Hierarchical Instantiation for Radiosity

Cyril Soler
Max Planck Institut für Informatik

François Sillion
iMAGIS-GRAVIR/IMAG-INRIA

Abstract.
We present the concept of hierarchical instantiation for radiosity. This new method enables an efficient, yet accurate determination of the illumination in very large scenes, where similar objects are replaced by instances of the same element. Instances are equipped with suitable radiative properties and are used to replace large amounts of geometry at multiple levels of the scene hierarchy. In essence, our algorithm replaces a single very large hierarchical radiosity problem by a collection of hierarchical radiosity problems within small sets of objects at a time, at several hierarchical levels. We prove the applicability of our method on architectural scenes with replicated geometry. However we reach the best time and memory gains on plant models thanks to the high degree of self-similarity in such kinds of scenes. This allows us to compute lighting simulations on scenes including a very large number of polygons in a short time on machines with limited memory.

1 Introduction

In scenes that include a large number of input geometric primitives, hierarchical radiosity algorithms are rapidly limited by their memory and computation costs.

We propose a new radiosity algorithm based on the concept of hierarchical instantiation, which allows this calculation to be performed while dramatically reducing the resources needed as compared to previous approaches. The hierarchical instantiation idea takes advantage of the inherent repetition and structure in a scene, while allowing full differentiation in terms of lighting conditions.

The notion of instantiation refers to the replacement of similar parts of the complete geometrical model by copies of a single geometry. This is commonly used in ray tracing algorithms, where only a geometric transformation needs to be stored with each instance, all ray intersections being computed against the unique geometrical model common to all instances. In a more general approach, the approximate self-similarity in large models lets us use instances to replace parts of the geometry that are largely, although not exactly, identical. At first sight, instantiation does not seem really suitable for radiosity calculations because geometrically similar objects may receive totally different illumination. Therefore our algorithm cleanly separates the representation of the light distribution from the geometry management. Thanks to a well-chosen computation mechanism and associated traversal of the hierarchical structure, only a fraction of the total geometry ever needs to be present in memory, making the algorithm practical even for very large models such as plants. Instantiation also appears to be a promising approach to radiosity simulations in very complex architectural scenes involving some repetition.

The remainder of the paper is organized as follows: we first review existing approaches for lighting simulation with instantiation. Section 3 presents the hierarchical instantiation algorithm in detail, and analyzes the expected benefits of the approach. In Section 4 we propose a complete set of solutions to the different problems arising in the radiosity simulation using hierarchical instantiation. Section 5 presents our results, including the validation of the hierarchical instantiation algorithm and the evaluation of its performance.

2 Previous work

The general principle of *instantiation* is to replace the geometric definition of several identical objects by pointers to a unique original object and the geometric transformation needed to map the original to the instanced object.

Ray-tracing instanced objects simply involves transforming the ray back to the coordinate system of the original object before computing the intersection. Complex models can be ray traced with little memory, since only the geometric representations of original objects must be kept in memory [SB87, KK86]. Applied to plants [Har92] or to fractal objects [BW94] the gain is even more interesting, because the self similarity of such models appears at different levels of their hierarchical representation.

When the objects to be instanced are not exactly identical to a given original object (as is the case for most scenes), approximate instantiation can be used to save memory while ensuring a similar visible result. This has been used to render images of giant outdoor scenes [DHL+98].

Instancing objects for radiosity simulation is a much more difficult problem, because radiosity operates on a representation of illumination traditionally attached to geometry (typically in the form of a mesh). Ouhyoung [OCL96] defined the concept of *re-usable radiosity objects* to share precomputed geometric information between instances of a given object, in order to accelerate the computation of visibility and of the form factors involving these objects. However, the representation of radiosity inside different instances of the same object can not be shared, because it not only depends on internal characteristics of the object but also on its position in the scene. Besides, when dealing with approximate instantiation the geometry of the original can not simply be used as a support geometry for the instances. Therefore this method requires the complete model of all similar objects to be present in memory.

An exception to this rule exists, however, for the case where the radiosity of the instance is not that much influenced by its position in the scene, and it can therefore be considered constant. This is the case for the *canned light sources* objects [HKSS98], for which an inside radiosity solution is precomputed and stored with the model. Such an object can be further instanced and used in any radiosity solution.

3 Hierarchical Instantiation Radiosity

3.1 Using instances in radiosity

Radiosity algorithms compute an explicit representation of illumination, typically associated with the geometry in the form of a mesh. Copies of a given object each have their own, unique illumination. Instantiation for radiosity is therefore more elaborate than for simpler rendering techniques, since it should differentiate between the geometry (that is easily shared) and the illumination (that varies from one instance to another).

