Dynamics for Computer Graphics: A Tutorial

Jane Wilhelms
University of California, Santa Cruz

ABSTRACT: There is a move in computer graphics toward more correctly simulating the world being modeled in hopes of achieving more realistic and interesting still images and animation. An important component of this move is the use of dynamics, i.e. considering the world as masses acting under the influence of forces and torques. Dynamics can be useful in providing inverse kinematics, constraints, collisions, and, in general, helping produce realistic positions and rates of motion. However, it is computationally expensive, involved to program, and complex to control. This paper is a summary of some useful information necessary for simulating the motion of bodies for computer animation.

What is Dynamics and What can it buy us?

Dynamics refers to the description of motion as the relationship between forces and torques acting on masses. If we treat the objects modeled in computer graphics as masses and apply forces and torques to them, we can use physics to find out the motion these masses should undergo. This motion should mimic the motion that would actually occur to such masses in the real world, hence dynamics *simulates* the motion, rather than just *animating* it.

Dynamics is useful for a number of reasons: it can help restrict motion to that which is realistic in the world modeled; it can automatically find many kinds of complex motion with minimal user input (e.g., motion due to gravity); it can automatically impose many kinds of constraints (e.g., preventing intersection of colliding bodies); it can be used to move complex bodies in natural way; etc.

Dynamics is problematic as a technique for motion control in computer animation because it is (often) computationally expensive, and because controlling the motion is (often) difficult. However, it shows considerable potential for manipulating and animating bodies, and merits further investigation. This paper attempts to provide enough basic information to let anyone simulate simple objects using dynamics.

This work was supported by National Science Foundation grant number CCR-8606519 and UCSC fellowship 660177-19900. An earlier version of this paper was presented at the 1987 USENIX Computer Graphics Workshop.

How to do it

To use dynamics to find the motion of objects, first the *dynamics equations of motion* which describe how *masses* will move under the influence of *forces* and *torques* must be set up. Though there are a number of ways to formulate the equations, they all should give the same solution (they refer to the same world). Second, the equations must be solved for *acceleration.* Third, they must be *integrated* to find the new *velocity* and *position*, given the acceleration. Once the new positions are available, the object can be animated.

There are many books discussing dynamics; unless some specific reference needs to be made, most of the physics in this paper relies upon these references.[8,12,21,23,26] Robotics books are often useful.[16,22] The following references pertaining to use of dynamics for computer animation may also be useful.[1,2,3,27,28,29,30,31]

A right-handed coordinate system with a right-hand screw rule for rotations is assumed, and vectors are premultiplied by matrices to change coordinate frames. (This is more in keeping with robotics and physics usage than computer graphics.) Note that considerable variation in conventions is found in the literature; keep in mind which frame and which screw rule is used.[16,22]

Matrices will be in upper case boldface type (**J**), vectors in lower case boldface (**f**), and scalars in italic type (m). Subscripts will be used to describe the axis for vectors ($\mathbf{c}_x$ is the position of the center of mass along the x-axis), and to further describe the value when necessary ($\mathbf{f}_{grv,i,x}$ is the force of gravity acting on the i-th segment along the x-axis). Superscripts will be used to indicate the frame of reference being used, when necessary ($\mathbf{c}_{i,x}^{j}$ is the above seen in terms of the instantaneous position of the j-th coordinate frame).

Table 1 is a handy reference for the meaning of terms.

Table 1. Meaning of Terms

Matrices	
$\mathbf{J}$	= *inertial tensor matrix*
$\mathbf{R}^{topar}$	= *rotation matrix segment to parent*
$\mathbf{R}^{frompar}$	= *rotation matrix parent to segment*
$\mathbf{R}^{toworld}$	= *rotation matrix segment to world*
$\mathbf{R}^{fromworld}$	= *rotation matrix world to segment*
$\mathbf{D}$	= *rotation matrix seen as direction cosines*
$\mathbf{I}$	= *identity matrix*
$\mathbf{K}$	= *recursive coefficient matrix*
$\mathbf{M}$	= *recursive coefficient matrix*
3D Vectors	
$\mathbf{f}$	= *force*
$\mathbf{f}_{grv}$	= *force due to gravity*
$\mathbf{f}_{ext}$	= *external applied force*
$\mathbf{f}_{son}$	= *force applied by child of a segment through a joint*
$\mathbf{f}_{topar}$	= *force applied onto parent of a segment through a joint*
τ	= *torque*
τ_{grv}	= *torque due to gravity*
τ_{ext}	= *external applied torque*
τ_{son}	= *torque applied by child of a segment through a joint*
τ_{topar}	= *torque applied onto parent of a segment through a joint*
$\mathbf{p}$	= *position*
$\mathbf{v}$	= *linear velocity*
$\mathbf{a}$	= *linear acceleration*
$\mathbf{a}_{grv}$	= *gravitational acceleration*
$\delta\mathbf{u}$	= *change in angular position*
ω	= *angular velocity*
$\dot{\omega}$	= *angular acceleration*
$\mathbf{l}$	= *vector to joint of son segment from parent frame*
$\mathbf{c}$	= *vector to segment center of mass defined in segment frame*
$\mathbf{d}$	= *recursive coefficient*
$\mathbf{f}'$	= *recursive coefficient*
Scalars	
m	= *mass*
δt	= *time step between samples*
I_x, I_y, I_z	= *moments of inertia*
I_{xy}, I_{xz}, I_{yz}	= *products of inertia*

Particles: Point Masses

To illustrate the method on a very simple object, consider the motion of a point mass (a *particle*) in three-dimensions. Dynamics can be done in two dimensions and it's much easier, but also much less interesting.

Some simple bodies can be adequately modelled using dynamics on point masses. It is also possible to effectively model flexible bodies dynamically as masses connected by springs.[13,18] Flexible bodies can also be modelled using elasticity theory.[25]

Information Needed

INVARIANT INFORMATION The only extra piece of constant information necessary to dynamically animate particles is the *mass* of the particle. (Dynamics can be done on a particle of changing mass but it's probable that, for computer graphical purposes, constant mass is a reasonable assumption.)

VARIABLE INFORMATION Variable data needed for dynamically animating particles include the present position **p** (a 3D vector representing x, y, and z-coordinates) and the present velocity **v** (also a 3D vector representing the present motion of the particle). (Again, other coordinate systems could be used, but the Cartesian x,y,z system seems reasonable.) The fact that three numbers are needed to specify the position implies that the particle has three degrees of freedom of motion.

Also needed are the force **f** (a 3D vector with components pulling along the x, y, and z-axes) being applied. If a number of forces are pulling at once, the vectors representing the individual forces are added to get a net force.

Equations

According to Newton's Second Law, the dynamics of a particle can be stated as

$$\mathbf{f} = m\,\mathbf{a} \qquad 1$$

where **f** is the force (a 3D vector representing the components of the force along each Cartesian axis) acting on the particle, m is the mass of the particle, and **a** is the acceleration that the particle will

undergo. Typically, force is in *Newtons* (*kilograms·meters/second*^{2}), mass is in *kilograms*, and acceleration is in *meters/second*^{2}.

