Approximate Ray Tracing

David Elliott Dauenhauer
Sudhanshu Kumar Semwal
Department of Computer Science
University of Colorado at Colorado Springs
Colorado Springs, Colorado 80933-7150

Abstract

We provide a general technique for all ray tracing space subdivision methods to perform what we term as "Approximate Ray Tracing." An implementation of the Approximate Ray Tracing called the Approximate Slicing Extent Technique or ASET is provided. ASET checks only one ray-polygon intersection per cell along the path of the ray. All other ray-object intersections are eliminated.

While the benefits of standard area subdivision techniques have proven to be fairly optimal, our experiments have shown an average of fifteen to fifty percent reduction in the time required to ray trace approximate images. Time savings are expected to be greater when more complex scenes are rendered. Irrespective of the scene complexity, ASET adds only a constant amount of memory overhead.

Keywords: Shading, Rendering Algorithms.

Introduction

Ray tracing has proven to be the most popular and effective method for generating realistic images from geometric and mathematical descriptions of objects. It involves the tracing of a ray or vector from a specified view point through a scene. As the ray strikes an object, a reflected and a refracted ray may be produced. These new rays in turn may strike other objects - repeating the above process. The aggregate effects of this continual process provide shading, reflections, and shadows. However, the repeated splitting of each parent ray into two new rays can produce an exponential number of rays.

Ray tracing, as described above, is too computationally time consuming for the average graphics system. It involves testing for a ray-object intersection with every object (i.e., polygon, sphere, etc.) in the scene volume. For example, ray tracing a 500 x 500 pixel scene with 5000 objects would require a minimum of 1.25 billion intersection attempts for initial or primary rays.

Consequently, researchers have pursued heuristics to reduce the number of calculations required to generate an image. One technique that has gained wide acceptance is space subdivision. The original volume (containing all the objects) is subdivided into smaller subvolumes. These small subvolumes may contain possibly fewer objects than the original undivided volume. Since rays traverse a predetermined linear path, a computational savings is realized by only considering intersection with objects in subvolumes along the path of the ray. Space subdivision has shown dramatic reductions in the number of calculations required to generate an image as the number of objects in a scene increase [3,4,7,8].

Some prevalent methods used for space subdivision include the oct tree and grid methods. Other popular techniques are the ARTS and EXCELL methods.

All the space subdivision techniques work well depending upon the scene and the desired view orientation.

The Oct Tree Method

The oct tree method [4] recursively subdivides a scene volume into 8 equal disjoint subvolumes depending upon the number of objects in the volume space. An oct tree division algorithm can be tailored to halt based on user defined parameters. These parameters are: no objects left in the subvolume; the number of objects in subvolume less than some constant; and the height of the oct tree greater than some constant.

Oct-tree works well for a scene that has large blank volumes and/or objects that are clustered densely. Since oct tree adapts its data structure to the scene, large volumes of blank data can be assigned to a single node.

When ray traversal begins, these large volumes can be bypassed with a single node check.

A drawback to using oct tree for space subdivision is the time involved in traversing the scene volume (i.e., moving from subvolume to subvolume) [3]. No constant increment can be applied to easily determine the next subvolume. Different levels of the oct tree are examined to move from one subvolume of the scene to another. This computational roller-coaster ride up and down the oct tree is time consuming and slows down ray tracing process.

The Grid Method

The Grid Method [3,7] subdivides the object space evenly along the x, y, and z axis. The result of the grid method is the creation of equally sized rectangular volumes called grid volumes or voxels. While ray tracing, it allows for easy traversal from voxel to voxel by simply adding the subdivision constant(s) to the appropriate axes (3DDA [3]).

The grid method works well for a scene that has a fairly equal dispersal of objects throughout the scene volume. It is crucial to somehow select optimal division constants for each axis. Ray traversal from voxel to voxel is easily computed and a fairly even number of objects in each voxel provide a fairly constant ray intersection computation time.

A scene ideally suited for oct tree would run very sluggishly under this method. Processing time can be wasted traversing through numerous empty voxels. The number of objects assigned to a grid voxel depend directly upon the dispersal of objects in the scene. Additionally, ray-object intersections can be duplicated when a large object is assigned to multiple voxels.