Hierarchical radiosity algorithms, especially those using clustering, try to avoid considering the inherent complexity of energy exchanges by computing transfers at fairly high levels of a scene hierarchy. However a complete traversal of the scene is needed to estimate the energy emitted or received by a cluster with any accuracy [SAG94]. The use of meta-objects, or impostors, has been proposed to avoid this descent in the hierarchy, and instead perform the computation with fairly large (and simple) objects [RPV93, OCL96].

This is a very good idea, resulting in important savings, but in order to guarantee a sufficient accuracy these meta-objects should be equipped with very precise radiometric information: they should respond appropriately (i.e. just like their actual contents would do) to incident light, and allow accurate determination of light transmitted through their contents.

An instance should therefore be able to participate in radiosity calculations without accessing its geometric content. In general, for each instance, this requires the knowledge of (a) an outgoing radiance distribution [SDS95], (b) a bidirectional scattering phase function to convert incoming energy into outgoing radiance, and (c) a transmittance function. The phase function of an instanciable object can be sampled by placing an external light source for a number of incoming directions and recording the outgoing radiance after solving for a global energy balance inside its geometry. To compute the outgoing distribution of light of an instance due to incoming energy along a link, we multiply its phase function for the incident direction by the irradiance along the link. For that reason, the phase function can also be called the *reflectance* function of the instance, as an extension to its usual definition on simple surfaces.

Obviously, these characteristics are quite costly to handle, both in terms of computation time and in terms of storage. Meta-objects are therefore especially useful when a sufficient number of such identical objects are present in the scene, since all instances share the same intrinsic characteristics. Phase and transmittance functions are precomputed and stored, and can be accessed whenever needed, to instance a part of the geometry of the scene that corresponds to it.

In summary, instantiation is a key element in making the accurate characterization of simplified objects viable. It is realized by identifying objects (in fact, clusters) that have a similar behavior in terms of light emission, reflection and transmission. Note that the radiometric behavior of an instance can therefore be an approximation of reality, just like its geometry is approximately that of the original. A flexible trade-off is possible between the accuracy of the representation and its compactness, largely controlled by the degree of auto-similarity in the model.

After a radiosity solution is obtained at the level of the instances, the illumination of objects *inside* the instances must still be determined. This involves a local hierarchical radiosity solution in which the contained geometry is subjected to the incident illumination computed for the considered instance. A major potential difficulty is that the contents of the instance might still be too complex to allow a hierarchical radiosity calculation. The hierarchical instantiation algorithm described below provides an elegant and efficient solution to this problem.

3.2 Hierarchical Instantiation

Our aim is to ensure that hierarchical radiosity calculations are always performed on a fairly small scene. To this end, we explore the possibility of creating instances at various hierarchical levels. This is possible whenever different scales of similarity exist in the model. (Plant models are the perfect example of such a property; we will therefore illustrate the idea with the case of plants in the results section).

Structure of the hierarchy. We want to reveal the redundancy present at different scales of the scene hierarchy. This information can easily be extracted from the architecture of plants for instance and we anticipate that it will be possible to obtain it in large-scale architectural scenes as well. If the cluster hierarchy made from *instantiable* clusters still has a large branching factor its efficiency toward hierarchical radiosity must be improved by inserting new levels of (non instantiable) clusters into the hierarchy. This can be done using a constrained clusterizer [HDSD99].

As a result, the entire scene can be described as a hierarchy of clusters, in which instantiable clusters appear at various levels (possibly one included in the other). However, during any call to the computation of a *local* solution using hierarchical radiosity, the part of the hierarchy that is considered always consists of a cluster hierarchy whose leaves are either non opened instances or polygons.

Hierarchical solution. The scene hierarchy is first loaded into memory with a depth limited to the first level of instances. Then it is processed by the hierarchical radiosity solver, which involves iteratively establishing (refining) links between clusters and propagating energy until convergence. Refinement of the links is limited to the level of instances, since their geometry is not available at this time. However the resulting solution is still much more accurate than if we had performed a hierarchical radiosity solution on the entire scene while limiting the link refinement to the level of the instances. This is due to the fairly precise representation of each instance's "phase function" or general reflectance property, which is precomputed and embodies the effect of light propagation and scattering inside the instance. In addition, it should be noted that no self-links are established on instances, because their reflectance function already accounts for internal light scattering.

We then traverse the scene hierarchy, and focus on each instance encountered in the following manner: (a) form in a new hierarchy by loading down to the next instantiation level the geometry contained in the instance, (b) compute a local radiosity solution inside the new hierarchy (with a recursive application of the algorithm) and (c) destroy the new hierarchy and replace it back with the instance. Therefore we essentially perform a depth-first traversal of the scene, always focusing on a given hierarchy of instances.

When we reach a level with no instances below, the algorithm is equivalent to hierarchical radiosity with clustering, and a complete solution is available for the current branch of the scene hierarchy, taking into account contributions from the entire scene.

