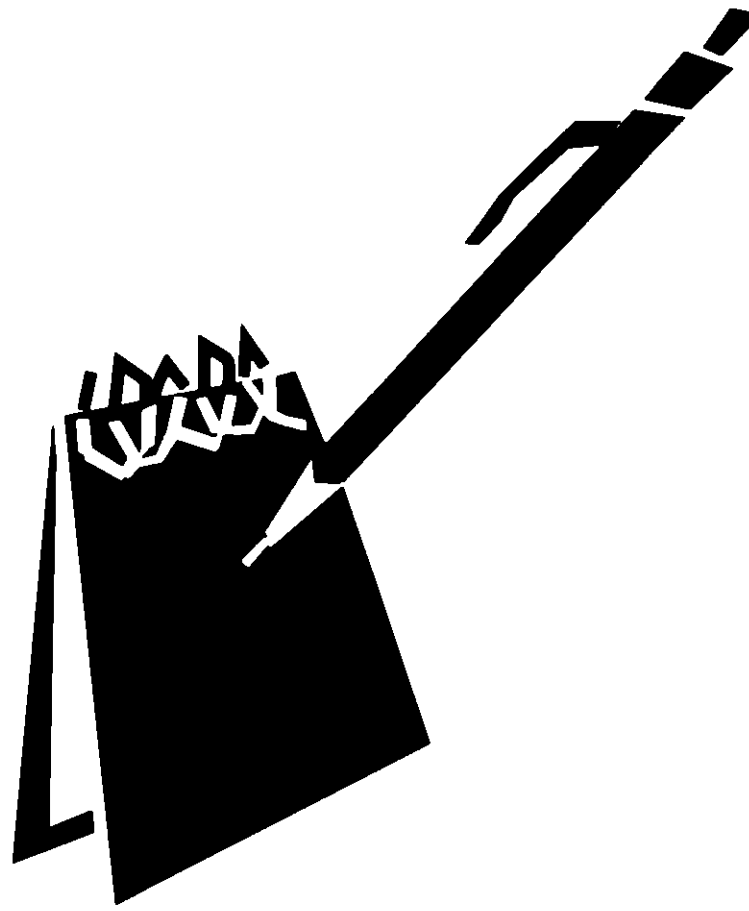
SIGGRAPH 98

33

Rendering with Radiance. A Practical Tool for Global Illumination



25th International Conference on Computer Graphics and Interactive Techniques
Exhibition **21 23 July** 1998 Conference **19 24 July** 1998
Orlando, Florida USA

course notes

33

Rendering with Radiance: A Practical Tool for Global Illumination

Organizer
Gregory Ward Larson
Silicon Graphics Inc.

Lecturers
Charles Ehrlich
Lawrence Berkeley National Laboratory

Gregory Ward Larson
Silicon Graphics, Inc.

John Mardaljevic
De Montfort University

Robert Shakespeare
Indiana University

25th International Conference on Computer Graphics and Interactive Techniques
Exhibition **21 23 July** 1998 Conference **19 24 July** 1998
Orlando, Florida USA

course notes

Rendering with *Radiance*
A Practical Tool for Global Illumination

ACM Siggraph `98 Course #33
Orlando, FL
July 21, 1998

Organizer
Greg Ward Larson
Silicon Graphics Inc

Speakers
Greg Ward Larson SGI
Rob Shakespeare Indiana University
John Mardaljevic DeMontfort University
Charles Ehrlich LBNL

Summary

This course provides essential information for artists designers and researchers interested in creating realistic images with the *Radiance* Lighting Simulation and Rendering System a free global illumination package developed at the Berkeley National Laboratory over the past 12 years This software runs on most UNIX platforms including Linux for the PC and Macintosh and has been ported to the PC under DOS and Windows It s widespread availability and reputation for accuracy have made it popular among cutting edge lighting designers and engineers who wish to visualize novel lighting and daylighting solutions Some of the more exotic uses include virtual sculpture rock show lighting theatrical backdrop rendering and validating a computer vision system for the space shuttle Computer graphics researchers and hobbyists have also taken to *Radiance* as a testbed for advanced global illumination and rendering algorithms In this course four *Radiance* experts including the author of the package will present their work give demonstrations and provide tips for using the software on practical problems Tutorial examples will be taken from lighting analysis theater lighting and daylighting design The author of *Radiance* will describe the underlying principles that make this ray-tracing software unique and the audience will be given ample opportunity to ask questions and offer suggestions for future development

Schedule

8 30a	Larson	Introduction
8 55a	Shakespeare	Tutorial Example
9 40a	Mardaljevic	Daylighting Applications
10 00a		Break
10 15a	Mardaljevic	Daylighting Applications (continued)
10 35a	Ehrlich	Lighting Design Considerations
11 20a	Shakespeare	Illumination of Large Structures
12 00		Lunch
1 30p	Shakespeare	Theater Lighting
2 15p	Larson	Calculation Methods Employed in *Radiance*
3 00p		Break
3 15p	Mardaljevic	Advanced Daylighting Calculations
3 45p	Larson and Mardaljevic	Validation Studies
4 00p	Larson and Ehrlich	Future Program Developments
4 15p	All	Working through an Example Design Problem
4 45p	All	Open Q&A Session

Syllabus

A Introduction (25 min) *Larson*
- 1 Program history
- 2 System design and limitations
- 3 Essential programs
- 4 Input and output

B Tutorial Example (45 min) *Shakespeare*
- 1 Visualize simple room with light source and actor
- 2 Closer look at *Radiance* materials surfaces instances etc
- 3 Introduction to the rad program for automating the rendering process

C Daylighting Applications (40 min) *Mardaljevic*
- 1 Basics daylight factor technique standard skies etc
- 2 Generating sky models with Radiance
- 3 Calculating DF values
- 4 Visualization - a daylit atrium scene

D Lighting Design Considerations (45 min) *Ehrlich*
- 1 Introduction
- 2 Software tools available
- 3 Modeling approach & methods
- 4 Material Properties
- 5 Analysis for Lighting Design

E Illumination of Large Structures (40 min) *Shakespeare*
- 1 Concept development and who will use the *Radiance* pictures?
- 2 Indoor outdoor and exploring the visual impact
- 3 Efficient handling of large data sets
- 4 Material/geometry considerations
- 5 Luminaire and lamp selection and placement
- 6 Technique for accurately aiming luminaires
- 7 Lines of light Cold Cathode and Neon
- 8 Transition from day to night
- 9 Selecting the views and post processing of pictures

F Theater Lighting (45 min) *Shakespeare*
    1 Actors sets and props
    2 Photometry acquisition and management
    3 Shaping the beam of light shutters and templates
    4 Organizing and aiming the lightplot
    5 The color of light
        a) dimming effects
        b) colored filters
        c) adaptation and normalization
    6 Special material considerations
    7 Atmospheric effects the *Radiance mist* primitive
    8 Summary images and potential for image based control systems
G Calculation Methods Employed in *Radiance* (45 min) *Larson*
    1 Direct calculation
        a) selective shadow testing
        b) adaptive source subdivision
        c) virtual light source calculation
    2 Indirect calculation
        a) specular sampling
        b) indirect irradiance caching
    3 Secondary light sources
        a) impostor surfaces
        b) computing secondary distributions
    4 Participating media (*mist*)
        a) single-scatter approximation
        b) the *mist* material type
    5 Parallel rendering
        a) goals
        b) methods
H Advanced Daylighting Calculations A Glare Analysis Case Study (30 min) *Mardaljevic*
    1 Outline of the design problem
    2 Possible approaches
    3 Final technique
    4 Results
    5 Discussion
I Validation Studies (15 min) *Larson and Mardaljevic*
    1 Electric light comparisons
    2 Daylight comparisons
J Summary and Future Program Developments (15 min) *Ehrlich and Larson*
    1 At LBNL
    2 At SGI
K Working Through an Example Design Problem (30 min) *All*
    1 Daylight study
    2 Lighting analysis
    3 Aesthetic issues
    4 Summary
L Open Panel Session for Audience Questions (45 min) *All*

Speaker Biographies

Gregory Ward Larson
Member of the Technical Staff
Silicon Graphics Inc
2011 N Shoreline Blvd M/S 07U 553
Mountain View CA 94043 1389
(650) 933 4878 2663 fax
(510) 528 2044 home
gregl@sgi com

Gregory Ward Larson is a member of the technical staff in the engineering division of Silicon Graphics Inc Previously he was a staff scientist at the Lawrence Berkeley National Laboratory He graduated with an A B in Physics in 1983 from the University of California at Berkeley and earned his Master's in Computer Science from San Francisco State University in 1985 His professional interests include digital photography and image standards physically based rendering global and local illumination, luminaire simulation electronic data standards, and lighting related energy and environmental conservation issues Greg has published numerous papers in computer graphics (including four SIGGRAPH papers) and illumination engineering He is the primary author of the widely used *Radiance* system for the analysis and visualization of lighting in design and author of *Rendering with Radiance* from Morgan Kaufmann He is also the inventor of an imaging gonioreflectometer for the measurement of reflectance of architectural materials and the developer of the Materials and Geometry Format for lighting information exchange

Robert A Shakespeare
Associate Professor
Indiana University
Rm 200 Theatre
Bloomington IN 47405 3085
(812) 855 8827 tel/fax
tcvc@indiana edu

Robert Shakespeare is a professional lighting designer Associate Professor at the Department of Theatre and Drama of Indiana University and Director of the Indiana University Theatre Computer Visualization Center He has lighted over 180 stage productions in 5 countries for companies including The Bristol Old Vicin England St Lawrence Center in Canada, the Utah Shakespearean Festival and the Lyric Theatre in Hong Kong His architectural lighting projects have included Times Square the Jin Jaing Hotel in Shanghai the Hong Kong Marriot atrium Tsing Ma and Kap Shui suspension bridges in Hong Kong and the top of Nations Bank Atlanta for the Olympics

Robert uses *Radiance* and other lighting simulation software as part of his design process and is coauthor of *Rendering with Radiance* Current projects include linking databases derived from lighting/computer visualization interactions directly to the technology of complex lighting control environments such as theaters and theme parks His professional affiliations include the Illuminating Engineering Society of North America, the International Association of Lighting Designers and the United States Institute for Theatre Technology He is a member of the IESNA Computer Committee and he was a speaker at two previous SIGGRAPH courses

John Mardaljevic
Research Fellow
Institute of Energy and Sustainable Development (IESD)
De Montfort University
The Gateway
Leicester LE1 9BH
UK
Tel +44 (0) 116 250 6242
Fax +44 (0) 116 257 7449
E mail jm@dmu ac uk
jm@dmu ac uk

John Mardaljevic is a Research Fellow at the Institute of Energy and Sustainable Development De Montfort University Leicester UK He received his B Sc (1982) in Physics and an M Phil (1988) in Astrophysics both from the University of Leicester In 1990 he took up a Research Assistant post with De Montfort University His first work there was on a project to assess dynamic thermal simulation programs for passive solar design In 1991 he began to look into daylighting design tools for complex spaces in particular atrium buildings The *Radiance* system seemed particularly well suited to coping with modern atria e g complex designs with a large number of specular or semispecular reflecting surfaces As data from the International Daylight Measurement Program became available the emphasis of John s work shifted towards validation and novel approaches to illuminance prediction Specifically a comparison of sky model performance based on internal illuminance predictions and the formulation implementation and validation of the daylight coefficient approach for the *Radiance* system This technique offers the potential for an efficient evaluation of the internal illuminance due to any sky condition by reusing pre-computed illuminance values from a discretized sky

In addition to pure research John has used *Radiance* to create renderings and to provide design advice for various architectural projects To date these have included atria (daylight factor and visualization) electrically lit offices shading analysis (a pre-process for thermal simulation programs) and the evaluation of the visibility of a large-scale video-display screen against daylight produced glare

Between 1993 and when he rejoined De Montfort in 1996 John worked as a Research Assistant at the University of Aberdeen Scotland Based at the Marine Laboratory Aberdeen he worked on oceanographic and ecosystem modeling projects John has published papers on astrophysics marine science and illumination modeling He is married and has a daughter

Charles Ehrlich
Principal Research Associate
Lawrence Berkeley National Laboratory
1 Cyclotron Rd 90 3111
Berkeley CA 94720
(510) 486 7916 4089 fax
CKEhrlich@lbl gov

Charles Chas Ehrlich is currently a research associate at Lawrence Berkeley National Laboratory and serves as the main point of contact for the *Radiance* and ADELINE software packages He has been working with *Radiance* in various capacities for almost 10 years Chas earned his bachelor s in architecture from the University of California at Berkeley College of Environmental Design in 1989 In 1990 he established the private consulting firm called Space & Light which provides *Radiance* training and project consulting for lighting analysis Chas is a member of the Illuminating Engineering Society of North America and the CIE He is active in the daylighting committee of the IESNA and is editor of the Daylighting applications chapter of the IES Handbook

Over the years Chas has performed work for dozens of clients including architects Mark Mack, Polsheck and Partners Skidmore Oewings and Merril and Cesar Pelli and Associates lighting designers Horton Lees Lighting Design of New York and Becca Foster of San Francisco energy consultants Energy Simulation Specialists of Tempe Arizona, Cunningham and Associates of San Francisco, and Stephen Winter and Associates of Norwalk, Connecticut laywers Alan Moss of San Francisco

Projects completed by Space & Light include the daylighting of the Inventure Museum in Acron Ohio exterior lighting of a Bank Headquarters in Winston-Salem, North Carolina, a theater in San Francisco the new International Lobby building at the San Francisco International Airport a terminal building interior at the Ben Gurion International Airport, a library in Southern California, a utility headquarters building for Southern California Edison a daylighting analysis for Wall Mart stores several legal cases including one train pedestrian accident and numerous other small projects

Introduction to *Radiance*

Greg Ward Larson
Silicon Graphics, Inc

Program History

- First applied to lighting in 1986
- Developed at LBNL in California and EPFL in Switzerland under government funding
- Version 1.0 released in January 1989
- Version 3.1 released in July 1997
- Work continues at LBNL and SGI

System Design and Limitations

- Ray-tracing engine follows light backwards from measurement point(s) to source(s)
- Diffuse interreflection handled by irradiance caching scheme for global illumination
- Optimizations for many light sources, BRDFs, planar mirrors, scattering & more
- No spectra, curved mirrors or polarization

Essential Programs

- **oconv** - compiles scene description files
- **rview** - interactive rendering program
- **rpict** - batch rendering program
- **pfilt** - picture filter and exposure control
- **rad** - executive control program for above

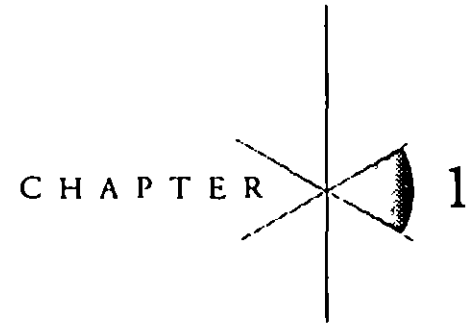
Input and Output

Today's Speakers

- Greg Ward Larson
  - principal *Radiance* author
- Rob Shakespeare
  - lighting designer, teacher and book coauthor
- John Mardaljevic
  - daylighting expert and chapter author
- Charles (Chas) Ehrlich
  - program expert, manager and chapter author

Morning Schedule

Afternoon Schedule



CHAPTER 1

Introduction

Radiance is a professional tool kit for visualizing lighting in virtual environments It consists of over 50 tools, many of which cannot be found anywhere else and, because of their almost endless possibilities, may appear complex to the beginner To make it easy to get started, this chapter is written as a complete introduction at the end of it you will be able to create and render scenes of your own More advanced concepts are elaborated in the remaining tutorials in this section

We start off by illustrating what distinguishes *Radiance* from other rendering tools namely its ability to predict reality Next, we introduce some of the important tools and concepts that will be needed to understand the material in this book Finally, we offer a short tutorial which is designed to give you some immediate hands-on experience with the software

1.1 Photorealism and Lighting Visualization

Rendering is the process of taking a 3D geometric description and making a 2D image from a specific view. This term is taken from traditional practice in architectural and artistic drawing, whose rules of perspective were developed centuries ago. These rules have been elaborated, refined, and codified in modern computer-aided design (CAD) software. More recent advances in computer lighting models (called *local* and *global illumination models*) have developed further into the field known as *photorealistic* rendering. In most cases, we call an image photorealistic if it "looks as real as a photograph." Although this is a laudable goal, there is still a big difference between something that *looks* real and something that is a good reproduction of reality. We begin this book with a hypothetical example to illustrate this important difference.

Imagine yourself as a third year design student in the architecture department of a large university. For your term project, you are charged with the design, modeling, and rendering of a three-story office complex. In addition to design drawings, you must produce full-color renderings of the inside and outside of your structure. You may produce the renderings by hand or using computer software. In addition, you must produce a daylight study of one room in your structure, using whatever means you have available. Most students are building scale models of their designs to photograph outdoors, but you want to use the computer both for renderings and for daylight analysis. (After all, the CAD program you are using, DesignWorkshop, has settings for the time of day and time of year and claims to do solar studies.)

The design and modeling phases of your project go well, and soon you have a complete set of drawings to hand in. You then turn your attention to rendering and daylighting analysis. You have some success rendering exterior views of your building, though you are a bit disappointed by the flat shading produced by the CAD software, which gives your renderings the sort of cheesy look so familiar in computer graphics. You do learn how to set the solar position, though, and you are emboldened to attempt rendering the interior for your daylight study.

Much to your dismay, you find that no matter how hard you try, you cannot get anything even remotely believable for your interior views. You finally decide that the CAD software is just not up to the task, and look into some of the other rendering programs at your disposal. You have heard good things about 3D Studio, so you make use of the export and import options to get your model over to this package and start to play around with it. First, you struggle for some time to get the sun in a known position, since the coordinate system is different and there is no clear mechanism for getting the right kind of light source in the right place. Finally, you get yourself reoriented and generate a view of the interior. Although the results

are an improvement over the CAD renderings, they still look very strange and light is not bouncing around as you would expect There is a sun patch on the floor which you expected but no light from this patch is reflected to the rest of the room In fact, the rest of the room appears to have a constant illumination that is unrelated to the light coming in (You try a number of sun positions to verify this hypothesis)

After spending some time with the 3 D Studio manual, you decide that the only way to get the effects you are looking for is to create what are called "ambient lights " invisible sources of illumination that brighten up those parts of the room you expect to be bright You experiment with these imaginary sources for a while until you get some results that you think are worth showing to your instructor Your instructor looks at them then asks you a very annoying question "How do you know this is what it will look like?"

You think about this for a moment before realizing that all you have done is cre ate a rendering that meets your expectations! In fact, you have learned nothing about daylight in the process and you have no real confidence that the actual space will look anything like your rendering Since the purpose of a daylight study is to determine how well a building lets light into its interior this method of rendering is useless because it is not predictive It may be photorealistic since it looks as if it *could* be a real photograph but it isn t *accurate* because it has no physical basis in reality Light does not interact in your rendering system the same way it would in a real environment so the results are not true to reality In fact, you had to introduce completely nonphysical nonexistent sources into the model just to get it to look reasonable you spent a lot of extra time and gained no new insights in the process

Fortunately, you have another option Using the *Radiance* export facility of DesignWorkshop, you can render your model with a valid lighting visualization program Between the reference manual on the CD ROM and the short tutorial at the end of this chapter you can learn enough about the programs and material definitions to complete your exported model and generate some simple renderings From Chapter 6 Daylight Simulation, you can learn the basics of accurate daylight calculation, and you will soon be generating some very nice renderings of your interior, renderings that not only look great but are predictive of the way the real space would appear As a bonus you can also determine accurate daylight factors at var ious points in the room and your exterior renderings will look better as well

This story illustrates the difference between *photorealistic rendering* and *lighting visualization* The former is useful in situations where you only want to fool the audience into thinking it s real The latter is what s needed when the appearance in the rendering must match actual physical conditions An additional benefit of lighting visualizations is that they often look more realistic as well since they do in fact correspond much better to reality

1.1.1 Requirements for Lighting Visualization

The first requirement for a valid lighting visualization program is that it correctly solve the *global illumination* problem. Specifically it must compute the ways light bounces among the various surfaces in the 3D model. If absolute quantities are desired from the simulation, it must further perform its computation in *physical units*, such as units of radiance or radiant exitance (radiosity).

The second requirement, which is equally important, is that the *local illumination* model also adhere to physical reality. This model describes the way light is emitted, reflected, and transmitted by each surface. Many lighting visualization programs are based on the *radiosity method* [Ash94] [SP94], which typically models surfaces as ideal Lambertian diffusers. This is at best a gross simplification, but it is a very convenient one to make computationally speaking. The best methods include specular and directional-diffuse reflection as well as in *Radiance*. (Note: Do not confuse the units with the methods named after them. See the Glossary for further explanations.) Most important, the local illumination model must include an accurate simulation of emission from light sources, because if this is not done correctly, nothing done afterward can save the result.

Past these basic requirements, there are some important practical issues to consider. Although opinions differ, we believe that the following goals must be met by any useful lighting visualization system, and that these capabilities are intrinsic to *Radiance*.

- **Accurately calculates luminance and radiance.** Luminance is the photometric unit that is best correlated with what the human eye actually sees. Radiance is the radiometric equivalent of luminance, and is expressed in SI (Standard International) units of watts/steradian/m^2. *Radiance* (the software) endeavors to produce accurate predictions of these values in modeled environments, and in so doing permits the calculation of other, derived metrics (for all metrics are derivable from this basic quantity), as well as synthetic images (renderings).
- **Models both electric light and daylight.** Since *Radiance* is designed for general lighting prediction, we wish to include all important sources of illumination. For architectural spaces, the two critical sources are electric light and daylight. Modeling electric light accurately means using measured and/or calculated output distribution data for light fixtures (luminaires). Modeling daylight accurately means following the initial intense radiation from the sun and redistributing it through its various reflections from other surfaces, and scattering from the sky. (Section 3.1 demonstrates the use of IES luminaire data and shows how to set up daylight simulations.)

- **Supports a variety of reflectance models** The accuracy of a luminance or radiance calculation depends critically on the accuracy of the surface reflectance model because that determines as much as the illumination how light will be returned to the eye *Radiance* includes some 25 different surface material types one of which is an arbitrary bidirectional reflectance transmittance distribution function (BRTDF) Each material type has several tunable parameters that determine its behavior and many have procedural and data inputs as well In addition these basic materials can be combined in all manners with 12 different pattern and texture types and even with each other Most important, every material type is based on reasonable approximations to the physics of light interaction with particular surfaces rather than derived with the more prevalent motive of algorithmic convenience
- **Supports complicated geometry** Great efforts are made in *Radiance* to minimize the impact of complicated geometry on the memory and processing requirements Storage complexity increases linearly with the number of surfaces and computational complexity increases sublinearly, on the order of the cube root of the number of surfaces or less To further reduce the memory overhead of complicated scenes *Radiance* employs *instancing* to maintain a list of repeated objects and their occurrences in the scene Using this technique it is possible to model scenes (such as a forest) with millions of surface primitives in only a few megabytes of RAM
- **Takes unmodified input from CAD systems** One of the basic precepts of *Radiance* is that scene geometry can be taken from almost any source We think it is unreasonable to restrict you to a rendering system for creating your geometry when CAD systems are available for just this purpose We also think it is unreasonable to require you to condition your CAD models by orienting surface normals or meshing surfaces, since this is pointless drudgery and must be repeated if the model is regenerated The one requirement in *Radiance* is that there be some way to associate materials with surfaces and this is more a prerequisite for interesting renderings than it is a *Radiance*-specific requirement

Now that we have outlined what *Radiance* does, let us look at how well it does it

1 1 2 Examples of Lighting Visualization

Plate 1 shows a *Radiance* rendering of a conference room The model for this room was derived by measuring the dimensions of the real space and furnishings shown in Plate 2 The similarity between the two images testifies to the accuracy of the luminance calculation even if no numeric values are shown Plate 3 shows the same image with superimposed isolux contours indicating lines of equal illumination on

room surfaces A lighting designer or architect could use this numerical information to assess the adequacy of the electric lights in simulation before installing them in reality

Figure 1 1 shows a comparison between measured illuminance values under daylight conditions and *Radiance* predictions based on simultaneous measurements of the sun and sky components [Mar95] This attests to the numeric accuracy of the daylight calculation in *Radiance*

Plate 4 shows a *Radiance* rendering of a daylighted office space Plate 5 shows a photo of the actual space, taken under similar conditions The reflectance function of the table was measured with a gloss meter, and these measurements were used in assigning the reflectance properties in *Radiance* Again, the similarity between the two images testifies to the accuracy of the calculation

Plate 6 demonstrates some of the material properties that can be modeled in *Radiance* The candleholders exhibit anisotropic reflection as though the metal had been brushed circumferentially The table also shows anisotropic behavior because of the application of varnish over the woodgrain, which can be seen in the elon gated highlights from two candles The woodgrain pattern was taken from a scanned photograph and staggered with a user-defined coordinate mapping procedure Finally, the silver box displays an anisotropic reflection pattern modeled with another procedure that simulates the effect of carving many S-shaped grooves in the surface Plate 7 shows the same scene rendered with diffuse surfaces such as one might obtain from a view-independent radiosity system

Plate 8 shows the interior of a stadium, which was modeled with AutoCAD and then exported to *Radiance* for rendering The scene contains tens of thousands of surfaces Plate 9 shows the exterior of the same structure The trees were included as instances, each one including many thousands of surfaces but requiring only a few bytes of additional memory

1 2 *Radiance* Tools and Concepts

Radiance is a lighting simulation program that synthesizes images from 3D geometric models of physical spaces The input model describes each surface s shape size location, and composition A model often contains many thousands of surfaces, and is often produced by a separate CAD program Besides arbitrary (planar) polygons *Radiance* directly models spheres and cones Generator programs are provided for the creation of more complex shapes from these basic surface primitives Exam-

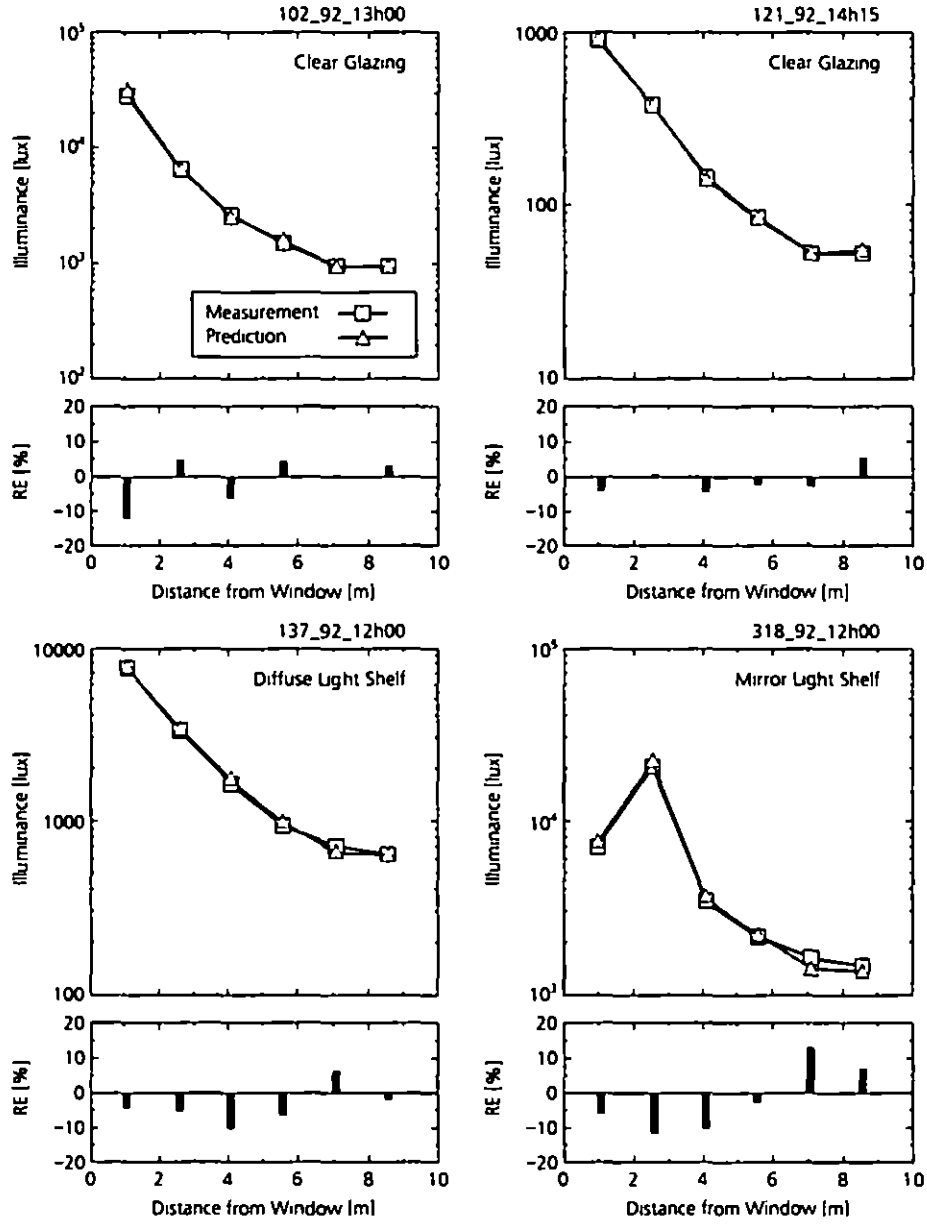


Figure 1 1 An experimental comparison between *Radiance* calculations and real measurements under daylight conditions [Mar95]

ples include boxes, prisms and surfaces of revolution A transformation utility permits the simple duplication of objects and the hierarchical construction of a scene

To be more specific about what *Radiance* does, let's look at some of its features one at a time We will start by breaking the calculation into segments for clearer discussion These are

- *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

1 2 1 Scene Geometry

Scene geometry within the rendering programs is modeled using *boundary representation* (B rep) of three basic surface classes, defined below

- *Polygon* An *n*-sided planar polygon with no fewer than three sides A polygon may be concave or convex as long as it is a well-defined surface (i e no two sides may intersect, though they may coincide) Surface orientation is determined by vertex ordering Vertices read counterclockwise from the front Holes in polygons are represented using *seams* If the vertices are nonplanar a warning is issued and the average plane is used, which may result in cracks in the rendering of adjacent surfaces
- *Sphere* Defined by a center and a radius Its surface may point outward or inward
- *Cone* Includes the truncated right cone the truncated right cylinder, and the ring (a disk with an inner and an outer radius)

Each surface primitive is independent in the sense that there is no sharing of vertices or other geometric information between primitives Besides the above mentioned local geometric types, there is one distant geometric type

- *Source* A direction and subtended angle indicating a solid angle of light entering the environment such as light that might come from the sun or the sky

From this short list of geometric entities you might conclude that the geometric model of *Radiance* is very limited If it were not for the object manipulators and generators you might be right Because generator commands are placed *inline* their output is expanded as more input effectively adding to the geometric entities supported by *Radiance* Some of these commands are listed below

- **xform** Scales, rotates, and moves *Radiance* objects and scene descriptions Combined with the inline command expansion feature permits easy creation of a scene hierarchy for easy modification and manipulation of complex environments Also provides an array feature for repeating objects
- **genbox** Creates a parallelepiped with sharp, beveled, or rounded corners
- **genprism** Creates a truncated prism extruded from a specified polygon along a given vector Optionally rounds corners
- **genrev** Generates a surface of revolution based on a user-defined function and a desired resolution The resulting object is built out of stacked cones
- **genworm** Generates a variable radius "worm" along a user-specified parametric curve in 3D space The object is built out of cones joined by spheres
- **gensurf** Generates a general parametric surface patch from a user-defined function or data set The object is created from optionally smoothed quadrilaterals and triangles
- **gensky** Generates a description of a clear intermediate overcast or uniform sky, with or without a sun
- **replmarks** Replaces special "mark" polygons with object descriptions Useful for separating light sources or detail geometry for manipulation in a CAD system

Although it is possible to create highly sophisticated scene geometries using nothing more than a text editor and the primitives and programs included with *Radiance*, most people prefer to use a CAD program to create their scenes Translator programs for a few different CAD formats are included with the main *Radiance* software Others are available from the ftp site *(ftp //radsite lbl gov/ http //radsite lbl gov/radiance/)* or other sources Listed below are some of the translators we can recommend

- **archicad2rad** converts from ArchiCAD RIB exports to *Radiance* (for Macintosh)
- **arch2rad** converts from Architrion Text Format to *Radiance*
- **arris2rad** converts ARRIS Integra files to *Radiance*
- **dem2rad** converts from Digital Elevation Maps to gensurf input
- **ies2rad** converts from the IES standard luminaire file format to *Radiance*
- **mgf2rad** converts from the Materials and Geometry Format to *Radiance*
- **nff2rad** converts from Eric Haines's Neutral File Format to *Radiance*

- **obj2rad.** converts from Wavefront's *obj* format to *Radiance*
- **radout** converts ACAD R12 to *Radiance* (ADS C add-on utility)
- **rad2mgf** converts from *Radiance* to the Materials and Geometry Format
- **stratastudio** converts Macintosh StrataStudio files to *Radiance*
- **thf2rad** converts from the GDS Things File format to *Radiance*
- **tmesh2rad** converts a basic triangle-mesh to *Radiance*
- **torad** converts from DXF to *Radiance* (AutoLISP routine must be loaded from within AutoCAD)

In addition to the listed surface primitive types, generators manipulators, and translators *Radiance* includes two additional features to make geometric modeling simpler and more efficient

- *Antimatter* Antimatter is a pseudomaterial that can be used to subtract portions of a surface implementing a sort of crude constructive solid geometry (CSG) CSG normally provides all possible Boolean operations between two volumes, including union and intersection However, subtraction is the most useful operation after union, and union is provided by default when two opaque surfaces intersect in *Radiance* (This occurs by virtue of the fact that the inside is not visible from the outside)[1]
- *Instance* An instance is defined in terms of a *Radiance octree* which contains any number of surfaces confined to a region of space Multiple occurrences of the same octree in a given scene will use only as much memory as that required for a single instance plus some small amount of additional memory to store the associated transformations for each instance's location This mechanism is most frequently used for furnishings and the like but can be applied to nearly anything, from building parts to a collection of furniture to trees in a forest *Radiance* scenes including millions of surface primitives have been rendered using this technique

When the geometry has been defined in one or more *scene files* this information is compiled into an octree using the **oconv** command The octree data structure is necessary for efficient rendering, and for including geometry with the instance primitive The oconv program compiles one or more *Radiance* scene description files into an octree file which the rendering programs require to accelerate the ray tracing process In this book the *oct* extension is added as a convention to identify octrees produced by oconv

---

1 Note that there are many limitations associated with the implementation of antimatter Most notably two antimatter objects cannot intersect or chaos will result It is generally wiser therefore, to express the desired object by conventional B rep methods such as collections of triangles

The following example converts three scene description files into an octree input file

```
% oconv materials.rad objects.rad lighting.rad > scene.oct
```

1 2 2 Surface Materials

Although the geometric model is very important, equally important to a rendering algorithm is its representation of materials, which determines how light interacts with the geometry. The most sophisticated geometric model in the world will look mundane when rendered with a simple diffuse plus Phong shading model. (Most radiosity programs are purely diffuse.)

For this reason, *Radiance* pays careful attention to materials, more perhaps than any other rendering system. Version 3.1 has 25 material types and 12 other modifier types. Many modifiers also accept data and/or procedures as part of their definitions. This adds up to unprecedented flexibility and generality, and to a little bit of confusion. It is sometimes difficult to choose from among so many possibilities the primitive that is appropriate for a particular material. Let's look at a few of the choices.

- *Light*. Light is used for an emitting surface, and it is by material type that *Radiance* determines which surfaces act as light sources. Lights are usually visible in a rendering, as opposed to many systems that employ non-physical sources, then hide the evidence. A pattern is usually associated with a light source to give it the appropriate directional distribution. Lights do not reflect.
- *Illum*. Illum is a special light type for secondary sources, sometimes called *impostors*. An example of a secondary source is a window where sky light enters a room. Since it is much more efficient for the calculation to search for light sources, marking the window as an illum can improve rendering quality without adding to the computation time.
- *Plastic*. Despite its artificial sounding name, most materials fall into this category. A plastic surface has a color associated with diffusely reflected radiation, but the specular component is uncolored. This type is used for materials such as plastic, painted surfaces, wood, and nonmetalic rock.
- *Metal*. Metal is exactly the same as plastic, except that the specular component is modified by the material color.

- *Dielectric* A dielectric surface refracts and reflects radiation and is transparent Common dielectric materials include glass water, and crystals A thin glass surface is best represented using the glass type, which computes multiple internal reflections without tracing rays thus saving significant rendering time without compromising accuracy
- *Trans* A trans material transmits and reflects light with both diffuse and specular components going in each direction This type is appropriate for thin translucent materials
- *BRTDfunc* This is the most general programmable material providing inputs for pure specular, directional diffuse, and diffuse reflection and transmission Each component has an associated (programmable) color and reflectances may be different when seen from each side of the surface The disadvantages of using this type are its complexity and the fact that directional diffuse reflections are not computed with Monte Carlo sampling as they are for the built-in types

Most other material types are variations on those listed above, some using data or functions to modify the directional-diffuse component Other variations provide anisotropy (elongation) in the highlights for materials such as brushed aluminum and varnished wood Finally, there are a few other light source materials for controlling this part of the calculation and materials for generating *virtual light sources* by specular reflection or redirection of radiation

All material types also accept zero or more patterns or textures, which modify the local color or surface orientation according to user-definable procedures or data This mechanism is very general and thus also serves as a source of confusion for the user so we will spend some time on the subject in the tutorials

1 2 3 Lighting Simulation and Rendering

Radiance employs a light backwards ray-tracing method extended from the original algorithm introduced to computer graphics by Whitted in 1980 [Whi80] 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 To take a typical example, a 512 by-512-pixel rendering of a bare light bulb in a lightly colored room would take about a month on the world's fastest supercomputer using a naive forward ray tracing method The same rendering takes about three seconds using *Radiance* (Mind you we are talking about a very fast computer here)

The chief difficulty of light-backwards ray tracing as practiced by most rendering software is that it is an incomplete model of light interaction In particular the original algorithm fails for diffuse interreflection between objects which it usually approximates with a constant "ambient" term in the illumination equation Without a complete computation of global illumination, a rendering method cannot produce accurate values and is therefore of limited use as a predictive tool *Radiance* overcomes this shortcoming with an efficient algorithm for computing and caching *indirect irradiance* values over surfaces, while also providing more accurate and realistic light sources and surface materials

Physically accurate rendering of realistic environments requires very careful treatment of light sources, since they are the starting points of all illumination If the direct component is not computed properly, it does not matter what happens afterwards since the calculation is garbage Most rendering systems since they do not care much about accuracy, pay little attention to direct lighting In fact the basic illumination equations frequently disobey simple physical laws for the sake of user convenience allowing light to fall off linearly with distance from a point source or even to remain constant

The details of the local and global illumination algorithms in *Radiance* are described in Part III Calculation Methods, Chapters 10 through 15 Here we will only mention the main rendering programs and what they produce

- rview The interactive program for scene viewing The displayed resolution is progressively refined until the user enters a command to change the view or other rendering parameters This is meant primarily as a quick way to preview a scene, check for inconsistencies and light placement and select views for final high quality rendering with rpict

  The example below selects an initial camera location (-vp vantage point) 10 feet along the negative *y*-axis looking in the positive *y* direction (vd view direction) with up in the positive *z* direction (vu view up) An ambient light level (-av ambient value) is added enabling the shadowed areas to be illuminated in the *scene oct* data set

  ```
  % rview  vd 0 1 0  vp 0  10 0  vu 0 0 1  av  1  1  1 scene oct
  ```

- rpict This rendering program produces the highest-quality raw (unfiltered) pictures A *Radiance* picture is a 2D collection of real color radiance values which unlike a conventional computer graphics image, is also valuable for lighting visualization and analysis The picture is not generally viewed until the rendering calculation is complete and the output has been passed through pfilt for exposure adjustment and antialiasing

The example below creates an image taken from a virtual camera located and oriented by the view file (vf) *scene vf* This view was determined, then written into the *scene vf* file, using functions built into rview The image will be 512 pixels square, and the program will report the status of the rendering progress every 30 seconds The output of rpict, namely the picture, is redirected (>) into the *scene pic* file In this book the *vf* extension is added to view files and *pic* to pictures

```
% rpict  vf scene vf  x 512  y 512  t 30 scene oct > scene pic
```

- **rtrace** This program computes individual radiance or irradiance values for lighting analysis or other custom applications Input is a scene octree (as for rview and rpict) plus the positions of the desired point calculations This program is often called as a subprocess by other *Radiance* programs or scripts

As we have mentioned above rtrace is also employed by other *Radiance* programs to evaluate radiance or irradiance for other types of analysis For example, **mkillum** computes radiance entering through windows skylights, and other "secondary sources" where concentrated illumination can be better represented in the calculation using the illum primitive (Secondary sources are introduced in the tutorial at the end of this chapter and explored in detail in Chapters 6 and 13) Another program that calls rtrace is **findglare**, which locates and quantifies glare sources in a scene Here is a list of similar lighting analysis tools

- **dayfact** An interactive script to compute illuminance values and daylight factors on a specified work plane Output is one or more contour line plots
- **findglare** An image and scene analysis program that takes a picture and/or octree and computes bright sources that would cause discomfort glare in a human observer
- **glare** An interactive script that simplifies the generation and interpretation of findglare results Produces plots and values
- **glarendx** A back end to convert findglare output to one of the supported glare indices Also called glare
- **mkillum** Converts specified scene surfaces into illum secondary sources for more efficient rendering

The findglare program is particularly interesting because it will accept a *Radiance* picture as input as well as the original scene description for rtrace Since a picture in *Radiance* contains physical radiance values it is equivalent to a large collection of rtrace evaluations, and findglare takes advantage of this fact In the next section, we look at some of the other *Radiance* tools tailored specifically for picture processing

1 2 4 Image Manipulation and Analysis

As we mentioned in the preceding section a *Radiance* picture is unlike any other computer graphics image you are likely to encounter First and foremost, the pixel values are real numbers corresponding to the physical quantity of radiance (recorded in watts/steradian/m^2) These values are stored in a compact, 4 byte/pixel run length encoded format (See the File Formats section of the CD-ROM for more details) Second the ASCII header contains pertinent information on the generating commands view options exposure adjustments, and color values that can be used to recover pixel ray parameters and other information needed for various types of image processing

The most essential *Radiance* image manipulation program is pfilt which adjusts the picture exposure and performs antialiasing by filtering the original image down to a lower resolution (This is called *supersampling*) More advanced features include the ability to adaptively filter overbright pixels caused by inadequate sampling [RW94] and add optional star patterns Here is a list of the most important *Radiance* picture manipulators

- **falsecolor** Converts a picture to a false color representation of luminance values with a corresponding legend for easy interpretation (See Plate 3 for an example) Options are included to compute contour lines and superimpose them on another (same size) picture change scales and interpretations and print extrema This program is actually implemented as a C-shell script, which calls other programs such as pcomb and pcompos
- **macbethcal** Calibrates color and contrast for scanned images based on a scan of the Macbeth Color Checker chart May also be used to compute color and con trast correction for output devices such as film recorders Output is a pixel mapping function for pcomb or pcond
- **pcomb** Manipulates pixel values in arbitrary ways based on the functional programming language used throughout *Radiance*
- **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 [LRP97] Also takes calibration files from macbethcal
- **pextrem** Finds and returns the minimum and maximum pixel values and locations
- **pfilt** Performs antialiasing and exposure adjustment A picture is not really finished until it has passed through this filter
- **pflip** Flips pictures left-to right and/or top to-bottom

- **pinterp** Interpolates or extrapolates pictures with corresponding z buffers as produced by rpict Often used to compute in-between frames to speed up walk-through animations
- **protate** Rotates a picture 90 degrees clockwise
- **pvalue** Converts between *Radiance* picture format and various ASCII and raw-data formats for convenient manipulation
- **ximage** Displays one or more *Radiance* pictures on an X11 windows server Provides functions to query individual and area pixel values and computes ray origins and directions for input to rtrace

In addition, there are many programs to convert to and from foreign image formats, such as AVS, PICT PPM, Sun rasterfile PostScript and Targa. These programs have names of the form ra_*fmt*, where *fmt* is the commonly used abbreviation or filename extension for the foreign image format For example, ra_ppm converts to and from Poskanzer Pixmap formats In most cases, reverse conversions (importing into *Radiance*) are supported by the same program with a -r option However, a few reverse conversions are too difficult or cumbersome and are not supported This is the case for the Macintosh PICT and PostScript formats In other cases not all representations within the defined format are recognized such as TIFF which contains almost too many data tags to enumerate, including a raw FAX type—the data stream sent over a phone line!

1 2 5 Integration

Having all these individual tools provides great flexibility but the number of commands and options can overwhelm the casual user Even an experienced user who understands most of what is going on does not want to be bothered with constantly having to think about the details We therefore introduce a few executive programs to simplify the rendering process The 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 Through the setting of intuitive control variables in a short ASCII file, rad sets calculation parameters and options for rview, rpict, and pfilt, and also automatically runs mkillum and updates the octree and output pictures with changes to the scene description files
- **trad** This is a graphical user interface (GUI) built on top of rad using the Tcl/Tk package [Ous94] To the utility of rad it adds process tracking, help screens and image file conversions

- **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 these tools within the UNIX *Radiance* distribution there are a few other systems that integrate *Radiance* in CAD or other environments and we should mention them here

- *ADELINE* A collection of CAD simulation and visualization tools for MS DOS systems, which includes a DOS version of *Radiance* Integration between components is of variable quality but it does include a good translator from DXF format CAD files and it includes LBNLs SUPERLITE program in addition to *Radiance* This package is available from LBNL and other contributors See the Website *radsite lbl.gov/adeline/index html* for details
- *ddrad* A user interface based on AutoCAD which includes the ability to export geometry and define *Radiance* materials interactively It was written by Georg Mischler and friends and is available free from the Website *www schorsch com/autocad/radiance html*
- *GENESYS* A lighting design package from the GENLYTE Group It runs on MS DOS computers It includes an earlier DOS version of *Radiance* and has a nice user interface for designing simple layouts with a large catalog of luminaires
- *SiView* An advanced, integrated system featuring *Radiance* for MS DOS and Windows platforms It is available from Siemens Lighting in Traunreut, Germany It requires the separate purchase of both AutoCAD and ADELINE

Other integrated systems have been created with *Radiance* but we are not aware of any that are publicly available at the time of this writing

Next we present a short tutorial which demonstrates the essential commands and techniques of the system

1 3 Scene 0 Tutorial

This tutorial is designed to give a quick introduction to the system We do not go into much depth because our purpose is to touch on as many aspects of the system as possible in a short space The tutorials in the chapters that follow will provide a more complete learning experience and are recommended to all readers who wish to use the system in a serious way If you find the condensed style of the following tutorial too confusing you may wish to skip to Chapter 2 and return to this later

We assume a certain amount of familiarity with the UNIX operating system and its text editing facilities You will need the *Radiance* reference manual on the CD-ROM to understand the following examples of scene creation and program interaction Text in *italics* is variable input

1 3 1 Input of a Simple Room

In this example, we will use a text editor to create the input for a simple room containing a box, a ball and a light source In most applications a CAD system would be used to describe a scene s geometry, which would then be combined with surface materials light fixtures, and (optionally) furniture To get a more intimate understanding of the input to *Radiance*, we will start without the advantages of a CAD program or an object library

The scene we will be working toward is shown in Figure 1 2 It is usually helpful to start with a simple drawing showing the coordinate axes and the relative locations of major surfaces

The minimum input required to get an image is a source of illumination and an object to reflect light to the "camera "[2] We will begin with two spheres, one emissive and the other reflective First we define the materials then the spheres themselves Actually, the order is important only insofar as each modifier definition (i e , material) must appear before its first reference (Consult the *Radiance* manual for an explanation of the primitive types and their parameters) Start your favorite text editor (vi in this example) to create the following file, called *room rad*

```
% vi room rad
#
#  My first scene
#

#
# The basic primitive format is
#
# modifier TYPE identifier
# number_string_arguments [string arguments    ]
# number_integer_arguments [integer arguments    ]
# number_real_arguments [string real    ]
#
```

2 In fact a *Radiance* renderer can be thought of as an invisible camera in a simulated world

```
# The special modifier "void  means no modifier
# TYPE is one of a finite number of
# predefined types  and the meaning of
# the arguments following is determined by
# this type  (See Radiance Reference
# Manual on the CD ROM for details)
# The identifier may be used as a modifier later
# in this file or in files following this one
# All values are separated by white
# space (spaces  tabs  newlines)
#
# this is the material for my light source

void light bright
0
0
3  100  100  100
#^ r_radiance g_radiance b_radiance

# this is the material for my test ball

void plastic red_plastic
0
0
5   7   05   05   05   05
#^  red  green  blue  specularity  roughness

# here is the light source

bright sphere fixture
0
0
4  2  1  1 5   125
#^  xcent  ycent  zcent  radius

# here is the ball

red_plastic sphere ball
0
0
4   7  1 125   625   125
```

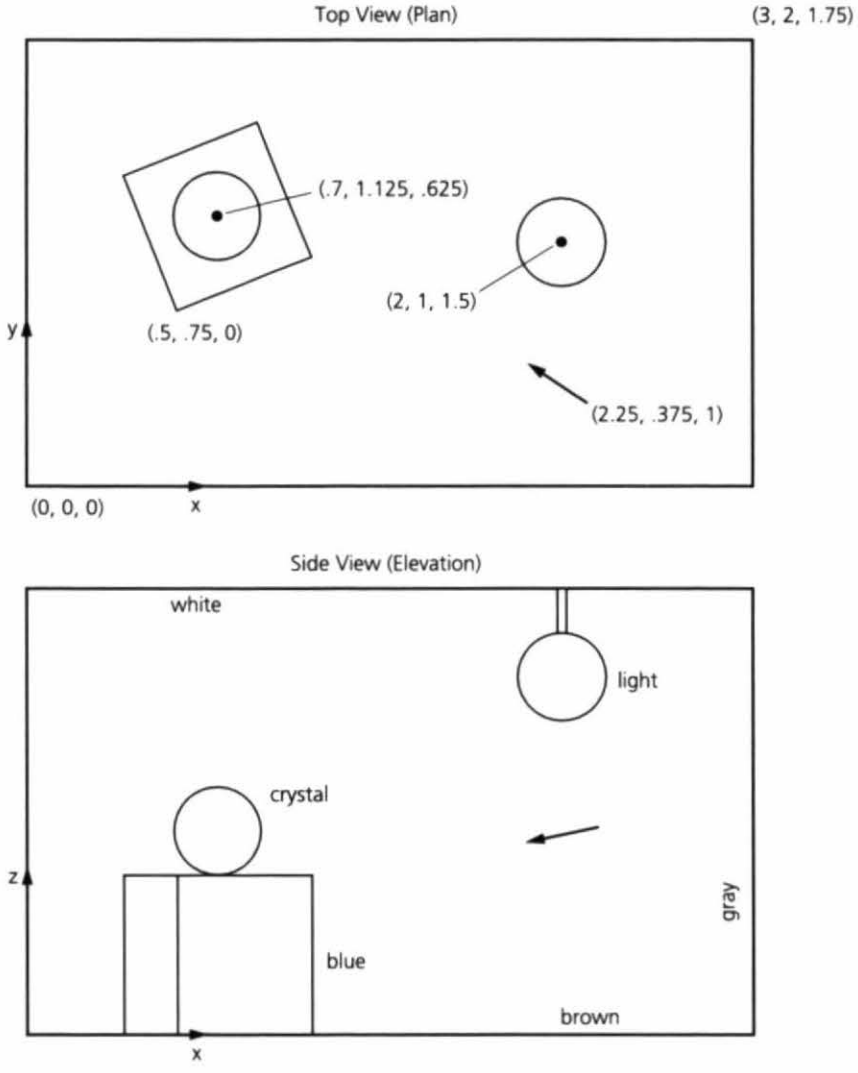


Figure 1.2 A simple room with a block, a ball, and a light source.

