VIRYA - A MOTION CONTROL EDITOR FOR KINEMATIC AND DYNAMIC ANIMATION

Jane Wilhelms
Computer Graphics and Imaging Laboratory
Computer and Information Sciences Board
University of California, Santa Cruz, CA. 95064, U.S.A.

Abstract

Virya is an interactive graphical motion control editor for kinematic and dynamic animation. Most animation is controlled *kinematically*, by designating objects' positions taken over time without consideration for the causes of the motion. An alternative is *dynamic* motion control, where objects are seen as masses moving under the influence of forces and torques. Dynamic motion control has some advantages in that motion more naturally simulates real world conditions and many complex motions can be automatically calculated, though calculating motion is quite expensive and control is sometimes less intuitive. The editor *Virya* works both for kinematic and dynamic motion control. It has two main tasks: to specify *control functions* representing positions (kinematic) or forces and torques (dynamic) controlling motion, and to specify *control modes* which designate how control functions are interpreted or whether joints are frozen in place, relaxed, or balanced. Using these control modes, the user can designate motion using a convenient kinematic method and still use dynamic analysis as a final step to constrain and add realism.

KEYWORDS: computer animation, human modeling, dynamics, simulation

1. Kinematic Motion Control Methods

Motion control is a central problem in computer animation and is the one aspect of animation that most sets it off from other areas of computer graphics. Common kinematic approaches to motion control are *3-D keyframing*, *motion control functions*, *parametric control*, and *animation languages* (these approaches can, and often are, combined).

In 3-D keyframing, the user typically positions the objects in the scene interactively, designating a sequence of configurations and the times when they should occur [10,11] The animation system then interpolates between these *keyframe* configurations to generate the *inbetween* configurations. 3-D keyframing is inherently superior to 2-D keyframing for 3-D animation because the problems of information loss do not occur. However, keyframing is limited by the necessity of creating many keyframes and the lack of complete control over the interpolation process defining the path and speed of motion between keyframes.

Parametric motion control involves designating certain parameters whose values define a particular configuration of the objects in the world [5]. For example, in the case of facial animation, parameters may designate the position of the mouth, the elevation of the eyebrows, etc. [8]. Parameters are convenient to use and allow association of reasonably complex motions (such as smiling) with one or a few parameters. Choosing parameters that cover the desired range of motion can be problematic, so the user may have to sacrifice complete motion control for ease of use.

Animation languages are an attractive alternative because complex motion can be described in the form of scripts [9]. Some languages are fairly low-level and merely provide a convenient interface to specify simple motions [7]. Higher-level languages would allow the user to specify motion in general terms (e.g. "walk forward") and depend upon an intelligent hierarchical interpretation system to find the specific low-level directions needed to draw the frames [16]. While high-level languages may provide the most convenience to the user in the long run, at present many issues involved in high-level motion control remain unresolved. Again, use of a script may limit the amount of control the user has over the motion.

Kinematic motion control functions represent motion at each degree of freedom in the form of position versus time curves. The control functions can be simply generated and succinctly stored using control points which generate the curve. These control functions are low-level and represent motion at individual degrees of freedom, but do allow very detailed specification of motion. An advantageous feature of control functions is that the final motion description of the other methods (changes to particular degrees of freedom over time) can be easily represented in this form. The use of an interactive control function editor allows the user to make individual changes to motion at the lowest-level and at the last minute, and can make up for some of the loss of exact control often concomitant with the above methods.

2. Dynamic Animation

Most animation systems at present are *kinematically-based*, that is, motion is considered as the relation of position versus time without consideration of the environmental influences causing the motion. *Virya*, the motion control editor described here, was designed mainly for use with the *dynamic* animation system *Deva*. In *Deva*, objects are considered as extended masses which act under the influence of forces and

torques. Using dynamic analysis, this relationship is formulated as the dynamics equations of motion; the solution of these equations is a kinematic description of the motion that would occur under the specified conditions in a "real" (simulated) world [13,14].