The solution for the current portion of the hierarchy is accessible at this stage only, because its supporting geometry will be deleted when closing the instance. We thus render the corresponding polygons into an off-screen buffer (or output the results to a file), thereby progressively forming the image during the traversal of the scene.

Opening instances. We detail here the operations involved in the "local" solution computed when opening an instance in the recursive traversal. This process is illustrated in Figure 1. On the left we see a solution computed at a given level. Oval shapes represent objects or clusters, while rectangles represent instances. Links are indicated by arrows, and have been created at varying levels of the cluster hierarchy.

When the lower-right instance is opened, we build a hierarchy with its contents, as shown on the right-hand side of the figure. In order to properly account for all incoming light, we create copies of all links that previously arrived on the instance (marked using dashed lines on the figure) and attach them to the root of the new hierarchy. We also add a self-link to the root if no self link exist on any parent levels, to account for all internal exchanges [Sil95].

We can then apply the solution procedure outlined above, that is first solve for radiosity, then traverse the hierarchy to open instances and recurse. The right side of Figure 1 illustrates the radiosity solution, in the opened level: Dashed links correspond to links that previously arrived at the instance level, and have been refined. Internal links issued from the refinement of the added self link are also represented. The recursion would then continue into the smaller instances before returning to the left-hand situation and opening the other instance.

Note that refinement is constrained in that only elements belonging to the considered hierarchy may be subdivided (either as emitters or receivers). Gathering and push/pull operations are also applied to the local hierarchy only, essentially treating all elements external to this hierarchy as fixed light sources.

Discussion. Our algorithm essentially gains by neglecting the correlation between objects lying in different instances at the same hierarchical level. For two such sibling instances, no link can ever be created between one object from each, because the contents of both instances are never simultaneously present in memory. This ensures that every local solution only involves a small number of objects, at the expense of a small approximation. For the

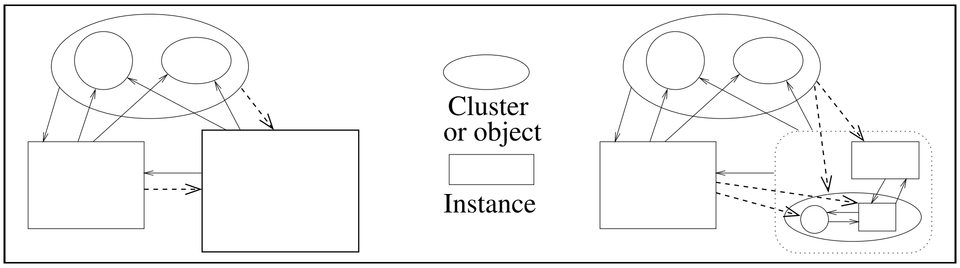


Fig. 1. Two consecutive stages of the recursive computation of the radiosity solution in the hierarchy of instances.

same reason, a complete solution is never present in memory, although every part of the global solution is available at some stage of the calculation. This explains why any results such as images or radiosity values written to a file must be output during the calculation as mentioned earlier.

The same result could be achieved in a normal radiosity algorithm, by preventing the refinement of an emitter if it is an "instantiable" object different from the receiver. However the global accuracy would be lower unless the emitter is already refined enough to obtain a high-quality representation of its internal light distribution. Since the phase functions of the original instances are pre-computed and stored, more computation time can be invested in this process than typically done in a hierarchical radiosity computation. For instance the effects of internal visibility in emitting clusters, which are usually not computed for cost reasons [SD95], are accounted for in our general reflectance functions (See Section 4).

The pseudo-code on Figure 2 summarizes the algorithm.

```
main()
    OpenOutputFile()
    OpenInstance(root)
    CloseOutputFile()

HierarchicalInstantiation(cluster H)
    if IsAnInstance(H)
        OpenInstance(H)
    if IsACluster(H)
        ForAllChildren(g,G)
            HierarchicalInstantiation(g)

    if IsAPolygon(H)
        Render(H)

OpenInstance(instance H)
    cluster G = LoadNextLevel(H)
    ReplaceInHierarchy(H,G)
    TransferLinks(H,G)

    ComputeLocalSolution(G)

    HierarchicalInstantiation(G)

    DeleteLinks(G)
    ReplaceInHierarchy(G,H)
    Delete G
```

Fig. 2. Pseudo code for the radiosity instantiation algorithm. The generic *Render* procedure replaces any output of the information as rendering the polygon to an off-screen buffer, or saving its radiosity to a file.

3.3 Cost considerations

Simple recursion arguments allow us to evaluate the cost of our algorithm in terms of storage and computation cost. Let us denote the number of instantiation levels by k, the number of elements at each instantiation level by N, and the number of these elements that are instances by p. This model is very simple because it assumes a uniform branching factor among all levels of the hierarchy of instances, and a uniform proportion of instances and polygons at each level of the scene hierarchy. Using these characteristics, the total number of polygons in the scene is :

$$n = (N - p)(1 + p + ... + p^{k-1}) + p^k N = O(p^k N) \tag{1}$$

Gain in memory. Let us denote the memory size of a polygon, an instance and an original (instanced) object by ε, I and o.

