3D COMPUTER ANIMATION: MORE AN EVOLUTION PROBLEM THAN A MOTION PROBLEM

N. Magnenat-Thalmann, D. Thalmann
MIRA Lab. HEC/IRO, University of Montreal, Montreal, Canada.

ABSTRACT

3D computer animation is too often restricted to very simple motions: e.g. logo rotations or camera motions around a 3D reconstructed object. Enormous effort is devoted to image rendering, but little to the motion itself. In fact, research in computer animation needs to look beyond the mere creation of motion towards the evolution of state variables over time according to appropriate laws. We call this "the evolution problem". In this paper, we show how new evolution laws may be introduced into an animation system. Several examples are discussed: collisions, brownian motions, chaotic attractors and models based on simultaneous differential equations. All examples are presented using the extensible director-oriented animation system MIRANIM.

RESUME

L'animation tridimensionnelle par ordinateur se limite trop souvent à des mouvements très simples comme des rotations de logos ou des mouvements de caméras autour d'objets construits par digitalisation. Un effort considérable est investi pour rendre les images réalistes, mais peu d'effort est consacré au mouvement proprement dit. En fait, la recherche en animation par ordinateur a besoin de dépasser le simple stade de création de mouvements pour mieux étudier l'évolution dans le temps des variables d'état; ce que nous appelons "le problème de l'évolution". Dans cet article, on montre comment de nouvelles lois d'évolution peuvent être introduites dans un système d'animation. Plusieurs exemples sont présentés: collisions, mouvements browniens, attracteurs chaotiques et modèles basés sur des systèmes d'équations différentielles. Tous les exemples sont formulés à l'aide du système d'animation extensible MIRANIM qui est dédié au réalisateur.

INTRODUCTION

Three-dimensional computer animation has become more and more popular. Every computer-generated film or sequence is considered as very exciting and innovative. In fact, computer animation is an innovative technique based on old-fashioned methods, especially kinematic laws. Most of the special effects used in advertisements are centred on combinations of translations, rotations and other matrix-based operations. Other effects, generally considered as spectacular, are based on the so-called interpolation technique. In fact, three-dimensional computer animation systems can be divided into two categories:

1. scripted systems

These systems are mainly based on the Mudur and Syngh [1] axiom, although some were designed before this theory was developed. They consist of the description of objects with a list of associated transformations. Animation is often limited to trivial transformations: i.e. rotations and translations of objects; however, some interesting effects may be generated using procedural models controlled by a few external parameters. Typical systems of this kind are ANIMA II [2], ASAS [3] , MIRA [4], CINEMIRA [5], and the SKELETON ANIMATION SYSTEM [6].

2. key-frame systems

Although mainly devoted to 2D computer-assisted animation, key-frame systems may also play an important role in 3D as discussed by Sturman [7]; typical 3D key-frame systems are BBOP [8] and MUTAN [9].

In our view, it is unfortunate that most computer animation is limited to these two main areas. The spectacular effects of computer-generated scenes is not due to their animation but to the image rendering phase. This is not surprising, because the image rendering phase has been the focus of more

than 10 years of research; every year, new image synthesis algorithms are presented and a high degree of realism can be achieved, although the methods used are still very expensive (e.g. ray-tracing techniques). However, one may ask whether the rotation of a virtual camera around a 3D reconstructed object with several light sources, texture and shadows is worthy of the term computer animation. Is computer animation merely able to show a metallic logo moving across the screen with two rotation movements? 3D computer animation is generally described [10] as composed of three phases:

- object modelling
- object and camera motion
- image rendering

The second step should be the most important. If only the first and third steps are considered, we should not call this computer animation. There is a great deal of literature on object modelling, and image synthesis is a major area of research. But papers on object and camera motion are very rare. Of course, papers describing computer animation systems and languages may be easely found [11]. However, most of them do not introduce any new theoretical ideas on motion. Many of them discuss shading and shadow problems. One significant paper which is for us the most important paper on computer animation in the last five years should be noted however. This paper is not very innovative in terms of the image synthesis technique which is quite simple, but it describes motion and transformations based on evolution laws. This is the Reeves [12] paper on particle systems. It should also be mentioned that several authors [13, 14] suggest taking forces into account in computer animation. This is certainly a step in the right direction, but the approach is still very classical and may be assimilated into a list of transformations applied to objects.

We believe there is a paradox here: computer animation is more popular than ever; but computer animation research is in a blind alley.

We believe that the main reason for the relative weakness of computer animation theory is that animation is defined as the essence of movement. Animation is therefore viewed as a problem of creating motion. This is not wrong in itself. However, motion implies more than a single list of image transformations or a concatenation of matrices. What is important in a rotation is not so much the rotation itself, but how the angle of rotation evolves over time. It is true that linear and cosinusoidal variations are often adequate . The fact that so many effects may be generated with these simple laws partly explains why innovations in animation are so rare.

