

Computer Animation

CMPS 80c, Winter 2001

Jane Wilhelms

1 Animation

Computer animation involves making a series of images that are shown in rapid succession to give the illusion of continuous motion. To view the animation, these images are shown on the CRT, or put frame by frame onto movie film (24 frames per second for viewing) or videotape (30 frames per second for viewing).

There are two ways to view the animation directly from the computer:

1. Pre-compute the frames one by one and store them, and then rapidly replace the images in the frame buffer with successive frames. This requires *double-buffering*, so that you don't see the replacement of the frame buffer. This can use a great deal of space, because, for example, a 500x500 pixel image is 250,000 pixels and, if full color, 750,000 bytes. An image of about video resolution is thus nearly a megabyte per image. At 30 frames per second, that is about 30 megabytes for a second of animation, or 1800 megabytes (almost 2 gigabytes) for a minute. Images can be *compressed*, by using various schemes to minimize storage of repeated information. However, even with compression, except for small demonstration animations, storing all frames is not practical on most systems. People making animations for movies or video normally make an image, preserve it on film or video, and then make the next image, do the same, etc. Machines are available to do this automatically. Compact discs and other media are used for storage as well.
2. Store enough information about changes in the animation to regenerate the image later. For example, you might remember to rotate 30 degrees around X in 1 degree increments. This requires much less space to store than whole images, plus the information stored is three-dimensional and can be re-used from different camera angles. On the other hand, though, it may take some time to redraw the two-dimensional images from the three-dimensional information. If the computer is fast enough to redraw images from motion information at the same rate as the real motion being simulated, this is called "realtime". ("Interactive speeds" in computer graphics usually means fast enough to feel comfortable making changes to the image and seeing it redraw. This is generally slower than realtime but still produces the sensation of motion, rather than a series of single frames.)

Animation information stored in the second format is succinct and fully describes the animation. The first format is used for the final filming, or to show animation that is too complex to regenerate the images in realtime.

The most common way of creating a computer animation is called *key-framing*. *Motion paths* and *morphing* can be seen as variations on keyframing. More advanced methods use *physical simulation*, *inverse kinematics*, *constraint-based animation*, *high-level control*, *behavioral animation*, etc. Sometimes motion is *captured* from live performances.

2 Kinematic Information

Descriptions of positions taken over time are called "kinematics". Kinematics means a description of the motion without any information about how or why these positions occur. That is, it doesn't indicate what forces or collisions might have caused the motion.

An *animation parameter* is one single thing that can be changed during an animation. This may be the X-position of an object, the Z-rotation, the amount of scaling, the saturation of the surface color, etc. There are thus many many animation parameters possible in any scene. If the animation parameter refers to a change in position, orientation, or scale, it is a *motion parameter*. A single point has 3 parameters for motion, also known as *degrees of freedom*: translation in X, Y, and Z. A *rigid body* has 6 motion parameters: translation in X, Y, and Z, and rotation about the X, Y, and Z axes. It has six degrees of freedom. (Scaling is not normally considered a degree of freedom, only translation and rotation.) A flexible body that is defined by edit or control points has 3 degrees of freedom for each control point. Joints of an articulated, animal-like body usually have 3 rotational degrees of freedom.

A single animation parameter can be represented by a numeric value that changes over time. In a graph, this can be shown with the time as the horizontal axis and the value of the parameter as the vertical axis. This *time graph* is a *mathematical function*. This means there can only be one single value of the parameter for any point in time. The time graph of a motion parameter that never changes in value (remains stationary) would be a horizontal line.

3 Keyframing

Keyframing is by far the most common form of animation. The idea came from old-fashioned, two-dimensional Disney-style animation first done in the 1920's and 1930's (and still being done very well today). All the work was then done by hand. Because there were few very good animators and they were expensive to hire, these experts would draw the most important frames of the 24 frames per second needed for an animation. It requires much talent and training to be able to predict what positions in frames should be to make the motion appear realistic. The frames done by these experts were called *keyframes*. Less experienced or talented animators then would create the *in-between* frames by interpolating between the keyframes they were given. Thus, the process is sometimes called *inbetweening* as well as *keyframing*.

