A Radiosity System for Real Time Photo-Realism

David Stuttard, Adam Worrall, Derek Paddon and Claire Willis

Kaleidoscope Computer Graphics Laboratory
Department of Computer Science
University of Bristol
Bristol, BS8 1TR, UK
derek@compsci.bristol.ac.uk

Abstract
The need to render changes in real time to scenes in a radiosity image is addressed by using parallel processing and distributing the tasks that originate from the scene changes amongst the processors. The mechanism for distributing these tasks is based on hashing the sample points that have been created to random processors. In this way it is ensured that the load balancing mechanism is optimised. In addition the global geometry changes are achieved by global broadcast of data. The combination of these mechanisms enables the system to generate new scenes by utilising most of the sample data from the previous radiosity image.

1 Introduction

One definition of the radiosity problem is as an attempt to generate sufficiently accurate (both numerically and visually) representations of the radiance functions that describe the surfaces. In this paper we will explore this definition and define a practical working system.

Most approaches used to solve the radiosity problem use a discretisation approach, in which the environment is divided into a finite set of patches. This approach was borrowed from thermal engineering and applied to diffuse reflection by Goral et al [Goral et al. 84]. The radiance function of each surface is approximated by the set of patches attributed to that surface. More recent methods have used polynomials [Zatz 93] and wavelet models [Gortler et al. 93] to represent the radiance functions over surfaces.

In many application areas of radiosity, such as, prediction and evaluation of lighting environments, the need for quick feedback is essential. Using radiosity for this type of problem is often unfeasible, even on state of the art hardware, since practical data sets are of great complexity. Although methods have been developed to deal with very large data sets [Teller et al. 94], the time taken to generate a solution is often too great to allow such solutions to be used as part of the design process.

We use distributed memory parallel processing to address both of these issues; this will allow very large data sets to be processed in an accelerated time scale [Chalmers et al 89, 91]. The radiosity system is based on CARM, a radiosity system that has been developed at Bristol.

1.1 Previous work

The actual methods for solving the physical radiosity equation are based on iterative techniques; the original method [Goral et al. 84] was an implementation of the Gauss Siedel method. It can be visualised as calculating how much energy each patch receives from every other patch. In matrix

terms, this requires a full solution of the entire matrix, with every possible interaction examined with equal attention. Progressive refinement [Cohen et al. 88], which is equivalent to Southwell Relaxation, can be seen as each patch shooting its energy out to the rest of the environment. This has the advantage that useful results can be generated quite early on in the solution, with most of the important energy transfers having been performed within the first few iterations. This is in contrast to the Gauss Siedel method, where usable results are generated only at the end of the solution. However, the Southwell Relaxation method has a slower convergence rate than the Gauss Siedel method; therefore, if the solution of the progressive refinement method is taken to full convergence then a result identical to that generated by the Gauss Siedel method will be obtained, but at the cost of more computation.

The most recent algorithm to be applied to radiosity is hierarchical radiosity [Hanharan 91], which was developed for the N-body problem. This method uses a rigorous error analysis to approximate form factors where such approximations are of the same order as other approximations made in the solution. The effect of this is to approximate whole blocks of the form factor matrix to just one form factor, so that far fewer form factors have to be computed. In fact, the algorithm runs in O(n) time (with n being the number of elements), as opposed to O(mn) (with m being the number of patches), as in other solutions. However, hierarchical radiosity requires an initial linking stage to determine which blocks of form factors can be approximated; this involves visibility calculations, and refinement of the patches into elements where necessary. This linking information, which is extensive, also has to be stored, which limits the size of the data sets that can be tackled. A final problem with hierarchical radiosity is that it works most efficiently when few initial patches are refined into a large number of elements; the technique is less effective for environments with complex initial geometry.

1.2 Overview of the CARM Radiosity System.

CARM uses a patch based system, with each patch holding a mesh of sample points. Although discretised systems have a number of shortcomings (such as aliasing) they can accurately represent very abrupt changes in the radiance function, which the analytical methods are not capable of.

