Voxel Occlusion Testing: A Shadow Determination Accelerator for Ray Tracing

Andrew Woo †
John Amanatides ‡

Department of Computer Science
University of Toronto
Toronto, Ontario
M5S 1A4

1. Abstract

A shadow determination accelerator for ray tracing is presented. It is built on top of the uniform voxel traversal grid structure. The accelerator proves to be rather efficient, requires little additional memory and the worst case scenario per shadow determination just reduces down to traditional voxel traversal. It can also be extended to model linear, area lights, as well as atmospheric shadows.

Keywords: grid, intersection culling, occlusion, penumbra, ray tracing, shadows, umbra, voxel traversal.

2. Introduction

The presence of shadows in a scene is important in conveying realism and aiding depth perception [Woo89b]. It is seen as a comparative darkness within an illuminated area caused by the interception of light by another object. The dark region produced can provide information such as the approximate shape and relative proximity of the intercepting object(s). It can also indicate the approximate location, intensity and size of the light sources.

Hard shadow determination algorithms for opaque surfaces have been categorized into six general classes: shadow generation during scanout [Appe68] [Bouk70], shadow volumes [Crow77] [Brot84] [Berg86] [Max86], shadow volume binary space partition tree [Chin89], area subdivision [Nish74] [Athe78], depth buffer [Will78] [Hour85] [Reev87], and ray tracing [Appe68] [Gold71] [Whit80]. The first four approaches have the general constraint that only planar polygons can be easily handled. The depth buffer approach [Will78] does not have this constraint but introduces additional aliasing artifacts. However, the aliasing artifacts have been reduced by filtering, as in the work done by [Hour85] [Reev87].

† Andrew can be reached at SAS Institute Canada Inc., 225 Duncan Mill Road, Don Mills, Ontario, Canada, M3B 3K9.

‡ John can be reached at Department of Computer Science, York University, North York, Ontario, M3J 1P3.

3. Shadow Determination in Ray Tracing

Ray casting [Appe68] [Gold71] was introduced as an elegant method for visibility calculations and a simple shadow determination approach without the previous shadow algorithm limitations. *Ray tracing*, as popularized by Whitted [Whit80], uses the ray casting technique to model reflections and refractions.

The principle behind shadow determination in ray tracing is simple. A shadow ray is shot from the visible point (point to be shaded) to the light source. If the shadow ray intersects any objects between the visible point and the light source, then it is in shadow; otherwise it is not. Each intersection check is very floating-point intensive and the naive ray tracing approach requires such checks with all objects in the scene. Since each ray-surface intersection check is very expensive, there is a need to calculate a small candidate set of objects that can possibly intersect the ray to reduce computation time. This is referred to as *intersection culling*.

4. Intersection Culling Algorithms

There has been a great deal of research in this aspect of ray tracing acceleration. One class of intersection culling algorithms uses spatial subdivision, of which there are two general approaches: *voxel traversal* [Glas84] [Fuji86] [Aman87] [Snyd87], and *hierarchical bounding volumes* [Rubi80] [Kay86] [Gold87].

Since the new shadow determination accelerator is built on top of the uniform voxel traversal grid structure *, it is necessary to go over the fundamentals of voxel traversal. Space encompassing all objects is placed in a grid of unit cubes called *voxels*. Each voxel contains a list of all objects which reside in that voxel. Each ray traverses the grid inorder and tests for intersection only with objects residing in the voxels traversed (inorder), until an intersection is found or the ray has completely traversed the grid.

* There are basically two variations of voxel traversal: variable sized voxels using hierarchical data structures [Glas84] [Fuji86], and uniformly sized voxels using a grid structure [Fuji86] [Aman87] [Snyd87]. Uniform voxel traversal is used in the implementation of the accelerator.

Voxel traversal culls all ray types (including shadow rays) and tends to provide small candidate set of objects that a ray needs to intersect with. However, many of the intersections performed when applied to shadow rays are still unnecessary and savings related to shadow calculations can prove to be valuable. Thus shadow ray cullers need to be looked into.

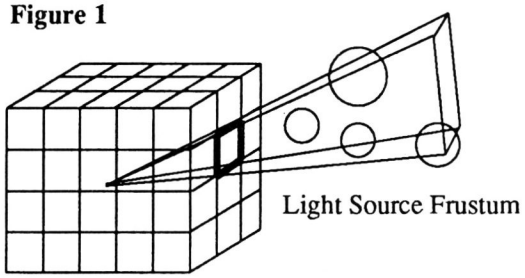
5. Shadow Ray Culling

Dealing only with opaque objects and hard shadow generation, shadow determination is just a binary decision. It is not concerned with information about object(s) that occlude it. Thus many intersection checks can be avoided. There have been only two attempts at shadow ray culling in ray tracing: *light buffer* and *hybrid shadow testing*.

