CASTING CURVED SHADOWS ON CURVED SURFACES

Lance Williams
Computer Graphics Lab
New York Institute of Technology
Old Westbury, New York 11568

Abstract

Shadowing has historically been used to increase the intelligibility of scenes in electron microscopy and aerial survey. Various methods have been published for the determination of shadows in computer synthesized scenes. The display of shadows may make the shape and relative position of objects in such scenes more comprehensible; it is a technique lending vividness and realism to computer animation.

To date, algorithms for the determination of shadows have been restricted to scenes constructed of planar polygons. A simple algorithm is described which utilizes Z-buffer visible surface computation to display shadows cast by objects modelled of smooth surface patches. The method can be applied to all environments, in fact, for which visible surfaces can be computed. The cost of determining the shadows associated with each light source is roughly twice the cost of rendering the scene without shadows, plus a fixed transformation overhead which depends on the image resolution. No extra entities are added to the scene description in the shadowing process. This comprehensive algorithm, which permits curved shadows to be cast on curved surfaces, is contrasted with a less costly method for casting the shadows of the environment on a single ground plane.

In order to attain good results, the discrete nature of the visible-surface computations must be treated with care. The effects of dither, interpolation, and geometric quantization at different stages of the shadowing algorithm are examined. The special problems posed by self-shadowing surfaces are described.

Key words: shadows, hidden surface algorithms, computer animation, computer graphics.

CR classification: 8.2

Introduction

The Z-buffer visible surface algorithm, first published by Catmull [1], was the first method to make possible computer generated shaded pictures of bicubic surface patches. The algorithm is extremely general and quite simple to implement but requires substantial memory.

A "frame buffer," in the current computer graphics parlance, is a memory that stores a complete digital picture. It may serve as an intermediary between the computer that produces the picture and a video driver which continuously refreshes a display. Some visible surface algorithms (e.g. [2]) require a frame buffer in order to compute an image. In this case, the frame buffer mediates the display process in a more substantial way.

The Z-buffer is an extension of this mass-memory approach to computer graphics which resolves the visible surfaces in a scene by storing depth (Z) values at each point in the picture. As objects are rendered, their Z values are compared at each point with the stored Z values to determine visibility. Since this determination requires only that a measure exist which orders the surfaces to be displayed, it is not too strong a statement to say that the Z-buffer algorithm provides a discrete solution to all scenes for which visible surfaces can be computed.

Z-buffer visible surface computation is of particular interest because it exhibits limiting-case properties [3]. The objects to be rendered do not have to be sorted beforehand, so indefinitely complex scenes can be handled. At the pixel level, the Z-buffer implicitly executes radix sorts in X and Y and simple indexing in Z. In X and Y, the sorts are bucket sorts, the special case of the radix sort where the radix encompasses the range of the keys, obviating all comparisons. In Z, the index of the sort is reduced to one, necessitating only a single comparison for each item.

Radix sorting is the only sorting method which grows only linearly in expense with the number of randomly-ordered items to be sorted, and the Z-buffer is the only visible surface algorithm the cost of which grows only linearly with the average depth complexity of the environment (that is to say, with the total screen area of all surfaces rendered, whether visible in the final image or not).

Thus the Z-buffer algorithm enjoys two key advantages over all other existing visible surface algorithms:

1. indefinitely large environments;

2. linear cost growth.

In addition, the final image computed has an asso-

ciated Z partition, a "depth map" [4] of the scene. This extra information permits a great many interesting post-processes on a computed image. Such algorithms are noteworthy because their expense does not vary with the size or complexity of the environment, but depends only on the image resolution. The shadow algorithm described here is one attempt to exploit the Z partition.

Shadow Information

The display of shadows may make the shape and relative position of objects in computer generated scenes more comprehensible. Shadows emphasize and may serve to clarify the three dimensional nature of the forms displayed.

The shadows cast by a point source of light onto a flat surface represent, like a perspective transformation, a projection of the scene onto a plane. This simplified situation offers a convenient way of understanding the information that shadows convey. A scene rendered with shadows contains two views in one image. If we are content to cast shadows on a single wall or ground plane, these two views are simple projections. In general, of course, shadows may fall across any surface in the scene. Two views are still sufficient to compute the shadows, however, if they are Z-buffer views.

The proposed algorithm works as follows:

1. A view of the scene is constructed from the point of view of the light source. Only the Z values and not the shading values need be computed and stored.