This vector equation really represents three scalar equations, one for each Cartesian axis. These three equations are

$$f_x = m\, a_x \tag{1a}$$

$$f_y = m\, a_y \tag{1b}$$

$$f_z = m\, a_z \tag{1c}$$

The Second Law Equation is a differential equation, because the acceleration is a function of time. The equation can be also stated

$$\mathbf{f} = \mathrm{m}\frac{\mathrm{d}\mathbf{v}}{\mathrm{dt}} \tag{2}$$

because the acceleration is really the derivative (rate of change) of the velocity over time. (The force may also vary with time.) Similarly, it could be stated

$$\mathbf{f} = \mathrm{m}\frac{\mathrm{d}^2\mathbf{p}}{\mathrm{dt}^2} \tag{3}$$

because the velocity is the derivative of the position over time, and, thus, acceleration is the second derivative of the position.

Solving the Equations of Motion

If the user provides the particle mass and the applied force, it is easy to see that solving these three independent equations will give the acceleration that the particle will undergo along each Cartesian axis, by dividing by the mass. For example, for x

$$a_x = \frac{f_x}{m} \tag{4}$$

Integrating to Find the New Velocity and Position

The above equations provide the acceleration, but not the position. A simple method of integrating this equation is referred to as the *Euler method.* It is a numerical (= approximate) solution whose inaccuracy increases as does the acceleration or the time steps used. The Euler method assumes the present velocity (e.g. at time i) is known and the velocity a bit (δt) further on in time is needed. The new velocity will be

$$\mathbf{v}_{i+1} = \mathbf{v}_i + \mathbf{a}_i \delta t \qquad 5$$

Again, this is really three separate equations. For example, for x

$$v_{i+1,x} = v_{i,x} + a_{i,x} \delta t \qquad 5a$$

This gives an approximation of the new velocity, but only an approximation. See Figure 1, which represents how the velocity is really changing over time. A point on the curve at time t_i represent the velocity at a particular time t_i. The arrow leaving the curve at a tangent represents the instantaneous acceleration at that time, found from Equation 4 in the previous section. The Euler approximation amounts to moving δt units along the time axis and assumes the new velocity is where the arrow is at time $t_i + \delta t$. Note this is not on the curve. How far off the curve it is depends on how much the curve is bending away from the arrow and how large δt is. With reasonably small time steps this method can be used without too much trouble arising.

Given the new velocity, the new position can be found by the same method

$$\mathbf{p}_{i+1} = \mathbf{p}_i + \mathbf{v}_i \delta t + \frac{1}{2} \mathbf{a}_i \delta t^2 \qquad 6$$

Again, this is really three separate equations. For x,

$$p_{i+1,x} = p_{i,x} + v_{i,x} \delta t + \frac{1}{2} a_{i,x} \delta t^2 \qquad 6a$$

The same inaccuracy problem occurs when finding the new position. There are better methods of numerical integration, such as the Runge-Kutta method.[6]

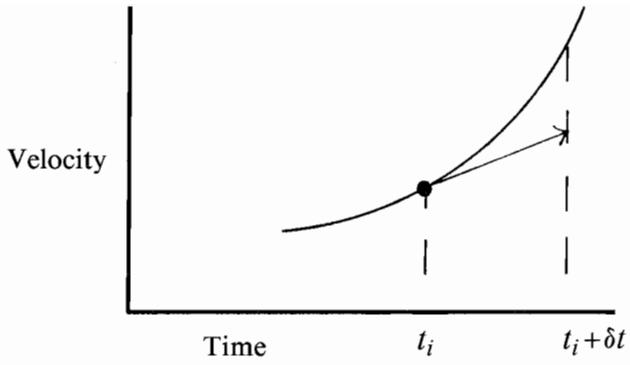

Figure 1.

Controlling the Motion

Controlling particles is fairly simple. The user need only supply an external force as one 3D vector, or as a normalized (length 1) 3D vector representing the direction of the force and a scalar magnitude representing the strength of the force. It might be desirable to have gravity act on the particle. The gravitational force $\mathbf{f}_{grv}$ is the product of a gravitational acceleration (about 9.81 $meters/second^2$ on earth, acting toward the earth's center) times the particle mass.

Others forces that might be of interest involve collisions with other objects, and are discussed briefly below.

Rigid Bodies: Extended Masses

Assuming that the objects are extended masses, not point masses, complicates things considerably. For now, assume that these extended masses are rigid, and do not change shape or mass.

Information Needed

INVARIANT INFORMATION The necessary constant information includes the mass m of the object, the *center of mass* $\mathbf{c}$ of the object (the balance point), and a way to describe how the mass is distributed about the center of mass. The mass is a simple scalar.

The center of mass is a 3D vector describing a location in space. This could be a vector from the origin of the world (inertial) space within which all objects are placed, but then this vector would change as the object moved. It is better to assume some *local* coordinate frame fixed to the object and describe the center of mass relative to this local frame. As long as the location of the local frame relative to the world frame is known, it is easy to find the world space center of mass if necessary. Typically such a local frame is already used to describe the geometry of objects for graphics. If the center of mass is not known, picking a point roughly at the center of the object generally is sufficient.

Describing the mass distribution can be more complex, particularly if the mass of the objects is not symmetrically distributed about the local coordinate frame. Mass distribution for symmetrical objects requires three *moments of inertia*, one about each axis.

$$I_x = \int (y^2 + z^2) dm \qquad 7a$$

$$I_y = \int (x^2 + z^2) dm \qquad 7b$$

$$I_z = \int (x^2 + y^2) dm \qquad 7c$$

i.e., the sum of the masses of each particle making up the object (dm) multiplied by the square of its perpendicular distance from the axis.

For symmetrical bodies there are simple ways of calculating these moments of inertia. For example, for a box with origin at the center of mass with dimensions c in x, b in y, and a in z, the moments of inertia around the center of mass are[26]

$$I_x = \frac{1}{12} m(a^2 + b^2) \qquad 8a$$

$$I_y = \frac{1}{12} m(a^2 + c^2) \qquad 8b$$

$$I_z = \frac{1}{12} m(b^2 + c^2) \qquad 8c$$

Often this bounding box is a close enough approximation.

If the object is not symmetrical, the three *products of inertia* must also be found. (For objects symmetrically arranged around a center of mass, the products of inertia relative to the center of mass are all zero.) The products of inertia are shown below. (Note that occasionally products of inertia are predefined as negative quantities, making terms involving them change sign in the dynamics equations.)[12]

$$I_{xy} = \int xy \, dm \qquad 9a$$

$$I_{xz} = \int xz \, dm \qquad 9b$$

$$I_{yz} = \int yz \, dm \qquad 9c$$

The units for moments and products of inertia in the SI (metric) system are *kilogram – meters*^{2}.