5.1. The Light Buffer

An observation made by Haines et al. [Hain86] is that, in many scenes, shadow determination may dominate the ray tracing processing time. This is especially true when multiple light sources are involved. The *light buffer* [Hain86] is constructed to lower the processing cost of shadow determination.

Figure 1



The light buffer consists of six grid planes forming a box surrounding the light source origin (point light source is assumed). The cells of the buffer contain information on the smallest *full occlusion* (completely fills cell) distance from the light source, and sorted approximate depth values of candidate occlusion objects obtained by projecting objects onto the surface cells during preprocessing. See figure 1, where the four spheres within the light frustum are stored in the cell's data structure of candidate occlusion objects.

For each intersected point in question, if the depth value of the corresponding cell is greater than the smallest full occlusion distance, then the intersection point is in shadow. Similarly, if the cell is void of projected objects, then the surface is not in shadow. Otherwise, shadow determination requires intersection tests with candidate occlusion objects of the cell, performed inorder with respect to the depth values until either an intersection hit is found (in shadow) or the depth value of the candidate occluding object is greater than the intersection point (not in shadow).

Haines et al. report a substantial improvement in the shadow determination phase. However, large memory space is required: $O(N^2n)$ per light source, where $N \times N$ is the resolution of one grid plane of the light buffer, and n is the total number of objects. The preprocessing is also expensive: $O(\max(EnN^2, N^2 n \log n))$ per light source, where E is the average number of edges per polygon, EN^2 accounts for the scan-conversion cost per object, and $n \log n$ represents the sorting required per cell.

5.2. Hybrid Shadow Testing (HST)

Hybrid Shadow Testing (HST) [Eo89] is another approach to accelerate shadow determination. It applies a shadow polygon approach [Crow77] [Brot84] [Berg86] [Max86] and voxel traversal [Glas84] [Fuji86] [Aman87] [Snyd87]. The shadow polygons are placed in the voxel traversal grid structure, and the shadow count is kept up-to-date as the ray is traversed. No shadow rays are generated for this scheme, but intersection calculations with the shadow polygons, if encountered in a traversed voxel, are necessary. When the closest intersected surface is found, the shadow count is checked. If the count is 0, then the surface is not in shadow; otherwise it is in shadow.

Eo et al. [Eo89] realize that the number of shadow polygons are large. Thus each ray may potentially need to intersect many shadow polygons. As such, the traditional shadow ray approach applying voxel traversal is used if many such shadow polygons need to be intersected. The storage and runtime complexity for the HST algorithm is $O(EN^3n)$ per light source, where E is the average number of edges per polygon, n is the number of objects in the scene and $N \times N \times N$ is the resolution of the grid structure.

6. A New Shadow Determination Accelerator

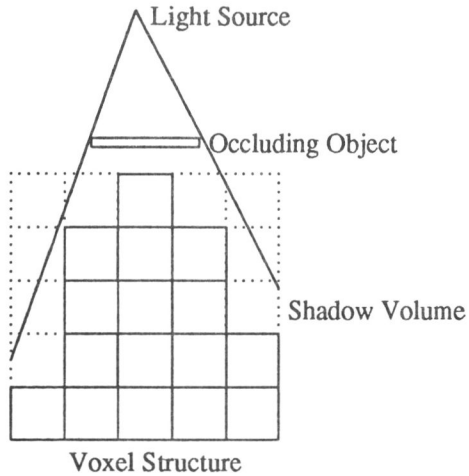
The new shadow determination accelerator, *voxel occlusion testing* [Woo89a], possesses favourable properties in that it uses a data structure already built for fast ray tracing, requires little additional storage, models directional and extended light sources, and has a good worst case scenario.

The implementation of the accelerator is built on top of uniform voxel traversal, i.e. each voxel is the same size. Given the uniform grid structure, an extra 2-bit field needs to be added per light source in each voxel. Its value indicates the level of opaque occlusion of the voxel with respect to each light source: *full, null* and *complicated occlusion*. Note that the remainder of the process description assumes only one light source, though the process is to be performed for each light source in the scene.

As a preprocessing step following object placement within the voxel data structure, all empty voxels are marked with *null occlusion* and non-empty voxels are marked with *complicated occlusion*. For each object, the shadow umbra [Crow77] is projected down to the relevant voxels. The voxels which reside solely within the shadow umbra are marked with *full occlusion*; the voxels that contain the shadow umbra edges are marked with *complicated occlusion*. Refer to figure 2, where the *dotted* voxels represent *complicated occlusion*, and the remainder of the voxels are marked with *full occlu-*

sion.