HOW CAN NEW LAWS OF EVOLUTION BE INTRODUCED

We shall use some examples based on our extensible director-oriented system MIRANIM [15] to explain how new laws of evolution can be introduced into computer animation. The system is mainly based on three components:

1. The object modelling and image synthesis system body-building.
2. The director-oriented animation editor ANIMEDIT.
3. The actor-based sublanguage CINEMIRA-2.

ANIMEDIT, when used by non-programmer animators is basically a scripted system based on the Mudur and Syngh approach. The director designs a scene with actors, cameras and multiple lights. Each of these entities is driven by animated variables, which are in fact, state variables following evolution laws. These laws are predefined and may be chosen from among a linear law, cosinusoidal acceleration and deceleration, circular motion, gravity etc... Although we have introduced less common laws like fuzzy laws, this is still limitative for the creator. CINEMIRA-2 allows the director to use programmers to extend the system. The great advantage of this is that the system is extended in a user-friendly way. This means that the director may immediately use the new possibilities.

It is essential that any new evolution law be defined in such a way that it may be applied to any state variable as follows:

in CINEMIRA-2:

```
law INNOVATIVE(P1,P2:T):VECTOR;
begin
  INNOVATIVE:=f(P1,P2,CLOCK)
end;
```

in ANIMEDIT:

```
VECTOR VAR,A,0,0,0
LAW PROCEDURAL,INNOVATIVE
...
EVOLUTION VAR,INNOVATIVE,0,10
```

A STRATEGY FOR DESIGNING EVOLUTION LAWS

The problem in the design of evolution laws is to find a way of expressing them analytically:

e.g. INNOVATIVE:=f(P1,P2,CLOCK)

Our strategy in CINEMIRA-2 is based on the following principle: if laws may be expressed analytically, we program a single law. If the evolution law is only expressed as a function of a previous state, we merely store values as global variables and the evolution law then gives a value computed from the global variables. The more general case occurs when evolution laws are modified during the animation process. This is generally not possible in current animation systems. We implement this as follows:

1. We initialize the animation system in the main CINEMIRA-2 program.
2. We define procedural objects, to initialize global state variables.
3. We define animation blocks which modify these global state variables at each frame.
4. We define evolution laws which depend on the global state variables. They compute values that depend on the value of the state variables which are accessed by the blocks during the animation process.

With this strategy, an evolution law may be completely changed at any time (and consequently at any frame).

EXAMPLES

1. Collision of bodies

Assume we have two moving objects with masses M1 and M2. We may describe this by defining two actors BODY1 and BODY2. A single movement is assigned to each one by using:

```
ACTOR BODY1, OBJ1
ACTOR BODY2, OBJ2
MOVE BODY1, POS1
MOVE BODY2, POS2
```

A very simple way of defining POS1 and POS2 is to use a linear law. For example:

```
VECTOR POS1,A,-100,-100,0
LAW LIN,2
...
EVOLUTION POS1,LIN,0,10
```

We now take into account the law of conservation of momentum and compute the collision if this occurs. We do not change the transformations associated with the actors BODY1 and BODY2. But a COLLISION law must be programmed. In fact, this law is not easy to implement and it consists (at present) of 400 lines of CINEMIRA-2 source code. The law heading is:

```
law COLLISION(M1,M2:REAL;P1,P2,SP1,SP2:
    VECTOR;COLLITYPE:INTEGER):VECTOR;
begin
...
end;
```

This law returns the position of the second body at any time before or after a collision; M1 and M2 are the two masses, P1 and P2 are the two initial positions and SP1 and SP2 the two initial speeds; COLLITYPE allows the type of the collision to be selected as elastic or inelastic.

This law must then be applied to the animated variables driving the motion of both bodies:

```
LAW COLLI1,COLLISION
3,1,-100,-100,0,141.4,0,0,25,25,0,-35.3.0,0,1
LAW COLLI2,COLLISION
1,3,141.4,0,0,-100,-100,0,-35.3,0,0,25,25,0,1
EVOLUTION POS1, COLLI1,0,10
EVOLUTION POS2,COLLI2,0,10
```

where

M1=3, M2=1, P1=<-100,-100,0>

P2=<141.4,0,0> SP1=<25,25,0>

SP2=<-35.3,0,0> COLLITYPE=1 (Elastic)

This law functions well, as we have checked using the real-time play-back implemented in our animation system [15].

