MOTION CONTROL: FROM KEYFRAME TO TASK-LEVEL ANIMATION

D. Thalmann

1. Introduction

1.1 Computer animation and simulation

We know from Computer Graphics courses that 2D and 3D graphical objects may be constructed using geometric modeling techniques. In a 3D space, scenes are viewed using synthetic or virtual cameras and they may be lighted by synthetic light sources.

These techniques are important because they allow to visualize any geometrical, physical or chemical situation to be visualized **at any given time**. However, the most interesting aspect of many phenomena is their evolution over time; e.g. motion of electromechanical devices (robots), chemical reactions, fluid motion, cloud motions, heat conduction. Experiments are often very expensive and sometimes impossible; e.g. crashes, explosions. It is generally easier and less expensive to produce computer simulation of phenomena.

Computer graphical simulation is based on animation techniques.

Computer **animation** consists of modifying a scene over time. Consider for example a 3D scene; we may say that it is composed of three types of entities: objects, cameras and lights.

Each entity has characteristics which may evolve over time according to arbitrary complex laws:

1) for objects:

- location (car)
- orientation (robot arm)
- size (plant evolution)
- shape (cloud, human heart)
- color (fire, sunrise)
- transparency (fog simulation)

2) for cameras:

- viewer position (flight simulator)
- interest point
- view angle (zoom in).

3) for light sources

- intensity
- location (car light simulation).

1.2 Real-time vs Frame-by-frame

Real-time computer animation is limited by the capabilities of the computer. A real-time image must be displayed in less than 1/15 second, because the illusion of continuous movement breaks down at slower speeds. This is a severe limitation, because only relatively simple calculations can be made at this time. Another mode of production is frame-by-frame. Frames are calculated, recorded and then projected at a rate of 24 (25 or 30) frames/second. The calculation of one frame may take a few seconds or several hours for very complex images.

We give an example: we move a car 100 meters along the x-axis in 5 seconds; the car is located at $\langle 5, 0 \rangle$. We assume a sequence in 24 frames per second, which gives 120 frames for 5 seconds.

The following program produces such a animated sequence:

in real time:

```
STEPX:=100 / 120;
create CAR;
place CAR ( $\langle 5, 0 \rangle$ );
draw CAR;
for IMAGE:=1 to 120
    wait;
    erase CAR;
    translate CAR ( $\langle \text{PASX}, 0 \rangle$ );
    draw CAR;
```

frame by frame:

```
STEPX:=100 / 120;
create CAR;
place CAR ( $\langle 5, 0 \rangle$ );
draw CAR;
for IMAGE:=1 to 120
    record the frame;
    wait;
    erase CAR;
    translate CAR ( $\langle \text{PASX}, 0 \rangle$ );
    draw CAR;
```

In the near future, very complex animation will be produced in a very short time, due to the research in parallel processing and multiprocessors. Image synthesis algorithms like ray-tracing and scan-line may be easily distributed between several processors. Moreover, animation may be considered as a set of parallel processes. with and without communications between them.

2. A classification of computer animation methods

Most authors (Hanrahan and Sturman 1985; Parke 1982; Magnenat-Thalmann and Thalmann 1985; Steketee and Badler 1985; Zeltzer 1985) distinguish between three types of three-dimensional computer animation: image-based key-frame animation, parametric keyframe animation and algorithmic animation.

2.1 Image-based keyframe animation

Keyframe animation consists of the automatic generation of intermediate frames, called inbetweens, based on a set of key-frames supplied by the animator. In image-based keyframe animation, the inbetweens are obtained by interpolating the keyframe images themselves. This is an old technique, introduced by Burtnyk and Wein (1971). Fig.1 shows the principles to create inbetween frames by linear interpolation between corresponding vertices. When corresponding images have not the same number of vertices, it is necessary to add extra vertices, as shown in Fig. 2. A linear interpolation algorithm produces undesirable effects such as lack of smoothness in motion, discontinuities in the speed of motion and distortions in rotations, as shown in Fig. 3. Alternate methods have been proposed by Baecker (1969), Burtnyk and Wein (1976), Reeves (1981). According to Steketee and Badler (1985), there is no totally satisfactory solution to the deviations between the interpolated image and the object being modeled.