The input geometry to the system is triangulated, allowing all object primitives and complex objects to be represented by meshes. The input triangles are then automatically tested for D0 discontinuities (i.e. tests for intersecting or abutting triangles) and for excessive size, and subdivided if required. The triangles left after such subdivision correspond to traditional hand-crafted radiosity patches.

Radiance values (or flux values) are sampled at various points, which are located at patch vertices, and also within the patch. These sample points are stored in a mesh, which can be triangulated and then interpolated to reconstruct the radiance function.

The system allows sample points to be added as the solution progresses, to allow the mesh to be refined; this is termed adaptive subdivision. When new sample points are added, then the mesh is re-triangulated; these triangles are analogous to traditional elements. Subdivision is controlled by examining the gradient and difference between neighbouring sample points; if either of these values exceeds a threshold, then a new sample point may be introduced. A final check is for the linearity of the radiance function between the neighbouring sample points; the function is sampled between them, and if the gradient is linear, then no new sample points are created since interpolation between the existing sample points is sufficient to model the function. Note that the new sample points are similarly checked before continuing with the rest of the scene.

Although there are some penalties in using the progressive refinement method on straightforward scenes, the method has unique advantages over both standard radiosity and hierarchical radiosity when volatile scenes are being manipulated. Unlike other methods which require global updates, progressive refinement allows small changes and movement of objects in a scene to be dealt with on a local basis, by only updating a subset of the sample point meshes in the scene. This local approach is physically incorrect, and will introduce errors into the radiosity solution. However, if the subset is carefully selected, then these errors will be no greater than those caused by not following the progressive refinement technique to convergence.

The method we use for form factor calculation is based on [Wallace et al. 89]. In this method, energy is shot from a source patch to receiving sample points, with the patch approximated to a disc, the vertex to a differential area, and the distance between the two assumed to be large compared to the size of the disc. This assumption is often broken, for example in corners, so analytical extensions are used for these cases.

Ray casting is used to determine visibility, with an adaptive technique to allow a trade-off for time versus accuracy. This operation has been accelerated with an octree data structure, which uses an algebraic method of traversal [Samet 89] as opposed to the standard approach [Glassner 84]; the Samet method climbs up the octree as far as necessary and then descends a mirror route to reach the neighbour, whereas the Glassner method descends from the top of the octree, selecting the most appropriate child until a leaf is reached. The Glassner method is prone to floating point error, and also slows down as the tree increases in size, whereas the Samet method is guaranteed correct, and has an average case performance independent of the size of the octree.

Discontinuity meshing for first and second order discontinuities [Lischinski et al. 92] is currently under development, since in the region of discontinuities this technique generates radiance function meshes of much greater accuracy and of much smaller size than those produced by the adaptive subdivision strategies.

2 Parallel model

The parallel model used in this system utilises a message passing architecture [Green et al 88] [Feda et al 91]. Each processor in the network has a small amount of memory and four serial links to other processors in the network. The processor (Inmos T800 Transputer) has hardware support for context switching allowing the use of several processes on one processor. This process model is usually known as *threading*. The system uses threads to support several tasks on one processor. This leads to two considerations to be made about the parallel network; intra-worker communication and topology and inter-worker communication and topology.

2.1 Intra-worker communication and topology

Each processor in the network is of exactly the same form. Several processes are executed concurrently on each processor. There are four main parts: the Work Distribution Manager (WDM), the Database Manager (DBM), the Worker Threads (WT) and the Worker Processes (WP). There can be any number of worker threads up to a pre-defined maximum. As the processor has hardware support for time slicing, these threads are almost free in terms of extra processing time. The use of threads means that several distinct tasks can be initiated at the same time, the processor then automatically switching to threads that are not waiting for data etc. This has the distinct advantage over explicit task control of very low overheads. However, allowing the processor exclusive control of task management using its inherent hardware context switching has several drawbacks. In the worst case, the threads can synchronise into a "request data, do work" rhythm giving a strobing effect. This problem is overcome by a combination of explicit task control and threads.

Tasks are initiated by the WDM along with a series of threads. The tasks are started at high priority and the threads at low priority. Only one task is running at any one time. When a task requests data for which it must wait (i.e. it is on another processor), it is explicitly de-scheduled. When no task is running, then the low priority threads are automatically time-sliced in to utilise any waiting time the processor may have. This combination of explicit task control and threads enables the highest utilisation of a processor.