Assuming r original objects are used to create the p instances, the memory footprint of the scene at the top level of calculation is :

$$M_{inst}(1) = (N - p)\varepsilon + pI + ro$$

Since the algorithm only loads the geometry of the branch of the hierarchy it is descending into, the maximum memory requirement is reached at the bottom of the hierarchy, where it is :

$$M_{inst}(k) = k(N - p)\varepsilon + kpI + kro \tag{2}$$

For the ideal case of a well balanced hierarchy of instances, the memory cost is thus logarithmic in terms of the total number of polygons in the scene. In any case, it is much less than the $O(n\varepsilon)$ memory size of the model itself.

As an example taken from our implementation and real data, consider $o = 1,100$ bytes (an original object holds two sampled directional functions at 528 bytes each), $\varepsilon = 150$ bytes (this rather large size accounts for geometry, radiometric and subdivision information) and $I = 200$ bytes (in our implementation, instances are also clusters and thus contain inherited information). For the tree presented in Figure 5, we have $n = 119,000$, $k = 4$, $r = 5$, $N \approx 30$ and $p \approx 8$. The expected memory size given by Expression 2 is 48 kb, which is much less than $n\,\varepsilon = 15,085$Kb, the expected size of the entire scene.

Although these numbers do not translate directly into required memory sizes, because of the missing constants and various fixed costs, we will see in Section 5 that a large memory reduction is observed, the gain increasing with scene complexity. It actually becomes feasible to simulate very large scenes that simply could not be treated by previous methods.

Since the accuracy threshold does not change when recursively computing the local solutions, the maximum number of links in memory can be estimated by the number of links that contribute to the illumination of a leaf element in a classical hierarchical radiosity solution on the entire scene, multiplied by the number of leaf elements at the lowest level, *e.g* $O(N \log n)$. This is much less than the $O(n \log n)$ links of the normal clustering radiosity method.

In scenes with limited instantiation depth (*e.g* k is small) the logarithmic equivalent does not hold anymore. In the worst case, the gain in memory is the number of instances times the ratio between the memory cost of an instance and the actual geometry.

Computation cost. We consider that a hierarchical radiosity solution in a scene of n elements equipped with a well balanced hierarchy can be performed in $O(n \log n)$ time.

Let $C(i)$ denote the cost of our algorithm for solving level i, then we get :

$$C(i) = N \log(N) + pC(i-1) \qquad \text{and} \qquad C(1) = N \log(N)$$

The cost is thus :

$$\begin{aligned} C(k) &= N \log(N) \left(1 + p + ... + p^k\right) \\ &= O(p^k N \log(N)) \end{aligned}$$

Considering that $n = O(p^k N)$, the value $C(k)$ appears to be equivalent to $O(n\log(N))$, which is very close to the cost of the classical hierarchical radiosity algorithm. However, our algorithm is faster in practice, as will be discussed in the results section.

4 Implementation

Precomputation and use of instance information. In the general case, we use a directional distribution to represent the transmittance information of an instance. We precompute it using graphics hardware, by rendering the concerned object off-screen in each direction sample and recording the percentage of the bounding box of the object that is not masked by the object itself. In some cases, the transmittance information can preferably be accessed through a simplified geometric representation of the real geometry, especially if the instances replace very simple objects. This representation can still be approximative since we allow the instantiation of not necessarily identical objects.

We compute the phase function by performing a series of accurate lighting simulations for directional incoming illuminations, and recording the distribution of outgoing light. Each lighting simulation is done using hierarchical radiosity. We record the outgoing light by rendering the illuminated object in an off-screen buffer and measuring the average values of pixels in the resulting image, so as to obtain an average radiance value per unit of projected area. One advantage of this technique is to account for self-occlusion for the outgoing light as well as for the incoming light in the objects. However, if instances are used to replace very complex geometry this approach can be costly. In such a case, a recursive application of the Hierarchical Instantiation algorithm with limited depth could be used to improve the computation time of the phase functions. An other possibility is to use generic phase functions, which works well for plant models.

To compute the contribution of an emitting instance to the radiosity of an element in a given direction θ, we then multiply its emmited radiance by the projected area $a(\theta)$ of its geometry in the direction of the receiving element. This projected area is easily deduced from the value of the directional transmittance $\tau(\theta)$, and the projected area $A(\theta)$ of the instance's bounding box:

$$a(\theta) = (1 - \tau(\theta))A(\theta)$$

Visibility computation. Computing visibility using exact geometric information is only viable for clusters that contain a small number of polygons. Therefore the geometric information of simple objects can be kept with the original cluster, for use by all the instances. At higher levels, a formulation based on extinction coefficients [Sil95] works well for isotropic clusters, like the ones we find on plant models. For all other cases, a more complex representation of visibility can be needed depending on the geometric configuration of the energy transfer it is involved in. This can be achieved using a simplified geometry. Indeed, even a highly decimated mesh produces much more realistic shadows than a box equipped with directional transmittance.