This method may be extended to three-dimensional objects. The principle is the same when objects are modeled in wire-frame. However, the technique is much more complex when objects are facet-based, because a correspondence between facets and between vertices must be found. Vertices and facets must be added in order to have the same numbers for both objects. A complete algorithm has been introduced by Hong, Magnenat-Thalmann and Thalmann (1988). Plate 1 shows an example.

Fig.1 Linear interpolation

Fig.2 Adding extra vertices before interpolation

Fig.3 In this example, linear interpolation produces undesirable reduction of the arm length

2.2 Parametric keyframe animation

Parametric key-frame animation is based on the following principle: an entity (object, camera, light) is characterized by parameters. The animator creates keyframes by specifying the appropriate set of parameter values at given time, parameters are then interpolated and images are finally individually constructed from the interpolated parameters.

We now give an example: the joint of a robotic arm is characterized by an angle varying during time t ; the following values have been selected:

$$t=0 \quad =10 \quad \quad t=2 \quad =20 \quad \quad t=5 \quad =45 \quad \quad t=8 \quad =100$$

The value of the angle every $\frac{1}{30}$ second, may be calculated by linear interpolation: e.g. for $t = \frac{1}{30}$, we have $= 10 + \frac{20-10}{2 \cdot 30} = 10.1666\dots$

However as shown in Fig. 4, there is a first-derivative continuity problem: values of angle for time $t=2-\frac{1}{30}$, $t=2$ and $t=2+\frac{1}{30}$ are respectively: $= 20 - \frac{20-10}{2 \cdot 30} = 19.81333$; $= 20$, $= 20 + \frac{45-20}{3 \cdot 30} = 20.2777...$ We observe a discontinuity around the value 2. In summary, linear interpolation causes first-derivative discontinuities, causing discontinuities in speed and consequently jerky animation. The use of high-level interpolation such as cubic interpolation or spline interpolation is preferable as shown in Fig. 5.

A good method is the Kochanek-Bartels spline interpolation (Bartels and Kochanek 1984) because it allows the curve to be controlled at each given point by three parameters: tension, continuity and bias. A time value should be added to each control point to control the motion. The method is valid for interpolation between scalar values like angles and vector values like positions.

To explain the method, consider a list of points $\mathbf{P}_i$ and the parameter t along the spline to be determined. The point $\mathbf{V}$ is obtained from each value of t from only the two nearest given points along the curve (one behind $\mathbf{P}_i$, one in front of $\mathbf{P}_{i+1}$). But, the tangent vectors $\mathbf{D}_i$ and $\mathbf{D}_{i+1}$ at these two points are also necessary. This means that, we have:

$$\mathbf{V} = \mathbf{T}\mathbf{H}\mathbf{C}^T \quad (1)$$

where $\mathbf{T}$ is the matrix $[t^3 \ t^2 \ t \ 1]$, $\mathbf{H}$ is the Hermit matrix, and $\mathbf{C}$ is the matrix $[\mathbf{P}_i, \mathbf{P}_{i+1}, \mathbf{D}_i, \mathbf{D}_{i+1}]$.

Fig.4 Linear interpolation of the angle

Fig.5 Spline interpolation

Kochanek and Bartels start from the cardinal spline:

$$\mathbf{D}_i = 0.5 (\mathbf{P}_{i+1} - \mathbf{P}_{i-1}) = 0.5 [(\mathbf{P}_{i+1} - \mathbf{P}_i) + (\mathbf{P}_i - \mathbf{P}_{i-1})] \quad (3)$$

This equation shows that the tangent vector is the average of the source chord $\mathbf{P}_i - \mathbf{P}_{i-1}$ and the destination chord $\mathbf{P}_{i+1} - \mathbf{P}_i$. Similarly, the source derivative (tangent vector) $\mathbf{DS}_i$ and the destination derivative (tangent vector) $\mathbf{DD}_i$ may be considered at any point $\mathbf{P}_i$.

Using these derivatives, Kochanek and Bartels propose the use of three parameters to control the splines—**tension**, **continuity**, and **bias**.

The tension parameter t controls how sharply the curve bends at a point $\mathbf{P}_i$; the parameter c controls the continuity of the spline at a point $\mathbf{P}_i$ and the direction of the path as it passes through a point $\mathbf{P}_i$ is controlled by the bias parameter b .

