FIAT:
Light-Driven Global Illumination

Alain Fournier [1]
Eugene Fiume
Marc Ouellette
Chuan Chee

Department of Computer Science
Sandford Fleming Building
University of Toronto
Toronto, Ontario
M5S 1A4
{alain | elf | vv1 | ckchee}@dgp.toronto.edu

Abstract

Computing the global illumination in a scene, which is the amount of light falling from every source on every surface, is an important, elusive and costly goal in computer graphics. Recent solutions include ray-tracing and its variants, radiosity-based algorithms and combinations of both.

We suggest here a new paradigm. Its main characteristics are: it is a light-driven approach, it balances the light energy budget of adaptively determined cells, and it uses the result as "incremental local light sources" in a conventional rendering.

We describe an implementation based on adaptive decomposition of the space in an octree of cells, light propagation within cells modelled by sampled beams with discretized directions, and a light update buffer for progressive rendering.

The advantages of this approach are the greater range of effects, over both radiosity-based techniques and ray-tracing, that can be included (indirect illumination, specular reflection, contributing media), the adaptive and progressive nature of the process (in particular unlit areas are not considered) and the flexibility achieved (parameters can be tuned to control the time spent on the global component).

CR Categories: I.3.3 [**Computer Graphics**]: Picture/Image Generation — *display algorithms*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms: algorithms.

Additional Keywords and Phrases: Octree, radiosity, power.

1. Introduction

It has always been known that behind the *ambient* term, which is set to a constant by most illumination models, lurks a problem of devastating complexity. If we want to achieve a convincing level of realism, however, it is indispensable to get at least a reasonable approximation of the amount and distribution of light falling on any object from every direction. This is so because our judgement about the nature, the quality and the orientation of the surfaces we see, and about the characteristics of the atmosphere around us, in short our whole sense of space, critically depends on it.

---

1. Current address: Department of Computer Science, University of British Columbia, Vancouver, BC, V6T 1W5.

The reasons for the difficulties are fairly evident. A local illumination model defines the intensity distribution of the outgoing light given the intensity distribution of the incoming light, and does so with only that local information. On the other hand a global illumination model has to define the intensity distribution of the incoming light, which is in turn a function of the intensity distribution of the outgoing light.

2. Previous Techniques

The first illumination models used light falling directly on the objects from the light (often ignoring shadows) to compute the light reflected in the direction of the viewer (see [Hall83a] for a survey of illumination models). Since it is obvious that in most environments a significant amount of light actually comes indirectly from other surfaces, a constant term, the *ambient intensity* is usually added.

2.1. Ray-tracing

Shown first in its full glory by Whitted [Whit80a], ray-tracing traces paths from the eye into the environment through refracted and reflected rays. Ray-tracing models very well pure reflection and refraction, and since in many circumstances this is what we notice most in scenes, it is a successful technique. Another reason for its wide appeal is that it is relatively simple to implement. Ray-tracing, however, has two serious drawbacks (in light of the running times exhibited in this paper we will not count excessive computing time as a drawback). First it is a *ray-driven* process, and as such it inherently point-samples. Various modifications have been proposed to alleviate this problem, some of them very effective, such as cone-tracing [a], distributed ray-tracing [Cook84a], stochastic sampling [Dipp85a] [Cook86a], stratified sampling [Lee85a] and pencil-tracing [Shin87a]. Second, ray-tracing traces from the eye, and as such cannot determine illumination coming indirectly from the light sources after reflection/refraction off other surfaces (or in general media). That means still being limited to a constant ambient light term. Various fixes have been suggested [Ward88a], most involving the elusive "ray-tracing from the light", none too successful.

2.2. Radiosity

Radiosity is a physical quantity expressing the *power* (quantity of energy per unit time, see Glossary) per area sent by a surface into the environment. The radiosity of a surface is a function of the radiosities of other surfaces in the environment, and therefore to obtain it one has to solve a system of n linear equations, n being the number of surface elements in the scene. From the introduction of an algorithm based on radiosity by Goral, Torrance and Greenberg [Gora84a] [Nish85a], to its first practical application through the use of *hemi-cubes* [Cohe85a], to the introduction of volume elements to include participating media [b], to an adaptive implementation that proves it can be a realistic choice for a rendering method [Cohe88a], radiosity-based methods have produced remarkable pictures, and have shown that the global illumination problem for diffuse surfaces is essentially solved.

The obvious limitation of radiosity is that it is a model of diffuse reflections. Several attempts have been made to introduce specular reflection within a radiosity approach, notably by Immel, Cohen and Greenberg [Imme86a], but this proves to be extremely costly, because of the necessity to use bidirectional reflectance which forces a jump in the size of the solution matrix from $O(n^2)$ to $O(n^4)$, where n is the number of elements in the scene. A very interesting and (after the fact) simple observation made by Neumann and Neumann [Neum89a] is that the same system of equations obtained for diffuse surfaces is obtained for a more general class of *separable* illumination models, that is where the function of the light direction is separable from the function of the eye direction. They thus obtain a limited range of specular models "for free" within the radiosity solution.

Essentially, radiosity methods are *patch driven* (actually there can be surface patches or volume elements), and anything that forces an increase in the number n of surface elements will dramatically increase the storage requirements and the running time (roughly as $\Theta(n^2)$).

2.3. Hybrid Solutions

The fact that ray-tracing and radiosity model different aspects of the global illumination problem has lead several researchers to attempt combining the two techniques. Wallace, Cohen and Greenberg [Wall87a] and [Devi87a] [Sill89a] have met with some success in that direction, but the gain is limited by the fact that they consider only a limited number of diffuse/specular surface interactions.

In a quite different vein, Buckalew and Fussell [Buck89a] have discretized light beam directions to replace the hemicube by discrete patch to patch links. We will use direction discretization, albeit for a different purpose. Other approaches to improve on the hemicube solutions are found in [Baum89a] and [Wall89a]. They are relevant to this work because they address discretization problems similar to the ones found in our approach.

2.4. The Rendering Equation

In 1986, Kajiya introduced what he called the *rendering equation* [Kaji86a]. As an equation, it states in an enlightening form the various elements of the energy (actually power) balance for each pair of points in the environment, and couches it in a form (the Neumann expansion) which makes it easy to relate to the various solutions described above. The paper also introduced new sampling techniques to better approximate the integrals involved in the solution of the rendering equation. The techniques suggested and implemented are variations of ray-tracing, but the main difference for our discussion is that they are *contribution-driven*; that is new paths are selected on the basis of their expected contribution to the result. While it can be very effective in many scenes, as exhibited by pictures showing together convincing effects of *colour bleeding* and concentration of light by glass spheres, the basic problem of this type of approach is that the real contribution of a ray is not known until we are finished tracing it, and it is therefore difficult to choose rays to trace on this basis. We present a more detailed analysis of this problem in [Fium87a].

2.5. Analysis

An obvious conclusion from this brief survey is that each class of methods has good and bad qualities. Most of the drawbacks of ray-tracing stem from its *ray-driven* character, that is it works only for perfect mirrors and refractors, and from the fact it traces from the eye. Most of the drawbacks of radiosity come from the fact that it is *patch-driven*, and introduces the light only after the form factors have been computed, and from the fact it really works only for separable sources and reflectors. They are both global illumination solutions which critically depend on a limiting assumption about the local behaviour of light. Hybrid solutions also invariably have problems because there is no neat dichotomy between specular and diffuse reflection, and real reflection/refraction is a continuum between these two extremes.