Now that we have a simple scene description, we may look at it with the interactive viewing program, rview. First, however, we must create the octree file that will be used to accelerate the rendering process. To do this, type the following command:

```
% oconv room.rad > test.oct
```

Note that the extension *rad* and *oct* are not enforced by the program but are merely a convenience to aid the user in identifying files later The command getinfo can be used to get information on the origin of binary (unviewable) files created by *Radiance* utilities Try entering the command

```
% getinfo test oct
```

The usefulness of such a function will be apparent when you find yourself with a dozen files called *test?.pic*

To make an image of our scene we must select a suitable set of view parameters telling *Radiance* where to point its camera To simplify our example we will use the same starting position for all our renderings and change views only once rview is started

```
% rview  vp 2 25  375 1  vd  25  125  125  av  5  5  5 test oct
```

The -vp option gives the view point the -vd option gives the view direction vector The -av option specifies the amount of light globally present in the scene, permitting portions of the scene that are not illuminated directly to be visible Rview has many more options and their default values may be discovered using

```
% rview  defaults
```

You should start to see an image of a red ball forming on your screen Take this opportunity to try each of the rview commands, as described in the manual If you make a mistake in a view specification use the last command to get back to where you were It is probably a good idea to save your favorite view using the following command from within rview

```
view default vf
```

You can create any number of viewfiles with this command, and retrieve them with

```
last viewfile
```

If you look around enough you may even be able to see the light source itself Unlike those in many rendering programs the light sources in *Radiance* are visible objects This illustrates the basic principle that underlies the program, which is the simulation of physical spaces Since it is not possible to create an invisible light source in reality, there is no reason to do it in simulation

Still, there is no guarantee that the user will create physically meaningful descrip tions For example we have just floated a red ball next to a light source somewhere in intergalactic space In the interest of making this scene more realistic let s enclose the light and ball in a room by adding the following text to *room rad*

```
% vi room.rad
# the wall material

void plastic gray_paint
0
0
5   5   5   5  0  0

# a box shaped room

!genbox gray_paint room 3 2 1.75  -i
```

The generator program **genbox** is just a command that produces a *Radiance* description. It is executed when the file is read. It is more convenient than specifying the coordinates of four vertices for each of six polygons, and can be changed later quite easily. (See the genbox manual page on the CD-ROM for further details.)

You can now look at the modified scene, but remember first to regenerate the octree.

```
% oconv room.rad > test.oct
% rview -vf default.vf -av  5  5  5 test.oct
```

This is better, but our ball and light source are still floating, which is an unrealistic condition for most rooms. Let's put in a box under the table and a rod to suspend the light from the ceiling.

```
# a shiny blue box

void plastic blue_plastic
0
0
5  .1  .1  .6  .05  .1

!genbox blue_plastic box  .5  .5  .5 \
        | xform -rz 15 -t  .5  .75 0

# a chrome rod to suspend
# the light from the ceiling

void metal chrome
0
0
5   .8   .8   .8   .9  0
```

```
chrome cylinder fixture_support
0
0
7
        2       1       1 5
        2       1       1 75
         05
```

Note that this time the output of genbox was "piped" into another program **xform** (The backslash merely continues the line) Xform is used to move, scale and rotate *Radiance* descriptions Genbox always creates a box in the positive octant of 3D space with one corner at the origin This was what we wanted for the room, but here we wanted the box moved away from the wall and rotated slightly First we rotated the box 15 degrees about the *z*-axis (pivoting on the origin) then we translated the corner from the origin to (5, 75 0) By no small coincidence this position is directly under our original ball

After viewing this new arrangement you can try changing some of the materials—here are a few examples

```
# solid crystal

void dielectric crystal
0
0
5   5   5   5  1 5  0

# dark brown

void plastic brown
0
0
5   2   1   1  0  0

# light gray

void plastic white
0
0
5   7   7   7  0  0
```

To change the ball from red plastic to the crystal defined above simply replace `red_plastic sphere ball` with `crystal sphere ball` Note once again that the definitions of the new materials must precede any references to them Changing the

materials for the floor and ceiling of the room is a little more difficult Since genbox creates six rectangles, all using the same material it is necessary to replace the command with its output before we can make the required changes To do this enter the command directly

```
% genbox gray_paint room 3 2 1 75  1 >> room rad
```

The double arrow >> causes the output to be appended to the end of the file rather than overwriting its contents Now edit the file and change the ceiling material to `white` and the floor material to `brown` (Hint The ceiling is the polygon whose z coordinates are all high And don't forget to remove the original genbox command from the file!)

Once you have chosen a nice view, you can generate a high resolution image in batch mode using the rpict command

```
% rpict  vf myview  av  5  5  5 test oct > test pic &
[PID]
```

The ampersand & causes the program to run in the background so you can log out and go home while the computer continues to work on your picture The bracketed number `[PID]` printed by the C shell command interpreter is the process ID that can be used later to check the progress or kill the program This number can also be determined by the `ps` command

```
% ps
```

The number preceding the rpict command is the process ID If you want to kill the process use the command

```
% kill PID
```

If you only want to get a progress report without killing the process use this form

```
% kill  CONT PID
```

This sends a continue signal to rpict, which causes it to print out the percentage of completion Note that this is a special feature of rpict and will not work with most programs Also note that this works only for the current login session If you log on later on a different terminal (or window), rpict will not send the report to the correct place It is usually a good idea therefore, to give rpict an error file argument if it is running a long job

```
% rpict  e errfile
```

Now sending a continue signal will cause rpict to report to the end of the specified error file Alternatively, you may use the -t option to generate reports automatically at regular intervals You can check the reports at any time by printing the file

```
% cat errfile
```

This file will also contain a header and any errors that occurred

1 3 2 Filtering and Displaying a Picture

If you are running *Radiance* under X11, you can use the ximage program to display a rendered picture Try the following command

```
% ximage  e auto test pic &
```

The e auto option tells ximage to perform a histogram exposure adjustment on the picture, to insure that all areas of the image are visible

You may notice that the pixels are jagged in the original output from rpict This is because the picture has not been *filtered*, and filtering is the principal means of antialiasing in *Radiance* The program pfilt performs this task as well as adjusting the exposure in a linear fashion, which does not disturb the physical meanings of the resultant pixels Try the following command sequence

```
% pfilt  x /2  y /2 test pic > testfilt pic
% ximage testfilt pic &
```

There is a space between the x option and its argument but there is no space between the / character and the 2 This sequence has the effect of reducing our original image size by one half and bringing it into the appropriate brightness range for direct display, without the e auto option

If you wish to print out a picture or convert it to another format a number of conversion utilities are available For example, the program ra_ps will convert a *Radiance* picture to a PostScript file, which may then be sent to a printer Try the command

```
% ra_ps  c testfilt pic | lpr
```

(You may have to substitute another command for lpr to send a PostScript job to your printer) This will print out the filtered picture on a color PostScript printer If your printer does not have color simply leave off the c option for grayscale out-

put If you wish to apply the same kind of dynamic range compression provided by the -e auto option of ximage, you may use the pcond program as follows

```
% pcond testfilt.pic | ra_ps -c | lpr
```

The pcond program offers many advanced features for reproducing scene visibility, and we recommend that you consult the manual page on the CD-ROM for more details

1.3.3 Addition of a Window

Adding a window to the room requires two basic steps The first step is to cut a hole in the wall and put in a piece of glass The second step is to put something outside to make the view worth having Since there are no explicit holes allowed in *Radiance* polygons, we use the trick of coincident edges (making a seam) to give the appearance of a hole The new polygon for the window wall is shown in Figure 1.3

To create the window wall, change the appropriate polygon in the scene file (modified part in italics) If you haven't done so already follow the instructions in the preceding section to change the genbox command in the file to its corresponding polygons so we can edit them

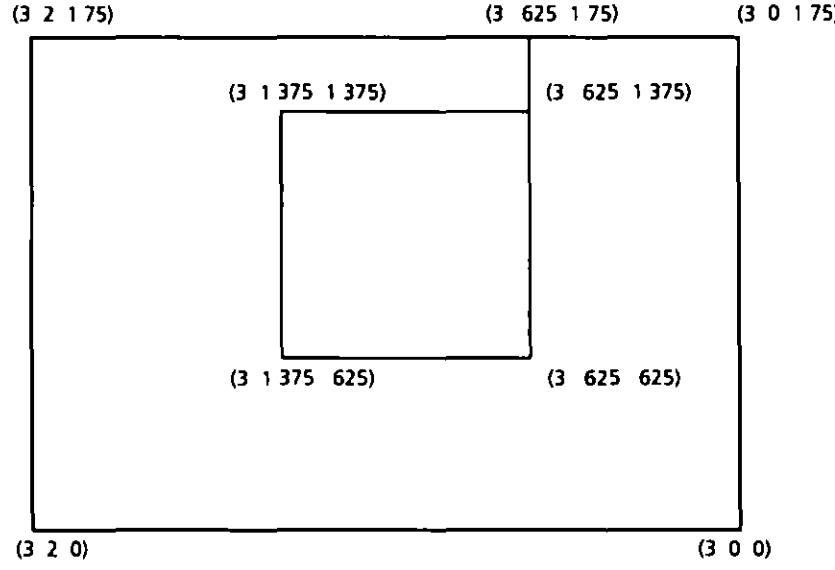


Figure 1.3 The window wall with a hole cut in it

```
% vi room.rad
gray_paint polygon room.5137
0
0
30
        3       2       1.75
        3       2       0
        3       0       0
        3       0       1.75
        3         .625  1.75
        3         .625    .625
        3       1.375     .625
        3       1.375   1.375
        3         .625  1.375
        3         .625  1.75
```

Next, create a separate file for the window (The use of separate files is desirable for parts of the scene that will be manipulated independently, as we will see in a moment)

```
% vi window.rad
# an 88% transmittance glass window has
# a transmission of 96%

void glass window_glass
0
0
3  .96  .96  .96

window_glass polygon window
0
0
12
        3         .625      1.375
        3       1.375       1.375
        3       1.375         .625
        3         .625        .625
```

The vertex order is very important, especially for polygons with holes Normally, vertices are listed in counterclockwise order as seen from the front (the room interior in this case) However, the hole of a polygon has its vertices listed in the

opposite order This ensures that the seam does not cross itself The front of the window should face into our room, since it will later act as a light source and a light source emits only from its front side

The next step is the description of the scene outside the window A special purpose generator, **gensky** will create a description of the sun and sky which will be stored in a separate file The arguments to gensky are the month, day, and hour (local standard time) The following command produces a description for 10 00 AM standard time on March 20 at latitude 40 degrees longitude 98 degrees

```
% gensky 3 20 10  a 40  o 98  m 105 > sky rad
```

The file *sky.rad* contains only a description of the sun and the sky *distribution* The actual sky and ground are still undefined so we will create another short file containing a generic background

```
% vi outside rad
#
# A standard sky and ground to follow
# a gensky sun and sky distribution
#

skyfunc glow sky_glow
0
0
4  9  9 1 15 0

sky_glow source sky
0
0
4 0 0 1 180

skyfunc glow ground_glow
0
0
4 1 4  9  6 0

ground_glow source ground
0
0
4 0 0  1 180
```

We can now put these elements together in one octree file using oconv

```
% oconv outside rad sky rad window rad room rad > test oct
```

Note that the above command causes the following error message

```
oconv  fatal  (outside rad)  undefined modifier  skyfunc"
```

The modifier is undefined because we put *outside rad*, which uses skyfunc before *sky rad*, where skyfunc is defined It is therefore necessary to change the order of the files so that skyfunc is defined *before* it is used

```
% oconv sky rad outside rad window rad room rad > test oct
```

Now let s look at our modified scene using the same command as before

```
% rview  vf default vf  av  5  5  5 test oct
```

As you look around the scene, you will need to adjust the exposure repeatedly to be able to see detail over the wide dynamic range now present To do this wait a few seconds after choosing each new view and enter the command

```
exposure 1
```

or simply

```
e 1
```

All commands in rview can be abbreviated by using one or two letters Addi tional control over the exposure is possible by changing the multiplier factor to a value greater than 1 to lighten or less than 1 to darken It is also possible to use absolute settings and spot normalization (See the rview manual page on the CD ROM for details)

You may notice that, other than a patch of sun on the floor the window does not seem to illuminate the room In *Radiance* certain surfaces act as light sources and others do not Whether or not a surface is a light source is determined by its mate rial type Surfaces made from the material types light illum, spotlight, and glow will act as light sources, whereas surfaces made from plastic metal glass, and other material types will not In order for the window to directly illuminate the room it is therefore necessary to change its material type We will use the type illum because it is specially designed for "secondary" light sources, such as windows and other bright objects which are not merely emitters but have other important visual prop erties An illum will act as a light source for parts of the calculation but when viewed directly will appear as if made from a different material (or disappear altogether)

Rather than modify the contents of *window rad*, which is a perfectly valid description of a nonsource window, let s create a new file, which we can substitute during octree creation, called *srcwindow rad*

```
% vi srcwindow rad
#
# An emissive window
#

# visible glass type for illum

void glass window_glass
0
0
3  96  96  96

# window distribution function
# including angular transmittance

skyfunc brightfunc window_dist
2 winxmit winxmit cal
0
0

# illum for window  using 88% transmittance
# at normal incidence

window_dist illum window_illum
1 window_glass
0
3  88  88  88

# the source polygon

window_illum polygon window
0
0
12
        3         625       1 375
        3       1 375       1 375
        3       1 375         625
        3         625         625
```

You should notice a couple of things in this file The first definition is the normal glass type `window_glass` which is used for the alternate material for the illum `window_illum` Next is the window distribution function which is the sky distribu tion modified by angular transmittance of glass defined in *winxmit cal* Finally comes the illum itself which is the secondary source material for the window

To look at the scene simply substitute *srcwindow rad* for *window rad* in the previous oconv command thus

```
% oconv sky rad outside rad srcwindow rad room rad > test oct
```

You can look at the room at different times by changing the gensky command used to create *sky rad* and regenerating the octree (Although the octree does not strictly need to be recreated for *every* change to the input files it is good to get into the habit until the exceptions are well understood)

1 3 4 Automating the Rendering Process

Until now we have been using the individual *Radiance* programs directly to create octrees and perform renderings By creating a control file, we can leave the details of running the right commands with the right options in the right order to the *Radiance* executive program, rad Similar to the UNIX `make` command rad pays attention to file-modified times in deciding whether or not the octree needs to be rebuilt or other files need to be updated Rad also has a lot of built in "smarts" about *Radiance* rendering options, and improves rendering time and quality by optimizing parameter values based on qualitative information in the control file instead of relying on defaults Finally, rad can quickly find reasonable views without forcing you to think too much in terms of *xyz* coordinate positions and directions

A control file contains a list of variable assignments generally one per line Some variables can be assigned multiple values these variables are given in lowercase Variables that can have only a single value are given in uppercase Here is a minimal control file which we'll call *simple rif*

```
# My first "rad input file
################################
# First  we must specify the  ZONE  for this
# scene  which gives the x  y  and z dimensions
# of our space   The  I  stands for
#  interior"  since we are interested in
# the inside of this space
```

```
ZONE= I  0 3  0 2  0 1 75
# xmin xmax ymin ymax zmin zmax

################################
# Next  we need to tell rad what scene input
# files to use and in what order   For this  we
# use the lowercase variable "scene"  which
# allows multiple values   Literally  all
# the values are concatenated by rad  in the
# order we give them  on the oconv command line

scene= sky rad outside rad
scene= srcwindow rad
scene= room rad

################################
# Technically  we could stop here and let
# rad figure out the rest  but it is very
# useful to also give an exposure value that
# is appropriate for this scene   We can discover
# this value from within rview using the "e ="
# command once we have found the exposure level
# we like   For the interior of our space
# under these particular lighting conditions
# an exposure value of 0 5 works well

EXPOSURE= 0 5
# This could as well have been " 1" (f stops)
```

Once we have this simple input file, we can start using rad to run our commands for us, as in this example

```
% rad  o x11 simple rif
```

The -o option tells rad to run rview under X11 instead of creating pictures (the default action) using rpict If you are using a different window system then you should substitute the appropriate driver module for x11 To discover what modules are available with your version of rview, type

```
% rview  devices
```

Once started, rad shows us the commands as it executes them first oconv then rview

Since we didn t specify a view in our control file rad picks one for us which it calls `X` This is one of the standard views and it means "from the maximum *x* posi tion " As another example the view `yZ` would mean "from the minimum *y* and maximum *z* position " The actual positions are determined from the `ZONE` specifi cation and are just inside the boundaries for an interior zone and well outside the boundaries for an exterior zone (Please take a few moments at this time to consult the rad manual page on the CD-ROM under "view," to learn more about these standard identifiers) We could have selected a different standard view on the com- mand line using the `v` option, as in this example

```
% rad  o x11  v Zl simple rif
```

This specification gives us a parallel projection from `Z`, the maximum *z* position (i e a plan view) Rather than executing another rad command we can get the same view functionality from within rview using the `L` command (This is a single letter command, corresponding roughly to the "last" command for retrieving views from files explained earlier) This command actually consults rad using the current control file to compute the desired view The complementary `V` command appends the current view to the end of the control file for later access and batch rendering For example you can put the default viewpoint into your control file using the rview commands

```
last default vf
```

followed by

```
V def
```

(Shorter view names are better because they end up being part of the picture file name which can get quite long) Move around in rview to find a few different views you like and save them (with sensible names) to the control file using the `V` com- mand If you make a mistake and save a view you later decide you dislike you must edit the control file and manually remove the corresponding line

Looking through the rad manual page you will notice that there are many vari- ables we have left unspecified in our simple control file To discover what values these variables are given, we can use the e option (together with -n and -s to avoid actually doing anything)

```
% rad  e  n  s simple rif
```

Some of these default values do not make sense for our scene In particular, the `VARIABILITY` is not `Low` because there is sunlight entering our space We should also change the `DETAIL` variable from `Medium` to `Low` because our space is really quite sim ple Once we are satisfied with the geometry in our scene, we will probably want to

raise the quality of output from the default value of Low It is also a good idea to specify an ambient file name, so that renderings requiring an indirect calculation will be more efficient We can add the following lines to *simple rif* to correct these problems

```
# We can abbreviate VARIABILITY with 3 letters
VAR= High
# Anything starting with upper or lower L is LOW
DET= L
# Go for a medium quality result
QUAL= Med
# The file in which to store indirect values
AMB= simple amb
```

If we want to create picture files for the selected views in batch mode we can run rad in the background, as follows

```
% rad simple rif &
```

This will of course, echo the commands before they are executed, which may be undesirable for a background job So we can use the "silent" mode instead

```
% rad  s simple rif &
```

Better still we may want rad to record the commands executed along with any error reports or other messages to an error file

```
% rad simple rif >& errs &
```

The >& notation is recognized by the C-shell to mean "redirect both the standard output and the standard error to a file " Bourne shell users should use the following form instead

```
% rad simple rif > errs 2>&1 &
```

1 3 5 Outside Geometry

If the exterior of a space is not approximated well by an infinitely distant sky and ground, we can add a better description to calculate a more accurate window output distribution as well as a better view outside the window Let's add a ground plane and a nearby building to the *outside rad* file we created earlier and call this new file *outside2 rad*

```
# Terra Firma

void plastic ground_mat
0
0
5  28  18  12 0 0

ground_mat ring groundplane
0
0
8
       0       0         01
       0       0       1
       0      30

# A big  ugly  mirrored glass building

void mirror reflect20
0
0
3  15  2  2

!genbox reflect20 building 10 10 2 \
       | xform  t 10 5 0
```

Note that groundplane was given a slightly negative z value This is very important so that the ground does not peek through the floor we have defined The material type *mirror*, used to define the neighboring structure, is special in *Radiance* Surfaces of this type as well as the types *prism1* and *prism2* participate in something called the virtual light source calculation In short, this means that the surfaces of the building we have created will reflect sunlight and any other light source present in our scene The virtual light source material types should be used sparingly, since they can result in substantial growth in the calculation It would be a good idea, in the example given above, to remove the bottom surface of the building (which cannot be seen from the outside anyway) and to change the roof type to metal or some nonreflecting material This can be done using the same manual process described earlier for changing the room surface materials

Now that we have a better description of the outside, what do we do with it? If we simply substitute it into our scene without changing the description of the window illum, the distribution of light from the window will be slightly wrong because

the skybright function describes only light from the sky and the ground not from other structures Using this approximation might be acceptable in some cases, but at other times it is necessary to consider outside geometry and/or shading systems to reach a reasonable level of accuracy There are two ways to an accurate calculation of light from a window The first is to treat the window as an ordinary window and rely on the default interreflection calculation of *Radiance* and the second is to use the program mkillum to calculate the window distribution separately so that we can still treat it as an illum light source Let s try them both

Using the default interreflection calculation is probably easier but, as we shall see it takes a little longer to get a good result in this case To use the interreflection calculation, we modify the scene specification and a few other variables in *simple rif* to create a new control file, called *inter rif*

```
ZONE= I  0 3  0 2  0 1 75
# new exterior description
scene= sky rad outside2 rad
# go back to simple window
scene= window rad
scene= room rad
EXP= 0 5
VAR= High
DET= L
QUAL= Med
# Be sure to use a unique name here
AMB= inter amb
# One bounce now for illumination
INDIRECT= 1
view= def  vp 2 25  375 1  vd   25  125   125
```

To look at the scene with rview simply run

```
% rad  o x11 inter rif
```

Probably the first thing you notice after starting rview is that nothing happens It takes the calculation a while to get going because it must trace many rays at the outset to determine the contribution at each point from the window area. Once rview has stored up some values the progress rate improves, but it never really reaches blistering speed

A more efficient alternative in this case is to use the program mkillum to create a modified window file that uses calculated data values to define its light output distribution Applying mkillum is relatively straightforward in this case Simply create a new control file from *interrif*, and name it *illum rif*, making the following changes

```
ZONE= I  0 3  0 2  0 1 75
scene= sky rad outside2 rad
scene= room rad
# window will be made into illum
illum= window rad
EXP= 0 5
VAR= High
DET= L
QUAL= Med
# Be sure to use a unique name here
AMB= illum amb
# No interreflections necessary with illum
INDIRECT= 0
# Options for mkillum
mkillum=  av 18 18 18  ab 0
view= def  vp 2 25  375 1  vd   25  125   125
```

The -av value given to mkillum is appropriate for the outside, which is much brighter, as suggested by the output of the gensky command stored in *sky.rad* The -ab option is set to 0 because outside the building we do not expect interreflections to play as important a role as they do in the interior (and we are also trying to save some time) To view the scene interactively, we again use rad

```
% rad  o x11 illum rif
```

You will notice that the calculation proceeds much more quickly and even produces a smoother-looking result However, aside from waiting for mkillum to finish, there is an additional price for this speed advantage The contribution from the sun patch on the floor is no longer being considered, since we are not performing an interreflection calculation inside our space The light from the window is being taken care of by the mkillum output, but the solar patch is not In most cases, we endeavor to prevent direct sun from entering the space, and in the morning hours this is true for our model, but otherwise it is necessary to use the diffuse interreflection calculation to correctly account for all contributions Note that the interreflection calculation is turned on automatically when the `QUALITY` variable in the control file is changed to `High`

1.4 Conclusion

By now you should have a fair idea of what *Radiance* has to offer and should even have gained some insight into the way it all works together. If the Scene 0 tutorial left you with some unanswered questions, we recommend that you continue with the Scene 1 tutorial in Chapter 2. After that, the Scene 2 tutorial in Chapter 3 provides some very interesting surprises. Chapter 4 continues with examples of "scripting" in *Radiance*. Part II, Applications (Chapters 5 through 9), gives application specific advice and case studies. Part III, Calculation Methods (Chapters 10 through 15) goes into graphic detail to describe what exactly is going on inside *Radiance*. this is important to the advanced user who wants greater understanding and control, as well as to the graphics researcher who wants to know.

Radiance has been used to visualize the lighting of homes, apartments, hotels, offices, libraries, churches, theaters, museums, stadiums, roads, tunnels, bridges, airports, jets, and space shuttles. It has answered questions about light levels, esthetics, daylight utilization, visual comfort and visibility, energy savings potential, solar panel coverage, computer vision, and circumstances surrounding accidents. If you can imagine it, and you want to know what it will *really* look like, *Radiance* is the tool that can show you.

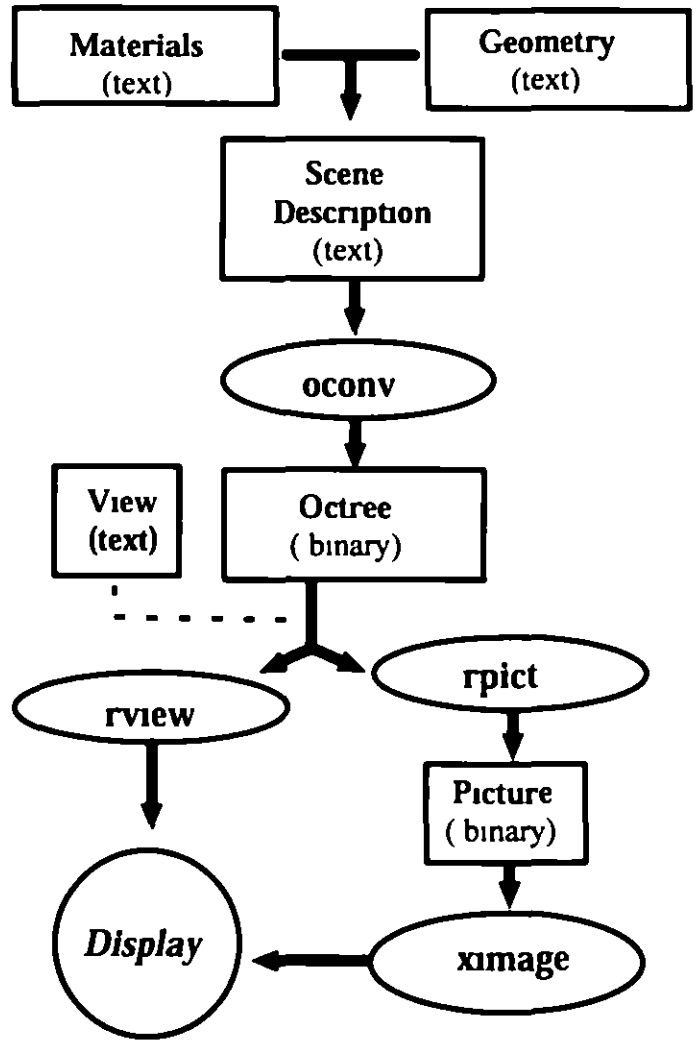
Tutorial by Rob Shakespeare

Tutorial Introduction

Several tutorials have been created to introduce *Radiance* to a variety of users While some are terse and serve the impatient reader who comes armed with a reasonable amount of computing savy others introduce *Radiance* in a more gentle and verbose manner Though the content of these tutorials embrace similar topics the examples range from spheres located in boxes through log cabins to sophisticated models such as an Art Gallery The *Radiance* distribution comes bundled with a few tutorials and several others can be found on the web and in print

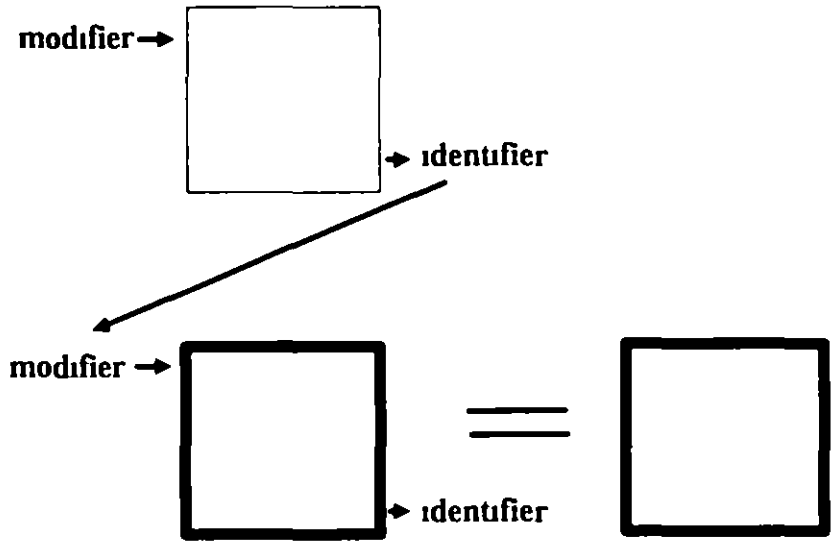
This section makes no attempt to compete with existing tutorials in presenting a comprehensive overview of how to use *Radiance* Rather it will jog along the basic modeling and rendering critical path pausing to glimpse at the more commonly used resources This provides a context from which to engage the advanced use of *Radiance* as detailed throughout the rest of this course

The following schematic illustrates the basic recipe used to make a *Radiance* picture



In the previous schematic the ellipses represent a few of the programs which are part of the *Radiance* suite The image making process begins by your defining the materials and geometry which are then assembled into a scene The oconv program compiles the scene data into an octree In general practice rview is used to interactively view a low quality picture or rpict is used to render a higher quality image which is stored as a *Radiance* picture Ximage enables the viewing of a *Radiance* picture

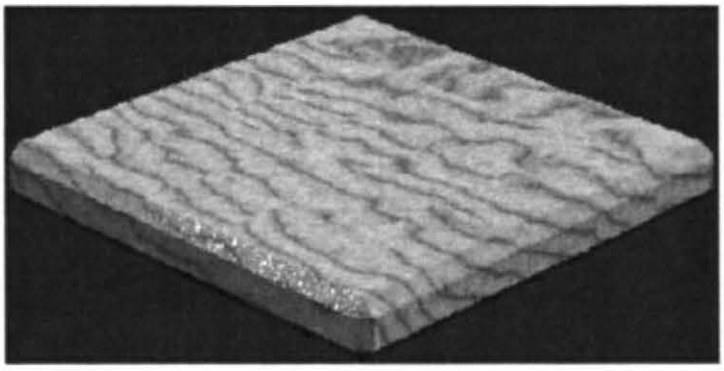
The majority of user effort generally centers on constructing the model This is where specific materials and geometry are manipulated and combined into the surfaces of the scene Radiance employs an inheritance scheme where each entity is assigned an identifier These identifiers can be used to modify other entities and are used to assign a particular material property to a surface primitive



This schematic demonstrates how a bordered square inherits a grid pattern resulting in a bordered square with grid pattern Identifiers are usually descriptive words such as wood_grain or table_leg The word 'void is inserted into the modifier position if no attributes are to be inherited

Modifying a material: inheritance

```
### maple wood tile with waxed finish
void        brightfunc          mottled
4           dirt       dirt.cal -s .01
0
1           .30


mottled     brightfunc          maple
6           zgrain     woodpat.cal      -s .04  -ry 90
0
1           .55


maple       texfunc             grainy
6           xgrain_dx ygrain_dx zgrain_dx  woodtex.cal -s .005
0
1           .075

grainy      plastic             tile_grained_maple
0
0
5           .689 .511 .298  .025 .025

!genbox  tile_grained_maple tile    1 1 .1  -b    .025
```



This figure illustrates the identifier-modifier inheritance process in practice. Without focusing on the specifics of these *Radiance* scene descriptions, we can clearly see the process by which the wood grained tile acquires its surface attributes.

Now that we have actually seen how an object is described, lets glimpse more closely at the scene description primitive whose form is the building block for all materials, surfaces, textures and patterns.

The *Radiance* scene description

Format:

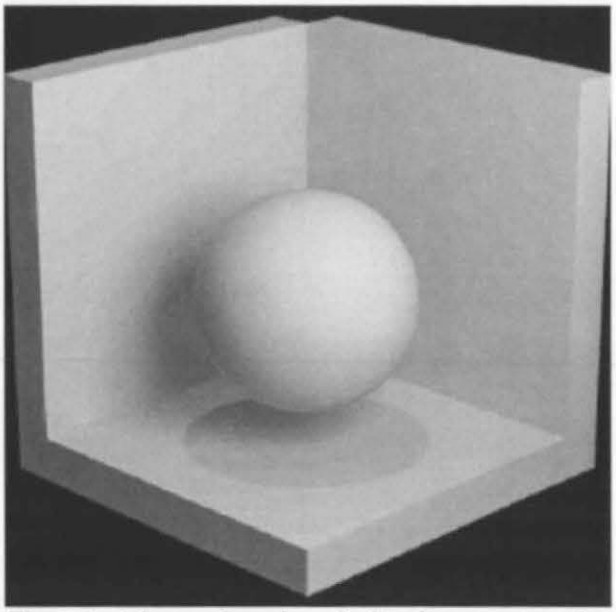
modifier	type	identifier
n	S1 S2 S3 ..Sn	[S is a string or word]
0		[reserved for future integers]
m	R1 R2 R3 ..Rn	[R is a real number]

A specific material primitive:

modifierplastic id
0
0
5 red green blue specularity roughness

A specific material description:

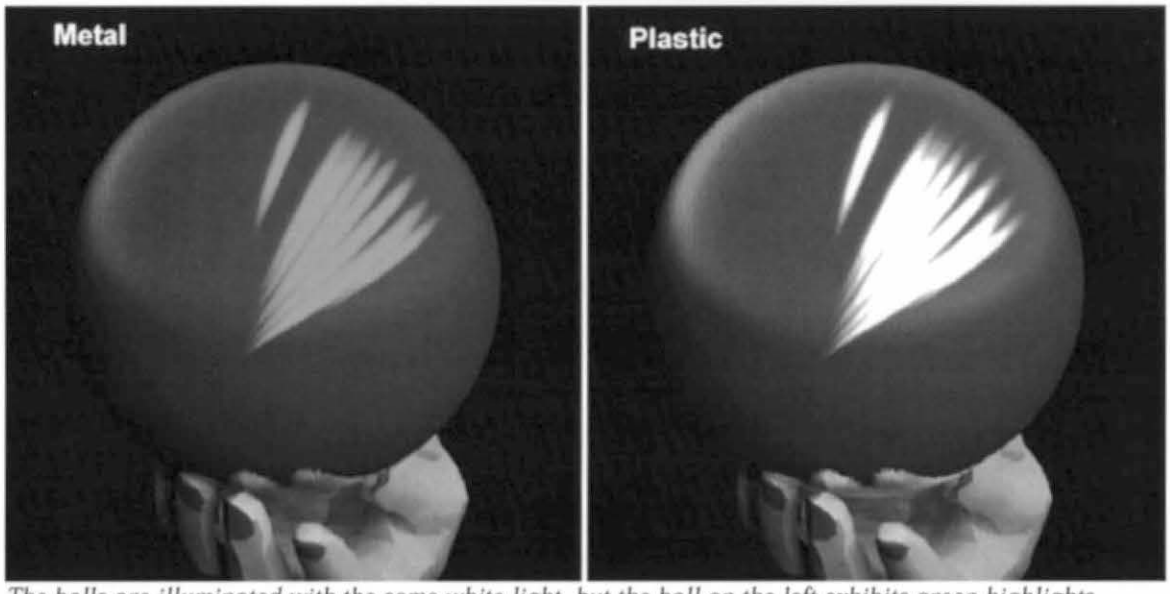
void plastic sand
0
0
5 .9 .5 .2 0 0



The material named sand applied to the geometry of a scene.

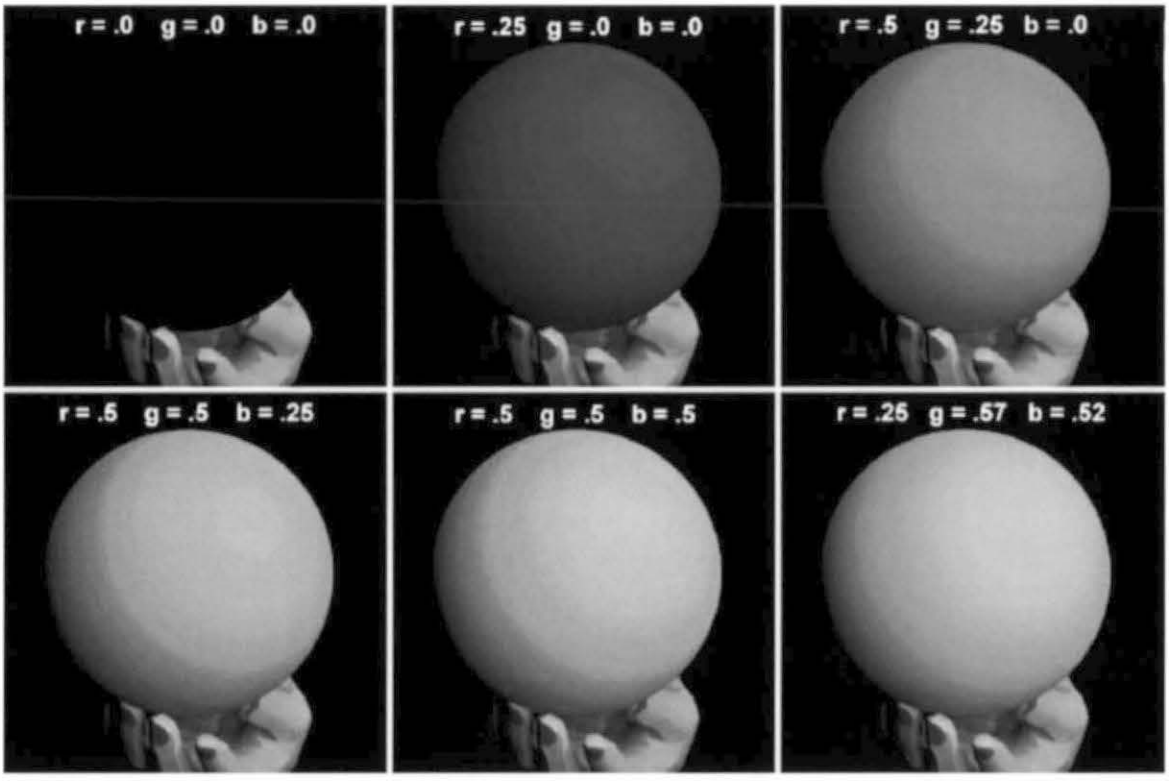
Radiance Materials: The Basic Inventory

Plastic is one of the basic, two sided materials from which many specific surfaces can be created. The most obvious difference between plastic and metal is seen in specular highlights. Plastic surfaces have uncolored highlights while metal surfaces modify the color of highlights.



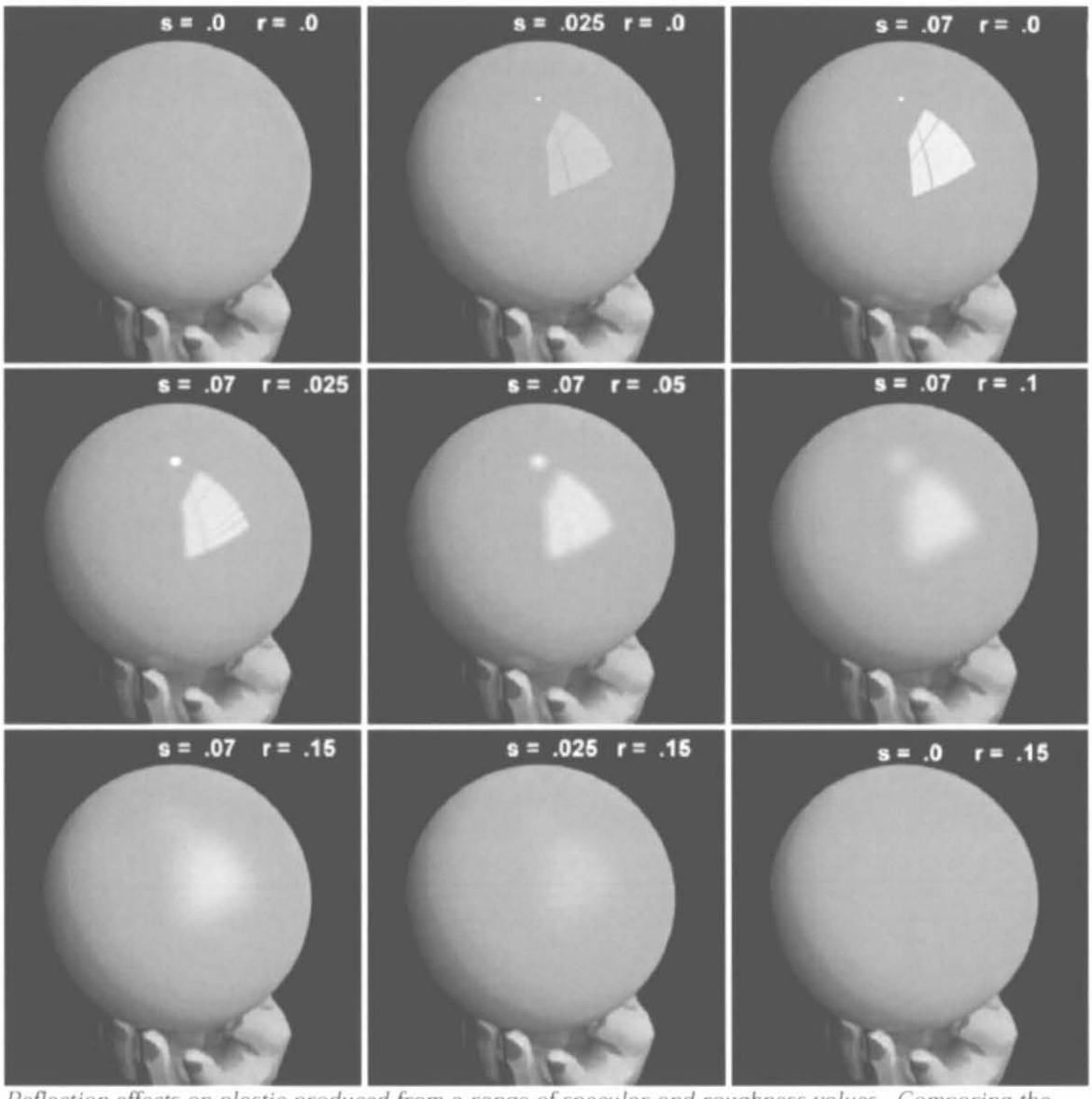
The balls are illuminated with the same white light, but the ball on the left exhibits green highlights. Unlike plastic, materials from the metal family modify the color of highlights.

The color of plastic is defined in the scene description as a combination of red green and blue values ranging between 0 and 1 as demonstrated in the following figure:



Nature does not make 100% reflectors so the r g b values are always less than one. If you accidentally increase a color value to greater than one, the object might begin to glow.

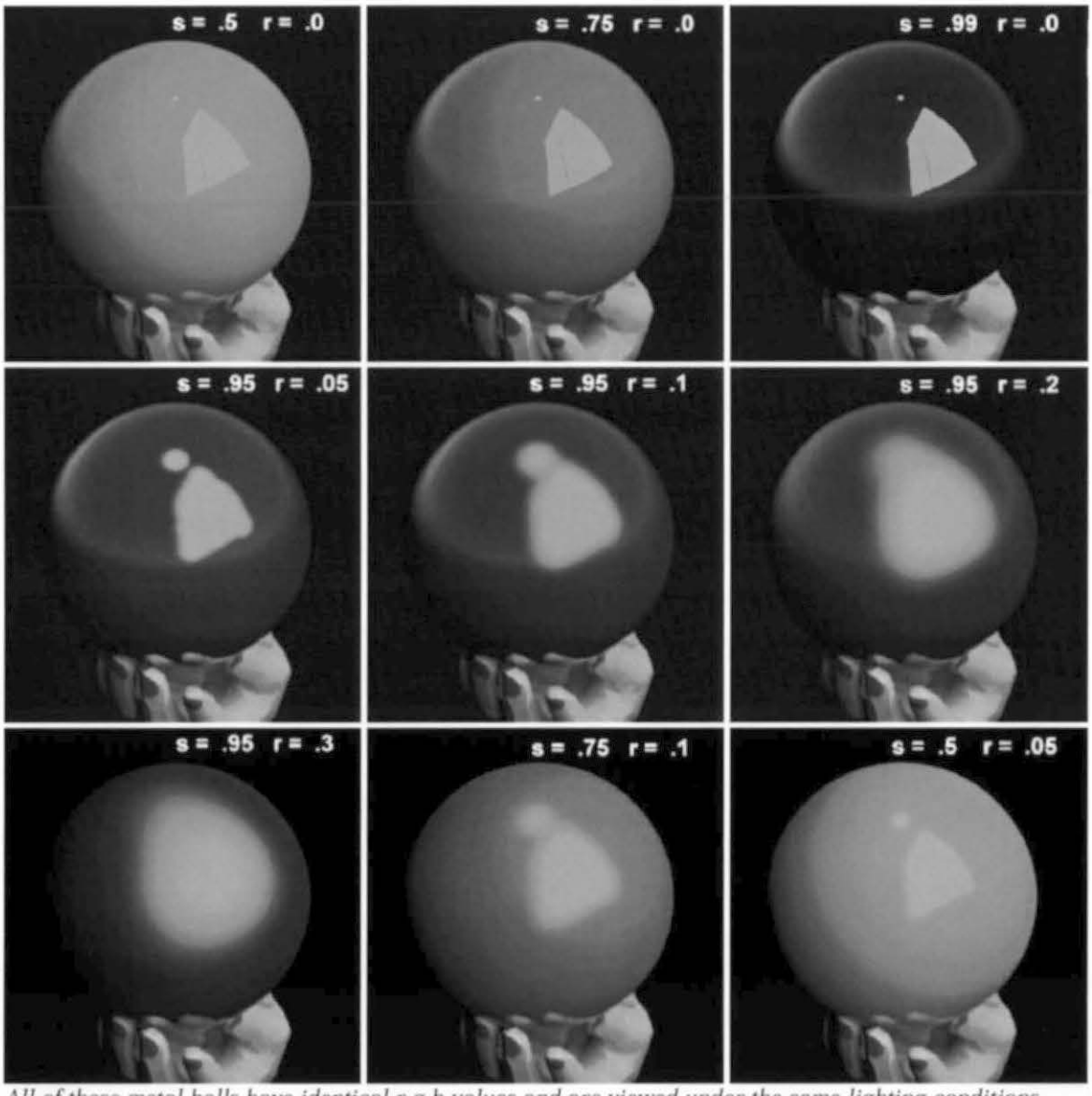
The specular and roughness values combine to produce a wide variety of reflection effects.
Be aware that a plastic surface rarely has a specularity greater than .3 and a roughness of greater than .2. If these values are increased beyond these limits, it is unlikely that you will find a physical match for the surface.