Equations combining the three parameters may be obtained:

$$\mathbf{DS}_i = 0.5 [(1-t)(1+c)(1-b) (\mathbf{P}_{i+1} - \mathbf{P}_i) + (1-t)(1-c)(1+b) (\mathbf{P}_i - \mathbf{P}_{i-1})] \quad (4)$$

$$\mathbf{DD}_i = 0.5 [(1-t)(1-c)(1-b) (\mathbf{P}_{i+1} - \mathbf{P}_i) + (1-t)(1+c)(1+b) (\mathbf{P}_i - \mathbf{P}_{i-1})] \quad (5)$$

A spline is then generated using Eq. (1) with $\mathbf{DD}_i$ and $\mathbf{DS}_{i+1}$ instead of $\mathbf{D}_i$ and $\mathbf{D}_{i+1}$.

2.3 Algorithmic animation

In this kind of animation, motion is algorithmically described by a list of transformations (rotations, translations etc.). Each transformation is defined by parameters (e.g. an angle in a rotation). These parameters may change during the animation according to any physical law. These laws may be defined using an analytical form or using a complex process such as the solution of differential equations. Control of these laws may be given by programming as in ASAS (Reynolds 1982) and MIRA (Magenat-Thalmann and Thalmann 1983) or using an interactive director-oriented approach as in the MIRANIM (Magenat-Thalmann et al 1985) system. With such an approach, any kind of law may be applied to the parameters. For example, the variation of a joint angle may be controlled by kinematic laws as well as dynamic laws. This latter approach has been recently introduced by several authors (Armstrong and

Green 1985; Wilhelms and Barsky 1985; Girard and Maciejewski 1985), but only for simplified and rigid articulated bodies with few joints, geometrical bodies (cylinders) and without any deformation. Only very short sequences have been produced, because of the lack of complete specification for complex motions and because of the CPU time required for certain methods. Moreover, although dynamics-based motions are more realistic, they are too regular, because they do not take into account the personality of the characters. It is unrealistic to think that only the physical characteristics of two people carrying out the same actions make these characters different for any observer. Behavior and personality of the human beings are also an essential cause of the observable differences.

For the creation of natural human motions, it is essential to take into account geometry, physics and behavior. Only a system based on these three factors may really work. However, this implies mechanisms for data acquisition, animation and image synthesis for each of these factors.

As an example of algorithmic animation, consider the case of a clock based on the pendulum law:

$$= A \sin (\omega t + \phi)$$

A typical animation sequence may be produced using a program such as:

```
create CLOCK (...);
for FRAME:=1 to NB_FRAMES
    TIME:=TIME+1/24;
    ANGLE:=A*SIN (OMEGA*TIME+PHI);
    MODIFY (CLOCK, ANGLE);
    draw CLOCK;
    record CLOCK
    erase CLOCK
```

2.4 An unified view

The three types of animation may be described in a more general and unified way. **An animated object (actor) is characterized by a set of state variables that drive the motion of the actors. The evolution of the state variables is defined by an evolution law.** The three types of animation may be redefined using the following terminology:

A) image-based keyframe animation: actors are characterized by their vertices; motion is specified by giving keyframes. Each keyframe consists of a series of values corresponding to the vertices for that keyframe. Inbetweens are calculated by applying an interpolation law for each corresponding vertex in the two keyframes. The interpolation law may be a linear law, a cubic law or a spline interpolation law.

state variables: vertices

evolution law: interpolation law (e.g. linear interpolation, Reeves interpolation)

B) parametric keyframe animation: actors are characterized by parameters; motion is specified by giving key values for each parameter. Inbetween values are calculated using an interpolation law such as a cubic law or a spline interpolation law.

state variables: parameters

evolution law: interpolation law (e.g. linear law, spline interpolation)

C) algorithmic animation: actors are objects with a motion defined by a list of transformations (rotations, translations etc.). Each transformation is defined by parameters (e.g. an angle in a rotation). These parameters may change during the animation according to any physical law. These laws may be defined using an analytical form or using a complex process such as the solution of differential equations.

state variables: parameters of the transformations

evolution laws: any physical law.

2.5 A comparison of methods

The various methods have advantages and disadvantages and may be compared using several criteria as shown in Table 1.

From this table, it is clear that no one model is superior to all the others. In particular, methods which are efficient (low CPU time) do not provide very realistic animation except when human intervention is very important (e.g. shape interpolation with many key-frames).