Dynamic animation has certain advantages lacking in kinematic systems. Motion is automatically constrained to respond to environmental conditions. For example, it is kinematically difficult to animate a body, such as a human figure, naturally responding to collisions, such as hitting the ground. One problem is that colliding objects should not move through each other. The user can avoid this, at considerable expense, by visually checking for unrealistic intersections, doing automatic collision detection, or implementing constraints using inverse kinematics [4]. Another problem is that when an articulated body collides, the motion of its many connected segments can be extremely complex and difficult to predict. Using dynamics, such collisions can be automatically simulated by applying an opposing force against the body when it collides. Taking this force into account during dynamic analysis results in a natural response to the collision, including a certain amount of bouncing and correct reaction of other connected body parts. Bodies will also automatically respond to gravity or the motion of connected body parts or external influences.

Dynamic animation does has some disadvantages as well. It is computationally much more expensive than kinematic animation [1,14]. When the system dynamics are complex, as in the case of articulated bodies with many degrees of freedom, numerical instability can be a problem. Dynamics also requires initial determination of object and environmental characteristics such as masses, joint limits, scale factors for springs and dampers used in collisions, etc. Perhaps the most serious disadvantage occurs in simulating controlled motion. While bodies respond naturally to environmental conditions, it is difficult to find when, where, and how strongly to apply internal forces and torques simulating muscles in humans and other animals. These issues are explored in detail elsewhere [13,14,15]. Suffice it to say that while dynamic animation is still in its exploratory stages, it does offer a method with considerable potential for simulating realistic motion.

3. Virya Motion Control Functions and Modes

Virya is a motion control editor used to develop, modify, and store motion control information in the form of *control functions* and *control modes*. *Virya* was designed to work within the dynamic animation system, *Deva*. Some of its features are only relevant to dynamic animation systems; other features are relevant to both kinematic and dynamic systems. *Deva* and *Virya* were specifically designed to explore the problems of controlling the motion of articulated bodies such as animals and robots. Though the same principles apply to controlling other, simpler objects, discussions will largely be from the standpoint of controlling articulated bodies.

Virya control functions can either be kinematic, representing positions over time for each degree of freedom, or dynamic, representing forces (sliding joints) or torques (revolute joints) for each degree of freedom. In *Virya*, control functions are represented by cubic interpolatory spline curves, piecewise curves that interpolate a sequence of user-defined control points with first and second derivative continuity [2,3]. Cubic interpolatory splines have the advantage that they interpolate specified control points; they have the disadvantages of being global (changes in one control point affect to varying degrees the entire curve) and given to occasional wild behavior. A local interpolatory spline, such as that of Kochanek [6], or a local approximating spline that can approach control points arbitrarily closely such as the beta-spline [3], may be more desirable. Control functions are easily constructed in *Virya* by picking control points on the screen using a puck and tablet.

Each degree of freedom of the body (e.g., flexion of the elbow or rotation of the head) can exist in one of five control modes for dynamic animation: *relaxed*, *dynamic control*, *frozen*, *balanced*, and *hybrid K-D* modes. Each degree of freedom can alternate between modes during the animation. One of *Virya*'s functions is to specify modes and their durations for each degree of freedom. A sixth, *pure kinematic* mode exists which completely by-passes dynamic analysis. In this case, control functions represent positions over time and are directly sampled to produce purely kinematic animation.

3.1. Relaxed Mode

Dynamic animations can be developed without any user-specified motion at all, merely by placing the body in an unstable position and letting it react to the gravitational force, its own joint limits, the ground, etc. The degrees of freedom in *relaxed* mode will move freely under environmental conditions with no internal controlling force or torque simulating a muscle contraction or a robot actuator motor. These relaxed degrees of freedom are constrained to remain within their joint limits and their motion is slightly damped. Other forces or torques due to collisions or motion of other body parts will still act upon them. (Within the system *Deva*, springs and dampers are used to mimic both joint limits and collisions.)

