

COMPUTER ANIMATION

by

Charles Csuri* The Ohio State University

INTRODUCTION

A comprehensive 3-D real-time computer animation system is based upon a broad range of research activities in the field of computer graphics. In many ways the requirements for such a system are more challenging and complex than for other graphics systems. This is particularly true if one builds a language and a system which is truly user oriented and which has viable production capabilities for researchers and film makers. Too often systems which are the result of a research experiment in hardware or software design do not go beyond a beautiful demonstration of potentialities. Such experimentation is essential to advance the state of knowledge but if computer animation is to become a new research and production instrument we must, in addition, provide more examples of useable systems.

Although each of the following topics are aspects of a real-time animation environment deserving a detailed explanation, they will be dealt with in a way to introduce to the reader basic requirements and problems. The topics are:

- I. STATE OF THE ART SYSTEMS, AND LANGUAGES
 - Used as references and as a basis of comparison.
- II. AN ANIMATION ENVIRONMENT
 - Several systems and languages are being implemented to run under RSX-11/D on our PDP-11/45 computer.
 - A. VISIBLE SURFACE SYSTEM
 - Allan Myers' algorithm
 - VILAN (VIsual LANguage)

- B. GRAPHICS SUPPORT SYSTEM
 - Manfred Knemeyer's system for handling hardware devices, data structures, management of transformations and time, and memory management for the graphics buffer.
- C. ANIMA
 - A new graphics programming language has been designed and is being implemented.
- D. DATA GENERATION SYSTEM
 - Some approaches to problems are briefly discussed.
- III. DISPLAY HARDWARE AND GRAPHICS ALGO-RITHMS
 - The problems presented by the order of transformations in an algorithm are briefly described.
- IV. HIGH PERFORMANCE GRAPHICS
 - some speculations
- I. STATE OF THE ART SYSTEMS

During the past ten years several state of the art graphics systems and algorithms have been used to produce computer animated films. Some of these involve 3-D graphics using high performance algorithms in software and/or hardware and have set a high standard for computer graphics. General Electric's multi-million dollar color display system built for NASA was the first major example(1). Concave 3-D objects with 250-300 edges were displayed in real-time. Rougelot(2) of General Electric, using some of the same concepts and technology, developed an "off line" system to produce color movies. Using some custom electronic circuits. his

* e.g., a hard-wired matrix multiplier handling coordinates of 6 edges at a time.

^{*} Professor of Art and Computer Science and Director, Computer Graphics Research Group

system could process a color scene of about 350 polygons (1400 edges) in one second. Greenberg(3), using the Rougelot system, produced a stunning animated film representing movement through the Cornell campus. The data for the Cornell campus had about 12,000 polygons (48,000-50,000 edges) and each movie frame required 20-30 seconds of calculation time.

The Rougelot system used the Schumacker(4) algorithm. This algorithm processes shaded and colored objects but its data representation requirements place a heavy burden upon the user. Essentially the user must organize the data based upon certain topological properties of the scene. This involves, among other things, faces whose fixed priorities can be computed relative to each other. The effect is to reduce certain computational problems for the algorithm. With a model of an entire university campus, this can be a horrendous task. The preparation of the data for the 12 buildings representing the Cornell campus took 12 students one semester. The algorithm's speed is dependent upon such data representation and thus is not so general as most later algorithms.

Another major system is the Watkins Box at Case Western Reserve University. The Watkins(5) solution to the visible surface problem for 3-D concave polyhedra was hard-wired by the Evans and Sutherland Corporation so that it can run in realtime. This system also includes hardware clipping, smooth shading, perspective calculation and light source calculation. The algorithm for clipping was developed by Sutherland and Sproull(6) while the one for smooth shading was developed by Gourand(7). The display system cost about \$500,000 (the computer is extra) and is capable of displaying shaded objects in real-time. A real-time image may contain up to 2000 edges at 512 x 512 resolution. Interesting results based upon the Watkins algorithm in software (including routines for clipping, perspective and smooth shading) are the University of Utah films on the "Face" and the "Hand". The "Face", for instance, involved about 750 edges. It is interesting to note that although many hours of computer time were required to make films several minutes in length traditional animation would have taken far longer.

