TECHNIQUES FOR INTERACTIVE MANIPULATION OF ARTICULATED BODIES USING DYNAMIC ANALYSIS

David R. Forsey

Computer Graphics Laboratory, Department of Computer Science,
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
Tel: (519) 888-4534, E-Mail: drforsey%watcgl@waterloo.edu

Jane Wilhelms

Computer Graphics & Imaging Laboratory, Computer & Information Sciences Board,
University of California at Santa Cruz, Santa Cruz, CA 95064

ABSTRACT

Manipulating articulated bodies for traditional key-frame computer animation is a laborious task because of the many degrees of freedom, the interaction between segments, and the difficulty of finding and/or constraining local joint positions to specific world space positions or paths. The recursive formulation (by Armstrong)[1] for the dynamics equations for rigid bodies provides a method for dealing with these problems at interactive speeds. Because of complex control issues, dynamics is not yet reasonable for total motion control, but it is a powerful tool for manipulating articulated bodies for keypositioning. This paper describes how dynamic analysis can be used to provide inverse kinematics, directed movement, and kinematic constraints. A system, *Manikin*, for interactive manipulation of articulated figures using dynamics is described.

KEYWORDS: dynamics, kinematic constraints, animation

1. Introduction

Articulated bodies are typically animated using *key-positioning*, i.e., the animator designates occasional key positions and the times at which they occur, and the system animates by interpolating between these positions. Key-positioning systems are generally *kinematic*, based on positions and angles varying over time and rather than objects with mass moving under the influence of forces and torques (*dynamics*). Kinematic control in most animation systems is typically quite low-level; users must deal with each degree of freedom independently despite the fact that, even kinematically, they are interdependent. (Sometimes articulated bodies are animated by copying recorded motion, as in *rotoscoping*; though the motion is very lifelike, it is not a general solution and will not be discussed here.)

Kinematic animation has the advantages that animators think of motion in terms of position changes and kinematic systems are easy to program for simple objects. However, low-level kinematic systems can be difficult to use effectively, particularly when animating articulated bodies such as humans and animals, since: there are many degrees of freedom; the motion of segments is interdependent; the relation between local and world space positions is complex (the *inverse kinematics problem*); constraints are usually non-existent; and it is difficult to specify the *timing* of motion.

Dynamic analysis is an appropriate way to deal with these problems, because it *simulates* rather than *animates*. To use dynamics, objects are modeled as masses connected by joints and acting under the influence of external and internal forces and torques such as gravity, muscles, pushes, and pulls. The relationship between these forces and torques and the body's motion is expressed as the *dynamics equations of motion*, whose simplest form is Newton's Second Law:[17] force is equal to mass times acceleration. It is not necessary for the animator to actually deal with these equations; they can be automatically set up and solved with very little user input beyond the usual kinematic body description.

Dynamics is "natural" in that it mimics how objects move in the real world, and it constrains motion to be realistic (for the reality being modeled). Consider the problems mentioned above. In *Manikin*, the animator need not specify motion at each degree of freedom or worry about interdependence of body segments because a force or torque on one segment will affect other segments to which it is attached. Bodies can be pushed, pulled, twisted, and bent similar to the way one might move a mannequin.

The difficulties involved in using dynamic analysis for animation are threefold. First, dynamic analysis is computationally expensive even using a linear, recursive formulation. Second, the dynamics equations are solved by using numerical methods, and when many degrees of freedom are involved numerical instability problems can arise. Third, and most serious, precisely controlling bodies using dynamics is complex because the largely kinematic world view of the animator must be translated to the internal dynamic world view of the system – where on the body should you pull, how hard, and for how long.

The simplest use of dynamics for animation is as a convenient method for manipulating and constraining

This research was supported by the Natural Sciences and Engineering Research Council of Canada and National Science Foundation Grant number CCR-8606519.

bodies, providing a useful tool for generating the key positions that remain the foundation of most computer animation. This will be referred to as *dynamically-manipulated modeling* and the paper will concentrate on this issue. However, dynamics is also potentially useful for complete motion control without keyframing. This use of dynamics will be referred to as *dynamically-generated motion.* While it has been investigated to some extent,[3,11,19,22] it is not at a point where it can be used for general motion control. The complexities of this use of dynamics are described elsewhere.[18]

Specialised dynamics has also been used to simulated body motion by Lundin[13] at NYIT. In this case, special purpose dynamics routines were used to post-process kinematic animation for greater realism. Special purpose dynamics is also sometimes used in CAD/CAM for engineering analysis, in crash simulations,[21] and for assembling models with positional constraints.[22]

Section 2 will deal with present methods used for motion control of articulated bodies. Section 3 will briefly describe how dynamics is done. Section 4 will deal with how the body can be manipulated using dynamics. Section 5 discusses the computation cost involved. Section 6 discusses the future work. Section 7 concludes the paper.