When computing the visibility through a blocker that is not an instance, the visibility is recursively computed down to next instances or polygons.

Instantiation policy. The basic principle of instantiation requires that a sufficient number of copies of an instantiable cluster exist in the scene to counterbalance the cost of keeping general reflectance functions of original clusters in the scene. It should be noticed that exact geometric similarity between objects instanced by the same original is not necessarily needed: it is sufficient that they only have similar radiometric properties at the level of the hierarchical radiosity solution they are involved in. After being opened, the instances are automatically replaced by more appropriate geometry. This allows us to instance branches of a plant for instance, whose geometry would differ in details but that still would have the same global shape. In general, a more complete approach would consist in precomputing and storing the necessary information to obtain a bound on the error for energy exchanges that involve approximate instances, and use it to decide whether to use a different but more appropriate original cluster to reduce the error.

Instancing at all levels of the hierarchy is not necessarily a good idea: In some cases in which clusters are too close to each other, replacing both of them by instances would suppress any interaction between pairs of sub-elements in these clusters. This is a common cause of error in hierarchical radiosity algorithms. Besides, clusters that lay too close to each others tend to be associated to poorly approximated form factors, which can cause divergence of the algorithm when a number of them sum up to too large a value. To correct both problems, our implementation allows to skip levels of instantiation to increase accuracy, at the expense of computation time and additional memory cost.

In our first implementation, this strategy is entirely user-defined, which is not acceptable if we want to be able to treat complex scenes automatically. It would also be a good idea to have the instantiation policy depend on the memory currently available by caching instance memory: the program could be allowed to skip instantiation levels whenever needed provided that sufficient memory is available.

5 Results

5.1 Application to classical architectural scenes

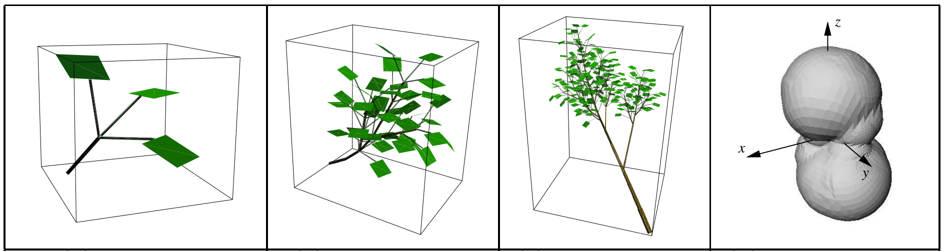
In Figure 6 (*See appendix*) we present a solution on a "classical radiosity scene" of $55,000$ polygons, in which all objects have been instanced (*e.g* 24 chairs, 3 tables, 3 orchids and 4 plants. Computation times for each of the four phase functions took between 1mn and 10mn). To be able to generate correct shadows for the instances the transmittance function is computed using a unique and shared copy of the geometry of each object. Note that shadows of external geometry (including instances) on instances themselves are also correctly generated (consider for instance the shadows on tables cast by the orchids).

The solution itself is computed in 8mn using 15MB of memory, whereas the computation time and memory cost of hierarchical radiosity is 45mn/67MB. However, it could be possible to improve the gain in this particular scene, using a higher depth of instantiation on the two plant models but we did not have access to their botanical structure. This raises the question of automatically finding potential instances in a scene.

5.2 Application to plant models

Hierarchical radiosity without clustering is of little help in the case of plants, mainly because the polygons that define the models are always very small, thus raising the cost of initial linking. Radiosity methods with clustering [SAG94, Chr95, Sil95] seem to be very promising since they have a much smaller computation cost, but they still require considering the entire model simultaneously in order to establish links between clusters, which becomes unpracticable for very large plants. Other calculation algorithms based on statistical or continuous approximations have been proposed [Gre89, MMKW97, Gov95], but are ill-suited to simulations at the scale of a few plants, which is more typical in virtual reality and computer graphics applications.

Conversely, our approach works very well because of the high degree of self similarity inside the plant models, and because instances are able to share properties such as phase functions. We observe indeed that phase functions for a given kind of axis (branch, whole plant) tend to reach a "limit" as its age increases. This allows us to instantiate older axes using the precomputed information of medium aged ones. Otherwise, precomputing the phase function of very complex objects (like old trees) using a brute force approach can be even longer than computing the solution for the scene itself. Finally, directional transmittance functions work pretty well because the transmittance of such kind objects is, to a large extent, uniform at the scale of the energy transfers.