Perhaps the most spectacular single computer picture to be produced is MAGI's (MathematicalApplications Group, Inc.) tree. A calculation at 500 x 500 resolution of a 3-D deciduous tree in color with hundreds of leaves took nearly three hours on an IBM 360/65(8). MAGI's(9) algorithm has been used to produce some of the best'computer animated films.

Staudhammer's(10) system at North Carolina State University approaches a real-time color video display for 3-D concave objects. Using another implementation of the Watkins algorithm, he and his associates have hard-wired the scan segment conversion section of the algorithm to increase its speed. Objects of 2000 edges complexity typically take about 20-25 seconds with simple objects taking eight seconds. Staudhammer's system does not normally provide for clipping, smooth shading or perspective, but these features can be introduced at some cost in performance. As is the case with all known systems, the speed is also dependent upon the type of object selected. Significantly lower in cost than other systems, his approach has potential for widespread computer animation.

There have been several attempts at animation/graphics programming languages with a concern for naturalness. An experimental effort of the Computer Graphics Research Group at O.S.U. produced the Graphics Symbiosis System (DeFanti)(11), and its derivative the Animation-Real-Time (ART) system. Both of these, providing an easy to use language syntax, are for users who have little or no programming experience. One can quickly learn how to build pictures, use picture manipulation commands (move, scale, rotate, warp, group, hide, etc.) and to write programs in the language. These systems can be easily used to make film and video-tape. After accomplishing these initial goals, it was noted that it is sometimes difficult for users to produce animation sequences to their liking. Analysis indicated that the easy to use nature of the systems enables users to go from naive to sophisticated so quickly that they overreach the systems capabilities. Thus, the current challenge is to retain some aspects of naturalness and real-time response and to build a language and system with powerful algorithmic features.

II. AN ANIMATION ENVIRONMENT

The Computer Graphics Research Group is in the process of constructing major portions of an animation environment integrated under the RSX-11/D multitasking operating system on a PDP-11/45 minicomputer. Portions selected to be implemented are either essential or are currently suitable for research. It seems clear that it is not yet time for a complete production system although we will ultimately use the environment described here as a production system. This procedure which was also used with our prior systems (IBM 1130 programs, GRASS and ART)(11) is guaranteed to show what significant research problems remain.

A. VISIBLE SURFACE SYSTEM

The essential basis of the best known real-time graphics systems is a significant algorithm which handles the visible surface problem. The requirements of complex algorithms such as hidden line removal and visible surface calculation are so great that language designers tend to suggest that such algorithms must be hard-wired. Since it is so difficult to obtain a sufficiently efficient implementation, they sometimes claim that the super-fast computers have not arrived to handle such problems in software. For example, in the particular case of the visible surface problem, one expert has stated "hidden surface algorithms take too long to compute to be useful in a real-time system"(12). What many people believe can only be accomplished in hardware, has now been achieved on a minicomputer in software.

Allan Myers, of O.S.U.'s Computer Graphics Research Group has designed and implemented a major new visible surface algorithm for concave polyhedra. It is based on some of the latest programming techniques including structured programming* and it solves (actually avoids) the sorting problem as expressed by Ivan Sutherland(13), et al. This algorithm functions by efficiently accomplishing the work to be done as opposed to using time consuming strategies to avoid this work, as is the case with previous algorithms. Thus, it represents the first of a new class of algorithms.

Myers' technique does not place a heavy burden upon the user for data representation as is the case with the Schumacker and Newell algorithms. Unlike any other known algorithm its computation time grows as a function no worse than linear as the number of faces increases.

