Modeling and Animating Three-Dimensional Articulate Figures

Danny G. Cachola
Gunther F. Schrack

The University of British Columbia
Vancouver, B.C.

ABSTRACT

This paper describes a method for representing and animating three-dimensional articulate figures. It permits the definition of a model consisting of segments and joints, and the specification of the model's motion at a high level of abstraction by the use of a structured programming language.

RESUME

Cet article a pour but de décrire une méthode de représentation et d'animation de modèles articulés en trois dimensions. L'article propose d'une part, la définition d'un modèle de segments et d'articulations et d'autre part, la précision du movement de celui-ci à un niveau élevé d'abstraction en utilisant un langage structuré de programmation.

KEYWORDS: Computer animation, figure modelling, movement representation

1.0 INTRODUCTION

Many advances have been made in computer animation in the last few years, especially in the area of figure modelling and motion specification. Several methods have been proposed including the modelling and control of figures using procedures (procedural modelling) [7], the control of a physical model by the application of forces (dynamic modelling) [1], the use of goal-directed systems for the generation of a model's motion [5,10], and the use of key frame animation [3], one of the oldest animation techniques still in use.

A different approach for the modelling of a three-dimensional articulate figure and the subsequent control of its motions will be presented here. It permits a user to define a model (representing a real three-dimensional figure) consisting of segments and joints [11], and to specify the desired motion of its joints using a high-level structured programming language. The problem of figure modelling and motion specification is dealt with in terms of kinematics: the study of position (displacement) and its time derivatives (velocity and acceleration). Considerations of force and mass (dynamics) [4,9], balance [8], and obstacle avoidance [6] are beyond the scope of this discussion.

2.0 DESCRIPTION OF MODELS

Before an attempt is made to specify a desired motion for a model, a method for specifying the model must be available. The model's individual rigid links (segments) are unspecified in this study. It is assumed, however, that the model's links can be defined as graphical objects, using a high-level graphics language, before the model is constructed.

2.1 DESCRIPTION OF JOINTS

A joint has up to three degrees of freedom, that is, it can be rotated about each of the X, Y, and Z axes. Joints may be restricted to one or two degrees of freedom by permitting the joint to rotate about only one or two of the axes. Thus, simple joints such as fingers (hinge joints), and complex joints, such as shoulders (ball-and-socket joints) can be simulated. A joint connects only two links. A joint can move independently of all other joints, hence the position of one joint does not affect the motion of another. The links are restricted in their movements about a joint. During a single joint's movement, one link (the primary link), is considered stationary and the second link (the secondary link) moves with respect to the stationary link. A single link can function as both a primary link and a secondary link if it belongs to two or more different joints. One link, the model's main link, is singled out from the others. All movement ultimately refers to the main link. Only one link may be designated as the main link and it must be the primary link in all joints it belongs to.

Each instance of a joint is assigned a unique identifier to permit subsequent motion specifications. The user may place restrictions on the range of angles through which a link may travel and may specify where the two links are

to be joined. A typical statement creating an articulate joint is

```
JOINT joint_identifier,
      primary_link,    relative_location_1,
      secondary_link, relative_location_2,
      x_extremes, y_extremes,  z_extremes
```

where 'joint_identifier' is the unique identifier for this instance of a joint; 'primary_link' and 'secondary_link' may be either previously defined graphical objects (which have been defined in independent coordinate systems) or submodels. 'primary_link' is the stationary link with which 'secondary_link' moves. 'relative_location_1' and 'relative_location_2' are vectors which contain the relative positions, to each of the graphical object's coordinate system, where the joint is to be attached. The extreme parameters are component vectors which store a pair of extreme angles beyond which the joint can not move. Extremes, as well as the joint's current angles, are given relative to the joint's predefined neutral position of (0°, 0°, 0°).

2.2 INTERNAL REPRESENTATION

The model of an articulate figure is described by a tree structure of nodes and arcs. Links are represented by nodes and the joints are represented by arcs. Each segment is defined on its own local coordinate system [2,5,10]. The nodes of each level move with respect to the nodes of the higher level and are considered stationary by the nodes below. The nodes at the leaves of the tree represent the outermost extremities of the model; the root node is considered the main link. Figure 1a is an example of a partial model of a human figure. Figure 1b contains the model's tree structure. If a root node is no longer required to act as the main link, a new node can be assigned to that role by the statement

```
FIGURE model_name MAIN major_link_in_body
```

It is possible, therefore, to animate a model with a different main link in different scenes.

