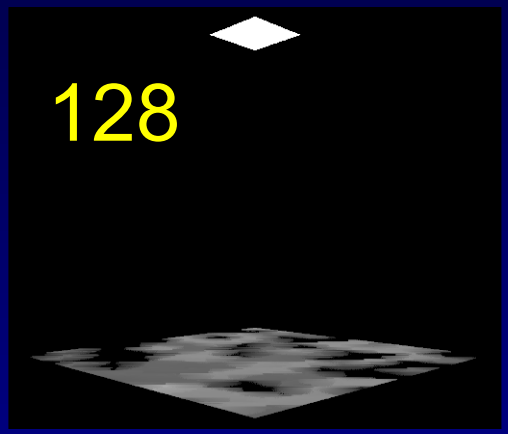
`-ad 32`

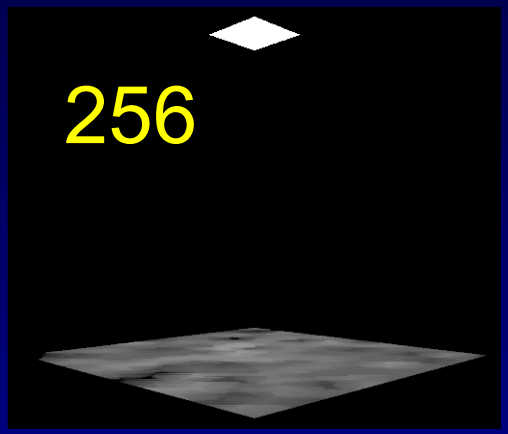


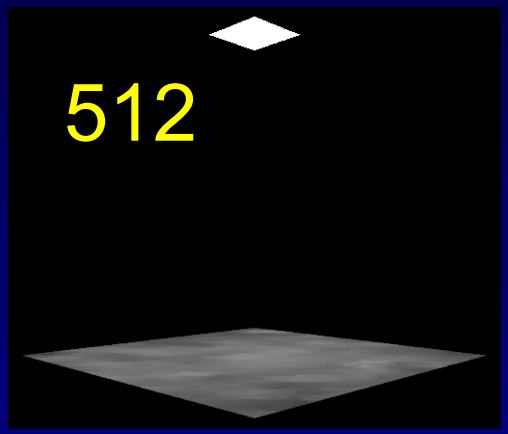
`-ad 64`

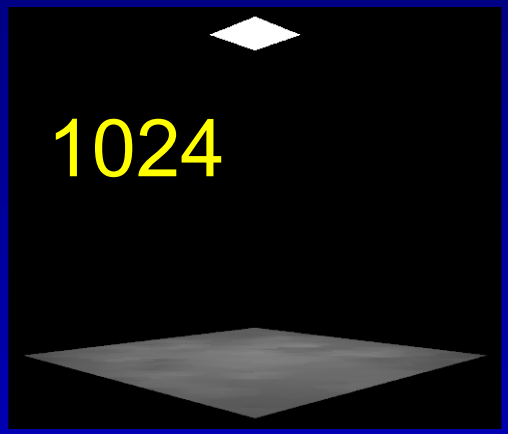
INCREASING THE NUMBER OF ad RAYS...

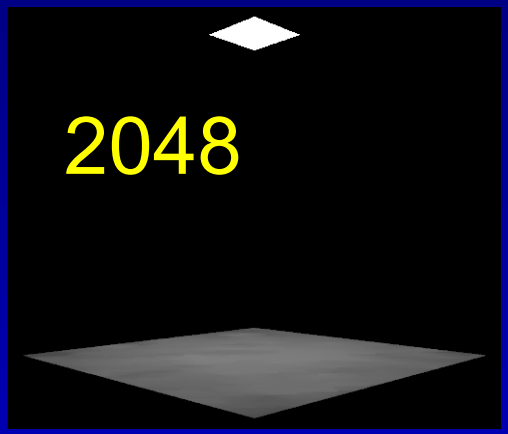
...does produce smoother renderings.

