CS-184: Computer Graphics

Lecture #21: Physically Based Animation Intro

Prof. James O'Brien
University of California, Berkeley

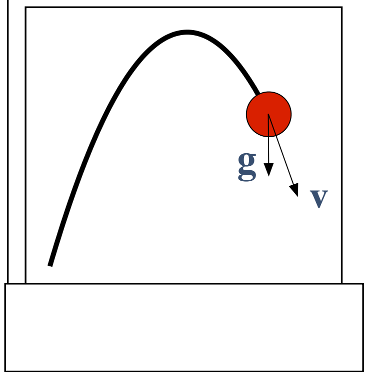
V2011-F-21-1.0

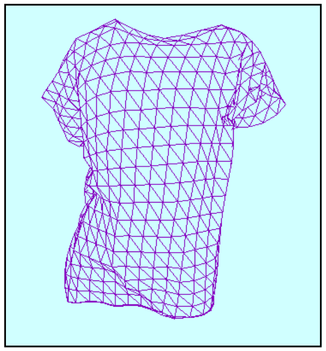
Today

- Introduction to Simulation
  - Basic particle systems
  - Time integration (simple version)

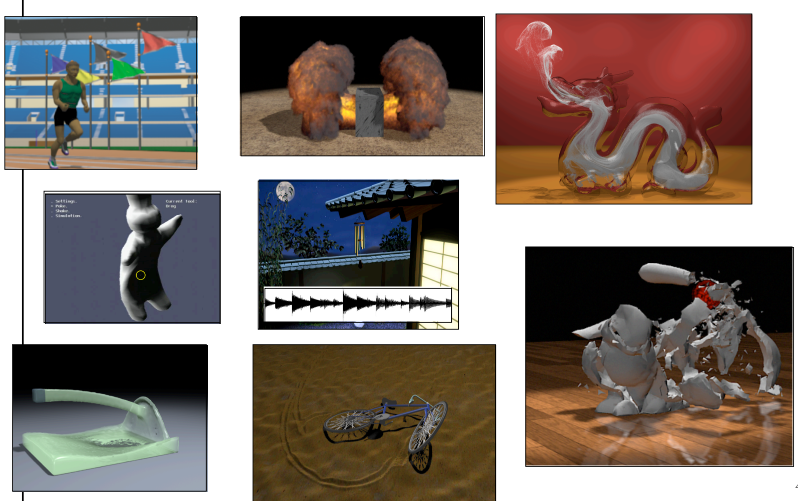
Physically Based Animation

- Generate motion of objects using numerical simulation methods





Physically Based Animation



Particle Systems

- Single particles are very simple
- Large groups can produce interesting effects
- Supplement basic ballistic rules
  - Collisions
  - Interactions
  - Force fields
  - Springs
  - Others...

Karl Sims, SIGGRAPH 1990

5

6

Particle Systems

- Single particles are very simple
- Large groups can produce interesting effects
- Supplement basic ballistic rules
  - Collisions
  - Interactions
  - Force fields
  - Springs
  - Others...

Feldman, Klingner, O'Brien, SIGGRAPH 2005

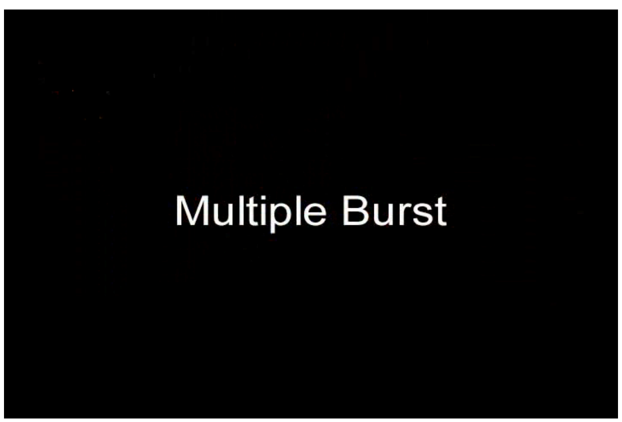
Basic Particles

- Basic governing equation $\ddot{\boldsymbol{x}} = \frac{1}{m}\boldsymbol{f}$
  - $\boldsymbol{f}$ is a sum of a number of things
    - Gravity: constant downward force proportional to mass
    - Simple drag: force proportional to negative velocity
    - Particle interactions: particles mutually attract and/or repell
      - Beware $O(n^2)$ complexity!
    - Force fields
    - Wind forces
    - User interaction