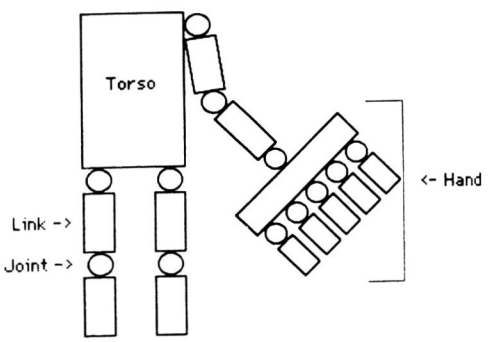

Figure 1a

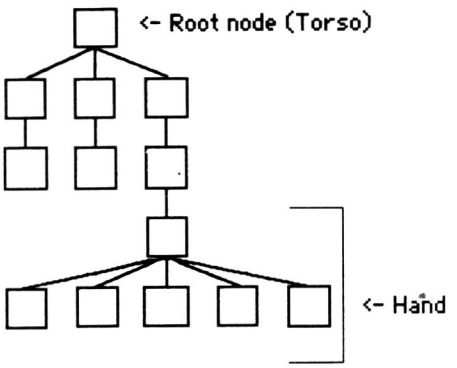

Figure 1b

a) Partial model of a human figure
b) Model's tree structure

Each node is associated with at least one arc, as in the case of the extremities which are secondary links. Typically, there are two arcs associated with each node, corresponding to a link (such as the femur of a leg) which acts as both a primary and secondary link. However, a node may have three or more arcs associated with it, as with the hand, which has six arcs (one representing the wrist joint, the other five representing the finger joints). A node may have at most n arcs associated with it. The branching factor n is theoretically unlimited, but a factor of 10 is deemed sufficient to define most articulate figures.

The tree structure permits the representation of open kinematic chains only. A kinematic chain is a linear sequence of links which are connected by joints. In an open chain, one end point is fixed and the remaining chain is allowed to move freely, as in Figure 2a. In a closed chain, more than one end point is fixed in space, as in Figure 2b. For example, if two hands are joined together and the arms are allowed to move while keeping the body motionless, a closed kinematic chain is formed by the arms. The motion produced in such a chain is more complex to analyze and is beyond the scope of this discussion.

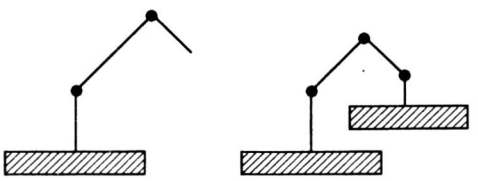

Figure 2a Figure 2b

a) Open kinematic chain
b) Closed kinematic chain

Each model has a table associated with it. The table contains information from the declaration of the model, in particular, the

identifier, the rotational extremes associated with each degree of freedom, the current rotational angles of each degree of freedom, and the instantaneous velocity and acceleration of the joints along each degree of freedom. The table describes the model completely at all times.

2.3 MODEL DEFINITION USING SUBMODELS

An articulate model can be defined using two basic techniques. The first allows the use of articulate submodels. Instead of simply using static graphical objects for each link, the secondary link can consist of a grouping of other links and joints, which permits the creation of intermediate models or submodels that can be referenced frequently. The statement

```
arm := JOINT 2, humerus,  (1.0, 5.0, 1.0),
               forearm,  (0.0, 0.5, 1.0),
               (0°,  135°), (0°, 180°)
```

creates a model of an arm consisting of a forearm and a backarm (humerus) joined at the elbow. The secondary link (forearm) is a model itself, consisting of links and joints. The relative location vector for the secondary link is given with respect to the main link in the model 'forearm'.

An advantage of this method is that symmetric models can be created with fewer statements. For example, creating a model of a human body first entails the creation of submodels for the right arm and the right leg. Once defined, each submodel can be duplicated and joined to both the right and left half of a human torso. A separate set of left limbs need not be defined. A problem arises here, however. The values of the joint identifiers should not be duplicated for the right hand and left hand limbs, or else the two sets of limbs will behave identically. Thus, the submodel joint identifiers must be modified before the submodels can be connected to the torso, to ensure unique identifiers. Consider the statement

```
arm := JOINT 2, humerus,  (1.0, 5.0, 1.0),
               forearm <TRANS BY 10>,
               (0.0,  0.5, 1.0),
               (0°,  135°), (0°, 180°)
```

which creates a model of an arm consisting of a forearm and backarm (humerus) joined at the elbow. The secondary link (forearm), which was previously defined, is an articulate model whose joint identifiers have been mapped onto a new range of identifiers starting with the identifier 10.