2. A view of the scene is then constructed from the point of view of the observer's eye. A linear transformation exists which maps X,Y,Z points in the observer's view into X,Y,Z coordinates in the light source view. As each point is generated in the observer's view, it is transformed into the computed view in the light source space and tested for visibility to the light source before computing its shading value. If the point is not visible to the light source, it is in shadow and is shaded accordingly.

Step (2) as defined is the "correct" form of the proposed algorithm, but in the ensuing discussion and pictures a modified procedure is assumed. The complete scene is computed from the observer's viewpoint, and the point-by-point transformation to the light source space and consequent shadowing is undertaken as a post-process. This modified algorithm incorrectly shades the hilights in the scene, since they appear in the shading process and then are merely darkened if they are found to lie in shadow; hilights should not appear in shadowed areas at all. The modified algorithm may also suffer more severely from quantization problems, since the Z coordinates of the visible points will have been quantized to the resolution of the Z buffer (16 bits in the cases illustrated here) before transformation. On the other hand, the expense of the transformation in the modified version does not depend on the complexity of the scene, as it does when all points are transformed as they are computed. Operating as a post process, the transformation is applied only to the points that are visible in the final picture. The expense is thus dependent only on the resolution of the image. Like most point-by-point operations, expense increases with the square of the resolution.

Limitations of Image Space

The generalization to curved surfaces and the linear cost growth which distinguish the proposed algorithm are both attributable to the fact that all computations are performed in image space. This approach carries with it certain limitations, however, which must be weighed against the advantages.

Since shadow determination is based on transformation between two images, the user must take care to ensure that all objects which may cast a shadow in the observer's image be within the field of view of the light source image. The assumption is that points transformed into the light source space which lie outside the viewing volume of the light source are illuminated. Shadows may only be cast within the viewing volume of the light source.

While it is not precisely true that the light source must lie outside the observer's field of view, it can cast shadows only within its own field. If a light source within the observer's viewing volume is to cast shadows in all directions, its sphere of illumination must be sectored into multiple views as suggested by Crow [5]. Computing these views in the Z-buffer is only slightly more expensive than computing a single view containing all the objects in the scene. Transforming points from the observer's image into the light source space becomes more expensive, however. Either each point must be transformed into each light source view (the correct approach in computing shadows for multiple light sources), or clipped against the light source viewing volumes in the observer's space and transformed into the coordinates of the light source view in which it falls. The major difficulty with this method is the increased memory required.

Severe perspective, either in the observer's view or required by a light source close to the scene, may increase the quantization problems attendant in transforming from one image to the other. In any case, quantization and aliasing are the chief drawback of image space algorithms. The aliasing problem must be addressed vigorously whenever image space techniques are applied. This is a large and complicated issue, outside the scope of this paper; for a general treatment of aliasing and visible surface algorithms, see [1], [6], and especially [7]. [3] will treat the special topic of aliasing, geometric quantization and the Z-buffer.

Self-shadowing surfaces rendered by the proposed technique constitute an excellent case study in image space sampling problems. When we transform a point from a surface in the observer's space onto a surface in the light source space, it should ideally lie right on the surface of which it is a part. Due to the imprecision of machine arithmetic and more particularly to the quantization of Z-buffer surfaces, it will fall above or below the surface. Since we want the point to appear illuminated if it lies on a visible surface, we subtract a bias from

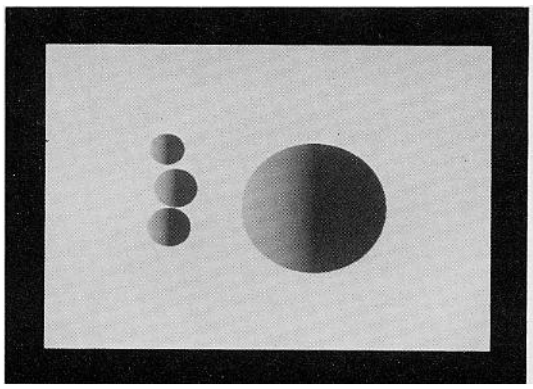

fig. 1
The observer's view of four spheres.

