REPRESENTATION OF COMPLEX ANIMATED FIGURES

David Zeltzer

Computer Graphics Research Group
Ohio State University
Columbus, Ohio

ABSTRACT

Humans and animals are extraordinarily complex mechanical systems whose articulated movements are difficult to represent, control, and display. This paper describes a method for representing articulated figures as skeletons, for use with a skeleton animation system currently under development. Our implementation includes a translator for mapping structured descriptions of skeletons into an internal representation, a movement processor which accepts input from movement scripts or motor control programs, and a graphical processor which allows debugging, display, and real-time playback of motion sequences. Output from the graphics processor can be interfaced to a variety of display algorithms so that 3-D shaded and anti-aliased animation can be produced using data generated by the skeleton animation system.

KEYWORDS: computer animation, movement representation, human movement simulation, graphics transformation systems.

1. INTRODUCTION

Despite many advances in graphics technology, the world of three-dimensional computer graphics is with few exceptions a desolate one. The reason is that humans and animals are extraordinarily complex mechanical systems whose articulated movements are difficult to represent, control, and display. The human figure, with over 200 degrees of freedom, is capable of motion so complicated that we are still learning how to measure and define it. If natural-looking, three-dimensional figures are to appear in computer-generated animation sequences, the burden of specifying motion in detail will have to be left to the animation software. Otherwise, articulated motion will remain prohibitively expensive and tedious.

Many animals (and mechanisms) can be abstracted as "skeletons". The essential qualities of motion can be captured adequately by a simplifieid figure modeled as a collection of connected, rigid segments. In contrast with "stick figure" representations, skeleton representations may be arbitrarily complex, so that users may model the actual skeletal structure of humans or other vertebrates. The ambiguities and perceptual difficulties associated with stick figures can be avoided with skeletons, since the user can represent both multiple segments and the curvature of bones. For example, longitudinal rotations of the human forearm are easily distinguishable when both the ulna and radius are present in the model. If, in addition, the display system can generate perspective views, observers generally have little difficulty in interpreting movements of the skeleton.

Reproducing the actions of muscle masses, soft tissue, and clothing on a figure presents a further set of difficulties which, in our view, can be approached independently. Once skeletal movement systems are well-understood and implemented, we believe, software can be developed to map polygonal or higher-order surfaces onto moving skeletons.

We are developing a goal-directed skeleton animation system that will allow artists, animators, and other non-computer specialists to define, control, and display complex motion of articulated skeletons. The current implementation includes a translator for mapping structured descriptions of arbitrary skeletons into an internal representation, a movement processor which accepts input from low-level movement scripts or motor control

This work was supported in part by National Science Foundation Grant No. MCS 7923670.

programs, and a graphics processor which controls the display, debugging, and real-time playback of motion sequences. A human skeleton has been defined, and we have completed work on a motor control program for generating straight-ahead gait over level, unobstructed terrain. Output from the graphics processor can be interfaced to a variety of display programs, so that high quality three-dimensional shaded and anti-aliased animation can be produced using data generated by the skeleton animation system.

There are a number of approaches to portraying complex motion. Key-framing (4,11), a technique drawn from conventional animation, is general with no restrictions on the kinds of figures that can be animated. However, it is essentially two-dimensional and difficult to extend to three dimensions. The same is true of Baecker's P-curves (2), intended to provide the animator with greater control, and of Reeve's recent extension (8). All of the above allow the representation of arbitrary figures but require explicit and detailed motion descriptions from the animator. Wessler's three-dimensional system (12) was based on a modification of key-framing, and was restricted to the portrayal of a human figure executing a series of stylized gaits. Hartrum (5) constructed a system based on physiological data which could generate sequences showing a walking human "stick figure". His work could not be extended to other kinds of figures, nor could the walk be easily altered. Badler's work (1) is based on the Labanotation movement description language. Labanotation offers the animator a very wide movement vocabulary, but it is not clear that it can readily be extended to describe the movement of other, possibly imaginary, figures. While it is not a figure animation system per se, the GRAMPS graphics system (7), with its facilities for defining articulated objects is closest in spirit to the ideas we present here.