This idea of having information about some but not all frames stored, and then interpolating to get the other frames, carried over into computer graphics. The name stuck also.

The basic idea of a keyframe is to position objects where they should be at a given time (for example, to be at a X-position= 6 at time= 0) and store all those positions. Then objects are repositioned (for example, to X-position= 20 at time= 10). These positions and times are called the *keyframes*. *Interpolation* is used to find values for positions between these times. There are different ways, mathematically, of doing interpolation. Using a simple, *linear* interpolation, the values between two keyframes can be found by noticing that adding 1.4 at each time step reaches 20 at time 10.

Time	0	1	2	3	4	5	6	7	8	9	10
Pos.	6	7.4	8.8	10.2	11.6	13.0	14.4	15.8	17.2	18.6	20.0

In fact, many key positions are usually stored, and a *spline* curve is used to interpolate between them and make a smooth change in position. This way, jerkiness is controlled. The splines are usually created in pieces (*piecewise*) and different splines have different characteristics. Some go through defining *control points* (*interpolating splines*), while others go near them (*approximating splines*). Sometimes the pieces of spline meet very smoothly (C^2 *continuity*), so that their second derivatives (curvature for position or acceleration for motion) are the same on either side of the join. *B-splines* and *NURBS* can have C^2 *continuity*. This means *position*, *velocity*, and *acceleration* will vary smoothly where the spline pieces join. Other splines only provide C^1 *continuity*, or continuity of first derivatives, tangent vectors, and velocity, at spline junctures. *Cardinal* and *Bezier* and *Hermite* splines can have this continuity. Where smooth interpolation isn't desirable, most animation systems (and spline formulations) give you other options. If the curve pieces just join, so there is no gap, this is called C^0 *continuity*.

In basic keyframing, then, one associates positions (or colors, rotations, etc., anything that can change) with times and the system fills in the times between those times for animation.

Sometimes keyframing in 3D computer graphics is called *key-positioning*. This is because a complete description of all parameters for each keyframe may not be kept. For example, no values need be stored for parameters that don't change.

3.1 Motion Paths

Motion paths are quite similar to keyframes in the sense that a minimal amount of information is stored to regenerate the motion. In the case of motion paths, however, a path through space is designed, and objects are associated with the path. There is a beginning time associated with the start of the path, and an end time associated with the end of the path, and the object moves along the path at just the right speed to reach the end at the desired time. Motion paths are really another way of looking at keyframing, because a series of key frames automatically creates a motion path, and the control points on a motion path can be seen as keyframes. Usually the term is used only for positions along a path, as opposed to rotations, scales, etc.

3.2 Morphing

Morphing refers to changes in shape of a single object to one or more new shapes. Each defining shape acts much like a keyframe. Morphing can be two-dimensional or three-dimensional.

Two-dimensional morphing is done by changing one digital image into another. You may have seen the Michael Jackson music video a few years ago ("It's black; it's white"). Various people's faces appear at center screen and look center, then right, then left, then back to center. The face gradually changes to another face with a smooth morph. This is done by associating parts of the 2D image (such as the eyebrows) with the same part in another image, and changing pixel values to smoothly go from one to another. Two-dimensional morphing was used in the movie "Terminator II", where the evil cyborg T-1000 changed into various shapes.

Three-dimensional morphing involves changing a 3D model from one shape to another, or to several new shapes in succession. The water creature in the movie "The Abyss" used three-dimensional morphing. In this case, each vertex on one model is associated with another vertex in the second model and moves using interpolation to the new position. It is really much like keyframing points, except that the points are connected to form a surface and after they are repositioned, the 3D model is rendered in the new shape.

3.3 Motion Capture

Motion capture refers to any method of measuring changing positions from the real world and then using these positions to cause motion in a simulated body. It is most commonly used for human and animal bodies (*articulated bodies* or *hierarchies*). These are usually modeled as rigid bodies connected together by joints. Joints usually have 3 rotational degrees of freedom, and can't translate (or they would come apart!). Motion capture is expensive and time-consuming, and the movement once measured is hard to change, but it is very difficult to generate realistic human or animal motion in other ways.