3.2. Dynamic Mode

To actually control the animation, pseudo-muscular forces or torques must be applied to certain degrees of freedom of the body. For example, to make the body wave its arm, torques must be applied to the shoulder and elbow. The most direct way to specify these forces and torques is to develop a control function for each degree of freedom whose motion is controlled in this way. These control functions represent a force (for sliding degrees of freedom) or a torque (for revolute degrees of freedom) over time. The control functions are sampled to find the appropriate controlling force or torque to apply to specified degrees of freedom at each time instant that dynamic analysis is done; these controlling forces and torques are then added to the automatically calculated forces and torques mentioned above to find the total forces and torques acting upon the body.

While these force/torque control functions have the advantage of directly specifying the controlling forces and torques needed for dynamic analysis, they leave the user at the disadvantage of not knowing intuitively what forces or torques will be necessary to produce the desired motion. This problem is accentuated by the complex interactions between different parts of an articulated body. For example, should the user find the correct force to lift the arm rigidly at the shoulder, and then add torques to the elbow to accompany the lifting by a bending action, he will find that the additional torque at the elbow inter-

feres with the smooth motion of the shoulder. For this reason, other modes have been added for user ease.

3.3. Freeze Mode

It is common during many movements of articulated bodies that some parts of the body remain locally stable. For example, in reaching movements the legs and hips may remain stable, and during walking the head usually is directed forward. Because of the complex interplay mentioned above, it is difficult to find the sequence of forces or torques that will ensure local stability. A simple solution to this is to simulate the stabilizing torque (or force) with a tight spring and damper clamped about the local position. This can be viewed as a temporary change in the range of the joint end limits.

3.4. Balance Mode

While automatic response to gravitational forces make certain motions easy to simulate, this can create problems with coordinated movements. For example, walking no longer becomes a question of merely manipulating the legs in the proper sequence, but also of balancing the upper part of the body to keep it from falling over. A simple way to achieve balance is to describe a world-space orientation vector for particular segments, such as the trunk, and apply an external applied force to counteract any motion away from this orientation. Experimentation shows this technique is acceptable in limited cases; whether it will provide realistic balance in all cases has not been explored in depth.

3.5. Hybrid K-D Mode

The freeze and balance modes do help with some of the motion control problems introduced by dynamics, but leave the major problem of determining forces or torques for controlled motion, rather than just stability. An approach to this problem that has been reasonably successful is to describe the desired motion in kinematic terms, as kinematic control functions specifying rotation (for revolute joints) or translation (for sliding joints) over time. The exact same *Virya* interactive interface can be used; the interpretation of the control functions merely changes.

These kinematic descriptions are then used to find the forces and torques applied at specified degrees of freedom. The method used to find these forces and torques is trivially simple, but surprisingly effective. The equations used are

$$delp = des\_pos - pos$$

$$delv = delp / deltime - vel$$

$$ft = delv * m / deltime$$

For sliding joints, *delp* is the difference between the desired position at the next time sample (*des_pos*) and the present position (*pos*). *delv* is *delp* divided by the time between samples (*deltime*) minus the present velocity (*vel*), in other words, the amount the velocity must be altered to achieve the desired position at the next time sample. ft is the estimated force that must be applied to achieve this, and m is the mass of the segment distal to this degree of freedom. For revolute joints, the same formula is used but the velocities are angular, ft is a torque, and m is a moment of inertia. Probably because dynamic analysis is done from 3 to 30 times as frequently as images are displayed, no feedback is needed to achieve smooth motion.

The advantage to this method is that the user can enter motion directions in the intuitive kinematic form for those joints whose particular motion is known (or desired) and yet retain the advantage of dynamics.

3.6. Pure Kinematic Mode

Kinematic mode is separate from the dynamic animation package. *Deva* can operate as a strictly kinematic animation system, in which case the control functions specified by *Virya* are taken to represent the actual desired motion for all degrees of freedom. The dynamic analysis routines are entirely bypassed and fast kinematic animation is possible.

