Profiling Graphic Display Systems

Peter Schoeler

Alias Research Incorporated
Toronto, Ontario
and
Department of Computer Science
University of Toronto

Alain Fournier

Computer Systems Research Institute
Department of Computer Science
University of Toronto
Toronto, Ontario
M5S 1A4

ABSTRACT

Graphics systems using three dimensional models, and computing a colour shaded image for a raster display are very common, and range widely in performance and cost. Despite the numerous variations in rendering techniques, visibility determinations, illumination models and modelling primitives manipulated, it is important to be able to compare them when rendering similar scenes.

We present here the first results of a series of profiling of different rendering systems displaying the same scenes on the same machine. The systems studied are a ray-caster, a system using a depth-buffer for visibility determination, and a system using a scan-line Watkins algorithm. The first and last systems have an antialiasing option. Two types of scenes were used, one made of a constant number of polygons varying in size, and the other made of parametric surfaces varying in level of subdivision.

The results, mainly useful for relative comparisons, confirm some predicted behaviours. The depth-buffer algorithm degrades considerably when the depth complexity increases. The ray-caster is not much influenced by the number of polygons, but by the total number of pixels covered. The most striking result is the large proportion of time spent on shading. It is a strong indication that work on ways to make shading computations less expensive, and to design special hardware for that purpose would be fruitful.

KEYWORDS: display systems, rendering techniques, profiling, shading, visibility determinations.

RESUME

Les systèmes graphiques qui utilisent des modèles à trois dimensions et qui produisent des images ombrées en couleur pour des affichages *rasters* sont maintenant très répandus et diffèrent énormément en puissance et en coût. En dépit des grandes variations dans les techniques de détermination de visibilité, les techniques d'ombrage et les techniques de modelage qu'ils utilisent, il est important de pouvoir comparer leurs performances quand ils rendent la même scène.

Nous présentons ici les premiers résultats d'une série de profilage de différents systèmes d'affichage produisant les mêmes scènes sur la même machine. Les sytèmes étudiés sont un *lanceur de rayon*, un système utilisant une *mémoire de profondeur* pour déterminer la visibilité, et un système utilisant l'algorithme de Watkins avec la conversion en ligne de balayage. Le premier et le dernier système ont tous les deux une option d'*antialiasing*. Deux genres de scènes ont été utilisées. Un était fait d'un nombre constant de polygones dont seule la taille changeait, et l'autre de surfaces paramétriques à des niveaux variés de subdivision.

Les résultats, surtout utiles pour des comparaisons relatives, confirment beaucoup de prévisions. La performance de l'algorithme de mémoire de profondeur se dégrade de façon considérable quand augmente la complexité de profondeur. Le lanceur de rayon n'est pas très influencé par le nombre de polygones, mais plutôt par le nombre total de pixels recouvertes. Le résultat le plus frappant est la grande proportion de temps consacrée aux calculs d'ombrage. C'est une forte indication du fait que plus de recherches pour améliorer l'efficacité de ces calculs et pour développer du matériel pour cet effet pourrait s'avérer payant.

MOTS CLES: systèmes d'affichage, techniques de rendu, profilage, ombrage, détermination de visibilité.

1. Motivations

A graphic display system, in the context of this study, is a combination of hardware and software which extracts object descriptions from an application database, applies geometric transformations to create instances of objects, determines their projections in a two-dimensional screen space, and computes the colour value of each pixel for the frame buffer of a raster display device. We will limit ourselves to the consideration of systems which handle three-dimensional models of objects, and aim at a *realistic* picture. Even with these restrictions, there exist systems which vary in performance from real-time to real-long-time (several hundred hours per frame), and from a few thousand dollars to a few million.

A display system has three main components (note that we are not considering the interaction with the user in this

study). The first one is the modelling component, which is really part of the application. By modelling here we do not mean the designing and creation of the models, but their retrieval and/or generation on the fly. For example extracting polygons from the database, computing points on a parametric surface, generating stochastic data are all modelling operations. The second component involves geometric operations. This includes clipping, perspective transformations, mapping to the screen. Two other important parts of the geometric operations are visibility determinations and shading. They are classified within the geometric operations because they use directly the geometric properties of the objects and the scenes for their computations and none of the screen geometric properties. And finally the third component includes all the display operations. In a raster system they are mainly the "scan-conversions", the sampling and filtering operations, and writing out the image (to a file or directly to the frame buffer).