Often the moments and products of inertia are arranged in a 3×3 *inertial tensor* matrix for using in the equations of motion.

$$\mathbf{J}_k = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix} \qquad 10$$

Estimating the moments of inertia for simple symmetrical bodies is not hard. It is also quite straightforward to find the moments and products of inertia about any axes or points in space given this information. For example, to find these values for the axes of a second coordinate system whose major axes are parallel to the previously described frame originating at the center of mass, but displaced by $(\Delta x, \Delta y, \Delta z)$, the new values are

$$I'_x = I_x + m(\Delta y^2 + \Delta z^2) \qquad 11a$$

$$I'_y = I_y + m(\Delta x^2 + \Delta z^2) \qquad 11b$$

$$I'_z = I_z + m(\Delta x^2 + \Delta y^2) \qquad 11c$$

$$I'_{xy} = I_{xy} + m\,\Delta x\,\Delta y \qquad 12a$$

$$I'_{xz} = I_{xz} + m\,\Delta x\,\Delta z \qquad 12b$$

$$I'_{yz} = I_{yz} + m\,\Delta y\,\Delta z \qquad 12c$$

Suppose that the new frame is both rotated and translated from the old. (Note that this case be may avoidable in simulations, however, it is worth examining.) The values relative to the new frame (f'') can be derived from the original center of mass frame (f'') in two steps. The first step is to create a frame (f') which is parallel to the original frame f but displaced so that its origin is at the desired location of the origin of frame f''. This new frame f' is created by using Equations 11 and 12. Next the values for the rotated and translated frame f'' can be derived from f' as follows. One needs the *direction cosines* describing how the new x-axis is related to the old x-axis (a_{00},a_{10},a_{20}), the new y-axis to the old y-axis (a_{01},a_{11},a_{21}), and the new z-axis to the old z-axis (a_{02},a_{12},a_{22}).[17,26]

The 3×3 rotation matrix representing the orientation of a frame can be thought of as three direction cosine (column) vectors defining the axes of the frame. Column 0 represents the new x-axis, column 1 the new y-axis, and column 2 the new z-axis. (To be convinced of this relationship, try transforming the original axes vectors ((1,0,0),(0,1,0),(0,0,1)) by the rotation matrix.)

$$\mathbf{D}_k = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \qquad 13$$

Now, the new moments and products of inertia (Ix'', etc.) given those found above in a frame parallel to that centered on the center of gravity (Ix', etc.) are found by

$$\mathbf{I}'' = \mathbf{D}_k\ \mathbf{I}'\ \mathbf{D}_k^T \qquad 14$$

which, expanded, is

$$I''_x = I'_x a_{00}^2 + I'_y a_{01}^2 + I'_z a_{02}^2 \qquad 15a$$
$$- 2I_{xy}a_{00}a_{01} - 2I_{xz}a_{00}a_{02} - 2I_{yz}a_{01}a_{02}$$

$$I''_y = I_x a_{10}^2 + I_y a_{11}^2 + I_z a_{12}^2 \quad 15b$$
$$- 2I_{xy} a_{10} a_{11} - 2I_{xz} a_{10} a_{12} - 2I_{yz} a_{11} a_{12}$$

$$I''_y = I_x a_{20}^2 + I_y a_{21}^2 + I_z a_{22}^2 \quad 15c$$
$$- 2I_{xy} a_{20} a_{21} - 2I_{xz} a_{20} a_{22} - 2I_{yz}\, a_{21}\, a_{22}$$

$$I''_{xy} = (a_{00}a_{11} + a_{01}a_{10})I_{xy} + (a_{00}a_{12} + a_{02}a_{10})I_{xz} \quad 15d$$
$$+ (a_{01}a_{12} + a_{02}a_{11})I_{yz} - (a_{00}a_{10}I_x + a_{01}a_{11}I_y + a_{02}a_{12}I_z)$$

$$I''_{xz} = (a_{00}a_{21} + a_{01}a_{20})I_{xy} + (a_{00}a_{22} + a_{02}a_{20})I_{xz} \quad 15e$$
$$+ (a_{01}a_{22} + a_{21}a_{02})I_{yz} - (a_{00}a_{20}I_x + a_{01}a_{21}I_y + a_{02}a_{22}I_z)$$

$$I''_{yz} = (a_{10}a_{21} + a_{11}a_{20})I_{xy} + (a_{10}a_{22} + a_{20}a_{12})I_{xz} \quad 15f$$
$$+ (a_{11}a_{22} + a_{12}a_{21})I_{yz} - (a_{10}a_{20}I_x + a_{11}a_{21}I_y + a_{12}a_{22}I_z)$$

This may seem like a drastic amount of trouble, but actually it can be programmed as subroutines and made invisible to the user. In fact, approximate quantities can be found by merely providing a boundary box around the center of mass and assuming some default density to the material (e.g. 1 *kilogram/meter*^{3}). The dimensions of the boundary box (a,b,c) can be used to find the volume ($a \times b \times c$ *meters*^{3}). Multiplying the density by the volume gives the mass. The center of mass can be assumed to be the center of the bounding box. The moments of inertia around the center of mass can be found from Equation 8 above; the products of inertia will be zero. If the frame is not at the center of mass but translated away from it, Equations 11 and 12 can be used to find the moments and products of inertia relative to this new frame. If the frame is rotated, Equation 14 (or its equivalent Equation 15) can be used to find the new moments and products of inertia.

VARIABLE INFORMATION Rigid bodies have six degrees of freedom. Three are translational degrees of freedom as with point masses. Three are rotational degrees of freedom describing how the body is oriented toward some frame of reference. Assuming a local coordinate frame fixed to the object, the translational degrees of freedom may represent displacement relative to a fixed inertial world frame axes, or along the present local frame axes (or any other axes). Similarly, the orientation degrees of freedom may

refer to rotation about the world space axes, or about the present local frame axes.

Here the order of rotations will be fixed as x-rotation, then y-rotation, then z-rotation. These are called *Euler* rotations. Euler rotations can come in various orders; using the order x, then y, then z, the x-rotation is relative to the original x-axis, the y-rotation is about the y-axis created by the x-rotation, and the z-rotation is about the z-axis created by the former two rotations. It is often sensible to assume the local z-axis represents the longitudinal axis of the body, when there is an obvious longitudinal axis.

The other variant information involves the forces **f** and torques τ which cause motion to occur. If a number of forces are acting on the body, their total translational effect can be found by merely summing them. The center of mass of the body will move translationally as if it were a particle mass influenced by one net force.

A torque is similar to a force, except that it causes a rotational motion about a particular axis. Torques can be represented as 3D vectors describing their components about an x, y, and z-axis. Torque vectors' net action can be found by summing them.

If all forces are applied at the center of mass, they produce no torque; however, a force acting at a point on the body other than the center of mass will also cause a torque. To find a torque about a coordinate frame's axes due to a force **f** (f_x,f_y,f_z) applied at point **p** (x,y,z) (both defined relative to this frame), compute the cross product:

$$\tau = \mathbf{p} \times \mathbf{f} \tag{16}$$

This gives:

$$\tau_x = f_z y - f_y z \tag{16a}$$

$$\tau_y = f_x z - f_z x \tag{16b}$$

$$\tau_z = f_y x - f_x y \tag{16c}$$

Often the motion of the rigid body in terms of its body-fixed frame is desired and the point of application of the force is in terms of this frame, but the external force is more naturally given in terms of the world inertial frame. An external force (or any

other quantity) defined in the inertial frame can be converted into the local frame by multiplying it by the matrix defining how the world frame is oriented as seen from the local frame. This matrix is the inverse (which in this case is the transpose) of the matrix defining how the local frame is defined relative to the world frame.

