ANIMENGINE : AN ENGINEERING ANIMATION SYSTEM

Tsukasa Noma and Tosiyasu L. Kunii
Department of Information Science,
Faculty of Science, The University of Tokyo,

ABSTRACT

Animation for engineering is very different from traditional animation for entertainment. We do not require realistic images and we cannot afford the cost, or the time, for their production. At the same time, there are other requirements which must be met. Firstly, it must be possible, in an engineering animation, to identify, unambiguously, each separate, mechanical part in a scene. Secondly, the animation must be produced quickly: ideally, in real-time.

As an attempt to satisfy these requirements, a new engineering animation system called ANIMENGINE has been implemented. To achieve clear visual identification of objects, we mark each of them with a unique, characteristic color or texture. For fast production and almost real-time display, we use a combination of on-line video display and recording. Algorithms have been developed for hidden surface removal with moving objects and a special, geometric model, the "template model" has been used to facilitate the production of scenes using many similar, common parts.

KEYWORDS: engineering animation, animation system design, interactive video, simulation, hidden surface removal

INTRODUCTION

Animation is one of the best ways to visualize the movement of an object. It suits not only entertainment but also flight simulation [1] or crash simulation [2]. Moreover, in the area of computer integrated manufacturing(CIM), an animation is very useful for engineering design, especially, as a means of communicating the results of assembly simulation.

ANIMENGINE is an animation system for engineering, and it aims to help designers to find and solve, the problems encountered in the design process with a superior man/machine interface, that is, an engineering animation. In contrast with costly and realistic image-oriented entertainment animation, an engineering animation system needs to meet the following requirements:

1. Exact and Unambiguous Display of Objects:

   Objects in engineering animation are parts, modules, and units of production and assembly machines such as robots. It must be possible to distinguish the different objects in a picture unambiguously. So the style of display must make this possible. This is more important than making the pictures realistic. Also, the parts must look solid. Line drawings do not convey the impression of solid working parts. For an engineer to gain an understanding of the three dimensional motion of the animation, it is much easier if the pictures have a solid appearance.

2. High-speed and automatic production of engineering animations:

   An engineering animation is also a way of communicating between designers, engineers, and producers. It needs to be produced as fast as possible so as to encourage their communication.

   In addition, engineering animations are usually produced by designers who are not professional animators. For this reason, an engineering animation system should work automatically.

3. Lower Host Dependency and Cost Saving:

   An engineering animation system must be constantly available during the design process. There is little point in having a fast system if you have to wait to use it. Typically, several work stations will share a central host computer and these must not put too large a load on the host. The work stations should be inexpensive too.

THE DESIGN OF ANIMENGINE

Based on the three requirements of the previous section, we made the following design decisions for ANIMENGINE.

1. Since a vector display works faster than a raster display, it is better for animation. In spite of this, we selected a raster display because we needed to display solid areas of color and patterns to make the components of our pictures identifiable, requirement 1 (above).

2. High-speed production of animation needs a high-speed display. A real-time animation system is ideal, but hardware for a real-time raster graphics animation system is too expensive, requirement 3. Instead, we decided to use an interactive video display combined with a video recorder. The designer can see his animation a frame at a time and record each frame for later display as a smooth animation. This system configuration produces frames fairly quickly, several frames per minute, and the designer can get a good impression of what the animation will look like as it is produced. We call this "pseudo real-time animation". We prefer video tapes to film because we can see the animation play back immediately and we do not need the very high definition of film.

3. In order to achieve the lower host dependency, we need to utilize an intelligent graphic device. If the device has a segment buffer and enables us to shift, rotate, and scale the figures up and down with simple commands, we can construct an engineering animation system which is practicable even if the connection with the host and the graphic device is only a low-speed serial interface, for example: a local area network.

For the internal data of the intelligent graphic device, three dimensional data are preferable, but two-dimensional, intelligent graphic devices are useful, too. This comes from the fact that much of the motion we wish to display is parallel motion combined with simple rotation and this can be done with a two-dimensional display.

