

A System for Computer Generated Movies

Edwin Catmull, University of Utah

With the recent developments in fast hidden surface algorithms and a method for smooth shading of half-tone pictures, it has become feasible to generate useful movies with the computer. This paper describes a system used to make computer generated movies. It also explains the methods used to attempt to solve the problems of object representation, object manipulation, concurrent motion, and ease of specifying motion. The system was first used to make a movie of a hand that lasts for little more than a minute.

KEY WORDS AND PHRASES: half-tone computer graphics, hidden surface, polygonal surface structure presentations, animation, graphic language, movie
CR CATEGORIES: 3.41, 4.29, 8.2

INTRODUCTION

Computer generated pictures and movies have been made by artists and computer scientists for several years now, and have recently caught the public eye. Most of the computer movies, however, have consisted of line drawings. The state of the art in half-tone pictures was not advanced enough to make it reasonable to produce half-tone movies. Half-tone pictures are displayed on a screen much as a TV picture is presented: surfaces are painted with differing shades on scan lines. The most basic problem with half-tone pictures was deciding which surface was in front and therefore should be painted on the scope. Several algorithms have been developed for different classes of surfaces. Most were quite time consuming. Recently there have been fast hidden surface algorithms for surfaces made up of polygons

* This research was supported in part by the University of Utah Computer Science Division and the Advanced Research Projects Agency of the Department of Defense monitored by the Rome Air Development Center, Griffiss Air Force Base, New York, under contract number F-30602-70-C-0300.

(1,2,3). This research uses Watkins (3) algorithm since it has been implemented in hardware and is remarkably fast. One of the objections to these algorithms is that all surfaces must be made up of polygons. This is a severe restriction for curved surfaces such as spheres or skin. Fortunately, Henri Gouraud (4) has developed a method, called smooth shading, for making polygonal surfaces appear curved.

With these developments, it has become feasible to generate useful half-tone movies with the computer. This paper describes a system used to generate movies. It explains the methods used to solve the problems of object representation, object manipulation, concurrent motion, and ease of specifying motion.

THE SYSTEM

Figure 1 shows a block diagram of a system for making movies. The high precision scope is a raster scope similar to a TV but with higher resolution and quality. The hidden surface removal is done by a routine developed by Gary Watkins (3) which operates on polygons. The smooth shading was developed by Henri Gouraud (4). It is necessary to send these routines the points of a polygon, the normals at each point for shading, and the whiteness or colour for each polygon. All objects displayed by this system must be made up of polygons. This is usually not a serious limitation because of the smooth shading.

DATA DESCRIPTION

For the purpose of this paper, an "object" or "part" is a group of polygons that will transform together and a "body" is an organized group of objects. For example, the body of a hand is made up of several smaller objects: the palm, the bottom of the thumb, the middle of the thumb, the top of the thumb, etc. The objects are organized in a tree structure. By using a tree structure, any transformation applied to an object at a node also applies to its children.