2.4 MODEL DEFINITION BY CONSTRUCT

The submodel approach may produce temporary models unnecessarily. For this reason, an alternate technique for creating models is provided which assumes that none of the model's subcomponents is required subsequently. This method creates only the resultant model with no intermediate articulate models. The following construct is an example

```
MODEL arm
    JOINT  1, palm, (0.0, -3.0, 1.0),
              little, (0.0, 1.5,  0.5),
              (-90°, 45°), (-45°,  45°)
    JOINT  2, palm, (0.0, -3.0, 2.0),
              ring, (0.0, 1.5,  0.5),
              (-90°, 45°), (-45°,  45°)
    JOINT  3, palm, (0.0, -3.0, 3.0),
              middle, (0.0, 1.5,  0.5),
              (-90°, 45°), (-45°,  45°)
    JOINT  4, palm, (0.0, -3.0, 4.0),
              index, (0.0, 1.5,  0.5),
              (-90°, 45°), (-45°,  45°)
    JOINT  5, palm, (0.0, -3.0, 5.0),
              thumb, (0.0, 1.5,  0.5),
              (-90°, 45°), (-45°,  45°)
    JOINT  6, ulna, (0.0, -5.0, 1.0),
              palm, (0.0, 3.0,  1.0),
              (-90°, 0°), (-25°,  25°)
    JOINT  7, humerus, (0.0, 5.0, 1.0),
              ulna, (0.0, 0.5,  1.0),
              (0°, 35°), (0°,  180°)
ENDMODEL
```

If the model 'hand' comprising the first five joints has been defined prior to the use of the MODEL construct, then the model definition can be simplified:

```
MODEL arm
    JOINT  6, ulna, (0.0, -5.0, 1.0),
              hand, (0.0, 3.0,  1.0),
              (-90°, 0°), (-25°,  25°)
    JOINT  7, humerus, (0.0, 5.0, 1.0),
              ulna, (0.0, 0.5,  1.0),
              (0°, 35°), (0°,  180°)
ENDMODEL
```

3.0 DESCRIPTION OF MOTION

A model can be animated by a number of different approaches. Conventional animation relies heavily on two-dimensional techniques such as key frames and interpolation ("in-betweening"), while computer animation usually expresses position and velocity as functions of time. Key frames are the frames used to provide the information which express the proper effects of movement. In animation studios, key frames are drawn by the head animators, while the frames required to create the smooth animation are produced by the in-betweeners.

One of the earlier approaches to computer animation consisted of having the computer assume the role of the in-betweeners [3]. While some very effective animations have been achieved, in-betweening is two-dimensional in origin and awkward to apply to three-dimensional figures. In-between frames are frequently linearly interpolated, resulting in temporal discontinuities and movements which only approximate actual trajectories, thus, deforming the animation.

Functions of time are evaluated on a frame-by-frame basis which involves specifying a path over time. This method has the advantage of producing motion with few temporal discontinuities. The description of a three-dimensional path as a function of time, however, is generally a difficult task.

3.1 MOTION SPECIFICATION

The proposed approach for motion specification treats motion somewhat differently than either the key frame or functional technique. Whereas key frame animation views a figure with an external perspective and the functional approach views a figure from a piecewise perspective, the proposed technique treats the model as a unit and views it from an internal perspective (i.e. from the model's point of view). Therefore, a model's positional orientation can be specified throughout time. Similarity exists with key frame animation since key frames or positional extremes are used throughout time, however, each of the model's joints is employed and dealt with on a high level of abstraction, thus allowing more flexibility in the model's animation.

The motion for a single joint is specified by the joint identifier, an initial starting position (angle), and a set of key frames (movements). Each movement contains the frame identifier at which the movement is completed, the position (angle) of the joint at the frame, and the interpolation method used to reach this positional extreme.

The position angles relate to a neutral position, arbitrarily defined as (0°, 0°, 0°). The neutral position is defined with respect to the primary link in the joint. The frame identifiers represent the number of ticks (units of time) which have passed during the motion sequence. The frame identifiers are given with respect to the start of the motion specification. The initial frame is arbitrarily assigned the value of zero. The interpolation techniques available are: linear, acceleration, deceleration, and a combination of both acceleration and deceleration.

The parameters above are sufficient to specify a single joint's movement throughout an animation sequence. Temporal discontinuity problems can arise if the interpolation method is not chosen carefully. The system can avoid these problems by determining a joint's instantaneous velocity and acceleration before the interpolation, thus making adjustments to the selected interpolation. This results in smaller temporal discontinuities.

A complete motion definition for a model consists of motion specifications for all of the joints present in the model. The specifications are independent of each other, therefore, there may be different numbers of key frames for each joint. All the joint key frames must, however, end at the same unit of time.