4. Virya User Interface

The *Virya* screen (see Figure 1) consists of three regions. The lower half of the screen contains small joint windows representing each degree of freedom of the system. The upper right quadrant consists of a menu of commands for designing and saving control functions. The upper left quadrant is a large window where control functions can be viewed and altered. The user selects menu items or designates points on the screen by using a graphics tablet and puck. *Virya* runs on an Evans and Sutherland PS300/340 graphics system.

Each joint window contains a label identifying the degree of freedom represented there; e.g. "J4 elbow (1) rz" refers to a z-rotation of the elbow (joint number 1). The line through each window joins the control points that define the curve representing motion control information for that degree of freedom. The user can alter these control points and thus define the shape of the motion control function. By specifying more or fewer points and spacing them differently, such control functions can be used to control both the path of the motion and its velocity (kinematics) or the strength of the force or torque applied (dynamics).

The menu consists of I/O commands, joint commands, vertex commands, and miscellany. The I/O commands are LOADBODY (input a body description including segments, joints, degrees of freedom, and present configuration); LOAD (input a previously created *Virya* file containing motion control information); and SAVE (save the present motion control description in a file). Joint commands are SHOWJ (bring a particular control function into the large window); COPYJ (copy the control function for one degree of freedom to another); ACTIVEJ (designate one control function as modifiable); REDRAWJ (redraw the active control function); and CURVEJ (use the control points to create a cubic interpolatory spline curve). The vertex commands all refer to the degree of freedom that has been designated "active". They are INITV, ADDV, DELV, MOVEV, and LOCV. INITV initializes the control function to a horizontal line along the time axis. ADDV, DELV, and MOVEV add, delete and move control points defining the control curve. LOCV gives the exact numeric value of a point on the screen. The miscellaneous commands are CONFIRM (confirm a change); QUIT (leave *Virya*); and TABLET (a binary switch between inputting control points from the terminal or from the graphics tablet).

The DKFRB (*dynamic / kinematic / frozen / relaxed / balanced*) option allows the user to designate control modes for each degree of freedom and a time span during which the modes are in force. The default control mode for dynamics is dynamic mode.

Virya motion descriptions are stored in an ascii file (which under present conditions makes up for space consumption with user convenience); part of a sample file is shown in Figure 3.

5. Use of Virya with 3-D Keyframing

Because motion of a complex articulated body such as a human figure is sometimes difficult to visualize in terms of angular or translational motion of individual degrees of freedom, it has been found convenient to initially define the motion as a series of keyframes developed by interactively positioning the body on the screen using *Deva*. A sequence of keyframes with the times they should occur are stored in an ascii file (part of such a file is shown in Figure 2). These files can be converted to *Virya* control function files and used to generate control curves which are sampled to drive the animation or called into the *Virya* editor for modification.

These key-frame-derived control functions initially place all degrees of freedom in hybrid K-D mode, but because this almost completely constrains the motion, there would be little point in doing dynamics. Typically the user modifies this file placing many degrees of freedom into relaxed, frozen, or balanced modes to allow dynamics to fulfill its purpose of adding realism.

6. Sample Session

A simple session using *Virya* will be described to illustrate its use. First, the user enters the animation system *Deva* and calls up a previously stored figure, in this case the 24-degree-of-freedom human figure *Joe*. Using dials and the keyboard, five keyframe configurations for Joe are found. These configurations are to occur at 0, 3, 6, 7, and 8 seconds in the animation. They are stored in ascii form in a keyframe file shown in part in Figure 2. The lines with a single number represent the times when the keyframes occur, and the following numbers are positions for each degree of freedom of motion at that time.

The keyframe file is converted using *Deva* to the format needed for interaction with *Virya*. In the keyframe file, positions are grouped by the times when they occur; in the *Virya* file they are grouped by degree of freedom. Figure 3 shows part of the *Virya* file originally derived from the keyframe file, but somewhat modified using *Virya*. Following the control positions (taken from the keyframes) for each degree of freedom are definitions of the states, or modes, over time. Initially, when the file is created from a keyframe file, all degrees of freedom are in the hybrid K-D mode (K) during the default time span (0-100 seconds). (The 4 numbers after the time span are only used in balance mode, where they represent the position vector and the amount of deviation from it allowed.) During the motion described (lifting the legs and swinging the arms), many degrees of freedom (such as the waist) are frozen into their local configuration and others (such as the right knee) are relaxed. The modes of these joints are changed from the default K-D mode.