If the movements of the displayed objects are parallel and the projection is a parallel projection or oblique projection, then we can treat shifting (translation) of an object in the three-dimensional object space as shifting in the two-dimensional projected plane. The case of parallel projection is shown in Fig. 1. The projection of an object M which is shifted by (x,y,z) in the object space is the same as the projection of M shifted $xP_x+yP_y+zP_z$ in the projected plane, where P_x, P_y, P_z are projections of unit vectors in the object space. Accordingly, we decided to employ a two-dimensional graphic device and to adopt parallel projection, especially, axonometric projection. This means that for parallel motion we need only transmit a projection of an object's shape to the work station once, and the animation can be produced locally. Of course, in the case of rotating objects we have to transmit a different projection for each frame. As a matter of fact, most of the rotations can be treated easily by "template models" described later.

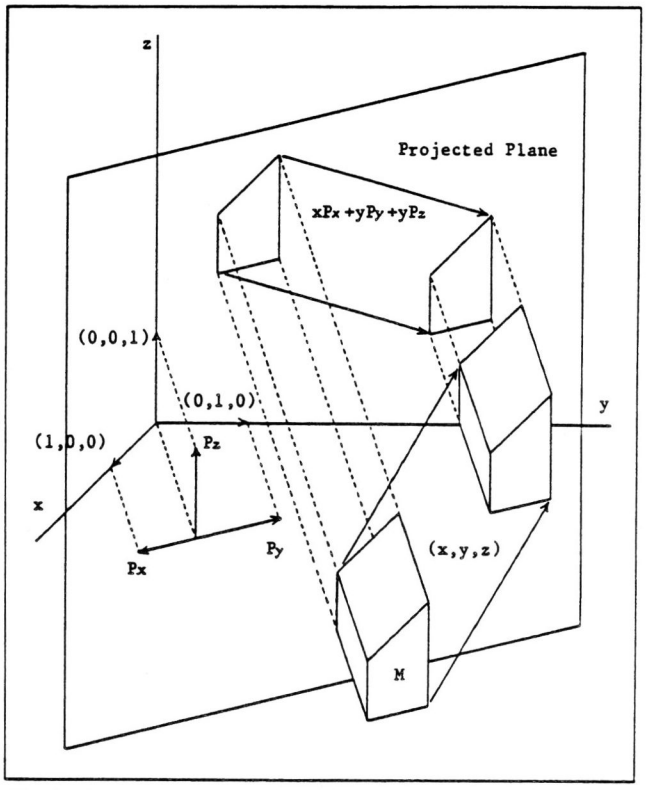

Fig.1. Parallel motion in the object space and parallel motion in the projected plane

The hardware of ANIMENGINE is shown in Fig. 2. The host computer is VAX 750 running UNIX. The graphic device is YAMAHA GC-100 with 752x480 pixels. It can put out RGB signals whose timing matches NTSC(RS-170A) signals, and with a PHOTRON ENCODER/DECODER ED-1000, we can get NTSC video signals directly from the GC-100.

We are using a Japan Victor U-matic video tape recorder. Ideally, the video tape recorder should be controlled by the host. For the moment, we are using a MSX personal computer with a Japan Victor time lapse unit as a video device controller and recording each picture at regular intervals frame by frame.

The software of ANIMENGINE consists of three units, called Global Controller, Display

Manager, and Recording Manager (Fig. 3). Global Controller manages the whole system and it has interfaces with other systems and users. Display Manager is computer graphics display software and it has interfaces with Object Database, Movement Database, and Template Database. Object Database contains the shape data of displayed objects and Movement Database contains data which describes their positions and motion. In a working application these databases would also be shared by other systems, for example, a computer aided design system and a strength simulation system. There is another database called Template Database, described later.