It is important to note that this subdivision is independant of the rendering scheme. For instance, consider a depth-buffer system and a ray-caster. They could have the same modelling primitives and operations, such as B-spline surfaces and adaptive subdivision; the geometric operations for the depth-buffer system are mainly as described above, and consist of ray intersection calculations and shading for the ray-caster; the display operations are scan conversions and depth comparisons for the depth-buffer, and distributing into "scan buckets", subdividing the screen, etc., for the ray-caster.

An indirect confirmation of the validity of this view is that when new algorithms or new hardware appear, they can easily be categorized following this scheme.

New modelling techniques are appearing regularly [FoFC82, Reev83, Gard84, Gard85]. In geometry, the basic operations do not change very much, but the shading techniques became more sophisticated and more expensive [Cook81]. The visibility problem remains a active area of research, and even more effort is expended to make it more difficult [Whit80]. In this respect ray-tracing and ray-casting are properly rendering methods, that is they involve the whole rendering scheme. Therefore they include the modelling, geometric and display operations. Recently the display side (notably the sampling and filtering operations) have received the most attention within that technique [Aman84, CoPC84, DiWo85, LeRU85].

Hardware developement, besides the wholesale implementation in hardware of the graphics display system for real-time flight simulators [Scha83], has seen attacks on specific components: the purely geometric operations [Clar82] or the scan conversion component [FGHS85]. The design of specialized hardware for modelling, especially complex modelling, has been only started [PiFo84]. Notable by its absence is the lack of hardware design for shading.

For most systems the goal is the greatest amount of realism for the least cost (in time and hardware to run it on). Given this, it is surprising that the literature is not more abundant on the performance evaluation of such systems. The information available so far, besides various raw timings for pictures ("Figure X took 450 hours of Vax time") is limited to profiling results on one particular system [ReBl85] and numbers and analysis of the performance of visibility determination algorithms [SuSS74]. Crow compared the times spent on modelling, geometric operations, shading and filtering [Crow81], which was mainly oriented towards a comparison of the latter operations.

While most of the work in performance analysis bore on visibility determination, there was mounting evidence that the cost of modelling, and even more shading was rapidly getting larger. Already Crow pointed out that trend in [Crow81]. The result of that is that we have to consider carefully the illumination models and the shading methods, especially as they relate to the visibility algorithms and the display operations. A fast visibility algorithm will degrade in performance if the depth complexity increases and it continues computing the shade for many invisible areas. At this point, an algorithm that computes the shading only for the visible surfaces might win, even if the visibility determination is less efficient.

The first task in comparing various systems is choosing the scene they will be run on. Here again it is a fairly complex problem, with not as many published results as its importance and interest require. Kaplan and Greenberg [KaGr79] and Parke [Park80] addressed the problem for the analysis of depth buffer algorithms in conjunction with various processor architectures. Schmitt [Schm81] did the same, but this time to determine empirically the complexity of various visibility algorithms. More recently Whelan [Whel85] considered the problem again within the context of multiprocessor architectures.

Of course the problem of choosing the right test data is not unique to graphics. The problem here is twofold. One problem is to determine how to measure scene characteristics, and the other is to decide what are the characteristics of "typical" scenes.

2. Methodology

For this first report we tried to keep the number of variables under control, but to have enough variety and relevance to be of use to practitioners. The tactic we have adopted is to have three different renderers displaying the same scenes on the same hardware. The difference between the renderers is mainly in their methods to determine visibility.

The first renderer is a ray-caster, which we will call RC[1]. To speed up ray intersection, it subdivides screen space into buckets, and each polygon is added to a bucket list if its bounding box intersects the bucket. For each ray only the polygons listed with the bucket interesected by the ray are examined. It has an antialiasing mode, where pixels are adaptively subdivided if the shades at each corner differ by more than a given threshold. It can adaptively

1. The ray-caster is based on software originally written by Mike Sweeney at the University of Waterloo Computer Graphics Laboratory. An improved version is now a component of the Alias 1 rendering module.

subdivide parametric surfaces that way, but this was not used here to allow easier comparisons.

