Computer Animation & Particle Systems

Some slides courtesy of Jovan Popovic & Ronen Barzel

Today

- How do we specify or generate motion?
  - Keyframing
  - Procedural Animation
  - Physically-Based Animation
  - Forward and Inverse Kinematics
  - Motion Capture
- What is a Particle System?
- Particle System Examples
- Advanced "Particle Systems"

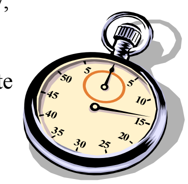
Keyframing

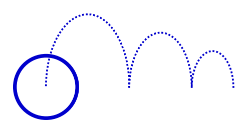
- Use spline curves to automate the inbetweening
  - Good control
  - Less tedious than drawing *every* frame
- Creating a good animation still requires considerable skill and talent

ACM © 1987 "Principles of traditional animation applied to 3D computer animation"

Procedural Animation

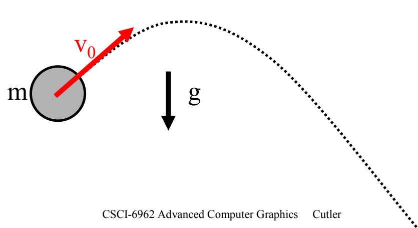
- Describes the motion algorithmically, as a function of small number of parameters
- Example: a clock with second, minute and hour hands
  - express the clock motions in terms of a "seconds" variable
  - the clock is animated by varying the seconds parameter
- Example: A bouncing ball
  - $Abs(sin(\omega t+\theta_0))*e^{-kt}$





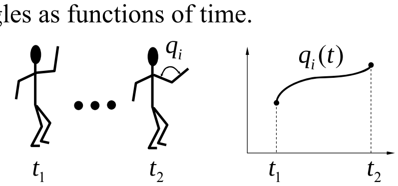
Physically-Based Animation

- Assign physical properties to objects (masses, forces, inertial properties)
- Simulate physics by solving equations
- Realistic but difficult to control



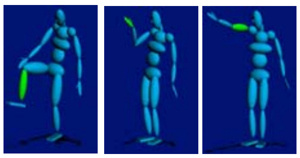
Articulated Models

- Articulated models:
  - rigid parts
  - connected by joints
- They can be animated by specifying the joint angles as functions of time.

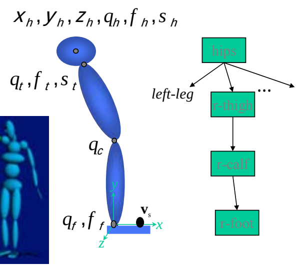


Skeleton Hierarchy

- Each bone transformation described relative to the parent in the hierarchy:

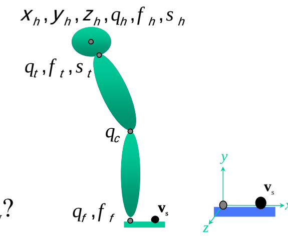


1 DOF: knee 2 DOF: wrist 3 DOF: arm



Forward Kinematics

- Given skeleton parameters p, and the position of the effecter in local coordinates V_s, what is the position of the effector in the world coordinates V_w?



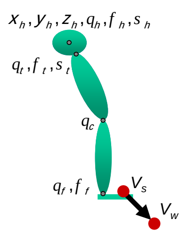
$V_w = T(x_h,y_h,z_h)R(q_h,f_h,s_h)T_hR(q_t,f_t,s_t)T_tR(Q_c)T_cR(q_f,f_f)V_s$

$V_w = S(p)V_s$

Inverse Kinematics (IK)

- Given the position of the effecter in local coordinates V_s and the *desired position* V_w in world coordinates, what are the skeleton parameters p?
- Much harder requires solving the inverse of the non-linear function:

  find p s.t. $S(p)V_s = V_w$



- Underdetermined problem with many solutions

IK Challenge

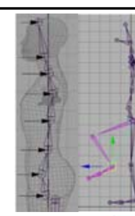
- Find a “natural” skeleton configuration for a given collection of pose constraints
- A *scalar objective function g(p)* measures the quality of a pose, g(p) is minimum for most natural poses
  - Example g(p): deviation from natural pose, joint stiffness, power consumption, etc…
- A *vector constraint function C(p) = 0* collects all pose constraints:

Motion Capture

- Optical markers, high-speed cameras, triangulation → 3D position
- Captures style, subtle nuances and realism
- You must observe someone do something

How Do They Animate Movies?