Table 1 A comparative table of animation methods

	animation quality	CPU time	human intervention	versatility	source of difficulty
shape interpolation	depends on the number of key-frames the type of interpolation	depends on the number of points and	very long lack of creativity	very bad a complex interpolation law	often unrealistic except with many key-frames or
parametric interpolation	depends on the number of key-values	depends on the number of parameters	shorter more creative	better	to find the best parameters
kinematic algorithmic animation	depends on the laws, but often unrealistic	depends on the laws, but not very expensive	may be difficult depends on the human interface	very good	realistic laws are not so easy to find
dynamic algorithmic animation	very realistic	very expensive	may be limited	good	complete dynamics-based models are too expensive for large sequences

3. Motion control of synthetic actors

One of the main challenges for the next few years is the development of an integrated animation system for the creation and animation of three-dimensional scenes involving human beings conscious of their environment. Such a system should be based on an interdisciplinary approach and integrate aspects and methods from animation, mechanics, robotics, physiology, psychology, and artificial intelligence. The system should achieve the following objectives:

- automatically produce computer-generated human beings with natural behavior
- improve the complexity and the realism of motion; realism of motion needs to be improved not only from the joint point-of-view as for robots, but also in relation to the deformations of bodies, hands and faces during animation.
- reduce the complexity of motion description

In future animation systems, based on synthetic actors, motion control is automatically performed using A.I. and robotics techniques. In particular, motion is planned at a task level and computed using physical laws. The simplest automatic control of motion is based on inverse kinematics. The problem involves the determination of the joint variables given the position and the orientation of the end of the manipulator with respect to the reference coordinate system. This is the key problem, because independent variables in a synthetic actor are joint variables. In a typical system based on inverse kinematics, the animator specifies discrete positions and motions for end parts; then the system computes the necessary joint angles

and orientations for other parts of the body to put the specified parts in the desired positions and through the desired motions. Such an approach works well for simple linkages. However, the inverse kinematic solutions to a particular position become numerous and complicated, when the number of linkages increases. Let us have an example, it is not difficult to determine how much to bend an elbow and a wrist to reach an object with the hand. It is much more difficult if we bring into play the rotation of the shoulder and the flexion of fingers. There are too many possibilities and the animator must supply more information. But which is the more natural solution and how to specify this choice ? Plate 2 shows an example of motion calculated using inverse kinematics.

A more complex, but more realistic approach is based on dynamics. The motion of a synthetic actor is governed by forces and torques applied to limbs. Two problems may be considered: the direct-dynamics problem and the inverse-dynamics problem. The direct-dynamics problem consists of finding the trajectories of some point as the end effector **with regard to the forces or the torques that cause the motion**. The inverse-dynamics problem is much more useful and may be stated as follows: given a trajectory as well as the forces to be exerted at the manipulator tip, find the torques to be exerted at the joints so as to move it in the desired manner. For a synthetic actor, it is possible to compute the time sequence of joint torques required to achieve the desired time sequence of positions, velocities and accelerations using various methods.

Techniques based on kinematics and dynamics have already been used in computer animation, but only for simplified and rigid articulated bodies with few joints, geometrical bodies (cylinders) and without any deformation. Only very short sequences have been produced, because of the lack of complete specification for complex motions and because of the CPU time required for certain methods. Moreover, although dynamics-based motions are more realistic, they are too regular, because they do not take into account the personality of the characters. It is unrealistic to think that only the physical characteristics of two people carrying out doing the same actions make these characters different for any observer. Behavior and personality of the human beings are also an essential cause of the observable differences.

4. Task-level animation

4.1 Robotics and animation

Similarly to a robot task-level system, actions in a task level animation system (Badler et al. 1985) are specified only by their effects on objects. In a robot task-level system (Lozano-Perez 1982), a task planner would transform the task-level specifications into manipulator-level specifications. In a task-level animation system, task-level commands should be transformed into low-level instructions such as a script for algorithmic animation or key values in a parametric keyframe approach. Zeltzer (1982) outlined one approach to task level animation in which motor behavior is generated by traversing a hierarchy of skills, represented as frames (Minski 1975) or actors (Hewitt 1977) in object-oriented systems.

Typical examples of tasks for synthetic actors are:

- walking from an arbitrary point A to another point B
- pick up an object at location A and move it to location B
- speak a sentence or make a high-level expression

Lozano-Perez divides task planning into three phases: world modelling, task specification and manipulator program synthesis. We shall use a similar philosophy to describe task planning for synthetic actors. We call the three phases: world modelling, task specification and code generation. It should be noted that essential differences exist between the robotics context and the animation context. These difference will be emphasized in the next sections.