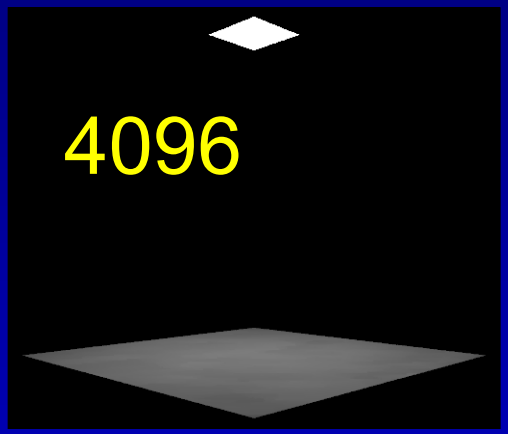








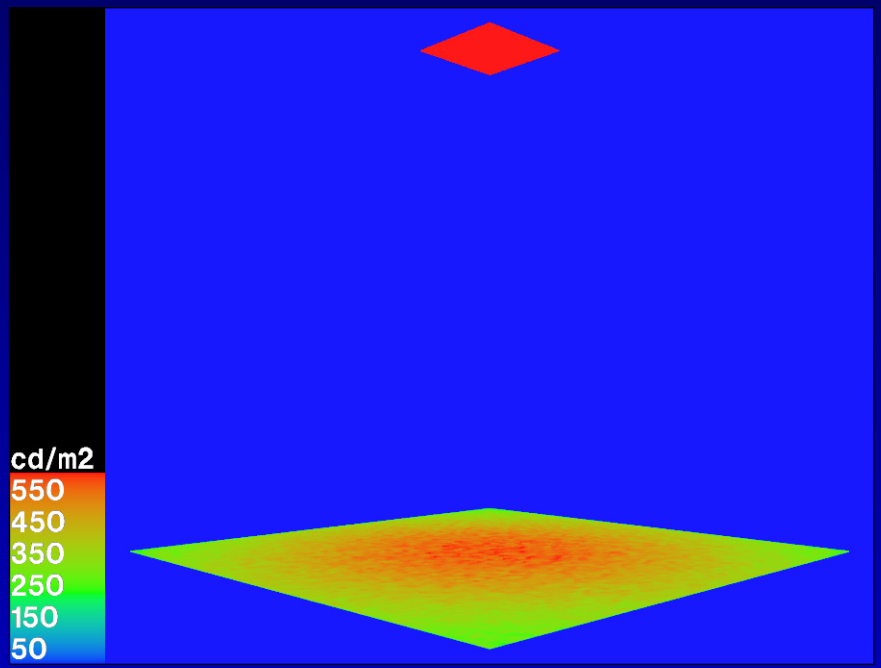




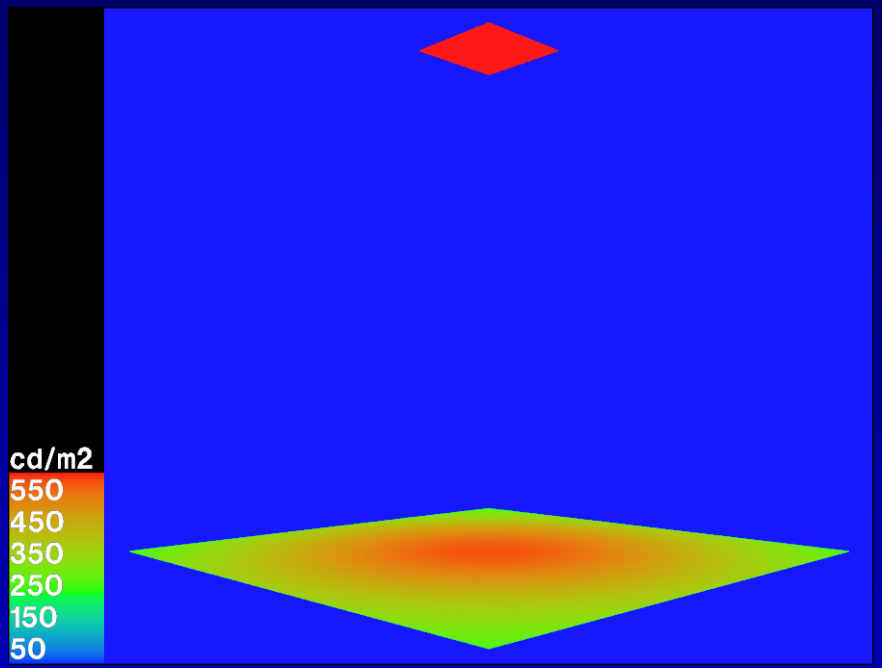
But Even With `-ad 4096`...