The alternative is a *light-driven* approach, where the global illumination problem is solved by propagating the light from sources (we use "source" in the broadest sense of anything radiating light) to other parts of the environment to the eye [2]. One way to achieve this goal is to use wave propagation, and it has been suggested before [Mora81a] [Baha87a] but we think that it is excessive and impractical at this time. Another important criterion is that the amount of effort spent on each part of the scene is somehow related to how much light will come from it. There is no need to compute much about the whole content of a closet whose door remains closed, even less need to compute anything if all the lights are off. What we are advocating, therefore, is a *light-driven, volume-oriented* approach.

3. The Paradigm

Consider a volume **V** within the environment to render, with a boundary surface **S**. If we know for every point of **S** the flux of light crossing it in any direction, then we can study separately the illumination inside and outside of **V**. Furthermore, even if we do not know the true situation at the boundary, but only some amount of light emitted inside **V** and the amount of light coming into **V**, and if we know how to solve the global illumination problem inside **V**, then we can assign to points of **S** the correct amounts of outgoing light. The outside of **V** can then be treated without considering **V** unless it is found that more light comes into **V** from outside. In this case we can solve the global illumination problem inside **V** again *independently* of the previous solution if we assume (and it is the only serious restriction) the *linearity* of the light effects. This means that we assume that all the operators $\mathbf{O}(I)$ on light intensity (or any quantities proportional to the light power) are linear:

2. We hasten to point out that it will not be "ray-tracing from the light", since we won't use "rays", and we won't follow just one reflection/refraction.

$$\mathbf{O}(\ a\ I_1 + b\ I_2) = a\ \mathbf{O}(I_1) + b\ \mathbf{O}(I_2)$$

If we cannot solve the global illumination problem for **V**, we can partition it into two or more sub-volumes, and so on recursively until each section is simple enough so that we can deal with it.

3.1. Power Balance

Given a volume **V**, how do we know whether its *internal global illumination* has correctly been determined? We can by considering its *power balance*. We use power instead of energy because we are only interested for the time being in the steady state of the system, and therefore we should use the energy exchanged by unit of time, in other words the power. If we call P_I the power we have so far determined to be going into **V** from outside, P_O the power we have so far determined to be going from **V** to the outside, P_A the power we have so far determined to be absorbed inside **V** (of course energy is conserved and P_A is transformed into non-visible forms of radiation, or non-radiative forms of energy) and P_E the power we have so far determined to be emitted within **V**, then the quantity:

$$\Delta P = P_E + P_I - P_A - P_O$$

is a measure of our ignorance about the power budget of this volume (see Figure 1).

Figure 1. Power Budget for volume **V**

A positive ΔP means that some power is not correctly re-emitted into the environment. A negative ΔP means that some power is lost by **V** into the environment and not properly accounted for. Only when ΔP is zero can we claim to have balanced power budget of **V**. It is crucial to note that once we have done this, because of our linearity assumption we do not have to remember any of the power components. After balancing **V**, we have to consider it again only if there is a modification to one of the power values, and then we only need to use its increment.

In the initial state no power is exchanged between volumes, and the only unbalanced volumes are the ones containing a light source ($P_E = \Delta P > 0$). After balancing any of these volumes, new volumes will become unbalanced because their P_I will become positive ($P_I = \Delta P > 0$). If when dealing with a volume we are careful not to "create" energy, that is we have a local illumination and transmission model with some physical credibility, then a "treated" volume will be balanced, and remain so until new light is transmitted into it from an adjacent volume. This amount of light cannot be more than its neighbour received (in power), and in fact will be less after any reflection/refraction, so we are guaranteed to eventually reach a state where all volumes are balanced. In particular we will not become trapped in local minima, and general global optimization algorithms are not necessary. To prove this, consider that the only way a cell can become unbalanced is with $\Delta P > 0$, as shown above. Therefore the total of all the ΔP is the sum of positive values. Each time a cell is balanced, its ΔP goes to 0, and some of its neighbours have their unbalances increase, but the total increase is P_O of the cell just balanced, since it is the power sent by the cell to its neighbours. The previous unbalance of the cell was $P_E + P_I$ (note that P_E appears only the first time the cell is dealt with), and since

$$P_O \leq P_E + P_I,$$

the difference being P_A, then the total unbalance can only decrease or remain constant. If we do not allow perfect reflectors/transmittors, then the total unbalance can only decrease, and we are guaranteed to converge to a global balance [3].

An issue of concern is the existence of cycles. They are very possible, as for instance when a light source is between two parallel mirrors, each one of these three objects being in distinct volumes (see Figure 2).

Figure 2. A cycle

In this case the power emitted by L into volume $\mathbf{V}_1$ will re-enter volume $\mathbf{V}_2$, then enter volume $\mathbf{V}_3$, come back to $\mathbf{V}_2$, etc.. If there is no loss of power at each reflection, the cycle can go on forever. In a similar way longer cycles can easily be created. One solution would be to try to detect such cycles ahead of time. But for many reasons it is contrary to the spirit of the algorithm, since we do not want to consider parts of the environment not reached by light. Another solution would be to try to detect cycles dynamically. There again it is not practical in our context, since we want to be able to forget what we have done as soon as a volume is balanced. We will in practice *preempt* cycles by limiting the amount of power that can be reflected to something below 100%, so that the amount of power transmitted in a cycle will be damped rapidly, and we will not consider it further [4].

3.2. Basic Algorithm

The straightforward implementation of this paradigm can then be described by the following pseudo-code:

```
begin Main
    Scene_Pointer Scene
    Volume_Pointer Volume
```

3. In practice we have to be watchful of numerical problems due to approximations in the shading model, truncations and round-offs.

4. Note again that a decrease in power does not necessarily mean a decrease in intensity for any particular beam.

```
        List of Volume_Pointer Volume_List
        Get_Scene_Description(Scene)
        Choose_Initial_Volume(Scene, Volume)
        Add_To_Volume_List(Volume, Volume_List)
        Balance_Everything(Volume_List)
        Display(Scene)
        Enjoy(Scene)
end Main

begin Balance_Everything(Volume_List)
        Volume_Pointer Volume, Sub_Volume
        List of Volume_Pointer Sub_Volume_List
        while (Volume_List not Empty)
                Chose_Next_Unbalanced(Volume_List, Volume)
                if (Compute_DELTA_P(Volume) < P_TOLERANCE) then
                        Remove_From_List(Volume_List, Volume)
                elsif Can_Deal_With_Volume(Volume) then
                        Balance_Volume(Volume)
                        Update_Neighbours(Volume, Volume_List)
                        /* Update_Neighbours can cause some of the neighbours
                           to be put on Volume_List */
                        Update_Local_Illumination(Volume)
                        Remove_From_List(Volume, Volume_List)
                else
                        Split_Into_Sub_Volumes(Volume, Sub_Volume_List)
                        for each Sub_Volume in Sub_Volume_List
                                Add_To_Volume_List(Sub_Volume)
                        end for
                end if
        end while
end Balance_Everything
```

4. Modalities

For the sake of brevity we will discuss at the same time the various modalities of our paradigm and some specifics of our implementation. We hope to make clear which is which.

4.1. Scene Description

The paradigm proposed is totally independent of the models chosen for objects, lights and illumination models, as long as there are local algorithms to deal with them. In our implementation we used a rather conventional front end, similar to the one used in our standard ray-tracer, *optik* [Aman87a]. The range of objects has been temporarily limited to collections of convex polygons (in the images shown, we only used triangles). Light sources, in addition to traditional point and directional light sources, can be polygonal extended light sources, in practice objects with an emissivity attached to their surface. In this case the source is considered to be a diffuse emitter, that is its *radiant exitance* (we will call it *intensity*, to follow computer graphics practice) is the same in all directions of the hemisphere above the plane of the polygon.

