Interactive Keyframe Animation of 3-D Articulated Models

David Sturman
Computer Graphics Laboratory, New York Institute of Technology, Old Westbury, NY 11568

Abstract

This paper discusses some of the issues concerned with keyframed computer animation of 3-D articulated models and the problems in designing interactive systems for this type of animation. Examples are taken from the four years of keyframe animation of articulated models done at the NYIT Computer Graphics Lab, and from our recent attempts to refine our original keyframe animation system, *BBOP*. This paper also addresses the importance of the interaction of animators with keyframe animation systems as an element in the design of such systems.

Résumé

Cet article aborde certains des aspects de l'animation par ordinateur traitée par images intermédiaires de modeles 3-D articulés et des problemes de la conception de systemes interactifs appliqués à ce genre d'animation. Des examples décrits sont extraits de nos quatre ans d'expérience en animation par images intermédiaires de modeles articuleś au sein du NYIT Computer Graphics Laboratory et de nos récentes tentatives d'améliorer notre systeme d'animation par images intermédiaires, BBOP. Cet article souligne aussi l'importance de l'interaction des animateurs avec des systemes d'animation par images intermédiaires comme un élement de la conception de tels systemes.

KEYWORDS: Animation, articulated model, keyframe, interactive, user-friendly.

Introduction

Animation for film and video has traditionally been a long and tedious task, with many animators drawing and painting each frame by hand. With the advent of computers, animators turned toward these machines to take the drudgery out of the animation process. Early systems worked with two--dimensional images, automating the tedious inbetweening task. Animators specified the correspondence between lines in successive key frames by tracing them in a fixed order. The computer would generate the frames in between by interpolating corresponding lines from the keyframes [1,2]. This technique has been refined and is still very popular in modern animation systems like the TWEEN system produced by CGL, Inc. [3]

As three dimensional animation became possible with faster machines and better display hardware, keyframing was adapted to 3-D animation. Instead of interpolating corresponding 2-D line segments, 3-D animation systems interpolate transformations at joints in a three-dimensionally represented model. Key frames consist not of 2-D images, but of 3-D positions of a model.

Because the models were represented in 3-space and projected on the image plane by the computer, these systems tend to produce more realistic-looking images, than those produced by an animator who approximates a 3-D representation with a 2-D drawing.

This paper discusses some of the techniques and problems in the design of 3-D keyframe animation systems, stressing the importance of animator interaction. It draws heavily on the NYIT Graphics Lab's keyframe animation system BBOP[4,5], and a more recent NYIT animation system, EM[6].

Models

The information stored in a model is an important aspect of any animation system. One simple way to define models is as a set of rigid objects jointed at nodes, organized hierarchically into an articulated body. At each node or joint, a 3-D transformation matrix controls the position of the portion of the body below that joint. Transformation matrices are nested in accordance with the body structure. The position of the model at any one instant is determined solely by the transformation matrices. The only intelligence contained in the model is the topology of the body parts and the degrees of freedom at each joint. Alone, the model is a static entity. To make the model move, the animator uses the animation system to control the 3-D transformation values at each joint. The "rigid object" stipulation allows scaling of the body parts (using the joint matrices) but not flexing or changing their basic geometries. These are the types of models used by the animation systems BBOP and GRAMPS[7]. The particular model structure was motivated by the Evans & Sutherland Multi-Picture System (MPS), on which the systems are based. The MPS manages nested transformations easily: performing real-time transformation, clipping, and display of lines.

Like BBOP, EM uses models constructed of parts connected at joint nodes. However, EM uses a geometric modeling language which allows parametric control over the transformations at each joint and over the geometry of the individual body parts. The set of parameters define the model's final position, shape, and characteristics. Parameters can be constrained and coordinated with respect to constants or other parameters in order to give the model intelligence in the way it moves. For instance, the motion of a ball can be dependent on the slope of the floor it rolls across, or swinging arms can be made to swing opposite to each other.

Scripted systems [8,9,10,11] use procedural models [12], which embed the possible motions of the model in the model description itself, leaving some parameters for external control.

