Particle Systems for Interactive Graphics

Steve Marschner
Cornell University
CS 569 Spring 2008, 4 March

The whole point of interactive graphics is that things can move. So far all our motion has come from moving the camera around, or applying simple transformations to objects in the scene. This is great for modeling applications, but for something like a game we want much more complex motion.

Types of animation in interactive systems

Broadly speaking, there are three sources of motion for graphics:

1. Keyframe animation
2. Motion capture
3. Physical simulation

The first two types are pre-recorded animation: an animator painstakingly adjusts control curves, or an actor does take after take, until the motion is perfect, and then it is stored away to be played back at the appropriate time. These kinds of animation are not inherently interactive: the motion is just there, independent of what the user does. In interactive applications they respond to user control by being triggered at the appropriate moment. For instance, the player hits the "kick" button and her character executes a pre-scripted kick, which knocks out the bad guy, who then executes a pre-scripted fall onto the floor. In a football game, after a player makes a touchdown the character might spike the ball and do a victory dance that was recorded from the real player using motion capture.

In games there are lots of motions that are designed to fit together in sequences like this. The simplest arrangement is a state machine: a character moves from pose to pose via scripted motions, and it can only execute motions that start from the current pose. If you get more sophisticated about it, you can blend motions together to construct smooth transitions in less constrained circumstances, producing a wider, but still limited, range of motions. We'll talk more about the techniques for playing back these motions in the next lecture.

In physical simulation, the motion of an object is computed by simulating the physics that govern its motion. The object starts in some initial state, is subject to forces over time that alter its motion, and the motion can just be computed. For appropriately simple physical models, this simulation can be done in real time—that is, it takes less than a frame of wall-clock time to compute one frame of motion. In this case, a simulated motion is inherently interactive: the object or system can be subject to all manner of forces and other inputs that are caused by the user, and the simulation automatically behaves appropriately.

In games and movies, simulation is used for secondary motion: billowing smoke, swirling fog, wrinkling clothes, splashing water, blowing hair, etc. It is also used for events that are primarily caused by passive motion: explosions, collapses, characters falling through the air, etc. In games a simulation can become part of the game play: the dynamics of a car the player is driving, the motion of passive obstacles, etc. There are some very early examples of games whose gameplay is governed by a simulation: asteroids is a simple particle system; lunar lander is a one-particle system with gravity. These games are challenging and fun partly because you have to understand the physics of your vehicle in order to play successfully.

The autonomy of simulations is both a blessing and a curse: the good thing is that the simulation determines the motion without any requirement for human supervision; the bad thing is that the simulation determines the motion without any opportunity for human supervision. The simulation does what it does, and in a game this can be awkward because it's hard to guarantee anything about what is going to happen. This is fine if the simulation is just for decoration, but if it affects the game's world (e.g. simulations of vehicle behavior or of breaking walls) then it can become harder for the game designer to control the course of the action—which is itself a blessing and a curse!

Types of simulations

Simulations can be categorized by what kind of physical model controls the motion:

- Rigid bodies, governed by momentum and angular momentum
- Cloth, governed by the dynamics of an elastic sheet
- Fluids, governed by pressure and velocity via the Navier-Stokes equations
- Elastic solids, governed by the dynamics of an elastic volume
- Particle systems, governed by very simple Newtonian mechanics

All these types of simulations are used frequently in offline applications (such as film production), but only the simpler ones can be simulated in real time (though of course that dividing line is constantly in motion as hardware performance and computational techniques improve). The simplest of all these is the particle system, which is therefore by far the most often used simulation technique in the interactive setting. (Rigid body simulation is also popular, because it can also be implemented very efficiently, but it is quite a bit more complex.)

One great thing about particles is that they can be used fairly effectively to approximate all the types of physics listed above.

Particle systems

Witkin and Baraff have done a great job of putting together slides and notes on particle systems for graphics, so rather than attempting to improve on that I will ask you to read their notes instead. They are available at the URL below and are linked from the course web site.

Simulating other things with particles

Particle systems can be set up to approximate all manner of physical systems. Particles are not always the fastest or most accurate approach, but they are intuitive and simple to get working, so they are very widely used.

Flexible bodies

Cloth, rope, hair, flesh, and other 1- to 3-D flexible objects can be approximated by collections of particles that are connected up into a fairly stiff mesh with springs. Cloth is a good example—it can be set up using stiff springs to resist extension and weaker springs to resist bending and shearing.

A slightly more sophisticated approach, described in Baraff and Witkin's influential paper, "Large Steps in Cloth Simulation," is to define the forces, not with springs, but with somewhat more general *n*-ary forces based on energy functions that penalize distortions in the cloth.

One problem with flexible bodies is that they tend to lead to *stiff* differential equations that are very difficult to solve stably using explicit methods like the ones we've discussed. They can be more successfully simulated using *implicit* methods, which can take much larger time steps at the expense of higher per-step computation and considerably increased algorithmic complexity.

Fluids

Liquids can be simulated using an approach known as *smooth particle hydrodynamics* or SPH. The main difference between this kind of approach and a spring mesh for flexible bodies is that the connections are not fixed. In SPH, the forces on a particle are calculated by searching for nearby particles and using their positions and velocities. For instance, there is a "pressure" force that pushes particles towards regions of lower particle density and a "viscosity" force that damps out differences in velocity between nearby particles. The key to getting it to work is nice smooth estimates of the properties of the fluid based on the particles' state.

Further reading

Witkin and Baraff's SIGGRAPH 01 course notes are a great source for learning this material, particularly if you are coming to the topic with a CS/graphics background but less knowledge of physics and numerical analysis.

Andrew Witkin and David Baraff, *Physically Based Modeling*. Notes for SIGGRAPH 2001 Course 25. URL: http://www.pixar.com/companyinfo/research/pbm2001/index.html

David Baraff and Andrew Witkin, "Large Steps in Cloth Simulation." SIGGRAPH 1998.

Matthias Müller, David Charypar and Markus Gross, "Particle-based Fluid Simulation for Interactive Applications." Eurographics Symposium on Computer Animation 2003.