4.2. Space Subdivision

There are many practical schemes for hierarchical space division, and choices have to be made. The first important decision is about the volumes: their shape, size, and how to determine them. The above description of the paradigm clearly leads to an adaptive solution. There is no need to examine a given part of the environment unless some light actually reaches it (or it contains a light source). A powerful and well-known way to subdivide space adaptively is by using *octrees* [Same84a]. The advantages of octrees are clear in our case: they contain simple, regular, object-independent cubic cells, which will facilitate the computations for the propagation of light. Octrees have been used many times before in computer graphics, notably to speed-up ray-tracing [Glas84a]. Among the alternatives are *point-octrees*, *k-d trees* [Same84a] and *binary space partitioning (BSP) trees* [Thib87a], but all of these structures are data-driven (in our case objects are the data), and we found it easier to deal with a space-driven data structure.

In our application, the criteria to decide whether to split a octree *cell* (as we will call the leaf cubes of the octree) is whether it has enough unbalance to warrant looking into it, and in this case whether it is simple enough so we can "deal" with it. There is therefore a strong analogy between this approach and Warnock's visibility algorithm [Newm73a], where the screen is subdivided as a quadtree until squares are simple enough to be dealt with, or too small to be further subdivided.

What is simple enough in our case? In our implementation, there are four kinds of cells which are not further divided:

- an empty cell. Light is merely propagated through the cell to its neighbours.
- a cell with only a light source in it. In the case of compound extended light sources, that means only one polygonal element of the source. In this case the light emitted by the source is propagated through the boundaries of the cell to its neighbours.
- a cell with only one polygon in it. In this case the local illumination model is used to compute, given the incoming light, the reflected, transmitted and absorbed (normally the absorbed light is computed by difference from the other ones; that is we assume that our local illumination model is physically consistent).
- a cell too small to be further subdivided. We will see later what the size criterion ought to be. In this case we compute the outgoing light from the incoming and the emitted light based on a local illumination model. Depending on what this model is, it might mean neglecting the reflections within the cell. The cell budget will not be wrong, but the distribution of light might be.

In the process of subdividing the cells, objects are not clipped. Each cell maintains pointers to the list of objects within the cells. If boxing is used, these objects might not actually intersect the cell, but this will be found when the cell is dealt with. Extended light sources are actually clipped, since we need to compute the power emitted in each cell, which is a function of the area of the light source within the cell.

4.3. Light at the Boundary and Light Transmission

The crucial implementation decision is how to represent light as it enters and leaves the cell. The goal is to represent as faithfully as possible the power and distribution through space, frequencies and phases of the light traversing the boundaries. Phases will not be represented, since we will not attempt to deal with the full wave aspect of light (besides most light sources are incoherent). The issue of frequencies is not different in our context than in other rendering system, and we will ignore it for the time being. Our current implementation uses an *RGB* model of light [5]. Power is the easiest quantity to deal with, since it is only a scalar value for any surface element of the boundary. The spatial distribution is the most difficult problem. A compromise has to be found between good space resolution *versus* reasonable storage cost and faithfulness to direction *versus* control of spread. The possibilities, considering only known techniques, include rays, cones, beams, parallel beams or pencils [Whit80a] [c] [Heck84a] [Shin87a].

Our current implementation uses sampled beams. Each face of a cell is divided into an $k \times k$ array of cell sections, or *sexells*. Each sexell is a square, which for convenience we have constrained to be of constant size in the world coordinate system. This means k is a function of the size of the cell. This also means that in this case we do not subdivide further any cell whose faces contain only one sexell ($k = 1$). For each sexell, the hemisphere inside the sexell is divided into a number of discrete directions. These directions have been chosen so that the tangents of the corresponding angles are ratios of simple integers, to facilitate the propagation and the filtering computations (see Figure 3). There are usually (this can be changed by a *#define* statement) 192 directions per hemisphere. Only one hemisphere is needed, since the neighbouring cell carries the information for the other. In our implementation each cell has the hemisphere of the directions pointing inside.

5. *Video Meliora Sed Atque Deteriora Sequor.*

Figure 3. The 8 cubes defining the directions around a sexell

The contribution of each beam is represented by a single power value, and as such the stored values are samples of a continuous distribution of the light energy through space. This means that we can expect sampling problems, and that some form of filtering is necessary to obviate them. Filtering is used in the current implementation of FIAT at two levels: in the propagation and in the shading.

The continuous distribution[6] of power through cell wall is some function of the position and direction. The corresponding intensity (power per area per solid angle) can be written:

$$I(x, y, \theta, \phi)$$

WLOG assume we deal with a face in the XY plane, and θ and ϕ are the usual polar angles. The total power through the face is:

$$P_{face} = \int_{\text{all } \theta} \int_{\text{all } \phi} \int_{\text{all } x} \int_{\text{all } y} I(x, y, \theta, \phi)\, d\theta\, d\phi\, dx\, dy$$

The power carried by a beam is (the indices ij denote the direction θ_i and ϕ_j, and the indices kl indicate the position $x_k y_l$):

$$P_{ijkl} = \int_{\text{all } \theta} \int_{\text{all } \phi} \int_{\text{all } x} \int_{\text{all } y} I(x, y, \theta, \phi)\ \ W((x - x_k), (y - y_l), (\theta - \theta_i), (\phi - \phi_j))\, d\theta\, d\phi\, dx\, dy$$

where $W(x, y, \theta, \phi)$ is a filter function. Energy conservation requires for the incoming power P_{cell}:

$$P_{cell} = \sum_{\text{all sexells kl}} \ \sum_{\text{all directions ij}} P_{ijkl}$$

By moving the sums inside the integrals, one can see that at any point all the filters must sum to 1. If $I(x, y, \theta, \phi)$ is constant (and equal to I), the power through the sexell is the intensity (which is then the

6. *Continuous* here means defined at every point.

Figure 4. Directions and solid angles for sexells. Cube shown represents 48 directions.

average intensity) multiplied by the 4D "volume" defined by the integral:

$$V = \int_{\text{all } \theta} \int_{\text{all } \phi} \int_{\text{all x}} \int_{\text{all y}} W((x - x_k), (y - y_l), (\theta - \theta_i), (\phi - \phi_j))\, d\theta\, d\phi\, dx\, dy$$

and P = I V is the power through the sexell. Note that the above equation also says that if W() is constant over a given region, and 0 elsewhere, then its value should be 1, and V is the real volume of the region. Possible models for the beams (parallel, cones, etc..) are special cases where the filter is non zero only in limited range of area and solid angle.

Once the power have been assigned to the beams, all is known about the flux entering the cell is the samples represented by the beams. To propagate (and shade) we need to reconstruct these samples into a continuous function. Assume that we are given a face and its sexells, all with assigned intensities for each of their directions. Given a point (x, y) on the face, and an arbitrary direction (θ, ϕ) the intensity of the ray at that point in that direction is:

$$I(x, y, \theta, \phi) = \sum_{\text{all sexells kl}} \; \sum_{\text{all directions ij}} WR((x - x_k), (y - y_l), (\theta - \theta_i), (\phi - \phi_j)) \;\; I_{ijkl}$$

WR() is the reconstruction filter.