Other Methods

Other space subdivision methods not detailed in this report include the ARTS and EXCELL methods. The ARTS method [3] combines the oct tree and grid methods. The EXCELL method [9] provides a spatial index to an adaptive cell structure created by recursive binary subdivision. An excellent review of these and other techniques appear in [3,4,6].

The Slicing Extent Technique

Another space subdivision method called the Slicing Extent Technique (SET) was developed in [8]. Independently, a similar technique appeared for a different application in [11][1].

[1] In [11], the scene consists of boxes or rectangular parallelepipeds, which may not be the case in SET.

The SET method uses 3 sets of 2-D projection planes or slices instead of volumes to "surround" the object.

These slices are perpendicular to either the x, y, or z axis. The intersection of slices create two-dimensional rectangular areas or cells on every slice. Each object in the scene is associated with at least 6 cells. Slices are tested (for ray-object intersection) in sorted order along the path of the ray. No tree traversal is necessary because there are no trees.

The SET method works well when a scene has a "reasonable" (not very large) number of objects that are not in "close" proximity to each other. A large number of objects may require a proportional number of slices – creating an unmanageable data structure. However, if objects are in close proximity to each other, then many objects can be assigned to a single cell. In this situation, image generation times are higher.

The MSET Method

To alleviate the drawbacks of SET, the Modified Slicing Extent Technique (MSET) was developed [7]. Instead of placing a possibly large number of slices proportional to the number of objects in the scene, a predefined number of slices are used along each axis. This has two advantages. First, a limit on the number of slices provides for a predictable and manageable data structure. Secondly, by spacing the slices evenly along each axis, slice traversal can be done easily by simply adding an incremental slice constant (resembling traversal in the Grid Method).

Preprocessing in MSET

A bounding-box is computed for each object. Slice cells that surround this bounding-box are "marked" as containing the projection of the object onto their slice. Each slice cell has two sides – an "up" side and a "down" side. Objects projected onto a slice are assigned to one of these sides of the cell.

If a ray traveling in the "up" direction would intersect the marked slice before intersecting the actual object, the object is assigned to the "up" side of the slice cell. "Down" cells are marked in a similar fashion. The object's address is added to the object-pointer-list for the appropriate side.

Unlike the Grid method, no duplicate ray-object intersections are performed because an object can only be assigned to the 6 slices surrounding the bounding-box. The "up" and "down" marking scheme ensures that an object is checked a maximum of one time for intersection with a specific ray. *This property is unique to both MSET and ASET.*

A grid-volume data structure is also maintained. Grid-volumes or voxels are the parallelepipeds created by slices. This grid-volume data structure allows secondary rays to check for a possible intersections that might occur within the voxel (containing the present position of the ray) before any object assigned to a slice-cell is checked for nearest intersection.

Logan's modifications to the SET technique produced a five-fold optimization over the SET implementation method [7].

Motivation

When no space subdivision technique is used, it has been shown that ray-object intersection calculations take up to 95% of the image generation time [4]. The oct- tree, grid, ARTS, EXCELL and SET/MSET techniques were designed to alleviate much of the ray-object intersection processing. However, a sizable amount of intersection checks are still performed. Rays must be checked for intersection with each object assigned to the subdivided space – *with no intersection guaranteed.*

Increasing the number of space subdivisions often reduces the number of required ray-object intersections – at the expense of space subdivision traversal overhead and memory. While the traversal overhead is often more than offset by the processing time saved by fewer ray-object intersections, memory is often the limiting factor.

We consider MSET (or any other viable alternative) to be highly efficient for the production of "final" ray traced images. Still, image generation time is high. Our goal is to produce quicker draft images[2]. The draft images are close in appearance, but are not exact duplicates (See Figures 3-8).