Implemented** on a PDP-11/45 minicomputer under the RSX-11/D multitasking operating system in a 32K (16 bit word) main memory partition, it can manage over 24 frames per second for small simple concave objects at 512 x 512 resolution. This includes clipping, perspective and smooth shading. The routine can manage over ten frames per second for multiple simple

- * Although most of the ideas behind structured programming are not new, they deserve the emphasis they have recently received.
- ** The basic visible surface capability has been operational since the end of 1974 and the entire program with clipping, smooth shading, transparency, etc., has been operational since March 1, 1975. Consequently, the frame rates given here are actual timed frame rates.

objects with independent transformations. The routine is not assisted by any special purpose hardware and uses only the point plotting, auto-increment, Z depth cuing and subroutine stack capabilities of the Vector General 3-D display. The routine is capable of displaying an image of up to ten thousand edges* from the 32K partition which contains the algorithm, data and free space**. Although complicated and/or large objects cannot be displayed in realtime, an image of five to ten thousand edges can be displayed in as little as several seconds. Further, scenes of reasonable complexity can be displayed in real-time at 100 x 100 resolution. At this resolution the user can make basic judgements about objects in terms of their position, intensity, speed of motion, clipping, etc. After achieving the desired visual effects, he can film the sequence at a higher resolution. Note that this is practical using Myers' aolorithm because the computation time decreases as a function greater than linear as the resolution decreases.

The routine was designed with the scientific experimenter in mind and includes the following capabilities:

- A. Scene Structure
 - 1. Multiple objects per scene.
 - 2. Multiple surfaces per object.
 - 3. Multiple faces per surface.

B. Image Quality***

- Clipping on a scene (a new, efficient method).
- 2. Perspective on a scene.
- Smooth shading of surfaces (if desired).
- Variable transparency of surfaces (not in real-time).
- Three independent gray scales (each of 2048 levels) for color during filming (the average is displayed for direct viewing).
- Face intensities based on the angle between the light source and the face (if desired).
- C. Manipulation Capabilities
 - 1. Scaling of objects.
 - 2. Rotation of objects.
- * Typically, the maximum possible number of edges is several thousand less than this.
- ** Although this may seem impossible after some simple calculations, it has been done.
- *** Note that individual parts of an object (i.e., surfaces) may be independently smooth shaded, faceted or transparent.

- 3. Translation of objects.
- 4. User specified face intensities (if desired).

Although interesting in itself, the algorithm with all of its capabilities does not satisfy our primary research goal. This is to make powerful algorithmic graphics capabilities available to the scientific research in a natural and flexible manner. Consequently, the kernel of a graphics language to be called VILAN (pronounced 'vī-lan) will be designed and implemented to allow a researcher the use of a sophisticated "solid" object manipulation and display capability.

B. GRAPHICS SUPPORT SYSTEM

We have a sophisticated graphics support system consisting of a privileged task and several device handlers under RSX-11/D. The graphics subsystem consists of four major components. These are, (i) the display driver and interrupt service routine, (ii) a refresh buffer, (iii) a communication area, and (iv) a subroutine library. If required, independent access is available to the user to each of these components. The general user, however, interfaces to the graphics subsystem through the driver and treats the display as an I/O device.

This system greatly facilitates hardware control, data structure manipulation, memory management for the graphics buffer, and the management of transformations and time. It should be noted that the graphics support system in and of itself provides more graphics capabilities than most graphics languages.

C. ANIMA

We have completed the detailed design specifications for a new graphics programming language called ANIMA and we are in the process of implementing it. The language has been designed for the scientific researcher who can use real-time animation and film as a research tool. Our conception of a graphics programming language is very inclusive with a full range of facilities, support algorithms, graphics algorithms, a compiler, powerful numeric capabilities, a generalized list structure for data, and other features not usually found in graphics languages. We plan to use 32K* (16 bit words) of main memory for the language and support algorithms and we already use 32K for picture buffer space.