The principle behind the computation of the light propagated to the outside of a cell propagation is that the incoming beams traverse the cell and intersect faces, where they constitute samples of the outgoing flux. New samples at the sexells centers for the assigned directions are computed by interpolating from these outgoing samples. The previous general formula is used, except that in this case I_{ijkl} are the intensities of the intersection of the incoming beams with the faces of the cell. They have the same orientations (ij) as the original beams, but they have different xy coordinates.

In practice it is not efficient to compute $I(x, y, \theta, \phi)$ for each outgoing direction of each sexell, since it would involve having to check on all incoming beams, many of which could be with no intensity and/or 0 weight. Instead we proceed as follows. For each incoming beam, we establish the list of outgoing beams for which WR() is non-zero. Then for that beam, we go through the list of the affected outgoing beams and add $WR((x - x_k), (y - y_l), (\theta - \theta_i), (\phi - \phi_j))\ I_{ijkl}$ to their intensity. Again kl here refers to the position of the incoming beam intersection with the opposite face. The intensity itself is the intensity of the beam unmodified.

The weight function we used is Gaussian spread, assuming separability of the spatial and angular contribution. In other words we compute $WR(x, y, \theta, \phi) = WRXY(x, y,) \times WR\Theta\Phi(\theta, \phi)$. The spatial term is then a traditional Gaussian spread. The angular term is interestingly very similar to the spread caused by a *Phong* like specular reflection. A given direction in practice has non-zero weight for a relatively small number of other directions (about 50 in practice). Once the intensity I_{in} is computed, the power to be stored is the intensity multiplied by the 4D volume V_{ijkl} represented by that particular beam. To each of the outgoing beams (*ijkl*) for which WR_{ijkl} is not 0, we assign the intensity

$$I_{ijkl_{out}} = \frac{I_{in}}{V_{ijkl}} \frac{WR_{ijkl} V_{in}}{\sum WR_{ijkl}}$$

4.4. Initialization

The intensities for the parallel beams have to be set initially based on incoming light. This light can come from light sources, or from reflection/refraction of parallel beams off the objects. The assignments have to be made in such a way that power is conserved, and that the distortion in the distribution of light is kept to a minimum.

We assume in our model that each sample beam represents the power collected all over the surface of the sexell in a direction centered around its assigned direction, and within a solid angle computed such that the whole 2π of the hemisphere is accounted for (as a partition of the hemisphere). Since our directions are determined by the centers of squares on the faces of 4 cubes sharing a corner at the center of the sexell (see Figure 4), the solid angle defined by each square can be computed from the definition of solid angle:

$$\Omega = \int_A \frac{\cos\theta \ dA}{r^2}$$

For a rectangle $(x_0, y_0), (x_1, y_1)$ in a plane at distance d from a point **P**, the solid angle sustained is:

$$\Omega = f(x_1, y_1) - f(x_1, y_0) - f(x_0, y_1) + f(x_0, y_0)$$

where

$$f(x, y) = \frac{1}{2} \sin^{-1}\left(\frac{(x^2 - d^2)\ (y^2 + d^2) - 2d^2x^2}{(y^2 + d^2)\ (x^2 + d^2)}\right)$$

These values are computed once and stored into a table, where they are indexed by direction indices when needed. Note that these values are needed mainly for transforming intensity samples into power. Depending on the filtering in practice a beam sample can have an influence beyond that solid angle.

4.4.1. Incoming Parallel Beams

Given a sexell and a parallel beam of direction (θ, ϕ) (with respect to the plane of the sexell) and incoming intensity (Exitance) $\mathbf{E_I}$, the power $\mathbf{P_I}$ incident on the sexell is

$$\mathbf{P_I} = \mathbf{E_I}\ \mathbf{A_I} \cos\theta$$

where A_I is the area of intersection between the beam and the sexell, measured in the plane of the sexell (see Figure 5). All this power is assigned to the direction (θ_i, ϕ_j) of the sexell closest to the direction (θ, ϕ). The power P_O outgoing from the sexell is equal to P_I. The outgoing intensity is

$$E_O = \frac{P_O}{A_S \cos\theta_i}$$

Figure 5. Incoming parallel beam

Figure 6. Point source

where A_S is the area of the sexell. So the intensity of the outgoing beam is

$$E_O = E_I \frac{A_I}{A_S} \frac{\cos\theta}{\cos\theta_i},$$

The reader can verify that the right values ensue when $A_I = 0$ (no solid angle/sexell intersection), or $A_I = A_S$ (solid angle entirely within the sexell) or $\theta = \theta_i$ (no change in direction).

Figure 7. Extended light source

4.4.2. Point Light Sources

For a point light source, the relevant intensity is the *radiant intensity*, expressed in power/solid angle (*e.g.* Watts/steradian). For an light source emitting uniformly through the sphere, its radiant intensity R_I is its total power divided by 4π. Assuming a point source **P** with radiant intensity R_I, the direction (θ_i, ϕ_j) for a given sexell, if the solid angle Ω_I is the angle covered by the intersection between the sexell and a cone based at **P** and equal to Ω_{ij} then the power to assign is:

$$P_I = R_I \ \Omega_I$$

Since in our model the intersection is always a rectangle (possibly null), the solid angle Ω_I can be computed by the formula given above. A possible approximation, valid when the source is relatively far from the sexell is to use to compute the value of the solid angle the ratio of the areas A_I of the intersection between the sexell and the solid angle Ω_{ij} and the area of the sexell A_S (see Figure 6).

$$P_I = \frac{R_I \ \Omega_{ij} \ \ A_I}{A_S}$$

The intensity of the sampled beam in that direction is:

$$I_O = \frac{P_I}{\Omega_{ij} A_S} \ .$$

Note that in this context Ω_{ij} denotes not only the value of the solid angle, but its direction and cross section as well. One can, to simplify, take A_I to be either 0 or A_S, depending on whether the ray from **P** in the direction (θ_i, ϕ_j) intersects the sexell. This amounts to using point sampling instead of area sampling at the sexell level.

4.4.3. Extended Diffuse Light Source

The full development of the formulae in the case of area light sources is somewhat lengthy, so we summarize the results. If we have a polygon of constant diffuse radiant exitance $\mathbf{E_I}$, whose normal makes an angle ψ with the normal to the sexell, and if the intersection of the oblique projection of the polygon along the direction $(\theta_\mathbf{i}, \phi_\mathbf{j})$ and the sexell is $\mathbf{A_I}$ (see Figure 7), then the incoming power $\mathbf{P_I}$ is:

$$P_I = \frac{E_I \ \Omega_{ij} \ A_I \cos \psi}{2\pi} \quad .$$

The only approximations made are that ψ is considered constant within the solid angle Ω_{ij}, and the solid angle centered on a surface element dA of the polygon is considered totally inside or totally outside of the sexell, depending on the position of its center with respect to the sexell.

4.4.4. Lazy Conversion of Sources

There is no need to actually convert the sources before they are used inside a cell. The only thing needed for each cell is its unbalance, which can be obtained for the faces of the cells from the total power received from the source. A face would be given pointers to the lights which directly send light to them. This way light sources are not affected by the method of propagation until they traverse a non-empty cell.

The power contribution to the cell from the "unevaluated" source still has to be computed. The answer is actually given by the form factor $F_{source\ face}$ between the light source and the relevant face of the cell. This form factor is the fraction of power emitted by the source and received by the face. It has to be multiplied by the total power emitted by the source, which is in turn the product of its radiant exitance by its area. So:

$$P_I = E_I \ A_{source} \ F_{source\ face}.$$

The form factor between a triangle in a general position and a square is not given in standard lists of form factors such as [Howe73a] and is actually rather painful to determine analytically. A simplification is to compute it from the sum of the power which would be attributed to all the sexells of the face in all their directions if this source had been converted. We have not implemented this lazy evaluation in the current version of FIAT.