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

`glow`

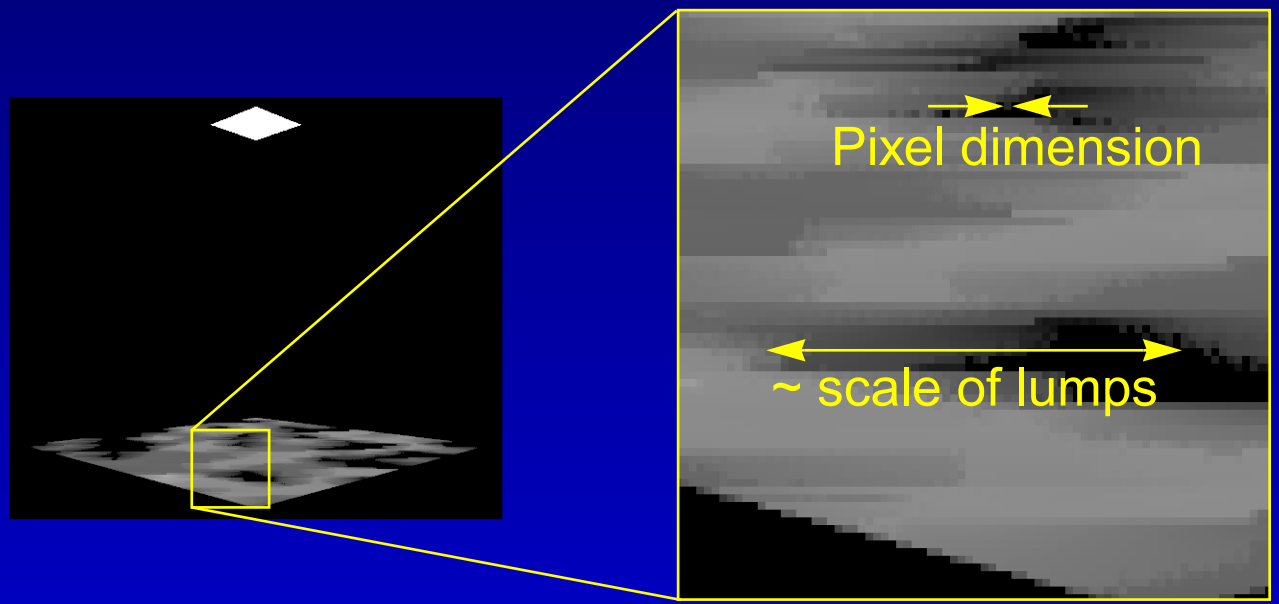


`light`



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.

