A Survey of Computer Animation

Xavier Pueyo
Daniela Tost

DMI01-87/ 1987

• 140019145?
copia 1

UNIVERSITAT POLITECNICA DE CATALUNYA
ESCOLA TECNICA SUPERIOR D'ENGINYERS INDUSTRIALS DE BARCELONA
DEPARTAMENT DE METODES INFORMATICS

A SURVEY OF COMPUTER ANIMATION

XAVIER PUEYO SANDEZ
DANIELA TOST PARDELL

REPORT DE RECERCA DMI01-87

MAY 1987

INDEX

TABLES AND FIGURES 'S INDEX

ABSTRACT

A survey on computer animation is presented with three specific goals:

- To clarify what people understand by computer animation.
- To separate the different aspects of computer animation and underline its main problems.
- To present the different solutions given to these problems.

Several classifications for computer animation systems are first given according to different criteria. After that, the report is basically divided into two section according to the historical classification of computer animation systems: computer assisted animation where traditional animation is just helped by the use of computers, and modelled animation where the use of computers is essential in different aspects of the animation process. Special emphasis is placed on motion specification and generation.

Finally, special attention is paid to human body animation because of the great interest of this specific field of computer animation and the peculiarity of its problems.

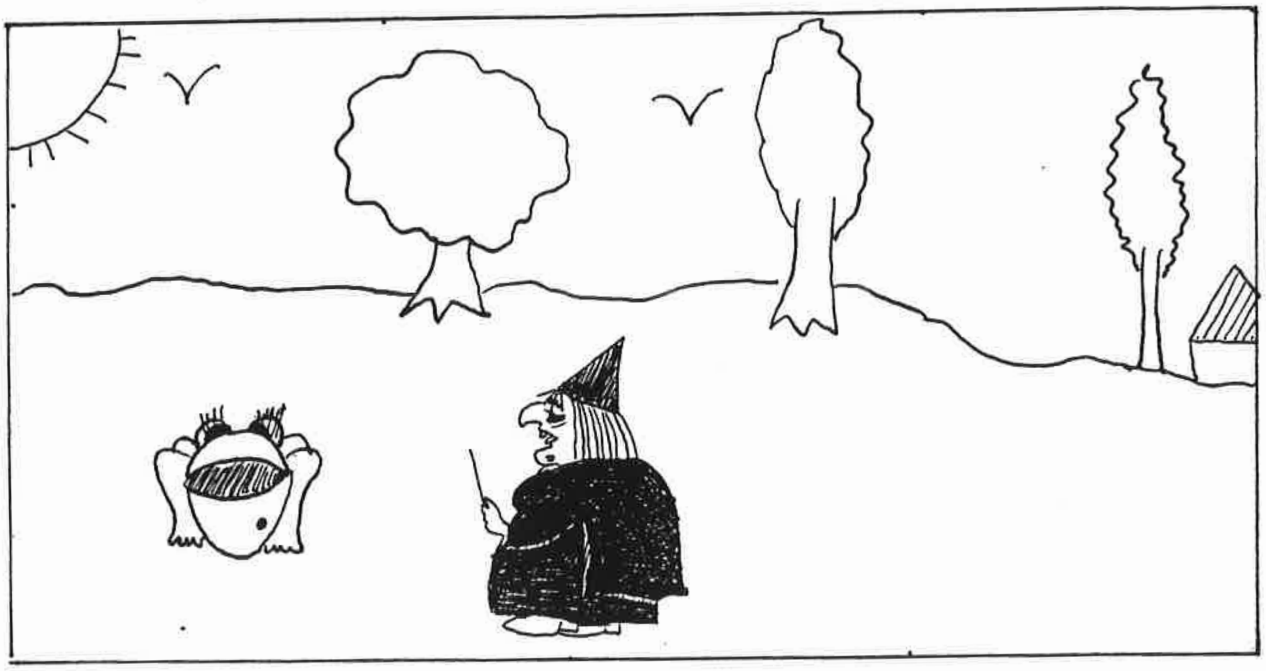
Hardware aspects and postprocessing have only been taken into account in the classification section but have not been studied.

1. INTRODUCTION

Computer animation covers a large range of different techniques that sometimes have very little in common. Almost all the systems that work with the temporal variation of one or more elements or parameters of computer generated pictures are called "Computer Animation Systems" . This is obviously a very general definition that includes everything from control motion systems to 3-D animation films or video games. Even if the variable time is the common characteristic of these systems , it seems not to be their main interest : in fact most of the papers about computer animation don't really talk about time but about rendering, scene edition, user interfaces ... (MaT85a).

It seems that, in order to clarify this topic it will be interesting to class the so-called computer animation techniques and try to separate them into their different parts in order to distinguish what exactly is "animation".

In a picture, as illlustrated in fig 1, a lot of variations can happen in a period of time. First of all if we consider a scene composed by actors, this composition can change: actors can appear or disappear from one frame to the other. These actors can have some kind of motion: geometrical motion such as scaling, translating or rotating and transformations such as changing shape, inflating or deflating. The attributes of the elements can change: colour or texture, and also effects of lighting and shading. In the case of natural phenomena modelled by particles systems (ReB85, Ree83), particles can born and dye following a stochastic process. Finally the movements of the camera , real or virtual, zoom,



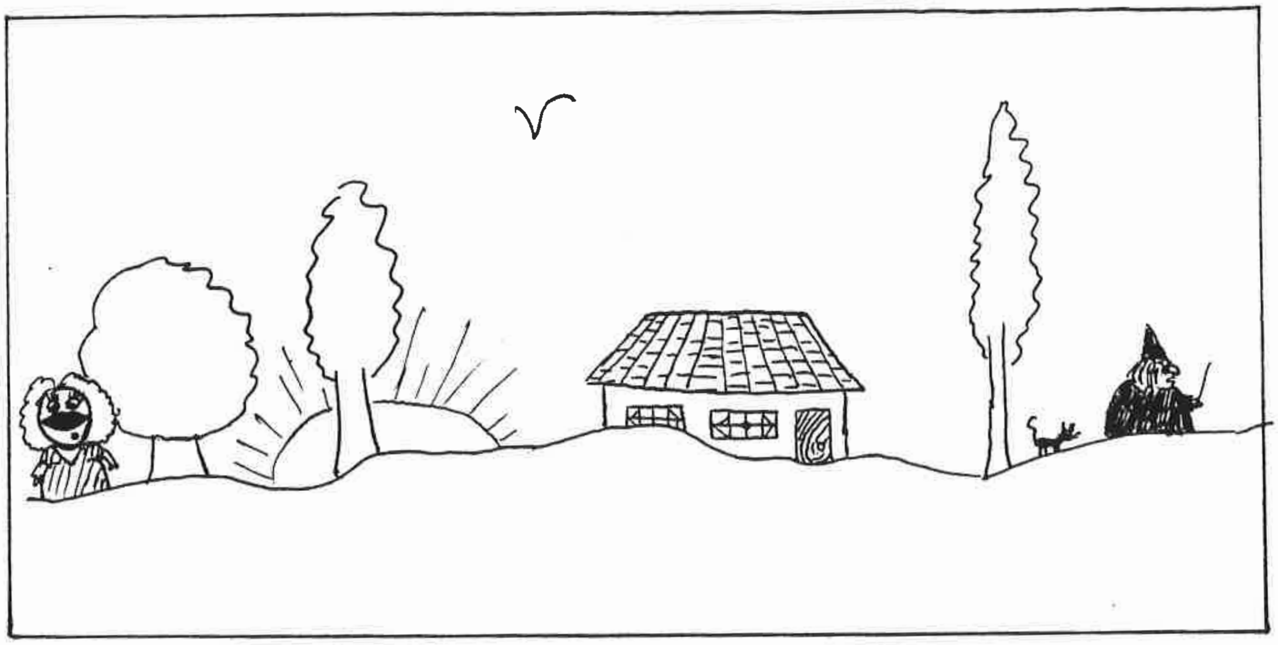


FIGURE 1. TEMPORAL VARIATIONS IN A SCENE.

instant t_1 : inicial
instant t2 : final

From t1 to t2 there have been the following changes:

- change in the orientation of the point of view.
- apparition of actors (the cat), and disappearences (the bird)
- transformation : the frog into a princess
- colors have changed (the trees)
- light also has changed (different lighting from morning to afternoon).

pan and other cinematografics effects constitue an other kind of temporal variation. All these changes are usually referred to as "actions" (Ros85a) and time is computed as a discrete variable. Animation is seen to be the production of a sequence of pictures one for each of the considered instants. Recently, related to the introduction of the stochastic models of representation of fuzzy objects mentioned below (Ree83) time has been considered as a continuous variable : the goal of animation is to determine an evolution law more than discrete values so that the process of animation is not a pre-determined one.

Computer animation is generally divided into four phases:

- scene edition and object modelling.
- animation itself.
- image rendering.
- post-processing: shooting and synchronization

It is important to keep in mind this separation in order to diferentiate the specific problems of each phase. We can consider, for instance, a computer animation sequence in which a sunset is simulated by varying only the colors and the lighting of the static picture. Here the problems of animation are in fact problems of rendering. In the production of a 3-D film such as Tron (Walt Disney Productions 1985) the three phases are clearly diferentiated and almost all of the possible temporal variations described below are used.

In a system of detection of collisions between objects (UOT83), only geometrical movements of the objects will be considered. The rendering is not important but the numeric results are.

2. CLASSIFICATIONS

There are many criteria of classification. Most of them are classical ones but they have rarely been exposed together. In table 1 we summarize all of them and then we develop each topic a little more. All these criteria are interdependent and they are all related to the capabilities of the hardware in which systems are implemented.

2.1 Historical :

Historically computer animation is related to the production of films : it was born as a simple process of mechanization of the long and tedious manual works of editing, generating key frames, colouring ... that were used in order to produce cartoons as we will explain further (section 3). This kind of animation is generally called "Computer Assisted Animation". It is now of current use in the production of commercial films.

In modelled animation, the computer stores a model of the objects that compose the scene and computes the different actions that can be taken (MaT85e). The process is not as operator-dependent as computer assisted animation and can have more applications than the production of films.

2.2 According to the application.

It is possible to distinguish two main applications of computer animation. The first one is the production of films for entertainment (cartoons), didactic or publicity purposes (commercial or political). The second one is the simulation of natural or industrial processes useful in engineering

TABLE 1 : CLASSIFICATIONS OF COMPUTER ANIMATION

1) Historical :

- computer assisted animation
- computer modelled animation.

2) According to the aplication:

- production of commercial or entertainment films
- industrial or scientific simulations.

3) According to the system's motion control

- guiding
- program level
- task level

4) According to the dimension of the system.

2D // 2D 1/2 // 3D

5) According to the system of generation of pictures.

- by drawing
- by instantiation of primitives
- by reconstruction of a real model

6) According to the "animation" model:

- based on kinematic data (positions, speed and accelerations) :
  * interpolation of calculated or shooted key-frames.
  * definition of geometrical tranformations
- based on dynamic data (forces and torques)

7) According to the complexity of the rendering: more or less realism.

8) According to the complexity of the post-processing.

9) According to time performance:

- real time animation
- batch animation + shooting

(robotics, CAD/CAM,detection of collisions, verification of paths), or science.

The objectives in one system or the other are quite different : in the first one the rendering is very important, the pictures are generally pre-designed by artists in such a way that all the unknowns are solved before creating the film . The film should be very "realistic" but the motions can be fancy: there are no physical restrictions.In the second one the numerical results are very important, actions are not pre-dicted by an operator but by equations.

In table 2 there is a non-exhaustive list of possible applications of computer animation.

2.3 According to the system's motion control.

Computer animation system can be more or less animator dependent. Zeltzer (Zel85) defined three levels of autonomy of the systems.

In the guiding systems, the animator should know a priori the different positions the objects of the scene will have at each moment and have to specify it to the computer. The computer assisted animation systems (BuW76, Cat78, Gom84) are typically guiding tasks.

In the program level systems, the computer loads the sufficient knowledge to interpret elemental commands specified in a script (program) by the animator. There is no need for the operator to compute the different positions of the objects. He just has to decompose the movements into elemental ones and write them orderly as sentences in a program . Some of these systems allow the so-called "adaptative motion " in

TABLE 2: A FEW REFERENCES OF COMPUTER ANIMATION APPLICATIONS

- FILMS
  - For art and entertainment :

    "TRON", "STAR TREK", "VOL DE REVE " (Lan82, MaT85a, GiV85,Com86)

  - For advertisments (ChE83, Wat82)

  - For education (Mei85)

- SIMULATIONS
  - Militar/Industrial :
    * Determination of paths, collisions's detection (UOT83,Fou86,All84)

  - Industrial/Robotics :
    * Visualization of mechanisms or robots in movement (NoK85, LaP84,HaC86).
    * Reconstruction of models based on the scenes viewed throught sensors (ZSS78) .
    * Analysis of vibrations (MaS82) .
    * Simulation of nuclear power plant (CuS85)

  - Physical science :
    * Simulation of waves propagation (CaD84)

  - Medical/surgery :
    * Reconstruction of human brain (Or86,BTTU84, FYZ85)

  - Bioquimichal science:
    * Reproduction of natural phenomena impossible to film (WRDD86)

  - Programming:
    * Programs illustration (BrS84, LLM76)

which for each period of time the information of the environment is used in the computations of the positions in the next period. A few 3-D modelled computer animation systems such as MIRA (MaT83c), GRAMPS (ODo81), ASAS (Rey85b) are program level systems (also called animator level system). They have the inconvenient that working with them suppose for the animator a certain knowlegde of computer science so that their use by artists is still limitated .