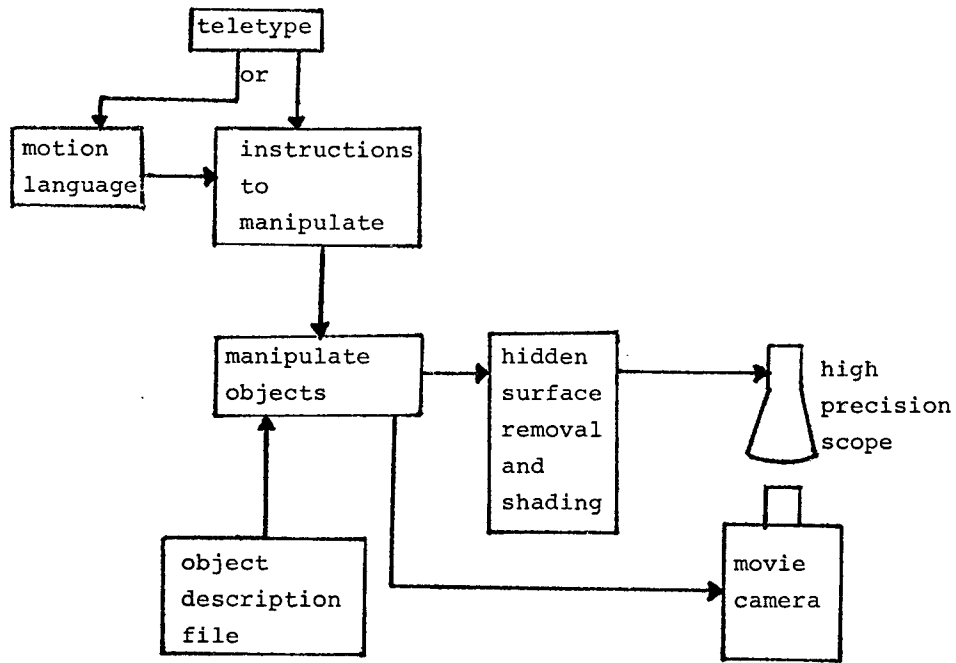


Figure 1.

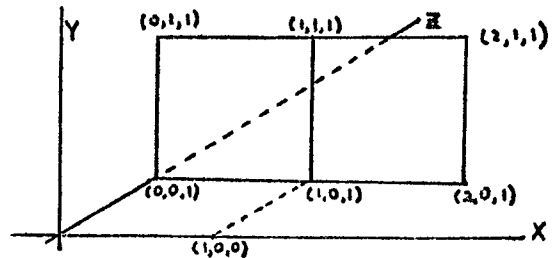
If one rotates the bottom of the thumb, the other two parts of the thumb follow automatically. Each object is described as follows:

father
brother
points
polygons
axes
named positions

The points are a numbered list of x , y , and z coordinates. Polygons are described by a list of numbers where each number refers to the correspondingly numbered point. If the number is preceded by a "+" the point is in the object named as father. If the number is preceded by a "-" then the point is in the object named as brother. This way objects can be connected together by having polygons with points in different objects to allow for flexible surfaces. Each object can be assigned a color.

One can define an axis of rotation for an object relative to its parent object. The axis of rotation is defined by three points where the axis passes through the second point and is perpendicular to the plane defined by the three points. A positive rotation is taken from the first to the third point. One can name different degrees of rotation. For example, in a forearm, we might call zero degrees OPEN and 170 degrees CLOSED. Then when one wants the arm to go straight, regardless of the position it is in, it is only necessary to give an instruction that the forearm go to position OPEN, rather than having to keep track of the position. This can prevent losing track of where one is at after making many changes.

For example, to describe a body consisting of two objects where each object consists of one polygon.



```

NAME=FLAP1
POINTS
1  0.  0.  1.
2  1.  0.  1.
3  1.  1.  1.
4  0.  1.  1.
5  1.  0.  0.
POLYGONS
1 2 3 4
END FLAP1
NAME=FLAP2
FATHER=FLAP1
POINTS
1  2.  0.  1.
2  2.  1.  1.
POLYGONS
1 2: ↑3 ↑2
AXIS 1 ↑1 ↑2 ↑5
FLAT=0
CLOSE=90
END FLAP2

```

This method of data representation is fairly convenient to use but still does not solve some of the more difficult problems of flexible surfaces. Careful inspection of the hand (figure 2) when the hand is closed will show that the joints become somewhat distorted. This is not just a matter of choosing wrong axes of rotation. Two possible solutions to this particular problem are:

1. Interpolation between two states taking into account rotation. Unfortunately, it would be extremely difficult to get the coordinate definition of a closed hand.
2. Generating the polygons of a flexible part using some curve fitting or B-spline technique. This has yet to be fully worked out.

MOTION - THE OBJECT MANIPULATION ROUTINE

The routine MOTION accepts instructions to manipulate objects, prepares the data for sending to the hidden surface algorithm, and controls the camera. The instructions to MOTION can come from either a teletype or a file. The most commonly used instructions are:

```

ROTATE <object name><axis number><degrees>
MOVE <object name><dx><dy><dz>
DISPLAY

```

There are other instructions to control resolution, background, intensity, camera, the instruction file, etc. There are instructions to return the current state of an object. For example, a controlling program may request the degrees of rotation that the top of the thumb has made or how many degrees the thumb must rotate to reach a given named position. This makes it usually unnecessary to keep track of the changes made to an object.