Figure 2



When a ray intersects a surface in a voxel that has either *full* or *null occlusion* (a 2-bit comparison), then any intersected surface in the voxel is ensured to be in shadow or not in shadow, respectively. No shadow ray nor intersection checks need to be performed; this is referred to as *first pass voxel occlusion testing*. If *complicated occlusion* is the voxel occlusion value, then the fastest method is to resort back to voxel traversal (and intersection checks with candidate set of objects) of the shadow ray. However, as each voxel is traversed, its occlusion value is also checked (another 2-bit comparison). If the occlusion value is either *full* or *null occlusion*, then traversal can be halted and the intersected point is ensured to be in shadow of this light source or not, respectively; this is referred to as *intermediate voxel occlusion testing*.

At worst, no known occlusion values will be found during traversal. As a result, considering the negligible 2-bit comparisons, the worst scenario per shadow determination is the same as for voxel traversal. Refer to the below pseudo code for the different shadow determination procedures under voxel traversal and voxel occlusion testing.

```
/* Voxel Traversal's Method for Shadow Determination */

VoxelTraversalShadow (point, light)
    {
    voxel = CalculateCurrentVoxel (point);
    ray = GenerateShadowRay (point, light);

    while ((voxel = TraverseNextVoxel (ray)) != outside grid)
        if (IntersectObjectsInVoxel (ray, voxel) == hit)
            return (in shadow);
    return (not in shadow);
    }
```

```
/* Voxel Occlusion's Method for Shadow Determination */

VoxelOcclusionShadow (point, light)
    {
    /* First Pass Voxel Occlusion Testing */
    voxel = CalculateCurrentVoxel (point);
    if (voxel occlusion value == full occlusion)
        return (in shadow);
    else if (voxel occlusion value == null occlusion)
        return (not in shadow);

    /* Intermediate Voxel Occlusion Testing */
    ray = GenerateShadowRay (point, light);
    while ((voxel = TraverseNextVoxel (ray)) != outside grid)
        {
        if (voxel occlusion value == full occlusion)
            return (in shadow);
        else if (voxel occlusion == null occlusion)
            return (not in shadow);

        if (IntersectObjectsInVoxel (ray, voxel) == hit)
            return (in shadow);
        }
    return (not in shadow);
    }
```

6.1. Overhead for Preprocessing

The additional storage necessary is $2N^3$ bits per light source, where $N \times N \times N$ is the resolution of the grid. Note that the storage necessary is independent of the number of objects.

The preprocessing mainly involves projections of shadow umbrae onto voxels. The silhouettes of the objects are projected, then scan-converted to determine the voxel occlusion values. Thus the complexity is $O(EnN^3)$, where n is the number of objects, E is the average number of vertices per object and EN^3 accounts for the projection and scan-conversion costs per object. This preprocessing is simple for polygons since the vertices can be projected, then joined to form the projected silhouette. However, the silhouette is more difficult to identify for quadric surfaces. Detailed projection mathematics for polygons and quadric surfaces are discussed in [Woo89a].

6.2. Non-Occluded Regions

If the exact definition of *null occlusion* is considered, this occlusion value can never be found in a voxel that contains a surface, even if the surface is completely unoccluded. *Complicated occlusion* will always be found in such voxels and thus first pass occlusion testing will fail. This can be partially solved for the case of only one object residing in the voxel. The voxel can be marked *null one* if the voxel is not already marked with other occlusion values. This indicates *null occlusion* for first pass occlusion testing, and *complicated occlusion* for intermediate occlusion testing. Thus, shadow rays do not need to be generated for voxels that contain only one convex object. If the object were concave, an intersection test against itself needs to be done before the occlusion status is determined.

With more populated voxels, it is inevitable that the occlusion value may be marked with *complicated occlusion* (to handle the case of occlusion from objects within the same voxel). For first pass occlusion testing, the ray bounding box approach [Snyd87] is used to reduce intersection checks within the initial voxel. Then hopefully, intermediate voxel occlusion testing can be used to determine the occlusion later on.

6.3. Non-Opaque Objects

Shadows as a result of transparent objects are much harder to deal with †. It needs the information of all occluding objects to generate coloured shadows. In addition, due to refraction, a single linear shadow ray is not sufficient to generate the correct results. The compromise commonly taken in ray tracing is to assume no refraction in the transparent occluding objects. This is also the assumption made in this shadow determination accelerator.

