FILM ANIMATION BY COMPUTER

by Dr E. E. Zajac
Bell Telephone Laboratories, Murray Hill, New Jersey

By feeding a cathode ray tube with data from a computer it becomes very easy to make animated motion pictures illustrating a mathematically complex sequence of events. The technique has great potential, in enabling research workers to visualize the results of computation and in preparing educational films

Suppose you are teaching a course in celestial mechanics. You want to show the satellite orbits that would result if Newton's universal law of gravitation were other than an inverse square law. On a piece of paper you write:

```
DELT = 1.0
TFIN = 1000
EXP  = −3
CALL ORBIT (DELT, TFIN, EXP).
```

Then you take the paper to the computation centre. After a few hours, you return to pick up a movie, which you then show to the class.

In the movie the positions of the satellite and parent bodies are drawn on a frame of film every minute of their orbits (DELT = 1.0) for the first 1000 minutes (TFIN = 1000). The satellite and parent body, obeying an inverse cube gravitational law (EXP = −3), start out in circular orbits. However, as you have shown in class mathematically, circular orbits for an inverse third power gravitational law are unstable. The 1000-frame (42-second) movie drives home the point; the students see the initial, almost perfectly circular orbits disintegrate, with the satellite making ever-closer swings about the parent body until the two bodies collide (Figure 1).

To those who have tried to make an animated scientific film by conventional means, this account may sound Utopian. Hand animation for a movie of this sort first requires the computation of the coordinates of the satellite for each frame of film. Then comes the tedious task of drawing and properly positioning the satellite 1000 times on celluloid "cels". Finally, the cels have to be individually photographed.

Yet, at the Bell Telephone Laboratories, I can today write the programme I have shown and obtain 16 mm film, ready for viewing, within three or four hours. At present, there is only a handful of installations in the world where this procedure is possible. But within a few years, I suspect it will be available at most major universities and industrial laboratories in the United States and Western Europe.

The cost of making the film, including computer time and processing, but not including my labour, would be about $30. However, this cost and the above description are deceiving, for they do not take into account the work of my colleague, F. W. Sinden, who constructed the "subprogramme" called ORBIT which computes the orbit and produces the film. To have a more balanced view, we have to consider how computer pictures are drawn and how subprogrammes such as ORBIT work.

The computer-driven cathode ray tube— Figures 2 and 3 are themselves computer drawn, to illustrate the picture-making process. The instructions or programme for the picture are fed into the computer, usually on punched cards (Figure 2). The computer then calculates numbers that specify the picture and the commands for film advances. These are read on to magnetic tape, shown in the bottom right of Figure 2. Next, the tape is rewound; it is then connected to a cathode ray tube and the film-advance mechanism of a camera (upper right, Figure 3). As the tape is run from its beginning, the numbers on the tape are translated into electron beam commands for the cathode ray tube; the picture appears on the face of the tube and is recorded on the camera film (the shutter is always open). When a frame is completed, a command from the tape to the camera causes the film to advance to the next frame. All this happens at electronic speeds, so that a film such as the celestial mechanics film described earlier might be made at the rate of 5 to 10 frames per second, or about one quarter of the rate of standard movie projection.

Programming of computer animation— The cathode ray tube performs only two basic tasks: (1) it "types" a small font of standard-sized letters and (2) it sweeps a straight line segment between two specified points. (The description here pertains specifically to the SC-4020, a computer-driven cathode ray tube manufactured by the General Dynamics Corporation. However, other computer-driven cathode ray tubes work in a similar manner.)

Although these tasks are performed with high precision and reliability, they are in principle extremely simple. The advantages of computer animation do not lie with an elaborately versatile cathode ray tube. Rather, they lie with the power of a high-speed digital computer. Indeed, the special virtue of computer animation is that it brings the power of computing to the film medium.

To illustrate this power, let us consider how one builds a programme for an animated film, starting with the line-drawing ability of the cathode ray tube. An example of the basic line drawing command is:

```
CALL LINE (−5, 3, 15, 22)
```

This command causes the electron beam to connect the point with horizontal coordinate X = −5 and vertical coordinate Y = 3 to the point with coordinates X = 15, Y = 22. The plotting area of the cathode ray tube is a square with X and Y ranging between −512 and + 512, giving more than one million reference points.