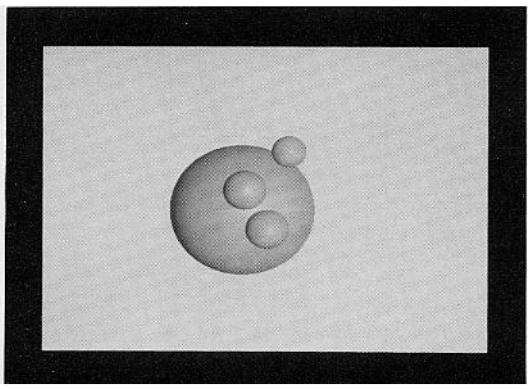

fig. 2
The four spheres viewed from the position of the light source.

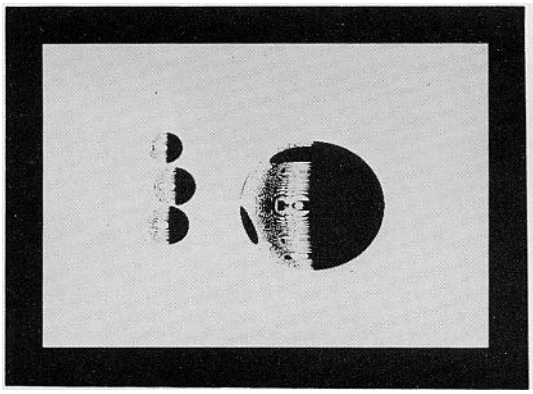

fig. 3
Shadow with reduced surface bias reveals quantization moire.

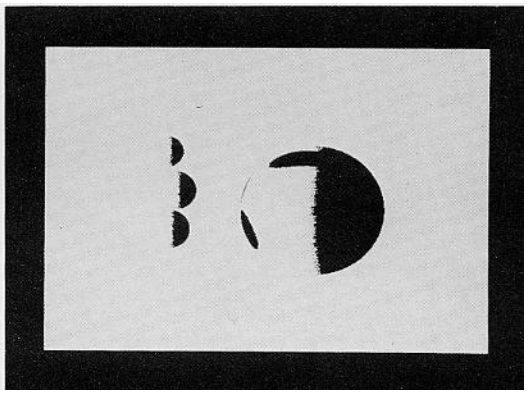

fig. 4
Increased bias and dither applied to shadow computation.

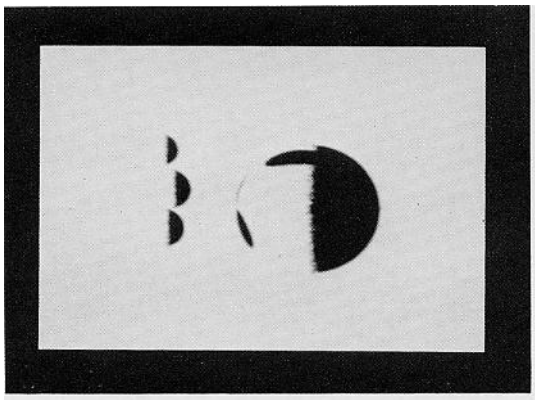

fig. 5
Low-pass filtering applied to shadow.

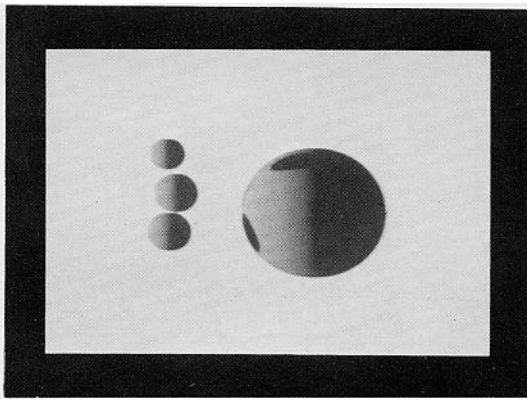

fig. 6
Shadow applied to the image of fig. 1

the Z value of the point after it has been transformed into the light source space (actually, of course, this bias is incorporated into the general linear transformation employed). The bias may move the shadow line slightly, but it has the desired effect of keeping surfaces from shadowing themselves where they are plainly visible to the light source.

As a surface curves smoothly away from the light, however, it must ultimately shadow itself. This is not a problem with polygons, the sharp edges of which make shadowing a rather sharply defined proposition. A smooth surface shadowing itself in a shallow curve may switch from light to dark on the strength of a least significant bit in the Z-buffer. Worse, it may switch back and forth as the quantizing error beats with the sampling grid, producing a vivid moire.