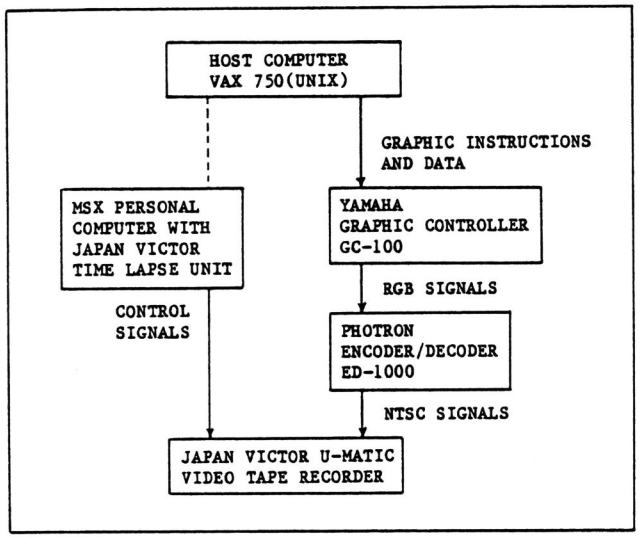

Fig.2. ANIMENGINE hardware configuration

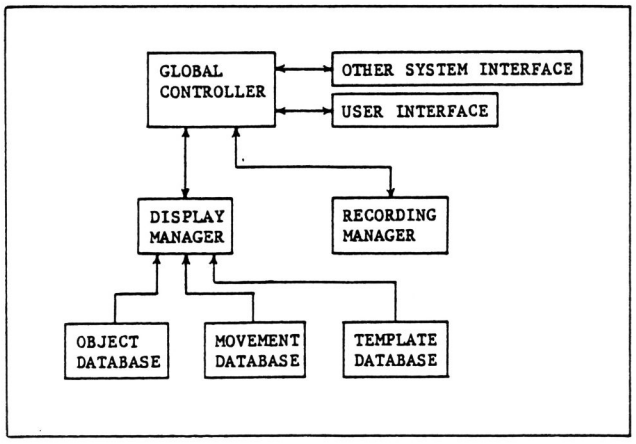

Fig.3. ANIMENGINE system architecture

Recording Manager controls recording and reports at the end of each frame, but because of the restrictions of existing hardware, described above, Recording Manager is only a time keeper.

COLORS, TEXTURES, AND SYMBOLS

As described in the first section, an engineering animation must have a solid appearance in order to help designers to recognize the shape, position, and direction of displayed objects. Methods of giving computer generated pictures a solid appearance are, for example, ray-tracing, shading, making distant objects darker and/or more obscure [2, 3], and the use of texture.

Now, in our case, Ray-tracing is unsuitable because of the amount of computation required. Since distant objects are often important in an engineering animation, we cannot make distant objects darker and/or more obscure. ANIMENGINE uses shading and texture to produce a solid appearance.

For shading, we use a diffuse reflection model with parallel illumination [4] owing to its simplicity. Consequently we need not change the colors of objects while they are moving in parallel in the object space. Colors of each object are not the same as those in the real world but decided from a characteristic color unique to each object and a shading effect. This makes identification of each object easier. Characteristic textures are also used for the same purpose.

ANIMENGINE can display symbolic marks and/or additional lines. Symbolic marks are used to express the positions of small holes or unseen points. Additional lines can represent the directions of movements or the center lines of shafts.

ANIMENGINE does not support anti-aliasing. This is because pictures of NTSC signals are less clear than RGB signals and anti-aliasing is unnecessary.

DRAWING EDGES

In engineering animation, it is important to recognize the edges of displayed objects. Edges of shaded objects are identified by sharp changes in color. It sometimes the case that two surfaces of different normal vectors are painted almost the same color. Considering this and similar cases, the designers wish to draw the edges themselves.

Drawing edges has three advantages:

1. Edges of visible surfaces make the shape of an object clear.

2. Edges of hidden surfaces enable us to recognize the whole shape.

3. If an object is represented as a set of polygons, some of the edges form an outline of the projection of the object. Also in this case, drawing edges can be useful in the identification of objects.

For this reason, ANIMENGINE has a function for drawing edges. It is interesting to note that when we use ordinary boundary representation, for example, winged edge data structure [5], it is difficult to distinguish boundary edges worthy of being displayed from useless ones.

A solid with a hole is shown in Fig. 4. If the holed face ABCD-EFGH is represented as two surfaces, that is, the hexagons ABCGFE and AEHGCD, then drawing all the boundary edges results in drawing two line segments AE and CG. They are not only useless but harmful.