The above instruction is written in a standard scientific programming language called Fortran, although Fortran users will note that, in the interest of clarity, I have taken certain liberties with standard Fortran. A Fortran programme is actually executed in two passes through the computer. On the first pass, the symbolic Fortran instructions are translated into the basic numerical code of the computer; the output is a set of punched cards or a magnetic tape. In the second pass, this numerical code is read from the cards or tape and executed; the result is the final numerical output of the computer.

Fortran was written to allow symbol manipulation, akin to ordinary algebra (Fortran stands for "FORmula TRANslator"). For example, the two commands

```
Y = 5
CALL LINE (−15, Y, 30, Y)
```

will cause a horizontal line to be drawn from the point X = − 15, Y = 5 to X = 30, Y = 5. Symbolic specification is convenient in the writing of "loops", for example

```
DO THRU*, I = 1, 100
Y = I
*CALL LINE (−15, Y, 30, Y)
```

The first instruction signals the formation of the loop. It commands the computer to execute 100 times the instructions following, up to and including the instruction marked by the asterisk. Each time through the loop, the Y coordinate of the line is given the value of the counter, I. So on the 29th pass through the loop Y = 29, on the 30th pass Y = 30, and so on. In this example, 100 horizontal lines, spaced vertically one unit apart and lying

between Y = 1 and Y = 100 would be drawn on a single frame of film.

Computer loops are tailor-made for animation. To illustrate, first introduce the instruction CALL FRAME. Fortran translates this into a command to the camera to advance the film by one frame. Insertion of CALL FRAME into the loop gives:

```
DO THRU* I = 1, 100
Y = I
CALL LINE (– 15, Y, 30, Y)
*CALL FRAME
```

The result is animation. We get 100 frames of a movie showing the steady upward motion of a single horizontal line.

Even this simple four-instruction programme illustrates several points. First of all, note that a computer loop is not the same as a film loop. A film loop repeats the same sequence of *events* over and over. A computer loop repeats the same sequence of *commands* over and over. A command, such as Y = I, may call for a change from the previous passage through the loop. Thus, the event produced in each passage through the loop generally *differs in a systematic way* from the event produced in the previous passage.

Secondly, note the ease with which we can change the programme to produce a new animated sequence. Suppose we want the action to go only one-half as fast. We simply replace the command Y = I by Y = 1/2, and the command DO THRU* I = 1,100 by DO THRU* I = 1,200. Or, we might decide that the line should extend from X = – 5 to X = 50. This requires only that we change the LINE call to CALL LINE (– 5, Y, 50, Y).

In fact we could allow for all of these changes at once by using symbols. The addition of an instruction, READ, allows us to specify particular values at the time of running. The complete programme then looks like this:

```
READ, SPEED, X1, X2
DO THRU* I = 1, 100 × SPEED
Y = I/SPEED
CALL LINE (X1, Y, X2, Y)
*CALL FRAME
STOP
3, 10, 72
```

The READ instruction says to the computer, "When the card containing STOP has been reached, all instructions in this programme will have been loaded in. Look for numbers on the next card. Set SPEED equal to the first number, XI to the second number, and X2 to the third number. Then execute the programme."

For each particular set of values of SPEED, X1, X2, we get a different film. This illustrates another point of great importance; a successful programme generates more than a single film; it makes possible a whole *family* of films. The family is a function of several variables just as in mathematics a family of curves can be a function of several variables. For each particular choice of values, a new film results. Different choices may give films that do not even resemble each other. Thus, in the celestial mechanics example, a single programme using the Newtonian law of gravitational force yields movies of circular, elliptical, parabolic and hyperbolic orbits, depending on the initial conditions.

Figure 1 *Two massive bodies interact according to an inverse cube law of force. The bodies either spiral in and collide, as shown, or spiral ever outward. From* Force, Mass and Motion.