To this end, we propose an augmentation to the MSET called the "Approximate Slicing Extent Technique" (ASET). This implementation includes a technique to preprocess a scene volume and assign a single color value to each slice cell (only if it had been assigned at least one object). These color values are to be used in lieu of color values returned from ray-object intersection processing. We also allow the specification of a recursion level. This is the level where ASET processing take control from regular MSET ray tracing. Rays which intersect the cells could either be terminated or recursively propagated.

Our primary objective is to completely eliminate ray-object intersection processing while ASET is in effect. We only perform a single ray-cell intersection to ascertain cell color. ASET would be used to draft initial and intermediate images. MSET could then be used for final image generation.

Implementation

To implement ASET, the MSET package of James Logan [7] and the Ray Tracing package of Jim Duke is used. The current configuration of this software runs on an HP9000 graphics workstation.

Three different options[3] were designed to assign a single color value to each slice cell [2]. The option we have implemented is:

1. Modify the SET data structure to contain a texture field for both the "up" and "down" sides of the slice cell. The texture field is a structure that contains all the parameters used to define the attributes of an object including its color, transmittance, and reflectivity.
2. Use the *expanded* bounding-box around each object to compute or update the texture of every slice cell. The smallest bounding-box around an object is *expanded* so that box-walls are in fact the slices. Each cell that is marked (because of the projection of the expanded bounding-box) will have its corresponding texture updated to reflect the existence of the object. Each object that is contained in a cell will have an equal effect on the color of the slice. In other words, the textures of all the objects (whose expanded bounding-box projects onto a cell) are averaged.

Figures 1 and 2 show the pseudo-code for ASET implementation.

Results

The ASET package was tested and compared against MSET – producing some very interesting and faster

[2]Image generation time may also be reduced by using more than one processors. A lower resolution image can also be traced for quicker draft images.

[3]A detailed comparison of the three options appears in [2].

Following is the pseudo-code for ASET Preprocessing:

```
Procedure ASET_Preprocessing

Begin
    - Slice the scene volume along the x, y, and z axis
    - Initialize the cells on each slice.
    - For each cell on a slice:
        - set value of object counters to zero
        - assign null object pointer lists
        - assign null material to texture pointers
    - end for.
    - For all objects (polygons and spheres):
        - Compute 3D bounding-box around object
        - Project bounding-box onto 6 nearest outer slices
        (2 slices per axis)
    -end for.
    - For each slice cell that object is projected onto,
    "mark" the slice cell by:
        - Incrementing the slice cell object counter
        - Assigning the object to the slice cell pointer list
        - if aset is invoked:
            - compute temporary texture by averaging
            object material texture with slice cell texture
            - search existing texture list for texture match
            - if texture match is found return texture address
            else
                add texture to texture list and return new address
            - assign returned texture address to cell texture pointer
    - end for.

End ASET_Preprocessing
```

Figure 1: ASET Preprocessing.

results. MSET was used for primary[4] rays and associated shadow-rays[5]. For some image renderings, this can account for more than 60% of the rays produced.

The 813 object SNOW scenes (Figures 3-6) were used for the majority of ASET tests. Two other scenes, TETRA (Figure 7) and GEARS (Figure 8), are from Eric Haines's Standard Procedural Databases [5].

Analysis

Detailed analysis of the above data produced some surprising results. Prior to this research, we believed that ray-object intersection processing was still taking an overwhelming percentage of the image processing time. Indeed, it does take some time to process ray-object intersections and that is the whole idea behind the implementation of area subdivision techniques. When sufficiently fine subdivision is used, the area subdivision techniques (e.g., MSET, ARTS, etc.) *usually* have close to optimal cell density of one object per non-empty cell. For example, when 100 grid slices are used on the SNOW scene, the average number of objects assigned to non-empty cells is 1.3 (see Table 2).

The following results gathered from numerous ASET and MSET runs confirm our belief of near-optimality for the scene containing polygonal objects. These are:

1. On the HP9000, a ray-polygon intersection averages approximately .00032 seconds of processing time.

[4]Primary rays are the rays starting from the view-point. All other rays are secondary rays.

[5]Shadow rays are generated from point of intersection to the light source.

