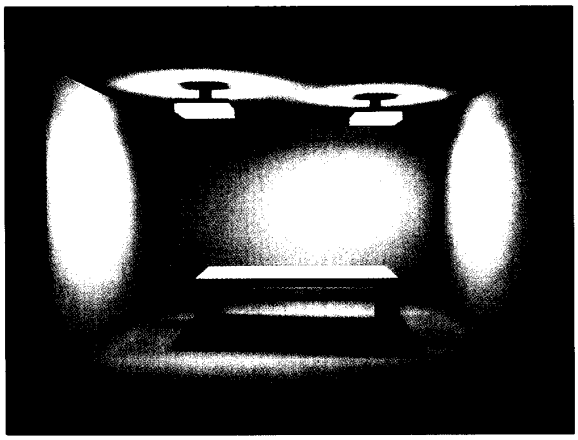
Radiosity

Radiosity Redistribution for Dynamic Environments

David W. George, François X. Sillion, and Donald P. Greenberg
Cornell University



We extend the radiosity algorithm to dynamic environments, providing global-illumination simulations to scenes that are modified interactively. The illumination effects introduced by a change in position, shape, or attributes of any objects in the scene are computed very rapidly by redistributing the energy already exchanged between objects. Corrections are made by shooting positive and negative energy, accounting for both increased illumination and the creation of shadows. We use object coherence to minimize computation and progressive-refinement techniques to accelerate convergence. The new algorithm yields excellent approximations to the exact solutions at interactive speeds.

The depth cues and visual realism provided by global-illumination algorithms greatly improve the ability of the user to understand the content of an environment. Although current versions of the ray-tracing and radiosity algorithms provide the necessary shadows and reflections, their computation time is too long to be applicable to changing environments. Techniques are needed to provide global-illumination effects, such as shadows with penumbrae, to environments that are being manipulated interactively.

Such techniques must be fast enough to allow the user to interact with the environment, immediately providing a new global-illumination solution whenever changes to the scene are made. The user must be able to add, remove, change, or move objects without restrictions on their geometry, their surface characteristics, or the path of their motion. The user must also be able to move the observer position freely throughout the environment.

We present an algorithm, based on progressive-refinement radiosity, which approaches interactive speeds and meets all these criteria. In response to a change in the environment, the algorithm updates the existing radiosity solution. For moderately complex environments, the algorithm provides a good approximation to the new radiosity solution in near real time

0272-17-16/90/0700-0026$01.00 ©1990 IEEE

on today's workstations. Given more time, the algorithm continues to refine the solution and eventually provides a new converged solution.

The algorithm capitalizes on temporal coherence in object space, recognizing that the changes in the environment from one time step to the next are likely to be small. To achieve interactive speeds, the global-illumination solution is not completely recomputed each time the environment changes. Rather, the existing solution is modified. Thus, computation is initially limited to the parts of the environment that have changed. Furthermore, the algorithm processes changes in the environment in order of decreasing importance, handling the greatest energy changes first.

Until recently, the calculation cost for realistic imaging was considered too high to be applied to nonstatic scenes. Previous efforts to display dynamic environments using global illumination have been limited to scenes in which the paths of all dynamic objects are known in advance.[1,2]

Radiosity

The original radiosity method solved the diffuse global-illumination problem for environments made up of polygonal patches.[3] The amount of light leaving each patch can be expressed as a combination of its emitted light and its reflected light. The radiosity leaving patch i is given by

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j \qquad (1)$$

where

B_i = radiosity of patch i
(energy\unit area\unit time)
E_i = radiosity emitted from patch i
(energy\unit area\unit time)
F_{ij} = form factor from i to j
(fraction of all energy leaving patch i that arrives at patch j)
ρ_i = reflectivity of patch i
n = number of patches in the environment

All B_i values can be solved using either the full-matrix method or the progressive-refinement approach. The full-matrix algorithm solves each B_i value, one at a time, by "gathering" light contributions from all other patches in the scene. The progressive-refinement approach simultaneously solves all patch radiosities by repeatedly choosing a patch to "shoot" and distributing that patch's energy to all other patches.[4] The latter approach is attractive because it provides a very good approximation to the final solution after only a few iterations.

Errors are frequently introduced in the form-factor calculation because of the discretization of the environment or the resolution of the visibility calculation.[5] The quality of the solution can be improved by patch subdivision[6] and by the use of ray tracing to calculate form factors.[7] Both techniques are used in the form-factor calculations for the redistribution algorithm, which is described in the following sections.