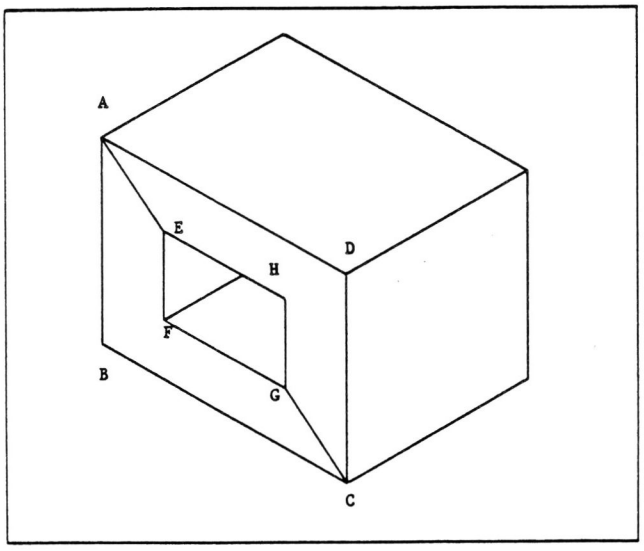

Fig.4. A solid with a hole

Another example is given in Fig. 5. It is clear that a number of rectangles represent the curved surface. We don't have to draw edges between such rectangles. (When wire frame models are drawn, such edges are useful. In our case, however, it is important to recognize "sharp" edges.)

We thought that the visibility of the boundary edge should depend on the angle between two surfaces which share the edge. So we adopt the following algorithm.

```
ALGORITHM EDGE_VISIBILITY_DECISION(E)
begin
  (* Symbols PFACE and NFACE are used as defined
  by Baumgart [5] *)
  vp <- normal vector of PFACE(E);
  vn <- normal vector of NFACE(E);
  vp <- vp/|vp|;
  vn <- vn/|vn|;
  if(vp.vn<1-ε) then
    set E visible;
    else
    set E invisible;
end
```

This can decide the visibility of each edge. This is not enough. We need another algorithm to avoid losing profile lines.

```
ALGORITHM MAKE_PROFILE_LINE_VISIBLE
(* E1,...,En are all the invisible edges. *)
begin
  for i=1 to n
  begin
    vp <- normal vector of PFACE(Ei);
    vn <- normal vector of NFACE(Ei);
    vv <- normal vector of projected plane;
    if((vp.vv)*(vn.vv)<=0) then
      set E visible;
  end
end
```

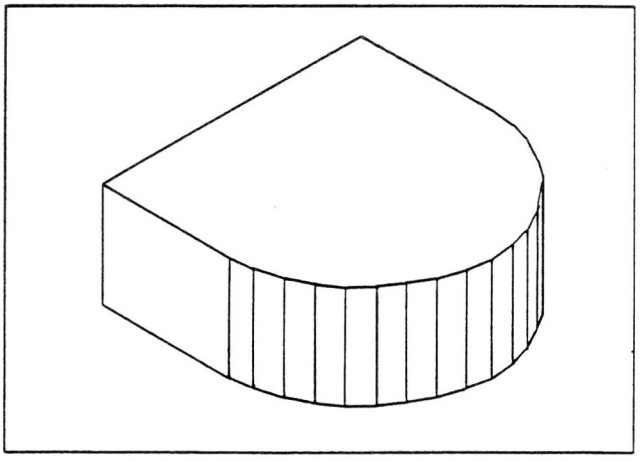

Fig.5. A solid with an approximated curved surface

HIDDEN SURFACE PROBLEM IN A DYNAMIC ENVIRONMENT

Hidden surface removal [6] is usually done with a static environment and a static view-point. If we apply it to an animation, it takes a lot of time because it is necessary to repeat the calculation for each picture.

A method of decreasing the amount of computation for each picture with appropriate preprocessing has been reported [7]. It is only for a static environment and a dynamic view-point for such applications as flight simulation. Conversely we want to remove hidden surfaces in a dynamic environment with a static view-point. Therefore we thought that global hidden surface removal should be done. "Global" means that hidden surface removal of several frames is done at the same time. Our algorithm is based on the painter's algorithm [4], to make the most of the intelligent functions of graphic devices. We paint each scene from back to front so that hidden surface removal becomes a matter of deciding in which order to draw each face. Our algorithm deals with objects only in parallel and uniform motion.

