Chapter 8

Conclusions

The simulation of global illumination is an important tool in generating realistic images. It is challenging because the illumination in an environment is interdependent; the light reflected by any object in the scene potentially depends on the light reflected by all other objects in the scene.

Finite element-based radiosity methods solve for the global illumination of a scene containing diffuse (matte-like) surfaces. They are useful because their output is view-independent illuminated geometry, which can be re-rendered from any viewpoint, without repeating the global illumination calculations.

8.1. Contributions

In this dissertation, I have described a method for performing radiosity calculations on scenes containing large, detailed models. I call this method *face cluster radiosity*. It can result in large performance gains when model detail is greater than the natural element size needed to solve for scene radiosities. Because it uses surface-based clusters, it avoids a number of potential artifacts possible with the previously-used volume-based clustering algorithm. In particular, irradiance can be interpolated across the cluster in a natural way. Thus solution quality is also improved.

My algorithm is capable of performing the most time-consuming part of the radiosity simulation, solving for element radiosities, in a time sub-linear in the

number of input polygons. This compares favourably to the $k\log k$ complexity of the previous best algorithm, hierarchical radiosity with volume clustering. In particular, after a certain point, arbitrary tessellation of the input scene does not affect the solution time. That is, in the limit, the complexity of the solution phase of the new algorithm is constant time.

The algorithm does rely on a preprocessing phase which is $O(k\log k)$, but this cost is a per-model one that can be amortized over a number of different model instantiations and scenes. The complexity of the post processing phase during which mesh vertices are actually coloured must of course be $O(k)$, but the constant is quite small.

My algorithm has three essential components.

- **The use of a surface-oriented *face cluster hierarchy.*** These hierarchies can be pre-calculated on a per-model basis. The hierarchies can be built with a variant of Garland & Heckbert's quadric-based simplification algorithm, one that works on the geometric dual of the model in question. This has the advantage that it optimizes for the production of flat clusters.
- **Radiosity calculations based on *vector irradiance*, rather than scalar irradiance.** This allows us to avoid the push-to-leaves phase normally associated with radiosity clusters, while still modelling the effect of the variation in surface normal during the final model-colouring stage. This saves considerably on memory use and time, as unnecessary model detail need never be paged in.
- **Bounds on the projected area of a cluster of connected polygonal faces.** I have established constant-time methods for finding relatively tight upper and lower bounds for the projected area of a face cluster. This is a crucial step in accounting for the error introduced by the vector radiosity approximation, and thus driving refinement of clusters to the point where this error is acceptable.

Together, they make it possible to calculate radiosity solutions for scenes that were not tractable with previous methods, on a modest-sized personal computer. They also make radiosity on such scenes much faster than the best existing ray-tracing approach, which uses diffuse-sample caching. It is possible, for example, to compute a radiosity solution for a scene of three million polygons in four minutes on a 450 MHz Pentium processor with 160 MB of memory. The improvements to the radiosity algorithm presented here will help make it more usable in industry for applications such as architectural CAD, movie special effects, and

video games, especially as the number of polygons used to represent model and scene detail continues to climb.

Finally, I have also

- Modified Garland's original face cluster creation algorithm to be robust in the presence of flat mesh regions, produce more balanced hierarchies, and run in essentially linear time. The new implementation, *make-fch*, can cluster a million polygon model in under three minutes.
- Implemented the face cluster radiosity algorithm within the context of my radiosity program *Radiator*. This program also handles most previous radiosity algorithms, and has a visual front-end for demonstration purposes.

Together these programs make up more than 80,000 lines of C++ code. The code and associated documentation is freely available for research or production use from the web site http://www.cs.cmu.edu/~ajw/thesis-code/.

8.2. Future Directions

There are a number of ways to extend the method presented in this dissertation. The following avenues of research show particular promise.

Higher-order irradiance vector representations

The irradiance vector plays a prominent part in face cluster radiosity. However, it is to vector radiosity what the Haar basis is to scalar radiosity; a constant quantity averaged over the finite element in question. Thus there exists the opportunity to exploit smoothness in the radiosity transport kernel by generalising vector irradiance to tensor representations of various ranks. This could also make interpolating vector irradiance unnecessary, though experience with wavelet bases in the scalar domain indicates this may be unlikely.