The other two renderers share the same front end, known as *3d* [AmBr86] at the University of Toronto Dynamic Graphics Project. The second renderer, which we will designate as DB, uses a file depth-buffer to determine visibility. It does not have an anti-aliasing option.

The third renderer uses Watkins algorithm [Watk70] to compute a scanline per scanline visibility. We will call it WS. It has an antialiasing option which uses the full precision in the X direction, and four subscanlines in the Y direction.

They both use a variety of rendering options, with a choice of illumination model, and include texture mapping, except for DB.

2.1. Modelling Primitives

Since the systems have to render the same scenes, they will have to use the same modelling primitives. They are polygons and B-splines patches, the most prevalent in current practice. The scene description actually is defined in a *scene description language*, and various filters generate the files for each renderers.

2.2. Geometric Primitives

Even though it was not mandatory for this study, the three systems all use triangles internally as geometric primitives.

	P1	P2	P3
Modelling Polygons	244	244	244
Geometric Triangles	488	488	488
Effective Triangles	485	485	482
Average Depth	0.60	2.32	8.32
Average Pixels Covered	321	1253	4526
Average Area	326	1313	5376

Table 1. Scene characteristics for polygons

2.3. Display Primitives

The three renderers produce a raster image, and were all set to output the image to a run-length encoded file. They therefore have basically the same output method. They were set to output a 512x512 image of 8 bit each of red, green and blue pixels.

3. Scene Characteristics

In order to isolate only a few variable, we decided to keep the number of modelling primitives constant for each series of scenes. In the first series, we distributed 40 cubes (6 faces each) roughly uniformly over the window. The spacing was chosen so that there was little overlap between cubes. Then for the subsequent scenes the cubes were linearly doubled around their centres so that the depth complexity, the average area of the polygons and the number of covered pixels all increased regularly.

In the second series, we designed a "glass" made of a 6 by 6 array of B-spline patches, and made three copies of it. There are therefore 3 primitives if primitives are control point networks, but 108 primitives if each patch is considered a primitive. The level of subdivision was set at 2, 4, 8 and 16 segments to a side. In this series the depth complexity and the total number of pixel covered is practically constant. The number of geometric primitives increases and the average size of each decreases to keep the product almost constant. Table 1 and 2 gives the main numbers for each series. Figures 1 to 3 and 4 to 6 are line drawings of the first six scenes.

The statistics given here were chosen to indicate the complexity of the scene. The depth complexity is the average number of object in one pixel, and will allow to gauge the efficiency of visibility determination and of shading. The average area of the polygons, computed analytically from the screen coordinates of the vertices, will help in determining the "polygon set-up time" *vs* the cost of pixel calculations. A pixel is deemed covered by a polygon if its center is inside the polygon. For the scenes used here the last two numbers are almost equal, but as the polygons become thinner, the difference can become important. Other statistics which are not included here can also be important. The number of edges, and the number of pixels containing edges is an example. In further studies about the role of filtering and antialiasing, we will have to consider them, as well as distinguish between silhouette edges and internal edges. If the scenes are used to test parallel algorithms, the distribution of the primitives in space, and their aspect ratio should be taken into account.

	V1	V2	V3	V4
Modelling Patches	108	108	108	108
Geometric Triangles	864	3744	15552	63360
Effective Triangles	864	3744	15552	63358
Average Depth	0.61	0.59	0.59	0.59
Average Pixels Covered	185	41	9	2
Average Area	185	41	10	2.4

Table 2. Scene characteristics for B-splines

The scenes were all lit by three local light sources. The illumination model used was the same across systems, being the Lambert cosine law for the polygons, and Phong illumination model for the patches, with a *shinyness* of 50. The backfacing polygons were not culled, and every polygon was uniformly shaded (no Gouraud shading).