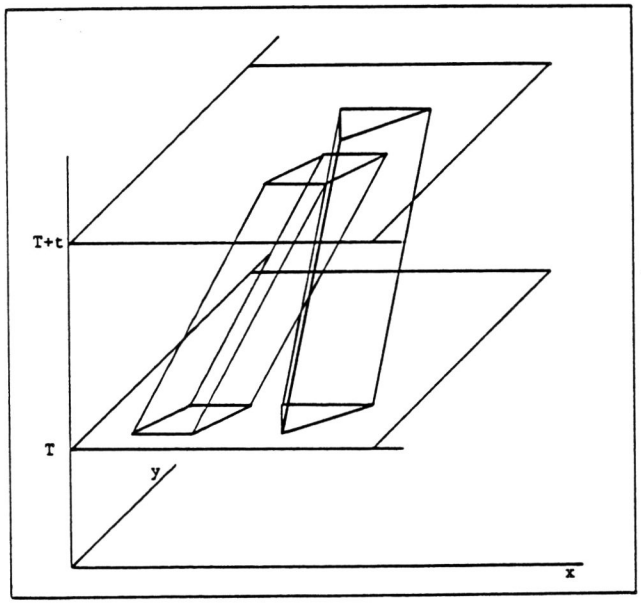

Fig.6. Interference detection over time t

The most important point of this algorithm is to determine whether or not the projections of two faces intersect each other in the projected plane within a given period. This is equivalent to interference detection between two oblique prisms in the three-dimensional space composed of two axes in the projected plane and a time axis. In Fig. 6, two oblique prisms represent the motion of the projections of two faces over a time t. If the two oblique prisms intersect each other then the projections of the two faces intersect each other within the time t. If we compute in the three-dimensional space, we can decide whether they intersect each other or not. This is the most general method but inefficient for an engineering animation.

Accordingly, we want to compute in the projected plane. One of the alternative ways is to examine the loci of the projections of the two faces (Fig. 7). If the loci do not intersect each other, the projections do not intersect within the given period. The converse is not true. The loci can intersect although the projections of the objects do not interfere. We can use this test only for preprocessing.

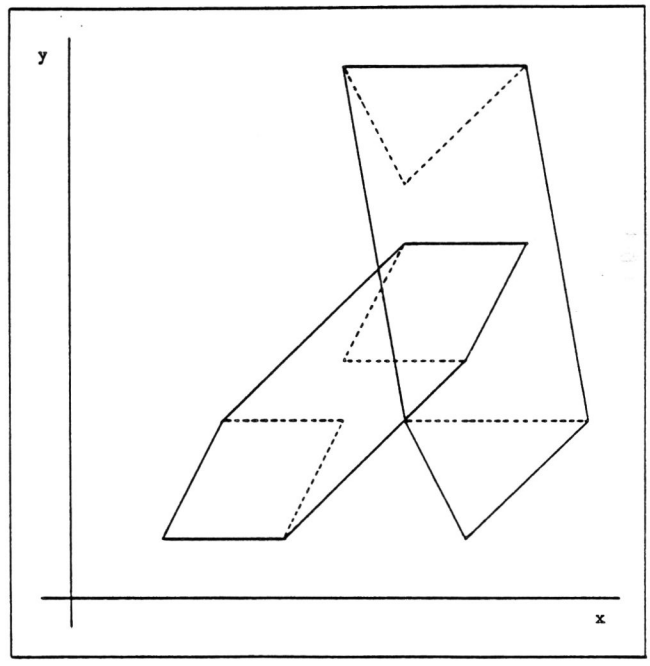

Fig.7. Interference detection by loci of two projections

Now, objects of engineering animation are mainly rigid ones. In parallel motion, faces of rigid objects do not change the shape of their projections. If we take a new origin of coordinates from a fixed point in one of the object's projections, the motion of the other object's projection in the new coordinates will be a difference of the original two motions. The projection of the first object intersects the locus of the projection of the second object in the new coordinates if and only if the original

two loci intersect (Fig. 8). The algorithm follows.

```
ALGORITHM
GLOBAL_DEPTH_BUFFER_OF_TWO_SURFACES(A,B)
begin
  ta <- end of the current parallel and uniform
  motion of A;
  tb <- end of the current parallel and uniform
  motion of B;
  tmin <- min(ta,tb);
  split A into triangles A1,A2,...,Ak;
  split B into triangles B1,B2,...,Bl;
  for i=1 to k
    for j=1 to l
     GLOBAL_DEPTH_BUFFER_OF_TWO_TRIANGLES(Ai ,Bj
     );
end

ALGORITHM
GLOBAL_DEPTH_BUFFER_OF_TWO_TRIANGLES(P,Q)
begin
  LQ <- locus of Q in the relative coordinate;
  split LQ into triangles LQ1,...,LQm;
  for i=1 to m
    if(P and LQi intersect) then
    begin
      set priority between P and Q until tmin;
      break;
    end
end
```