If multiple forces and torques are acting upon a body, these six important net values (three force, three torque) can be easily found (for motion relative to the local frame) by summing the forces (in local terms) to find the net **f**, finding the torques caused by these forces using Equation 16, and summing these torques with any active pure torques to find the net torque (τ). This effectively removes the torque component from the forces. After this is done, the net force effectively is applied to the origin of the local frame. The local frame need not be at the center of mass for this to be true.

Equations

With rigid bodies, dynamics become somewhat less trivial. There are a number of formulations, and here a brief description of the Euler method is presented. The Euler method is, perhaps, one of the more intuitive formulations. The Armstrong method for articulated body dynamics presented in the next section can, of course, also be used for a simple non-articulated body.

The Euler method creates six equations: three are the translational equations of motion relating the linear acceleration and mass to the force, and three are the rotational equations of motion relating the angular acceleration and mass distribution to the torque. Altogether, they specify the behavior of the six degrees of freedom of a free rigid body. Much of this discussion comes from Wells.[26]

The 3D vector version of the translational equations describing the motion of the center of mass is familiar, e.g.

$$\mathbf{f} = \mathrm{m}\,\mathbf{a} \qquad 17$$

or, as 3 scalar equations,

$$f_x = ma_x \;\; ; \;\; f_y = ma_y \;\; ; \;\; f_z = ma_z \qquad 17\text{a,b,c}$$

where **f** is the net force and **a** is the linear acceleration of the center of mass relative to inertial space. This is because the center of mass acts as if the whole body mass were located there and all forces are acting at that point. The effect of these forces on rotation comes out in the rotational equations.

The force and linear acceleration could be expressed relative to any axis, e.g. the instantaneous local axis fixed to the body, by taking the proper components. However, they must both be expressed relative to the same frame. This is an important point, if the user inputs the force $\mathbf{f}^w$ relative to the inertial world coordinate frame and wants the linear acceleration $\mathbf{a}^l$ in terms of the local frame, direction cosines (= rotation matrices) can be used to find the components of the world space force relative to the local frame. Another way of looking at this is to take the dot product of the force vector (f_x,f_y,f_z) with each axis vector (e.g., for the x-axis, (a_{00},a_{10},a_{20})). The force component along the local x-axis would be

$$f_x^l = f_x^w a_{00} + f_y^w a_{10} + f_z^w a_{20} \qquad 18$$

The rotational equations for motion about the center of mass are also quite simple, assuming the products of inertia are zero and that *either* the local frame is at the center of mass or the origin of the local frame is fixed in world space. In this case,

$$\tau_x = I_x\dot{\omega}_x + (I_z - I_y)\omega_y\omega_z \qquad 19a$$

$$\tau_y = I_y\dot{\omega}_y + (I_x - I_z)\omega_x\omega_z \qquad 19b$$

$$\tau_z = I_z\dot{\omega}_z + (I_y - I_x)\omega_x\omega_y \qquad 19c$$

where all values are assumed relative to the local body-fixed frame. ω is the angular velocity of the local frame relative to the inertial frame but expressed in terms of local frame axes. $\dot{\omega}$ is the angular acceleration. ω is typically in *radians/second* and $\dot{\omega}$ in *radians/second*^{2}. τ is the torque acting on the body.

Should one not be so lucky, the more general form of the equations is below. All values are relative to a single coordinate frame, which may be an inertial frame, but is (for our case) probably the instantaneous position and orientation of a body-fixed local coordinate frame. **c** refers to the location of the center of mass relative to this frame. **a** refers to linear acceleration of

the origin of this frame. All other values are in terms of this frame as well. First, the vector form of the equations is

$$\mathbf{f} = m\mathbf{a} - m\mathbf{c} \times \dot{\boldsymbol{\omega}} + m\boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{c}) \quad (20)$$

$$\boldsymbol{\tau} = \mathbf{J}\dot{\boldsymbol{\omega}} + m\mathbf{c} \times \mathbf{a} + \boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega} \quad (21)$$

Expanded this gives

$$f_x = m(a_x - c_x(\omega_y^2 + \omega_z^2) + c_y(\omega_y\omega_x - \dot{\omega}_z) + c_z(\omega_x\omega_z + \dot{\omega}_y)) \quad (20a)$$

$$f_y = m(a_y + c_x(\omega_x\omega_y + \dot{\omega}_z) - c_y(\omega_x^2 + \omega_z^2) + c_z(\omega_y\omega_z - \dot{\omega}_x)) \quad (20b)$$

$$f_z = m(a_z + c_{a_x}\omega_z - \dot{\omega}_y) + c_y(\omega_y\omega_z + \dot{\omega}_x) - c_z(\omega_x^2 + \omega_y^2)) \quad (20c)$$

$$\tau_x = m(a_zc_y - a_yc_z) + I_x\dot{\omega}_x + (I_z - I_y)\omega_y\omega_z + I_{xy}(\omega_x\omega_z - \dot{\omega}_y) - I_{xz}(\omega_x\omega_y + \dot{\omega}_z) + I_{yz}(\omega_z^2 - \omega_y^2) \quad (21a)$$

$$\tau_y = m(a_xc_z - a_zc_x) + I_y\dot{\omega}_y + (I_x - I_z)\omega_x\omega_z + I_{yz}(\omega_y\omega_x - \dot{\omega}_z) - I_{xy}(\omega_y\omega_z + \dot{\omega}_x) + I_{xz}(\omega_x^2 - \omega_z^2) \quad (21b)$$

$$\tau_z = m(a_yc_x - a_xc_y) + I_z\dot{\omega}_z + (I_y - I_x)\omega_x\omega_y + I_{xz}(\omega_y\omega_z - \dot{\omega}_x) - I_{yz}(\omega_x\omega_z + \dot{\omega}_y) + I_{xy}(\omega_y^2 - \omega_x^2) \quad (21c)$$

Solving and Integrating the Equations

Note the equations are simple to solve in the direct direction, given accelerations, find the forces and torques; however, here one wants to find linear and angular accelerations given forces and torques (assuming the present position and velocity are known). Thus, there are (at worst) six equations in six unknowns (a_x,a_y,a_z, $\dot{\omega}_x$,$\dot{\omega}_y$, $\dot{\omega}_z$). Gaussian elimination or similar methods can be used to solve for accelerations. The Euler method of numerical integration is sometimes adequate for integration.

Controlling the Motion

Rigid bodies can be controlled by a combination of applied torques and applied forces. Applied torques cause a rotational motion about the axes they refer to (e.g. the body-fixed local frame) and require a 3D vector. Applied forces involve a 3D force vector (as with point masses) and also a 3D location vector

describing where the force is being applied. Typically the location vector will be specified in the local frame.