Reflection effects on plastic produced from a range of specular and roughness values. Comparing the upper left and lower right image shows that roughness has no effect if the specularity is set to 0.

Metal specular values generally range between .5 and .99 while the roughness component rarely exceeds .2. The scene description primitive for metal follows:

modifier metal id
0
0
5 red green blue specularity roughness

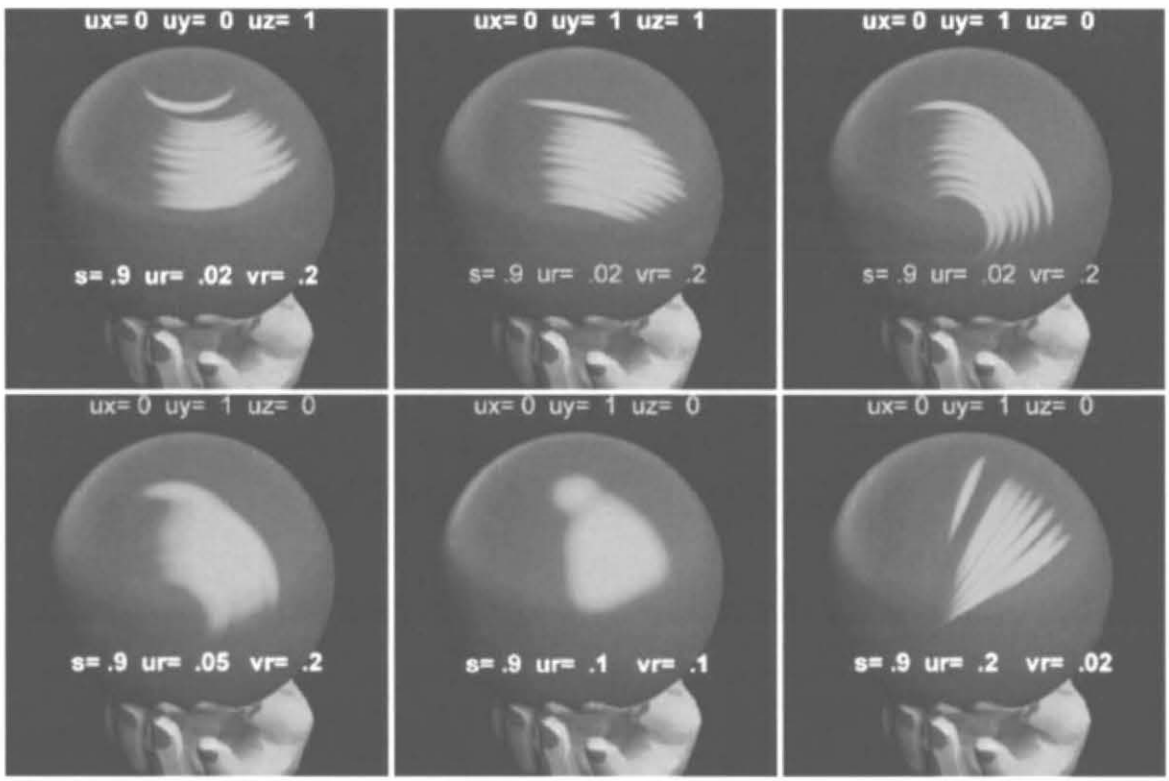


All of these metal balls have identical r g b values and are viewed under the same lighting conditions. Note how the balls in the top row reflect their surroundings and appear to get darker as specularity is increased. As the roughness component is increased across the second row, the reflected shape of the window is diffused and loses definition.

Metal2 and **Plastic2** are anisotropic surfaces which exhibit elliptical instead of round highlights. Examples of this surface effect can be seen when light reflects off brushed aluminum or from a poorly lacquered surface where the brush strokes are evident.

Color and specularity are the same as for plastic and metal, but there are two additional sets of parameters which establish the orientation and shape of the elliptical reflections. The first is an unnormalized vector (ux uy uz) which orients the anisotropy. If no specific function file is required, a "." is located after the vector as a placeholder. The shape of the elliptical highlight is defined by two values which effect the roughness along the orientation vector (ur) and roughness perpendicular to the vector (vr).

```
void              metal2          id
4    ux    uy    uz    .
0
6    red  green  blue  specularity  ur  vr
```



The changes in highlights in the top row of pictures result from redirecting the orientation vector in the metal2 material. The bottom row demonstrates how the shape of the reflection changes by varying the two roughness values. When the roughness values are the same, round reflections result as shown in the center picture..

Plastic2 is defined with a similar scene description primitive:

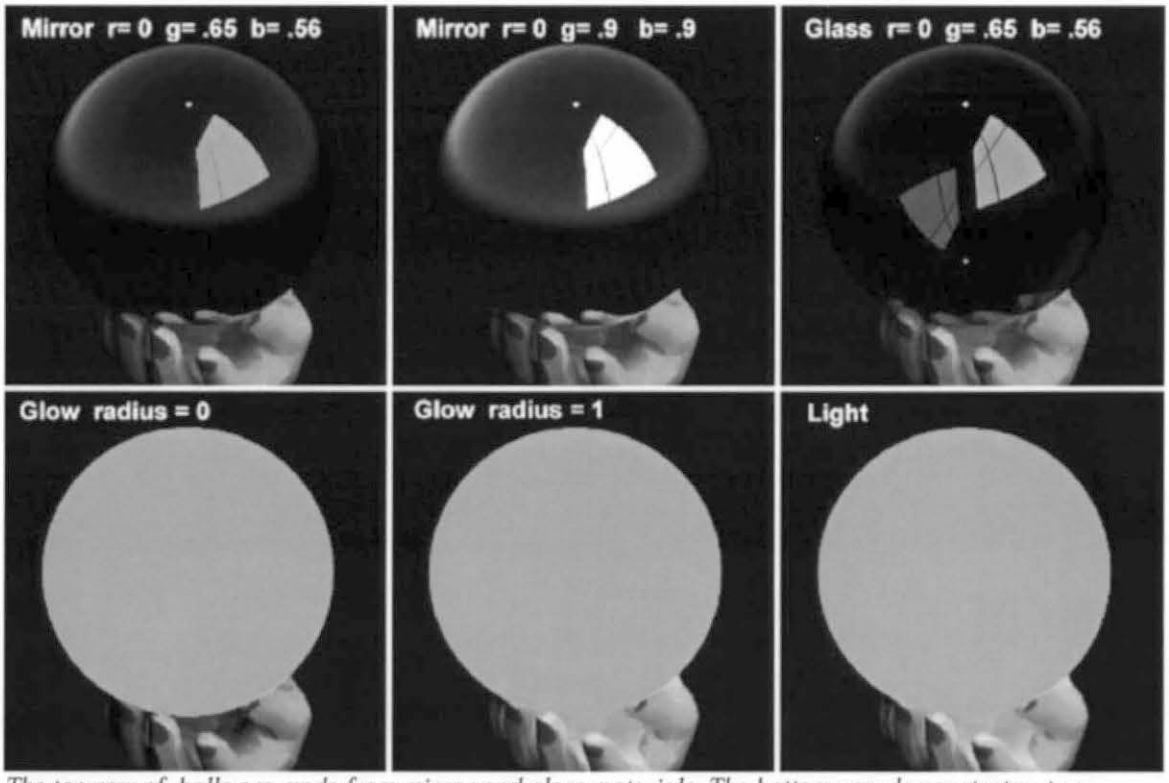
```
void              plastic2        id
4    ux    uy    uz    funcfile_or_"."
0
6    red  green  blue  specularity  ur  vr
```

Glass is often encountered when modeling architectural projects. Windowpanes and other thin glass surfaces are made from this material which has a color variable and fixed refractive index.

void glass id
0
0
3 red green blue

Mirror is another common material, but unlike metal and plastic, it is one sided. The direction of the surface normal determines the mirrored side, and this direction results from the order in which the vertices are listed. The right hand rule is a convenient way to predict the direction of a surface normal in *Radiance*.

void mirror id
0
0
3 red green blue



The top row of balls are made from mirror and glass materials. The bottom row demonstrates two different luminous materials. Note the hand in each of these pictures. The glow material in the lower left does not illuminate the hand. The center ball can illuminate other surfaces within a 1 unit radius of its center. Because the hand is within this 1 foot radius, there is no visible difference between the constrained effect of the glow material and the unconstrained effect of light.

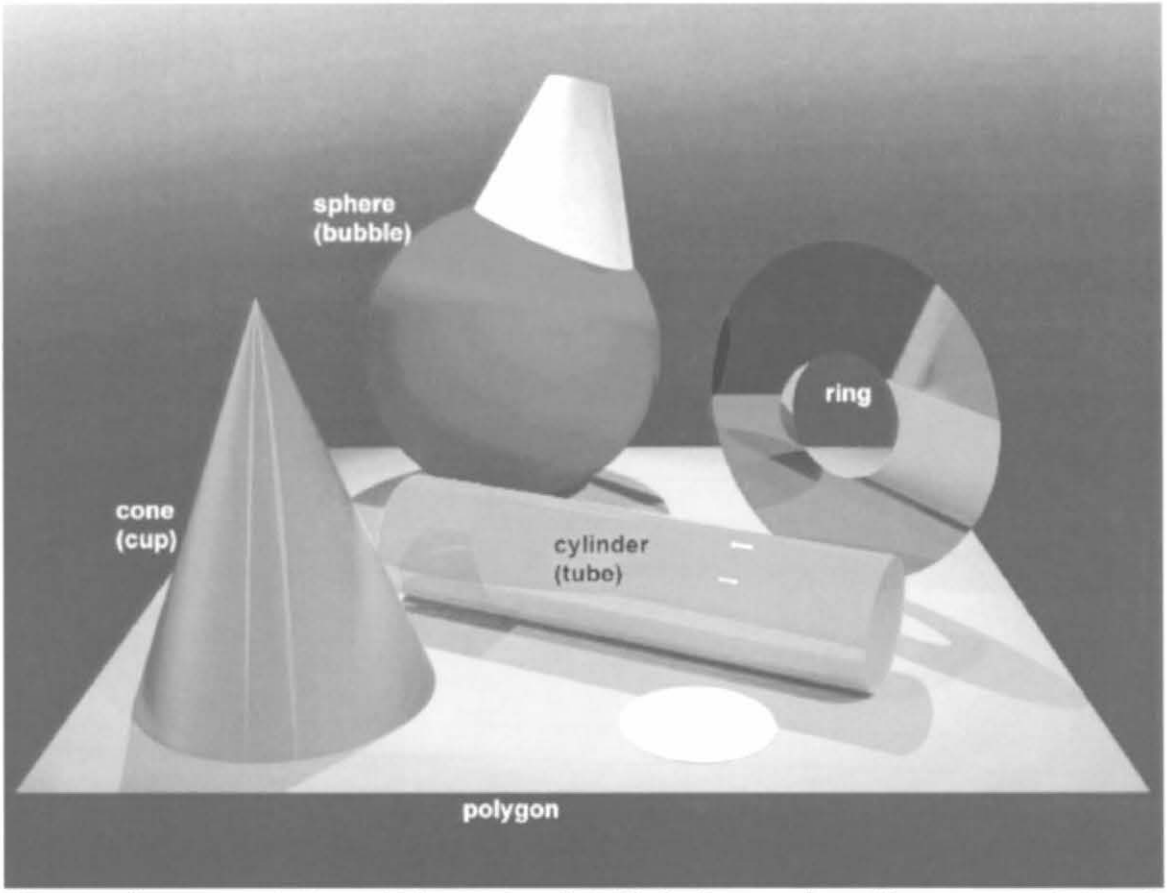
The material **light** enables surfaces to become light sources within a scene. The r g b values of light, in conjunction with the size of the surface area, define the quantity of luminous flux emitted from the surface. A *Radiance* program named lampcolor can be used to determine these r g b values

void light id
0
0
3 red green blue

Glow is similar to light but the range of its effect can be constrained within a radius. Any surface outside of this radius will not be illuminated by the glow material. Again, the lampcolor program is a handy way to establish the glow r g b values.

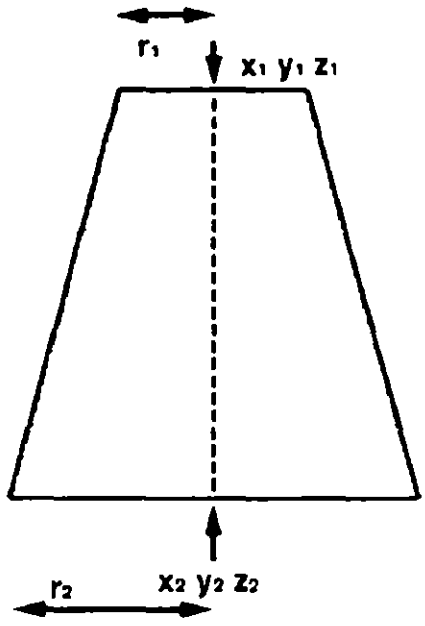
```
void            glow            id
0
0
4    red  green  blue  radius
```

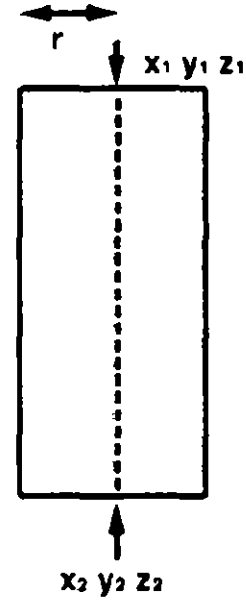
Radiance Geometry Primitives



The suite of geometry primitives which comprise all visible Radiance surfaces. The geometry primitives in parentheses ( ) have inward pointing normals.

Actually, the family of cones includes the cylinder and the ring. The **cylinder** is defined by a beginning and ending vertices along with a radius. The **cone** is also defined by two vertices, but each has its own radius. Finally, the **ring** is defined using one vertices and an orientation vector along with an inner and outer radius.



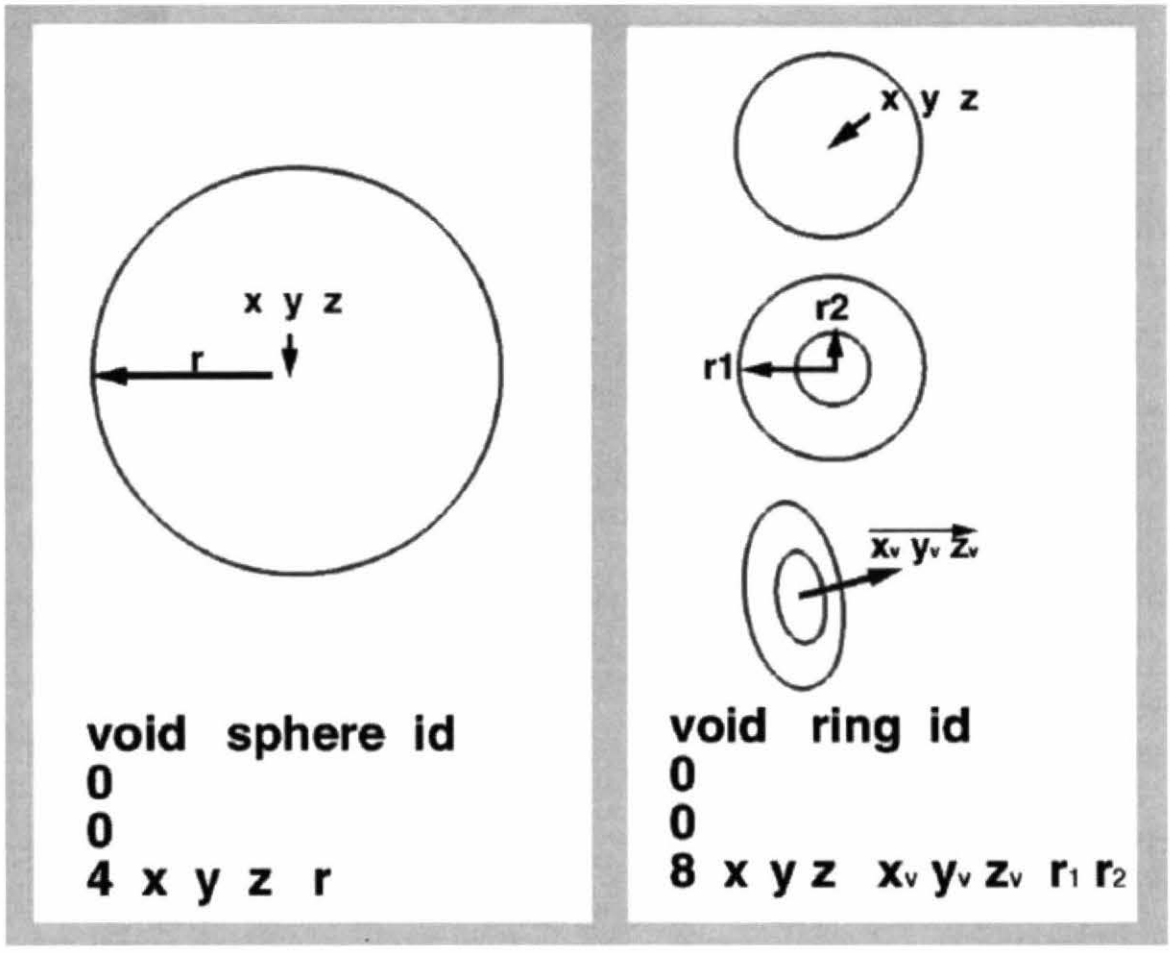


void cone id
0
0
8 x1 y2 z3 x1 y2 z3 r1 r2

void cylinder id
0
0
7 x1 y2 z3 x1 y2 z3 r

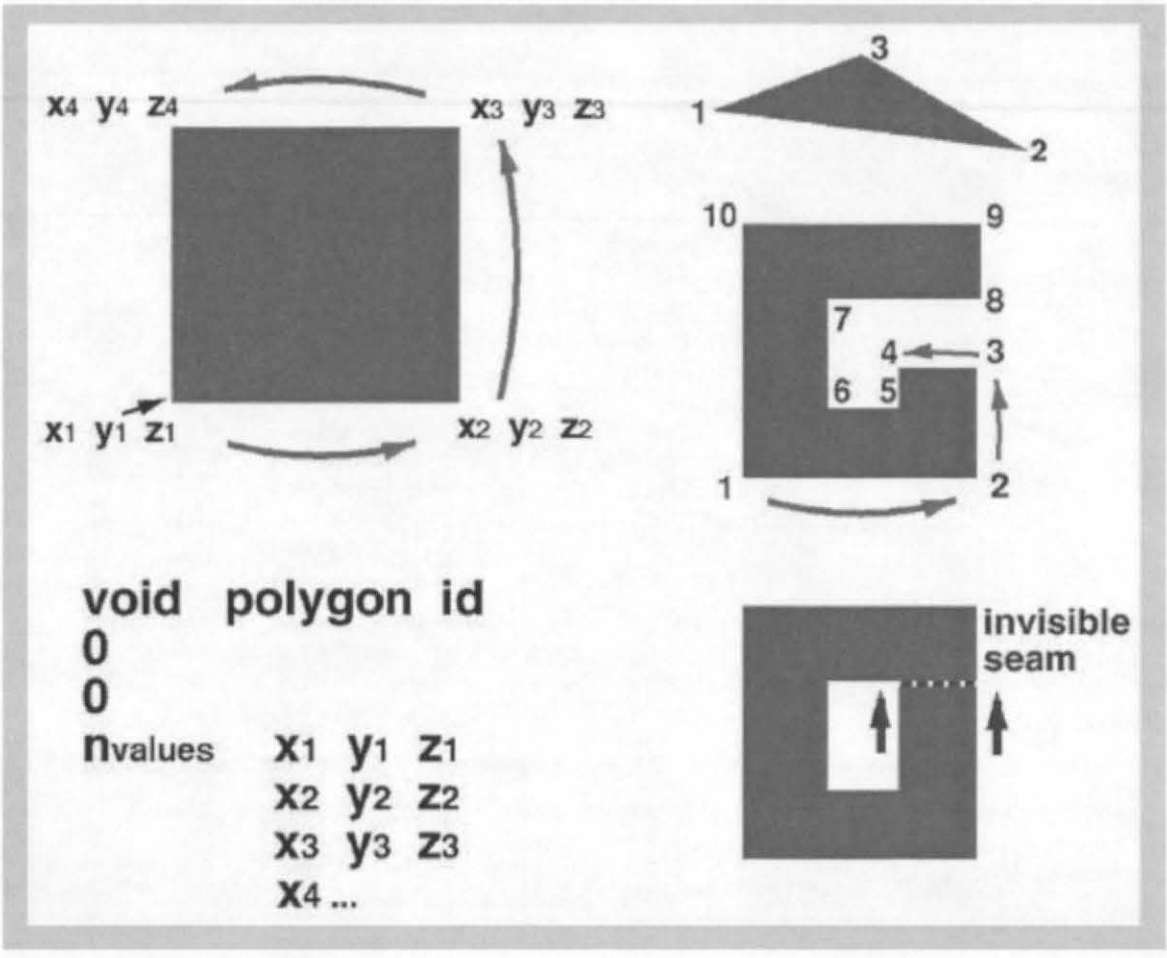
The surface normal of a cone and cylinder points outwards Replace the word cone with cup and cylinder with tube in these scene descriptions to make the normals point inwards Unlike many CAD programs which create huge arrays of polygons (sometimes called 3d faces) when generating similar shapes Radiance defines these geometry primitives with equations

The **sphere** has a very simple definition comprised of a vertices and a radius.



By setting the value of one radius to 0, there will be no hole in the ring. The orientation vector of a ring also determines the surface normal. By setting the value of one radius to 0, the hole in the ring is eliminated.

Finally the **polygon** completes the suite of geometry primitives.



Polygons are planer and defined by at least 3 vertices. The order in which the vertices are listed determines the surface normal, as presented in the top left figure. In this case, the normal points towards you, but if the order were reversed, the normal would be on the other side of the polygon, pointing away from you. The right center and bottom figures illustrate an invisible seam technique which creates the effect of a polygon with a hole in it.

Radiance Polygon Generators

It would be a very tedious task to build a model by typing in one polygon at a time. *Radiance* comes to the rescue by providing several functions which generate polygons for you. The simple polygon generators are presented here.

To create the six polygons which comprise a box, *Radiance* provides the **genbox** function. This function can be called directly from the command line, or by preceding it with an "!", you can locate the function within the scene description file. "!" tells the oconv program that a function follows. The box is build between 0 0 0 and a point which defines the opposite corner (x y z).

!genbox modifier id x y z

The **xform** function can then be used to scale, rotate, translate, mirror and/or array the box.

The following sequence pipes the output of a genbox command through xform to creates a row of four boxes, 2 units apart.

```
!genbox sand boxes 1 1 1  |  xform -a 4 -t 2 0 0 -i 1
```

This could be read: create a 1 foot square box, and make 4 copies of it, each 2 feet along the x axis from the previous. Twenty-four polygons are generated for you.

The **genprism** polygon generator can be though of as extruding a polygon in the direction and distance defined by a length vector. The polygon is defined on the xy plane with pairs of contiguous coordinates.

The following command produces a triangle extruded to 6 units tall, which is then duplicated and mirrored on either side of the y axis:

```
!genprism sand tri 3  4 -1  5 1  3 2  -l 0 0 6 | xform -a 2 -mx -i 1
```

GENPRISM **a polygon generator to extrude more complex shapes than genbox. This is accomplished by providing genprism with a list of contiguous xy points and a 3d extrusion vector.**

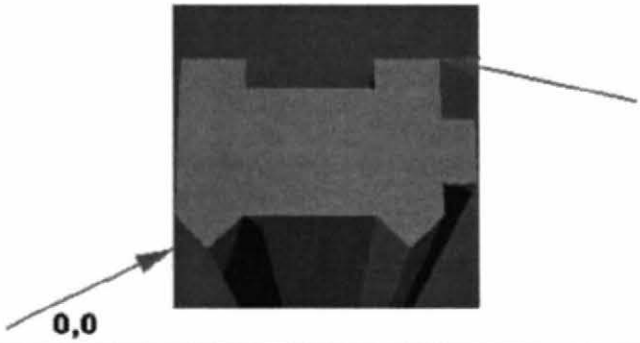
```
!genprism material id n  x1 y1  x2 y2 ... xn yn \
                      -l  xmag ymag zmag
```

The following two statements produce the same object:

```
!genprism brown moulding 4  0 0  .33 0  .33 .3  0 .3 \
                         -l 0 0 7
!genbox   brown moulding .33 .3 7
```

An example of a more complex extrusion :

```
!genprism brown moulding 16  .05 0  .1 .05  .3 .05  .35 0 \
    .4 .05  .4 .1  .45 .1  .45 .2  .4 .2  .4 .3  .3 .3 \
    .3 .25  .1 .25  .1 .3  0 .3  0 .05 -l 0 0 7 | xform -t 2 0 0
```

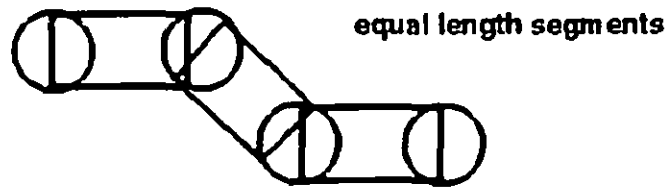


Radiance cone and sphere generators

A wine glass or a barrel could be described by linking a series of different size cones into a single shape This could be performed longhand but *Radiance* provides a function named genrev to speed up the process Additionally a curved tube could be constructed by alternating spheres and cones seamlessly into a curved shape by using genworm The final shapes are determined by the expressions or data files which you include with the command

GENWORM

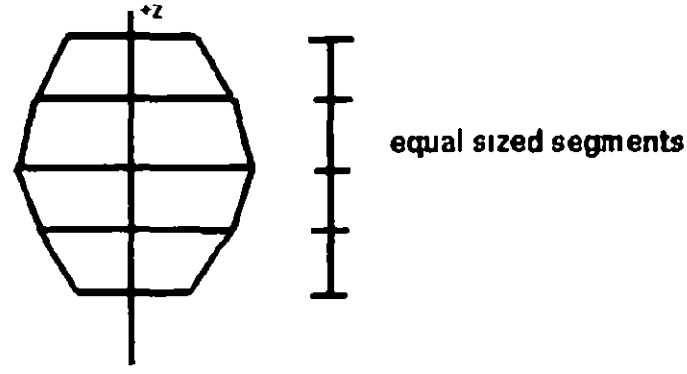
genworm mat name 'x(t)' 'y(t)' 'z(t)' 'r(t)' nseg [-e expr][-f file]



Creating shapes with cones

GENREV

genrev mat name 'z(t)' 'r(t)' nseg [-e expr] [-f file] [-s]

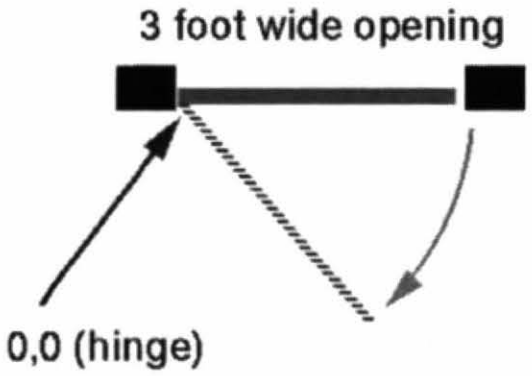


A simplified description of the anatomy generated by genworm or genrev

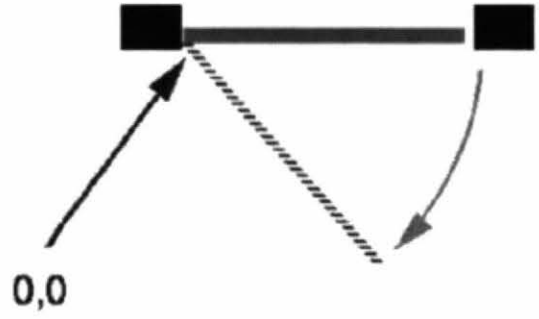
These generators are extraordinarily adaptable and are limited only by your math prowess

Assembling related objects

There are many approaches to assembling a scene in *Radiance* The description which follows demonstrates the construction of a door frame and a door that can open or close The first method is self contained in one file and inserts a simplistic genbox door The second figure inserts the file of a more complex door into the door frame file In both cases the door can be opened and closed at its hinge by simply changing an rz value in the xform command The second approach enables doors and frames from a library to be rapidly combined into many permutations




```
#simple door frame
!genbox wood left_jamb .33 .3 7 | xform -t -.33 -.05 0
!genbox wood rght_jamb .33 .3 7 | xform -t 3 -.05 0
!genbox wood top_jamb 3.66 .3 .33 | xform -t 0 -.05 6.67
# simple door
! genbox wood door 2.94 .1 6.6 | xform -t .03 0 .03
# door open 45 degrees
!genbox wood door 2.94 .1 6.6 | xform -rz -45 -t .03 0 .03
```



Create a more complex door by:

1. build door, door knob, and details in one file such as **fancy_door.rad**
2. build door frame in a second file called **doorway.rad**
3. insert fancy_door.rad into doorway.rad using xform:

```
!xform -n nicedoor -rz -45 -t .03 0 .03 fancy_door.rad
```

If many of the same door are to be included in a scene then an instance of the door can be created To be eligible the surfaces of each copy must not change though each copy could be modified by a different material type Only one description of the object really exists and pointers to that description are located in the scene It is a very efficient way to include hundreds if not thousands of identical objects in a scene without proportionately increasing the size of the data set The method begins by converting a description into an octree

```
% oconv  some materials  doorway rad  >  doorway oct
```

The octree is then inserted into an instance scene description We will call the file door ins

```
# filename  door ins
void      instance   door
1  doorway oct
0
0
```

The file can then be inserted many times into a scene file using xform commands

```
#filename  scene rad
# insert 90 doors in a circle  facing inwards
!xform  rz  90  t 50 0 0  a 90  rz 4  i 1  door ins
```

Assembling the scene

Complex scenes might include hundreds of different objects each comprised of many components There are several organizational strategies which can be employed and they generally include

> separate material files enabling expedient global updates
> several library directories perhaps one containing lighting and furniture in another
> a single file containing instances of objects which often move such as actors and chairs

But the real key to managing a large scene and rendering it appropriately is to use a *Radiance* Input File (nf) and the rad program

Managing rendering with rad

Rad performs many tasks including optimizing the rendering variables based on a few parameters which you include in the nf file By comparing the time and date signatures of files rad also manages the updating of the octree and rendering process You can use it to view a scene interactively with rview or you can render a long list of views by running in batch mode This is an immensely useful tool and every Radiance user should learn of its power A typical nf file follows

```
### Downtown_Hong_Kong nf
INDIRECT=        1
AMBFILE=       scene amb
DETAIL=          Low
VARIABILITY=  Medium
QUALITY=       High
ZONE=          Exterior  500  1500   -300  900     1  200
PICTURE=       pics/daytime
RESOLUTION=  2000 1500
```

```
UP =              Z
REPORT=           2
EXPOSURE=        -3
MATERIALS=    lib/times_square.mat   lib/roadways.mat     lib/harbor.mat    auto_lib/car.mat
MATERIALS=    lib/day_sky.mat        light_lib/lighting.mat  morgue/skin_clothes.mat
RENDER=          -st .01
SCENE=           lib/buildings.all   scene.all   roadways.all  harbor.all  morgue/people.all
Objects=         lib/tower1.rad   lib/tower2.rad  ....
view=    pk1     -vf  from_peak.vf
view=    hbr     -vf  from_harbor
```

Now to view the scene interactively for the purpose of establishing a new view, we simply change the value of INDIRECT to 0 to turn off the interreflection calculation which speeds up our rendering process. Rview is then launched with the following command:

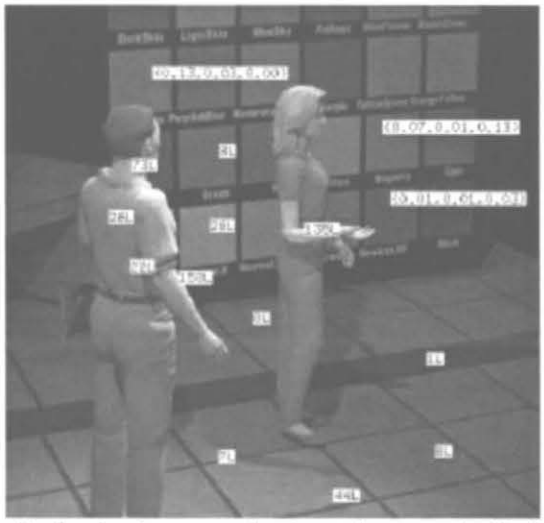
```
% rad -o x11 Downtown_Hong_Kong.rif
```

Alternatively, to render the two pictures named pics/daytime_pk1.pic and pics/daytime_hbr.pic, simply enter the following command and return the next morning to see in they are finished.

```
% rad Downtown_Hong_Kong.rif
```

Using ximage

Once a series of *Radiance* pictures are rendered, you have much more that a standard picture on your hands. The *Radiance* picture format is a 2D collection of real color radiance values. To the lighting designer, this means that the luminance values of surfaces can be directly accessed from the picture. Picking a pixel with the cursor and pressing the "l" on the keyboard, temporarily displays the luminance value of that pixel. Pressing the "c" provides the color value. If a larger areas is selected, the average luminance or color is displayed. Pressing the "=" key adjusts the exposure of the picture to the area of interest.



Ximage can display luminance and color values on the Radiance picture.

Light Revisited

What separates *Radiance* from the majority of rendering software systems is how it handles light No meaningful survey of *Radiance* can make light over this fact You have no doubt heard of the validation experiments which indicate that if you provide *Radiance* with valid physical data, then the data in the resulting simulation correlates very closely to its physical counterpart Well without a lot of bells and whistles here is a high speed exercise which directly introduces you to physically based data and results This exercise also provides a brief summary of how to get started with *Radiance*

We will measure a real lamp with a light meter locate a virtual lamp in a room sized box and compare the results To simplify the modeling we will test a 100 watt globe shaped lamp which has a fairly uniform distribution The G40 lamp has a radius of 2 5 (21) and an initial lumen output of about 1100 lumens at 120 volts We can use a sphere to simulate the lamp s geometry but how do we determine the r g b values for the light material?

Radiance comes to the rescue with its lampcolor program After answering a few questions the resulting r g b values can be pasted into our scene description

```
% lampcolor
Program to compute lamp radiance  Enter  ?  for help
Enter lamp type [WHITE]          WHITE
Enter length unit [meter]        feet
Enter lamp geometry [polygon]    sphere
Sphere radius [1]                21
Enter total lamp lumens [0]      1100
Lamp color (RGB) =               37 99   37 99   37 99
```

First we will create a test box and locate our lamp in its center With such a small scene we will combine all of our descriptions in one file and dispense with separate material and surface files Create the following file

```
## testbox rad
# materials( for the wall  lamp and a bulls-eye target)
void     plastic   wall_white
0
0
5        9  9  9  0  0

void     plastic   red
0
0
5        9   1   05 0 0

void     light      G100_40
0
0
3    37 99   37 99   37 99



# build a 10  square room and locate the lamp in the center  with its surface 5  above the floor

!genbox  wall_white  room  10  10  10 | xform  t  5  5 0
```

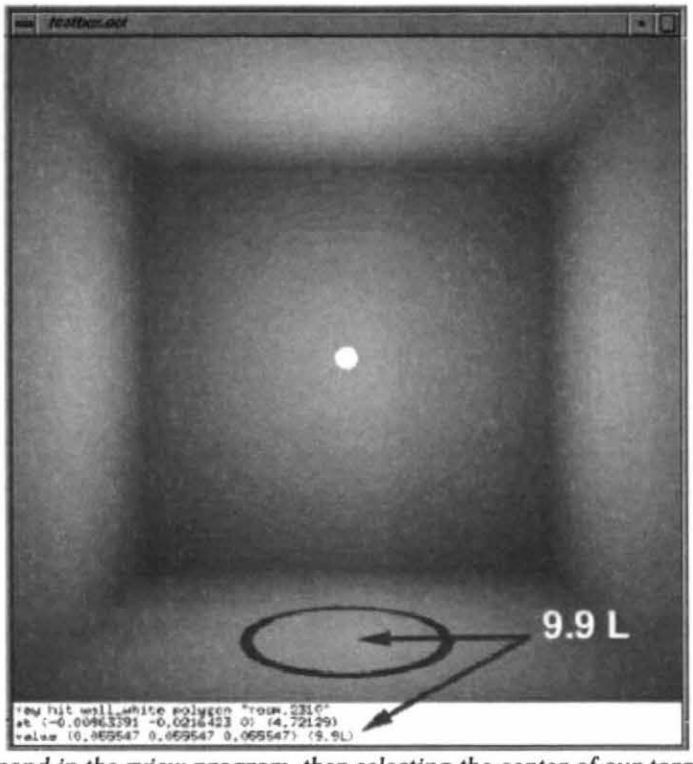
```
l00G_40sphere              lamp
0
0
4          0  0  5.21   .21

# add a red target on the floor
red                        ring                target
0
0
8          0 0 .001    0 0 1    2     1.75
# end file
```

Now create the octree.

% oconv testbox.rad > testbox.oct

We need to peer through the wall to see the effect of the lamp in the room so we will apply a clipping plane. If our vantage point (-vp) is located 15 feet from the center of the room, then a clipping plane 10.5 feet in front of the vantage point (-vo) will let us see through the wall. Enter the following rview command to view the scene:

% rview -vp 0 -15 0 -vo 10.5 testbox.oct



Using the trace command in the rview program, then selecting the center of our target area, provides several lines of data including the luminance at that point (9.9 cd/m^2)

This equation converts luminance into an incident light value or illumination:

Luminance_value * 1/reflection_of_surface * PI = illumination

(9.9) * (1 / .9) *(3.14159) = **34.56 lux** or **3.45 Fc**

No interreflection variable was included in the rview command so we can compare this virtual direct illumination with the illumination of a real lamp whose center is 5.21 feet from a light meter in a low reflection room. If the lamp is new and operating close to its rated voltage, then we can expect to measure a similar illumination value. This extraordinarily simple exercise is impossible to execute with most rendering software.

With the lamp located in the center of the testbox, the lighting of the wall is very uniform, and we would expect most sample points to be in the range of 30 to 35 lux. But suppose the lamp were closer to a corner. In this case the range of illumination would vary considerably, and if we wanted to quantify this, many samples would have to be taken.

Again, *Radiance* comes to the rescue with the falsecolor program. First we create a picture of the direct lighting component using the rpict program. Ambient bounce (-ab) is set to 0, and because we want a medium quality picture, we render it at twice final size then pipe it through the pfilt program to reduce its size by ½. This latter procedure provides anti-aliasing.

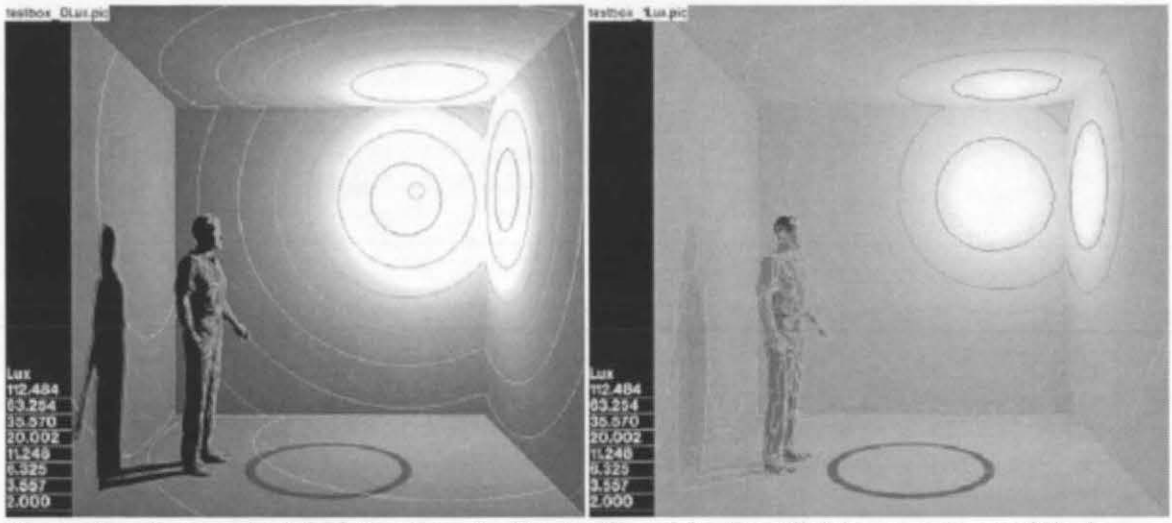
%rpict -vf testbox.vf -x 1600 -y 1600 -ab 0 -t 30 testbox.oct | pfilt -1 -x /2 -y /2 testbox_0.pic

To generate and overlay illumination contour lines on testbox_0.pic, we also need to render an second picture. Rpict is instructed to create an irradiance image (the -i option) which depicts illumination levels (incident light, not luminance).

%rpict -i -vf testbox.vf -x 800 -y 800 -t 30 -ab 0 testbox.oct > testbox_irr0.pic

Now falsecolor has all the data it needs to calculate contour lines from the irradiance picture (-i testbox_irr0.pic) and overlay them onto the of the previously rendered testbox_0.pic (-p testbox_0.pic). The following falsecolor command uses log2 to scale 10 contour lines (-cl) between 0 and 150 lux (-s 150). We will call the resulting picture testbox_0Lux.pic

% falsecolor -i testbox_irr0.pic -p testbox_0.pic -cl -s 150 -log 2 -l Lux > testbox_0Lux.pic



The contour lines on the left picture show the illumination of the direct lighting component while the picture on the right shows the increased illumination after a full interreflection. Though it would be more expedient to use a rif file to determine the rendering variables appropriate to interreflection, a simplified version of the rpict and the complete proceedure follows:

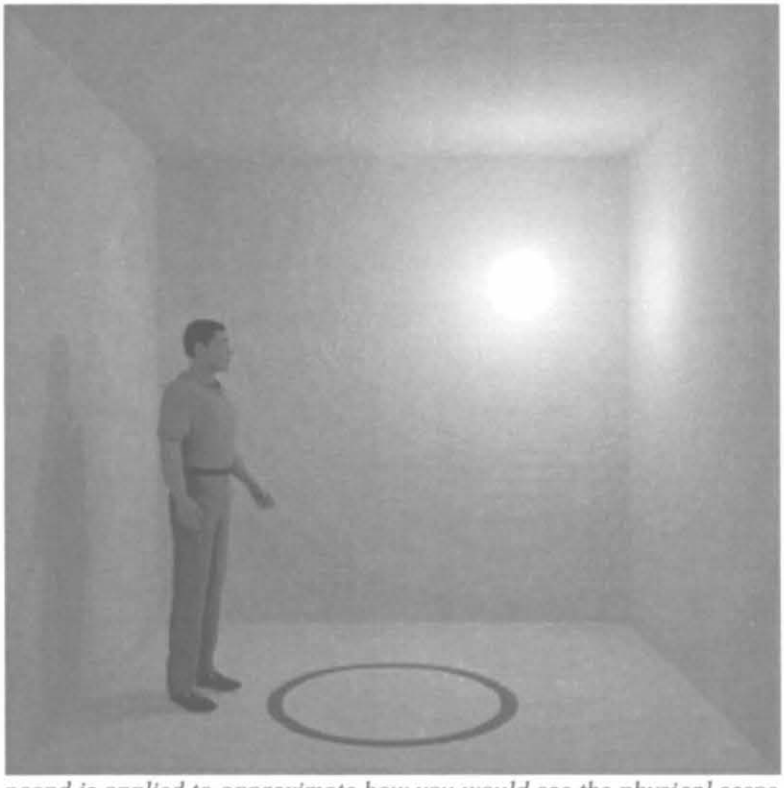
*%rpict -vf testbox.vf -x 1600 -y 1600 -**ab 1** -t 30 testbox.oct | pfilt -1 -x /2 -y /2 testbox_1.pic*

*%rpict -i -vf testbox.vf -x 800 -y 800 -t 30 -**ab 1** testbox.oct > testbox_irr1.pic*

% falsecolor -i testbox_irr1.pic -p testbox_1.pic -cl -s 150 -log 2 -l Lux > testbox_1Lux.pic

When a range of luminance values, such as in our testbox_1.pic, exceed the luminance gamut of a monitor, very bright areas become uniformly white and obscure portions of the picture. *Radiance* includes a suite of post process filters which can modify a picture to fall within the luminous range of your monitor. They can also be applied to approximate how you would see the physical scene within this luminous range. The final image in this section is produced by applying the pcond program to testbox_1.pic. The -h+ option calls a combination of filters based on human vision factors.

% pcond -h+ testbox_1.pic > testbox_1pc.pic



pcond is applied to approximate how you would see the physical scene.

One more material needs to be mentioned. If the lamp in our test box were made from the material illum, then the room would still be accurately illuminated but we would not see the light source. Illum is the invisible version of the material named light and solves many problems such as delivering daylight through windows.

Summary

Though we have had a tertiary glance at several important components of *Radiance*, we have only just scratched the surface of this profoundly resourceful rendering system. The *Radiance* manual contains full descriptions of the functions and programs demonstrated in the section.

C Daylighting Applications

John Mardaljevic

Summary

A description of just some of the many ways in which Radiance can be employed to solve daylighting problems

1 Introduction

This section describes a few basic daylighting analysis techniques. Daylight simulation has already been covered at some length in the book *Rendering with Radiance*, and duplication of some of the material from the book chapter is unavoidable since one of the techniques (daylight factors) is fundamental to daylight analysis. Otherwise, this section contains much new material and it includes a description of a new (freely available) *Radiance*-based lighting analysis tool.

2 Tutorial I: Daylight factor basics

2.1 Preamble

The daylight factor at any point is the ratio of the interior illuminance at that point to the global horizontal illuminance under CIE standard overcast sky conditions. The daylight factor (DF) is normally expressed as a percentage:

$$DF = \frac{E_{in}}{E_{out}} \cdot 100 \qquad \textbf{(C-1)}$$

The interior illuminance is usually evaluated at workplane height. Direct sunlight is, of course, excluded from the calculation. Overcast skies will generally be the dullest, so the daylight factor method should be considered a "worst case" evaluation, primarily suited to calculating minimum values. The sky luminance in the CIE overcast model does not vary with azimuth, so the orientation of the scene about the *z*-axis has no effect on daylight factors.