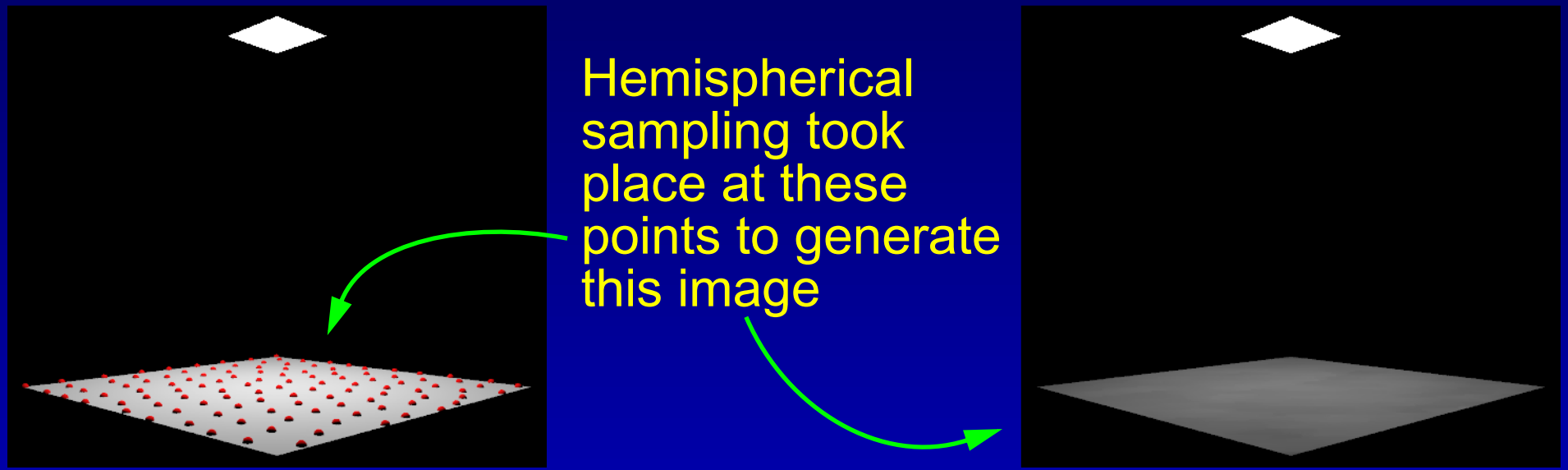
If hemispherical sampling occurred at each of these pixels, then the number of rays sent would be 25,000 times the **`ad`** number:

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.



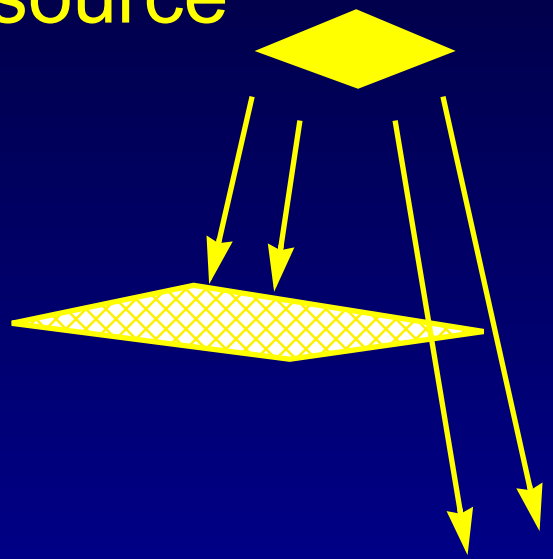
-ad 4096

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.

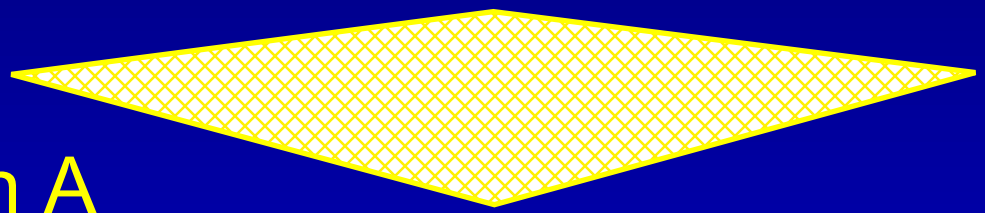
In the previous example, the scene didn't allow for inter-reflection. Here, we modify the scene by adding an occluding polygon to see how hemispherical sampling is used to compute indirect or (inter-reflected) light.

Scene With Occluding Polygon



Polygon B
(occluding)

Occluding polygon shades lower polygon from light source.