Net force is found by summing force vectors irrespective of point of application. Net torque is found by taking the torque caused by these forces (using Equation 16) as well as any pure torques and summing these. These six values are used in the six equations of motion.

Articulated Bodies

Articulated bodies can be thought of as rigid segments connected together by joints. There are numerous formulations of the dynamics equations for rigid bodies, but again, they all come down to the same thing. Some possible choices are the Euler equations,[26] the Gibbs-Appell formulation,[14,21,27] the Armstrong recursive formulation,[1,2] and the Featherstone recursive formulation.[7] The Euler method does not deal very nicely with constraints at joints. The Gibbs-Appell equations, described in appalling detail elsewhere,[27] have been used for graphical simulation but in a non-recursive form that is $O(n^4)$ in complexity. This is computationally untenable, but if a recursive formulation could be found it still might be a reasonable method, as it allows considerable flexibility in designing joints. The Featherstone method is recursive and linear in the number of joints, and is flexible in the types of joints, and is worth exploring.

The Armstrong method is recursive and linear in the number of joints and will be described in some detail here. It has the slight disadvantage that it can only accommodate bodies with freedom of movement relative to the world (6 degrees of freedom from the body tree root and the world) and three rotary degrees of freedom at each joint. Also bodies must be representable as tree structures. This is fine for most animalistic figures, and further constraints can be applied on top of the basic dynamics using external forces or other more devious methods. The Armstrong method has been used in graphics modeling and the author is using it at present, using a modified version of code originally provided by Bill Armstrong and Mark Green at the University of Alberta.

The Armstrong method can be thought of as an extension of the Euler equations with multiple segments (connected rigid bodies). Again, there are at most six equations for each joint (one for each degree of freedom of motion). The real difference comes in the components of the torques and forces. One must consider not only applied forces and torques on the segment, but forces percolating down onto the segment from the child segments, and reaction forces at the joint between the segment and its parent. The following equations are described in detail in Armstrong and Green's 1985 paper.[2] They are repeated here in slightly different terms to show their equivalence to the Euler formulations above.

Information

The same information is needed for articulated bodies made of rigid segments as for non-articulated rigid bodies, plus a tree describing how the segments are connected together. Each segment can have at most one parent and zero or more children. For convenience, the local frame should originate at the proximal (nearer to the root) joint of a segment and the longitudinal axis of the segment should be the local z-axis. If this convention is followed, the third Euler rotation at a joint will always cause a longitudinal rotation.

In simulating people and other animals, biology and biomechanics books are useful sources of information on the nature of organic tissue, dimensions, etc. NASA's book on anthropometry is also a handy reference.[20]

Armstrong Formulation

The Armstrong formulation is based upon the six Euler equations described above as Equations 20 and 21. Everything is expressed in terms of the instantaneous location and orientation of the frame of the i-th segment.

$$\mathbf{f}_i = m_i\mathbf{a}_i - m_i\mathbf{c}_i \times \dot{\boldsymbol{\omega}}_i + m_i\boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{c}_i) \qquad 22$$

$$\boldsymbol{\tau}_i = \mathbf{J}_i\dot{\boldsymbol{\omega}}_i + m_i\mathbf{c}_i \times \mathbf{a}_i + \boldsymbol{\omega}_i \times \mathbf{J}_i\boldsymbol{\omega}_i \qquad 23$$

In Equation 22, the first term on the right comes from the linear acceleration of frame i, the second from the angular acceleration of frame i, and the third term from the centrifugal force due to rotation of the frame. In Equation 23, the first term on the right is the rate of change of the angular momentum, the second is due to the acceleration of the frame. and the third is due to the rotation of the frame.

If the body is articulated, the influence of neighboring segments must be considered as well as external applied pushes and pulls. The force can be further broken up into

$$\mathbf{f}_i = m_i \mathbf{a}_{grv,i} + \mathbf{f}_{ext,i} + \sum \mathbf{f}_{son,i} - \mathbf{f}_{topar,i} \qquad 24$$

All these are expressed in terms of the i-th local frame. $m\mathbf{a}_{grv,i}$ $(= \mathbf{f}_{grv,i})$ is the force due to gravity acting on the mass of segment i. $\mathbf{f}_{ext,i}$ is the net external force acting on frame i. $\mathbf{f}_{son,i}$ is the net force due to each son of segment i acting on segment i through the joint joining them. $\mathbf{f}_{topar,i}$ is the net force that segment i is applying to its parent. This force is applied by the parent back onto the son to keep the two from separating (as described in Newton's Third Law), so it is negative in this equation.

The torques acting on segment i can also be broken into components

$$\tau_i = m_i \mathbf{c}_i \times \mathbf{a}_{grv,i} + \tau_{ext,i} + \sum (\tau_{son,i} + \mathbf{l}_{son} \times \mathbf{f}_{son}) - \tau_{topar,i} \qquad 25$$

The first term on the right, $m_i \mathbf{c}_i \times \mathbf{a}_{grv,i}$ $(= \tau_{grv,i})$, describes the effect of gravity acting on the center of mass of the segment and causing a torque at the proximal joint. $\tau_{ext,i}$ is the net external torque applied to the segment i. $\tau_{son,i}$ is the torque that a son of segment i is applying to segment i at the joint between them. $\mathbf{l}_{son} \times \mathbf{f}_{son}$ is the torque due to the force a son segment is applying onto segment i. $\mathbf{l}_{son}$ is a vector from the origin of segment i to the joint between segment i and its son son in terms of frame i. $\tau_{topar,i}$ is the torque that segment i is applying to its parent segment. Forces acting directly on segment i are assumed to have been analyzed to find their torque component acting on segment i and this added to the applied external torques τ_i.

Finally, one more vector equation is needed that relates the acceleration of the parent and son segments. The right side describes the acceleration of the son's proximal hinge due to the

the acceleration, angular acceleration, and centrifugal acceleration of the parent i. All are in terms of the axes of frame i.

$$\mathbf{a}_{son} = \mathbf{a}_i - \mathbf{l}_{son} \times \dot{\omega}_i + \omega_i \times (\omega_i \times \mathbf{l}_{son}) \qquad 26$$

One thing to keep in mind is that though the motion is being described in terms of the axes of frame i, the motion is relative to inertial space, not the parent. That is, the velocity is not relative to the parent, which may also be moving on its own, but an inertial motion that includes the motion of the segment about its joint to the parent plus any motion that parent may be involved in relative to the world.