The task level systems create a database with the necessary information of the world in which objects can move. The animator gives only additional information (initial position, forces to be applied ...) and the computer generates itself the movements, following the physical or mechanichal laws it knows. The problem is one of degrees of freedom : the computer can generate one or more possible motions related to the input data. The task level systems are one of the most important current subjects of reseach in computer animation even if the existing systems aren't yet totally of task-level : TEMPUS (Bad80), SAS (Zel82) (TsD84).

They arouse special interest because they constitute the real way to solve the problem of the interface with user: they should be user_friendly, ready-for-use by non-specialized users (non-programers). Research in this topic is related to another very recent field of study: artificial intelligence and the development of expert systems.

2.4 According to the dimension of the system

Animation systems can be two- or three- dimensional or even 2.5- if they work in some way with the depth of the pictures. Generally 3D animation systems, more complex, are concerned with modelled animation while 2D systems are usually Computer Assisted systems.

2.5 According to the system of scene generation

The generation of the frames can be done in several ways. The most common system, specially in computer assisted animation, is with a graphic editor generating a 2-D picture. In 3-D an object can be modelled, from its 2-D projections, by sweep. Curves or surfaces can be generated by the information of their control points.

It is also possible to work with instantiation of elemental primitives: in a human body system to select arms, legs , heads., or cubes, cilinders, screws.. in an industrial environment.

Another way used in special applications is generating a 3-D model from single views of the object, or of its sections, as used in interactive surgical planning (BTT84).

It is also possible to film a real sequence, and digitalize it .

2.6 According to animation model

As shown in table 1 from the user's point of view, it is possible to disguinsh two main ways of giving actions to a scene: by kinematic or dynamic data.

The most common kinematic model is key-frame based. The user has to give some key frames and by interpolating between them with some method (linear or with splines), the intermediate positions can be found. It is the classical method used in traditional animation and in computer assisted animation and it will be explained in more detail in section 3. It allows shape transformations or movements.

In addition to key-frames, paths can be given that help the process of in-betweening.

Another way to specify actions kinematically is to define explicitly the elemental actions to be taken by each entity of the scene, such as rotating around an axis, translating to a determinated point.... This supposes a more sophisticated system and is generally related to operator level systems.

Finally, in the dynamic model, users must define forces and torques to be applied, or, in a stochastic model of particles, a life law. Entities move as the result of the application of these forces and following physical laws and constraints.

2.7 According to the rendering

The rendering of the frames can be more or less complex. The traditional aspects of image rendering in computer graphics are:

- removal of hidden surfaces
- shading : light reflexion,transparency,textures, spatial anti-aliasing...

In addition there are specific aspects of rendering in computer animation such as the temporal anti-aliasing called motion blur which tries to remove the breaking and jerking appearance that computer animation sequences have .(PoI83, KoB83, Re85).

The introduction of the effects of a virtual camera that can move also brings about specific problems

All these calculations involve a great deal more CPU time if we consider that they should be performed for each frame. In this sense very little has been done via software to find specific solutions to this problem in computer animation. Some algorithms used for frame to frame coherence to save hidden surface calculations (HuZ82, Hub80, ShG82) constitute for the moment the only research that has been done in this field.

On the other hand, there is a special technique of moving lights and shadows (FLM86) which is interesting and should be the object of further study. The research herein has been mainly centered on increasing hardware performance, with special workstations (StF83,Nic84), cameras (MaB80) or parallel processing (Kah78), but, as far as hardware is beyond the scope of this report it will no be treated herein.

2.8 According to the post-processing.

The post-processing can be also more or less complex and depends basically on the specifications of the final product. For instance, a process of synchronization of sound, music and pictures (FLT83) may or may nott be needed.

2.9 According to time performance.

To produce the sensation of continuity for a human eye pictures must be displayed at a speed of 24 frames/scd. Depending on the level of complexity of the pictures and on the capabilities of the computer it will or will not be possible to display display frames at this speed. Two-dimensional video-games with very limitated drawing are real time systems.

Producing a single 3-D realistic picture, however, can take several hours. In this case the frames produced are first recorded from the screen and then projected at 24 fr/s. An alternative is to save the value of the pixels for each frame in a frame store display station and then display it at real time: it is the so-called "real-time play-back " (AcW80,EHM-84). Other techniques are the slit scan and scan based animation that work with the movements of the real camera and the opening of its shutter (Par80).

3. COMPUTER ASSISTED ANIMATION

Nowadays almost everything has been said about computer assisted animation. On the one hand it is a limited technique and is, within its scope of action, totally under control. On the other hand it was the first subject of research in computer animation so that since the sixties, there has been extensive bibliography about it.

We will explain briefly the different steps of Computer Assisted Animation (MaT85e, Cat78) , and talk more specifically about interpolation methods and about inherent problems of this technique.

3.1 Traditional Animation

The traditional (manual) animation process is divided into the following steps (Cat78):

	- story written (1)
pre-proc.	- lay out (2)
	- sound track (3)
	- design of the key frames (4)
animation	- in-betweening (5)
	- xeroking and inking (6)
	- painting on acetate (7)
post-proc.	- checking (8)
	- shooting (9)
	- editing (10)

The story is written grafically as a story-board (1): an illustrated script which is laid out (2) designing the characters and their actions and drawing the background. The third phase of recording the sound track (3) allows actions to be synchronized with sound and music : it is, in traditional animation, part of the pre-processing. Then some frames of the actions (<u>key-frames</u>) are designed and form the base for generating intermediate frames by interpolation between them. This is the so-called <u>in-betweening</u>.

Then it is necessary to ink the lines, to transfer the drawings to acetates by special photocopying (the xeroking) (6) , to paint (7) and shoot the frames (9). Acetates are used because of their transparency which allows us to work with layered cells each of them with only one actor : this makes the process of composing the final scene , checking and modifying it, easier.

The computer can assist almost all of these steps and specially (Coh85, Lan82) the process of key-frame generation , in-betweening and colouring.

3.2 Computer Assisted Animation

3.2.1 <u>Key-frames generation and painting</u>

Graphic editors are the first module of computer assisted animation systems. As for other fields of aplication of computer graphics, (MaT86) , more or less sophisticated, they should allow at least the selection of elemental drawing primitives: points, lines , circles , arcs and eventually other curves, working with libraries of created shapes, applying geometrical transformations (rotations, translations..) , zooming, panning....

The colouring can be a part of the graphic editor or constitute by itself an independent module simulating the tools of a painter and called Paint Systems , largely used in art either for creation, or for pedagogical tasks (GiV85,-Mei85).

3.2.2 Automatic In-Betweening

In-Betweening consists of generating intermediate pictures between two key frames by interpolation (BuW71). This is generally made up in two phases : (see fig.2)

1. decomposition of the scene into individual elements for interpolating where correspondances between points of one and the next frame are found.
2. definition of the path between the corresponding points: interpolation itself.

Reeves (Ree81) considered four criteria in order to evaluate in-betweening algorithms:

- the generality (usable in many cases)
- the smoothness
- the efficiency
- the economy

There are various method of interpolation (MaT85e) : the linear one (interpolating two positions with a line) is the one most generally used, even if it produces effects of breaking in the final product. Some research have been done to introduce splines interpolation in order to increase the smoothness . The drawback of these interpolations is that they are very time consuming. In this sense the key frames should give, rather than exact points in each frame, control points of the temporal curves, in such a way that the animator could interactively modify these control points to obtain the accurate paths.

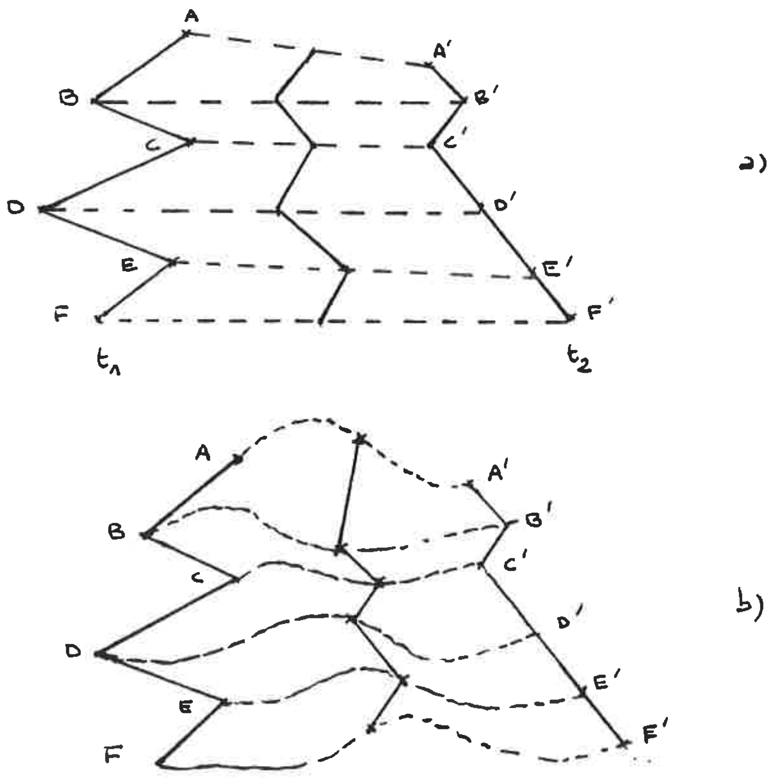


FIGURE 2. INTERPOLATIONS BETWEEN KEY FRAMES
a) LINEAR
b) NON LINEAR.

The design in two dimensions must deal with two specific problems when automatically interpolating frames, as explained by Catmull (Cat80).

First of all, as two-dimensional pictures are in fact projections of a three dimensional world, they should recreate by their sequence, the coherence of this 3-D world. In a manual interpolation, this is possible because all the information about this world is in the animator's head. In a scene where a man is rotating his head the animators know that , even if it does not appear in the profile, the other eye exists and at which frame it should begin to be drawn. All this passive information is lost when automatically interpolating so that alternative solutions must be given.

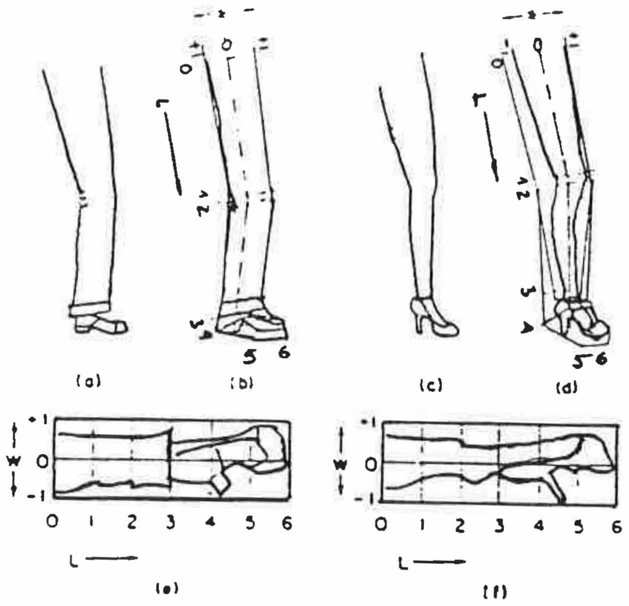
In order to override this problem different solutions can be taken. For commercial purposes or for some entertainment films it is always possible to simplify the drawings in order to avoid this difficulty : most of television cartoons are very elementary.

If more sophisticated pictures are needed it will be probably necessary to define a 3-D model from which to use modelled animation. In this sense it would be especially interesting to reconstruct the 3-D world based on 2-D projections : it is an open, and active research field but still in development (WeN85, TuT84, Ros85b, JiC85).

In 2-D it will be necessary to give additional information for the interpolation, increasing the number of key-frames. This involves tedious edition work and it is the reason why special methods of interpolation have been created, to simplify this work.

The first technique for simplifying in-betweening is not interpolating the pictures in their totality but locally, as used by Julian Gomez in his 3-D system <u>TWIXT</u> (Gom84). For a particular object in a scene, a key-frame is needed wherever this object is involved in some kind of action, so that the global key-frames should reflect all the changes of all the objects in the scene. In <u>TWIXT</u> a collection of local inbetweenings is considered in the <u>track</u> of the different parameters or attributes of the entities of the scene : for each element of the scene an interpolation is made of its characteristics along its "life" whenever some <u>event</u> happens to it and not between two arbitrary instants.