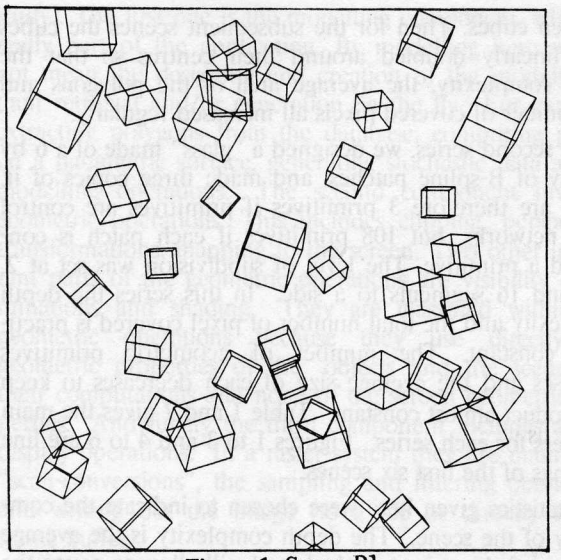

Figure 1. Scene P1

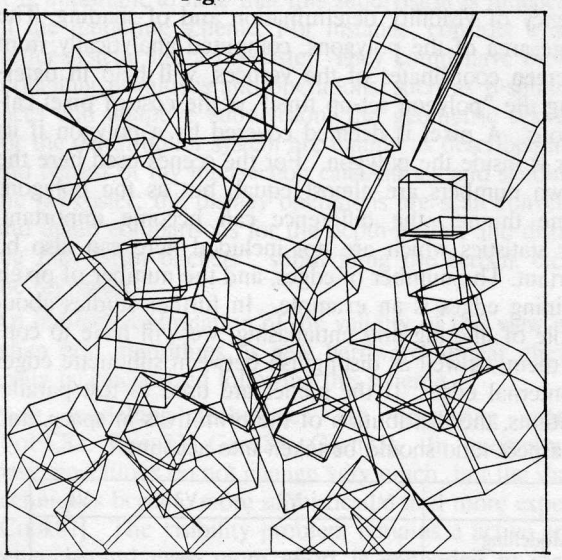

Figure 2. Scene P2

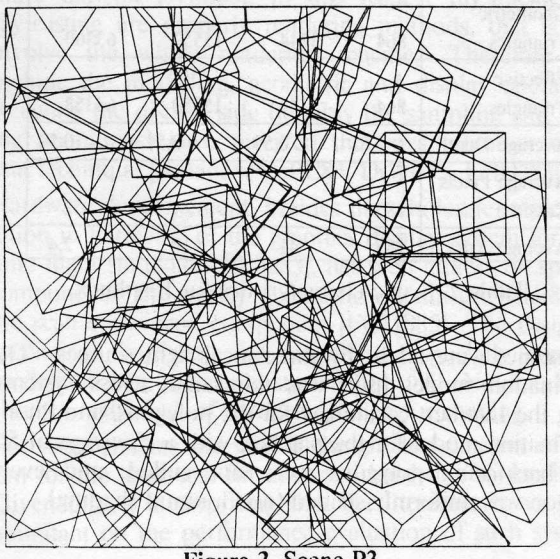

Figure 3. Scene P3

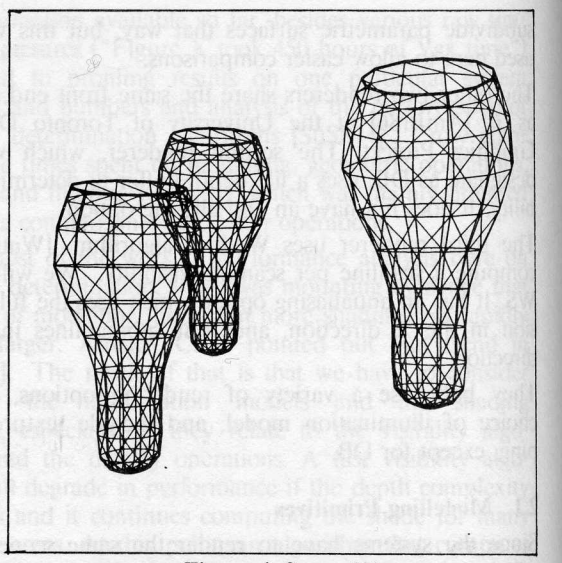

Figure 4. Scene V1

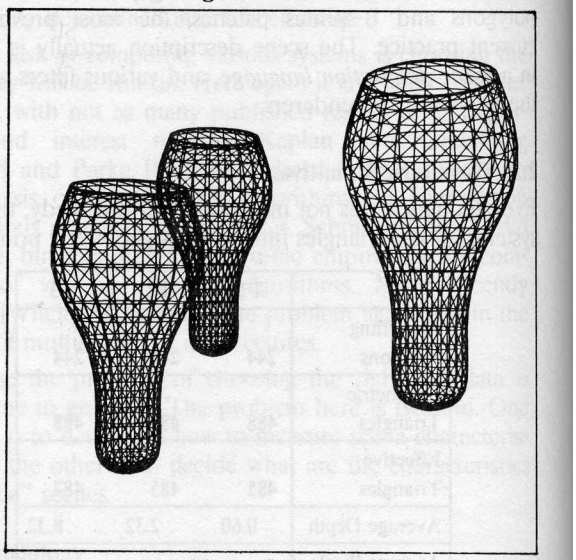

Figure 5. Scene V2

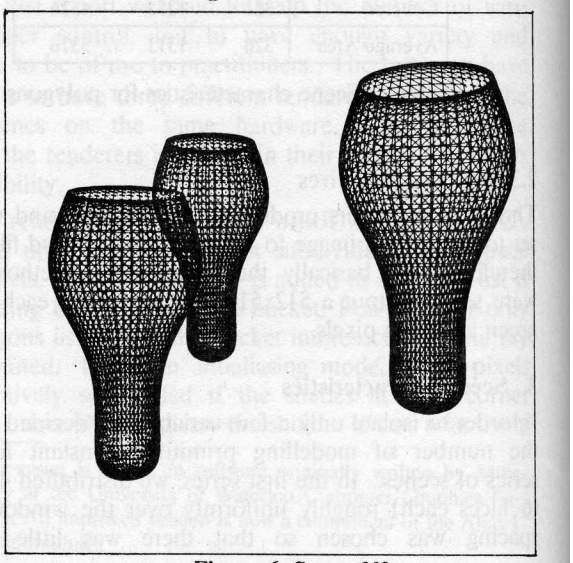

Figure 6. Scene V3

4. Hardware and operating system characteristics

All three systems were run on a *Celerity C1200* with 4MB of main memory and three 120 MB disks[2]. The processor also has a 64K cache. The operating system was Unix† version 4.2 BSD. The three systems were compiled with the standard C compiler, using the profiling, optimization and hardware floating point options. The only difference was that RC had to use the option which prevents single floats to be cast into double, and the other two could not be run with that option. It is one more reason to be cautious about any comparison of the *absolute* times.

5. Results

Each of the seven scenes were displayed 5 times (RC, RC antialiased, DB, WS, WS antialiased). The last two scenes did not run with WS, because the allocated memory was not big enough. In the interest of brevity, we will only give two tables and three plots.

The time directly spent on I/O was not included in the tables, but is fairly constant for each system across scenes.

The first thing to note is that modelling plays no role in the first series of scenes, and the geometric transformations play little role. The dominant factors are shading and visibility determinations. In fact shading takes from 20 to more than 90% of the total time. The plots of Figures 7 and 8 show the times for visibility determination and shading for all five renderings as a function of the average polygon coverage. The plot of Figure 7 shows that the cost of visibility determination continues to climb briskly for the ray-caster even in the 5000 polygon range. It is clear from the plot of Figure 8 that DB pays the price for shading many non-visible areas as the depth complexity increases. Since all the other systems tend to flatten out as the depth complexity increases, the depth-buffer is the worse from the middle of the range explored here.