In this paper we discuss our approaches to representing skeleton descriptions and skeletal motion, and give some extensions to the standard tree-traversal algorithms for calculating graphic transformations. The system is general and can operate on arbitrary figures.

2. SKELETON DESCRIPTIONS

What are the minimum requirements for a skeleton description? Articulated motion is usually implemented (6) by constructing a transformation tree in memory where each node represents some primitive transformation on an object. Objects nested more deeply in the tree will be transformed by matrices stored at ancestor nodes when the tree is traversed. At the very least, a skeleton description must name all the joints or segments in the skeleton and specify how they are to be connected in order to construct a transformation tree.

Since our system is to be accessible to non-expert users, the description must be both human-and-machine-readable so that animators may readily define new figures. In addition, skeleton descriptions must be conveniently modifiable and estensible to give the animator maximum freedom to alter figures as work on a sequence progresses.

Most programming languages provide facilities for describing nested control and data structures. We have chosen an analogous linguistic representation for skeletons, based on a context-free grammar that is simple enough to allow compact representation of real sekeltons, yet general enough to allow arbitrary complexity. The use of a language for skeleton representation has several advantages. It will not be unfamiliar to users of programming languages and animation languages. In the sense that it is a naming of parts of the body it is not unlike a "natural language" description. And finally, the description can be altered and extended easily with any text editor.

Like programs written in a number of structured programming languages, skeleton descriptions have two parts: a declarations part and a description part. In the declarations part, the user specifies certain data to be maintained at each node. Each joint may rotate about up to three local coordinate axes, specified 'x', 'y', and 'z'. For convenience, these are intially taken to be parallel to the global coordinate axes. In addition, the user provides a set of rotational constraints for each axis of rotation. Figure 1 shows a portion of the declarations part of the skeleton description for a six-legged dragon, listing the degrees of freedom for each joint and the associated constraints.

In the description part the user specifies the transformation hierarchy using two structures: the "compound", which defines a joint where two

```
/* Dragon declarations */
body: x 0 360 y 0 360 z 0 360;
/* Rotations of the skull */
skull: x -60 60 y -180 180 z -60 60;
/* Actions of the jaw */
jaw: x -60 0 y -10 10;
/* Actions of the shoulders and hips
6-limbed dragon has "front shoulders", "mid
  shoulders", and "hips" */
l_fshoulder: x -90 180 y -60 60 z -45 180;
r_fshoulder: x -90 180 y -60  60 z 45 -180;
l_mshoulder: x -90 180 y -60 60 z -45 180;
r_mshoulder: x -90 180 y -60 60 z 45 -180;
l_hip: x -90 180 y -60 60 z -45 180;
r_hip: x -90 180 y -60 60 z 45 -180;
/* Actions of the elbows and knees */
l_felbow: x 0 -150; /* "hinge" joints */
r_felbow: x 0 150;
l_melbow: x 0 -150;
r_melbow: x 0 150;
l_hknee: x 0 -150;
r_hknee: x - 150;
```

FIGURE 1
Portion of the declarations part of the description of a dragon

or more segments meet, such as at the pelvis or wrist; and the "limb", which defines a sequence of connected segments. Compounds and limbs may be nested to any level.

Compound joints are indicated by "begin-end" blocks, where the first string after begin is taken to be the name of the compound joint and all strings up to end are the names of dependent joints. For example:

```
begin wrist thumb index middle ring little end
```

defines a wrist joint with five movable (though rigid) fingers. In this compound, the dependent joints are taken to be each attached at the wrist, rather than attached sequentially, as in a limb. A limb is indicated by a set of strings enclosed in parentheses, e.g.

```
(indexl index2 index3)
```

defines a three jointed finger. A whole hand could be defined by

```
begin wrist
  (thumbl thumb2)
  (indexl index2 index3)
  (middlel middle2 m
  (ringl ring2 ring3)
  (littlel little2 little3)
end /* wrist */
```