(a) Small axis	(b) Small branch	(c) Large branch	(d) phase function

Fig. 3. (a) to (c) : Axes corresponding to three different levels of possible instantiation. Each level also corresponds to a cluster in the hierarchy that defines the tree. (d) represents the reflectance function for the axis in (a). The two main bulbs come from the shape of the bounding box and the smaller ones from the common orientation of the polygons in the cluster. See text also.

We import our scenes as a depth-first description of the cluster hierarchy and the geometry. In the case of plants, each cluster (before constraining the hierarchy) corresponds to a plant axis in the architectural sense [dREF$^+$88], and contains information about the kind of axis and its age. We use this information as a criterion of similarity between clusters that can be instantiated. The clusters in Figure 3 represent different levels of instances used for the images of Figures 5 and 7 (*See appendix*). With each of these clusters is also stored its position and orientation in space. Whenever it is deemed possible to instantiate a cluster, its geometric definition is skipped and replaced by an instance with the correct geometrical transformation.

In our current implementation, phase functions only account for incoming light direction in the instances, the outgoing radiance per unit of projected area is assumed to be uniform. This allows us to store only radiosity values on instances instead of complete directional radiance distributions. While incurring some approximations, this allowed us to test our algorithm in a first approach, and does not depend on any limitations of the algorithm itself. Such a phase function is represented in Figure 3d that corresponds to the cluster in Figure 3a.

Finally, leaves are modeled by two-sided polygons with diffuse transparency. The radiosity on one side is computed as the sum of two terms: (1) the irradiance on the same side multiplied by its reflectance, and (2) the irradiance on the other side multiplied by a diffuse transmittance coefficient.

5.3 Comparison with hierarchical radiosity

In Figure 5 (*See appendix*) we compare the results given by our radiosity algorithm to that of a hierarchical radiosity simulation with clustering, performed with the same set of parameters but without instantiation. Three light sources are used (the 3rd one is masked by the tree). The model of the tree is a 30-year old poplar tree (*populus*) consisting of 119 000 polygons. Experiments have been conducted on a *SGI Origin2000* computer.

Although the results seem identical at first sight, some subtle differences can be found, that mainly concern a variation of the energy in some parts of the plant where instantiation has been used. This is due to the fact that the radiosity stored in the instances is not directional but represented as a single value. On the *right* image for instance, the light arriving from the light source on the center of the tree has been distributed behind the instance because of this approximation. As expected by our estimations, the gain is very important (15 times less memory in this particular case), especially considering that there are fixed memory costs (Our radiosity program requires a fixed amount of 5MB of RAM).

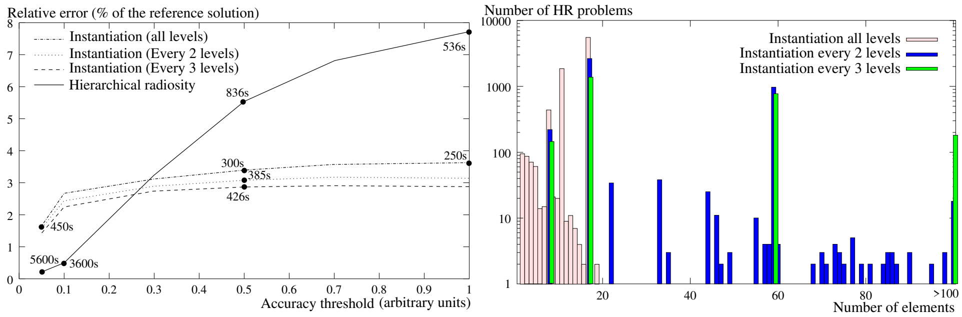


Fig. 4. Left: *L^2 error in percentage of the maximum value of the reference solution (computed using hierarchical radiosity) for various values of the accuracy threshold used in link refinement. The numbers in black on the curves indicate computation times in seconds.* Right: *histogram of the number of hierarchical radiosity solutions on local hierarchies with variable numbers of leaf elements. When instancing at fewer levels, hierarchical radiosity computations tend to involve more elements.*

An important gain in computation time is also apparent (Hierarchical Instantiation is 8 times faster). Since our algorithm does not allow bidirectional refinement of energy links between instances, it refines fewer links and computes fewer form factors. The ratio between the two numbers of links is the average number of elements an emitter is subdivided into. This of course depends on the relative positions of the elements and on the refinement algorithm. If the instances are not too close to each other, we can consider that our algorithm is equivalent to normal refinement; for really close instances it is more approximate. Visibility calculations are also faster with instances, since they do not require geometric operations, instead using the stored mean transmittance value in the relevant directions.