The conventional method to evaluate daylight factors, still very much in use, is from illuminance measurements taken inside scale models under artificial sky conditions. Unlike thermal,

acoustic, or structural models, physical models for lighting do not require any scaling corrections While a detailed physical model may indeed provide reliable results, such models can be very expensive to construct, especially if several design variants are to be evaluated Increasingly, architects and design consultants are looking to computer simulation to offer an alternative solution approach

Daylight factors are usually evaluated for uncluttered spaces Since we are not interested in visual impression, the scene description usually accounts for only the important structural features of the space, and furniture and so on is not included

Illuminance (and DF) are quantities that we derive from the irradiance predicted by the rtrace program Often you will see that the irradiance values from the standard output of rtrace are converted directly to illuminance (or DF) Wherever in the text we refer to illuminance (or DF) prediction, we shall use the term to mean irradiance prediction followed by conversion to the appropriate units The following tutorial describes, in general terms, how the mode of analysis influences the setting of key *Radiance* parameters

2 2 Procedure

Create a *Radiance* scene description for your model Here are a few guidelines

Include the following

- All walls, floors, ceilings and significant internal/external obstructions
- Window(s) and window frame bars either explicitly or as a reduced window area (Figure C-1a)
- The wall thickness where windows are present (Figure C-1b)
- An external ground plane, usually a ring of diameter ~2 times the maximum dimension across the scene contents (Figure C-1c)

Exclude the following

- Scene detail that is unimportant for significant light transfers, e g furniture, clutter, decoration, etc
- Light sources (other than the sky), e g luminaires, spotlights, desk lamps, etc

Do

- Use a convenient coordinate system - CIE overcast skies are invariant to rotation about the z-axis (Figure C-1d)
- Use gray reflectance and transmission values unless reliable spectral information for the scene materials is available
- Use - for most cases - the ambient calculation only In which case, do not convert the windows to illum sources

And finally

- Be prepared to experiment a little at first - it will save you time later on

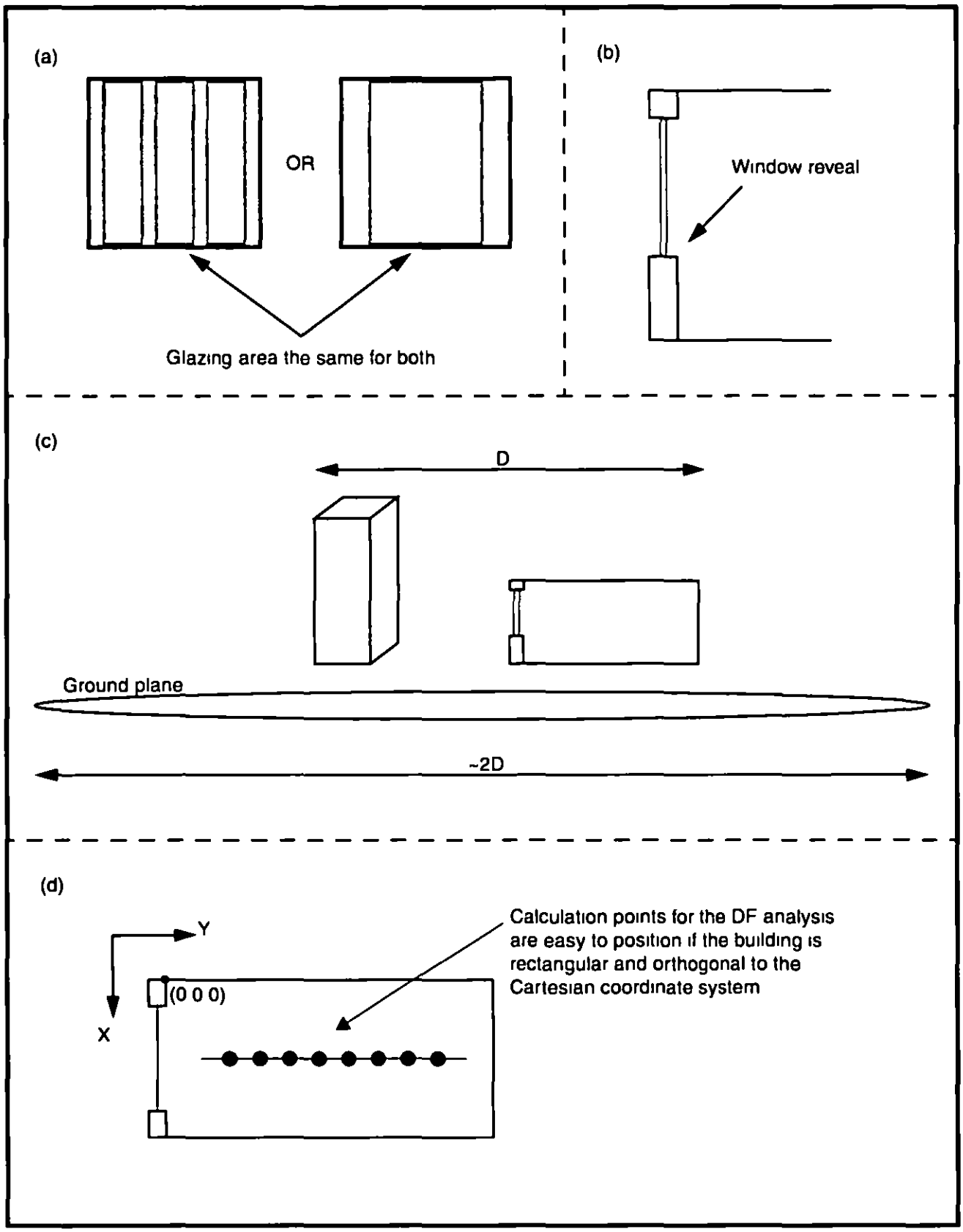


Figure C-1 Illustrations

2 3 Rendering the scene

Even if you intend to make renderings of the model later, it is still worthwhile to predict the daylight factors using as simple a scene description as possible To facilitate this, arrange the scene description files so that it is easy to assemble a basic model for the daylight factor analysis and a fully worked up model for the later renderings The files could be organised as shown in Table C-1

File contents	Mode	
	Daylight factor analysis	**Renderings**
Sky	CIE overcast	Any though skies with sun generally make for better looking more interesting renderings
Materials	Usually gray and based on measurements	RGB values either based on measured spectral properties or guesstimate values that look acceptable
Building	Main building structure	
Windows[a]	Ordinary `glass` material	Usually an `illum` material
Furniture	None	As required
Ground	Ground plane	As required
External obstructions	Model as simple shapes with average reflectance values	As required
Trees/foliage	None unless they offer significant obstruction to daylight entering the space In which case model as simple shapes e g cones	As required but usually exclude from ambient calculation

Table C-1 File organisation for daylight factor analysis and renderings

a Both modes will share the same window geometry but the use of separate files is desirable for those parts of the scene that will be manipulated independently

The following section shows how to calculate daylight factors with *Radiance* It demonstrates how the results can be very sensitive to the ambient parameter settings

3 Tutorial II Daylight factor analysis

The daylight factor is a ratio of the internal to the external illuminance, the absolute brightness of the standard CIE overcast sky that is used to derive the daylight factors is therefore not important It is good practice however to use, wherever possible, realistic values for all materials, luminous sources etc Daylighting practitioners commonly describe a sky in terms of the diffuse horizontal illuminance that is produced by that sky The CIE overcast model does not include the sun, so here the global horizontal illuminance will be the same as the diffuse horizontal illuminance The CIE overcast sky can therefore be fully characterized by the horizontal illuminance, usually given in lux A realistic horizontal illuminance for a (brightish) overcast sky is 10,000 lux This is a convenient figure to work with, for example, a daylight factor of 5% corresponds to an illuminance of 500 lux The gensky program gives us two ways in which we can generate a 10,000-lux CIE overcast sky 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 following example The irradiance that corresponds to an illuminance of 10,000 lux is 10,000/179 = 55 866 w/m^2 Note that the conversion factor is the *Radiance* system's own value for luminous efficacy[1] and is fixed at K_R = 179 lumens/watt (lm/w) The scene file for the sky and ground glow should look like this

```
# CIE overcast sky with diffuse horizontal illuminance = 10 000 lux
!gensky  ang 45 0  c  B 55 866
skyfunc glow sky_glow
0
0
4 1 1 1 0
sky_glow source sky
0
0
4 0 0 1 180
skyfunc glow ground_glow
0
0
4 1 1 1 0
ground_glow source ground
0
0
4 0 0  1 180
```

3 1 Predicting Internal Illuminances

In this example, we demonstrate how to predict DF levels for a simple scene We show how to automate the execution of the rtrace program and how this can be used to test for convergence in the ambient calculation The section concludes with an introduction to the dayfact script

A Simple Space

The room we will use is 3 meters wide, 9 meters deep, and 2 7 meters high These dimensions are typical of a deep-plan office module The long dimension is aligned north south, the room has a single south-facing window of width 2 6 meters and height 1 5 meters The south wall is 0 2 meter thick and the window is set in the middle of this wall, so there are internal and external windowsills of depth 0 1 meter The room description is maintained in three scene files

- *room rad*—walls, floor, ceiling geometry
- *mat_gray rad*—material description for walls, floor, ceiling geometry
- *window rad*—window geometry and material description

3 2 Computing Daylight Factor Values

A typical analysis might begin by determining the daylight factor along the midpoint of the room The file *samp1d inp* contains the coordinates of the positions at which the DFs will be

1 This quantity 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

evaluated Executing the rtrace command from a shell script is a convenient way to automate systematic explorations of parameter settings The following script shows how to automate the DF calculation and test the sensitivity of the prediction to the number of ambient bounces For this test, we cover the range -ab 1 to -ab 5

```
#!/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 \
        < sampl.inp | rcalc -e\
        '$1=($1*0.265+$2*0.670+$3*0.065)*179/10000*100
end
```

For all other parameter settings, the current rtrace defaults will, of course, be applied The predictions follow a characteristic pattern as shown in Figure C-2(a) close to the window, the predictions for the range of -ab are relatively similar (17% to 20% at 0.5 meter) Farther away from the window, where inter-reflection becomes more important, they agree less (0.24% to 1.26% at 5 meters We expect the predictions for -ab 5 to be greater than those for -ab 1, but sampling variance may mask that We also expect the illuminance, and therefore the DF, to gradually decrease away from the window The DF curves in Figure C-2(a) nevertheless confound our expectations the predictions are simply not good enough to show a consistent pattern in the data This is especially noticeable at the rear of the room, where the curves are very jagged

You may be relieved to learn that we don't *always* have to work through a series of -ab simulations before we can discover that one or more of the other ambient parameter settings was too coarse We can, for many situations, use the -ab 1 as a diagnostic to help us make better choices for some of the other settings Recall that for -ab 1, the illuminance predicted will be that due to the portion of sky that is directly visible from the point of calculation, that is, the direct sky component This component is usually the major contributor to the total illuminance at that point If we get the direct sky component (-ab 1) wrong, our predictions for the total illuminance (ab > 1) will be also poor For this space, we know that some sky should be visible from all the points for which we want to predict the DF Examination of the data for -ab 1 reveals that for several points at the back of the room, the DF was predicted to be zero This tells us that too few rays were spawned to guarantee adequate sampling of the window from all points in the DF plane To remedy this, we should set -ad to a higher value, say 1024 We can further improve our estimates at -ab 1 by enabling the ambient supersampling option (-as) in the rtrace calculation The value we set for -as is the number of extra rays that will be used to sample areas in the divided hemisphere that appear to have high variance In other words, for this scene, additional rays will be used to sample around the window—assuming, of course, that the ambient division sampling picked up the window in the first instance

We now repeat the DF predictions with -ad 1024 and -as 64 The ambient accuracy is the same as before, but the ambient resolution has been relaxed to -ar 16 These DF predictions look much better as shown in Figure C-2(b) The curves are fairly smooth and the rank order is the same at all points along the DF plane Which of these predictions, if any, are correct? Before we can answer this, we need to distinguish between absolute accuracy and useful accuracy For daylighting purposes, it is important to obtain reliable predictions of the DF distribution in

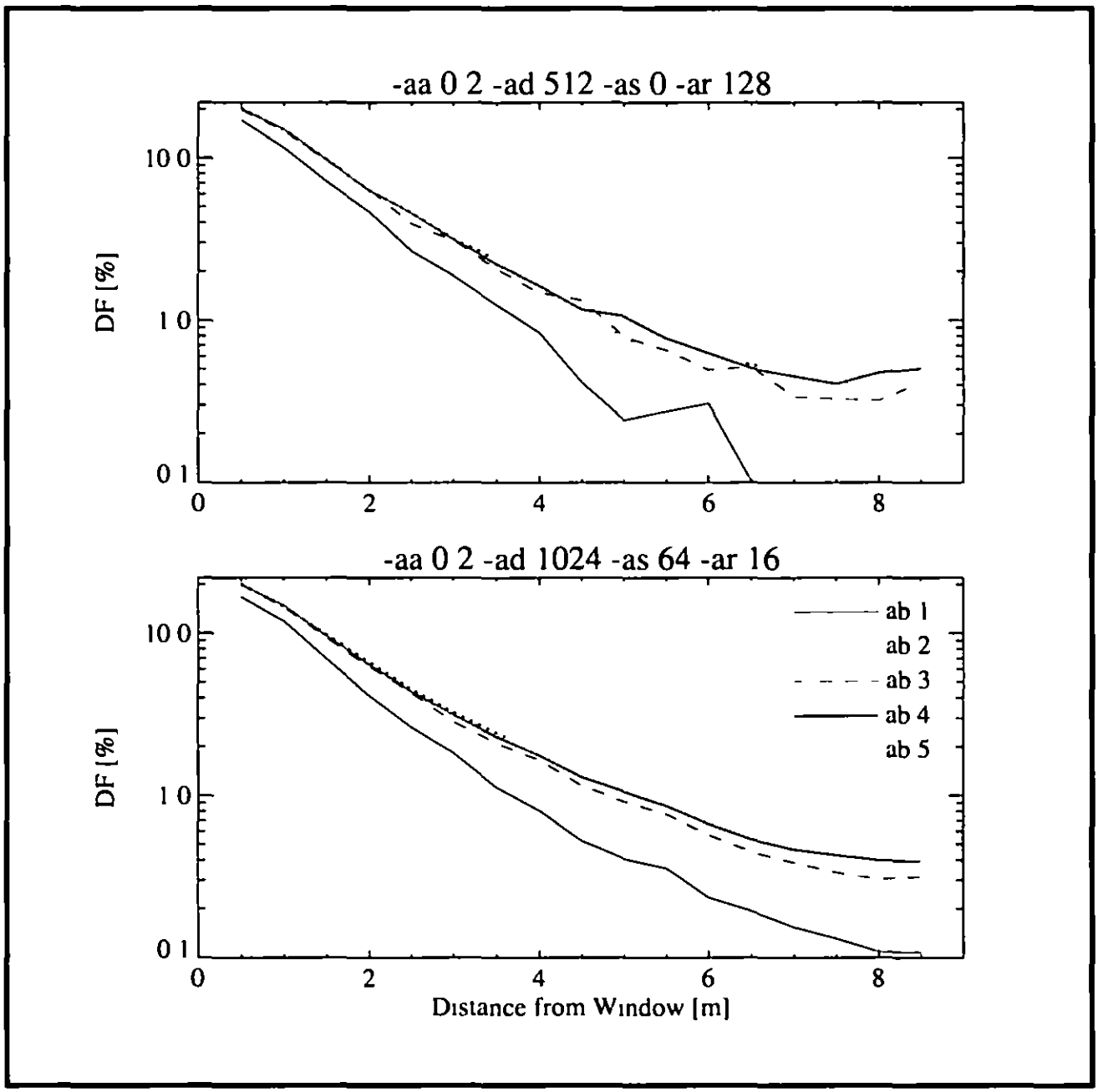


Figure C-2 Daylight factor plots showing the effects of the ab parameter The top graph (a) uses fewer samples over the hemisphere, -ad 512 as 0, than the bottom graph (b) which uses ad 1024 -as 64

the critical range 10% to 0 5% The recommended minimum DF for full daylighting is 5%, and the 1% value is generally considered to be a minimum below which the provision of daylight can be considered negligible Thus, we need to be fairly certain of the DF down to the 1% level There is little practical use in resolving the 0 1% DF boundary, or in distinguishing between the 0 02% and 0 05% levels With this in mind, there is little to choose between the -ab 4 and -ab 5 curves Would it be worthwhile predicting the DFs for -ab greater than 5? For this case, no We can see from the curves that the difference between successive DF predictions for higher -ab gets smaller each time Remember, the predictions will never be exact, so the DF curves for

scenes like this will never be perfectly smooth The basic tenets for setting the ambient parameters are

1 Set -ad high enough to capture the visible luminous features at the first bounce
2 Give sufficient ambient bounces to redistribute the light
3 Set the remaining ambient parameters to sufficiently high resolution to deliver *acceptably* smooth results

The next section shows how DFs can be used to estimate daylighting provision over long time periods

4 Estimates of long term daylight availability

In this section we demonstrate a simple technique which can give an estimate of the long-term daylighting provision of a space based on predicted daylight factor values The technique has general application and it does not matter how the daylight factors were derived - by lighting simulation, analytical means or scale model It is not therefore particular to the *Radiance* system, but it is of value and also very easy to apply

The technique is based on the cumulative availability of diffuse daylight during working hours over a period of one year This is usually derived from a standard meteorological dataset appropriate to the locale of the proposed building The dataset will usually contain hourly measurements of diffuse horizontal irradiance and direct normal irradiance for 365 days For this technique, we are only interested in the diffuse irradiance measurements The cumulative availability is derived from the diffuse irradiance time-series as follows

1 Convert the diffuse horizontal irradiance to diffuse horizontal illuminance using a simple luminous efficacy model The simplest efficacy model of all is a constant of conversion - usually something in the region 80 - 120 lm/W (Don't confuse this with the *Radiance* system's own value of 179 lm/W)
2 Subset the illuminance time-series taking only those values that fall within the normal working day, say 09h00 to 18h00
3 From the time-series subset, compute the cumulative diffuse daylight availability for the working year

A plot of the cumulative diffuse daylight availability derived from measurements taken at Kew (UK) is shown in Figure C-3 With this information and a daylight factor value we can estimate the percentage of the working year for which a target illuminance is achieved

For example, say we have predicted a daylight factor of 2% for, say, the middle of an office space, and that we are interested in the cumulative availability of (internal) daylight illuminances of 100, 200 and 500 lux Applying the daylight factor calculation in reverse, so to speak, we can deduce that, at 2% DF, internal illuminances of 100, 200 and 500 lux are provided by (CIE overcast) skies with diffuse horizontal illuminances of 5,000, 10,000 and 25,000 lux, respectively Reading from Figure C-3, we see that these diffuse sky illuminances are achieved for about 85, 70 and 30% of the working year

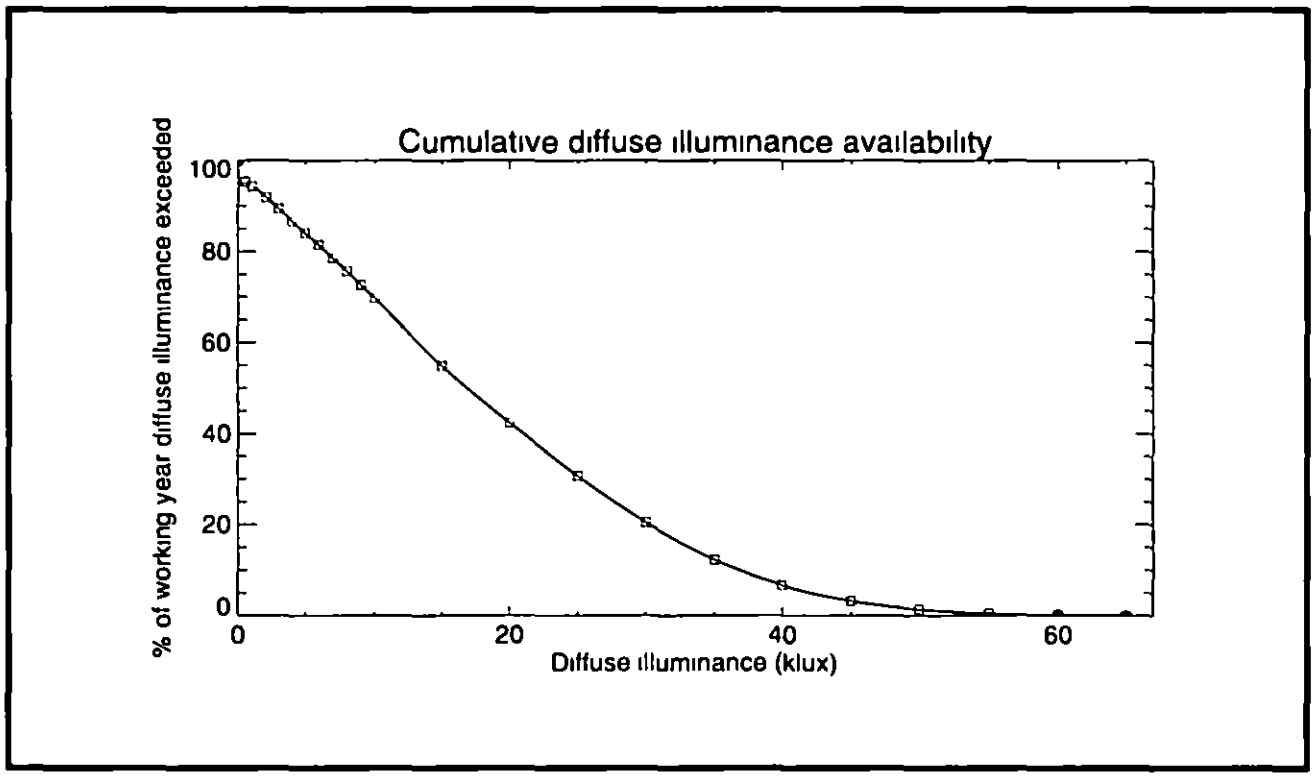


Figure C-3 Cumulative diffuse daylight availability for Kew (UK)

How reliable are these estimates? In applying the technique, we make the implicit assumption that all the skies - bright and dull - have a luminance distribution that conforms to the CIE overcast sky model In reality of course, the full range of sky types will occur according to the location of the site, e g Northern European, Iberian, Mid-West USA, etc Note also that the dullest skies in the distribution (0 - 10,000 lux) will comprise overcast skies *and* clear or intermediate skies with low altitude sun Nevertheless, the technique gives us a reasonable first order approximation for internal daylight availability The omission of the solar illuminance contribution direct and reflected - will generally result in an under-estimation of the total availability of daylight illumination It will often be the case however that the penetration of direct solar illumination into a space will precipitate the closing of blinds etc , lowering the daylight levels overall Lights may then be switched on to provide illumination and/or reduce contrast levels For buildings where the redirection of direct solar illumination is important (e g those incorporating light shelves), the technique will have less applicability

To reliably predict daylight illuminance over long time periods, we need to account for the changing sky *and* sun conditions The following section describes a new *Radiance*-based tool that is designed to predict time-varying internal illuminances

5 The Dynamic Lighting System New *Radiance*-based software

Computer programs to assist lighting designers and manufacturers have been produced, but few can model the complexities of advanced daylighting systems or predict the varying illuminance over long time periods The benefits of natural light are only realised if an appropriate artificial lighting system and controls are installed This section discribes the

Dynamic Lighting System (DLS) - a new *Radiance*-based software tool to predict time-varying internal illuminances The DLS was developed at the Institute of Energy and Sustainable Development, De Montfort University, UK The DLS software is currently (May 98) being tested and it will be made freely available in the near future The following sections are taken from the final report on the project [EPSRC 97]

5 1 Overview of the DLS System

The methods used by the DLS system to predict both natural and artificial illuminance are based on the daylight coefficient approach [Tregenza 83] In this context, a coefficient is the numerical relationship between the intensity of a source of light - a luminaire or a patch of sky - and the amount of that light arriving at a point of interest, Figure C-4a The total illumination is calculated by considering the contributions from all such sources of light, i e all the luminaires and all the sky patches (Figure C-4b)

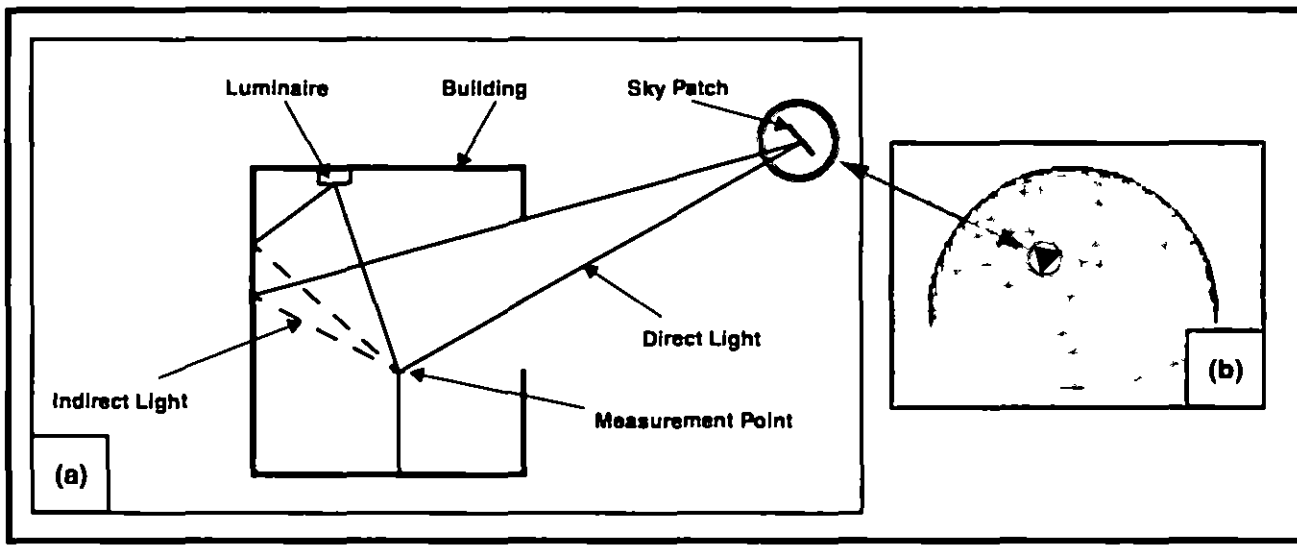


Figure C-4 Sources of illumination (a) and sky dome discretisation (b)

The system uses an advanced physically-based ray-tracing program, *Radiance*, to calculate all the coefficients *Radiance* was chosen because it is capable of calculating complex inter-reflections, and places no theoretical limitation on the complexity of the building geometry Previous research at De Montfort University had validated the numerical accuracy of *Radiance*, when used in its native mode, for calculating illuminance values under complete skies [Mardaljevic 95], and when it was used in a demonstration of concept study, to calculate daylight coefficients [Cropper 97]

The DLS system has two distinct functions (i) to generate coefficients, and (ii) to evaluate the time-varying performance of the lighting scheme under consideration The first function uses new code to generate a sky dome, and to define the position and solid angle of patches of sky Definitions of luminaires, including complex distribution patterns, provide sources of artificial illuminance *Radiance* is then used to calculate *daylight coefficients* and *artificial light coefficients* These operations are performed only once for a given building geometry and lighting layout The program's second function uses the coefficients to predict illuminance, by assigning the actual luminance values to each sky patch or light source, scaling those values using the coefficients, and summing the light arriving at each prediction point This function may

be used repeatedly, i e at each time step of the calculation, without re-calculating the coefficients

5 2 Theoretical basis

The basic daylight coefficient scheme [Tregenza 83] was re-formulated to make effective use of *Radiance*'s hybrid deterministic-stochastic ray tracing approach In the new formulation, the total daylight illuminance at a point, E, was evaluated as the sum of four components of illuminance

$$E = E^{d} + E^{i} + E^{sd} + E^{si} \qquad \text{(C-1)}$$

Where E^{d} and E^{i} are, respectively, the direct and indirect components of illumination due to skylight Similarly, E^{sd} and E^{si} are the direct and indirect components of illumination due to solar radiation The direct components account for window and room configuration, external obstructions and glazing transmittance The indirect quantities account for the inter-reflected light components, which for both cases, sun and sky, include internal and external reflections In contrast to a previous theoretical scheme [Littlefair 92], the illuminance components used here are defined by type - direct or indirect - and luminous origin - sun or sky External obstructions and reflections etc are absorbed in these four categories In the daylight coefficient matrix notation [Tregenza 83], the total illumination vector, $\mathbf{E}$, (sun and sky, direct and indirect) is given by

$$\mathbf{E} = (\mathbf{D}^{d} \times \mathbf{c}) + (\mathbf{D}^{i} \times \mathbf{c}) + \mathbf{D}^{d}_{\beta} S^{sun} L^{sun} + \mathbf{D}^{i}_{\beta} S^{sun} L^{sun} \qquad \text{(C-2)}$$

Where $\mathbf{D}^{d}$ and $\mathbf{D}^{i}$ are, respectively, the daylight coefficient matrices for the direct sky component and the indirect sky component The vector $\mathbf{c}$ is the product of the solid angle and the luminance for all the patches of sky The vector for the direct component of illuminance from the sun was calculated by multiplying column β of the direct sky component matrix (i e $\mathbf{D}^{d}_{\beta}$) by the product of the sun solid angle S^{sun} and the sun luminance L^{sun} The column index β identifies the direct sky component for that patch which was closest to the actual sun position Similarly, the indirect component of illumination from sun was calculated (last term in the above equation) using column β of the direct sky component matrix (i e $\mathbf{D}_{\beta}$)

The artificial light coefficients are specific to each (user-placed) luminaire, in contrast to the daylight coefficients which are related to the discretised sky Artificial light coefficients therefore have a less general application than daylight coefficients, and a simpler theoretical basis for each lamp, the illuminance at a point is divided by the luminous output of the luminaire to obtain the coefficient

5 3 Program modules within the DLS

The major program modules which comprise the DLS are shown in Figure C-5 The DLS possesses a comprehensive graphical user interface (GUI) At key stages of an analysis scenario, the user is presented with various graphical windows to either aid configuration of the problem, or to display certain facets, e g a wire-frame image of the building, Figure C-6a The

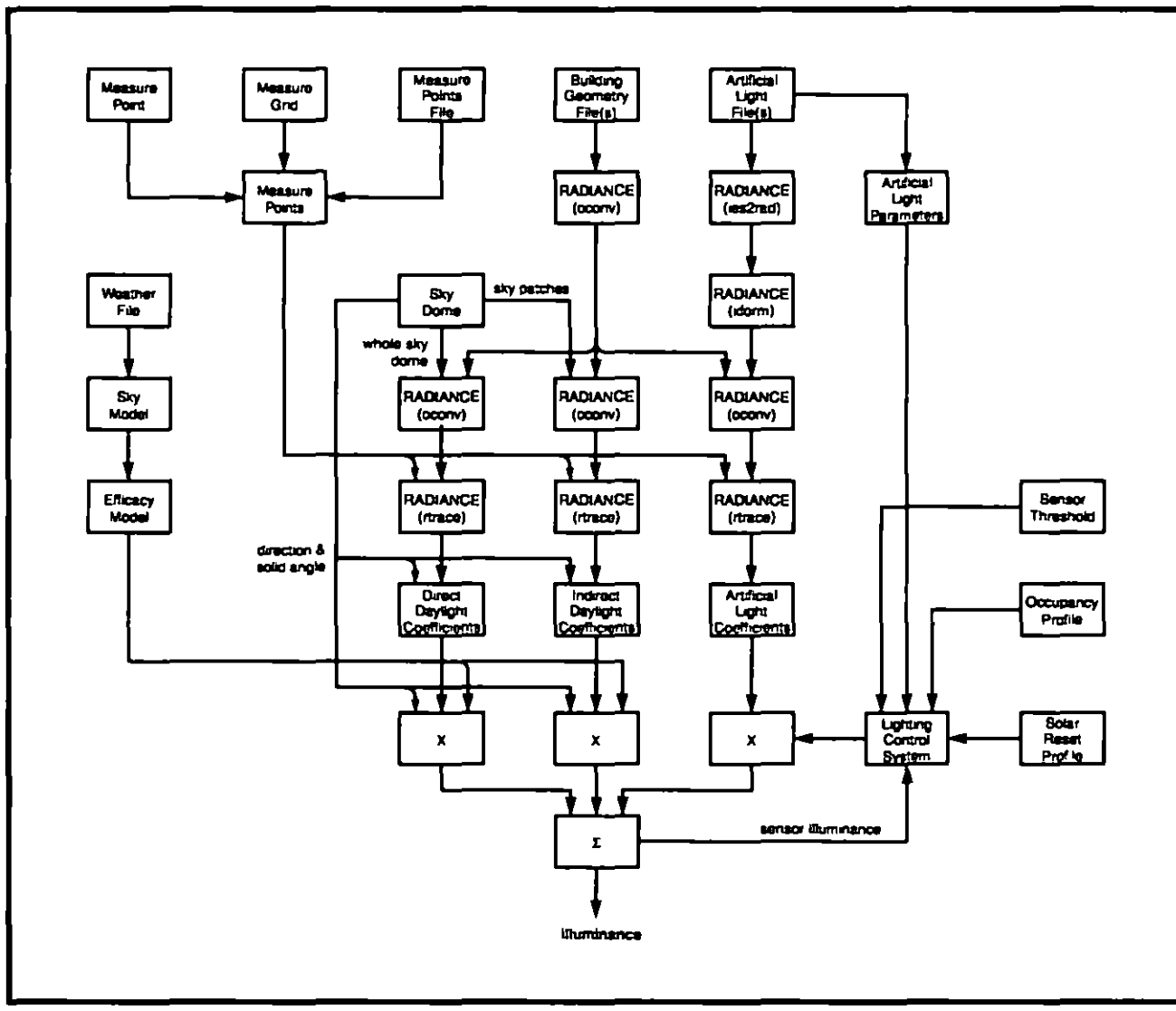


Figure C-5 System diagram for the DLS

system includes plotting routines to display a time-series of predicted illuminance values, Figure C-6b, and other results

Defining the building geometry and artificial lights

Building geometry is defined in data files, described using the standard *Radiance* input format, which may be obtained by translation from a computer aided design (CAD) drawing, e g AutoCAD These file also contain information about the colour, reflectivity, roughness and specularity of surfaces and the transmission properties of glazing materials

The DLS includes an extensive database of luminaires In addition to referencing the luminaire description file, the DLS database stores information about each luminaire, such as power consumption, luminous output, etc , used by the system when calculating artificial light coefficients and when evaluating a lighting design The luminaire database, and a program used to maintain it, are described in more detail in the full report [IESD 97] When each luminaire is added, its position, a representation of its geometry and an identification number are shown by the geometry viewer (Figure C-6a) The user is able to rotate the wire-frame

image to view the building from any direction, enlarged or reduced and moved around within the viewer window.

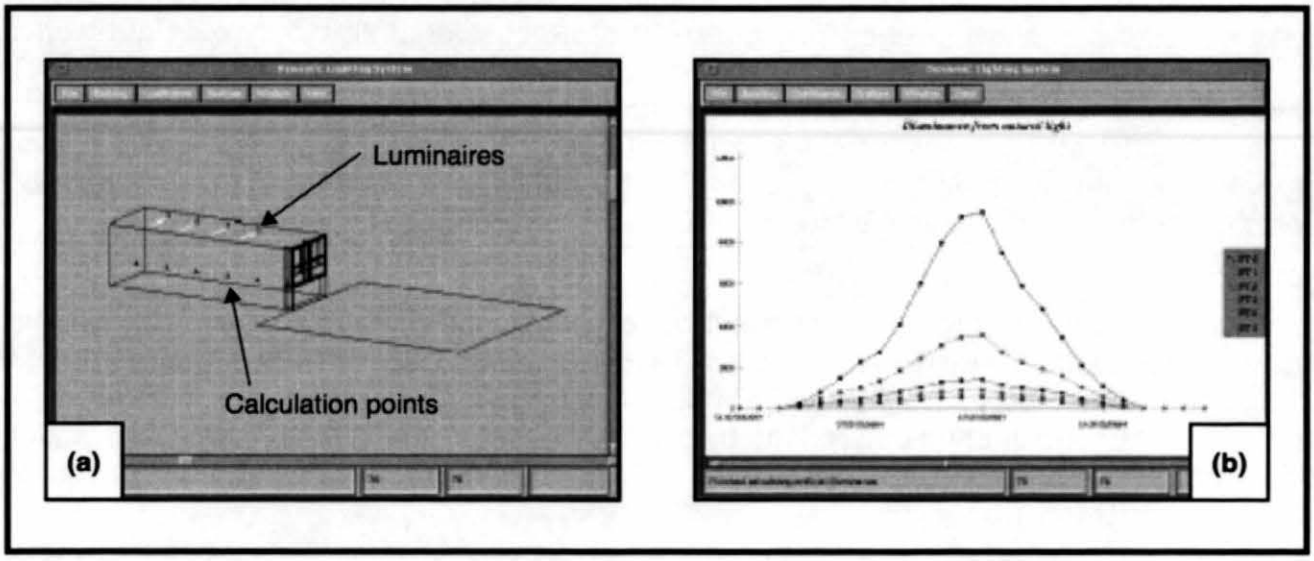


Figure C-6. Display of building model by geometry viewer (a) and example of results for daylight illuminance prediction (b)

5.4 Calculating the lighting coefficients

Daylight coefficients are calculated in two stages. Indirect illuminance is determined by using each patch of sky in turn as an individual light source. Each patch is combined with the building geometry and the amount of light from that patch arriving at the measurement point(s) is determined using the *Radiance* inter-reflection calculation. For the direct component, rays are aimed towards segments (i.e. patches) of a complete sky hemisphere and the resulting illuminance is evaluated by summing each ray's contribution.

The daylight coefficients are also used to predict illumination from the sun. The direct and indirect coefficients for the sky patches nearest to the position of the sun, calculated at each time step, are used to calculate the solar illuminance. This strategy accepts a small sun position error, in exchange for greater flexibility of the software. In addition, the resulting coefficients are invariant to the orientation or world position of the building. The finest level of sky discretisation currently used by the DLS is comparable to that which gave the lowest errors in the validation exercise.

The coefficient approach is also used to predict illumination from artificial lights. Each luminaire is combined in turn with the building geometry, and *Radiance* used to determine the resulting illuminance. The illuminance value is divided by the luminous output of the luminaire to obtain a coefficient. This normalisation of the coefficient value allows the luminous output to be varied during the prediction phase, either as a result of automatic or manual dimming, or by the DLS user specifying different luminous outputs.

This approach means that coefficients can be calculated for any number of luminaires, including different types of luminaire at the same location. The final design can then evolve rapidly by comparing predictions for alternative luminaire combinations without repeating the coefficient calculation phase.

Lighting control systems

The function of the lighting control system is to determine if the lights should be on or off The DLS currently provides two models for lighting control systems The manual switching model [Hunt 80] is based on the probability that a person entering a room will switch on the lights in response to a perceived level of illuminance, the lowest level of illuminance on the working plane For each occupancy period the probability that the lights will be switched on at the start of the period is found This probability is compared with a random number, to determine if the lights should be switched on If the lights are not switched on, the decision making process is repeated at intervals through the occupancy period The manual switching algorithm is described in more detail in the full report [IESD 97]

The other lighting control model is based on photo-cell switching which compares the illuminance value at a measurement point, designated by the user as a sensor position, with one or more specified thresholds The result of this comparison is used to determine if the lights should be switched on The lights are switched off either when the illuminance rises above the appropriate threshold, or as determined by solar reset switching In either case, the lights are switched off at the end of the working day The DSL currently includes two types of photo-cell switching [IESD 97]

5 5 Proof of concept Validation

To prove the daylight coefficient concept, the illuminance predictions obtained by the method were validated The goal of the validation was to show that the accuracy of the illuminance predictions was good in absolute terms, and that it was comparable with that already demonstrated using individually modelled skies [Mardaljevic 97] The validation was carried out using a unique dataset of measurements taken at the BRE The measurements for the 754 entries in the validation set covered a range of naturally occurring skies, from heavily overcast, through intermediate to clear sky conditions In order to match the sky scanner measuring pattern, the sky discretisation schemes used for the validation were modified from the default triangular-patch version used in the DLS [IESD 97]

5 6 Availabilty

The DLS will be made available for download and use soon - check the *Radiance* web site for links to the DLS in Summer 1998

6 Solar penetration study

Another use to which we can put *Radiance* is the evaluation of solar penetration into a space The pattern of solar penetration into a building can be assessed by generating a rendering that reveals which interior surfaces are illuminated by the sun An image sequence for, say, each hour of the day, will show where and when solar penetration occurs The most straightforward way to assess the degree of solar penetration into a space is to visualise the floor plan from above Fisheye views from just below the ceiling can achieve this, but they introduce distortions and much of the floor plan can be hidden by projecting furniture etc A better way to obtain a rendering of the floor plan is to enable the clipping plane options in rpict (and rview) These can be set to eliminate foreground and background surfaces allowing the user to 'see through walls' without affecting the light transfer in the simulation, Figure C-7

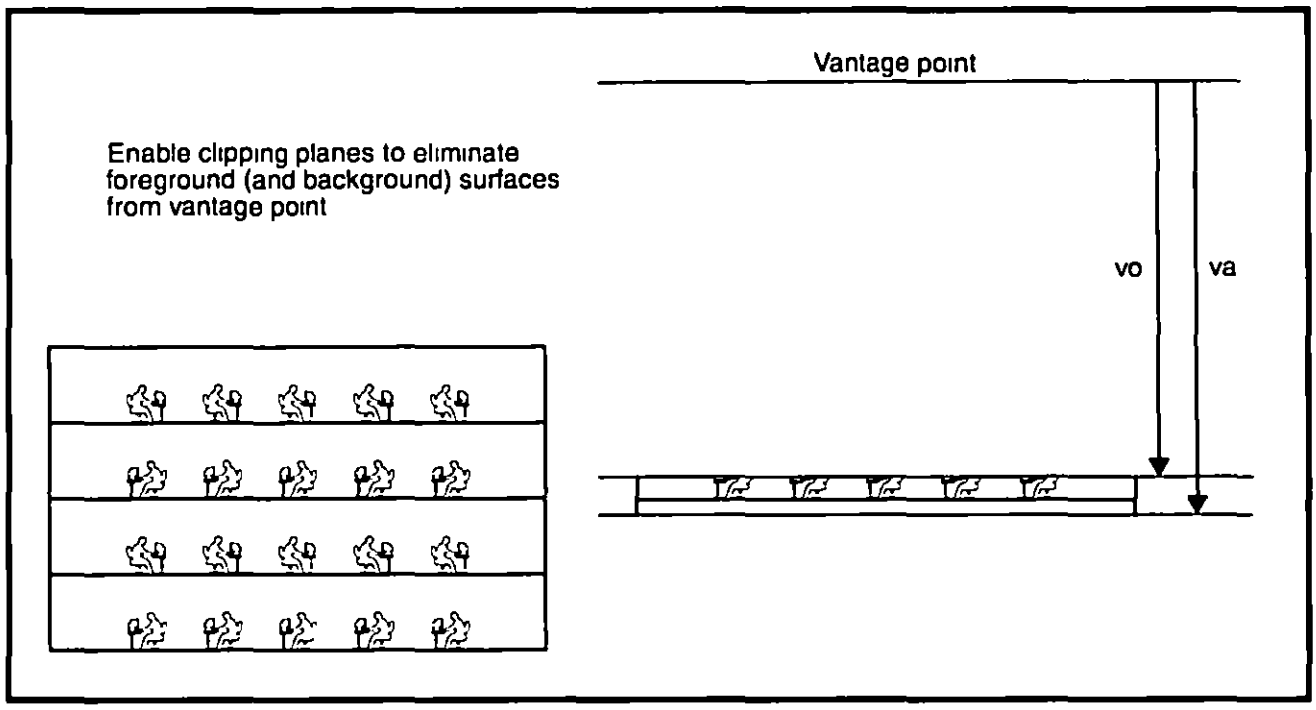


Figure C 7 Clipping planes

A typical use of this technique is to assess a louvre design and/or to compare design variants For example, the renderings given in Figure C-8 show vertical and horizontal external louvre systems The external louvre design was intended to exclude direct sun penetration from the main working floor areas of the building An analysis was carried out for level 3 of the building for several louvre designs - two of which are presented here

The pattern of solar penetration across the floor plan was assessed from renderings that revealed which surfaces were illuminated by direct sun at a particular time The sun position varies continuously throughout the year A profile for sun penetration was therefore established by considering a winter, spring/autumn and summer case For the first day of January, March and June, renderings were generated for the hours 09h00, 10h00, , 20h00 Image sequences[1] are presented for two design variants, Figure C 9 The renderings were generated using a small ambient component to show up some of the furniture (rows of desks) and partitions

The details of this particular analysis were not important, what we have demonstrated is the basic principle

1 The individual renderings were assembled into a single image using the **pcompos** program

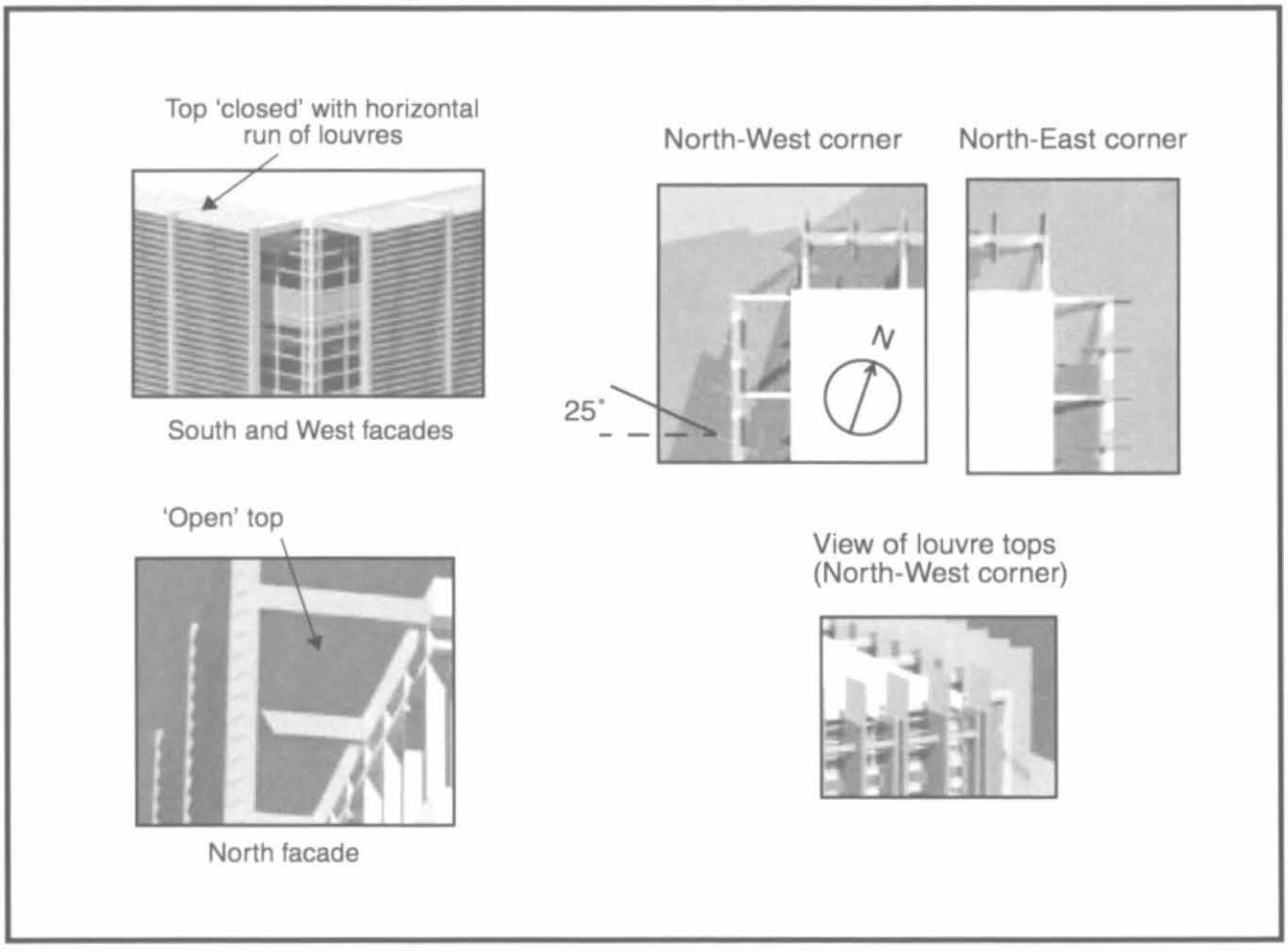


Figure C-8. Example renderings showing different louvre designs

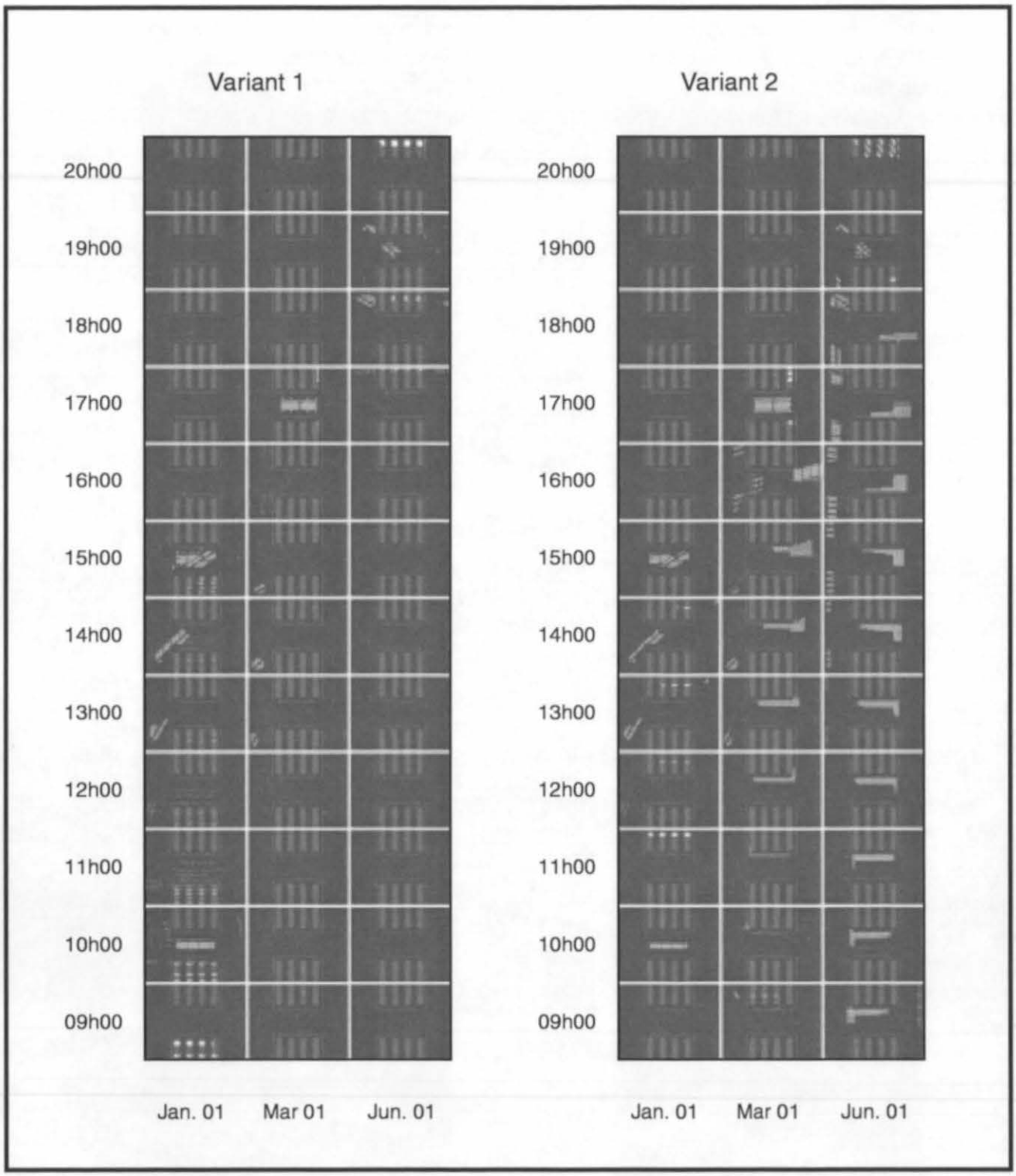


Figure C-9. Example solar penetration image sequences

7 Summary

These notes have demonstrated just *some* of the ways that *Radiance* can be used to solve daylighting problems We hope that the user may learn from these examples and apply the techniques demonstrated to their own problems More importantly, we hope that they will go on to devise new ways of solving lighting problems using the *Radiance* system - particularly the ones that we haven't yet thought of - they are always the most interesting

Acknowledgements

The solar penetration analysis was based on work carried out at the IESD on behalf of Dyer Warner Partnership, Leicester, UK

References

[Cropper 97] Cropper P, Lomas K J, Lyons A and Mardaljevic J A *Dynamic Lighting System Background and Prototype* Lux Europa 97 proc 480-492 (Amsterdam, 1997)

[EPSRC 97] Cropper P, Mardaljevic J, Lomas K J and Lyons A *The Dynamic Lighting System* Report on EPSRC Grant GR/J88753, Institute of Energy and Sustainable Development, De Montfort University, Leicester (1997)

[IESD 97] Cropper P, Mardaljevic J, Lomas K J and Lyons A *The Dynamic Lighting System* Complete internal report on EPSRC Grant GR/J88753, Institute of Energy and Sustainable Development, De Montfort University, Leicester (1997)

[Mardaljevic 95] Mardaljevic, J *Validation of a lighting simulation program under real sky conditions* Lighting Res Technol 27 (4) 181-188 (1995)

[Mardaljevic 97] Mardaljevic, J *Validation of lighting simulation program a study using measured sky brightness distributions* Lux Europa 97 proc 555-569 (Amsterdam, 1997)

[Petherbridge 83] Petherbridge P and Oughton D R *Weather and solar data*, Build Serv Eng Res and Technol , 4 (4), 147-58 (1983)

[Tregenza 83] Tregenza P and Waters I M *Daylight coefficients* Lighting Res Technol 15 (2) 65-71 (1983)

Rendering with Radiance

A Practical Tool for Global Illumination

Lighting Design Considerations

by Charles Ehrlich

Introduction

Guide to learning how to

- Create lighting accurate 3D models
- Measure / define accurate material properties
- Perform analyses with Radiance for lighting design

What is Lighting Design?