Basic Particles

- Properties other than position
  - Color
  - Temp
  - Age
- Differential equations also needed to govern these properties
- Collisions and other constrains directly modify position and/or velocity

Particle Rules



Bryan E. Feldman, James F. O'Brien, and Okan Arikan. "**Animating Suspended Particle Explosions**". In *Proceedings of ACM SIGGRAPH 2003*, pages 708–715, August 2003.

Integration

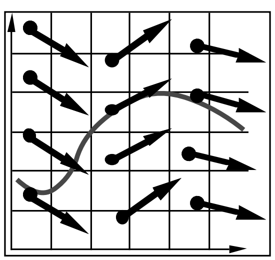
- Euler's Method
  - Simple
  - Commonly used
  - Very inaccurate
  - Most often goes unstable

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^{t} + \Delta t$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t$$

Integration

- For now let's pretend $\boldsymbol{f} = m\boldsymbol{v}$
  - *Velocity* (rather than acceleration) is a function of force



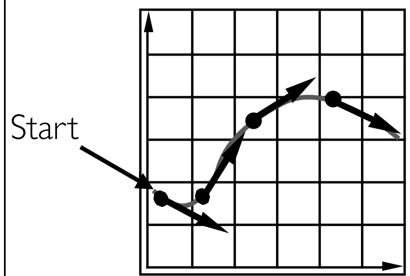
Witkin and Baraff

$$\dot{\boldsymbol{x}} = \mathsf{f}(\boldsymbol{x}, t)$$

Note: Second order ODEs can be turned into first order ODEs using extra variables.

Integration

- For now let's pretend $\boldsymbol{f} = m\boldsymbol{v}$
  - *Velocity* (rather than acceleration) is a function of force

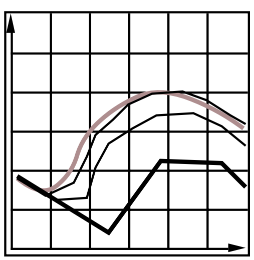


Witkin and Baraff

$$\dot{\boldsymbol{x}} = \mathsf{f}(\boldsymbol{x}, t)$$

Integration

- With numerical integration, errors accumulate
- Euler integration is particularly bad

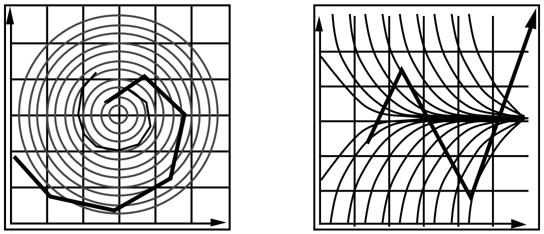


Witkin and Baraff

$$x := x + \Delta t \ \mathsf{f}(\boldsymbol{x}, t)$$

Integration

- Stability issues can also arise
  - Occurs when errors lead to larger errors
  - Often more serious than error issues



$\dot{\boldsymbol{x}} = [\,-\sin(\omega t)\ ,\ -\cos(\omega t)\ ]$

Witkin and Baraff

Integration

- Modified Euler

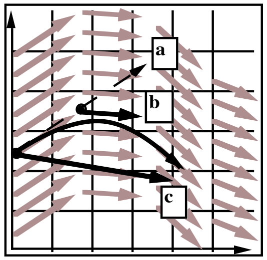
$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \frac{\Delta t}{2}\left(\dot{\boldsymbol{x}}^t + \dot{\boldsymbol{x}}^{t+\Delta t}\right)$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t\ \ddot{\boldsymbol{x}}^t$$

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t\ \dot{\boldsymbol{x}}^t + \frac{(\Delta t)^2}{2}\ \ddot{\boldsymbol{x}}^t$$

Integration

- Midpoint method
  a. Compute half Euler step
  b. Eval. derivative at halfway
  c. Retake step
- Other methods
  - Verlet
  - Runge-Kutta
  - And ***many*** others...



