Abstract

In computer graphics there is a great desire to create visually realistic images. The difficulty of this task lies in the highly complex and varied nature of real environments such as: the needles of a pine tree, the terrain in a landscape, the uniqueness of a snow flake, or the subtle shadows in the corner of a room. This thesis deals with issues of the latter nature, examining how light interacts with the objects contained within an environment.

The global illumination problem consists of determining the complete interaction of light within an arbitrary scene. Subsets of this problem have been addressed and partially solved by classical techniques based on ray tracing or radiosity. Several variations on these techniques have also failed to provide a complete and general solution. This thesis introduces a general solution to the global illumination problem that essentially models the flow of light through an environment.

This thesis analyzes the modalities of one such light-driven algorithm that adaptively subdivides the environment into more manageable volumes called cells. Different methods of defining cells as well as various ways of representing the flow of light between cells are discussed. The issues relevant to the representation of the light flow emanating from a light source that enters a cell are also addressed. Approximations to the interaction between the light entering a cell and the contents of cell are followed by the formulation of a procedure to generate an image based on the modelling of the light flow through the environment. Finally, the potential of this paradigm is evaluated.

Acknowledgements

Many people made significant contributions that permitted this thesis to be completed. First and foremost, I am indebted to my supervisor, Eugene Fiume, for the numerous hours he dedicated to this thesis. His knowledge of mathematics and computer graphics, and his command of the English language greatly improved the contents and style of this thesis. I am very grateful to Alain Fournier: more than a second reader, his expertise in illumination sparked fascinating discussions and provided critical input into this research.

Without Chuan Chee, FIAT would have taken a lot longer to get off the ground: I probably owe him many "all-nighters"! The dedication of Marie-Claire Forgue and George Drettakis for reading this thesis is greatly appreciated: their numerous suggestions helped improve this thesis.

Special thanks to my friend Anthony Scian, for technical assistance, and of course, for his help in obtaining and setting up my terminal at home in record time. I also want to thank everybody else who in some way helped me complete this thesis, including all my friends who kept me from going insane. Merci à ma famille, pour tout leurs encouragements.

I am grateful to the Natural Science and Engineering Research Council of Canada, the Department of Computer Science and the University of Toronto for funding this research. Finally, I would like to thank the Lord for providing me with the ability and perseverance necessary to complete this work.

Contents

List of Figures

Chapter 1

Introduction

It has always been a fundamental goal of researchers in computer graphics to produce realistic images using computer generated scenes. Since the early 1970's, there has been a progression in the visual realism of computer generated images. This has been achieved in part through better understanding of the interaction of light with objects. It is the understanding of this interaction that we propose to explore.

In computer graphics, the production of an image begins with the description of the light sources and the objects which form a scene. The interaction of light with the objects can then be modelled, based on a description of the physical characteristics of the elements of the scene. The problem of determining the complete interaction between the various elements of a scene is known as global illumination.

The global illumination problem for scenes with objects having perfectly specular surfaces is well understood, and is partially solved using a technique known as ray-tracing. This problem can also be solved for scenes composed of objects that are both opaque and possess rough surfaces, using the radiosity technique. However, neither of these algorithms can successfully model the interaction of light in scenes containing objects with arbitrary types of surfaces and light sources.

The purpose of this thesis is to introduce the First Illumination Algorithm at Toronto (FIAT), a new computational model for global illumination. FIAT is a light-driven algorithm, where adaptively determined cells partition the environment; their light energy budget is iteratively balanced in an effort to simulate the flow of light through a scene. A local illumination model is used at the cell level, to determine the interaction of light with the objects within this cell. Global illumination is computed by having cells exchange information concerning the light flux at their boundaries. The lighting information for the objects contained in the cells is then used to compute an image.

In the following chapters, topics relevant to this new paradigm will be discussed. Chapter 2 will survey the current methods for determining global illumination and briefly introduce FIAT. In chapter 3, the overall aspects of FIAT will be high-lighted. Chapter 4 will examine the propagation of light between cells. Light sources will then be analyzed in chapter 5. The local interaction of light with objects inside a cell will be the focus of chapter 6. Chapter 7 will present a method for computing an image using FIAT. Chapter 8 will discuss the current implementation of FIAT, focusing on some of the problems and results. Finally, chapter 9 will present conclusions with regard to this algorithm, and suggest further considerations.

Chapter 2

Global Illumination

Determining global illumination consists of calculating the correct distribution of light reaching each element of a scene. This illumination can come from various sources, through direct illumination from lights or as the result of the interaction of emitted light with the environment. After an introduction to concepts necessary to the understanding and formulation of the problem, the following sections will survey previous techniques used for computing the global illumination: ray-tracing, radiosity, hybrids and others. The final section of this chapter will provide a brief introduction to FIAT.

2.1 Terminology

Under certain circumstances, the task of determining the interaction of light with the elements of a scene can be reduced to a geometrical problem. In particular, if we assume that light travels in a straight line, that there is no participating medium interfering with propagation, and that all light operators are linear[1], then a geometrical model can be constructed. Very few lighting phenomena such as phosphorescence exhibit a non-linear behaviour, and thus the linearity assumption is not very restrictive.

[1]A light operator T is linear if and only if $T(aL_1 + bL_2) = aT(L_1) + bT(L_2)$, where L_1 and L_2 are light quantities, and a and b are constants.

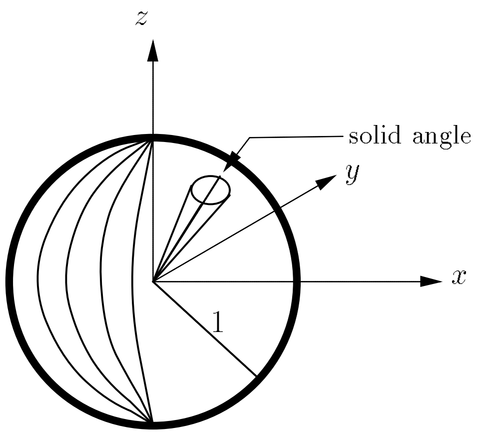


Figure 2.1: Solid Angle

As most existing global illumination algorithms do, we will make these assumptions; they will enable us to consider global illumination as a geometrical problem. We will also only consider the steady state of the system, where the flow of light through a scene remains constant. Under these assumptions, power, which is energy per unit time, and energy are interchangeable terms. We will use the term power for both quantities. We will also ignore the spectral character of light, and assume that the global illumination problem can be determined independently for any frequency.

Let a scene $S \subseteq \mathbb{R}^{\nvDash}$ consist of a list of n light sources, $\{L_1, \ldots, L_n\}$, and of m objects, $\{O_1, \ldots, O_m\}$. Each light source definition describes that light source's type, spectral and spatial distribution, shape, and location. Each object is defined as a description of its properties: type of surface, shape, and location. If light sources are to be considered as objects with respect to the reflection/refraction of light impinging on them, they will also be assigned object properties. If the distinction between object and light source is irrelevant, we will refer to an arbitrary component as an element.

The *solid angle* value associated with a cone C of arbitrary shape, centred at the origin of a sphere of unit radius, is defined as its intersection with the sphere's surface, as illustrated in figure 2.1. In terms of Cartesian coordinates, the polar coordinates (θ, ϕ) for a vector (x, y, z) are defined such that: θ is the angle measured from the z-axis, and ϕ the counterclockwise angle of the vector projected onto the xy-plane, as illustrated in figure 2.2. If the bounds of integration of the cone are defined in terms

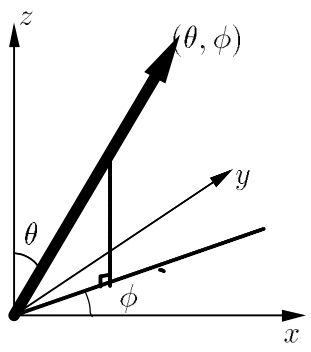


Figure 2.2: Polar Coordinates

of polar coordinates, θ and ϕ, then the solid angle value ω associated with C is the double integral:

$$\omega = \int_\phi \int_\theta \sin\theta \, d\theta \, d\phi \qquad 0 \le \phi \le \pi,\ 0 \le \theta \le 2\pi, \text{ and } (\theta,\phi) \in C. \tag{2.1}$$

The range for θ is only a convention, and we could just as well use the range $[-\pi, \pi]$. If the cone that defines the bounds of integration is an entire hemisphere, the associated solid angle is:

$$\int_0^\pi \int_0^{2\pi} \sin\theta \, d\theta \, d\phi = 2\pi. \tag{2.2}$$

The solid angle that corresponds to the entire hemisphere is 2π.

Let E be an element of the scene S, and let dA_E be an infinitesimal surface element of E, and let $d\omega$ be an infinitesimal solid angle element centred in the direction of a vector W originating from dA_E. The directed radial power associated with this direction is the power emitted from, received through, or transmitted from dA_E and through $d\omega$ per unit of time. The outgoing directed radial power is the sum of the emitted and transmitted component. The received directed radial power will also be referred to as the incoming power.

Given a scene S, computing the global illumination consists in determining the relationship between the outgoing and incoming directed radial power of the various elements of the scene, for each infinitesimal solid angle and element of surface area.

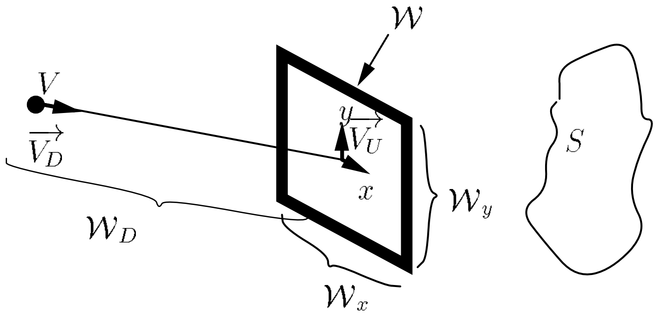


Figure 2.3: View Dependence

The inherent intractability of an exact and general solution is due to the uncountable set of dA_E's for each $E \in S$, and the uncountable set of $d\omega$'s defining a sphere of illuminance about each dA_E. Because of the complexity of this general problem, it is difficult to obtain an analytic solution, and a discrete approximation is usually computed.

View dependence can be used to reduce the storage requirements of a solution to this problem. Let the eye position, or viewpoint, V consist of a point in $\mathbb{R}^3$, and associate with it a viewing direction, $\vec{V_D}$. Define up by a vector $\vec{V_U}$. Define also a window $\mathcal{W}$, as a $\mathcal{W}_x$ by $\mathcal{W}_y$ rectangular cross-section of space, centred on the line $\{V + t\vec{V_D} \mid t \in \mathbb{R}\}$, at a distance $\mathcal{W}_D \in \mathbb{R}$, from V. The axes of $\mathcal{W}$ correspond to the standard Euclidean plane axes, with the y-axis and $\vec{V_U}$ collinear, as in figure 2.3.

The purpose of the window is to gather light information flowing through it, towards the viewpoint. Typically, a window is associated with an image display and is tessellated into pixels. If V is located between S and $\mathcal{W}$, then one can think of $\mathcal{W}$ as mimicking the film in a camera. To simplify depth calculations with respect to the viewpoint, $\mathcal{W}$ is usually positioned between S and V.

If we imposed viewing conditions on a scene, with viewpoint V and window $\mathcal{W}$ as previously defined, we can define a set A of *visible* infinitesimal surface elements as:

$$\begin{aligned} A \; = \; & \{dA \in S \mid \exists t \geq 0, \;\; \text{point } P \in \mathcal{W} \text{ such that } V + t(P - V) = dA \text{ and} \\ & \forall t', 0 \leq t' < t, \nexists dA' \text{ such that } V + t'(P - V) = dA'\}. \end{aligned} \tag{2.3}$$

For each dA we associate a vector W directed towards V. This set A contains only the infinitesimal surface elements visible from V, and global illumination has to be determined for members of A. In practice, the entire scene still might have to be considered in the computation, but only the illumination values for members of A need to be explicitly determined.

2.2 Local Illumination

A local illumination model describes the behaviour of light at the surface level; it establishes a relationship between light received and light transmitted through reflection or refraction. Computer graphics researchers have experimented with numerous such models.

The first attempts at producing realistic images in computer graphics dealt with a simplified local illumination model. An ambient illumination term was used to compensate for the missing global illumination information, and objects were shaded using non-physical methods. Scenes were rendered by using a visibility algorithm [Warn69][Watk70] to determine the elements of the scene that were visible to the eye. The colour of the object was used to shade the appropriate section of the image where this object was seen. Procedures for producing smoother shades for curved surfaces were then introduced, notably Gouraud's polygonal shade interpolation [Gour71] and Phong's normal interpolation. Phong also introduced an improved local illumination model [Phon75], which modelled more than perfectly diffuse surfaces, adding specular highlights. Many other models have since been proposed, notably in: [Blin77] [Whit80] [Cook81] [Hall83].

Local illumination models form an important component of the global illumination problem. We shall present an general model that characterizes most illumination models. Let P represent the point of intersection of a vector W_{in}, representing an incoming light beam, with a surface element dA, as in figure 2.4. Let N be the unit surface normal of dA at P. Let R point in the reflected direction from the incident direction W_{in} and T point in the refracted direction. According to the reflection laws

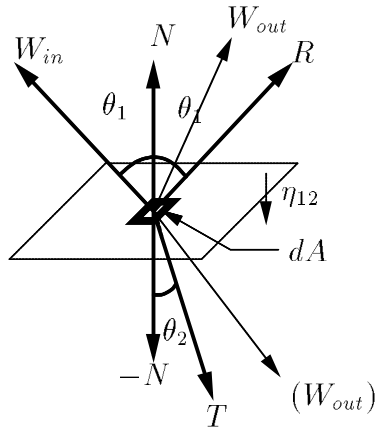


Figure 2.4: Local Illumination Model

we know that W_{in}, N, R and T lie in the same plane and that the angle θ_1 between W_{in} and N is the same as the one between R and N. Furthermore, T is such that $\sin\theta_1 = \eta_{12}\sin\theta_2$, in accordance with Snell's law, where θ_1 is the angle between W_{in} and N, and θ_2 is the angle between T and $-N$, as illustrated in figure 2.4, and $\eta_{12} \in \mathbb{R}$ is the index of refraction from medium 1 to medium 2.

Bidirectional reflectance relates light received in the incoming direction, to light transmitted, through reflection or refraction, in the outgoing direction. The following description of a local illumination model's bidirectional reflectance is simplified by breaking down the transmitted light into several components.

Ambient An estimated contribution of light from the environment in general; used to compensate for a lack of global illumination information.

Diffuse Reflection The quantity of light that is uniformly distributed in the hemisphere of directions above the surface element.

Specular Reflection The component of the light that is reflected in all directions based upon a distribution function. This function depends on the surface properties and will typically achieve a maximum in the direction of R.

Specular Refraction The component of light that is refracted through the interface, incoherently, and typically concentrated about the refracted direction, as determined by a distribution function.

If W_{in}, the direction of incoming light, and a viewing direction W_{out}, originate from the same side of the interface, then the power reflected through an infinitesimal solid angle centred on W_{out}, from dA, due to the incoming light is:

$$P_r(W_{out}, dA) = k_a I_a + k_d f_d(W_{in}, N) + k_s f_s(W_{in}, W_{out}, N). \tag{2.4}$$

Otherwise, the refracted light is:

$$P_t(W_{out}, dA) = k_t f_t(W_{in}, W_{out}, \eta_{12}, N). \tag{2.5}$$

In the above formulae, I_a represents the ambient component, f_d represents the diffuse reflectance function, f_s the specular reflectance function and f_t the specular refractance function. The constants k_a, k_s, k_d, k_t are used to assign weights to the corresponding components in the illumination model.

Ideally, a local illumination model defines the interaction of light with a given material, determining the outgoing power given the light impinging on the surface. Since this is difficult to achieve, approximations are used, such as assuming an isotropic illumination model[2].

2.3 Global Illumination

In the past decade, new techniques have been proposed that provide approximate solutions to instances of the global illumination problem. We now examine ray-tracing, radiosity, hybrids and other existing techniques used to solve this problem.

2.3.1 Ray-Tracing

First introduced to computer graphics by Whitted [Whit80], ray tracing consists of tracing light paths into the environment. Traditionally, these paths originate from

[2]A model is said to be *isotropic* if f_s and f_t do not depend on the planar orientation of the surface, with respect to W_{in} and W_{out}. See [Kaji85][Yoko88][Poul89] for a discussion of reflection models which are not isotropic.

the viewpoint, and are cast through pre-determined points of the window, such as the centre of each pixel. Rays are then recursively cast from the intersected objects, in the ideal reflection and refraction directions. Once the tree of paths has been determined, either because there are no more paths to explore or the tree has attained some predetermined depth, a reverse walkthrough (i.e., starting from the last intersected object) is performed. Local illumination information is percolated back towards the viewer, and is ultimately used in assigning a shade to the pixel. This type of recursive ray tracing is often referred to as "ray tracing from the eye".

Ray tracers have used more realistic local illumination models than the previous non-physical models, to allow them to account for secondary reflection and refraction components. Modified versions of both Blinn [Blin77] and Phong [Phon75] models have been used ([Whit80], [Hall83].) These typically include both a specular and a diffuse reflection component, as well as a specular refraction component. However, most ray tracers still use an ad hoc ambient term, in order to compensate for an incomplete calculation of the global illumination.

We now consider how ray-tracing approximates the global illumination across a single pixel. Let P_{ij} correspond to the $(i, j)^{th}$ pixel of the window $\mathcal{W}$. In simple ray tracing, a single ray is cast through the centre of P_{ij}, and the intersection point P of the ray with the closest object, O in S, is found. If no such object exists, the flux crossing this pixel is considered to be null. If such an object O exists, the shade of pixel P_{ij} is determined by the intensity leaving the infinitesimal element of surface dA of O in an infinitesimal solid angle $d\omega$, centred at P, and about the direction W from dA to V, as illustrated in figure 2.5.

A local illumination model is used to calculate this power. Each *visible*[3] light source's contribution to the outgoing power is calculated according to the model. In addition, both the ideal reflection and refraction directions — with respect to W —are sampled for secondary illumination, as illustrated in figure 2.5.

In order to sample these secondary directions, rays are cast from the intersection

[3]Visibility determination of an object from a light source requires occlusion tests; these estimate the fraction of light leaving the source and reaching the intersection point.

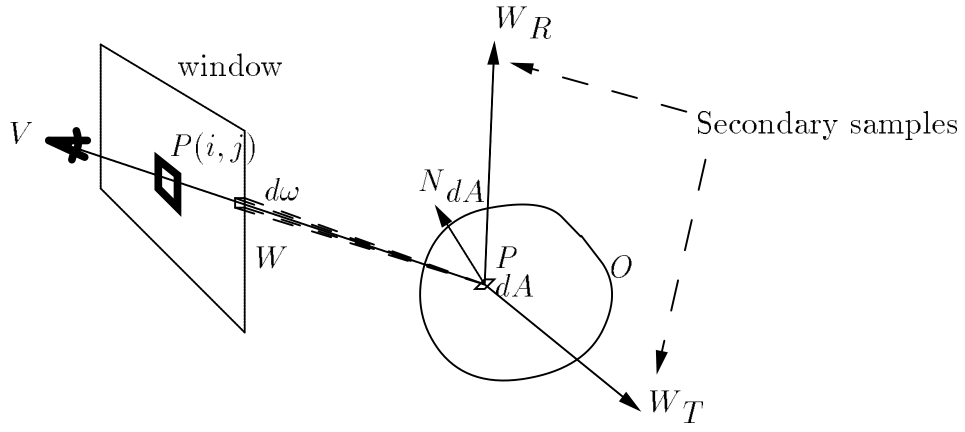


Figure 2.5: Secondary Sampling Directions

point in both the ideal reflection and refraction directions. The secondary contribution from the newly intersected object O' corresponds to the power leaving O' through an infinitesimal solid angle centred at the intersection point on O', and about a vector defined by the two intersection points. This power is determined in the same fashion as it was for the previous point, using primary lighting information, and secondary information from the ideal reflection and refraction directions. This recursive sampling of incoming light in the ideal reflection and refraction directions terminates either when no intersecting object is found in all secondary directions, or a maximal depth has been attained for the tree representing the ray-trace.

For each leaf in the tree representing an intersected object, the outgoing power is determined strictly from the primary light sources. The local illumination model is used to determine the power directed towards the intersected object associated with the node's immediate ancestor in the tree. In a progression through levels, and towards the root of the tree, the missing secondary illumination at each node's intersected object is obtained from the child(ren)'s outgoing power. The power at the root of the tree is eventually obtained, and this is used to determine the shade of the pixel.

Figure 2.6 illustrates a ray-trace. The ray cast from the view point through the pixel P_{ij} intersects an object at the surface element dA_1. The secondary sample in the ideal reflection direction is determined from the surface element dA_2 that lies

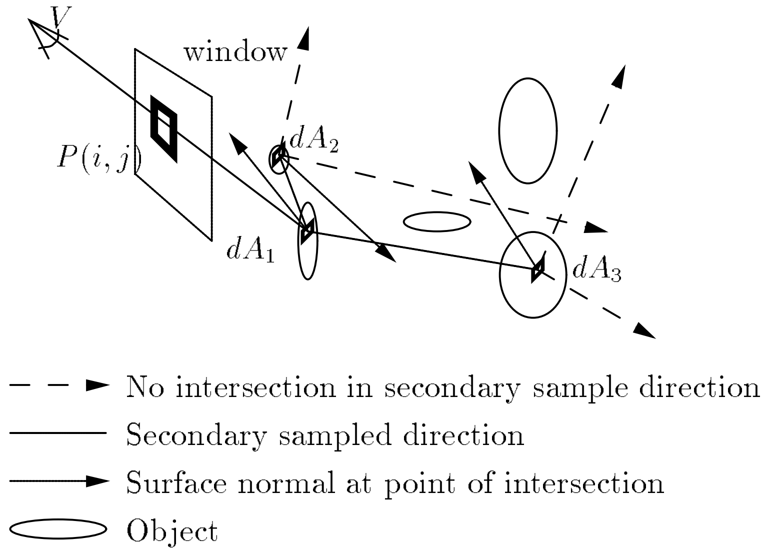


Figure 2.6: Sample Ray Trace

in the path of the reflection. The secondary sample in the refraction direction is determined from the surface element dA_3 that lies in the path of the refraction. The power for each of dA_2 and dA_3 is calculated strictly from the primary lighting (not shown), since the secondary sample directions do not intersect any object. The power from both secondary directions, together with the primary lighting, is finally used to determine the power transmitted towards the view point.

In summary, for each pixel P_{ij}, a tree of rays is constructed by casting rays through the scene. The locally determined illumination is percolated towards the parent node, and finally to the root. This provides additional lighting information in the reflective and refractive directions. The outgoing power at the root, in the direction of the viewpoint, is used to compute the shade of P_{ij}.

One obvious problem with ray tracing is that only one infinitesimal solid angle is sampled for each pixel. Because objects are not always appropriately represented along the sight line that goes through the centre of a pixel, it can be the case that details such as small objects are missing. Techniques such as adaptive super-sampling [Whit80] can be used to alleviate these problems. Instead of computing global illumination for a single infinitesimal solid angle per pixel, a number of these are sampled, across the extent of the pixel, depending on the distribution of objects, and the results

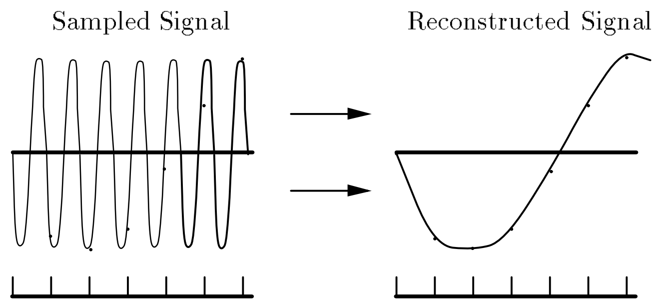


Figure 2.7: An Example of One-Dimensional Aliasing

are integrated into a single shade for the pixel.

One of the side-effects of the pixel level sampling problem is that problems appears in the image, in the form of unwanted aliasing artifacts. A classical illustration of this problem occurs when a high frequency illumination signal is under-sampled, and reconstructed from these samples, showing up as a signal of a lower frequency (see figure 2.7). With a regular sampling rate, as is the case in tracing paths through the centre of equally spaced pixels, this aliasing phenomenon can occur if the frequency of the signal is above the *Nyquist limit.* This limit requires the sampling frequency to be at least twice the maximum frequency of the signal.

Many heuristics have been suggested to alleviate sampling artifacts:

Stochastic Sampling Solid angles are sampled in a pseudo-random fashion throughout the solid angle determined by the the pixel and the viewpoint [Cook85],[Dipp85].

Stratified Sampling The sampling space is partitioned into subspaces. Each subspace is sampled often enough to satisfy a local error estimate [Lee85].

Cone Tracing Cones are traced into the environment, with area intersection approximations replacing ray intersections. As a result, coverage information is collected on a pixel basis [Aman84][Aman88].

Beam Tracing The viewing frustum is traced into an environment of polygonal objects. Each intersecting polygon produces a subdivision of the beam being traced. [Heck84]

Pencil Tracing Small bundles of rays are traced through the environment, using paraxial ray properties, with error control [Shin87].

Note that cone, beam, and pencil tracing provide additional information at more than the pixel level, because beams of non-zero cross section are traced into the environment.

By appropriate sampling and filtering[4] of the various rays cast into the environment, many optical effects can be emulated. By treating a viewpoint as an aperture of finite non-zero dimension, and sampling by using stochastically determined points in this aperture, depth of field phenomena such as blurring can be obtained. If the samples are distributed through time, motion blur can be simulated in a dynamic scene.

A more important sampling problem is the insufficient amount of global illumination computed at each node of the tree. Thus, even with a better selection, or larger selection, or both, of infinitesimal solid angles for each pixel, the light energy calculated for each of these would still suffer from undersampling. For any node in the ray-traced tree, the illumination received is only computed from the primary light sources, and from the ideal reflection and refraction directions. In some solutions to the undersampling problem, additional illumination information is gathered for each ray, either through additional ray samples or area sampling. While this enhances features such as the concentration of light due to optical effects, and penumbrae, only a small fraction of the volume of incoming light is considered at each node. Because of this lack of information concerning incoming light, only partial global illumination is reflected in the final image. As a result, secondary illumination is generally ignored. Suppose that a mirror is placed such that the light from a primary light source is reflected towards an intersection point in the ray-trace tree, as illustrated in figure 2.8. Since the mirror doesn't lie in the ideal reflection or refraction direction with respect to the viewing direction V, a standard ray-tracer would not detect this important secondary illumination[Fium87].

[4] See [Cook84] for further details.

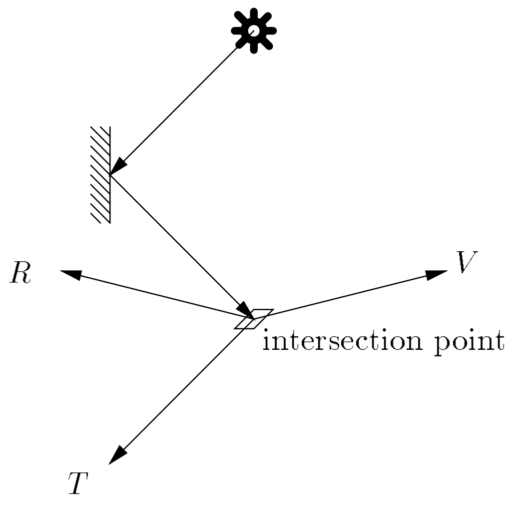


Figure 2.8: Missing Secondary Illumination

Additionally, the illumination received through the small set of sampled solid angles is not completely accurate global illumination. In the case of light originating at the primary light sources, the contribution will be inaccurate if the interaction with intervening objects is not carefully considered. The illumination received in the ideal reflection and refraction directions, will suffer additional inaccuracy due to the limited global illumination calculated for the children of this node, as per the above discussion. Ray-tracing's recursive nature compounds the sampling problem by using samples whose inaccuracy is a function of previous samples.

An often over-looked problem occurring with ray-tracers is the physical inaccuracy of the local illumination model with respect to energy conservation. Often, the global illumination determined for an arbitrary pixel is a physically meaningless number, because ray tracing programs can bypass any physical interpretation, and convert this number to a shade. Because traditional ray tracing only deals with a tree of rays traced from the viewer, it is very tempting to neglect the physical veracity of the local illumination model, ignoring the fact that quantities should represent power or energy.

Finally, another problem with the ray-tracing strategy is dealing with objects that intervene between a source and a point of intersection in the tree of rays. To calculate primary lighting information for a point, we must determine the light that directly reaches it. If an object lies between the source and the point, it may prevent some of the light from directly reaching this point. For this reason, the simplest solution is to treat it as secondary light, and hope that the secondary samples account for this light. The occlusion test itself becomes complicated when light sources of extended area are used. This explains why most ray-tracers only support light sources of infinitesimal surface area, known as *point* light sources.

From a computational point of view, ray tracing is generally more expensive then earlier empirical rendering methods, mostly due to the cost of multiple ray intersections with the various objects of the scene. Innovations such as space subdivision have contributed to the reduction of the intersection cost[Glas84][Fuji86][Snyd87][Aman87].

Even though ray tracing is portrayed as a simplistic approach to solving a small subset of the global illumination problem, it is nonetheless a powerful image synthesis tool. Because it includes the ideal reflection and refraction directions in its global illumination computation, scenes composed of objects such as glass spheres, and mirrors can be convincingly rendered. Since the human eye detects and appreciates these sharp reflections and refractions, ray traced images do resolve an important aspect of the global illumination. Unfortunately, these limitations imply that diffusely reflecting surfaces are poorly handled, and this lack of global illumination reflects in the quality of images for such scenes; in particular, most secondary illumination is missing. For this reason, ray tracers include an ad hoc ambient term in the local illumination model, to compensate for this lack of light propagation.

2.3.2 Radiosity

The global illumination technique known as *radiosity*, is derived from a thermal engineering method used for the calculation of radiative heat transfers. First introduced to computer graphics by Goral et al. [Gora84], this paradigm has evolved into an effective solution to the global illumination problem for polygonal scenes composed of perfectly diffuse emitters and reflectors. A set of equations relating incoming power to outgoing power is derived and solved. These equations establish a geometric relationship among the various elements of the scene. Once solved, the equations give an accurate description of the global illumination everywhere in the scene, which eliminates the need for an ad hoc ambient term.

Consider a scene S such that every object in S is a perfectly diffuse[5] (Lambertian) surface, and every light source is a perfect diffuse emitters. Generally, the scene forms a closed environment, but artificial objects can be added to S to ensure this. Every object is assumed to be opaque; no transmission of light through objects will be handled by radiosity. Each element of S is subdivided into a number of patches, and each one is assumed to be of uniform composition, illumination, and emission power[6].

Let the set of patches so obtained be called $A_{\mathcal{S}}$, and let $n_{\mathcal{S}} = |A_{\mathcal{S}}|$. For each patch $P_i \in A_{\mathcal{S}}$ in this environment, the flux of power leaving a patch is equal to the sum of the patch's emitted power, and of the fraction of the received power that it reflects. Because all patches are assumed to be ideal diffuse reflectors, the illumination contribution of each patch can be geometrically related to compute the global illumination accurately. Using the following notations,

B_j : Power leaving P_j.

E_j : Power directly emitted by P_j.

F_{ij} : Fraction of radiant power leaving P_i and impinging on P_j (form-factor)

ρ_j : Reflectivity of P_j.

we can express the radiosity relation for any patch P_i, to all other patches as:

$$B_i = E_i + \rho_i \sum_{j \neq i} B_j F_{ji}. \tag{2.6}$$

Determining the form factors F_{ji} requires computing the average visibility of patch i from patch j. As is shown in figure 2.9, this requires estimating the quadruple integral

$$F_{ji} = \frac{1}{A_j} \int_{A_j} \int_{A_i} \frac{\cos\theta_j \cos\theta_i}{\pi r^2} HID\, dA_j\, dA_i \tag{2.7}$$

[5]In the sense that it emits/transmits light intensity equally through every infinitesimal solid angle originating on the surface.

[6]The patches are assumed uniform — each infinitesimal element of surface is a perfectly diffuse emitter/receiver, and each is assumed to emit/receive the same power — for computational purposes, although they seldom really are.

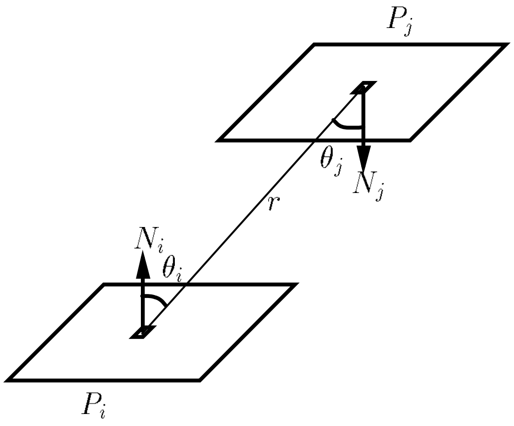


Figure 2.9: Form factors

where HID is a visibility boolean function. An approximation to this integral is calculated, if the analytic solution is untractable.

Once the form factors are calculated, a matrix representing the $n_{\mathcal{S}}$ equations in $n_{\mathcal{S}}$ unknowns, is solved. The computed B_i's represent the diffuse power leaving the respective patches. For a patch P_i, each infinitesimal surface element is such that the outgoing power is the same for each infinitesimal solid angle, and the total contribution is B_i. Various methods have been used [Gora84][Cohe85][Cohe88] to solve the system of equation, but a complete solution generally takes $O(n_{\mathcal{S}}^2)$ operations. After the system has been solved, the value of the B_i's represent global illumination information, and are used to generate an actual image of the scene.