Modification of an existing radiosity solution

Ultimately, the radiosity approach will be useful for dynamically changing environments only if the radiosity solution can be updated very quickly in response to user input. Unfortunately, complete recomputation of the solution is currently too slow. This section describes how—starting with a radiosity solution computed using the progressive-refinement radiosity algorithm—a progressive transition to the new solution can be achieved.

By using a second level of progressive refinement, the energy that has already been "propagated" is "redistributed," yielding a good approximation to the newly converged solution very quickly. A single iteration of the traditional progressive-refinement radiosity algorithm is referred to as a *propagation* operation, because it propagates energy through the environment. A different operation, called *redistribution*, computes a correction for the energy leaving a given patch. This accounts for the changes in the environment.

Redistribution of the energy

The energy in an environment is redistributed whenever objects are added, removed, moved, or changed.

Adding an object

Consider the work necessary to adjust a radiosity solution when a new object is added to the environment. The energy that has been propagated through the environment must now be redistributed to account for the new object. Some patches will now re-

ceive more energy, and some will receive less (see Figure 1).

Obviously, the patches on the newly introduced object will receive more energy in the new solution than in the original solution. During the initial propagation process, the new object was not in the environment, so it did not receive any energy. Patches on the new object, which are now visible from any illuminated patch, should receive their share of the energy propagated from that patch. The correction is made by shooting "positive" energy from that patch toward visible patches on the new object.

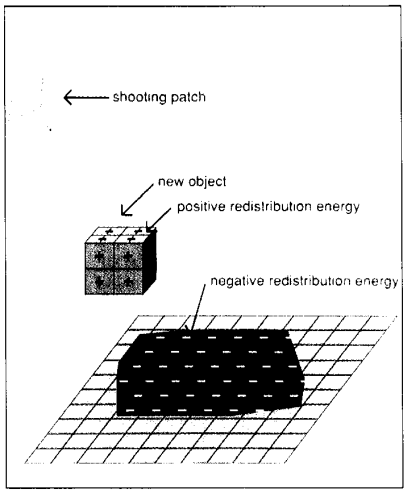


Figure 1. Patches on the new object receive more of the energy propagated from a patch, and patches occluded by the new object receive less energy.

Patches in the shadow of the new object will receive less energy than in the original solution. If a patch is partially or completely occluded from an illuminated patch by the new object, then some of the energy previously received from the illuminated patch should be canceled. This is accomplished by shooting "negative" energy.[8]

If the most influential patches in the environment redistribute their energy first, a good approximation to a new solution is obtained after only a few redistribution shots. We discuss methods for rating each patch's importance later.

Eventually, patches that have received energy adjustments will propagate those adjustments through the environment. That is, each patch that has received redistribution energy will shoot some fraction of that energy to other patches.

Removing, moving, or changing an object

The result of removing an object from the environment is nearly symmetric to the result of adding one. From each of the most important patches in the environment, positive energy is shot toward patches that are no longer occluded from the shooting patch by the disappearing object. This erases the object's shadow. The radiosities of patches on the removed object are set to zero because the object is no longer in the environment. Energy that was distributed by patches on the dynamic object is canceled by shooting negative energy.

Moving an object or changing its shape can be reduced to removing it from the environment, making the appropriate changes, and adding it back into the environment.

Changing illumination attributes, such as reflectivity or self-emittance of an object, is a simpler problem, because it is sufficient to compute and propagate a radiosity correction for each patch on the object.[9] No redistribution of other patches' energy is required.

A mathematical foundation

A mathematical foundation for energy redistribution is derived from the basic radiosity equation (see Equation 1). For simplicity, we consider only the "add object" operation here.

In the radiosity equation for a changing environment, time-dependent values are represented by specifying the time as a superscript. Values for the original radiosity solution are represented by superscript $t-1$, and values for the new radiosity solution, which must be computed, are represented by superscript t. Superscript Δt is used to represent changes in time-dependent values between time $t-1$ and time t.

