Digital Filmmaking

Some of the most sophisticated and spectacular applications of computer graphics are found in current films.

by Alvy Ray Smith

Computed pictures are now a fixture of human culture. Their impact is particularly evident in the field of motion pictures. *Futureworld*, *Looker*, *Star Trek II: The Wrath of Khan*, and *Tron* feature some outstanding examples of full-color raster-graphics realizations of 3-dimensional, solid, illuminated, and animated models [see the *Graphics Glossary*]. Several other films—represented by *Star Wars*, *Alien*, and *Return of the Jedi*—contain animated line-drawing, or calligraphic, scenes (see the cover picture).

But all these scenes are rudimentary compared to those promised by computer graphics for the future. Both the quantity and the quality of such scenes will increase sharply. However, there are several hurdles to leap before the computer ceases to be an expensive curiosity and joins the camera and optical printer as equals in the production of images for the movie screen [see the *Movie Glossary* and *Optical Printer* figure].

This article provides an overview of a computer graphics 3-D (3-dimensional) animation system suitable for feature film work. This system differs from those designed for less ambitious projects in terms of the size, speed, and resolution required and also the complexity of imagery desired. Its general nature, however, holds for any synthetic imagery application where artistic merit is the principal goal. The intent is to present a vocabulary for the subject, an appreciation of the basic simplicity of the theoretical foundations of the art, a sense of the remaining technological challenges, and an ability to discriminate between more and less sophisticated computed images. The digital revolution will certainly extend the language of film; this article is intended to serve as a "film appreciation" primer for this new aspect.

There are two main categories of tasks in computer-mediated 3-D animation, those dealing with modeling and those dealing with rendering. By *modeling* is meant the creation of the 3-D database which serves as the "world" to be portrayed in a synthetic computer-graphics sequence. Time ordering of a database, that is, the "animation" of it, is considered to be part of the modeling process. *Rendering* is a frame-by-frame realization of the database into two dimensions. Of course, one database may be rendered in an infinity of ways. The output of a single program is seldom an entire frame, but rather is one component of it. Combining these components into a final frame, the "compositing of its elements," is considered part of the rendering process. Compositing is a nontrivial task which can be compared in difficulty and importance to editing in conventional filmmaking.

Modeling

One of the most difficult parts of computer animation is get-

Alvy Ray Smith received his Ph.D. in Computer Science at Stanford in 1969 with a dissertation on cellular automata theory; shortly thereafter, he united his computer science education and artistic interests by plunging into computer graphics. In 1975 he went to the Computer Graphics Laboratory at the New York Institute of Technology, and spent five years there generating stills, videotapes, and films, which have appeared in print, in museums, and on TV. In 1980 he joined the Computer Division of Lucasfilm Ltd. where he is Project Leader of the division's computer graphics effort, which has produced material for *Star Trek II: The Wrath of Khan* (the "Genesis Demo" sequence) and for Lucasfilm's *Return of the Jedi*.

Graphics Glossary

Raster graphics. An image modulates a standard arrangement of scanlines. This arrangement is that of ordinary television and is called a *raster*. The required electronics for a raster pattern are well-established, and hence not expensive. The most sophisticated computer graphics are realized this way. Antialiasing is required.

Calligraphic graphics. An image is formed from scanlines oriented in arbitrary directions and drawn in an arbitrary order. Expensive electronics are required, but spatial antialiasing is not. Typical of this style of graphics are the "wire-frame" models which were considered synonymous with computer graphics in the early days.

3-dimensional. In computer graphics this refers to the three spatial dimensions stored for each point of a model in a database, rather than to the perception of three dimensions as obtained, say, with two images and polarized glasses. Stereo graphics is possible, of course, but is only a subset of 3-dimensional graphics.

CAD/CAM. Acronym for Computer-Aided Design/Computer-Aided Manufacturing. This is a rapidly-growing branch of computer graphics, currently relying primarily on calligraphics but now branching out to incorporate raster graphics. It is used to design auto parts, for example, and for integrated circuit design. Realizability of the modeled objects is of great importance.

Pixel. Short for "picture element." Refers to the smallest computational unit in a computed picture. A computed picture is typically composed of a rectangular array of pixels (e.g., 1000 by 1000). The number of pixels in one dimension is frequently used as the *resolution* of the picture (e.g., a 1000-line resolution image).

Framebuffer. An especially adapted piece of digital memory for storing a computed picture. Framebuffers typically have a video control and display for viewing the contents of the memory, assumed to be a 2-D picture.

Spline. A piecewise polynomial with at least first-order continuity between the pieces. A mathematically simple and elegant way to connect disjoint data points smoothly, hence used not only for generating smooth curves and surfaces between sparse data points, but also for smooth motions between parameters sparsely located in time—such as those used to describe the keyframes in an animation.

Patch. A piece of a curvilinear surface, typically with three or four sides. These are attached together at their edges with at least first-order continuity to form complex 3-D surfaces. The edges of a patch are frequently described with polynomials or ratios of polynomials. For example, if both dimensions are described with cubic polynomials, then the patch is said to be *bicubic*. If ratios of cubic polynomials are used, then the patch is a *rational bicubic* patch. Although difficult to deal with, one patch can take the place of hundreds of flat polygons and thus greatly reduce the size of a database.

Antialiasing. A most important and all-too-often neglected aspect of computer graphics, the lack of which results in such problems as "jaggies" or "stairsteps" along what should be smooth edges, "strobing" of edges in time (the temporal equivalent of jaggies in space), and the "wagon wheel revolving backwards" phenomenon. A sure sign of the infancy of computer graphics, just as flicker is in film.

Fractals. A branch of mathematics recently codified by Benoit Mandelbrot which deals with curves and surfaces with non-integral, or fractional, dimension. In computer graphics applications, this relates to a technique for obtaining a degree of complexity analogous to that in nature from a handful of data points.

Texture. Any 2-D pattern which is used to add the appearance of complexity to a 3-D surface without actually modeling the complexity. Paintings or digitized photographs are frequently used. Whereas fractals actually add complexity to a 3-D database, textures do not. The 2-D arrangement of pixels in a computed picture is frequently compared to the warp and woof of textiles, hence the term.

Field of view. The limits of what a simulated camera can see, usually expressed as a horizontal angle centered at the camera. For simplicity of computation, computer graphicists assume that what a camera sees lies within a pyramid—rather than a cone—with apex at the camera [*Perspective* figure].

Aspect ratio. A ratio, typically of width to height, of the cross section of the pyramid seen by a simulated camera. This plus the field-of-view angle gives the vertical field-of-view angle.

Clipping plane. If an object gets too near, or behind, a simulated camera, it becomes meaningless. Hence it is customary to clip—not display—any objects or parts of objects behind the camera or between it and the so-called "near" or "hither" clipping plane. Similarly, objects very far from the camera are not displayed if they fall beyond the "far" or "yon" clipping plane [*Perspective* figure]. The portion of the pyramid, as seen by a simulated camera, between the near and far clipping planes is known as the *viewing frustum.*

Edge sharpening. Any real digitizing process tends to soften edges—that is, to remove high frequency information—by convolving the shape of the scanning spot with the object digitized, an integration process. This can be compensated for somewhat by passing a so-called "Laplacian" filter over a digitized picture which sharpens edges by differentiation.

Contrast enhancement. A real digitizing process typically involves some kind of non-linear detector which destroys the light-to-dark relationships of the object scanned. The correct contrast can be reintroduced if the characteristics of the detector are known. In fact, the contrast can even be heightened if desired.

Movie Glossary

Optical printer. The instrument used for cross-dissolves, fades, format conversions, and, most importantly, compositing of images filmed separately. It is one of the three most important instruments for making a movie, and the least known or understood. The camera and audio recorder are well appreciated by the public, but the optical printer is recognized only by filmmakers. There are currently several attempts being made to digitize this all-important machine.

Fade. A frequently used effect for beginning or ending a scene. A scene begins by apparently raising the light level from black to full intensity for a *fade in*. Similarly, a scene ends by apparently lowering the light level to black for a *fade out*.

Cross-dissolve. A frequently used effect for changing from one scene to the next. The first scene is faded out while the next is faded in. To simply abut the two scenes is called a *cut* and doesn't require an optical printer.

Matting. The combining, or *compositing*, of two or more separately filmed images through the use of auxiliary pieces of film called *mattes*. These are opaque where an image is desired and transparent elsewhere. Matting is a difficult art.

Bluescreen matting. A popular matting technique which requires foreground objects to be shot against a blue screen, typically an intense blue sheet illuminated from behind by an array of lights. The blue is replaced with a desired background by a process which requires multiple passes through an optical printer and several auxiliary pieces of film for each foreground object. This process is non-algorithmic since it requires human judgment in its execution. It is difficult because of partial transparencies—for instance, at the edges of objects blurred because of rapid motion.

Editing. The task of splicing many pieces of film together to form a whole. Much more film is shot than will be used. The editor selects just those frames needed to create an effect, and the rest end up "on the cutting room floor." Obviously, a great amount of artistic control is exercised at this point in the filmmaking process. The lack of editing is generally the hallmark of "home movies."

Aperture. The size of the opening in a camera or projector which passes light to or through the film. Hence, the size of the actual frame of film [*Apertures* figure].

Academy aperture. A common 35mm aperture measuring 22.05mm by 16.03mm for an aspect ratio of 1.375. A central "composition area" of 20.96mm by 11.33mm is frequently used for a wide format of aspect ratio 1.85.

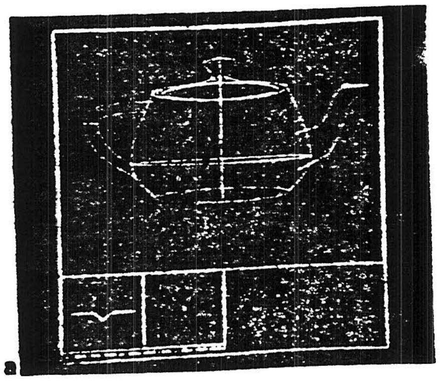
Panavision or Cinemascope. A popular wide-screen aperture measuring 22.05mm by 18.67mm on 35mm film. During projection, the horizontal dimension is magnified by a factor of two with a so-called ***anamorphic*** or "***scope***" lens. The resulting aspect ratio is about 2.35.

Vistavision. Another wide-screen aperture measuring 37.72mm by 25.17mm on 35mm film. The horizontal dimension runs the length of the film rather than the breadth typically used by other formats. Ordinary 35mm slides are approximately this size. Since this aperture requires special equipment, it is not often used in theaters. Its large size is used internally in special effects houses, however.