Regardless of the implementation of its models, an animation system should allow animators to easily modify model structure and movement. Interaction and visual feedback are important. Systems like BBOP, EM, and GRAMPS are significant in this regard because the animator can immediately view his changes.

Positioning models

Because keyframed animation is based on key "still" frames, the method of creating these frames is a vital component of the system. The animator must be able to easily set up all the parameters necessary to define a keyframe. One basic method of positioning a model in a keyframe is to type in a joint identifier, the parameter to be changed at that joint, and the value for that parameter. Although functional, this method is not easy to use.

In BBOP the animator selects a joint using a joystick to traverse the transformation tree of the model. With a set of function keys, the animator specifies that he wants to modify the translation, rotation, or scaling parameters at that joint. Each parameter set has three elements, one for each of the x, y, and z axes. Using a three-axis joystick, the animator can modify these three values and watch the model move on the screen. BBOP provides no constraints or rules about moving the model, except that it follows the x, y, z movements of the joystick. When manipulating a human body model, for instance, limbs can disconnect from the body structure and joints can be bent at unrealistic angles, even causing a limb to enter the body itself. At each joint, the interaction is the same: the x, y, and z, translation, rotation, and scaling parameters are controlled by the three outputs of the joystick. This makes the system simple to use, and an inexperienced person can learn to manipulate a model in just a few minutes.

GRAMPS, which has a similar interactive input method, takes a step beyond BBOP by allowing several devices for input. By means of simple functional assignment statements, values from eight dials, a data tablet, and a joystick can be used to modify the model's parameters. For instance, the animator can assign the inverse value of a dial to the rotation of a character and set bounds on the values the rotation can assume. EM builds on BBOP and GRAMPS, allowing input modes to be defined dynamically with complicated dependency expressions including other parameters and multiple input devices.

We noticed an interesting phenomena in systems which have user configurable inputs. In BBOP, the interaction modes are predefined and remain the same from joint to joint and model to model. An animator can sit down with a new model and immediately begin to manipulate it. In EM, the interaction modes can be configured specific to each parameter and joint in the model's tree structure, and by each user. Each model and each user can have different input modes. Thus, it is difficult to sit down with a new model and animate it right away. It takes time for the animator to get used to the model's control characteristics. Once familiar with a model's movement, however, configurable inputs have proven to be very effective. Especially useful has been the ability to interactively control parameters of several joints of the model at the same time. For instance, the animator can rotate the shoulder, wrist, and waist of a model simultaneously. This was not possible with BBOP.

Modeling camera movement is also an important part of an animation system. There are many ways to move the camera interactively. We first must define the coordinate axes in which the camera can be moved. We define the pivoting of a camera on its own axes, i.e. tilting, panning, and rolling the camera, as rotation in the camera's local coordinate system. Camera movement around an external center point is movement in a global coordinate system. This is exemplified by a camera mounted on a crane that moves around the space of the "animation studio." In BBOP, the center of global camera movements is always the center of world space and cannot be changed. Movement along arbitrary coordinate axes would be an important enhancement to the camera model. This would aid in tracking a movement of the model or moving the camera along a particular trajectory while maintaining a constant object of interest.

The second aspect of camera movement is how the camera moves relative to the model or scene. One approach to interactive camera movement is to have the joystick (or other interactive device) operate as if it were attached to the camera. This is the local or "airplane" method of camera control, because the joystick models the pitch, yaw, and roll of an airplane joystick. Pushing the joystick forward causes the camera to tilt down; pulling the stick to the side causes the camera to tilt to that side. The second method is to model the input as if the animator were controlling the world. In this "global" mode, joystick control creates just the opposite effect of the "airplane" method. Pushing the joystick forward causes the camera to tilt up (or apparently, the world tilt to down, away from the eye). The two methods are computationally equal since moving the camera to the left is indistinguishable from moving the world to the right. Different people favor different approaches; however, the majority prefer the "move the world" approach because the picture they see moves in the same direction as the joystick. BBOP and EM also can simulate multiple cameras, enabling an animator to display the same animation as seen from several viewpoints.