4.2 World modelling

World modelling for a task consists mainly of describing the geometry and the physical characteristics of the synthetic actors and the object. The legal motions of the synthetic actors depend on constraints due to the presence of objects or other actors in the environment. The form of the constraints depends itself on the shape of the objects and the actors, which requires geometric descriptions of all elements. The most common way of modelling objects in the context of animation of synthetic actors is: facet-based representation, CSG or soft objects. What is also essential is that synthetic actors are generally based on deformable bodies.

Another important aspect in task planning is based on the limits of the primitive capabilities of the synthetic actor, e.g. joint limits.

Moreover physical characteristics should also to be taken into account;. Let us have an example: move a block B from a location X to a location Y. It is not sufficient to know the shape and the size of the block; it is necessary to know its mass, in order to generate the sequence of elementary movements to perform the task.

We should note that an attribute-based model is particularly suitable for task planning and implicit animation. In such a model, the scene has attributes, the objects and actors also have attributes and the animation is considered as a relation between attributes.

4.3 Task specification

There are three ways of specifying tasks in a task-level system:

1. by example
2. by a sequence of model states
3. by a sequence of commands

The specification by example means for the operator "to perform the task at least once in order to explain it to the system." This is suitable in robotics, because the task may be physically specified by manually guiding the robot. This is of course impracticable in animation.

In the second type of method, the task is considered as a sequence of model states; each state is given by the configuration of all the objects in the environment. The configuration may be described by a set of spatial relationships. But what is the level of these relationships ? High-level relationships correspond for example to indicating that at a given time an object A must be at a certain height and in front of another object B. The problem in this case is that the set of relationships should be converted into a set of equations and inequations which may be very difficult to solve. Moreover, a set of configurations may overspecify a state. Low-level relationships may correspond to the coordinates of the objects at a certain time, which is a simple keyframe description.

The specification by a sequence of commands is the most suitable and popular. As stated by Zeltzer (1985), the animator can only specify the broad outlines of a particular movement and the animation system fills in the details. A non-expert user may be satisfied with the default movements, as generated by a task specification like WALK FROM A TO B. However, a high-end user may want nearly total control over every nuance of an actor's movement to make a sequence as expressive as possible. This means that the animator does need to access different levels of the control hierarchy in order to generate new motor skills and to tweak the existing skills.

Most commands include a goal statement involving spatial relationships between objects.

We give an example in the AUTOPASS robotics language (Liebermann and Wesley 1977):

```
PLACE bracket IN fixture
      SUCH THAT bracket.bottom CONTACTS cartop
      AND bracket.hole IS ALIGNED WITH fixture.nest
```

The language proposed by Calvert and Chapman (1978) for dance should also be mentioned; for example, the command WALK may be used as follows:

```
WHEN (DancerB Touches) WALK VERY SLOWLY TO LEFT
      FRONT UNTIL (Edge of Stage)
```

4.4 Code generation

In robotics, the output of the synthesis phase is a program in a manipulator-level language; this consists of a sequence of low-level commands specific to a particular manipulator.

In computer animation, several kinds of output code are possible:

1. The complete animated sequence under the form of a series of frames ready to be recorded

2. The value of parameters (e.g. joint angles) for each frame; this allows the easy calculation of each frame in the case of a parametric keyframe animation system
3. The value of parameters (e.g. joint angles) for certain keyframes; this allows the easy calculation of each frame by parametric interpolation
4. A script in an animation language like ASAS (Reynolds 1982), MIRA (Magenat-Thalmann and Thalmann 1983)
5. A script in a command-driven animation system such as MIRANIM (Magenat-Thalmann et al. 1985)

Note that the transformation from a high level specification to a sequence of elementary motions is very similar to the problem of compiling. As in the processing of programming languages, three cases are possible: translation into a low-level code (classical compilers), translation into another programming language (preprocessor) and interpretation. This latter case has already been used by some authors. In particular, Zeltzer (1982) developed the Skeleton Animation System; in this system, a task manager accepts task descriptions and decomposes them into a list of component skills. Each skill represents some class of motions the figure can perform: walking, running, grasping and so on. Skills are implemented by procedures called motor programs, which invoke a set of primitive procedures, called local motor programs (LMP). An LMP can access and modify a fixed list of joints. The walk controller is implemented as a simple finite-state machine in which its four states represent pairs of LMPs to be executed concurrently.