Format conversion. Converting film with one aperture for use with another—for example, for converting Vistavision to Panavision.

ting the object for animation into the computer in the first place—i.e., the creation of the database. There are two main ways to accomplish this modeling process, input scanning and interactive synthesis from primitives.

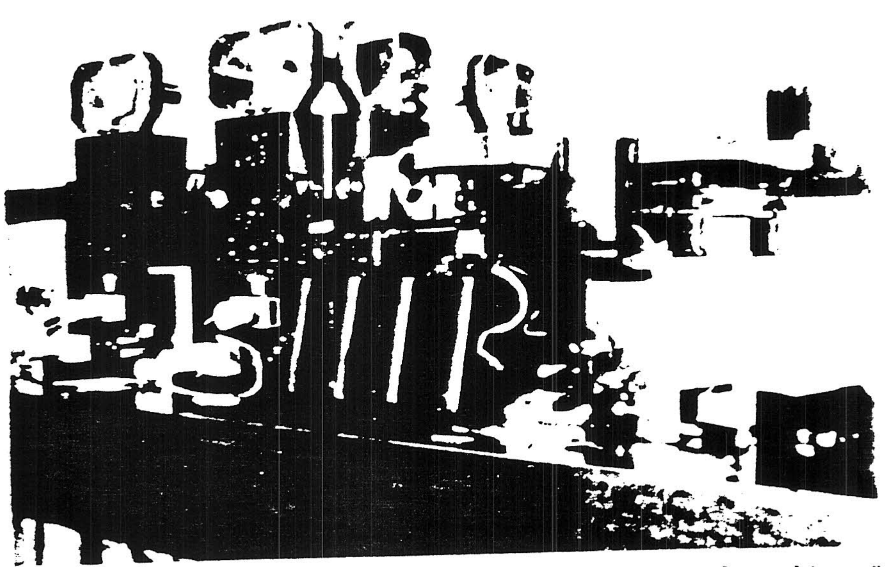
Digitization. Input scanning, or digitization, can mean several things. The most direct 3-D digitization technique for a given object is simply to enter the 3-D coordinates of the object into the computer. Special devices, "3-D digitizers," are just becoming available. One such device bounces laser light off an object to determine its distance from the laser; this, combined with the known location of the laser, provides the 3-D information. With another, three orthogonal magnetic fields are transmitted to a wand, the tip of which a modeler touches to an object: the strengths of the three received fields are used to determine where the wand is located and establish a data point on the object's surface. Without such a device, direct entry is forbiddingly difficult. One consequence of this is the paucity of 3-D databases and the popularity of those, such as Martin Newell's teapot, which exist in the public domain. This teapot is the example used at some time or another by nearly all computer-graphics houses [*Teapot* figure].


a

3-D reconstruction from 2-D information is more common. For example, the carefully drawn plans or blueprints of the object may be digitized using readily available tablets (or 2-D digitizers). Several views—say, two or three orthogonal projections—are entered this way, and the computer is used to derive 3-D coordinates from the given information. Jim Blinn used this method to enter the Voyager spacecraft model into the Jet Propulsion Laboratory computers for his Jupiter/Saturn flyby simulations.

In another technique, grids are projected onto the object of interest, which is then photographed from two or more different angles with cameras in a known spatial arrangement. The photographs are hand-digitized at the grid points. Enough information is provided for the computer to obtain 3-D coordinates from triangulation. This method was used by Information International Inc. to enter actor Peter Fonda's head into their computers for the movie *Futureworld.*

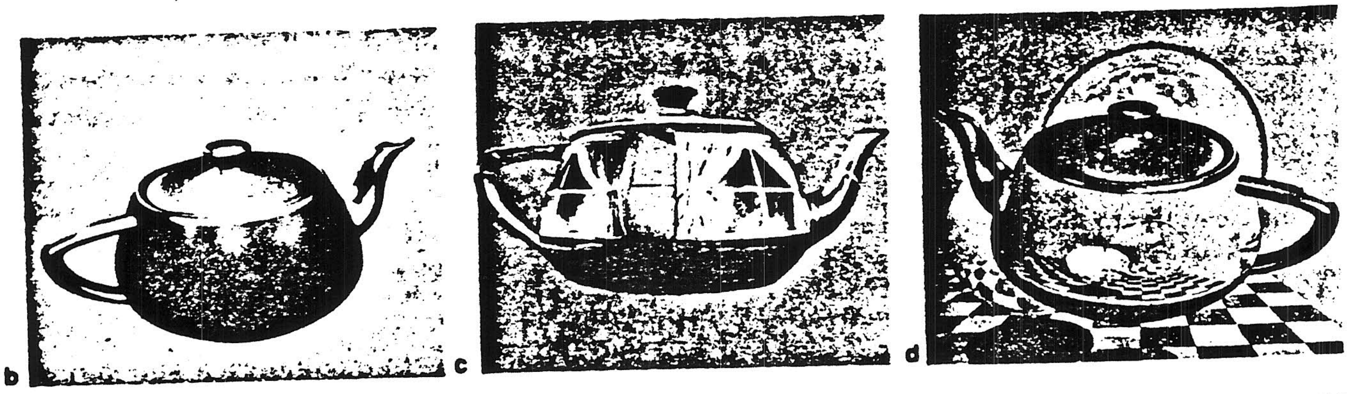
There are several other variations on these triangulation methods, but the main thing to note is that all the digitization processes are quite tedious. They also share the problems of organizing the mass of data so entered into a usable database, removing spurious points, and smoothing out the noise typical of this kind of point measurement.

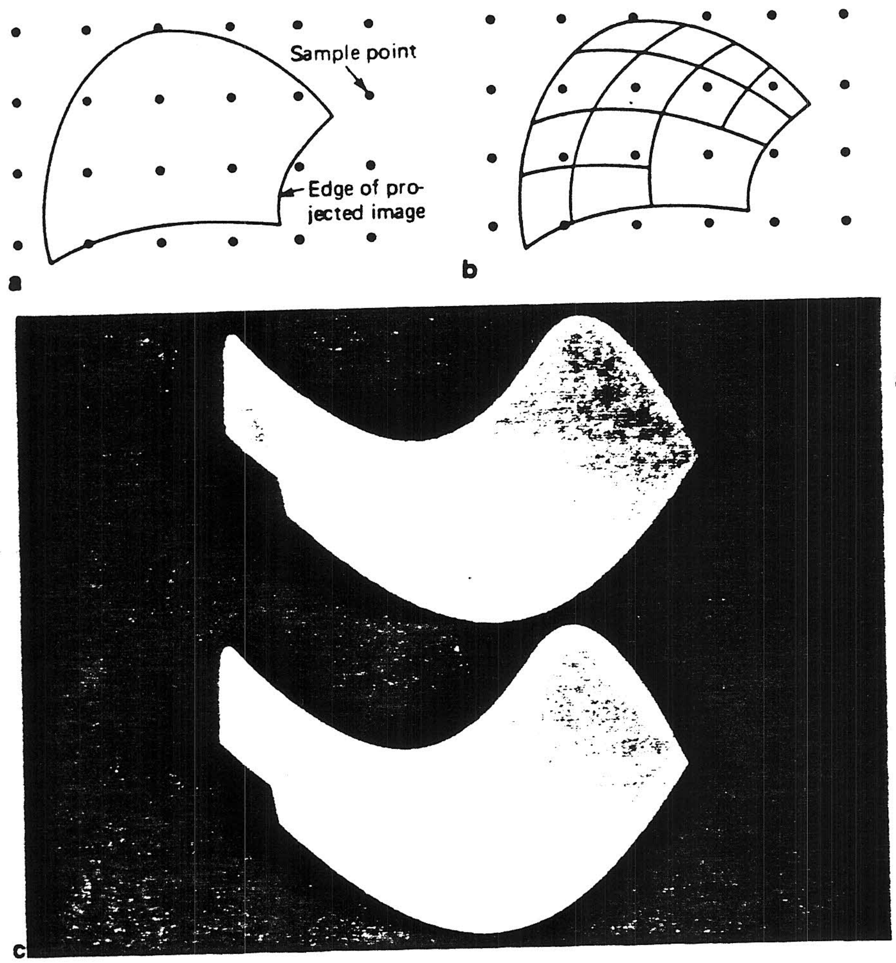

Optical Printer. A state-of-the-art optical printer used in the making of *The Empire Strikes Back.* [Courtesy ILM, Lucasfilm.]

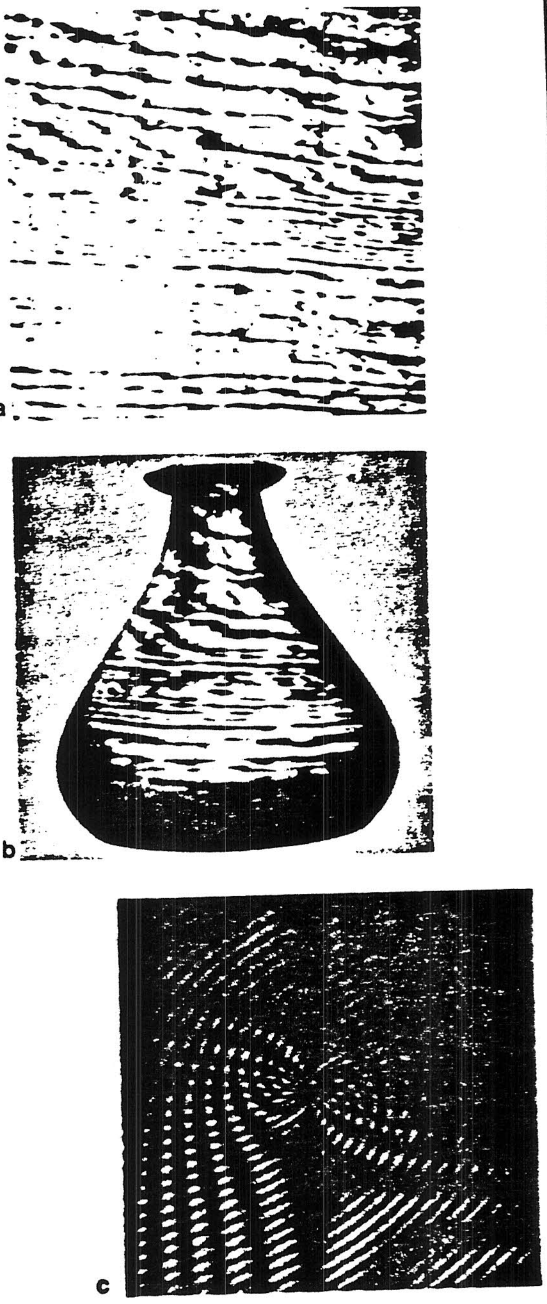
Synthesis from Primitives. Very accurate models may be entered directly into a machine using interactive programs for synthesizing objects from primitive shapes. This is called "solids modeling" in the context of CAD/CAM, but, in 3-D animation for movies, many of the features necessary for CAD/CAM systems may be omitted. For example, dimensioning of parts and specification of tolerances are unnecessary. Whereas many CAD/CAM systems insist on realizable objects, movies are fortunately not restricted to reality.