Following is the pseudo-code for ASET ray tracing:

```
Procedure ASET_Ray_Object_Intersection_Processing

Begin

        - For each ray
         - If ray base point is internal to scene volume and
         it is a first level ray
              - Check grid-volume against object list for
              possible ray-object intersection
         - If intersection is found, return results of
         ray-object intersection (for possibly generating
         lower level reflected and/or transmitted ray)
    - end for.
     - While ray propagation is internal to scene volume:

        - Find next intersected slice cell
             - If any objects assigned to cell:
                  - If MSET
                       - Check all objects assigned to cell
                       for possible intersection with ray
                       - If intersection occurs before next
                       slice intersection, return results of
                       ray-object intersection (for possibly
                       generating lower level reflected
                       and/or transmitted ray)
                  - else if ASET:
                       - compute ray-cell intersection
                       - return results of ray-cell
                       intersection (for possibly generating
                       lower level reflected and/or transmitted ray)
    -end while.

End ASET_Ray_Object_Intersection_Processing
```

Figure 2: Ray Object Intersection Processing.

2. The other overhead associated with a single ray (not including ray-object intersection times) averages approximately .0088 seconds.
3. Consequently, the overhead of initialization, recursion, slice traversal, etc. for a single ray takes about 22 to 30 times more processing time than does a single ray-polygon intersection.

Why Use ASET?

ASET outperforms MSET by a bigger margin when the number of slices or the grid size[6] is reduced. This is because the cell density (objects per cell) increases with the reduction of the grid size (See Tables 1 and 2). Why not use a large number of slices and get a more accurate picture in less time? The answer lies in the use of memory. If memory is at a premium, ASET may be the better way to go. It simply outperforms MSET when a small number of slices are used.

Another reason for using ASET is the strange and interesting special effects that are created because of the approximation (See Figure 6). ASET processing often produces interesting images and artifacts[7] that cannot be produced normally. Since a portion of computer graphics, and ray tracing in particular, is the production of images purely for visualization's sake, ASET provides this capability.

[6]number of divisions along x, y and z axes.

[7]These artifacts can be controlled by delaying the use of ASET during ray tracing (See Figures 3,4,6).

Scene	Rendering time		No of Rays		Rays per sec	
	MSET	ASET	MSET	ASET	MSET	ASET
Snow 100	7943.9 s	8644.6 s	667,239	841,883	78	121
Snow 50	11540.0 s	11330.6 s	667,239	851,461	47	114
Snow 30	22434.6 s	17939.3 s	667,239	780,719	21	137
Teapot 15	3976.0 s	3089.9 s	353,480	402,145	55	138
Tetra 10	16791.8 s	12182.6 s	1,049,047	1,668,085	57	148
Gears 15	3475.4 s	3102.9 s	320,312	327,640	56	92
Gears	10952.4 s	5543.6 s	697,660	643,389	57	148

Table 1: Rendering times for different images.

Scene	Cell density objs/cell	Avg. non-empty cells visited per ray		Attempted intersections in non-empty cells	
		MSET	ASET	MSET	ASET
Snow 100	1.3	18	1	21	1
Snow 50	2.1	32	1	64	1
Snow 30	3.9	43	1	162	1
Teapot 15	4.6	8	1	36	1
Tetra 10	11.4	5	1	61	1
Gears 15	6.8	3	1	18	1
Gears	6.8	3.5	1	25	1

Table 2: Cell density and attempted intersections.

In Table 1, we have shown some of our results for MSET and ASET implementation. Frequently there are more rays being traced in ASET. For all scenes, ASET traces more rays per second than MSET.

In MSET, on the average more number of ray-polygon intersections are performed and more non-empty cells are traveled before the nearest point of intersection is found (Table 2).

In Table 3, the hit ratio, total intersections attempted and memory usage for MSET and ASET is given. The ASET implementation uses only 20-30 percent more memory than MSET. Hit ratio is higher in ASET than MSET for any grid size. The total number of intersections attempted with the objects in the non-empty cells are lower in ASET than the MSET, contributing towards better hit ratio.