Glossy Illumination

Any work on radiosity methods is almost compelled to mention extensions to handle specular or more realistically glossy reflection, although this is rarely carried out. In our case, however, the work described in **Chapter 4** holds some promise. In the face cluster radiosity algorithm, we are essentially modelling a collection of small, connected diffuse surface elements with a single larger element. If our lower and upper bounds could be generalised to surfaces with more general radiance distributions, there exists the possibility that the same could be

done with non-diffuse surface elements. The results could be invaluable in either controlling a glossy global illumination algorithm, or determining "good" sampling directions for a Monte Carlo-based algorithm.

Better Visibility Approximations

Currently fractional visibility is used during the radiosity simulation, and (when interpolation is used) area-to-point visibility is interpolated for the final output pass. This relatively coarse approach to sampling visibility could be usefully refined by employing higher-order approximations.

Also, in this dissertation, visibility sampling has been carried out by using a nested grid representation of the original scene. This can result in large amounts of memory being used to store the grids. Although we can employ a form of approximate visibility to reduce memory and time requirements, we must pick a single approximation level, and this must be more complex than the radiosity solution, to avoid self-shadowing problems. Thus the resolution of our visibility queries is often much greater than strictly required, especially when sampling visibility over the larger finite elements in the scene.

An interesting alternative currently under investigation is to use the face cluster hierarchy to enable multiresolution visibility queries. Queries would have some notion of the region over which visibility was being sampled, and we would only descend as far down the cluster hierarchy, which acts as a bounding box hierarchy, as needed to properly cover this region. Moreover, the projected area of a face cluster, which I have investigated heavily in the course of this thesis, is intimately related to the probability of a ray hitting the surface within that cluster.

Previous algorithms proposed to speed up visibility sampling for radiosity have used density grids to avoid ray-casting against the original scene geometry, but such visibility estimates are isotropic. For instance, a long flat surface will take on the width of the grid's cell size. By taking advantage of the various projected visible area estimates we have for face clusters, we can avoid this isotropy.

Handling disconnected topology

The algorithm presented in this dissertation performs best on connected surfaces. It attempts to exploit the coherence such surfaces present. Groups of disconnected surfaces are handled by using the volume clustering approach much used by previous hierarchical radiosity algorithms. More could be done to bridge the gap between these two representations. When we have a number of disconnected surfaces which are similarly oriented, and do not overlap much, we could use the same approach as for face clusters with little loss of accuracy. When we have a

number of randomly oriented surfaces, with significant shadowing occurring, we should fall back to a more volume-oriented approach, but it is possible we can still produce constant-time projected-area bounds by using a sample interpolation method similar to that presented in **Chapter 4**. Identifying these situations and exploiting them is an avenue for further research.

Visual Masking of Bumpy Surfaces

While working with face cluster radiosity, I have observed that when a surface contains many bumps and ridges, and consequently the scalar irradiance of the surface is changing rapidly, errors in the radiosity field are much less noticeable. It seems plausible that the visual masking techniques of Ferwerda et al. could be used to add a "masking" weight to each cluster [Ferw97]. Clusters with small weights could then be refined less than smoother clusters.

Illumination Maps

One of the most exciting possible further directions of this research is the adaption of the algorithm to produce an *irradiance vector map*. The output of the solution algorithm, before the final post-processing stage, essentially partitions the geometry into a set of regions, each with a number of (possibly interpolated) irradiance vectors representing the illumination of that region. We can refer to this set of (disjoint) regions as the irradiance vector map.

This map could then be used to render illuminated geometry quickly and efficiently, as the viewpoint changes, taking into account specular reflection. Rendering each frame would only require evaluating the local illumination, and only for a single direction. It has the great advantage that this re-rendering would be independent of the number of light sources in the scene. This kind of vector-based shading also lends itself well to some of the more recent graphics hardware [Heid99, Kaut99].

This would not result in a complete global illumination solution, as specular reflections are limited to the final stage of any illumination path. Moreover, approximating the incoming light as uni-directional may be unrealistic where specular BRDFs are involved; a small number of the most significant transport links may have to be used instead. However, such an approach may be able to match the quality of the radiosity / ray-traced post-process algorithms commonly in use for architectural global illumination solutions.

Indirect output primitives

Currently, the focus of the algorithm has been the output of illuminated geometry: a (possibly refined) version of the input scene, together with vertex colours. Interesting alternatives include the generation of shadow maps and environment maps or virtual light sources. Commodity graphics hardware is starting to support these primitives, and at some point it will become reasonable to essentially perform the final pass of the radiosity algorithm in hardware.