4.5. Solving the Cell

4.5.1. Choosing the Next Cell to Solve

The best choice for the next cell to attempt to solve is the one with the highest unbalance, since it is the cell with the largest amount of undistributed power. This excess power is present because the cell contains light sources, or because it receives a lot of light not properly reflected or transmitted, or both. This particular choice of cell to deal with does not affect the convergence of the algorithm, but is important for the *progressive* nature of the information generated. Choosing the highest unbalance means that large amounts of light are distributed first. In this respect it is analogous to the choice made in the progressive refinement version of the radiosity-based algorithm [Cohe88a]. It should be again mentioned that cycles are avoided by not allowing reflection/refraction without any absorption.

4.5.2. The Local Illumination Model

If the cell is empty the only thing to do is to propagate the light through it (undistributed lights are merely pointed at if lazy evaluation is used). If the cell contains only one light source, solving the cell consists of propagating the light to its boundary, using the formulae given above. If only lights are present in a cell, one can deal with it either by assuming that light is not blocked by light sources (not a very realistic assumption) or that by actually testing whether any light beam direction within the cell is blocked before it reaches the boundaries.

If the cell contains any objects, we can deal with it by using a local illumination model if the cell contents are such that no interreflection is possible. This would be the case with the convex surface of opaque objects, but in our implementation we have reduced it to the easiest case, that of single polygons. In this case, we have to reflect or refract each beam of each sexell, and distribute the reflected/refracted intensity to the appropriate directions of the appropriate sexells. The two essential steps of this process, computing the ray/object intersection and computing the intensity of the reflected/refracted light, are part of every ray-tracer, and we have actually used the relevant modules of *optik* in our implementation. Care has to be taken, however, as most ray-tracers are rather loose with energy. In particular most local illumination computations are not normalized, and therefore do not guarantee that the surface does not reflect more energy that it receives. After each cell has been treated, we update the global unbalance and make sure that it has not increased. It is interesting to note that most systems have a reasonable model for the reflected ray (*optik* uses the Phong model), but not for the refracted ray, which often only gets a Fresnel coefficient. Hall and Greenberg are an exception [Hall83a], and use a Phong model for the exiting refracted beam. We treated the refracted ray exactly as the reflected ray, using a Phong-like distribution around the refracted direction.

4.5.3. Unsplittable Cells

If the cell cannot be dealt with and is too small to be split, one has to approximate the answer given by a determination of its global illumination. In our implementation we treat it like a cell with only one polygon: the first intersection in the direction of each sexell is taken, and the reflected/refracted intensity is computed. This, as mentioned before, is equivalent to ignoring the inter-reflection within the cell. Note that there are only six sexells in this case, one for each face of the cell if "too small" means a cell of a sexell unit size.

4.6. The Light Update Buffer

It is clear that the ultimate purpose of the method is not merely to balance the cells, not even to obtain the local illumination for every element of the scene, but to determine how much light is sent in the direction of the eye. In a maximal solution, one could envision keeping all the sexell information at the highest resolution possible in a view-independent manner, and once the view point is chosen selecting for the sexells closest to the window the direction closest to the direction of the eye, and take this array of intensity as the picture. For a $1\mathbf{K} \times 1\mathbf{K}$ picture, the storage cost would be in the terabyte neighbourhood. The first observation is that the sexell information is not needed anymore as soon as the cell is balanced (at the minor cost of re-allocating the sexells if the cell is unbalanced again). What is really needed is the local illumination information close enough to the surfaces to be used for rendering purposes. This information is available when we just finished updating a cell, and before we throw away the sexells and their information. If we are adopting a view-dependent solution, we only need to keep the intensity of light reflected in the direction of the eye for the visible parts of the objects (visible here means directly visible, for instance the first surface for a transparent object). Using again the linearity of the light effects, we can update this information as the algorithm progresses. To update the information about the visible surfaces, we only need an efficient way to access this information from a given cell.

4.6.1. Initial Dark

The information to be collected is very similar to that contained in a classic Z-buffer. The main difference is first in the fact that it normally is updated (as opposed to replaced in a normal frame buffer), and second that the cost of the information is rather high, and it is therefore prudent to ascertain that it will be needed before computing it. In our implementation we have decided to determine visibility from the eye only once. A frame buffer of a resolution determined by the user, but equal to or higher than the display resolution, is created. We call it the *Light Update Buffer*, or LUB for short. Each element of the LUB buffer contains the depth of the visible surface, and a light intensity initially set to 0. In our implementation the depth information is determined by a ray-caster (it could be as well the result of a Z-buffer algorithm). The LUB buffer is updated when a cell has just been balanced, and before its sexell information is thrown out.

4.6.2. Local Illumination Filtering

It is rather obvious that filtering the illuminating beams is important to remove artifacts from the spatial and angular discretization. This filtering is distinct from the traditional filtering to obtain pixel values, and the latter can be done in addition and independently of the former. Even though the filtering used for local illumination does not have to be the same as the one used for propagation, it is advantageous to keep them similar (for efficiency of coding) and consistent. The consistency is important, because, as Figure 8 indicates, the same object can have part of its surface illuminated through different number of stages of propagation filters, if they are in different cells. To obtain consistent results, the illumination filter should be able to equalize the local illumination conditions.

The filtering scheme we use for local illumination is as follows. For each visible point, we position a virtual sexell, with its normal set of direction, in its neighbourhood. Here neighbourhood means at the closest possible sexell position (remember that sexells have fixed sizes, and therefore imply a regular 3D grid in

Figure 8. Consistent Filtering is Important

space). For each direction of each real sexell (this is done in the propagation loop) find out if the beam has a non-zero contribution (using the same weights as used in the propagation filtering) contribution to any of the beams of this virtual sexell. If it does, add this intensity to the relevant beam of the virtual sexell. In pseudo-code:

```
Assign Virtual Sexells to the Visible Points within the Cell
for each direction of each sexell
    if a ray in that direction intersects a virtual sexell then
        Compute its contribution to the beams of the virtual sexell
    end if
    if the ray does not intersect a polygon then
        Propagate ray to outgoing sexells
    end if
end for
```

At the end of this process, each visible point has a set of average local beams, which are used in a straight-forward manner to compute the reflected light towards the eye (or towards the visible point direction, which could be different if it is an indirectly visible point). In pseudo-code:

```
for each direction of each virtual sexell
    for each visible point pointed at by this virtual sexell
        Compute the intensity of the reflected/refracted light
        Add intensity to LUB
        end for
end for
```

Note that if several visible points are close enough to have the same virtual sexell, they will have the same average local illumination. This is especially useful if the sexell resolution is much less than the pixel resolution in this neighbourood. To make this process more efficient, the first step of shading within a cell is to make a list of all the virtual sexells associated with the visible points within that cell. Then these are linked (and have a pointer to their associated visible point(s), and they are treated during the propagation step, since the process to determine the intensity of their beams is identical to the one to determine the intensity of the outgoing beams.

The propagation and local illumination filtering is not always enough to remove artifacts at the cell boundaries. One effective way to remove most of these artifacts to "filter" visible points across cells. A given visible point belongs to the cell which includes it, but also to the neighbourhood cells, with a different weight. It is quite enough to extend the influence of the cell to one quarter of its linear size all around it. See Figure 9

Figure 9. Cell Filtering

for illustration. While shading, then, the visible point's LUB entry is updated by more than one cell. Of course the weights given to the cells have to be normalized. This is not been implemented in our current version of FIAT, but does not require much additional code.