For a movie of a hand, each frame may require as many as 14 instructions, i.e. the fingers close while the hand

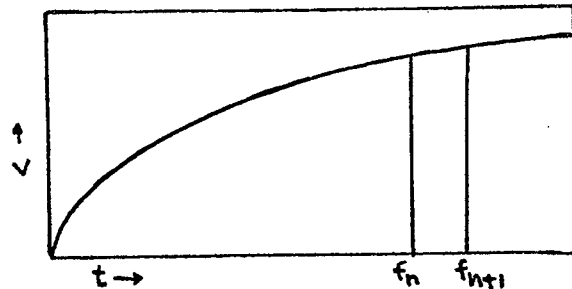
rotates. In order to make a movie of only one minute (1440 frames), about 13,000 instructions must be executed. It is immediately apparent that it is unreasonable to specify frame by frame a complex behavior that lasts for more than a few frames. The problem is compounded by the fact that objects may move independently in time, thereby making normal looping techniques inadequate to generate the instructions needed.

MOP - A MOTION PICTURE LANGUAGE

Typical programming languages are much too sequential in nature for specifying easily the kinds of simultaneous and overlapping action we expect in a movie. There have been some animation systems that have tried to solve this problem (5,6,7,8). The language MOP is designed to allow concurrency from the point of view of the user. It generates instructions for MOTION which changes and displays the objects.

Figure 2 shows a sequence of 11 frames where a hand first closes and then opens in a fanout way. Notice that the motion starts slowly, speeds up, and then slows before stopping. The rate at which an object moves or changes can be specified by either some mathematical formulation (in this case a sine curve) or a table of empirical data. The program then integrates the area under some portion of the curve to determine the movement for each frame.

For example, if one wanted an object to accelerate (animators refer to it as a slow-in), a table could be created as follows:



The user would have to specify the table either by guesswork or from empirical data. The table can be used for any frame period. The number of frames in the period is the number of divisions along the time axis. The movement in each frame can then be easily calculated.

Most statements in MOP are of the following form: <label> <frame period> <instruction> <parameters>. The label is an alphanumeric name followed by a colon. It is optional and is used to name a statement. It is not used for transfer of control since control is not sequential.

Figure 2. A sequence of 11 frames showing the hand close and open. Notice the rate of change is not constant.