A new occlusion value *full transparent occlusion* can be added. However, upon reaching such a voxel, shadow determination cannot be stopped since information about all occluding objects are needed. Only transparent objects need to be checked for intersection. However, this has little advantage and requires the number of bits per voxel to increase to three. It is generally recommended that *complicated occlusion* be marked for the shadow umbrae of transparent objects.

7. Implications of the Accelerator

7.1. View Independence

In a rendering technique to calculate radiosity [Gora84], view independence can be achieved. Ray tracing is generally a view dependent rendering technique since its illumination calculations are dependent on the ray shot from the eye. But since shadows from opaque objects are only fractional value occlusions multiplied with intensity calculations, they can be view independent if desired. In this paper, voxel occlusion testing provides a view independent approach for shadow calculations in ray tracing. Thus the 2-bit voxel occlusion values need only be generated once and can be retained for future frames in any fly-by animation. In addition, changes in the power or spectral description of any light source require no change in the voxel occlusion values since the fractional occlusions are still the same.

7.2. Shadows on Bump-mapped Surfaces

As stated in [Woo89b], shadows on bump-mapped surfaces look perfectly smooth because the shadow ray is not perturbed to reflect the bumpiness. An analytic solution to the perturbation of the shadow ray was proposed in [Woo89b], but is expensive in general. However, the perturbation of the shadow ray only needs to be done near the silhouette of the shadow; the region in the middle usually ends up with the same shadow results. Thus, when the voxel occlusion value is *full occlusion*, it is assumed that it is not near the silhouette region and no work is done to perturb the shadow ray. When the voxel occlusion value is *complicated occlusion*, then the shadow ray is perturbed. This should save some computations in perturbation of the shadow rays.

† Actually, these are not shadows, but filtering of light [Woo89b].

7.3. Voxel Occlusion may not Accelerate

Our testing suggests that voxel occlusion testing should not be used on low resolution images, e.g. 64×64. The preprocessing time may dominate the rendering process. However, the larger the resolution or the higher the sampling rate, the better the improvement of the runtime performance since the preprocessing is independent of resolution and becomes much less dominant compared to the actual rendering process.

Even at reasonable resolutions, this accelerator may not provide accelerated runtime performance over voxel traversal at times (on a single frame basis). Some additional work should go into a quick check whether as to use voxel occlusion testing during the preprocessing stage (on a single frame basis). This can be done since the algorithm is just a simple attachment onto the voxel traversal approach after the grid structure is produced.

A simple solution to this problem is to evaluate the scene after the grid initialization for voxel traversal is done. The equation to be evaluated is presented below, where n represents the total number of objects, the numerator represents the total region occupied by the objects, and when divided by the grid resolution N^2, represents a percentage of the total volume used:

$$\frac{1}{n\,N^2}\sum_{i=1}^{n} \textit{number of voxels occupied by object } i$$

and voxel occlusion testing is not to be used if the above is below some *threshold value* (empirically derived), assuming that the small objects do not project a large shadow umbra. Note that this is only a quick check and may not be valid all the time.

8. Testing and Analysis

8.1. Some Numerical Results

Testing has been done on a *SUN 3/280* with *fpa*, and the algorithm was implemented on a ray tracer at the University of Toronto: *optik*. The voxel traversal implementation in *optik* is described in [Aman87]. All test images are rendered at a resolution of 512×512 with one sample per pixel.

Table 1 illustrates the effectiveness of voxel occlusion testing. A couple of images were taken from the procedural database [Hain87]. Note that in the column *grid*, a grid subdivision level of N×N×N is assumed. In addition, *Voxel* and *Occ* indicate the **total CPU time** in seconds taken to render the image using voxel traversal and voxel occlusion, respectively. Finally, *%Shad* represents the percentage of shadow rays over all ray types.

In terms of the savings acquired, table 2 is given for the identical images in table 1. *%Known* is the percentage of shadow determination requests that does not require shadow ray generation. In addition, *#Trav* and *#Int* is the total number of traversals and intersection tests saved for using voxel occlusion testing, respectively. Finally, *#Intermed* is the total number of successes using intermediate voxel occlusion testing.

Table 1

Image	#light	Grid	#Obj	Voxel	Occ	%Shad
Graph	2	40	12	1518.2	1220.3	63%
Quadric	3	50	15	1057.2	880.6	54%
DGP	1	20	21	797.5	658.7	41%
Bars	1	30	88	1419.6	1052.6	49%
Flakes	3	50	92	3234.7	2970.9	70%
Tetra	1	40	4096	747.5	985.1	10%

Table 2

Image	%Known	#Trav	#Int	#Intermed
Graph	58%	11209198	347883	129073
Quadric	67%	7071098	333625	44227
DGP	74%	7016086	174635	13138
Bars	68%	662781	14314895	16099
Flakes	36%	1066370	21094921	214037
Tetra	1%	12872	97692	10756

8.2. Voxel Occlusion vs. Voxel Traversal

In this section, a synopsis of the comparison between the two methods is given. A more indepth analysis can be found in [Woo89a].