3.2 TREE TRAVERSAL

Before a model's tree structure is traversed, the symbol table is updated with each joint's current positional angle, and its current velocity and acceleration. This information is obtained from both the motion specification and the interpolation routines. The tree structure is completely traversed each time the model is displayed (i.e. once each frame). The traversal algorithm is a simple recursive post-order routine. It assembles each model's instance (from the extremities inwards) with respect to the main link in the model. The resulting instance is displayed by the high-level graphical language. Recursive routines are employed in the tree traversal because they allow storage of the the model's primary-secondary link relationship in the recursion stack. Also, they allow the use of the same routines on models whose main link (root) has been changed.

3.3 EXPLICIT DEFINITION

A motion can be specified using different approaches. There are two different methods for explicitly defining a motion. The first technique involves the use of a construct for the definition of a motion. A typical example is

```
MOTION motion_name
    JOINT  joint_id
         POSITION angle_x,  angle_y, angle_z
         FRAME frame_id
             POSITION angle_x, angle_y, angle_z
             INTERPOLATE interpolation_technique

         FRAME frame_id
             POSITION angle_x, angle_y, angle_z
             INTERPOLATE interpolation_technique
        . . .
    ENDJOINT

    JOINT  joint_id .....
         FRAME ...
         . . .
    ENDJOINT
    . . .
ENDMOTION
```

With the construct, the exact definition of the motion can be specified. 'motion_name' names the set of joint-motion specifications. The joint identifiers correspond to those defined in the model. Each joint can have at most n key frames, where n is a predetermined value. The construct permits a structured approach to producing a motion specification. It, however, does not provide for the use of other control constructs.

A less structured method is also available. A typical example of the definition technique is

```
motion_name [joint_id, key_frame_id] :=
    FRAME  frame_id
         POSITION angle_x,  angle_y, angle_z
         INTERPOLATE interpolation_technique
```

The approach allows direct access to the motion variable. It can be employed within other constructs and it permits the use of iteration to produce the motion specifications, thus reducing the number of statements needed and the amount of work needed to define the specifications.

3.4 IMPLICIT DEFINITION

Once a motion has been defined, the specification can be viewed as a unit (motion primitive), thus it can function as a building block for the definition of more complex motions. A primitive motion algebra has been introduced for this use. For example, to create a motion which enables a human model to hop, skip, and jump 10 times, the following statement can be used:

```
new_walk := 10 *  (hop + skip + jump)
```

'new_walk' is the resulting action. 'hop', 'skip', and 'jump' are previously defined motions which allow a model to hop, skip and jump respectively. The constant 10 is the repetition factor that is applied to the actions 'hop', 'skip', and 'jump'.

This technique relies heavily on the availability of predefined motion primitives. It is recognized that the explicit definition of such primitives can be both difficult and time consuming, therefore, operations have been introduced which allow a more convenient definition of the motion primitives. For example, if a motion primitive (walk) exists which allows a model to walk, it may be desirable to employ the action of the legs in a different motion. The operation STRIP has been introduced, which creates a partial motion primitive from a given motion. In the statement

```
walking_legs :=  STRIP walk, 5, 6, 7,  8, ...
```

'walking_legs' is a new motion primitive which when applied to a model, animates the legs. The values 5, 6, 7, 8, ..., are the joint identifiers present in the model's legs. Using this method, several motion primitives can be created that animate only portions of a given model.

The operation SYNCHRONIZE has been introduced, which when given a set of partial motions, creates a new motion that animates portions of the model concurrently. The statement

```
my_walk := SYNCHRONIZE walking_legs,
                       swinging_arms
```

defines a motion primitive (my_walk) which causes the figure to walk while swinging its arms. This approach allows an increase in the number of motion primitives, but at a much higher level of abstraction.

Such an approach to the creation of motion gives rise to a large number of operations which can tailor motion primitives to the needs of the current animation sequence. Possible operations include a scaling factor which stretches or shrinks a motion primitive temporally, a scaling factor which modifies the amplitude of a motion's movement, and a negation which reverses the order of the movements present in a motion primitive. These operations, as well as the operations * and +, have the advantage that intimate knowledge of the model's structure and motion definitions are not necessary. They allow the use of motion at a high level of abstraction.

3.5 USE OF A MOTION DEFINITION

Once a model is created and motions have been specified, a model can be animated.

```
ANIMATE model_name FROM start_frame TO  end_frame
      USING motion_name
```

The frame identifiers are given relative to the start of a scene. If the motion does not span the entire animation sequence, the motion will automatically repeat until the time span is covered.