In Table 4, number of primary and secondary rays are given. In ASET number of secondary rays are twice that of MSET. We noticed that some rays are trapped inside the boxes and do not contribute significantly to the overall intensity of the pixel; but still generates several reflected rays. To decrease the number of rays in ASET, these rays were curtailed[8].

In conclusion, our experiments show that any combination of the following would allow for faster ASET rendering times than MSET:

1. More complex scene (more objects) are rendered.
2. Grid size is reduced.
3. More rays are to be generated as the maximum height of the ray tracing tree is increased.

Future Directions

We must also focus on decreasing the overhead associated with the tracing of a ray and its children. In today's space subdivision techniques for ray tracing, this appears to be the area that takes up the majority of the processing time.

[8]The cut off criterion depends upon the distance traveled by the ray before it hits another cell [2]. If this distance is more than certain percentage of the diagonal distance of the voxel cell then the ray is not pursued further.

Scene	Hit Ratio		Total intersection attempted		Memory in bytes	
	MSET	ASET	MSET	ASET	MSET	ASET
Snow 100	4.8%	10.7%	6,421,644	4,541,077	5,808,708	7,488,824
Snow 50	1.9%	3.5%	19,367,605	13,241,396	906,864	1,146,980
Snow 30	0.6%	1.2%	51,621,849	36,169,829	311,362	390,404

Table 3: Snow Scene Analysis.

Scene	Primary	MSET (Secondary)	ASET (Secondary)
Snow 100	511,547	155,692	330,336
Snow 50	511,547	155,692	339,914
Snow 30	511,547	155,692	269,172
Teapot 15	244,291	109,189	157,854
Tetra 10	245,449	803,598	1,422,636
Gears 15	260,088	60,224	67,552

Table 4: Total Number of Rays.

Acknowledgements

Special thanks are due to Jim Duke for providing us with the basic ray tracing package, and, James Logan for his MSET implementation. Tracy Vanderventer and David Von Bank provided us the input/output and display routines. We would also like to thank Daven Kalra for sending us the Eric Haines's Data base for standard images.

References

1. (Glassner 1989) Glassner Andrew S. (editor) , "An Introduction to Ray Tracing", Academic Press, 1989.
2. (Dauenhauer 1989) David E. Dauenhauer, "Approximate Slicing Extent Technique (ASET) for Ray Tracing," M.S., University of Colorado at Colorado Springs, Summer 1989.
3. (Fujimoto, 1986) Fujimoto, A., Tanaka, T., and Iwata, K., "ARTS: Accelerated Ray-Tracing System," IEEE CG&A, April 1986, pp. 16-26.
4. (Glassner, 1984) Glassner, A. S., "Space Subdivision for Fast Ray Tracing," IEEE CG&A, October 1984, pp. 15-22.
5. (Haines, 1987) Haines, E.A., "A Proposal for Standard Graphics Environments," IEEE CG&A, November 1987, pp. 3-5.
6. (Kessener, 1985) Kessener, L.R.A., Peters, F.J., van Lierop, M.L.P., Data Structures for Raster Graphics, Chapter 4, 1985.
7. (Logan, 1989) Logan, J.R., "The Modified Slicing Extent Technique for Space Subdivision While Ray Tracing," M.S., University of Colorado at Colorado Springs, 1989.
8. (Semwal, 1987) Semwal, S.K., "The Slicing Extent Technique for Fast Ray Tracing," Ph. D. Dissertation, University of Central Florida, Orlando, Summer 1987.
9. (Tamminen, 1981) Tamminen, M., The EXCELL method for efficient geometric access to data, Thesis, Acta Polytechnica Scandinavica, Mathematics and Computer Science. Series no. 34, Helsinki, 1981. (Indirect Reference).
10. (Whitted, 1980) Whitted, T., "An Improved Illumination Model for Shaded Display," ACM Graphics and Image Processing, June 1980, Volume 23, Number 6, pp. 343-349.
11. (Yossef, 1986) Yossef, S., "A New Algorithm for Object Oriented Ray Tracing," Computer Vision, Volume 34, 1986, pp. 125-127.