An other method is the well known "skeleton technique " (BuW76) in which only a stick figure representation of the actors is used to interpolate . The skeleton coordinates configure a network of polygons that can be deformed and



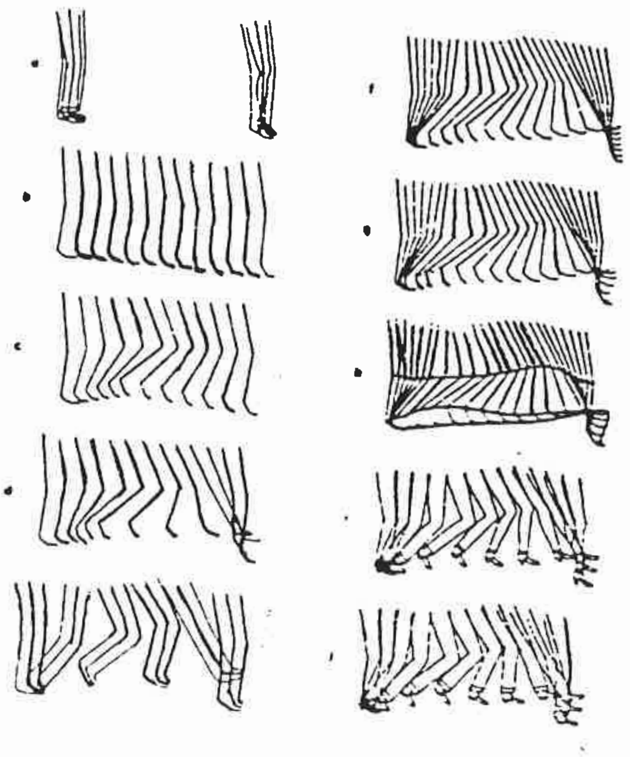


FIGURE 3. TECHNIQUE OF THE SKELETON

distorsted throughout the motion in such a way that user's control can be greater, easier (more interactive) and consequently can allow complex motions to be developed. The skeleton selection is made as interactive as possible giving to the user the possibilities of modifying it (Fig 3).

Finally a third method is that of Reeves's (Ree81) moving constraint points. Between two or more frames, the total real path of some critical points can be defined by the user who gives an orientation and some constraints to the interpolation.

The second problem stated by Catmull is related to the kinematics of the scenes: how specifying accelerations and decelerations.

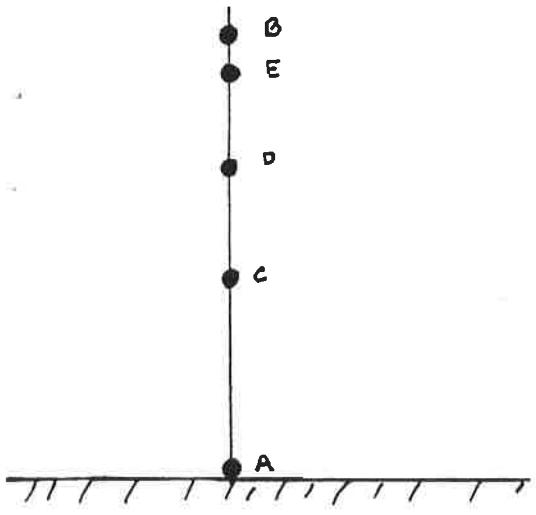


FIGURE 4.

The ball goes up and down. If only three positions are given (A,B,A), the deceleration (A->B) and acceleration B->A) will not be reflected : more key frames are needed (c,d,e)

In order to solve this problem the P-curves where introduced by Baecker in his system : GENESYS (Bae69). They consist in a dynamic curve of path where the separation between points gives an indication of time as shown in the figure 5 .

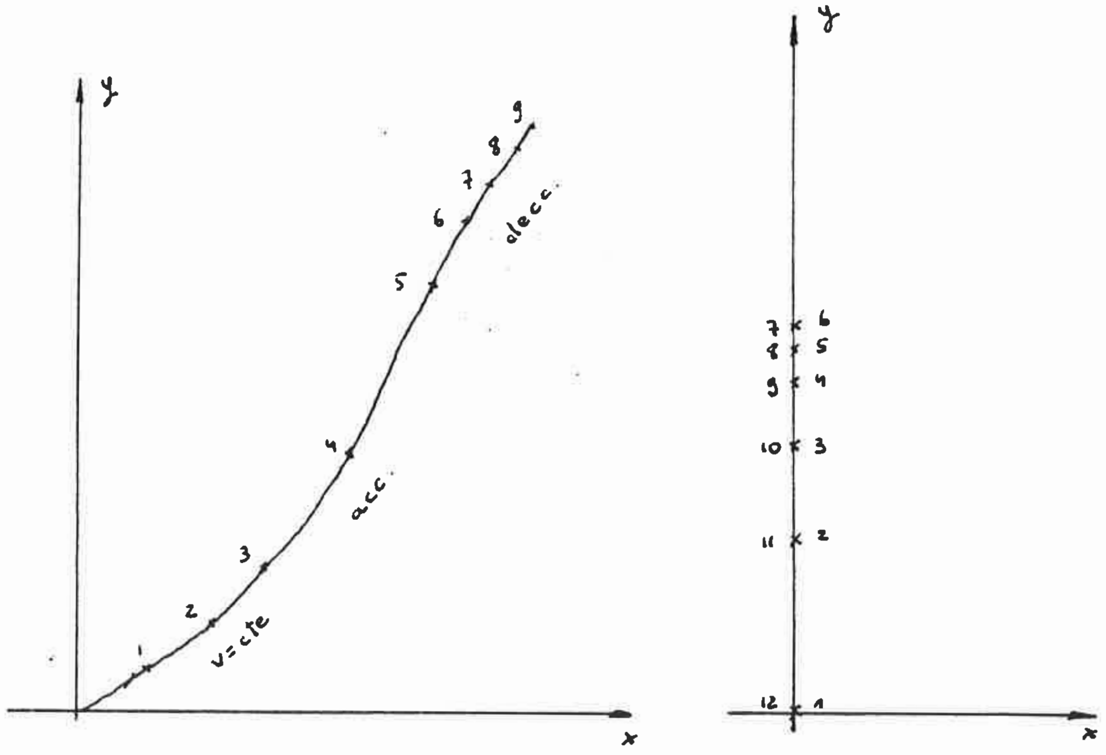


Figure 5. a) A P_curve : the separation between points is the temporal dimension.
b) Application to the picture 4 's case.

4. MODELLED ANIMATION

As explained in section 2. modelled animation includes three different but related aspects which we now analyze :

- scene edition and object modelling
- movement definition
- scene visualization

4.1 Scene description and object modelling.

As in other computer graphics fields [Par85], scenes to be animated must first be generated (edited), stored using a geometric model to describe its elements and structured in some way in order to define relationships between different elements. Probably because animation systems use well known techniques to solve these aspects, computer animation papers rarely treate this topic except as a side effect of motion control techniques.

Several helpful features may be remarked for the representation of objects and scenes in order to facilitate the production of animation :

- In keyframe systems, object models must provide a set of singular <u>points</u> to be interpolated.

- The computation of movement is often based on a <u>schematic representation</u> associated to the geometric model [LuC86]. This is specialy interesting in dynamic systems using a <u>mechanical model</u>.

- A <u>simplified geometric model</u> with a less accurate description of the object than the actual geometric model [ChE83] is useful in two different ways. Preliminary tests may be speeded up using simplified models. In another sense, these models may be used for final production when the details of the objects will be obscured either for a specific visualisation (e.g. projection in a small display area) or quick motion where details are not assessed (Ple86)

- The possibility of assigning <u>relationships</u> between objects or the elements of the objects allows us to relate their motion (e.g. a constant distance to be kept between two objects).

2-D objects edition is usually performed by means of standard drawing systems, and 3-D elements are often generated by object instanciation. In the second case we may distinguish between two different systems:

- Instanciation of primitive objects like parallelepipeds and quadrics (Sug83, Veg84, Her78). In these systems more complex objects are built by glueing or theoretical union of primitives.

- Instanciation of complex parametrised objects previously modelled (MaT85b). Now the animator may generate different objects of the same class, just giving different parameters values. For example, different dogs may be instanciated giving specific values to the parameters defining their size, color, texture, ...

Because of the special interest of some animation applications in reproducing real life scenes, these are often captured from sequences of pictures or images produced by scanning, which are used to reconstruct 3-D objects (WeN86), (HsZ85). This interest of some computer animation applications in reproducing the real world has, on the other hand, stimulated the use of fractals [Opp86] to model trees, mountains ...

Whole scenes and specific objects are usually structured hierarchically. This organisation offers two interesting features. First, hierarchical description of objects allows the animator to specify motion at different levels of this hierarchy giving, for example, a movement to the whole object and different specific movements to each of its elements. Another ability of hierarchical description of scenes and objects is the posibility of using simplified descriptions of them by means of despising low levels of the structure containing unnecessary details either for a specific view or for initial tests.

Particle systems (Ree83, ReB85) have been developed for computer animation in order to model special objects whose movement is inherent to their life such as fire, waves..etc. It can be also applied to other objects such as trees, grass .. in movement caused by wind. The objects are considered to be composed of particles. A stochastic law of evolution is given in function of which the "age" of the particles is determined. The position of the particles depends on their age. So motion is involved in the model of the object.

4.2 Movement definition

4.2.1 Types of movements

A computer-generated image is defined by a set of parameters describing:

- the structure of the scene, the objects and their attributes.
- the position of the observer (camera position and

camera state)

- light source position and attributes.

As stated in section 1, computer animation is attained by varying the values of some or all of these parameters and recording the images obtained for successive values of the parameters. Given this, these changes may start at different moments and have different behaviour : they must be synchronized. A few works have been published which focus on light sources and camera animation whereas the description of the movement of objects is the main research field in computer animation.

ANIMEDIT (FLM85) is an animation editor in MIRANIM system which allows the definition of four types of lights: ambient light, directional sources, positional sources and spots. Each defined light has a name, except for ambient light, and a list of parameters whose number changes from one type of light to another. For example, to define a positional light source the animator uses the command:

```
SOURCE <source identifier> <RGB intensity colour> P <Loca-
        tion vector>
```

where P specifies the type of light we are defining. Parameters may be either constants or variables. In the second case, each variable may be affected by any evolution laws.
We next show an example taken from (FTMT85) where a positional source is defined and evolution laws are applied to its parameters (intensity and location):

```
SOURCE LUX INTENSITY LOCATION
VEC INTENSITY A <1,0,0> <0,1,0>
LAW LINEAR LIN
EVOLUTION INTENSITY LINEAR 0 10
VEC LOCATION A <10,20,30>
LAW ALTERNATE OSCILLATION
EVOLUTION LOCATION ALTERNATE 0 10
```

VEC command gives the initial (and final for intensity) values of the parameters, LAW selects an evolution law and EVOLUTION applies the law to the parameter during 10 seconds.

In MIRANIM (MaT86) system, the command:

```
 CAMERA <camera identifier>, <eye point>, <interest point>
```

defines a camera by its location (vector eye point) and the point towards which the camera is directed (vector interest point). Transformations of the observed scene like zoom and spin may be easily applied to the camera referred to by its identifier. The user may also apply to the camera several colour filters and fog filters.

These location, viewing and attribute parameters may be animated by applying evolution laws to them in a similar way to that used to animate light sources (FLMT85) explained above. In addition to the possibity of animating zoom and spin, and producing typical movements like panning, MIRANIM allows the animator to select a path to be followed by the camera. This path can be defined in three diferent ways:

- applying a predefined law
- defining a curve
- applying a procedural law.

In the next example taken from (MaT86) a predefined, circular movement (CIRC) around axis <0,1,0> passing through point <5,6,7> with a speed of 1rd/s is applied to the camera location (EYE) during ten seconds.

```
CAMERA MYCAM, EYE, INT
VECTOR EYE, A, <10,20,30>
LAW TURN, CIRC, <5,6,7>, <0,1,0>, 1, 0
EVOLUTION EYE,TURN,0,10
```

In MIRANIM system we may also assign the movement of an animated object in the scene to the camera.

4.2.2 Movement definition modes

In this section we present the different ways of defining motion in computer animation from the point of view of the animator (movement specification) and from the point of view of the c.a. system designer (movement modellization).

Movement specification

The animator may specify the motion of the elements defining an image using three different tools :

- GUIDING (Key frame and interpolation): in these systems the animator defines a set of frames giving the parameters that define them, and intermediate frames are generated by interpolation, as explained in section 3.
- PROGRAM LEVEL (languages): in which the animator specifies the motion algorithmically.
- TASK LEVEL (Motor program handling) : where high level commands perform predefined or computable movements.

From the animator's point of view each of these three definition modes presents several advantages and drawbacks but the fundamental criteria should be the simplicity of their use. Motor handling systems are easy to learn whereas the use of programming languages needs computer science knowledge, thus a longer training period.

Motor handling systems, friendlier to use, allow the animator to obtain a complete product quickly. The major drawback of these systems is their accuracy, because the user has no control over the motion of individual elements. Furthermore, the animator often needs new motions not yet defined or computable so that a computer scientist must help him to implement the necessary tools.

The main advantages of key-frame systems are that no computer science knowledge is needed at all and that the animator may control the motion at any level of detail. Their main drawback is that specifying a complex movement requires a great amount of data to be given by the animator. This may be highly laborious and in some cases practically impossible (see section 3.).

Finally the use of programming languages is a powerful tool that provides the possibility of controlling the motion of any element or set of elements offering different levels of abstraction. The problem of these systems is, in addition to the above-mentioned need for users with knowledge of computer science, that some movements are sometimes very dificult to specify algoritmically.

Because of the different advantages and disadvantages of each motion-controlling technique, computer animation systems often provide different possibilities of specifying motion. For example, a key frame based system may be supported by a script defining relations between elements whose motion affects each other (ChE83). On the other hand, programming languages and motor handling based systems (Ze85) sometimes use key-frame facilities either to specify some special movements or to define new motors.

Some published works presenting the different techniques used in motion definition (Be85) give a longer list of methods that are, in fact, special cases of the above-mentioned techniques, or a combination of them.

Movement modellization

Independently of the way the animator specifies movement, different models are used to describe it. These models may, in a first step, be classed as kinematic models and dynamic models.