Design of the lit environment of buildings

- electric lighting
- daylighting (previous topic)
- day- and electric lighting combined

Purpose of a rendering for a lighting designer is to visualize the effect of surface, material, and lighting choices

Why is Lighting Design Important?

The Design of our lit environment affects all of us every moment of our life

Visual comfort / discomfort

Veiling reflections / disability glare

Minimum levels of illumination

Productivity and well-being (SAD)

How does Radiance help?

Does not place arbitrary limits or significant processing burden on complex scene geometry

Accepts input of measurable information about the behavior of materials and light sources

Produces rendered images which contain real-world values suitable for quantitative and qualitative analysis

No arbitrary limits on scene complexity

Efficiently renders thousands of lights sources

Efficiently renders very complex geometry

Rendering time sub-linearly proportional to number of surfaces

Accepts input of measurable information

Visible surface reflectance and transmittance

265074126*R + 670114631*G + 064811243*B

Surface specularity

Surface roughness (RMS facet slope)

Lamp color temperature using lampcolor csh

Rendered images contain real-world values

Dynamic range of image format encompasses the faintest starlight to beyond brightness of the sun
Images can be post-processed to retrieve point value information
Images can be post-processed to introduce artifacts of the limitations of the human eye

Software Tools Available

Questions answered in this section

What are the minimum general qualifications of 3D CAD modeling tools for Radiance?
What tools are integrated with Radiance?
What tools export to Radiance directly?
What tools support Radiance through plug-in?
What intermediate geometry formats are supported?
How do I begin to write my own converter?
What can I do with Radiance without a CAD tool?

Minimum general qualifications of tools

Creates/exports 3D surfaces
Allows attachment of material name or other exportable attribute to individual surfaces
Polygon vertices are ordered and co-planar
Exports or converts to a supported geometry format
May or may not support export of materials and views

Tools integrated with Radiance

SiView Siemens, AG
Genesys by Genlyte
ADELINE by IEA Task 12 (LBNL and Fraunhoffer Institute of Stuttgart)
Radiance Daylighting Tool by LBNL

Tools which export Radiance directly

BRLCAD (US Army, public domain)
SCED (UC Berkeley, public domain)
Design Workshop (Artifice, Inc)
also supports material and pattern export

Tools supported through plug-in

Arris (plug-in module arris2rad)
AutoCAD (plug-in modules torad, radout, ddrad)

Intermediate geometry formats

OBJ (obj2rad)
DXF v10 (dxf2rad) v13 (ADELINE)
3DS v2 (3ds2rad)
NFF (nff2rad)
MGF (mgf2rad)
USGS Digital Elevation Models (dem2rad tar)
IESNA Candlepower Distribution Data (ies2rad c)

Converters for other tools

General triangle mesh (tmesh2rad c)
Stratastudio (stratastudio sea)

Architron (old format arch2rad c)
GDS Things file (thf2rad c thf2rad2 c)

What is possible w/o CAD tool?

Geometry modeling within Radiance
- genbox (boxes with optional rounded corners)
- genrev (surfaces of revolution with smoothing)
- gensurf (arbitrary parametric surface w/smooth)
- genblinds, genclock, genprism, genworm
- xform is a general uniform transformation tool
- use "antimatter" for simple CSG

Write your own converter

A good intermediate format to use mgf2rad c
A good example of how to do it arch2rad c

Availability of tools

BRLCAD	http //web arl mil/software/brlcad/
AutoCAD	
torad	ftp //radsite lbl gov/translators
radout	http //www schorsh com/
ADELINE	http //radsite lbl gov/adeline
SCED	http //http cs berkeley edu/~schenney/sced/sced html
Radiance Daylighting Tool	
LBNL	http //radsite lbl gov/desktop

Modeling Approach & Methods

Questions answered in this section
- What are the minimum requirements of scene?
- What are the limits of scene complexity?
- What types of geometry, materials and views and light sources are possible?
- What types of patterns and textures are possible?

Minimum requirements of scene

One surface, N-sided closed, planar polygons
- right-hand vertex ordering for surface normals
- must be able to become part of valid octree

One material associated with surface
One view (default is at origin looking along +Y)
One light (or background ambient value)

Limits of scene complexity

Size of scene limited mostly by hardware
Complexity sometimes limited by octree
use instances and keep geometry axis-aligned
The software limits that do exist can be over-ridden in source code and re-compiled

Largest to smallest dimension
from many thousands of miles down to an inch
Bitmapped and procedural textures, patterns and mixtures

Types of geometry

Closed polygons with holes and phong smoothing
Spheres and bubbles
Rings and disks
Cylinders and tubes (flat ends only)
Cones and cups (flat ends only)
Infinitely distant "source" for skydome
Groups of surfaces within "instance"

Types of materials

lambertian (diffuse) distribution with visible reflectance
diffuse surface with specular highlights
semi-specular highlights with roughness
anisotropic roughness with non-uniform highlights
clear transmitting dielectric with color and Tvis
semi-transparent and diffusing with light redirection
mixtures
BRDF and BRTF

Types of views

perspective views with off-axis shifts (-vh, -vv)
parallel projection views (plan and axonometric)
fish-eye views (linear-180°, angular- 360°)
cylindrical views (quicktimeVR)

Types of light sources

polygon, disk, cylinder, sphere, and source
normal with $1/D^2$ fall-off (light)
limited range of effect (glow)
limited cone of effect (spotlight)
invisible surfaces for imposter geometry (illum)
statistical sampling used for optimization
can use candlepower distribution data (IESNA)

Dealing with ambient light

no need to create "fake" lights to make the scene appear realistic
method empolyed is "ambient interreflections"
no rigid requirements on geometry of scene
rendering time sub-linearly related to # of surfaces
very complex scenes rendered with minimal h/w

Special requirements of lights

pay attention to surface normal
large-area sources subdivided or penumbras

can use candlepower distribution data (IESNA)
complex luminaires require imposter geometry
  specular reflectors (emitting sufaces can't be found)
  intervening geometry (to avoid unrealistic shadows)
  use boxcorr to properly map light distribution
  if very close to other surfaces use lboxcorr

Special requirements of windows

use "mkillum" to pre-calculate light distribution
  pay attention to surface normal
  large-area sources subdivided or penumbras
  can reduce ambient bounces by one or two
Complex glazings require imposter geometry
  venetian blinds
  light shelves
  transom glazing between office and interior space
surface normal must point into scene

Special requirements of Combined Day- and Electric Lighted Scenes

If using illums in windows, must turn on source sampling
Source sampling will inordinately affect rendering time if there are many electric lights because they will also be sub-divided (source sampling is a rendering option, not a material option)

Repository of Objects

Radiance tp site
  ftp //radsite lbl gov/pub/objects
Avalon repository
  http //www cdrom com/avalon

Repository of Materials

Included with Radiance distribution
  ray/lib/materials dat
  includes measured values German RAL standard
  includes examples of common patterns
  includes several glazing types

Repository of Luminaires

More and more lighting manufacturers are providing luminaire data on www sites
  http //www ledalite com
Other companies provide software with specific information about their products
  Genlyte's Genesys program includes Radiance
  http //www lightolier com/

Gallery of Images

http //radsite lbl gov/radance/

Material Properties

Questions answered in this section
- Which measuring devices and methods exist?
- What software for colorspace conversion exists?
- How to go beyond diffuse materials?

Measuring Devices & Methods

Read *Rendering with Radiance* for how to
- Estimate reflectance with gray scale chart
- Estimate color with color picker software
- Use a luminance meter
- Use a calibrated scanner
- Use a spectrophotometer

Software for Colorspace Conversion

Scripts provided with Radiance
- ray/src/cal/cal/
- *Rendering with Radiance* book

Colortron software
- Light Source Images Technologies

Software for high end spectrophotometers

Going Beyond Diffuse Materials

How to model properties of
- Reflective materials
- Transmissive materials
- Emissive materials (light)

What it is appropriate use of
- Patterns (variations in brightness or color)
- Textures (large-scale variations in surface)
- Mixtures (combinations of other types)

Reflective Materials

Diffuse reflectance

Specularity

Isotropic roughness
- plastic, metal

Anisotropic roughness
- plastic2, metal2

Arbitrary distribution
- plasfunc, plasdata, metfunc, metdata

Transmissive Materials

Visible transmittance
- dielectric, glass

Diffuse transmittance
  trans, transdata, transfunc
Specular transmittance
  trans, trans2
Arbitrary distribution
  BRTDfunc

Emissive Materials

Visible light sources
  light, glow, spotlight
Invisible light sources
  illum
Secondary light sources
  mirror, only valid on polygon and disk (ring)
Light re-direction for glazings
  prism, prism2

Applying Patterns

A pattern is a variation in brightness or color
Procedurally defined
  brightfunc, colorfunc
Mapped from an external data file
  brightdata, colordata
Mapped from an image
  colorpict
To apply text
  brighttext, colortext

Applying Textures

A texture is a large-scale variation in surface normal simulating bumpiness
Procedurally defined
  texfunc
Mapped from an external data file
  texdata

Applying mixtures

A mixture is used to combine the effects of other patterns or materials (materials are a recent addition)
  notion of a "foreground" and "background"
Mapped with a procedure
  mixfunc
Mapped from an external data file
  mixdata
To apply ascii text
  mixtext

Analysis for Lighting Design

Questions answered in this section

- What is the goal of a lighting design/analysis?
- What is a Radiance image?
- Why do I apply human sensitivity mapping?
- What glare calculations are possible?
- How do I calculate workplane illuminance?
- What other false color plots are possible?

Goal of Lighting Design/Analysis

The lighting designer needs to know about the design

- that it meets minimum lighting levels
  - for safety
  - for the type of space
  - for the kinds of activities
  - for the age of the occupants
- that there are not serious issues with visual discomfort
- that the design is aesthetically pleasing to the occupants

Radiance provides tools to answer these questions

What Is a Radiance Image?

File format uses a 4-bytes per pixel

- red, green, blue, exponent for red, green and blue
- allows for great dynamic range, so long as none of the individual components vary by more than a 32 bit exponent

Stores real-world luminance and illuminance

Can post-process image for further analysis

Can create a version of image which introduces artifacts demonstrating the limitations of human vision

Human Sensitivity Mapping

pcond maps image luminances to display brightnesses

- former linear mapping (with gamma) suffers from clamping
- human sensitivity mapping uses the eye's performance to simulate a subjective response to the image
  - contrast sensitivity full range of image luminances have an appropriate display brightness
  - low-light loss of acquity (blurring) and color perception
  - veiling glare
  - automatic, center weighted exposure adjustment

Glare Calculations

findglare is used analyze images for glare

- Guth Visual Comfort Probability
- Guth Disability Glare Ratio
- CIE Glare Index
- BRS Glare Index
- Unified Glare Rating
- Daylight Glare Index

xglaresrc is used to display sources of glare

Workplane Illuminance

Point-by-point using rtrace

- **rlux** takes care of converting RGB output to avg

Using falsecolor program to post-process images

- use **rpict** "plan" view to show surface illuminances
- use **rtrace** to calculate arbitrary plane in space

Other False Color Plots

falsecolor can create

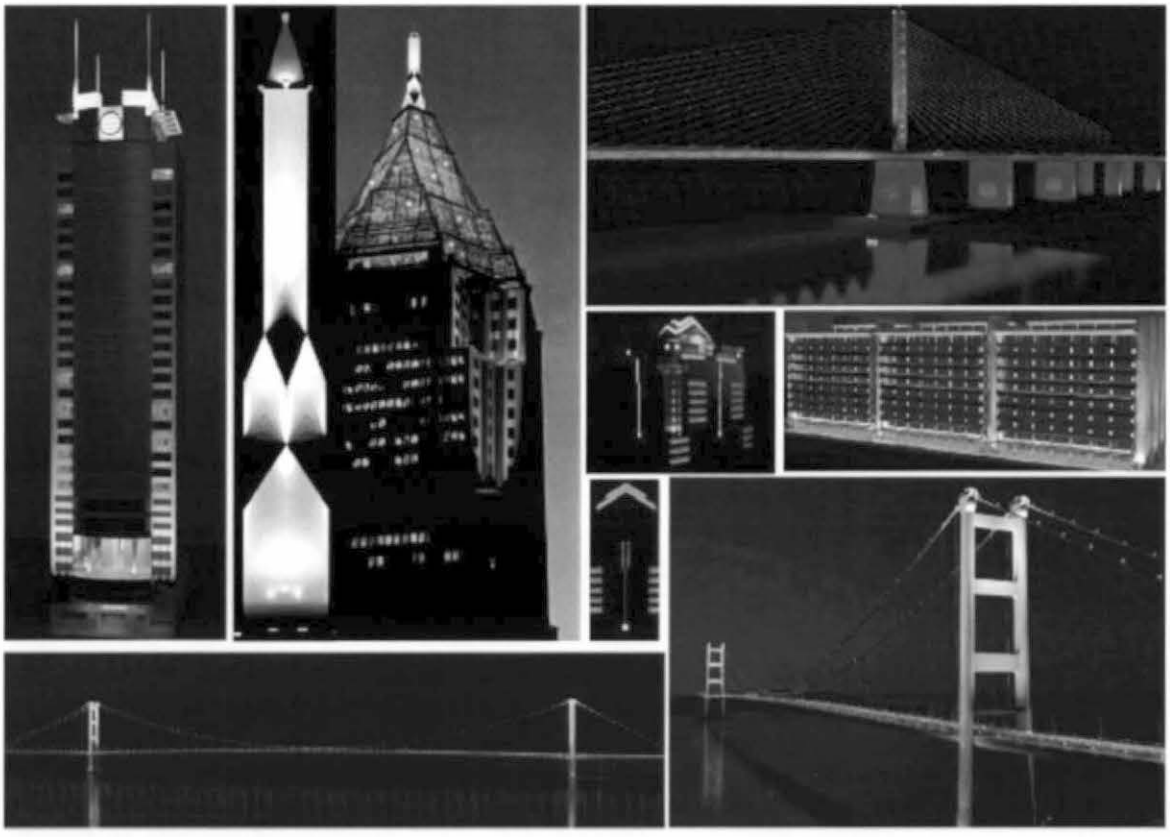
- continuous color variations
- isolux contour lines and bands
- can overlay these lines on top of a true-color image

units conversion is accomplished with scalefactor

- to display illuminance in footcandles
- **falsecolor -ip -s 179/9 91**

Rendering with Radiance
Illumination of Large Structures

by Rob Shakespeare
Indiana University
Theatre Computer Visualization Center
http://appia.tcvc.indiana.edu/~tcvc



Lighting large structures

Societal Issues

- light trespass and pollution
- safety: shipping, aircraft, roadways
- establishing landmarks

Lighting large structures

Investor Issues

- security
- privacy
- property value
- prestige

Who will use the Radiance Pictures?

Lighting Designers

- private preliminary explorations
- collaborative concept development
- final design review
- portfolio

Who will use the Radiance pictures?

Clients

- architects and engineers
- regulatory officers
- developers
- marketing/sales departments

Radiance pictures and concept

Concept conversations

- ambiguous sketches ~ passive comments
- detailed simulations ~ strong reactions
- faster than physical mock-ups
  - ~ more design ideas explored

Radiance pictures and concept

Concept conversations (con't)

- budget cuts ~ visual consequences
- pictures transcend language barriers

Radiance pictures and concept

- detailed engineering of photometry,
  - luminaire placement, and accurate surfaces
    - ~ accurate concept pictures
    - ~ design idea CAN be implemented

Data sets

Import from CAD

- quicker startup
- often larger data sets
- potentially slower rendering
- easy to maintain plans in CAD

Data sets

Build in Radiance

- slower startup
- effectively smaller data sets
- generally faster rendering
- harder to maintain plans in CAD

Large data sets

How to manage?

- partitioning data sets
- using instances
- simplifying test scenes

Large data sets

Insufficient environment space

- review object sizes
- review oconv options
- reduce number of surfaces
- review primitive types
- increase computer resources

Material and geometry considerations

- detail vs image resolution
- detail in context
  - backgrounds
  - separate data sets for close-up
- organic shapes

Luminaire and lamp selection

•Managing photometry and ies2rad
•keeping data current
•large quantities of files
•in line vs off line conversion

Luminaire and lamp selection

• the floodlight testbox
• white light design vs color
• issues of metamerism in the age of HID

Locating light sources

•the power of array
•you cannot instance a light source!

Controlling light sources:

•grouping by style, function, zone
•switching and dimming

Techniques for aiming luminaires

•consistent rotation and transformation procedures
•one at a time method
• attaching housings to Radiance lights
• offsetting lightsources

Techniques for aiming luminaires

Design aids

- 100% reflector targets
- test pattern projection
- rays of glow
- virtual camera at luminaire

Lines of Light: Cold Cathode and Neon

- needle in a haystack
- optimizing rendering
- the lighting metrics of cold cathode
- building a cold cathode tube in Radiance
- shape limitations

Transitions from Day to Night

- using gensky
- visual adaptation issues
- bringing the skyline to light

Special Tool Kit:

- water
- sparkle
- mist

Future practices

- off line editing of light changes
- immersive visualization

•augmented reality

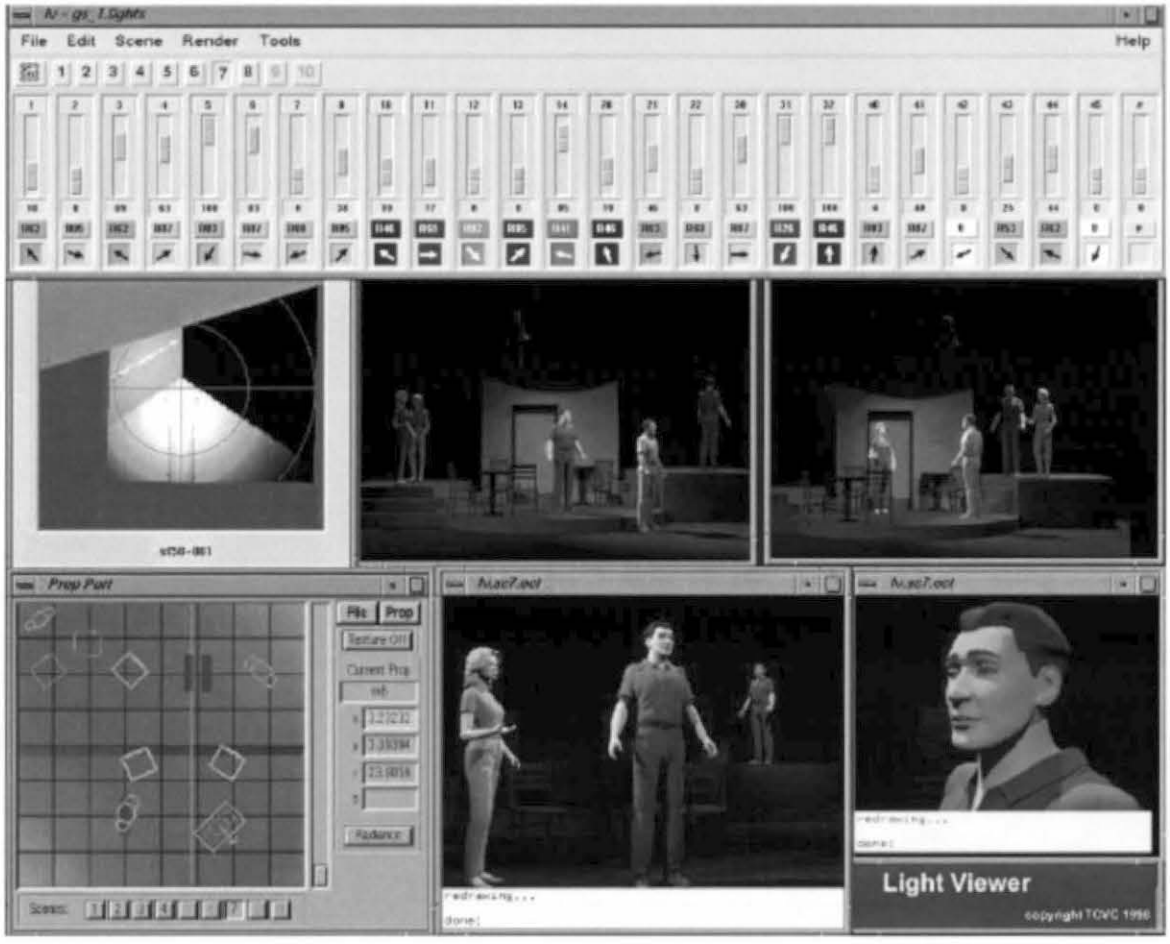
Radiance benefits

•complete design process in
a virtual environment
(without implementation…of course!)

Photo of opening night- Hong Kong Bridges

Rendering with Radiance
Theatre Lighting

by Rob Shakespeare
Indiana University
Theatre Computer Visualization Center
http://appia.tcvc.indiana.edu/~tcvc



Defining the challenge

A world of illusion

- driven by subjective response
- creates its own realities

Defining the challenge

Actors

- detailed faces
- gestures
- characters' costumes
- large data sets

Defining the challenge

Scenes

- hundreds of staging arrangements
  - scenic changes
  - actors blocking
- hundreds of associated lighting "looks"

Defining the challenge

Special materials

- faux finishes
- perspective painting
- draperies, scrims, projection, rp screens

Defining the challenge

Luminaires

- hundreds of uniquely focused luminaires
  - thousands of gel colors
  - different shaped beams
  - individual dimmer settings (red shift)
  - many luminaire types

Defining the challenge

Lighting

- suggesting natural lighting effects
- creating abstract effects
- patterns
- visible shafts of light
- strobes, lasers, flash pots, "lightning",
- INTERACTIVE CONTROL

Photometry

Aquisition and management

- appropriate IESNA format
- measuring methods
- modeling a variable focus ERS
- modeling a variable focus Fresnel

Photometry

Shaping the beam of light

- patterns
- shutters
- shutters in a variable focus ERS

The color of light

Colored filters (CIE Yxy)

- acquiring color data
- CIE Yxy to Radiance rgb via xyz_rgb cal
  - rcalc input and output formats
  - creating mycvt cal
  - generating the rgb color file
- ies2rad and the rgb color data (see figure 1)

The color of light:

Colored filters (spectral data)

- acquiring spectral transmission data
- converting spectral data to Radiance rgb
  - mgfilt creates illuminant's spectra
  - combine illuminant and gel spectra
  - convert to CIE xy using mgfilt
  - convert to rgb with rcalc and mycvt cal

The color of light

Color shifts resulting from dimming

- dimming curves and lumen multipliers
- color temperature vs Voltage (see figure 2)

The color of light

Combining dimming , color filters and photometry

- creating the input data format
- creating the ies2rad output file format
- managing the process with instr cal
- running the rcalc script
- ies2rad commands in the output file

The color of light:

Adaptation and normalization

- considerations
  - single light color environments
  - changing colored lighting
- determining "white" by modifying instr cal

Organizing the light plot

Controlling chaos!

- inline ies2rad and xform commands
- channel issues
  - duplicate luminaires - same channel
  - differing luminaires - same channel

Organizing the light plot

•separate aiming and photometry files

- pros change color/intensity without
- •rebuilding the whole octree
- cons traversing two files

Modeling special materials

•trans and the scrim effect

•mist and rock and roll

Interactive simulation

•focusing an instrument aiming, focus and shutters

•actor positioning

•painting the scene with light

Future applications built on Radiance

•off line editing

•new process based design interfaces

Figure 1
Gel Colors Conversion
CIE Yxy color data to *Radiance* rgb values

What follows are sample files which are used to convert CIE Yxy color data to *Radiance* rgb values

```
input fmt
$(name)   ${transmission}   ${myx}        ${myy}

cie in
straw     82 32             509           445
```

```
amber     70 32          533          425
red       7 21                 695          281
pink      70 63          482          388
ltblue    63 51          422          404 .
dkblue    5 71           204          187

 output fmt
$(name)   $(myR)    $(myG)    $(myB)

cvt cal

myX = myY*myx/myy
myY = transmission/100
myZ = myY*(1-myx-myy)/myy
myR = R(myX myY myZ)
myG = G(myX myY myZ)
myB = B(myX myY myZ)
```

Execute the this *rcalc* command to create a list if rgb values in the file *gel lst*

```
%rcalc -o output fmt -i input fmt -f cvt cal -f xyz_rgb cal cie in > gel lst
```

gel lst (contents of output file)

```
ltstraw   1 343    0 727    0 137
straw     1 421    0 669    0
amber     1 414    0 492    0
red       0 370    0        0 002
pink      1 330    0 510    0 166
ltblue    0 851    0 590    0 211
dkblue    0 019    0 057    0 209
```

Figure 2
Values to insert into the *Radiance* ies2rad program to approximate the effects of dimming a Tungsten Halogen light source

Control Level	Lumen output multiplier	3200°K lamp r	g	b	3000°K lamp r	g	b
100%	1 00	1 377	0 913	0 351	1 450	0 890	0 292
90%	81	1 407	0 904	0 326	1 481	0 880	0 268
80%	64	1 440	0 893	0 299	1 517	0 868	0 243
70%	50	1 479	0 881	0 270	1 559	0 854	0 216
60%	35	1 527	0 865	0 237	1 610	0 837	0 186
50%	25	1 584	0 845	0 201	1 672	0 816	0 152
40%	16	1 660	0 820	0 158	1 751	0 788	0 115
30%	09	1 760	0 785	0 112	1 864	0 748	0 070
20%	04	1 876	0 743	0 065	1 876	0 743	0 065
10%	01	1 876	0 743	0 065	1 876	0 743	0 065
0%	00						

To produce a *Radiance* light source of a spotlight which has been dimmed to 50% on a fader use the following command

ies2rad l default m 25 -c 1 672 0 816 0 152 -o channel1 spotlight ies

Radiance Calculation Methods

Greg Ward Larson
Silicon Graphics, Inc

Overview of Calculation

- Solve the following integral equation

- Direct calculation removes large incident
- Indirect calculation handles most of the rest
- Secondary light sources for problem areas
- Participating media (adjunct to equation)
- Parallel rendering to accelerate process

Direct Calculation

Selective Shadow Testing
- Only test significant sources

- Adaptive Source Subdivision
  - Subdivide large or long sources
- Virtual Light Source Calculation
  - Create virtual sources for beam redirection

Selective Shadow Testing (1)

Selective Shadow Testing (2)

- Sort potential direct contributions
  - Depends on sources and material

Test shadows from most to least significant
- Stop when remainder is below error tolerance

- Add in untested remainder
  - Use statistics to estimate visibility

Selective Shadow Testing (3)

Selective Shadow Testing (4)

Adaptive Source Subdivision

Virtual Light Source Calculation

Indirect Calculation

- Specular Sampling
  - sample rays over scattering distribution
- Indirect Irradiance Caching
  - sample rays over hemisphere
  - cache irradiance values over geometry
  - reuse for other views and runs

Indirect Calculation (2)

Specular Sampling

Indirect Irradiance Caching

- Indirect irradiance is computed and interpolated using octree lookup scheme

Secondary Light Sources

- Impostor surfaces around sources
  - decorative luminaires
  - clear windows
  - complex fenestration

- Computing secondary distributions
  - the **mkillum** program

Impostor Source Geometry

- Simplified geometry for shadow testing and illumination computation
  - fits snugly around real geometry which is left for rendering direct views

Computing Secondary Distributions

- Start with straight scene description
- Use **mkillum** to compute secondary sources
- Result is a more efficient calculation

Participating Media

Single-scatter approximation

The *mist* material type
- light beams
- constant density regions

- Rendering method

Single-scatter Approximation

- Computes light scattered into path directly from specified light sources
- Includes absorption and ambient scattering

The *Mist* Material Type

- May demark volumes for light beams

  May increase medium density or change scattering properties within volume

Rendering Method

- After standard ray value is computed
  - compute ambient in-scattering, out scattering and absorption along ray path
  - compute in-scattering from any sources identified by *mist* volumes ray passes through

    this step accounts for anisotropic scattering as well

Parallel Rendering

Goals
- Parallel computation on variety of architectures

  multiprocessor machines

  networked machines
- Data sharing for best speed and memory

- Method
  - Large grained parallelization
  - Data sharing locally and over NFS

Portable Parallelization

- Don't depend on libraries or architectures

  Use only standard UNIX and NFS features
  - share memory via `fork(2)` system call
  - share new data over NFS with lock manager

  Avoid threads using coarse-grained method

Rendering Animations

- Multiple rendering processes on one or more machines sharing data with NFS

  Each process opens next unstarted frame using `open(2)` system call with `O_EXCL`
- **Ranimate** program manages overall animation process and handles recovery

Rendering Large Images

Break image into small, equal-sized blocks

- Assign blocks to processors sequentially
- Usually achieves over 95% utilization with negligible redundancy (i e , linear speedup)

Conclusions

- *Radiance* has been researched and developed over the past 12 years
- Provides a practical compliment of tools and methods for lighting visualization
- Field tested effective for large models and novel electric lighting and daylight systems

References

- Larson, G W and R A Shakespeare, *Rendering with Radiance the Art and Science of Lighting Visualization*, Morgan Kaufmann Publishers, 1998
- Larson, G W , H Rushmeier, C Piatko, ' A Visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes," LBNL Technical Report 39882, January 1997
- Ward, G , "Making Global Illumination User-Friendly," Sixth Eurographics Workshop on Rendering, Springer-Verlag, Dublin, Ireland, June 1995
- Rushmeier, H , G Ward, C Piatko, P Sanders, B Rust, " Comparing Real and Synthetic Images Some Ideas about Metrics," Sixth Eurographics Workshop on Rendering Springer-Verlag Dublin, Ireland, June 1995
- Ward G , "The RADIANCE Lighting Simulation and Rendering System," *Computer Graphics*, July 1994
- Rushmeier, H , G Ward, ' Energy Lighting Simulation and Rendering System,' Computer Preserving Non-Linear Filters," *Computer Graphics*, July 1994
- Ward, G , A Contrast-Based Scalefactor for Luminance Display " *Graphics Gems IV*, Edited by Paul Heckbert, Academic Press 1994
- Ward, G , 'Measuring and Modeling Anisotropic Reflection, ' *Computer Graphics*, Vol 26, No 2, July 1992
- Ward, G P Heckbert "Irradiance Gradients " Third Annual Eurographics Workshop on Rendering, Springer-Verlag, May 1992
- Ward, G "Adaptive Shadow Testing for Ray Tracing ' Photorealistic Rendering in Computer Graphics proceedings of 1991 Eurographics Rendering Workshop, edited by P Brunet and F W Jansen, Springer-Verlag
- Ward, G , "Visualization," *Lighting Design and Application*, Vol 20, No 6 June 1990
- Ward G , F Rubinstein, R Clear, "A Ray Tracing Solution for Diffuse Interreflection," *Computer Graphics*, Vol 22, No 4, August 1988
- Ward, G , F Rubinstein, "A New Technique for Computer Simulation of Illuminated Spaces," *Journal of the Illuminating Engineering Society*, Vol 17, No 1, Winter 1988

See the RADIANCE Reference Materials page at radsite lbl gov/radiance/refer for additional information

H Advanced Daylight Calculations: Case Study

John Mardaljevic

Design Problem

What is the likelihood of daylight produced glare impairing the visibility of a large "video-wall" display against an expanse of South-facing glazing?

1 Introduction

This section describes how a new design evaluation methodology was devised and then applied There was no off the shelf solution available that could solve the problem outlined below What follows is how a project diary might have looked It explains stage by-stage how a workable solution was arrived at in a short space of time A few alternative approaches that were considered - briefly - but which fell by the wayside are discussed also

2 Getting started

2 1 The building model

The essentials of the design problem were discussed with the client, in this case, a consulting engineer At this stage, we are not aware of an accepted evaluation technique that we can apply to this problem - a fact that we may or may not disclose to the client[1] We are nonetheless confident that we will come up with a workable solution, and we agree to do the work

The Trafford Centre Development, Manchester, provided the setting for the analysis The area of interest for the glare study was a large, circular (on the southern flank) food court hall (diameter ~50m) This area will have a circular arc of glazing along the southern wall Facing into the food court, just in front of the glazing, there will be a "video-wall" installation, i e an array of TV monitors configured to mimic a single large screen Beyond the glazing there will be a

1 The client was of course in this case informed of this fact

open piazza, which will be bounded by a colonnade arc concentric with the glazing Otherwise, the area to the south of the piazza will be fairly open and there are not expected to be any major obstructions to views south beyond the colonnade

We examine the architects drawings and we begin to plan out how we will construct a *Radiance* model of the scene The detail we give to the model will depend on one or more of the following factors

- the nature of the problem, in this case daylight glare,
- the view parameters - if the analysis is image based, and,
- the available time

Even without a fully developed evaluation methodology we are still thinking about this - a little careful consideration of the problem should give us an idea on how to proceed From the viewpoint of a person inside the proposed building - looking south towards the video-wall - problem areas of high luminance are likely to be either

- glazing through which the sky is directly visible, and/or,
- light coloured structural/decorative surfaces which are directly illuminated by the sun

For either case, it will be the magnitude of the high luminance areas and their extent in the field of view that are likely to be the principal factors for any measure of glare Note that we have made no mention of the contribution of inter-reflected light, from either the sun or the sky This is, we hope, a reasonable assumption, and later we will see that it has critical bearing on the mode of the final analysis

Having decided what light transfers are important, we can decided how to proceed with the modelling Structure that is important to the two light transfer mechanisms identified above needs to be modelled in detail These structures include the glazing and those internal/external surfaces that will be seen in the field of view and which might be illuminated by direct sun Our exact view points have yet to be decided, but we can guess that they will be looking south from the main body of the building The building model that was finally assembled gave an accurate representation of the (internal) view of the sky through the glazing which forms the south-facing facade of the piazza Internal and external obstructions modelled were those that might obscure the view of the sky, or alternatively, be directly illuminated by sunlight Therefore, the ceiling structure and the East/West sides of the building were not modelled in detail The rest of the building can be modelled as a light tight enclosure, Figure H-1 We will assess the likelihood of daylight glare from two positions in the Food Court View A is from the middle of the ground floor area looking up towards the video-wall and View B is from the mezzanine level Renderings of the model showing external and internal views are given in Figure H-2

2 2 Formulating a methodology First thoughts

During the modelling, we are actively thinking about how we will approach the analysis Here are some of the 'first thought' ideas that were considered, and why they were not used

The standard *Radiance* release includes, of course, a collection of programs designed to locate sources of glare in a rendering These are **findglare**, **glare**, **glarendx** and **xglaresrc** We could, for example, generate a rendering for a sky condition that is likely to cause problems,

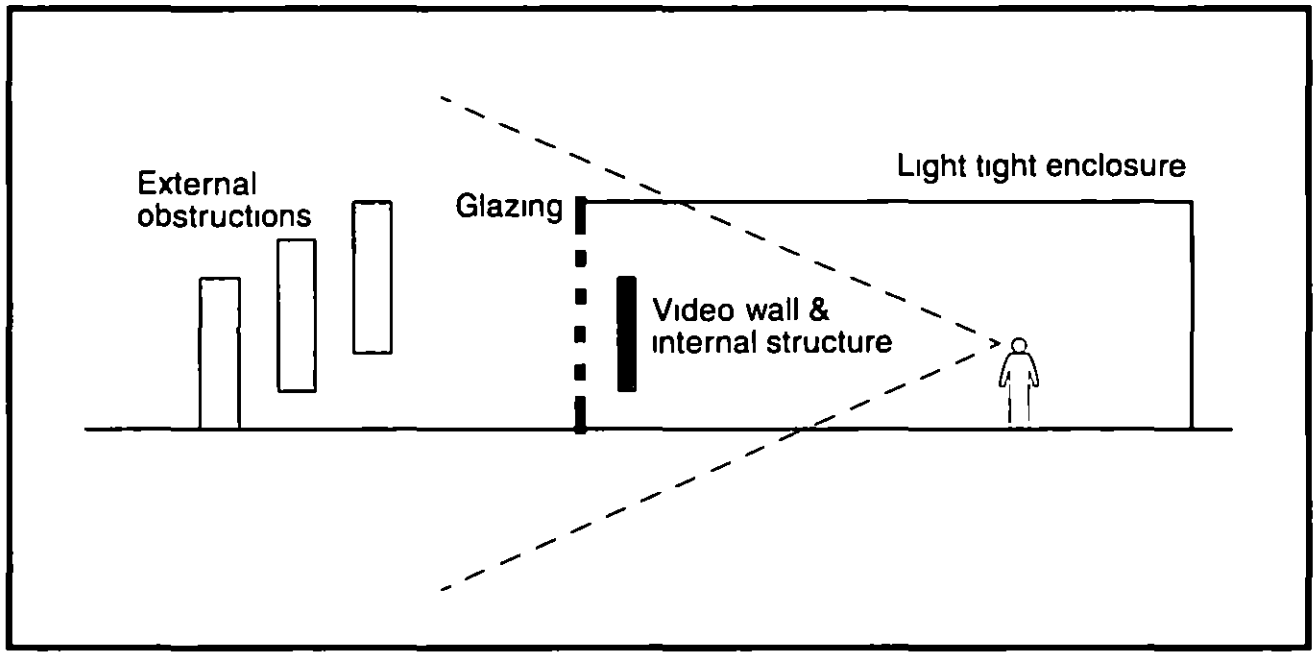


Figure H-1 Schematic of model detail

say, low sun around midday And then use the findglare program to locate potential sources of glare in the field of view In the heat of the moment, this might seem a reasonable way to proceed However, we soon realise that we need to consider a few more issues before we should go further along this route The questions that we should ask are

1 The glare formulations built-in to the *Radiance* (glare) programs, are they appropriate for daylight glare?

2 What does one, or even a handful, of sky conditions tell us about the overall likelihood of glare?

3 From what data should we generate sky conditions?

4 What sky model, or models, should we use?

To answer these, we need to acquaint ourselves with some of the current research on daylight glare and sky models

2 3 Glare a brief review

The basic studies on discomfort glare examined the veiling effects of small glare sources (~0 01 sr) in the field of view [Hopkinson 63] From these experiments, a quantitative measure for glare was derived which depended on the glare source characteristics (source luminance, solid angle, position factor) and the background field luminance of the viewing environment Increasing the glare source size beyond the small solid angles used in these experiments did not lead to the expected increase in perceived glare due to the adaptation effects of the eye Further studies, carried out at the Building Research Establishment (UK) and Cornell University (USA), addressed the problem of large glare sources i e windows, and eye adaptation The

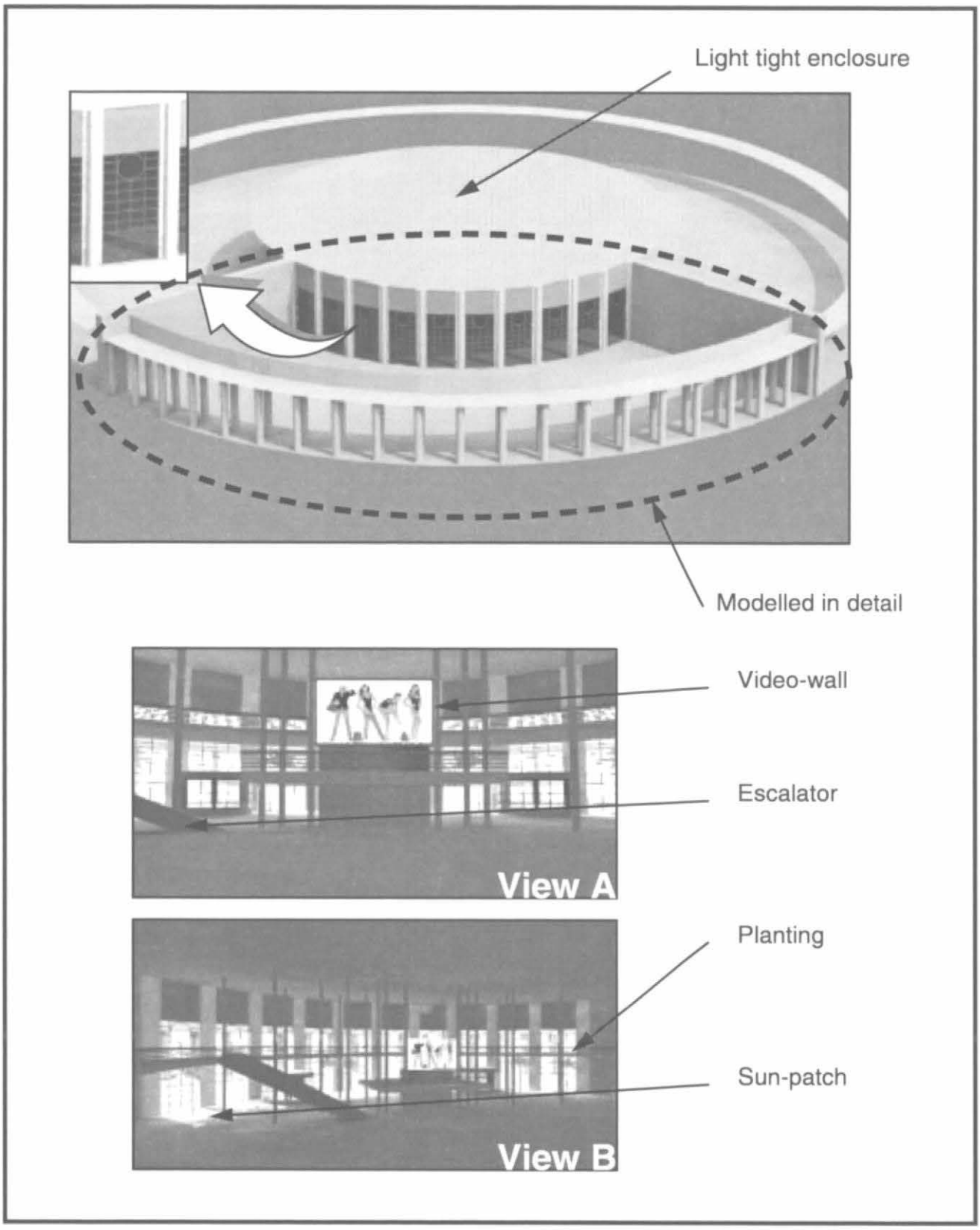


Figure H-2. Renderings of the building model

relation derived from this later work, now known as the 'Cornell formula', gave G, the glare constant as

$$G = K\frac{L_s^{1\,6}\Omega^{0\,8}}{L_b + 0\,07\omega^{0\,5}L_s} \qquad \text{(H 1)}$$

where K is a constant depending on the units and where L_s, L_b and ω are respectively the source luminance, the surround luminance and the solid angle subtense of the source at the eye Ω is the solid angle subtense of the source modified for the effect of the position of its elements in the field of view A relatively recent review by Chauvel *et al* [82] concludes however, that discomfort glare from a window is practically independent of window size and distance from the observer, but that it is critically dependent on the sky luminance In that case, the relationship between a quantitative measure for daylight glare and the sky luminance may be simpler than that given by the Cornell formula The authors of this review paper note that the general conclusions presented were based on "limited experimental data", and that "there is a need to make further investigations to derive a prediction method which correlates more closely with the differing experimental results"

If we follow Chauvel's findings, we need only consider the perceived sky luminance, and, for our purposes, sun illuminated surfaces However, it might be instructive to also determine the extent of the high luminance areas in the field of view We shall therefore need to write an analysis program of our own This program will determine, from a *Radiance* rendering, the frequency of occurrence of high luminance pixels in the image We shall not therefore need the *Radiance* glare programs

2 4 Meteorological data

Next we should consider the scope of the analysis - how many different sky conditions do we need to evaluate? The sky brightness can vary enormously from one moment to the next, and it is not possible to generate a single sky luminance distribution which is completely representative of the naturally occurring range of conditions The actual sky brightness distribution is generally not known, and instead it is usually generated from an integrated quantity which is a measure of the total illumination (or irradiation) due to the sky

A worst case scenario - low midday sun with clear sky conditions - is a one-off calculation, but it tells us nothing about how likely an event this is To have any credibility, the analysis needs to be based on the likelihood of daylight glare occurring over a full year How than can we account for all the variation in sun and sky conditions that occur in a year? Before we try to answer this, let us first consider a related issue from what data do we generate the sky conditions? Ideally, we need a time-series of measurements from which we can derive daylight conditions For this analysis, an annual time-series of hourly values for diffuse horizontal irradiation and direct normal radiation was used These data were a standard meteorological set recorded at the Finningley station (Sheffield) - climatically similar to nearby Manchester A plot of the Finningley data shows the hour-by-hour variation in the irradiation parameters The data are presented as 365 by 24 arrays (day number by hours), and the hourly irradiance value has been mapped to colour, Figure H-3 The variation in the direct normal irradiance is particularly apparent, but what the plot doesn't show is the changing sun position An arc describing the passage of the

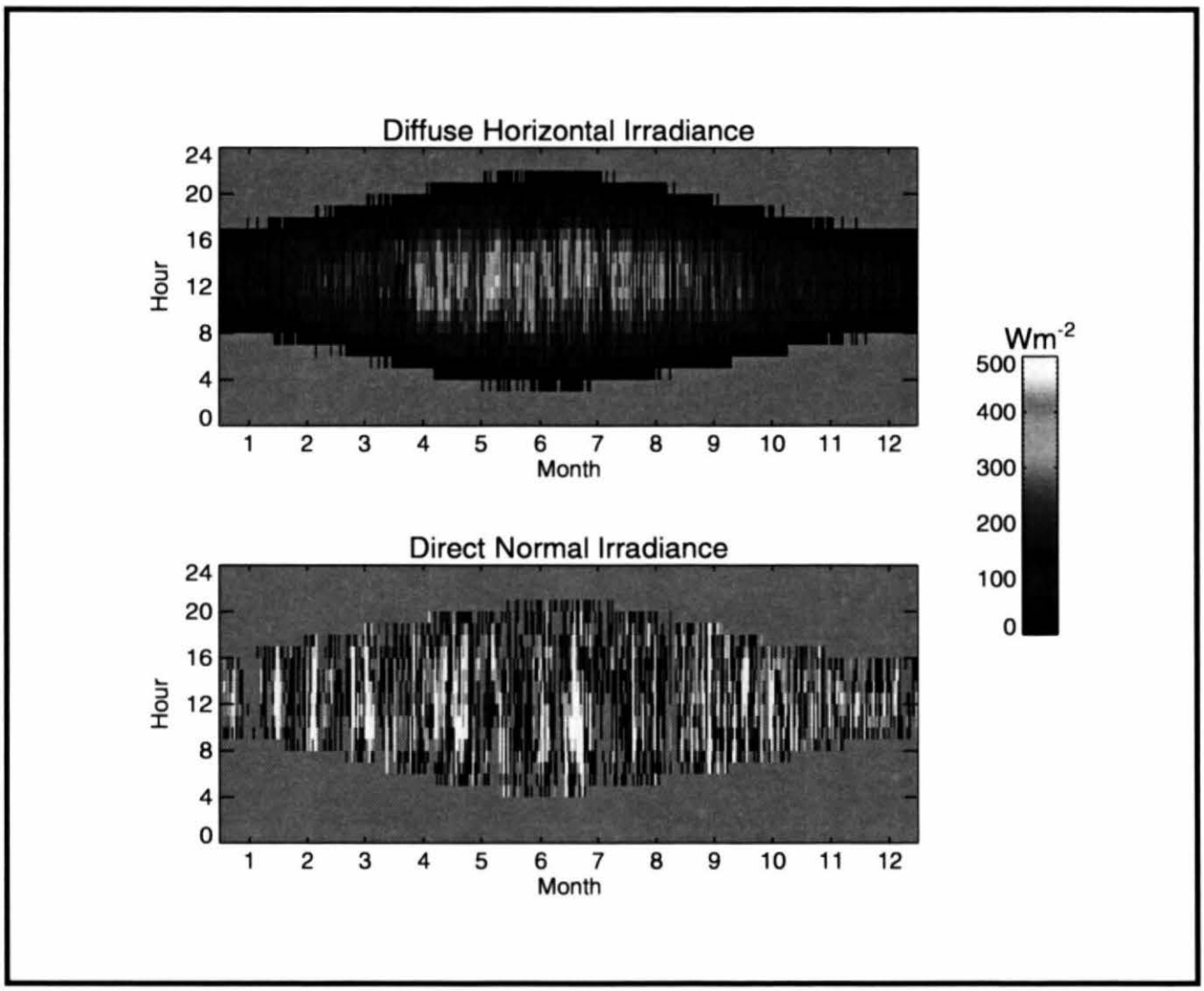


Figure H-3. Annual maps for ir*radiance* values from TRY data

sun across the sky vault will increase in extent from the winter to the summer solstice, thereafter it will decrease at the same rate.