Some common primitives are the quadrics—sphere, cone, cylinder, ellipsoid, paraboloid, hyperboloid—and their superquadric generalizations, the torus (sometimes included as an honorary quadric), polygons, and patches—bilinear, biquadratic, and particularly bicubic patches

Teapot. The canonical teapot. (a) A calligraphic display. (b) A bronze and copper teapot illuminated by three light sources. (c) The background picture, created by Turner Whitted with a ray-tracing algorithm, is texture-mapped onto a bronze teapot. (d) A bump-mapped teapot. [(a)-(c) by Rob Cook, Loren Carpenter, and Bill Reeves, Lucasfilm. (d) courtesy Information International Inc. and SIGGRAPH.]





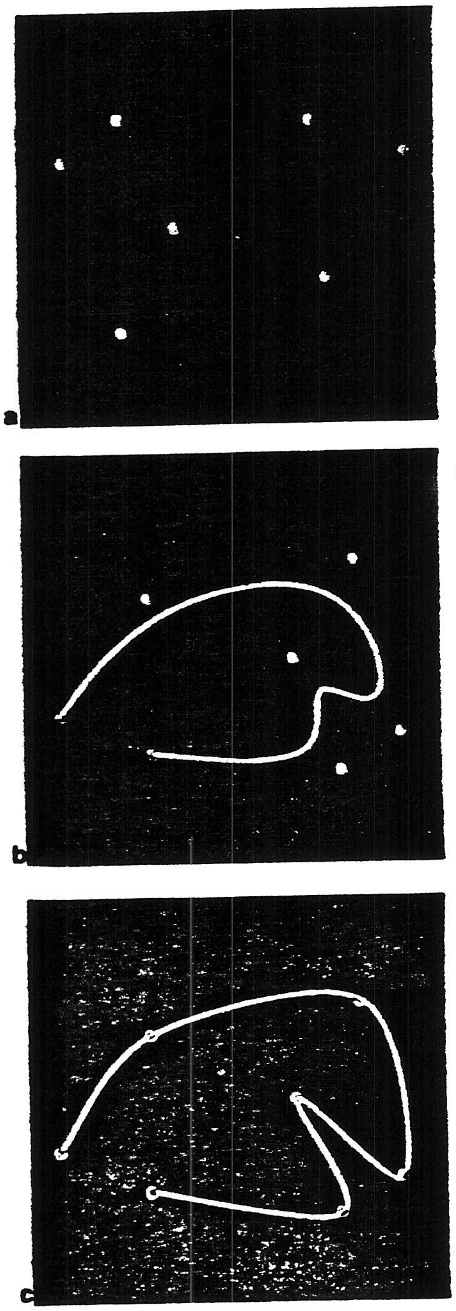


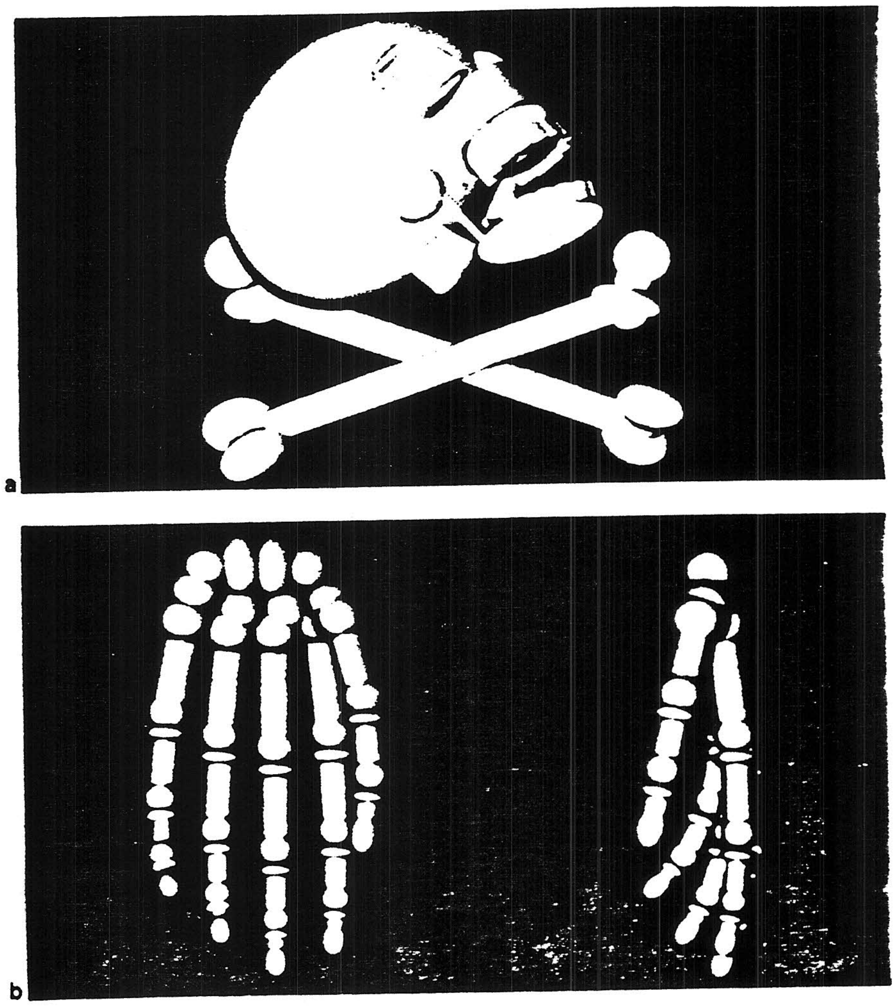
Patch. **(a) A typical bicubic patch, its edges defined by cubic polynomials. (b) The same patch recursively subdivided so that no sub-patch covers more than one sample point. (c) Another patch rendered with and without antialiasing. [(a), (b) courtesy Ed Catmull. (c) by Loren Carpenter, Lucasfilm.]**

Textures. **(a) A scanned-in and digitized photograph of wood texture. (b) The same texture used to control the coloring and lighting of a synthetic vase modeled as a surface of revolution. (c) An algorithmic texture. [(a), (b) courtesy Rob Cook, Lucasfilm. (c) courtesy New York Institute of Technology.]**

[*Patch* figure (c)]. Other primitives are point spots, lines, and surfaces of revolution (with silhouettes defined by polygons or splines [*Textures* (b) and *Splines* figures]. These will be further discussed in the section on Model Animation.) Some of these "primitives"—a sphere, for instance—are only primitive at the modeling stage where their intuitive simplicity serves them well. At the later rendering stage, a sphere might be represented by several hundred small flat polygons. Whatever they are chosen to be, primitives are arranged together to form complex "sculptures" or models. A trivial example is the formation of a cube from six square polygons. Slightly more complex is a femur modeled from two cylinders and five ellipsoids [*Articulation* figure (a)].

Procedural Models. Another basic type of building block too sophisticated to be called primitive is the program-defined or *procedural* model. An excellent example of this general class of models is the wall of fire, by Bill Reeves of Lucasfilm, used in the "Genesis Demo" sequence of the movie *Star Trek II* (portraying the "Genesis effect" which instantaneously transforms death into life) [*Procedural* figure (a)]. A program was written which created hundreds of thousands of particles and then kept track of each as it arched





Splines. (a) Given data points. (b) The data approximated with a B-spline. (c) The same data interpolated with a Catmull-Rom spline. [Courtesy Lucasfilm.]

Articulation. An articulated skeleton modeled solely from cylinders and ellipsoids and animated by varying the elements of the 4×4 matrix conceptually stored at each joint. This model was created and animated by the author, using programs by Garland Stern and Tom Duff. (a) Skull and two femurs. (b) Two hands including wrists. [Courtesy New York Institute of Technology.]

through a trajectory and changed colors as it cooled. The sheer complexity of the fire model required computer control. Its success depended as much on the simulated physics as on the primitive image elements, which were short, smooth line segments. Lest procedural models be taken as synonymous with *ad hoc* models, it should be noted that the fire procedure has been used to model other processes, such as fountains, fireworks, and grass [*Procedural* figure (b)]. Reeves has termed this class of procedural models "particle systems."

Another relatively new type of model is the *fractal*, a rich category of structures recently identified by Benoit Mandelbrot. This includes such forms as mountain ranges, coastlines, tree branching systems, river systems, the vascular system, and a host of mathematical delights such as the "space-filling" curves which have the property of being one-dimensional (they are curves) but visiting every point in the plane, and hence are two-dimensional in some sense. Mandelbrot has suggested that these should really be given a fractional dimension between one and two; hence the name. The concept also generalizes to dimensions greater than two.

In computer graphics, the fractal concept has been successfully applied to the creation of landscapes. [*Fractal* figure (a)]. Loren Carpenter of Lucasfilm contributed the beautiful fractal landscapes of the Genesis Demo sequence in *Star Trek II*. A fractal model is really a primitive model to which a controlled randomizing process has been applied to create a more detailed model. The procedure creates complexity recursively [*Fractal* figure (b–d)].

Model Operations. The synthesis technique of database creation requires that an artist or designer create objects from the primitives or procedural models available. This implies that there is some interactive station available, or at least a keyboard-driven language, which provides copies, or "instances," of the primitives and supplies a set of operators for positioning and combining these instances.

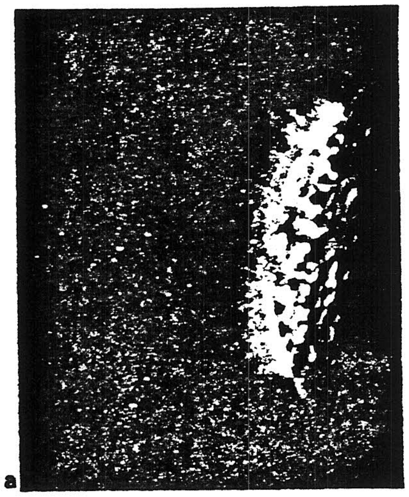
Some typical operators include the "boolean" operators of intersection, union, and set difference. These are particularly popular in CAD/CAM contexts. They are difficult to realize in the case of higher-order surfaces such as the bicubic or rational bicubic patches because of the difficulty of deriving analytic expressions for intersection curves. Numerical, or iterative, approaches must be used instead.