It is theoretically possible to use the view-independent information to generate several view-dependent LUBs at the same time (for instance for several frames of an animation if nothing moves in the scene except the point of view). The main obstacle to this is storage cost, and it is not clear that it would be often useful.

In our prototype implementation, we transfer the LUB intensity values directly to a display buffer, so that we can see the picture "develop" as the algorithm progresses. The display is initially black, then the objects appear as the cells containing light sources are treated, and rapidly we have an picture with most of the direct illumination computed. The rest of the time is spend "refining" this picture with indirect illumination.

4.6.3. Initial Bright

A straightforward observation is that if we could compute first the illumination of the visible surfaces taking only into account the light reaching directly the surface, then this intensity can only increase as more light reaches the surface indirectly. Note that in this case the objects blocked from the light are initially black. We can store this information in the LUB as the initial intensity values. We can then run an algorithm similar to the precedent, with the difference that we will not add (or compute) the local illumination from light coming to any surface directly, since it is already stored. That entails being able to distinguish between two kinds of light: the light that reaches the surface directly (or more exactly the light that reaches the surface directly and which has already been used to compute the initial shade) and the rest (light reaching the surface indirectly and light we could not initially deal with). To do this we need to have a dual system of light beams for each sexell, the direct and the indirect light. This at worst can almost double the memory requirement. In practice, however, the increase is less than this since we allocate dynamically sexell information, and most of the time direct and indirect light do not overlap. This approach presents two decisive advantages: the picture is usable as soon as the ray-caster is done initializing the LUB, and the shadows, which are an important clue in most scenes, are sharp when they are supposed to be.

In our implementation we use the same *optik* ray-caster to determine first the direct illumination and the shadows. Extended light sources are decomposed into point light sources for this purpose, since the ray-tracer does not deal with them directly. As soon as light is reflected/refracted off a surface it is transferred to the indirect light beams, and only these are used to update the LUB information. Note that in this context to be in shadow means not to receive light directly. After all is said and done, it might turn out that the points originally "in shadow" are very brightly lit because a glass sphere or a concave mirror has concentrated indirect light on them.

4.6.4. Sharp Reflection/Refraction

The goal of the "initial bright" solution is to make sure that the directly visible points (DVP) are rendered correctly and quickly, no matter how indirect light is computed or represented. In many applications, the images of objects seen by reflection and/or refraction should also be reasonably sharp. This is especially important in the case of simple transparency.

This can be achieved within FIAT using a slight extension of the "initial bright" strategy. Instead of simply ray-casting, the scene is ray-traced to some level. Let us assume for the time being that we ray-trace only to the first opaque object seen, an indirectly visible point (IVP). We then store in the LUB (or in the cell if visible pixels are pointed at from the cell) the relevant visible points, along with the direction from which it is visible. In the ray-casting case, this direction is always the eye direction, and does not have to be stored explicitly. For an IVP, it is the direction of the reflected or refracted ray. An other piece of information has to be stored, the extinction coefficient, that is the decrease in intensity due to the reflection/refraction along the path. The shading computation proceeds as before, by determining the average intensity falling on the visible point (directly or indirectly visible), and computing the light reflected/refracted in the direction of the ray (usually the eye direction, but for the indirectly visible points in the ray direction). Figure 10 illustrates the possible cases. Ps denote pixels (window pixels), VPs directly visible points, IVPs indirectly visible points. At P1, the visible point VP1 is stored. At P2, instead of the directly VP VP2, instead IVP2 is stored, with a coefficient of extinction corresponding to the absorption of light through the transparent object (we assume there is no significant refraction at VP2, therefore the direction associated with IVP2 is the same as for VP2). At P4, IVP4, seen by reflection, is stored instead of VP4. When the cells containing VP1, IVP2 or IVP4 are shaded, the light is reflected in the direction of the ray (be it direct or indirect) and multiplied by the relevant extinction coefficients, before being added to the LUB.

In this way sharp reflected and refracted images are obtained, and the only additional cost is storage for the direction and attenuation coefficient if needed. This is what we have done in the current implementation.

Of course storing only one IVP might be limiting. In many cases only one point does contribute significantly to the clear image, and this is quite adequate. Note that the other indirectly visible points will contribute through the indirect light mechanism. It is only that their influence will be through the indirect light contribution. If there is a need to consider more than one sharp RR combining at a given point, then all the relevant IVP have to be stored for a given pixel. This is possible (efficient techniques for encoding of trees of traces are given in [Sequ89a]) but practicality suggests stopping well before we store a sizeable tree of

Figure 10. Sharp Reflection/Refraction

ray traces for each pixel (again recall that the IVP not stored still contribute through the sexells).

Another consideration is that the first visible point (the one at which the first RR took place) can also contribute to the light sent towards the eye. For instance in Figure ?.? VP4 can also contribute light to the LUB at P4. If it contributes mainly in a specular way, it is the case just discussed previously, namely we can add the reflected or refracted direction to the tree of traces. If it contributes in a more diffuse way, in this case we also should add it to the list of VP at this pixel for shading purposes. The difference between the two situations is that for mirror-like RR, only the leaves of the tree of traces can tribute to the shading, when in the diffuse case, all the nodes of the tree can potentially contribute. A practical compromise is to always store the direct VP, and store at most one indirectly visible point. If we have a choice between a refracted or a reflected ray, we of course take the one with the smallest extinction coefficient.

5. Results and Analysis

5.1. Sample Scenes

5.1.1. Modelling

The scenes are currently modelled with triangles. A variety of standard modelling tools have been used, in particular generalized cylinders. Lights are modelled as objects, with a light attached to them, in the same way surface characteristics are attached to a surface. Even though it is not the case in our current implementation, light sources will be texture mapped if so desired.

5.1.2. Local Illumination

The local illumination is computed by the relevant modules from *optik*, using a standard Phong illumination model. The model is used for both the reflected and the refracted direction. The same exponent is used for both (mainly because the data structure has only one field for it), but there is no reason to believe it should be the same.

5.1.3. Basic Radiosity Effect

The first picture shows the result of running FIAT with a scene identical to the one used in the first "radiosity" paper [Gora84a]. All the surfaces are diffuse, and a light is about where the eye is. In this picture, each sexell had 192 directions, and a level of octree subdivision of a maximum of 2, and a cell had a minimum size of 2 sexells on a side. There is no filtering for propagation or shading. It is very similar to the result obtained with radiosity, albeit with obvious discontinuities due to the discretization of the beams and the effect of the cell boundaries. The total memory requirements were about 51 Mb.

5.1.4. A Specular Surface

The second picture shows a specular ashtray, modelled with about 600 triangles, lit by a directional light source to the left. Again there was no filtering, and the discretization shows. One can, however, distinguish the characteristic caustics formed by the reflection of the light on the sides of the ashtray. Thi sis an important effect, hard to obtain from any other method without *ad hoc* treatment. The maximum subdivision level was 3, the minimum size of a cell was 3 sexells on a side, the maximum memory requirement was 200 Mb, and the running time was 5h 38mn with 16Mb of main memory on a Silicon Graphics Iris 2D/20.

5.2. Parameters and Costs

The current implementation runs on a Silicon Graphics IRIS 4D/20 with 8MB of real memory and about 100MB of virtual memory.

The storage cost of the LUB is 16 or 20 bytes per element, or about 20 MB for a 1Kx1K buffer. Since current workstations have a frame buffer of this size already available, this is not too much of a demand.

