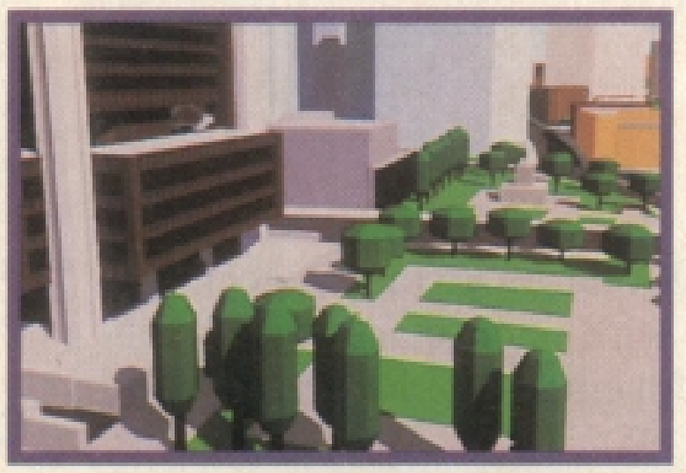
A General Version of Crow's Shadow Volumes

Philippe Bergeron, Digital Productions*

In 1977 Frank Crow introduced a new class of algorithm for the generation of shadows. His technique, based on the concept of shadow volumes, assumes a polygonal database and a constrained environment. For example, polyhedrons must be closed, and polygons must be planar. This article presents a new version of Crow's algorithm, developed at the Universite de Montreal, which attempts a less constrained environment.

The method has allowed the handling of both open and closed models and nonplanar polygons with the viewpoint anywhere, including any shadow volume. It does not, however, sacrifice the essential features of Crow's original version: penetration between polygons is allowed, and any number of light sources can be defined anywhere in 3D space, including the view volume and any shadow volume. The method has been used successfully in the film *Tony de Peltrie* and is easily incorporated into an existing scan-line, hidden-surface algorithm.



Shadows can be a problem in generating realistic images. They are visible from the viewpoint but invisible from the light source. Shadow casting vastly improves realism and gives useful information about relationships between objects, but the technique itself is extremely complex. It was for this reason that early implementations of shading algorithms eliminated the need for shadows, assuming that the light source was positioned at the observer's point of view. If more than one light source was used, the environment was generated as if it were a cloudy day with diffuse illumination.

These constraints were clearly unacceptable in some cases. Shadow casting was vital to applications that called for a realistic appearance, such as computer-animated sequences for flight simulation or for entertainment.

In 1977, Frank Crow documented existing shadow-generation methods and presented a technique for shadow casting that was based on shadow volumes.[1] He found that three classes of shadow generation methods were prevalent:

1. shadow computation during display
2. polygon shadow generation based on clipping transformation
3. shadow volumes

In 1978 Lance Williams introduced a fourth class:[2]

4. z-buffer shadow computation

and in 1980 Turner Whitted presented the fifth:[3]

5. generation of shadows using rays

*Digital Productions is a division of Omnibus Simulations. The work described in this article was done at Universite de Montreal in Canada.

0272-1716/86/0900-0017$01.00 © 1986 IEEE

Table 1. Classification of existing shadow algorithms.

Class No.	Description	Hidden-Surface Algorithm
1	Shadow computation during display	Scan-line, frame buffer
2	Polygon shadow generation with clipping transformation	Scan-line, frame buffer
3	Creation of shadow volumes	Scan-line, frame buffer
4	*z*-buffer shadow computation	Frame buffer
5	Shadows using rays	Ray tracing

Table 1 shows typical hidden-surface algorithms that can be associated with each class. Other specific conditions were addressed by Liu, Burton, and Campbell[4] and by Max.[5]

The technique described here, which we developed at the Universite de Montreal, is a general version of Crow's shadow-volume technique. It permits shadow casting in an unconstrained environment, while still allowing interpenetration of polygons and any number of light sources anywhere in 3D space—features of Crow's original version. It was used in the 35-mm, 3D animated short film *Tony de Peltrie*,[6] which was shown at the close of the SIGGRAPH 85 Film and Video Show.

Shadow-generation techniques

We will now look at the five classes of shadow-generation techniques in more detail. The general version of Crow's algorithm is categorized in Class 3.

Class 1

Shadow computation during display, the first method ever used, has been demonstrated independently by Appel[7] and Bouknight and Kelley.[8] Basically, shadow boundaries on polygons are detected by a scan-line algorithm while the image is displayed. The boundaries are formed by projecting potential shadow edges onto the polygon being scanned. Shadow edges are then projected onto the image plane. The color of a scan segment changes when it crosses the edge of a shadow.