Kinematic models produce motion from positions, speeds and accelerations (easies), whereas dynamic models describe motion by a set of forces and torques from which kinematic data are derived. In order to be able to apply dynamic models, the objects to be animated must be defined using mechanical elements. These are of two types: material (masses) and joints (rods,springs).

Most of the currently operative computer animation systems use kinematic models which consume less computer time than dynamic models and are more intuitive to use. On the other hand,dynamic models are useful for describing complex and quick movements that could be laborious and even impossible to model with kinematic techniques.

Four types of kinematic models are used:

- Interpolating models. Motion is represented by a set of scenes located in time (or provided with speed and acceleration) and an interpolation method to compute intermediate scenes.

- Procedural models. Here motion is described by an algorithm and a set of input data. These models are used to implement typical movements like oscilation, circular motion, ... or special movements specific to each application.

- Recorded models are files storing data that define realistic or pre-computed motion. This information has been previously recorded from the real world using different techniques and devices or computed from other models.

- Natural models. Motion is described using commands of a near natural language notation. A set of rules is applied to the parameters of these commands to produce the motion.

4.2.3 Guiding systems

Key-frame techniques already dealt with in section 3 for computer assisted animation are also used in modelled animation in the sense that they are applied to 3-D objects, virtual cameras or light sources. Furthermore, the idea of key frame systems where motion is in a first step specified by a set of images to be interpolated, is extended to the use of a sequence of forces and torques to be applied in successive time spaced scenes (i.e. frames). So the animator must first define a sequence of scenes or an initial one, plus a sequence of torques and forces. VIRYA system (Wil86, WiB85 allows the use of both techniques (kinematic and dynamic) in combination.

In the simplest key frame systems, the interpolation between key frames (in betweening) is performed linearly in space but may be non-linear in time in order to accelerate and decelerate motion. More evolved systems allow the animator to apply mathematically defined movements between key-frames or to describe path curves (ChE83). A path curve is the definition of a space versus time curve to be followed by a specific point between successives frames, in kinematic systems, or the definition of a curve of forces (torques) versus time to be applied to a specific point.

If the elements of the animated objects are rigids, defining all singular points as vertices at each key frame is enough, but for non rigid objects, different types of interpolation are employed (MaT85) to derive the shape of the object between key frames from in-betweened singular points and key frame data.

More sophisticated key frame systems like the one proposed by Chauang and Entis (ChE83) are based on a script that provides the possibility of defining motion relationships between different elements. In this way, the motion described for a given element may automatically affect the motion of another element. Chaung-Entis's system also allows the animator to modify interactively the results of the interpolation frame by frame by means of a simplified first visualization.

Hanrahan and Sturman [HaS85] present a hibrid system combining programming languages and in-betweening approaches for computer animation where control motion is in fact exclusively performed by keyframe interpolation. From the point of view of movement specification the only facility given by the language is just reading key frames from a data base.

4.2.4 Program level systems

Computer animation programming languages are usually extensions of general purpose languages including new tools to help the animator's task. These are essencially specific control stuctures, data abstraction and parallel programming facilities. The former are basically special kinds of loops where each iteration is associated to a frame. Data abstraction allows the instanciation of objects provided with their own motion. Finally, parallel facilities help the sincronisation of movements.

The script concept (description of a sequence) appears at different levels in these languages and their environments:

- Control structures which are loop statements producing a frame after each iteration and controlling the motion of a sequence.
- Lists of commands to be applied to a specific object.
- Complet programs or procedures controlling the production of a sequence or the whole film.
- High level commands using procedures designed with a computer animation language.

We next present some computer animation programming languages including the above-mentioned tools and script implementations.

HOPE's animation extension

Arya presents in (Ar84) an extension to the functional programming language HOPE which provides useful data structures and functions to produce animation. A movie (list of frames where a frame is considered as a pair (number/object)) is created by the function:

```
 DEC ANIMATE : COST x SCRIPT x STAGE -------> MOVIE
```

where cost is a list of actors, stage is a list of commands describing the motion of actors which is specified as follows:

```
  data script  = = list(command)
  data command = = name x action x entry x exit
  data action  = = list(  mov_act  (..)  ++  rot_act  (..) ++
                      grow_act ++ shrink_act(..) );
```

Parameters entry and exit are frame numbers where command is activated and desactivated. An actor is a hierarchical description of objects such that each element may be animated independently. The frame may be generated individually, using function :

```
  DEC  MAP_COM : COST x COMMMAND  ------> FRAME
```

So the production of a complete film will be performed by a combination of animate and map_com functions. Motions (actions) corresponding to non mathematical behaviours may be specified by functions such as:

```
   DEC TWEEN : FRAME X FRAME    -------> LIST(FRAME)
```

which takes frames as input and returns a list of frames resulting from the in-betweening operation.

We next present an example taken from (Ar84) which animates a bird:

```
 ANIMATE (bird, fly, sky)
 fly = = REPLICATE(COMPOSE(twist-act("wing"+70),
                           twist_act("wing"-70))
```

Bird is the identifier of an object with two wing objects in a lower level of the bird object hierarchy. Action fly consists in repeating (replicate) sixty times the composition (compose) of function twist_act with different parameters in such a way that in each frame wings are rotated about their origin -70 and then +70 units.

More recent work of Arya on the use of functional languages for computer animation is presented in [Ary86].

ASAS (ACTOR/SCRIPTOR ANIMATION SYSTEM

In ASAS (Re82), an extension of lisp programming environment, the production of a film is specified by a script function. This operation includes the initialization of the variables and a sequence of expressions which are usually animate blocks and has the following definition :

```
 (SCRIPT <IDENTIFIER>
       (local <variables initialization>)
       (animate <list of cue expression>)
       (animate.....))
```

An animate block is a loop wich produces a frame after each iteration. Its body is composed of a series of cue expressions each one controlling the motion of an actor. So, an animate block may be considered as an operation returning a sequence of the film. The structure of a cue expression is :

```
(CUE (at <frame number>
     (<operation> <actor identifier> <list of paramaters>))
```

where frame number specifies the frame where operation is activated. The operation is typically starting (start) an actor's motion and stopping it.

Animate block execution and actor's motion are stopped by the operator cut which accepts a frame number as parameter.
An actor is a process associated to an object, which is run once for each iteration of the animate block. The actor is on; i. e. between its start and stop associated operations.

We next present an example of a program to produce animation in ASAS taken from (Re82):

```
(SCRIPT spinning_cubes
      (local: (runtime 96)
              (midpoint(half runtime)))
      (animate ( cue (at 0)
                     (start (spin_cube_actor  green)))
               ( cue (at midpoint)
                     (start (spin_cube_actor  blue)))
               ( cue (at runtime)
                     (cut))))
```

Two instances of "spin_cube_actor" are activated, one at frame 0 and the other at midpoint (48), each one of a different color. Both are stopped at "runtime". The actor's definition is :

```
(DEFOP spin_cube_actor
       (param: colour)
       (actor( local: (angle 0)
                      (d_angle(quo 3 runtime))
                      (my_cube(recolor color cube)))
       (see (rotate angle y_asig my_cube))
       (define angle
            (plus angle d_angle))))
```

This operation receives a colour used to create an instance (recolor) of a predefined object (cube) and returns "my_cube" rotated by "angle" about y_axis . Finally "angle" is updated by an incremental velocity ("d_angle") of the angle for the next frame.

In ASAS, actors may exchange messages in order to mantain predefined relationships. This goal is attained by using operators send and receive. The first one takes an actor's identifier to which the message is to be sent , and a message. Receive is a case structure in which a function is activated depending on the clause matching the incoming message type.

MIRA 3_D

MIRA 3-D is a graphical and animation language in MIRANIM animation system (MaT83) which is an extension of Pascal including abstract graphical data types. Two different tools are provided by MIRA 3_D in order to produce object animation: figure and actor data types.

Type figure allows the programmer to define objects procedurally and has the following structure:

```
 type <identifier> = figure <list of parameters>;
         var<variables declaration>;
         begin <list of statements> end;
```

where declared variables may be of the other figure types. The list of statements include geometric constructors and transformation operators that we will describe in section 4.2 as well as standard Pascal statements. The list of parameters may contain frame numbers which are used to compute actual parameter values. In this way, the instantiation of several variables of the same figure type with successive frame numbers gives the parameter values of an object in these frames.

Four fundamental statements allow the programmer to handle variables of type figure :

```
  CREATE <figure> (<parameter list>)
  DELETE <figure>
  DRAW <figure>
  ERASE <figure>
```

So, the production of a sequence of a moving object may be attained by a sequence of create, draw and erase blocks of statements, where each create state instantiates an object whose parameter values depend on the frame number.

We next present an example taken from (MaT83) illustrating the definition and use of the figure types to animate a bird.

```
   type  BIRD   =  figure  (FRAME:INTEGER;H:HALFBODY;W:WING;-
                       C,D:VECTOR);
              var
                 RELATIVE:0..CYCLE;
                 FRACTION,BETA:REAL;
                 W2:WING;
                 RIGHTPART,LEFTPART:FIG;
              begin
                 RELATIVE:=FRAME mod CYCLE;
                 if RELATIVE > (CYCLE div 2) then
                    RELATIVE := CYCLE-RELATIVE;
                 FRACTION := (RELATIVE*2)/CYCLE;
                 BETA := LAW(ACCEDECE,ANGLEMAX,FRACTION);
                 ROTATION(W,C,BETA,D,W2);
                 UNION(H,W2,RIGHTPART);
```

```
                    delete H,W2;
                    SYMYZ(RIGHTPART,LEFTPART);
                    include RIGHTPART,LEFTPART
                 end;
```

And the creation and draw of the bird will be as follows :

```
        procedure DRAWBIRD (FRAME:INTEGER);
        var FIRSTBIRD:BIRD;
        begin
             create FIRSTBIRD (FRAME,RIGHTBODY,RIGHTWING,C,D);
             TRANSLATION    (FIRSTBIRD,<<0,0,FRAME*BIRDSTEP>>,-
                          FIRSTBIRD;
             draw FIRSTBIRD;
             delete FIRSTBIRD;
        end;
```

Actor data types are figure types with their own animation. So, the instantiation of an actor variable includes the production of an object with the actual values of its time varying parameters for a sequence of frames.

The structure of these data types is basically the same as the figure type structure plus the definition of the interval of time in which the object will be active (moving) and the definition of animated types :

```
  type <identifier> = actor <parameter list>;
       time <time interval>;
       type <declaration of animated types>;
       var <declaration of variables>
       begin <list of statements> end;
```

Declared variables may be of animated types which are defined as follows :

```
 <identifier> = animated <object's parameter type>
                    val <initial parameter value< <final param.
                         value>
                    time <time interval>
                    law <motion law>
```

So variables declared in the actor's type var section of a specific animated type will move as specified in the law section, in the time interval stated in the time section of the animated type declaration, and will take values starting with initial parameter value and ending with final parameter value.

We next present an example of an actor type declaration for the moving bird.

```
type
    BIRD = actor(H:HALFBODY;W:WING;C,D:VECTOR;T1,T2:REAL);
           time T1..T2;
           type
            ANG = animated REAL;
                      val 0.0..ANGLEMAX;
                      time T1..T2;
                      law ACCDEC(...)
                      end;
            POS = animated VECTOR;
                      val ORIGIN..UNLIMITED;
                      time T1..T2;
                      law BIRDSTEP*BIRDSPEED
                      end;
            var
               TRANS                  :POS;
               BETA                   :ANG;
               W2                     :WING;
               RIGHTART,LEFTPART      :FIG;
            begin
               init TRANS;
               init BETA;
               ROTATION(W,C,BETA,D,W2);
               UNION(H,W2,RIGHTPART);
               DELETE H,W2;
               SYMYZ(RIGHTPART,LEFTPART);
               include RIGHTPART,LEFTPART
           end;
```

Then an instanciation of the actor type is needed to produce and initialize a moving object:

```
init <identifier> (<parameter list>) .
```

In the example explained above to initialize a bird , the following sentence should be used:

```
init FIRSTBIRD(H,W,C,D,10,16)
```

Other references related with MIRA 3-D language are [MaT82], [MaT83b] and [MaT85b].

CGAL#1

CGAL#1 (Com 86) is a language similar to Pascal from the point of vue of its control structures. In CGAL#1 the motion control is attained by the use of two special types of loop: the script and from statements defined as follows.

```
 SCRIPT <start frame> TO <end frame> Do <simple/composed
         statement>
 FROM <start frame> TO < end frame> DO <simple/composed>
         statements.
```

The first one produces a frame automatically after each iteration and, so, allows the definition of a sequence. The second one specifies a motion between two frames of an active period of a script statement.

We next present an example of these statements where three balls are animated (rotated). Blueball starts its motion at frame 50 and stops at frame 150 whereas redball starts at frame 100 and stops at 200. Finally greenball is moving throughtout the whole sequence.

```
SCRIPT 1 TO 250 DO
 begin;
  ROTATE_ABOUT_X (green_ball,1.5);
  FROM 50 TO 150 DO
    ROTATE_ABOUT_Y (blue_ball,1.5);
  FROM 100 TO 200 DO
    ROTATE_ABOUT_Z (red_ball,1.5);
 end;
```

So each ball will be rotated by 1.5 degrees at each frame about an axis (x for gree_ball, y for blue_ball and z for red_ball)

NEM

This is a language proposed [MMZ85] for the representation and animation of humanoids. It allows the possibility of moving objects by composing primitive motions or LISP like functions describing more complex movements, implemented by the user. Composition mechanisms are, in addition to sequential execution:

- Parallel execution.

```
  action_1 & action_2 & action_3 & ...
```

- Guarded execution.

```
  while  event do action
  wait event do action
  perform action_1 except event then action_2
```

For example, to "open" a pyramid by moving its three vertices simultaneously we should write,

```
    while (actualtime < (t+10))  do
      [
       bend vertex_1 &
       bend vertex_2 &
       bend vertex_3
      ]
```

Some programming languages ([DoO81], [HaS85]) presented as computer animation oriented are, in fact, just graphical languages giving some facilties to create, store and recover keyframes to be in-betweened. On the other hand, EXPERTMIRA [MaT86] must be classed as a task level system because it offers a "near natural" language to specify motion.