Modern computing allows still another freedom. We can make our programme into a subprogramme. First, we decide on a name. In Fortran, we are limited to names of six symbols. We pick HRZLIN (standing for horizontal line). Then we modify the previous programme as follows:

```
SUBROUTINE HRZLIN (SPEED,
  X1, X2)
DO THRU * I = 1,100 × SPEED
Y = I/SPEED
CALL LINE (X1, Y, Xw2, Y)
*CALL FRAME
```

This programme is stored in the computer or on cards once and for all. If in some film in the future, we decide we want to show an ascending horizontal line, we would write:

```
READ, SPEED, X1, X2
CALL HRZLIN (SPEED, X1, X2)
STOP
2,5,50
```

If we did not want to experiment with several different SPEEDs, X1s and X2s, but rather knew that these variables should have the values 2, 5, and 50, the programme would be shorter still:

```
CALL HRZLIN (2, 5, 50)
```

This illustrates the powerful "naming" or subroutine capability of programming. By this means we can build up a library of programmes as we go along. This library differs markedly from the usual film library. Each entry represents a family of films rather than a single film. Each entry probably embodies loops. But these are computer loops, not film loops.

The reader can now perhaps understand the celestial mechanics example given at the outset. Having F. W. Sinden's general programme, ORBIT, for producing a family of orbit films as a function of the relevant variables, it is simply a matter of writing the instruction CALL ORBIT and specifying values to obtain the particular orbit film desired.

Applications of computer animation— Computer results usually emerge in the form of numbers printed on paper, sometimes hundreds of thousands of numbers per sheet on hundreds of thousands of sheets of paper. With computer animation one can translate the numbers into a series of pictures which can be scanned in succession as a movie. Figure 4 is a frame from such a movie.

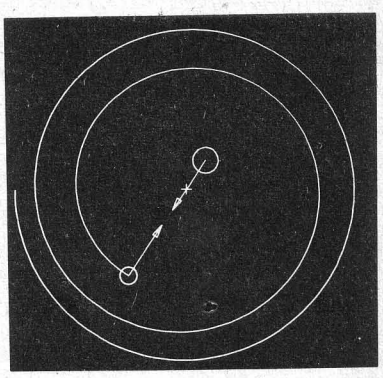


Figure 2 *A programme on punched cards for a picture is fed into the computer. The computer writes the numbers representing the computed picture on to magnetic tape. From* A Computer Technique for the Production of Animated Movies

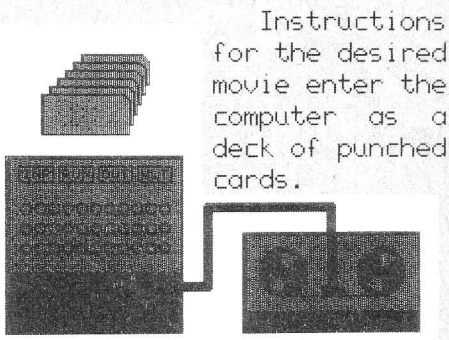


Figure 3 *The computed picture is read from magnetic tape to drive a cathode ray tube and camera film advance mechanism. From* A Computer Technique for the Production of Animated Movies

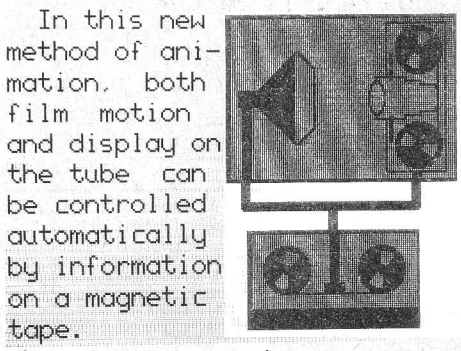
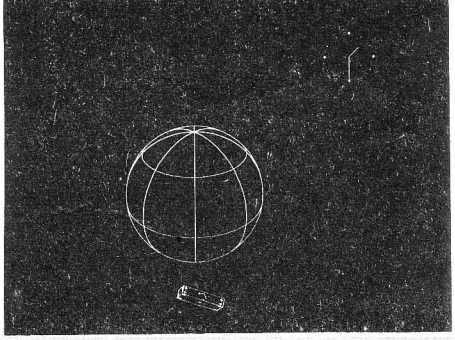