This *Virya* file was used to drive dynamic analysis and the resultant, predicted motion stored in another, similar *Virya* file. This second, output file is a kinematic description of the dynamically predicted motion, and was sampled to produce the animation shown in Figure 4. Keyframes are approximately in locations (0,0), (1,3), (3,0), (3,2), and (3,5) (row major order). (Actual dynamic analysis was done 300 times per second; not all of these configurations are stored for the kinematic description and display.)

7. Conclusions

The interactive graphical editor *Virya* is used to design and store motion control commands for kinematic or dynamic animation using control functions and control modes. For kinematic animation, the user designs control functions representing positions over time for each degree of freedom. For dynamic animation, control functions may represent either kinematic information (positions over time) or dynamic information (forces or torques over time). To alleviate some of the control problems that accompany the advantages of dynamic animation, the freeze, balance, and relaxed control modes are also available. The kinematic output of dynamic analysis routines and motion control information derived from other higher-level control methods can be stored the form of *Virya* data files. This format is convenient for low-level modification and sampling for animation generation.

References

1. William W. Armstrong and Mark Green, "The Dynamics of Articulated Rigid Bodies for Purposes of Animation," pp. 407-415 in *Proceedings of Graphics Interface '85*, Computer Graphics Society, Montreal (May, 1985).

2. Brian A. Barsky and Spencer W. Thomas, "TRANSPLINE - A System for Representing Curves Using Transformations among Four Spline Formulations," *The Computer Journal*, Vol. 24, No. 3, August, 1981, pp. 271-277.

3. Richard H. Bartels, John C. Beatty, Brian A. Barsky, *An Introduction to the Use of Splines in Computer Graphics*, Technical Report No. UCB/CSD 83/126, Computer Science Division, Electrical Engineering and Computer Sciences Department, University of California, Berkeley, California, USA (August, 1983).

4. Michael Girard and A. A. Maciejewski, "Computational Modeling for Computer Generation of Legged Figures," pp. 263-270 in *Proceedings of ACM SIGGRAPH '85*, **19**, San Francisco, Ca. (July, 1985).

5. Patrick Hanrahan and David Sturman, "Interactive Animation of Parametric Models," pg. 102-111 in course notes for the tutorial in *Introduction to Computer Animation*, ACM SIGGRAPH '85.

6. Doris H. U. Kochanek, Richard H. Bartels, and Kellogg S. Booth, *A Computer System for Smooth Keyframe Animation*, University of Waterloo, Waterloo, Ontario (December 1982).

7. T. J. O'Donnell and Arthur J. Olson, "GRAMPS - A Graphical Interpreter for Real-Time Interactive Three-Dimensional Picture Editing and Animation," pg. 133-142 in *Proceedings of ACM SIGGRAPH '81*, **15**, Dallas, TX (July 1981).

8. Frederic Parke, "Parameterized Models for Facial Expression," *IEEE Computer Graphics and Applications*, **2**, No. 1 (November 1982), pg. 61-68.