Voxel occlusion testing, on average, shows a good improvement over voxel traversal: about 15% improvement per light source on the **total CPU time** on our selected images. The ratio of shadow rays over all ray types tends to play a large role in the improvement level. The larger the percentage, the better the improvement. In particular, the addition of more light sources, thus more shadow rays, exhibits approximately linear rate of improvement.

For a small number of objects (less than 250 objects), different subdivision levels seem to hamper voxel traversal much more than voxel occlusion testing. Thus the necessity to find the optimal subdivision level with voxel occlusion is less important; an educated guess may be sufficient. In addition, the optimal subdivision level for voxel traversal is usually larger compared to voxel occlusion testing. Grid sparseness with small densely populated regions also seems to hamper voxel traversal much more than voxel occlusion testing. This variable is important to analyze since it gives voxel traversal a great deal of trouble: a deep subdivision level will require extensive traversal, but a not deep enough subdivision level may require intersection checks with many objects.

However, the above paragraph's observations are not true for large number of objects. Grid sparseness causes intersection savings to be few if the subdivision level is small. By increasing the subdivision level, the storage cost increases further to cause swapping problems. This is one prime reason to perform a quick check to determine if voxel occlusion testing should be used on top of voxel traversal. However, a better solution to deal with this problem is proposed in §9.2.

8.3. Voxel Occlusion vs. Light Buffer

As compared to the light buffer, voxel occlusion seems to perform better under some informal analysis. The lighting frustum, which encompasses all candidate objects per cell in the light buffer, grows larger as it gets farther away from the light. Thus the probability of finding known occlusion gets much smaller as the distance to the light source increases. This also results in a larger candidate set of objects to intersect with.

Voxel occlusion relies on the grid structure to locate candidate sets of objects to intersect with. The candidate sets tend to be much smaller since it does not grow as it gets farther away from the light source. For the same reason, the probability of finding known occlusion in a voxel should be higher when compared to the light buffer.

Another disadvantage of the light buffer is the need for large storage requirements. Haines et al. report improvement factors of 4-10. However, this is misleading since it is compared to traditional ray tracing without intersection culling (and the additional storage requirements of the culling approach). Voxel occlusion, at 2 bits per voxel, requires a lot less memory.

8.4. Voxel Occlusion vs. HST Algorithm

As can be seen, voxel occlusion testing just reduces down to voxel traversal at the worst case. Hybrid Shadow Testing (HST) attempts to acquire this upper bound also by switching between shadow polygon and traditional shadow ray. However, this upper bound cannot be guaranteed.

On average, the number of intersections required seems to favour voxel occlusion testing: the HST algorithm deals poorly with hierarchy of shadow volumes. A point shadowed by many objects needs to intersect against each of those objects, whereas our approach only requires checking a single occlusion value. In addition, it must be noted that HST requires a much larger memory capacity.

9. Proposals for Improvement

9.1. Complex Surfaces

This accelerator needs to project objects with known silhouettes (less exact silhouettes can generally be acquired though). Thus, implicit and parametric surfaces present a problem if numerical iteration (direct rendering) is used to render them. Approximate shadow projections have to be considered; i.e. have a known shadow region, and a small, uncertain shadow region around the silhouette.

Another method to render such surfaces would be polygonization. This does not provide a problem for our accelerator since it is just projection of polygons. However, the silhouette of the shadows appear polygonal-like. Perhaps approximate shadow projections (as discussed above) can be used so that the silhouette region can be smoothed out.

9.2. Lazy-Evaluation Occlusion Testing

The voxel occlusion preprocessing done might be a waste if many of the voxels are not even traversed during ray tracing. On the other hand, the voxel may be too large to contain any useful information; i.e. mostly *complicated occlusion*. As such, a lazy-evaluation approach should be considered, perhaps along the lines of the work done by Jevans et al. [Jeva89]. Initially, a small subdivision grid formation is created and preprocessed with the voxel occlusion information. When calculating shadows, if the voxel occlusion at the visible point is *complicated occlusion*, then further subdivision of the voxel is required. The newly created voxels are then processed with occlusion information and used for the current and future shadow calculations.