Figures 1 and 2 illustrate a simple scene from the point of view of an observer and from the point of view of the light source, respectively. The Z bias subtracted from the transformed points in computing the shadows of figure 3 has been deliberately reduced to reveal the quantizing moire. The light source is at infinity, rotated one hundred degrees about the vertical axis to the left of the observer's bearing. Note that the quantizing error is greatest in two areas: at the edge of the vertical solid shadow line, and in a dark curved band to the left of it. These correspond to, respectively, the right edge of the spheres in the light source view, and the right edge of the spheres in the untransformed observer's view. These edges are aliased in the original views, discrete periodic samples of non-bandlimited images. Another way of viewing the problem is that the edges are quantized in the original views, quantized to the nine bits of resolution available in X and Y. Clearly the problem is much greater than in Z, where sixteen bits are available. Unfortunately, extra lateral resolution is purchased at square law expense.

Bandlimiting the Z partition before sampling, if it were practical, would not improve matters. Z values at the edges of the spheres would be a smooth blend of the depth of the edge and the depth of the far clipping plane, meaningless points as far as the scene is concerned. The aliasing effects observed are local, however, and it is reasonable to treat the smooth surfaces within the ragged edges as correctly sampled. This assumption implies that a filter to reconstruct the surface between samples is in order. Indeed, interpolating the Z values of the light source image to derive Z values at the exact X,Y coordinates of the transformed observer points (rather than using the Z value of the nearest neighbor for comparison) improves matters somewhat, reducing shadow noise in the form of isolated pixels.

Treating shadow noise as a quantizing rather than an aliasing problem improves the image further. The error signal in a quantizing system correlates quite strongly with the signal. Addition of random noise in the range of a single quantum breaks up this correlation, reduces the resulting periodicities (moire) to which the human eye is so sensitive, and whitens the spectrum of the error. Figure 4 illustrates the sphere shadows with increased negative Z bias and bilinear interpolation of light source Z values to the X,Y of the transformed observer points, which have been dithered by the addition of normally distributed random values in the range -.5 to +.5. The shadow image is subsequently dejagged by an edge dequantizing filter similar to one advanced by Freeman [8], then low pass filtered to further smooth the contours and merge the dithered edge of self shadowing (figure 5).

As a final, not unimportant observation, figure 6 illustrates that the problem of self shadowing surfaces may not be terribly significant in practice. Figures 3 through 5 were of the computed shadows alone; figure 6 displays the shaded surfaces with their shadows. Shading the spheres according to the position of the light source casting the shadows causes a smooth shadow transition which obscures the quantization error. In practice, translucent shadows (their translucency corresponding to the additive "ambient" term in most surface shading formulations) generally look better than deep black shadows, and low pass filtering of the shadows before they are applied to the image subjectively approximates the soft penumbra cast by real light sources.

Conclusions

The algorithm described operates successfully on scenes of curved surface patches, and does so with a cost that increases only linearly with the complexity of the environment. The cost is roughly twice the cost of rendering the scene normally, plus the cost of transforming the points of the observer's image into the light source image. In the originally stated algorithm, the cost of transformation increases linearly with the depth complexity of the scene. The cost of transformation in a modified version of the algorithm which performs shadowing strictly as a post- process is fixed by the resolution of the screen, and corresponds to a scene with an average depth complexity of one. The rendering cost is only "roughly" twice the cost of rendering the scene normally, since the light source view requires no shading computation. Depending on the complexity of the shading rules applied [9], this may represent a substantial savings.

Speed does not directly correspond to "computational expense" when special hardware can be applied. The enormous interest in real-time graphics has led to the development of specialized transformation hardware, specifically, digital devices to multiply four by four transformation matrices by four element homogeneous point coordinates [10]. The modified version of the shadow algorithm, developed for animation purposes, is particularly suited to pipelining of the coordinate transforms. The intent at NYIT is to apply the Floating Point Systems AP120-B array processor to such problems.

The complexity of software necessary to implement the shadow algorithm is minimal if the necessary memory is available. Although it has long been suggested that two passes of a visible surface algorithm is sufficient to compute shadowing [5], relating the data provided by the two passes is very difficult for many algorithms. The Z-buffer provides a straightforward means of relating data in different views since the visible surfaces are three dimensional and hence subject to general

three dimensional transformations.