To make some account of the changing sun path throughout the year, we could look at, say, one day at the start of each month. This at least covers much of the *range* of possible sun paths. We might then try to average the meteorological data to obtain a single day that would be representative of each month. A comparison of the frequency distribution of the diffuse and direct normal illuminances for the raw and averaged data reveals however that the averaging process is suppressing, to an unacceptable degree, the high brightness days that are likely to be the main cause of glare. One way to account for these conditions might be to include a limited number of bright sunny skies. Selective use however of bright sunny days can also result in erroneous findings because, not only must the limited number of bright days be somehow representative of the entire year, but the same should be the case for the sun angles. The conclusion from this preliminary study was that a significant bias would be introduced if averaged and/or selected skies were used. It would appear from this that we need to consider *every* hour of daylight in the time-series, approximately 4000 different skies!

A further analysis of the annual time-series revealed however that the occurrence of bright days was largely the same in the first six months as the last six months And, importantly, the occurrence of sun angles in the first (or last) six months is representative of those for the entire year In the light of these findings, we decide to pursue a "semi-brute-force" approach where the analysis will be based on the predictions for the first 182 days of the meteorological data at the 1 hour time step The target period would be all the skies of non-zero brightness between the hours of 08h30 and 18h30 The number of daylight hours in the target period for the first 182 days of the time-series was 1,820 The simulation results will be scaled to represent the probability of glare for a full year

2 5 Sky models

Thus far, we have deduced that the analysis should account for the magnitude and extent of the high luminance areas of the renderings, and that a credible evaluation requires the evaluation of approximately 2000 skies for each viewpoint The remaining question that we must answer is this what sky model(s) is best suited to this application?

The official *Radiance* sky generator program, **gensky**, supports these sky models

- the uniform luminance model,
- the CIE overcast sky model,
- the CIE clear sky model, and,
- the Matsuura intermediate sky model

The absolute luminance of any of these sky types is controlled by supplying the program with either the zenith radiance or the diffuse horizontal irradiance The clear and intermediate sky models allow the option to automatically create a description for the sun In which case, the solar radiance is either directly supplied to the program or calculated from horizontal direct irradiance The sun position can be either supplied as altitude and azimuth arguments, or it may be calculated from the time and the geographical coordinates The uniform luminance model is unrepresentative of any naturally occurring sky conditions and is therefore excluded from any further consideration

The CIE Standard Overcast Sky, originally known as the Moon and Spencer Sky [Moon 42], was devised to represent the luminance distribution observed for overcast skies Adopted as a standard by the CIE in 1955, this description is the one most frequently used for illuminance modelling In this model, the sky brightness increases gradually with altitude from the horizon to the zenith, but it does not vary with azimuth

The description the brightness distribution for a CIE clear sky requires a fairly complex mathematical representation The complexity arises from a number of observed effects that are accounted for in the model These include brightening of the sky close to the solar position and a luminance gradient which may change sign along an arc from the sun across the zenith and down to the horizon The scale of these effects are related to the solar position and the relative magnitude of the illumination produced by the sun and sky

The overcast and clear CIE models are representations of extreme sky types - densely overcast or completely clear Intermediate skies, that is thin/moderate cloud cover and/or hazy

atmospheric conditions, are more likely occurrences than totally clear or overcast skies for many geographical locations

The "Matsuura intermediate sky" model describes sky conditions that have a higher turbidity than the CIE clear sky model In comparison to the CIE clear sky model, the intermediate formulation generally predicts lower luminance for the circumsolar region and slightly higher zenith luminances Additionally, horizon brightening which can be a prominent feature of the clear sky model, is usually absent

The basic characteristics of these sky models are shown in Figure H-4 Here, the sky point luminance along an arc from the horizon due South, across the zenith to the horizon due North, is plotted for the overcast, intermediate and clear sky models Each sky model was generated to provide the same diffuse horizontal illuminance (30,000 lux) The sun altitude and azimuth was set to 45 and 180 respectively The sun position provides the locus for the circumsolar region of the sky for the intermediate and the clear sky models, but the sun itself was not modelled

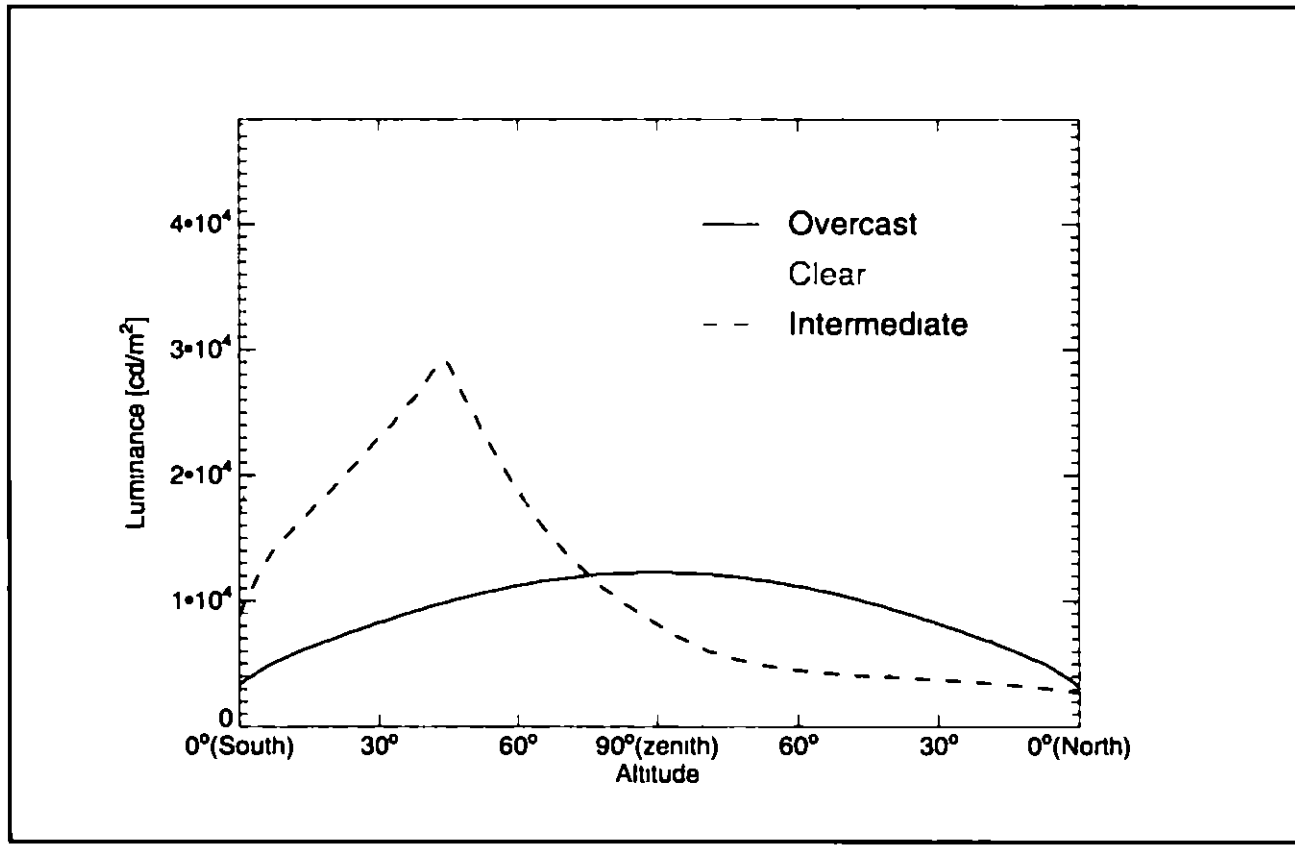


Figure H-4 Sky luminance profiles for 3 sky models

What this example demonstrates is how very different the sky luminance distribution can be, depending on the sky model type At the sun position, the sky luminance is (approximately) 10^4, $3x10^4$ and $4x10^4$ cdm ^{2} for the overcast, intermediate and clear skies respectively The outcome of the glare analysis that we have in mind will therefore be very sensitive to the sky model that we use In reality, of course, we could not hope to use just one of the above sky

models to represent all of the naturally occurring sky brightness distributions We could try to estimate, at each hour of the time series, the most likely sky conditions - overcast, intermediate or clear - from, say, the absolute magnitude and ratio of the irradiance quantities Or, more ambitiously, we could try to make blends of two or more of the sky models based on some quantity derived from the time-series, e g clearness index Approaches such as these are perfectly valid, but they tend to be the preserve of experts in sky models We need something simpler Fortunately, help is at hand in the form of the **gendaylit** program This produces a sky brightness distribution based on the Perez All-Weather model [Perez 93] An attractive feature of this model is that the full range of naturally occurring sky conditions are accommodated within a single theoretical scheme The appropriate sky type is generated automatically from the basic input parameters This is the sky model that we will use

Note, all the sky models generate continuous sky luminance distribution patterns The discontinuous aspects of skylight - instantaneous cloud patterns - are not addressed And the spectral distribution of skylight - its colour - is not predicted by any of these models

Consistent with the overall uncertainties in this analysis, global and diffuse illuminances were derived from the irradiance time-series using a fixed value for luminous efficacy (100 lm/W)

3 Running the simulations

We are now ready to prepare the way for generating the huge number of renderings - 1,820 for each viewpoint - that we need for our analysis We could, if disk space allows, save all the renderings and post process them later Otherwise, we can examine each rendering for high luminance areas (that is, count the number of high luminance pixels) as it is created, and then delete it If disk space is limited and we choose the second option, we must ensure that we extract from the rendering all the information that we need, otherwise we will have to repeat the entire sequence later

The execution of each sequence of 1,820 renderings and the reduction of the data can be achieved in many ways For the case study which this 'diary of events' is based on, the author used a data analysis package called PV-WAVE Programs written in PV-WAVE were used to control the sequence of simulations and process all the renderings For each hour in the time-series, the 'executive' program determined the gendaylit input parameters and wrote them to a temporary file The program then spawned a C-shell script (see Appendix) which

- loaded the gendaylit parameters and created the *Radiance* sky description,
- added the sky description to an octree of the building model, and,
- executed the rendering pipeline command

The PV-WAVE process waits until the child process - the rendering is finished before continuing The 'executive' program could have been written in virtually any programming language, including shell scripts

The bulk of the computational effort will be expended on the rendering pipeline command Because we are not enabling the inter reflection calculation, the rendering time should be fairly short, and linearly dependent on the image size, that is the number of pixels We can therefore, from just one rendering, estimate how long the entire sequence will take Batch simulation of this magnitude are usually run overnight or at weekends If, from the test rendering, we find that the estimated time for the sequence to complete is too long, we can reduce the image

dimensions to fit the available time Renderings of just a few hundred pixels across (maximum dimension) should be sufficient

The simulations will generate a lot of input/output traffic To reduce impact on the local network, the storage disk should be on the SCSI bus connected to the processor which is doing the renderings Note that the sky description is added to the building description octree This is computationally much faster than recreating the entire scene octree for every sky in the time-series

4 Results

Luminance data from all the renderings were aggregated into annual profiles which show the frequency of occurrence of high luminance areas in the field of view and cumulative totals

Annual profiles for the frequency of occurrence and cumulative totals are presented for View A (Figure H-5) and View B (Figure H-6) Each non-zero entry in the map indicates by shade either the number of hours (frequency of occurrence) or the percentage of the year (cumulative total) for which a given threshold luminance is exceeded across a percentage range (or percentage) of the screen The arrows overlaid on Figure H-5 demonstrate how this is read Here the arrows show that, for View A

1 a luminance of 8,000 Cd/m^2 is exceeded across 2 to 3% of the field of view for approximately 40 hours during the year, and that,
2 for approximately 15% of the year, a luminance of about 3,000 Cd/m^2 is exceeded across about 5% of the field of view

The frequency maps reveal the distribution in the exceedence of the threshold luminances, but for overall assessment the cumulative maps are more useful Comparison of these for the two viewpoints shows that the Mezzanine level view (B) has a lower propensity for high luminance areas, and that these generally occur across smaller percentages of the field of view than for the floor level (A) viewpoint

For View A, luminances ≥ 9,000 Cd/m^2 (i e potential glare sources) were predicted to occur for ~5% of the working year, but only across ~1 5% of the field of view (Figure H-5b)

From the Mezzanine level viewpoint (B), where high altitude sky was not visible, luminances in excess of 6,000 Cd/m^2 were not predicted Thus views towards the "video-wall" from the mezzanine level will generally be less prone to daylight glare than ground floor views

Since the building model was (necessarily) an incomplete description of the finished structure, the actual obstruction of view to the sky will almost certainly be greater than that modelled here Therefore, although glare is unlikely to be eliminated entirely, the frequency of occurrence of and the percentage of the field of view affected would be even less than that predicted On the basis of this analysis therefore, it seems unlikely that the visibility of the "video-wall" will be impaired by daylight glare for a significant period of the year

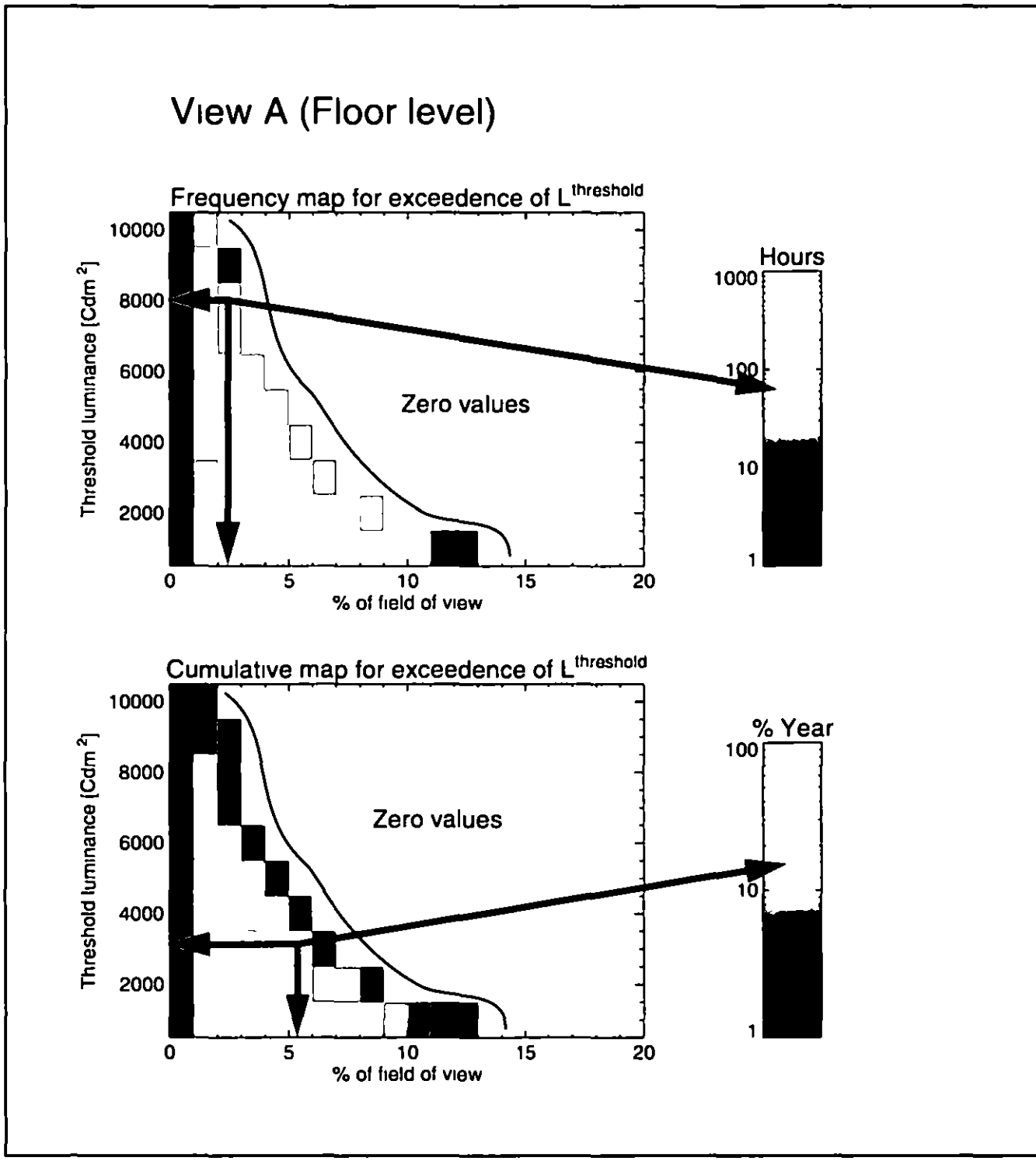


Figure H-5 Annual profiles - frequency of occurrence (a) and cumulative totals (b) for View A

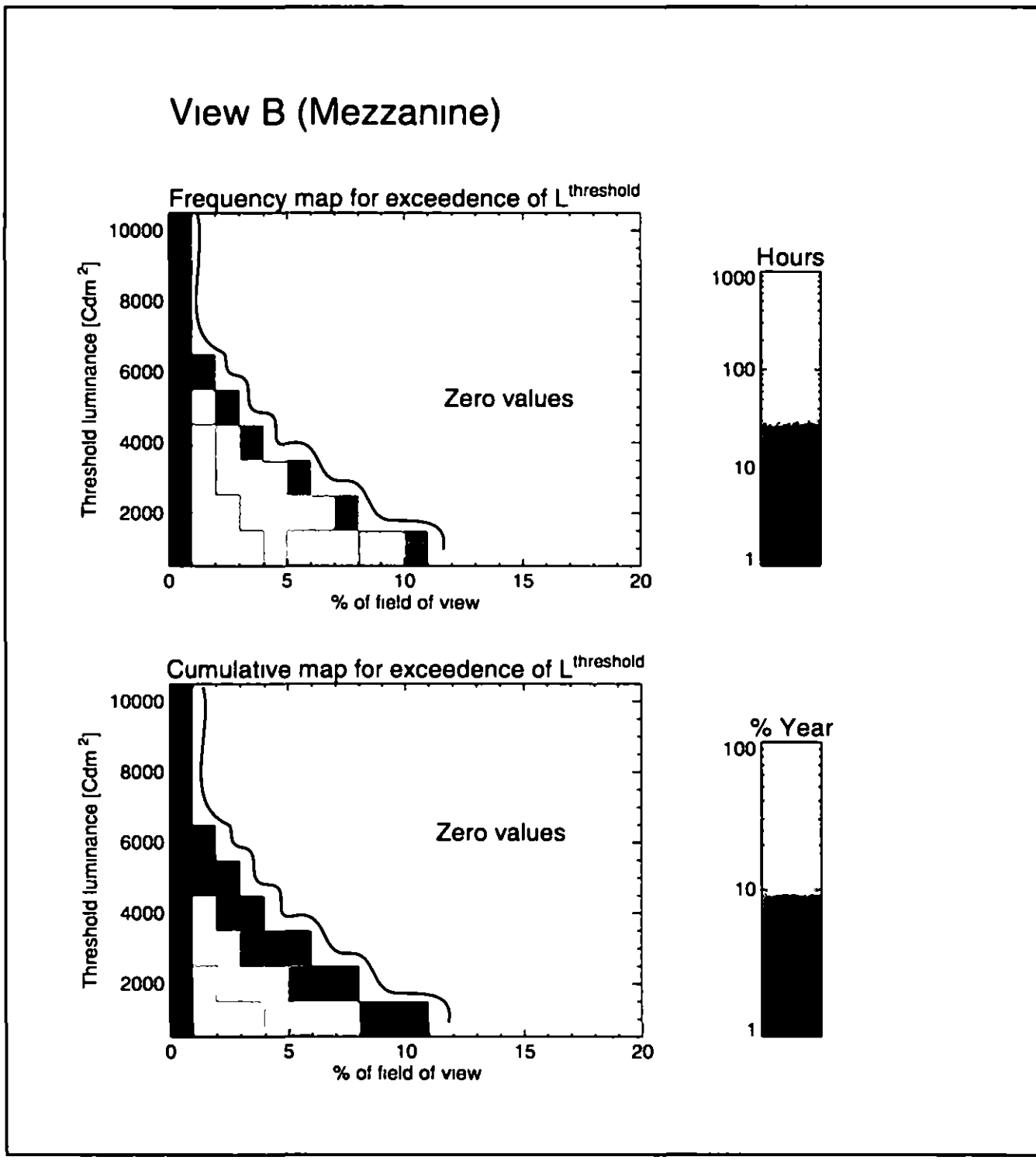


Figure H-6 Annual profiles - frequency of occurrence (a) and cumulative totals (b) - for View B

5 Discussion

The simulation based approach reported here has demonstrated that an annual estimate for the likelihood of glare based on an analysis of several thousand skies is an achievable goal The methodology described, and the various working assumptions employed, were commensurate with the uncertainties associated with sky modelling and the current understanding of glare

Sky luminance distributions have recently been measured at various stations across the world as part of the International Daylight Measurement Programme [Kendrick 89] The data from these stations have been used to test the accuracy of predicted sky luminance distributions against measurements of real skies for many different sky conditions [Littlefair 94] The sky luminances were measured using a 145 patch scan pattern The luminance of the patch closest to the sun position, however, was not recorded, and so the sky model predictions for this, usually, the brightest region of the sky, could not be assessed The degree of predicted glare will, on non-overcast days, be particularly sensitive to the magnitude and (angular) distribution of sky luminance about the circumsolar region To test sky models for this particular application therefore, would require further validation of sky model luminance predictions, at high resolution (> 145 patches) and including the circumsolar region

The accepted glare formulations, as already noted, cannot readily be applied to daylight glare evaluation with a high degree of confidence This is particularly so for this application, since display installations, in contrast to desktop monitors, are generally not intended for long term viewing of small scale image features It is likely therefore that disability glare criteria derived from experiments in an office environment might be significantly relaxed for the less exacting demands of entertainment, advertising or headline broadcast

It is possible to relate pixel location in the simulated image to a direction vector and associated solid angle With this information it would be a relatively straightforward task to apply a position factor weighting to each pixel in the glare estimation The analysis could then be modified to use a more complex glare formulation if it was found to be appropriate

Acknowledgments

The analysis described here was based on work carried out by the Institute of Energy and Sustainable Development (IESD) for WSP Consulting Engineers for and on behalf of their client and end user, The Trafford Centre Limited

References

[Chauvel 82] Chauvel P, Collins J B, Dogniaux R and Longmore J, *Glare from windows current views of the problem*, Lighting Res Technol 14 (1) 31-46 (1982)

[Hopkinson 63] Hopkinson R, *Architectural physics lighting*, London HMSO, (1963)

[Kendrick 89] Kendrick J D, *Progress towards the CIE International Daylight Measurement Year, 1991*, Proc CIE Conf "Daylight and solar radiation measurement" Berlin (1989)

[Littlefair 94] Littlefair P, *A Comparison of sky luminance models with measured data from Garston, UK* Solar Energy 53 (4), 315 322 (1994)

[Perez 93] Perez R, Michalsky J and Seals R, *An all-weather model for sky luminance distribution* Solar Energy 50 (3), 235 245 (1993)

Appendix C-shell script

```
#!/bin/csh -f
#
# Shell script to create rendering as luminance data for
# glare analysis
#
# Set shell variable perpars to contents of file perdat
# Set variables for month day, hour, direct normal illuminance and
# diffuse horizontal illuminance
#
set perpars = `cat perdat`
set mon = $perpars[1]
set day = $perpars[2]
set hr = $perpars[3]
set dnil = $perpars[4]
set dhil = $perpars[5]
#
# Set geographical coordinates
#
set vf   = -b
set rundir =  /runs/test1
set lat   =  53.0
set lon   =  -2.35
set mer   =  0.0
set coord = ( -a $lat -o $lon -m $mer)
#
# Set rendering and gendaylit parameters
#
set ab =    0
set dim = 300
set res =   ( -x $dim -y $dim)
set view = ( -vf $vf vf)
set foct = wsp_main_rz.oct
set skypar = ($mon $day $hr $coord -L $dnil $dhil)
set av =    1
set av  = ($av $av $av)
set paramb=(-ab $ab -av $av)
#
# Add sky description to (frozen) octree of building  Execute rpict and pipe
# output to pvalue for conversion to binary data format
#
oconv -w -i $foct \!gendaylit "$skypar" -o sky_glow.rad > sky$foct
set fileid = $mon$day$hr.lum
rpict $view $paramb $res sky$foct \
        | pvalue -H -h -df -o -b \
        > $rundir/$fileid
rm sky$foct
```

I.2 Validation Studies: Daylight

John Mardaljevic

Summary

The illuminance predictions from the lighting simulation program Radiance are compared with measurements taken in full scale experimental rooms under real sky conditions The simulation program used sky luminance patterns based directly on measured sky brightness distributions Uncertainties in the model sky representation are therefore greatly reduced, allowing a detailed evaluation of the absolute accuracy of the program under realistic conditions Results are presented for 754 skies covering all types from heavily overcast to very clear The error characteristics of the illuminance predictions at each of the room photocell positions were analysed

1 Introduction

Recent advances in computer graphics techniques allow, in principle, the modelling of realistic architectural scenes for visualisation and illuminance prediction [Sillon 94, Ward 94] Validation studies of these new programs have, to date, been of restricted value, one reason being that comparison against scale models measured in artificial skies are made using necessarily idealised sky brightness distributions [Selkowitz 82] Also, where illuminance predictions have been compared with measurements taken in real rooms under real sky conditions, the sky brightness distribution used by the program was based on a theoretical sky model generated from bulk values e g global and diffuse horizontal illuminance [Bellia 94] Differences between the real sky luminance distribution and that used in the program are not known It is therefore impossible to determine where the errors arise, in the basic algorithms or the representation of the sky

For this validation study, comparing predictions with illuminance data from full size rooms under real sky conditions, the simulation program uses model sky luminance patterns based directly on measured sky brightness distributions

The validation was carried out using a unique dataset of measurements taken at the Building Research Establishment (BRE), Garston, UK The measurements for the 754 entries in the validation set covered a range of naturally occurring skies, from heavily overcast, through intermediate to clear sky conditions The absolute accuracy of the direct light source calculation in *Radiance* was verified by others early in the development of the system [Grynberg 89] [Ward 91] So it is not in question here

2 The validation dataset

Long term measurements of the sky luminance distribution were carried out at the Building Research Establishment during 1992 and 1993 The sky monitoring was conducted as part of the BRE contribution to the International Daylight Measuring Programme (IDMP) Together with measurements of global horizontal, direct normal and four vertical illuminances, a sky scanning device was used to measure the sky luminance at 145 positions evenly distributed over the sky vault every fifteen minutes during daylight In conjunction with the sky monitoring programme, the BRE conducted an evaluation study of the light redistribution properties of five innovative glazing systems against standard clear glazing [Aizlewood 93] The sky monitoring apparatus were positioned on the roof directly above the experimental rooms Room illuminance and sky luminance measurements were recorded within seconds of each other

2 1 Internal conditions illuminance measurements

Two full-size mock offices with south-facing glazing were constructed adjacent to each other Room dimensions were almost identical, 9 metres deep, 3 metres wide and 2 7 metres high The rooms were left unfurnished, though the surface reflectances were chosen to correspond to a typical office The window of one office was adapted so that an innovative daylighting system could be installed, the other has conventional single glazing, Figure I-1a Six illuminance cells positioned at work plane height (0 7m), regularly spaced along the centre line of each room, were used to monitor the illuminance distribution in the room, Figure I 1b

2 2 External conditions monitoring the sky and sun

The instrument used to measure the sky brightness distribution was a PRC Krochmann sky scanner, Figure I-1c The sky scanner measured the sky luminance distribution every 15 minutes during daylight hours The scanner was configured to begin each sky scan at the solar azimuth position For each row of fixed altitude, measurements were taken as the scanner rotated anti-clockwise, i e $N \rightarrow W \rightarrow S \rightarrow E$ The measurement pattern, though regular, possessed therefore a rotation about the zenith axis which was different for each scan A scan consisted of 150 readings according to the pattern recommended by the CIE [Perez 91] and took 25 seconds to complete Of the 150 measurements taken, 145 were for unique positions on the sky vault (the zenith luminance was recorded 6 times during each scan) The scanner acceptance angle was 11 giving a sky coverage of ~68%, Figure I-1d The scanner did not measure the sky luminance at the position closest to the sun, and a scan could contain one or more occurrences of 'out of range' measurements It was therefore necessary to use interpolation to estimate for missing sky luminance values In addition to estimating the missing and out-of-range values, the sky luminance measurements were re-aligned to an anchored-grid pattern, Figure I-2

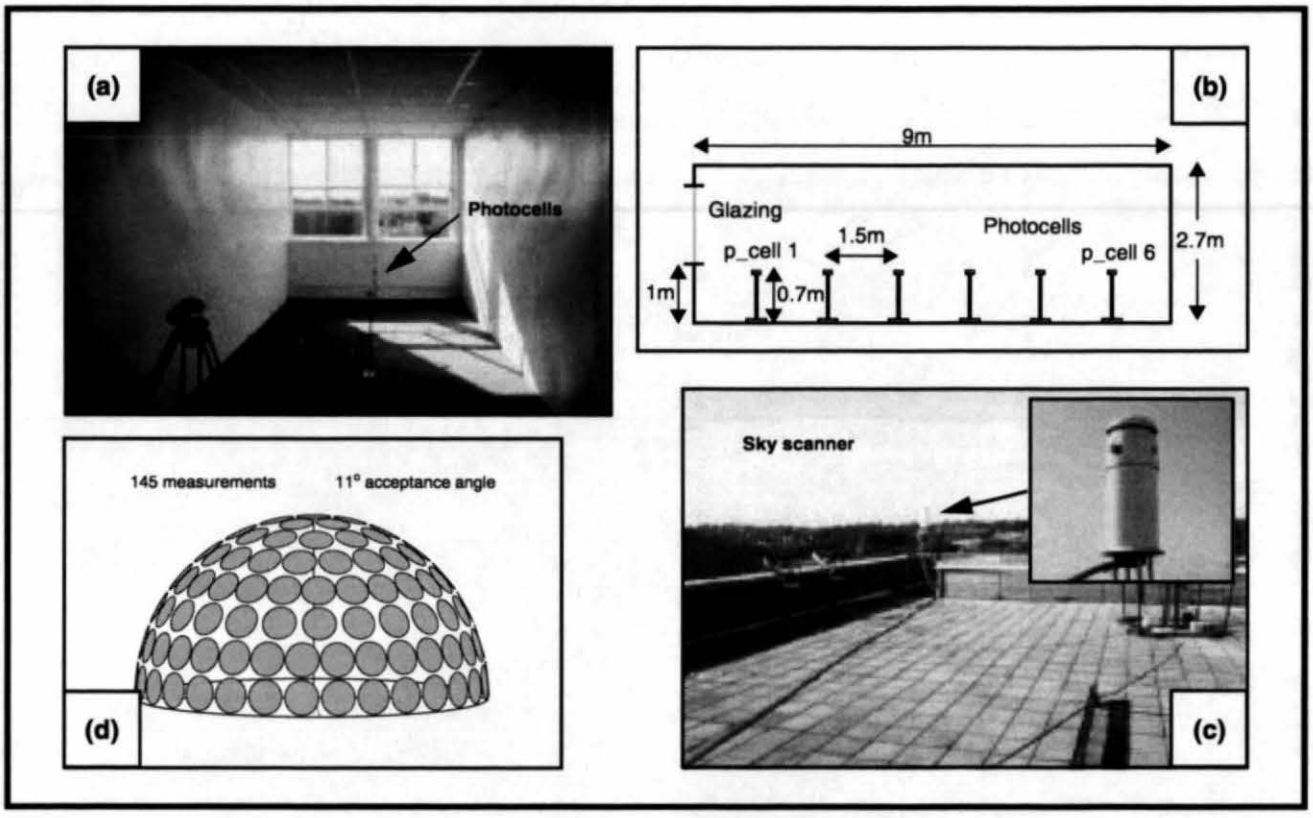


Figure I-1. Photograph (a) and figure (b) showing the office room. Photograph (with detail) of the sky scanner on the roof above the office (c) and graphic showing the scanner measurement pattern (d)

Monitored sky and room data were obtained for 754 skies covering 27 unique days in the year 1992, Figure I-3. Because the fixtures in the innovative glazings room were cycled throughout the monitoring period, the single glazing was exposed to the largest number of skies. Just how representative these 754 skies were of the full range of naturally occurring sky condition in the UK can be judged from Figure I-4. Here, the distribution in the sky clearness index for the validation dataset and for a standard test reference year (TRY) are compared. The TRY data were recorded at Kew which is close to the validation site. The TRY time-series contains hourly measurements of the diffuse sky irradiance and the direct normal solar irradiance for one year. The distribution in sky types for the validation dataset was broadly similar to that for the TRY. In the validation data, heavily overcast skies (bin 1) were somewhat over-represented whilst the very clearest skies were under-represented.

3 The Radiance model descriptions

3.1 The Experimental Room

Geometrically, the model room generated for the simulations was a very close representation of the experimental office. The dimensions of the clear glazed office room were measured to an accuracy of ~1cm and the room described in the model as a collection of rectangular polygons. Particular attention was paid to the window bars and glazing panes which were measured to an accuracy of ~0.2cm and modelled as discrete elements. The illuminance meters themselves are not modelled, rather the horizontal illuminance at that point was

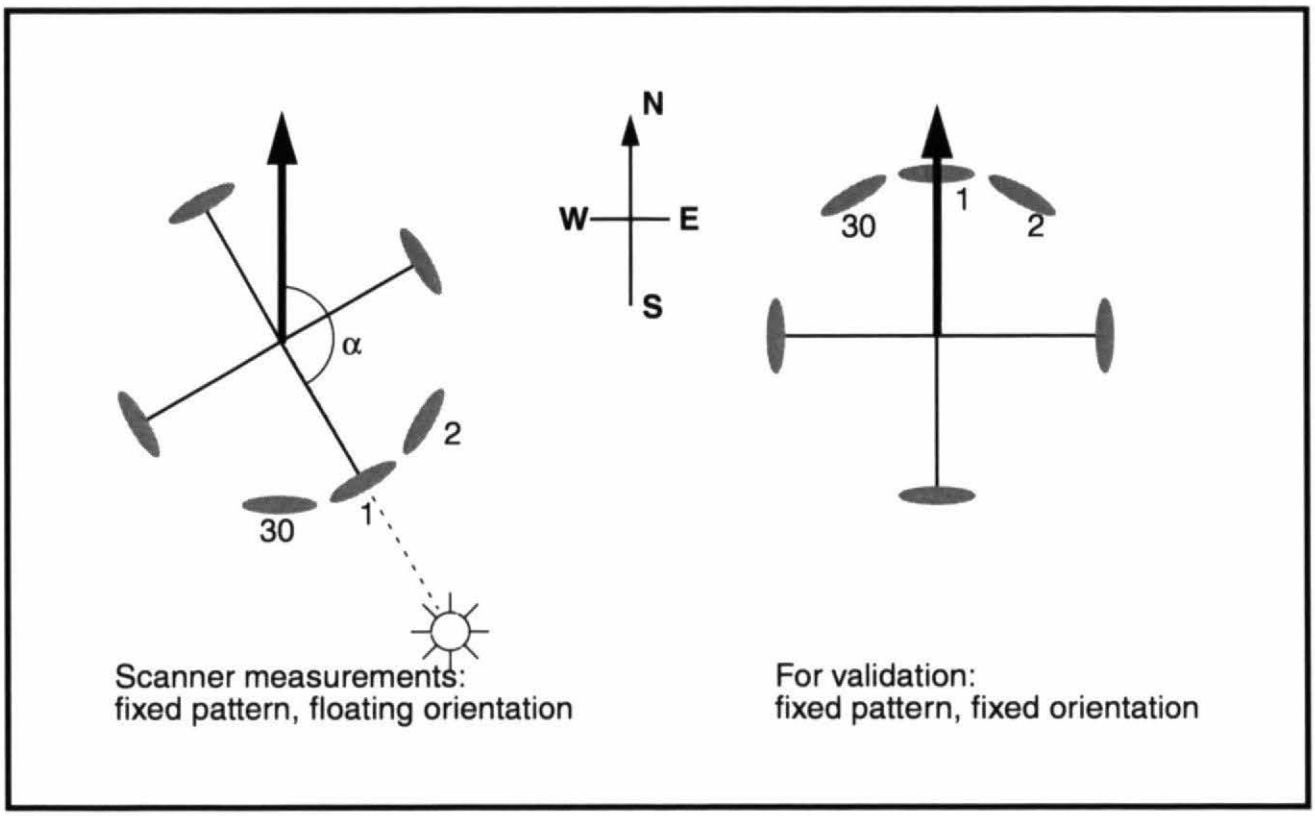


Figure I-2. Conversion of the measurement pattern and orientation for validation

calculated. All opaque surfaces were modelled in the first instance as gray diffuse reflectors. The reflectances used in the model were the average of the values measured at the beginning and end of the monitoring period: walls 0.83, ceiling 0.80 and carpet 0.095 [Aizlewood 93]. Window transmittance was that for standard single glazing. A glazing maintenance factor was incorporated into the transmittance and was based on the average of the range recommended by the experimenters. A circular ground plane of radius 30 metres, reflectivity 0.2 and centred on the room, was the only non-luminous external object.

3.2 Room with Innovative Glazing Fixture

The innovative glazing fixtures modelled for the companion study were internally mounted diffuse and specular (mirror) finish light shelves [Mardaljevic 95]. Both shelves were the same size: full room width, 1.00 metre deep and fixed at a height of 2.08m. The diffuse finish light shelf was coated with a paint similar to that used on the ceiling and so was a assigned a reflectivity of 0.80. The upper surface of the specular shelf, in reality a polished aluminium sheet, was modelled as a mirror having a reflectivity of 0.90. Some uncertainty exists here: specular light shelf reflectivity was not directly measured and the value used in the model was based on typical value for this material. Otherwise, this room was identical to that having clear glazing.