The WDM has exclusive access to the rest of the network via the processor's external links. Any requests to the network in general for data must go via this process. This process is also responsible for the initiation of the Worker Threads and management of Work Processes. The DBM controls all the data for the node.

2.2 Inter-worker communication and topology

Inter-worker communication is a more difficult problem to solve. Bi-directional communication must be supported by the WDM of each processor. It is inevitable that at some point a processor will be sending data or task information to a processor which is at the same time sending a request. A traditional solution to this problem is to buffer the messages. This approach only succeeds in postponing the same problem until the buffers are full. This system enforces a strict protocol so that the deadlock situation cannot arise. A simple handshake protocol is used in the form of an *ack* message. This ack indicates to a processor that the buffer of the processor it is attempting to communicate with is empty. This alleviates the deadlock problem. The topology used in the system is a tertiary tree. The ack system is even simpler in this case as ack need only be sent to child processes. Any communication going down the tree need not use the handshake system, thus avoiding deadlock.

The system can cope with very large data sets by virtue of an off-line backing store in the form of a hard disk on the root node of the tree. Any data not contained in the network can be brought from backing store.

3 The radiosity algorithm

The use of a distributed memory environment causes problems with the progressive refinement algorithm. These are covered below.

3.1 Shooting of patches

Shooting a patch **P** is the process of distributing the energy present at **P** out to the rest of the environment. The algorithm presented in [Wallace et al. 89] is employed; for every sample point **S** in the environment, an approximation to the form factor is determined, and the energy is transferred. Finally, the 'unshot' energy total for **P** is set to zero.

The form factor is approximated using the analytical result for a disc to a differential area. To test for occlusion, a single ray can be cast from **S** to a point on **P**. To test for partial occlusion, a number of rays can be cast to various points on **P**, and an approximation made to the degree of occlusion.

3.1.1 Determining visibility

Determining visibility introduces several problems for the parallel system. By far the largest amount of communication in the system is determining if a ray from a source patch to a sample point is occluded. This reduces down to a simple ray traversal of the octree. The octree is distributed across the network with duplication allowed. When a ray is spawned from a shooting patch, it is first traced into an initial voxel. A request for this voxel is initiated by the data manager on the node. If the information is present on that processor, then the data is returned to the task in order to determine visibility. If the data is not on the processor, then a request is sent out to the network. If the model is very large and cannot be contained in the memory of the network, then it may be held on disc backing store which can be polled for the data if it is not found in the network.

3.1.2 Use of caching in octree data

The system utilises a caching strategy for the octree geometry data. This means that data from other processors in the network can also be held in a dynamic cache on the processor requesting the data. This means that for a particular source patch shooting to a cluster of sample points, there will be a certain amount of coherence for the data required. The cache currently used is a simple fully associative cache with a temporal replacement scheme. The octree data structure and message packets in the system consist of the voxel itself and all geometry contained therein. Consequently, voxels are smaller than would normally be used in a ray tracing problem.

3.2 Adaptive mesh refinement

Without pre-processing, the initial distribution of sample points throughout the environment will not be sufficient to accurately model the radiance functions of the surfaces in the environment. As the solution progresses, new sample points have to be created, and the meshes updated to include them. Discontinuity meshing [Lischinski et al. 92] is currently the best method to determine where to place new sample points; however, it is computationally expensive.

Another method of increasing the accuracy of the sample point mesh is adaptive subdivision [Cohen et al. 86]. After a patch has been shot, it examines all patches in the scene, and if any fail a certain criterion (usually their radiosity gradient being too high) then more sample points are added. Although this method does not model the radiance functions as well as discontinuity meshing, and also creates more sample points, it operates much quicker. Care has to be taken when selecting the values of the tolerances for adaptive subdivision since ineffective values can cause too little or too much subdivision.