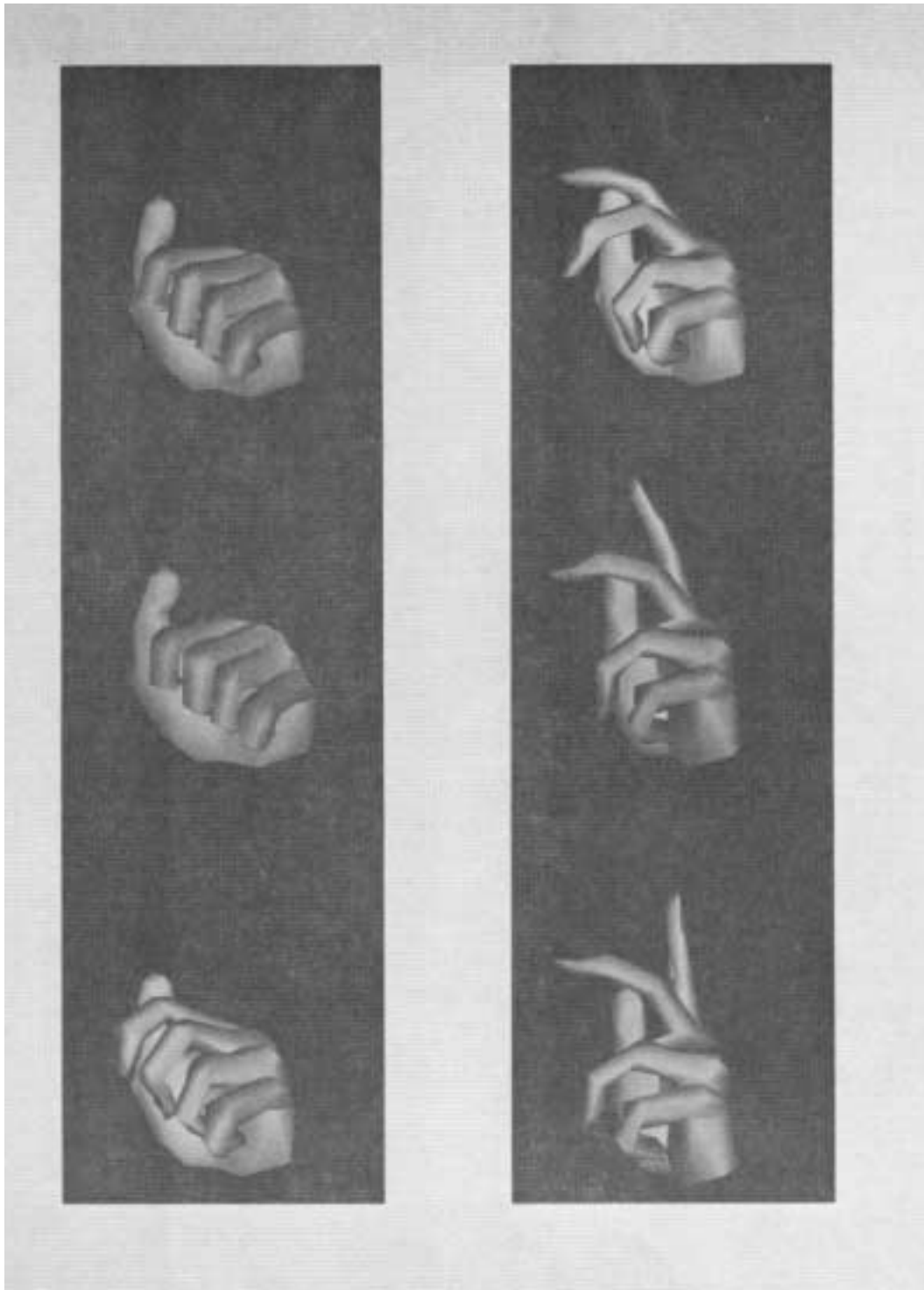




Figure 3. A line drawing of the hand as it was input.

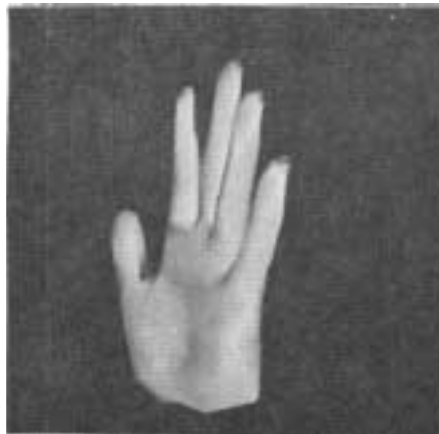


Figure 4. The hand as displayed using hidden surface and smooth shading algorithms.



Figure 5. The hand as displayed using hidden surface algorithm but no smooth shading. The individual polygons can be seen.

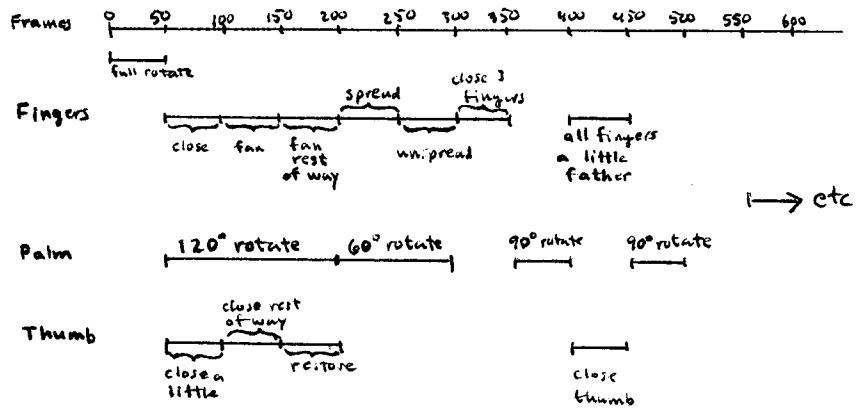


Figure 6.