We can write the basic radiosity equation as a time-dependent equation.

$$B_i^t = E_i + \rho_i \sum_{j=1}^{n} F_{ij}^t B_j^t \qquad \textbf{(2a)}$$

and

$$B_i^{t-1} = E_i + \rho_i \sum_{j=1}^{n} F_{ij}^{t-1} B_j^{t-1} \qquad \textbf{(2b)}$$

Using the relations $B_i^t = B_i^{t-1} + B_i^{\Delta t}$ and $F_{ij}^t = F_{ij}^{t-1} + F_{ij}^{\Delta t}$, we can rewrite Equation 2a in terms of the known values from the previous time step $(t-1)$ plus the incremental changes during the period Δt.

$$B_i^{t-1} + B_i^{\Delta t} = E_i + \rho_i \sum_{j=1}^{n} (F_{ij}^{t-1} + F_{ij}^{\Delta t})(B_j^{t-1} + B_j^{\Delta t}) \qquad \textbf{(3)}$$

Expanding all terms and using Equation 2b to subtract B_i^{t-1} from both sides gives the equivalent expression.

$$B_i^{\Delta t} = \rho_i \sum_{j=1}^{n} F_{ij}^{\Delta t} B_j^{t-1} + \rho_i \sum_{j=1}^{n} F_{ij}^{t} B_j^{\Delta t} \quad (4)$$

A *redistribution term*, R_i, is defined to be

$$R_i = \rho_i \sum_{j=1}^{n} F_{ij}^{\Delta t} B_j^{t-1} \quad (5)$$

Substituting Equation 5 into Equation 4 gives the radiosity redistribution equation.

$$B_i^{\Delta t} = R_i + \rho_i \sum_{j=1}^{n} F_{ij}^{t} B_j^{\Delta t} \quad (6)$$

The radiosity redistribution equation (Equation 6) is in the same form as the basic radiosity equation (Equation 1). There are only two differences: The emission term, E_i, has been replaced by the redistribution term, R_i, and the unknown variable in the equation has changed from B_i^t to $B_i^{\Delta t}$. Thus, we can find $B_i^{\Delta t}$, the change in radiosity at each patch, in two steps: (1) compute all redistribution terms, and (2) solve $B_i^{\Delta t}$ with the traditional radiosity techniques used to solve B_i.

The redistribution algorithm

This algorithm computes the redistribution term for each patch in the environment.

"Shooting" approach

The redistribution terms can be computed using either "gathering" or "shooting." With gathering, R_i is computed one patch at a time. With shooting, each patch can shoot all of its redistribution energy, partially solving for many patches' R_i terms with each shot. Patch j shoots its redistribution energy by adding $\rho_i F_{ij}^{\Delta t} B_j^{t-1}$ to R_i for each patch i so that $F_{ij}^{\Delta t} \neq 0$. If the most important patches shoot their redistribution energy first, then many R_i terms approach their final value after a small number of shots. Therefore, shooting is the preferred method for computing redistribution terms.

Positive and negative corrections

Suppose that a new object is added to the environment and that patch j is chosen to shoot its redistribution energy. Patches in the environment that have changed form factors with respect to j (that is to say, the set of i so that $F_{ij}^{\Delta t} \neq 0$) fall into two categories: patches on the new object and patches that have become occluded from j by the new object.

Each patch i on the new object was previously not in the environment, so $F_{ij}^{t-1} = 0$. In the new environment, some of those patches may be visible to j, in which case $F_{ij}^{\Delta t} = F_{ij}^{t} > 0$. Therefore, j shoots "positive" redistribution energy toward patches on the new object (see Figure 1).

Patches whose visibility from j has been fully or partially occluded by the new object have decreasing form factors with respect to j. For those patches, $F_{ij}^{\Delta t} < 0$; therefore, j shoots "negative" redistribution energy toward them (see Figure 1). Note that both F_{ij}^{t-1} and F_{ij}^{t} must be computed to evaluate $F_{ij}^{\Delta t}$.

Using partially converged solutions

We can also use the energy-redistribution algorithm to adjust partially converged radiosity solutions. Only the energy that was propagated before the environment changed should be redistributed. Thus, the radiosity subject to redistribution from patch j equals $B_j - \Delta B_j$, where ΔB_j is the component of B_j that patch j had not shot at the time of the change.

Algorithm

The algorithm to redistribute a single patch's energy is given in Figure 2. The *find_most_important_patch* and *find_patches_in_shadow* operations are described next.

Implementation

We designed the current implementation for interactive use, where the user constantly supplies input by adding, moving, deleting, or changing objects in the environment. A simple model for the implementation is shown in Figure 3; a more complete model is presented in Figure 4.