Another set of operators consists of what might be called "articulation" operators. Their purpose is to force a hierarchy on a set of primitives. At the New York Institute of Technology, there are programs for arranging ellipsoids and cylinders into tree structures representing, for example, humans or animals [*Articulation* figure]. The nodes of the tree are the joints of the corresponding animal, which is why these are called articulation operators. Included are operators which allow a user to manipulate or traverse the tree structures created with other operators.

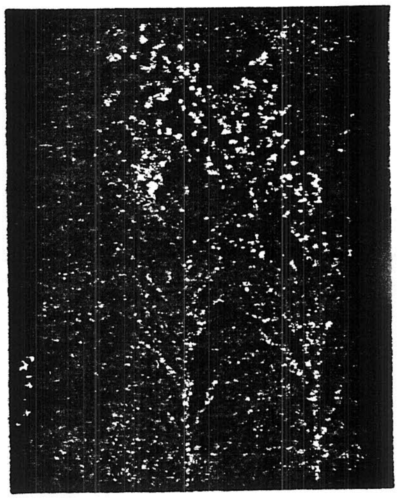
Underlying any interactive 3-D viewing situation are the rotation, scaling, translation, skewing, and perspective transformations which are the given language of 3-D computer graphics. It is well-known that one 4 × 4 matrix will accomplish all these transformations simultaneously or, using a series of them, sequentially. Some hardware commonly available, notably the Evans & Sutherland Picture System, has a 4 × 4 matrix multiplier built into its hardware since this is such a common operation in graphics. Very briefly, if a model is thought of as a set of 3-D points, then these points are turned into 4-D points by appending a unity 4th coordinate, the so-called *homogeneous* coordinate. Then a 3-D transformation is accomplished by multiplying a matrix of form

$$\begin{bmatrix} s & S & S & p \\ S & s & S & p \\ S & S & s & p \\ t & t & t & g \end{bmatrix}$$

times each 4-D point in the model. (S stands for skew, s for scale, t for translate, p for perspective, and g for global scale. Rotations are a combination of s and S entries.) Then the 4-D points are converted back into 3-D points by dividing the first three coordinates by the homogeneous coordinate [*Matrices* and *Perspective* figures]. Articulated tree models frequently have a 4 × 4 transformation matrix associated with each

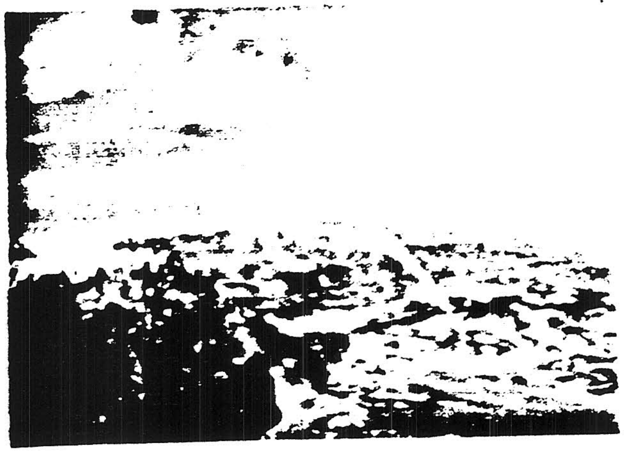

a

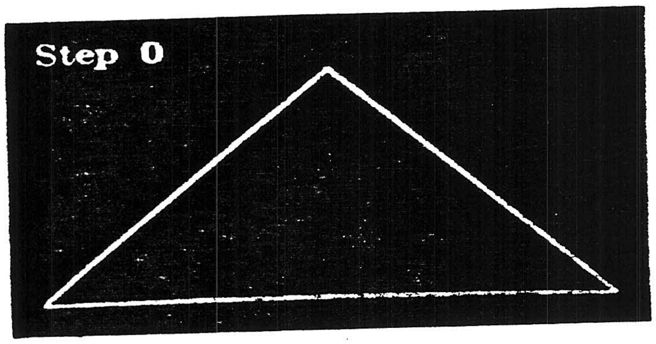
b

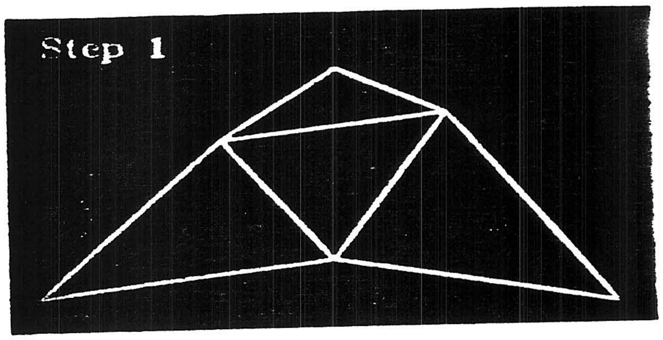


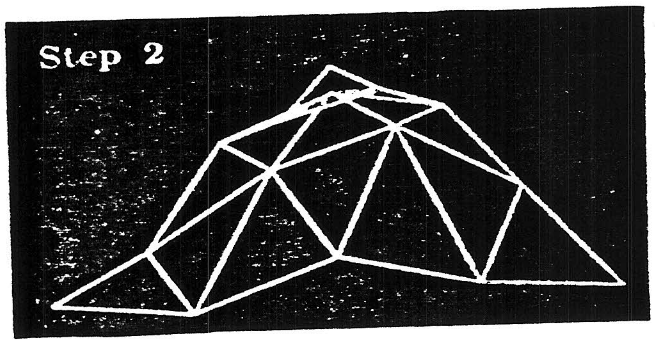
Procedural. **Two examples of procedural models. (a) Fires by Bill Reeves in *Star Trek II: The Wrath of Khan*. They are composited with bump-mapped craters by Tom Duff and point-spot stars by Tom Porter. (b) Plants by the author with grass by Bill Reeves. [Courtesy Lucasfilm.]**

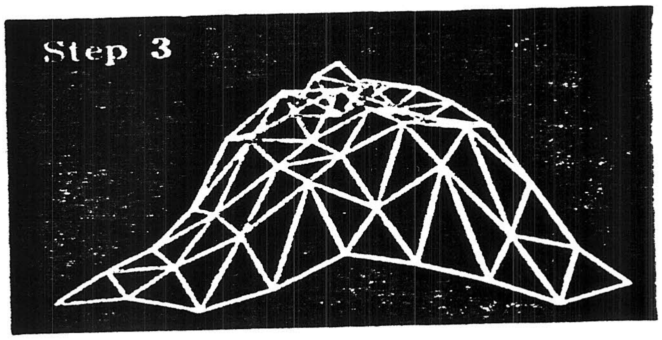
Fractal. (top right) A fractal landscape with painted clouds and ray-traced rainbow. (Steps 0–7) A triangle and its first seven fractal subdivisions. At each step, the sides of all triangles are broken at their midpoints. Each midpoint is displaced a random amount proportional to the length of the corresponding side. The new points are connected to the old and to one another, forming four new triangles for each previous triangle. By Step 7 the original triangle has become mountainlike. [Landscape: mountains by Loren Carpenter; clouds by Mike Pangrazio, Chris Evans, and Frank Ordaz; rainbow by Rob Cook, Lucasfilm. Steps 0–7: Courtesy Lucasfilm].

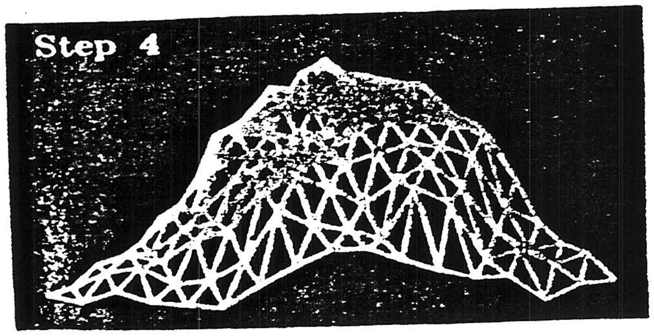


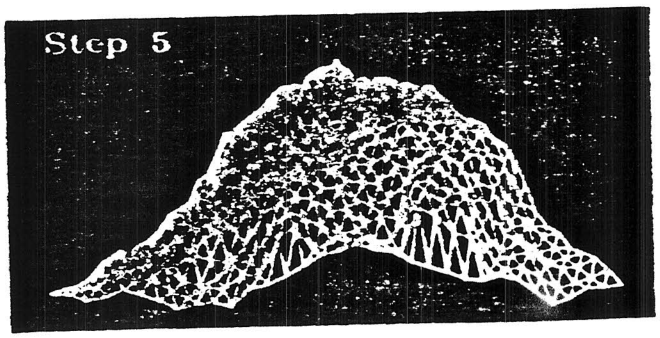


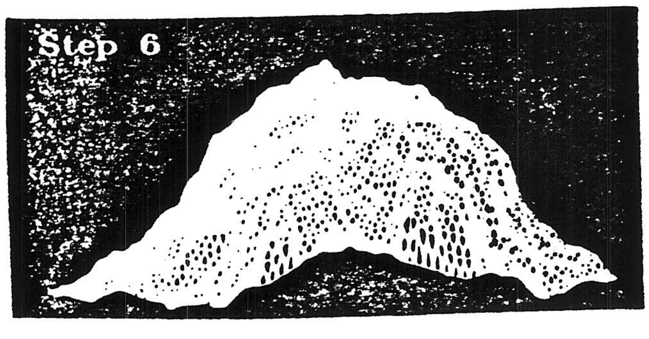


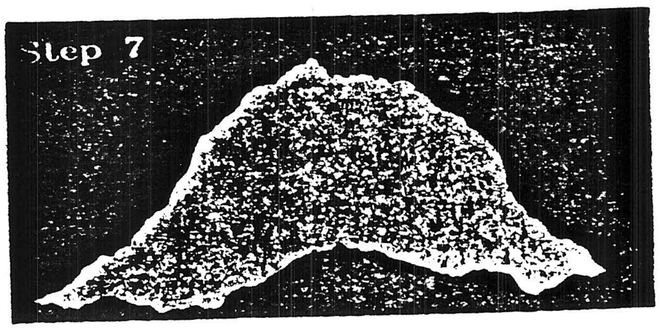












Matrices

The *projectivities* are those 3-D transforms which can be specified with a 4×4 matrix A and which take a vector $[x\ y\ z]$ into a vector $[x'\ y'\ z']$ as follows:

$$[X\ Y\ Z\ W] = [x\ y\ z\ 1] * A,$$

$$[x'\ y'\ z'] = \left[\frac{X}{W}\ \frac{Y}{W}\ \frac{Z}{W}\right].$$

Some examples follow:

(a) Scale x by 1, y by 2, z by 3 has

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[x'\ y'\ z'] = [x\ 2y\ 3z].$$

(b) Move x by -1, y by 3, z by 2 has

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 3 & 2 & 1 \end{bmatrix}$$

$$[x'\ y'\ z'] = [x-1\ \ y+3\ \ z+2].$$

(c) Rotate 30 degrees about the z axis has

$$C = \begin{bmatrix} \cos 30 & \sin 30 & 0 & 0 \\ -\sin 30 & \cos 30 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[x'\ y'\ z'] = [.866x-.5y\ \ .5x+.866y\ \ z].$$

(d) Perform (a) then (b) then (c) has

$$D = C*B*A,$$

$$[x'\ y'\ z'] = [.866x-.5y-1\ \ x+1.732y+6\ \ 3z+6].$$

Matrices. Matrix transformation examples. The matrices correspond to the transformations (a) scale x by 1, y by 2, and z by 3; (b) move x by -1, y by 3, and z by 2; (c) rotate 30 degrees about the z axis; (d) perform (c) then (b) then (a).

node to specify relative placement and size of each part.