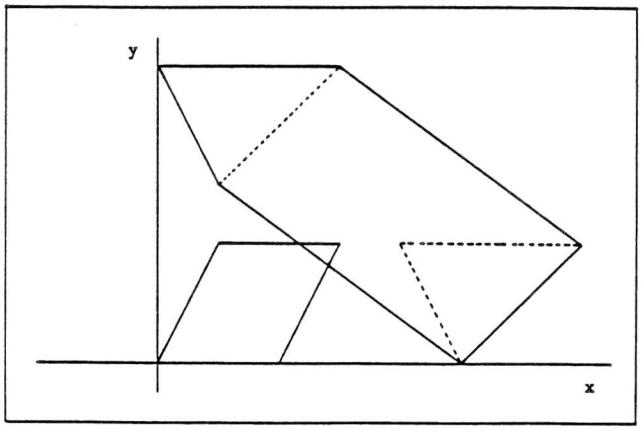

Fig.8. Interference detection by loci of two projections in the relative coordinates

This methodology can also be applied to the interference/collision detection of moving rigid objects in three-dimensional space [8, 9, 10].

TEMPLATE MODEL

ANIMENGINE adopts boundary representation as its figure model. It is a general and advantageous model for animation. It is not always appropriate to have boundary representation data for all objects. For example, shafts, rings, bolts, nuts, and screws appear quite frequently in an engineering animation. They have the same shape within a particular type of part and only the sizes differ from one to another.

Just as a template is used, when drafting, to draw figures which often appear but are hard to draw, so ANIMENGINE enables us to specify regular parts as "template models".

A template model consists of the the part name and a list of numbers to specify its size, and when the template model is drawn, drawing information in Template Database is used. Generally, drawing information in Template Database does not include solid models or full specifications of the shape in three-dimensional space. It only provides rules for drawing. Therefore invisible 'back' surfaces can be omitted.

Two examples of template rules for drawing a cylinder are shown in Fig. 9. Fig. 9 (a) is with shading effects on the curved surface and Fig. 9 (b) is without shading. Note that the back faces are not considered. Whether Fig. 9 (a) or (b) is adopted is determined by the entry in Template Database.

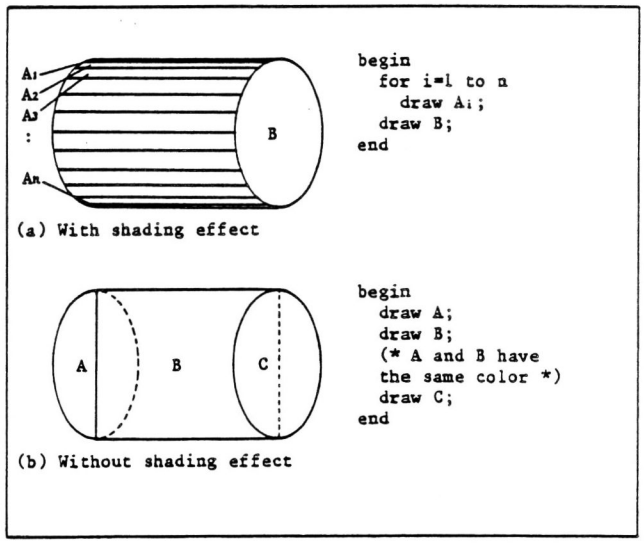

Fig.9. Template rules for drawing a cylinder

Template models have another advantage, for handling object rotation. For example, when a shaft, specified as a "template model", rotates around its center line, it need neither be redrawn nor does it require its shape data transferred from the host computer. This is be-

cause its rotation does not change the way it is drawn. Note that the way in which the template is designed is important here. Suppose a shaft is defined by an inappropriate boundary representation, say a polygonal approximation, the view of its boundary differs after rotation (Fig. 10). So it becomes necessary to redraw its shape repeatedly during its rotation. What is even worse, is that in this case, the shape data would be transmitted from the host, repeatedly.