Witkin and Baraff

Integration

- Implicit methods
  - Informally (incorrectly) called backward methods
  - Use derivatives in the future for the current step

$$\boldsymbol{x}^{t+\Delta t} = \boldsymbol{x}^t + \Delta t\ \dot{\boldsymbol{x}}^{t+\Delta t}$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^t + \Delta t\ \ddot{\boldsymbol{x}}^{t+\Delta t}$$

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \mathsf{V}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}, t+\Delta t)$$
$$\ddot{\boldsymbol{x}}^{t+\Delta t} = \mathsf{A}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}, t+\Delta t)$$

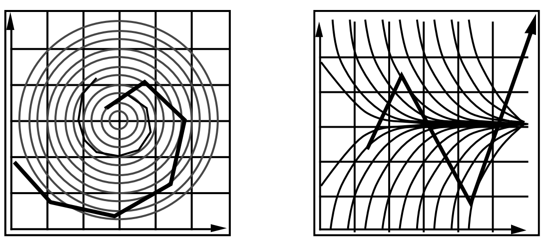
Integration

- Implicit methods
  - Informally (incorrectly) called backward methods
  - Use derivatives in the future for the current step

$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t \; \mathsf{V}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}, t+\Delta t)$$
$$\dot{\boldsymbol{x}}^{t+\Delta t} = \dot{\boldsymbol{x}}^{t} + \Delta t \; \mathsf{A}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}, t+\Delta t)$$

  - Solve nonlinear problem for $\boldsymbol{x}^{t+\Delta t}$ and $\dot{\boldsymbol{x}}^{t+\Delta t}$
  - This is fully implicit backward Euler
  - Many other implicit methods exist...
  - Modified Euler is ***partially*** implicit as is Verlet

Temp Slide



Need to draw reverse diagrams....

Integration

- Semi-Implicit
  - Approximate with linearized equations

$$\mathsf{V}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}) \approx \mathsf{V}(\boldsymbol{x}^t, \dot{\boldsymbol{x}}^t) + \mathbf{A} \cdot (\Delta \boldsymbol{x}) + \mathbf{B} \cdot (\Delta \dot{\boldsymbol{x}})$$

$$\mathsf{A}(\boldsymbol{x}^{t+\Delta t}, \dot{\boldsymbol{x}}^{t+\Delta t}) \approx \mathsf{A}(\boldsymbol{x}^t, \dot{\boldsymbol{x}}^t) + \mathbf{C} \cdot (\Delta \boldsymbol{x}) + \mathbf{D} \cdot (\Delta \dot{\boldsymbol{x}})$$

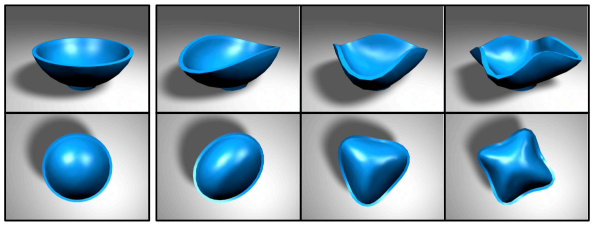
$$\begin{bmatrix} \boldsymbol{x}^{t+\Delta t} \\ \dot{\boldsymbol{x}}^{t+\Delta t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^t \\ \dot{\boldsymbol{x}}^t \end{bmatrix} + \Delta t \left(\begin{bmatrix} \dot{\boldsymbol{x}}^t \\ \ddot{\boldsymbol{x}}^t \end{bmatrix} + \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \dot{\boldsymbol{x}} \end{bmatrix} \right)$$

Integration

- Explicit methods can be conditionally stable
  - Depends on time-step and ***stiffness*** of system
- Fully implicit can be **un**conditionally stable
  - May still have large errors
- Semi-implicit can be conditionally stable
  - Nonlinearities can cause instability
  - Generally more stable than explicit
  - Comparable errors as explicit
    - Often show up as excessive damping

Integration

- Integrators can be analyzed in modal domain
- System have different component behaviors
- Integrators impact components differently



Suggested Reading

- Physically Based Modeling: Principles and Practice
  - Andy Witkin and David Baraff
  - http://www-2.cs.cmu.edu/~baraff/sigcourse/index.html
- Numerical Recipes in C++
  - Chapter 16
- Any good text on integrating ODE's