- Keyframing mostly
- Articulated figures, inverse kinematics
- Skinning
  - Complex deformable skin, muscle, skin motion
- Hierarchical controls
  - Smile control, eye blinking, etc.
  - Keyframes for these higher-level controls
- A huge time is spent building the 3D models, its skeleton and its controls
- Physical simulation for secondary motion
  - Hair, cloths, water
  - Particle systems for “fuzzy” objects

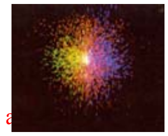


Images from the Maya tutorial

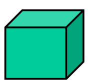
Questions?

Today

- How do we specify or generate motion?
- What is a Particle System?
  - Each particle maintains its state (position, velocity, color, etc.)
  - Describe the external forces with a
  - Integrate the laws of mechanics (ODE Solvers)
  - Collisions (later…)
- Particle System Examples
- Advanced “Particle Systems”



Types of Dynamics

- Point 
- Rigid body 

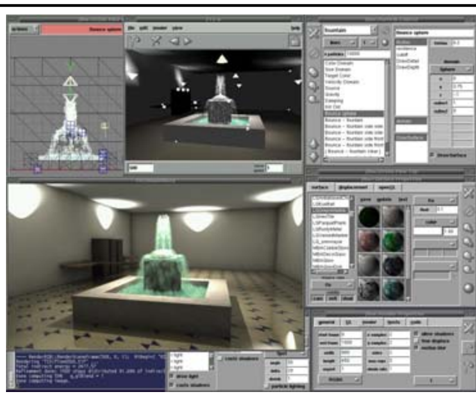
ACM© 1988 “Spacetime Constraints”

- Deformable body (include clothes, fluids, smoke, etc.)

Carlson, Mucha, Van Horn, & Turk 2002

What is a Particle System?

- Collection of many small simple particles
- Particle motion influenced by *force fields*
- Particles created by *generators*
- Particles often have *lifetimes*
- Used for, e.g:
  - sand, dust, smoke, sparks, flame, water, …



Demos by Mark B. Allan
http://users.rcn.com/mba.dnai/software/flow/

Particle Motion

- mass m, position x, velocity v
- equations of motion:

$$\frac{d}{dt}x(t) = v(t)$$

$$\frac{d}{dt}v(t) = \frac{1}{m}F(x,v,t)$$

- Ordinary Differential Equation:

$$\mathbf{X} = \begin{pmatrix} x \\ v \end{pmatrix} \qquad f(X,t) = \begin{pmatrix} v \\ \frac{1}{m}F(x,v,t) \end{pmatrix}$$

Numerical solutions to ODEs

- Analytic solutions can be found for some classes of differential equations, but most can't be solved analytically (e.g., 3-body problem)

$$\frac{d\,\mathbf{X}(t)}{dt} = f(\mathbf{X}(t), t)$$

- Given a function $f(\mathbf{X},t)$ compute $\mathbf{X}(t)$
- Typically, *initial value problems*:
  - Given values $\mathbf{X}(t_0)=\mathbf{X}_0$
  - Find values $\mathbf{X}(t)$ for $t > t_0$
- Also, boundary value problems, constrained problems, etc…

Solving ODEs for animation

$$\mathbf{X}(t) = \mathbf{X}_0 \qquad t = t_0$$
$$\tfrac{d}{dt}\mathbf{X}(t) = f(\mathbf{X}(t), t) \qquad t \geq t_0$$

- For animation, want a series of values:

$$\mathbf{X}(t_i) \qquad t_i = t_0, t_1, t_2, \ldots$$

  - samples of the continuous function $\mathbf{X}(t)$
  - i.e., frames of an animation

Questions?

Path through a field

- $f(\mathbf{X},t)$ is a vector field defined everywhere
  - E.g. a velocity field which may change over time

- $X(t)$ is a path through the field

Higher order ODEs

- E.g., Mechanics has 2nd order ODE:

$$\frac{d^2}{dt^2}x = \frac{1}{m}F$$

- Express as 1st order ODE by defining $v(t)$:

$$\tfrac{d}{dt}x(t) = v(t)$$
$$\tfrac{d}{dt}v(t) = \tfrac{1}{m}F(x, v, t)$$

$$\mathbf{X} = \begin{pmatrix} x \\ v \end{pmatrix} \qquad f(X,t) = \begin{pmatrix} v \\ \frac{1}{m}F(x,v,t) \end{pmatrix}$$

E.g., for a 3D particle

- We have a 6 dimension ODE problem:

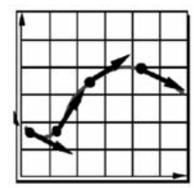
$$\mathbf{X} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{pmatrix} \qquad f(\mathbf{X},t) = \begin{pmatrix} v_x \\ v_y \\ v_z \\ \frac{1}{m}F_x(\mathbf{X},t) \\ \frac{1}{m}F_y(\mathbf{X},t) \\ \frac{1}{m}F_z(\mathbf{X},t) \end{pmatrix}$$

and for a collection of 3D particles…

$$\mathbf{X} = \begin{pmatrix} p_x^{(1)} \\ p_y^{(1)} \\ p_z^{(1)} \\ v_x^{(1)} \\ v_y^{(1)} \\ v_z^{(1)} \\ p_x^{(2)} \\ p_y^{(2)} \\ p_z^{(2)} \\ v_x^{(2)} \\ v_y^{(2)} \\ v_z^{(2)} \\ \vdots \end{pmatrix} \quad f(\mathbf{X},t) = \begin{pmatrix} v_x^{(1)} \\ v_y^{(1)} \\ v_z^{(1)} \\ \frac{1}{m_1} F_x^{(1)}(\mathbf{X},t) \\ \frac{1}{m_1} F_y^{(1)}(\mathbf{X},t) \\ \frac{1}{m_1} F_z^{(1)}(\mathbf{X},t) \\ v_x^{(2)} \\ v_y^{(2)} \\ v_z^{(2)} \\ \frac{1}{m_2} F_x^{(2)}(\mathbf{X},t) \\ \frac{1}{m_2} F_y^{(2)}(\mathbf{X},t) \\ \frac{1}{m_2} F_z^{(2)}(\mathbf{X},t) \\ \vdots \end{pmatrix}$$

Intuitive solution: take steps

- Current state X
- Examine f(X,t) at (or near) current state
- Take a step to new value of X
- Most solvers do some form of this



Euler's method

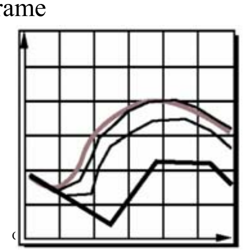
- Simplest and most intuitive.
- Define **step size** *h*
- Given $\mathbf{X}_0 = \mathbf{X}(t_0)$, take step:

$$t_1 = t_0 + h$$
$$\mathbf{X}_1 = \mathbf{X}_0 + h\, f(\mathbf{X}_0, t_0)$$

- Piecewise-linear approximation to the curve

Effect of step size

- Step size controls accuracy
- Smaller steps more closely follow curve
- For animation, may need to take many small steps per frame



Euler's method: inaccurate

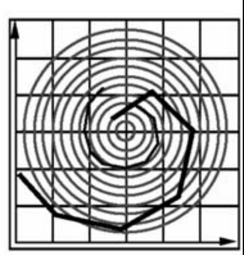
- Moves along tangent; can leave curve, e.g.:

$$f(\mathbf{X},t) = \begin{pmatrix} -y \\ x \end{pmatrix}$$

- Exact solution is circle:

$$\mathbf{X}(t) = \begin{pmatrix} r\cos(t+k) \\ r\sin(t+k) \end{pmatrix}$$

- Euler's spirals outward
- no matter how small *h* is



Euler's method: unstable

$$f(x,t) = -kx$$

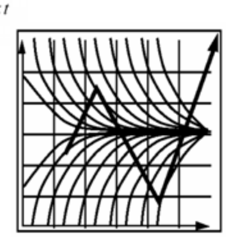
- Exact solution is decaying exponential:

$$x(t) = x_0 e^{-kt}$$

- Limited step size:

$$x_1 = x_0(1 - hk)$$

$$\begin{cases} h \le 1/k & \text{ok} \\ h > 1/k & \text{oscillates} \pm \\ h > 2/k & \text{explodes} \end{cases}$$



- If *k* is big, *h* must be small

Analysis: Taylor series

- Expand exact solution $\mathbf{X}(t)$

$$\mathbf{X}(t_0+h) = \mathbf{X}(t_0) + h\left(\tfrac{d}{dt}\mathbf{X}(t)\right)\Big|_{t_0} + \tfrac{h^2}{2!}\left(\tfrac{d^2}{dt^2}\mathbf{X}(t)\right)\Big|_{t_0} + \tfrac{h^3}{3!}(\cdots) + \cdots$$

- Euler's method approximates:

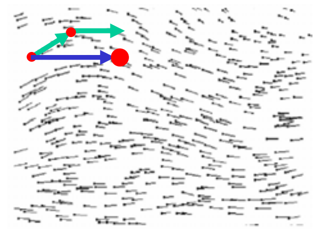
$$\mathbf{X}(t_0+h) = \mathbf{X}_0 + h\,f(\mathbf{X}_0, t_0) \quad \ldots + O(h^2)\text{ error}$$

$$h \to h/2 \;\Rightarrow\; error \to error/4 \text{ per step} \times \text{twice as many steps} \to error/2$$

- First-order method: Accuracy varies with h
- To get 100x better accuracy need 100x more steps

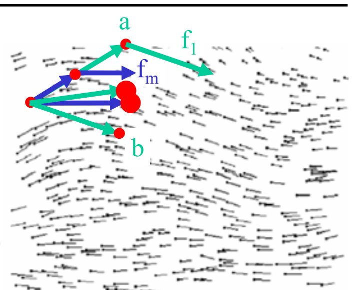
Can we do better than Euler's method?

- Problem: f has varied along the step
- Idea: look at f at the arrival of the step and compensate for variation

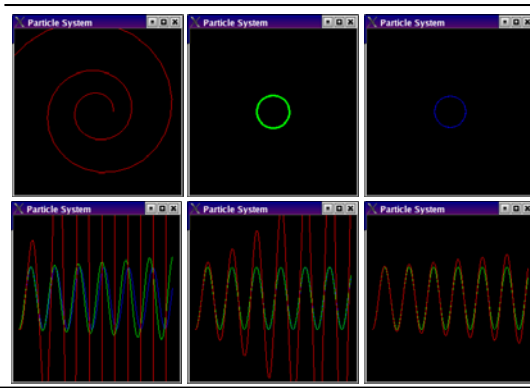


2nd-order methods

- Midpoint:
  - ½ Euler step
  - evaluate f_m
  - full step using f_m
- Trapezoid:
  - Euler step (a)
  - evaluate f_1
  - full step using f_1 (b)
  - average (a) and (b)
- Not exactly same result
- Same order of accuracy



Comparison: Euler, Midpoint, Runge-Kutta



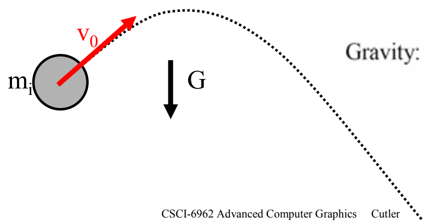
Questions?

What is a force?

- Forces can depend on location, time, velocity
- There can be multiple force sources

Forces: Gravity on Earth

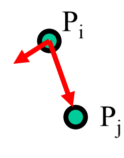
- depends only on particle mass:
- *f*(**X**,*t*) = constant
- for smoke, flame: make gravity point up!



$$\text{Gravity: } f^{(i)} = \begin{pmatrix} 0 \\ 0 \\ -m_i G \end{pmatrix}$$

Forces: Gravity for N-body problem

- Depends on all other particles
- Opposite for pairs of particles
- Force in the direction of $p_i p_j$ with magnitude inversely proportional to square distance
- $F_{ij} = G\, m_i m_j / r^2$



Forces: Damping

$$f^{(i)} = -d v^{(i)}$$

- force on particle i depends only on velocity of i
- force opposes motion
- removes energy, so system can settle
- small amount of damping can stabilize solver
- too much damping makes motion too glue-like

Forces: Spatial fields

- force on particle i depends only on position of i
  - wind
  - attractors
  - repulsers
  - vortices
- can depend on time
- note: these add energy, may need damping too

Forces: Spatial interaction

- e.g., approximate fluid: Lennard-Jones force:

$$f(x^{(i)}, x^{(j)}) = \frac{k_1}{\left|x^{(i)} - x^{(j)}\right|^m} - \frac{k_2}{\left|x^{(i)} - x^{(j)}\right|^n}$$

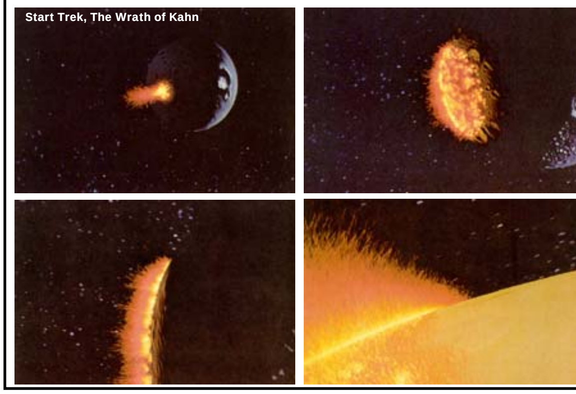
- $O(N^2)$ to test all pairs
  - In practice, only local

Questions?