While it is beyond the scope of this paper to describe details, a few important features will be identified. We have devised a systematic method for the specification of the common transformations which are essential for basic object manipulation capabilities. Our format for handling a general class of transformations is different from the typical method which is to develop a package of transformation matrix generating subroutines. Within our language transformations may be specified by expression definitions which are re-evaluated for every regeneration cycle. The implications of our method are important relative to reduced overhead costs in response time, the options and the facilities provided for the user, and the terms in which the user develops a solution to his problem. It is extremely difficult to appreciate the difference this makes with respect to the user actually being able to accomplish what he wants to do with a reasonable amount of effort.

The language is designed so that within a single "motion command" capabilities exist which include the following: limits, and mode. Limits specify the limits of the motion; mode specifies the action to be taken upon reaching the limits. Mode includes such things as repeat action, reverse action and halt. For instance, a requirement in a real-time animation environment might involve a 3-D shaded butterfly flapping its wings as it moves in 3 dimensional space. The flapping might be accomplished by means of a rotate command per wing with appropriate limits and with an appropriate mode specification to indicate whether the movement should restart, reverse or stop when the limit is reached.

In a real-time environment there are many instances where one must re-evaluate the parameters that modify transformations. Using analog input devices as well as variables and invoking programs to make the necessary changes in the motion sequence, the problem is a complex one. Everything cannot be re-evaluated in realtime, but we have devised a method of notation for the specification of continuous motion. For example, ".ROTATE NOSE,5, @(7*A)" where the picture "NOSE" will be rotated a constant $\mathbf{5}^{\degree}$ about the X axis and a variable amount about the Y axis specified by "7*A" which is re-evaluated for every display regeneration using the current value of A. Note that including a local clock in the re-evaluated expression produces continuous motion. If the above specifications were ".ROTATE NOSE, 5, @(7*.F)" then the picture "NOSE" will be rotated about the Y axis by an angle depending upon the current "frame clock" .F. This expression is re-evaluated on every regeneration cycle.

TIME MANAGEMENT AND TRACKING FACILITY IN ANIMA

The scientific researcher in a real-time

^{*} Our PDP-11/45 has 32K of MOS memory and 64K of core memory.

animation system requires accurate and repeatable images on the graphics display. It is necessary for the user to perceive the dynamics of a model correctly if it is to be a research tool. There are many algorithms that exceed the capacity of the computer to calculate and display pictures at a real-time rate. Consequently, time management will often be a significant problem: it will often be time to display the next "frame" well before the required computations for the frame are complete.

Traditional methodology for time management involves system clock support. The usual implementation of such support involves timer queues and communication between the clock support component and the task scheduler. A significant shortcoming of most implementations of system clock support relative to real-time animation is that there is no time-mapping-there is no way to correlate "action time" with film frame number. There is also significant overhead involved with system clock support. We developed a technique for such support which is to be controlled by our graphics language; and this support is embedded in the graphics system itself.

A tracking facility, which depends very heavily on time management, has the following capabilities: (1) the ability to dynamically record in real-time sufficient information to replay the totality of the visual effects produced originally; (2) the ability to dynamically modify in real-time and under user control the playback of a previous track thus producing a new track, for example, adjust the rotation of one particular object over one particular span of time; (3) explicit commands designed to perform often-desired effects, essentially a superset of editing and similar operations performed on film; for example, inserting fades, combining ("splicing") tracks, modifying time relationships, etc.; and (4) the capability of putting a track on film with the visual effects essentially unmodified.

D. DATA GENERATION SYSTEM

One of the major problems remaining in computer animation is data generation or data input of three dimensional objects. One input technique uses a physical model, for instance, a plaster cast of a head, covered with lines establishing convex polygons and then set upon a digitizing table(14). The user applies an electronic probe to the vertices of the polygons, in an order suitable for a shade algorithm, and the three dimensional coordinate data is automatically recorded on cards or Sutherland(15) describes an input tape. technique using two electronic pens se lecting coordinate data from a draftsman's representation of several views of a three dimensional object. The multiple pens

enable the user to indicate a single point simultaneously in two such views, thus defining the three dimensional position of a point. Provided that one has the blueprints, this is very useful for a representation of a model to a shade algorithm. However, in the typical case with animators the object is usually just something in their imagination. At best there may be a few rough sketches of the model on paper with many details missing.