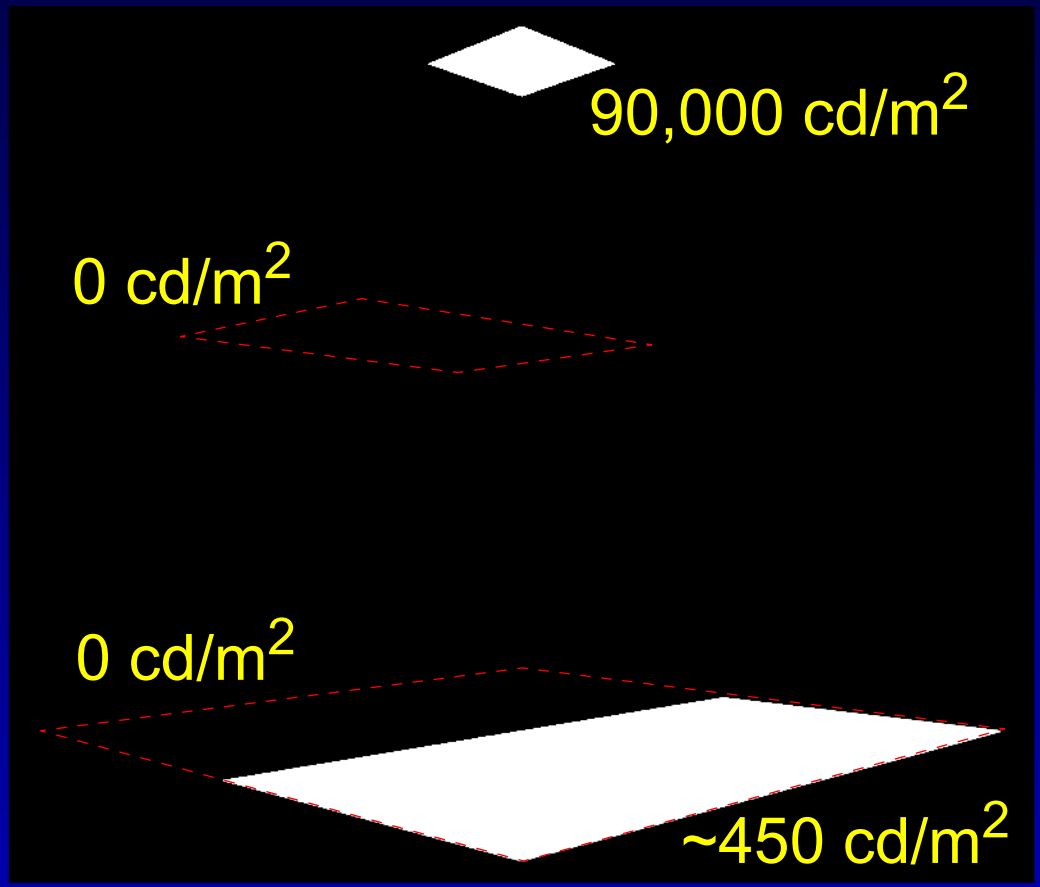


Polygon A

RENDERING FOR OCCLUDING SCENE `-ab 0`

Underside of polygon B not illuminated

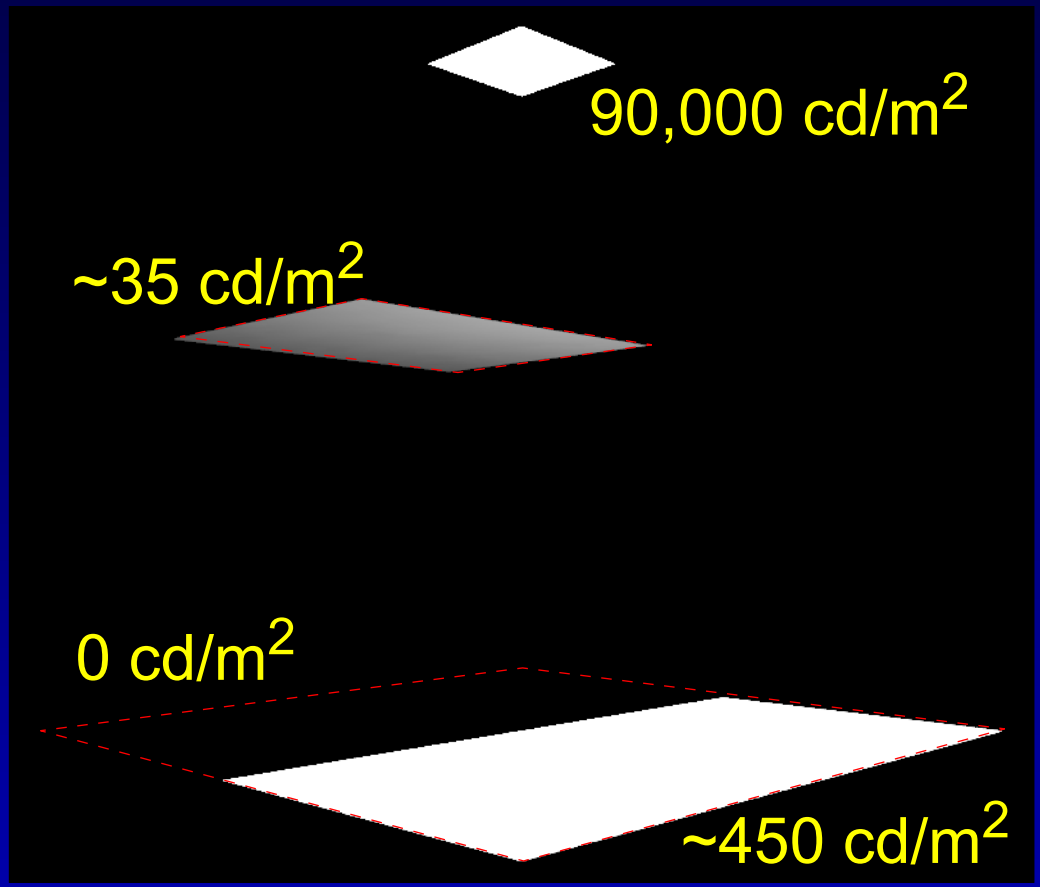