Today

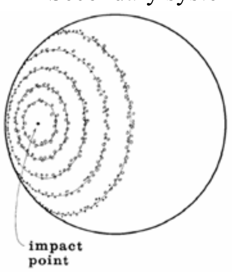
- How do we specify or generate motion?
- What is a Particle System?
- Particle System Examples
- Advanced "Particle Systems"

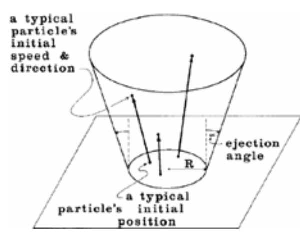
Particle Animation [Reeves et al. 1983]



How they did it?

- One big particle system at impact
- Secondary systems for rings of fire.





ig. 2. Distribution of particle systems on the planet's surface.

Where do Particles come from?

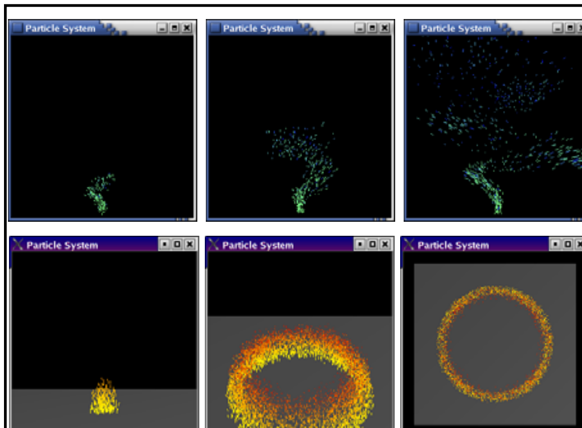
- Often created by *generators* (or "emitters")
  - can be attached to objects in the model
- Given rate of creation: particles/second
  - record t_{last} of last particle created

$$n = \lfloor (t - t_{last}) rate \rfloor$$

  - create n particles. update t_{last} if $n > 0$
- Create with (random) distribution of initial x and v
  - if creating $n > 1$ particles at once, spread out on path

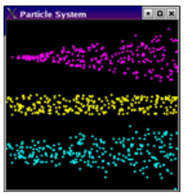
Particle Lifetimes

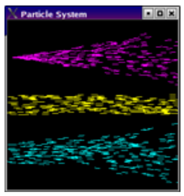
- Record time of "birth" for each particle
- Specify lifetime
- Use particle age to:
  - remove particles from system when too old
  - often, change color
  - often, change transparency (old particles fade)
- Sometimes also remove particles that are offscreen



Rendering and Motion Blur

- Particles are usually not shaded (just emission)
- Often, rendered last with z-buffer disabled
- Draw a line for motion blur (x, x + vdt)
- Sometimes use texture maps (fire, clouds)





"Particle Dreams" by Karl Sims

- 1988
- "A Connection Machine CM-2 computer was used to perform physical simulations on thousands of particles simultaneously, one processor for each particle."
- Karl Sims, "Particle Animation and Rendering Using Data Parallel Computation", SIGGRAPH 1990
- http://www.genarts.com/karl/

Particle Modeling

- grass made from particles [Reeves et al. 1983]

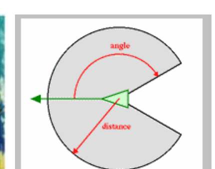
Questions?

Today

- How do we specify or generate motion?
- What is a Particle System?
- Particle System Examples
- Advanced "Particle Systems"

Advanced "Particle Systems"

- Each bird modeled as a complex particle ("boid")
- A set of forces control its behavior based on location of other birds and control forces

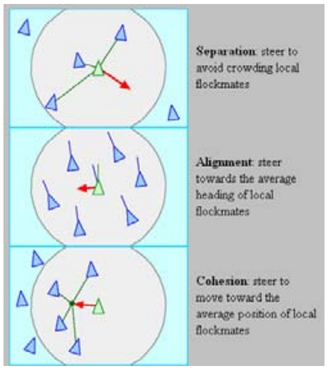



a boid's neighborhood

- Craig Reynolds http://www.red3d.com/cwr/boids

Advanced “Particle Systems”

- “Boid" is an abbreviation of "birdoid", as the rules apply equally to simulated flocking birds, and schooling fish.
- Craig Reynolds



Fish School, Crowds

“Finding Nemo” Pixar

Questions?