The description part, then, is simply an indented listing of joint names delimited by the appropriate parentheses and begin-end pairs. Figure 2 shows a description of a simplified human skeleton.

```
begin body /* mandatory top node */
  begin pelvis /* hips and spine "depend on
  pelvis */
   (l_hip l_knee begin l_ankle
    (l_balloffoot l_toe)
    (l_heel) end)
   (r_hip r_knee begin r_ankle
    (r_balloffoot r_toe)
    (r_heel) end)
   begin spine /* unarticulated spinal column */
    (l_clavicle l_shoulder l_elbow begin l_wrist
      (l_thumbl l_thumb2)
      (l_indexl l_index2 l_index3)
      (l_middlel l_middle2 l_middle3)
      (l_ringl l_ring2 l_ring3)
      (l_littlel l_little2 l_little3) end)
    (r_clavicle r_shoulder 4_elbow begin r_wrist
      (r_thumbl r_thumb2)
      (r_indexl r_index2 r_index3)
      (r_middlel r_middle2 r_middle3)
      (r_ringl r_ring2 r_ring3)
      (r_littlel r_little2 r_little3) end)
    (skull jaw)
   end /* spine */
  end /* pelvis */
end /* body */
```

FIGURE 2
Description of a simplified human skeleton

To keep the tree traversal algorithms simple, the output of the parser is a binary transformation tree containing a node for each named joint. When necessary the parser inserts extra nodes at compound joints which transforms them into a series of binary subtrees as in Figure 3.

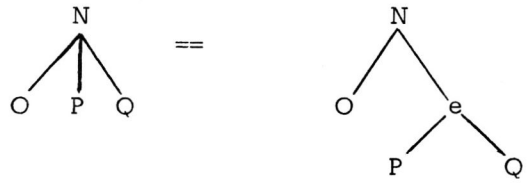

FIGURE 3
Extra node (e) is inserted to generate a binary subtree

Users may thus create skeletons of arbitrary complexity since there are no constraints on the branching factor at any joint.

In addition to the transformation tree, the parser generates a symbol table with entries for each joint which contain information from the declarations part of the skeleton description, i.e., the number and names of each degree of freedom (i.e., axis of rotation), and the rotational constraints for each degree of freedom. The initial coordinates of each joint, input from a second file, are also stored in the symbol table. Metric information (joint coordinates) is input independently of the topological description in order to insulate skeleton descriptions from the varying display spaces of data generation and display programs. The current rotation angles, transformation matrix, joint coordinates, and additional "bookkeeping" information are also maintained in the entries for each joint.

These data structures -- the transformation tree and the symbol table -- are the objects on which all other parts of the movement system operate. Motor programs and user movement scripts update the rotation data at each joint. The movement processor traverses the tree and computes new coordinate axes and position coordinates for each joint. The display processor generates a vector display directly or outputs data suitably formatted for one of several available raster display programs.

3. MOVEMENT REPRESENTATION

Our movement simulation is based entirely kinematics, without attempting to model the effects of physical forces on the skeleton. A large and useful movement subset for any figure can be completely specified kinematically; dynamic simulations could be added where necessary. Velocities and accelerations can be computed from changing joint angles. If the mass of a figure were taken into account, it would be possible to compute a parabolic trajectory to simulate a leap, for example. More difficult motions would require more detailed information on the physical properties of the figure, e.g., the masses associated with each body segment. Thus the kinematic simulation could by systematically extended to generate ballistic movements.

Human locomotion (and in general, vertebrate locomotion) is entirely the result of rotary motion about the joints (10). The apparent translation of the figure through space is the result of rotations about the many links of a complex kinematic chain. At times, an endpoint of the chain may be a fixed center of rotation; at other times, the same endpoint may be free-swinging. Centers of rotation may shift within the chain as well. In human walking, for example, the pelvis rotates about alternate hip joints as the figure progresses through stance phase, then swing phase, then stance again. The path traced out by the body's center of gravity is sinusoidal and not linear. A simple pre-order traversal of the transformation tree will not capture these fundamental properties of creature motion unless some provision is made for altering centers of rotation as the figure moves. Ad hoc solutions based on the geometry of a particular figure are of course feasible (5,12). For animating arbitrary skeletons, however, a more general treatment is required.