This subdivision, which occurs dynamically, has an impact on the parallel solution to radiosity. Sample points which are close may be able to benefit from image coherence, that is they will probably require the same data for object intersection tests. Since sample points on the same patch are likely to be close, it is desirable to keep such sets of sample points together on the same processor, so that they can all use the data that has to be fetched. However, this can adversely affect the load balancing of the system; several initially sparse patches could quite quickly contain the majority of the sample points in the system, which would leave other processors idle. One solution to this problem is to distribute the sample points randomly throughout the system, but this would lose the benefits of coherence, and therefore vastly increase the number of off-processor data-fetches. Another solution is to divide patches when they contain too many sample points; then one of the new patches can be passed to another processor for load balancing. This preserves the coherency, and allows dynamic load balancing, but at the cost of making the geometry data volatile. However, since geometry data is only used for intersection tests, it is not important to other processors whether a surface is two patches or four, provided that they cover exactly the same area. Therefore this volatile aspect will not necessitate any broadcast updates to ensure that all duplicates remain consistent.

4 Implications of volatile geometry

In an interactive system, it is necessary to deal with geometry being altered. Even a very small modification will change all other parts of the scene, due to the nature of global illumination.

The system is capable of generating a complete radiosity solution quicker than a sequential version of the radiosity engine; this is required to allow radiosity to be used in a design process. However, for small changes in the scene, updates can be made more quickly than is possible with a sequential system.

4.1 Data distribution

Broadcast updates will be required when geometry moves or changes in some way, since geometry data is allowed to be duplicated between processors. This is carried out by updating the backing store copy of the data, and at the same time sending a broadcast event cascading down the tree network to all nodes, updating all copies of the data as it is propagated. Sample point data will also be updated but this is simpler to do as they are uniquely represented in the system, and can be quickly accessed through the hashing strategy implemented.

The tasks in the system are essentially the sample points, or at least calculating form factors from shooting patches to sample points. Thus, when the scene is changed, sample points are altered. The distribution of these new sample points in the network is undefined and in the worst case all may reside on one processor. The load balancing strategy (detailed in section 5.4) will automatically re-

distribute these sample points throughout the network, thus utilising all processors to update the radiosity solution to reflect the new scene.

4.3 Updating the radiosity solution

Another issue for volatile geometry is how to correct the existing radiosity solution. One method for achieving this is incremental radiosity [Chen 90]. In this technique, the light sources shoot out negative light to cancel out their previous effect. Then the geometry is subtracted, and added in the new location. Finally, the light sources are shot again. This obviously requires twice as much work as shooting the light sources in the first place; if this is to occur every time a part of the geometry is moved, then it will not be feasible to use radiosity in an interactive system, due to the computational burden of recomputing the whole solution nearly every frame.

Intuitively, a change in geometry will affect the entire environment; this is way in which [Chen 90] operates. However, not all areas will be equally affected. For example, in the case of a chair in the middle of an empty room, moving the chair will greatly affect the radiance function over the floor surface, but will have negligible affect on the diffuse reflection reaching the walls. This suggests that some prioritisation is possible, with the radiance function on the floor receiving the most computation, and the radiance functions on the walls being left unaltered until there is spare computational capacity available in the system. Research is currently underway to examine ways in which existing radiance functions can be reused, since complete recalculation of a mesh may not be required every frame. This would allow a good approximation to be made from the existing radiance function for a number of frames, before a new radiance function approximation has to be generated.

5 Data distribution

The two types of data in the system require different methods of distribution, since they are used in different ways.

5.1 Geometry data

The geometry data consists of a set of surfaces, each of which has constant surface characteristics. Each surface in turn consists of a set of triangular patches. Since the form factor calculations employ a ray-casting approach for visibility determination [Wallace et al. 89], some form of acceleration is required; an octree spatial subdivision [Glassner 84] system is used for this.

This means that there are two distinct structures that have to be maintained; the hierarchy of surfaces and patches, and the octree, which is essentially an index into all the sets of patches.

Radiosity models generally tend to have large data-sets. The reason for this is that realistic lighting simulations are usually performed on environments with real-world complexity. An average size for a model to undergo radiosity treatment will be several thousand polygons. Most networks of parallel processors have nodes with a few megabytes of memory only and consequently it is impossible to contain the entire geometry of the model on a single node especially if one has to retain sample points which can exceed this by a factor of four. It is primarily for this reason that the geometry data in this system is distributed through all the processors.