To render the image corresponding to a chosen viewpoint V and window $\mathcal{W}$, one chooses the set of infinitesimal solid angles $d\omega$ to be sampled for each pixel. Although a uniform radiosity has been assumed throughout the initial phase of the algorithm, the B_i of the patch intersected by the relevant $d\omega$ is not used directly. To avoid shading patches uniformly, the B_i is treated as the average radiosity of patch P_i, such that all infinitesimal solid angles, originating at the centre of the patch emit the same power, and their power sum to B_i. Using neighbouring patches, a gradient is established between the average radiosity at the centre of each patch, and values are interpolated (or extrapolated) at the corner of each patch. Once this has been done, the global illumination received by the pixel through $d\omega$ is approximated using a bilinear interpolation of the calculated corner intensities of the intersected patch P_i.

Since no assumption depends on a fixed viewpoint, the choice of viewing parameters V and C is deferred until this point. It is possible to solve the system once for a scene, and subsequently to render images of this scene from different viewpoints, using the same global illumination. [Note: In practice, animated sequences deal with moving objects within a scene. This affects the global illumination, and different equations have to be computed and solved.]

The principal advantage of the radiosity method is its accurate handling of diffuse inter-reflections in the environment. For scenes containing only diffuse elements, the global illumination problem is effectively solved. Resulting images confirm this, especially through noticeable features such as soft penumbral shadows, and colour bleeding near abutting objects.

However, the radiosity method has several drawbacks, some of which are unavoidable. First, determining form factors is a non-trivial task. Second, solving the system of linear equations is a computationally expensive operation. Third, high gradients of intensity can force an exceedingly large number of patches to be generated, compounding the first two problems. Fourth and foremost, radiosity's inability to handle more than diffuse inter-reflections limit the generality of the solution.

The first problem is determining the form factors, which requires solving an intricate quadruple integral, for each pair of patches. In some cases, these can be reduced to a few line integrals, of lower complexity[Gora84]. However, to solve the general case, only an approximation is feasible. In [Cohe85], a method is described which provides reasonable approximations to form factors, and also takes into consideration occlusion of objects. Form factors can be approximated, but as $O(n^2)$ of them are required, storage space can become prohibitive[7].

The second problem is that using a straightforward elimination/substitution scheme, solving a system of n linear equations in n unknowns takes $O(n^3)$ time. With an iterative approach such as Gauss-Seidel, the computation time is reduced to $O(n^2)$.

[7] Recently, a new technique has been developed to calculate form factors using ray tracing[Wall89].

Because realistic scenes are elaborate, they often contain a large number of patches. A progressive method has also been developed [Cohe88] which performs partial propagation of the light emitted from the patch with the most power which has not been processed. This alleviates the storage problem, since form factors can be computed on the fly, with no more than n needed to propagate the undistributed power of one patch. It also permits progressive rendering of an image, for faster response under interactive display environments. Each iteration generates an image in $O(n)$ running time, and provides progressive refinement to the global illumination.

The third problem associated with radiosity lies in the great number of patches which have to be used when the gradients of light intensity are sharp. In transitional areas, such as penumbral shadows, the number of patches required per object becomes excessive. Instead of creating the additional patches necessary to a more accurate image, patches are used to estimate the distribution of power in the scene, and those with sharp luminosity gradients are subdivided into *elements*. Each patch is assumed to have a uniformly distributed power output, but the incoming power is only assumed constant over an element. For each element of the patch, the incoming power is calculated as if it replaced the entire patch in the original algorithm. This gives a more accurate distribution of the global power, without an increase in the number of patches. Therefore, a roughly correct global distribution of power is obtained, and the estimate of the power received by a patch is divided among its elements.

The fourth and principal drawback of the radiosity based methods is the inability to efficiently handle efficiently non-diffuse reflectors. Several attempts have been made at integrating specular reflection into radiosity algorithms, with only limited success. In [Imme86], a radiosity algorithm which takes into account such effects is introduced, but it turns out to be too computationally expensive for any practical application.

2.3.3 Hybrids and Others

Attempts have been made at efficiently increasing the range of environments that radiosity based algorithms can solve. In this section, others techniques which attempt to solve the global illumination problem are surveyed.

While classical ray-tracing handles the purely specular inter-reflections, and radiosity techniques handle the purely diffuse inter-reflections, they do not account for all of the global illumination. Various attempts have been made to handle both, by either improving or combining these two techniques.

Most have enjoyed only limited success, because in reality neither light reflection nor light itself can be categorized as specular or diffuse. Rather, these terms only apply to the two ad hoc components introduced to facilitate the expression of local illumination models. Some global illumination algorithms have inherited these attributes from the predominant component of the underlying local model, treating light as being transported either specularly or diffusely.

We first examine how each of these two paradigms can be extended to solve the entire global illumination problem.

2.3.3.1 Extended Ray-Tracing

In [Kaji86], Kajiya introduces the *rendering equation*, which relates energy received by a surface, in a given direction, to the energy emitted and transmitted in that direction by other elements of the scene. A solution to the equation is approximated using a more thorough sampling of the directions at each node of a ray-trace from the eye. For each pixel, a number of light *paths* — trees with a branching factor of one — are traced into the environment. At each node, information is gathered from the light sources, and from a subtree rooted at the node. Rather than sampling both the ideal reflection and refraction directions, a random sample direction is selected, weighted according to the local illumination model. In effect, a number of paths are traced into the environment for each pixel.

While in the limit this technique performs an integration of all lighting information in the scene and solves the entire global illumination problem, it can still suffer from insufficient sampling. Because only a small number of paths can be traced in a tractable solution, the choice of these paths is critical. To achieve a convincing level of realism, the "important" directions must be sampled at each node. Unfortunately, these directions cannot be known a priori, since a ray's contribution is not known until the global illumination is fully computed. The weighting factor for choosing the next branch from a node is based mostly on the local illumination model, since the global illumination has yet to be fully determined. Because of the intractability of determining ray contributions [Fium89], this theoretical solution, which is known to converge to the global illumination solution, is difficult to implement in an efficient fashion for all but the simplest scenes.

In [Ward88], Ward, Rubenstein and Clear introduce a ray-tracing solution for diffuse inter-reflections. Additional diffuse reflection information is calculated at the node level, extending the standard ray-tracing algorithm. An array of directions is recursively sampled about the intersection point, and converted to a single diffuse term via the local illumination model. For each corresponding intersection point in the ray-traced tree, a neighbourhood is calculated, in which this diffuse component is roughly constant. If a subsequent node in any pixel's ray-traced tree falls into this neighbourhood, the diffuse reflection component is obtained through interpolation.

Since only a view-dependent diffuse term and pure specular directions' information is used towards determining a shade, only ideally specular and diffuse surface can be handled. This technique also faces another problem: how can one define neighbourhoods over which diffuse reflection is the same? The authors have recognized that sharp gradients can go undetected, and categorize this as a sampling problem. The above drawbacks preclude the use of this algorithm as a viable solution to the global illumination problem.

2.3.3.2 Extended Radiosity

In [Imme86], Immel and Cohen present a modified radiosity algorithm which theoretically solves the global illumination problem for any environment. The major modification consists of defining form factors between discrete solid angles about the centre of each patch in the scene. For patches P_i and P_j in A_S, and for solid angles Ω_i of P_i, and Ω_j of P_j, the form factor $F_{\Omega_i \Omega_j}$ represents the fraction of power leaving a solid angle Ω_i centred at P_i and received through an Ω_j centred at P_j. With these form factors, a relationship similar to the standard radiosity equation is set up between the pairs of solid angles.

The need for an accurate solution forces the subdivision of the hemisphere of each patch into a large number of solid angles. This increases the number of form-factors by the square of the number of solid angles per patch. The number of equations and unknowns which have to be solved increases proportionally. In practice, this means that the solution time increases by several orders of magnitude, precluding all but simple environments from being processed in such a fashion. As is the case for Kajiya's rendering equation, this algorithm theoretically solves the global illumination problem for any environment, but is very expensive to compute.

2.3.3.3 Hybrids

Several attempts have been made at combining ray-tracing and radiosity techniques, in the hope of handling more of the general global illumination problem. Wallace, Cohen and Greenberg [Wall87] first suggest such an approach in their two pass rendering of a scene. Given an environment partitioned into a set A_U of patches of uniform radiosity, and a set A_S of purely specular patches, form factors are calculated between each pair. Additional indirect form factors are calculated between ideally diffuse patches which can "see" each other in purely specular (e.g., mirrors, windows) surfaces in A_S, creating virtual worlds. Once the radiosity algorithm has been applied using these special form factors, a view-dependent ray-trace is performed from the eye V, through the screen C. This amounts to standard ray-tracing where the diffuse reflection term is replaced by an interpolation of the results obtained in the first phase, and the specular terms are calculated over a discrete solid angle, as opposed to an infinitesimal one.

Notice that only purely specular surfaces can also be additionally handled in the new radiosity calculations. Surfaces which are neither purely specular nor purely diffuse cannot be handled. Compounding this limitation is the difficulty of calculating form factors between patches which see each other through multiple levels of indirection. This means that form factors can only account for a small number of indirect sub form factors, typically one or two. It is doubtful if this technique could be generalized to handle arbitrary surfaces.

One positive aspect of the ray-tracing phase used in this solution is that a frustum of directions centred about the purely specular directions are sampled, instead of the usual single infinitesimal angle. While this increases the probability of gathering all of the "important" light impinging on a patch, it does not properly solve the arbitrary global light distribution. Determining where the light is coming from cannot be known a priori, as this requires knowing global illumination, and hence importance sampling does not produce a complete solution, only an improved one.

Another attempt at combining ray-tracing and radiosity in a two pass process was presented by Devillers, Puech and Sillion [Devi87]. Two general equations similar to Kajiya's rendering equation are exhibited, one relating purely specular transmission, the other purely diffuse. In an initial pass, a ray-tracer is used to calculate form factors between diffuse patches, including multiple levels of indirection. This means that patches are allowed to "see" each other in the purely specular directions, and this produces more accurate form factors. Again, only ideal specular surfaces are properly handled. Once the radiosity method has been applied, a ray-tracer is used to determine the complete illumination. As in the preceding algorithm, the diffuse component is extracted from the radiosity phase, and the specular components from the ray-tracing one. While this solution is more general than [Wall87] regarding form factor calculations, it still only handles the two extreme types of surfaces.

A third attempt by Shao, Peng and Liang [Shao88] appears more promising. The hemi-cube technique [Cohe85] is repeatedly applied to an environment, in order to calculate the total energy distribution for each patch. In a first pass, all patches are assumed perfectly diffuse, to get an initial approximation of the energy distribution in the environment. In subsequent passes, form factors between patches P_i and P_j that are not perfectly diffuse are adjusted to take into consideration the current radiosity of each patch, and the specularity of each of them. The estimates of the energy leaving each patch are used to determine the amount of light that would fall on P_i and be transmitted to P_j. The authors are not clear as to whether each patch is treated as a diffuse emitter for the purpose of these calculations. By repeatedly solving for the radiosity of each patch with the updated form factors, and calculating new form factors, one hopes that this process converges to a stabilized system representing the true energy distribution.

If convergence is obtained then some global illumination value is determined for each patch. Associated with each patch is an amount of energy radiated by that patch, whose directional character is not clearly explained in the paper. To avoid prohibitive subdivision of the environment for the radiosity solution, the final illumination is determined through a variant on ray-tracing. This is done because directional information kept from the first phase is only accurate at the centre of each patch. The ray-tracing pass can then help in determining the specular component.

There have been other approaches to solving the global illumination, notably some involving wave propagation theory of light[Mora81, Baha87]. Like the hybrids previously discussed, they have not brought about the desired breakthrough in global illumination.

2.4 FIAT

The final section of this chapter will give an overview of the subject of this thesis, FIAT, a light-driven global illumination algorithm. The previously mentioned shortcomings of both ray-tracing and radiosity based solutions to the global illumination problem motivate the design of FIAT. By properly modelling the interaction of light between the various objects in a scene, the light flux values obtained represent complete steady-state global illumination and can be used to generate an image of the scene.

Given a scene S, a bounding volume $\mathcal{V}$ is defined such that all objects O of S are included, as well as those light sources that also act as objects[8]. The principal

[8]Point light sources, which typically represent sources such as the sun, are not included in the bounding volume, since their participation in the true global illumination is limited to emitting light. For instance, we can safely assume that any light reflected towards the sun will not be reflected by the sun and reach the viewer.

objective of FIAT is to model the interaction of the light emitted from the sources with the objects contained in $\mathcal{V}$. FIAT differs from traditional approaches in that the environment is adaptively partitioned into regions. This divide and conquer approach eases the task of modelling this interaction.

Consider a volume (cell) C within $\mathcal{V}$, with surface boundary $\mathcal{S}$. If we ignore the flux of light power crossing the boundary $\mathcal{S}$, we can separately process the illumination inside and outside of C. If instead we know the incoming flux crossing $\mathcal{S}$, and the light emitted within C, we can model the interaction of this light with the cell's contents independently of the contents of the remainder of $\mathcal{V}$. Once this has been done, the resulting outgoing flux crossing $\mathcal{S}$ can be used to process other cells in $\mathcal{V}$ independently of C. The linearity of all light effects guarantee that we can repeatedly solve the global illumination problem for any cell, until the entire global illumination problem has been resolved.

If the flux crossing the boundary between cells is maintained, cells can be processed independently of each other, and of order. The main thrust of FIAT is that the bounding volume $\mathcal{V}$ is subdivided into disjoint cells C_i such that the global illumination problem can easily be solved within each C_i. We refer to the act of solving the global illumination within a cell as *balancing* a cell, and that of exchanging the boundary flux information between cells as *propagation*. Given a scene S, FIAT proceeds to balance cells as outlined in the algorithm of figure 2.10

At each step k, there is a set of n_k disjoint cells $\{C_i\}$ which cover $\mathcal{V}$. At each of those steps, a cell is chosen and if we can compute the global illumination within it, we do so, and propagate exiting light to the appropriate neighbours. Otherwise, we subdivide the cell and replace the entry in the list by the newly created cells. This iteration process continues until all of the light power has been propagated. In practice, we can stop when the undistributed light, which consists of light at boundaries plus unprocessed emitted light, falls below some threshold τ. Once the light processing phase is complete[9], a window $\mathcal{W}$ and viewpoint V can be defined.

[9]Since we assume additivity of light, is is possible to predefine V and $\mathcal{W}$, and to collect illumination information as the light processing occurs.

```
Determine bounding volume C_0 = 𝒱
while( not done ){
        choose a cell C_i ∈ 𝒱
        if( can deal with C_i ){
                Propagate incoming and emitted light
                Update boundary of neighbouring cell
        } else {
                Split the cell C_i
        }
}
```

Figure 2.10: FIAT Algorithm

The light power gathered during the iteration, which is directed towards V and crosses $\mathcal{W}$, is used to define the image.

Chapter 3

FIAT

This chapter presents the main components of FIAT, explaining the following aspects of this algorithm:

- Cell structure
- Power balance.
- Deciding if a cell must be split.
- Dealing with a cell.
- Halting the iteration process.
- Obtaining an image.

3.1 Cell Structure

Many plausible methods for hierarchical space subdivision exist that could be used in FIAT. Due to the adaptive nature of FIAT, only schemes that could be applied adaptively could be considered. Many data structures exist to partition a scene, but such techniques as *point-octrees*, *k-d trees* [Same84] and *Binary Space Partitioning (BSP) trees* [Fuch80] are data-driven, and lead to adaptive division of the environment. Since the data in this case are the objects, we found it simpler to deal with space-driven data-structures, and chose to use *octrees*.

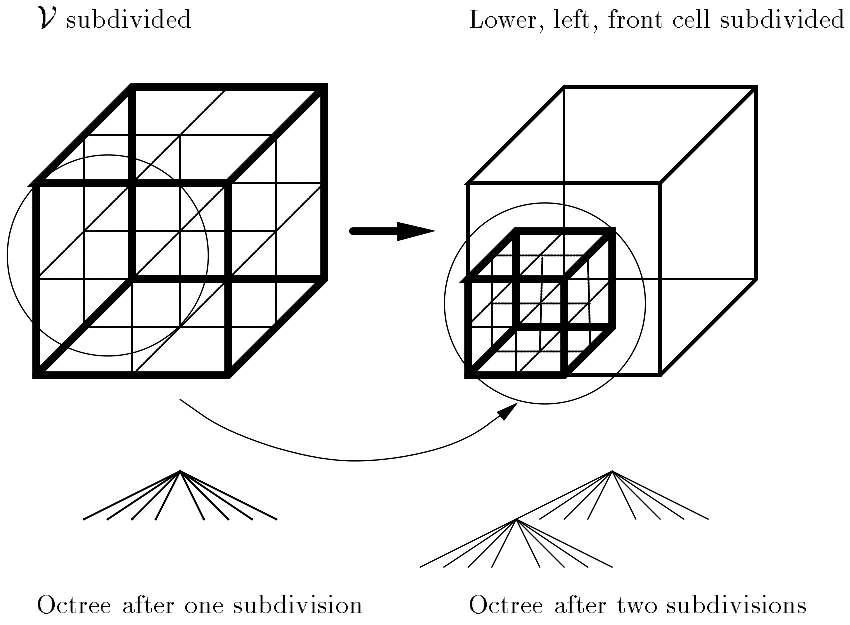


Figure 3.1: Space Subdivision Using Octrees

The octree space subdivision scheme is simple. Consider the bounding volume $\mathcal{V}$ for a given scene. If any iteration step of the FIAT algorithm requires that a cell C within $\mathcal{V}$ be subdivided, C is split into eight children of equal dimension. These eight children replace the parent cell C in the partition of $\mathcal{V}$ into cells.

We can construct a tree with a branching factor of eight that represents the subdivisions. The root of the tree represents a cell that corresponds to the entire bounding volume. Every node in the tree represents a cell within $\mathcal{V}$. If a node has any children, the corresponding cell has been split, and this node has eight children. The eight corresponding cells are disjoint, and form a partition of the parent cell. If a node is childless, it is called a *leaf*, and corresponds to a cell which has not been split. Figure 3.1 illustrates the cells within the bounding volume and associated octree, after one and two subdivisions.

We can associate with each iteration step of FIAT a *partition octree* defined according to the above rules. The set of all the cells that correspond to the leaf nodes form a partition of the bounding volume. At the start of each iteration step, the cell that is to be processed is chosen from this partition. This implies that once a cell is subdivided, it will never be again selected in an iteration step, although one or more of its children may be selected. [Note: It is conceivable to recombine subdivided cells if saving storage is sufficiently important.]

Notice that if the bounding volume is a cube, each one of its children, and, by extension, each one of its descendants, will also be a cube. To achieve this, we force the initial bounding volume to be a cube. In the remainder of this thesis, we will assume that cells are always cubic.

Since light information will be exchanged between adjacent cells, then for any cell, we must be able to determine the ones that are adjacent to it. We will ignore cells that are only adjacent through a corner, or an edge — because no power can be transmitted through an intersection of measure 0 — considering only those that share part or all of one face. Given a cell C at some iteration step k, we define its immediate neighbours as the set of all leaf cells — cells corresponding to leaves of the k^{th} partition octree — that share part or all of one face with one of C's face. Assuming that we process C at step k, the only cells whose flux information can be directly affected by a change in the flux information at C's boundary are precisely those in the immediate neighbour set.

Because of the regular nature of an octree, it is possible to determine the immediate neighbour set of a cell by examining the partition octree at the current step. Given a face f of a cell C, f is said to be a *pseudo-interior* face of C if f lies inside C's parent cell. Otherwise, f is a *pseudo-exterior* face of C, and f is part of one of the faces of the parent cell. All the faces of the cell corresponding to the bounding volume are defined to be pseudo-exterior.

Consider the chosen cell C in the partition octree of step k, and one of its faces f. If f is a pseudo-interior face of C, it coincides with the same pseudo-interior face of one of its siblings. If this sibling is a leaf cell, it is the only immediate neighbour of C on f. Otherwise, the partition sub-octree rooted at the sibling must be examined: the leaf cells that share one of their faces with f are all the immediate neighbours of C on f. If f is a pseudo-exterior face, we determine the first ancestor of C in the partition octree that does not contain f as part of one of its faces. If no such ancestor exists, f was a true exterior face: f lies on one of the faces of the bounding volume,

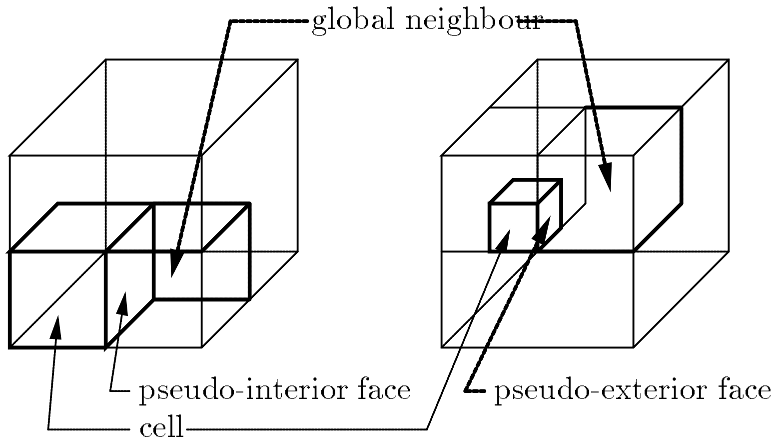


Figure 3.2: Examples of Global Neighbours

and hence C has no immediate neighbours on that face[1]. If such an ancestor is found, we examine the eight children of the ancestor cell, and choose the one that contains f on one of its pseudo-interior faces, but that does not contain C. The partition sub-octree rooted at this cell contains all the immediate neighbours of C. If this cell is a leaf cell, it is the only immediate neighbour. Otherwise, the set of immediate neighbours is the set of all leaf cells of the sub-octree that share one face with all or part of f.

As we can see, the set of immediate neighbours of C on f is found by examining the leaves of a partition sub-octree. We define the cell corresponding to the root of the sub-octree as the global neighbour, since it contains all of the immediate neighbours of C on f. The global neighbour cell is illustrated for a pseudo-interior and a pseudo-exterior face in figure 3.2.

This concept of global neighbour is useful, and it is easy to maintain. When a cell is subdivided, eight nodes are added as the children of the node for the subdivided cell. The global neighbour definition implies that these do not change for existing nodes. Hence we only have to determine a global neighbour once: this is done for each newly created node. The global neighbour on a pseudo-exterior face is inherited from the parent. The global neighbour of a pseudo-interior face is the sibling that shares this pseudo-interior face.

[1]This implies that the cell corresponding to the bounding volume has no immediate neighbours.

The immediate neighbours will be needed when light information is to be transferred from one cell to the other. Given a point of intersection on a face f of $\mathcal{C}$, we will find the global neighbour on that face, and traverse the sub-octree rooted there to find the immediate neighbours. If the global neighbour does not exist, this implies that the power leaves the environment, and it will not have to be further considered. Otherwise, this information is transferred to the cells which are immediate neighbours.

3.2 Power Balance

In any iterative algorithm it is important to achieve convergence in as few steps as possible. Since FIAT is a light propagation algorithm, convergence is achieved when all the light has been propagated through the environment, either by leaving it, by being absorbed, or both. Initially, the bounding volume is one cell, and there is either incoming light, internally emitted light, or both, that have to be processed. These define the incoming and emitted light of the bounding cell. At each step of the algorithm, a cell C is chosen, and if we can compute its internal global illumination, the energy entering C and emitted from within C is used to determine the cell's global illumination information. If the contents of a cell are simple enough, we apply a local illumination model to the objects within C, in conjunction with the incoming and emitted light, to determine how this power is transformed into light leaving the cell and being absorbed.

If we call C's incoming power P_I, the outgoing power P_O, the absorbed power P_A and the emitted power P_E, then the law of conservation of energy for a closed environment such as a cell, tells us that:

$$P_I + P_E = P_O + P_A. \tag{3.1}$$

The quantity $\Delta P = P_I + P_E - P_O - P_A$ represents the *power balance* of this cell.

For any FIAT iteration where a cell's global illumination is calculated, we must ensure that the local illumination model brings the power balance to zero. This implies that all the incoming and emitted energy must be transformed into outgoing and absorbed power. Since the power balance is always reduced to zero, we define the absorbed power implicitly, in terms of the other quantities.

Suppose that a cell C_O is balanced, and that it is adjacent to a cell C_I. Since C_O and C_I share a common boundary, the flux of light crossing it is identical for both, except that one cell's incoming power is the other's outgoing. To avoid duplicating information, each cell keeps track of only half of the common flux. Without loss of generality, each cell will only be responsible for the light entering it, but will have access — through its neighbour(s) — to the light leaving it.

The power balance is the simplest cell selection criterion. If a cell has a positive power balance, then there is a surplus of incoming and emitted light. If a cell has a negative power balance, then there is a surplus of outgoing and absorbed light. Since the ultimate goal of FIAT is to determine the global illumination information in order to produce an image of a scene, then on a cell basis, we only need to remember the complete flux information for both the light entering the cell, and that emitted inside. If the final power balance is complete, the only light that can illuminate an object within a cell comes from precisely these components.

Because of the linearity assumption, once a cell has been balanced, the current incoming and emitted power components correspond to light which has either already been propagated as outgoing power, or absorbed. We no longer need to consider that light as part of the power balance of the cell, but need to remember it so that it can be used towards calculating the global illumination. Doing so, we reset the four components of the power balance to zero.

Initially, the only power in the system is due to the light sources, and hence only P_E, P_I, or both, can be non-zero. At any stage, if a cell C is balanced, its ΔP is reset to 0, as are all the components. If the cell is not chosen, only its P_I can increase, and hence both the outgoing and absorbed power are not affected by this step. If a cell is split, the flux information is spread to the child(ren), and the incoming or emitted power components may be reduced, if the information has to be divided amongst more than one cell. This shows that the outgoing and absorbed power of a cell is always zero at the start of any step, and thus $\Delta P = P_I + P_E$, at the start of any step, and $\Delta P \geq 0$. Therefore the power balance of each cell is completely determined by P_I and P_E, and we redefine

$$\Delta P = P_I + P_E \tag{3.2}$$

for each cell. This measure of power balance will be the selection criterion for choosing a candidate cell for balancing.

3.3 Splitting a Cell

Now that we have decided how a cell is chosen, we need to be able to decide if we can deal with a cell, and how to subdivide it if we cannot. Ideally, we should only deal with cells for which there exists a simple solution to the global illumination problem[2]. The following types of cell are unarguably simple enough:

Empty Cell. The illumination information at the boundary $\mathcal{S}$ can be transferred through the cell to its neighbours.

Single Planar Object in Cell. A local illumination model can be used to compute the outgoing power, given the power at $\mathcal{S}$.

Single Opaque Convex Object in Cell. A local illumination model can be used to compute the outgoing power, given the power at $\mathcal{S}$.

Single Light Source in Cell. The emitted light can be transferred to the neighbouring cells, assuming the light source is not too complex. (e.g., planar)

In addition to these simple cases, some cells with combinations such as multiple co-planar objects, or multiple non-participating light sources[3] can also be easily balanced.

[2]An actual implementation may be forced to deal with non-simple cells due to system-imposed constraints on memory usage.

[3]Non-participating in the sense that the source emits light but is has not been assigned object properties.

The only proviso is that no light should encounter more than one object boundary on its way from its origin to a destination outside the cell. If this is not the case, we are faced with a complicated global illumination problem within a cell.

If a cell cannot be easily balanced, it should be subdivided. Ideally, the resulting cells would all be simple enough to be balanced. If such is not the case, a child may get subdivided repeatedly as it becomes a candidate for balancing. The current scheme subdivides the cell into eight children of equal dimension, using the previously explained octree structure.

3.4 Balancing a Cell

If a chosen cell does not need subdivision, it is balanced. We have to process both the incoming light, and the light emitted within the cell. Note that light is only emitted once as we only consider power, being only treated the first time we balance a cell. There may be new incoming light, but never any new emitted light. The following description of the treatment of an incoming ray is understood to apply also to light which must be emitted when we first process this cell.

For every incoming ray, we must determine how it interacts with the contents of the cell. If it encounters nothing, we simply propagate it to the correct neighbouring cell. If an object lies in its path, a local illumination model is used to determine the outgoing flux of light from the intersection point. A sphere of rays of light is then propagated from that point to the neighbours, with the appropriate boundary being updated. Because no ray in the chosen cell can reflect/refract more than once (due to our selection criterion), the cell's global illumination information is correctly determined using a local illumination model.

3.5 Halting

The iteration process repeatedly selects cells, balances them, and propagates the light power to neighbouring cells. As light is traced through the environment, we

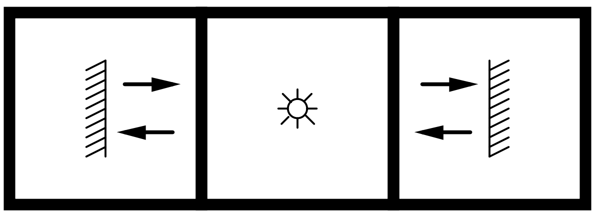


Figure 3.3: Example of a cycle

must ensure that this eventually accounts for most of the light entering the initial bounding volume. If we can guarantee that the total unprocessed power drops below a certain threshold τ, then we will be able to account for the proper distribution of most of the global illumination. We now introduce sufficient terminating conditions.

Let

$$\Delta E_k = \sum_{i=1}^{n_k} \Delta P_i \tag{3.3}$$

be defined as the undistributed power of the system at step k. If we use the undistributed power at iteration k as a stopping criterion, FIAT will terminate when $\Delta\ E_k\ <\ \tau$.

We must ensure that this condition is eventually reached. In particular, we wish to avoid scenes in which ΔE_k does not decrease to 0, which will happen if cycles with no damping are present in the scene. Consider the case where two facing parallel perfect mirrors and a non-participating (as an object) light source are contained in contiguous cells such as in figure 3.3. In such a pathological case, the impulse of light will indefinitely propagate back and forth between the two mirrors.

One simple method of lowering the probability of cycles occurring, is by imposing the following constraint:

Halting Condition 1 *For any non-empty balanced cell,* $P_A > 0$

This condition should result in all of the light power initially entering the system being eventually dissipated. Since most, if not all, real objects modelled absorb some power, this is quite a reasonable constraint to impose on the system. In practice, after each iteration, $\Delta E_{k+1} < \Delta E_k$ and this should eventually result in convergence for some $N \in \mathbb{N}$ such that $\Delta E_N < \tau$.

From a mathematical point of view, it is still possible that $\lim_{k\to\infty} \Delta E_k > \tau$, even if we impose halting condition 1. For instance, the sequence

$$2,\ 1\frac{1}{2},\ 1\frac{1}{3}, \ldots, 1\frac{1}{k}, \ldots$$

is monotonically decreasing, but does not converge to 0. We can circumvent this possibility by enforcing the following stronger halting condition:

Halting Condition 2 *There exists a $\delta > 0$ such that at any iteration stage k, P_{A_k}, the power absorbed by a non-empty balanced cell $\mathcal{C}_{b_k}$ is such that $P_{A_k} > \delta\, \Delta P_{b_k}$.*

Roughly speaking, this says that at least a fraction δ of the incoming power is dissipated after the cell has been balanced.

Proposition 1 $\Delta E_k \to 0$ *as* $k \to \infty$.

Proof: Let n_k be the number of cells at stage k. Since cell b_k was chosen at step k, then $\Delta P_{b_k} \geq \Delta P_i$, for all i. Now

$$\Delta E_k = \sum_i \Delta P_i \leq n_k \Delta P_{b_k}$$

implies that

$$-\Delta P_i \leq \frac{-\Delta E_k}{n_k}.$$

After cell b_k has been balanced, the remaining power in the system is

$$\Delta E_{k+1} = \Delta E_k - P_{A_k},$$

since the absorbed power, P_A, has been dissipated. By halting condition 2, $P_{A_k} > \delta \Delta P_{b_k}$, therefore

$$\begin{aligned}
\Delta E_{k+1} &< \Delta E_k - \delta \Delta P_{b_k} \\
&\leq \Delta E_k - \delta \tfrac{1-\Delta E_k}{n_k} \\
&= \Delta E_k \left(1 - \tfrac{\delta}{n_k}\right).
\end{aligned}$$

If we assume that n_k is bounded above, say by $\mathcal{N}$, then $1 - \frac{\delta}{n_k} < 1 - \frac{\delta}{\mathcal{N}}$ and hence, for all k,

$$\begin{aligned} \Delta E_{k+1} &< \Delta E_0 \prod_{i=0}^{k} \left(1 - \frac{\delta}{n_k}\right) \\ &< \Delta E_0 \left(1 - \frac{\delta}{\mathcal{N}}\right)^k \end{aligned}$$

and hence $\Delta E_k \to 0$ as $k \to \infty$. ■

In practice, halting condition 1 will be sufficient for FIAT to halt, but condition 2 will guarantee termination since in any implementation, the bound $\mathcal{N}$ on the number of cells will exist.

3.6 Computing an Image

Once the iteration phase of FIAT has been completed, all the power has been traced through the environment. Only a certain threshold τ remains unaccounted, and if this quantity is negligible, FIAT has in fact approximated the complete global illumination. By keeping all the lighting information of the participating elements of the scene, we can generate images of the environment with correct global illumination.