3.1. TYPES OF ROTATION: BENDS AND PIVOTS

Each body segment will have, let us say, an initial or proximal endpoint, and a distal endpoint. In some cases, there may be more than one distal endpoint. It may be convenient, for example, to define the initial endpoint of the pelvis as being midway between the hips. Either end of the pelvis then becomes a "distal" endpoint. Now at each node of the transformation tree we can specify a center of rotation as well as an angle of rotation for each degree of freedom, giving several possible kinds of rotation.

We will say a "bend" is any rotation about a proximal endpoint. All non-supporting movements, such as swinging the arms or legs freely, turning the head, and so on, can be implemented as bends. A rotation about a distal endpoint we call a "pivot", such as the various rotations of the

pelvis about the hips as support alternates between limbs during locomotion.

Transformations can also be assigned to the top node, "body", for global (whole figure) movements. We can thus specify global bends and pivots as well. A global bend corresponds to a rotation of the entire body about its center of gravity (the body's "proximal endpoint"), as in diving or tumbling. In locomotion, the body rotates over various support points, and this can be implemented using appropriate global pivots about the heels, toes, and balls of the feet in addition to local bends and pivots.

In general, non-support movements are implemented using bends, while support movements can be implemented using global pivots or a combination of local and global pivots. The process is analogous to shaping the movements of a flexible doll or mannikin. To execute a kneebend, for example, the body is pivoted forward about the ankles, backward at the knees, and forward at the hips. The result is that the feet remain stationary and the body appears to drop into a squat. If local bends alone were used, the body would appear to hang in space with the feet drawn upward, and additional downward translation would be necessary to correct this. For more involved support motions, such corrections may be non-linear and difficult to deduce. The use of global pivots eliminates such difficulties since the correct movements of the figure through space are automatically generated. Significantly, this means that we can simulate the interaction of arbitrary figures with their environment. Reliable data on the position of the figure is generated automatically by the movement processor and is continuously available to motor control programs. This is essential for goal-directed simulation where the animation software is expected to generate the path of a moving figure through a possibly complex environment.

3.2. THE TRANSFORMATION TREE: PROGRAM ACCESS AND TRAVERSAL

The skeleton transformation tree may be accessed in two ways. First, an interactive user interface is available to call up a skeleton and manipulate it "by hand", i.e., by entering keyboard commands to rotate various joints. Input may be redirected to disk so that long sequences of movement commands may be entered automatically. Second, motor control programs may generate movement commands. In either case, write access to the skeleton data base is restricted to calls to a single movement primitive called "bend". Requests to bend specify, among other things, the joint to rotate, the joint to rotate about (proximal or distal endpoint, called the "pivot" joint), and the angle of rotation.

Each call to bend invokes a partial tree traversal which begins with the rotating joint and continues until the leaves of the tree are visited. Whenever a pivot is requested (i.e., the pivot joint is a distal endpoint of the rotating joint), the subtree whose root is the pivot joint is not traversed. Thus the pivot joint remains stationary, while the moving segment (or the whole figure) rotates about it.

Local coordinate axes and the current position of each joint are maintained explicitly in the symbol table. For each traversal, we compute the following transformation: translate to the current location of the pivot joint (proximal or distal), rotate about the given degree of freedom (axis), and translate back. The rotation matrix provides for rotation about arbitrary axes (9). As each node is visited, the local coordinate axes and joint coordinates are transformed by this matrix and stored. Thus the data base always reflects the current location and orientation of all the body parts; skeleton movements are always with respect to the current configuration; and bends, pivots, and global pivots may be requested in any order.

4. CONCLUSION

The skeleton animation system provides the user with convenient access to articulated objects -- the skeleton software is entirely responsible for maintaining the integrity of the skeleton so that the user cannot input a command that would cause, say, the skeleton's arm to fall off. Moreover, motion is abstracted at a higher level: the user can think of movement in terms of joints and body parts rather than primitive objects and transformations.