In each case, the correspondence between the task specification and the motion to be generated is very complex. Consider three very essential tasks for a synthetic actor: walking, grasping and talking.

4.5 Walking

To generate the motion corresponding to the task "WALK from A to B", it is necessary to take into account the possible obstacles, the nature of the terrain and then evaluate the trajectories which consist of a sequence positions, velocities and accelerations. Given such a trajectory, as well as the forces to be exerted at end effectors, it is possible to determine the torques to be exerted at the joints by inverse dynamics and finally the values of joint angles may be derived for any time. In summary, the task-level system should integrate the following elements: obstacle avoidance, locomotion on rough terrains, trajectory planning, kinematics and dynamics.

For obstacle avoidance, the system should determine possible trajectories for the actor based on the environment. This obstacle avoidance, may be subdivided into two sub problems:

1. avoidance of static obstacles (decor) or objects grasped by actors
2. avoidance of dynamic obstacles

Problems are related to robotics, but also aesthetic criteria are very important in computer animation.

4.6 Grasping

This problem is well-known in robotics. The system determines joint angles from the position of the tip hand and the position of the object to be grasped. This is the classical problem of inverse-kinematics. Once the angles have been obtained from a kinematics point-of-view, the principle of inverse-dynamics may be used in order to obtain the data necessary to a dynamic process (forces and torques); this will allow us to solve motion equations and generate new motions.

To generate the motion corresponding to the task "PICK UP the object A and PUT it on the object B", the planner must choose where to grasp A so that no collisions will result when grasping or moving them. Then grasp configurations should be chosen so that the grasped object is stable in the hand (or at least seems to be stable); moreover contact between the hand and the object should be as natural as possible.

Once the object is grasped, the system should generate the motions that will achieve the desired goal of the operation. A free motion should be synthesized; during this motion the principal goal is to reach the destination without collision, which implies obstacle avoidance. In this complex process, joint evolution is determined by kinematics and dynamics equations. In summary, the task-level system should integrate the following elements: path planning, obstacle avoidance, stability and contact determination, kinematics and dynamics. Plate 3 shows an example of object grasping.

4.7 Talking

To generate the motion corresponding to the task "SAY THE SENTENCE How are you? ", the system must analyze the sentence and separate it into phonemes, and then facial expressions corresponding to these phonemes must be selected. These expressions are themselves expressed as face deformations caused by muscles: jaw opening, eye opening, face folds etc. Once the expressions have been selected, the system should indicate to the computer at which times the expressions must be activated and generate the frames according to a law (spline for example). In summary, the task-level system should integrate the following elements: phonemes detection, selection of facial expression selections, handling of facial parameters, animation generation.

4.8 The design of task specification languages

Let us have an example; we first introduce typical task commands:

```
walk to <location>  
put <object> on <object>  
pick up <object>  
sit down  
say <sentence>
```

We apply the following command to the synthetic actress Marilyn:

```
put GLASS on TABLE
```

Two cases are possible:

1. Marilyn is near the table and she has the glass in her hand; she only has to perform the required action. No other action is assumed.
2. Other actions are necessary to perform the required action. For example, Marilyn is sit down on a chair and the glass is located on the bar. The following sequence of actions has to be performed:

```
walk to BAR  
pick up GLASS  
walk to TABLE  
put glass on TABLE
```

Such a sequence is not too difficult to generate. We assume a knowledge database consisting of the description of the environment. For example, we have the following facts:

```
ABOVE(GLASS, BAR)  
SIT_DOWN ( MARILYN, CHAIR)
```

The system is also assumed to know the relative locations of the actress and each object.

However, the main problem in our example is that there is a great deal of uncertainty in the actions to be performed. For example, where should the glass be placed on the table ? What is the exact meaning of "walk to TABLE" ?

4.9 The use of symbolic spatial relationships

As already stated in the previous section, tasks may be defined as sequences of states of the world model. Each model state may be given by the configurations of all objects in the environment. Several methods for obtaining configuration constraints from symbolic spatial relationships have been proposed (Popplestone et al. 1980, Taylor 1976, Lozano-Perez 1976). We describe the method of Popplestone et al. by using an example, as described by Lozano-Perez. Fig. 6. shows two blocks BLOCK1 and BLOCK2.