After having established a viewpoint V and window $\mathcal{W}$, then for each pixel $P_{ij} \in \mathcal{W}$, a number of infinitesimal solid angles can be sampled, from the eye through the pixel. At each intersection point, the proper infinitesimal solid angle of the infinitesimal surface element centred on the intersection point is used to obtain the correct power impinging on the pixel. These values, which will still suffer from the problems which are associated with any reconstruction process based on sampling, are then used to compute the proper shade of the pixel.

As will be seen in chapter 7, it is sometimes not practical to keep all the lighting information, but to impose view dependency to reduce storage requirements. Since we have assumed the additivity of light, it is possible to incrementally compute the power crossing each pixel in the direction of the eye. Given a cell and a pixel, it is possible to determine if the sample point(s) for the pixel lie within the cell. Every time a cell is balanced, the light entering the cell is used in conjunction with the contents of the cell to determine the light crossing a pixel for each sample point within this cell. The information associated with the incoming light can then be discarded, thus reducing the storage requirements.

Chapter 4

Light Propagation

Any attempt to implement FIAT hinges on the design of a reliable light propagation model. This is the subject of this chapter, as we focus on the different modalities of this light driven algorithm. We discuss both methods of representing the flow of light at the cell boundary, and the type of beams that will be used to model the propagation of this flow between cells.

4.1 Boundary Information

Since generally we cannot possibly keep track of a possibly uncountable set of light rays traversing a face of a cell, we need a method of approximating this flow of light. Our approach consists of subdividing each face of a cell into small square elements, called *cexels*[1].

Before we discuss how we model the flow of light at the cexel level, we examine two different definitions of a cexel: a fixed portion of the area of a face, or a square of a fixed size.

[1]The term is a misnomer, being derived from the combination of section of cell, and can also be written as sexell. For historical reasons (i.e., too much work to change it everywhere!) the spelling cexel is retained in this thesis.

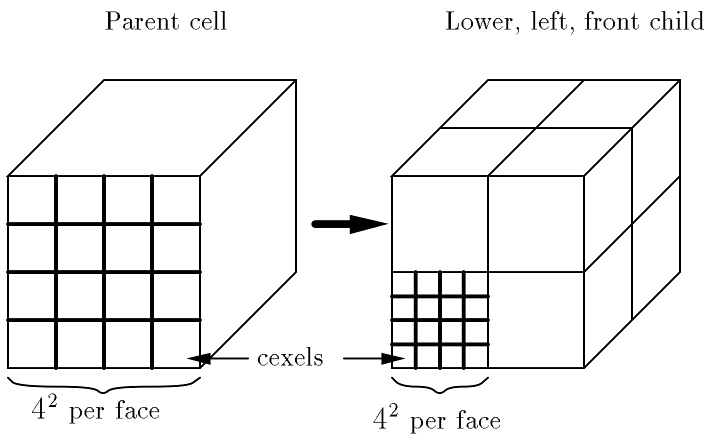


Figure 4.1: Fixed Number of Cexels per Face

4.1.1 Fixed Number per Face

If a cexel is defined as a fixed portion of the face of a cell then a fixed number of cexels cover the face of any cell. This also implies that the definition of a cexel is dependent of the cell to which it belongs. For example, suppose that we want 16 cexels to cover the face of a cell. Figure 4.1 illustrates what happens to the size of a cexel when a cell is subdivided, and new cexels are defined for its children.

Because of the adaptive nature of FIAT, it is probable that cells of different a size will be adjacent, during some iteration(s). If one of these cells is balanced during such an iteration, we may have to update the flux information at a boundary between cells of a different size. If this information is kept in cexels, it will be transferred between cexels of a different size. If any cell is subdivided, the parent's cexel information must be transferred from its faces to the pseudo-exterior faces of the children. Again, this involves transferring information between cexels of a different size. These practical difficulties prompted us to adopt a cell independent definition of a cexel where these problems do not occur.

4.1.2 Fixed Size

To avoid the problem of cexel information transfer between cexels of a different size, we fix the dimensions of a cexel. As a result of subdivision, children cells can now

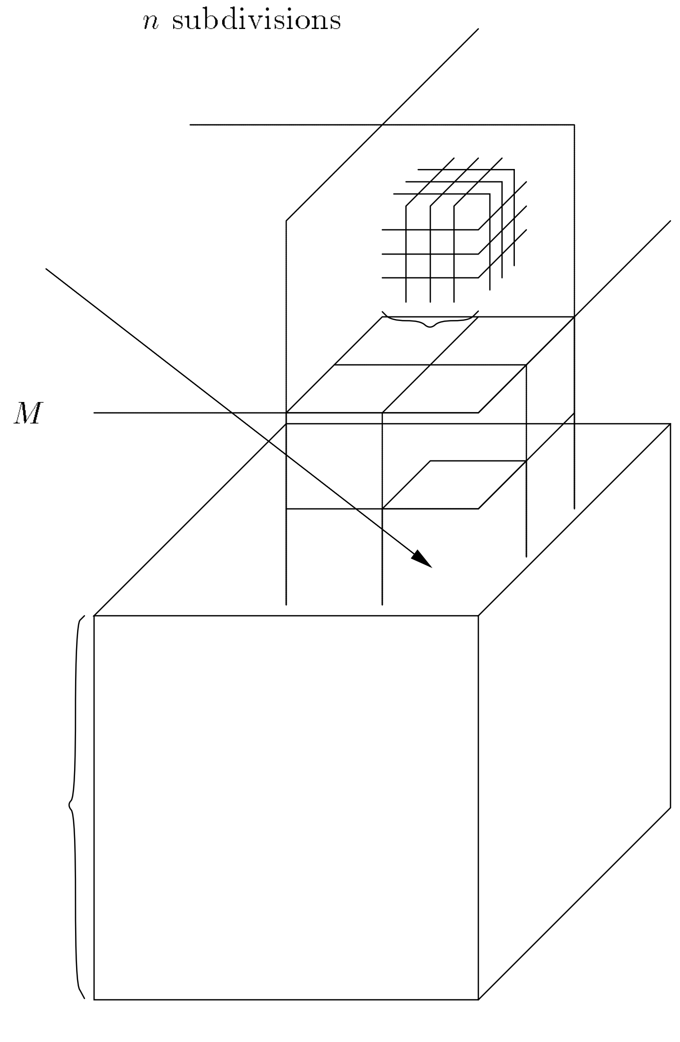


Figure 4.2: Definition of a Fixed Size Cexel

directly inherit the cexels of the parent cell. To simplify subdivision, we also want to ensure that the face of an arbitrary cell can always be exactly covered by an integral number of cexels. This will be the case if the side length of a face is always an integral multiple of the side length of a cexel. If such is the case, the face of a cell will never be smaller than a cexel.

Formally, suppose n is the maximum depth allowed for the partition octree, and that M is the length of the side of the bounding volume $\mathcal{V}$. These assumptions imply that n subdivisions are possible, and that the smallest possible cell will have a side length of $\frac{M}{2^n}$. Define a parameter m such that m^2 cexels exactly cover the smallest possible cell. Define the constant size of a cexel (CS) as the size of one of the cexels used to cover the smallest cell:

$$CS = \frac{M}{m \cdot 2^n}. \tag{4.1}$$

This is illustrated in figure 4.2

Given any cell in the octree, we can now cover it with an exact number of cexels of size CS. Specifically, if cell $\mathcal{C}$ is the result of $0 \leq k \leq n$ subdivisions of $\mathcal{V}$, then each face of $\mathcal{C}$ will be a square of side length $M/2^k$. Thus an M' by M' tiling of cexels

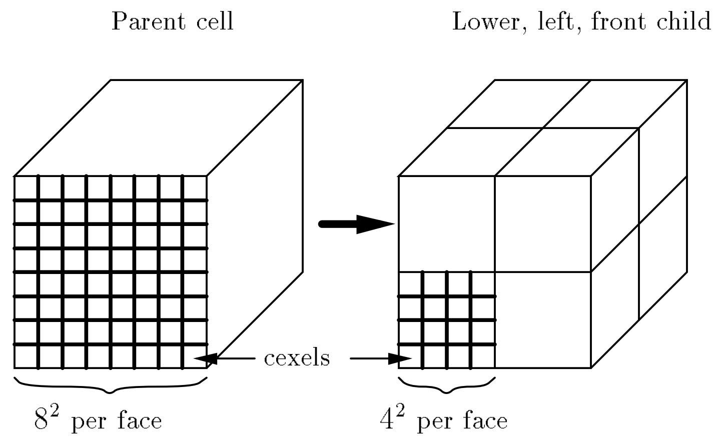


Figure 4.3: Fixed Number of Cexels per Face

will cover $\mathcal{C}$ provided that there exists an $M' \in \mathbb{N}$ such that:

$$M' \cdot CS = \frac{M}{2^k}. \tag{4.2}$$

Since $CS = M/m2^n$, then

$$\begin{aligned} M' &= \frac{M}{CS\, 2^k} = \frac{M\, m\, 2^n}{M\, 2^k} = m\frac{2^n}{2^k} \\ &= m\, 2^{n-k} \end{aligned} \tag{4.3}$$

and therefore $M' \in \mathbb{N}$, since $n - k \geq 0$.

Under this definition of a cexel, each cell can now be tiled with these cexels, in order to represent the light flux at the boundary. Because cexels are of a fixed size, the information associated with the cexels of a parent cell can be directly transferred to the corresponding cexels of the children. Figure 4.3 illustrates the use of fixed size cexels. The parameters m and n can be used to control the level of detail that we wish to keep at the surface boundary. We will assume from now on that we are dealing with fixed size cexels.

If we assume that each cexel contains the same amount of information, the parameters m and n can be tailored to the storage capacity of the target implementation machine. The upper bound for the cexels' storage requirement is proportional to the maximal number of cexels. Since n levels of subdivision are possible, the maximum number of cells is $(2^n)^3$. Since each cell has 6 faces, the m^2 cexels cover the face of the smallest possible cell, the maximum number of cexels used is:

$$6 \cdot 2^{3n} \cdot m^2. \tag{4.4}$$

A natural upper bound on the maximum number of cexels is the quotient of the storage capacity and of the storage requirement of one cexel. Note that in practice, not all cexels will be present at the same time.

4.2 Modelling Light Propagation

We now have to decide on some representation scheme for the flux crossing a cell. Since flux information will be kept in fixed-size cexels, we will define such a scheme at the cexel level. Ignoring such problems as phase and frequency[2] of light, it is important to preserve the power and the spatial distribution of the flux of light crossing a cexel. Since outgoing power for a cexel can be represented as the incoming power for the corresponding cexel of a neighbouring cell, it is sufficient to only consider the incoming power for each cexel. Power can be represented easily, as it is a scalar quantity. The problem then consists of determining the directions in which the incoming flux will be sampled. This is a difficult optimization problem, as we try to preserve the directional character of light, to control the spread of light, and to maintain a reasonable storage cost.

Each cexel consists of a collection of infinitesimal surface elements, which can receive incoming power in a hemisphere of infinitesimal solid angles, as illustrated in figure 4.4.

Since it is not possible to precisely represent all the arbitrary fluxes that can cross a cexel, we must represent a flux using a finite number of values. It is obviously impossible to represent arbitrary signals — in this case three dimensional fluxes —

[2]Information can be stored independently for selected frequencies of light, in identical structures.

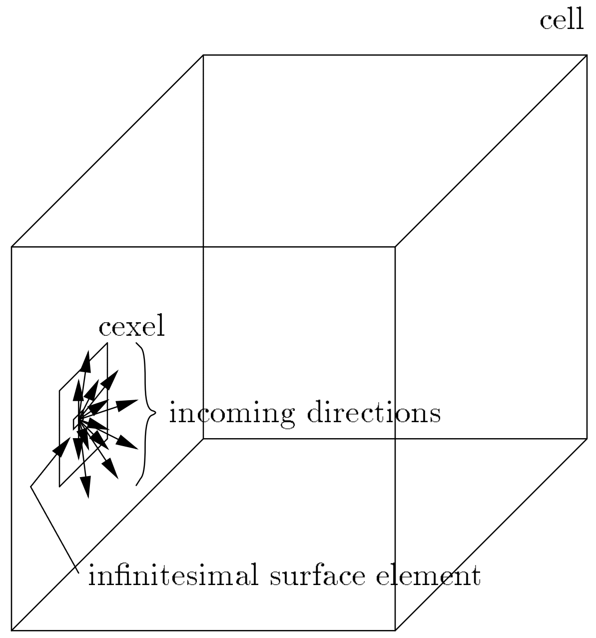


Figure 4.4: Hemisphere of Incoming Directions for a Surface Element

using finite sets of samples.[Pavl88] Therefore the information stored at the cexel level is only a representation, and in most cases, an approximation to the flux.

To illustrate the impossibility of reconstructing an arbitrary flux from a finite number of samples, consider the following illustration of the sampling theorem. Since only a finite number of samples are used to approximate a flux, there must exist two different fluxes f_1 and f_2 such that f_1 and f_2 map to the same representation f_R. When we reconstruct a flux from f_R, we will get the same answer for f_1 and f_2 since they both map to f_R. Therefore at least one of these arbitrary fluxes will be incorrectly reconstructed.

For each cexel, we partition the hemisphere of incoming directions into a set of solid angles, each of which has an associated representative direction, as illustrated in figure 4.5. For a cexel $\mathcal{C}$, the set of all such solid angles is $\mathcal{S}(\mathcal{C})$, and the set of the corresponding representative directions is $\mathcal{D}(\mathcal{C})$. The problem we wish to address is how we can propagate light through empty space, with minimum distortion, using only a discrete direction representation at the cexel level.

We now proceed to examine different propagation mechanisms and use of cexel's set of solid angles. The main goals we wish to achieve are:

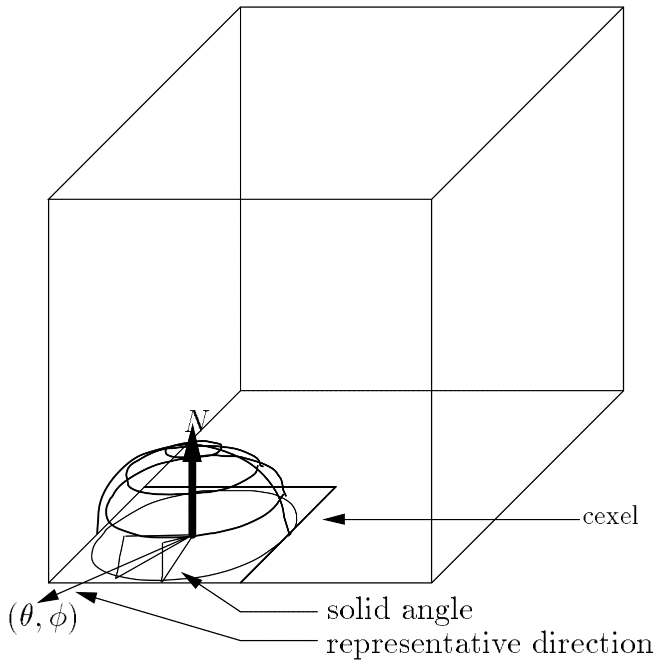


Figure 4.5: Hemi-Sphere of Directions — Sample Solid Angle

1. Conservation of power
2. Linear propagation of light
3. Conserving radial exitance

Notice that the first goal is essential, and the last two are also important but are conflicting. For each of rays, frustum, and parallel beams, we will examine one or more propagation mechanisms, and discuss their properties. The following is a synopsis of this discussion:

Rays
- Infinitesimally thin beam of light
- Can be propagated in exact straight line
- Good model for directional light effects

Frustums
- Solid angles of light with constant radial exitance
- More difficult to propagate correctly
- Better model for diffuse light effects

Parallel Beams
- Beams of uniform brightness, with constant cross-section.

- Can be propagated in exact straight line
- Good model for directional light effects
- Two possible types of cross-section:
  - Cexel-dependent
  - Cexel-independent

Parallel beams are used in the current version of FIAT because they are easy to implement. A cexel-dependent cross-section is used, since it is the simplest to define. The choice of cross-section is only relevant when shading computations are done for the final image, since both propagation and local illumination are approximated by the behaviour of a beam's representative ray.

4.2.1 Rays

Rays of light correspond to infinitesimally-thin beams of light, each carrying an infinitesimal amount of power. Given a partition of the incoming hemisphere of a cexel into its set of solid angles $\mathcal{S}(\mathcal{C})$ and representative directions $\mathcal{D}(\mathcal{C})$, we associate with each representative direction d a ray in this direction. The power contained in this ray will be a function of all incoming rays which are enveloped by the solid angle. Thus the flux at each cexel will be approximated using a finite collection of rays in the representative directions.

Because of discretization, we know that we can only approximate a flux. Given a flux approximated by a set of cexel rays in $\mathcal{D}(\mathcal{C})$, we will propagate the flux by propagating each ray in a straight line. This is the most we can accomplish, in the sense that any additional information concerning the flux represented at the cexel is no longer available. This mechanism will then behave correctly for a flux which can be expressed as a combination of the available rays in $\mathcal{D}(\mathcal{C})$. Hence, the mechanism will tend to preserve the directional aspect of light, but will at the same time reduce non-directional light flows to a finite collection of directional ones. For instance, narrow beams of light in a direction $d \in \mathcal{D}(\mathcal{C})$ will correctly be propagated, but light

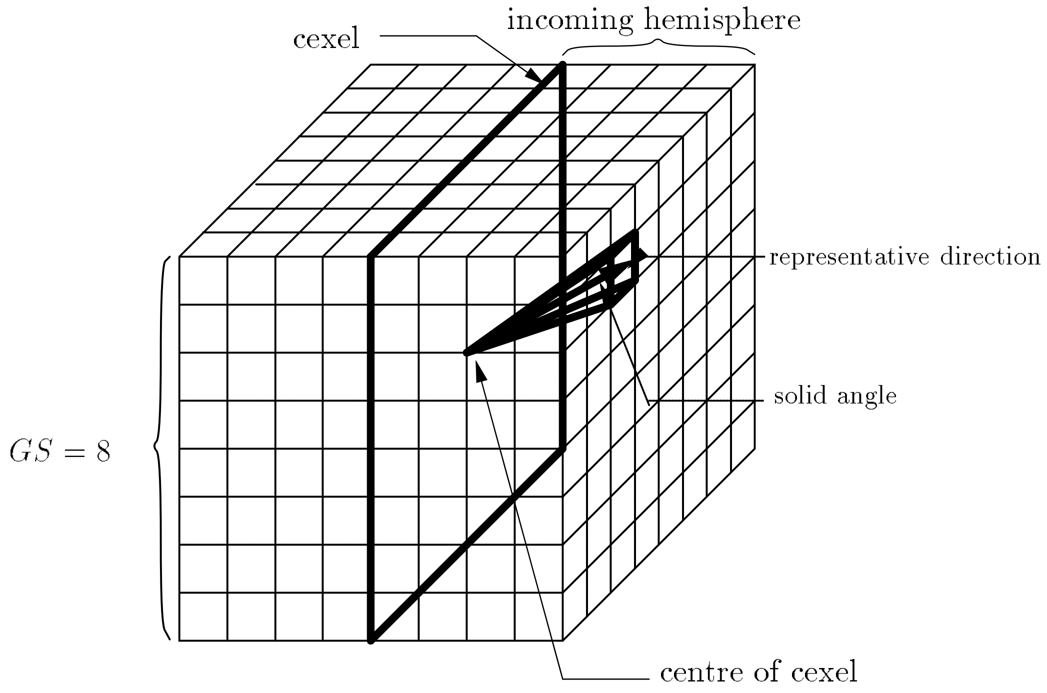


Figure 4.6: Global Array of Directions

from a diffuse light source will only be approximated as a (potentially large) set of directional beams.

We now define a partition of hemisphere into solid angles, as well as a linear propagation mechanism, which will allow us to propagate representative rays across cells in a straight line.

4.2.1.1 Hemisphere Partition

Consider a cexel $\mathcal{C}$ and define a cube centred about $\mathcal{C}$, with each face of the cube the same size as the cexel, as illustrated in figure 4.6. Notice that the cexel splits the cube into two identical parallelepipeds, and is parallel to two of the faces.

A parameter GS can be used to control the amount of information kept at each cexel. On each face of the cube, draw a grid of size GS by GS, giving rise to GS^2 squares per face. For each square, consider the solid angle delimited by a frustum whose base is the square and apex the centre of the cexel. These solid angles partition the entire sphere of directions about the cexel into a set $\mathcal{S}$ of solid angles associated with the squares. The representative direction for each solid angle is the vector defined by the centre of the cexel and the centre of the square.

Since we are only interested in the hemisphere of incoming light, we will only consider the solid angles which lie within the half of the cube that lies inside the cell. These solid angles define the partition $\mathcal{S}(\mathcal{C})$. Since the grid covers both halves of a cube covering the incoming and outgoing hemisphere of a cexel, the squares that define the solid angles will individually lie in only one half of the cube if and only if the grid size is even. To simplify the partition of the incoming hemisphere into a set of solid angles, we will assume that the grid size is even.

For a given grid size GS, the total number of base squares is $6GS^2$, and hence there are $3GS^2$ incoming representative directions to be kept for each cexel. Since $3GS^2$ is a measure of the level of information kept at the cexel level, it is evident that the parameter GS can be used to control this level.

For a given cexel $\mathcal{C}$, we can define the hemisphere of directions $\mathcal{D}(\mathcal{C})$ as

$$\begin{aligned}\mathcal{D}(\mathcal{C}) \quad = \quad & \{(x, y, z): \text{ two of } x, y, z \text{ have magnitude } 2i+1, 2i'+1, \\ & \text{where } 0 \leq i, i' \leq \frac{GS}{2}, \text{ one has magnitude } GS, \\ & \text{and } (x, y, z) \text{ represents an incoming direction}\}. \end{aligned} \tag{4.5}$$

4.2.1.2 Straight Line Propagation

We now introduce a method of propagation which allows straight-line propagation in similar fashion to Bresenham's line drawing algorithm. This deals only with the case of a ray travelling in a straight line and traversing a cell without intersecting an object or a medium.

Let a cell C, a ray r and cexels $\mathcal{C}_{IN}$ and $\mathcal{C}_{OUT}$ be such that r enters C through $\mathcal{C}_{IN}$ and exits through $\mathcal{C}_{OUT}$, actually entering the next cell through $\mathcal{C}_{OUT}$. Assume that r enters $\mathcal{C}_{IN}$ through one of the representative directions, say $(x, y, z) \in \mathcal{D}(\mathcal{C}_{IN})$. [Note: Three coordinates are used to represent a two-dimensional space of directions for simplicity, even though exactly one of x, y, z must have an absolute value GS.] This is illustrated in figure 4.7

In order for r to travel in a perfectly straight line, it is both sufficient and necessary

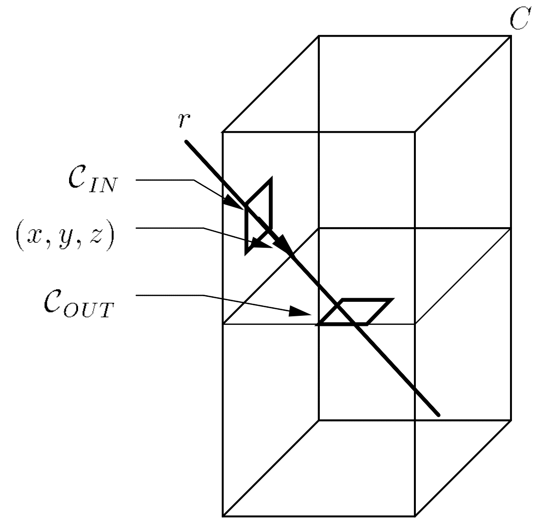


Figure 4.7: Ray Traversing a Cell

to satisfy the following two conditions:

1. $(x, y, z) \in \mathcal{D}(\mathcal{C}_{OUT})$.
2. The entry point of r into $\mathcal{C}_{OUT}$ must be the same as the point from which it will leave $\mathcal{C}_{OUT}$ when r is next propagated.

Matching Representative Directions. This condition is guaranteed by the definition of each $\mathcal{D}(\mathcal{C})$.

Proof: Since $(x, y, z) \in \mathcal{D}(\mathcal{C}_{IN})$, then x, y, z are such that two of them have a magnitude that satisfy $2i + 1, 2i' + 1$, where $0 \leq i, i' \leq \frac{GS}{2}$, and one of them has magnitude GS. Since this vector also penetrates $\mathcal{C}_{OUT}$, it is also part of its incoming hemisphere. The two preceding statements satisfy the definition of $\mathcal{D}(\mathcal{C}_{OUT})$, and hence $(x, y, z) \in \mathcal{D}(\mathcal{C}_{OUT})$. ■

Entrance and Exit Points. Suppose that rays are only allowed to enter and leave a cexel through their centre. This implies that any ray leaving a cexel, and travelling in one of the representative directions must intersect a cexel exactly at the centre. Given any representative direction, we must ensure that a ray that leaves *any* cexel in this direction enters the next cexel through the centre.

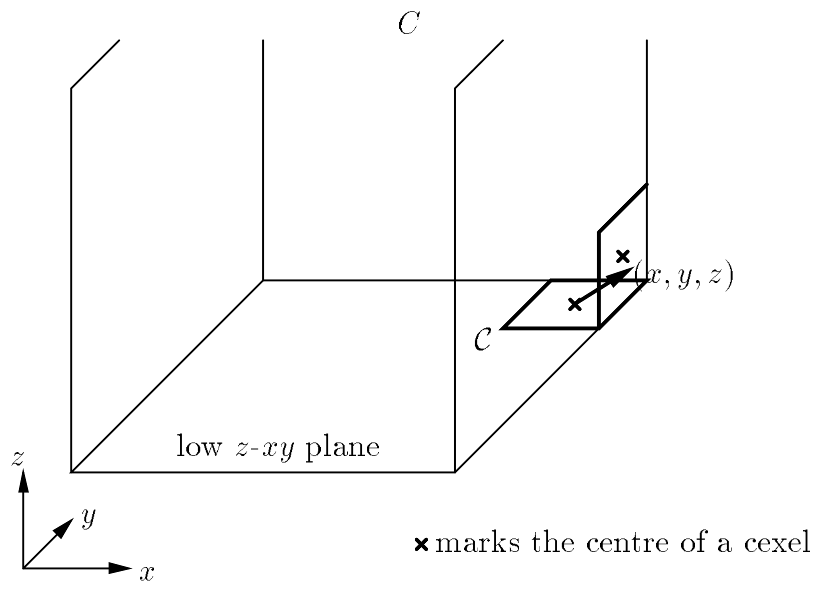


Figure 4.8: Cexel To Cexel Vector

Because all the representative directions must be present in all the $\mathcal{D}(\mathcal{C})$, this severely limits the number of representative directions that can be used.

Consider one representative direction (x, y, z) in a cexel's set of representative directions. Without loss of generality, assume that x, y, and z are relatively prime, positive, and that x has the largest magnitude. Consider a ray r leaving a cexel $\mathcal{C}$ such that $\mathcal{C}$ lies on the xy face of the cell C that has the smallest z coordinate, as illustrated in figure 4.8. In addition, suppose that $\mathcal{C}$ has the greatest x and y coordinates of all the cexels on that face. A ray leaving the centre of $\mathcal{C}$ in the (x, y, z) direction cannot intersect the next cexel in the centre. Since x has the largest magnitude, the next cexel intersected is adjacent to $\mathcal{C}$ but on the yz face. If both y and z are less than x, then as illustrated in figure 4.8, the point of intersection cannot be at the centre. The point of intersection will only be at the centre if precisely one of y and z has the same magnitude as x, and the other has magnitude 0.

What this illustrates is that only vectors with x, y, and z coordinates of magnitude 0 or 1 — and only some of those combinations — can leave a cexel at the centre and always enter the next one through the centre. For any other

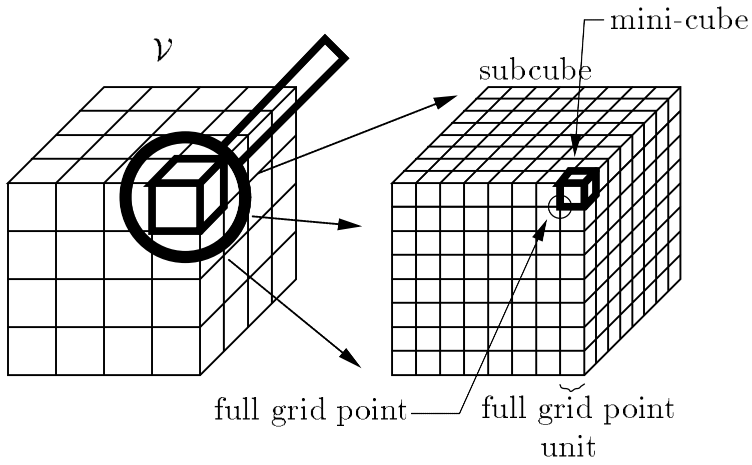


Figure 4.9: Full Grid Points

direction, one can construct an example where a ray leaving in that direction from a certain cexel cannot enter the next cexel through the centre.

Because of the small number of directions that can be used for centre-to-centre propagation, we must enhance our method of propagation. Specifically, if we can describe the entrance point of a ray into a cexel, say as an offset from the centre of a cexel, we will be able to use directions which do not always enter and exit through the centre of the cexel. If we can keep a compact and exact description of this offset, we will be able to propagate rays in exact straight lines.

Before we introduce the offsets that will be used, we need to define the concept of a *full grid point.* Given that a grid size GS has been chosen for determining representative directions, we define a full three-dimensional lattice of points within the bounding volume $\mathcal{V}$. As each face of $\mathcal{V}$ can be tiled with GS^2 cexels, we can partition it into subcubes whose face is the same size as a cexel. We then divide each subcube into $(2GS)^3$ mini-cubes of identical size, with each corner of a mini-cube corresponding to a full grid point. The set of points thus obtained constitutes a three-dimensional lattice of evenly-spaced points spread uniformly through $\mathcal{V}$, as illustrated in figure 4.9, for a grid size of 4.

Since the centre of a cexel falls on a full grid point, any ray originating from the centre of a cexel passes through a full grid point. The following claim will show that any ray originating from a full grid point can be propagated in a straight line. This will require keeping the rays coincidental by recording an offset from the centre of the intersected cexel. This offset describes a full grid point in the neighbourhood of the intersected cexel, through which the ray passes.

Lemma 1 *Let a ray r originate at a full grid point, and be oriented in one of the representative directions. Let r intersect some cexel $\mathcal{C}$, whose centre has full grid coordinates $(x_{\mathcal{C}}, y_{\mathcal{C}}, z_{\mathcal{C}})$. Then there exists a full grid point (x_f, y_f, z_f) which lies exactly on r's path, and such that $|x_{\mathcal{C}} - x_f| < 2GS$, $|y_{\mathcal{C}} - xyf| < 2GS$ and $|z_{\mathcal{C}} - z_f| < 2GS$, where distance is measured in terms of full grid points, which corresponds to the side-length of a mini-cube.*

If we can prove lemma 1, then any ray originating from a full grid point and intersecting a cexel can be propagated in a straight line if we record the offset $(\delta_x, \delta_y, \delta_z)$ in the range $\{-2GS, \ldots, 2GS\}$ such that:

$$\begin{aligned} \delta_x &= x_f - x_{\mathcal{C}} \\ \delta_y &= y_f - y_{\mathcal{C}} \\ \delta_z &= z_f - z_{\mathcal{C}}. \end{aligned}$$

The offset, together with the centre of the intersected cexel, constitute sufficient information to allow collinearity, by defining a point on the original ray's path. Thus the ray can be propagated from the intersected cexel, not only in the correct direction, but also in coincidental fashion with respect to the intersecting ray, guaranteeing straight line propagation. Since we know a full grid point that lies on the path of the ray, we can propagate the ray so that it will leave the intersected cexel from this point, and not the centre of the cexel. If no offsets were kept, either the number of representative directions would be severely restricted, or propagation errors would occur as rays would not necessarily enter cexels at the centre, but would leave from there.

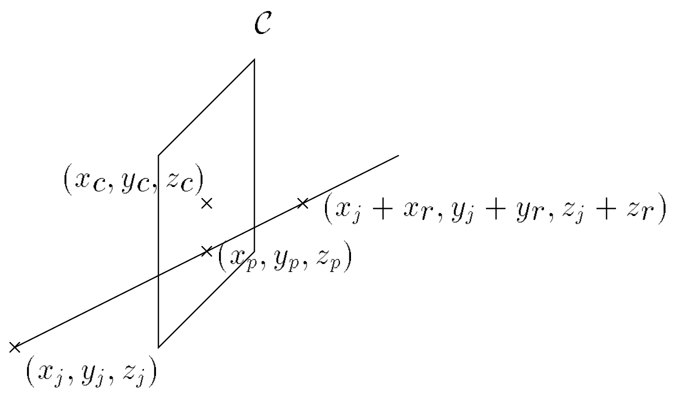


Figure 4.10: Offset From Centre to Intersection Point

Proof: Since r travels in one of the representative directions, there exists an $(x_r, y_r, z_r) \in \mathcal{D}(\mathcal{C})$ such that (x_r, y_r, z_r) is in the same direction as r. Since r originates on a full grid point, say (x_0, y_0, z_0), then r crosses another full grid point at $(x_0+x_r, y_0+y_r, z_0+z_r)$. In general, if (x_i, y_i, z_i) is a full grid point on r's path, then so is $(x_i+x_r, y_i+y_r, z_i+z_r)$.