On the left side of Figure 4 we show a comparison of the accuracy of hierarchical radiosity and hierarchical instantiation for values of the accuracy threshold used in the refinement of links, for the scene of Figure 5. It appears that for larger values of the error threshold, hierarchical instantiation is much faster then hierarchical radiosity with a better accuracy. This confirms that using phase functions that accurately account for the internal scattering of light is more efficient than the traditional approximation used for clusters in hierarchical radiosity. For small error thresholds hierarchical radiosity is still more accurate than hierarchical instantiation. We believe that this is caused by the omni-directional approximation of the reflectance functions in our current implementation, which is confirmed by the fact that instancing fewer levels in the hierarchy only marginally increases the accuracy. The repartition of the hierarchical radiosity solutions required by our algorithm corresponding to each case is shown on the right side of Figure 4. It clearly appears that when instancing all levels, only hierarchical radiosity problems with small number of elements occur. When instancing every other level, the average number of elements per hierarchical radiosity problem increases. Finally, when instancing every three levels, the algorithm tends to mainly solve hierarchical radiosity problems with more than 100 leaf elements.

Figure 7 (*See appendix*) presents a lighting simulation in a scene of 1 210 925 input polygons, built using poplar trees of various ages[1]. The simulation took 9 hours and 53 minutes to compute on a *Sgi-Origin 2000* computer, using only 49Mb of memory. When comparing this computation time with the one of the solution on Figure 5, it should be noticed that the source we used here is much more complex (Each bulb contains an average of 1 000 small polygons grouped into clusters) and thus drastically increases the number of links in the scene. Computation time with a single polygon source is less than 2 hours.

[1]Models generated by the AMAPHydro software. Courtesy of Pr.Philippe de Reffye and F.Blaise, CIRAD

6 Conclusions and future work

We have proposed a new hierarchical instancing technique that drastically reduces the amount of resources needed for the lighting simulation with hierarchical radiosity in scenes with approximate geometrical redundancy. This enables for the first time the precise simulation of light in very large models, such as entire trees, with limited memory. This instantiation technique essentially bridges the gap between explicit radiosity solutions and view-dependent approaches operating on instanced geometry, allowing a precise calculation in object space with reasonable resources.

Whereas in the case of plants it appears that the phase functions from large models can be approximated by the ones for small models, (*e.g* by the phase functions from instances at lower levels in the hierarchy) this is not true in architectural scenes. Some investigation is required to search for a way to deduce phase functions of large instances without computing a hierarchical radiosity in the entire model. We are thinking about considering an incremental approach, where phase functions are initially considered uniform, but are incrementally updated after opening the corresponding instance.

We have seen in the case of architectural scenes that directional transmittance functions do poorly in the computation of shadows cast by the instances and we used geometry based transmittance. Indeed, in the method we have presented, the instances are "opened" to improve the computation of their internal light distribution only. The regions where shadows are cast by an instance could also be considered as to be improved when opening this instance. Therefore, a formulation of our algorithm based on the accuracy of energy transfer along links that encounter an instance – rather than based on the accuracy of light on approximated geometry – would form a more general approach that could also permit to accurately compute shadows due to the geometry in the instances while keeping an approximate transmittance function within the instance.

The hierarchical instantiation algorithm could perhaps be improved by detecting instances with simple contents receiving a nearly uniform distribution of incoming light. In such cases, a simple hierarchical push/pull may replace the local radiosity solution at a fraction of the cost. Smarter and more generic instantiation policies could be used, especially as we start applying these ideas to scenes with less self-similarity. The vector quantization technique of [DHL$^+$98], based on the difference between phase functions, should prove particularly useful.

As could be observed in the results section, the compromise between storage requirements of the phase functions and the accuracy of the simulation leads to approximations in the solution for highly refined scenes. This choice was not due to a particular restriction of our algorithm but for the sake of simplicity for a first approach. Experiments could be conducted on bidirectional phase functions, and the kind of phase function to use (uniform, mono or bi-directional) could be adapted to the case of each object to be instanced. The precise distribution of illumination computed in our method could also be used to create high quality renderings, using a rendering step in which each leaf is subjected to the irradiance from links but uses a more accurate BRDF, including specular/shiny effects which are visually important despite their limited global impact.

Finally, it is clear that this work has a direct application on physiological plant growth simulation where the precise illumination on leaves is needed to compute the production of vegetal matter. We are currently working on such a project in collaboration with the LIAMA (French-Chinese Laboratory for Computer Science, Beijing, China) and the CIRAD (International Research Center in Agronomy and Development).

Acknowledgements. This work was supported in part by the INRIA Cooperative Research Project *SOLEIL* (INRIA-LIAMA-CIRAD). We wish to thank Philippe De Reffye and Frédéric Blaise (CIRAD) for their active participation. It was also supported by the European Union under Esprit LTR project #24944 ARCADE "making radiosity usable".