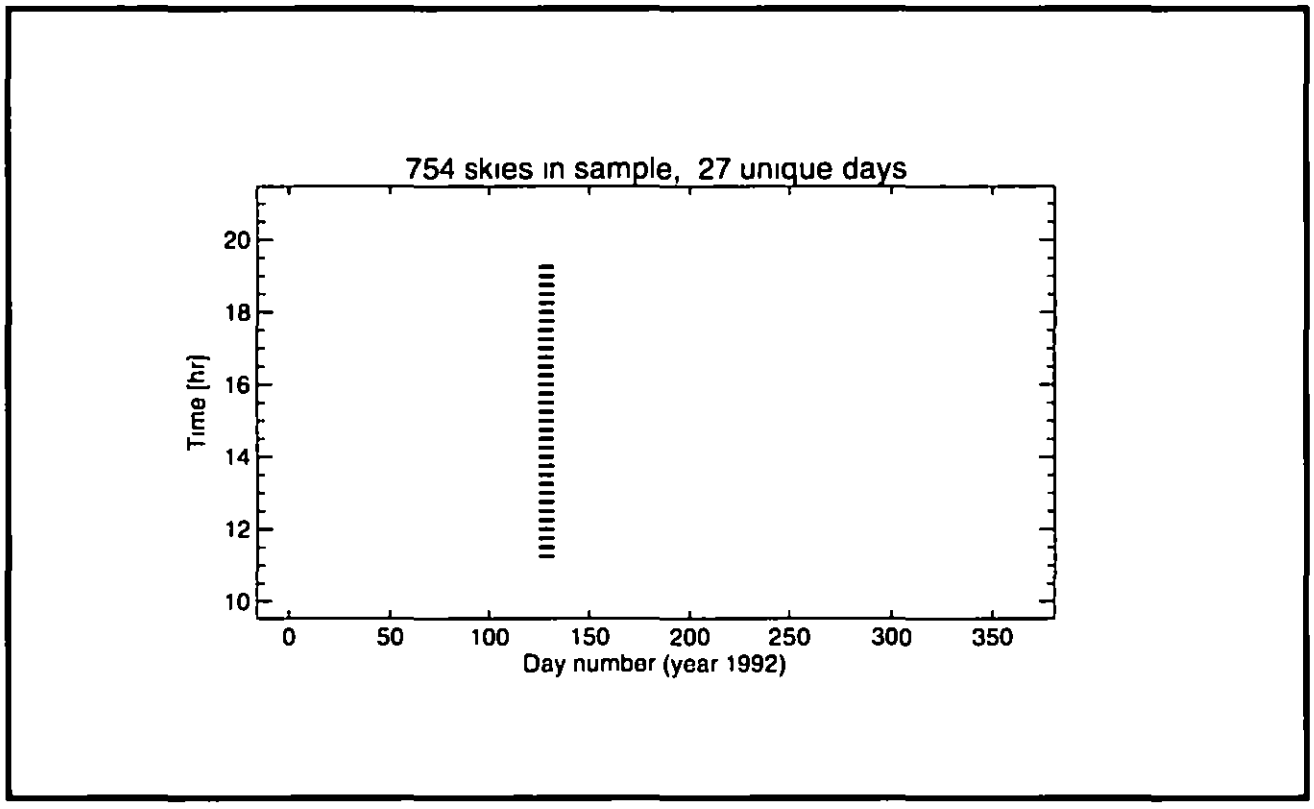


Figure I-3 Distribution of skies in 1992

3 3 The sun and sky representations

The solar disc was described as a source angle whose radiance was determined directly from measurements of the direct normal illuminance The processed sky luminance measurements were applied as pattern modifier to the usual *Radiance* sky glow using the **brightdata** pattern type

In the course of this validation exercise it was discovered that large relative errors (|RERI > 50%) in the illuminance predictions were most likely to occur when the patch of sky about the sun position was visible from the photocell location This was believed to arise due to the uncertainty in the sky brightness distribution about the solar position - which was not measured by the scanner

The absolute accuracy of a *Radiance* prediction (illuminance or luminance) for a real scene will depend on both the faithfulness of the model description - materials and geometry - and the resolution of the simulation parameters It is important to try to distinguish between errors in prediction which are due to incomplete or inaccurate model representation, and those errors which can be directly attributed to the algorithms used in the simulation, since the former can often overwhelm the latter A robust method to achieve this was devised, whereby each sky-scan and photocell combination in the validation dataset was classed as either reliable or potentially unreliable data

Sky scan and photocell combinations which were classed as potentially unreliable were identified as *those instances where any part of a 6° disc centred on the sun position was 'visible'*

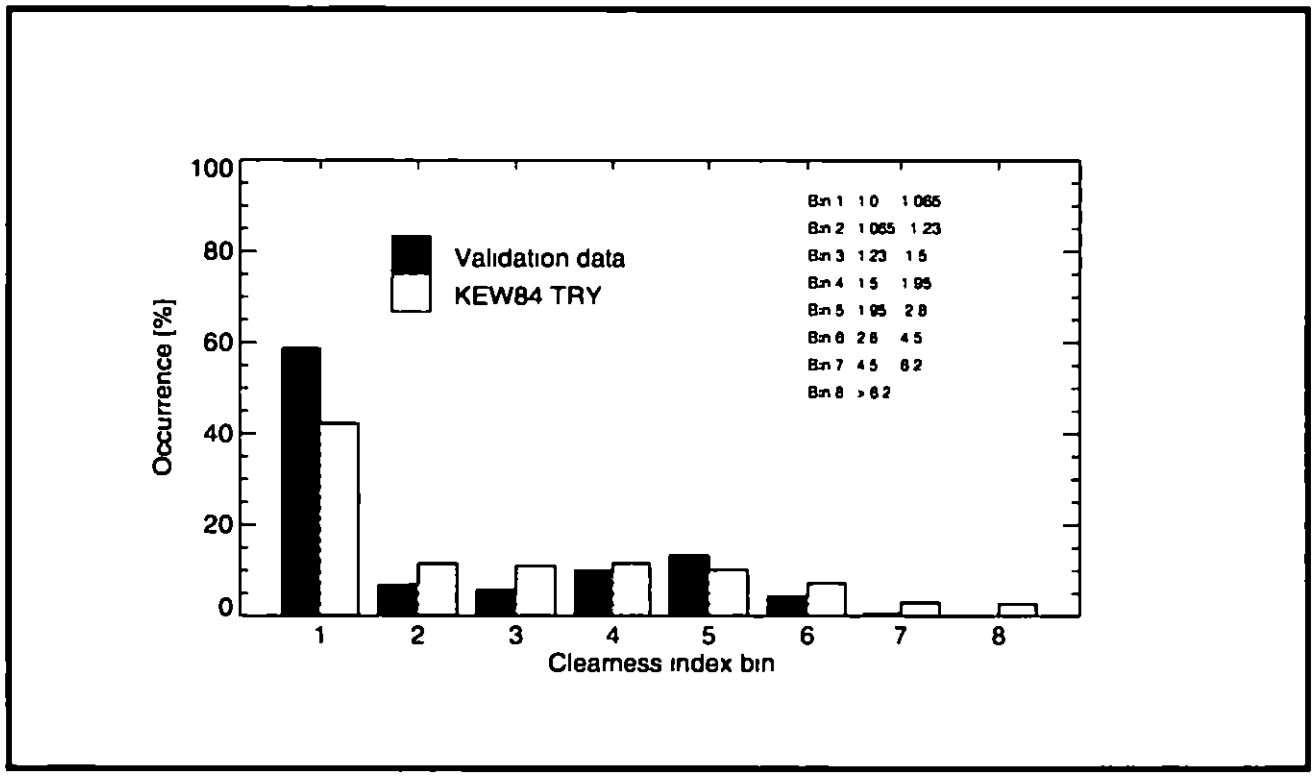


Figure I-4 Distribution in clearness index compared to TRY

from the photocell A 6 opening angle for the disc was employed because this was equal to the acceptance angle of the tracking photocell that measured the direct solar illuminance This is because the, potentially very large, luminance gradients about the solar position were not measured by the sky scanner Illuminance predictions from these cases were then eliminated from the analysis of the results The overall error characteristics were therefore evaluated using a subset of the total number of predictions Occurrences of a photocell 'seeing' any part of the 6 solar disc were of course more likely at the front of the office space (that is, near the window) than at the back The number of predictions included in the error analysis were therefore different for each photocell, e g N_{scan} = 357 for photocell 1 and N_{scan} = 724 for photocell 6 Note that by excluding the instances where the 6° solar disc was visible, the validation sample is now biased towards those instances where inter-reflected light - rather than direct - was the dominant component Inter-reflected light is the most difficult component of illumination to predict direct light from the sun or sky is easy to model given a sufficiently accurate description of the sky/sun luminance magnitude and distribution

Visibility testing was achieved by re-running the simulations for the direct sun component, but in place of the normal sun specification, a 6 source solid angle was used Instead of carrying out an illuminance calculation at the photocell location, a 'bundle' of rays in a 6 cone were aimed from this position towards the centre of the 6° disc A greater than zero luminance for a returned ray indicated that the ray intersected with the 6 disc which was the only luminous material in the scene This testing was carried out for all six photocells and for each of the 754 sun positions

For comparison, illuminance predictions using the potentially unreliable photocell-scan combinations are presented also

4 Comparison of Model Predictions with Measurements

4 1 Individual Skies

Detailed comparisons for a limited number of skies have been presented in a previous paper [Mardaljevic 95] Two of these are reproduced here The first, case 102_92_13h00 (day_year_time), is for the clear glazed room illuminated by an intermediate sky in Spring, Figure I-5 This shows surface and aligned perspective contour plots of the sky luminance distribution after processing for input to the model, together with plots of log illuminance (measured and predicted) vs distance and the relative error for the predictions The relative error (RER) is defined as

$$RER = \left(\frac{I_{predicted} - I_{measured}}{I_{measured}} \right) \times 100$$

The largest relative error for this sky is +12 1% (at photocell 1), for the other five photocells the predictions are within ±6%

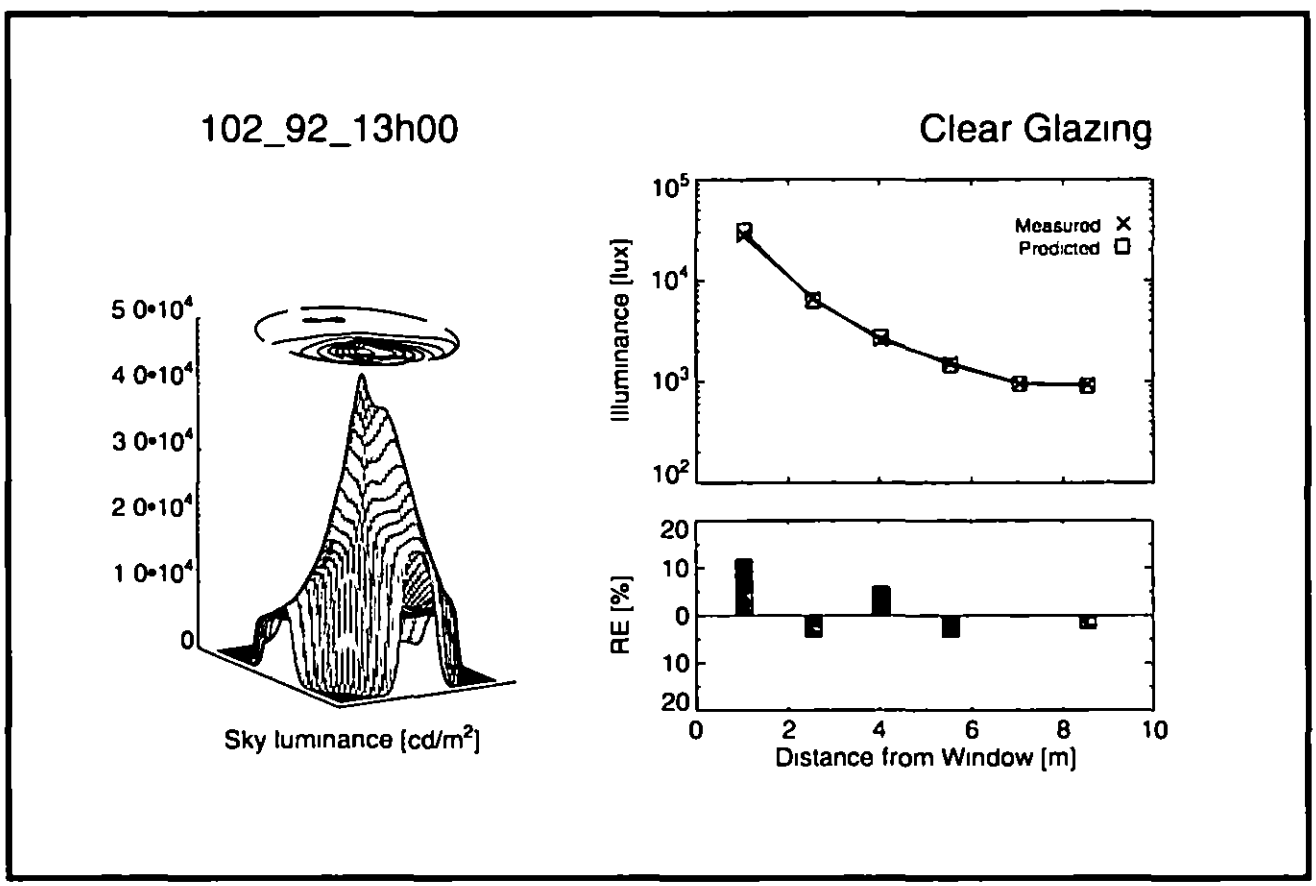


Figure I-5 Individual skies comparison clear glazing

The second example is for case 318_92_12h00 where the innovative glazings room was fitted with a mirror light shelf and the scene was illuminated by a clear sky The surface/contour plots show a characteristic clear sky pattern with horizon brightening and the (Winter) sun at low altitude The illuminance predictions for this case were good also the mean of the absolute values of the relative errors was 8 5%

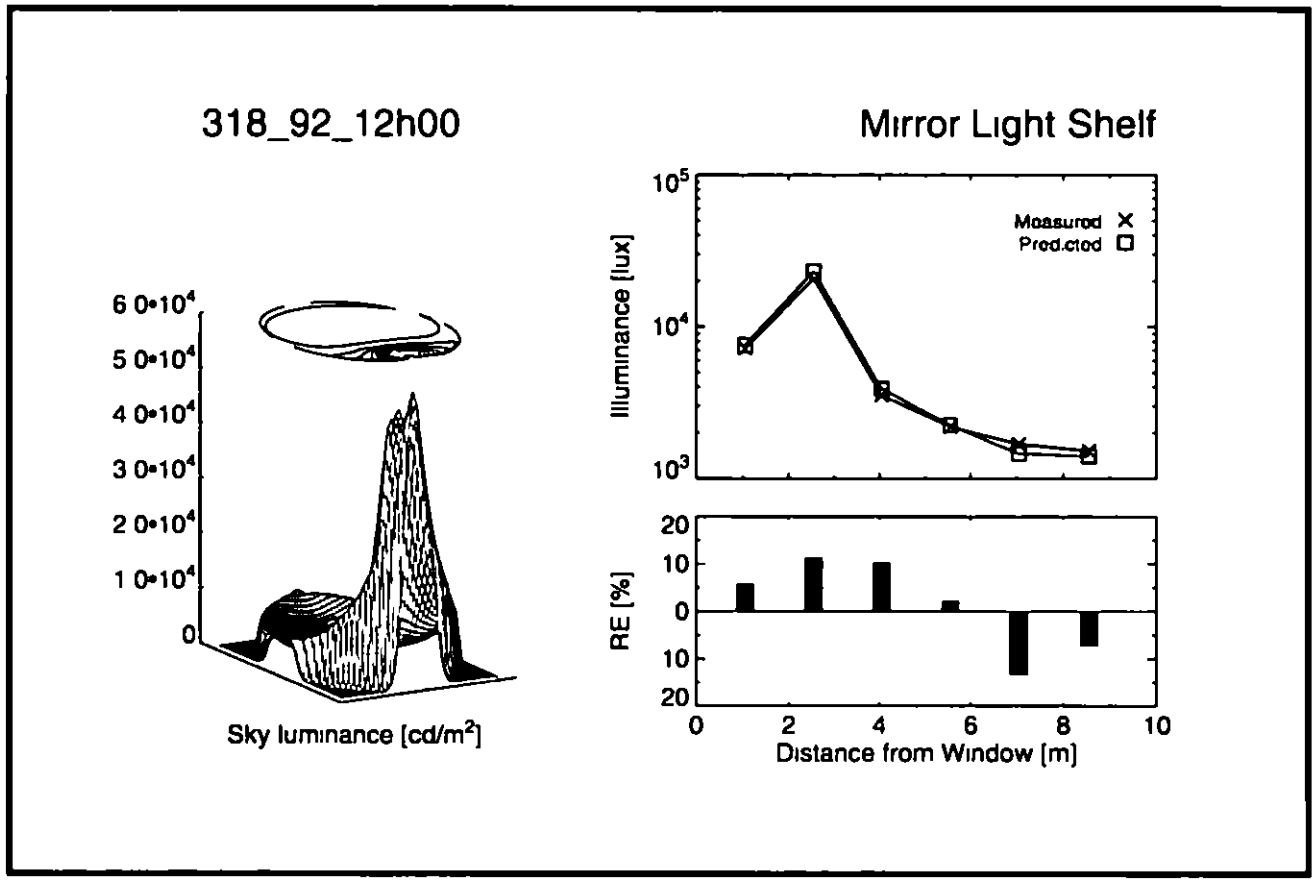


Figure I-6 Individual skies comparison - mirror light shelf

4 2 All Skies

The illuminance predictions at each of the six photocells for the 754 skies were partitioned into sets designated as either 'reliable' or 'potentially unreliable' depending on the visibility of the circumsolar region from each of the photocell positions The relative error in the predictions for each of the sets were collated into frequency distribution histograms The RERs for the 'reliable' and the 'potentially unreliable' sets were aggregated into frequency distribution histograms The RER binsize was 5% and the number in each distribution was normalised Each histogram is annotated with the photocell number, the number of predictions in the sample, the overall mean bias error (MBE) and the root mean square error (RMSE)

$$MBE = 100\% \sum \left[\frac{E_{pred} - E_{meas}}{E_{meas}} \right] / N$$

$$RMSE = 100\% \sqrt{\sum \left[\frac{E_{pred} - E_{meas}}{E_{meas}} \right]^2 / N}$$

Where E_{pred} and E_{meas} were, respectively, the predicted and measured illuminances, and N was the number of predictions (i e scans)

Considering first the predictions from the 'reliable photocell-scan combinations, Figure I-7

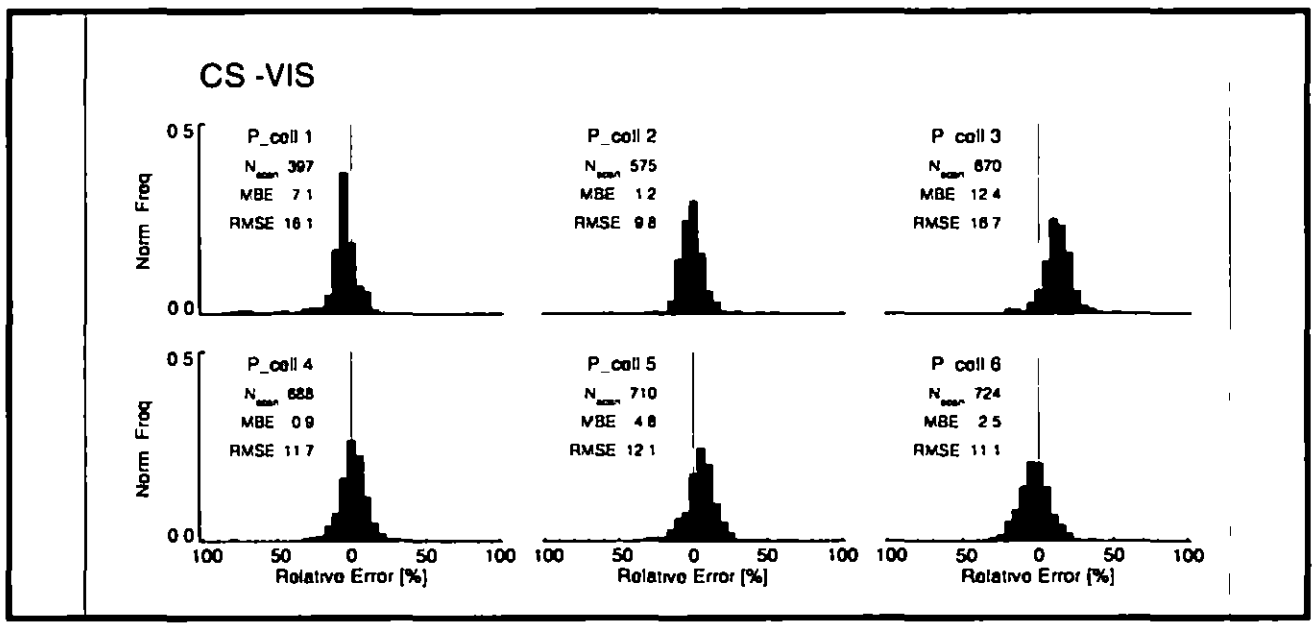


Figure I-7 Predictions for 'reliable photocell-scan combinations

Here, each of the distributions, with the exception of pcell 3, appears fairly symmetric about the 0% line, and the main body of the distribution is contained within the range +-17 5% From pcell 1 at the front of the room to pcell 6 at the rear, there is a reduction in the kurtosis (or 'peakiness'), of the distribution For all pcells, with the exception of number 3, the MBE is very low, and the RMSEs are never greater than 17%

The predictions from the 'potentially unreliable photocell-scan combinations are very different, Figure I-8 Note that not only are the MBEs much larger than for the 'reliable' data, but they are all positive This is because over prediction can give (positive) RERs > 100%, but the RER limit for under prediction is -100% Significant over prediction in illuminance can occur when a photocell is predicted to be in sun when in reality it was in shade The smallest of differences in geometry between the model and the actual room could cause this The small inset histogram for each photocell shows the distribution in the fraction of the 6° disc that was visible for each sample For example, near the back of the room (pcell 6) the photocell never 'saw more than about half of the disc

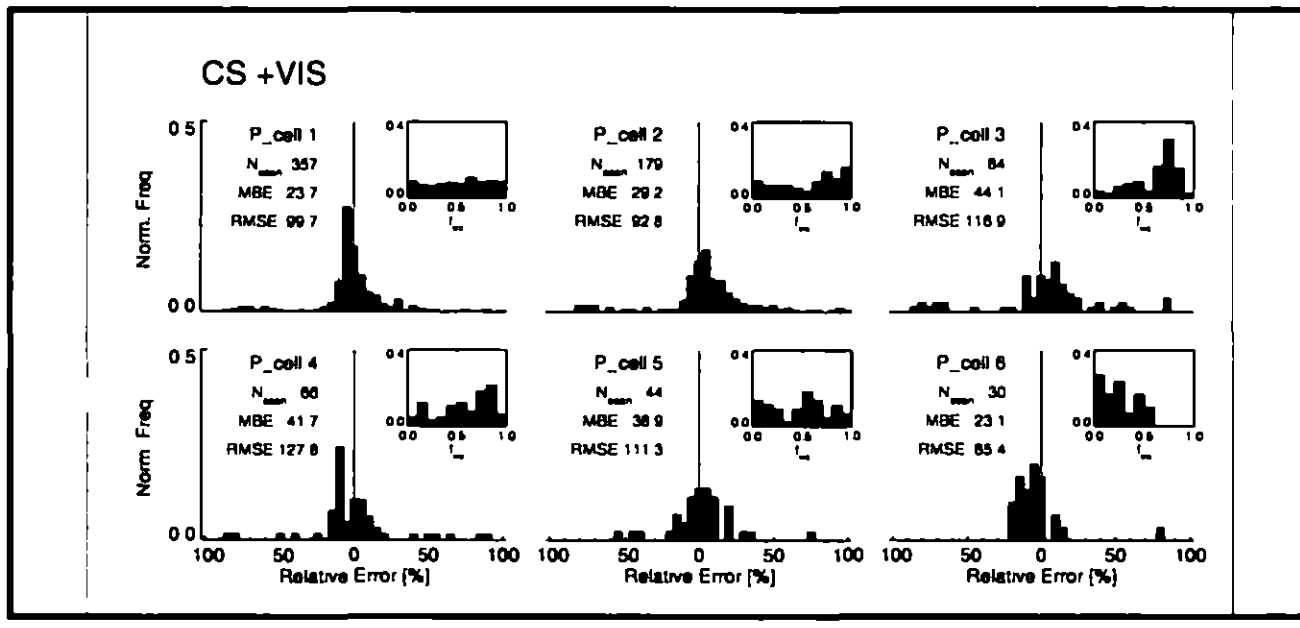


Figure I-8 Predictions for potentially unreliable' photocell-scan combinations

It is clear from the distributions in Figure I-8 that many accurate illuminance predictions are nevertheless classed by the visibility criteria as 'potentially unreliable' Might it be possible to include the most overcast skies where large luminance gradients about the solar position are unlikely - as 'reliable' even though the (dull) circumsolar region was visible to the photocell? A test of this hypothesis is to further partition the data according to sky clearness index bin, and then determine for each set the overall MBE and RMSE The results for this test are shown in Figure I-9 For both MBE and RMSE, the accuracy from the 'reliable' set is always better than

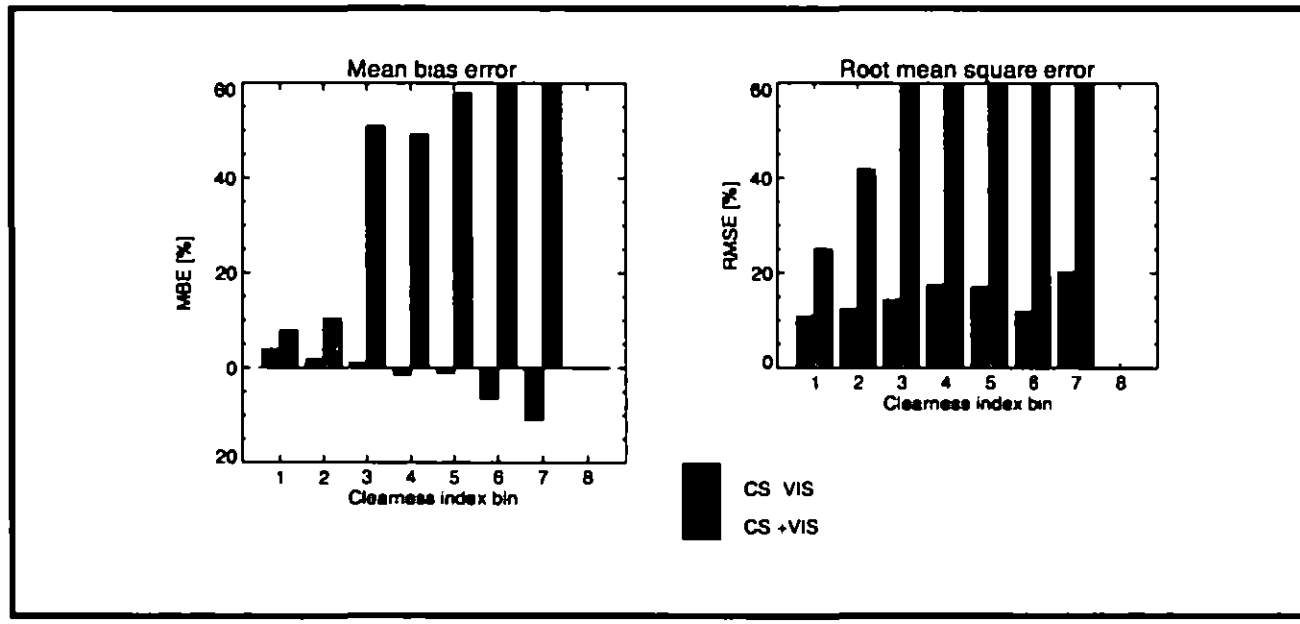


Figure I-9 MBE and RMSE stratified by clearness index

from the 'potentially unreliable' As might be expected, the difference is less for the overcast skies (bin 1), but it is nevertheless significant

Finally, the RER at each photocell is plotted together with the time-series of global horizontal, diffuse horizontal and vertical South illuminances A pair of plots are given for each of the 27 days They are grouped together in Figure I-10 to Figure I 13 Here the relative error at each photocell is marked by a shaded square at the time of the measurement The 'reliable photocell scan combinations are shaded magenta (■) and the 'potentially unreliable' combinations are shaded cyan (■) The illuminance predictions were made every 15 minutes, which was the sampling frequency of the sky scanner The three external illuminances values however are plotted at 5 minute intervals, which was the sampling frequency at which these data were obtained It is apparent from some of the plots (129_92 and 273_92) that the occasional poor accuracy from 'reliable' data might be related to rapidly varying sky conditions

5 Conclusion

The results presented here show that the *Radiance* lighting simulation system can predict internal illuminance to high degree of accuracy for a large sample of skies which cover a wide range of naturally occurring sky conditions As far as the author is aware, this validation study is the only one to date that has made use of measured sky brightness distributions and simultaneous internal illuminance measurements

A hypothesis concerning potently unreliable entries in the validation data was presented and verified

Acknowledgments

Sky luminance and room data were supplied by Paul Littlefair and Maurice Aizlewood of the UK Building Research Establishment Greg Ward of the Lawrence Berkeley Laboratory, USA advised on the conversion of sky luminance data to *Radiance* format

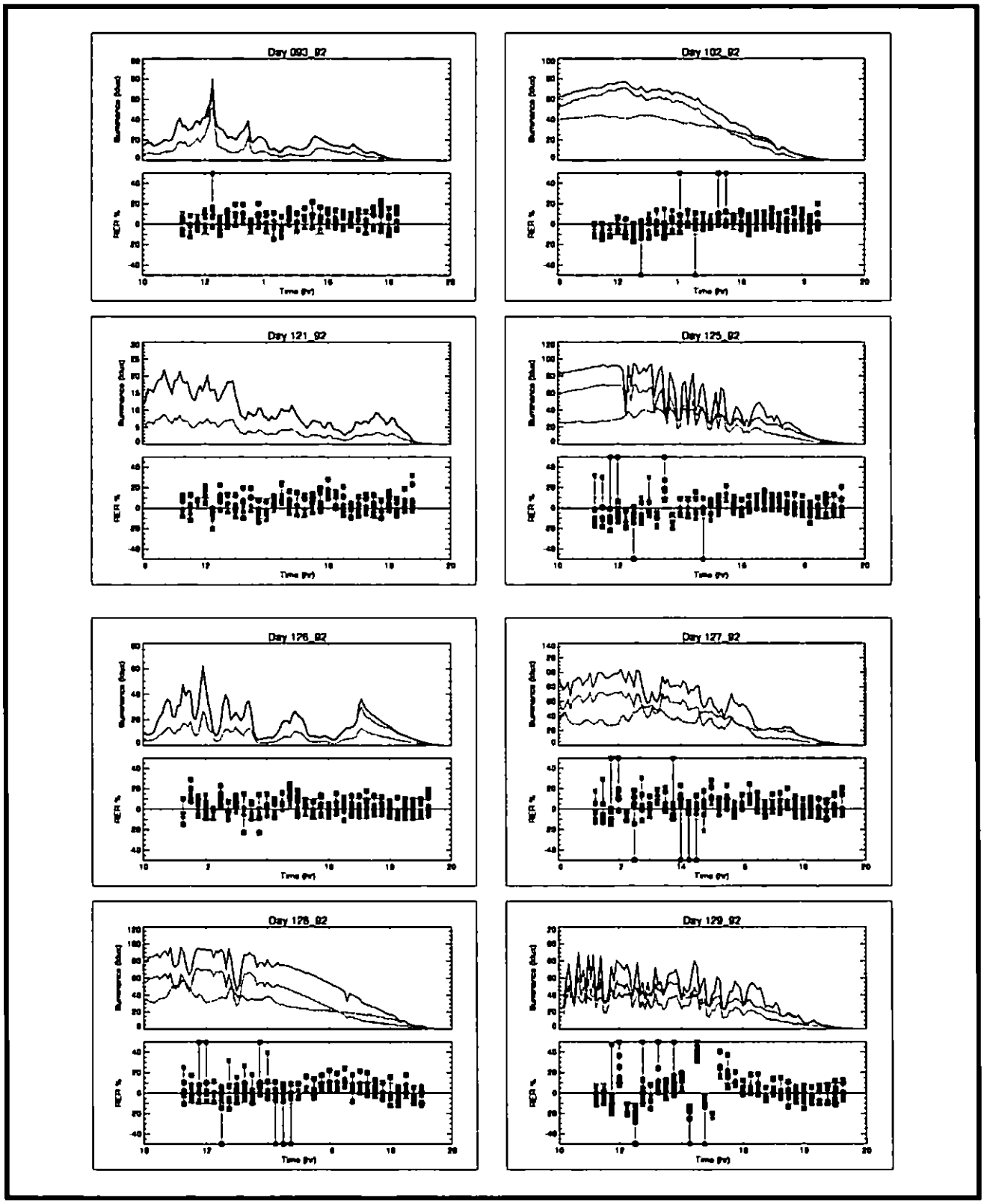


Figure I-10 RER time-series 093_92 to 129_92

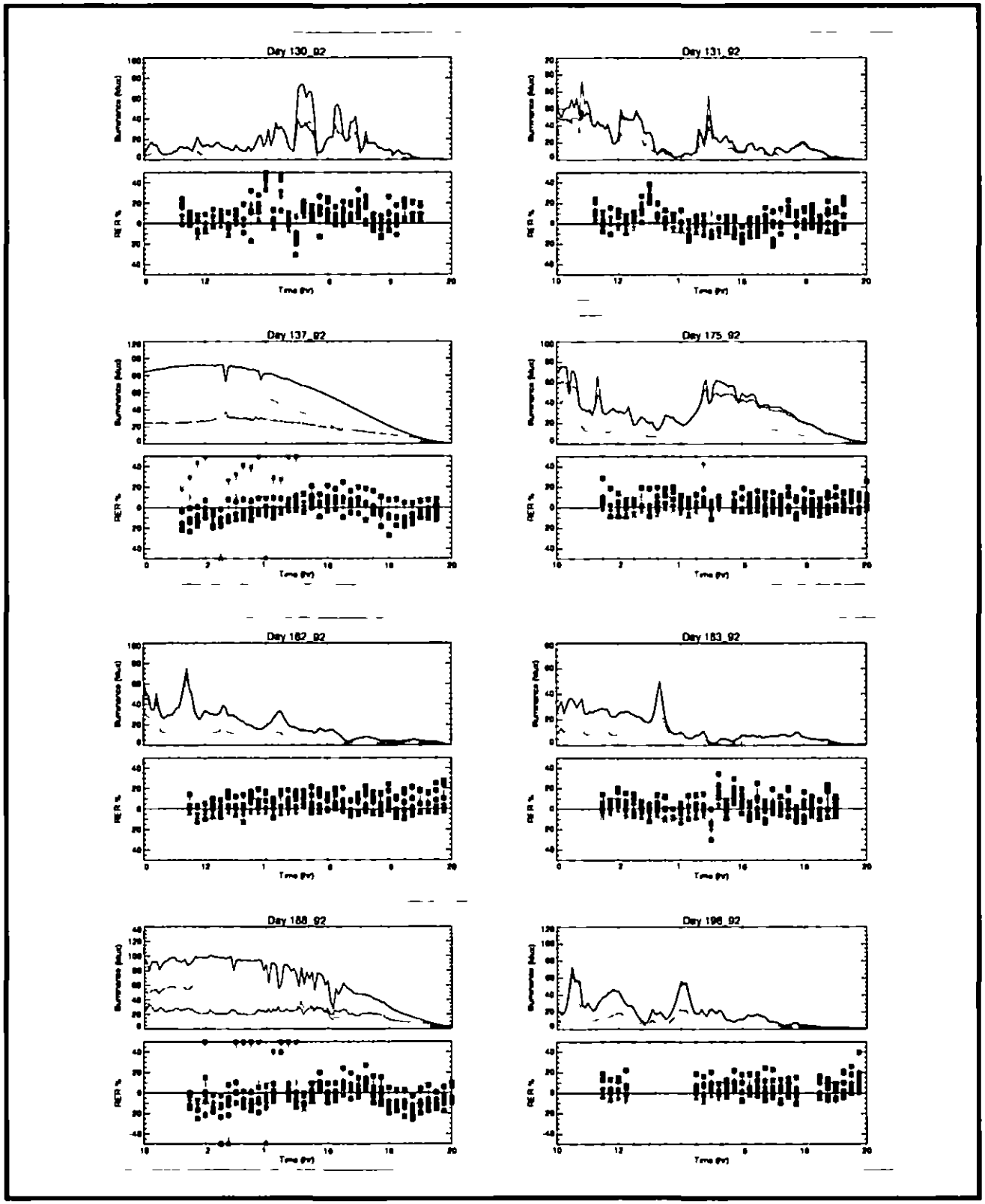


Figure I-11 RER time-series 130_92 to 196_92

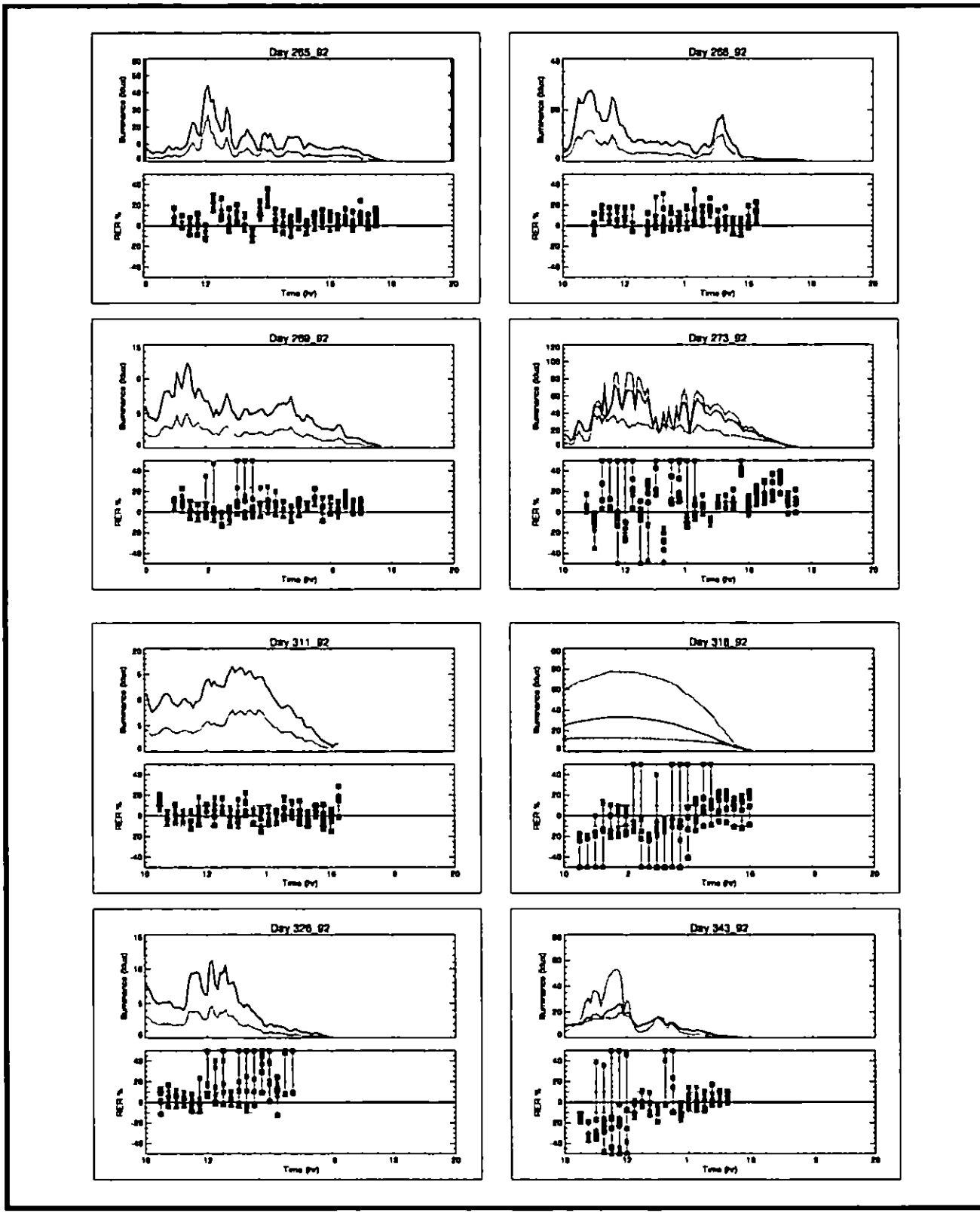


Figure I-12 RER time-series 265_92 to 343_92

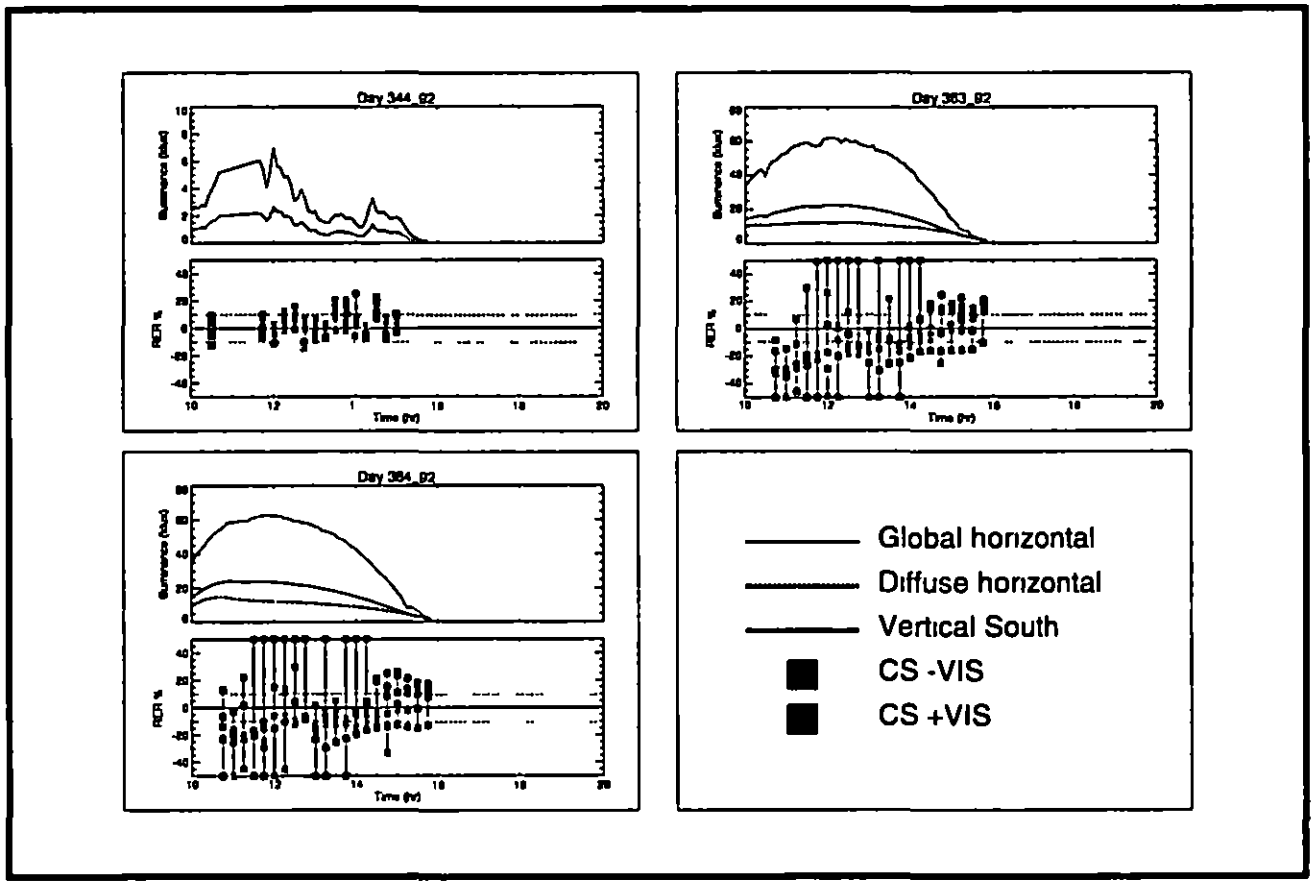


Figure I-13 RER time-series 344_92 to 364_92 and legend

References

[Aizlewood 93] Aizlewood M E *Innovative daylighting systems An experimental evaluation* Lighting Res Technol 24 (4) 141-152 (1993)

[Bellia 94] Bellia L, Cesarano A and Sibilio S *Daylighting contribution to interior lighting Experimental verification of software simulation results* Lighting Res Technol 26 (2) 99-105 (1994)

[Grynberg 89] Grynberg, Anat, *Validation of Radiance*, LBID 1575, LBL Technical Information Department, Lawrence Berkeley Laboratory, Berkeley, California, July 1989

[Mardaljevic 95] Mardaljevic, J *Validation of a lighting simulation program under real sky conditions* Lighting Res Technol 27(4) 181-188 (1995)

[Perez 91] Perez R and Kendrick J D (eds) *Guide to recommended daylight measurement* CIE TC 3-07, 7th draft (1991)

[Selkowitz 82] Selkowitz S, Kim J-J, Navvab M and Winkelmann F *The DOE-2 and Superlite daylighting programs* Lawrence Berkeley Laboratory Report 14569 (1982)

[Sillon 94] Sillon F X and Puech C *Radiosity and global illumination* (USA Morgan Kaufmann) (1994)

[Ward 91] Ward, Gregory, *Adaptive Shadow Testing for Ray Tracing* Second EUROGRAPHICS Workshop on Rendering, Barcelona, Spain, April 1991

[Ward 94] Ward G J *The **Radiance** lighting simulation and rendering system* Computer Graphics, Proceedings, Annual Conference Series, 459-472 (1994)

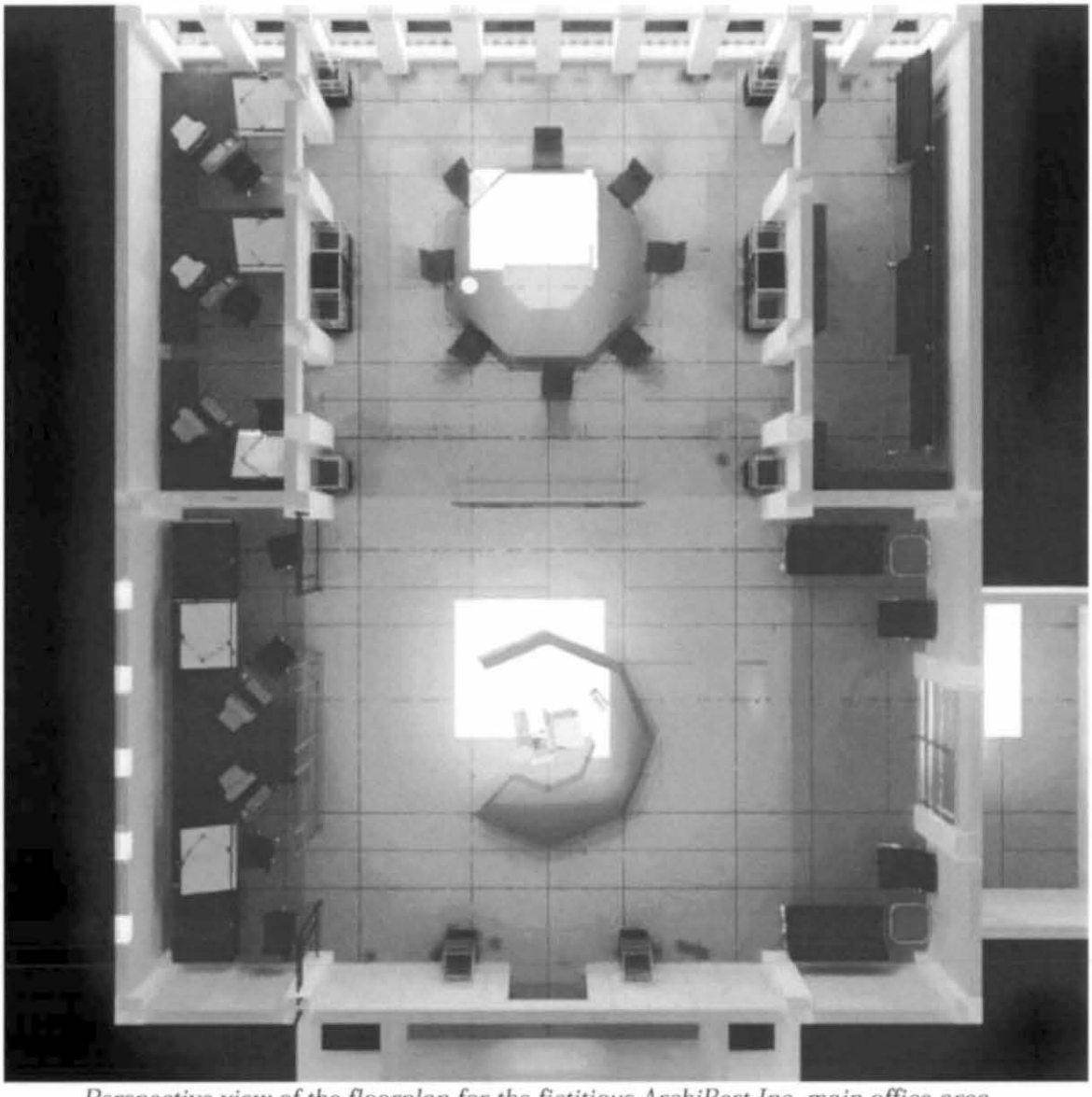
Working Through an Example Design Problem

Moderator:

Greg Ward Larsen

Presenters:

Chas Ehrlish
John Mardaljevic
Rob Shakespeare



Perspective view of the floorplan for the fictitious ArchiBest Inc. main office area. .

The fictitious ArchiBest Inc. architects offices are comprised of a reception area, conference room, two design studio areas and a resource room located in the top floor of an 6 story building located near Indianapolis. The entrance is via an interior elevator lobby to the south east. Private offices,

outside of the scope of this project, are accessed through the arch on the south wall The corporate image of the company is upscale, inventive and trend setting

Our task is to provide a lighting design for this environment which expresses the corporate image, optimizes the visual tasks and to make recommendations concerning the integration of daylight Radiance and its many resources will be featured as the basic "means of communication" as the participants present, then merge their individual design concepts into a unified approach

The design charette' has the following roles

- Rob explores the visual impact of the reception/conference areas and presents several designs which reflect the corporate upscale and inventive image of the client

- Chas explores the tasks which are performed in the various spaces identifies the challenges and develops a responsible lighting design to optimize visual task productivity

- John explores the ramifications of daylight and its impact on tasks performed throughout the space and in relation to electric lighting systems

- Greg mediator, time keeper, crowd control heckler Responsible for eliciting the atmosphere of the "design charette'

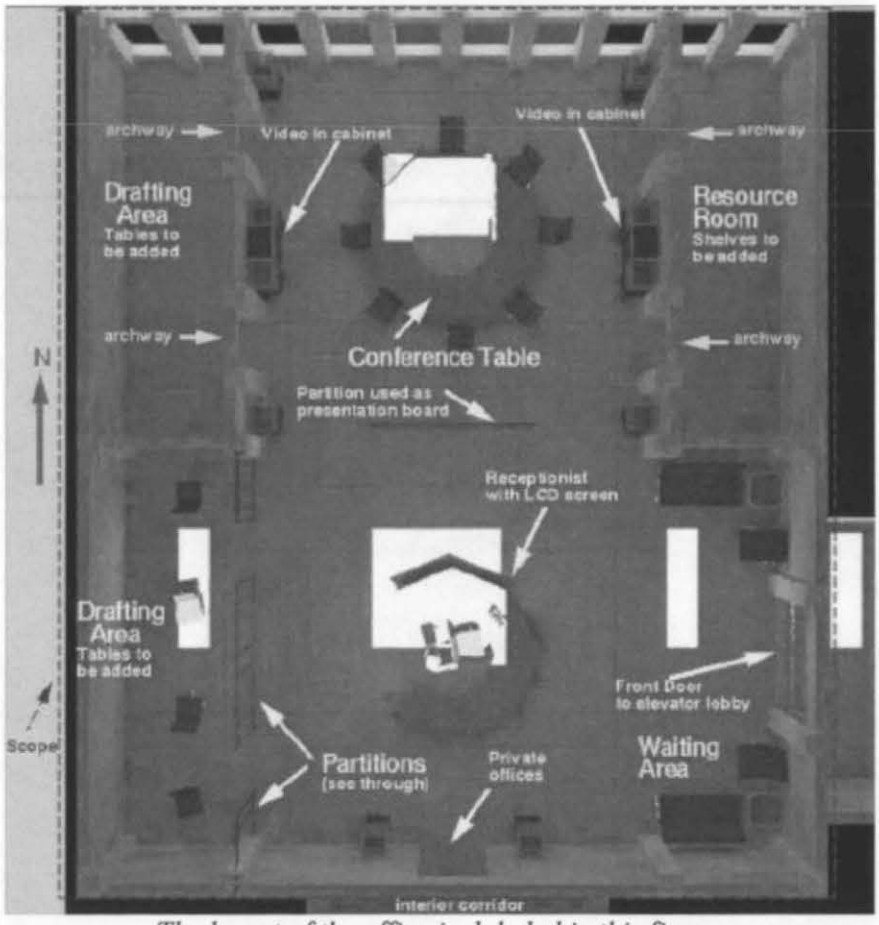
The presentation

Greg will play master of trouble, timekeeper and challenge/expand on our *Radiance* techniques Also as the principle design reviewer, he might encourage the audience to "blurt" out opinions to keep the participants on their toes

Each participant will have 5 minutes to present their own studies early designs, and any special tools or techniques which aided the concept/analysis process

During the final 15 minutes of the demonstration the participants will interact with each other and show some prerendered examples, early compromise solutions and other materials to demonstrate how they compromised their ideas to come up with a unified and strong design

The final few minutes will open to a discussion on collaborating with *Radiance* pictures The panelists will have observed how they engaged each other communicated through *Radiance* and will comment on what they learned from each other

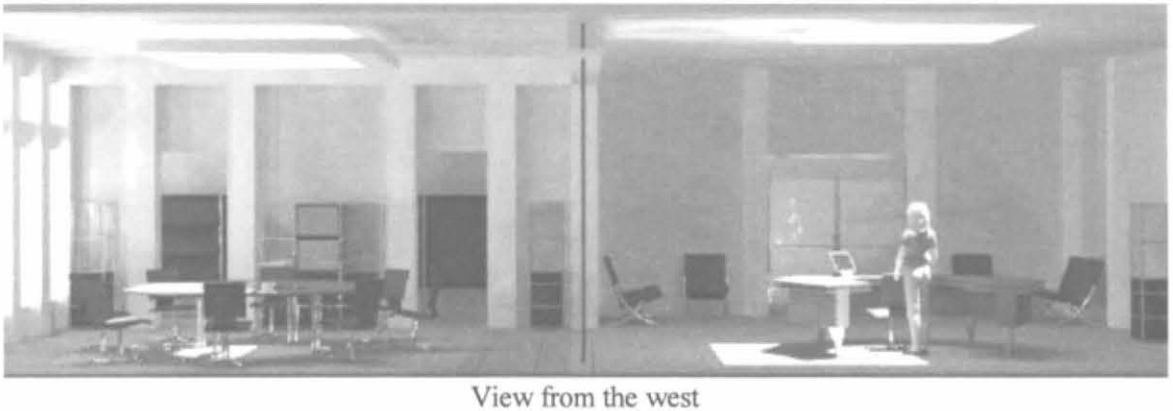


The layout of the office is labeled in this figure.