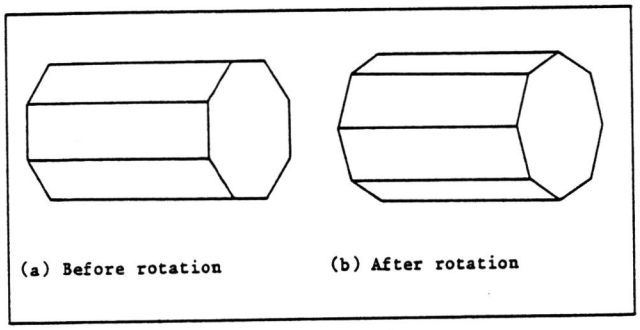

Fig.10. Rotation of a shaft specified by the polygonal boundary representation

EXAMPLES

Fig. 11 is a simulation of assembling a copy machine. Each module is identified with its characteristic colors and a robot hand and its arm, gripping a drum module are identified with a texture. Two sets of three red triangles are symbolic marks. The right-hand set represents the positions of screws on a drum module and the left-hand set is the corresponding positions on a base module. Edges of hidden surfaces help to represent the whole shape of each module.

Fig. 12 is a simulation of setting rings onto a shaft. The shaft and the rings are defined as a template model. Additional lines indicate the center line of the shaft.

CONCLUSION AND FUTURE WORK

ANIMENGINE satisfies our requirements for engineering animations, that is, exact and unambiguous display of objects, high-speed and automatic production of engineering animation, low host dependency and modest cost.

The presentation of colored and shaded areas needed for easy identification of picture components is a requirement which conflicts with the need for fast display. We have used a novel method for hidden surface removal which enables us to produce displays of sufficient quality, fast enough to be of practical use.

There is room for improvement in the following points.

1. Our method for hidden surface removal cannot deal with cyclic overlap now. This we hope to correct soon.

2. Engineering animation needs exact display. When the displayed objects are too complicated, it should be possible for the system to simplify them automatically when the detail is not required.

3. There are some characteristics of engineering which make the job of animation simpler than in a more general context. We have tried to take advantage of this by means of our "Template model". There may be other such characteristics which can be exploited.

4. We are now utilizing a frame-by-frame video tape recorder, but a video tape recorder needs several seconds for prerolling. In order to relieve this bottle-neck, for example, we could use a video disk recorder.

ANIMENGINE is a growing system and the developed system will be described in a succeeding paper.

ACKNOWLEDGEMENTS

This work was partially supported by Nippon Gakki Co., Ltd., Ricoh Co., Ltd., and Japan Victor Co., Ltd. Prof. Geoff Wyvill's critical comments and review were useful to brush up the paper.

UNIX is a trademark of AT&T.

MSX is a trademark of MICROSOFT.

REFERENCES

1. B.J. Schachter, "Computer Image Generation for Flight Simulation," IEEE CG&A, Vol.1, No.4, Oct. 1981, pp.29-68.

2. N.I. Badler, J. O'Rourke, and H. Toltzis, "A Spherical Representation of a Human Body for Visualizing Movement," Proc. IEEE, Vol. 67, No. 10, Oct. 1979, pp. 1397-1403.

3. K. Yokokawa, and T.L. Kunii, "A Definition of Neighborhood of a Region for Picture Processing," Computer Graphics and Image Processing, Vol. 14, No. 2, Oct. 1980, pp. 112-144.

4. S. Harrington, Computer Graphics --- A Programming Approach, McGraw-Hill, 1983.

5. B.G. Baumgart, "A Polyhedron Representation for Computer Vision," AFIPS Conf. Proc., Vol. 44, May 1975, pp. 589-596.

6. I.E. Sutherland, R.F. Sproull, and R.A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," Comput. Surveys, Vol. 6, No. 1, March 1974, pp. 1-55.

7. H. Fuchs, Z.M. Kedem, and B.F. Naylor, "On Visible Surface Generation by a Priori Tree Structures," Computer Graphics, Vol. 14, No. 3, July 1980, pp. 124-133.

8. P.G. Comba, "A Procedure for Detecting Intersections of Three-Dimensional Objects," JACM, Vol. 15, No. 3, July 1968, pp. 354-366.

9. J.W. Boyse, "Interference Detection Among Solids and Surfaces," CACM, Vol. 22, No. 1, Jan. 1979, pp. 3-9.

10. D.M. Esterling, and J. Van Rosendale, "An Intersection Algorithm for Moving Parts," Computer-Aided Geometry Modeling (NASA Conf. Publication 2272), April 1983, pp. 129-133.