Finding the patches in shadow

To redistribute a patch's energy, form factors must be computed from that patch to patches on the new object as well as patches occluded by the new object. Before redistributing a patch's energy, a shadow volume is constructed to determine which patches might be occluded by the new object.[9] Thus, we do not consider a large number of patches whose form factors have not changed. To compute the form factors that have changed, we used ray-tracing techniques.[7]

Patch importance

Due to the time limitations required for interactivity while the user is supplying input, only a few patches can redistribute their energy before the display is up-

```
j = find_most_important_patch();
B_shoot = B_j − ΔB_j; /* energy shot in the past */
/*shoot positive energy toward the new object */
for i = each patch on the new_object†
    Rdist = ρ_i F_ij^Δt B_shoot;
    B_i = B_i + Rdist;
    ΔB_i = ΔB_i + Rdist;
endfor
/* shoot negative energy toward the new object's
     shadow */
shadow_list = find_patches_in_shadow
     (j, new_object);
for i = each patch in the shadow_list†
    Rdist = −1 × ρ_i F_ij^Δt B_shoot;
    B_i = B_i + Rdist;
    ΔB_i = ΔB_i + Rdist;
endfor
```

Figure 2. Pseudocode for a single redistribution shot.

†If patch subdivision is used, energy is shot to sample points instead of patches.

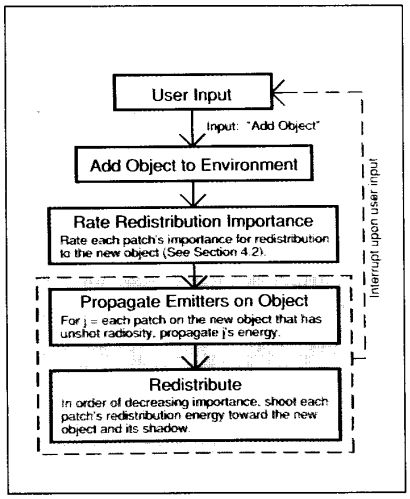


Figure 3. A simple model for an interactive implementation. Once inside the shaded area, the display is updated and the input device is polled after every few operations. If new input is received, the algorithm is interrupted.

dated and more input is accepted. Therefore, the few patches that are chosen should be those with the most energy to redistribute. In contrast to the usual progressive-refinement radiosity algorithm, where the relevant ranking parameter is the total unshot energy at each patch, only the energy shot toward the dynamic object is considered.

To accelerate the computational speed, a heuristic method can be used to predict which patches will shoot the most redistribution energy to the object. One such heuristic algorithm uses a quick estimate of the relative amounts of energy that each patch would shoot toward the new object. If a sample point, s, is chosen, for example, in the geometric center of the dynamic object, an estimate of the energy received at s from every patch i in the environment can be obtained very rapidly. Assume a differential area ds, positioned at s, with its normal always pointing toward i. An estimated form factor between ds and patch i is computed. The product of that form factor and the radiosity of patch i is used to rank patch i in the shooting order of the patches.

If there are multiple dynamic objects or multiple sample points within a single dynamic object, then the relative importance of shooting patch i is found by summing the estimated values computed at all sample points.

The accuracy of the estimate depends on three factors. First, the new object must be small relative to the distance between s and i to avoid inaccuracies in the form-factor calculation.[5] Second, the estimate will be more accurate if the projected area of the new object is nearly the same in all directions, since the actual amount of energy that patch i would shoot toward the object is proportional to the area of the object projected in the direction of i. Third, the estimate will be inaccurate if the sample point is occluded from i while some parts of the new object are not occluded, or vice versa. Results generated using this heuristic approach are presented in the section on our results.

Interleaving redistribution and propagation shots

When a new object is added, many patches in the environment may have unshot radiosity. If the new object is an emitter, patches on its surface have radiosity to propagate. If the initial radiosity solution is not converged, other patches have unshot radiosity as well. After several redistribution shots, additional unshot radiosity will have accumulated at patches on the dynamic object, and unshot negative radiosity will have accumulated at shadow patches. To reach a fully converged solution, we must propagate all unshot radiosities throughout the environment.

Propagation and redistribution shots can be interleaved. We discuss two possible interleaving strategies below.

Strategy one: Redistribute first

The first strategy gives priority to redistribution shots. When an object is added to the scene, patches are chosen in order of decreasing importance to shoot their redistribution energy. Redistribution continues until the amount of energy redistributed during each iteration falls below some energy threshold. This threshold may be zero, in which case all patches' energy is redistributed before any further energy is propagated.

After the threshold is reached, propagation shots are interleaved with any remaining redistribution shots. We need a heuristic method to choose between the patch with the most energy to redistribute and the patch with the most unshot propagation energy. Note that any energy propagated after an object was added is not redistributed.