Figure 9 gives the plot of the times for visibility determination and shading for RC, RCa and DB. The main features of the statistics for the series of patch scenes are that while shading is still an important factor, the visibility determination becomes more important for the non ray-casting systems as the polygons get smaller. In fact, as expected, the RC and RCa are relatively insensitive to the size of the polygons, especially for the shading. The growth of the cost of visibility determination is less than could have been expected. In fact the ray-caster is a winner from around 5000 triangles (remember the warning against taking these absolute numbers too seriously). For the first time the cost of modelling begins to be felt, in particular for RC in scene V4. But it should be stressed that we are using fairly simple primitives here. It should also be kept in mind though that sooner or later the storage requirements will hinder RC, and they prevented us to run WS and WSa on the last two scenes.

6. Conclusions

These results are only a small sample of the profiling done so far. We tried here to define two series of scenes and choose three renderers so that the number of independent variables was relatively small. Most numbers confirmed our prejudices. Renderers using depth buffers do poorly when the depth complexity increases (here it had problems above 2) and ray-casters do well when the polygons become small. They confirmed that shading is a large part of the cost of the rendering, and it is therefore important to help with efficient routines and specialized hardware. It is important to note that DB and WS spend 10% or more of their time doing vector normalization. The computer used has hardware square root, so it was not as much a factor as it usually is in that type of programs.

The data for antialiasing is also mainly indicative. For both RC and WS antialiasing about doubles the cost and the increase comes mainly from more shading computations. We did not study here the relative cost of different illumination models and shading techniques, but we will do so as part of this study. Considering the high cost of shading, it has a significant impact on the total cost.

This brings up the issue of the quality of the picture. Of course most of these costs are assumed in the belief that a better picture will ensue. Our test pictures were as identical as we could expect, and therefore are not much help in this respect. We plan to complete a similar study with complex objects that have been modelled for other purposes with shophisticated shading and up to several hundred thousand polygons. Then the picture quality, especially as it relates to antialiasing will have to be judged subjectively.

Acknowledgements

We wish to acknowledge the support of the National Science and Engineering Research council. We also want to thank Alias Research Corporation for making available the computers and the modelling systems to create the scenes and run the profilings. John Amanatides and Tom Nadas wrote the various versions of 3D, and helped considerably in making them fit to our demands.

2. All the profiling was done on one disk 90% full.

† Unix is a trademark of AT&T Bell Laboratories

Times	RC	%	RCa	%	DB	%	WS	%	WSa	%
P1										
Total	152	100	409	100	181	100	120	100	234	100
Geometry	6	3	6	1	2	1	2	1	2	0
Visibility	81	53	292	71	5	2	11	9	34	14
Shading	50	32	96	23	167	92	103	85	194	82
P2										
Total	346	100	624	100	542	100	292	100	497	100
Geometry	12	3	12	1	2	0	2	0	2	0
Visibility	184	53	409	65	8	1	21	7	73	14
Shading	135	39	188	30	529	97	265	90	418	84
P3										
Total	614	100	794	100	964	100	365	100	645	100
Geometry	32	5	33	4	2	0	2	0	2	0
Visibility	403	65	568	71	8	0	43	11	161	24
Shading	164	26	178	22	950	98	316	86	477	73

Table 3. Some statistics for the polygon scenes

Times	RC	%	RCa	%	DB	%	WS	%	WSa	%
V1										
Total	156	100	307	100	176	100	166	100	338	100
Modelling	1	0	1	0	1	0	1	0	1	0
Geometry	6	3	6	1	0.2	0	0.2	0	0.2	0
Visibility	79	50	215	70	12	6	36	21	102	30
Shading	63	40	77	25	162	92	129	77	235	69
V2										
Total	168	100	320	100	204	100	240	100	538	100
Modelling	4	2	3	0	2	0	2	0	2	0
Geometry	10	5	9	2	0.2	0	0.2	0	0.2	0
Visibility	85	50	223	69	27	13	103	42	249	46
Shading	62	36	73	22	175	85	134	55	286	53
V3										
Total	213	100	378	100	287	100	-	-	-	-
Modelling	13	6	13	3	5.5	1	-	-	-	-
Geometry	33	15	33	8	0.2	0	-	-	-	-
Visibility	98	46	249	65	84	29	-	-	-	-
Shading	62	29	74	19	196	68	-	-	-	-
V4										
Total	376	100	591	100	574	100	-	-	-	-
Modelling	46	12	45	7	23	4	-	-	-	-
Geometry	121	32	122	20	0.2	0	-	-	-	-
Visibility	136	36	336	56	303	52	-	-	-	-
Shading	63	16	78	13	247	43	-	-	-	-