Animation. So far, the discussion of graphical database creation has centered on the generation of still frames. One of the powers of—and principal motivations for—computer graphics is time modeling or "animation." Strictly speaking, the term "animation" means "giving life to something," and is the domain of artists we call animators. In the context of computer graphics, it means only "giving motion to something," and this is how we shall use the term here. Of course, in the hands of an animator they are much the same thing. The problem is conveniently broken into the two subproblems of animating the model and animating the camera which views the model.

Model Animation. The usual notion of animation in both 2-D and 3-D computer graphics is called *keyframe animation.* Only selected, important frames are modeled using some favorite modeling program (perhaps with articulation operators), and the computer is used to generate the "inbetween" frames. For example, in complex animation, perhaps every third or fourth frame is provided as a keyframe to the "inbetweening" program, which then fills in the missing two or three frames between keyframes. It was an early hope and is still a common misconception that inbetweening is somehow "easy" for computers. But the depth information missing from a 2-D model must be deduced by an animation program to know, for example, whether a swinging arm passes over or under the body to which it is attached. This is, in general, a difficult if not impossible problem, requiring human intelligence. No one has yet written an artificially intelligent 2-D computer inbetweener. The availability of depth information in 3-D models means, surprisingly, that computer-aided animation is easier in 3-D than it is in 2-D.

An adequately complete animation program—assuming keyframe creation is accomplished elsewhere—includes the following facilities: the ability to specify path of movement; placement of frames in time along this path (linear interpolation is seldom sufficient in either time or space); and real-time playback of generated frames. Also typically needed are means for changing frame numbers attached to frames, for changing the number of inbetween frames, for copying motion used in one set of frames to another set of frames, and for creation of new keyframes, called *breakdowns,* from old inbetween frames when it becomes clear that the given set does not give fine enough control. The features required for a powerful and comfortable animation system are surprisingly numerous. The design of interfaces for such systems, easily understood by computer-naive artists, is an art form in itself.

Since models typically become quite complex, even in line-drawing form, there must be some way to represent primitives in a simplified fashion while exercising the animation program. For example, if spheres are represented by lines of longitude and latitude, then the number of these lines might be reduced. Alternatively, just a box barely enclosing each sphere might be used to represent it.

Smooth motion paths are best specified by passing splines (piecewise cubic polynomials here) through keyframe parameters. The two most useful splines currently known are the *B-spline* (the *B* stands for *basis* for reasons that will not concern us here) and the *Catmull-Rom spline* (named for two of its discoverers). Both of these are cubic splines with local control; that is, a change in a keyframe parameter being splined affects the parameter only at the frames between the given keyframe and the two keyframes before and the two after the given keyframe. The B-spline is an *approximating* spline and the Catmull-Rom spline is an *interpolating* spline, sometimes called a "cardinal" spline [*Splines* figure]. An approximating spline passes near the parameter values at the keyframes (the "knot" values of the spline) yet not necessarily through them, but an interpolating spline is guaranteed to pass through them. The approximating spline tends to be more graceful than the interpolating spline because it preserves second-order continuity. The interpolating spline may have undesired kinks in it.

These two splines are so easily specified, and so useful, that they should be taught along with the classic trigonometric functions. Given a list of x-coordinates, and a parameter u which will take us along the spline connecting (or approximately connecting) one coordinate x_0 to the next x_1 as the parameter is varied from 0 to 1, a new x-coordinate is obtained for each value of u from the four nearest given x-coordinates (two behind, two ahead, along the curve) by $U * M * X^T$, where

$$U = [u^3\ u^2\ u\ 1],$$
$$X = [x_{-1}\ x_0\ x_1\ x_2],$$

and M is the "magic matrix"

$$\frac{1}{2} * \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

for Catmull-Rom splines or

$$\frac{1}{6} * \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

for B-splines. The y-coordinates and z-coordinates of a given set of points to be interpolated would be treated similarly, as would any other parameter to be smoothly interpolated through the animation—for example, the angle of rotation about the z-axis.

Perspective

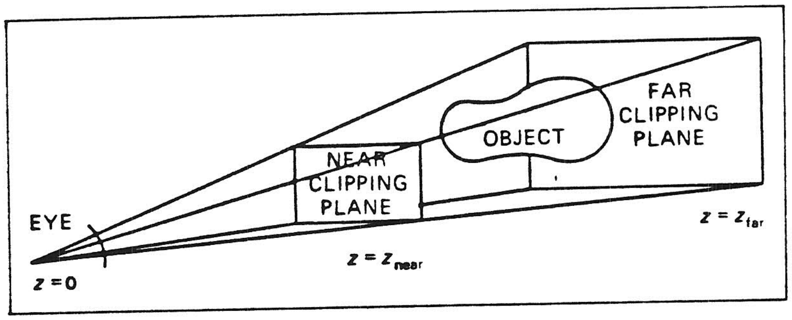
Perspective is a *projectivity* (cf. *Matrices* figure). A common method for specifying perspective maps x and y onto $[-1,1]$ and z onto $[0,1]$ with the matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e & 1 \\ 0 & 0 & f & 0 \end{bmatrix}$$

where

$$d = 1 - \frac{z_{near}}{z_{far}},\quad e = \frac{1}{d},\quad f = \frac{-z_{near}}{d}$$

and z_{near} and z_{far} are the near and far "clipping" planes as shown in the following diagram. That is, all parts of an object nearer than z_{near} or farther than z_{far} from the eye are to be ignored.



Perspective is applied first. So the transform (d) in *Matrices* figure would be projected into perspective space by the transform $D*P$. So for $z_{near} = .1$ and z_{far} at ∞:

$$[X\ Y\ Z\ W] = [.866x - .5y - 1\quad x + 1.732y + 6\quad 3z + 5.9\quad 3z + 6],$$

$$[x'\ y'\ z'] = \left[\frac{.866x - .5y - 1}{3z+6}\quad \frac{x + 1.732y + 6}{3z+6}\quad \frac{3z+5.9}{3z+6}\right].$$

Perspective. Conventional perspective with a 4×4 matrix. The perspective transformation and the viewing frustum, showing the viewing parameters.

Camera Animation. Animating the camera's view of the object or objects making up a synthetic scene is much like the process of model animation discussed above. In fact, it can be combined with that procedure, but I have isolated it as a separate part of the task of animation because it is usually thought of as an independent process, if only because the terminology tends to be different. Whereas one tends to use commands such as "rotate 30 degrees about x (y)" for animating a model, the related command to the camera might be "rotate 30 degrees in elevation (azimuth)."

The so-called "viewing parameters" are placed under control for camera animation, but are typically set to some standard setting for model animation where the object changes but the camera doesn't budge. These parameters are related to the perspective transformation applied to the synthetic scene: field-of-view (which can cause "fish-eye' lens distortions, if desired); near and far clipping planes [*Perspective* figure]; aspect ratio (such as Panavision or Vistavision [*Apertures* figure]).

It is convenient to provide users with several "cameras" so that a scene can be viewed from several different vantage points. These different cameras may also be thought of as keyframes for the camera animation. Another convenient tool is an *oracle camera* which views the scene and the other cameras as well. They might be represented, for example, by their viewing frustums.

The animation of the path of the camera (along with any other parameters describing its motion) is done using B-splines or Catmull-Rom splines just as for models.

Modeling Interfaces. The interface to a modeling program is of utmost importance. There are very few successful interfaces, where success is measured by grace and power. Grace in turn is measured by economy of motion, intuitiveness of control mechanisms, and the "feel" of the actual, physical controls. For example, it is difficult to write a convenient interface which allows a user to "attach this sphere at this point to that cone at that point." It is easy to write a program which assumes only one coordinate system—that attached to and moving with the object being constructed—but it is harder to write one that references the fixed screen coordinate system, and hence, unfortunately, this is seldom done.