This interleaving strategy is based on the observation that the user is normally perceptually focused on the changing parts of the scene. The portions of the image with the greatest changes are the dynamic objects and their shadow regions. Therefore, redistribution shots are given priority to refine these parts of the environment quickly.

If a new object is an emitter, the user's attention is focused on the effect that its light source has on the environment. In this case, light energy at patches on the dynamic object is propagated before any energy is redistributed.

This strategy quickly reduces radiosity errors at patches on the new object and in its shadow. Note that each redistribution shot is much faster than a propagation shot, because redistribution shots affect only the dynamic object and its shadow region.

In Figure 4, we present this strategy by the path branching to the left after the "propagate emitters on object" box.

Strategy two: Interleave redistribution and propagation

In the second strategy, we interleave propagation and redistribution shots completely (see the branch to the right in Figure 4). Before each shot, we compare the patch with the most unshot propagation energy to the patch with the most energy to redistribute. Redistribution shots only adjust for the most recent change in the environment; this restriction will be lifted in the next section.

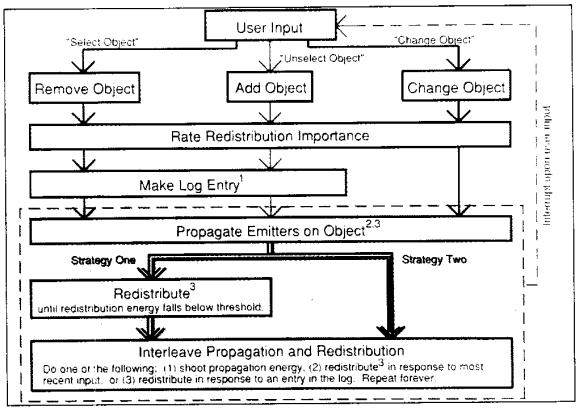


Figure 4. A complete model of the interactive implementation. Paths through the flow chart are color coded for the three different types of user input.

[1]Log entry is either "remove object" or "add object," depending on input.
[2]If input is "select object," negative energy is propagated from each patch on the object that shot energy in the past.
[3]If input is "change object," radiosity changes are stored in a separate buffer.

This strategy provides visualization of two effects more quickly than the first strategy. First, light reflected from the dynamic object may be propagated very quickly. Second, patches in the dynamic object's shadow may propagate their negative energy much earlier.

Unlike the first strategy, there is no guarantee that errors on a new object's surfaces and in its shadows are reduced quickly.

Using a log to eliminate error

The cost of providing interactivity is that error is introduced into the radiosity solution. Whenever the user specifies a change to the environment, the solution process is interrupted, as shown in Figure 3. Energy that has not been redistributed at the time of the interrupt is never redistributed, causing the radiosity solution to converge to an incorrect solution.

This error can be eliminated at a later time if we keep a log of the history of all changes to the environment. Each log entry records a single environment change and the patch with the most energy left to redistribute in response to the change.

To minimize the number of events stored in the log, we record a series of changes to a single object as a single "remove" and "add" operation. When the user selects an object for change, the object is removed from the radiosity solution. While the object is selected, radiosity that is redistributed to adjust for changes of the object is stored separately, added to the radiosity solution only at display time, and erased if

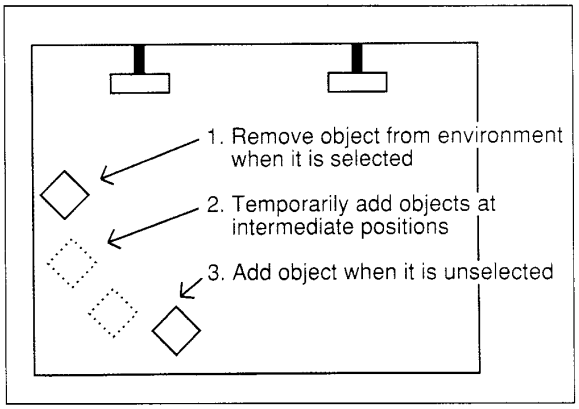


Figure 5. Reducing a series of changes, in this case "move" operations, to a single "remove" and "add" operation.

the object is changed again. Because these intermediate changes are temporary, they do not introduce any error into the solution. When the object is unselected, it is added back into the environment. Figure 5 shows a cube being selected, moved through a room, and unselected.

Results

We have presented the performance of the redistribution algorithm for two changing environments, one simple and one complex. In response to changes in each of the environments, the redistribution algorithm converges from the previous radiosity solution to a new solution much faster than the traditional progressive-refinement radiosity algorithm.