Solving the Equations Recursively

Because the body is limited to a tree structure, effects of other segments on a particular segment is limited to effects of sons and parent on this segment. This makes it possible to solve the equations recursively. First the linear relationship between angular and linear acceleration, and between linear acceleration and the reactive force on the parent, must be recognized. **K** and **M** are recursive coefficient matrices which relate linear acceleration to angular acceleration ($\dot{\omega}$) and to reactive force on the parent ($\mathbf{f}_{topar}$), respectively. **d** includes other constituents of the angular acceleration and **f′** includes other constituents of the force on the parent. For each segment i,

$$\dot{\omega}_i = \mathbf{K}_i \mathbf{a}_i + \mathbf{d}_i \qquad 27$$

$$\mathbf{f}_{topar,i} = \mathbf{M}_i \mathbf{a}_i + \mathbf{f'}_i \qquad 28$$

Note that the reactive force $\mathbf{f}_{topar,i}$ acting on the parent j of segment i is one of the $\mathbf{f}_{son,j}$ forces seen from this parent (see Equation 24). By some deft maneuvering (described in more detail in Armstrong and Green's 1985 paper), the dynamics equations can be restated using this relationship. The four recursive coefficients for each segment can be found in an inward pass from the leaves of the body tree to the root. Then this information can be used to find the accelerations of each segment from the root back to the leaves. The root segment has no parent, so it has no reactive force on a parent and Equation 28 can be

solved for the root's linear acceleration. This can be used in Equation 27 to find the angular acceleration of the root. This process is repeated outward using the relationship in Equation 26 to find the linear acceleration of the son links and using this to find their angular acceleration.

The actual steps are shown below.[2] Note that $\mathbf{R}^{topar}$ signifies a 3×3 rotation matrix that takes vectors in a local frame into its parent frame, and $\mathbf{R}^{frompar}$ signifies a 3×3 rotation matrix that takes vectors in a parent frame into a son frame, and that these two are transposes of each other. $\mathbf{R}^{toworld}$ signifies a 3×3 rotation matrix that takes vectors in a local frame into the world frame, and $\mathbf{R}^{fromworld}$ signifies a 3×3 rotation matrix that takes vectors from the world frame into a local frame, and these two are also transposes of each other.

It is useful to compute the cross-product operation using a *tilde* matrix. The tilde matrix for a vector $\mathbf{a}$ is a 3×3 matrix that when postmultiplied by a vector $\mathbf{b}$ gives the same result as the cross-product $\mathbf{a} \times \mathbf{b}$. Thus, $\tilde{\mathbf{a}}\mathbf{b} = \mathbf{a} \times \mathbf{b}$.

$$\tilde{\mathbf{a}} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \qquad 29$$

INWARD PASS The inward pass computes the 4 recursive coefficients and some other useful quantities that are used often. (I have slightly simplified this step. Readers are invited to find more quantities to efficiently precompute.) This step can be divided into two passes: one (the *slowband*) need only be done occasionally; the other (the *fastband*) needs to be done each time through the dynamics loop. Remember subscripts indicate which segment the value refers to, and superscripts indicate which frame the value is in terms of (the default is frame i). The equations are repeated for each segment. Summations are over all sons of segment i.

The slowband calculations for a segment i are these:

$$\mathbf{a}_{c,son} = \omega_i \times (\omega_i \times \mathbf{l}_{son}) \qquad 30$$

$$\mathbf{Q}_{son} = \mathbf{R}^{topar}_{son}\mathbf{M}^{son}_{son}\mathbf{R}^{frompar}_{son} \qquad 31$$

$$\mathbf{W}_{son} = \tilde{\mathbf{I}}_{son}\,\mathbf{Q}_{son} \quad 32$$

$$\mathbf{T}_i = (\mathbf{J}_i + \sum(\mathbf{W}_{son}\,\tilde{\mathbf{I}}_{son}))^{-1} \quad 33$$

$$\mathbf{K}_i = \mathbf{T}_i(\sum \mathbf{W}_{son} - m_i\tilde{\mathbf{c}}_i) \quad 34$$

$$\mathbf{M}_i = (m_i\tilde{\mathbf{c}}_i)\mathbf{K}_i - m_i\mathbf{I} + \sum(\mathbf{Q}_{son}(\mathbf{I} - \tilde{\mathbf{I}}_i\mathbf{K}_i)) \quad 35$$

Along the way, torque and force information is accumulated for each segment, $\tau_{\sigma,part}$ accumulates torques, and f_σ accumulates forces. Note this assumes that external torques ($\tau_{ext,i}$) are being defined in terms of the local frame (and include torques due to external forces), but external forces ($\mathbf{F}_{ext,i}$) are in terms of the world space frame.

$$\tau_{\sigma,part,i} = -\omega_i\times(\mathbf{J}_i\times\omega_i) + \tau^i_{ext,i} + (m_i\mathbf{c}_i)\times\mathbf{R}_i^{fromworld}\mathbf{a}^{world}_{grv,i} \quad 36$$

$$f_{\sigma,i} = -\omega_i\times(\omega_i\times(m_i\mathbf{c}_i)) + \mathbf{R}_i^{fromworld}(\mathbf{f}^{world}_{ext,i} + m_i\mathbf{a}^{world}_{grv,i}) \quad 37$$

The following equations are the fastband, and should be done each time through the dynamics loop.

$$\tau_{\sigma,i} = \tau_{\sigma,part,i} - \tau_{topar,i} + \sum(\mathbf{R}^{topar}_{son}\tau^{son}_{topar,son}) \quad 38$$

$$\mathbf{d}_i = \mathbf{T}_i(\tau_{\sigma,i} + \sum(\mathbf{l}_{son}\times(\mathbf{R}^{topar}_{son}\mathbf{f}'^{son}_{son} + \mathbf{Q}_{son}\mathbf{a}_{c,son}))) \quad 39$$

$$\mathbf{f}'_i = \mathbf{f}_{\sigma,i} + (m_i\mathbf{c}_i)\times\mathbf{d}_i + \quad 40$$
$$\sum(\mathbf{R}^{topar}_{son}\mathbf{f}'_{son} + \mathbf{Q}_{son}(\mathbf{a}_{c,son} - \mathbf{l}_{son}\times\mathbf{d}_i))$$

OUTWARD PASS This completes the work traversing the tree inward. Now the tree is traversed outward; again the work can be divided into a slow and fastband depending on whether the information should be updated each time. First the important accelerations of the root segment, the only one capable of translating freely.

$$\mathbf{a}_{root} = -(\mathbf{M}_{root})^{-1}\mathbf{f}'_{root} \quad 41$$

$$\dot{\omega}_{root} = \mathbf{K}_{root}\mathbf{a}_{root} + \mathbf{d}_{root} \quad 42$$

For the rest of the segments on the way out to the leaves

$$\mathbf{a}_i = \mathbf{R}_i^{frompar}(\mathbf{a}_{c,i_{par}} + \mathbf{a}^{par}_{par} - \mathbf{l}^{par}_i\times\dot{\omega}^{par}_{par}) \quad 43$$

$$\dot{\omega}_i = \mathbf{K}_i\mathbf{a}_i + \mathbf{d}_i \quad 44$$

$$\mathbf{f}_{topar,i} = \mathbf{M}_i \mathbf{a}_i + \mathbf{f'}_i \quad 45$$

if needed to check the solution.

INTEGRATION Now integration can be performed to find the new positions and velocities. This again consists of a step that needs to be done each time period, and a step that can possibly be done less often.