Topside of polygon A half in shade



FOR -ab 1

Underside of polygon B now illuminated

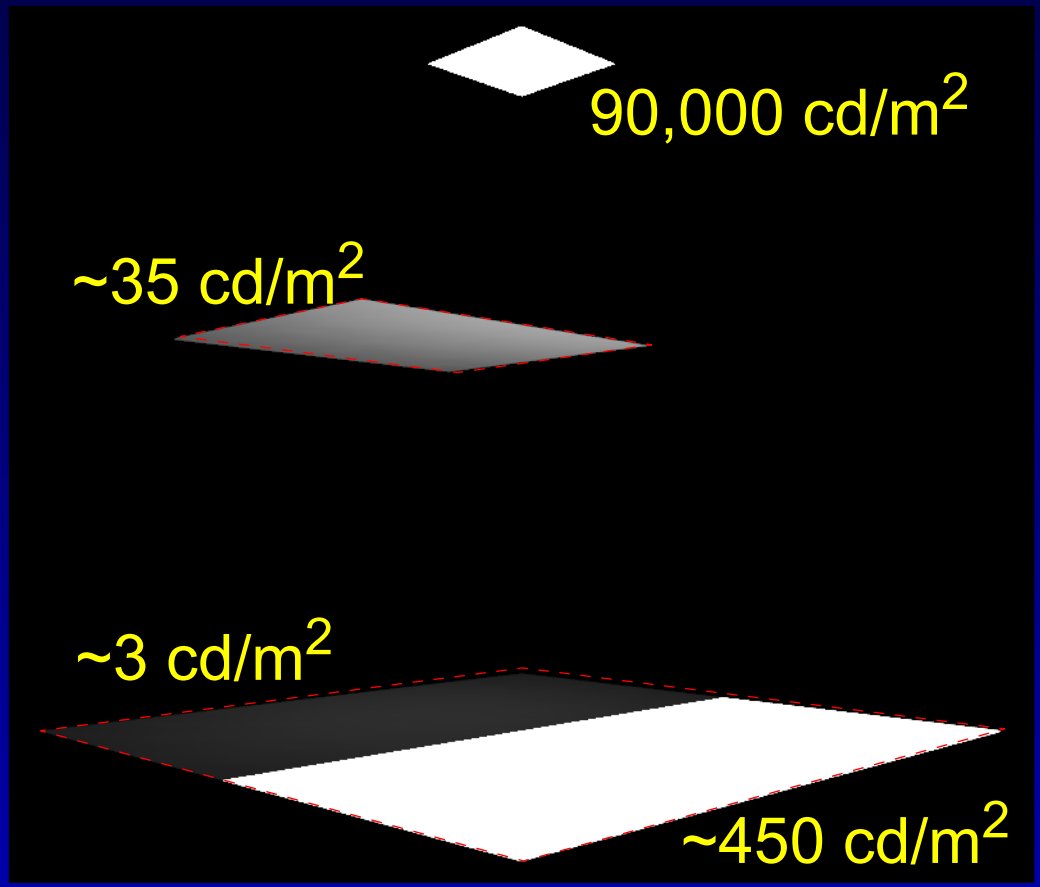
Topside of polygon A still half in shade