2. Kinematic Motion Control For Articulated Bodies

The majority of computer animation systems use *key-positioning*. Interpolation between key values is typically done with some consideration for smoothness of motion, using ease-in/ease-out or splines.[16] Expert animators exploiting years of experience are capable of producing lifelike motion using this method; ordinary mortals are likely to find it difficult to choose key positions which will result in realistic animation. Both are likely to become discouraged by the sheer number of values that must be specified, particularly for articulated bodies.

Two capabilities do much to ease the laborious work of keypositioning articulated bodies: *inverse kinematics* and *constraints*. Inverse kinematics refers to the ability to deduce local joint positions that will place a part of the body at a desired worldspace position. This helps to relieve the animator from the tedious job of specifying each degree of freedom separately. Constraints is a very general term; here it refers to the ability to restrain part of the body relative to worldspace allowing the animator to position the figure without, for example, worrying if the feet will penetrate the floor.

2.1. Inverse Kinematics

The complexity of inverse kinematics increases greatly with the complexity of the body. It is possible to find analytical solutions to the inverse kinematics problem for simple bodies by specifying equations describing the configuration of the body and solving them. Analytical solutions have the advantage of providing all possible solutions, but finding such solutions for complex bodies quickly becomes prohibitively expensive.[12] Numerical methods provide one solution from a large, possibly infinite, number, and the problem becomes the selection of an acceptable solution.

Girard and Maciejewski [8,9] have described the use of inverse kinematics to produce locomotion of articulated bodies. In their system, the user designates world space foot positions and inverse kinematics is used to find the angles of the leg to the hip. Interpolation is used between these positions. A pseudo-inverse jacobian matrix (which describes how small changes in local joint positions are caused by small changes in the goal segment's world space position and orientation) is used for inverse kinematics, aided by a vector with which the user describes desired joint angles for the limbs and how important it is to approach the desired values. This method has produced the some of the best computer-animated gaits seen (apart from methods such as rotoscoping).

Korein has suggested an interesting heuristic approach using hierarchical workspaces for each body segment[12] which minimises the position changes necessary at each joint. A problem with this method is that the user has no control over which solution is achieved, and there is no reason to believe the solution will be a natural one for complex articulated bodies.

2.2. Constraints

The second feature of a convenient key-positioning system for articulated body motion is the ability to define positional constraints. To take a simple example, when the animator bends the knees of a figure, the body will find itself suspended in space above the floor. A more desirable solution would have the feet constrained to remain on the floor while the whole body sinks towards the ground. Similarly, when walking, the body rotates first about one foot, then both, then the other.

Constraints on articulated bodies often involve the use of inverse kinematics, because their effect on the body may involve finding local joint positions that accommodate the constraint. If the constraint is a single point, it can sometimes be modeled simply by restructuring the hierarchical tree around the constraint (possibly requiring the addition of a false segment if the constraint is not at a joint).[6]

Badler and Manoochehri[5,14] have developed a system for positioning articulated bodies using kinematic constraints. The user is allowed to designate multiple constraints and rate their importance, a useful feature when they cannot all be resolved. A simple iterative inverse kinematics solution is used.

Witkin[22] models constraints as energy functions solved by following the energy gradient through the model's parameter space. While not dynamic in the sense of solving the equations of motion, the energy constraints do behave rather like forces in that they "pull" the parameters into the desired configuration.

Problems with kinematic constraints, as with kinematic inverse kinematics, involve choosing a desirable solution from among many possible body positions. Isaacs[11] solves sets of equations which explicitly describe the dynamic and kinematic constraints of the models being animated. Unfortunately this approach is, at the moment, too computationally expensive for interactive use.

3. Dynamic Analysis in Manikin

There are many formulations for describing the dynamics equations of motion. The one used in *Manikin* system described in this paper was developed by Armstrong[4] and is recursive and linear ($O(n)$) in the number of joints.

There are some disadvantages to the Armstrong method over other available methods:[10,11,20] it assumes three rotary degrees of freedom at each internal joint (no sliding joints), so non-spherical joints must be constrained by appropriate torques; and the body must be modeled as a tree structure without cycles. However, given the absence of unlimited computational resources, the potential of the Armstrong formulation for real-time interactive simulations makes it the method of choice for a system like *Manikin*.