9.3. Soft Shadow Generation

Thus far, the discussion has been applied to hard shadows of opaque objects. The domain of light sources to generate this type of shadows are directional and point light sources (assuming no inter-reflections of light). However, higher dimensional lights (linear, area lights) should produce soft shadows. Thus the problem is no longer a binary decision as to whether a point is in shadow or not, the fraction of occlusion needs to be calculated. The traditional methods for soft shadow generation in ray tracing can be seen in work done by Cook et al. [Cook84] and Amanatides [Aman84]. However, both approaches tend to be very expensive computationally.

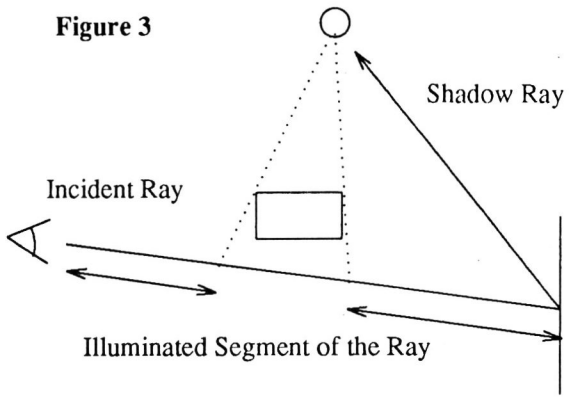
Nishita et al. [Nish85] propose a method to identify the umbra and penumbra regions on the projected plane emanating from a light source. This can be applied to voxel occlusion testing in the following manner: identify the umbra region on the voxel planes and mark the enclosed voxels as *full occlusion*, then mark the penumbra region as *complicated occlusion*. The umbra region can be detected using voxel occlusion, as before. The penumbra regions cannot be preprocessed per voxel since the level of penumbra is different for each point. Thus some shadow rays need to be shot to determine the level of penumbra, and distributed ray tracing [Cook84] along with intermediate voxel occlusion testing seems to be the most obvious choice.

9.4. Atmospheric Shadows

Sunlight scattering in the air causes the atmosphere to glow. This glow is particularly visible in a shadowed environment. Thus for a ray shot to calculate the closest visible surface, the critical problem is not only to determine whether the intersection point is in shadow. Identifying the segments of the viewing ray which are visible from the light is just as crucial. This information is necessary to acquire atmospheric shadows assuming only a single scattering model for light diffusion. See figure 3, where the two arrowed lines indicate the illuminated segments of the incident ray, thus atmospheric illumination needs to be taken into account.

Max [Max86] and Nishita et al. [Nish87] use shadow volumes to calculate the ray segments that are visible from the light. Similarly, our accelerator can be extended for voxel occlusion testing: the voxel occlusion values are checked as the ray traverses through the voxels. *Null occlusion* voxels indicates full illuminance of the ray segment within the voxel, and *complicated occlusion* indicates partial illuminance of the ray within the voxel. The fraction of illuminance of *complicated occlusion* voxels requires some additional partial sampling.

Figure 3



10. Acknowledgements

The first author wishes to thank Pierre Poulin, Caroline Houle and Alain Fournier for providing some very useful suggestions towards this paper. The authors are also grateful to NSERC and ITRC for their financial support.

11. Conclusions

A new shadow determination accelerator is proposed. This accelerator, on average, outperforms voxel traversal and existing shadow ray cullers. However, as with all intersection culling algorithms, the accelerator performs well under some circumstances and poorly under other circumstances. Lazy-evaluation voxel occlusion methods proposed should improve the situation when the basic approach performs poorly. The accelerator can also be extended to model linear, area lights, as well as atmospheric shadows.

12. References

[Aman84] J. Amanatides, "Cone Tracing", Computer Graphics (Proc. SIGGRAPH 84), 18(3), July 1984, pp. 109-115.

[Aman87] J. Amanatides, A. Woo, "A Fast Voxel Traversal Algorithm for Ray Tracing", Conference Proceedings of EuroGraphics '87, August 1987, pp. 1-10.

[Appe68] A. Appel, "Some Techniques for Shading Machine Renderings of Solids", Proc. AFIPS JSCC, vol. 32, 1968, pp. 37-45.

[Athe78] P. Atherton, K. Weiler, D. Greenberg, "Polygon Shadow Generation", Computer Graphics (Proc. SIGGRAPH 78), 12(3), August 1978, pp. 275-281.

[Berg86] P. Bergeron, "A General Version of Crow's Shadow Volumes", IEEE Computer Graphics and Applications, 6(9), September 1986, pp. 17-28.

[Bouk70] W. Bouknight, K. Kelley, "An Algorithm for Producing Half-Tone Computer Graphics Presentations Shadows and Movable Light Sources", AFIPS Conf. Proc. vol. 36, 1970, pp. 1-10.