References

BW94. Ph. Bekaert and Y. D. Willems. Raytracing 3d linear graftals. *Winter School of Computer Graphics 1994*, January 1994. Held in held at University of West Bohemia, Plzen, Czech Republic, 19-20 January 1994.

Chr95. Per Henrik Christensen. *Hierarchical Techniques for Glossy Global Illumination*. Ph.D. thesis, Technical Report, Department of Computer Science and Engineering, University of Washington, Seattle, Washington, 1995.

DHL+98. Oliver Deussen, Patrick Hanrahan, Bernd Lintermann, Radomr Mech, Matt Pharr, and Przemyslaw Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. *Proceedings of SIGGRAPH 98*, pages 275–286, July 1998. ISBN 0-89791-999-8. Held in Orlando, Florida.

dREF+88. Ph. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. In *Computer Graphics(Proceedings of SIGGRAPH 88)*, volume 22(4), pages 151–158, 1988.

Gov95. Y. M. Govaerts. *A Model of Light Scattering in Three-Dimensional Plant Canopies: A Monte Carlo Ray Tracing Approach*. PhD thesis, Departement de Physique, Universitat Catholique de Louvain, Louvain, Belgium, 1995.

Gre89. Ned Greene. Voxel space automata: Modeling with stochastic growth processes in voxel space. *Computer Graphics (Proceedings of SIGGRAPH 89)*, 23(3):175–184, July 1989. Held in Boston, Massachusetts.

Har92. J. C. Hart. The object instancing paradigm for linear fractal modeling. In *Proc. of the Graphics Interface '92*, pages 224–231, Vancouver, Canada, 1992.

HDSD99. Jean-Marc Hasenfratz, Cyrille Damez, François Sillion, and George Drettakis. A practical analysis of clustering strategies for hierarchical radiosity. In *Computer Graphics Forum (Proc. Eurographics '99)*, volume 18(3), September 1999.

HKSS98. Wolfgang Heidrich, Jan Kautz, Philipp Slusallek, and Hans-Peter Seidel. Canned lightsources. *Eurographics Rendering Workshop 1998*, pages 293–300, June 1998. ISBN 3-211-83213-0. Held in Vienna, Austria.

KK86. Timothy L. Kay and James Kajiya. Ray tracing complex scenes. *Computer Graphics*, 20(4):269–278, August 1986.

MMKW97. Nelson Max, Curtis Mobley, Brett Keating, and En-Hua Wu. Plane-parallel radiance transport for global illumination in vegetation. In Julie Dorsey and Phillip Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 239–250, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.

OCL96. Ming Ouhyoung, Yung-Yu Chuang, and Rung-Huei Liang. Reusable radiosity objects. *Computer Graphics Forum*, 15(3):C347–C356, C483, September 1996.

RPV93. Holly Rushmeier, Charles Patterson, and Aravindan Veerasamy. Geometric simplification for indirect illumination calculations. In *Proceedings Graphics Interface '93*. Morgan Kaufmann publishers, 1993.

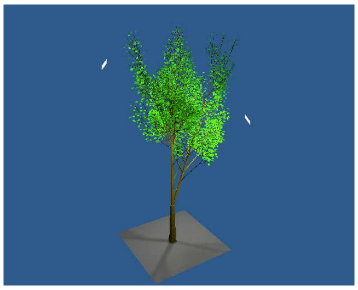
SAG94. Brian Smits, James Arvo, and Donald Greenberg. A Clustering Algorithm for Radiosity in Complex Environments. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 435–442, 1994.

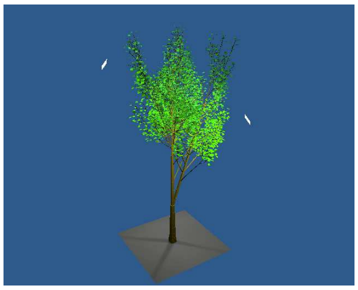
SB87. John M. Snyder and Alan H. Barr. Ray tracing complex models containing surface tessellations. *Computer Graphics*, 21(4):119–128, July 1987.

SD95. François Sillion and George Drettakis. Feature-Based Control of Visibility Error: A Multiresolution Clustering Algorithm for Global Illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (ACM SIGGRAPH '95 Proceedings)*, pages 145–152, 1995.

SDS95. F. Sillion, G. Drettakis, and C. Soler. A clustering algorithm for radiance calculation in general environments. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.

Sil95. François Sillion. A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3), September 1995.





H.Radiosity (**119**mn/**123**MB) H.Instantiation (**14**mn/**8**MB)

Fig. 5. *Comparison between hierarchical radiosity and hierarchical instantiation*

Fig. 6. *Application of our algorithm to a "classical radiosity scene".*

Fig. 7. *Hierarchical Instantiation on a scene of* $1M$ *input polygons.* **9**h**53**mn/**49**MB.