Each processor will contain a cache of geometry data, which will be used to feed the form-factor calculations. If the data is not present in the cache, then it will have to be fetched from another processor. In the case of very large model databases, this data may not reside on any processor, and is fetched from disc by a central controller.

The other important issue of the storage of geometry data is how to implement the octree. In traditional uniprocessor systems, a leaf node in the octree will contain a list of pointers to patches, and a branch will contain pointers to its children. However, this use of pointers is not possible in a distributed memory machine, and so another approach has to be used.

The first option is to use a naming scheme, in which each patch and voxel is given a unique name. Then, a voxel can contain the names of the patches which are relevant, and the cache can be partitioned into two areas, one for patches and another for voxels. If the required patch/voxel is not in the cache, then a request is put out as before. The main drawback of this technique is in the increased amount of network traffic it will generate; if a voxel is fetched from another node, and contains *n* patches, these patches will also be required, which will require a further *n* data requests.

The second approach is to actually embed the patch information in a voxel. This makes the voxel much larger (in terms of memory), and more importantly makes the size of a voxel dynamic; empty voxels will be far smaller than voxels which contain a number of patches. It has the advantage of reducing the number of data requests made, at the expense of making the message size larger.

5.2 Sample point data

The ray casting approach used by the radiosity system means that we are tracing rays from a source patch to a sample point in the geometry of the model. In the normal radiosity model, the sample point is part of the geometry of the model, usually a polygon of some sort. As we are dealing with points in this system, they are distinct from the geometry, and as such are distributed amongst the parallel processing elements as separate entities. As was mentioned earlier, the geometry is simply broken down into voxels and distributed anywhere in the network with duplication allowed if there is space.

Since sample point data is dynamic, the data must be balanced so that it is not all concentrated on one processor (which may happen if subdivision occurs in one very small area in the scene). Re-distribution of "hot spots" is easily achieved by re-hashing several of the sample points when a certain tolerance level is reached. As the memory becomes more saturated then this tolerance for re-hash redistribution necessarily has to be increased.

In the initial version of the radiosity system, sample points were indexed using a hashing strategy based on the Cartesian world co-ordinate position and normal of the sample point. In the parallel model, we also index the sample points like this, but instead of giving a location in a hash table, we use this value to give us a processor number. This number is then used to send a message to interrogate the processor suspected of containing the desired sample point. If the relevant information is present, then it is passed back to the calling processor. In some cases, sample points will not be placed on their first choice of processor, for reasons of space or sometimes due to load balance issues. In this case, a re-hash of the sample point is done at the target processor, and another message sent off to the second choice, and so on. A return message is then sent to the original processor, along with the eventual target address when the sample point is located. In this way, the sample points are distributed evenly through the network.

5.3 Hashing strategy

It is necessary to have the facility to retrieve sample points from the network to allow complete patch radiance functions to be reconstructed. This is required to determine which patches are to be shot in the next iteration. Because of this, a simply random method of distributing data is insufficient; it has to be retrieved.

The main motivations behind the hashing strategy used in this system are to provide a random method of distributing data. It also provides a system to re-distribute data in mid-execution, with memory contention taken into account. This strategy also provides a method for direct retrieval of data from any node if required without the need for search broadcasts. It is very difficult in a distributed data model to keep track of where a particular data item resides. This model allows the data to be sent into the network for storage with a quick and simple method for finding that data again. For instance, if there is no room for data on a particular processor, then it will be sent out to another processor for storage, that processor being identified according to a hash function on Cartesian co-ordinate and normal vector. If this processor's memory is, in turn, full, then a re-hash will be used to send the data on again to another processor.

It is intuitively obvious that the effectiveness of the hashing function directly effects the number of re-hashes needed, and consequently the number of messages needed in order to hit the desired sample point. However, the subdivision of the model and creation of new sample points also effects this. The hashing function can be made more effective, but the subdivision of the model is completely undefined. This means that in the worst case, we may need to search all nodes in the network for a particular sample point.

5.3.1 Use of hashing to deal with hot spots

Although all processors start with a roughly equal burden of sample points, this will change through the course of the solution as the radiance functions are refined adaptively by introducing new sample points. It is likely that certain parts of the sample mesh will start to generate large numbers of additional sample points, as effects such as shadow boundaries are modelled.