There have also been research efforts to use a laser beam scan of a physical model for the automatic recording of data. This technique has the problem of excessive data, and more importantly getting the system to recognize sub-pictures or boundaries, for instance, in a human head--eyes, ears, or a nostril. Techniques of artificial intelligence and pattern recognition are also being applied to the data generation/input problem. It's an intriguing approach, but the results at best are tentative and there are horrendous problems to be solved before such techniques become practical. One suspects that some aspects of the data generation problem are likely to be solved much sooner using more straightforward methods.

Other methods employ mathematical techniques to generate basic forms such as polyhedra or quadric surfaces and planes. These forms are combined to construct a model of a car, tree, tank, building, etc. Although these methods have produced beautiful representations of various objects, they are not general enough to meet the needs of different animators.

We need methods that are somewhat analogous to the sculptor working in clay to create three dimensional models. One should have the freedom to quickly describe arbitrary surfaces, to smooth one surface into another one in order to establish continuity, to add to or subtract from small sections of surfaces, to stretch, twist or even squeeze a form with kinesthetic feedback--essentiallyto shape it into the final form. Then with simple commands one should be able to specify the color, the texture and the degree of transparency of surfaces.

The analogy of the sculptor's approach to data generation is an artist's fantasy in working with computers and one which usually brings gales of laughter from software designers and systems programmers (not ours). Obviously, when the user's data generation problem involves an interactive process and there is no formal model, then the software requirements become very complex. The communication problem between man and machine in an interactive graphics environment is still a challenging one. Much of this interaction depends upon the kinds of definitions that can be permitted by the software. One must be concerned about

approaches to data structures, the type of interactive language to be used, the relationship between the data structures and the language and the mathematical and algorithmic techniques to be used.

Data generation will be another system, running under RSX-11/D and our graphics support system, which will exploit a three dimensional sonic pen with a special interface. Dr. Leslie Miller of our group has developed some algorithms to easily create surfaces. These are techniques to build three dimensional curvilinear forms and to add surfaces together and through interpolation techniques provide for smooth continuity. When the user is finished his forms are automaticallyput into a format for visible surface calculation. Although Miller's approach has promise for a reasonably general class of forms, considerable work remains before it might be considered a system. Eventually, when files are created with the data generation system, they will be transferable to either the ANIMA system or Myers' shade system (VILAN).

III. DISPLAY HARDWARE DESIGN AND ALGO-RITHMS

As one considers how to integrate graphics algorithms into a language, and how to order transformations to optimize speed in relationship to a graphics display, one clearly recognizes that display designers did not have animation in mind when they built current hardware. Although film is made with these systems, the dollar market is with customers who need assistance for drafting problems, circuit diagram design and other engineering applications. These are usually applications where there is a requirement for displaying large amounts of data with "occasional" small changes to this data. Typically, the display designer's approach to transformations has serious drawbacks for real-time animation. Even the most general state of the art hardware implementation (microprogrammedor digital transformations and read-back) is not general enough to permit certain algorithms to efficiently use the capabilities represented by the hardware.

Myers' visible surface routine is a specific example of an algorithm requiring calculations in an order different from that assumed by the display designer. The algorithm needs to do the transformations of the objects before most of the processing. In the case of a microprogrammed display, there are two ways one might attempt to use the capabilities it represents. The first is to use it to do the transformations. This has the difficulty that transferring the relevant information back and forth between the general purpose processor and the display consumes a significant proportion of the time that was to be saved. The second way is to use it to perform the final phase of the algorithm. There are two difficulties with this second alternative: the display was optimized for transformations and is now being used for something else, and it may well have insufficient memory or no memory at all to perform the algorithm that was intended.