This step is done each time period. δu signifies an angular change vector accumulating orientation changes. Remember that while these values are defined in terms of the local frame orientation, they are inertial, including motion not only at the joint to the parent but all motion of all ancestors back to the world. For each segment,

$$\omega_{new} = \omega_{old} + \delta t \dot{\omega} \quad 46$$

$$\delta u_{new} = \delta u_{old} + \delta t \omega \quad 47$$

For the root segment, the linear motion is also of interest. The linear motion of the other segments (here relative to the world space frame orientation) can be calculated from their angular motion.

$$\mathbf{v}_{new}^{world} = \mathbf{v}_{old}^{world} + \delta t \mathbf{R}^{toworld} \mathbf{a}_{new} \quad 48$$

$$\mathbf{p}_{new}^{world} = \mathbf{p}_{old}^{world} + \delta t \mathbf{v}_{new} \quad 49$$

Finally, the rotation matrices at the slowband rate are updated from distal to proximal (leaves to root). (Reset $\delta \mathbf{u}$ to zero after this operation.)

$$\mathbf{R}_{new}^{topar} = \mathbf{R}_{old}^{topar}(\mathbf{I} + \delta \mathbf{u}) \quad 50$$

This matrix should be orthonormalized to reduce error accumulation.[9]

Finally, each $\mathbf{R}^{topar}$ and its inverse can be calculated

$$\mathbf{R}_{new,son}^{topar} = \mathbf{R}_{new,i}^{fromworld} \mathbf{R}_{new,son}^{toworld} \quad 51$$

Armstrong and Green[2] suggest that the numerical instability that sometimes accumulates and causes bodies to flail about can be reduced by reducing the time step δt or by artificially increasing the moments of inertia about longitudinal axes. The latter method may produce some anomalous behavior, however.

Control Issues

It is not terribly difficult to write subroutines to do the dynamics explained above (or to borrow the code from a friendly spirit who has done it before). The open questions involve how to use this dynamic ability to get desirable motion and simulate constraints nicely. Some hints at solving these problems are presented in this section, but a great deal of work remains to be done before we can watch simulated animals moving realistically about on our computer screens under total dynamic control.

Clearly, the way to control the motion is to supply forces and torques that cause or restrict motion, either directly or through sophisticated preprocessors. Control could also be supplied in the form of extra constraint equations that limit the degrees of freedom involved. This method will not be discussed here.

Automatically Obvious: Gravity

The effect of gravity is easily calculated given the gravitational acceleration (about $9.81 m/sec^2$ on the earth's surface). Assuming the y-axis points away from the center of the earth, the force acting on the center of mass of each rigid body is

$$\mathbf{f}_{grv} = (0, -9.81, 0)\, m \qquad 52$$

The torque due to this force acting in the body fixed coordinate frame is

$$\tau_{grv} = \mathbf{c} \times \mathbf{f}_{grv} \qquad 53$$

External Dynamic Control

The user can shove the body about by applying forces and torques directly.

External Applied Torques

A pure *external* torque causes rotation of the body about an axis, and is specified by a 3D torque vector which is added to the net torque vector τ used in the dynamics equations for rotation.

External Applied Forces

Forces require both a 3D vector for the force itself and a 3D vector for its point of application. If is often most convenient to specify the force in terms of world space coordinates (converting it to the coordinates of the local frame of the segment upon which it is acting before doing the dynamics equations). The force itself is added to the net force used in the translational equations of motion **f**.

The position of the force is essential because the force may also cause a torque, depending upon where it is applied. It is usually most convenient to specify the torque in terms of the local coordinate frame, e.g., pick a local point of application **p**. The torque due to the force is found by Equation 16.

Internal Control

Internal control is mostly relevant to moving an articulated body in the way robots and animals move themselves, by applying torques and forces between neighboring segments. As the dynamics formulation described for articulated bodies only accommodates rotary joints, only internal torques, not forces will be mentioned.

Internal Torques

For the torque to be *internal*, e.g., simulating a muscle that acts upon two neighboring segments in an equal and opposite fashion, it should contribute to the net torque on one segment and its negative should contribute to the net torque on its neighbor.

Internal torques are also useful for simulating joint limits, e.g., to keep the arm from bending backwards at the elbow. Rotary spring and damper combinations or exponential torques can be used to simulate them.

Positional Suggestions

Moving bodies about by suggesting forces and torques is less than intuitive. Motion is usually imagined kinematically, as changes in position. It is still possible to take advantage of dynamics but have the user think in positional terms by providing a (more or less) intelligent preprocessing step that converts positional suggestions to forces and torques that will accomplish them.

Internal Positional Control

The user could suggest local positional changes at joints, e.g., rotate the elbow from 45 degrees to 60 degrees in 10 seconds. The system could take into account the mass of the segments moving and their present velocity and guess how much internal torque will do this. Using super- or adaptive sampling or feedback, reasonable torques can be found to accomplish the desired motion. Before asking "why use dynamics at all," consider that only a few joints of the body need be under positional control at any time. The rest may be left in a simple state that is automatically dealt with, e.g., relaxed and hanging loosely, or frozen into a local configuration.

External Positional Control: Goals

It is sometimes handy to pick a point on a body and then a point in world space where you would like that point to be (a goal). In this case, a force can be applied starting at the desired body point and directed toward the goal. Finding the amount of force to pull the body to the goal at a reasonable speed without overshooting it or oscillating is sometimes tricky.

Environment Interactions

It would be nice if bodies were to react automatically and realistically to their environment as well. Attempting this will add to the cost of the system, because considerable collision detection may have to be done. A simple brute force method of finding collisions is to check for the intersection of all the bounding vertices of an object with the bounding planes of all other objects.

Floors

Floors can be simulated with reasonable success by modeling them as a combination of a spring and a damper. A spring supplies a force dependent upon the amount its compressed, δc, times a constant k.

$$f_{spr} = k\delta c \qquad 54$$

Similarly, a damper supplies a force dependent upon its velocity times a constant.

For complex articulated bodies, it may be well not to use a constant constant for these equations, but find some way of automatically calculating a reasonable proportionality constant for the body considering its total motion.

Other Collisions

Collisions with other objects are not fundamentally different from floor collisions, though the objects' shapes may be different and they may be expected to move in response as well. In this case, the collision should be recognized and the collisions forces found before dynamics is done on the individual objects to find their motion in response to the collisions. For simple bodies, one might prefer to calculate the effects of collisions directly, rather than simulating them with springs and dampers. It is also possible to use analytic collision solutions for some bodies.[19]

Numerical Issues

Dynamics is considerably more expensive than kinematics, but not unreasonably so, given the rapidly decreasing cost of compute power. Probably simple dynamics can be done on modern personal computers without too much trouble. The bells and whistles are costly, e.g. collision detection, joint limits, internal preprocessed control, etc. Lots of work remains to be done on this. Use of recursive dynamic formulations is a real boon. More sophisticated numerical integration methods can also help, Runge-Kutta integration is somewhat more complex to program and takes longer per time step but much larger time steps can be used than with the Euler method and results are more accurate. Adaptive calculations can also help, e.g. use large time steps when the body is falling freely but very small ones when it hits the floor. A clever adaptive idea (thanks to Ralph Abraham, UC Santa Cruz) is to do a 5th order and a 4th order Runge-Kutta integration and if they deviate more than some allowed amount, redo them with a smaller time step.