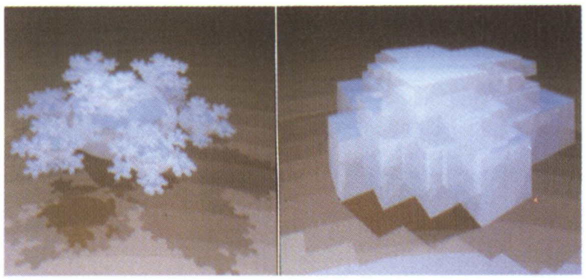

Figure 3: MSET (left) and ASET (right). ASET is used for the primary rays.

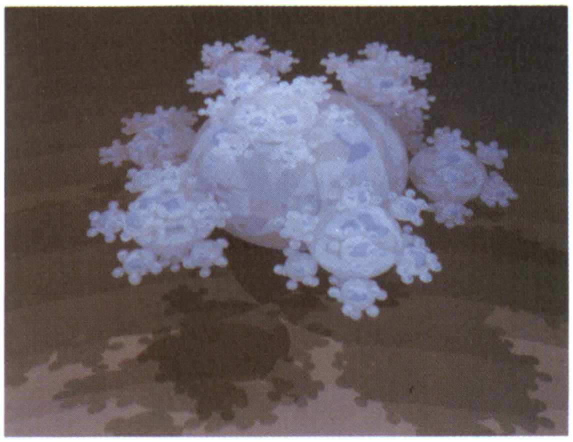

Figure 6: ASET. Strange Effects. Skull.

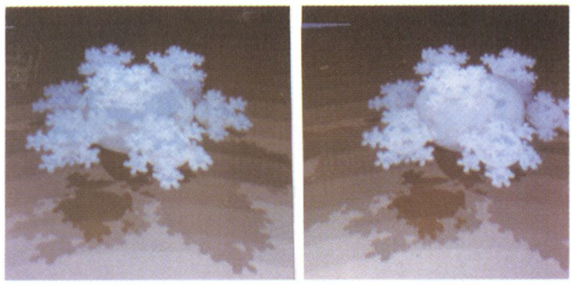

Figure 4: MSET (left) and ASET (right) for grid size of 100.

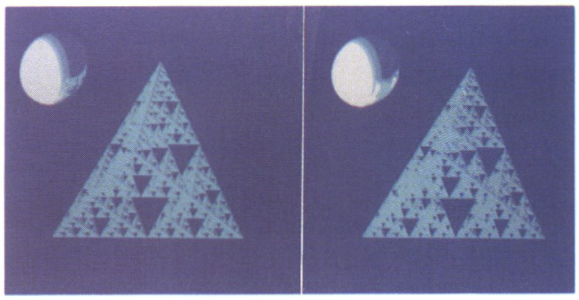

Figure 7: MSET (left) and ASET (right). Tetra and a Moon.

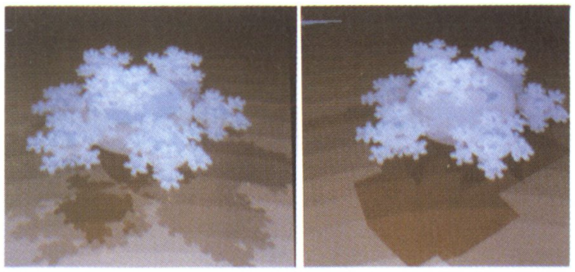

Figure 5: MSET (left) and ASET (right). ASET used for tracing shadow rays.

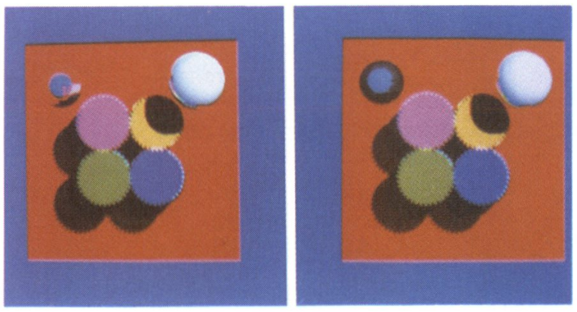

Figure 8: MSET (left) and ASET (right). Gears.