In addition to camera movement, there is local and global movement at the joints of the model. That is to say, transformations at a joint can take place in the coordinate system local to the joint or in the coordinate system of the next higher, or "parent," joint. (In an ideal implementation the animator would be able to effect transformations in any coordinate system. This would aid, for instance, in making a figure walk so that it rotates over the balls of the feet, not around the center of the body.) Because BBOP has just one interaction mode – joystick x, y, z controlling model parameters x, y, z, – the joystick's motion often has little physical relation to the motion of the model on the screen. To get around this problem, the animator can display the local coordinate axes as they move with the current joint. EM fosters more natural interaction because the animator designs the input modes, adapting each to a particular manipulation of the model. An example of this is to set the tablet x/y to be the model x/z position on a floor, and the joystick x, y, z to the model's rotation around its z, x, y axes respectively. Pushing the joystick away from you would make the model tilt away from you.

The most natural input method perhaps would allow the animator to point directly to a joint and "drag" it around on the screen. Thus the animator would manipulate the model by pushing and pulling it into position. There are, however, implementation problems in trying to manage 3-D control from a 2-D view. For instance, determining the coefficient of movement along the axis perpendicular to the screen can be quite complicated. A typical solution would be to set that "z" coefficient to zero, but a more sophisticated solution would be better.

Finally there are non-kinetic parameters such as color, reflectance, elasticity, etc. which the animator may need to control. These qualities may be required if a raster version of the animation is desired, and sometimes are perceptible only in a fully rendered scene. The time it takes to render scenes is usually prohibitive to display the actual property to the user in an interactive fashion. An alternative form of viewing these values is necessary. In EM, the animator can learn the current value of any parameter by typing its name. With color vector devices, part of this problem can be solved by coding various properties with different colors.

Keyframing

A frame in an animation is basically a description of the particular state of the world at a particular instant in time. In a keyframe system, the animator need not describe each frame. Instead he describes a set of "key frames" from which the animation system can interpolate the frames in between. Interpolation of intermediate values can be linear, cubic spline, cosine, etc.

The information stored at a keyframe may vary from a total description of the scene and animation to the value of a single parameter used in the generation of a frame. In scripted systems, the "key" information is more like a cue to a stage performer [9]. The cues tell the models (or actors) to start, stop, and in some cases to modify or switch their behavior to some other pre-programmed mode. In goal-oriented systems, the animator describes a goal state and the model moves towards that state using its knowledge of itself and the world [11,13]. Goal states can be considered the keys with the model itself generating the inbetween movements.

To set up keyframes in BBOP or EM, the animator positions the model on the display screen. Then he can record the position of the whole model, a subset of the model, or just the value of a single parameter as a keyed position in a particular numbered frame. In 3-D systems, each parameter has its own set of keys, unlike 2-D systems which key a whole image. In BBOP, every value at every joint for every frame is saved. Those values to be used as keys have a note to that effect. This is a simple but storage-intensive implementation. When an animator specifies a position as a key in EM, the system saves only the values of keyed parameters, the frame number, and interpolation information about the interval between that key and the next. EM has been implemented in a way that eliminates the need to maintain values at the inbetween frames. This scheme is more complicated than BBOP's, but more space-efficient. EM also can save key positions by name, separate from the interpolated sequence of numbered keyframes. Numbered keyframes can contain references to named keys, thus providing a sort of macro facility to the keyframe process. For example, if the same position is needed in multiple keyframes the position can be described once and then referenced by each key. If the position needs to be changed, modification of the original copy modifies it in the other keys.

An animator often uses the same key many times. Copying keyframes from one position to another at first seems to be conceptually simple, but when looked at carefully a number of issues become apparent. Frequently, an animator copies (or moves) keys forward or backward in a sequence to change the pace or timing of an animation. Moving a keyframe forward to expand a sequence can be done by moving the key to the new location, pushing subsequent keys forward in the process. This expands the particular key interval but also lengthens the entire animation. For modification of the timing of an entire animation, or non-synchronized motions, this is a valid approach. However, moving a key for a motion that is, at some point, synchronized or cued to other motions will push the subsequent keys out of synchronization. Clearly the subsequent keys cannot be moved forward with any integrity unless corresponding keys for the entire animation are also moved forward. Thus, lengthening a keyframe interval for a subset of a model's parameters requires a corresponding contraction of another keyframe interval. The reverse is also true.