The frame period is used to specify over what frames the statement is executed and at what rate changes occur. It is of the form:

```
<arith. expr.>,<arith. expr.>,<table name>
```

The first expression is the starting frame, the second is the last frame, and the third indicates which table or arithmetic formula is to be used to dictate the rate of change.

The instruction is either a reserved instruction or the name of another statement.

The parameters are arithmetic expressions or strings, the value of which may either be passed to another statement or passed to a reserved function for output.

When the program is read in, the frame sequence is either known or unknown. For example:

```
76,175,B ROTATE "PALM",1,30
```

has a known frame sequence 76-175 and will cause the palm to be rotated 30 degrees about axis 1 during that 100 frames. The amount rotated each frame is determined by the table or formula named "B". On the other hand:

```
LAB: FIRST,FIRST+50,B ROTATE "PALM",1,30
```

has an unknown frame sequence as long as FIRST does not have a value. Each statement is initially put into one of two lists: one for instructions where the frame sequence is known, the other for the unknown.

One of the reserved instructions is "BEGIN" which means that all of the following statements until an "END" will be grouped under the same label. However, the execution of each statement in the group is still determined by its own frame period specification.

A frame counter is started that is incremented for each frame. During each frame, every statement in the "known" list is checked to see if the frame counter is within its frame specification. If so, the statement is executed with appropriate modifications made depending on how far into the frame sequence the frame counter is. A statement may call by label a statement or group of statements from the "unknown" list to be in the "known" list by giving values to its undefined variables via the parameters. The new statements may take on the frame specification of the caller. This way routines of motion may be written.

The output is a file of instructions for the MOTION routine. A page and a half of coding in MOP to manipulate a hand to various positions for a minute movie will generate the some 13,000 changes that are made to the hand.

The statements to MOP may be in any order. This feature makes a considerable difference in programming ease.

THE MAKING OF A MOVIE

As an illustration of the capabilities of this system, a movie of a hand was made. A hand was chosen because it was difficult enough to show some of the possibilities and weaknesses of the system. The hand has a flexible surface and parts that move relative to other parts. The following steps were taken.

Data Acquisition

Although outside the scope of this paper, it should be noted that this is still a formidable task. For the hand, a plaster of paris mold was made. (A word of caution to anyone wishing to make a similar mold: it is more comfortable to shave the back of the hand before making the mold than to have the hairs pulled out by the mold.) Next, a plaster model was made from the mold and covered with a thin layer of latex material. Polygons were drawn on the latex. It was necessary to extract the three dimensional coordinates of some 270 corner points defining the surface of the hand, organize them into about 350 polygons, and organize the polygons into the parts of the hand. This was followed by determining the axes about which each part rotated. Figures 3, 4, and 5 show the polygons drawn as lines, polygon surfaces, and the smooth shaded version.

Mapping Out the Action

This consists of deciding what actions are to take place and when they are to occur. See figure 6. Some of this work can be done interactively. Baecker (7) has developed a scheme in two dimensions for specifying motion, however, this has not yet been worked out satisfactorily in three dimensions. Traditionally, the creation of an animated film begins with a storyboard. The system described in this paper is meant to be used with a storyboard format. However, the way one lays out the sequence of events is a matter of personal style.

Programming

The first step is to write routines of motion. As an example, the fingers move often enough that one might like to be able to type:

```
N,N+50 ROTFIN 45,45,50,60;
```

where the parameters specify how much the parts of the respective fingers would rotate. Once the routines of Motion have been written (in MOP) it is relatively trivial to program the movie

using the plan specified in the previous step. Each bar segment in figure 6 can usually be taken care of by a single statement. Again, the order of statements is inconsequential.