Let r cross $\mathcal{C}$ at a point (x_p, y_p, z_p), such that this point is on the cexel, where each coordinate lies on or between full grid points. Since $(x_{\mathcal{C}}, y_{\mathcal{C}}, z_{\mathcal{C}})$ is the centre of the cexel, then in full grid point units, $|x_p - x_{\mathcal{C}}| \leq GS$, $|y_p - y_{\mathcal{C}}| \leq GS$, and $|z_p - z_{\mathcal{C}}| \leq GS$. This follows since a cexel is a square of side $2GS$ in terms of full grid point units, and no coordinate can be further away from the centre than a distance of GS units, as illustrated in figure 4.10

Let (x_j, y_j, z_j) be the full grid point closest to (x_p, y_p, z_p) on r's path, prior to r's intersection with $\mathcal{C}$. Since $(x_j+x_r, y_j+y_r, z_j+z_r)$ is also a full grid point on r's path, it must lie on the other side of $\mathcal{C}$. This implies that x_p is between x_j and $x_j + x_r$, where $x_r \leq GS$, and similarly for y and z. Thus for the coordinate x,

$$\begin{aligned} |x_{\mathcal{C}} - x_j| &= |x_{\mathcal{C}} - x_p + x_p - x_j| \\ &\leq |x_{\mathcal{C}} - x_p| + |x_p - x_j| \\ &\leq GS + GS = 2GS, \end{aligned} \tag{4.6}$$

This also holds for y and z. Hence the claim is valid. ■

Claim 1 *If each cexel $\mathcal{C}$ maintains a set of sample directions $\mathcal{D}(\mathcal{C})$ and an offset triple*

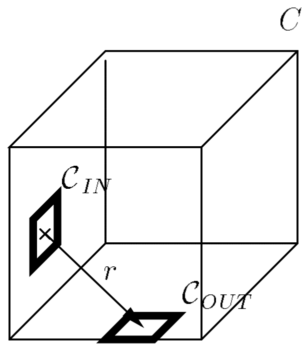


Figure 4.11: Straight Line Propagation

in $\{-2GS,\ldots,2GS\}^3$ *for each member of* $\mathcal{D}(\mathcal{C})$*, then exact straight line propagation can be achieved.*

Proof: Let a ray r originate from a cexel $\mathcal{C}_{IN}$, traverse cell C, and enter the neighbouring cell through cexel $\mathcal{C}_{OUT}$, as illustrated in figure 4.11 Since r originates at $\mathcal{C}_{IN}$, it does so from a full grid point, namely the centre of the cexel. By lemma 1, a triple offset in $\{-2GS,\ldots,2GS\}^3$ is sufficient to reconstruct a full grid point lying directly on r's path, before it intersects $\mathcal{C}_{OUT}$. Therefore, we can traverse one cell and preserve the straightness of the path of r. The proof of lemma 1 did not rely on the fact that no other cexels could be traversed, and hence it can be applied to *any* cexel lying on r's path. Therefore we can always reconstruct a point on a ray's path, and the representative direction is always available at the cexel traversed. This implies that we can propagate the ray in an exact straight line through an arbitrary number of cexels, thus achieving exact straight line propagation. ∎

Notice that the results just shown apply only to a single ray. In practice, more than one ray can be represented by a specific direction in a given cexel. If only one offset triple is kept, then, in general, it is impossible to propagate both rays in the same line. Perfect remedies to this problem exist, such as keeping track of the power of ray for every possible offset, (i.e., $(4GS+1)^3$ power fields for each possible offset for *one* possible representative direction) and creating a dynamic list of offsets for each member of $\mathcal{D}$. Unfortunately, any perfect remedy can cause a dramatic increase in the amount of storage required to model propagation through the use of cells and cexels, when arbitrarily large numbers of rays are to be propagated.

To keep the memory requirements to one triple offset per member of $\mathcal{D}$, offsets of conflicting rays [3] have to be combined in some fashion. The simplest combinations include setting the offset to 0, in the hope that the errors would balance, and selecting the offset of the *most important* ray. The power of each ray could then be added to obtain the power of the resulting ray. A more intuitively appealing solution consists of calculating a weighted average of the offsets, where the weight of an offset corresponds to the power contained in the beam. Notice that independently of the technique applied to combine the offset, this new combined ray will propagate in a straight line until it needs to combine with other rays.

4.2.2 Frustums

Although light rays propagate light in a straight line, they are only doing so in a finite number of sample directions. This property is biased towards the propagation of directional light. For instance, the light emanating from a diffuse light source (i.e., one with a constant radial power emanating from each point of the surface) can only be propagated approximately by using all the available representative directions. The light source is essentially approximated by a number of light sources which emit light in one of the representative directions.

To handle the propagation of light which exhibits a constant radial emissivity, we can select a different use for the partition of a cexel's incoming hemisphere of light. Instead of selecting a number of sample directions in which to propagate rays, we could choose a number of solid angles with constant radial power. The members of the hemisphere partition now represent frustums of light, with the associated power representing the constant radiant exitance of the representative pyramidal beam of light.

As in the case for rays, a finite number of solid angles can be used to discretize the light flux crossing an arbitrary cexel; a solid angle will not necessarily originate at the centre of a cexel. With each solid angle, we associate a beam in the shape of

[3] Conflicting in the sense that they would need a different value in the same offset field to satisfy their respective straight line propagation.

the solid angle, of constant radiant exitance. A flux of light crossing a cexel will then be represented as a combination of frustum beams of various, though constant, radial exitance. If we can propagate these frustums such that the shape and radial power of the beam is conserved, then we will have a better model for propagating light from diffuse sources and surfaces.

Let $\mathcal{F}$ denote the set of solid angles and associated frustums used to model the flux of light at the cexel level. Given a member f of $\mathcal{F}$, we would like to be able to propagate it *perfectly*. In the case of rays, perfect propagation implied straight line propagation. With frustums, we define perfection as the ability to propagate a frustum of light through cells such that each cexel traversed receives the same flux of light coming from the previous cexel traversed, as it would have had, had the original frustum of light impinged directly onto it.

We now show that no such perfect propagation mechanism can be constructed with a finite $\mathcal{F}$.

Claim 2 *Let $\mathcal{F}$ be a finite set of solid angles used to model propagation at each cexel using associated frustums of constant radial exitance. Then $\mathcal{F}$ is not sufficient for perfect frustum propagation.*

Proof: Consider a solid angle Ω_0 which contains the normal to a cexel, $\mathcal{C}_0$. If the cexel immediately in the direction of the normal is labelled as $\mathcal{C}_1$, then the frustum associated with Ω_0 will have to be propagated to $\mathcal{C}_1$, and represented under a new solid angle Ω_1, which is properly contained within Ω_0. This must hold if the frustum is to be propagated perfectly, hence the frustum at $\mathcal{C}_1$ must indicate that the origin is at $\mathcal{C}_0$, forcing Ω_1 to be properly contained within Ω_0. Notice that Ω_1 still contains the normal to both cexels.

By induction, a solid angle Ω_i containing the normal to $\mathcal{C}_0$, and perfect propagation, imply the existence of Ω_{i+1} at $\mathcal{C}_1$, and hence the existence of Ω_{i+1} at $\mathcal{C}_0$, where Ω_{i+1} is properly contained within Ω_i. The proper inclusion implies that all those solid angles are distinct, and members of $\mathcal{F}$. This contradicts the assumption that $\mathcal{F}$ is finite. ∎

Since no finite set of solid angles $\mathcal{F}$ exists such that frustums can be propagated perfectly, we will examine different methods of flux representation through the use of frustums.

Method 1.

One method that can allow reasonable frustum propagation is based on the representative scheme used for rays. We define a set of solid angles $\mathcal{S}$ and a set of representative directions $\mathcal{D}$ as we did for rays, and use the straight line propagation property to propagate a ray along the representative direction. A frustum is defined to be the solid angle to which the representative direction corresponds, as illustrated in figure 4.6.

To propagate a frustum, we simply propagate the representative ray at the centre of the frustum, and record the initial origin of the solid angle. The power associated with the transmitted frustum is also transferred to the receiving frustum. If all frustums originate (i.e., their projected apex) on a full grid point, such as the centre of a cexel, then it is sufficient to register the starting full grid point, along with representative direction, to fully characterize any propagated frustum.

In addition to the offset term introduced for the straight line propagation of the central ray, note that we also record the origin of the frustum. An exact solution would require the ability to record an arbitrarily large number of origins, if we are to model arbitrary scenes containing diffuse light sources. Each origin, ray offset, and representative direction triplet would then represent a specific frustum, and associated power.

Since storage requirements preclude the use of a large number of origins and offsets for each representative direction of a cexel, we must be able to combine the offsets, as before, and also the origins. Unlike offsets, whose total error is only a function of the size of a cexel,[4] origins can differ by arbitrarily large distances. It is not clear how these origins should be combined, with minimal error, but a weighted average of the origins according to power — similar to the offset averaging — would seem reasonable.

[4]All offsets are in $\{-2GS, \ldots, 2GS\}^3$, and therefore of size no more than one cexel's width.

The origin of a frustum of fixed shape provides information as to how far apart the frustum has spread. Since the frustum carries a constant radial power, the power received by a cexel depends on the distance that the frustum of light has travelled. The further a frustum has travelled, the smaller the solid angle of intersection the cexel will make with the beam, and the smaller the amount of power that the cexel will receive.

Method 2.

Due to the nature of the first suggested implementation, each frustum is assigned to exactly one cexel as the associated pyramidal beam of light is propagated through cells. This means that all the power associated with a frustum is assigned to a particular cexel, and gets concentrated in the central direction. While this is not worse than propagating rays of light, it would seem intuitively better to disseminate the flux information to all the cexels intersected by the frustum as it traverses cells. This should presumably improve the distribution of power across cexels, and provide a better approximation to the flow of power originating from diffuse light sources. In turn, the flow of power to each cell will be better represented, and hence the cell selection criterion, which is based on the amount of undistributed power, would be improved.

We extend the first proposed solution to allow a better distribution of the information characterizing a frustum of light being propagated. We use the same method to store the description of a propagated frustum, but instead of assigning it only to the cexel intersected by the central ray, we disseminate this information to all cexels intersected by the frustum. Each cexel stores identical information regarding the shape of the frustum, but only records the fraction of the power impinging on its surface. This fraction corresponds to the portion of the original frustum that impinges on the cexel. The total sum of the propagated power is then equal to the originating power, but each cexel, and by extension, each neighbouring cell, knows what portion it receives. This is illustrated in figure 4.12

The trade off with this scheme involves a higher storage cost, which cannot be avoided if the information is to be better distributed. The extra storage cost is

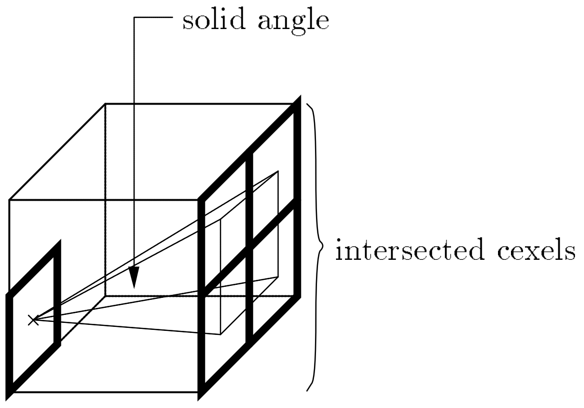


Figure 4.12: Distributing the Power of a Frustum

dependent on the size and number of cexels used: the more cexels are used, the more expensive method 2 is, but the more accurate is the distribution of power. The distribution of information also increases the risk of collisions between frustums with the same representative direction but differing origins, introducing further sources of propagation error. Additional calculations, in the form of determining which cexels are intersected by a frustum, and the fraction of power received, are also introduced.

Other alternatives include using filtering methods to combine and reconstruct flux information, and have yet to be explored. As future research, these should be investigated to determine how close to perfect frustum propagation one can get, and to get some measure of the error induced by each propagation model and mechanism.

4.2.3 Parallel Beams

A third and final alternative considered for modelling the flow of light makes use of parallel beams. Beams of constant cross-section are used to model the flow of light. As we did in the first two models, we partition the cexel's incoming hemisphere into a set of solid angles $\mathcal{S}$, and an associated set of representative directions $\mathcal{D}$. We define a beam of constant cross-section in each of the directions in $\mathcal{D}$, and use this set of parallel beams to model the flux of light at the cexel level.

In order to properly propagate these beams through empty cells, it is sufficient to propagate the representative ray, as long as the cross-sectional area of the beam is preserved. Since we can propagate rays in an exact straight line, we will assume we

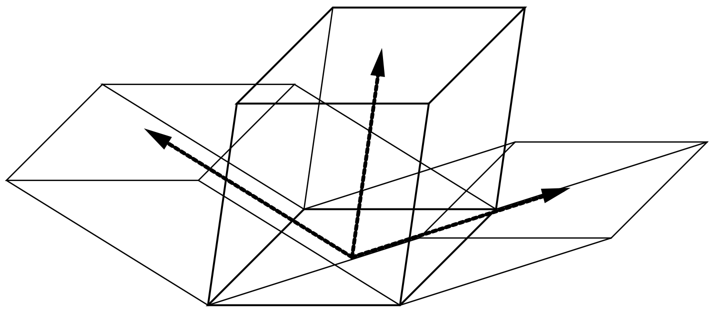


Figure 4.13: Orientation Dependent Parallel Beams

can do this for the representative ray. We instead examine two different methods of defining the cross-sectional area of a beam.

4.2.3.1 Orientation Dependent Projection

In this first approach, we consider an arbitrary cexel $\mathcal{C}$, and the incoming representative directions $\mathcal{D}(\mathcal{C})$ as they were defined for rays. Along each representative direction, we project the cexel, thus defining the cross-sectional area of the beam, as illustrated in figure 4.13. The $3GS^2$ parallel beams associated with the representative directions define the set $\mathcal{B}(\mathcal{C})$ of parallel beams for $\mathcal{C}$.

Since each parallel beam is the result of the projection of the cexel it belongs to, cexels with different orientations will produce distinct sets of beams. Each cexel possesses its local definition of parallel beams as an oblique projection of itself. As three different orientations of cexel are possible, namely along the xy, xz or yz planes, we end with $3 \times 3GS^2$ possible parallel beams, or three local sets of beams for each representative direction. Each representative direction is found in three different types of cexel, with respect to orientation, and hence each defines three different cross-sections for a beam in that direction. If we want each cexel to be able maintain a consistent definition of any of the three types of parallel beams that can reach it, the size of $\mathcal{B}(\mathcal{C})$ will be three times the size of $\mathcal{D}(\mathcal{C})$. If the available storage is limited, these three different types of parallel beams per representative direction may have to be combined.

Consider a cexel $\mathcal{C}$, and its set of representative directions, as well as its local definition of beams. To try to quantify the rate at which these beams spread, we will calculate the minimum distance from the centre of $\mathcal{C}$ to a point (x, y, z) inside C which does not fall in the path of any of the parallel beams. If we ignore the offset in ray propagation, we notice that the flux traversing $\mathcal{C}$ is modelled through this set of parallel beams, $\mathcal{B}(\mathcal{C})$. Since none of these beams contain the point (x, y, z), we label this point as being in "geometrical darkness"[5] with respect to $\mathcal{C}$. The minimum distance to such a point of geometrical darkness gives us a measure of how quickly an arbitrary flux modelled at $\mathcal{C}$ can diverge, as a function of the grid size.

Claim 3 *Let $(x_{\mathcal{C}}, y_{\mathcal{C}}, z_{\mathcal{C}})$ be the centre of a cexel $\mathcal{C}$ of a cell C. There exists a point of geometrical darkness (x, y, z) inside C such that for any $\varepsilon > 0$, $|(x, y, z) - (x_{\mathcal{C}}, y_{\mathcal{C}}, z_{\mathcal{C}})| < GS + \varepsilon$, where $|(x, y, z)| = \sqrt{x^2 + y^2 + z^2}$ and distances are measured in terms of grid size units, with a cexel side being of length $2GS$.*

Proof: Without loss of generality, assume that $\mathcal{C}$ is centred at the origin, with the positive z axis pointing inside the cell. Let $(x_r, y_r, z_r) \in \mathcal{D}(\mathcal{C})$, and let b be the projection of $\mathcal{C}$ along this direction. At a height of δ, the centre of the beam will be at position $\delta(\frac{x_r}{z_r}, \frac{y_r}{z_r}, 1)$, and any point within the beam and having height $z = \delta$ will have an x coordinate in the range $[\delta\frac{x_r}{z_r} - GS, \delta\frac{x_r}{z_r} + GS]$ and a y coordinate in the range, $[\delta\frac{y_r}{z_r} - GS, \delta\frac{y_r}{z_r} + GS]$. Since both x_r and y_r are at most GS in magnitude, and z_r is at least 1, the magnitude of the x and y coordinate of any point within *any* beam at height δ will be at most $\delta + GS$. Thus any point with coordinates $(0, \delta + GS, \frac{\delta}{2})$ will not be in any parallel beam, and will have a magnitude of:

$$
\begin{aligned}
|(x, y, z)| &\leq \sqrt{0 + (\delta + GS)^2 + (\frac{\delta}{2})^2} \\
&= \sqrt{GS^2 + 2GS\delta + 1.25\delta^2}. \qquad (4.7)
\end{aligned}
$$

If we choose δ such that $\delta < \frac{2}{\sqrt{5}}$, then

$$
(GS + \varepsilon)^2 = GS^2 + 2GS\varepsilon + \varepsilon^2
$$

[5] An area of "geometrical darkness" may still be illuminated by this cexel since we neglect both the offset and the method used for shading.

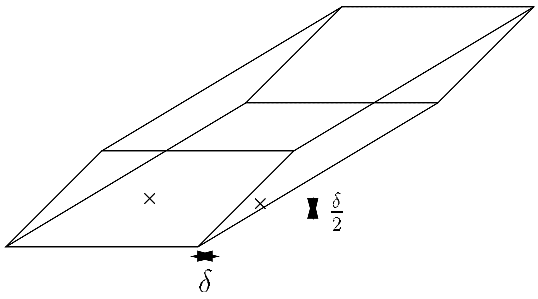


Figure 4.14: Trivial Geometric Darkness

$$
\begin{aligned}
&> GS^2 + 2GS\delta + 1.25\delta^2 \\
&= |(x,y,z)|^2. \qquad (4.8)
\end{aligned}
$$

Such an (x, y, z) is then a point of geometrical darkness whose distance to the centre of the cexel is less than $GS + \epsilon$. This is illustrated in figure 4.14. ■

By the above claim, we can get points of geometrical darkness arbitrarily close to the edge of the cexel, independently of the grid size. Therefore, if we only use the local definitions of beam, we can place small objects arbitrarily close to the edge of $\mathcal{C}$, such that any beam leaving the centre of $\mathcal{C}$ will not intersect this object. One might argue that these objects are trivial because the angle between them and the normal to $\mathcal{C}$ has to be virtually at a right angle with these objects for them to be close enough to the cell's boundary and to be in geometrical darkness.

To obtain another measure of the rapidity with which these beams split, we may want to ignore objects whose angle of azimuth is closer to 90 degrees than any member of $\mathcal{D}$. In other words, we wish to determine how far away from the centre of a cexel that two beams separate. As we will see, such non-trivial geometrical darkness points occur at a distance from the centre of the cexel which is practically independent of GS.

Claim 4 *If $GS \geq 4$, there exists a non-trivial point of geometrical darkness at a distance of $\varepsilon + \sqrt{4GS^2 + 9}$ from the centre of $\mathcal{C}$, for $\varepsilon > 0$, where distances are measured in terms of grid size units, a grid size unit being defined such that $2GS$ units is the side length of a cexel.*

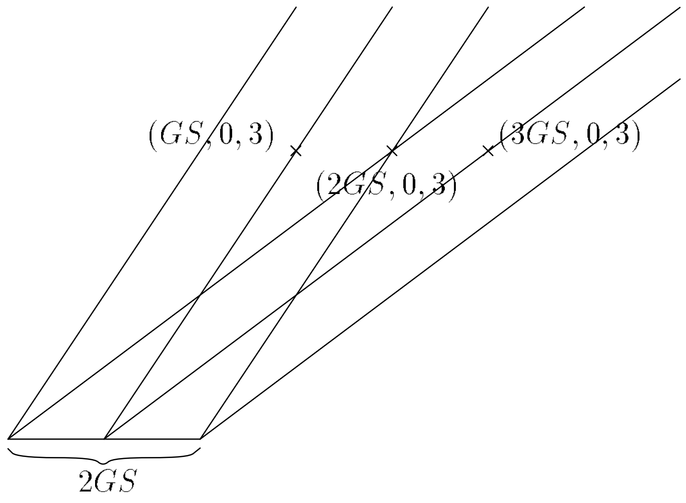


Figure 4.15: Non-Trivial Geometric Darkness: A Cross-Section

Proof: Consider the non-trivial point $(2GS, 0, 3)$, as illustrated in the figure 4.15 Again, assume that $\mathcal{C}$ is centred at the origin, with the positive z axis pointing inside the cell. Since z is always positive, $\mathcal{D}(\mathcal{C}) = \{(x, y, z) : |x|, |y|, z \in \{1, 3, \ldots, GS - 1, GS\}$ exactly one of which is $GS\}$. If $x = GS$, $z = 1$, and $GS > y \geq 1$, then at height 3, the centre of the (x, y, z) beam is at position $(3GS, 3y, 3)$. Since every beam projected along one of these directions has x ranging from $2GS$ to $4GS$, the non-trivial point is at most on the boundary of each of these beams.

If $x = GS$, $GS > z \geq 3$, and $GS > y \geq 1$, then at height 3, the centre of each of these beams has x coordinate at most GS. Since every beam projected along one of these directions has x ranging from 0 to $2GS$, the non-trivial point is at most on the boundary of each of these beams.

If $y = GS$, $GS > z \geq 1$, and $GS > x \geq 1$, then at height 3, the centre of each of these beams has y coordinate at least GS, if $z \leq 3$. Since every beam projected along one of these directions has y ranging from 0 to $2GS$, the non-trivial point is at most on the boundary of each of these beams. If $z > 3$, then x is at most $GS - 1$, and hence the x coordinates of points in the beam at a height of 3 is less than $2GS$, forcing the non-trivial point to be outside these beams.

If $z = GS$, $GS > x \geq 1$ and $GS > y \geq 1$, then at a height of 3, x is less than 3, which is less than GS. Again, the x range of the beam does not include the x

coordinate of our non-trivial point, thus it is outside these beams.

If x or y are negative, a similar argument will show that the beams thus defined do not properly contain our non-trivial point.

Therefore, $(2GS, 0, 3)$ is not properly contained within any of the locally defined beams associated with $\mathcal{D}$. Since this will lie on the outer boundary of at least two beams, a point which is ε further in a direction between the representative directions of the beams making contact with this point, will be outside all of the beams. It will also be at a distance of $\sqrt{4GS^2 + 9} + \varepsilon$, satisfying our claim. ■

Since,

$$\begin{aligned} 4GS^2 + 9 &< 4GS^2 + 12GS + 9 \\ &= (2GS + 3)^2 \end{aligned} \tag{4.9}$$

then we can always find a non-trivial point of geometrical darkness whose distance to the centre of a cexel is at most $2GS + 3$. In terms of length of cexels, this absolute point will be at most $1 + \frac{3}{2GS}$ cexel side lengths away, since distances were measured in full grid point units, where $2GS$ units is the same length as the length of the side of a cexel. This surprising[6] result suggests that we attempt to find a definition of parallel beams which does not deviate at such a fast rate.

4.2.3.2 Orientation Independent Projection

Consider a cube whose six faces are subdivided with a grid of size GS, and define $\mathcal{GD}$ as the set of all the representative directions. Along each representative direction, we project a cexel centred at the centre of the cube, and parallel to the face of the cube which the representative direction first traverses. This defines a global set of parallel beams called $\mathcal{GB}$, and is illustrated in figure 4.16

For each cexel $\mathcal{C}$, we define $\mathcal{D}(\mathcal{C})$ exactly as we did in the case of rays. The local set of parallel beams $\mathcal{B}(\mathcal{C})$ used to model the flow of light is the subset of $\mathcal{GB}$ whose

[6]Surprising in the sense that there are arbitrarily close gaps between beams, regardless of the size of the grid used to define them.

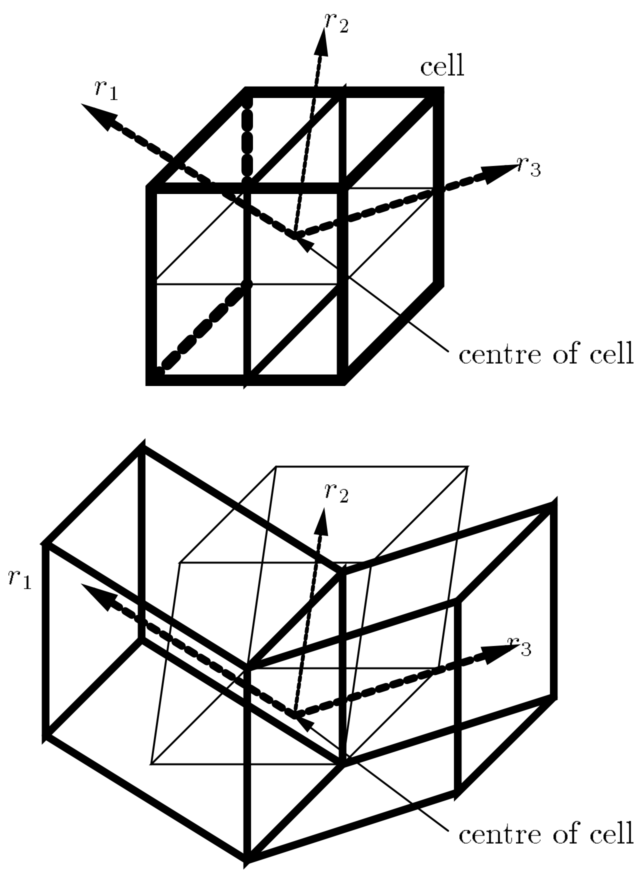


Figure 4.16: Orientation Independent Parallel Beams

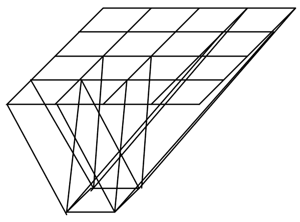


Figure 4.17: Separation of Orientation Independent Beams

representative direction corresponds to one in $\mathcal{D}$.

The definition of $\mathcal{B}(\mathcal{C})$ implies that only one type of beam has to be kept per representative direction, regardless of the orientation of the cexel. This scheme requires only one third the memory as the orientation dependent scheme, for accurate propagation of arbitrary single parallel beams. In addition, geometrical points of darkness occur further away from the centre of a cexel in the cexel independent scheme. Intuitively, this is explained by the thicker average cross-sectional area of the parallel beams used in this scheme. Whereas in the first scheme, the cross-sectional area of a beam ranges from roughly $\frac{1}{n}$ to 1 times the area of a cexel, this factor is always at least $\frac{1}{\sqrt{2}}$ in the current scheme.

We now show a stronger result about the distance of geometrical points of darkness, as a function of GS.

Claim 5 *The closest geometrical point of darkness is at least GS^2 away, where $2GS$ is the side length of a cexel.*

Proof: Consider the cube with the grid of size GS. Since the dimensions of the cube do not affect the representative directions, let each square resulting from placing the grid on each face be the same size as a cexel. Starting from the centre of a cexel, the beams in $\mathcal{B}(\mathcal{C})$ will traverse the cube exactly at the square whose centre corresponds to their respective representative directions. Therefore, all of the beams separate precisely at the cube's boundary. Since the cube is GS^3 cexel volumes, the distance from the centre to a face is $GS/2$, with each cexel of length $2GS$ giving a total distance of GS^2 grid size units, as required. This is illustrated for a grid of size 4 in figure 4.17

■

If we normalize the distance to the closest geometrical point of darkness by the size of a cexel, we notice that a grid size of GS yields a distance of $GS/2$ cexels. Thus for a fixed cexel size, the distance to the closest geometrical point of darkness is a linear function of GS, and this is a marked improvement over the orientation dependent result. Combined with the smaller memory requirement, this technique is definitely superior with regard to propagation.

Because of the smaller memory requirements of parallel beams, and their straight line propagation properties, we chose to implement the FIAT algorithm using parallel beams. Our initial implementation considered only the orientation dependent parallel beam definition, but due to the better properties of orientation independent beams, we shall also consider this approach in the following discussion.

Chapter 5

Light Sources

Since we have assumed that we are using a parallel beam representation of the flux at the cexel level, we know that we can propagate these beams through empty cells in a perfectly straight line. We now consider the problem of converting an arbitrary flux impinging on a cexel, to an *optimal* parallel beam representation. In this chapter, we discuss the fundamental case of converting the light emitted by a light source to a parallel beam representation. In particular, we examine how both diffuse and directional point light sources, as well as diffuse and directional light sources of extended area, can be converted to a parallel beam representation.

5.1 Introduction

Given an arbitrary flux entering a cell at the cexel level, our goal is to approximate it using some combination of the parallel beams. Let us define a set of properties that we will want this representation to have. Recall that for a given cexel $\mathcal{C}$, the set of solid angles is $\mathcal{S}(\mathcal{C})$, and the set of representative directions is $\mathcal{D}(\mathcal{C})$. Suppose that an arbitrary flux f crosses a cexel, and that the set of values assigned to the cexel's sample beams is F. We introduce two notions of power conservation:

Total Power Conservation The total power of f and F should be the same.

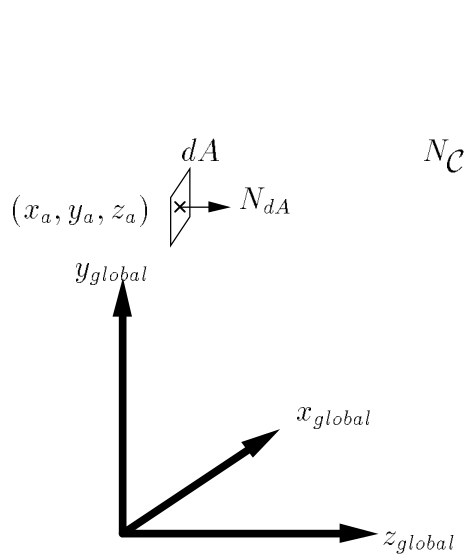


Figure 5.1: Surface Element Light Source

Directional Power Conservation For each solid angle $\Omega \in \mathcal{S}(\mathcal{C})$, the power of the parallel beam associated with Ω should be equal to the total power crossing the cexel in the envelope of directions contained within Ω.

The presumed additivity of power implies that if the second property holds, the first property will also hold. We use this second property to define the *optimal* solution as the one for which the directional power is completely conserved. We derive for four different light sources a method to compute the power received by a cexel, through each solid angle. In an implementation, this power will be assigned to the associated parallel beams of the cexel; this discretization is the approximation to the flux from a light source.

5.1.1 Calculating the Exact Directional Power

Let dA be an infinitesimal element of surface from which light emanates, and let $\mathcal{C}$ be a cexel on which part of this light impinges. Suppose that dA is located at (x_a, y_a, z_a) with respect to global three dimensional axes x_{global}, y_{global} and z_{global}, and that dA has a normal N_{dA}. Without loss of generality, assume that the centre of $\mathcal{C}$ is located at $(\mathcal{C}_x, \mathcal{C}_y, \mathcal{C}_z)$, that it lies in the xy plane, and that it has a normal $N_{\mathcal{C}}$, as illustrated in figure 5.1.

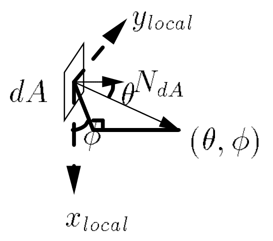


Figure 5.2: Local Axes

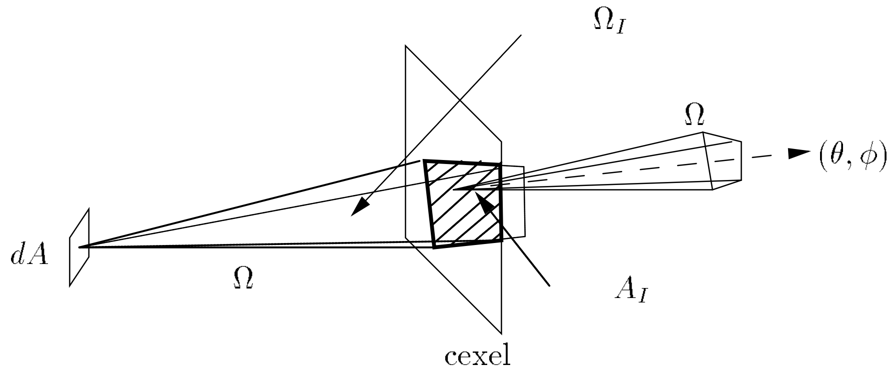


Figure 5.3: Solid Angle of Intersection

Let the local xy plane be the plane tangent to dA, and perpendicular to its normal, with fixed x_{local} and y_{local} axes, as illustrated in figure 5.2. Without loss of generality, assume that dA only emits light in the upper hemisphere, that is the one containing N_{dA}. Suppose that the radiant power emanating from dA is a function of θ and ϕ, say $RP(dA, \theta, \phi)$. As presented in chapter 2, θ is the azimuth angle, as measured from N_{dA}. The angle ϕ is measured counterclockwise from the positive x_{local} axis.