Their legal configurations must satisfy the following relationship:

$$(F_3 \text{ against } F_1) \text{ and } (F_4 \text{ against } F_2) \quad (6)$$

A task command could be:

place BLOCK1 so $(F_3 \text{ against } F_1)$ and $(F_4 \text{ against } F_2)$

Such a relationship is quite naturally expressed, because each block has natural faces. But how can one express the exact action of grasping an object, when the hand and the object are modelled using a faceted representation ? The following formulation is possible, though it is not very user-friendly.

grasp BOTTLE so $(F_{47} \text{ against } F_{32})$ and $(F_{133} \text{ against } F_{76})$ and $(F_{198} \text{ against } F_{89})$ and $F_{214} \text{ against } F_{93})$ and $(F_{267} \text{ against } F_{104})$

Fig.6 Two blocks

where F_{47} , F_{133} , F_{198} , F_{214} , and F_{267} are assumed to be the facet numbers of the hand and F_{104} , F_{32} , F_{76} , F_{89} , F_{93} and F_{104} correspond to facets of the bottle.

This approach is similar to the method described by Thalmann and Magnenat-Thalmann (1986). An ideal approach would be a high-level specification such as:

grasp BOTTLE so FINGERS TOUCH NECK

or just

grasp BOTTLE by NECK

Configuration constraints are much more difficult to find when symbolic spatial relationships are of a very high level.

For relationships like (6), Popplestone et al. derive two equations:

$$\begin{aligned} \text{BLOCK1} &= F_3^{-1} M R_1^T \begin{matrix} T \\ <0,y_1,z_1> \end{matrix} F_1 \text{ BLOCK2} \\ \text{BLOCK1} &= F_4^{-1} M R_2^T \begin{matrix} T \\ <0,y_2,z_2> \end{matrix} F_2 \text{ BLOCK2} \end{aligned} \quad (7)$$

The name of an entity denotes its configuration. Positions (or configurations) of entities are expressed by 4x4 matrices. It should be noted that transformation matrices, when postmultiplied against the standard set of axes, produce the axes of the entity. M is the reflection matrix. R are matrices of rotations of around the x axis. T_v is the matrix for a translation v .

The above equations are two independent constraints on the configuration of BLOCK1 that must be satisfied simultaneously. Popplestone et al. show how to solve such equations and obtain:

$$y_1 = 0, y_2 = 1 \text{ and } z_1 = z_2$$

It shows (as well-known) that the position of BLOCK1 has one degree of freedom corresponding to translations along the z-axis.

The indetermination about z may be suppressed at the level of the task specification. For example:

place BLOCK1 so (F_3 against F_1) and (F_4 against F_2) by moving perpendicular to F_1

will generate a motion with a constant z.

Popplestone et al. propose a set of contact relationships called "against", "fits" and "coplanar" applied to planar facets, spheres, cylindrical shafts and holes, edges and vertices.

Acknowledgments

This research has been sponsored by the Fonds National Suisse pour la Recherche Scientifique.

References

- Armstrong WW and Green MW (1985) Dynamics for Animation of Characters with Deformable Surfaces in: N.Magnenat-Thalmann and D.Thalmann (Eds) Computer-generated Images, Springer, pp.209-229.
- Badler NI, Korein JD, Korein JU, Radack GM and Brotman LS (1985) Positioning and Animating Figures in a Task-oriented Environment, The Visual Computer, Vol.1, No4, pp.212-220.
- Baecker R (1969) Picture-driven Animation, Proc. AFIPS Spring Joint Comp. Conf., Vol.34, pp.273-288
- Burtnyk N, Wein M (1971) Computer-generated Key-frame Animation, Journal SMPTE, 80, pp.149-153.
- Burtnyk N, Wein M (1976) Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation, Comm. ACM, Vol.19, No10, pp.564-569.
- Calvert TW and Chapman J (1978) Notation of Movement with Computer Assistance, Proc. ACM Annual Conf., Vol.2, 1978, pp.731-736