For details describing the dynamics equations, readers are referred to papers by Armstrong.[1,2,4]

Keeping in mind Newton's Second Law, three types of information are necessary for dynamic analysis: the present state of the body (its mass, mass distribution, configuration, and velocity); the forces and torques acting on the body; and the acceleration the body undergoes. Given the present body state, one can either supply the desired accelerations and solve for forces and torques (of interest in robotics), or supply the forces and torques and solve for accelerations (of interest in computer graphics). Using the accelerations, numerical integration is used to find new velocities and new positions. These positions are used to update the graphics display.

The Euler integration method initially used was soon abandoned because of major instability and inaccuracy problems. A fourth-order Runge-Kutta scheme,[7] in which analysis actually takes place four times for each time interval, vastly improved accuracy and stability. More sophisticated integration methods that provide an error estimate for the integration, such as the Adams-Moulton method or the Sarafyan embedded form of Runge-Kutta, are essential for adaptively altering the step size of the simulation for the best speed and accuracy.

4. User Control in Manikin

The *Manikin* system was developed to explore dynamic manipulation of articulated bodies. Because animators do not normally think in terms of masses, forces, and torques, facilities have been encorporated into *Manikin* to make the system more intuitive. A number of control options are needed for a dynamic manipulation system: automatic calculation of mass information; controllable joint flexibility; ability to pull and push the body about in space; directed motion; positional constraints; and collision detection and response.

Table 1 shows the commands through which the user manipulates the body. It is possible to have multiple forces, torques, goals and constraints applied simultaneously. *Manikin* works as follows. First, a body is loaded and the system is initialised. A control loop checks for user input, calculates the relevant forces and torques, performs dynamic analysis and updates the display. Key positions can be stored for later playback using spline interpolation of joint angles. The user can request the system to run dynamics continuously for a time without asking for control input while storing key-frames, and can also control how often the graphics screen is updated.

Table 1. Manikin User Options

Command	Description
grv	gravity vector
ma	calculate mass data
stop	stop motion
merge	merge to parent segment
freeze	hold relative to parent
jlimits	local joint limits
damp	global damping factor
external forces	pull on body
external torques	torque on one segment
internal torques	muscle torque
goals	attraction point
floors	planar floor

4.1. Designating Body Mass and Configuration

The system contains a reasonable density value for animal tissue (750 $kg/meters^3$). The mass, centers of mass, and inertial tensor for each segment can be automatically calculated from this data. The mass of each segment is the product of the volume of the boundary box times the density of the material. The center of mass of each segment is assumed to lie at the center of the boundary box. The moments of inertia can be calculated from the mass and the dimensions of the boundary box.[17] The products of inertial are zero if the mass is uniformly distributed about the center of mass, as is assumed here.

4.2. Control of Gravity and Velocity

The amount of gravity acting on the body can be altered by changing the gravitational acceleration vector. For manipulation, it is often convenient to remove gravity entirely. At other times it is useful to have, or to have a small amount of it.

It is also convenient to be able to stop the body in a particular configuration, for example, to store it for keyframing or to stabilise it and start moving other segments. Stopping is also useful if motion seems to be getting out of hand, because it removes all the momentum from the system. Stopping the body halts all internal and external motion by zeroing the angular and linear velocity vectors.

4.3. Joint Flexibility

If the segments of the body are left relaxed, they will react quite freely to applied forces and torques. This is generally undesirable (unless one is trying to imitate the limpness of a rag doll). Some way of automatically limiting joint motion is needed. *Manikin* offers four ways in which this is accomplished: *merging* segments so that no motion is possible between them *damping* joints, to reduce their angular velocity; *freezing* joints, dynamically holding them in place locally by applying appropriate constraining forces; and using *joint limits*, so the range of motion is restricted. Freezing and joint limits are described by Wilhelms.[20]

Merging is conceptually (and dynamically) simple. It is useful when total immobility of some joints is desired for example, merging the lower body in a particular stance while manipulating the arms and torso. Merging also reduces the computational cost, because it removes degrees of freedom from the dynamics calculations. For example, in moving the body as a whole one might like to ignore fingers, which contribute little to the total motion. One cannot simply declare a joint immovable by refusing to change the joint angle, as this interferes with the motion predicted by dynamics and produces unrealistic motion as a whole. It is preferable to merge segments so that they are seen by dynamics as one single mass. By restructuring the body tree (the data structure that describes how segments are connected), two or more segments are combined into a single segment whose dynamic characteristics are those of the merged segments without a connecting joint. Segments can be later *unmerged.*