Figure 4 *Box representing a satellite changes attitude according to a mathematical model of the Earth's gravity. Clock counts off orbits. From* Simulation of a Two-Gyro, Gravity-Gradient Attitude Control System



FILM ANIMATION BY COMPUTER *continued*

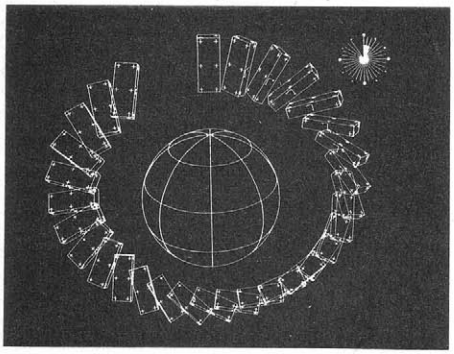


Figure 5 *Superposition of every fifth frame during one orbit from* Simulation of a Two-Gyro, Gravity-Gradient Attitude Control System

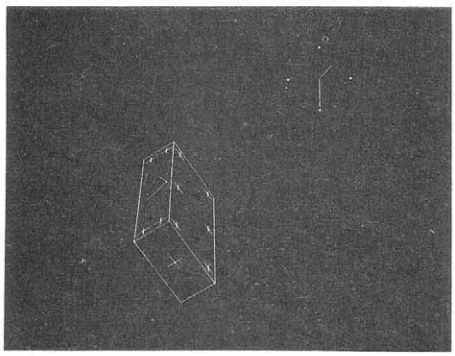


Figure 6 *Satellite as seen in an orbiting reference frame. From* Simulation of a Two-Gyro, Gravity-Gradient Attitude Control System.

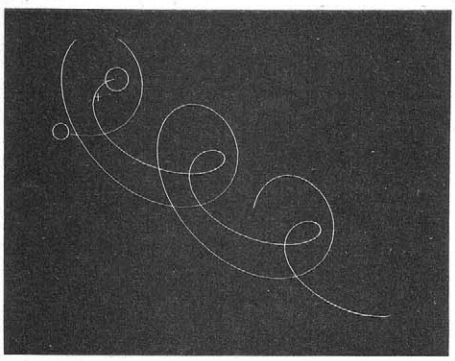


Figure 7 *Orbits of bodies with inverse square law interaction. The plus sign is the moving centre of mass. From* Force, Mass and Motion

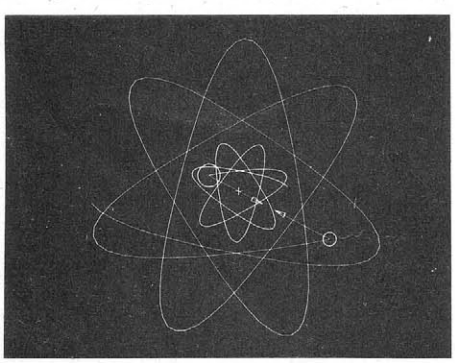


Figure 8 *Orbits of two bodies with a direct cube force law of interaction. From* Force, Mass and Motion

In this case, I was studying, by digital computer simulation, the orientation control of a satellite. At first, I had the computer print out numbers giving the satellite orientation at successive instants of time. The problem of visualizing the satellite motions from the printed numbers was formidable. So I wrote a subprogramme which took the numbers that would normally be printed out and used them to compute a perspective drawing of a box representing a satellite. Figure 5 shows the superposition of every fifth frame of the movie for the first satellite orbit (in the movie itself the Earth turns). An orbital clock in the upper right-hand corner counts off orbits.

In the programme it is easy to change the point of view of the perspective. In Figure 6, also taken from the movie, the viewer is travelling in orbit with the satellite.