Motion capture is done in various ways: for example, by attaching measuring devices to limbs to measure joint angles; by attaching lights to limbs and measuring their positions; or by videotaping the body from various angles and measuring the limb positions from the video.

4 Advanced Animation Techniques

The above methods based on key-framing (key-framing, motion paths, and morphing) all require a great deal of talent and experience on the part of the user to specify the motion, and (particularly) the timing, to make the animation appear realistic. This is particularly difficult for motion involving *articulated bodies*, e.g., humans, animals, and robots. Motion capture is often used for these jointed bodies. But motion capture equipment is expensive and the motion measured is difficult to change.

A number of techniques have been developed to make it easier to produce realistic animations without training as an animator. These techniques attempt to do much of the animators' work automatically, without them having to give all the details of the motion.

4.1 Physical Simulation

Physical simulation (also called *dynamics*) uses the Newtonian laws of physics to predict the motion. In basic physical simulation, the objects in the world are modeled as having mass (and if extended bodies rather than just points, mass distribution characteristics such as moments and products of inertia). The world "simulation" as opposed to "animation" is sometimes used to indicate that objects are behaving in a way closely relating to behavior of such objects in the real world. Objects in the world move because forces act upon them. A force is like a certain amount of push in a certain direction. The user in this case somehow has to provide the system with forces to cause motion. Sometimes, forces are automatically calculated by the system based on the environment.

Newtonian physics describes how objects that have a certain mass will move under the influence of applied forces. *Newton's Second Law* for a particle is that the net force applied is equal to the mass of the particle times its acceleration. By using a mathematical process called *integration*, it is possible to calculate the velocity of a particle and the new position of a particle after a force is applied. The equations are: (F is the force, m is the mass, a is the acceleration, v is the velocity, p is the position, dt is the amount of time the force is applied.)

$$F = ma \tag{1}$$

$$a = F/m \tag{2}$$

$$v_{new} = v_{old} + a * dt \tag{3}$$

$$p_{new} = p_{old} + v_{old} * dt + 0.5a * dt * dt \tag{4}$$

$$\tag{5}$$

Thus, if the force F is provided, a dynamics program will eventually find p_{new} , the new position of the particle under the influence of that force. For rotation, the equations are similar, and the force acts as a *torque* causing a turning motion.

Though particles may seem very simple objects, it is possible to realistically model flexible materials like cloth and skin as particles connected by simulated springs. The springs allow particles to stretch apart and move relative to each other, while keeping them connected. Springs apply a force between the two objects they connect that is dependent on how much the spring is deformed from its rest position.

If the masses and forces are what one would encounter in the real world, the motion resulting will be correct (or close) to that found in the real world. This makes it possible for untrained users to develop animations that appear realistic, in theory.

This process works very well for simple solid objects such as a baseball that is thrown in the air and come down due to gravity (which is just another force). It is also very effective for flexible bodies such as cloth, hair, and jello. Unfortunately, it doesn't work as well for complex bodies like humans, because the actual forces applied by muscles to create animal motion are so complex that simulating them is very difficult. Some progress is being made, but much needs to be done.

4.2 Collision Detection and Response

Another important problem you may have already encountered is that when you are arranging objects in your scene they are perfectly happy to intersect (go through) each other. Also, it is difficult to exactly place an object on another, such as a lamp on a table. *Collision detection* and *response* help solve this problem.

Collision detection means that the computer calculates for every object whether it is intersecting any other objects. It can do this quite accurately by comparing the positions of vertices, lines, and polygons, but it takes some time.

Collision response means that, given a known collision, objects are rearranged so that they no longer collide. This could be done by pushing objects apart just enough to get rid of the collision, and is good for placing objects on tables, etc. If the scene is animated, it may mean calculating the collision forces that occur when the objects strike and the correct motion away from each other. Collision response is difficult to calculate really correctly, especially if you take into account *friction* and *elasticity*. There are many simplifications that can be applied, however.