Dampers are forces or torques that restrict motion by an amount proportional to the velocity and a damping constant. Since the motion of an articulated body is complex and interrelated no single damping constant will work for all joints in all situations. In practice, the constant is dependent upon the behaviour, not only of the segment itself, but of the mass and motion of its neighbours. *Manikin* allows the user to specify a global damping factor that attempts to apply reasonable damping torques to all joints or a damping factor for a particular joint. The global damping factor is multiplied by both the local velocity of the segment and the sum of the masses of all segments distal to this joint to get the torque due to damping, which is then subtracted from the accumulated internal torque at the joint. This gives noticeable damping, especially at those light joints where it is most important, but is not a completely acceptable solution. We would prefer to find a simple method for giving a wide range of stiffness to joints, from complete freedom to immobility.

4.4. Local Joint Control

Local positioning (relative to the parent segment) can be achieved either dynamically or kinematically. Kinematics is satisfactory for merely manipulating individual joint angles when no kinematic constraints are involved (other than the constraint that, as tree-structured models, bodies are defined by the position and orientation of each segment relative to its parent).

Joint angles can also be changed using dynamics by applying forces or torques. An external torque affects only one specific segment. An internal torque acts like a muscle with equal and opposite forces acting on the two body segments defining the joint. We have found torques to be of little use for dynamically-manipulated modeling because of the difficulty of determining the appropriate torque to apply to a figure of arbitrary configuration. Simple feedback control strategies have been tried to automate this procedure,[3,20] but they have not been implemented in *Manikin* because an animator seldom would know what specific angle is required and when simple torques are imposed upon a dynamically constrained body, the resulting motion is highly dependent upon the current state of the body and thus is not easily controlled.

Torques have been useful in dynamically-generated motion created with *Manikin* where figures must act as if muscles are controlling motion.

4.5. Kinematic Constraints in Manikin

The recursive formulation of the dynamics equations used in *Manikin* precludes their reformulation to satisfy kinematic constraints in the fashion accomplished by Isaacs.[11] In *Manikin*, positional constraints are modeled by attaching extremely heavy segments, at their center of mass, to the point on the figure to be constrained. These *pseudo-segments* are invisible and no gravitational force is ever applied to them. (Considered this operation as analogous to attaching a segment the mass of the Queen Elizabeth II to the figure.) The extreme inertia constrains the rest of the body to move around the pseudo-segment without greatly effecting the position of the pseudo-segment. An external correcting force (modeled as a spring/damper combination) is applied to the pseudo-segment as necessary to prevent it from slipping away from its constraint point. If the slippage in either position or velocity exceeds a specified constant, the body position is reset to the beginning of the previous time-step and re-tried with a modified correcting force.

This method is convenient and moves the complexity of finding appropriate constraining forces into the dynamics formulation where the appropriate information is available. This method also prevents the oscillations that tend to occur if a simple spring and damper combination is used to hold a point in place.

4.6. Pushing and Pulling the Body

The body can be pushed or pulled into a particular configuration by defining a force (magnitude and direction) and a point on the body where it is applied. This effectively creates a rope or rod with a fixed worldspace orientation which pulls or pushes on the body. The force magnitude and direction can be altered interactively during its application.

4.6.1. Directed Motion (Goal Points)

Simple forces maintain magnitude and direction until changed by the user. A second class of forces, *directed forces*, change both in magnitude and in

Figure 1: Positioning Human Figure in Chair with Forces.

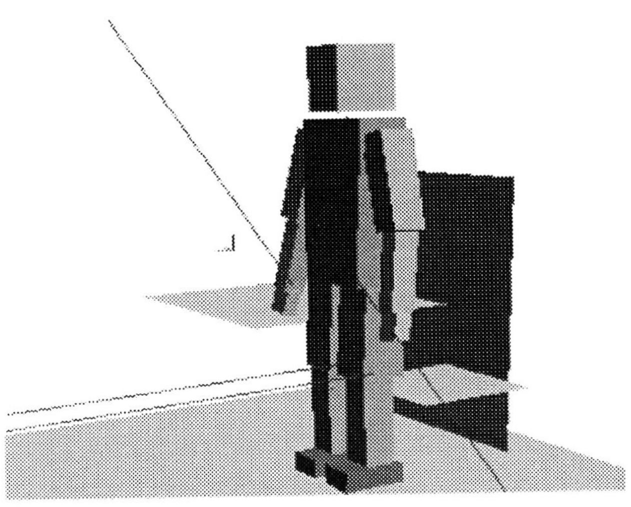

Figure 1a, Initial Position: Forces are attached to bring the knees forward, the waist down and backwards, and the hand off the table. The feet are fixed to the floor.