Figure 1. The *Virya* Screen

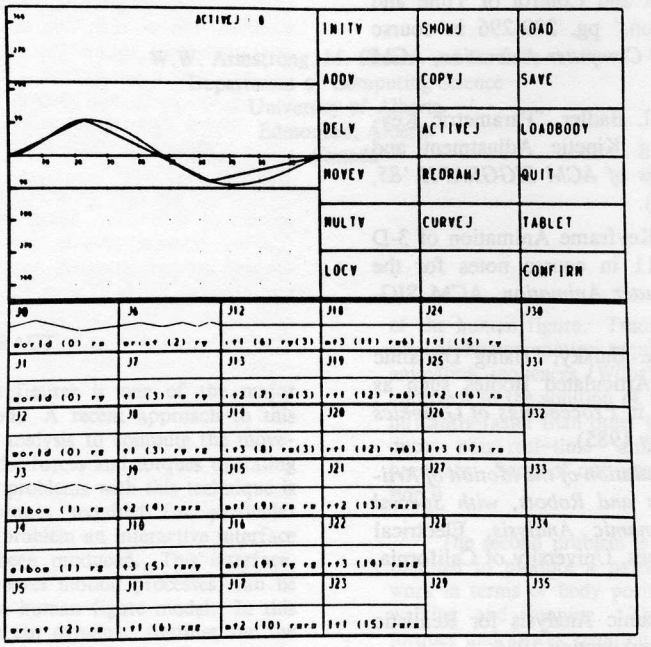

Figure 2. Partial Keyframe File

```
0.000000
180.000000 180.000000 180.000000 0.000000
0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 -180.000000 0.000000
0.000000 0.000000 -180.000000 0.000000
0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000

3.000000
180.000000 180.000000 180.000000 0.000000
0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 -150.000000 0.000000
30.000000 0.000000 -210.000000 0.000000
25.000000 0.000000 30.000000 0.000000
0.000000 0.000000 0.000000 0.000000

6.000000
180.000000 180.000000 180.000000 0.000000
0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 -210.000000 0.000000
30.000000 0.000000 -180.000000 0.000000
25.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000
```

Figure 3. Partial *Virya* File

```
dof 18 Pjnt 7 Type 0 Numv 5  j_hipr
Control 0.000000 0.000000
Control 3.000000 0.523599
Control 6.000000 0.000000
Control 7.000000 0.000000
Control 8.000000 0.000000
States K 0 6 0 0 0  0
States F 6 8 0 0 0  0
Q

dof 19 Pjnt 7 Type 1 Numv 5  j_hipr
Control 0.000000 0.000000
Control 3.000000 0.000000
Control 6.000000 0.000000
Control 7.000000 0.000000
Control 8.000000 0.000000
States F 0 8 0 0 0  0
Q

dof 20 Pjnt 8 Type 0 Numv 5  j_kneer
Control 0.000000 0.000000
Control 3.000000 0.000000
Control 6.000000 0.000000
Control 7.000000 0.000000
Control 8.000000 0.000000
States R 0 8 0 0 0  0
Q
```

9. Craig Reynolds, "Computer Animation with Scripts and Actors," in *Proceedings of ACM SIGGRAPH '81*, **15**, Dallas, TX (July 1981).

10. Craig Reynolds, "Description and Control of Time and Dynamics in Computer Animation," pg. 289-296 in course notes for the tutorial in *Advanced Computer Animation*, ACM SIGGRAPH '85.

11. Scott Steketee and Norman I. Badler, "Parametric Keyframe Interpolation Incorporating Kinetic Adjustment and Phrasing Control," in *Proceedings of ACM SIGGRAPH '85*, **19**, San Francisco, CA (July, 1985).

12. David Sturman, "Interactive Keyframe Animation of 3-D Articulated Models," pg. 102-111 in course notes for the tutorial in *Introduction to Computer Animation*, ACM SIGGRAPH '85.

13. Jane Wilhelms and Brian A. Barsky, "Using Dynamic Analysis for the Animation of Articulated Bodies such as Humans and Robots," pp. 97-104 in *Proceedings of Graphics Interface '85*, Montreal (27-31 May 1985).

14. Jane Wilhelms, *Graphical Simulation of the Motion of Articulated Bodies such as Humans and Robots, with Special Emphasis on the Use of Dynamic Analysis*, Electrical Engineering and Computer Sciences, University of California, Berkeley, CA (July, 1985).

15. Jane Wilhelms, "Using Dynamic Analysis for Realistic Animated Motion," submitted for publication 1986.

16. David Zeltzer, "Motor Control Techniques for Figure Animation," *IEEE Computer Graphics and Applications*, **2**, No. 9 (November 1982), pg. 53-59.

Figure 4. Dynamic Animation Based on *Virya* File