Statistics for test environments

The rate of convergence of each algorithm was measured by plotting the algorithm's solution quality versus its execution time. The quality of a partial solution is determined by comparing it to a converged solution for the same environment. For each patch, we convert the difference between current and converged

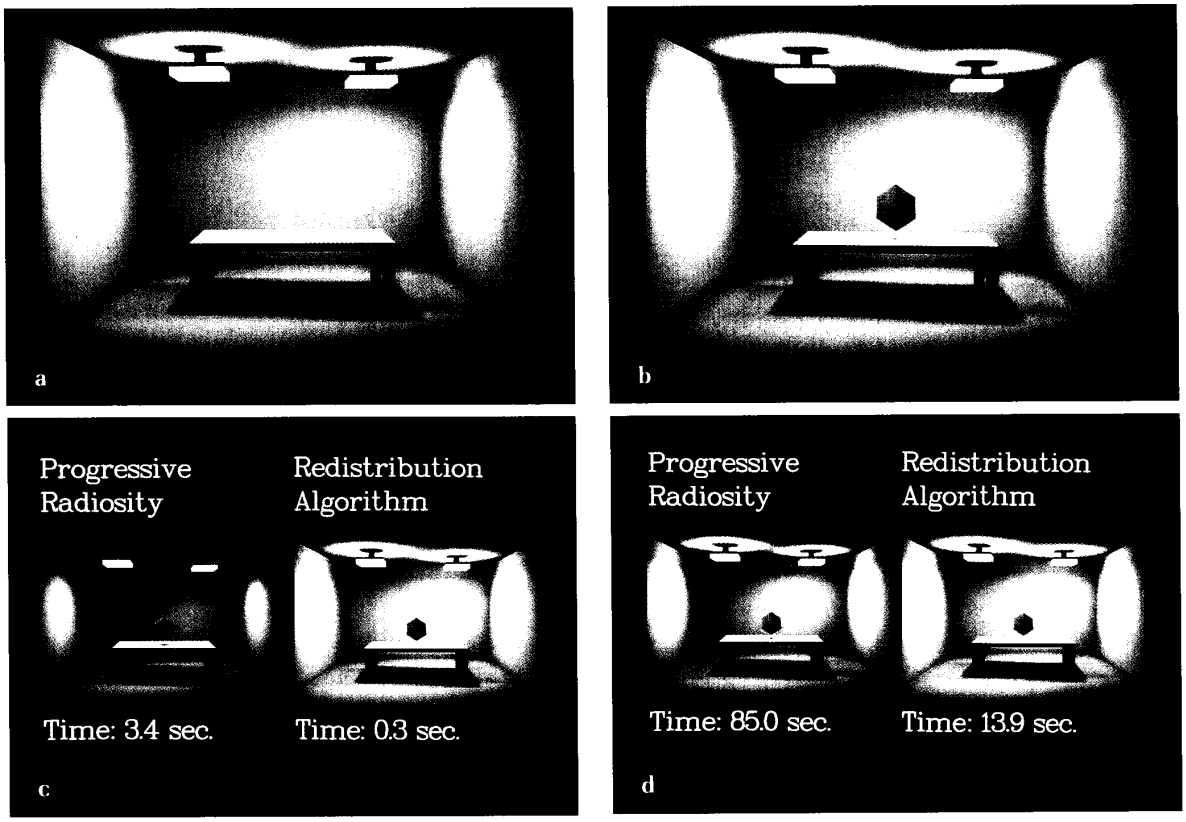


Figure 6. Comparison of the progressive-refinement and redistribution algorithms for the "Cube" scene (450 patches, 1,750 sample points). (a) Converged without cube, (b) Converged with cube, (c) Comparison after two iterations, (d) Comparison after 50 iterations.