Consider a solid angle $\Omega \in \mathcal{S}(\mathcal{C})$ and let θ and ϕ correspond to the polar coordinates of the associated representative direction in $\mathcal{D}(\mathcal{C})$. Only the power emitted from dA within the solid angle Ω can contribute to the power collected by $\mathcal{C}$ over that same solid angle. Of that power, only the portion impinging on $\mathcal{C}$ over the intersection area A_I determined by the projection from dA of a solid angle of shape and direction Ω with $\mathcal{C}$, contributes to the power received by the cexel through that solid angle. The actual power impinging on the intersection area is then determined by the solid angle of intersection Ω_I formed by A_I and dA, as illustrated in figure 5.3. Since the flux crossing this intersection is precisely dA's contribution to Ω of $\mathcal{C}$, this quantity is calculated as:

$$P(\mathcal{C}, \Omega, dA) = \int_{\Omega_I} RP(dA, \theta, \phi) \sin\theta \, d\theta \, d\phi, \tag{5.1}$$

where $P(\mathcal{C}, \Omega, dA)$ is the power received by $\mathcal{C}$, through Ω, due to dA.

If dA is a diffusely emitting surface element, $RP(dA, \theta, \phi)$ is a constant, say P_k, for all θ's and ϕ's. Recall from chapter 2 that the solid angle corresponding to a hemisphere is 2π. If the total power emanating from the surface element over the hemisphere is $P(dA)$, then P_k is the power emitted through an infinitesimal solid angle, thus:

$$P_k = \frac{P(dA)}{2\pi}. \tag{5.2}$$

The power received from a diffusely emitting surface element is then:

$$\begin{aligned} P(\mathcal{C}, \Omega, dA) &= \int_{\Omega_I} P_k \sin\theta \, d\theta \, d\phi \\ &= P_k \int_{\Omega_I} \sin\theta \, d\theta \, d\phi \\ &= P_k \Omega_I \\ &= P(dA) \frac{\Omega_I}{2\pi}. \end{aligned} \tag{5.3}$$

Consider a two-dimensional light source L of area A_L, the same cexel $\mathcal{C}$, and the same solid angle Ω. The total contribution of the extended light source L to the power received by $\mathcal{C}$, through Ω, due to L, is the integral of $P(\mathcal{C}, \Omega, dA)$ over all dA's of L:

$$\begin{aligned} P(\mathcal{C}, \Omega, L) &= \int_{A_L} P(\mathcal{C}, \Omega, dA) dA \\ &= \int_{A_L} \left(\int_{\Omega_I} RP(dA, \theta, \phi) \sin\theta \, d\theta \, d\phi \right) dA \end{aligned} \tag{5.4}$$

If L is an ideal diffuse light source, $RP(dA, \theta, \phi)$ is constant for all possible values dA, θ and ϕ, over L. The contribution from such a light to the given cexel and solid angle becomes:

$$P(\mathcal{C}, \Omega, L) = P_k \int_A \left(\int_\Omega h(d\omega, dA) d\omega \right) dA, \tag{5.5}$$

where

$$h(d\omega, dA) = \begin{cases} \cos(\psi) & \text{if a vector } d\omega \text{ from } dA \text{ intersects } \mathcal{C} \\ 0 & \text{otherwise,} \end{cases} \tag{5.6}$$

and ψ is the angle between $N_{d\omega}$ and $N_{\mathcal{C}}$.

We defer the evaluation of these integrals to subsequent sections of this chapter that deal with solutions to these integrals for specific types of light sources.

5.2 Point Light Sources

Point light sources correspond to points in space emanating light in all direction. Since they have often been used in computer graphics, and since the solution represents part of the solution to the more general case, we start with this simple case. We deal with both diffuse and global directional sources, where conceptually, a global directional light source is a diffuse source placed at an arbitrarily large distance from the scene, such that all light appears to arrive from the same direction.

5.2.1 Global Directional Light Source

Suppose that the light emitted by a light source L appears to come only from one direction V_L with respect to the scene. Only the solid angle(s) containing this direction can receive power from this light source. If the direction lies at the boundary between solid angles, we can either assign all the resulting power to one solid angle, or distribute it equally to each angle. We choose the former, as it is simpler, and as there is no evidence showing that the other approach is necessarily better.

To calculate the power of the light impinging on cexel $\mathcal{C}$, we first calculate the cross-sectional area of a parallel beam projected onto the cexel in the direction V_L. The cross sectional area of the beam is the area of the cexel, $A_{\mathcal{C}}$, scaled by the cosine of ψ, the angle between V_L and $N_{\mathcal{C}}$, as illustrated in figure 5.4. If the total power of the light source is RP, the power contained in an infinitesimal solid angle $d\omega$ is: $\frac{RP}{4\pi}$. We associate this power with a ray of light in the direction V_L. Hence the power

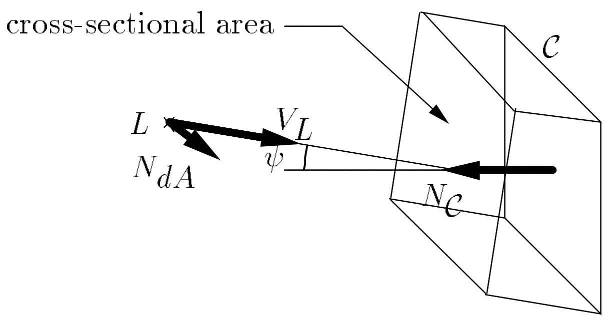


Figure 5.4: Global Directional Light Source

impinging on $\mathcal{C}$ through Ω due to dA is:

$$P(\mathcal{C}, \Omega, dA) = \frac{RP}{4\pi} A_{\mathcal{C}} \cos(\psi). \tag{5.7}$$

The approximation involved with computing the power received from this light source lies with computing the power assigned to the parallel beam in the representative direction corresponding to the solid angle Ω.

5.2.2 Diffuse Point Source

A diffuse point source corresponds to a point in space that emits light equally in all directions. In most case, a point light source is sufficiently far away from a scene that all the light emanating from the source and reaching the scene is contained in one hemisphere. Since this is roughly equivalent to an element of surface emanating light in one hemisphere, we will treat point sources as if they were infinitesimal elements of surface area sources. The motivation behind this is that we will need infinitesimal area sources in a following section when extended diffuse sources are explained. It is understood that the illumination emanating from a point light source could be similarly computed.

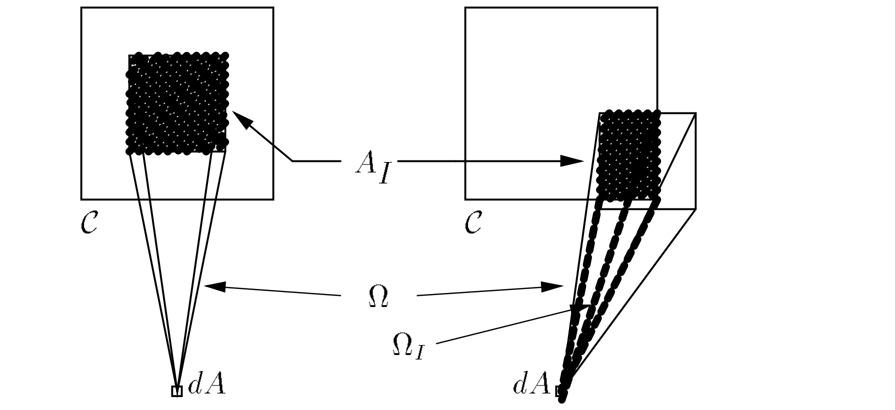
Consider an infinitesimal element of surface area dA, placed at a finite distance from the scene, with a total radial power of RP emanating evenly over the entire upper hemisphere of the plane containing dA. Without loss of generality, assume that the light source L is outside the bounding volume $\mathcal{V}$, since the solution is similar when the light source is inside.

To determine the solid angles which contribute light to a particular cexel $\mathcal{C}$, consider the four vectors defined by dA and each of the four corners $\mathcal{C}_c$ of $\mathcal{C}$. The solid angle delimited by dA and $\mathcal{C}$ contains all the directions which can impinge on $\mathcal{C}$ from dA, as illustrated in figure 5.5.

Recall that $\mathcal{S}(\mathcal{C})$ is a partition of the incoming hemisphere defined by mapping a grid of size GS onto a cube. In general, each intersection point on the grid is shared by four squares which define a solid angle. For a grid of size GS, there are $6(GS)^2$ solid angles defined. The number of intersection points on the grid is $2+6(GS)^2$, since each face defines $6(GS+1)^2$ intersection points, but each of the $12(GS-1)$ intersection points lying on an edge is counted twice, and each of the 8 corner intersection is counted three times. If we define a vector from the centre of a cube to the centre of each of the squares and to their corners, there are only two more corner vectors than centre vectors. Let us call this set of corner vectors $\mathcal{S}_c(\mathcal{C})$, and identify V_c with a member of this set. To determine if the solid angle Ω can contribute light to the cexel $\mathcal{C}$, we will intersect Ω from dA with the plane of the cexel and determine if this projection intersects $\mathcal{C}$.

Project each corner vector V_c from dA, onto the plane containing the cexel. The solid angles that can contribute power to $\mathcal{C}$, from dA, include those for which at least one V_c vector lies within $\mathcal{C}$. However, the test is only conclusive if it is positive. A general solution involves a walk starting with any solid angle containing some vector in the cexel, through the neighbouring solid angles. Starting with this solid angle, examine the intersection of its projection with $\mathcal{C}$, and determine if there is an intersection. Successively examine each neighbouring solid angle for an intersection, until a surrounding border of solid angles which do not intersect the cexel has been found.

For each contributing solid angle, we must now determine the actual contribution. If the four corner vectors of solid angle Ω's lie within the cexel, the entire solid angle lies within the solid angle determined by $\mathcal{C}$ and dA. This implies that the solid angle of intersection is Ω itself, and that the power, $P(\mathcal{C}, \Omega, dA)$, received by $\mathcal{C}$ through Ω



Four corners inside $\mathcal{C}$ One or more corners outside $\mathcal{C}$

Figure 5.5: Diffuse Surface Element Source

due to dA is:

$$P(\mathcal{C}, \Omega, dA) = RP\frac{\Omega}{2\pi}. \quad (5.8)$$

If not all four lie within the cexel, we must calculate the solid angle of intersection of Ω with $\mathcal{C}$. To do this, we project Ω from dA onto the plane in which $\mathcal{C}$ lies, and calculate the resulting area of intersection A_I. The contribution of dA to $\mathcal{C}$ is then directly proportional to the solid angle of intersection Ω_I determined by dA and I, as illustrated in figure 5.5:

$$P(\mathcal{C}, \Omega, dA) = RP\frac{\Omega_I}{2\pi}. \quad (5.9)$$

We can either compute this quantity by evaluating an integral to obtain the exact solid angle of intersection, or approximate it using the area of intersection. The discussion of such approximations will be deferred to the next section.

5.3 Extended Area Light Sources

We now consider light sources whose emanating surface extends over a finite area and is planar. To simplify our discussion of these sources, we only consider planar polygons as light sources, and again deal with both diffuse and directional sources. A directional source emits light in only one direction, and acts as a spot light: it sends a beam of light in the shape of the polygonal light source. The diffuse extended area light source will act as if each of the infinitesimal element of surface area composing

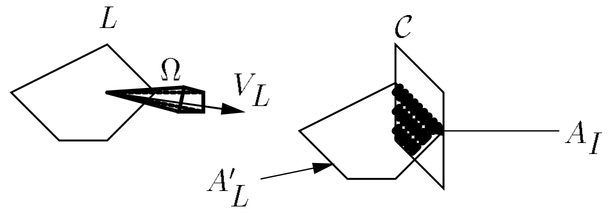


Figure 5.6: Directional Polygonal Light Source

it was a infinitesimal surface element light source.

5.3.1 Directional Polygonal Light Source

Suppose we have a polygonal light source L of area A_L that emits light in direction V_L, with power P_L. As was the case for directional point light sources, only one solid angle of the cexel $\mathcal{C}$ will receive a non-zero contribution from light source L. The contributing solid angle Ω will again be the one that contains V_L, with an arbitrary assignment being made for a V_L that belongs to more than one solid angle. This is illustrated in figure 5.6.

The power to associate with Ω is the fraction of the beam leaving L impinging on $\mathcal{C}$, multiplied by P_L. Given the area of intersection A_I of the projected area A'_L of L onto the plane containing $\mathcal{C}$, and of $\mathcal{C}$, the portion of the beam impinging on $\mathcal{C}$ is $\frac{A_I}{A'_L}$. The impinging power is then:

$$P(\mathcal{C}, \Omega, L) = P_L \frac{A_I}{A'_L}. \tag{5.10}$$

The flux from the light source is then approximated at the cexel level by assigning this power to the parallel beam in the representative direction corresponding to Ω.

5.3.2 Diffuse Polygonal Light Source

Physical light sources, such as fluorescent lights, both possess a finite area and radiate light in all directions; we should be able to model them more realistically using extended diffuse area light sources than any of the light sources examined thus far.

This provides the motivation for handling this last type of light source. It is more difficult to convert the flux of such a light source to a polygonal beam representation. We shall only do so for convex polygons. Since every polygon can be partitioned into a set of convex polygons, the techniques we present can be extended to handle arbitrary polygonal light sources.

5.3.2.1 Contributing Solid Angles

Suppose we have a convex polygon L consisting of a collection of infinitesimal elements of surface area dA, each radiating power equally in all directions, with a total power output of P_L. As was the case for diffuse point light sources, a cexel $\mathcal{C}$ can receive light from L through many solid angles. Our first task is to determine the set of solid angles S such that any infinitesimal ray of light which leaves L and impinges on $\mathcal{C}$ lies within a member of this set. Assume that the light source lies on only one side of the cexel, and that $\mathcal{C}$ lies in the xy plane. We present an algorithm for determining S; a similar technique can then be employed for cexels lying in a different plane.

Suppose that the extent of light source L is defined by the list of n vertices $\{V_i | 1 \leq i \leq n\}$. For each vertex V_i, calculate the four vectors from the vertex to each of the four corners of the cexel. Map these vectors onto the $z = 1$ plane, thus defining the set of vectors $\mathcal{Z}_1$ as

$$\begin{aligned} \mathcal{Z}_1 = \{(x_k, y_k) | \ \exists t, \text{ vertex } V \text{ and cexel corner } \mathcal{C}_c \text{ such that } x_k = t(V_x - \mathcal{C}_{cx}) \\ y_k = t(V_y - \mathcal{C}_{cy}) \text{ and } t(V_z - \mathcal{C}_{cz}) = 1\} \end{aligned} \tag{5.11}$$

Now determine the convex hull $\mathcal{H}$ of $\mathcal{Z}_1$ in the $z = 1$ plane, that is the set of (x, y) points such that:

1. $(x, y) \in \mathcal{H}$ if $(x, y) \in \mathcal{Z}_1$
2. $(x, y) \in \mathcal{H}$ if there exist i, j, t with $(x_i, y_i), (x_j, y_j) \in \mathcal{H}$ such that $(x, y) = (x_i, y_i) + t(x_j - x_i, y_j - y_i)$, and $0 \leq t \leq 1$.

Claim 6 *There exists a vector transporting light from a point L_p on L to a point $\mathcal{C}_p$*

on $\mathcal{C}$ if and only if this vector, when scaled such that its z coordinate becomes 1, lies in $\mathcal{H}$.

Proof: First, consider a point L_p on the light source and a point $\mathcal{C}_p$ on the cexel. We consider five possible cases, each of which is illustrated in figure 5.7.

1. **L_p on a vertex, $\mathcal{C}_p$ on a corner of $\mathcal{C}$.**

   If such is the case, then the resulting scaled vector is a member of $\mathcal{H}$, by the first property.

2. **L_p on a vertex, $\mathcal{C}_p$ on a side of $\mathcal{C}$.**

   Consider the corners $\mathcal{C}_c$, and $\mathcal{C}_{c'}$ that define the side on which $\mathcal{C}_p$ lies. By property 1, their respective projections belong to $\mathcal{H}$. Since $\mathcal{C}_p$ lies on the line which joins the pair, there exists a t such that
   $\mathcal{C}_p = \mathcal{C}_c + t(\mathcal{C}_{c'} - \mathcal{C}_c)$, and $0 \leq t \leq 1$. The vector directed from L_p to $\mathcal{C}_p$ that is, $L_p - \mathcal{C}_p$ can now be written as:

   $$\begin{aligned} L_p - \mathcal{C}_p &= L_p - (\mathcal{C}_c + t(\mathcal{C}_{c'} - \mathcal{C}_c)) \\ &= (L_p - \mathcal{C}_c) + t((L_p - \mathcal{C}_{c'}) - (L_p - \mathcal{C}_c)) \end{aligned} \tag{5.12}$$

   If we normalize $(L_p - \mathcal{C}_c)$ and $(L_p - \mathcal{C}_{c'})$, such that their respective z coordinate is 1, they belong to $\mathcal{H}$, by the first property. Noticing that $\mathcal{C}_c$, $\mathcal{C}_{c'}$, and $\mathcal{C}_p$ all have the same z coordinate, since all three belong to the cexel plane, we scale each vector by the same value to make the z coordinate become 1. Observe that

   $$\frac{(L_p - \mathcal{C}_p)}{L_{p_z} - \mathcal{C}_{p_z}} = \frac{(L_p - \mathcal{C}_c)}{L_{p_z} - \mathcal{C}_{p_z}} + t\left(\frac{(L_p - \mathcal{C}_{c'})}{L_{p_z} - \mathcal{C}_{p_z}} - \frac{(L_p - \mathcal{C}_c)}{L_{p_z} - \mathcal{C}_{p_z}}\right),$$

   which implies that the scaled vector lies in $\mathcal{H}$ by property 2.

3. **L_p on a vertex, $\mathcal{C}_p$ inside $\mathcal{C}$**

   Since $\mathcal{C}_p$ is inside $\mathcal{C}$, there exist two points on the sides of $\mathcal{C}$ such that $\mathcal{C}_p$ lies on the line joining them. As in the previous case, the scaled mappings of these points belong to $\mathcal{H}$, and by a similar argument, the projection of the vector from L_p to $\mathcal{C}_p$ onto the $z = 1$ plane must also lie in the convex hull $\mathcal{H}$.

4. **L_p lies between two vertices, $\mathcal{C}_p$ inside $\mathcal{C}$**

   Suppose that L_p lies between the vertices V_i and V_j. By the previous case, the projection of the vector between a vertex and any point inside the cexel is a member of $\mathcal{H}$. Since the three vectors are formed by joining a point on the light source to the same point on the cexel, and since the points on the light source are collinear, it follows that the three vectors are in the same plane. Since the vector L_p to $\mathcal{C}_p$ is in between the other two vectors, it's projection onto the $z = 1$ plane will also lie in between the projection of these two vectors. Formally, there exists a t such that:

$$L_p = V_i + t(V_j - V_i)$$

   therefore, the vector between L_p and $\mathcal{C}_p$ can be written as:

$$\begin{aligned} &(L_{p_x} - \mathcal{C}_{p_x}, L_{p_y} - \mathcal{C}_{p_y}, L_{p_z} - \mathcal{C}_{p_z}) = \\ &(V_{i_x} - \mathcal{C}_{p_x}, V_{i_y} - \mathcal{C}_{p_y}, V_{i_z} - \mathcal{C}_{p_z}) \\ +t((V_{j_x} - \mathcal{C}_{p_x}) - (V_{i_x} - \mathcal{C}_{p_x}), &(V_{j_y} - \mathcal{C}_{p_y}) - (V_{i_y} - \mathcal{C}_{p_y}), (V_{j_z} - \mathcal{C}_{p_z}) - (V_{i_z} - \mathcal{C}_{p_z})). \end{aligned} \tag{5.13}$$

   If we normalize this vector such that the z component becomes 1, there still exists a constant, namely $t' = \frac{(V_{i_z} - \mathcal{C}_{p_z})t}{(V_{i_z} - \mathcal{C}_{p_z})t + (1-t)(V_{i_z} - \mathcal{C}_{p_z})}$, such that the normalized vector lies on the line segment joining the projection of the $\mathcal{C}_p$ to vertex vectors. Since these projections lie in the convex hull, then so does the one for $(L_p - \mathcal{C}_p)$, by property 2.

5. **L_p lies inside L, $\mathcal{C}_p$ inside $\mathcal{C}$**

   In the last case, if we choose two arbitrary points on the light source and on the cexel, we show that the scaled vector is always in $\mathcal{H}$. Since L_p is inside L, and since L is convex, there exists two points, each lying between two vertices, and such that the line segment joining them is inside L, and contains L_p. By a

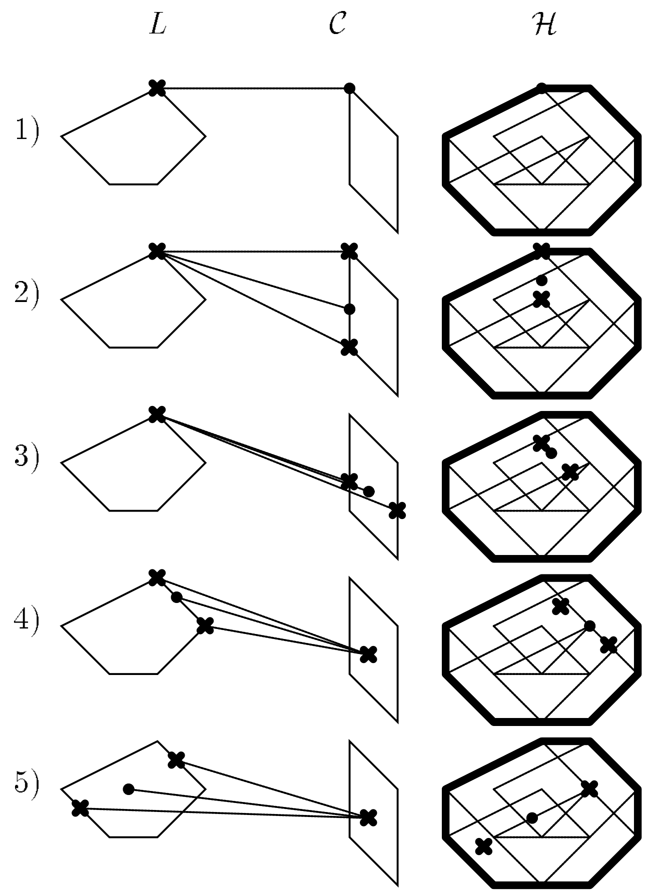


Figure 5.7: Convex Hull Examples

similar argument to the last case, we can write the projection of $(L_p - \mathcal{C}_p)$ onto the $z = 1$ plane as a combination of the projection of the vectors formed by each of the two points and $\mathcal{C}_p$. Since these projections are in $\mathcal{H}$ by the previous case, then by property 2, so is the projection of $(L_p - \mathcal{C}_p)$.

The five cases are illustrated in figure 5.7. The first case shown is a vector between a corner of the cexel and a corner of the light source, and the mapping of this vector in the convex hull. The following four cases illustrate how property 2 can be applied to show that a vector also belongs to the convex hull. The line segments between the light source and the cexel illustrate how the middle segment lies in the same plane as the other two. The corresponding vectors are shown in the hull: the two x's represent the outside vectors, the circle the middle one.

On the other hand, if we have a point which lies inside $\mathcal{H}$, then by reversing the above argument, we can construct a vector between a point of L and a point of $\mathcal{C}$ which gets mapped precisely to this point. This shows that the convex hull contains precisely those directions that do receive light. ■

Now that we have determined the exact directions from which $\mathcal{C}$ can receive light from L, we proceed — as we did for a diffuse point light source — to project the corners of the solid angles onto the $z = 1$ plane. A solid angle will receive light from L if the projection of at least one of its corners onto the plane lies within the convex hull. As in the case of a point source, this condition is sufficient but not necessary: we must "scan convert" the convex hull. A walkthrough has to be conducted starting with any solid angle containing a vector belonging to $\mathcal{H}$, and ending when a surrounding belt of non-intersecting solid angles has been found. For each member of the set S of all such solid angles, we must determine the amount of power that the cexel receives from the light source within the directions defined by each solid angle.

5.3.2.2 Contribution of a Solid Angle

For a given solid angle Ω, we can calculate the power of the light impinging on $\mathcal{C}$ through Ω by solving the iterated integral given by equation 5.5. Recall that:

$$P(\mathcal{C}, \Omega, L) = P_k \int_A \left(\int_\Omega h(d\omega, dA) d\omega \right) dA,$$

where h is as defined in equation 5.6. The radiant intensity P_k is the power of L divided by its area, A_L, and by the hemisphere of emittance. That is:

$$P_k = P_L \frac{1}{2\pi A_L}.$$

Before presenting approximations to this iterated integral, we examine a simple case for which we can get a precise answer without solving the integral. If for Ω each projection of L onto the cexel plane in each of the corner directions defined by Ω cover $\mathcal{C}$, the convexity of L implies that each point of $\mathcal{C}$ receives light from every infinitesimal solid angle of Ω. This is illustrated in figure 5.8. Instead of integrating

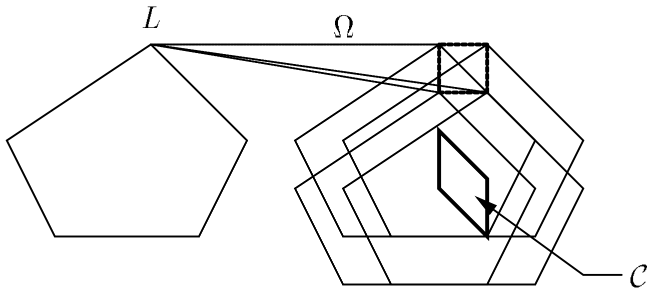


Figure 5.8: Extended Light Source — A Simple Case

over the light source, we integrate over the projected area $A_{\mathcal{C}'}$ of the cexel onto the plane containing the light source the contribution becomes:

$$\begin{aligned} P(\mathcal{C}, \Omega, L) &= P_k \int_{\mathcal{C}'} \left(\int_{\Omega} h(d\omega, dA) d\omega \right) dA \\ &= P_k \int_{\mathcal{C}'} \Omega dA \\ &= P_k \Omega \int_{\mathcal{C}'} dA. \end{aligned}$$

If the area of the cexel is $A_{\mathcal{C}}$, the projected area of the cexel is $A_{\mathcal{C}'} = A_{\mathcal{C}} \cos(\psi)$, where ψ is the angle between the normals of the cexel and the light source. If we substitute this value, and the value derived above for P_k, the power received by the cexel is:

$$P(\mathcal{C}, \Omega, L) = P_L \frac{A_{\mathcal{C}}}{A_L} \frac{\Omega}{2\pi} \cos(\psi). \tag{5.14}$$

The general integral for P is not trivially calculated, but a simple approximation can be obtained if we reverse the order of integration. After reversing the order of integration, we rewrite the power as:

$$P(\mathcal{C}, \Omega, L) = \int_{\Omega} \left(P_k \int_{A} h(d\omega, dA) dA \right) d\omega.$$

Suppose that ψ, the angle between the normal to the cexel and an infinitesimal solid angle $d\omega$ within Ω, is constant over Ω, making $\cos(\psi)$ also constant. Since

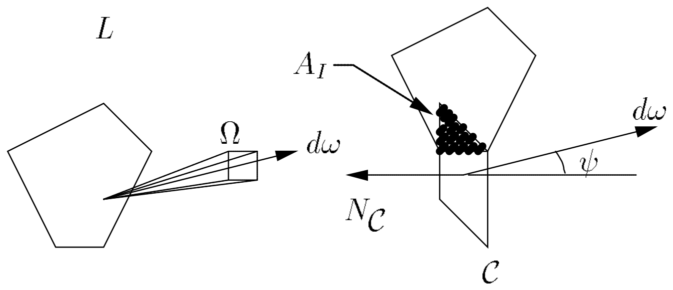


Figure 5.9: Projection of Light Source onto Cexel Plane

$\frac{h(d\omega, dA)}{\cos(\psi)}$ is 1 when a $d\omega$ from dA impinges on $\mathcal{C}$, and 0 otherwise, the integral

$$\int_A \frac{h(d\omega, dA)}{\cos(\psi)} dA$$

is precisely the intersection between a cexel and the projection of the extended light source in the direction $d\omega$ onto the cexel's plane, as illustrated in figure 5.9. If we assume that the intersection area is the same for every $d\omega$, then a first approximation to the impinging power is

$$P(\mathcal{C}, \Omega, L) = P_k \Omega \int_A h(d\omega, dA) dA. \tag{5.15}$$

Choose the representative direction $d\omega$ of Ω, and calculate the projected intersection area A_I. Since we assume that $\cos(\psi)$ is constant over Ω, we use the representative direction to approximate the cosine of the angle between an infinitesimal solid angle $d\omega$ and the normal to the cexel. Since

$$\int_A \frac{h(d\omega, dA)}{\cos(\psi)} dA = A_I,$$

then

$$\int_A h(d\omega, dA) dA = A_I \cos(\psi),$$

and equation 5.15 is further approximated by:

$$P(\mathcal{C}, \Omega, L) = P_L \frac{A_I}{A_L} \frac{\Omega}{2\pi} \cos(\psi), \tag{5.16}$$

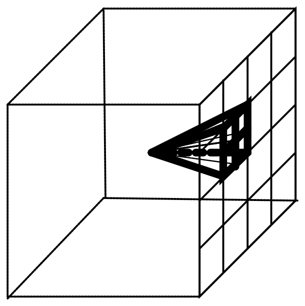


Figure 5.10: Quadrants of a Solid Angle

where P_k has been replaced by its value.

If upon examining the convex hull for contributing solid angles we discover that the projection onto the $z = 1$ plane of Ω is only partially contained within $\mathcal{H}$, we can deduce that only the $d\omega$'s which do project onto the convex hull can ever contribute light to $\mathcal{C}$. We can improve on the last approximation by replacing Ω by the fraction of Ω that maps inside $\mathcal{H}$.

To attempt to improve the accuracy of our calculation of the impinging power over a solid angle Ω, we take advantage of the way in which the solid angles are defined.

Consider the same solid angle Ω as before, and divide the square that defines this solid angle into four equal quadrants, as illustrated in figure 5.10. Instead of assuming that $\cos(\psi)$ is constant over the entire solid angle, suppose that it is constant over each of the solid angles defined by each of the quadrants. The power received in Ω can then be written as the sum of the power received through each sub-solid angle:

$$\begin{aligned} P(\mathcal{C}, \Omega, L)) &= P_k \int_A \left(\int_\Omega h(d\omega, dA) d\omega \right) dA \\ &= P_k \int_A \left(\sum_{i=1}^{4} \int_{\Omega_i} h(d\omega, dA) d\omega \right) dA \\ &= P_k \sum_{i=1}^{4} \left(\int_{\Omega_i} \left(\int_A h(d\omega, dA) dA \right) d\omega \right), \end{aligned} \tag{5.17}$$

where the Ω_i's correspond to the four quadrants of the solid angle. Since we assume ψ is constant over each quadrant, let each of the four ψ_i values correspond to the angle between the corner vector and the normal to $\mathcal{C}$. The inner integral is then approximated as before, by calculating the product of $\cos(\psi_i)$ and of the projected intersection area of L — in the corner vector direction, and onto the cexel's plane — with the cexel. This approximates the integral by treating it as a constant over the entire quadrant.

We are then left with the sum of four simple integrals, each of which calculates a solid angle. We can approximate each integral as a solid angle one fourth of Ω. This in effect makes our approximation to the power the product of the light source's point radiant intensity with the solid angle and the average intersection area of four projections. Because the integral calculated for each corner vector can be reused, only two more intersection calculations will have to be performed.

The final approximation is:

$$P(\mathcal{C}, \Omega, L) = P_k \Omega \sum_{i=1}^{4} \cos(\psi_i) A_i, \tag{5.18}$$

where A_i is the area of intersection between L projected in the direction of corner i, and $\mathcal{C}$. If we compare the cost of this approximation with the previous one, we notice that only two extra projections are required over the whole sphere of solid angles. In addition, if the four projections turn out to cover the entire cexel, then we have detected a simple case, in which the exact contribution is computed as in equation 5.14. As we did for the first general approximation, we can improve the solution by substituting the appropriate fraction of the solid angle than does contribute, instead of always using the full quantity Ω.

Note that the four term approximation is not necessarily better than the one term one. One can construct examples where the first approximation is more accurate, simply by moving the light source far away, such that the solid angle centred anywhere on the light source only intersects the cexel in its central portion. By choosing the light source to be very far away, the central vector can be made arbitrarily closer to any vector within a solid angle of intersection, than any of the four corner vectors. Similar examples can be constructed where the four corners technique outperforms the single projection calculation.

Problems with these approximations arise mostly when projections of the light source along the sample direction(s) of a solid angle completely miss the cexel. In such circumstances, consider the direction within a solid angle Ω that yields the greatest projected intersection area with the cexel. This intersection area is bounded by the minimum of the cexel area, and that of the projected light polygon in this direction. Since we have no efficient method of discovering this direction, we instead choose a suitable direction from Ω, either the centre or a corner, and project L onto $\mathcal{C}$'s plane. We then use the minimum of this area, and of that of the cexel, as the average intersection area. Multiplying this by the fraction of the solid angle which can impinge on $\mathcal{C}$, and by the radial power $\frac{P_L}{2\pi A_L}$, we obtain an estimate for the power impinging onto the cexel within Ω:

$$P(\mathcal{C}, \Omega, L) \approx \frac{\min(A_L \cos(\psi), A_{\mathcal{C}})}{A_L} \frac{\Omega}{2\pi} P_L. \tag{5.19}$$

This approximation is no worse than the two methods previously discussed, as it uses similar arbitrary information in the same approximation scheme, but allows us to get a (non-zero) approximation for each member of the set of solid angles which we know to illuminate $\mathcal{C}$.