The bright outlook for memory technology bodes well for mass-memory graphics. The shadow algorithm discussed here is one simple example of a wide class of extremely general algorithms which exhibit very desirable cost growth properties. The challenge of this approach to computer graphics is to cope successfully with the problems posed by the discrete nature of image space scene representations.

References

[1] Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces," PhD. thesis, Dept. of Computer Science, University of Utah, 1974.

[2] Newell, M. G., Newell, R. G., and Sancha, T. L., "A Solution to the Hidden Surface Problem," Proceedings of the 1972 ACM National Conference.

[3] Williams, L., forthcoming PhD. thesis, University of Utah.

[4] For the application of this representation to scene analysis, see: Levine, M. D., O'Handley, D. A., and Yagi, G. M., "Computer Determination of Depth Maps," Computer Graphics and Image Processing, No. 2, 1973.

[5] Crow, F. C., "Shadow Algorithms for Computer Graphics," Siggraph 1977 Proceedings, Vol. 11, No. 2, Summer 1977.

[6] Blinn, J. F., "A Scan-Line Algorithm for the Display of Bicubic Surface Patches," PhD. thesis, Dept. of Computer Science, University of Utah, 1978.

[7] Crow, F. C., "The Aliasing Problem in Computer-Synthesized Shaded Images," PhD. thesis, Dept. of Computer Science, University of Utah, 1976.

[8] Freeman, H., "Computer Processing of Line Drawing Images," ACM Computing Surveys, Vol. 6, No. 1, March 1974.

[9] Blinn, J. F., "Models of Light Reflection for Computer Synthesized Pictures," Siggraph 1977 Proceedings, Vol. 11, No. 2, Summer 1977.

[10] Sutherland, I.E., "A Head-Mounted Three-Dimensional Display," Fall Joint Computer Conference 1968, Thompson Books, Washington, D.C., 757.

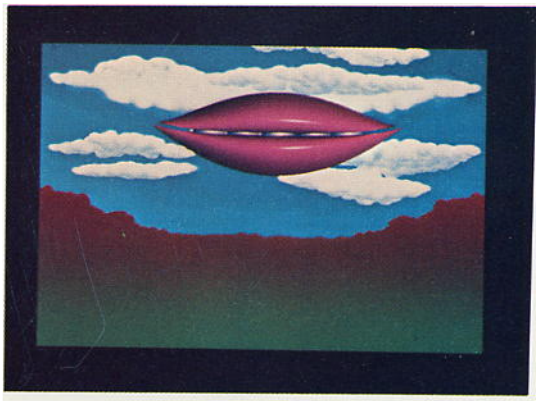

fig. 7a
3d smile sculpted by Alvy Ray Smith.

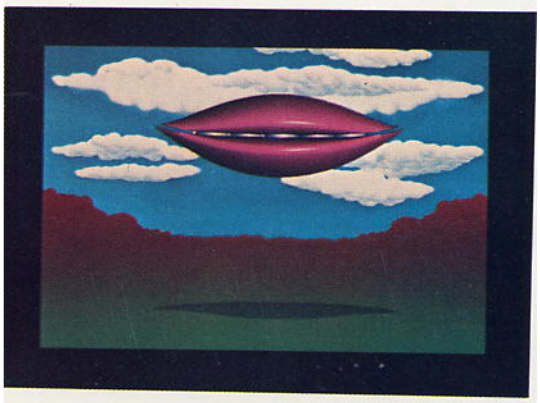

fig. 7b
evinces the shadow of a smile.

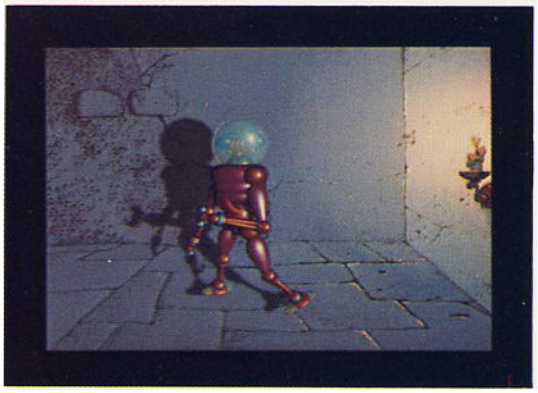

fig. 8
The robot casting his shadow on the wall and floor is composed of over 350 bicubic surface patches.

Especial thanks are due David DiFrancisco and Garland Stern for photographic assistance.