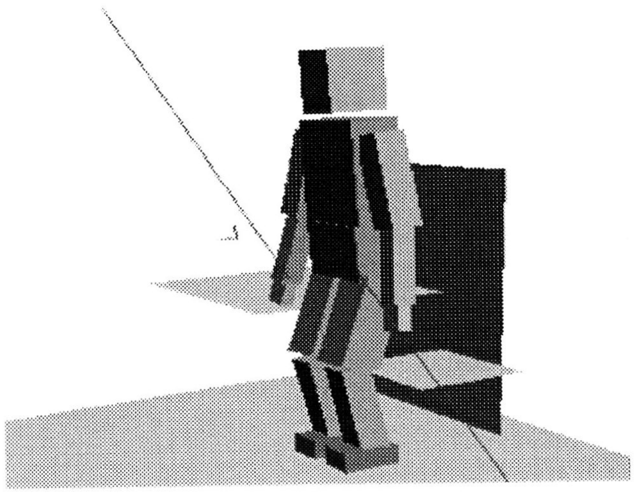

Figure 1b: The forces on knees are removed, the body continues to be pulled into the chair.

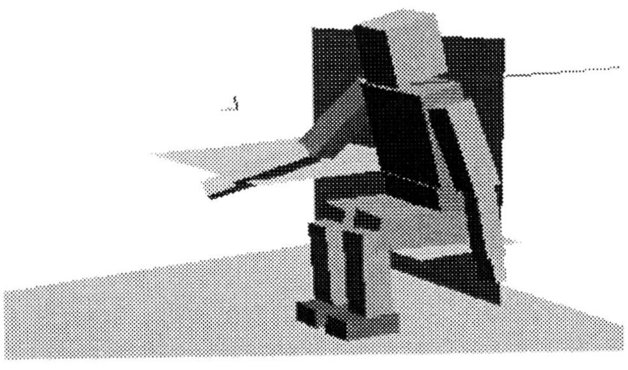

Figure 1c: The figure reaches the chair and all body motion is stopped. A force is attached to the neck to bring the body back into the chair.

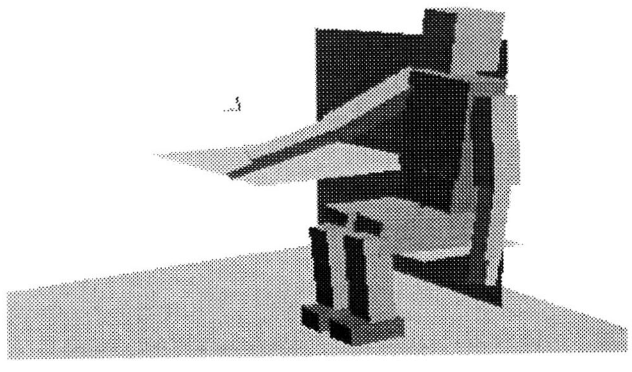

Figure 1d: Final Postion.

direction pulling towards a specified *goal point*. In *Manikin*, the animator species a point on a body segment and its goal position, and the system automatically calculates a force to get it there.

To implement goals, a heavy pseudo-segment (as described in the previous section) is attached to the figure at the point the animator wishes to control. By considering the pseudo-segment as a point-mass and, using inverse-dynamics on this segment alone, forces can be calculated to move the pseudo-segment along a defined path or towards the goal point. Because of the huge disparity between the mass of the pseudo-segment and the rest of the body (enforced by the system), forces arising from the attachment of the pseudo-segment have little effect on the motion of the pseudo-segment and are easily compensated by the correcting force and retry mechanisms.

Because the system is designed for interactive manipulation and dynamics can only provide 5-10 updates/second, forces chosen are large enough to cause rapid motion in terms of actual speed but are reasonable in terms of the motion the user sees. The system achieves a speed of 1 to 2 meters per second in moving toward the goal. The force applied is always in the direction of the goal and, initially, has a magnitude equal to the mass of the segment to which it is applied. A damping force is also applied opposite the direction of motion. As the body point approaches the goal, the force is reduced and damping is increased to prevent overshooting and oscillation. The net result of this approach is that the segment moves directly toward the goal at an acceptable speed and slows down as it closes in.