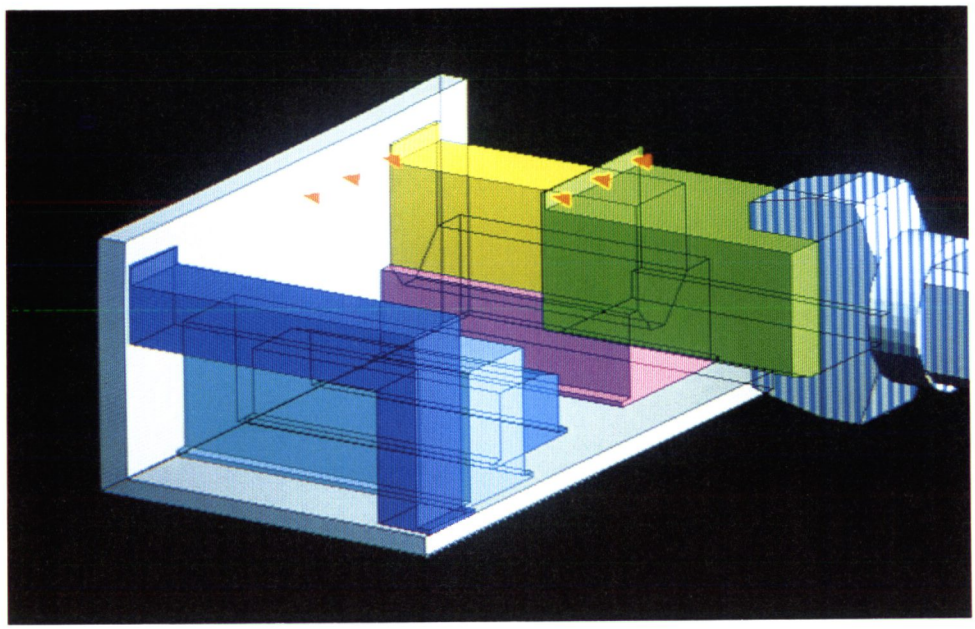

Fig.11 A simulation of assembling a copy machine

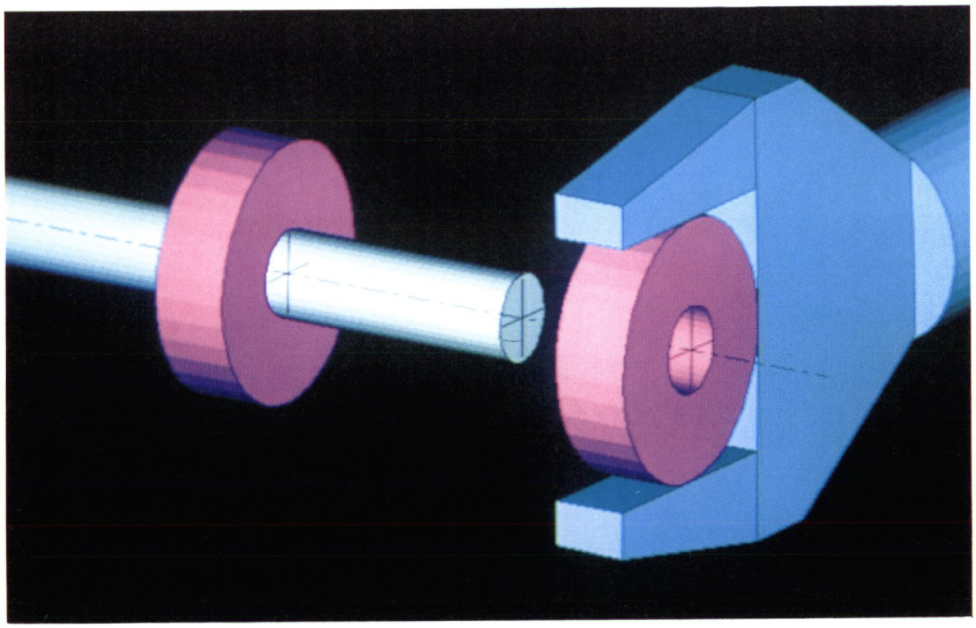

Fig.12 A simulation of setting rings onto a shaft