- Girard M and Maciejewski AA (1985) Computational Modeling for Computer Generation of Legged Figures, Proc. SIGGRAPH '85, Computer Graphics, Vol. 19, No3, pp.263-270
- Hanrahan P and Sturman D (1985) Interactive Animation of Parametric Models, The Visual Computer, Vol.1, No4, pp.260-266.
- Hewitt CE (1977) Viewing Control Structures as Patterns of Passing Messages, Journal of Artificial Intelligence, Vol.8, No3, pp.323-364
- Hong T.M., R.Laperrière, D.Thalmann, A General Algorithm for 3-D Shape Interpolation in a Facet-Based Representation, Proc. Graphics Interface'88, Edmonton, 1988
- Kochanek D and Bartels R (1984) Interpolating Splines with Local Tension, Continuity and Bias Tension, Proc. SIGGRAPH '84, pp.33-41.
- Liebermann L and Wesley M (1977) AUTOPASS: An Automatic Programming System for Computer Controlled Mechanical Assembly, IBM Journal Research and Development, 21, 4
- Lozano-Perez (1976) The Design of a Mechanical Assembly System, Artificial Intelligence Laboratory, MIT, AI TR 397
- Lozano-Perez (1982) Task Planning in: Brady M (Ed.) Robot Motion: Planning and Control, MIT Press, Cambridge, Mass.
- Magenat-Thalmann and Thalmann D (1985) Computer Animation: Theory and Practice, Springer, Tokyo
- Magenat-Thalmann N, Thalmann D (1983) The Use of High Level Graphical Types in the MIRA Animation System, IEEE Computer Graphics and Applications, Vol. 3, No 9, pp. 9-16.
- Magenat-Thalmann N, Thalmann D, Fortin M (1985) MIRANIM: An Extensible Director-Oriented System for the Animation of Realistic Images, IEEE Computer Graphics and Applications, Vol.5, No 3, pp. 61-73.
- Minsky M (1975) A Framework for Representing Knowledge, in: The Psychology of Computer Vision, P.Winston, McGraw-Hill, NY, pp.211-277.
- Parke FI (1982) Parameterized Models for Facial Animation, IEEE Computer Graphics and Applications, Vol.2, No 9, pp.61-68
- Popplestone RJ, Ambler AP and Bellos IM (1980) An Interpreter for a Language for Describing Assemblies, Artificial Intelligence, Vol.14, pp.79-107
- Reeves WT (1981) Inbetweening for Computer Animation Utilizing Moving Point Constraints, Proc.SIGGRAPH '81, pp.263-269.
- Reynolds CW (1982) Computer Animation with Scripts and Actors, Proc. SIGGRAPH'82, pp.289-296.
- Steketee SN, Badler NI (1985) Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control, Proc. SIGGRAPH '85, pp. 255-262.
- Taylor RH (1976) The Synthesis of Manipulator Control Programs from Task-Level Specifications, Artificial Intelligence Laboratory, Stanford University, AIM-282
- Thalmann D and Magenat-Thalmann N (1986) Artificial Intelligence in Three-Dimensional Computer Animation Computer Animation, Computer Graphics Forum, Vol.5, pp.341-348
- Wilhelms J, Barsky B (1985) Using Dynamic Analysis to Animate Articulated Bodies such as Humans and Robots, in: N.Magenat-Thalmann, D.Thalmann (Eds) Computer-generated Images, Springer, pp.209-229.
- Zeltzer D (1982) Motor Control Techniques for Figure Animation, IEEE Computer Graphics and Applications, Vol.2, No9, pp.53-59.
- Zeltzer D (1985) Towards an Integrated View of 3D Computer Animation, The Visual Computer, Vol.1, No4, pp.249-259.

Plate 1: A 3D shape interpolation

Plate 2: The use of inverse kinematics

Plate 3. Object grasping

Daniel Thalmann is currently full Professor and Director of the Computer Graphics Laboratory at the Swiss Federal Institute of Technology in Lausanne, Switzerland. Since 1977, he was Professor at the University of Montreal and codirector of the MIRALab research laboratory. He received his diploma in nuclear physics and Ph.D in Computer Science from the University of Geneva. He is member of the editorial board of the Visual Computer and cochairs the EUROGRAPHICS Working Group on Computer Simulation and Animation. He was director of the Canadian Man-Machine Communications Society and is a member of the Computer Society of the IEEE, ACM, SIGGRAPH, and the Computer Graphics Society. Daniel Thalmann's research interests include 3D computer animation, image synthesis, and scientific visualization. He has published more than 60 papers in this areas and is coauthor of several books including: *Computer Animation: Theory and Practice* and *Image Synthesis: Theory and Practice*. He is also codirector of several computer-generated films: *Dream Flight*, *Eglantine*, *Rendez-vous à Montréal*, *Galaxy Sweetheart*.