In the case of hardware with digital transformations and read-back, there is one of the same problems as with the microprogrammed hardware. This problem is, once again, that transferring the relevant information back and forth between the general purpose processor and the display consumes a significant proportion of the time that was to be saved. Note that for almost all practical purposes, the display hardware cannot be used to effectively implement any algorithm except those for which it was originally designed.

IV. HIGH PERFORMANCE ANIMATION/GRAPHICS

Advances in computer technology are bringing more powerful minicomputers with greater memory capacity, relatively lowcost dynamically-alterable-horizontallymicroprogrammed processors and displays, and microcomputers. In view of these developments it is important to re-examine the basic "black-box" premise under which high performance graphics has been operat-It should be possible to increase ing. the generality and flexibility of graphics beyond the black-box approach. The blackbox approach to graphics, epitomized by Sutherland, et. al., has long been the "right" way to do graphics. In the early stages of computer graphics it was the only way to achieve the performance necessary for real-time transformations such as clipping and perspective.

Now there are minicomputers available which, in a relatively straightforward multiprocessor configuration, are capable of real-time graphic transformations for a moderate cost. For example, the following configuration (see illustration, next page) would, with current technology, provide performance of the same order of magnitude as special purpose hardware. Although the performance on tasks that are traditionally hard-wired (e.g., rotation, clipping) would be less, the performance on tasks that are not hard-wired would be considerably greater than the traditional single processor system with special purpose hardware. Note that in the black-box approach the special hardware is of no use except in the special cases, whereas a general purpose system (microprogrammedor otherwise) lacks only a program to be of use in any task.

In the not too distant future it should be possible to equal or exceed the current

MYERS' FANTASY MACHINE



Note: There is no special purpose transformation hardware.

capabilities of the hard-wired approach even in the special cases. For example, the Watkins shading hardware at Case Western Reserve University is capable of calculating in real-time the visible surfaces described by 2000 edges. The upper limit of the hardware with severe flicker is 4000 edges. Since this is a hardwired implementation of a particular algorithm, and since the computation increases at a rate greater than linear, there is a fairly restrictive upper limit on the number of edges this hardware can handle in a reasonable amount of time. If, however, the algorithm were not hardwired, then there would be the possibility of increasing the buffer size and replacing the algorithm by a better one.

One can anticipate the "graphics machine" in the preceeding diagram being expanded in parallel. By extending the same organization over and over it would be possible to allocate <u>one graphics machine</u> <u>per picture</u>, thus creating "intelligent pictures". Although such a configuration is not currently very economical, the advent of low cost microcomputers and memory is rapidly making such approaches possible.

ACKNOWLEDGEMENTS

The author is indebted to Allan Myers, Manfred Knemeyer, Sam Cardman and Tony Lucido for their assistance on this paper, particularly their comments on the current status of ANIMA and their ideas on the state of current and future technology. The work of the Computer Graphics Research Group mentioned in the paper is supported in part by the National Science Foundation grant number DCR74-00768 A01.

Allan Myers' Visible Surface Algorithm.

The following pictures were produced using Myers' algorithm. There are several variables which determine the calculation time of a particular picture. The most significant of these are the number of edges, the orientation of the object(s) and the area of the faces (roughly measured by the size of the objects). The table gives the average time per frame for a number of objects.

The average is calculated as

$$\frac{\sum_{i=1}^{256} Z_i + \sum_{i=1}^{256} Y_i}{512}$$

where: Z_1 is the time for one "transform, calculate and display" step of the 256 steps taken for one revolution about the Z axis. Y_1 is the time for one "transform, calculate and display" step of the 256 steps taken for one revolution about the Y axis. These times are for 512 x 512 resolution. The designation large means an object almost the size of the area displayed and small means 1/4 (in height) of large.

Timings (averages for 512 different orientations at 512 x resolution.)