This means that certain nodes in the system may develop hot spots, which will seriously unbalance the system. Such hot spots will cause memory contention, which will force a rehash automatically. The hashing strategy will cope with this by randomly distributing the new sample points to the network. Obviously, if memory is not in short supply in the network, then serious imbalances may occur before the hashing system starts to distribute the extra sample points to the rest of the network. If processors have few tasks (i.e. below some defined level) then they send broadcasts to the network requesting a redistribution of data, then, all processors above a certain pre-defined level re-hash all their tasks. In this way, load balancing can be achieved in a non-memory saturated problem.

Other possibilities currently being investigated include direct transfer of large task loads to very lightly loaded processors in an extreme case of imbalance. This does not strictly follow the hash system and means that retrieval through the hash function becomes inefficient and may require worming through the entire system for retrieval. Any data request to a specific node may be routed via several intermediate nodes. In worst case scenarios, which have multiple rehashing, the network may be traversed a number of times; to reduce the retrieval time in such cases, each intermediate node is checked for the requested data as well as the destination node.

In redistribution of tasks from hot spots, a technique known as skew hashing may be used to bias re-distribution to lightly loaded nodes. This avoids the problem of distributing tasks to other heavily loaded nodes due to the random nature of the standard hashing function.

6 Rendering of scenes

SCOPE is the data structure used by the Bristol Graphics Group (Kaleidoscope). pSCOPE is the parallel radiosity system.

Currently, all rendering for pSCOPE is done using a SGI Iris Indigo Elan. The object database is updated from the parallel network via the ethernet. This system produces reasonable results. It is anticipated that very large models could be visualised using this system with the database on the rendering engine being supplied from a super-set of data on the parallel network. In this way, super-large walk-through models can be visualised in reasonable time.

Most applications at this time for radiosity and real time systems are of the architectural walk-through variety. For very large models the environment is usually divided up into rooms and corridors. Visibility determination from rooms can cut down the amount of the model that is needed for real-time rendering [Teller 94]. As the parallel system calculates an iteration of the radiosity solution, it also determines which part of the model needs to be rendered and provides the current data-set to be rendered to the rendering engine. In this way, the renderer only processes a portion of the entire model and therefore can provide much more interactive rates.

The current system uses a simple link via ethernet to the parallel system which means the update rate of the database from the parallel system is slow. A better link interface such as through a SCSI link directly to Transputers would provide a much better solution.

7 Evaluation of the system

The efficiency of the system is being evaluated and improved by establishing the parameters needed for judging the ratio of tasks to threads that are required to reduce processor wait time to zero. The optimum size of a system in terms of processing elements and system configuration is known to be dependant on the scene complexity and also on the size of the data set. Parameterising this aspect of efficiency is a difficult problem which requires the evaluation of cache coherence strategies, message passing protocols, the original decomposition of the problem and its representation in an octree data structure. The crucial point for effective load balancing in this system is the way in which data is re-hashed to distant processors. The relationship between the creation of excess data through scene changes (or some form of local scene complexity) and the computational requirement for that data is subtle and needs careful evaluation. Active research is currently being undertaken to find solutions to this problem.

8 Conclusion

A basic radiosity system has been developed that produces high quality images from scene data of varying complexity. Through the combination of discontinuity meshing, adaptive subdivision of meshes and ray casting for calculating the form factors, the system is able to correctly render shadow areas, highlight regions and geometrically complex surfaces.

To accelerate the system to cope with real time changes to a radiosity scene, it has been parallelised and implemented on Transputer processing elements. This has enabled the data sets to be expanded in size and complexity and allowed the system to return solutions in a shorter period of time. The load balancing mechanism is a combination of task and thread management whose work units consist of original data and hashed data. The effectiveness of this combination is crucial to the overall efficiency of the system. The development of the system is continuing with the aim of determining the parameters which optimise individual processor element performance and also the overall system performance.

References

Chalmers A. G., Paddon D. P. (1989) *Parallel Radiosity Methods*.. Proceedings of the First International Conference on the Application of Transputers. Liverpool.