In expanding or contracting an interval, an animator may wish to preserve the characteristics of the original motion in the interval. For the most part, interpolating functions take care of that. However, there are cases in which the modified timing of the intervals produces a motion with undesirable characteristics. This is especially true with cubic spline interpolation where keyframe spacing affects the shape of the interpolating curve.

Creating motion cycles is another technique that requires copying keyframes. Repeated motions are common in animation, as in the cycle of a walk or swing. To generate a cycle, an animator can manually copy keyframes of the basic motion to multiple locations, one for each repetition of the cycle. This method is tedious and error-prone. Often the original keys contain information unnecessary to the cycle and need to be trimmed down. Additionally, if the motion is to be repeated many times, the keyframe copy process is unwieldy. Ideally, some form of cycle operator or interpolation should exist in the animation system.

An alternative approach to modifying animation pacing or cycle generation is to manipulate the sequence of frame playback. An animator might draw a curve as a function of time to indicate to the animation system the frame playback sequence he wants. This curve could be saved and later used to control frame playback. A ramp function might be used to produce linear playback of frames at a speed dependent on the slope of the ramp. Cosine curves would create cyclical animation. Acceleration and deceleration could all be described in terms of this function. With further refinement, different parts of a model or animation could become subject to different pacing functions.

As well as being able to key motions, the animation system should make it possible for the animator to key attributes such as color, reflectance, etc. A flexible database is necessary if these attributes, varied in number, are to be added to a model without destroying previous animation.

Using keyframe systems, an animator must manage scores of joints across hundreds of frames; perhaps hundreds of individual keys. Several systems have successfully tackled the problem of giving such control to the animator. One such system, MUTAN[14], divides a model or group of models into tracks – separate entities that can be keyed individually. For each track, the system displays a sort of ruler that spans the range of frames. Tick marks indicate keys and include notations about the action at that key. Animators manipulate the tick marks to change key positions. MUTAN also has a mode which the animator can use to deal with the set of keys at a particular frame. Another system, DIAL[15], employs a specialized notation that the animator edits on a regular alphanumeric terminal. It displays frames horizontally and tracks vertically on the screen. The animator can view parallel tracks at once to facilitate coordination of keys. BBOP and EM have a special motion editor to manipulate keys along a single track (or parameter), and a command that prints the list of key frames for a particular parameter.

Interpolation

The interpolation process has been given little attention in many animation systems. Usually they only support linear and cubic spline interpolation. Some systems allow cosine interpolation and acceleration or deceleration functions. Animators' experiences with BBOP indicate that control of the inbetween frames is very important. There are cases in which the animator only needs linear or cubic spline interpolation, but there are also cases in which a more sophisticated motion is desired. One way to accomplish this is to add more keyframes. With this method, however, the animator quickly gets lost in a forest of keys, and the efficiency of the computer inbetweening is lost. Another solution is to offer a variety of interpolation types which can be selected for the intervals between keyframes. This scheme is better but the interface must give the animator a clear picture of the various interpolation types and the motions they produce. Limiting the animator to typing in keyframes and interpolation information may not be sufficient.

BBOP addresses this problem by providing a motion editor. The animator uses the motion editor to view the values of a parameter across the span of the animation. Key frames are clearly marked along the curve. The animator can add or delete keys and specify one of several functions to interpolate values in individual keyframe intervals. In addition, the animator can hand draw the desired motion between keyframes to achieve unique movement.

With more than one type of interpolation possible, especially across the path of a single parameter, it is important how curves of different interpolation types are joined. To control the continuity of the overall animation, the animator must be able to determine the degree of continuity across keyframes. Also, the interpolant's behavior at key boundaries is an important factor in its behavior between keys. ASAS solves this problem by using piecewise cubic curves with a selectable degree of continuity at the joints. MUTAN lets the animator specify acceleration or deceleration functions at keys. The BBOP motion editor supports an ease-in/ease-out function to match slopes at key boundaries. Using BBOP, an animator usually positions models at key frames, previews the motion generated by the default cubic interpolation of the inbetween frames, and then fine-tunes the movement in the motion editor. The motion editor allows an animator to give characters idiosyncratic motions, limps, jerkyness, and human-like qualities that linear and cubic interpolation would not provide.