4.2.5 Task_level systems

As explained above, task-level systems are those in which the user only has to active a "motor" that produces motion. These motors are in fact programs which may be of two types depending on the system:

- those that execute predefined movements by means of an animation language (MaT85).
- those that generate motion from some input data (LuC86), (MaT86).

In the first case, the animator executes commands activating for a specific period of time an element (camera,actor..) provided with a pre-defined motion. So the main inconvenient of these system's is the restriction of movements imposed by the set of existing programs. In the second type the user specifies either an external action to be applied to the object, or a set of facts to be com bined with a set of predefined rules in order to obtain the desired motion.

As explained in section 1, the bases of the second type of task-level systems where given by Zeltzer (Zel85). The TEMPUS system is problably the first near-task-level system and from it others systems are now being developped (Wil86).

EXPERMIRA (MaT86) is a language based in MIRA (see section 4.2.4) and PROLOG incorporating concepts of artificial intelligence. We class it as task-level becaude it provides a near natural way of specifying motion. The path of an object, for example, is described in terms of positional relation-ships between the object and other elements of the scene (e.g. near the house, between two trees,..). Predefined rules are applied to this description of the path and other "facts" as the duration of the motion, and one or more solutions are presented by the system. Then the animator may introduce more information in order to bound the number of solutions (i.e. give more rules) until obtaining a satisfactory one or let the system taking an arbitrary solution

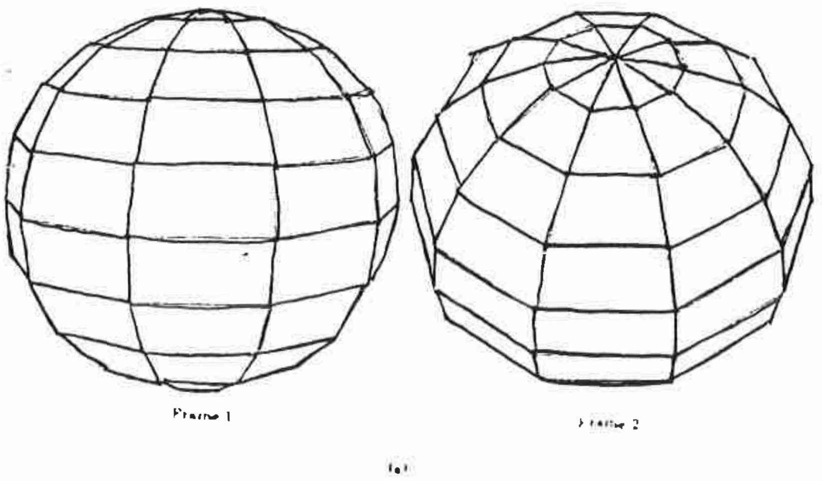
4.3 Scene visualization.

As has been explained in section 2 , rendering papers related to computer animation are often not specific to this area of computer graphics and only deal with new techniques to render geometric models which are of special interest in Computer Animation [ReB85], or to improve the efficiency of hidden line/surface elimination [CHP79] or shading [FLM85].

4.3.1 Hidden surface algorithms.

Removing the hidden surfaces frame by frame involves a great amount of CPU time . In order to reduce these computations, the similarities of the successives frame (generally known as "frame by frame coherence ") could be taken into account. From one frame to the next, very few elements of the scene change and in particular most of the hidden surfaces remain the same, in such a manner that it is only an necessary to update them from one frame to the other.

Hubshman's master thesis from Mac-Gill University in 1982 (Hub82) constitutes the main work in this field. He presents an algorithm of hidden surface computation using frame by frame coherence in a static scene where only the viewing position is changing . Another important restriction of this algorithm is that it is valid only for closed, convex polyhedra that do not intersect. Hubshman's algorithm consists in updating in each frame the list of visible polyhedra and for each of the partly visible polygon its



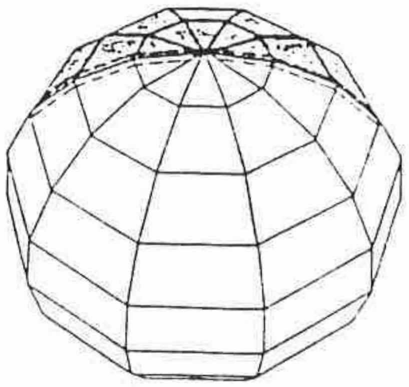
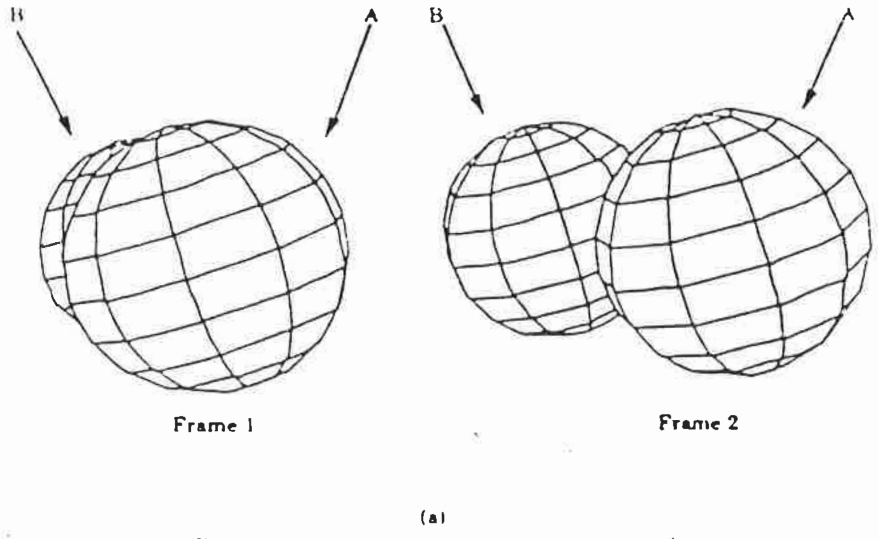


FIGURE 6.
Self-occlusion
of a polyedron



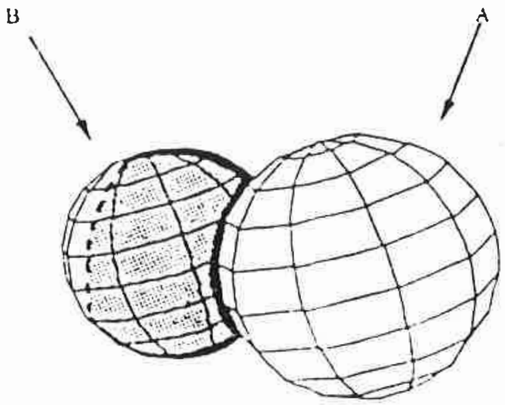


FIGURE 7.
Occlusion of a
polyhedron by
another

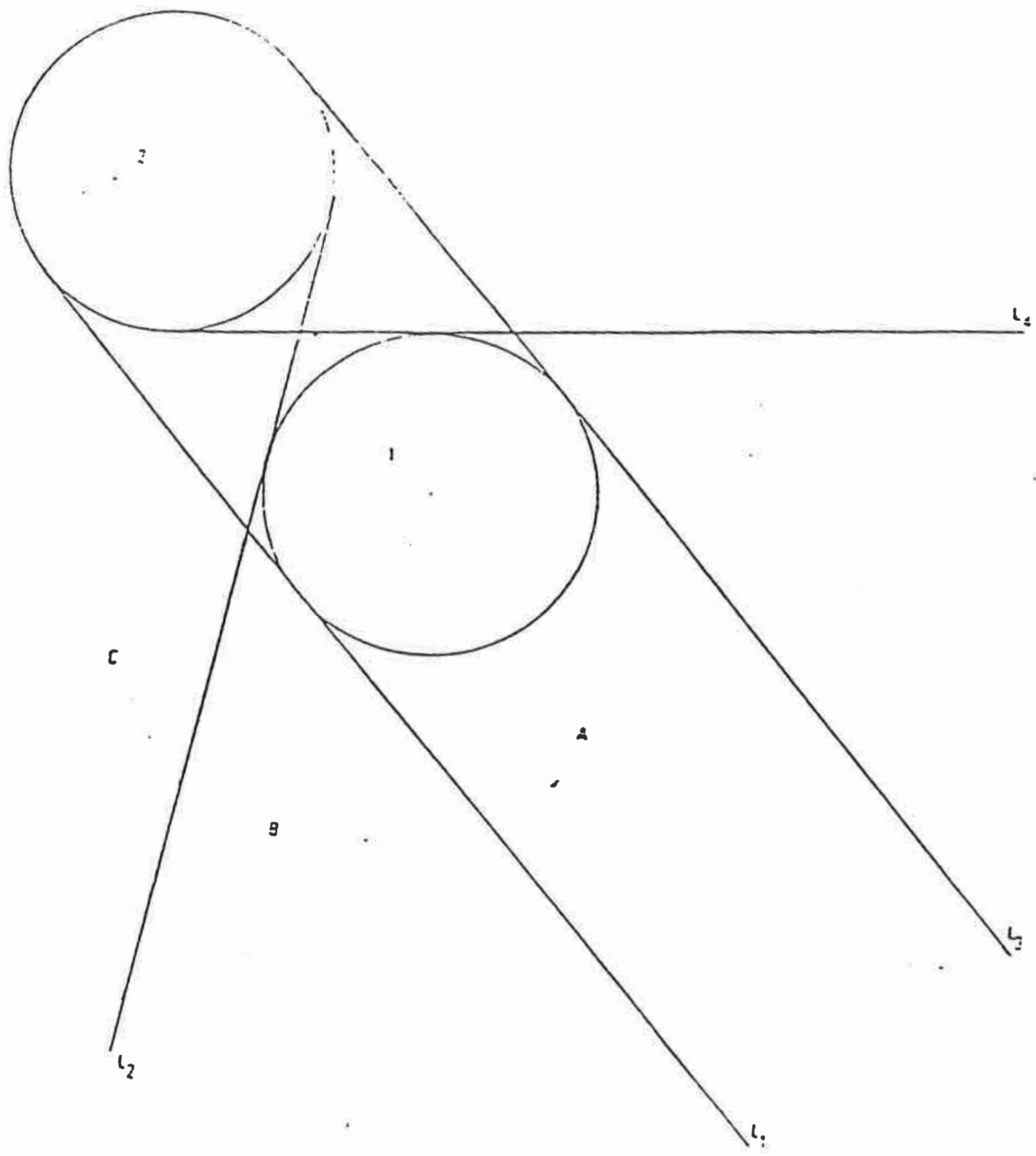


FIGURE 8. Space partition

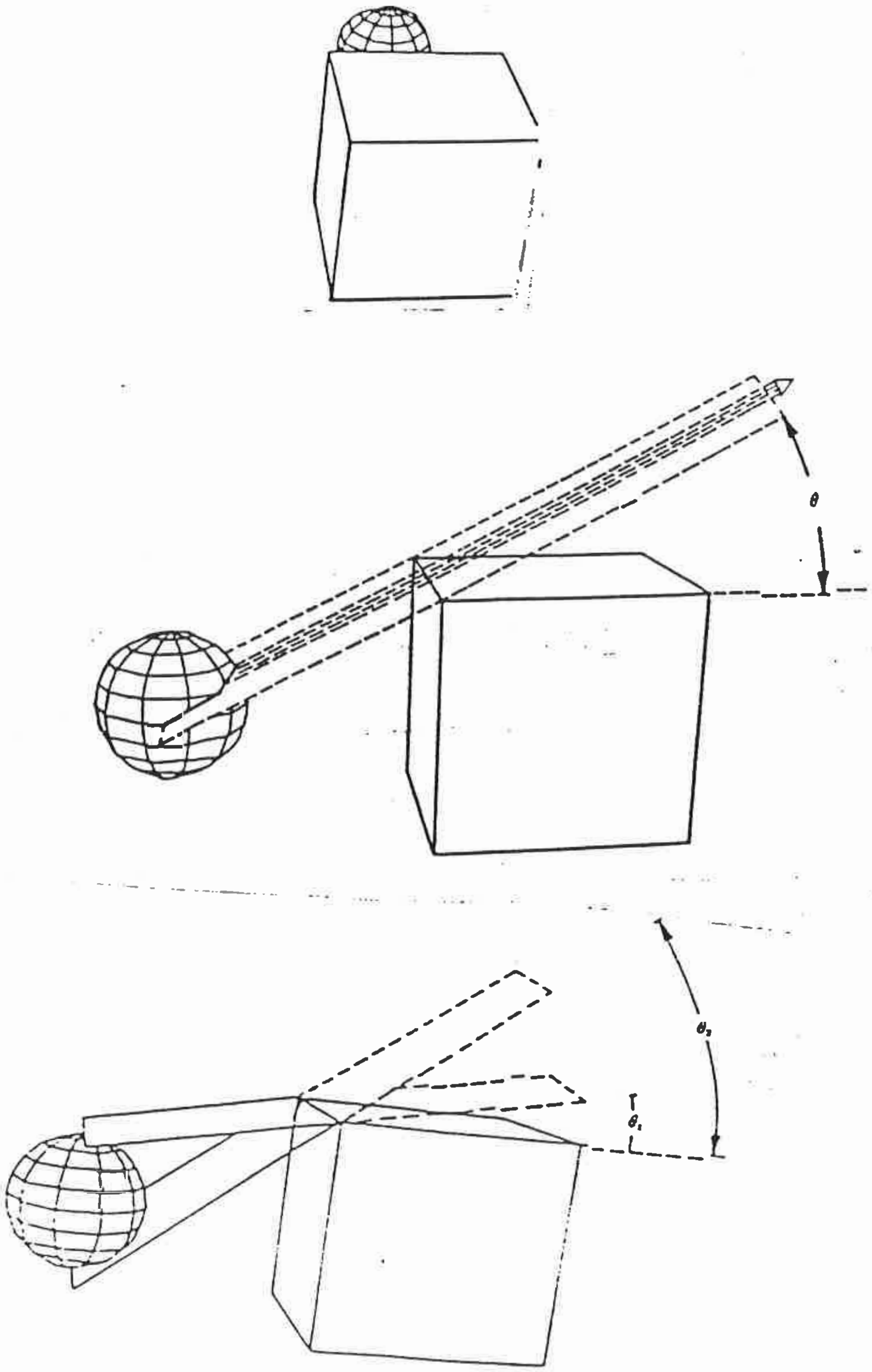


FIGURE 9. Viewing plane ray and support plane rays.