Alternately, if the goal point is very close, the forces are calculated to push the segment to the goal during the next time step. Thus if the goal point is updated by the user as the dynamic simulation continues, the segment will strive to follow the goal point. Given enough computational power the goal point could be followed closely as the user moved it interactively – giving the illusion of grabbing the model to move it into position.

4.7. Floor Contact

Floor contact is a restricted domain of the general collision problem which is beyond the scope of this paper. Collisions with the floor (and, potentially, other objects) can be simulated using springs and dampers to approximate the collision forces. The spring pushes back with a force or torque proportional to the amount it is compressed and the damper pushes back with a force or torque proportional to the velocity. More detailed descriptions of how to simulate floors (and other types of collision) can be found in Moore. [15]

5. Computational Cost

Manikin was created with the intention of utilising the speed of the recursive dynamics formulation to provide interactive manipulation of articulated bodies to the animator. Table 2 shows the cost of running Manikin with figures of various complexity with dynamic analysis performed with a time step size of 0.01 seconds using a 4th order Runge-Kutta numerical integration. The times are for "chains" of links (i.e. pendulums) except for those called "bushy" in which all links are attached to the root node. These raw times do not include user-time for interaction or graphics processing.

Table 2
Performance of MANIKIN
(8Mhz MC68020/MC68881)

Links	steps/sec	sec/link/step
One	103.09	0.0097
Two	40.32	0.0124
Five	14.45	0.0138
Five (VAX 8600)	53.76	0.00137
Ten	6.86	0.0146
Ten bushy	6.92	0.0145
Twenty	3.33	0.0150
Twenty bushy	3.36	0.0149
Five, 1 force	14.16	0.0141
Five, 3 forces	14.01	0.0143
Five, 5 forces	13.69	0.0146
Five, 5 goals	12.53	0.0159

6. Future Work

The system needs computational power, and more precise generation of automatic forces and torques for goals and collisions. At present, problems sometimes occur, producing oscillations, inappropriate speed, and floors which are too soft or too springy.

The premise in using dynamics for manipulation, rather than numerical inverse kinematics, is that the behaviour of the figure being manipulated will more closely approximate the real world and thus be easier for the animator to use. *Manikin* needs to move out of the laboratory and into the hands of the animator where its usefulness in comparison with other techniques can be evaluated.

7. Conclusions

Dynamics is a useful technique for positioning and moving articulated bodies in a natural way. The complex physical reality of the body and its environment can be specified in sufficient detail to encourage realistic configurations and movement. Dynamics allows simultaneous use of goal-direction, inverse kinematics, positional constraints, and collision simulation. The movements it produces are physically realistic for the model used. Use of dynamics for manipulation can be integrated with dynamically-driven animation.

On the negative side, dynamics is more expensive than kinematic motion control and doesn't provide realtime manipulation of typical human models using presently available hardware.

Figure 2: Positioning Human Figure with Goals and Forces.

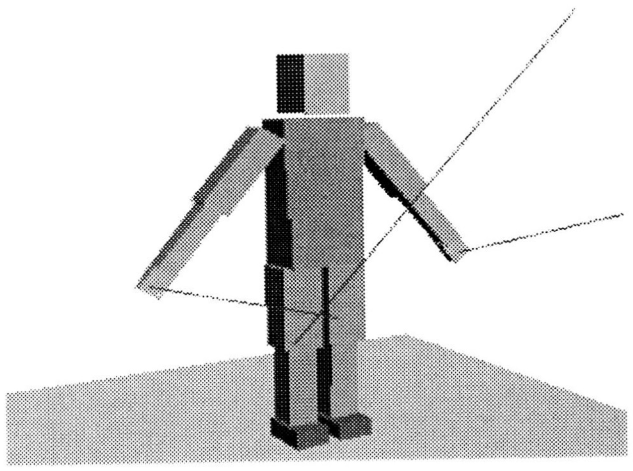

Figure 2a, Initial Position: Forces are attached to bring the arms up and the knee forward. The waist, wrists, neck and non-moving knee are merged while the non-moving foot is fixed to the floor.