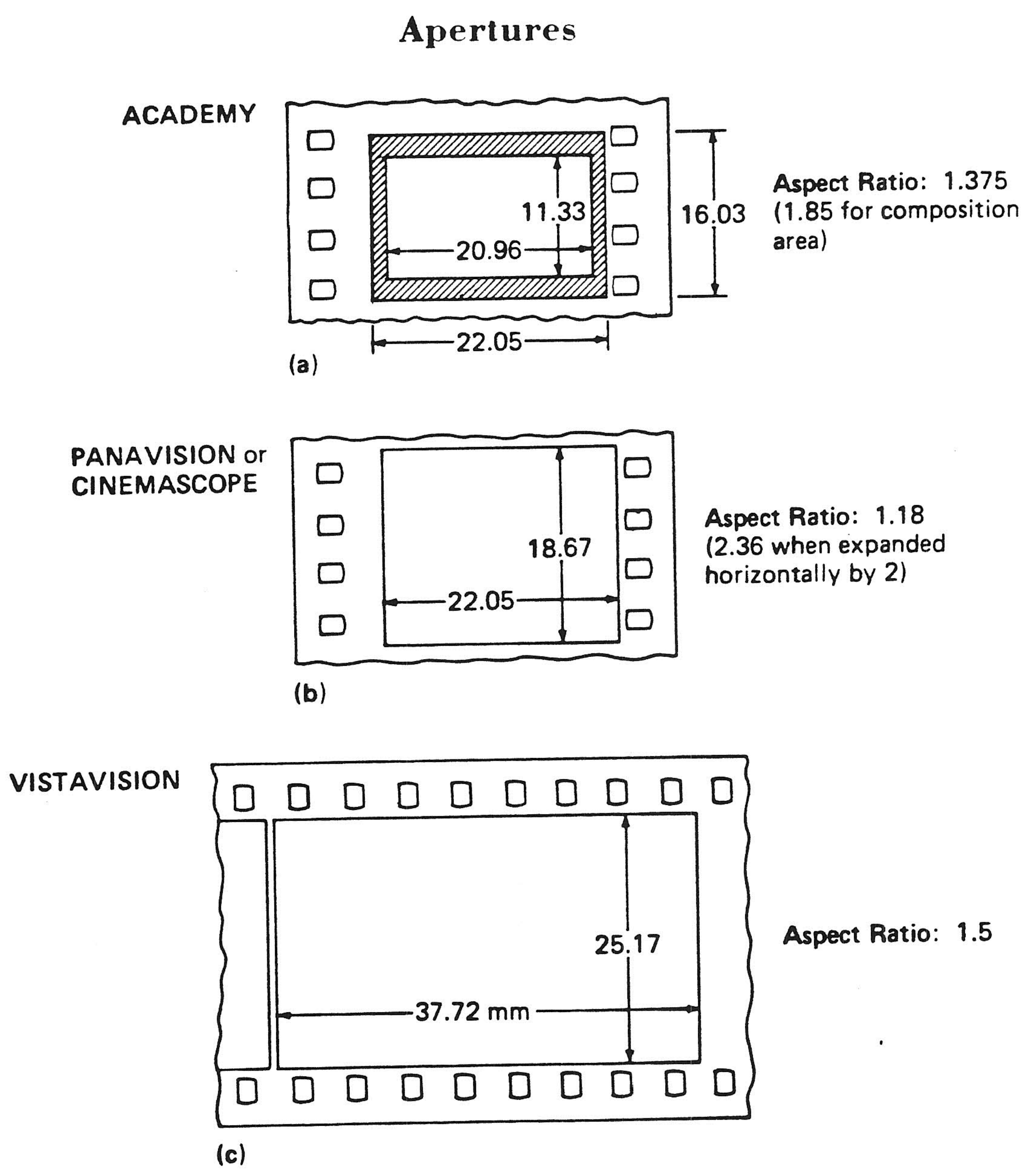
A class of modelers which are particularly difficult to interface are the patch design systems. Patches—and, particularly, rational bicubic patches—have many degrees of freedom. For example, bicubic patches have 16 degrees of freedom each, and rational bicubic patches have 32 each. They obtain their power from this freedom. The most promising technique for interaction with these modeling primitives (or any primitives, actually) is via stereo pairs of 3-D line drawings. No such system is currently in use.

Rendering: 2-Dimensional Problems

After a set of models has been digitized or synthesized, combined into scenes, and animated, we presumably have a set of frames in some conventional 3-D database format which then has to be rendered frame-by-frame into final 2-D form in full color with all hidden surfaces removed.

One of the most powerful tricks used in computer graphics is that of *texture mapping.* This is the wrapping of a 2-D picture onto a 3-D surface to give it much more detail than the modeling process feasibly admits. That is, if the heights of surface details on an object are very small with respect to the gross dimensions of the object, the details may be approximated by details with no height. At a sufficient distance a picture of bricks suffices in place of a 3-D model of bricks with rough surfaces and recessed mortar separators. This is similar to the old movie trick of the painted backdrop to represent scenes at a great distance from the camera. I include in the *rendering* section of this article the generation of the textures used for this process since this is basically a 2-D surface problem and not a 3-D object definition problem. Although the *rendering* domain yields strictly 2-D pictures, it is naturally subdivided into intrinsically 2-D and intrinsically 3-D problems, 3-D in this case meaning that the source of the information is 3-D, such as a model database. Two-dimensional problems will be considered first.

Texture Generation. The richest source of readily available imagery for simulated scenes is the real world. Since it is generally difficult to model the real world with anything like its complexity, synthetic models are enriched by mapping 2-D digitized real-world scenes onto their surfaces [*Textures* figure (a), (b)]. 2-D input scanning is simpler than 3-D—it requires only one camera, no triangulation, and cheaper equipment—but it suffers from some of the same problems (noise, in particular). A partial solution is so-called *flat-field correction.* In the case of reflected light, this is accomplished by scanning in a flat, white sheet of paper. Any variation in the digitiza-


Apertures
ACADEMY
11.33
20.96
16.03
Aspect Ratio: 1.375 (1.85 for composition area)
22.05
(a)
PANAVISION or CINEMASCOPE
18.67
22.05
Aspect Ratio: 1.18 (2.36 when expanded horizontally by 2)
(b)
VISTAVISION
25.17
37.72 mm
Aspect Ratio: 1.5
(c)

tion of the resulting signal is assumed to be noise. All subsequent scans are modified by the flat-field correction. Various filters for edge sharpening and contrast enhancement may also be used.

Another source of textures are the so-called "paint" programs which allow an artist to paint into a computer memory in a way that feels like painting directly onto a color video monitor. Trying to explain a paint program to someone who has not seen one is quite difficult. One demonstration is worth a thousand pictures; and words are almost useless until after the fact. Nevertheless, a paint program can be described as a tablet-driven interface to a color monitor. An artist moves an electronic stylus over an electronic tablet which returns (x,y) coordinate pairs to a computer. The computer maps these locations to corresponding locations on the video monitor and displays a cursor, such as a small arrowhead, there. A small switch in the tip of the stylus can be activated by the artist by simply pressing lightly against the tablet surface. This causes any number of things to happen, depending on where on the tablet the pressure is exerted. The action from which the program derives its name is a simulation of painting. If the artist moves on the tablet while keeping the switch closed, the video monitor displays a trail of color corresponding to the path traced on the tablet. Since it happens at computer speeds, it feels as if the artist painted the path of color directly on the screen. The size, shape, and color of the painted stroke can be selected by the artist through other tablet-controlled actions. Despite the clumsiness of this description, the reality of a good paint program is grace and simplicity. Children learn to use such programs readily.

These programs may be quite elaborate. In fact, a full paint program is really a system of programs. Such a system typically includes basic (simulation of) painting; filling of areas of arbitrary shape; line and curve drawing aides; picture saving and restoring; magnification, cursoring, color palette setup, and brush definition programs. Some paint stations include 2-D input scanning also. There are now several commercially available paint stations: "Superpaint" from Aurora Systems Inc. in San Francisco, "Images" from the Computer Graphics Lab Inc. in New York, and "Paintbox" by Quantel in Great Britain, to name a few. The most sophisticated such program has recently been written by Tom Porter at Lucasfilm [*Fractal* figure (b)].

The third technique of texture definition is algorithmic. This of course is as general as can be imagined in the context of computer programming. For example, a program which generates z as a function of x and y and represents z with shades of a color, or different colors, is a powerful source of often intriguing patterns for texture-mapping [*Textures* figure (c)].

Usually a texture is stored in a computer as a set of samples taken at equally spaced values of x and y. Textures may go through changes on quite a large scale during the mapping process onto 3-D surfaces. For example, a Mercator projection of the earth might be mapped onto a tiny sphere occupying only a hundredth of the screen in the first frame of an animation, and then onto a very large sphere—so close that only a portion of it near the equator is visible—in the last frame. One might think of this as one sphere and one texture mapping which gets larger during the animation; but, to avoid cumulative errors, it is the custom in computer graphics to enlarge the sphere and recompute the texture mapping for each frame. Theoretically it is possible to reconstruct a set of samples representing the texture, make a change of scale on the reconstruction, and resample with no loss of information. Practically, however, finiteness precludes exact reconstruction and resampling so it is done only approximately. Other approximations are introduced to speed up computations. One such technique is to prescale a texture destined to go through many scale changes in an animated sequence. The prescaling is typically done to powers of two. That is, the texture is prescaled by factors of 2, 4, 8, . . ., and ½, ¼, ⅛, ... Then, when a scale change is required, the texture already nearest in size is used for the slow, final mapping. This technique was first described by Ed Catmull at the University of Utah many yeas ago, and has been refined into the *mipmap* technique by Lance Williams of the New York Institute of Technology. It was also used by Tom Duff at Lucasfilm in the Genesis Demo sequence in *Star Trek II* [*Procedural* figure (a)].

The mechanics of assigning textures to surfaces is a large problem in itself, and has not yet been adequately solved. The earth-like planet in the Genesis Demo sequence was created by painting a texture with the Lucasfilm paint program (Chris Evans from Industrial Light and Magic, the Lucasfilm special effects division, did the painting). Then Tom Duff mapped this texture onto a sphere for the finished planet. Spheres are one kind of surface for which the texture assignment problem has been solved. Assigning a texture to the surface of a single bicubic patch is also straightforward. The difficult thing is mapping textures to complexes of quadric surfaces or patches.

Matting. Another 2-D color rendering problem is that of compositing several 2-D pictures into a single final frame. Since interesting scenes tend to be complex, they are typically broken into several *elements*. For example, in the Genesis Demo sequence, every frame is composed of from two to five elements. A frame showing a projectile impacting a planet surface has a starfield element, a planet element, an explosion element, and a shockwave element. Each of these is the final result of a 3-D rendering problem, but none is a complete frame [see also *Procedural* figure (a)]. A final 2-D *matting*, or compositing, step is applied to combine the elements. This is straightforward because a perfect matte is generated with each element as part of the 3-D rendering process.

A *matte* is simply a picture which is assigned the value 0 wherever another picture of the same size is to be transparent, 1 where it is to be opaque, and a fraction between 0 and 1 for corresponding partial transparency. Let a be a color from picture A and b be a color from picture B. A is to be placed over B with a registered over b. Let α be the partial transparency of pixel a—in other words, α is the pixel in the matte for A which is in registration with pixel a. The operator for combining the two pictures according to the given matte is used so often in computer graphics that it has a special name, *lerp*, short for "linear interpolation". Lerp is defined as the pixel-by-pixel operation

$$\alpha*A + (1-\alpha)*B = B + \alpha*(A-B).$$

That is, it is the linear interpolation of B to A by amount α known to be between 0 (transparent) and 1 (opaque).

A closely related film technique is called *bluescreen matting* [see the *Movie Glossary* and *Bluescreen* figure]. The main difference is that the matte used for lerping a foreground to a background with the bluescreen technique is not given, but must be extracted from the foreground image, a task generally requiring human intelligence. In a feature filmmaking environment, a complete digital imagemaking system must simulate the bluescreen process, since this is the principal way of integrating real-world and synthetic imagery. Tom Porter has implemented such a system at Lucasfilm.

Bluescreen. **A digital simulation of the bluescreen process. The original foreground and background elements are at the top. The cleaned-up foreground and resulting composite are at the bottom. [Courtesy Tom Porter, Lucasfilm.]**

Matting is one way of partitioning a scene so that hidden-surface removal becomes simple. The more general, more difficult problem of hidden surfaces in 3-D is covered in the next section.

Rendering: 3-Dimensional Problems

The 3-D problems discussed in this section tend to be the focus of computer graphics. This is because they are the most difficult problems among all those in a full 3-D animation system; they require the most sophisticated algorithms and use the largest number of computer cycles. Nevertheless, they are just part of the final picture.