Most of the applications of computer animation so far have been of this sort—usually short sequences visually displaying the results of a scientific computation. Examples are:

A movie of successive iterates of an iteration procedure for solving an optimization problem (Bell Telephone Laboratories; Sandia Corporation)

Flow of a viscous fluid, including the formation of a von Karman vortex street (Los Alamos)

Propagation of shock waves in a solid (Los Alamos; Lawrence Radiation Laboratory, Livermore, California)

Lines of constant pressure, temperature, precipitation, etc. in a dynamic meteorological model of the Earth's weather (Lawrence Radiation Laboratory)

Vibration of an aircraft (Boeing Aircraft)

Simulation of an aircraft carrier landing (Boeing Aircraft)

Most of the animation has been in black and white; however, the movies produced at the Lawrence Radiation Laboratory have been in colour.

The possibilities of the use of computer animation in this way as an output means for standard computing are limitless. Indeed, one can argue that graphical output is "natural," and the usual numerical computer output is woefully inefficient. The human eye has great pattern recognition ability; for this very reason the usual first step in handling scientific data is to plot it. The human eye also quickly picks out a moving object from a static background. Movies allow one to take advantage of this ability by adding the dimension of time to the familiar spatial dimensions in studying computer output.

Moreover, many of the calculations done on a computer are of essentially dynamic phenomena. One wants to see the unfolding or evolution of a process either as a function of time or of some other variable. Motion pictures are the obvious way of accomplishing this.

Finally, as the sample frames of Figures 4-6 show, it is easy to make perspective views of solid objects. It is likewise easy to make two perspectives side by side, that is, to make stereographic movies. This line of research has been pursued by A. M. Noll of the Bell Telephone Laboratories. Among other things, he has made a movie of a four-dimensional cube rotating about one of its axes, as seen when projected into three dimensions.

Educational use of computer animation—The use of computer animation for the display of the results of scientific computations is a form of education—one scientist passing information to another. It suggests the use of computer animation for the classroom.

One immediately thinks of a whole host of phenomena and concepts that uniquely lend themselves to illustration by computer movies. The idea of a "limit" in the calculus, for example, is essentially a dynamic one, as the standard terminology suggests—"f(X) *approaches* $f(X_a)$ as $X\text{-}X_a$ *approaches* zero." Other examples are Newton's laws of motion, kinetic theory in physics and chemistry, fluid flow in engineering—in fact, one can argue that a good deal of the instruction in the physical sciences is in terms of "movies" of mathematical models of nature—except that the student does not normally see the movies; he only hears verbal descriptions of them.

Perhaps more important is the opportunity computer animation gives to the scientist and engineer to make his own movies. As our examples have shown, the user programmes computer animation in the language of mathematics, a language in which the physical scientist or engineer is proficient. Moreover, the scientist can make films with a minimum of dependence on directors, producers, and animators. Much as in writing a book, where one submits a manuscript and receives back a proof in print ready for reading, so can one now submit a programme and receive back a proof film, ready for viewing. The scientist can be master of his own house; if he wishes, he can maintain complete creative control over the films he makes.

One might expect the professional science film-maker to feel threatened by this development. Paradoxically, the opposite has been the case with the few filmmakers who have had a chance to try computer animation and to appreciate its flexibility and power. One professional animator told me that he has always been envious of composers. All a composer needed was a piano to try out his latest creation. The animator on the other hand

could only with great labour try out his ideas. For example, if the action were too fast, cels would have to be redrawn and the scene laboriously rephotographed. In computer animation, such things as speed can be left as a variable. The film-maker has much more freedom to experiment. He is free to bring his full artistic abilities to bear in his partnership with scientists in film making.

One of the few purely educational computer animated films is *Force, Mass and Motion,* by F. W. Sinden of the Bell Telephone Laboratories. The film illustrates Newton's laws in two dimensions. Orbits are shown of two massive bodies under central force action for various laws, such as inverse cube, direct cube, etc., as well as for the familiar inverse square law. Figure 1 and Figures 7 and 8 are frames from Sinden's movie.