Table 4. Some statistics for the patch scenes

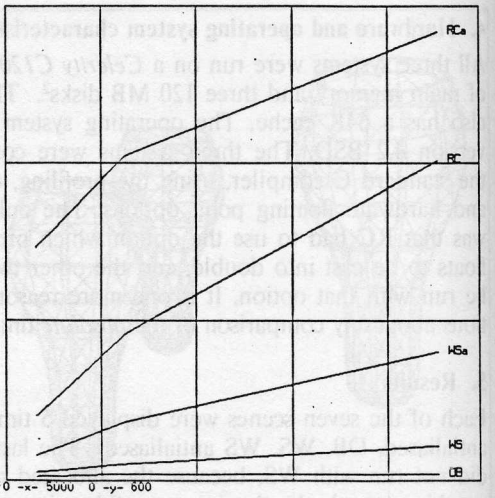

Figure 7. Visibility determination vs polygon coverage

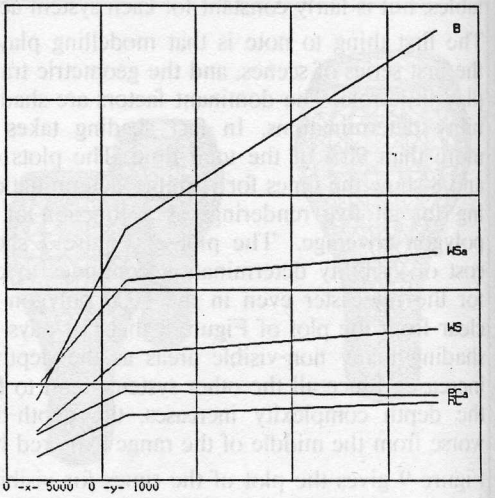

Figure 8. Shading vs polygon coverage

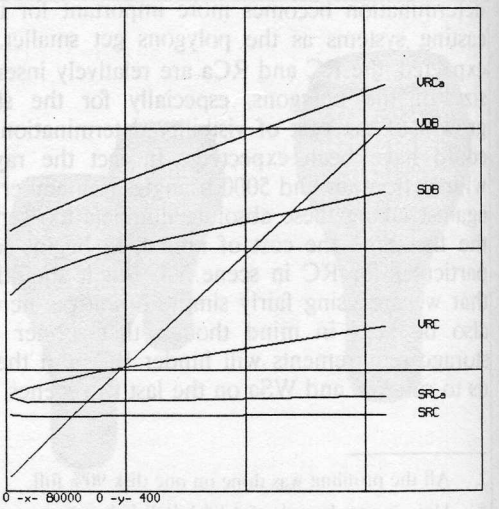

Figure 9. Visibility and shading vs subdivision

References

[Aman84]
Amanatides, J., "Ray Tracing with Cones", in *Proceedings of SIGGRAPH'84*, also published as Computer Graphics, 18, 3, (July 1984), 129-135.

[AmBr86]
Amanatides, J. and Brown, E., *3d User's Manual*, Dynamic Graphics Project, CSRI, University of Toronto, (January 1986).

[Cook81]
Cook, R. L. and Torrance, K. E., "A Reflectance Model for Computer Graphics", in *Proceedings of SIGGRAPH'81*, also published as Computer Graphics, 15, 3, (July 1981), 307-316.

[Clar82]
Clark, J. H., "The Geometry Engine: A VLSI Geometry System for Graphics", in *Proceedings of SIGGRAPH'82*, also published as Computer Graphics, 16, 3, (July 1982), 127-133.