Several models of a class (i.e. models of identical structure) may be animated by the same motion. Models of the same class and with different-sized links will react identically. Models with an equal number but with different types of joints can still be driven by these motion specifications, however, the resulting model movements may be difficult to predict. The models place restrictions on their movements. For example, if a rotation is employed which violates a joint's extremes, the joint will enforce its extreme rotation limits over the motion specification given. This allows two models of the same class, but different restrictions, to behave realistically using the same motion set.

Once all elements for a scene have been defined, it can be explicitly specified with static and dynamic components. A typical statement creating a scene is:

```
SCENE  scene_identifier
     DISPLAY background,  ...
     DISPLAY any  static models

     . . .

     ANIMATE model_name  FROM ....
     ANIMATE any  dynamic models

     . . .

     movements which  vary with time
     (camera movements,  panning,
      zooming,  etc.)

     . . .

ENDSCENE
```

The resulting scenes are displayed by the statement:

SHOOT SCENES

The scenes are displayed in sequential order ranging from the lowest scene identifier to the highest one. The use of scene constructs is advantageous because the animation sequence can be broken into a set of independent scenes, thus allowing its construction on a piecewise basis. The creation of independent scenes also permits changes in the scene ordering without any knowledge of the frame identifiers involved.

4.0 ENHANCEMENTS

The system is designed as an extension to a host language, to be executed in a batch environment. A preprocessor translates all extended language statements into procedure calls. Other implementations, e.g. as a command language to be interpreted by an executive kernel, are possible. The batch environment creates problems in the use of both predefined models and motions. At present, these definitions must be made at the beginning of a program before they can be employed. Ideally, there should be a method to save any models and motions defined in a program. Therefore, subsequent programs would need only load the definitions for models and motions before using them. This would allow the construction of a library of models and their movements.

A problem with the use of library definitions is that unless a user has defined a model and its motions, or has previously worked with them, it is difficult to determine how they will appear. It would be useful to have a viewing utility which permits the viewing of predefined models and motions in an interactive environment. This would allow the user to determine exactly what had been previously defined and what further definitions must be made in the program.

5.0 CONCLUSION

The project has several advantages over existing systems. It permits the modelling and animation of figures at a high level of abstractions. Where many systems have problems working with rotations, this system effectly deals with rotations. The use of rotations enables the creation of generalized motions which can be applied to more than one model, while motion specifications using paths and forces can only be used on the specific models for which they were designed. Interpolation currently works with articulate figures at the point level since it only has access to the two-dimensional projection of a figure. The approach presented deals with interpolation on a rotational basis (i.e. the rotational extremes correspond to the key frames in two-dimensional interpolation), thus allowing the animation of three-dimensional figures.

6.0 REFERENCES

[1] Armstrong, W.W. and M. Green, The Dynamics of Articulated Rigid Bodies for Purposes of Animation, Graphics Interface '85, 1985, pp. 407-415.

[2] Badler, N.I. and S.W. Smoliar, Digital Representations of Human Movement, ACM Computing Surveys, Vol. 11, No. 1, 1979, pp. 19-38.

[3] Burtnyk, N. and M. Wein, Computer-Generated Key-Frame Animation, Society of Motion Picture + TV Engineers, Vol. 80, 1971, pp. 149-153.

[4] Horn, B.K.P., Kinematics, Statics, and Dynamics of Two-Dimensional Manipulators, Artificial Intelligence: An MIT Perspective, Vol. 2, 1979.

[5] Korein, J.U. and N.I. Badler, Techniques for Generating the Goal Directed Motion of Articulated Structures, IEEE Computer Graphics and Applications, Vol. 2, No. 9, 1982, pp. 71-81.

[6] Perez, L. and M.A. Wesley, An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles, Communications of the ACM, Vol. 22, 1979, pp. 560-570.

[7] Magnenat-Thalmann, N. and D. Thalmann, Actor and Camera Data Types in Computer Animation, Graphics Interface '83, 1983, pp. 203-207.

[8] McGhee, R.M., Control of Legged Locomotion Systems, Proc 1977 Joint Automatic Control Conference, Vol. 1, pp. 205-213.

[9] Paul, R.P., Robot Manipulators, MIT Press, Cambridge, Mass., 1981.

[10] Zeltzer, D., Motor Control Techniques for Figure Animation, IEEE Computer Graphics and Applications, Vol. 2, No. 9, 1982, pp. 53-59.

[11] Zeltzer, D., Representation of Complex Animation Figures, Graphics Interface '82, 1982, pp. 205-211.