[Brot84] L. Brotman, N. Badler, "Generating Soft Shadows with a Depth Buffer Algorithm", IEEE Computer Graphics and Applications, 5(12), October 1984, pp. 5-12.

[Chin89] N. Chin, S. Feiner, "Near Real-Time Shadow Generation Using BSP Trees", Computer Graphics (Proc. SIGGRAPH 89), 23(3), July 1989, pp. 99-106.

[Cook84] R. Cook, T. Porter, L. Carpenter, "Distributed Ray Tracing", Computer Graphics (Proc. SIGGRAPH 84), 18(3), July 1984, pp. 109-115.

[Crow77] F. Crow, "Shadow Algorithms for Computer Graphics", Computer Graphics (Proc. SIGGRAPH 77), 11(2), August 1977, pp. 242-248.

[Eo89] D. Eo, C. Kyung, "Hybrid Shadow Testing Scheme for Ray Tracing", Computer Aided Design, 21(1), January 1989, pp. 38-48.

[Fuji86] A. Fujimoto, T. Tanaka, K. Iwata, "ARTS: Accelerated Ray-Tracing System", IEEE Computer Graphics and Applications, 6(4), April 1986, pp. 16-26.

[Glas84] A. Glassner, "Space Subdivision for Fast Ray Tracing", IEEE Computer Graphics and Applications, 4(10), October 1984, pp. 15-22.

[Gold71] R. Goldstein, R. Nagel, "3-D Visual Simulation", Simulation, January 1971, pp. 25-31.

[Gold87] J. Goldsmith, J. Salmon, "Automatic Creation of Object Hierarchies for Ray Tracing", IEEE Computer Graphics and Applications, 7(5), May 1987, pp. 14-20.

[Gora84] C. Goral, K. Torrence, D. Greenberg, "Modelling the Interaction of Light Between Diffuse Surfaces", Computer Graphics (Proc. SIGGRAPH 84), 18(3), July 1984, pp. 213-222.

[Hain86] E. Haines, D. Greenberg, "The Light Buffer: A Shadow-Testing Accelerator", IEEE Computer Graphics and Applications, 6(9), September 1986, pp. 6-16.

[Hain87] E. Haines, "A Proposal for Standard Graphics Environments", IEEE Computer Graphics and Applications, 7(11), November 1987, pp. 3-5.

[Hour85] J. Hourcade, A. Nicolas, "Algorithms for Antialiased Cast Shadows", Computer and Graphics, 9(3), 1985, pp. 259-265.

[Jeva89] D. Jevans, B. Wyvill, "Adaptive Voxel Subdivision for Ray Tracing", Graphics Interface, June 1989, pp. 164-172.

[Kay86] T. Kay, J. Kajiya, "Ray Tracing Complex Scenes", Computer Graphics (Proc. SIGGRAPH 86), 20(4), August 1986, pp. 269-278.

[Max86] N. Max, "Atmospheric Illumination and Shadows", *Computer Graphics* (Proc. SIGGRAPH 85), 20(4), August 1986, pp. 117-124.

[Nish74] T. Nishita, E. Nakamae, "An Algorithm for Half-Tone Representation of Three-Dimensional Objects", Information Processing in Japan, vol. 14, 1974, pp. 93-99.

[Nish85] T. Nishita, I. Okamura, E. Nakamae, "Shading Models for Point and Linear Sources", ACM Transactions on Graphics, 4(2), April 1985, pp. 66-75.

[Nish87] T. Nishita, E. Nakamae, "A Shading Model for Atmospheric Scattering Considering Luminous Intensity Distribution of Light Sources", Computer Graphics (Proc. SIGGRAPH 87), 21(4), July 1987, pp. 303-310.

[Reev87] W. Reeves, D. Salesin, R. Cook, "Rendering Anti-Aliased Shadows with Depth Maps", Computer Graphics (Proc. SIGGRAPH 87), 21(4), July 1987, pp. 283-291.

[Rubi80] S. Rubin, T. Whitted, "A 3-Dimensional Representation for Fast Rendering of Complex Scenes", Computer Graphics (Proc. SIGGRAPH 80), 14(3), July 1980, pp. 110-116.

[Snyd87] J. Snyder, A. Barr, "Ray Tracing Complex Models Containing Surface Tessellations", Computer Graphics (Proc. SIGGRAPH 87), 21(4), July 1987, pp. 119-128.