Filming

The instructions created by MOP are given to the MOTION program which gets the objects, makes changes on them, sends necessary information to the picture processors, and advances the camera. For debugging purposes, the half tone processor may be replaced by processors that paint the polygon edges on a vector or storage scope (figure 3).

Computers and Animation

The kinds of control needed for a film making language are:

1. that it must be set in a context of frames and concurrent actions.
2. the creation and manipulation in general of shapes and relations.
3. how motion occurs - specified by mathematical equations or empirical data.
4. to prevent conflict of objects because of accident or data error, i.e. one would not like to see an elbow bent the wrong way or a car driving through a tree.
5. keep track of where objects are at.
6. create routines of motion, i.e. a walking routine that could be applied to a person for an arbitrary number of frames.
7. naturalness in describing action, i.e.

IN FRAMES T TO T+400 WALK CHARLIE TO TABLE;

The system described in this paper gives most of the above controls to some extent. MOP is convenient for specifying concurrent actions in frames. It allows for rotation of objects in three dimensions in a very natural way. However, in general, there are many kinds of flexible surfaces, i.e. cloth and water, that are difficult to model. The best that this animation system can do is interpolate between extreme configurations of a surface. A great deal of research still needs to be done on the creation and manipulation of three dimensional objects.

For the kinds of motion allowed, the method of specifying action is very general. The problem still remains with the animator to dictate the rate of change of motion. This is a non-trivial problem if a nice-looking action is desired.

Conflict of objects is only partially taken care of. There is no check to keep one object from passing through another. This problem is partially taken care of because it is very easy to keep track of

the location and amount of rotation of any object.

MOP allows for the writing of routines of motion, but does not have a very natural syntax. It has been used for making several small movies, none of which had a story theme. The next step is to create a language with a better syntax.

CONCLUSION

A movie of the hand lasting more than a minute was made using the above system. The making of a movie is trivial compared to the acquisition of data at this time. Data is currently being gathered for the surface definition of a whole human body. At the time of this writing, a hardware implementation of Watkins algorithm has been completed, which is capable of displaying the picture data at a rate of 30 frames per second. At present it takes three to ten seconds to transform the object description to a format acceptable for the Watkins processor, but with the addition of a hardware clipper and matrix multiplier, the real time display of computer generated half-tone animated objects is a real possibility.

The system described in this paper has the kind of control needed to make movies that look good. For making movies, it is more desirable to be a director than a programmer, recognizing, of course, that being a good director is still a lot of work.

ACKNOWLEDGMENT

I would like to thank R. E. Stephenson, Ivan Sutherland, Fred Parke, and Barry Wessler for ideas, cooperation, and encouragement.

REFERENCES

1. Warnock, J., "A Hidden Surface Algorithm for Computer Generated Halftone Pictures," Technical Report 4-15, Computer Science, University of Utah, Salt Lake City, Utah, June 1969.
2. Bouknight, W. J., "An Improved Procedure for Generation of Half-tone Computer Graphics Presentations," Report R-432, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois, September 1969.
3. Watkins, G. S., "A Real-time Visible Surface Algorithm," UTEC-CSc-70-101, Computer Science, University of Utah, Salt Lake City, Utah, June 1970.
4. Gouraud, H., "Computer Display of Curved Surfaces," UTEC-CSc-71-113, Computer Science, University of Utah, Salt Lake City, Utah, June 1971.
5. Citron, J. and Whitney, J. H., "CAMP-Computer Assisted Movie Production,"

Proceedings of FJCC, 1968, vol. 2.

6. Weiner, D. D. and Anderson, S. E.,
"A Computer Animation Movie Language for
Educational Motion Pictures," Proceedings
of FJCC, 1968, vol. 2.

7. Baecker, R. M., "Picture-Driven Ani-
mation," Proceedings of SJCC, 1969.

8. Gracer, F. and Blasgen, M.W., "Karma-
A System for Storyboard Animation," Com-
puter Graphics, Vol. 5, No. 1, p. 26,
1971.