Automatic Animation

Robert Bridson

December 12, 2011

In the last part of the book, we turn to (semi-)automatic methods for animation. While the techniques we examined earlier make it possible to animate virtually anything, the degree of artist involvement — designing the motion curve for each and every animated parameter — makes many things infeasible. This is especially true in some games, where not every desired motion can be pre-animated and stored ahead of time.

Instead we need ways to better leverage computer power, with automatic or semi-automatic methods to generate animation. This broad field can be roughly divided into three areas, but with considerable overlap between them:

- *Procedural animation*, where a program generates the animation based on rules and procedures, perhaps involving random numbers or noise functions to incorporate unpredictable variation. Often this involves a creative process of reverse-engineering the appearance of the desired phenomena.
- *Physics-based animation*, essentially a specialized subset of procedural animation but which has grown into a major area of its own. Here equations are derived from the underlying physical principles of the desired motion, usually an elaboration of Newton's $F = ma$, and the computer is used to numerically approximate their solution (simulating the physics).
- *Data-driven animation*, where the motion data is recorded from the actual motion itself in the real world, then played back in the computer (perhaps with additional processing). *Motion capture* is the preeminent example of this.

1 Procedural Animation

Lots more to say, here are the important bullet points to jog your memory of class:

- Using Perlin noise to add small, non-periodic variations to motion.
- Particle systems, with rules for seeding/emission, evolution, and destruction of particles. Rendering options for particles: as points, connected in curves or meshes, with video sprites, or wrapping an implicit surface such as blobbies around them.
- Rendering implicit surfaces - raytracing in particular.
- Particles as full-fledged agents - crowds, flocks of birds, etc.
- Random processing of geometry, e.g. faking fracture with Voronoi diagrams.

2 Physics-Based Animation

- *Eulerian* versus *Lagrangian* descriptions.
- Ripples/waves from simple rules about acceleration up or down in a height field:

$$\begin{aligned}\frac{\partial h_{ij}}{\partial t} &= v_{ij} \\ \frac{\partial v_{ij}}{\partial t} &= k\left(\frac{h_{i+1,j}+h_{i-1,j}+h_{i,j+1}+h_{i,j-1}}{4} - h_{ij}\right)\end{aligned} \tag{1}$$

- Gravity on particles
- Spring forces between particles:

$$\vec{F}_i j = -k\left(\frac{\|\vec{x}_i - \vec{x}_j\| - L_{ij}}{L_{ij}}\right)\frac{\vec{x}_i - \vec{x}_j}{\|\vec{x}_i - \vec{x}_j\|} \tag{2}$$

- Time integration; using a model problem and recurrence relations to analyze stability. Forward Euler is bad, Symplectic Euler is good if the time step is small enough:

$$\begin{aligned}v_{\text{new}} &= v + \Delta t\, a \\ x_{\text{new}} &= x + \Delta t\, v_{\text{new}}\end{aligned} \tag{3}$$

- Include some damping, even just with `v*=0.99`.

3 Data-Driven Animation

Key idea of marker-based motion capture is just a generalization of the matchmove we worked out before, only now solving for the position of each marker based on several camera views.