5.3.2.3 Arbitrarily Close Approximations

Now that we have covered a set of first approximations, let us consider how we could construct arbitrarily close approximations. Suppose we wish to construct an approximation which deviates from the real power of the flux by at most $\varepsilon > 0$. If the real power is P_r, and the approximated power is P_a, then

$$P_r(1+\varepsilon) \geq P_a \geq P_r(1-\varepsilon) \tag{5.20}$$

Let ε_M be $\frac{\varepsilon}{\Omega}$, and suppose that we can divide the solid angle Ω into n solid angles $\{\Omega_1, \ldots, \Omega_n\}$ such that over each Ω_i, the cosine of ψ_i, the angle between any vector in Ω_i and the normal to the cexel $N_{\mathcal{C}}$ varies by less than ε_M. Then, for every $d\omega \in \Omega_i$,

$$|\cos(d\omega, dA, \mathcal{C}) - \cos(\psi_i)| < \varepsilon_M. \tag{5.21}$$

Therefore, over each Ω_i, if h' is such that

$$h'(d\Omega, dA) = \begin{cases} 1 & \text{if a vector d}\Omega \text{ from } dA \text{ intersects } \mathcal{C} \\ 0 & \text{otherwise,} \end{cases} \tag{5.22}$$

then

$$\left| \int_{\Omega_i} h(d\Omega, dA) d\Omega - \cos(\psi_i) \int_{\Omega_i} h'(d\Omega, dA) d\Omega \right| < \varepsilon_M \Omega_i. \tag{5.23}$$

Therefore, if we calculate the error over the entire solid angle,

$$\begin{aligned} \left| \int_{\Omega} h(d\Omega, dA) d\Omega - \sum_{i=1}^{n} \cos(\psi_i) \int_{\Omega_i} h'(d\Omega, dA) d\Omega \right| &< \sum_{i=1}^{n} \varepsilon_M \Omega_i \\ &= \varepsilon_M \Omega \\ &= \varepsilon. \end{aligned} \tag{5.24}$$

Since the inner integral's approximation is inaccurate by at most ε, then the iterated integral is erroneous by at most $A\varepsilon$. Therefore, by carefully choosing ε' such that $\varepsilon' < \varepsilon / A$, we can achieve arbitrarily accurate approximations, as long as we supersample, choosing sufficiently fine partitions of the solid angle.

Given some small error ε_S, how can we choose a partition such that each portion generates at most an error of ε_S? Without loss of generality, consider a cexel in the xy plane, with a normal of $(0, 0, 1)$. Let (i, j, k) be a vector contained within a member of the partition of the solid angle satisfying the ε_S error requirement. The cosine of the angle between these two is then obtained by using the identity

$$V_a \cdot V_b = N_a N_b \cos(\theta_{ab})$$

which yields

$$\begin{aligned} \cos(i, j, k) &= \frac{(i, j, k) \cdot (0, 0, 1)}{|(i, j, k)||(0, 0, 1)|} \\ &= \frac{k}{\sqrt{i^2 + j^2 + k^2}}. \end{aligned} \tag{5.25}$$

Let δ be such that any vector which differs in any coordinate by at most δ from (i,j,k), differs by at most $\frac{\varepsilon_S}{2}$ in cos. This will be the case if

$$\left| \frac{k}{\sqrt{i^2+j^2+k^2}} - \frac{k}{\sqrt{(i+\delta)^2+(j+\delta)^2+(k+\delta)^2}} \right| < \varepsilon_S,$$

or equivalently, if we measure the angle in polar coordinates, if we choose δ such that the cosine of the azimuth angle differs by less than half the small error, for any vector within δ of (i,j,k). This is achieved if the angle varies by less than $\cos^{-1}(\varepsilon_S)$. Given a vector, $\cos^{-1}$ can be calculated, and this will give us a range, in polar coordinates, of vectors within that small angle. A straightforward calculation will convert this to rectangular coordinates, and ultimately, will allow us to partition members of $\mathcal{S}$, the solid angles, for arbitrarily good approximations.

In practice, the flux from a light to a cexel will only be calculated once, and can be considered as overhead. It may therefore be reasonable to perform more exhaustive computation to obtain accurate values for the power of the initial flux of light from the light sources to the cexels. In a simple implementation, this goal may be overlooked, and the suggested approximations can be used to obtain an approximation of the flux at the first cexels encountered. One may also choose to view extended light sources as emanating power in precisely the representative direction, or the corner directions, and emanating a constant power in each of these directions: this amounts to combining directional extended diffuse light sources.

Chapter 6

Local Illumination

Recall that the FIAT global illumination algorithm begins by propagating the power from the light sources to the first cexel(s) encountered. It is also possible to defer the conversion of a light source's flux to a parallel beam representation until it enters a non-empty cell. The conversion to the cexel representation would happen at this point. In the iteration phase, a cexel's incoming power must either propagate through empty space to a neighbouring cell, or, if objects lie in the path of the light, the interaction of the light with this object must be modelled in order to determine how the light will propagate through the rest of the scene. Now that we have determined how to convert the initial flux from a light source to a discretization in terms of the parallel beams maintained at the cexel level, and how we can propagate parallel beams through empty space, we shall discuss how to deal with light interacting with objects inside a cexel.

Consider the case of a single parallel beam b entering a cell through some cexel $\mathcal{C}$. To model the interaction of this beam with the contents of the cell, a general solution forces us to approximate the behaviour by using a number of samples. By increasing the number of cexels, the thickness of the parallel beams is decreased, as is the error due to this simplification. For this reason, techniques that discretize the parallel beam into smaller parallel beams will not be considered.

6.1 Overview of Surface Models

Before we can analyze what happens to an infinitesimal beam of light as it interacts with an object, we must first examine how surfaces can be modelled. Because the main objective of FIAT is to reduce the global illumination problem to distinct phases of light propagation and local illumination, only local illumination models are considered. While the FIAT algorithm is independent of the actual local illumination model used, it does rely on the latter conserving energy.

In computer graphics, there is an abundance of schemes used to simulate various types of surfaces. In the early 70's, surfaces were first modelled as ideal diffuse reflectors [Gour71], but a desire for greater visual realism soon justified the addition of a specular highlight term[Phon75]. In 1977, James Blinn [Blin77] formulated a more realistic model of local illumination, based on the research in optics of Torrance and Sparrow [Torr67], incorporating the concept of the micro-facet composition of surfaces. Cook and Torrance [Cook81] then furthered this research to include spectral energy distribution.

Since then, global illumination algorithms have made use of variations of these local illumination models. In most radiosity algorithms, surfaces are modelled as perfectly diffuse reflectors, whereas in ray-tracing, more elaborate models involving specular and transmitted components are used. Increasing research has been done in the area of local refraction models[Hall83],[Thom86],[Musg87], and also in the area of anisotropic illumination models, where specular highlights become dependent of the orientation of a surface[Kaji85][Yoko88][Poul89].

6.2 FIAT's Local Illumination Models

The conservation of energy is the fundamental criterion of any local illumination model employed in FIAT. Specifically, the sum of the power reflected, transmitted, and absorbed by an infinitesimal element of surface area, must equal the power of the incoming beam of light. This can be achieved with any illumination model, as long as the quantities of power transferred are normalized to equate to the incoming power. In the current implementation of FIAT, two illumination models are available. Both model surfaces using a diffuse and specular reflection component, as well as a specular refraction component. Conservation of power is achieved by normalizing the transmitted and absorbed power such that they equal the incoming power.

The specular function of the first model is based on Phong's[Phon75] illumination model. Given an incoming beam of light db, a viewing direction V, and the normal N_S to the surface S as in figure 6.1, db is reflected through N_S giving db_R. The cosine of the angle θ between db_R and V is then raised to some exponent n. The power received by V, from dA, due to a specular reflection from db is:

$$P_s(V, dA, db) = \cos^n \theta. \tag{6.1}$$

This exponent gives a measure of how quickly the specular reflection decreases as one moves away from the ideal reflection direction. The specular refraction component is calculated in similar fashion, using the cosine of the angle θ' between the viewing direction V' and the ideal refraction direction db_T, raised to an exponent n':

$$P_t(V, dA, db) = \cos^{n'} \theta'. \tag{6.2}$$

The second model is based on Blinn's[Blin77] local illumination model, which proposes that surfaces are composed of micro-facets oriented in various directions. Given a viewing direction V, and incoming light beam db, as illustrated in figure 6.2, a pseudo-normal H is defined as the vector that bisects the angle formed by V and db. The specular component of db reflected towards V, is a function of the fraction of micro-facets within dA, an infinitesimal element of the surface S, whose normal coincides with H. In the current implementation, this is simply the cosine of θ, the angle between N_S and H, raised to a surface dependent exponent n:

$$P_s(V, dA, db) = \cos^n \theta. \tag{6.3}$$

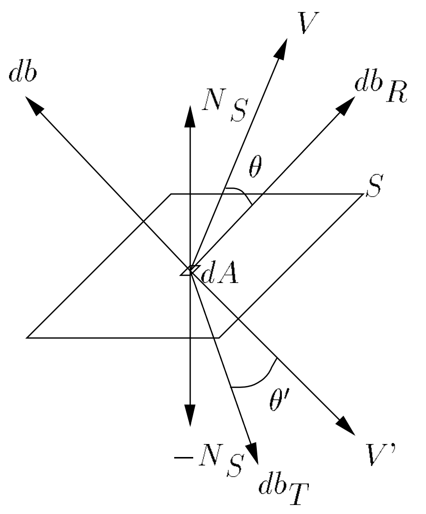


Figure 6.1: Phong-Based Local Illumination Model

The refraction component is calculated in similar fashion. A pseudo-normal H' is calculated such that light coming from db and refracting through a surface with the same refraction index as S, but whose normal coincides with H', would be transmitted in the V' direction. Since the distribution of the micro-facets is fixed for a given surface, this function is the cosine of θ', the angle between $-N_S$ and H', raised to the same exponent n:

$$P_t(V, dA, db) = \cos^n \theta'. \tag{6.4}$$

The specular refraction is a function of the proportion of micro-facets within dA whose normals coincide with H'.

6.3 Parallel Beam Meets a Surface

For simplicity, only the interaction of db, a parallel beam of infinitesimal thickness, with the first object it encounters inside C, is considered. The behaviour of the entire beam is approximated by the behaviour of db by assigning the full power of b to db. Since FIAT possesses more than one local illumination model, an arbitrary model is used in the analysis of the interaction of an infinitesimal beam of light db, with an infinitesimal surface element dA, and of the subsequent distribution of power to cexel(s) of the neighbouring cell.

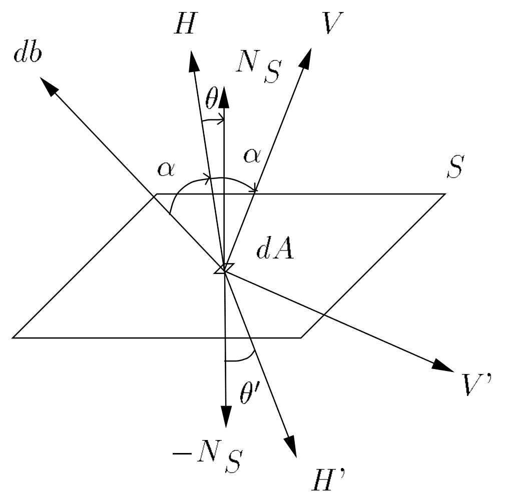


Figure 6.2: Blinn-Based Local Illumination Model

For the remainder of this chapter, a local illumination model $\mathcal{LM}$ is assumed, and it is normalized to conserve power. Given an incoming infinitesimal beam db of power P_{db}, and a viewing direction V, the resulting outgoing power $P(V, dA, db)$ is defined as

$$P(V, dA, db) = \mathcal{LM}(db, V)P_{db}. \tag{6.5}$$

Given an incoming beam db, the local illumination is applied to determine the outgoing power distribution. Assume for now that none of the light leaving dA is intercepted by an object before it leaves the cell. In essence, db transforms dA into a point light source, whose radial distribution of power is defined by the local illumination model. Ideally, for each cexel of the current cell, the set of solid angles which can receive energy from dA should be computed, and then the power radiated by dA in each direction should be assigned to the outgoing parallel beam. This problem is a generalization of the one presented in the preceding chapter, because dA is no longer a perfectly diffuse light source.

Given a cexel $\mathcal{C}$, we must first decide which solid angles can receive light from dA. Since this problem is independent of the distribution of power emanating from dA, the solution is precisely the same as the one described for diffuse point light sources in the preceding chapter. For each solid angle Ω in the set of solid angles which can receive light from dA, the power received from dA due to db must be computed. This

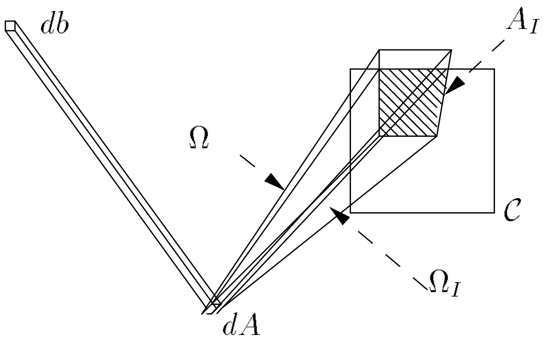


Figure 6.3: Solid Angle of Intersection

power can then be assigned to the outgoing parallel beam of light in the representative direction, as was done in the case of light source flux to cexel conversion.

Given Ω, determine the area of intersection A_I of a solid angle of shape Ω originating at dA. As illustrated in figure 6.3, the solid angle of intersection Ω_I is defined by dA and A_I. Denote the set of all such solid angles of intersection as S_Ω. The power received by $\mathcal{C}$ through Ω from dA due to db is:

$$\begin{aligned} P(\mathcal{C}, \Omega, dA, db) &= \int_{\Omega_I} \mathcal{LM}(db, d\omega) P_{db} d\omega \\ &= P_{db} \int_{\Omega_I} \mathcal{LM}(db, d\omega) d\omega, \end{aligned} \tag{6.6}$$

as P_{db} is fixed.

6.4 An Approximate Solution

A full solution to the local illumination problem requires that the set of solid angles S_Ω of intersection over all cexels of the current cell be determined, as well as the light flux from dA through each member of S_Ω. Now that we understand how the power from an infinitesimal parallel beam of light db is distributed to the cexels of the cell containing the object that db intersect, we can present approximations to these solutions.

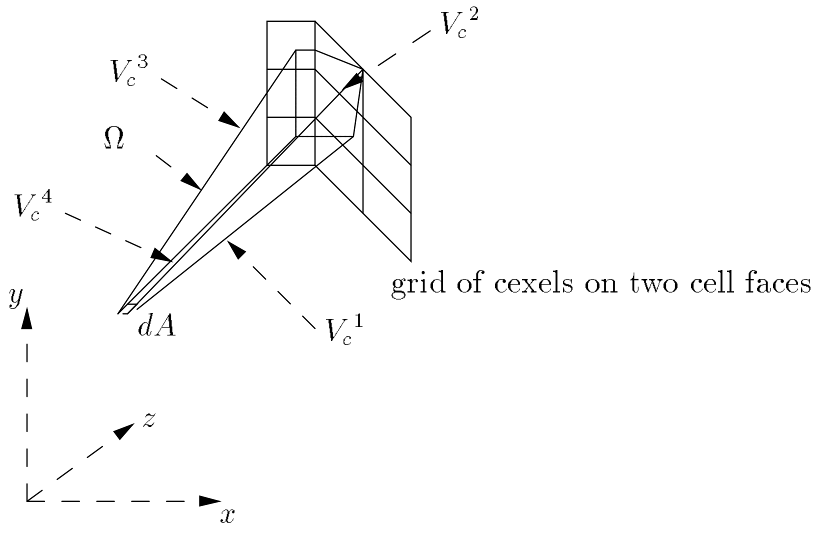


Figure 6.4: Projection of the Four Corners of a Solid Angle

6.4.1 The Solid Angle Intersection Set

The first part of the problem is to determine the set of solid angles of intersection for each cexel. For a fixed solid angle Ω, determine which cexel(s) it intersects. Recall that Ω has four corner vectors $V_c{}^1, \ldots, V_c{}^4$ which are defined to be the four edges of the solid angle, as illustrated in figure 6.4. For each vector, the projected point of intersection can lie on one of three possible cell faces. By projecting these vectors from dA onto the plane associated with each possible face of intersection, we determine the point of intersection lying on the boundary of the cell. Using the slope of each vector, and the distance to travel in each of the x, y and z directions, as illustrated in figure 6.5, the closest point of intersection can be determined.

If all four intersection points lie within the same cexel $\mathcal{C}$, the convexity of both the cexel and the solid angle's projected area of intersection implies that the entire solid angle impinges on $\mathcal{C}$. If the intersection points lie in the same plane, but not all in the same cexel, then the cexels which can receive power from dA through that solid angle must all lie within that plane. These cexels are precisely the ones that are intersected by the convex hull $\mathcal{H}$ that is determined by the four corner vectors projected onto the cexel plane. This is illustrated in figure 6.6. To approximate the solid angle of intersection for each cexel, the appropriate area of intersection with $\mathcal{H}$

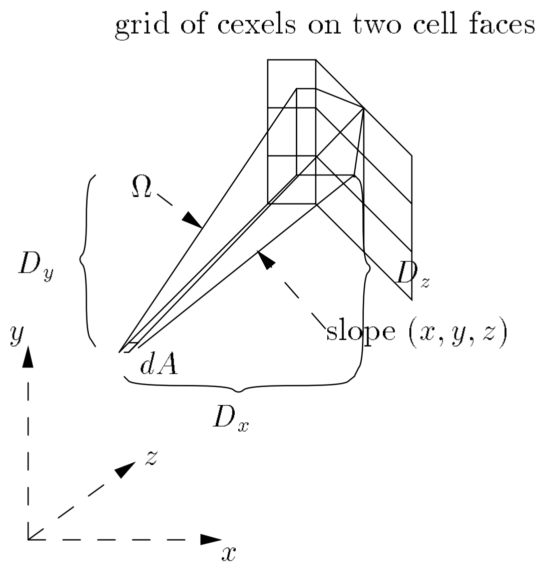


Figure 6.5: Using Slopes to Calculate Projection

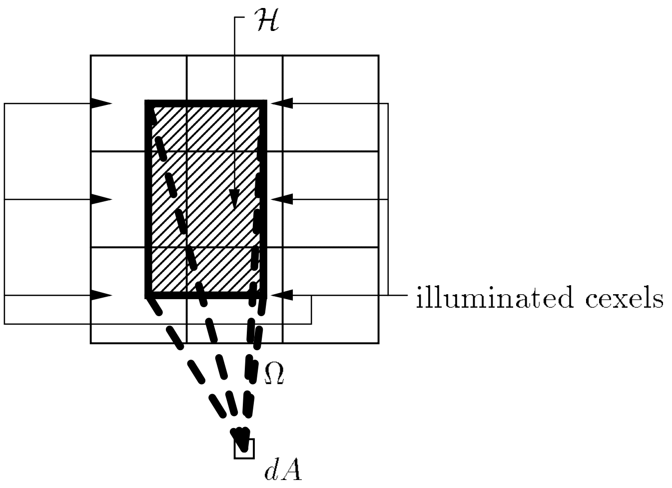


Figure 6.6: Cexels Within Convex Hull

can be used. To avoid further calculations, Ω could also be divided equally among the intersecting cexels.

If not all intersection points lie on the same face of the cell, the cexels which intersect the solid angle can still be determined by defining a convex hull of the four corner vectors onto *each* plane which contains an intersection point. The cexels which lie on one such plane, and intersect that plane's convex hull form the set of cexels receiving energy from Ω. The power from Ω can then be distributed approximately according to either the area of intersection with the appropriate convex hull, or again, divided equally among the cexels.

The advantage of this technique is that the number of intersection calculations is directly proportional to the number of solid angle corner vectors. Since there are only two more corner vectors than there are solid angles, as was seen in the preceding chapter, the number of intersection calculation is proportional to the number of solid angles. As a result, for some solid angles this algorithm will quickly determine the only cexel that is intersected, and for others, it will both determine the cexels which receive part of this light, and estimate the power received.

If execution speed is crucial, this algorithm can be further simplified such that exactly one point of intersection is calculated for each corner vector, giving only two intersection calculations more than there are solid angles. If only one projection is done, the algorithm is simplified in the following fashion. In the case where the four projected vectors of a solid angle do not lie in the same face, only the cexels with an intersection point will receive a fraction of the power. This fraction will be proportional to the number of points falling inside a cexel. Since this is only the case for solid angles which intersect one or more edges of the cell, this approximation will only be severely distorted when a solid angle has to cover a large distance before exiting the cell.

6.4.2 Power per Solid Angle of Intersection

The projection of a representative solid angle from the set of all possible solid angles $\mathcal{GS}$ onto the cexels of the current cell produces one solid angle of intersection per cexel it intersects. The size of S_Ω, the set of solid angles of intersection, is at least the size of $\mathcal{GS}$. Because the local illumination model involves an exponentiated cosine, the integral of the power over a solid angle can only be approximated. Instead of trying to approximate this integral over each solid angle of intersection, we only do so for each member of $\mathcal{GS}$.

For each member Ω of $\mathcal{GS}$, determine the power emitted through a solid angle Ω by dA due to db:

$$P(\Omega, dA, db) = P_{db} \int_\Omega \mathcal{LM}(db, d\omega) d\omega. \tag{6.7}$$

If the value of $\mathcal{LM}(db, d\omega)$ is constant over Ω, the power becomes $P(\Omega, dA, db) = P_{db}\mathcal{LM}(db, d\omega)\Omega$, for some $d\omega$ in Ω. Using d, the central direction of Ω, in that equation, the resulting approximation to the power reflected/refracted through Ω deviates from the exact answer by

$$P_{db} \int_\Omega (\mathcal{LM}(db, d\omega) - \mathcal{LM}(db, d))\, d\omega. \tag{6.8}$$

If a better approximation for the power emanating through a particular Ω is desired, we should take advantage of our knowledge of how Ω intersects the cexels of the cell. If the set of cexels that can receive power through the solid angle is known, then instead of trying to refine the approximation to Ω, it is better to calculate one for some or all of the solid angles determined by the participating cexels. As illustrated in figure 6.8, a sample direction from dA to each cexel is chosen, say the one to the centre of each cexel, and used to approximate the power impinging on that particular cexel. Assuming cexel i defines a solid angle of size Ω_i, the power received by $\mathcal{C}_i$ through Ω_i from dA due to db is approximated by:

$$P(\mathcal{C}_i, \Omega_i, dA, db) \approx \Omega_i \mathcal{LM}(db, V_i) P_{db}, \tag{6.9}$$

where V_i is the vector defined by the centre of $\mathcal{C}_i$ and dA.

The solid angle determined by each cexel can also be approximated. Given a set of n participating cexels, and n vectors V_i, to the centre of each cexel, assign a weight

w_i of 1 to cexel i if its chosen vector is not within[1] the actual solid angle, and 4 otherwise.

The choice of the weight factors is only a heuristic, and is roughly the ratio of the expected size of intersection area when the central vector is inside versus outside the solid angle. Consider a square of side 1, as illustrated in figure 6.7. Without loss of generality, the intersection between this square and a projected solid angle is determined by a point (x', y') and its perpendicular projection onto both the x and y axes, as illustrated in figure 6.7. If the intersection area contains the central point $(\frac{1}{2}, \frac{1}{2})$ then (x', y') is such that:

$$\frac{1}{2} \leq x', y' \leq 1.$$

The expected area of intersection in such a case is:

$$\frac{1}{4} \int_{\frac{1}{2}}^{1} \int_{\frac{1}{2}}^{1} xy \, dx \, dy = \frac{9}{16}. \tag{6.10}$$

If the intersection area does not contain $(\frac{1}{2}, \frac{1}{2})$, the expected area of intersection is:

$$\frac{3}{4} \left(\int_{0}^{1} \int_{0}^{\frac{1}{2}} xy \, dx \, dy + \int_{0}^{\frac{1}{2}} \int_{\frac{1}{2}}^{1} xy \, dx \, dy \right) = \frac{7}{48}. \tag{6.11}$$

The ratio of the two expected areas of intersection is:

$$\frac{9/16}{7/48} = \frac{27}{7} \approx 4. \tag{6.12}$$

Summing up these weights, say to $\mathcal{W}$, the estimate of the solid angle Ω_i is then:

$$\Omega_i \approx \frac{w_i}{\mathcal{W}} \Omega. \tag{6.13}$$

[1] This can be achieved by projecting the chosen vector onto the same plane of the cube which contained the square defining Ω.

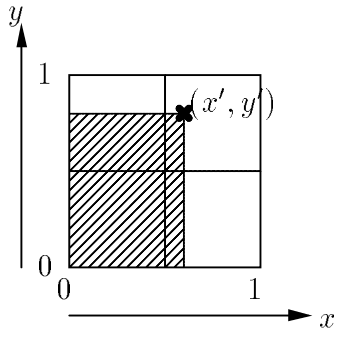


Figure 6.7: Weighted Area of Intersection

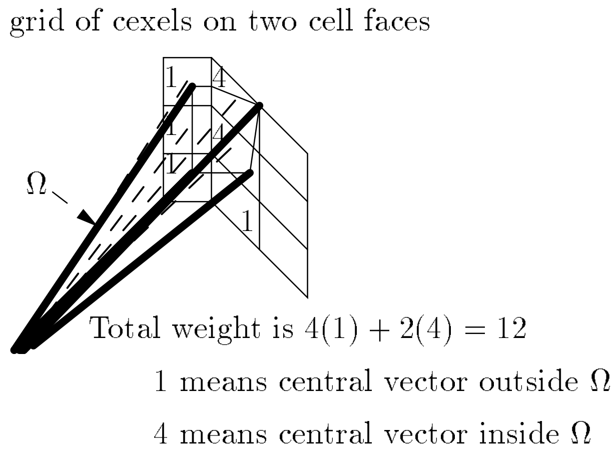


Figure 6.8: Power of a Solid Angle

Replacing this for Ω_i in equation 6.9, the approximate power received by $\mathcal{C}_i$ is:

$$P(\mathcal{C}_i, \Omega_i, dA, db) \approx \frac{w_i}{\mathcal{W}} \Omega \mathcal{L} \mathcal{M}(db, V_i) P_{db} \tag{6.14}$$

If the solid angle belongs to only one cexel, then we can choose additional sample values of $d\omega$ over Ω.

Finally, observe that the error due to these approximation can be reduced by increasing the resolution of the grid which defines the solid angles, that is, by increasing the number, and decreasing the average size, of the solid angles. In the results chapter, we will show the effects of varying the number of solid angles on the solution to the global illumination problem.

Chapter 7

Computing an Image

In the preceding chapters, we analyzed how FIAT can model the flow of light through an environment as it interacts with objects and transforms them into secondary light sources. We now delve into the problems that arise when we compute an image using the information gathered through the process of simulating the flow of light.

7.1 Storage Requirements

The choice of method used to generate an image from this data is restricted by the substantial computational requirements of the simulation. We cannot discuss image generation without first considering storage issues. As the number of cexels and beams necessary to model the flow of light through the environment controls the storage requirements of the algorithm, we must strive to minimize these. We now illustrate this point through a simple example.

Suppose a grid of size 8 is used to define the set of solid angles and associated representative directions. As previously noted, this defines a set of size $6(GS^2) = 6(8^2)$. Because each cexel is only responsible for the solid angles which are part of its incoming hemisphere, each cexel is required to maintain information for up to $3(8^2) = 192$ of those solid angles. If we wish to implement an algorithm for which 6 levels of subdivision are permitted, and such that the entire face of the smallest cell is covered by exactly one cexel, then the initial bounding volume can be partitioned into

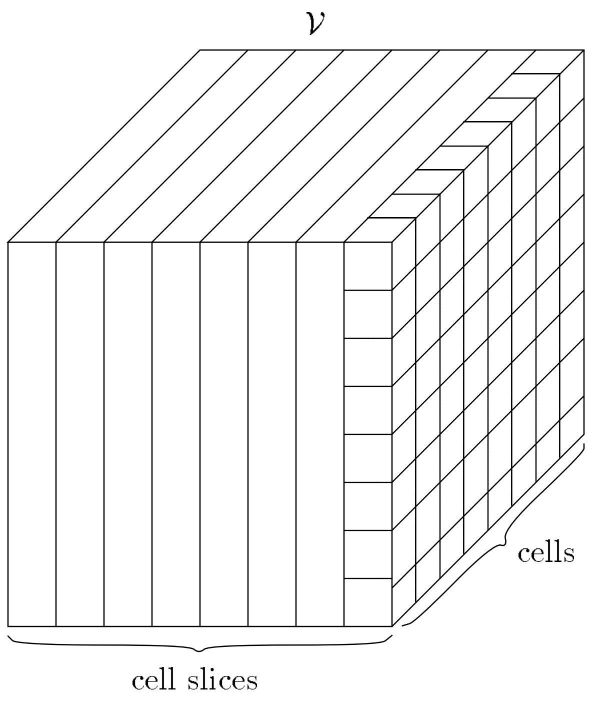


Figure 7.1: Slices of Cells Across Bounding Volume

$(2^6)^3 = 2^{18} = 256K$ such cells. Since each smallest cell has 6 faces which are covered by one cexel, a cell will potentially record flux information for $6{\cdot}192 = 1152$ different directions. In the worst case scenario, where maximum subdivision is achieved, and each cell is forced to maintain complete cexel information, the number of light beams stored will be $256K{\cdot}1152 = 288M$.[1] If at least one byte is used to store power and offset information at each cexel, the worst case storage requirement could exceed 288 megabytes.

If instead of preserving all the flux information throughout the entire environment, we only maintain the non-propagated flux information within the cexels, we can substantially reduce the storage cost of FIAT. Rather than maintaining all the illumination information, we discard the light which has already propagated through a cell.

Imagine an implementation of FIAT that successively iterates over cell slices of the bounding volume, as illustrated in figure 7.1. Each slice is processed until no flux

[1]The letters K and M represent the numbers 2^{10} and 2^{20} respectively.

exists between cells within the same slice. Neighbouring slices are then processed in similar fashion, until the algorithm halts. Since only one slice is active at any given time, only one slice can have all of its cell maintaining lighting information on all faces. All other slices have no inner flux, and hence each cell maintains information on at most two faces, assuming that propagated light is discarded. This corresponds to a two thirds savings in terms of number of light beams stored.

In general, it is difficult to estimate the savings achieved by discarding the light that has propagated through a cell, since the order in which the cells are processed is not known beforehand. Because of FIAT's voracious appetite for storage, we opted for this technique in our propagation of light. This requires that the light entering a cell is discarded as soon as we have determined how it interacts with the contents of the cell, and propagate it to neighbouring cell. In our current view dependent implementation, the incoming light is first used to maintain a *Light Update Buffer*(LUB).

7.2 The Light Update Buffer

Before we introduce the LUB, let us recall the concept of view dependence. Given a fixed viewing position V, as well as a window $\mathcal{W}$, positioned along the line of sight, the visible scene is defined by the viewing frustum V_F centred at V and passing through $\mathcal{W}$. Let $\mathcal{W}$ be partitioned into a $w \times h$ array of pixels $P(i,j)$ of equal size, as in figure 7.2, such that $1 \leq i \leq w$ and $1 \leq j \leq h$. The task of constructing a view-dependent image of the visible scene then corresponds to determining the lighting information coming from the scene, through each pixel, to the viewpoint.

The idea of the LUB is based on the following. Since we assume that light accumulates linearly, the order in which we collect lighting information is irrelevant. We take advantage of this by collecting lighting information each time we balance a cell. Since objects that fall within V_F are the only ones that can be directly visible, then for each pixel we can determine the surfaces which are directly visible from V, using a simple depth comparison. An element of surface within the viewing frustum is said to be directly visible if it is the first element of surface intersected by some

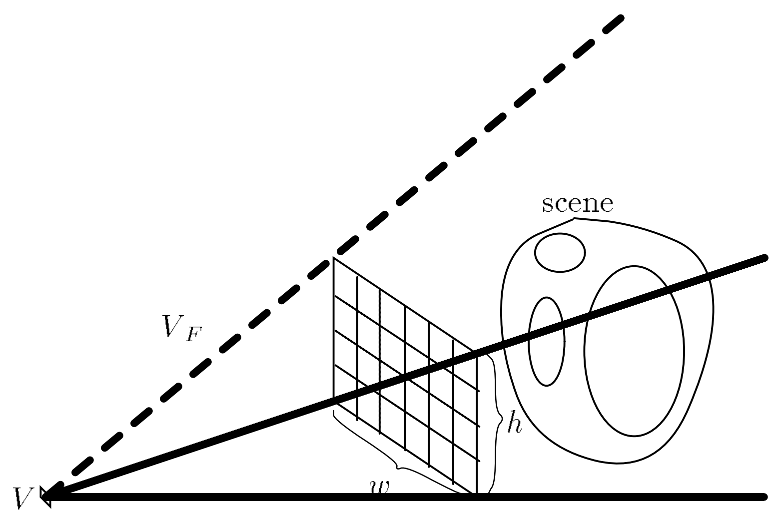


Figure 7.2: View Dependence

ray originating at V. In particular, each time we balance a cell, we can determine the set of visible surfaces, and the corresponding set of pixels that can receive lighting information. The Light Update Buffer is simply an array used to accumulate the incremental power received by each pixel, at each iteration of the cell balancing algorithm.

7.2.1 Shading a Pixel

Using the LUB, we can collect lighting information that allows us to shade each pixel of the window. Given a pixel, we wish to determine the power crossing it due to the elements within the scene. The cost of this is a function of the number of incoming beams — which is a function of the number of cexels and number of representative directions — and of the precision required for this calculations. The precision of an approximation is dependent on the size of the viewing frustum determined by the pixel and the viewpoint. The goal is to maximize the accuracy of the lighting computed, while keeping the cost low.