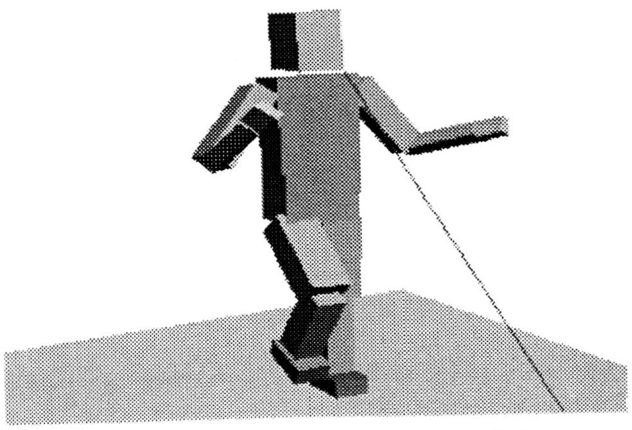

Figure 2b: The forces on the arms and knee are removed. A force is attached to shoulder to bring body forward. The hips are merged with the legs.

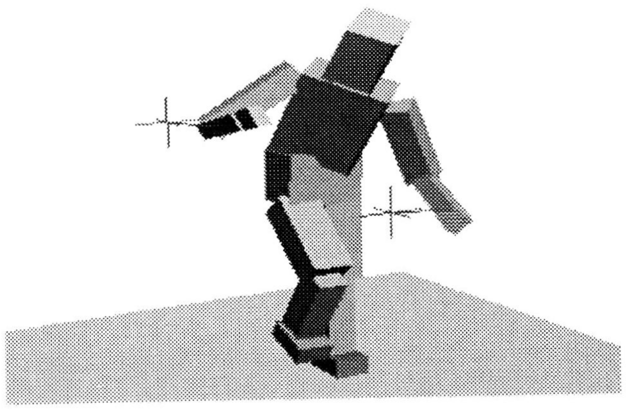

Figure 2c: The shoulder force is removed and goals attached to the left wrist and right elbow.

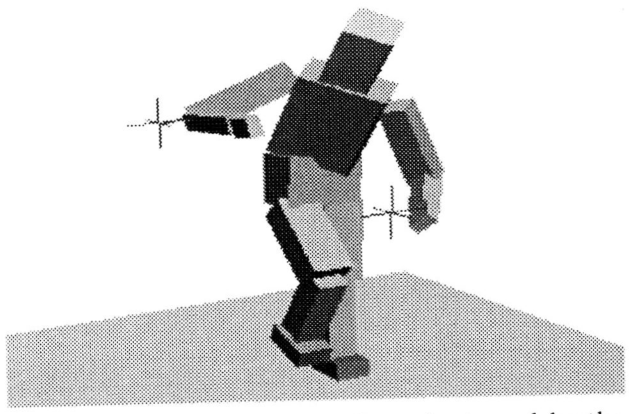

Figure 2d: The motion of the figure is stopped by the user before the goal positions are attained.

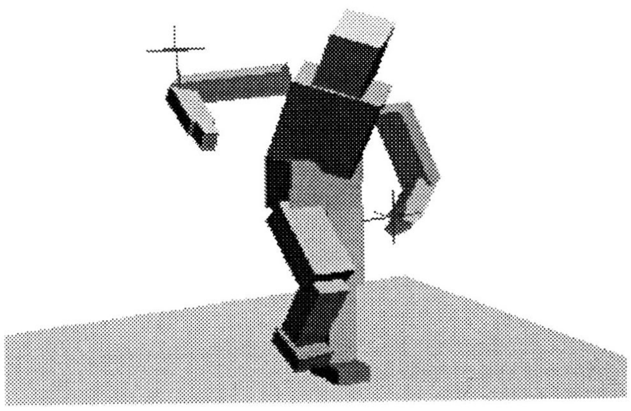

Figure 2e: The goal position for the right elbow is moved.

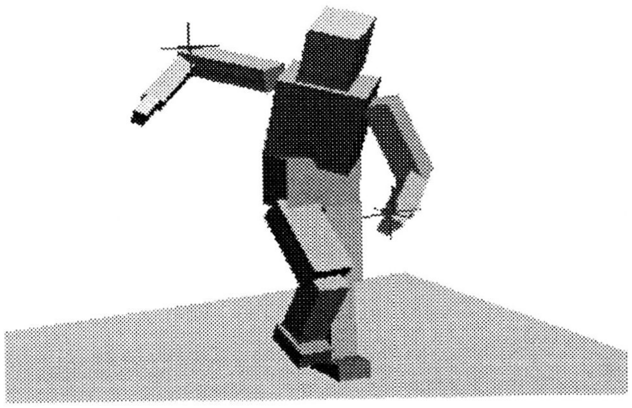

Figure 2f, Final position: Goals have been attained.

8. Acknowledgements

We would like to thank Pat Hanrahan for suggesting the use of dynamics for interactive modeling; Bill Armstrong for the basic dynamics subroutines; Ruby Loo for the **ARIA** graphics system; and Alan Paeth for his image manipulation tools.