AND LASTLY FOR `-ab 2`

Underside of polygon B now illuminated

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



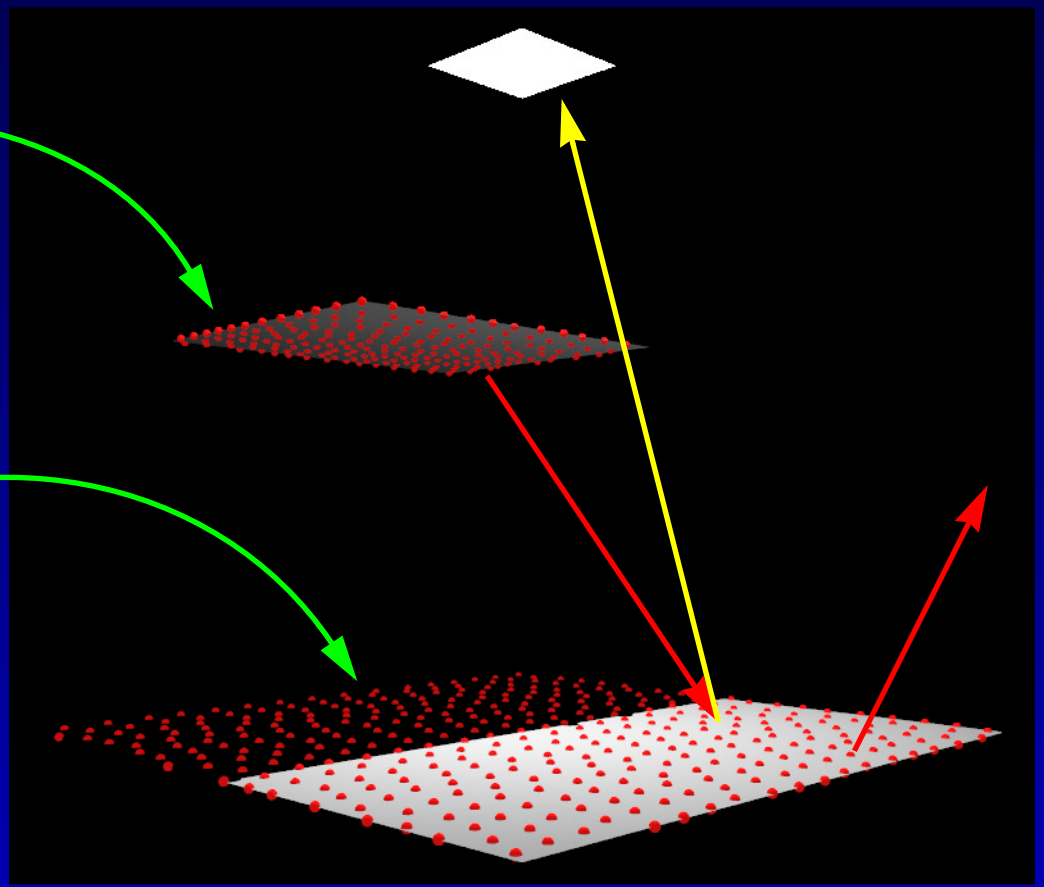
HEMISPHERICAL SAMPLING TOOK PLACE...