Who is doing it?

This is by no means a complete list, but the people and places that are now involved in using dynamics for computer graphics include the author and Matthew Moore[18,19,29] (at UC Santa Cruz), Bill Armstrong and Mark Green (at the University of Alberta),[1,2,3,4] Dave Forsey (at the University of Waterloo),[31] Michael Girard and A. Maciejewski[10,11] (at the Ohio State University), Dave Haumann[13] (also at the Ohio State University), Isaacs and Cohen[15] (at Cornell), Al Barr (at CalTech) with Kurt Fleischer, John Platt, Dimitri Terzopoulos, and Andrew Witkin (at Schlumberger Palo Alto Research or SPAR).[5,25,32] (The last reference, while not really dynamics, is intuitively similar and could be integrated into a dynamic system.) There are no doubt many others involved in engineering uses of computer graphics who also use dynamic simulations, but the author has no specific references to their work. There are also software packages

available for doing dynamics, though they tend to be expensive and (being extremely general) slow.

Summary

Use of dynamics for computer graphics has reached the point where it can be used usefully for animation and modeling. It is possible not only to generate motion for flexible or articulated rigid bodies using the dynamics equations of motion, but also, at least simplistically, to model constraints, collisions, and control the motion through positional suggestions. Certainly, more sophisticated and efficient methods need to be developed. Beyond these low-level issues of dynamic control, we now need to look ahead to issues of high-level control and coordination.[10,24,33] For this we should explore the related fields of robotics, mechanical engineering, biomechanics, biology, and control theory.

References

1. W. 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*, pages 1343-1346, American Society of Mechanical Engineers (1979).
2. W. W. Armstrong and M. W. Green, The Dynamics of Articulated Rigid Bodies for Purposes of Animation, *Proceedings of Graphics Interface '85*, pages 407-415, Canadian Information Processing Society (May 1985).
3. W. W. Armstrong, M. W. Green, and R. Lake, *Proceedings of Graphics Interface 86*, pages 147-151 (May, 1986).
4. W. W. Armstrong, M. W. Green, and R. Lake, Near-Real-Time Control of Human Figure Models, *IEEE Computer Graphics and Applications* 7(6) pages 52-61 (June 1987).
5. A. H. Barr, Dynamic Constraints, *SIGGRAPH '87 Tutorial Notes: Topics in Physically-Based Modeling* (1987).
6. S. D. Conte and C. de Boor, *Elementary Numerical Analysis: an Algorithmic Approach, 3rd Edition,* McGraw-Hill Book Company, New York (1980).

7. R. Featherstone, The Calculation of Robot Dynamics Using Articulated-Body Inertias, *International Journal of Robotics Research* **2**(1) pages 13-30 (Spring, 1983).

8. R. P. Feynman, R. B. Leighton, and M. Sands, in *The Feynman Lectures on Physics*, California Institute of Technology, Pasadena, CA (1963).

9. D. T. Finkbeiner, II, *Introduction to Matrices and Linear Transformations,* W. H. Freeman and Company, San Francisco, CA (1960).

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

11. M. Girard and A. A. Maciejewski, Computational Modeling for the Computer Animation of Legged Figures, *SIGGRAPH '85 Conference Proceedings* **19**, pages 263-270 (July, 1985).

12. D. T. Greenwood, in *Principles of Dynamics*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

13. D. Haumann, Modeling Flexible Bodies, *SIGGRAPH 1987 Tutorial Notes: Topics in Physically-Based Modeling* (July, 1987).

14. R. Horowitz, Model Reference Adaptive Control of Mechanical Manipulators, PhD dissertation, Mechanical Engineering, University of California, Berkeley (May, 1983).

15. P. M. Isaacs and M. F. Cohen, Controlling Dynamic Simulation with Kinematic Constraints, *SIGGRAPH 1987* (July, 1987).

16. C. S. George Lee, R. C. Gonzalez, and K. S. Fu, *Tutorial on Robotics,* IEEE Computer Society Press, Silver Spring, MD (1983).

17. W. G. McLean and E. W. Nelson, *Theory and Problems of Engineering Mechanics,* Schaum's Outline Series, McGraw-Hill Book Co., New York (1978).

18. M. Moore, A Flexible Object Animation System, Masters thesis, Computer & Information Sciences, University of California, Santa Cruz (September, 1987).

19. M. Moore and J. Wilhelms, *Collision Detection and Response for Computer Graphics*, Submitted for Publication, 1987.

20. NASA, *Anthropometric Source Book,* NASA Scientific and Technical Information Office (1978).

21. L. A. Pars, *A Treatise on Analytical Dynamics,* Ox Bow Press, Woodbridge, CT (1979).

22. R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control,* The MIT Press, Cambridge, MA (1981).

23. R. Resnick and D. Halliday, *Physics Part I,* John Wiley & Sons, Inc., New York (1966).

24. C. W. Reynolds, Flocks, Herds, and Schools: A Distributed Behavioral Model, *Computer Graphics* **21**, pages 25-34 (July, 1987).

25. D. Terzopoulos, J. Platt, A. H. Barr, and K. Fleischer, Elastically Deformable Models, *SIGGRAPH 1987* (July, 1987).

26. D. A. Wells, *Lagrangian Dynamics,* Schaum's Outline Series, McGraw-Hill Book Co., New York (1969).

27. J. Wilhelms, Graphical Simulation of the Motion of Articulated Bodies such as Humans and Robots, with Particular Emphasis on the Use of Dynamic Analysis, PhD dissertation, Computer Science, University of California, Berkeley (July, 1985).

28. J. Wilhelms, Virya - A Motion Control Editor for Kinematic and Dynamic Animation, *Proceedings of Graphics Interface 86*, pages 141-146 (May, 1986).

29. J. Wilhelms, Using Dynamic Analysis for Animation of Articulated Bodies, *IEEE Computer Graphics and Applications* **7**(6) (June, 1987).

30. J. Wilhelms and B. A. Barsky, Using Dynamic Analysis for the Animation of Articulated Bodies such as Humans and Robots, *Proceedings of Graphics Interface '85*, pages 97-104 (May 1985).

31. J. Wilhelms, D. Forsey, and P. Hanrahan, *Manikin: Dynamic Analysis for Articulated Body Manipulation,* Computer and Information Sciences Board, University of California, Santa Cruz (April, 1987). Tech. Report UCSC-CRL-87-2.

32. A. Witkin, K. Fleischer, and A. H. Barr, Energy Constraints on Parameterized Models, *SIGGRAPH 1987* (July, 1987).

33. D. Zeltzer, Motor Control Techniques for Figure Animation, *IEEE Computer Graphics and Applications* **2**(9) pages 53-60 (November, 1982).

[*submitted Oct. 22, 1987; revised Nov. 9, 1987; accepted Nov. 19, 1987*]