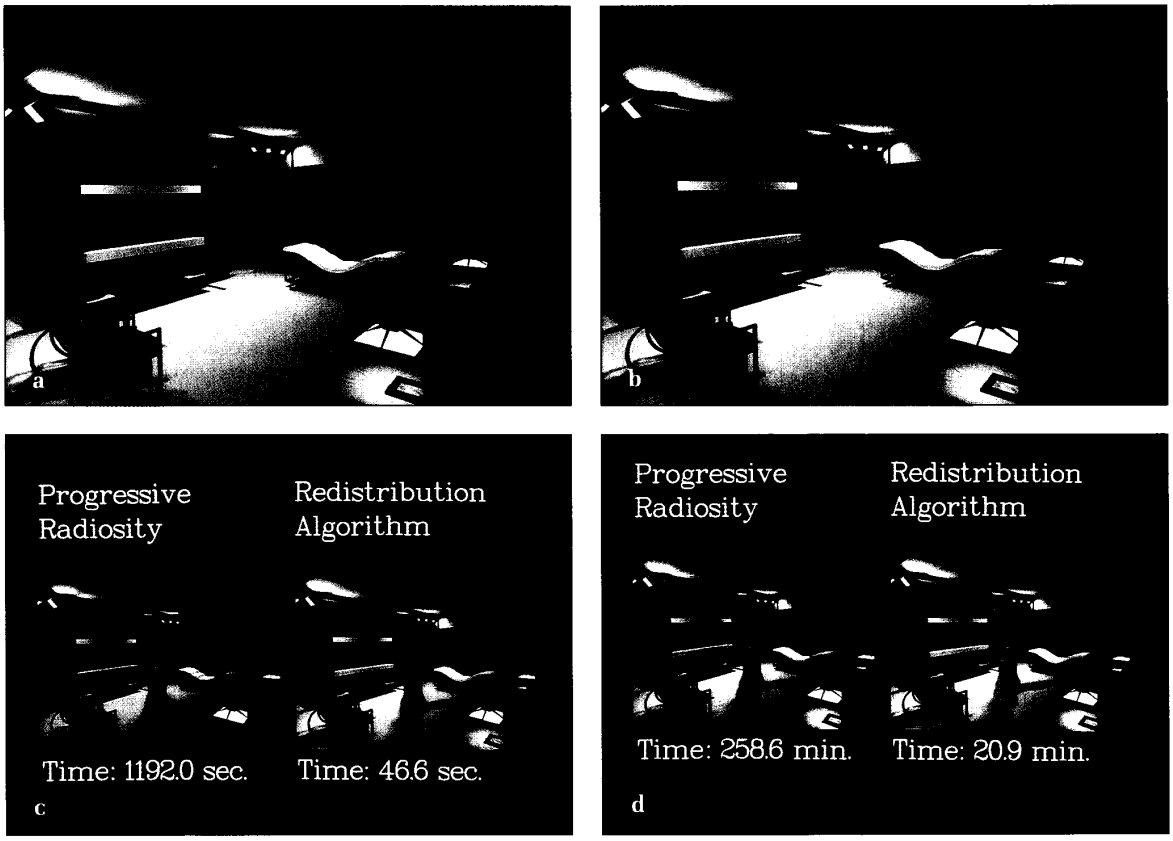


Figure 7. Comparison of the progressive refinement and redistribution algorithms for the "Robot" scene (12,700 patches, 45,900 sample points). (a) Converged without robot, (b) Converged with robot, (c) Comparison after five iterations, (d) Comparison after 100 iterations.

radiosity values to the CIE XYZ color system, and we find the difference in luminance (the Y coordinate) between the two. To compute the error term, the root mean square of the area-weighted luminance differences is computed.

$$Error = \sqrt{\frac{\sum_{i=1}^{n} ((y_i^{ref} - y_i^{cur})^2 A_i)}{\sum_{i=1}^{n} ((y_i^{ref})^2 A_i)}} \quad (7)$$

This error term is not a good measure of the perceived change to the environment when a new object is added. The addition of a new object causes the radiosity of a few patches to change very much, but most patches are barely affected, so the perceived difference is much larger than the error term indicates. Therefore, an alternative error term was also computed using only the 1 percent of the patches whose radiosities change the most between the previous converged solution (without the new object) and the new converged solution (with the object). In the "Cube" environment (see Figure 6), for example, the patches changing the most are on the cube and in the shadows cast on the table.

In Figures 6 and 7, we illustrate a comparison of the redistribution algorithm and the progressive-refinement radiosity algorithm for simple and complex changing environments. In both cases, we present fully converged reference solutions with and without a dynamic object. We then compared the two algorithms at different stages in the computation after adding the object to the scene.

Conclusion and future research

We have extended the progressive-refinement radiosity method to environments that change over time. The new redistribution algorithm uses progressive refinement and object-space coherence to update the existing radiosity solution when a change in the environment takes place. For small and moderately

sized environments, a good approximation to the new radiosity solution can be provided at interactive rates.