Each time a cell is balanced, we use the incoming light to update the LUB, incrementally determining the power received at V, through each pixel. Given a pixel $P(i,j,)$, we determine the set of infinitesimal surface elements $A_{vis}(i,j)$ that are visible from V, through $P(i,j)$, as illustrated in figure 7.3. To allow us to calculate

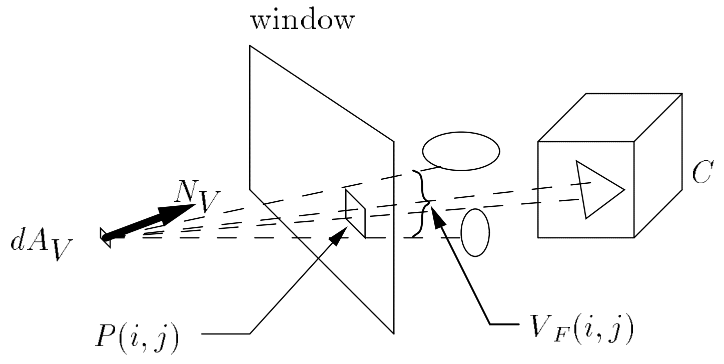


Figure 7.3: Contribution to LUB From a Cell

the power impinging on the view point due to a visible surface element, we place an infinitesimal element of surface dA_V at the view point, whose normal coincides with the viewing direction. For each $dA \in A_{vis}(i,j)$ inside the cell C, we calculate the power $dP(dA, dA_V)$ emanating from dA in the direction of dA_V.

We have assumed that the normal to dA_V coincides with the viewing direction, and denote the subset of $A_{vis}(i,j)$ that lies within C as $A_{vis}(C,i,j)$ The incremental power contribution that we must add to the LUB entry for $P(i,j)$, due to this cell is:

$$\Delta P(C,i,j) = \int_{A_{vis}(C,i,j)} P(dA, dA_V)dA. \tag{7.1}$$

To estimate this integral, we use the surface element directly visible from V, through the centre of $P(ij)$, as our sample. Since we use one sample for the entire visible frustum of $P(i,j)$, the approximation to the power is not over $A_{vis}(C,i,j)$, but over $A_{vis}(i,j)$. It is an approximation to the incremental power contribution $\Delta P(i,j)$ that must be added to the LUB for $P(i,j)$ due to the entire scene:

$$\Delta P(i,j) = \int_{A_{vis}(i,j)} P(dA, dA_V)dA. \tag{7.2}$$

The solid angle formed by dA as viewed from V is

$$\frac{dA \cos \psi''}{D^2}. \tag{7.3}$$

As illustrated in figure 7.7, ψ'' is the angle between the vector from dA to dA_V and N_{dA}, and D is the distance between dA and dA_V. Define $V_F(i,j)$ as the solid angle formed by the view dependent frustum that intersects $P(i,j)$, as illustrated in figure 7.3. If we approximate the power received by dA_V through $P(i,j)$ by supposing that the power per infinitesimal solid angle is constant and equal to dA's power per infinitesimal solid angle, the approximation to the power received by that pixel becomes

$$\Delta P(i,j) \approx V_F(i,j) \frac{D^2}{dA \cos \psi''} P(dA, dA_V) \tag{7.4}$$

If this surface element is not within the cell, the cell does not cause the corresponding entry in the LUB to be updated. The LUB will is updated when the cell containing the representative elementary surface element will is balanced.

7.2.2 Shading a Surface Element

Having determined a sampling for pixel updates, we now consider the calculation of $P(dA, dA_V)$. Given a selected surface element dA that is within the cell C that we are in the process of balancing, we use the incoming flux to determine the quantity of light power re-emitted by dA towards dA_V. This problem can be separated into two distinct tasks. Given the set of incoming parallel beams, we must determine which beams illuminate dA, and for each of those, the flux transmitted in the direction of the view point.

7.2.2.1 Determining the Contributing Beams

Given the set of incoming parallel beams, we must determine which ones illuminate the surface element that we are trying to shade. It is sufficient to show how we can decide which beam(s), from an arbitrary incoming direction, impinge onto dA. The actual shape of the parallel beams, which has had little relevance in the phases of FIAT that we have thus far presented, will now become essential in determining the contributing beams.

For a fixed direction D, a simple method to determine contributing beams from

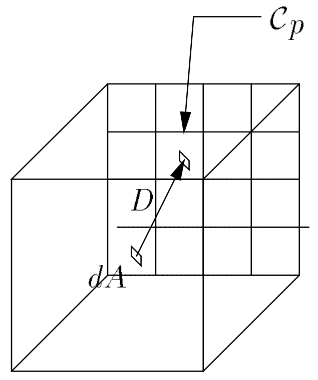


Figure 7.4: Determining the Contributing Beams

that direction is to project dA onto the closest face of the cell, as illustrated in figure 7.4. If the orientation dependent beams of chapter 4 are used, and if no offset terms are used, then the only possible contributing beam is the one belonging to the cexel $\mathcal{C}_p$ containing the projected surface element.

If we use these same beams, but also maintain a propagation offset term, the set of cexels which *can* contribute a beam is not limited to the cexel containing the projected surface element: the offset term can make the centre of a parallel beam deviate from the centre of a cexel by as much as a full cexel. All cexels on the same face as $\mathcal{C}_p$, and that are immediate neighbours, can be part of this set of cexels. Their centre can deviate enough to coincide with the centre of $\mathcal{C}_p$, thus they can also illuminate dA. This can be readily determined by comparing the distance from the offset centre of each cexel, to the projected point of intersection.

Since the offset term can differ by as much as a cexel in all three directions, it is possible that a cexel on a different face of the cell than $\mathcal{C}_p$ would be able to illuminate dA. A simple solution is to project the surface onto the plane containing each of the three possible intersection faces, as shown in figure 7.4. The possible intersection faces are the (at most) three that can be intersected by a vector in the representative direction, which originates anywhere in the cell. For each plane, we determine the set of cexels that can illuminate dA, and proceed to examine the cexels that are actually within the cell, in the same fashion as previously explained.

If we wish to use orientation independent parallel beams, the problem is more

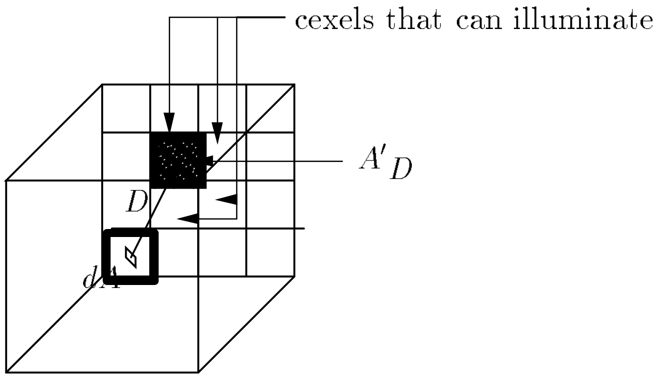


Figure 7.5: Contribution from Orientation Independent Beams

difficult. For the given direction D, we know that the cross-section is independent of the orientation of a cexel. If we ignore the offset term, the cexels that can illuminate dA are those whose beam in the direction D contains the ray (dA, D) that originates at dA, and is parallel to D, as shown in figure 7.5. Project a beam of the same cross-section — as all the parallel beams in the chosen direction — and centred on (dA, D) onto the faces of the cell, giving a projected area A'_D. Because the cross-section of all D beams is the same, a beam from cexel $\mathcal{C}$ will only illuminate dA if its centre is within A'_D. Since at low grazing angles, this projection can cover many cexels, the number of cexels to consider can be larger than for the orientation-dependent parallel beams.

To take into account the propagation offset, the projected area in the above argument becomes much larger. In effect, the projected area that determines the set of possibly contributing cexels is determined by placing a cube whose side length is twice the side length of a cexel. Recall that the offset term can vary by as much as a full cexel in each coordinate. Therefore, the centre of a parallel beam that projects onto dA, can be anywhere within a D beam cross-section of a point within the cube centred at dA. Thus to determine the set of relevant cexels, we project a beam of same cross-section from each corner of the cube, onto the faces of the cexel, and determine the convex hull of the projected areas (see figure 7.6). Because of the convexity of the cube and of the cross-sectional area of the beams, only the cexel whose centre lies within the convex hull can possibly illuminate dA. The offset term and location of

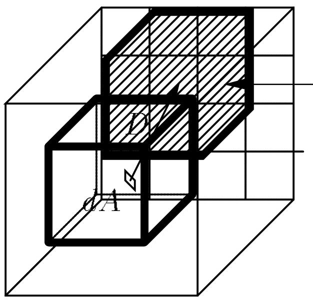


Figure 7.6: Independent Orientation and Error Terms

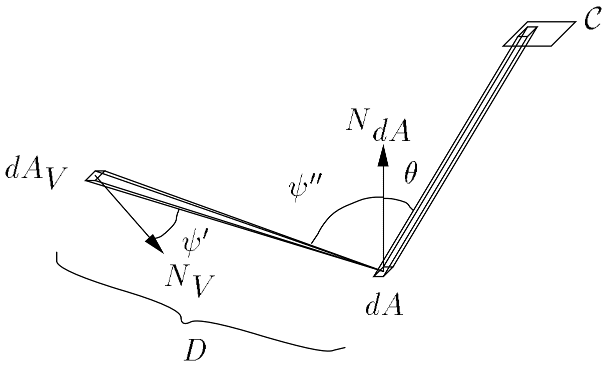


Figure 7.7: Local Shade of a Surface Element

each can then be used to determine if its beam in direction D illuminates dA.

7.2.2.2 The Contribution of One Beam

Having determined the beams which impinge on dA, we must calculate the power for the beam that is transmitted towards the viewing position. Suppose a parallel beam b impinges on dA, and possesses a power of P_b, as illustrated in figure 7.7. Letting the angle between b and N_{dA} be θ, a cross-section of size $dA\cos\theta$ of the beam is intercepted by dA.[2] If the cross-sectional area of the beam is A_b, then the power impinging on dA is

$$dP(dA,b) = P_b \frac{dA\cos\theta}{A_b}. \tag{7.5}$$

Using the local illumination model, we next compute the fraction of $P(dA,b)$ that is reflected or refracted towards dA_V. For any model $\mathcal{LM}$, the fraction of power

[2]The cross-sectional area of the beam depends on the type of parallel beams used, but can be easily tabulated before hand.

transmitted through an infinitesimal solid angle $d\omega_V$ is:

$$\mathcal{LM}(db, d\omega_V). \tag{7.6}$$

where db is the infinitesimal beam of light from b which impinges on dA, and the local model applies for both reflection and refraction.

Since it is crucial that our local illumination model conserve power, we normalize the contribution, dividing by the multivariate integral of the local model over both the hemisphere of reflection and the hemisphere of refraction. The solid angle corresponding to an entire sphere is 4π; thus we write this integral as:

$$\int_{4\pi} \mathcal{LM}(db, d\Omega) d\Omega. \tag{7.7}$$

As illustrated in figure 7.7, the angle between the normal to dA_V and the viewing direction from dA to V is ψ', and the distance between the two is D. The solid angle subtended by dA_V from dA is then $\frac{dA_V \cos\psi'}{D^2}$. By assuming that the power transmitted through any infinitesimal solid angle within that solid angle is constant and as indicated by expression 7.6, then the power received by dA_V is

$$P(dA_V, dA) = \frac{dA_V \cos\psi'}{D^2} P(dA, dA_V) \frac{\mathcal{LM}(db, d\omega_V)}{\int_{4\pi} \mathcal{LM}(db, d\Omega) d\Omega}, \tag{7.8}$$

While the expression in the numerator is straightforward to calculate, the integral in the denominator must be approximated for most local illumination models.

For example, suppose we have a Blinn based illumination model, such that the specular reflection component $\mathcal{LM}_S$ is

$$\mathcal{LM}_S(db, d\Omega) = k_S \cos^n \theta, \tag{7.9}$$

where k_S is a specular reflection constant, θ is as illustrated in figure 6.2, and n is a measure of how quickly reflections fade as we move away from the mirror reflection direction. The area under the micro-facets distribution function is:

$$k_S \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \cos^n \theta \, d\theta. \tag{7.10}$$

Using numerical integration methods, this integral can be calculated, and stored in a table indexed by each value of n required by the scene. If a large number of n's are used, we can approximate each integral using an interpolation table. Since the specular refraction component is similarly based on a micro-facet model, the distribution of facets is unchanged, and the integral remains the same. Dividing by this integral, the resulting function is a statistical distribution of the micro-facets.

The total specular power due to db is then bounded above by:

$$\left((k_S + k_t) 2\pi \int_0^{\frac{\pi}{2}} \cos^n \theta \, d\theta \right) P(dA, b), \tag{7.11}$$

where k_t is the specular refraction component. The total power due to db can then be obtained by adding the total diffuse power to the total specular power. The diffuse reflection component represents a fraction of the power of db that is reflected uniformly through all infinitesimal solid angles of the reflecting hemisphere. In its simplest form, the total power diffusely reflected is a pixed fraction of P_{db}, say k_d. The total transmitted power due to db is then bounded above by:

$$\left(k_d + (k_S + k_t) 2\pi \int_0^{\frac{\pi}{2}} \cos^n \theta \, d\theta \right) P(dA, b). \tag{7.12}$$

With the knowledge that the total transmitted power due to db is, say, P_t, we can normalize the evaluation of the local model in the view direction, to give a power per infinitesimal solid angle, and hence an approximation to the surface element's contribution:

$$P(dA_V, dA) \approx \frac{dA_V \cos \psi'}{D^2} \frac{\mathcal{LM}(db, d\omega_V)}{P_t} P(dA, db), \tag{7.13}$$

Once the contribution from each incoming parallel beam has been collected, we approximate the power received by dA_V, in the frustum defined by $P(i,j)$, by applying equation 7.4. Suppose that the total power re-emitted by dA towards dA_V is $P_t(dA, dA_V)$. The total power received at V due to the pixel can be approximated by:

$$\begin{aligned}
\Delta P(i,j) &\approx V_F(i,j)\frac{D^2}{dA\cos\psi''}P_t(dA, dA_V) \\
&= V_F(i,j)\frac{D^2}{dA\cos\psi''}\frac{dA_V\cos\psi'}{D^2}\frac{\mathcal{LM}(db, d\omega_V)}{P_t}P(dA, db), \\
&= V_F(i,j)\frac{D^2}{dA\cos\psi''}\frac{dA_V\cos\psi'}{D^2}\frac{\mathcal{LM}(db, d\omega_V)}{P_t}P_b\frac{dA\cos\theta}{A_b}, \\
&= V_F(i,j)\frac{\cos\psi'\mathcal{LM}(db, d\omega_V)cos\theta}{\cos\psi'' P_t A_b}dA_V P_b \qquad (7.14)
\end{aligned}$$

This quantity is the incremental power that has to be added to the LUB entry for pixel (i,j).

Chapter 8

Implementation Modalities

In this chapter, the modalities of the current implementation of FIAT are discussed. First, the current implementation details are compared to the theoretical discussions of the preceding chapters. Next, the problems encountered in the current implementation are discussed, and where possible, some solutions proposed. Finally, the general capabilities of FIAT are discussed.

8.1 Modalities

The differences between the current implementation and the earlier discussion of each modality are highlighted in this section. The propagation of light, the conversion of light sources to cexel fluxes, and the application of the local illumination model are the major differing components in the implementation with respect to the previous discussions.

8.1.1 Light Propagation

In the current implementation, light propagation is done using parallel beams principally modelled as rays. For the purpose of propagation, the behaviour of a beam of light is approximated by that of the ray in the representative direction of the beam. This is correct if the beam travels through empty space, but the interaction of a beam with an object may be missed if the representative ray does not intersect it. When an object is intersected, the behaviour of the entire beam is again approximated by the behaviour of the central ray.

The only time in which beams are not modelled as rays arises when the LUB is updated. As was explained in chapter 4, various types of cross-sectional areas can be used to define parallel beams. Since beams are not modelled as rays only at image generation time, the actual cross-sectional area of a beam is relevant only when the contribution to a pixel is to be determined.

In the current version of FIAT, an orientation-dependent definition of parallel beams is used. Consider a surface element dA whose contribution to a pixel $P(i, j)$ must be determined. Using orientation-dependent cross-sectional areas, the set of beams which can illuminate dA from a given representative direction is easily determined by projecting the point onto each plane corresponding to a face of the current cell, as explained in the previous chapter. Centring a cexel at the projected points, only the cexels that are intersected by this "projected cexel" can contain beams that can illuminate dA. It is easier to determine contributing beams when orientation-dependent beams are used instead of orientation-independent beams because the cross-sectional area of a beam crossing a face of a cell is always defined by a cexel centred about the central ray, and parallel to this face. For orientation-independent beams, the cross-sectional area of a beam is defined by the direction (c.f., chapter 4) of the beam, and hence the intersection of this beam with a face is not necessarily the same size as a cexel. The actual cross-sectional area of a beam is the area of a cexel, multiplied by the cosine of the angle between the representative direction of a beam and the normal to the face it leaves. This cross-sectional area is used when computing one beam's contribution to the power received at the viewpoint from dA, in equation 7.14

8.1.2 Light Sources

In the current version of FIAT, three types of light sources are supported: global directional, extended directional, and extended diffuse. Diffuse point light sources have yet to be implemented; this is not a major concern as point light sources should be easy to implement, and are the least realistic of the light sources. Global directional and extended directional are straightforward to implement, and were implemented as outlined in chapter 5, except that only triangles are currently handled as extended directional light sources. It is straightforward to construct a mesh of triangles to synthesize larger light sources.

Only triangular extended diffuse light sources are implemented. The technique used is simpler than the one outlined in chapter 5. Instead of two distinct phases where in the first the set of solid angles contributing to a cexel is determined, and in the second each solid angle's contribution is calculated, the two phases are combined into one.

Given a solid angle and its representative direction, the triangle corresponding to the light source is projected in the representative direction onto the cexel plane. The solid angle is then added to the set of contributing solid angles for each of the cexels intersected by the projection. The contribution assigned to each intersected cexel is the power contained in the solid angle leaving the light source, multiplied by the fraction of the projected area of the source that intersects the cexel. This is the first approximation suggested for computing a solid angle's contribution in chapter 5.

The major drawback of this method is that the set of contributing solid angles for a cexel is not correctly computed. This is a consequence of the simplicity of the test that decides if a solid angle contributes to a cexel; it is possible for a cexel to receive light from a solid angle, without receiving any of it from the representative direction. In particular, it is possible to completely ignore the contribution from a light source to a cexel. In addition, because the projection of a solid angle's corner vectors are not used to determine the contributing solid angles, it is more difficult to determine when the calculation for the power impinging on a cexel can be reduced to the simple case of chapter 5[1]. Since only the representative direction is used, a

[1]Recall that if a cexel lies within each of the projections of the light source along the four corner vectors corresponding to a solid angle, the power impinging on the cexel due to the light source is exactly as described in equation 5.14.

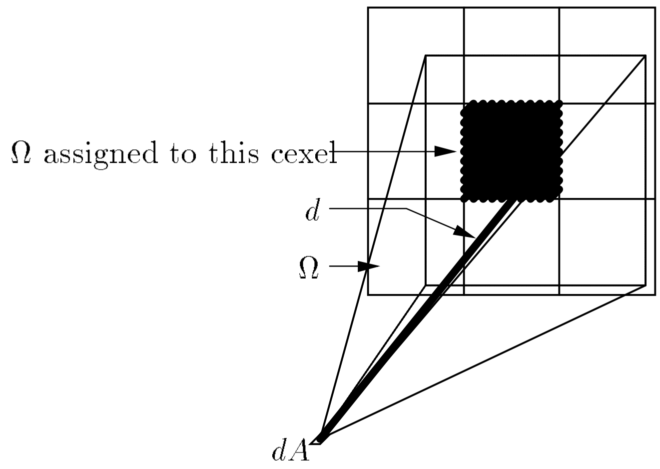


Figure 8.1: Solid angle assigned to one cexel

test to determine if the power received by a cexel can be reduced to the simple case would involve projecting the light source in all of the representative directions of the solid angles adjacent to the current one. If each projection contains the cexel, the power received by the cexel can again be calculated using equation 5.14. If not all the projections contain the cexel, it is still possible that the contribution from the solid angle could be calculated using the simple formula, but this cannot be determined without additional tests.

8.1.3 Local Illumination

We now examine how the current treatment of local illumination in our implementation differs from the approach suggested in chapter 6. Given an incoming beam impinging on some surface element dA within a cell, the power re-emitted by dA due to this beam is calculated for all possible outgoing solid angles. Recall that the problem of distributing the power leaving dA can be decomposed into two distinct problems: determining which solid angles contribute to the illumination of a cexel, and determining the contribution of each solid angle.

Currently, each solid angle Ω is handled such that it only contributes to the cexel intersected by a ray in its representative direction d, as illustrated in figure 8.1. If the number of cexels is small, with respect to the number of solid angles, this approximation is reasonable since many solid angles will correctly contribute to the illumination of only one cexel. On the other hand, if the number of cexels in the cell is the same or greater than the number of solid angles, many solid angles will impinge on more than one cexel. This would justify a more careful calculation of the set of cexels to which a solid angle can contribute.

For each solid angle Ω, the entire power leaving dA within Ω is assigned to the cexel intersected in the representative direction. Due to the practical restrictions that limit the depth to which cells can be subdivided, it is possible that many objects lie within the same cell. When such is the case, a test is performed to determine if an object lies between dA and the intersected cexel: if there is one, the current implementation assumes that this object absorbs all the light. This behaviour is reasonable in the sense that only small cells should contain multiple objects. In such cases the light propagates back and forth between close objects, causing most of the power to be absorbed, never reaching the eye.

The total power contained in a solid angle is approximated using the infinitesimal power leaving dA in an infinitesimal solid angle centred about the representative direction. Assuming that this power is constant over the solid angle, the total power is obtained by multiplying the infinitesimal power by the size of the solid angle. To ensure that the local illumination model conserves power, the total power is calculated as the sum of the approximation for each solid angle. Knowing the total power that will be assigned to cexels due to light leaving dA, it is simple to normalize the outgoing power such that energy is conserved.

To illustrate this, suppose that a beam b impinging on a surface element dA contains a power of P_b, and that a fraction k_s of this power is to be specularly reflected. Suppose that we calculate the reflected power over the reflection hemisphere with respect to dA, and that this hemisphere is partitioned by the set of n solid angles $\{\Omega_i | 1 \leq i \leq n\}$. Let P^s_{ai} be the approximation to the specular power in solid angle Ω_i. The total specular power contained in the approximations over the reflection hemisphere is:

$$P^s_a = \sum_{i=1}^{n} P^s_{ai}.$$

If we use:

$$P_i^s = k_s \frac{P_{ai}^s}{P_a^s} P_b$$

as the actual specular power assigned to each solid angle Ω_i, the total specular power will be $k_s P_b$, as desired. The diffuse reflection, and specular refraction components can be similarly scaled to conserve power.

8.2 Implementation Difficulties

In this section, the major difficulties encountered in the implementation will be discussed. In particular, we examine how light propagation and local illumination are affected by discretization, and shall suggest some improvements.

8.2.1 Discretization Problems

Modelling the flow of light as it interacts with the environment is the principal idea behind FIAT. The parameters that affect the accuracy of propagation are the type of beams used, the number of solid angles used to sample the sphere of possible light directions, and the size of the cexels which in turn define the size of the beams used. Since the current implementation only deals with parallel beams, we will only examine the effects of the last two parameters in relation to propagation using parallel beams.

As was illustrated in chapter 7, the use of cexels and solid angles to model the flow of light through an environment is very expensive. Because of the physical limitations of the target machines on which FIAT is currently implemented, the number of solid angles and cexels that can be used is very restricted. For example, a simple scene with eight polygons, computed using an instance of FIAT which has a grid size of 8 (i.e., 384 representative solid angles), a maximal number of 5 subdivisions, and a 4×4 grid of cexels on each of the smallest possible cell, requires in excess of 140 megabytes of memory. The memory requirements is not so much a function of the number of objects as it is a function of the complexity of the scene and the interaction of light between the various components. In particular, doubling the number of polygons in

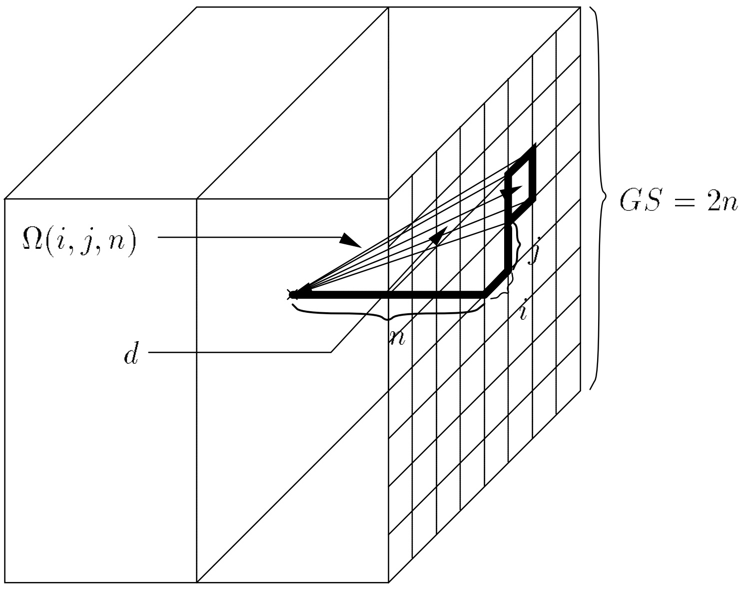


Figure 8.2: Solid Angle

the scene does not imply doubling the memory requirements of the algorithm. This justifies examining the effects of using a small number of solid angles, using larger parallel beams, or both.

8.2.1.1 Solid Angles

Because the amount of storage required to maintain information at the cexel level for a grid of size GS is proportional to the square of GS, it is important to use as few directions as possible in order to reduce the overall storage requirements of the implementation. This is a trade off since the greater the number of solid angles used, the smaller will be the errors in representing the possible light directions.

Suppose that we use a grid of size $GS = 2n,\ n \in \mathbb{N}$, to define the solid angles used in an instance of FIAT. Since any direction that falls within a solid angle Ω is assigned to the representative direction d of Ω, a measure of the directional error incurred by this is the maximum angle between d and any vector within Ω. We now calculate this maximum error over all solid angles for a given grid of size $GS = 2n$.

Let a square on the grid whose vertices correspond to the points $\{(i, j), (i, j+1), (i+1, j), (i+1, j+1)\}$ define the solid angle $\Omega(i, j, n)$, as illustrated in figure 8.2. Without loss of generality, assume that $0 \leq j \leq i \leq n$, since the solid angles are

n	1	2	3	4	5	6	7
maximal error	35.264	19.471	13.263	10.025	8.049	6.721	5.768

Figure 8.3: Maximal Error for Sample Grid Sizes

defined symmetrically in i and j by a grid of even size. The representative direction d of $\Omega(i,j,n)$ is defined as $d = (x_d, y_d, n) = (i+\frac{1}{2}, j+\frac{1}{2}, n)$. We shall find a direction $d' \in \Omega(i,j,n)$ such that the angle between d' and d is maximal. It can be shown that this occurs when $d' = (i,j,n)$, in which case the cosine of the angle between d and d' is:

$$\cos(\angle(d,d')) = \frac{(x_d, y_d, n)\cdot(i,j,n)}{|d|\cdot|d'|}, \tag{8.1}$$

where $|\ |$ is the euclidean norm, and $\angle(d,d') < \frac{\pi}{2}$. It can also be shown that this quantity achieves a minimum when both i and j are 0. Since the angle between two vectors is inversely related to the cosine of the angle, the maximum deviation within a solid angle occurs for the solid angle $\Omega(0,0,n)$. The vector $(0,0,n)$ is the one that deviates the most from the representative direction. The cosine between this vector and the representative direction is:

$$\begin{aligned}
\cos(d,d') &= \frac{(\frac{1}{2},\frac{1}{2},n)\cdot(0,0,n)}{|(\frac{1}{2},\frac{1}{2},n)|\cdot|(0,0,n)|} \\
&= \frac{n^2}{(\frac{1}{2}^2 + \frac{1}{2}^2 + n^2)^{\frac{1}{2}}(0^2+0^2+n^2)^{\frac{1}{2}}} \\
&= \frac{n}{(n^2+\frac{1}{2})^{\frac{1}{2}}}.
\end{aligned} \tag{8.2}$$

This can be converted to a maximal error. The table in figure 8.3 gives the maximal error in degrees for a set of solid angles for various grid sizes $GS = 2n$.

The images in figure 8.4 illustrate a simple scene rendered with different grid sizes. It consists of a white triangular diffuse light source in the background, and three planes in the foreground. The left plane represents a mostly red wall which reflects light in a purely diffuse fashion. The right plane represents a mirror, and the bottom plane is a white floor which reflects light predominantly in a specular fashion.

The seven images illustrate the effects of the grid sizes $2n$, for $n = 1, \ldots, 7$. All seven images were computed under the same conditions, using one level of subdivision, and only one cexel per smallest cell. Notice the gradual improvement in the continuity of colour on the left wall, and on the floor: the areas of darkness (middle, front for left wall, right, front for floor) progressively disappear as the sampling of angles increases.

While the quality of the image improves with the number of solid angles, the cost of maintaining this information may not justify using a very high resolution. For instance, the image computed with a grid of size 7 uses 3 times as many solid angles as the image with a grid size of 4, but does not necessarily produce an image with three times the quality. For this reason, the current version of FIAT uses a grid of size $n = 4$ for most images computed, compromising directional accuracy for smaller storage requirements. In the remainder of this discussion, we will omit the definition of a grid size unless it is not $n = 4$.

8.2.1.2 Cexels

The second important factor which contributes to the amount of storage required by an instance of FIAT is the number of cexels at which information concerning the representative solid angles is stored. Recall from chapters 3 and 4 that given a maximal subdivision of n, and a coverage number m such that m^2 cexels cover the face of the smallest possible cell, the maximum number of cexels required is $6 \cdot m^2 \cdot 2^{3n}$. If in addition the length of the side of the bounding volume is M, the length of the side of a cexel is $\frac{M}{m \cdot 2^n}$.

Since the parallel beams used have a cross-sectional area that is proportional to the area of a cexel, the parameters m and n can be used to control the width of the parallel beams. As was the case for selecting a set of representative solid angles, there is a tradeoff between accuracy of the parallel beam approximation and storage requirements. This will be discussed in greated detail as it pertains to the propagation of light through empty space.

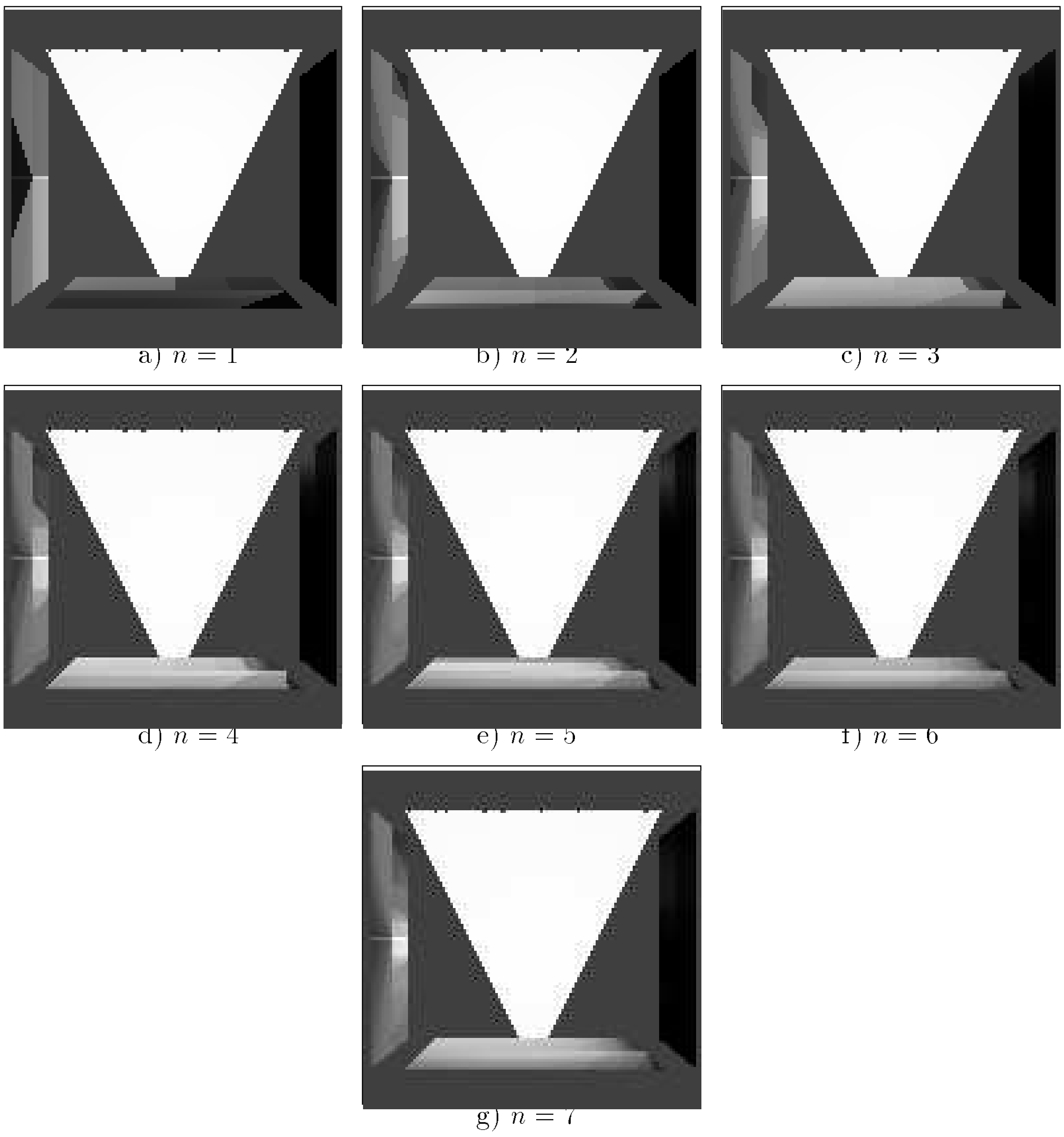


Figure 8.4: Simple Scenes, Varying Number of Solid Angles

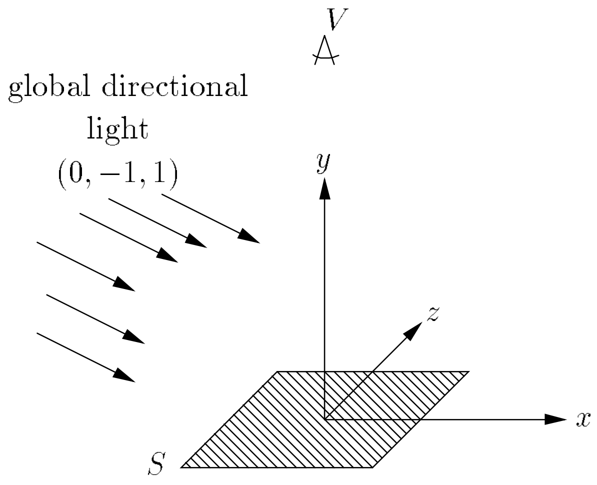


Figure 8.5: Simple Propagation Scene

8.2.2 Light Propagation

To understand how the size of the cexels affects propagation through empty space, we will illustrate this with a simple case. Consider a surface S consisting of a perfectly diffuse reflecting square polygon lying in the xz plane, and centred about the origin, as illustrated in figure 8.5 A global directional light source is placed such that light comes from the direction $(0, -1, 1)$. The viewpoint V is placed at an arbitrary point along the positive y axis, with the viewing direction towards the origin. As was mentioned in chapter 7, the power received at the viewpoint is calculated using an infinitesimal element of surface placed at the viewpoint whose normal coincides with the viewing direction.