Objects	Number of Edges	Size	Frames/ Second	Seconds/ Frame
Violin	330	large	2	0.5
		small	4	0.2
Mask	672	large	1	0.9
		small	3	0.3
Smooth Apple	1446	large	0.7	1
		small	1	0.7
"Mathematical" Surface	4158	large	0.3	3
		small	0.8	1



Smooth Apple



Virtual intersection of two violins.



Mask



"Mathematical" Surface



Virtual intersection of an apple, mask and violin.

REFERENCES

- Rougelot, Rodney S.; Schumacker, Robert--"G.E. Real-Time Display", NASA Contract NAS 9-3916, Defense Electronics Division, General Electric Company, Syracuse, New York.
- Wild, C.; Rougelot, R.; and Schumacker, R. A., "Computing Full Color Perspective Images", General Electric Technical Information Series R71ELS-26, (May, 1971).
- Greenberg, Donald, "Computer Graphics in Architecture", Scientific American, pp. 98-106, April, 1974.
- Sutherland, Ivan E.; Sproull, Robert F.; Schumacker, Robert A., "A Characterization of Ten-Surface Algorithms", Computing Surveys, Vol. 6, No. 7, March, 1974, pp. 1-55.
- Watkins, Gary S., "A Real-Time Visible Surface Algorithm", Computer Science, University of Utah, Technical Report UTEC-CSC-70-101, July, 1970.
- Sproull, Robert F., and Sutherland, Ivan E., "A Clipping Divider", Proceedings of the Fall Joint Computer Conference, pp. 757-764, Thompson Publishing Company, 1968.
- Gourand, Henri, "Computer Display and Curved Surfaces", University of Utah, UTEC-CSC-71-113, June, 1971 and IEEE, TC-20, p. 623.
- Brooks, Joan, et. al., "An Extension of the Combinatorial Geometry Technique for Modeling Vegetation and Terrain Features", MAGI, Inc., NTIS report no. AD-782883, June, 1974.
- Goldstein, Robert A., "A System For Computer Animation of 3-D Objects", Proceedings of the Tenth Annual UAIDE Meeting, 1971, pp. 3-128-3-139.
- 10. Eastman, Jeffrey F.; Staudhammer, John; "Computer Display of Colored

Three-Dimensional Objects", The 2nd Annual Symposium on Computer Architecture (ACM-SIGARCH)Vol. 3, No. 4, December, 1974, pp. 23-27.

11. National Science Foundation report 1972, "Real-Time Film Animation", see section I, "Graphics Symbiosis System", by T. DeFanti and The Computer Graphics Research Group (contact C. Csuri for copies).

> Csuri, Charles, "Real-Time Film Animation", Proceedings of the Ninth Annual UAIDE Meeting, 1970.

"Real-Time Computer Animation", Proceedings of the IFIP Congress 74, Stockholm, Sweden, pp. 707-711. North-Holland Publishing Company.

Gillenson, Mark, "The Interactive Generation of Facial Images on a CRT Using a Heuristic Strategy", Ph.D. Dissertation in the Department of Computer and Information Science, The Ohio State University, March, 1974.

- 12. Clark, James H., "3-D Design of Free-Form B-Spline Surfaces", University of Utah, Ph.D. dissertation and NTIS report no. AD/A-002 736, September, 1974.
- 13. Sutherland, Ivan E.; Sproull, Robert F.; Schumacker, Robert A., "Sorting and the Hidden-Surface Problem", Proceedings of AFIPS 1973 National Computer Conference, Vol. 42, pp. 685-693.
- 14. Sutherland, Ivan E.; Sproull, Robert F.; Schumacker, Robert A., "A Characterization of Ten-Surface Algorithms", Computing Surveys, Vol. 6, No. 7, March, 1974, pp. 1-55.
- 15. Sutherland, Ivan E., "Three-Dimensional Data Input by Tablet", Proceedings of the IEEE, Vol. 62, No. 4, pp. 453-462, April, 1974.