Another educational movie by Roger N. Shepard and the author is entitled *A Pair of Paradoxes.* It combines two psychological phenomena discovered recently. One, due to L. S. and R. Penrose of the University of London, is a winding staircase that seems to go ever upward while at the same time closing upon itself after each circuit. The other, due to Shepard, is a tone that seems to go ever upward while at the same time remaining near the middle of the scale. Figure 9 is a frame from this movie.

The future—It is now several years since films were first animated by computer (some computer films are known to have been made on the Whirlwind computer at MIT in 1951). However, they have made relatively little impact, especially in science education where their prospects are perhaps brightest. Probably the reason is the lack of a suitable champion. As I have indicated above, most of the development has been in industrial laboratories, where the interest has been in straightforward technical applications as a computer output device. Very little work has been done at universities, and that which has been done has also been aimed at scientific rather than educational research.

To exploit the full educational potential of computer animation will require the partnership of three specialities: computing, film-making, and the subject-matter science or technology about which a film is to be made. A central facility where these skills could come together would give the educational uses of computers a great push forward. So far no such facility exists.

Another need is for higher-level computer languages. We have already seen an example of a computer language in Fortran, which allowed us to write formulas to programme in terms of symbols rather than numbers, and to name sequences of instructions as subprogrammes.

Recently, a great deal of activity has gone into improving on Fortran, especially in the mechanism for naming. In particular, computer languages in certain specialities have been written. These contain, built into the language, names for the concepts and operations peculiar to that speciality. Such a language for movie-making would be very useful. It would contain convenient ways of specifying and moving objects mathematically and ways of projecting the object into a picture plane; it would allow the programmer to imagine himself as a cameraman with commands for panning, zooming in or out, dissolving, fading, etc.

A big first step in the direction of a universal movie language has already been made. K. C. Knowlton of the Bell Telephone Laboratories has written a language for movie making called BEFLIX which allows one to do many of the things mentioned. Figures 2 and 3 illustrating computer animation are taken from a 17-minute movie made in BEFLIX entitled *A Computer Technique for the Production of Animated Movies.*

The instructions in BEFLIX look quite different from those in Fortran. An example is:

PAINT, A, B, WRITE, 2

which says: "Paint the rectangular area specified by the opposite diagonal points A and B. First erase what is now in this area and then write in 2s". By filling rectangular areas of the screen with various alphabetical and numerical characters, Knowlton generates the different textures of grey shown in Figures 2 and 3. Unlike the satellite and orbit examples of Figure 1 and Figures 4–6, each frame of which was a line drawing, a frame of a film in BEFLIX is completely filled with varying shades of grey. Thus, BEFLIX is more akin to the scanning technique of television. BEFLIX can be easily learned in a few weeks by persons with no knowledge of mathematics or computing.

Special-purpose computer languages are attempts to make easier the specification of image inputs to the computer. Another recent development in this direction is the light pen, the stylus-like device that allows one to draw computer-recognizable pictures on the face of a cathode ray tube. The combination of the light pen and special-purpose languages is perhaps the ultimate in a graphical man-machine communication system—one which is still in its infancy.

The future for computer animation is, in my opinion, very bright. It is, however, too early to tell exactly what its impact will be, especially in science education. Here, the computer does best the animation that is couched in mathematics—precisely the animation that is hardest to do by hand. So hard, in fact, that only a few examples of it have been tried in the classroom. We therefore have little experience with which to predict the future.

But all those who have tried computer animation so far are excited by its possibilities. I think their expectations will be more than fulfilled.

Author's note—Some of the films mentioned are available on loan from the Technical Information Library, Bell Telephone Laboratories, Murray Hill, New Jersey, USA. These are all 16 mm films:

Simulation of a Two-Gyro, Gravity-Gradient Attitude Control System (4 min, sound)

Force, Mass and Motion (10 min, sound)

A Pair of Paradoxes (2 min, sound)

A Computer Technique for the Production of Animated Movies (17 min, silent)

Figure 9 *Ball bouncing along an impossible staircase. From* A Pair of Paradoxes, *which combines the Penrose and Penrose visual illusion with the auditory illusion of R. N. Shepard*