2. The Henon chaotic attractor

The Henon chaotic attractor [16] has recently been made more popular because of its relationship with fractal measures [17]. The Henon map is defined as:

$$x_{i+1} = 1 + y_i - ax_i^2$$

$$y_{i+1} = bx_i$$

x_o and y_o may be chosen independently. This means that the map is two dimensional. However, when the map is iterated, the sequence of iterations appears to lie on what looks like a very tangled curve. The attractor is apparently thicker than a curve, but not thick enough to be two dimensional. It is then a fractal as defined by Mandelbrot [18].

Such a chaotic attractor may be defined as a law in CINEMIRA-2, which could for example be applied to a particle system. The law is defined by the following program:

```
program CHAOTIC:
var VCURR:VECTOR;
      FLAG:BOOLEAN;
  law HENON(A,B:REAL; VO:VECTOR): VECTOR;
  var HEN:VECTOR;
  begin
     if FLAG then
     begin
        HEN:=<<PROJY(VCURR)+1
         -A*SQR(PROJX(VCURR)),
         B*PROJX(VCURR)>> ;
        VCURR:=HEN;
        HENON:=HEN
     end
     else
     begin
        HENON:=VO;
        VCURR:=VO;
        FLAG:=TRUE
     end
  end;
begin
   FLAG:=FALSE
end.
```

3. Simple brownian motion

A very simple model of brownian motion [19] may be described as follows:

When a small particle is immersed in a liquid, it can be seen under a microscope to move in a zig-zag motion. In our model we assume that a particle moves along three one-dimensional lines (along the axes X,Y and Z) where it may hop from one point to one of its two neighboring points with equal probability. To obtain a law which gives the position at any time of a particle with such a motion, we have to store a global state variable for each particle. These variables are modified by an animation block and the law then merely gives the current position by accessing the state variables. In CINEMIRA-2, this takes the following structure:

```
program BRMOTION;
var STATE:array[1..MAXPART] of
        record
           POSITION:VECTOR;
           ...
        end;
law BROWNIAN(PARTN:INTEGER):VECTOR;
begin
   BROWNIAN:=STATE[PARTN].POSITION
end;
block UPDATE;
begin
   (* updates the particle position *)
end;
begin
   (* initialize the particle position *)
end.
```

4. Evolution laws based on differential equations.

Many complex motions are exactly described by simultaneous differential equations. An analytical solution is generally impossible to obtain. Instead of approximating the motion by a series of rotations and translations, which may be very difficult to find, it is better to use the original equations. For example, consider the classical pilot-ejection problem [20]. We have to determine combinations of aircraft velocity and air density which will enable an ejected pilot to

miss his aircraft's vertical stabilizer; high aircraft velocity and low altitude will prevent successful ejection. The equations are:

$$dX/dt = V\cos(\text{theta}) - VA$$
$$dY/dt = V\sin(\text{theta})$$
$$dV/dt = 0 \quad \text{for } 0 \le Y < Y1$$
$$= -D/M - G\sin(\text{theta}) \quad \text{for } Y \ge Y1$$
$$d(\text{theta})/dt = 0 \quad \text{for } 0 \le Y \le Y1$$
$$= -(G\cos(\text{theta}))/V : \quad \text{for } Y \ge Y1$$
$$D = (RHO*CD*V)/2$$

where VA is the aircraft velocity
M is the mass of pilot plus seat
G is the gravitational constant
D is the drag force
CD is the drag constant
Y1 is the length of ejection rails
RHO is the air density
X,Y are the trajectory coordinates to be obtained and theta is the angle to the vertical.

To solve such a problem and obtain the solution in the form of analytical evolution laws, we have designed a general CINEMIRA-2 program which solves differential equations using three methods: Euler, modified Euler and Runge-Kutta (order 4). The program looks like this:

```
program CONTINUOUS;
const
  MAXSTATE=20;
  MAXPAR=20;
  MAXDERIV=20;
type
  STATE=array[1..MAXSTATE] of REAL;
  PARAM=array[1..MAXPAR] of REAL;
  DERIV=array[1..MAXDERIV] of REAL;
var
  Y:STATE; (*state variables*)
  P:PARAM; (*equation parameters*)
  G:DERIV; (*differential equations*)
procedure DIFFEQ;
  (*defines the differential equations
    under the form: G[I] :=f(Y[J],P[K])
    should be provided by the user *)
end;
procedure INITSIMULATION;
(* defines the values of the parameters P[K]
   and the initial values of the state
   variables Y[J], the integration method,
   the time scale *)
end;
procedure EULER;
end
procedure EULERMODIFIED;
end
procedure RUNGEKUTTA;
end;
block SIMULATION;
  (*responsible for the integration of the
    equations and for calculating the values
    of state variables at the current time*)
end;
law RESULT:VECTOR;
  ...
  (*several laws may be defined by the user
    in order to obtain the values of state
    variables*)
end;
begin
   INITSIMULATION
end.
```