Class 2

Polygon shadow generation involves two passes through an object-space hidden-surface algorithm. The first pass separates shadowed and unshadowed polygons. Polygons partially shadowed are divided by determining the hidden surfaces from the viewpoint of the light source. The colors of shadowed polygons are modified. The second pass processes the altered data from the normal viewpoint. The best known algorithm of this class is named Atherton-Weiler.[9]

Class 3

The third class of algorithm was introduced by Crow.[1] It consists of constructing shadow volumes, which can be defined as the invisible volume of space swept out by the shadow of an object. In the scan-line process the shadow polygons forming the volumes are treated as an invisible surface that, when pierced, causes a transition into or out of an object's shadow.

Shapiro Brotman and Badler developed a method for generating soft shadows by combining a *z*-buffer algorithm with shadow volumes.[10]

Class 4

The *z*-buffer shadow computation approach, introduced by Williams,[2] works with a *z*-buffer hidden-surface algorithm. A view from each light source is constructed, and the *z* values are stored. A view of the scene is then constructed from the observer's point of view. The *x*-*y*-*z* position of each point in the observer's view is then tested for visibility by transforming it to each light source. If the point is not visible to the light source, it is in shadow.

Class 5

The fifth class of shadow-generation techniques, presented by Whitted,[3] involves a ray-tracing hidden-surface algorithm. For each pixel a ray is created. The ray goes from the eye, passes through the pixel, and stops when a model is encountered. A new ray is directed toward the light source. If a new model is pierced before reaching the light source, the pixel is in shadow; otherwise it is not. This idea obviously extends to more than one light source, but it is extremely costly in terms of computation time.

Ray tracing is a very powerful shading technique that lends itself naturally to the casting of shadows (as well as to transparency and refraction). Any type of model can be used in ray tracing (polygons, patches, simple types, etc.).

For a polygonal database, the ray-tracing technique, like our general version of Crow's algorithm, allows any shape of model or polygon. However, displaying large polygonal databases with ray tracing takes too much time to be practical—which is why we developed a general shadow algorithm compatible with scan-line techniques.

The shadow volume approach

Crow describes the shadow volume approach this way:[1]

> The third class of shadow algorithm includes shadow volumes in the hidden-surface computation by adding their surfaces to the data. Assuming a polygonal object, the shadow surface is given by planes defined by contour edges and the light source position. Each such edge defines a polygon whose boundaries are the edge itself, the two lines defined by the light source position and the endpoints of the edge and the bounds of the field of view [see Figure 1]. The sense of the polygon must be maintained so that the near surface of a shadow volume (frontfacing polygons) may be distinguished from that far surface (backfacing polygons). Thus the polygons facing the light source plus the set of projected shadow polygons for an object define its shadow volume.
>
> Shadow polygons may be treated just like the rest of the data when applied to a scan-line, hidden-surface algorithm; only the shading for visible surfaces must be handled differently. Shadow polygons are themselves invisible, thus they do not count in the determination of visibility. However, the depth order of shadow surfaces and visible surfaces determines shadowing. A frontfacing shadow surface puts anything behind it in shadow while a backfacing shadow surface cancels the effect of a frontfacing one. For example, a post or column might cast a shadow surface consisting of a single polygon pair. Any surface lying between those two shadow polygons would be in shadow while surfaces lying in front of or behind both polygons would be shaded normally.

The algorithms in this class can be divided into two processes: the creation of shadow volumes (in object space), and the display of shadows in the scan-line process (in image space).