In the long term we see the role of the computer animator becoming more like that of a director of cinema or the theater as we strive to

give a measure of autonomy to the figures and objects we simulate. The work described above constitutes the set of movement primitives for our skeleton animation system. We are committed to the implementation of goal-directed systems using these movement primitives. The resources of robotics and artificial intelligence thus become increasingly necessary tools.

We have discussed the application of robotics control techniques to animation systems elsewhere (13). Ultimately, we are striving for a system which will accept "natural language" scripts as input to the animation controller, which coordinates the motion of predefined skeletons.

Using the movement primitives described above, a set of hierarchically organized motor control programs, and a skeleton defined in our skeleton description language, we are able to generate realistic sequences of straight-ahead gait over level, unobstructed terrain. Each frame takes 2-3 seconss to compute on a VAX 11/780. The internal buffer of our stroke-refresh display can store about 130 frames which we can then play back in realtime -- an enormous help in debugging the movement controller. Figure 4 shows a typical display. Figure 5 shows a complete sequence of 88 frames played back with the "trace" feature on. With tracing turned off, sequences can be played back forward or in reverse; the figure appears to walk forward and at approximately normal speed. Figure 6 shows a more detailed skeleton in a scene displayed on a raster monitor using the ANTS animation language (3) and walk control data generated by the skeleton animation system. A figure composed of polyhedra or other volume elements could be controlled using the same data. The single frame was extracted from a sequence showing the skeleton walking across a plane with several objects in the background.

The skeleton description language provides the animator with a new tool for defining articulated figures. Moreover, the movement primitives we have implemented (bends and pivots) enable the animation system to correctly generate the complex motions we take for granted in the real world.

5. ACKNOWLEDGEMENTS

The author would like to thank Don Stredney for assistance in developing gait models and for data generation; Mike Collery for the ANTS animation interface and photography; and Professor Charles Csuri and the Computer Graphics Research Group.

REFERENCES

1. Badler, N. and Smoliar, S., "Digital Representation of Human Movement," ACM Computing Surveys, Vol. 11 (1979).
2. Baecker, R., "Interactive Computer Mediated Animation," Ph.D. Thesis, MIT (1969).
3. Csuri, C., Hackathorn, R., Parent, R., Carlson, W., and Howard, M., "Towards an Interactive High Visual Complexity Animation System," Proc. Siggraph 1979, (August 1979).
4. Gracer, F. and Blasger, M., "KARMA, A System for Storyboard Animation," Proc. Ninth Annual UAIDE Meeting, (1970).
5. Hartrum, T.C. "Computer Implementation of a Parametric Model for Biped Locomotion Kinematics," Ph.D. Thesis, The Ohio State University (June 1973).
6. Newman, William and Sproull, Robert, Principles of Interactive Computer Graphics, 2nd Edition, McGraw-Hill, New York, (1979).
7. O'Donnel, T.J. and Olson, Arthur J., "GRAMPS -- A Graphics Language Interpreter for Real-Time, Interactive, Three-Dimensional Picture Editing and Animation," Computer Graphics, Vol. 15 (3), SIGGRAPH-ACM, (August 1981).
8. Reeves, W., "Quantitative Representation of Complex Dynamic Shapes for Motion Analysis," Ph.D. Thesis, University of Toronto, (1980).
9. Rodgers, David F. and Adams, J. Alan, Mathematical Elements for Computer Graphics, McGraw-Hill, New York, (1976).
10. Steindler, Arthur, Kinesiology of the Human Body, Charles C. Thomas Books, Springfie , Illinois, (1955).
11. Wein, M. and Burtnyk, N., "A Computer Animation System for the Animator," Proc. Tenth Annual UAIDE Meeting, (1971).
12. Wessler, B., "Computer-Assisted Visual Communication," Ph.D. Thesis, University of Utah, (1973).
13. Zeltzer, D. and Csuri,C., "Goal-Directed Movement Simulation," Proc. Conf. Canadian Society for Man-Machine Interaction, (June 1981).