Conclusions

The art of 3-D animation goes beyond positioning models, setting keyframes, and interpolating the inbetweens. Although many animations can be made with these methods, a wide range of situations require more. Scripted animation systems provide one set of solutions. They tend to allow a high level of control over an animation, simplifying many types of motion control, and are often used to model algorithmic or functionally-defined motions. These systems are well suited to goal-directed animation and the simulation of mechanical processes. However, scripted systems are not good at producing the idiosyncratic and non-algorithmic "natural" motions that professional animators favor in their productions. In addition, they do not provide immediate feedback – an important element in animation systems geared towards animators. Clearly, some combination of scripted and interactive keyframe animation is desirable.

The combining of the two approaches has been researched over the past year at NYIT, primarily by Pat Hanrahan. The animation system, EM, that has grown out of this research, is more clearly detailed in a paper submitted to the ACM SIGGRAPH '84 conference.

Acknowledgements

BBOP was created by Garland Stern, who developed and inspired many of the ideas in this paper. Kenneth Wesley enhanced BBOP to its current state. The BBOP motion editor was developed by Thaddeus Beier. EM was conceived of by Pat Hanrahan, and implemented with the help of the author. Jacques Stroweis provided the French translation to the abstract. Pat Hanrahan, Paul Heckbert, and Jane Nisselson were extremely helpful in providing constructive review and correction to this paper.

References

(1) Burtnyk, N. and Wein, M., "Computer generated key frame animation." *Journal of SMPTE* 80 (March, 1971): 149-153

(2) Catmull, Edwin, "The problems of computer-assisted animation." *Computer Graphics (SIGGRAPH '78 Proceedings)* 12 (August 1978): 348-353.

(3) *Tween Users Manual.* (New York: CGL Inc., [1983]).

(4) Stern, Garland, "Bbop – a system for 3D keyframe figure animation." *SIGGRAPH '83, Course 7, Introduction to Computer Animation,* July 1983: 240-243.

(5) Stern, Garland, "Bbop – a program for 3-dimensional animation." *Nicograph '83 Proceedings.* Tokyo, Japan, December 1983: 403-404.

(6) Hanrahan, P., and Sturman, D., "Interactive control of parametric models." submitted to *Computer Graphics (SIGGRAPH '84).*

(7) O'Donnell, T. J. and Olson, Arthur J., "GRAMPS – A graphical language interpreter for real-time, interactive, three-dimensional picture editing and animation." *Computer Graphics (SIGGRAPH '81 Proceedings)* 15 (July 1981): 133-142.

(8) Hackathorn, Ronald J., "Anima II: a 3-D color animation system." *Computer Graphics (SIGGRAPH '83 Proceedings)* 17 (July, 1983): 91-102

(9) Reynolds, Craig W., "Computer animation with scripts and actors." *Computer Graphics (SIGGRAPH '82 Proceedings)* 16 (July 1982): 289-296.

(10) Magnenat-Thalmann, N., and Thalmann, D., "The use of high-level 3-D graphical types in the Mira animation system." *IEEE Computer Graphics and Applications* 3 (December 1983): 9-16.

(11) Zeltzer, David, "Knowledge-based animation." *Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop, Motion: Representation and Perception.* Toronto, Canada, April 1983: 187-192.

(12) Newell, Martin E., "The utilization of procedure models in digital image synthesis." Ph.D. dissertation, Department of Computer Science, University of Utah, 1975.

(13) Korein, James U. and Badler, Norman I., "Techniques for generating the goal-directed motion of articulated structures." *IEEE Computer Graphics and Applications* 2 (November 1982): 71-81.

(14) Fortin, D., Lamy, J.F., and Thalmann, D., "A multiple track animator system for motion synchronization." *Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop, Motion: Representation and Perception.* Toronto, Canada, April 1983: 180-186.

(15) Feiner, S., Salesin, D., and Banchoff, T., "Dial: a diagrammatic animation language." *IEEE Computer Graphics and Applications* 2 (September 1982): 43-53.