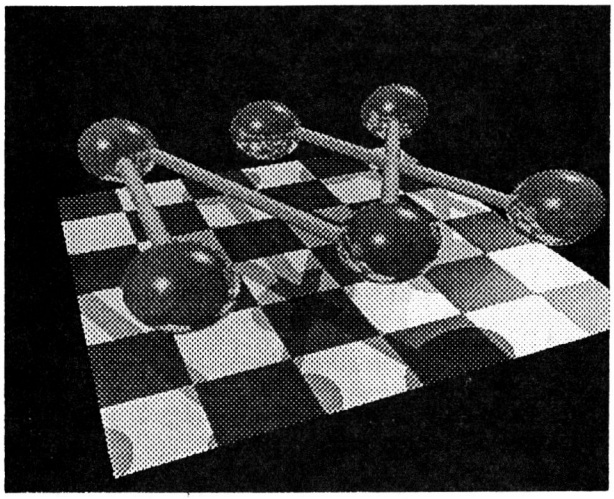
[Whit80] T. Whitted, "An Improved Illumination Model for Shaded Display", Communications of the ACM, 23(6), June 1980, pp. 343-349.

[Will78] L. Williams, "Casting Curved Shadows on Curved Surfaces", Computer Graphics (Proc. SIGGRAPH 78), 12(3), August 1978, pp. 270-274.

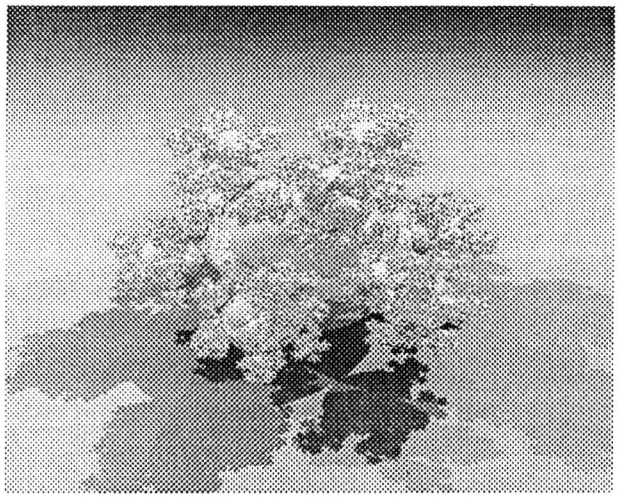
[Woo89a] A. Woo, "Accelerators for Shadow Determination in Ray Tracing", M.Sc. Thesis, Department of Computer Science, University of Toronto, 1989.

[Woo89b] A. Woo, P. Poulin, A. Fournier, "A Survey of Shadow Algorithms", Department of Computer Science, University of Toronto, submitted for publication, 1990.

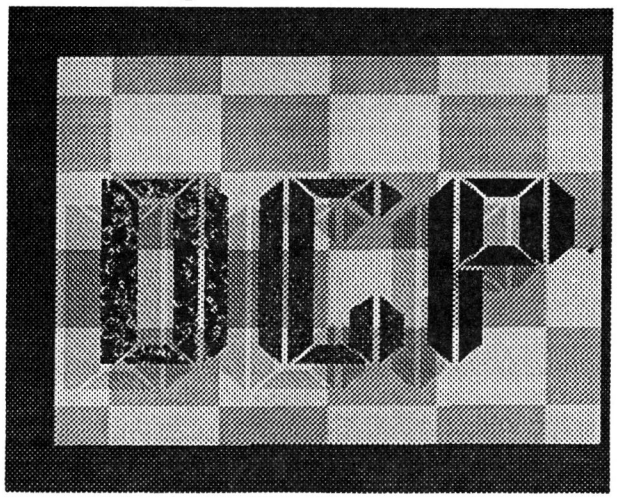
Graph Image



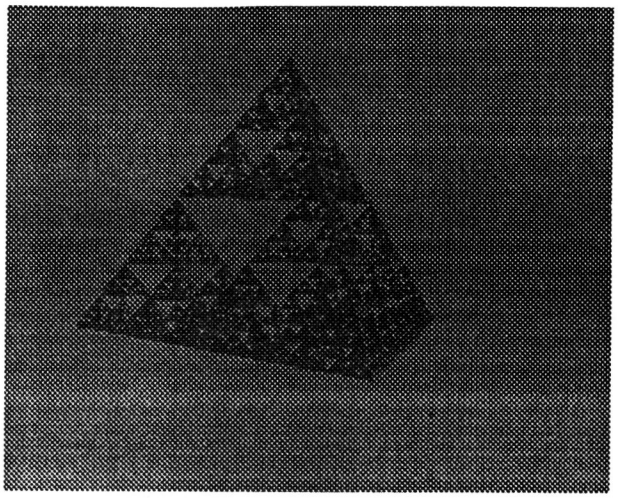
Flakes Image



DGP Image



Tetra Image



Bars Image