View from the south, looking north.

View from the east.

View from the north. Resource room on the left, conference area center and drafting/CAD on the right. Note the clear partition between the conference room and reception.



View from the west

K. 1 Example Design Problem: Daylight

John Mardaljevic

Daylight evaluation

The daylighting performance of the design will be evaluated in terms of basic daylight provision and solar penetration The standard daylight factor technique will be used to assess the level and uniformity of daylight in the space A sequence of renderings will used to demonstrate where and when solar penetration will occur

Recommendations for modifications to the design will be based on the outcome of these and the complementary studies

Example Design Problem

Focus on Electric Lighting Design/Analysis

The office contains the following space types

- reception area
- manual drafting room
- CAD drafting room
- library / study
- conference room

Example Design Problem

Occupants are assumed to be of all ages (20 to 70)

The following workplane illuminances have been chosen

reception area	750 lux
manual drafting room	1000 lux
CAD drafting room	100 lux
library / study	200 lux
conference room	500 lux
display cases	1000 lux

Example Design Problem

To meet target illuminances and to provide aesthetic variety, the following types of lighting have been chosen

reception area	direct/indirect pendant, task
manual drafting room	direct/indirect pendant
CAD drafting room	indirect, wall washer, task
library / study	recessed cans, wall washer
conference room	indirect pendants, dimmable spot
display cases	dimmable low voltage spot

Example Design Problem

The next question to answer is, which luminaires, how many luminaires, and where to locate them

In addition, the switching of these luminaires should take sources of daylight illumination into consideration

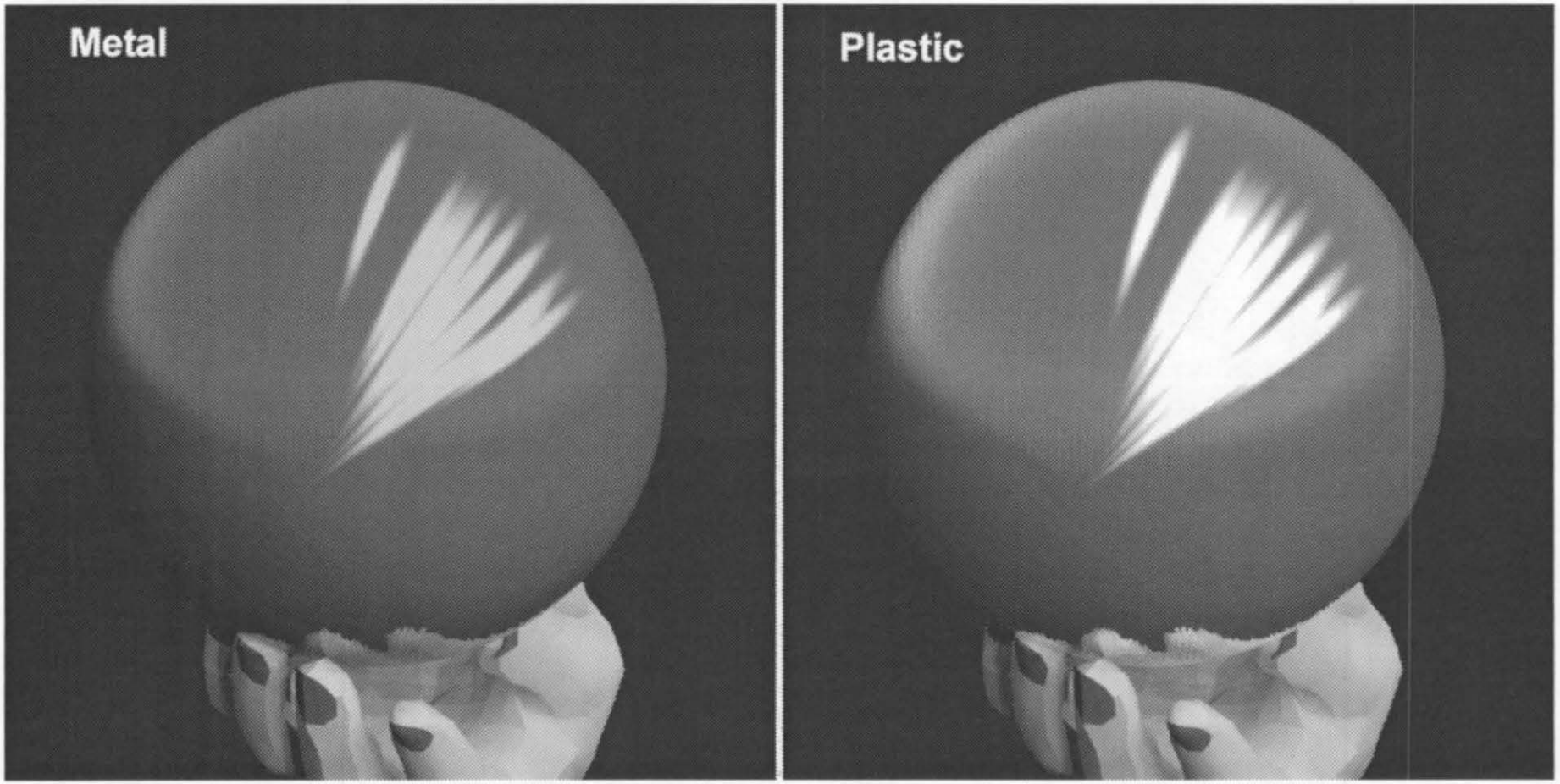
- group luminaires into zones by distance from window
- provide photosensor and occupancy sensor controls

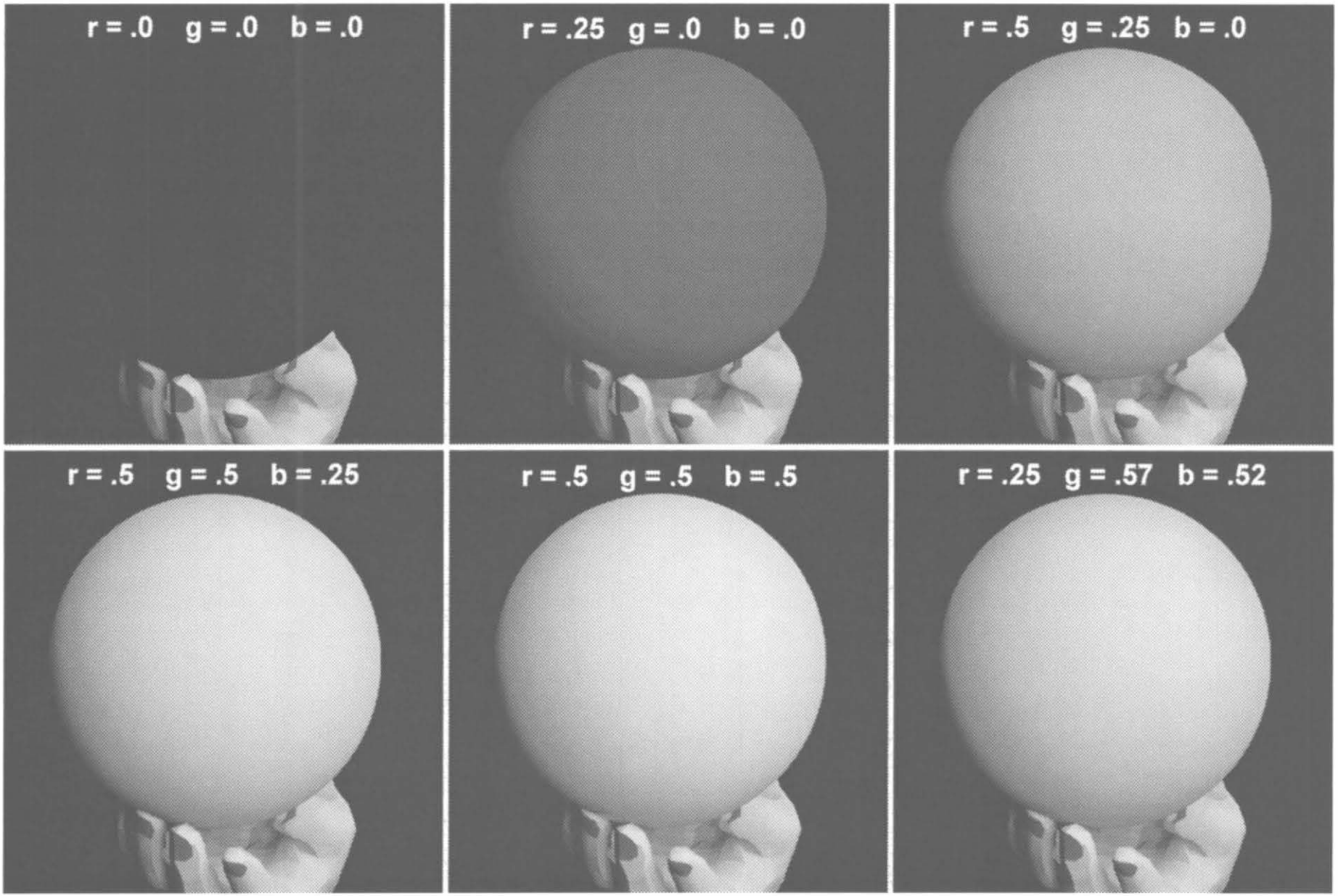
Example Design Problem

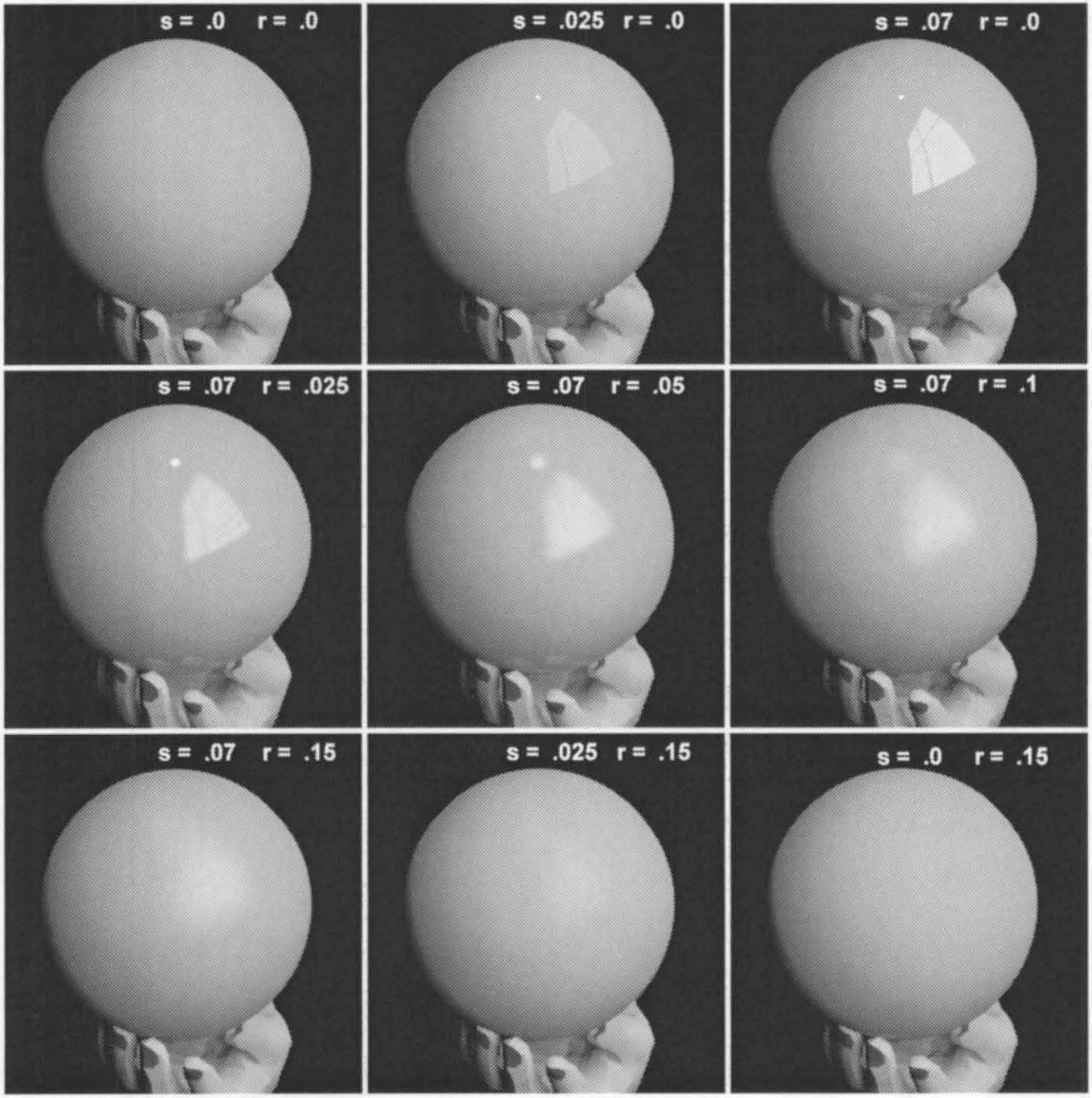
Finally, the lighting designer should review the interior designer's selection of surface materials and colors to assure that the type of light source is compatible, and to find potential synergies between the materials, electric light sources, and other sources of natural illumination

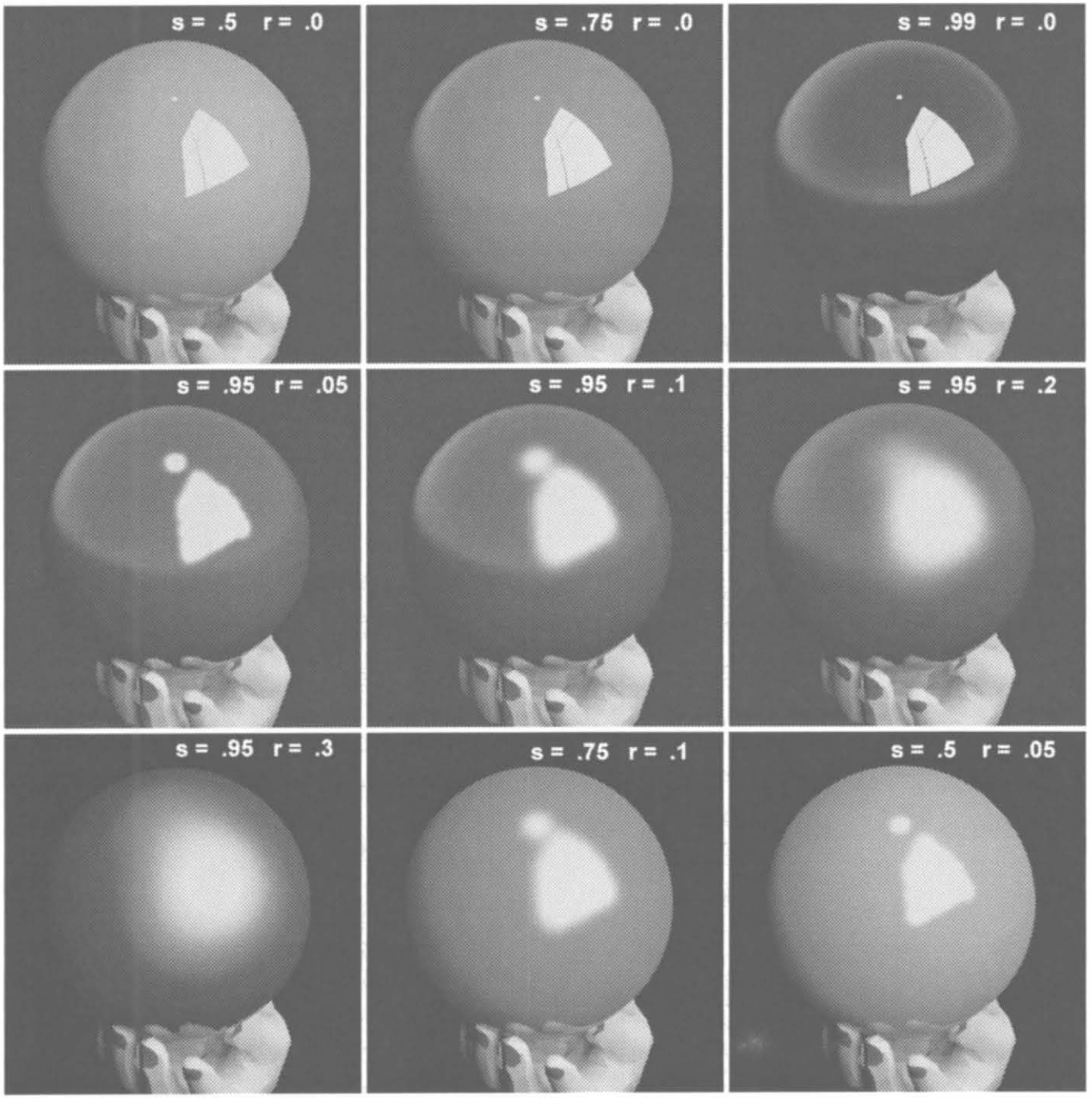
light colors around windows to bring light in and to reduce contrast

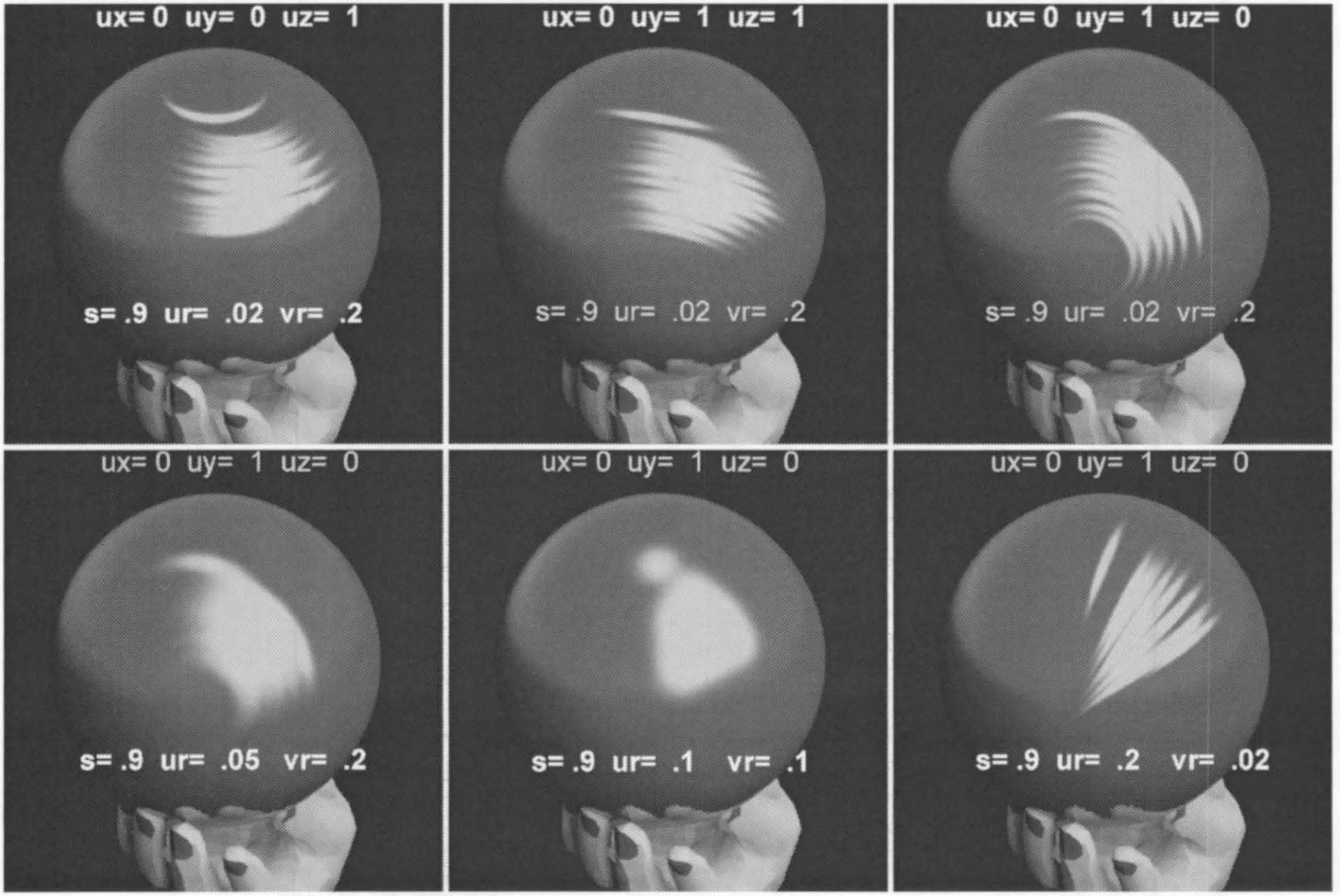
- appropriate use of window shades to provide control over excessive daylight levels

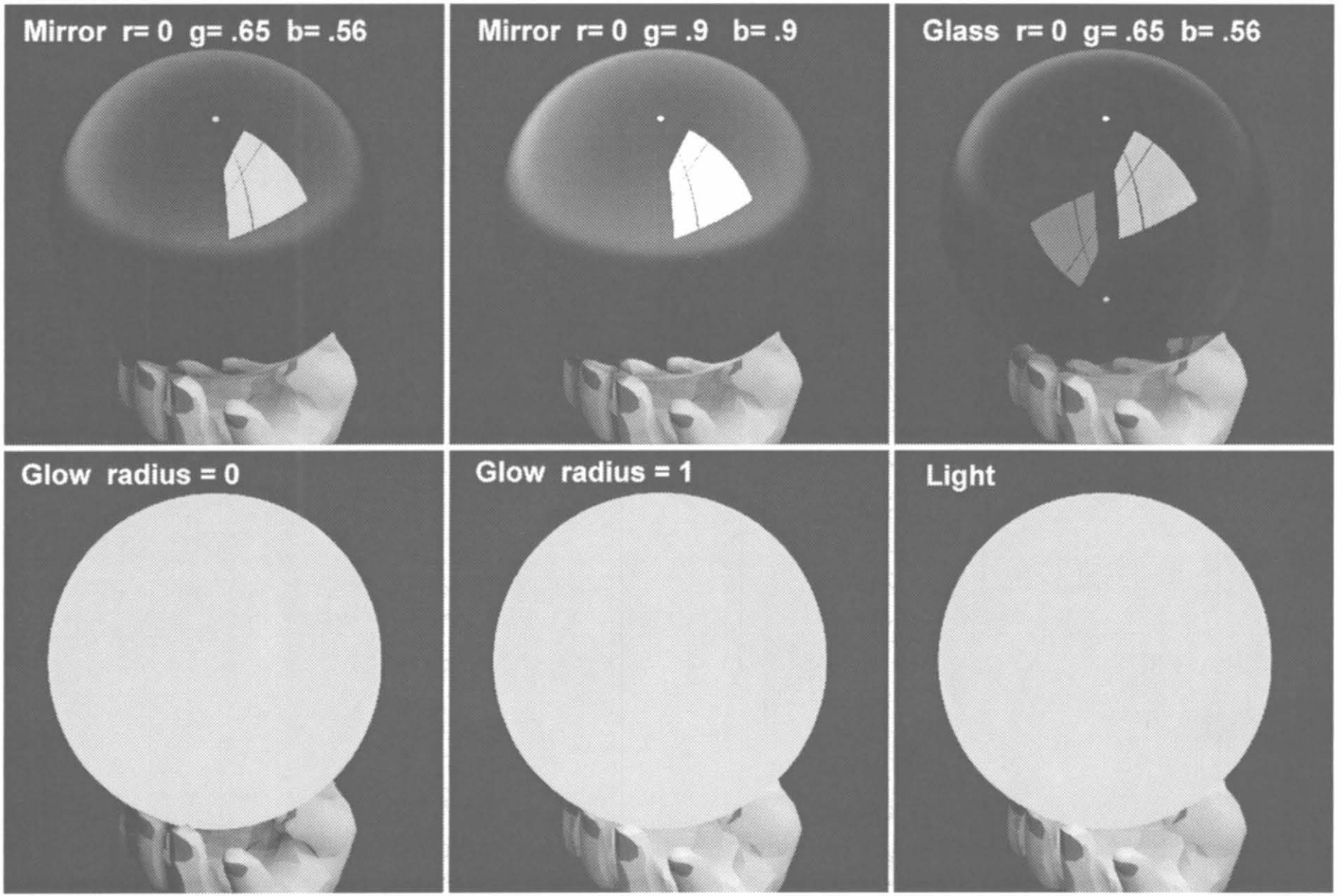

Metal
Plastic

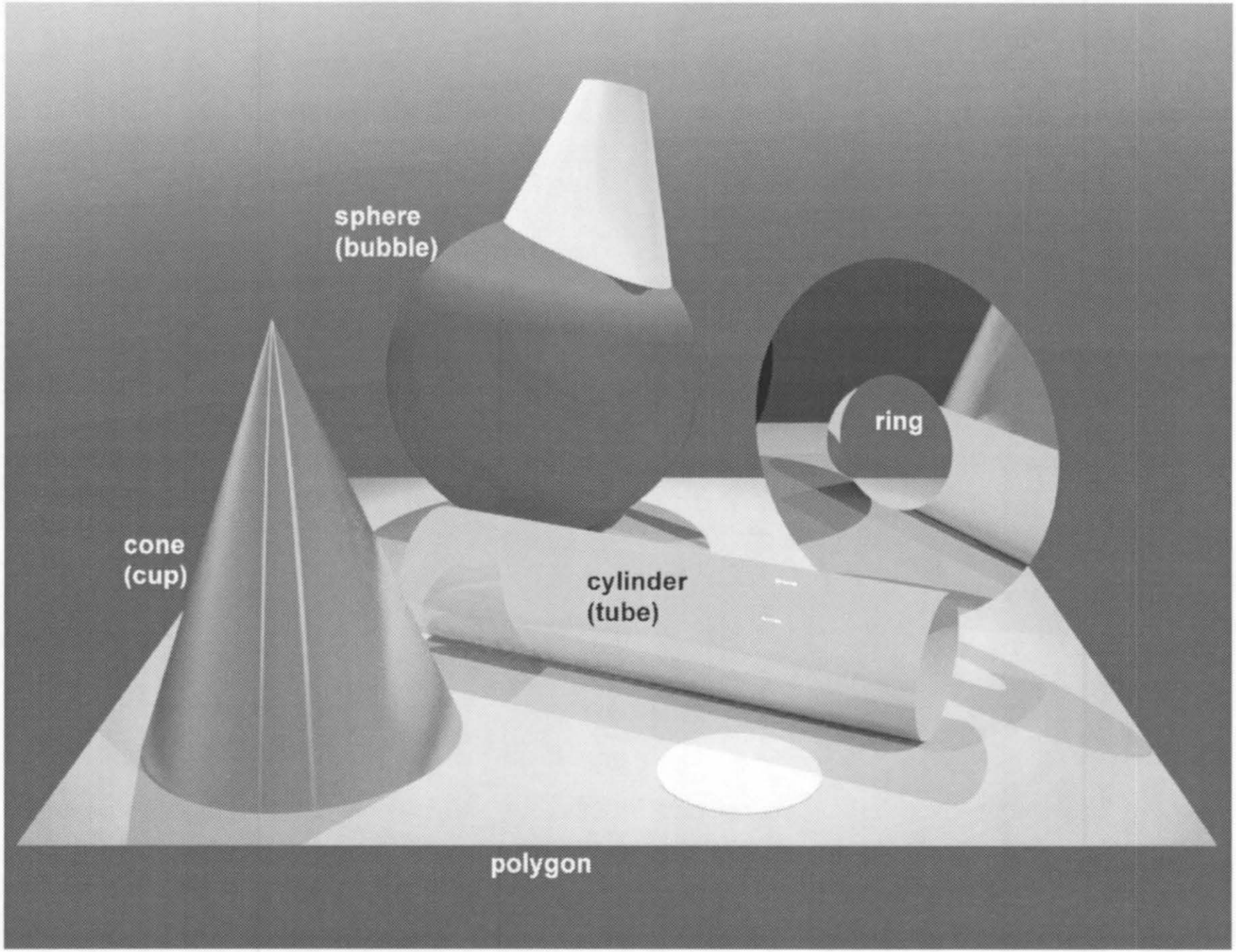

r = .0 g = .0 b = .0
r = .25 g = .0 b = .0
r = .5 g = .25 b = .0
r = .5 g = .5 b = .25
r = .5 g = .5 b = .5
r = .25 g = .57 b = .52

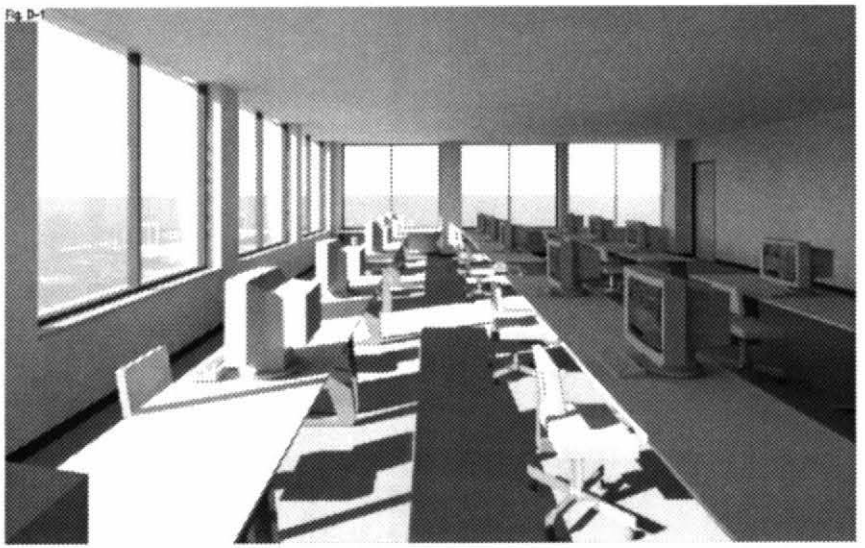

s = .0 r = .0
s = .025 r = .0
s = .07 r = .0
s = .07 r = .025
s = .07 r = .05
s = .07 r = .1
s = .07 r = .15
s = .025 r = .15
s = .0 r = .15

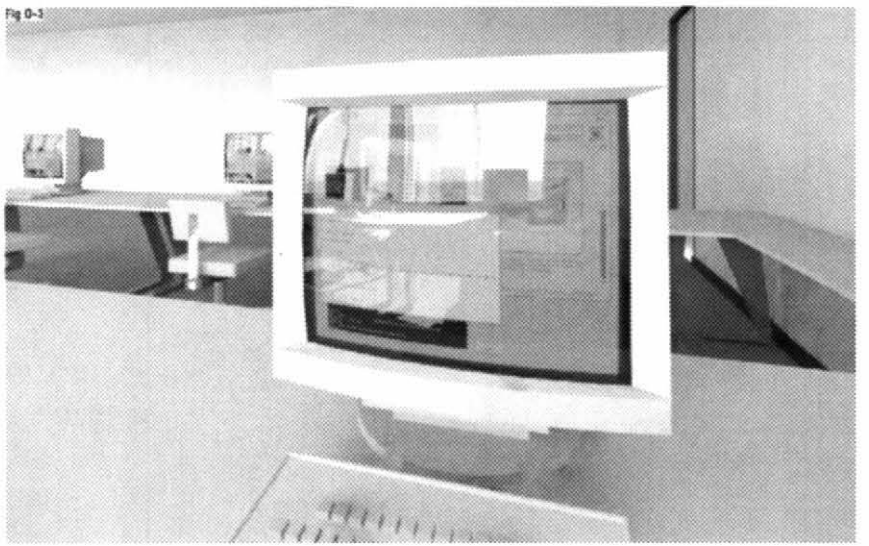

s = .5 r = .0
s = .75 r = .0
s = .99 r = .0
s = .95 r = .05
s = .95 r = .1
s = .95 r = .2
s = .95 r = .3
s = .75 r = .1
s = .5 r = .05

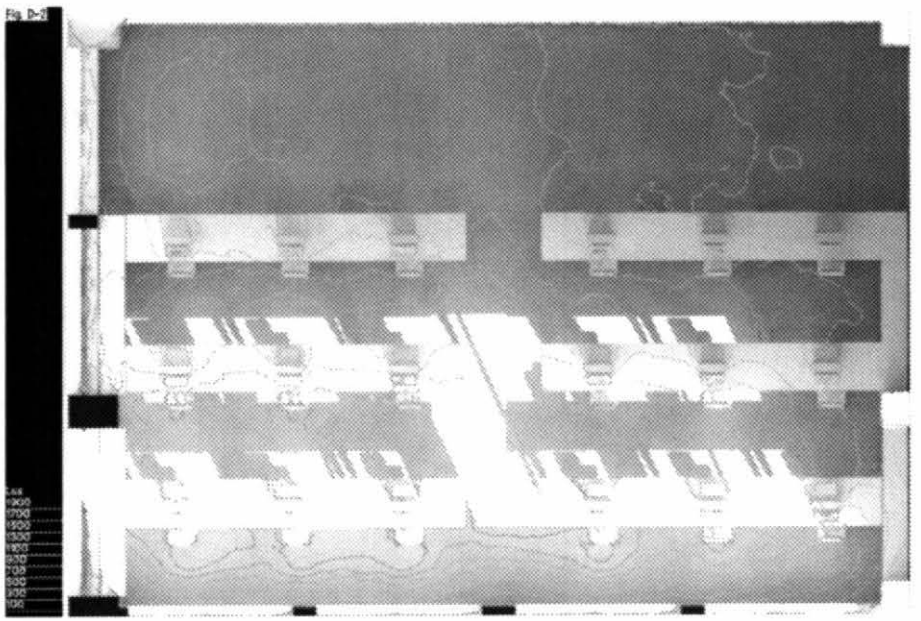

ux= 0 uy= 0 uz= 1
s= .9 ur= .02 vr= .2
ux= 0 uy= 1 uz= 1
s= .9 ur= .02 vr= .2
ux= 0 uy= 1 uz= 0
s= .9 ur= .02 vr= .2
ux= 0 uy= 1 uz= 0
s= .9 ur= .05 vr= .2
ux= 0 uy= 1 uz= 0
s= .9 ur= .1 vr= .1
ux= 0 uy= 1 uz= 0
s= .9 ur= .2 vr= .02

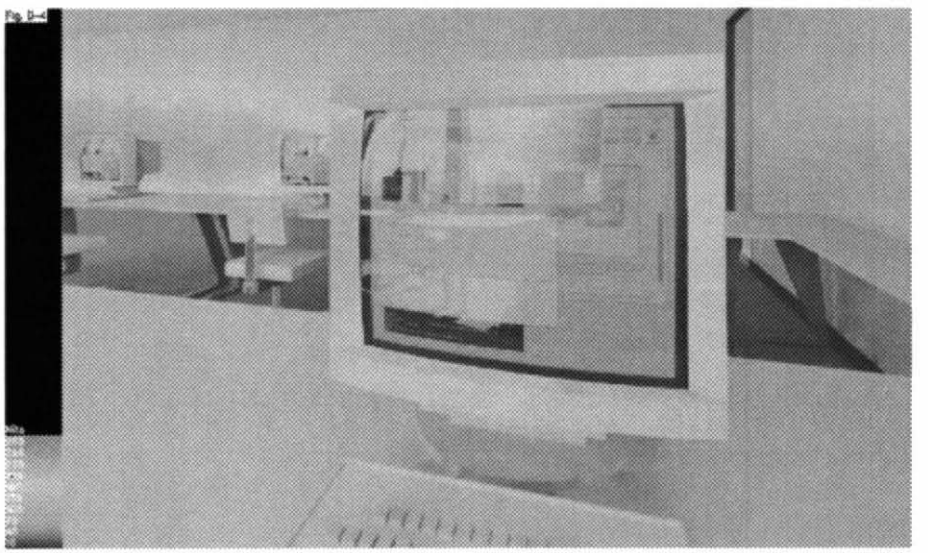

Mirror r= 0 g= .65 b= .56
Mirror r= 0 g= .9 b= .9
Glass r= 0 g= .65 b= .56
Glow radius = 0
Glow radius = 1
Light

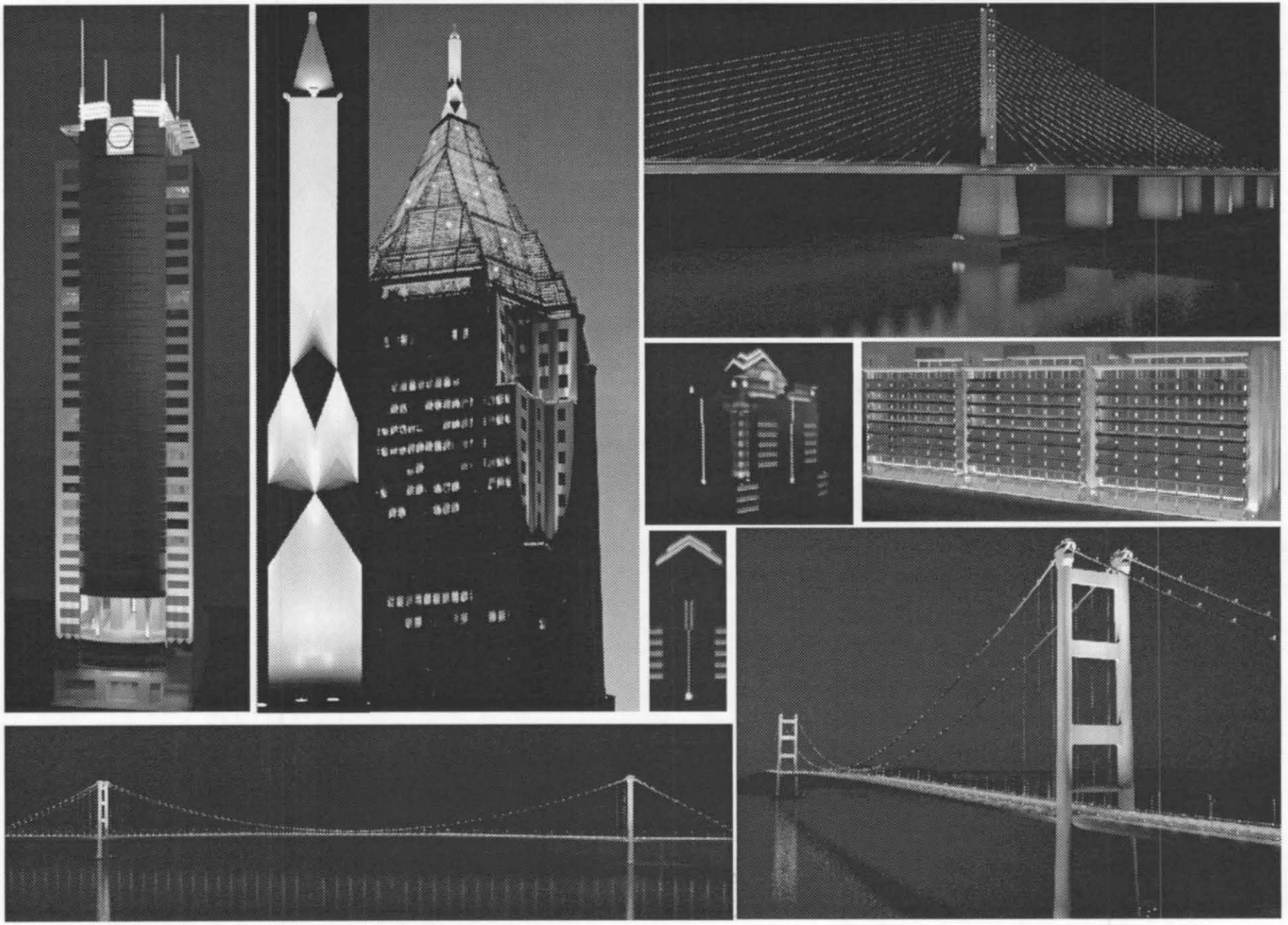

sphere
(bubble)
ring
cone
(cup)
cylinder
(tube)
polygon

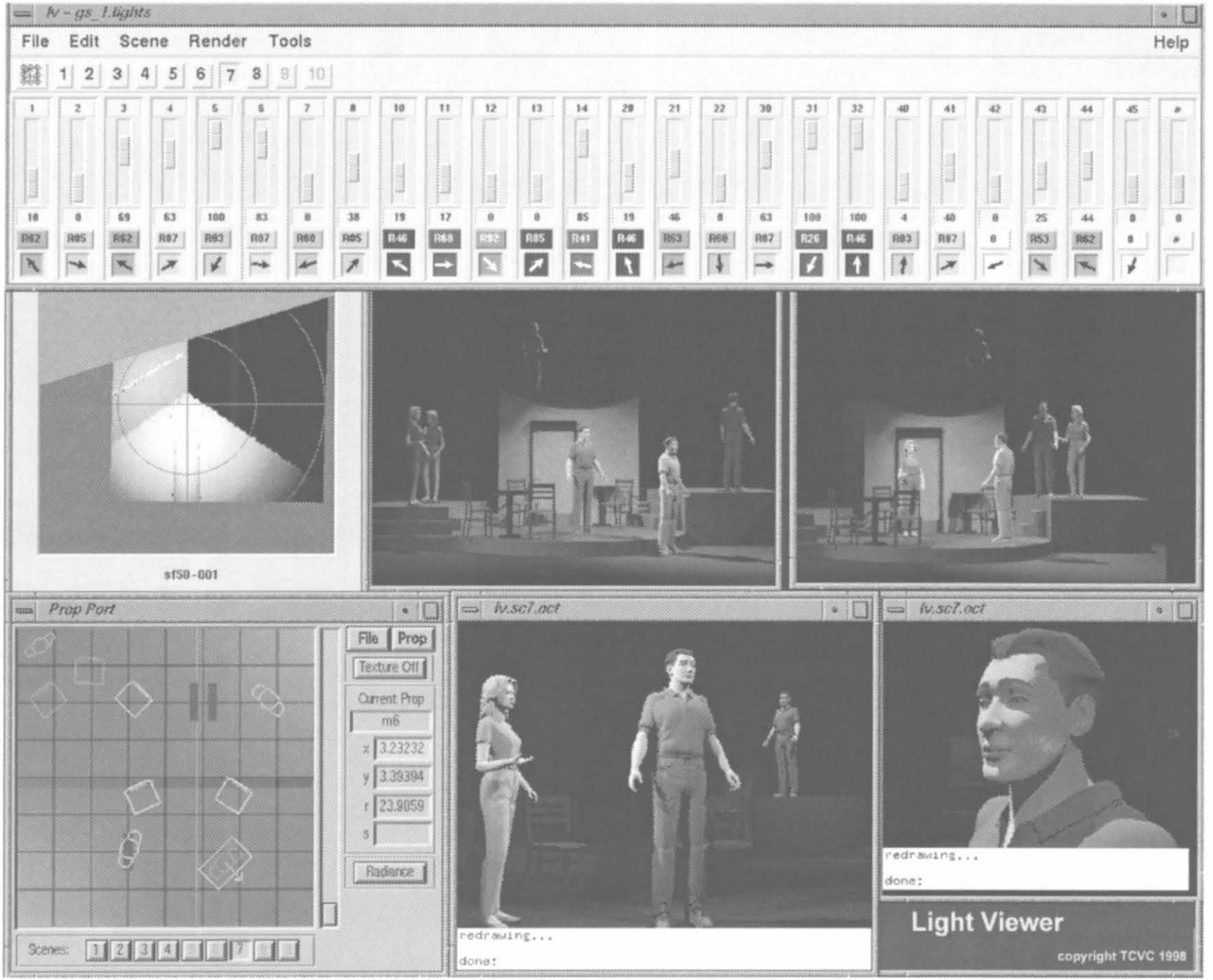











fv - gs_1.lights
File Edit Scene Render Tools Help
1 2 3 4 5 6 7 8 9 10
sf50-001
Prop Port
File Prop
Texture Off
Current Prop
m6
x 3.23232
y 3.39394
r 23.9059
s
Radiance
Scenes: 1 2 3 4 7
fv.sc7.oct
redrawing...
done:
fv.sc7.oct
redrawing...
done:
Light Viewer
copyright TCVC 1998

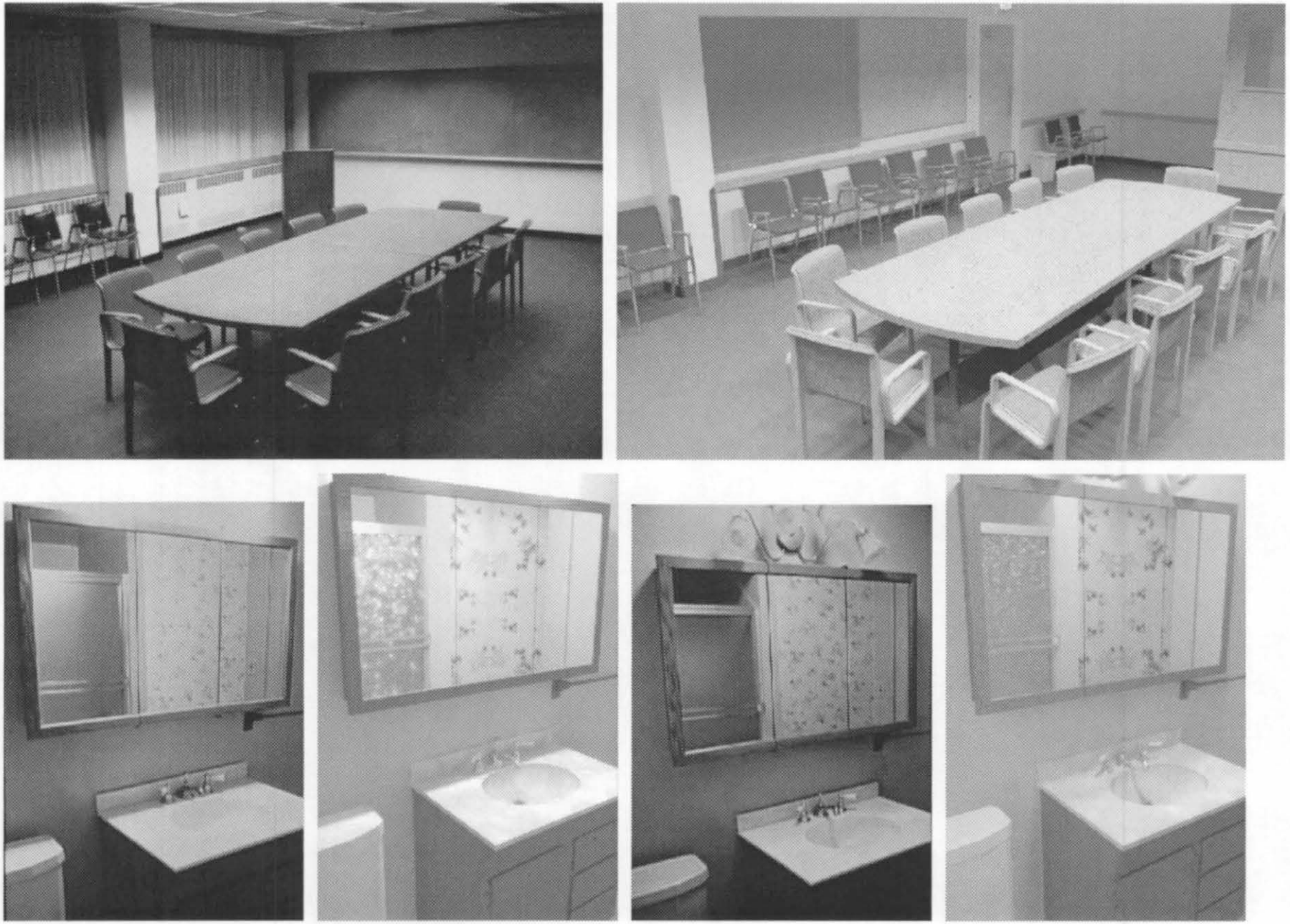


Figure I-1. Real vs. Radiance simulations of conference room and bathroom.