It can be shown that the expected picture as seen from V is a square, with the intensity of light perceived decreasing as a function of the radial distance from the centre of the square. The first image computed is for the (unsubdivided) initial bounding cell, with one cexel covering exactly the side of the cell. Because of the discretization of angles into a set of representative directions corresponding to the solid angles, the direction $(0, -1, 1)$ is not available, thus the global light will appear to come from the direction $(1, -8, 7)$[2].

[2]This direction was chosen arbitrarily, and could have been any of the equidistant directions available, such as $(-1, -8, 7)$, $(1, -7, 8)$, or $(-1, -7, 8)$.

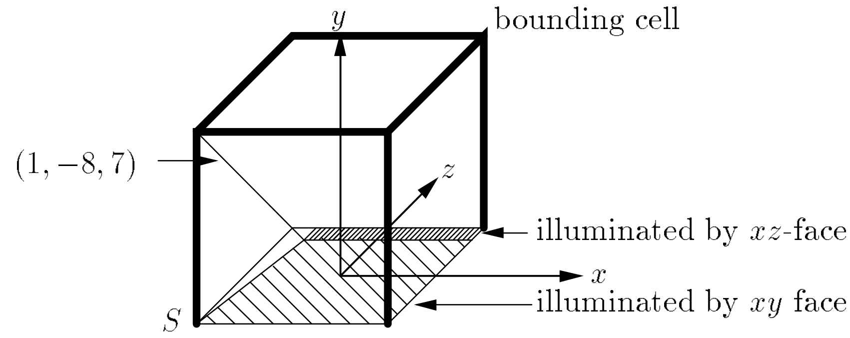


Figure 8.6: Bounding Cell

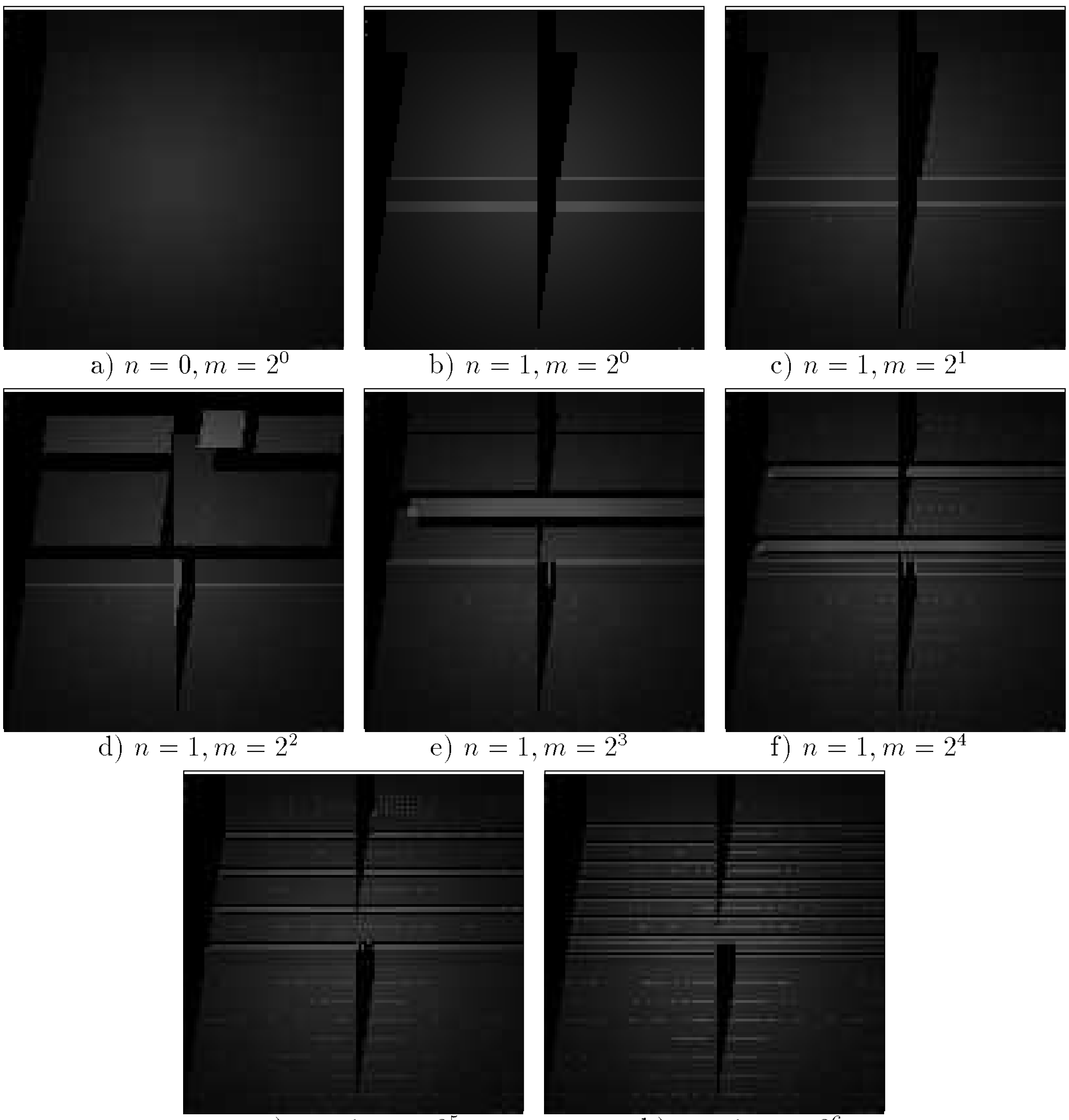
To explain the differences between this image and the expected one, consider the bounding cell that lies above the surface, as illustrated in figure 8.6. Using orientation-dependent parallel beams, the light which impinges on the surface comes from both the top face (in the top xz plane) and the front face (in the front xy plane). Since parallel beams are defined as the projection of the cexel from which the beam originates, the result is that both faces are projected along the direction $(1, -8, 7)$ onto the surface, as indicated in figure 8.6. Since the directional light was defined in the direction $(0, -1, 1)$, no light can impinge on a yz face; this explains the area of darkness on the left side of the surface.

To examine the effect of various subdivision levels and cexel size, this scene was rendered for values of the cexel defining parameters n and m, such that:

$$0 \leq n \leq 1$$
$$m = 2^i \text{ such that } 0 \leq i \leq 6. \tag{8.3}$$

These images are shown in figure 8.7.

We now explain how the image with subdivision 1, with 1 cexel covering the face of the smallest cell is computed, but omit the explanation for the others, since the calculations involved are similar. Consider how the surface that lies within the lower-left-front cell gets shaded. Using the full grid point units introduced in chapter 4, let the origin of the bounding cell be $(0, 0, 0)$. Note that the side length of the bounding cell is 32 full grid point units. If we again assume that the beams are orientation



a) $n = 0, m = 2^0$ b) $n = 1, m = 2^0$ c) $n = 1, m = 2^1$

d) $n = 1, m = 2^2$ e) $n = 1, m = 2^3$ f) $n = 1, m = 2^4$

g) $n = 1, m = 2^5$ h) $n = 1, m = 2^6$

Figure 8.7: Effects of Discretization on Propagation

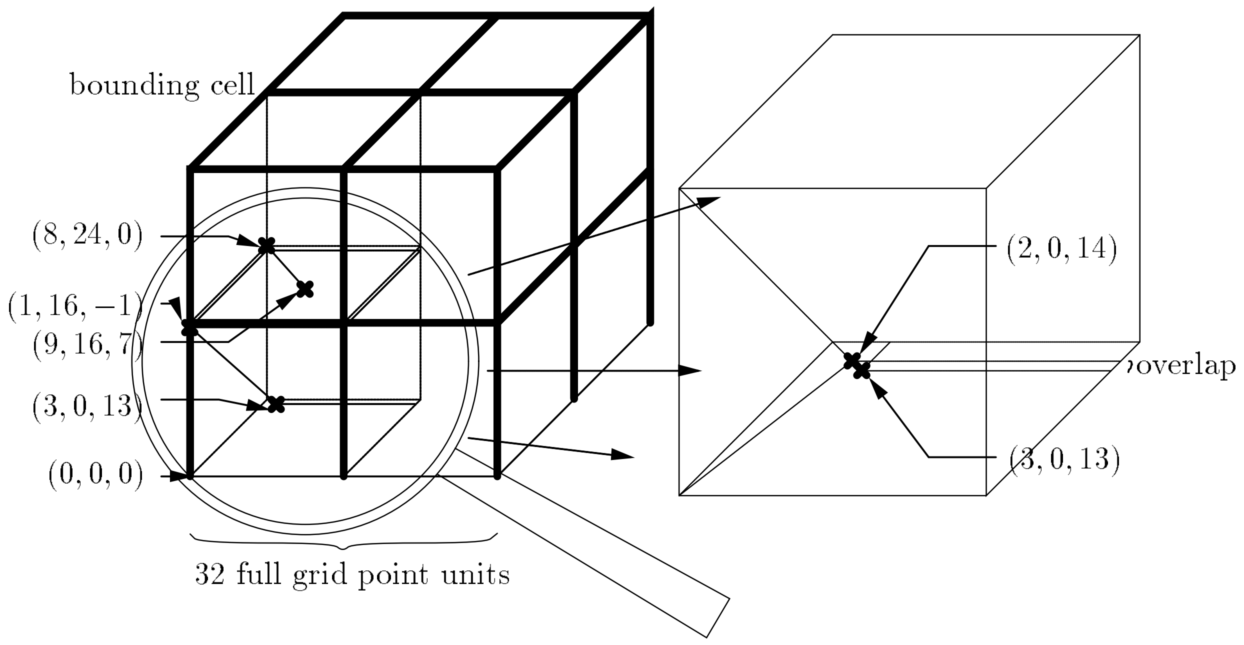


Figure 8.8: One Subdivision

dependent, the following accounts for all the light which impinges on the portion of the surface that lies within the lower-left-front cell.

The global directional source shines equally on all the xy plane faces, and on the xz ones. The only two beams that illuminate the quadrant of the surface nearest inside the lower-front-left cell are the two beams which originate from the two front-left yz faces. As illustrated in figure 8.8, the upper beam's representative direction is propagated through empty space from $(8, 24, 0)$, in full grid point coordinates, to $(9, 16, 7)$. Projecting a beam in the xz plane centred at $(9, 16, 7)$, the resulting projection intersects slightly with the surface, as illustrated in the inset. Also shown in the inset is the projection of the lower beam onto the surface. Notice that because of the orientation dependent definition of beams, there is an area in which the two projections overlap. In figure 8.7b), this shows up as a brighter band in the lower left quadrant.

As a result of the use of orientation-dependent beams, a uniform "sheet" of light can be propagated such that it overlaps. This happens when beams are propagated between cexels that have different orientation, since the new orientation redefines the cross-sectional area of the beam. By increasing the number of cexels, therefore decreasing the size of beams, the area of overlap is decreased, as can be observed in the sequence of images. If the storage capacity of the target system is sufficient, it is possible to reduce the area of overlap to less than a pixel by decreasing the cross-sectional area of the beams.

Due to limitations of the available hardware, the current implementation cannot allow the size of the beams to less than 2^{-7} of the area of the bounding cell. This suggests examining other methods of propagating beams in the hope that they would yield a more consistent representation of a flux of light. Using orientation-independent beams, and discretizing the beams that are to be propagated are but two of the possible alternatives. If orientation dependent beams are used, the cross-section of beams will never be dependent of the face it exits, hence beams that do not overlap, will never become overlapping, and beams that do overlap will always overlap. If the beams are discretized, the number of induced overlapping regions will increase, but the size of the overlap will decrease.

8.2.3 Local Illumination

Another major problem encountered in the implementation of FIAT relates to local illumination. There are two main aspects to this problem: how to compute the power emitted from a surface for highly specular surfaces, and how to deal with cells containing multiple objects,

8.2.3.1 Highly Specular Surfaces

One of the problems that was encountered as the current version of FIAT was being developed was that highly specular surfaces were poorly handled. This is mainly due to the sampling technique used to compute the power that leaves a surface through a solid angle due to an incoming beam. As was explained earlier on in this chapter, only one sample is used to approximate the power leaving a surface element dA through a solid angle Ω due to an impinging beam b. The approximation can be severely skewed in two different ways: it can either underestimate or overestimate the power.

With a highly specular surfaces, most of the power that impinges on a surface is reflected through a small solid angle centred about the ideal reflection direction. If a Blinn-like specular function is used, a highly specular surface is equivalent to raising the cosine of the angle between the normal and the pseudo-normal to a large exponent n. For n sufficiently large, 90% of the light which is specularly reflected will be within a very small solid angle centred about the ideal reflection direction. For a grid size of $GS = 2{\cdot}4$, recall that the angle of maximal error is 10.025 degrees. If n is chosen such that 90% of the re-emitted power is within a solid angle of radius of, say, 5 degrees of the ideal reflection direction, it is quite possible to not have a sample within the 90% zone. On the other hand, if the 90% zone is sampled, and if this sample is scaled by the size of the solid angle, the approximation will be too high.

One solution is to define a zone of 90% power, and to use many samples to calculate a more accurate approximation to the power flowing through solid angles intersecting this zone. For a Blinn-like specular function, the 90% zone corresponding to a surface exponent n can be computed numerically such that:

$$\frac{\int_0^z \cos^n \theta \, d\theta}{\int_0^{\frac{\pi}{2}} \cos^n \theta \, d\theta} = 90\%, \tag{8.4}$$

where z is the angle such that 90% of the specularly reflected light is reflected off micro-facets whose normal deviates from the surface normal by less than z. The distribution peaks for pseudo-normals that deviate from the surface normal by 0 degrees, which explains why the integral starts at 0. A similar computation can be done for other local illumination models, including Phong-based models. Given the value z, the local illumination calculations can be modified to be more diligent when computing values inside the 90% zone.

It can also be possible to compute an analytical solution to the problem of determining the outgoing power of a solid angle. One could approximate the integral of the specular function over the domain of this function (namely the hemisphere of directions), and evaluate this approximation over the desired solid angle. Notice also that the problem of determining the power leaving through a solid angle is independent of FIAT.

8.2.3.2 Multiple Objects in a Cell

The second problem encountered when computing local illumination arises in handling cells that need to be balanced, but that can no longer be subdivided, and still contain multiple objects. Given an impinging beam on a surface element and an outgoing solid angle, the current solution is to determine if an object lies between the surface element and the cexel that would be intersected in the representative direction of the solid angle. If there is one, the power is deemed to have been completely absorbed by this object, and the light leaving through the solid angle is discarded.

The main problem with this approach is that the interaction of the light with the intervening object is neglected. This could lead to critical global illumination not being accounted for, although it is quite possible that the light power is dissipated through inter-reflections among nearby objects, as previously mentionned in this chapter.

The other problem is computational in nature. If a cell contains many objects, significant computation time may be spent testing for light being blocked[3]. This would justify additional research into techniques for decreasing the time spent testing for intervening objects. One interesting possibility would be to calculate for a cell the probability that light leaving an arbitrary surface element would be blocked before reaching the boundary of the cell. A probabilistic decision could then be made, thus avoiding numerous object intersection calculations.

8.3 The Verdict

Is FIAT feasible? As a light flow modeller, it already is successful. Though the images presented suffer from discretization problems, the global illumination computed is acceptable. With some of the improvements that have been suggested, and possibly

[3]In certain instances, intersection calculation accounted for 90% of the execution time.

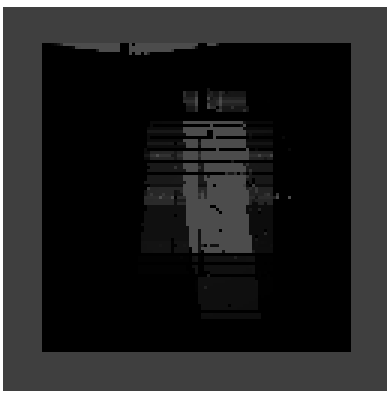


Figure 8.9: One Mirror Scene

with more sampling at the pixel level, it should be possible to obtain realistic images. For instance, the use of either orientation-independent beams or the discretization of the parallel beams when they are propagated should reduce the undesired artifacts that occur when beams that do not initially overlap do so after being re-emitted from a cexel with a different orientation. Better modelling of light sources, and of local illumination should also improve the quality of the computed global illumination. These improvements represent further refinements that will be carried out in the near future.

An image was computed that illustrates a scene consisting of three mirrors separated by 120 degrees. All three mirrors are perpendicular to a white, diffusely reflecting floor. A red directional light source is reflected off the first mirror onto the floor, a green directional light source is reflected off the second mirror onto the floor, and a blue directional light source is reflected off the third mirror onto the floor. The viewpoint is located above the floor, directly above the point where the three reflections meet. This is an example of how FIAT can handle secondary specular illumination whereas neither ray tracing nor radiosity can. A gray-scale version of a similar scene with only one of the three mirrors is shown in figure 8.9.

From a computational point of view, FIAT is an order $O(n)$ algorithm, since the running time is proportional to the number of cells, which is roughly proportional to the number of objects in the scene. Of course, the constants necessary to make FIAT an $O(n)$ algorithm are very large, since the running time per cell is proportional to the number of cexels and number of solid angles sampled. As for FIAT's storage requirements, we have already established that they are very large: anywhere from a few megabytes to well over 200. These storage requirements are dependent of both the number of cexels used — which is dependent of the number of subdivisions, and of the number of cexels that cover the face of the smallest cell — and of the number of representative directions used. It is suspected that the number of cexels used at any one point in an interaction is proportional to the number of cexels necessary to cover all faces of the bounding volume, since the power entering a cell is discarded once it has been propagated to the neighbours. At the present time, insufficient statistics are available to verify this hypothesis.

Chapter 9

Conclusion

In this thesis, we first presented a survey of local and global illumination models. Ray tracing, radiosity as well as hybrids and techniques that try to combine both ray tracing and radiosity were presented. It was concluded that neither ray tracing nor radiosity can completely solve the entire global illumination problem, but that both could be extended to a full solution, albeit to a prohibitive computational cost. The main topic of this thesis, FIAT, a light-driven global illumination algorithm was then introduced.

FIAT is a divide and conquer approach to solving the global illumination problem. A scene that is to be rendered is adaptively divided into cells, and a light energy budget is maintained for each cell. Since it is assumed that all light operators are linear, the global illumination problem can be solved by iteratively computing global illumination within each cell. This involves both using local illumination models to model the flow of light within a cell, and propagating light information between adjacent cells, through a common boundary.

An overview of the main components of FIAT was then presented. For simplicity, octrees were used to adaptively subdivide the volume enclosing the scene to be rendered. Next, the concept of the power balance of a cell was introduced which is a budget of the incoming, outgoing, absorbed and internally emitted power of a cell. This power balance was then used to determine the cell to be processed in the next iteration. Having chosen a cell to process, a decision had to be made: either subdivide this cell if the global illumination problem within it could not be easily solved, or solve the problem within it and update the power balance of its immediate neighbours. It was then shown that for must purposes, FIAT would halt when all but some threshold of the power initially contained in the scene had dissipated. Halting can be guaranteed upon the imposition of some basic constraint, namely that a minimal fraction of the light processed at each step must either be absorbed or discarded. Finally, it was shown how an image could be computed using the global illumination information.

We then proceeded to an indepth analysis of the major phases of FIAT. In chapter 4, we saw how the flow of light through the face of a cell can be discretized by subdividing each face into smaller sections called cexels. For simplicity, a fixed-size cexel was adopted. A discussion of various schemes for discretizing the flow of light at the cexel level followed: rays, frustums, and beams of light. It was concluded that parallel beams would be more suitable for a first implementation, as they are simpler to implement, and can be propagated in a straight line.

Chapter 5 introduced methods for converting the flux coming from a light source to a parallel beam representation at the cexel level. This was done for point-light sources, global directional light sources, polygonal directional, and polygonal diffuse light sources. Several different methods for calculating the set of solid angles through which a light source illuminates a cexel were presented. First-order and arbitrarily precise approximations were also developed for calculating the power leaving a light source and reaching a cexel within a given solid angle.

Chapter 6 discussed the problem of modelling the interaction of light entering a cell with the objects inside, in order to compute the illumination within this cell and to appropriately propagate light to its neighbours. We decided to approximate the interaction of a beam of light with the interaction of the representative ray with the first object intersected. Given a point of intersection, we saw that the problem of converting the resulting outgoing flux due to the incoming ray could be subdivided into two phases: determining which cexels are intersected by a particular solid angle from the point of intersection, and determining the contribution received by each cexel. Various approximations were presented for these problems.

In chapter 7, a technique for computing an image using the FIAT algorithm was introduced. Using view dependence and the linearity of light assumption, a light update buffer was defined. At each iteration step, the power leaving the visible elements of surface within the cell towards the eye had to be computed. This was done using a series of approximations, based on one sample per pixel, and calculated by placing an artificial element of surface at the viewpoint. This was again presented as a two-phase problem: first, we determine which beams illuminate an element of surface area, and second, for each beam we calculate the power impinging on the artificial element of surface placed at the viewpoint.

Finally, chapter 8 discussed some of the implementation modalities, highlighting the differences between some of the theoretical results of the previous chapters with the current implementation of FIAT. In particular, more work remains to be done in the area of handling light sources, light propagation and local illumination. The difficulties encountered during our current implementation of FIAT were centred around discretization problems. The selection of the number of solid angles used to sample the full sphere of directions was seen to be driven by the need for directional accuracy, traded off against the need to reduce storage requirements. Similarly, problems in light propagation were a result of having to use beams that were as small as possible, yet required only a small amount of storage.

Some suggestions were also made on how to improve the quality of images produced by FIAT. Even though the images suffered from discretization problems, they conveyed global illumination information that cannot be achieved using classical solutions to the global illumination problem such as standard ray tracing and radiosity. This suggests that further research into the discretization problems existing in the current version, as well as a full implementation of the techniques suggested in chapters 4 through 7, would be fruitful. Through better modelling of local illumination, light source flux conversion, and possibly the discretization of beams being propagated, it is hoped that convincing images will be produced in the near future, for arbitrary scenes.

The problems encountered so far are related mostly to the storage requirements of the algorithm. Because we model the flow of light through an environment, the number of samples required to produce images of great visual appeal quickly exceeds the runtime storage available on any of the local machines. Nevertheless, the great advantage of this algorithm is that it models the flow of light through an *arbitrary* environment, and allows us to compute better global illumination than any of the existing techniques. The quality of the global illumination in computed images is thus improved, and this is particularly noticeable in scenes where secondary illumination is dominant. Finally, we would like to stress that this algorithm is independent of the type of surface used, and thus can compute global illumination for a full range of surfaces varying from purely specular to purely diffuse.

This algorithm also opens the door to a number of related research topics. Currently, research is being done in the area of participating media, such as fog, to try to take advantage of the way in which FIAT models the propagation of light. As well, research is being conducted into parallelizing this algorithm, and early results appear promising. With a distributed version, greater computational power, as well as additional run-time storage should alleviate some of the problems encountered in the initial version, by allowing an increase in the resolution of beams and directions. The product of all this research should be a general renderer that can convert arbitrary scenes to images of great visual realism.

Bibliography

[Aman84] Amanatides, J., “Ray Tracing with Cones”, *Computer Graphics*, Vol. 18, No. 3, July 1984.

[Aman87] Amanatides, J., Woo, A., “A Fast Voxel Traversal Algorithm for Ray Tracing”, *Proceedings of Eurographics'87*, North Holland [Elsevier Science] Publishing Co., Amsterdam, 1987.

[Aman88] Amanatides, J., “A Solid Angle Approach to Ray Tracing”, Ph. D. Thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, 1988.

[Baha87] Bahar, E. and Chakrabarti, S., “Full-Wave Theory Applied to Computer-Aided Graphics for 3D Objects”, *IEEE Computer Graphics and Applications*, Vol. 7, No. 7, July 1987.

[Blin77] Blinn, J.F., “Models of Light Reflection for Computer Synthesized Pictures”, *Computer Graphics*, Vol. 11, No. 2, July 1977.

[Cohe85] Cohen, M.F. and Greenberg, D.P., “The Hemi-Cube, A Radiosity Solution for Complex Environments”, *Computer Graphics*, Vol. 19, No. 3, July 1985.

[Cohe86] Cohen, M.F., Greenberg, D.P., Immel, D.S. and Brock, P.J., “An Efficient Radiosity Approach for Realistic Image Synthesis”, *IEEE Computer Graphics and Applications*, Vol. 6, No. 3, 1986.

[Cohe88] Cohen, M.F., Chen, S.E., Wallace, J.R. and Greenberg, D.P., “A Progressive Refinement Approach to Fast Radiosity Image Generation”, *Computer Graphics*, Vol. 22, No. 3, August 1988.

[Cook81] Cook, R.L. and Torrance, K.E., “A Reflectance Model for Computer Graphics”, *Computer Graphics*, Vol. 15, No. 3, August 1981.

[Cook84] Cook, R.L., Porter, T. and Carpenter, L., “Distributed Ray Tracing”, *Computer Graphics*, Vol. 18, No. 3, July 1984.

[Cook85] Cook, R.L., “Stochastic Sampling in Computer Graphics”, *Transactions on Computer Graphics*, Vol. 5, No. 1, 1986.

[Devi87] Devillers, O., Puech C. and Sillion, F., "CIL: un Modèle d'Illumination Intégrant les Réflexions Diffuse et Spéculaire", Research Report of the LIENS, October 1987.

[Dipp85] Dippé, M., and Wold, E.H., "Antialiasing through Stochastic Sampling", *Computer Graphics*, Vol. 19, No. 3, July 1985.

[Fium87] Fiume, E., and Fournier, A., "The Complexity of Abstract Ray Tracing", submitted for publication, 1987.

[Fium89] Fiume, E., "The Mathematical Structure of Raster Graphics", Academic Press, Inc., Toronto, Ontatio, 1989

[Fuch80] Fuchs, H., Kedem, Z., and Naylor, B., "On Visible Surface Generation by a Priori Tree Structures", *Computer Graphics*, Vol. 14, No. 3, August 1980.

[Fuji86] Fujimoto, A., Tanaka, T., and Iwata, K., "ARTS: Accelerated Ray-Tracing System", *IEEE Computer Graphics and Applications*, Vol. 6, No. 4, April 1986.

[Glas84] Glassner, A., "Space Subdivision for Fast Ray Tracing", *IEEE Computer Graphics and Applications*, Vol. 4, No. 10, October 1984.

[Gora84] Goral, C.M., Torrance, K.E., Greenberg, D.P. and Battaile, B., "Modeling the Interaction of Light Between Diffuse Surfaces", *Computer Graphics*, Vol. 18, No. 3, July 1984.

[Gour71] Gouraud, H., "Continuous Shading of Curved Surfaces", *IEEE Transactions on Computers*, June 1971.

[Hall83] Hall, R.A., Greenberg, D.P., "A Testbed for Realistic Image Synthesis", *IEEE Computer Graphics and Applications*, Vol. 3, No. 8, November 1983.

[Heck84] Heckbert, P.S. and Hanrahan, P., "Beam Tracing Polygonal Objects", *Computer Graphics*, Vol. 18, No. 3, July 1984.

[Imme86] Immel, D.S., Cohen, M.F., Greenberg, D.P., "A Radiosity Method for Non-Diffuse Environments", *Computer Graphics*, Vol. 20, No. 4, August 1986.

[Kaji85] Kajiya, J.T., "Anisotropic Reflection Models", *Computer Graphics*, Vol. 19, No.3, July 1985.

[Kaji86] Kajiya, J.T., "The Rendering Equation", *Computer Graphics*, Vol. 20, No. 4, August 1986.

[Lee85] Lee, M., Redner, R.A., and Uselton, S.P., "Statistically Optimized Sampling for Distributed Rayt Tracing", *Computer Graphics*, Vol. 19, No. 3, July 1985.

[Mora81] Moravec, H.P., "3D Graphics and the Wave Theory", *Computer Graphics*, Vol. 15, No. 3, August 1981.

[Musg87] Musgrave, F.K., Hitchner, L.E., and Murphy, J.L., "A Realistic Model of Refraction for Computer Graphics", submitted for publication, 1987.

[Pavl88] Pavlidis, T., "Algorithms for Graphics and Image Processing", Computer Science Press, Rockville, Maryland, 1988

[Phon75] Phong, B., "Illumination for Computer Generated Pictures", *Communications of the ACM*, Vol. 18, No. 6, 1975.

[Poul89] Poulin, P., "Anisotropic Reflection Models", Master's thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, 1989.

[Same84] Samet, H., and Tammine, M., "Efficient Octree Conversion by Connectivity Labelling", *Computer Graphics*, Vol. 18, No. 3. 1984.

[Shao88] Shao, M., Peng, Q. and Liang, Y., "A New Radiosity Approach by Procedural Refinements for Realistic Image Synthesis", *Computer Graphics*, Vol. 22, No. 3, August 1988.

[Shin87] Shinya, Mikio, Takahashi, Tokiichiro and Naito, Seiichiro, "Principles and Applications of Pencil Tracing", *Computer Graphics*, Vol. 21, No. 4, July 1987.

[Snyd87] Snyder, J., and Barr, A., "Ray Tracing Complex Models Containing Surface Tessellations", *Computer Graphics*, Vol. 21, No. 4, July 1987.

[Thib87] Thibault, W.C., and Naylor, B.F., "Set Operations on Polyhedra Using Binary Space Partitioning Trees", *Computer Graphics*, Vol. 21, No. 4, July 1987.

[Thom86] Thomas, S.W., "Dispersive Refraction in Ray Tracing", *The Visual Computer*, Vol. 2, No. 1, January 1986.

[Torr67] Torrance, K.E. and Sparrow, E.M., "Theory for Off-Specular Reflection from Roughened Surfaces", *J.Opt.Soc.Am*, Vol. 57, No. 9, 1967.

[Wall87] Wallace, J.R., Cohen M.F. and Greenberg, D.P., "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ra"y Tracing and Radiosity Techniques, *Computer Graphics*, Vol. 21, No. 4, July 1987.

[Wall89] Wallace, J.R., Elmquist, K.A., and Haines, E.A., "A Ray Tracing Algorithm for Progressive Radiosity", *Computer Graphics*, Vol. 23, No. 3, 1989.

[Ward88] Ward, G.J., Rubinstein, F.M., and Clear, R.D., "A Ray Tracing Solution for Diffuse Interreflection", *Computer Graphics*, Vol. 22, No. 4, August 1988.

[Warn69] Warnock, J.E., "A hidden-line algorithm for halftone picture representation", TR 4-15, Department of Computer Science, University of Utah, Salt Lake City, Utah, 1969.

[Watk70] Watkins, G., "A Real Time Hidden Surface Algorithm", Ph. D. dissertation, Department of Computer Science , University of Utah, Salt Lake City, Utah, 1970.

[Whit80] Whitted, T., "An Improved Illumination Model for Shaded Display", *Communications of the ACM*, Vol. 23, No. 6, June 1980.

[Yoko88] Yokoi, S., and Toriwaki, J., *Computer Science and Technologies, Ch. 3.7 Realistic Expression of Solids with Feeling of Materials*, JARECT, Vol. 18, Ohmsha Ltd., 1988.