visible subpolygon edges. Two cases are considered : the self occlusion of a polyhedron (fig 6) and the occlusion of a polyhedron by another (fig 7). The new occlusions of polyhedrons can be deduced from a partition of the space (fig 8) and based on the geometrical information of the viewing plane ray and the support planes of the polyhedron (fig. 9).

The engineering animation system ANIMENGINE (NoK85) presents the opposite problem: solids are moving but the viewing point is static. In this case also limiting hypotheses are made, based on the real needs of an engineering system. The motions considered can only be parallel, which allows work in the 2-D projected plane (fig 10) . For each moving solid the volume swept by its motion is projected in a 2-D plane in order to determine which of these volumes intersects and which solid is occluding which other (fig 11).

Finally a third paper must be referenced in which frame by frame coherence is used (ShG82). In this key-frame based system, the path of the objects is determined by their position in each frame. The coherence of the path is used to reduce the computation of culling in each frame.

The complexity of these analyses and the strong restrictions to which they are subject indicate that research in this field in opened.

4.3.2 Shading

A few works has been published that deal on specific shading techniques for computer animation. Most of the research carried on this field focuses on temporal anti-aliasing [Par85b], [KoB83], [PoC83]. The problem is that pixels swept by a moving object are sometimes not activated because of the

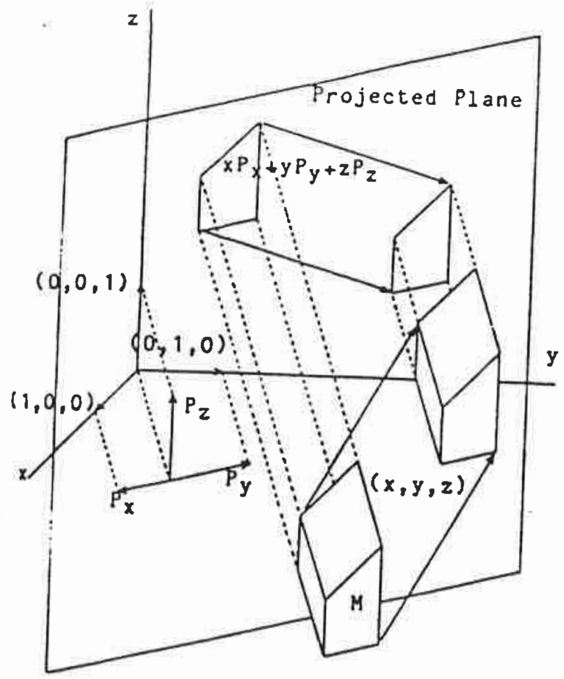


FIGURE 10.
3-D parallel motions in a projected plane

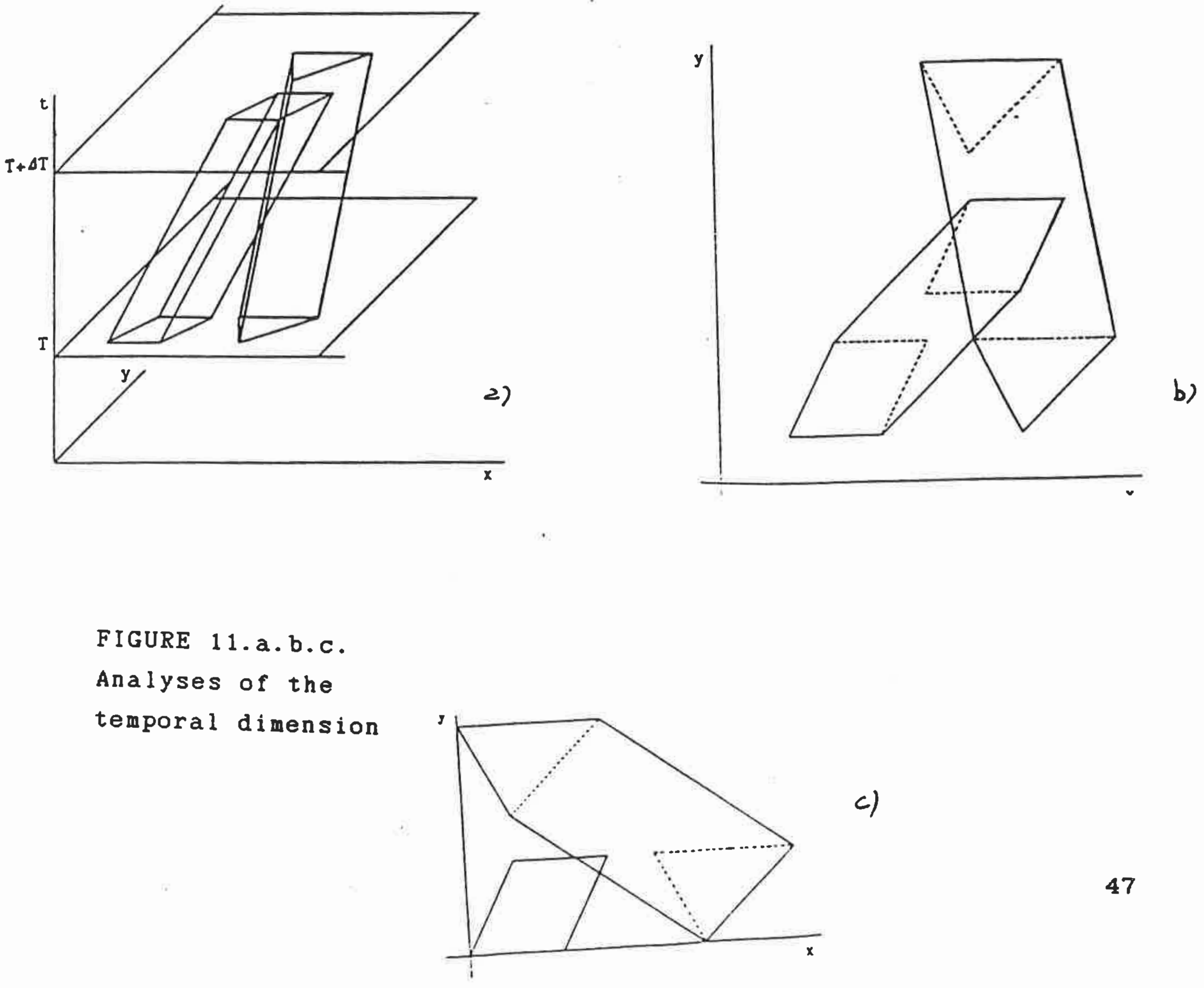


FIGURE 11.a.b.c.
Analyses of the temporal dimension

discretisation of motion in frames. This discontinuity is smoothed using techniques similar to those employed for spatial anti-aliasing. So different types of filtering strategies are used to activate additional pixels with less intensity than the pixels representing the actual object in the frame's location. These additional pixels give the illusion of motion for a static object filling the disconti-nuity between two consecutive frames.

On the other hand, the attributes of light sources may be animated as explained in section 4 [FLMT85], and camera attributes like color filters and focus may be simulated and animated [MaT86].

5. HUMAN BODY ANIMATION

Three-dimensional, realistic human body animation is one of the most challenging goals of computer animation. It is, however, a very complicated task: the human body is of irregular shape, difficult to model, and in addition, it involves more than 200 degrees of freedom (Zel82). This complexity is not, however, only in the design of the systems but also in their use: as human motion is rather intuitive, it is almost impossible for an animator to take into account all the links involved in a simple movement, because this coordination occurs in reality in an unconscious way (BRK80).

A way of avoiding these dificulties is the so-called "rotoscopy" (MaT85) which consists in filming real motions and drawing with the computer the two-dimensional pictures: the task of the animator is here rather one of cinematographic production. These are computer assisted animation systems and cannot be used for simulation, in robotics for instance which is one of the main field of application of human body animation. A technique derived from rotoscopy is the "scripting-by-enactment" of the MIT (Massachusettes Institute of Technology) (Bo81, GiM83) where the animator himself performs the actions he wants the graphic actors to perform. Its motion is tracked and parametrized : the principle is the same as for marionettes: the animator really performs the actions but hidden under another appearance.

In most of the applications, in any case, a model of the human body is needed. Nowadays , a realistic , user-friendly system modelling the human body has not been yet found in such a manner that the current systems bring partial solutions to the problem: simplifying the model of human body, reducing the approach to a kinematic one or limiting the dynamic analisis. Badler from the University of Pennsylvania, summarized at an early stage the principal characteristics that a human body animation system should have (BaS79) and in particular underlined the desirable capabilities from the point of view of the user:

" (...) 2. The body will be moved with an implicit respect of balance and support independent of a conscious effort to mantain these states by the user.
(...) 4. The dynamics and phrasing of a movement should be based on empirical evidence for human performance dynamics.."

It seems rather logical to think that the future of human body animation is related to artificial intelligence research and its aplication to the development of task level systems (Ze82) which could merge all these diferent approaches.

The first technique involved in human body movement modelling is not computer animation but choreography which needs a notation system of the human body and its movements in order to schedule the different phases of the ballet . Two different models of notation exist: the kinematic one of Laban (Labanotation : Hut70) and Eshkol (Esh58) and the dynamic one (effort/shape) (Del70).

Labanotation consists in representating human body as a set of links (fig.12.a) and in using a symbolic notation for the different actions that can be performed by the links :

- direction
- revolution
- orientation
- contact
- shape description

The Eshkol notation is almost similar but the body is modeled by joints associated with the longitudinal axis (fig.12.b)

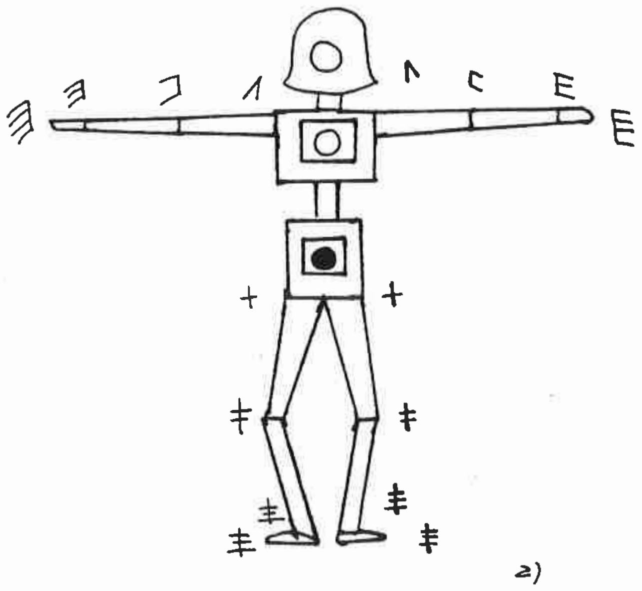
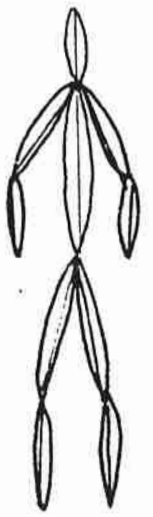



FIGURE 12. a) Laban representation of the human body
b) Eshkol representation.

The effort/shape notation instead tries to give a dynamic model of the movements by assigning to them qualities such as:

- tension
- weight
- time
- space....

The first notation technique is exclusively kinematic, and makes it difficult to model natural movements, whereas effort/shape notation is more flexible but needs a complex data structure and sophisticated algorithms.

There are three main models of the human body (BaS79) :

- stick figures
- surface models
- volume models : cylinders/ elipsoids/ spheres/parallilepipeds

In the first one (Wi70) the body is considered as a skeleton made up of joints and segments . The segments are considered rigid and capable at maximum of 6 degrees of freedom restricted by the attachment at the joints to neighbour segments. This model is related to the above mentioned technique of Labanotation. It presents the main disadvantage of producing unrealistic, schematic pictures.