Chen S. E. (1990) *Incremental Radiosity : An Extension of Progressive radiosity to an Interactive Image Synthesis System.* Computer Graphics 24 (4) 135-144,

Cohen M. F., Chen S. E., Wallace J. R., Greenberg D. P. (1988) *A Progressive Refinement Approach to Fast Radiosity Image Generation.* Computer Graphics 22 (3), 75-84.

Cohen M. F., Greenberg D. P., Immel D. S., Brock P. J. (1986) *An Efficient Radiosity Approach for Realistic Image Synthesis.* IEEE Computer Graphics and Applications 6 (2), 26-35.

Feda M., Purgathofer W. (1991) *Progressive Refinement on a Transputer Network.* Proceedings of the Eurographics Workshop on Rendering, Barcelona.

Glassner A. S. (1984) *Space Subdivision for Fast Ray Tracing.* IEEE Computer Graphics and Applications 4 (10), 15-22.

Goral C. M., Torrance K. E., Greenberg D. P., Battaile B. (1984) *Modelling the Interaction of Light Between Diffuse Surfaces*. Computer Graphics 18 (3), 213-222.

Gortler S. J., Schröder P., Cohen M. F., Hanrahan P. (1993) *Wavelet Radiosity*. Computer Graphics 27 (4), 221-230.

Green S., (1991) *Parallel Processing for Computer Graphics.* Research Monograph in Parallel and Distributed Computing. Pitman.

Green S., Paddon D. J., Lewis E. (1988) *A Parallel Algorithm and Tree Based Computer Architecture for Ray Traced Computer Graphics.* Proceedings of Parallel Processing for Computer Vision and Display, Leeds.

Hanharan P., Salzman D., Aupperle L. (1991) *A Rapid Hierarchical Radiosity Algorithm.* Computer Graphics 25 (4), 197-206.

Lischinski D., Tampieri F., Greenberg D. P. (1992) *Discontinuity Meshing for Accurate Radiosity.* IEEE Computer Graphics and Applications 12 (6), 25-39.

Samet H. (1989) *Neighbour Finding in Images Represented by Octrees.* Computer Vision, Graphics and Image Processing (46) 367-386.

Teller S., Fowler C., FunkhouserT., Hanharan P. (1994) *Partitioning and Ordering Large Radiosity Computations.* Computer Graphics 28 (4), 443-450.

Wallace J. R., Elmquist K. A., Haines E. A. (1989) *A Ray Tracing Algorithm for Progressive Radiosity*. Computer Graphics 23 (3), 315-324.

Zatz H. (1993) *Galerkin Radiosity : A Higher Order Solution Method for Global Illumination*. Computer Graphics 27 (4), 213-220.

Biographies

David Stuttard is a PhD student in the Department of Computer Science at the University of Bristol where he obtained a BSc in 1991 from the University of Bristol.

His research interests include photo-realistic graphics, real-time graphics, virtual reality and advanced parallel processing methods.

Adam Worrall is a PhD student in the Department of Computer Science at the University of Bristol where he obtained a BSc in 1993 from the University of Bristol.

He is currently researching the area of interactive realism in computer graphics. He is particularly interested in adaptive meshing techniques for radiosity rendering.

Derek Paddon is a Senior Lecturer in the Department of Computer Science at the University of Bristol and is Joint Director of the Kaleidoscope Computer Graphics Laboratory. He obtained his BSc in 1970 from the University of Wales, his MSc in 1971 from the University of Reading and his PhD from the University of Wales in 1979.

His research interests are in computer graphics, parallel computing and software engineering. His research group is actively researching methods for photo-realistic rendering, volume visualisation, virtual reality, animation and the parallelisation of computer graphic methods.

Claire Willis is Joint Director of the Kaleidoscope Computer Graphics Laboratory at the University of Bristol. She obtained her BSc in 1984 from University College London and her PhD from the University of Bristol in 1993.

Her research interests are in computer graphics, parallel computing and software engineering. Her research group is actively researching methods for photo-realistic rendering, virtual reality, user interfaces and the parallelisation of computer graphic methods. A further interest is in the use of graphical systems in situations that require safety critical software.