4.3 Inverse Kinematics

Inverse kinematics deals with a particular problem encountered in modeling and animating articulated (jointed) bodies. *Kinematics* refers to motion described only in terms of positions, not considering physics. Inverse kinematics solves the following problem: Given that you want a jointed body to reach for a certain position in space, how do you move the joints to accomplish this motion. For example, say you want your human model to reach out its arm for a coffee cup on a table. Normally, because of the way the human model is stored in the computer, you can only move the hand by rotating the joints of the wrist, elbow, shoulder, trunk, etc. or by moving the whole body over. In most cases, there are many ways to rotate the joints to get to the final position. Some are unnatural or uncomfortable. Inverse kinematics solves this problem and finds the joint angles to get to the proper position. There are many methods to do this, and it is also an important problem for robots, such as those welding a car body. The difficulty is finding a method that gives not just any solution but a natural one. Lightwave has some ability to do inverse kinematics.

4.4 Constraint-Based Animation

Constraint-based animation refers to a number of techniques which may include physical simulation, collision effects, and inverse kinematics, but is more general. A *constraint* is something that must hold true in the animation. For example, if you are modeling a person jumping over an obstacle, you might give a constraint that the body must rise in the air enough to clear it. Or you might give a constraint that the body must pass through a certain number of points along a path, while still maintaining the rules of physical simulation. There are mathematical ways to solve this problem, which we won't go into in any more detail.

4.5 Behavioral Animation

Behavioral animation (sometimes called *stimulus-response animation*) is animation where the objects moved are given some rules and determine their own motion based on the rules and what is happening in their simulated environment.

An early example of this was an animation by Reynold's of birds and fishes, part of which appeared on one of the videos we watched. In this case, the birds were given simple rules such as "keep to the center of the flock", "go in a certain direction", and (perhaps most important) "don't run into anything". These rules were incorporated into the animation program. When the program was run, each simulated bird would first check for collisions, and if one was about to happen, that rule would take precedence and it would move away from the collision. Otherwise, the other rules would be used.

5 Hierarchies and Articulated Bodies

Human, animals, and robots fall under the category of articulated bodies. These are modelled as rigid *segments* (that may have a soft surface) connected at *joints*. A characteristic of these bodies that is noticeable is that the segments (like a finger joint, or the lower arm) do not normally come loose from the rest of the

body, and they only rotate at the joints connecting them to the center of the body. Thus, the parts of an articulated body cannot move totally independently in space using all six degrees of freedom, but only three rotational degrees of freedom.

There can be as many joints and segments as necessary to model the body. The real human body contains a few hundred segments and joints. Most computer models are simpler, for example, by not including all the finger segments and joints, or not including all the vertebrae in the spine.

Articulated bodies are modeled as a *hierarchy*. A hierarchy is conceptually like an upside down tree, that has one root and many branches. The body hierarchy has a root point usually on the trunk or abdomen that is assumed to be the connection of the body to the world. This point can move or rotate freely. When you move the root, the whole body moves with it as if it were solid.

The root body point is a fixed part of the root segment. Let's say the root segment is the torso. It has *children* segments attached to it at specific joints, such as the neck, left arm, right arm and abdomen. The segments are called children of the torso, and the torso is called the *parent* of these segments. If you move a parent segment, the children always move with it, as if they were part of a solid object. However, if you move the child segment, it moves independently by rotating at the joint with its parent. The parent does not move. However, all the children of the child segment move when it moves. That is, if you rotate the upper arm at the shoulder, the lower arm and hand and fingers must move as well. This is much like real bodies work.

Each segment has one parent, but can have many children. For example, the right arm might have one child, the lower right arm. The lower right arm might have one child that is the hand. The hand could have 5 children, the thumb and four fingers. Each finger can have a child for each bendable joint connected to it.

Generally, each joint is assumed to allow 3 degrees of freedom of rotation. Some joints, like the shoulder, are very mobile and use these degrees of freedom often. Others, like the fingers, mostly just rotate about one axis like a hinge. In general, though, most joints allow some amount of rotation about three axes. The whole body can move with the full 6 degrees of freedom about in the world.

Within a single degree of freedom, *joint limits* may further constrain motion. For example, you can rotate your lower arm relative to your upper arm by less than 180 degrees. If you rotated further, one arm would intersect the other. Muscles, joints, and connective tissue also constrain the amount of motion that is possible.