As the complexity of the environments increases, however, updates to the radiosity solution become too slow for use in interactive applications. Although faster processors will be available in the future, the complexity of environments will continue to grow as well. Future research should concentrate on the parallelization of this algorithm.

Temporal coherence should be used to reduce computation. For example, patches that remain completely occluded from a light source by moving objects for several frames should retain redistribution energy from one frame to the next.

These topics are currently under investigation. However, we believe that the use of object coherence and the concept of shooting positive and negative energy within the context of a progressive-refinement solution will ultimately be the correct strategy for providing interactive global-illumination simulation for dynamic environments. ■

Acknowledgments

This research was conducted at Cornell University's Program of Computer Graphics under a National Science Foundation grant (#DCR-8203979) entitled "Interactive Input and Display Techniques." The authors gratefully acknowledge the generous equipment grant from Hewlett Packard, on whose workstations this research was conducted. The Cube Room was modeled by Rod Recker, and the Robot Room was created by Wendy Burgess and Keith Howie. The photography was done by Emil Ghinger. Thanks to Roy Hall for his useful comments on this article and to CUPCG students and staff who helped with our research.

References

1. A.S. Glassner, "Spacetime Ray Tracing for Animation," CG&A, Vol. 8, No. 2, Mar. 1988, pp.60-70.
2. D.R. Baum et al., "The Back-Buffer Algorithm: An Extension of the Radiosity Method to Dynamic Environments," *Visual Computer*, Vol. 2, No. 5, Sept. 1986, pp. 298-308.
3. C.M. Goral et al., "Modeling the Interaction of Light Between Diffuse Surfaces," *Computer Graphics* (Proc. SIGGRAPH), Vol. 18, No. 3, July, 1984, pp. 213-222.
4. M.F. Cohen et al., "A Progressive Refinement Approach to Fast Radiosity Image Generation," *Computer Graphics* (Proc. SIGGRAPH), Vol. 22, No. 3, Aug. 1988, pp.75-84.
5. D.R. Baum, H.E. Rushmeier, and J.M. Winget, "Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors," *Computer Graphics* (Proc. SIGGRAPH), Vol. 23, No. 3, Aug. 1989, pp. 325-334.
6. M.F. Cohen et al., "An Efficient Radiosity Approach for Realistic Image Synthesis," *CG&A*, Vol. 6, No. 3, Mar. 1986, pp. 26-35.
7. J.R. Wallace, K.A. Elmquist, and E.A. Haines, "A Ray Tracing Algorithm for Progressive Radiosity," *Computer Graphics* (Proc. SIGGRAPH), Vol. 23, No. 3, Aug. 1989, pp. 315-324.
8. C. Puech, F. Sillion, and C. Vedel, "Improving Interaction with Radiosity-Based Lighting Simulation Programs," *Proc. Symp. Interactive 3D Graphics*, Mar. 1990, ACM, N.Y., pp. 51-57.
9. T. Nishita and E. Nakamae, "Continuous Tone Representation of Three-Dimensional Objects Taking Account of Shadows and Interreflections," *Computer Graphics* (Proc. SIGGRAPH), Vol. 19, No. 3, July 1985, pp. 23-30.

David W. George is a graduate student in the Program of Computer Graphics at Cornell University. His research interests include realistic image synthesis and parallel programming.

George received a BS in math and computer science from Cornell University and expects his MS in computer graphics in August 1990.

François X. Sillion is a post-doctoral associate in the Program of Computer Graphics at Cornell University. His research interests include light-reflection modeling, global-illumination algorithms, interaction, and evaluation of image quality.

As a student of the Ecole Normale Superieure in Paris, Sillion received an undergraduate education in math and physics, a graduate degree in Solid State Physics in 1986, and a PhD in computer science in 1989.

Donald P. Greenberg is director of the Program of Computer Graphics and the originator and former director of the Computer Aided Design Instructional Facility at Cornell University. During the last 15 years he has been primarily concerned with research advancing the state of the art in computer graphics and the use of these techniques as they may be applied to a variety of disciplines. His specialties include hidden-surface algorithms, geometric modeling, color science, and realistic image generation.

In 1987 Greenberg received the ACM SIGGRAPH Steven A. Coons Award, and in 1989 he received the NCGA Academic Award. He is on the editorial boards of *Computers and Graphics* and *Computer-Aided Design*, and he is a member of ACM SIGGRAPH and IEEE.

Greenberg received his PhD from Cornell in 1968.

The authors can be contacted at the Program of Computer Graphics, 120 Rand Hall, Cornell Univ., Ithaca, NY 14853.