The second one constructs the body as an assembly of polygons or surface patches. It deals with a high level of complexity that gives it a high computation cost. The links are specially difficult to represent because in a real human body, skin is elastic and can be deformed in the articulations (in the movement of opening and closing an arm for instance) whereas surfaces are not deformable.

Finally, the third one approaches the body as a collection of elemental primitives: spheres, ellipsoids... that don't present the problem of the links. The NUDES system of Herbison-Evans (He78) works with elipsoids. It is useful for modelling stylised human forms (in fact it was originally a coded notation of choreography) and offers an ingenious solution to the hidden surface problems by limiting them to the calculation of obscuration points and interpenetration points between two elipsoids.

The BUBBLEMAN (Ba80) modellizes human body at two levels: that of the skeleton and the envolvent surface composed by spheres and bubbles.

The main techniques for giving motion to a human body are (Ba79):

- key-frames using the skeleton technique described in section 3.
- kinematic: by giving positions and directions or goals of movement (KoB82)
- dynamic: controlling motion by torques and forces (AmG85,AGL86)

- constraint specification (Sp75).

All four are at present the subject of research, but the final objective should be to integrate all these kinds of motions into a unique system that could, in addition, take into account collision detections as explained by Badler (Ba86).

The main groups of research in this area are:
- The University of Pensylvannia (Badler O'Rourke, Smoliar, Korein), the main centre of development of the theoretical background of human body animation. They base their work on an extension and adaptation of "Labanotation" . They have also investigated the field of facial representation (BaP81)

- The Ohio State University in which the human body animation is related to the task level systems research.The human body is considered as a skeleton .

- The Simon Fraser University in which the research attempts to deal with both symbolic inputs of labanotation and analog inputs derived from electro-goniometers.

- MYIT creator of the graphical marionette explained above.

- The New-York Institute of Technology specialized in a particular problem of human body representation : facial animation (Pa82). They have developed a parametrized model of the face : where attributes such as color, dimension of nose, eyes.. and eyes expressions (opening, direction of vision..) are the parameters of a complex system of animation that deals with procedures, rotation and interpolations.

- In Canada three research groups exist currently: in Alberta working with the dynamic analysis of human animation (AGL86), in Vancouver with a kinematic approach through the use of hight-level structured programming and in Montreal (MaT85) at MIRA laboratory, where the research is centered on face modelling and kinematic movement modellization.

6. CONCLUSIONS

In our bibliographical study we conclude that there is a shortcoming in two areas: first, the small number of papers synthesizing the essential aspects of computer animation; and second, the incomplete level of description of most of the papers about computer animation systems.

Movement description is, obviously, the topic most widely dealt with in computer animation literature, given that it is a specific aspect of this computer graphics field. Other aspects (objects modelling, scene structure, rendering) are rarely dealt with, although computer animation production could be improved in different aspects by proposing new or modified geometric models as well as rendering algorithms. Current systems usually employ well known geometric models, rendering algorithms, interpolation methods, ... without justifying their choice.

The required features of computer animation systems are quite different for systems oriented towards entertainment production (e.g. publicity) and systems used in simulation environments (e.g. robots simulation). In the first case the main goal is realism of movements and images, whereas simulation requires precision and near real time animation. These differences affect essencially the features of the algorithms employed in the different steps of animation production.

From our study we can conclude that current and/or future research work may be summarized as follows:

- Task level systems will be improved by means of providing freindlier and less restrictive user interfaces. This research is related to the artificial intelligence field.

- Interpolation techniques (splines) should be more extensively used in motion control (key frame interpolation) and object shape modelling.

- In CAD/CAM systems, geometric solid models probably need to be adapted to animate them in order to simulate their movements.

- Finally, the use of frame-to-frame coherence will substantially improve the efficiency of hidden line/surface algorithms and shading as well as other rendering aspects like shadows.

BIBLIOGRAPHY

[AcW80] Real Time Animation Playback on a frame store display system .
ACKLAND,B; WESTE,N
Proceeding Siggraph '80

[AGL86] Near Real time Control of Human Figure Models
ARMSTRONG,W.; GREEN,M.; LAKE,M.
Graphic Interface '86

[Al184] Images de synthèses dans les simulations d'entrainement
ALLAIN,G.
Proc.CESTA 84, pp.643-649, 1984

[ArG85] Dynamics for animation of characters with deformable surfaces.
ARMSTRONG,W.; GREEN,M.
Proc. Graphics interface '85, pp.203-208, 1985.

[Ary84] A functional approach to picture manipulation
ARYA,K.
Computer graphics forum, vol. 3, pp. 35-46, 1984

[Bad82] Human body models and animation
BADLER,N.
IEEE computer graphics and applications 82

[Bad86] Animating Human Figure: perspectives and directions
BADLER,N.
Graphic interface '86

[Bae76] A conversational extensible system for the animation of shaded images
BAECKER,R.
Proc. Siggraph '76, pp. 32-39, 1976

[BaS79] Digital representations of human movement
BADLER,N ; SMOLIER
Computer survey (vol 11) March 1979

[BaR80] Special problems in human Movement Simulation
BADLER,N; O'ROURKE,J.
Proceeding Siggraph '80

[Bel85] Control of facial expressions and body movements the computer generated animated short "Tony de Peltrie"

P. BERGERON, P. LACHAPPELLE
Siggraph '85 Tutorial notes

[Ber85] Techniques for animating characters
BERGERON ,P.
Siggraph '85 : tutorial notes

[Bli82] Light reflection functions for simulation of clouds and dusty surfaces.
BLINN,J.
Proc. Siggraph '82, pp.21-29

[Bli85] Systems aspects of computer image synthesis and computer animation.
BLINN,J.
Tutorial Siggraph '85

[Blo85] Modeling the Mighty Maple
BLOOMENTHAL , J.
Proceeding Siggraph '85

[BrS84] A System for Algorithm Animation
BROWN,M; SEDGEWICK ,R.
Computer Graphics July 1984

[BTT84] Interactive Surgical Planning
BREWSTER,L.; TRIVEDI,S.; TUY,H; UDUPA,J.
IEEE C.A and D mar 1984

[BuW76] Systems aspects of computer image synthesis and computer animation.
BURTNIK,N; WEIN,M
COM. ACM,vol.19,pp. 174-185, 1976

[CaD84] Visualisation animée d'une propagation d'onde simulée par ordinateur.
CARNET,J; DUJARDIN,J.
Proc. CESTA 84, pp.215-220,may 1984

[CaS86] Modelling and animating 3-D Articulated Figures
CACHOLA,D; SHRACK ., G.
Graphic Interface '86

[Cat78] The problems of computer assisted animation
CATMULL,E
Proceeding Siggraph '78

[CCP80] The integration of subjective and objective data in the animation of Human Movement
T.W. CALVERT, J. CHAPMAN, A.PATLA
Proceeding Siggraph '80

[CCP82] Aspects of the kinematic simulation of human movement.
T.W. CALVERT, J. CHAPMAN, A.PATLA
IEEE Comp.Graphics and Applications, pp.41-50, 1982

[ChE83] A 3_D Shaded Computer Animation Step By Step
CHUANG,R.; ENTIS,G.
Comp.graphics, Proc.Intergrafics'83, pp.350-359

[CHP79] Towards an interactive High level Complexity Animation System .
CSURI,C; HACKAHORN,K; PARENT,R; CARLON,M; HOWARD,M
Proceeding Siggraph '79

[Coh85] Computer Animation
COHEN,E
Comp. animation tutorial of Siggraph 85

[Com84] Outil de production industrielle de déssin animé par ordinateur.
COMPARETTI,P.
Proc. CESTA 84, pp.191-193, apr.1984

[CoM84] De la visualiszation des resultats á la création artistique
COLONNA,J; MOWGLI
Proc. CESTA 84, pp. 201-213, may 1984

[Com86] The CGAL animation environment and its applications in the entertainment industry.
COMNINOS,P.P
Proc. CESTA 86, pp.324-332, apr 1986

[Coo85] Shade trees
R. COOK
Siggraph 85 1985

[CuS85] Data analysis through a generalised interactive computer animation method (DATICAM)
CURTIS,J.N; SCHEITZER,D.H.
ACM Computer and graphics vol.9 n.2 pp.153-157,1985

[DaP86] Modelling waves and surf
DARWIN; PEACHEY
Proceeding Siggraph '86

[DeM86] Une methode d' interpolation pour le cinéma d'animation
DEVAUX,M.;MARCE,P.
Proc. CESTA 86, pp.276-282, apr. 1986

[DeT86] A diferential compiler for C.A.

DENBER,M.; TURNER ,P.
Proceeding Siggraph '86

[Die85] Visual Intelligence: the first Decade of Computer Art (1965/ 1975)
DIETRICH ,F
IEEE CGA July 1985

[DoO81] Gramps: A graphic Language Interpreter for real time interactive 3-D picture editing and animating.
O'DONELL; OLSON
Proceeding Siggraph '81

[Dup84] Visualisation des resultats e simulations en mecanique des fluides.
DUFRE,F.
Proc. CESTA 84, pp.221-226, may 1984

[EHM84] Analyzing and comparing the performance of two real time play-back systems.
HEGAN,J; HART,J; MAC ELROY,R.D.
ACM Computer and graphics, vol 8, pp.67-79, 1984

[Fah85] Moving picture synthesis at Linkoping University
FAHLANDER,O.
Computer graphics Forum, vol.4, n.4, 1985

[Fan83] Extended memory use in the Z-grass graphics system
DE FANTI,T.
Comp.graphics, Proc. Intergraphics' 83 pp.380-386

[FFD78] Basic Zgrass- A sophisticated Graphics Language for the Bally Home Library computer
DE FANTI,T; FENTON,J;DONATO,N.
Proceeding Siggraph '78

[FLM85] Animating lights and shadows
FORTIN,M.; LEONARD,N; MAGNEMAT-THALMANN,N.; THALMANN, D.
Proc. Graphics Interface '85, pp.45-55, 1985

[FoR86] A simple model of ocean waves
Alain Fournier- Williams T Reeves
Proceeding Siggraph '86

[Fou86] GY10.000: Un système de géneration d'images synthétiques pour les simulateurs de chars et d'hélicoptères.
FOUCHÉ,Y.
Proc. CESTA 86, pp.832-837, apr.1986

[FoW83] Computer animation primer

FOX,D.; WAITE,M.
Mac Graw Hill

[Fox85] Do you itself : C.A.
FOX,D.
Siggraph '85 : tutorial notes

[FTL85] A multiple Track Animator System for Motion Synchronization
FORTIN,M; THALMANN,D; LAMY,R
Tutorial of Siggraph 1985

[FYZ85] Animated 3_D CT IMAGING
FARELL / YANG/ ZAPULLA
IEEE C.A and Design 1985

[GaL83] Techniques graphiques et CAO
GARDAN,Y; LUCAS ,M.
Hermes Publishing 1983

[Gar85] Visual simulation of clouds
GARDNER ,G.
Proceeding Siggraph '85

[GiM83] Graphical marionette
C. GINSBERG , D. MAXWELL
Proceedings Siggraph 1983

[GiV85] The paint Problem
GILOTH,C; VEEDER,J.
IEEE C.A and D jul. 1985

[Gom84] TWIXT: a 3-D Animation System
GOMEZ,J.
Computer and graphics, vol.9, n.3, pp.291-298, 1985

[GZD84] A tool set for 3-D Computer Animation
GOMEZ,J; ZELTZER,D; MC DOUGAL ,P
Tutorial notes : Siggraph 1984

[HaC86] Simulation graphique des mouvements d' un robot.
HADDAD,M.; COUESSON,T.
Proc. CESTA 86, pp.685-689,apr. 1986

[HaS85] Interactive Animation of parametric models
HANRAHAN,P; STURMAN,D
Computer animation tutorial of Siggraph 85

[HeE78] NUDES 2: a numeric utility displaying Ellipsoids Solids. Version 2
HERBISON,D; EVANS
Proceeding Siggraph '78

[Hub80] Frame to frame coherence and hidden surface computation: constraints for a convex world.
HUBSHMAN,H
Master Thesis Mac Gill University, Montreal, Canada November 1980

[HuZ82] Frame to frame coherence and hidden surface computation: constraints for a convex world.
HUBSHMAN,H; ZUCKER,S.W
ACM Trans. on graphics, vol.1, n.2, pp.129-162, april 1982

[Jah84] Application de la mecanique à l'animation interactive.
JAHIDI,K
Proc. CESTA 84, pp.663-669,may 1984

[JeA84] Graphic Simulation of planar dynamic mechanisms
JENNESS,J.; ANDREW,G.
ACM Computer and graphics vol.8, n.4, pp.411-421, 1984

[JiC85] Determination of 3_D Human Body Postures from a single view.
JIAN LEE,Z.; CHEN,Z
Computer Vision Graphics and Image Processing 1985

[KaH78] Dynamic Graphics Using quasi parallelism
KAHN,K; HEWITT,C
Proceeding Siggraph '78

[KoB82] Tecniques for generating the goal-directed motions of articulated pictures.
KOREIN,J; BADLER,N.
IEEE CGA November 1982

[KoB83] Temporal antialiasing in Computer Generated Animation
KOREIN,V; BADLER,N.
Proceeding Siggraph '83

[KoB85] Interpolating splines with local tensions continuity and biass control
KOCHANEK,K; BARTELS,B
Comp. Animation Tutorial of Siggraph 1985