The storage cost per sexell is about 2.2K with the information we store in our current implementation. Computing the number of sexells necessary to cover the faces of the initial cell allows a rough estimate of the number of sexells used at any point in the algorithm. The argument is that light in the algorithm propagates from cell to cell from the sources as a "front" as cells are reached from the sources. The number of sexells necessary to cover this front is of the order of its surface, and the maximum surface is when the boundaries of the original cell are reached. For a $128 \times 128 \times 128$ initial cell, this represents about 200MB of storage. So the total storage space needed for this type of resolution is about 300MB. This is large, but not unobtainable even on a workstation.

The complexity of the algorithms chosen to implement the paradigm is very favorable, even though the constants are fairly high. There is no operation or storage requirement which grows faster than linearly with respect to the number of objects in the scene. The number of cells in the octree is a linear function of the number of objects. As a result so is the number of sexells and of directions of sexells. There are operations within a cell which are at worst proportional to the square of the number of sexell and/or directions, and this number, even though a constant, can be quite high. The costliest operation is by far the shading, and it is a good strategy to use a good, if expensive, shading model for direct illumination as entered in the light update buffer, and to use a cheaper model to account for the reflected light.

5.3. Analysis

5.3.1. General Advantages

There are no assumptions about the local illumination model. It actually can be changed without modifying the structure of the solution. Its only impact is in the number and intensity of beams sent to adjacent cells, and therefore in the number of unbalanced cells each cell can create. For instance if only a reflected and/or refracted ray is computed at the local level, the solution will very much look like a (fuzzy) ray-traced picture.

There is very little limitation on the light/matter interaction. For instance fluorescent material can be included. It only means that incoming light is redistributed in a different colour and in a diffuse mode. The only thing barred is non-linearity. For instance material whose index of refraction changes strongly with temperature can be a problem since the reflected/refracted light would be a function of the power previously absorbed.

The algorithm is adaptive in two distinct and useful ways. It is adaptive in space, since cells are subdivided only if they contain too many things. It is adaptive in power, since cells are looked at only if they receive or emit significant amounts of light. The latter is especially useful since it allows putting a time limit on the rendering. If only primary sources are to be considered one can put the tolerance very high, and no local illumination will be produced. By lowering the tolerance one can obtain more and more detailed information. The same is true about the size of sexells, but it cannot be modified during the course of a scene computation.

The algorithm is also progressive. By choosing the cell with the highest unbalance to be balanced next, this ensures propagation of the strongest light first. Near-cycles (that is cycles where the light intensity being circulated decreases, but only slowly) are a potential problem in this case, because the algorithm can spend too much time on large amounts of light being merely "recycled". Note that one can slightly bias the choice of cells towards the ones nearest to the eye, on the assumption that they are more likely to contain visible surfaces.

It is difficult to decide if view-dependence is an advantage or a problem. As we described it, the approach is basically view-independent, but the information gathered is view-dependent. As well in most cases the choice of the size of sexells will be view and resolution dependent. The answer is mostly a matter of applications. If the scene is a structure to be viewed from different angles, as in most design applications, view-independence is an asset. In most other applications some elements of the scene change from frame to frame and view-dependence is not a burden.

5.3.2. General Problems

The most obvious problem is the need to discretize space and directions. This only exacerbates the sampling problems plaguing computer graphics. In a mostly diffuse environment it will not be too noticeable, but it becomes serious with transparent and highly specular material. This is mitigated by the fact that it is the only method so far which deals with both reasonably well. If the direct illumination is computed first then this guarantees sharp highligths and sharp shadows (where they should be sharp).

The other general problem is storage space. Even though it does not reach unmanageable proportion, it is a serious limit to the use of the present implementation on an ordinary workstation.

6. Prospective

We obviously have a lot of exploration of the various parameters left to do.

6.1. Filtering

We have to investigate the filters given the best tarde-off between diffusion and coherence.

6.2. Local Illumination

The independence from the local illumination model is a very useful feature of our approach. Standard local illumination models present serious problems since in general they do not guarantee the conservation of energy. We are currently developing a relatively simple model parametrized so that it always do so. As a result of the flexibility in the choice of local model, we can envision the use of more sophisticated models. We plan to integrate a anisotropic reflection model that we are developing [Poul89a] into the current system.

As mentioned before, we have temporarily adopted an RGB model for shading. This was expedient, but unacceptable in the long run. We plan to change this. Another related issue is that the power computation is a function of the "meaning" of the primaries used. They do not have to be weighted equally. The two roles played by the power, as a criterion for energy balance and as a guide to select the next cell to be treated, can be separated. In particular for the purpose of the second role, the power (or its functional equivalent) can be computed taking the luminous efficiency into account, so that the most visible effects are dealt with first.

6.3. Participating Media

It is very important to include *atmospheric effects* into a realistic rendering system. This has been done in the context of radiosity [Rush87a], and we intend to add it to FIAT. At the local level, the inclusion of such effect is easy. There might be, however, a problem with cost, since if the medium fills most of the cells, every cell can become costly to compute [Klas87a]. A decision has also to be made about whether to apply a differential model, which would be easier to formulate, but which would force more subdivision of cells, or a macroscopic model, where the effects are integrated and can be computed with bigger cells. Participating media also force us to include the eye position in the original cell, which might make the cell much bigger. In addition we have to keep information about the light reaching the eye in the vicinity of the eye, to complement the information normally collected in the LUB, and we have to combine the two to determine the final picture.

6.4. Parallelism

Since this paradigm is rather demanding in storage and computation, it is natural to consider parallel implementations. It is obvious at first glance that there are features of the implemented algorithms very favourable to SIMD approaches, such as the computations on sexells and directions of sexells. There are, in addition, many opportunities for a coarser grain of parallelism. In particular many cells can be treated concurrently, with no exchange of information between processors once the cell and its list of objects have been distributed. Synchronization is only needed once the cell is dealt with, and substantial amounts of information flow only when the sexell data is transmitted to neighbours and when the LUB is updated. A report on our experience with parallelism with FIAT has been submitted to Eurographics 1990.

7. Conclusions

We have presented a different paradigm for solving the global illumination problem, driven not by objects or rays, but by the power balance of sections of space. This approach, combined with traditional rendering for the visible surfaces, leads to a system which can effectively render scenes with any mixture of diffuse and specular surfaces. The result is an adaptive, progressive rendering system which can deal with a wider range of effects than any other approach used so far in computer graphics. Even though its space and time requirements are not modest we feel that it is powerful, flexible and promising enough to warrant much more investigation.

Glossary

Symbol	Meaning	Unit	Relations	Remarks
A	Area	$\mathbf{m}^2$		
E	Energy	J (Joule)		
P	Power	W (Watts)	E/time	Called flux for radiation
E	Intensity	W.$\mathbf{m}^{-2}$	$\frac{\mathbf{P}}{\mathbf{A}}$	Normally called *Exitance* for a source *Irradiance* for a receiver
R	Radiant Intensity	W.sr^{-1}	$\frac{P}{\Omega}$	
I	Intensity	W.$m^{-2}.sr^{-1}$	$\frac{P}{A\Omega}$	
Ω	Solid Angle	sr (steradian)	$\frac{A}{r^2}$	A area on sphere of radius r
cell	cube in space			Variable size
sexell	square on face of cell			Fixed size in space

Acknowledgements

We acknowledge the support of NSERC through operating grants, a University Research Fellowship and an equipment grant which considerably facilitated this research. The support of the Province of Ontario through an Information Technology Research Centre Grant is also gratefully acknowledged. The first author also benefited from substantial support from the University of British Columbia. Many members of DGP and Imager contributed mightily. In particular we thank George Drettakis for major conceptual and software support, John Buchanan for the modelling (based on software by Andrew Chasin), Andrew Woo for all-around *optik* expertise, Ian Small and George Drettakis again for taming and managing the machines and systems, Pierre Poulin for healthy skepticism, and Brian Allum of Silicon Graphics for making various machines available when needed. We also thank Giovanni Agnelli and the Lever Brothers.