Hidden Surface Removal. In general, the removal of hidden surfaces from a synthetic scene cannot be solved with simple tricks such as 2-D matting of elements as discussed above. Consider a complex scene which has been animated and is ready for rendering into a frame. Assume that this scene is of a complexity approaching that of the real world, and that it is to be rendered at film resolution. No one has yet decided what the optimal film resolution for digital pictures is, but it is safe to say that there will be millions of pixels involved (with the range being anywhere from 1 to 64 million). Further assume that all primitives used in the modeling stage are subdivided into polygons for this rendering stage. This is not necessarily the case, but it is done often enough to make our example reasonable. So each leaf, car, face, blade of grass, rock, river, etc., is sufficiently subdivided to show no sampling effects. By looking around in the real world, it is apparent that depth complexity—the number of surfaces crossed by a straight line from the eye to infinity—is typically something between ten and one hundred. So potentially we are talking about many millions of polygons of arbitrary shape and orientation, stacked, perhaps, about twenty deep at any one pixel. The analysis of which surfaces hide which in this scenario has to be solved for every frame! Since a movie has 100,000 to 150,000 frames, it is easy to see just how computationally intensive fully-realized 3-D computer animation may be, and why so much work has been—and still is—put into finding algorithms for solving the hidden-surface problem.

I will not attempt to analyze the problem fully here, but will just indicate some of the major approaches. There are many tricks for reducing the complexity of a scene before the brute-force hidden-surface algorithms are put to work. One is the use of matting discussed previously. This is just a special case of what is called *clustering*, or breaking a scene into modules which can be treated in complete independence of one another.

A process of *culling* may quickly reduce the "environment" of objects which have to be considered in a frame. This is simply the removal from the working database of all objects which are clearly offscreen or behind the camera.

The use of *coherence* can greatly reduce the amount of computation. Coherence is a measure of how much a frame is like the one just solved, or how much a scanline resembles the one just processed, etc. In other words, many algorithms are sped up by noticing when a computation is to be performed again and saving the current result for that later time. There are many different types of coherence which may be exploited to save time.

In the brief summary of the major hidden-surface solution techniques which follows, I will use the terms *object space* and *image space* frequently. Object space is the coordinate system used for modeling the scene—or perhaps a translation-rotation of that system into a more convenient one, such as a coordinate-system based at the camera. In any case, it is a Euclidean 3-D space for describing the modeled scene with real numbers and continuous curves (at least to within the floating-point accuracy of a computer). Image space is the coordinate system of the plane in which the final image of the model is projected; it is typically represented with integers in a computer. It implies perspective projection and the representation of the projected model by numbers at or near the resolution of the display. Although the image is 2-D, a third dimension obtained from the perspective projection is kept for depth comparisons.

Ray Tracing. Some of the most beautiful computer-graphics pictures so far obtained have used *ray tracing* algorithms for hidden-surface detection and removal [*Teapot* figure (c)]. The idea is to follow all the visible rays emitted from light sources in a modeled scene through all their reflections and refractions by various objects until they strike the image plane, and then the eye or camera. Thus the laws of optics are applied faithfully. For computational purposes, these rays are traced in the reverse direction, from the eye to the emitting source. Since a ray can bounce off any object in the modeled scene, the scene cannot be culled. For example, a mirrored sphere in the camera's view may reflect what is behind the camera—and the camera itself. This type of algorithm is performed in object space; it tends to be slow, and makes neighborhood integration for antialiasing filters difficult (see section on *antialiasing* below), but it does provide shadows and handles transparency well. It can be used for any surface primitive for which intersection with a ray can be solved. Essentially, a new hidden-surface problem has to be solved every time a ray bounces off an object, so the problem is recursively difficult.

Depth Buffer. A very simple hidden-surface solution uses a large piece of memory called a *Z buffer*, or *depth buffer*. A Z buffer has one location for each pixel in the final image, so that a Z-buffer type of algorithm is intrinsically an image space algorithm. A pixel portion of a surface in a scene is written into the final image only if its z-coordinate is less than that stored in the Z buffer at that point. (It is fairly common practice in computer graphics to use an image space coordinate system with x being horizontal, y vertical, and z increasing into the screen away from the viewer.) If the new pixel hides the old, then its z value replaces that of the old pixel in the Z buffer.

The benefits of this approach are: No ordering of the environment is required—that means, no sorting. Several different programs can be used to generate objects independently and sequentially so long as they can access a common Z-buffer. Such programs are very simple to write. The method is not based on any one kind of primitive.

However, this approach fails to render transparency properly, since a knowledge of order of the surfaces is lost. Furthermore, it does not support antialiasing since it is basically a point-sampling idea (see the section on *antialiasing* later in this article).

Sorting and Clipping. The majority of algorithms which have been devised for hidden-surface resolution are polygon-based. This is really no great restriction so long as all primitives used for modeling can be subdivided into polygonal approximations. There have been perhaps twenty to thirty such algorithms devised so far; they differ by the order of sorting performed, the type of coherence exploited, and the question of whether image space or object space is used.

A typical example of this type of algorithm assumes a culled database which has been sorted in order of the maximum y for each polygon. Now the data space is subdivided for further sorting. For example, it may be divided into scanlines—that is, into horizontal slices which are to be resolved into one horizontal scanline of a display device or medium. In such *scanline oriented* algorithms, the polygons which intersect the given scanline, defined by a plane passing through the scanline and the viewpoint, are further sorted on their minimum *x* coordinates. Then a variety of depth-sorting techniques are used to make the final resolution. Since depth information may change only slightly from scanline to scanline, information on one line may be used to simplify computations on the next. This is an example of the use of coherence.

Another way to subdivide the space for further sorting is by dividing the image plane into a grid of squares and solving within the squares. The database of polygons is *clipped* against the current square of interest to subdivide polygons into subpolygons with new edges coinciding with the current square's edge. A square which is particularly difficult to solve may be further subdivided into smaller squares before final resolution is attempted. Some algorithms use polygons in the scene to clip against rather than an arbitrary grid of squares.

A few of these algorithms handle shadows; most do not. Some handle transparency, others don't. Most can be made to deal appropriately with antialiasing. They are sort intensive and tend to bog down for large numbers of polygons (millions). Some are plagued by a rapid increase in the number of subpolygons generated at intermediate steps, and others by the extreme shapes these subpolygons may take (slivers). Generally they are not designed to solve the optics problems solved by the ray-tracing algorithms; essentially they handle only one bounce of a ray.

Coloring. After the hidden-surface problem has been resolved, there still remains the actually rendering of color at each pixel in the visible surfaces, which may be only transparently visible. The color is a combination of many factors: the sources of illumination, including the location, spread, and color of each of them; the surface attributes of the object being illuminated, including the surface material and its roughness and glossiness; and the mappings used by computer graphicists to give non-modeled complexity to the surface.

Lighting Models and Surface Attributes. An environment may include several light sources, each of which affects the color of an object—either directly, or by shadowing, or by filtering through partially transparent objects or partially reflective objects. A light source may be local and distributed over space, or it may reside at a point at infinity. It may be in the scene viewed or off-camera. It may be colored. Simple computer graphics tends to use one white light source, located at infinity and out of camera range. But in general, this is restrictive, and insufficiently realistic.

A typical model for illumination by a light source will have a specular component, a diffuse component, and an ambient component. The specular component is that seen when the viewpoint is at or near the point where the angle of reflectance of a light ray from a surface is equal to the angle of incidence. It dominates on shiny surfaces. The diffuse component is more uniformly distributed over an illuminated surface, and dominates on matte, or dull, surfaces. The ambient component is a constant term, added to capture the notion of a low background glow due to the diffuse reflections of many low-level light sources. All lighting models substitute approximating terms for factors which would require very complex integrations for truly accurate modeling.

One common approximation (which should, however, be avoided for good color renditions) is that which gives the specular, diffuse, and ambient components the color of the light source. Real objects modify these three components in three different ways. For example, metals have specular highlights colored by the metal, not by the light source (and almost no diffuse component). Notable improvements in computer graphics scenes are obtained by the simple measure of using better lighting models. Rob Cook at Lucasfilm has achieved striking results by choosing his lighting model approximations carefully and heeding the significant aspects of surface materials [*Teapot* figure (b), (c)].

Texture Mapping. As has already been mentioned, a great amount of complexity may be apparently added to a synthetic scene by mapping a 2-D picture, or "texture," onto a 3-D surface [*Textures* figure]. The logistics of doing this mapping may be one of the most difficult parts of a 3-D animation exercise. Textures tend to require large amounts of disk space. Since disks are as yet relatively slow devices, it is important to have texturing algorithms which minimize disk accesses.

Bump Mapping. A simple and very powerful way of adding further complexity to a scene is based on the same principle as is texture mapping. This principle is that if the amplitude (height) of the detail is small relative to the amplitude of the object with the detail, then a satisfactory approximation is obtained by reducing the detail amplitude to zero—that is, by using a 2-D representation of the detail. Most lighting models vary the shade of an object as a function of the normal to the surface of the model. *Bump mapping*, or *normal perturbation*, invented by Jim Blinn, simulates wrinkles and bumps of low amplitude in a surface by appropriately wiggling the surface normal in the vicinity of the bumps and wrinkles as if they really existed in the model [*Procedural* figure (a)]. Only at the silhouette of the object is the secret given away—there are no bumps or wrinkles actually visible—but with enough complexity, viewers tend to ignore this fact.

Environment Mapping. An alternative to ray-tracing is what is sometimes called *environment mapping*. If an object does not move with respect to its environment, then it is possible to compute a texture map which takes optics fully into account. When this texture is mapped onto the surface at rendering time, the effect is the same as if ray-tracing had been performed. In fact, it has—but only once, not for every frame. This environment mapping trick can be used in cases involving little or no motion to simulate difficult surfaces such as chrome bumpers and shiny doorknobs.

Antialiasing. I have mentioned antialiasing several times in this section. It is one of the most important topics in computer graphics, and is still not fully understood by the community of graphics users. Unfortunately, it is much simpler to ignore antialiasing when writing a rendering program than to think it out and include it. As we have seen, some algorithms do not even allow for the incorporation of antialiasing. These will fall away as the importance of antialiasing becomes widely known.

Aliasing is most often seen as "stairsteps" or "jaggies" in computer-generated frames [*Patch* figure (c)]. In motion, these jagged edges ripple along, destroying the illusion that we are looking at the edge of a hard object. Other manifestations of aliasing are polygons that flash on and off and "strobing" or temporal aliasing. Perhaps the most familiar example of temporal aliasing is provided by the stagecoach wheels that seem to spin backwards in westerns.