When the user has defined the equations, the initial conditions and the laws required, the extended ANIMEDIT system is used as follows:

```
LAW NEW,RESULT
EVOLUTION VAR,NEW,0,10
BLOCK SIMULATION,0,10
```

OTHER EXPERIMENTS

We are also attempting to use this strategy to add particle systems to our MIRANIM system. Systems may be initialized by procedural objects and are then updated by animation blocks, and the laws may return any state variables. For example, we may obtain the position of the center of a particle system and consider it as the interest point for the camera. The only problem is that we cannot use the normal display algorithm. One way of solving the problem is to invoke a special display procedure in the animation block; however this solution is not very satisfactory, because it cannot be combined with other objects.

Other projects involve the modeling of an explosion; positions of any part of an object destroyed by an explosion will be accessed by an evolution law; this will allow, for example, the possibility of viewing an explosion from a part of the exploded object.

CONCLUSION

The possibility of defining any evolution law within an animation system may provide many new possibilities for motion, thus breaking with the traditional approach using only matrix operations. We are at an experimental stage with such capabilities. However, we believe that far more complex motions may be generated with this approach.

ACKNOWLEDGMENTS

The authors are grateful to Ann Laporte who has revised the English text. This work was sponsored by the Natural Sciences and Engineering Council of Canada.

REFERENCES

[1] Mudur, S.P. and Singh, J.H., "A Notation for Computer Animation", IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-8, No. 4, 1978, pp. 308-311.

[2] Hackathorn, R., "ANIMA II : a 3-D Color Animation System", Proc. SIGGRAPH '77, pp. 54-64.

[3] Reynolds, C.W., "Computer Animation with Scripts and Actors", Proc. SIGGRAPH'82, pp. 289-296.

[4] Magnenat-Thalmann, N. and Thalmann, D., "The Use of 3D High-level Graphical Types in the MIRA Animation System", IEEE Computer Graphics and Applications, Vol. 3, No. 9, pp. 9-16.

[5] Thalmann, D. and Magnenat-Thalmann, N., "Actor and Camera Data Types in Computer Animation", Proc. Graphics Interface '83, pp. 203-210.

[6] Zeltzer, D., "Representation of Complex Animated Figures", Proc. Graphics Interface '82, pp. 205-211.

[7] Sturman, D., "Interactive Keyframe Animation of 3-D Articulated Models", Proc. Graphics Interface '83, pp. 35-40.

[8] Stern, G., "Bbop- a System for 3D Keyframe Figure Animation", SIGGRAPH'83 Tutorial Notes on Computer Animation, Vol. 7, 1983, pp. 240-243.

[9] Fortin, D., Lamy, J.F. and Thalmann, D., "A Multiple Track Animator System for Motion Synchronization", Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop on Motion, Toronto, 1983, pp. 180-186.

[10] Magnenat-Thalmann, N. and Thalmann, D., Computer Animation, Theory and Practice, Springer-Verlag, Tokyo, 1985.

[11] Magnenat-Thalmann, N. and Thalmann, D., "An Indexed Bibliography on Computer Animation", (submitted for publication).

[12] Reeves, W., "Particle Systems : a Technique for Modelling a Class of Fuzzy Objects", ACM Transactions on Graphics, Vol. 2, 1983, pp. 91-108.

[13] Wilhelms, J. and Barsky, B.A., "Using Dynamic Analysis for the Animation of Articulated Bodies as Humans and Robots", Proc. Graphics Interface '85, (These proceedings).

[14] Armstrong, W.W. and Green, M., "The Dynamics of Articulated Rigid Bodies for Purposes of Animation", Proc. Graphics Interface '85, (These proceedings).

[15] Magnenat-Thalmann, N., Thalmann, D. and Fortin, M., "MIRANIM : An Extensible Director-oriented System for the Animation of Realistic Images", IEEE Computer Graphics and Applications, Vol. 4, No. 3, March 1985.

[16] Hénon, M., Comm. Math. Phys., Vol. 53, 1976.

[17] Doyne Farmer, J., "Dimension, Fractal Measures and Chaotic Dynamics", Evolution of Order and Chaos, Springer-Verlag, 1982, pp. 228-246.

[18] Mandelbrot, B., The Fractal Geometry of Nature, W.H. Freeman, 1982.

[19] Haken, H., Synergetics, Springer-Verlag, 1977, p. 69.

[20] Korn, G.A. and Wait, J.V., Digital Continuous System Simulation, Prentice-Hall, 1978, pp. 98-103.