[CoPC84]
Cook, R. L., Porter, T. and Carpenter, L., "Distributed Ray-Tracing", in *Proceedings of SIGGRAPH'84*, also published as Computer Graphics, 18, 3, (July 1984), 137-145.

[Crow81]
Crow, F. C., "A Comparison of Antialiasing Techniques", IEEE Computer Graphics and Applications, 1, 1, (January 1981), 40-48.

[DiWo85]
Dippe, M. A. Z. and Wold, E. H., "Antialiasing through Stochastic Sampling", in *Proceedings of SIGGRAPH'85*, also published as Computer Graphics, 19, 3, (July 1985), 69-78.

[FGHS85]
Fuchs, H., Goldfeather, J., Hultquist, J. P., Spach, S., Austin, J. D., Brooks, F. P., Eyles, J. G. and Poulton, J., "Fast Spheres, Shadows, Textures, Transparencies and Image Enhancements in Pixel-planes", in *Proceedings of SIGGRAPH'85*, also published as Computer Graphics, 19, 3, (July 1985), 111-120.

[FoFC82]
Fournier, A., Fussell, D. and Carpenter, L. "Computer Rendering of Stochastic Models", Comm. ACM, 25, 6, (June 1982).

[Gard84]
Gardner, G. Y., "Simulation of Natural Scenes Using Textured Quadric Surfaces", in *Proceedings of SIGGRAPH'84*, also published as Computer Graphics, 18, 3, (July 1984), 11-20.

[Gard85]
Gardner, G. Y., "Visual Simulation of Clouds", in *Proceedings of SIGGRAPH'85*, also published as Computer Graphics, 19, 3, (July 1985), 297-303.

[KaGr79]
Kaplan, M. and Greenberg, D. P., "Parallel Processing Techniques for Hidden Surfaces Removal", in *Proceedings of SIGGRAPH'79*, also published as Computer Graphics, 13, 3, (July 1979), 300-307.

[LeRU85]
Lee, M. E., Redner, R. A. and Uselton, S. P., "Statistically Optimized Sampling for Distributing Ray Tracing", in *Proceedings of SIGGRAPH'85*, also published as Computer Graphics, 19, 3, (July 1985), 61-67.

[Park80]
Parke, F. I., "Simulation and Expected Performance Analysis of Multiple Processor Z-buffer Systems", in *Proceedings of SIGGRAPH'80*, also published as Computer Graphics, 14, 3, (July 1980), 48-56.

[PiFo84]
Piper, T. and A. Fournier, "A Hardware Stochastic Interpolator for Raster Displays", in *Proceedings of SIGGRAPH'84*, also published as Computer Graphics, 18, 3, (July 1984), 83-91.

[ReBl85]
Reeves, W. T. and Blau, R., "Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems", in *Proceedings of SIGGRAPH'85*, also published as Computer Graphics, 19, 3, (July 1985), 313-322.

[Reev83]
Reeves, W. T., "Particle Systems- A Technique for Modeling a Class of Fuzzy Objects", in *Proceedings of SIGGRAPH'83*, also published as Computer Graphics, 17, 3, (July 1983), 359-376.

[Scha83]
Schachter, B. J. (Ed.), *Computer Image Generation*, Wiley & Sons, (1983).

[Schm81]
Schmitt, A., "Time and Space Bounds for Hidden Line and Hidden Surface Algorithms", *Proceedings of Eurographics'81*, North-Holland, (1981), 43-56.

[SuSS74]
Sutherland, I. E., Sproull, R. F. and Schumacher, R. A., "A Characterization of Ten Hidden-Surface Algorithms", ACM Computing Surveys, 6, 1, (March 1974), 1-55.

[Watk70]
Watkins, G. S., *A Real-Time Visible Surface Algorithm*, UTECH-CSC-70-101, Department of Computer Science, University of Utah, (June 1970).

[Whel85]
Whelan, D. S., *A Multiprocessor Architecture for Real-Time Computer Animation*, Computer Science Department, California Institute of Technology, CITCS 5200:TR:85, (May 1985).

[Whit80]
Whitted, J. T., "An Improved Illumination Model for Shaded Display", CACM, 26, 6, (June 1980), 343-349.