9. References

1. William W. Armstrong, Recursive Solution to the Equations of Motion of an N-link Manipulator, *Proceedings Fifth World Congress on the Theory of Machines and Mechanisms*, pp. 1343-1346 Am. Soc. of Mech. Eng., (1979).

2. William W. Armstrong, Mark Green, and Robert Lake, Near-Real-Time Control of Human Figure Models, *IEEE Computer Graphics and Applications* **7**(6) pp. 52-61 (June, 1987).

3. William W. Armstrong, Mark Green, and R. Lake, Near-Real-Time Control of Human Figure Models, *Proceedings of Graphics Interface 86*, pp. 147-151 (May, 1986).

4. William W. Armstrong and Mark W. Green, The Dynamics of Articulated Rigid Bodies for Purpose of Animation, *Proceedings of Graphics Interface '85*, pp. 407-415 (May, 1985).

5. Norman I. Badler, Kamran H. Manoochehri, and Graham Walters, Articulated Figure Positioning by Multiple Constraints, *IEEE Computer Graphics and Applications* **7**(6) pp. 28-38 (June, 1987).

6. Susan Van Baerle, Character Animation: Combining Computer Graphics and Traditional Animation, *SIGGRAPH 86 Tutorial Notes: Computer Animation, 3d Motion Specification and Control*, (July, 1986).

7. S. Conte and C. de Boor, *Elementary Numerical Analysis, 3rd Edition,* McGraw-Hill Book Company, New York (1980).

8. Michael Girard, Interactive Design of 3D Computer-Animated Legged Animal Motion, *IEEE Computer Graphics and Applications* **7**(6) pp. 39-51 (June, 1987).

9. Michael Girard and Antony A. Maciejewski, Computational Modeling for the Computer Animation of Legged Figures, *Proceedings of SIGGRAPH '85*, pp. 263-270 (July, 1985).

10. Roberto Horowitz, Model Reference Adaptive Control of Mechanical Manipulators, PhD Thesis, Mechanical Engineering, University of California, Berkeley, California (May, 1983).

11. Paul M. Isaacs and Michael F. Cohen, Controlling Dynamic Simulation with Kinematic Constraints and Inverse Dynamics, *Proceedings of SIGGRAPH '87*, pp. 215-224 (July, 1987).

12. James U. Korein and Norman I. Badler, Techniques for Generating the Goal-Directed Motion of Articulated Structures, *IEEE Computer Graphics and Applications* **2**(9) pp. 71-81 (November, 1982).

13. Richard V. Lundin, Motion Simulation, *Proceedings of Nicograph 1984*, pp. 2-10 (November, 1984).

14. Kamran H. Manoochehri, *Articulated Figure Positioning by Multiple Constraints and 6-Axis Input,* University of Pennsylvania, The Moor School of Electrical Engineering, Pittsburgh, Pennsylvania (August, 1986). Masters Thesis

15. Matthew Moore and Jane P. Wilhelms, Collision Detection and Response for Computer Animation, *Proceedings of SIGGRAPH '88*, (July, 1988).

16. Craig Reynolds, Description and Control of Time and Dynamics in Computer Animation, *Advanced Computer Animation Course Notes*, pp. 289-296 ACM SIGGRAPH '85, (July, 1985).

17. Dare A. Wells, *Lagrangian Dynamics,* Shaum's Outline Series, McGraw-Hill Book Co., New York (1969).

18. J. Wilhelms, Using Dynamic Analysis for Realistic Animation of Articulated Bodies, *IEEE Computer Graphics and Applications* **7**(6) pp. 12-27 (June, 1987).

19. Jane P. Wilhelms, Virya - A Motion Control Editor for Kinematic and Dynamic Animation, *Proceedings of Graphics Interface 86*, pp. 141-146 (May, 1986).

20. Jane P. Wilhelms and Brian A. Barsky, Using Dynamics Analysis for the Animation of Articulated Bodies such as Humans and Robots, *Proceedings of Graphics Interface '85*, pp. 97-104 (May, 1985).

21. K. D. Willmert, Visualizing Human Body Motion Simulations, *IEEE Computer Graphics and Applications* **2**(9) pp. 35-43 (November, 1982).

22. Andrew Witkin, Kurt Fleischer, and Alan Barr, Energy Constraints on Parameterized Models, *Proceedings of SIGGRAPH '87*, pp. 225-232 (July, 1987).