Spatial. Aliasing is a sampling problem. Computer graphics typically uses discrete samples of real surfaces rather than continuous surfaces, and a given surface is eventually displayed in a pixelated form (a pixel being a sample of the continuous surface). Sampling theory indicates that the samples are adequate for the purpose of representation if there are no direction changes in the surface within the distance between pixels or samples. More strictly stated, the sampled surface should have no frequencies, in the Fourier sense, of higher spatial frequency than half that of the sampling grid. Unfortunately, polygons, which are very often used in computer graphics, have very high frequencies near their edges. Some kind of local filtering or averaging must be performed to rid the surface of these frequencies, or else they will appear as ragged edges in a sampled display.

Temporal. A movie is a set of samples taken twenty-four (for film) or thirty (for television) times per second of a 3-D event, where one of the dimensions is time. Just as in the spatial case, aliasing will occur if the image changes due to motion occur with a frequency higher than twelve or fifteen times per second. A sharp edge moving rapidly across a frame should be seen as a smooth sweep. The frame sampling causes instead a ragged succession of spatially separated instances of the edge. This is referred to as "strobing". It is worsened by the fact that current projectors flash each frame two or even three times before moving to the next frame, confounding the motion prediction function of the human brain and causing us to perceive several copies of the temporally jagged edge. The solution is called *motion blur.* The idea is to simulate the image that would be exposed on a frame of movie film by leaving the shutter open for some length of time during which the object being filmed moves. Unfortunately, motion blur is now usually omitted from computer graphics since it is so computationally expensive. However, the stars and fires in the Lucasfilm *Star Trek II* sequence were motion-blurred [*Procedural* figure (a)]. Full motion blur of articulated objects moving independently of the camera is yet to be accomplished.

Logistics and Process

A subject which usually takes a back seat to the more glamorous aspects of computer animation—modeling and rendering—is that of logistics. It is readily apparent, however, in any real computer animation project that the management of resources—disks, magtapes, CPUs, people, film units, painting stations, and so forth—is one of the toughest problems, severe bottlenecks resulting if it is not adequately handled. In fact, one of the principal uses of the computer in digital filmmaking is the management of these resources. Lucasfilm's Malcolm Blanchard was the first computer scientist to get film credit for database management (in *Star Trek II*).

One powerful logistics tool is the *exposure sheet,* which is commonly used to specify what elements and what backgrounds go into an animated sequence, frame by frame. The exposure sheet contains information about compositing and camera or videotape control (advance, skip, reverse, etc.). Facilities for exposure sheet generation and maintainance are mandatory and can become surprisingly complex. Since mistakes always happen, any exposure sheet facility must allow easy recovery in case of error. With complex relationships between the elements of a sequence and where they are stored, this can be very difficult and lead to further errors in the recording.

It is important to note that a great deal of creativity may be exercised in the creation of an exposure sheet. The digital computer permits a freedom of arrangement of elements in space and time never before available. The notion of filmmaking as we currently know it hinges on the power inherent in the freedom to edit, discovered in the early days of the art. Editing is the freedom of arranging scenes in time. The computer enhances this freedom by extending it to the arrangement of elements within the frames of a scene.

Final Remarks

Computer imagery will be used in a self-referential way for the next five or ten years. That is, movie-makers will advertise their "made-with-computer graphics" and exploit the technical novelty involved. By the end of this decade, this medium will have been incorporated into our visual language, its novelty will have worn off, the mechanics of its production will have become sleek and efficient, and directors will use it as just another tool—as they now use camera imagery, not for the novelty of photography but for the kind of expression it can deliver to film. But whereas the camera's extension of the language was not obvious in the beginning, that of the computer is.

The consequences of the extension of editing into the spatial domain, made possible by the computer, are yet to be fully explored or even understood. The notion of compositing will be lifted from its current realm as a "special effect" to that of a "tool of the trade."

The full impact of the digital computer on filmmaking involves much more than just the visual aspects discussed here. Digitally synthesized audio, digitally controlled mixing of sound tracks, computer-assisted editing, digital database control, computer-controlled cameras, and electronic storyboarding are examples of parallel research efforts, each of which is worth an article in itself. The Grand View has all these aspects closely interlinked by computer networking, and fully integrated into the filmmaking process.

Bibliography

Foley, J. D., and Van Dam, A. *Fundamentals of Interactive Computer Graphics.* Menlo Park, California: Addison-Wesley, 1982.

Mandelbrot, Benoit B. *The Fractal Geometry of Nature.* Revised Edition. San Francisco: W. H. Freeman and Company, 1982.

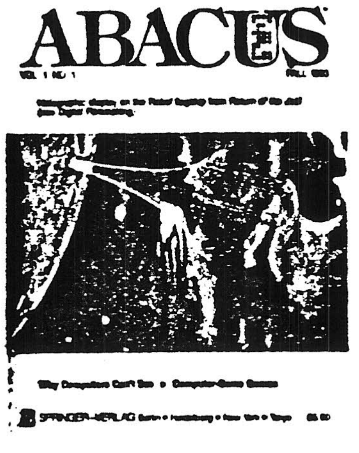
Newman, William M., and Sproull, Robert F. *Principles of Interactive Computer Graphics.* Second Edition. San Francisco: McGraw-Hill, 1979.

Smith, Alvy R. Special Effects for Star Trek II: The Genesis Demo, Instant Evolution with Computer Graphics. *American Cinematographer* 63 (1982): 1038–1039, 1048–1050.

ABACUS
VOL. 1 NO. 1 FALL 1983

Subscription Information: Vol. 1 (4 issues) will be published in 1983–84.
North America. Vol. 1, US$ 18.95 including postage and handling. Subscriptions are entered with prepayment only. Orders must be placed directly with: Springer-Verlag New York Inc., Service Center Secaucus, 44 Hartz Way, Secaucus, NJ 07094, USA.
All Other Countries. Vol. 1, DM 46,— plus postage and handling. Orders can be placed with your bookdealer or sent directly to: Springer-Verlag, Heidelberger Platz 3, D-1000 Berlin 33, Germany.

CONTENTS

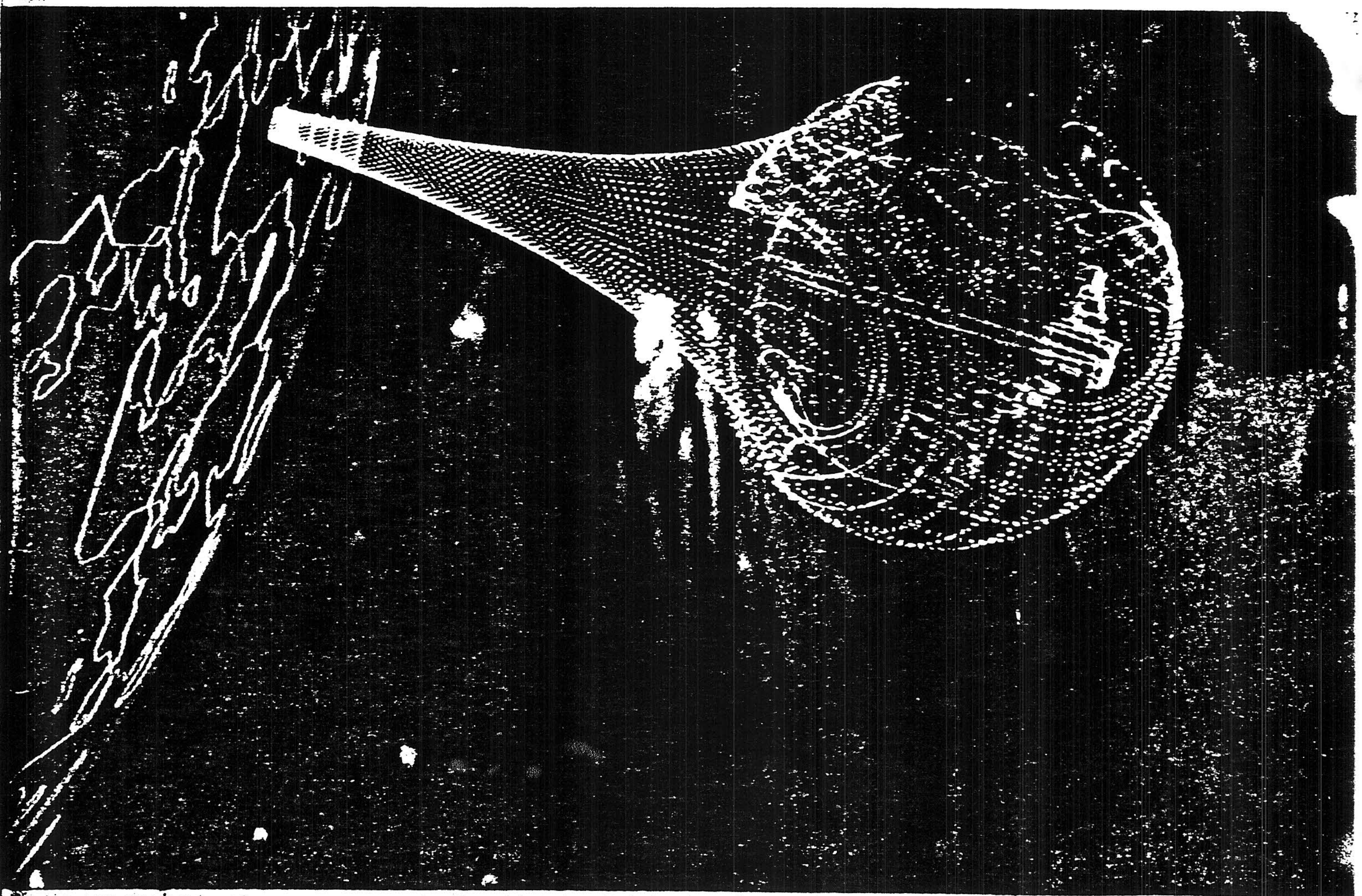


COVER PICTURE. William Reeves and Tom Duff of Lucasfilm's Computer Graphics Project generated this "holographic display" for a scene in *Return of the Jedi*, the third film in the *Star Wars* series.
This example of vector graphics was produced on an Evans & Sutherland Picture System II. The language C was used exclusively for the programming, under the Unix operating system on a VAX 11/750 from Digital Equipment Corporation [© 1983 Lucasfilm Ltd. All Rights Reserved.]

ABACUS™

VOL 1 NO 1 FALL 1983

Holographic display on the Rebel flagship from *Return of the Jedi* (*see* Digital Filmmaking)



Why Computers Can't See ■ Computer-Game Games

SPRINGER-VERLAG Berlin ■ Heidelberg ■ New York ■ Tokyo $5.00