...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)

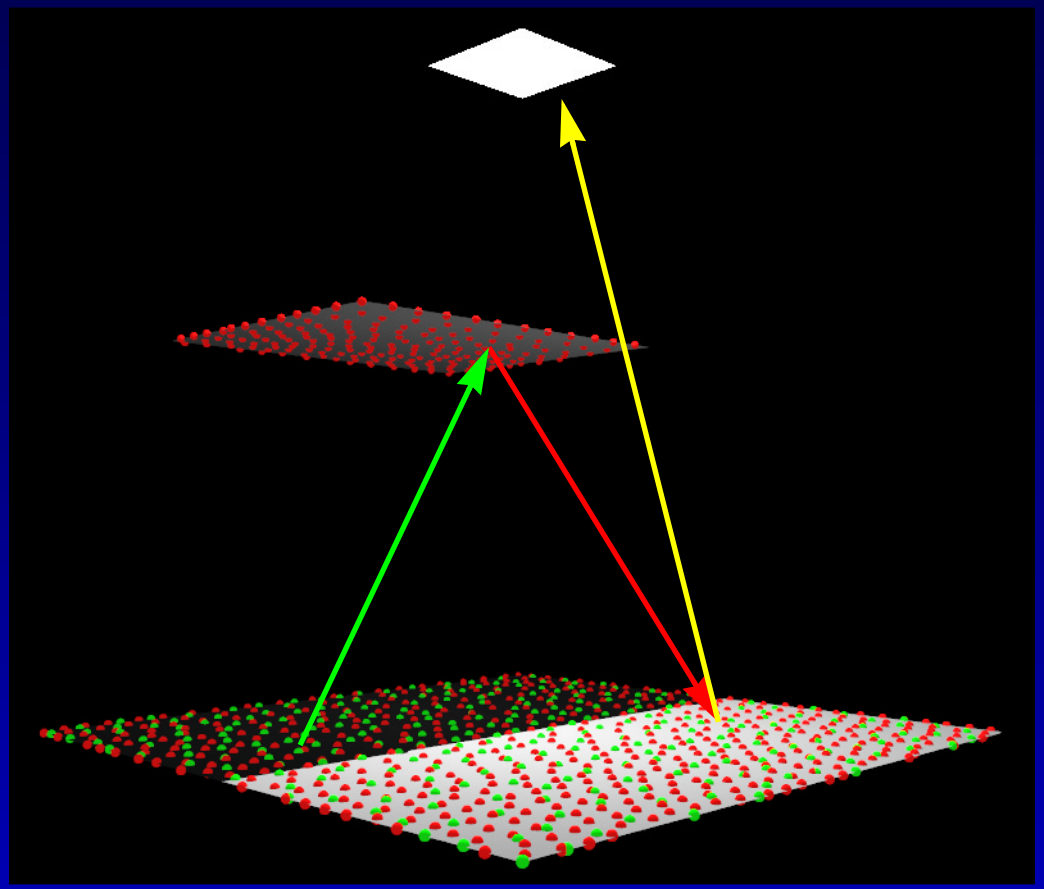


HEMISPHERICAL SAMPLING TOOK PLACE...

...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



THE AMBIENT CALCULATION WILL...

...be revisited later in the Course.

For now, we will noodle with some of the tutorial examples from the *Radiance* book.

We'll use this as an opportunity to recap on the work covered so far.