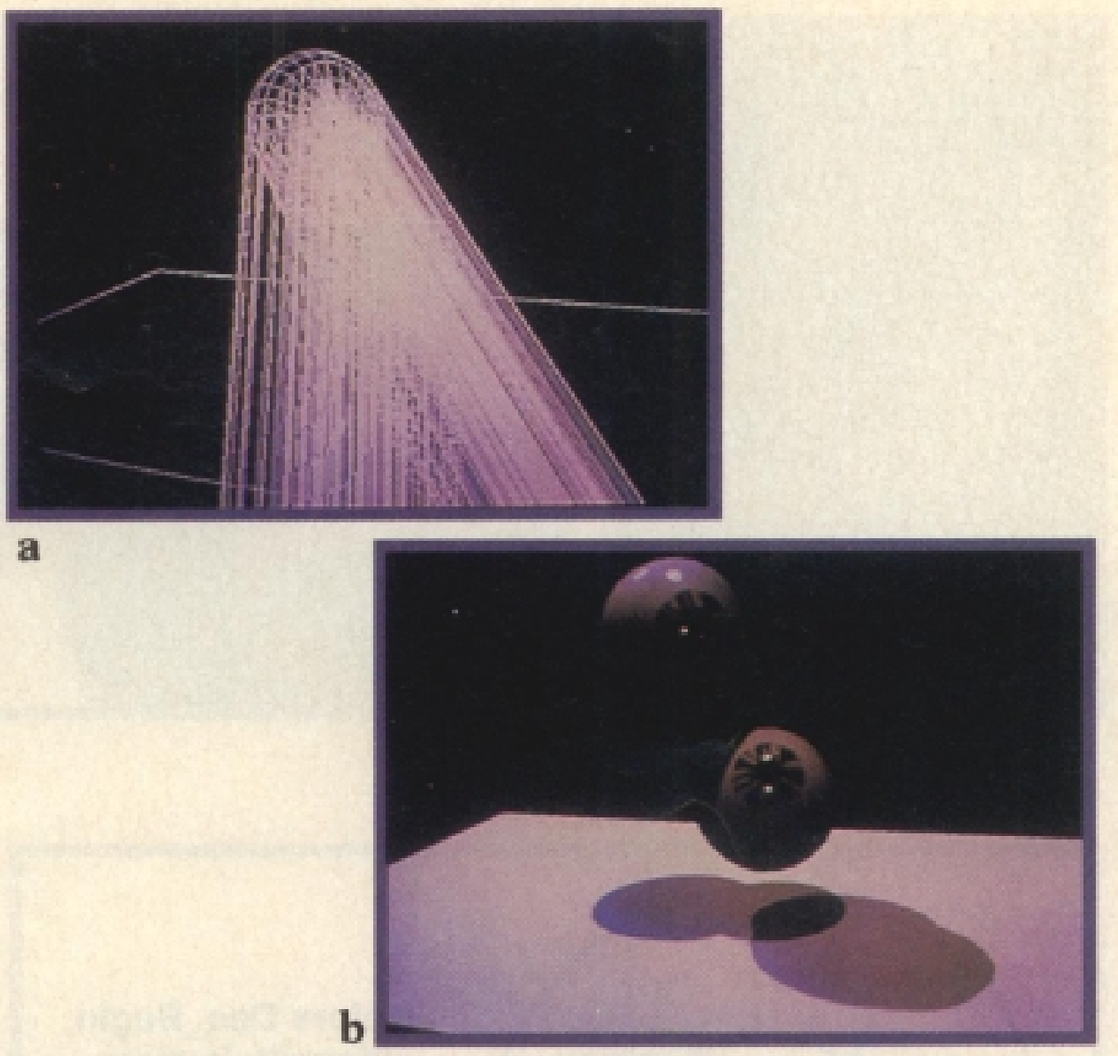
The environment

Let us describe the representation of models and the representation of lights in relation to the new version.

Representation of models

Most existing shadow algorithms based on a polygonal database assume a well-developed environment (convex polyhedrons or polygons, for example). Limitations vary from one algorithm to another. There is, however, one common restriction: Every algorithm (except ray tracing) assumes a planar polygonal database. If the original environment is not made of planar polygons, the user will have to subdivide each polygon into triangles before rendering.

Depending on the polygon, this process may be costly. It may also give results that are not artistically satisfying. Moreover, even if the original environment has been made planar to avoid subdivision, the user is forced to build models under severe and unnatural constraints. If the model is highly irregular, as in Figure 2, the polygons must be decomposed into triangular subsections to ensure planarity.



a

b

Figure 1. In (a) the normally invisible edges of the shadow polygons are projected from two spheres (the second sphere is obscured by the lines) lit by two light sources. In (b) you see the shaded images resulting from the models and shadow polygons of (a).

Our algorithm does not impose such limitations on the environment description. The data input consists of convex or concave, open or closed polyhedrons, with or without holes. Each polyhedron may be composed of convex or concave, planar or nonplanar polygons, with or without the holes.[11] Penetration between polygons is allowed. No subdivision is necessary at any stage during the process.

To use contour edges, the data structure provides links between adjacent polygons. An edge is shared by at most two polygons. A polygon observed from an outside view is defined clockwise.

Note that the shadow volume approach is incorporated into an existing scan-line algorithm: the Bouknight method.[12] Since this algorithm already handles the convex/concave, polygon/polyhedron cases, as well as penetration between polygons, we will not discuss those problems.

Representation of lights

Light can be modeled with either parallel or punctual light sources. Any number of sources can be defined, each of which may be anywhere in the