[Kri84] Application of computer generated animation in European space research
E. KRISTIANSEN

Eurografics 84

[Lan85] The influence of the UNIX operating system on the development of video games
LANGSTON, P.
Siggraph: tutorial notes 1985

[Lan82] Computer assisted Animation
LANDDOWN,J.
Computer World 1982

[Lan85] Object and Movement Description Techniques for Animation: an informal Review
LANDSDOWN,J
NATO ASI series 1985

[LaP84] Couplage d'un systeme de simulation de robot avec un terminal de synthèse d'images
LAUGIER,C.; PERTIN-TROCCAZ,J.
Proc. CESTA 86, pp.704-716, apr.1986

[Lar85] Computer modelling of the animation process
LARREA,J.J.
Computer animation tutorial of Siggraph 1985

[LLM76] Synthese d'etude de l'animation par ordinateur
LUCAS; LESTY; MARTINEZ; GUYOT
Rapport de recherche 1976

[LoC85] The a_buffer : an antialiased hidden surface method
LOREN; CARPENTER
Comp. Animation Tutorial of Siggraph 85

[Luc72] Computer Driven Animation on a Graphic Display
LUCAS,M.
8eme seminaire DECUS EUROPE 1972

[Luc77] Bibliographie : Aide a la conception et realization de films
LUCAS,M.
Extrait de these 1977

[LuC84] Modélisation et animation gestuelle d' objects. Le systeme ANIM
LUCIANI,A; CADOZ,C.
Proc. CESTA 84, pp.183-189, may 1984

[LuC86] Utilization de modèles mécaniques et geomètriques pour la synthèse et le controle d'images animèes.
LUCIANI,A; CADOZ,C.
Proc. CESTA 86, pp.704-716, apr.1986

[MaB80] Optical printing in Computer Animation
MAX,N.; BLUNDEN,M
Proceeding Siggraph '80

[Mar77] Techniques de passage d'un dessin a un autre par formations successives: application a un systeme d'animation.
MARTINEZ,F
Rapport de recherche, université de Nantes 1977

[Mar84] La synthèse d'images. Concepts, materiels et logiciels.
MARTINEZ,F
Editests, Paris 1984

[Mar86] Architectures specialisées et technologie en synthèse d'image.
F.MARTINEZ
Proc. CESTA 84, may 1984

[MaS82] Graphics for dynamic analysis/synthesis of vibration systems with arbitrary degrees of freedom (DAVIS)
N.MATSUMOTO, S.SUSUKI
ACM Computer and graphics 82

[Mat72] Hidden Lines Elimination for a Rotating Object
MATSUSHITA,Y.
Com. ACM,vol.15, n.4, pp.245-252, april 1972

[MaT82] Some unusual primitives in the MIRA graphical extension of PASCAL .
MAGNEMAT-THALMANN,N.; THALMANN, D.
Computer and Graphics vol.6, n.3, pp. 127-139, 1982

[MaT83a] An indexed bibliography on computer animation
MAGNEMAT-THALMANN,N.; THALMANN, D.
IEEE- Computer Graphic and desing 1983

[MaT83b] Special Cinematographic Effects with Virtual Movies Cameras.
MAGNEMAT-THALMANN,N.; THALMANN, D.
IEEE- Computer Graphic and applications, 1983

[MaT83c] The use of High Level 3_D GRAPHICAL Types in the MIRA animation system
MAGNEMAT-THALMANN,N.; THALMANN, D.
IEEE- Computer Graphic and applications, 1983

[MaT83d] The use of 3-D abstract graphical types in computer graphics and animation
MAGNEMAT-THALMANN,N.; THALMANN, D.

Comp.graphics, Proc. Intergrafics' 83 pp.360-373

[MaT85a] Three-dimensional computer animation: more an evolution problem than a motion problem.
MAGNEMAT-THALMANN,N.; THALMANN, D.
IEEE C.A and D oct. 1985

[MaT85b] Subactor data types as Hierarchical Procedural models for computer animation
MAGNEMAT-THALMANN,N.; THALMANN, D.
Eurographics 85 1985

[MaT85c] MIRANIM : an extensible Director Oriented System for the animation of realistic images
MAGNEMAT-THALMANN,N.; THALMANN, D.
IEEE C.A and Design mar. 1985

[MaT85d] Controlling evolution and motion using the CINEMIRA animation sublanguage.
MAGNEMAT-THALMANN,N.; THALMANN, D.
Proc. Graphics Interface '85, pp.249-259, 1985

[MaT85e] Computer animation: theory and practice
MAGNEMAT-THALMANN,N.; THALMANN, D.
Springer Editions , 1985

[Mei85] Bucolic: A program for Teaching Color Theory to art Student
MEIER,B
IEEE C.A and D jul. 1985

[MMZ85] A language for animation of actors and objects: NEM
MARINO,G; MORASSO,P; ZACCARIA ,Z
Proc. Eurographics 1985

[Nic84] The IRIS Workstation
NICKEL,R
IEEE C.A and Design mar. 1984

[NoK85] ANIMENGINE: An Engineering Animation System
NOMA,T; KUNII,T
Proc. Graphics Interface '85, pp.189-202, 1985

[Op86] Real time design and animation of fractal plants and trees .
OPPENHEIMER,C.M.
Proceeding Siggraph '86

[PaK86] Une simulation informatisée des mouvements du T' ai chi ch'uan
PARAS KAUL,J.
Proc. CESTA 86, pp291-294 ,apr. 1986

[Pal85] Computer Art: Depolarization and Unification
PALYKA,D
IEEE C.A and D jul. 1985 NYIT

[Par80] Adaptation of Scan and Slit Scan Techniques to C.A.
PARKE,F.I.
Proceeding Siggraph '80

[Par82] Parametrized Models for facial expressions
PARKE,F.,I.
IEEE CGA November 82

[Par85] Rendering Issues in animation
PARKE,F.I.
Computer animation tutorial of Siggraph 1985

[PlB81] Animating Facial expressions
PLATT,S; BADLER ,N.
Proceeding Siggraph '81

[Ple86] Un modèle hierarchique pour la représentation de scènes tridimensionelles .
PLEMENOS,D.
Rap.recherche, Université de Nantes, oct. 1986

[Pot83] Modelling Motion Blur in Computer Generated Images
POTMESIL,M.
Proceeding Siggraph '83

[Ray85] Splines tutorials and C.A. : notes
RAY SMITH ,R
Tutorial notes : Siggraph 85

[ReB85] Approximate and probalistic algoritms for shading and rendering structured particles systems.
REEVES,W; BLAU,R.
Proceeding Siggraph '85

[Ree81] In Betweening for C.A. utilizing Moving points constraints.
REEVES,W.
Proceeding Siggraph '81

[Ree83] Particles Systems A technique for modelling a Class of fuzzy objects
REEVES,W.
Proceeding Siggraph '83

[Rey85a] Description and control of time and dynamics in
REYNOLDS,C
Comp. animation tutorial of Siggraph 1985

[Rey85b] Computer Animation with scripts and actors
REYNOLDS,C.
Comp. animation Tutorial of Siggraph 1985

[RoC86] MIAMI: terminal d'animation 2,5 D
ROUX,C; CLAIRON,J.
Proc. CESTA 86, abstract, apr.1986

[Ros85a] Advanced Computer Animation : Basic Principes of computer animation
ROSEBUSH,J.
Comp. animation Tutorial of Siggraph 1985

[Ros85b] 3-dimensional reconstruction. A case study of a perspective problem.
ROSEBUSH,J.
Comp.graphics, Proc.Intergraphics '83, pp.374-386.

[Ros86a] Japanese Computer Animation
ROSEBUSH,J.
Computer graphics World 1986

[Ros86b] Designing for Computer Animation
ROSENTHAL,C
Computer graphics World 1986

[RVB85] Anoted Bibliographie : C. A.
ROSEBUSH,J; VICKERS ; BERGERON
Siggraph 85, tutorial notes

[Sch84] Le mouvement en graphique
SCHWEPPE,M.
Proc. CESTA 84, pp.663-669,may 1984

[ShG82] Path specifiaction and path coherence
SCHELLEY,K.;GREENBERG,D
ACM Comp.graphics vol.16 n.3,pp.157-166, jul.1982

[Sho85] Animating rotation with quaternion curves
SHOEMAKE, K.
Proceeding Siggraph '85

[Ste79] Soft-Cell An aplication of Raster Scan graphics on conventional animation.
STERN,G.
Proceeding Siggraph '79

[Ste85] Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control.
STEKETEE
Proceeding Siggraph '85

[StF83] Raster display of rotating objects using parallel processing.
STROHOTTE,T.; FUNT,B
Computer graphics Forum, vol.2,pp. 209-217

[Sug83] A robust description of time varying scenes for computer animation.
SUGIHARA,K.

[TsD86] Goal Directed Animation using English Motion Commands
TSOTSOS; DREWERY
Graphic Interface 1986

[TuT84] Direct 2_D display of 3D Display
TUY,h; TAN TUY,L
IEEE C.A and D oct 1984

[UOT83] Collision detection in motion simulation
UCHIKI,T.;OHASHI,T.;TOKORO,M.
ACM Comp.graphics vol.7 n.3-4,pp.284-293, 1983

[Veg84] Computer graphics as an aid to a robot dynamical simulation
analysis.
VEGA PEREZ, R.M.
ACM Computer and graphics,1984

[Wag86] Computer Animation at NYIT
Patrick M. Wagner
Computer graphics World 1986

[Wal81] Merging and transformation of raster images for cartoon animation
WALLACE ,R.
Proceeding Siggraph '81

[Wat82] Computer Animation in Broadscating
WATKINS, R.
Computer world 1982

[Wil86] Virya a motion control Editor for Kinematic and dinamic animation.
WILHEMS,J.
Graphics Interface ' 86

[WeN85] Towards the derivation of 3D descriptions from moving non convex objects
WESTPHAL; NAGEL
Computer Vision Graphics and Image Processing 1985

[WiB85] Using Dynamic Analysis to animate articulated bodies such as humans and robots .
WILHEMS,J.; BARSKY,B.A.
Proc.Graphics Interface '85, pp.209-229, 1985

[WRD86] Imagery and graphics animation in chemistry.
WEBER,J; ROCH,M.; DOUCET,J.P.; DUBOIS,J.E
Proc. CESTA 86, pp.295-299 april 1986

[Zel82] Motor Control Techniques for figure animation
D. Zeltzer
IEEE CGA November 1982

[Zel85] Towards an integrated view of 3-D Computer Animation.
ZELTZER,D.
Proc. Graphics Interface '85pp.239-248, 1985

[ZSS78] A sensor simulation and animation system
ZIMMERLIN,J ;STANLEY,J; STONE,W
Proceeding Siggraph '78

THEMATIC BIBLIOGRAPHY

COMPUTER ANIMATION: GENERAL PAPERS

[AcW80] , [Ary84] , [Bli85] , [Coh85] , [FoW83] , [Fox85] , [GaL83] , [Lan85] , [Lar85] , [LLM76] , [Mar84] , [MaT83a] , [Mat85e] , [Rey85a] , [Ros85] , [Ros86a] , [Ros86b] , [RVB85] , [Sug83]

COMPUTER ASSISTED ANIMATION

[Cat78] , [DeM86] , [KoB85] , [Lan82] , [Loc85] , [Luc72] , [Luc77] , [Luc84] , [Mar77] , [Par85] , [Ray85] , [Ree81] [Sho85] , [Ste79] , [Ste85] , [Wal81]

MODELLED ANIMATION

[ChE83] , [CHP79] , [Com86] , [DeT86] , [DoO81] , [Fan83] , [FFD78] , [Gom84] , [GZD84] , [MaT82], [MaT83c] , [MaT83d] , [MaT85a] , [MaT85b] , [MaT85c] , [MaT85d], [MMZ85] , [NoK85] [Rey85b] , [RHC86] , [TsD86] , [Veg84] , [Wag86] , [Wil86] [Zel85] , [ZSS78]

HUMAN BODY ANIMATION

[AGL86] , [ArG85] , [Bad86] , [BaS79] , [BaR80] , [BeL85] , [Ber85] , [CaS86] , [CCP80] , [GiM83] , [HaS85] , [HeE78] , [KoB83] , [Par82] , [PlB81] , [WiB85] , [Zel82]

COMPUTER ANIMATION APPLICATIONS

[All84] , [Blo85] , [BrS84] , [BTT84] , [CaD84] , [Com84] , [CoM84] , [CuS85] , [Die85] , [Dup84] , [Fou86] , [FYZ85] , [GiV85] , [HaC86] , [Jah84] , [JeA84] , [Kri84] , [LaP84] , [LuC86] , [MaS82] , [Mei85] , [PaK86] , [Pal85] , [RoC86] , [Sch84] , [Wat82] , [WRD86]

RENDERING IN COMPUTER ANIMATION

[Coo85] , [DaP86] , [Fah85] , [FLM85] , [FoR86] , [Gar85] , [Hu80] , [HuZ82] , [KoB83] , [Mat72] , [MaT83b] , [Opp86] , [Ple86] , [Pot83] , [ReB85] , [Ree83]

POST-PROCESSING IN COMPUTER ANIMATION

[EHM84] , [FTL85]

OTHERS

[JiC85] , [KaH78] , [Lan85] , [MaB80] , [Mar86] , [Nic84] [NoK85] , [Par80] , [Ros85b] , [StF83] , [TuT84] , [WeN85]