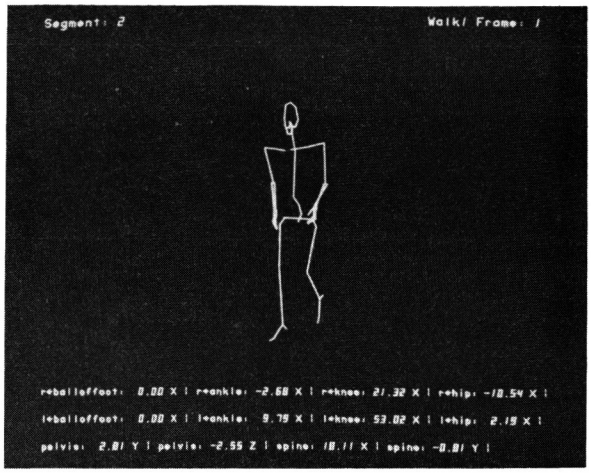

FIGURE 4. Human figure in a typical frame generated by the walk controller.

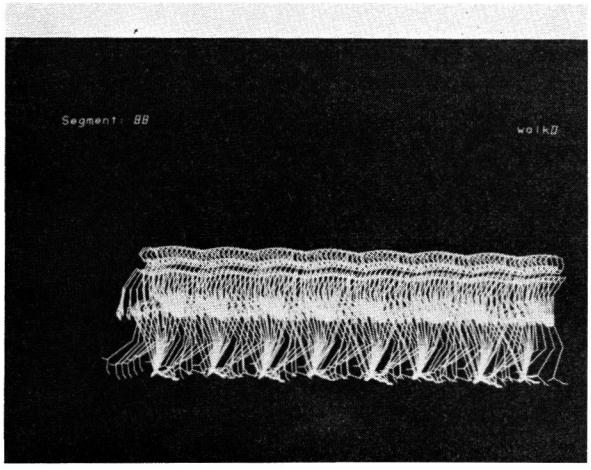

FIGURE 5. Trace of 88 frames of a walk sequence.

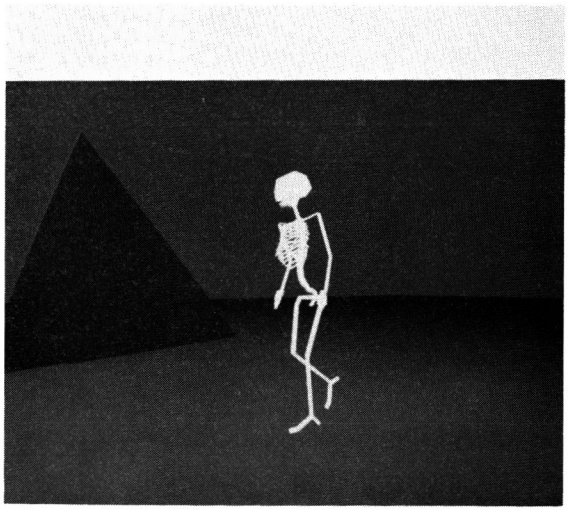

FIGURE 6. Single frame from an animated sequence displayed on a raster monitor, showing a more detailed skeleton controlled by data from the skeleton animation system.

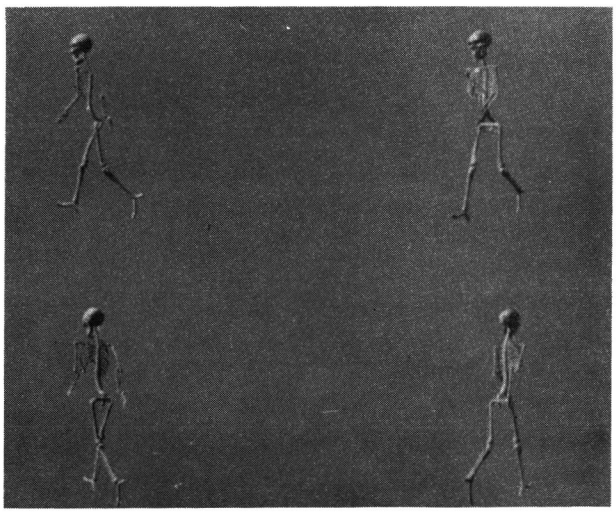

FIGURE 7. Four views of a walking skeleton.