References

Aman87a.
J. Amanatides and A. Woo, Optik Users' Manual,'' DGP Technical Report 1987-2, University of Toronto, August 1987.

Baha87a.
E. Bahar and S. Chakrabariti, Full-Wave Theory Applied to Computer-Aided Graphics for 3D Objects,'' *IEEE Computer Graphics and Applications*, vol. 7, no. 7, pp. 46-60, July 1987.

Baum89a.
D. R. Baum, H. E. Rushmeier, and J. M. Winget, Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors,'' *Computer Graphics*, vol. 23, no. 3, pp. 325-334, July 1989.

Buck89a.
C. Buckalew and D. Fussell, Illumination Networks: Fast Realistic Rendering with General Reflectance Functions,'' *Computer Graphics*, vol. 23, no. 3, pp. 89-98, July 1989.

Cohe85a.
M.F. Cohen and D.P. Greenberg, The Hemi-Cube: A Radiosity For Complex Environments,'' *Computer Graphics*, vol. 19, no. 3, pp. 31-40, July 1985.

Cohe88a.
M.F. Cohen, S.E. Chen, J.R. Wallace, and D.P. Greenberg, A Progressive Refinement Approach to Fast Radiosity Image Generation,'' *Computer Graphics (SIGGRAPH '88 Proceedings)*, vol. 22, no. 4, pp. 75-84, held in Atlanta, Georgia, August 1988.

Cook84a.
R.L. Cook, T. Porter, and L. Carpenter, Distributed Ray Tracing,'' *Computer Graphics*, vol. 18, no. 3, pp. 137-145, July 1984.

Cook86a.
R.L. Cook, Stochastic Sampling in Computer Graphics,'' *ACM Trans. on Graphics*, vol. 5, no. 1, pp. 51-72, January 1986.

Devi87a.
O. Devillers, C. Puech, and F. Sillion, CIL: Un Modele d'Illumination Integrant les Reflexions Diffuse et Speculaires,'' Technical Report LIENS-87-12, Laboratoire d'Informatique de l'Ecole Normale Superieure, Paris, October 1987.

Dipp85a.
M.A.Z. Dippe and E.H. Wold, Antialiasing through Stochastic Sampling,'' *Computer Graphics*, vol. 19, no. 3, pp. 69-78, July 1985.

Fium87a.
E. Fiume and A. Fournier, The Complexity of Abstract Ray Tracing,'' DGP Technical Memo DGP87-1, May 1987.

Glas84a.
A.S. Glassner, Space Subdivision for Fast Ray Tracing,'' *IEEE Computer Graphics and Applications*, vol. 4, no. 10, pp. 15-22, October 1984.

Gora84a.
C.M. Goral, K.E. Torrance, D.P. Greenberg, and B. Battaile, Modeling the Interaction of Light Between Diffuse Surfaces,'' *Computer Graphics*, vol. 18, no. 3, pp. 213-222, July 1984.

Hall83a.
R.A. Hall and D.P. Greenberg, A Testbed for Realistic Image Synthesis,'' *IEEE Computer Graphics and Applications*, vol. 3, no. 8, pp. 10-20, November 1983.

Heck84a.
P. Heckbert and P. Hanrahan, Beam Tracing Polygonal Objects,'' *Computer Graphics*, vol. 18, no. 3, pp. 119-127, July 1984.

Howe73a.
J. R. Howell, *A Catalog of Radiation Configuration Factors,* McGraw-Hill Book Co, N.Y, 1973.

Imme86a.
D.S. Immel, M.F. Cohen, and D.P. Greenberg, A Radiosity Method for Non-Diffuse Environments,'' *Computer Graphics*, vol. 20, no. 4, pp. 133-142, August 1986.

Kaji86a.
J.T. Kajiya, The Rendering Equation,'' *Computer Graphics*, vol. 20, no. 4, pp. 143-150, August 1986.

Klas87a.
R.V. Klassen, Modeling the Effect of the Atmosphere on Light,'' *ACM Trans. on Graphics*, vol. 6, no. 3, pp. 215-237, July 1987.

Lee85a.
M.E. Lee, R.A. Redner, and S.P. Uselton, Statistically Optimized Sampling for Distributed Ray Tracing,'' *Computer Graphics*, vol. 19, no. 3, pp. 61-67, July 1985.

Mora81a.
H.P. Moravec, 3D Graphics and the Wave Theory,'' *Computer Graphics*, vol. 15, no. 3, pp. 289-296, August 1981.

Neum89a.
L. Neumann and A. Neumann, Photosimulation: Interreflection with Arbitrary Reflectance Models and Illumination,'' *Computer Graphics Forum*, vol. 8, pp. 21-34, 1989.

Newm73a.
W.M. Newman and R.F. Sproull, *Principles of Interactive Computer Graphics (First Edition),* McGraw-Hill Book Co, N.Y, 1973.

Nish85a.
T. Nishita and E. Nakamae, Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflection,'' *Computer Graphics*, vol. 19, no. 3, pp. 23-30, July 1985.

Rush87a.
H.E. Rushmeier and K.E. Torrance, The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium,'' *Computer Graphics*, vol. 21, no. 4, pp. 293-302, July 1987.

Same84a.
H. Samet, The Quadtree and Related Hierarchical Data Structures,'' *Computing Surveys*, vol. 16, no. 2, pp. 187-260, June 1984.

Sequ89a.
C. H. Sequin and E. K. Smyrl, Parameterized Ray Tracing,'' *Computer Graphics*, vol. 23, no. 3, pp. 307-314, July 1989.

Shin87a.
M. Shinya, T. Takahashi, and S. Naito, Principles and Applications of Pencil Tracing,'' *Computer Graphics*, vol. 21, no. 4, pp. 45-54, July 1987.

Sill89a.
F. Sillion and C. Puech, A General Two-Pass Method Integrating Specular and Diffuse Reflection,'' *Computer Graphics*, vol. 23, no. 3, pp. 335-344, July 1989.

Thib87a.
W.C. Thibault and B.F. Naylor, Set Operations on Polyhedra Using Binary Space Partitioning Trees,'' *Computer Graphics*, vol. 21, no. 4, pp. 153-162, July 1987.

Wall87a.
J.R. Wallace, M.F. Cohen, and D.P. Greenberg, A Two-Pass solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods,'' *Computer Graphics*, vol. 21, no. 4, pp. 311-320, July 1987.

Wall89a.
J. R. Wallace, K. A. Elmquist, and E. A. Haines, A Ray Tracing Algorithm for Progressive Radiosity,'' *Computer Graphics*, vol. 23, no. 3, pp. 315-324, July 1989.

Ward88a.
G.J. Ward, F.M. Rubinstein, and R.D. Clear, A Ray Tracing Solution for Diffuse Interreflection,'' *Computer Graphics*, vol. 22, no. 4, pp. 85-92, August 1988.

Whit80a.
T. Whitted, An Improved Illumination Model for Shaded Display,'' *Comm. of the ACM*, vol. 23, no. 6, pp. 343-349, June 1980.