Real-time Game Physics

Practical Implementation:
Numerical Simulation

What is Numerical Simulation?

- Equations of frictionless collision response
  - They are "closed-form"
  - Valid and exact for constant applied force
  - Do not require time-stepping
    - Just determine current game time, t, using system timer
    - Plug t and t_{init} into the equations
    - Equations produce identical, repeatable, stable results, for any time, t, regardless of CPU speed and frame rate

What is Numerical Simulation?

- The above sounds perfect
- Why not use those equations always?
  - Constant forces aren't very interesting
    - Simple projectiles only
  - Closed-form solutions rarely exist for interesting (non-constant) forces
- We need a way to deal when there is no closed-form solution…

Numerical Simulation represents a series of techniques for incrementally solving the equations of motion when forces applied to an object are not constant, or when otherwise there is no closed-form solution

Finite Difference Methods

- What are They?
  - The most common family of numerical techniques for rigid-body dynamics simulation
  - Incremental “solution” to equations of motion
  - Derived using truncated Taylor Series expansions
- “Numerical Integrator”
  - This is what we generically call a finite difference equation that generates a “solution” over time

Finite Difference Methods

- The ***Explicit Euler*** Integrator:

$$\underbrace{\mathbf{S}(t + \Delta t)}_{\text{new state}} = \underbrace{\mathbf{S}(t)}_{\text{prior state}} + \Delta t \underbrace{\frac{d}{dt}\mathbf{S}(t)}_{\text{state derivative}}$$

- Properties of object are stored in a state vector, $\mathbf{S}$
- Use the above integrator equation to incrementally update $\mathbf{S}$ over time as game progresses
- Must keep track of prior value of $\mathbf{S}$ in order to compute the new
- For Explicit Euler, one choice of state and state derivative for particle:

$$\mathbf{S} = \langle m\mathbf{V}, \mathbf{p} \rangle \qquad d\mathbf{S}/dt = \langle \mathbf{F}, \mathbf{V} \rangle$$

Explicit Euler Integration

V_{init} = 30 m/s
Launch angle, ϕ: 75.2 deg (slow arrival)
Launch angle, θ: 0 deg (motion in world xz plane)
Mass of projectile, m: 2.5 kg
Target at <50, 0, 20> meters

Time	Position (m)			Linear Momentum (kg-m/s)			Force (N)			Velocity (m/s)		
	p_x	p_y	p_z	mV_x	mV_y	mV_z	F_x	F_y	F_z	V_x	V_y	V_z
5.00	10.00	0.00	2.00	19.20	0.00	72.50	0.00	0.00	-24.53	7.68	0.00	29.00

t_{init} $\mathbf{p}_{init}$ $m\mathbf{V}_{init}$ $\mathbf{F}=\mathbf{Weight} = m\mathbf{g}$ $\mathbf{V}_{init}$

$$\mathbf{S} = <m\mathbf{V}_{init}, \mathbf{p}_{init}>$$

$$d\mathbf{S}/dt = <m\mathbf{g}, \mathrm{V}_{init}>$$

Explicit Euler Integration

	$\Delta t = .2\ s$	$\Delta t = .1\ s$	$\Delta t = .01\ s$

$$\mathbf{S}(t+\Delta t) = \mathbf{S}(t) + \Delta t \frac{d}{dt}\mathbf{S}(t) = \begin{bmatrix} 19.2 \\ 0.0 \\ 72.5 \\ 10.0 \\ 0.0 \\ 2.0 \end{bmatrix} + \Delta t \begin{bmatrix} 0.0 \\ 0.0 \\ -24.53 \\ 7.68 \\ 0.0 \\ 29.0 \end{bmatrix} = \begin{bmatrix} 19.2025 \\ 0.0 \\ 67.5951 \\ 11.5362 \\ 0.0 \\ 7.8000 \end{bmatrix} = \begin{bmatrix} 19.2025 \\ 0.0 \\ 72.0476 \\ 10.7681 \\ 0.0 \\ 4.9000 \end{bmatrix} = \begin{bmatrix} 19.2025 \\ 0.0 \\ 72.2549 \\ 10.0768 \\ 0.0 \\ 2.2900 \end{bmatrix}$$

$$\text{Exact, Closed-form Solution} = \begin{bmatrix} 19.2 \\ 0.0 \\ 67.5951 \\ 11.5362 \\ 0.0 \\ 7.6038 \end{bmatrix} = \begin{bmatrix} 19.2 \\ 0.0 \\ 72.0476 \\ 10.1536 \\ 0.0 \\ 4.8510 \end{bmatrix} = \begin{bmatrix} 19.2 \\ 0.0 \\ 72.2549 \\ 10.0768 \\ 0.0 \\ 2.2895 \end{bmatrix}$$

A Tangent: Truncation Error

- The previous slide highlights values in the numerical solution that are different from the exact, closed-form solution
- This difference between the exact solution and the numerical solution is primarily ***truncation error***
- Truncation error is equal and opposite to the value of terms that were removed from the Taylor Series expansion to produce the finite difference equation
- **Truncation error, left unchecked, can accumulate to cause simulation to become unstable**
  - This ultimately produces floating point overflow
  - Unstable simulations behave unpredictably

A Tangent: Truncation Error

- Controlling Truncation Error
  - Under certain circumstances, truncation error can become zero, *e.g.*, the finite difference equation produces the exact, correct result
    - For example, when zero force is applied
  - More often in practice, truncation error is nonzero
  - Approaches to control truncation error:
    - Reduce time step, Δt
    - Select a different numerical integrator
  - See text for more background information and references

Explicit Euler Integration – Truncation Error

Lets Look at Truncation Error (position only)

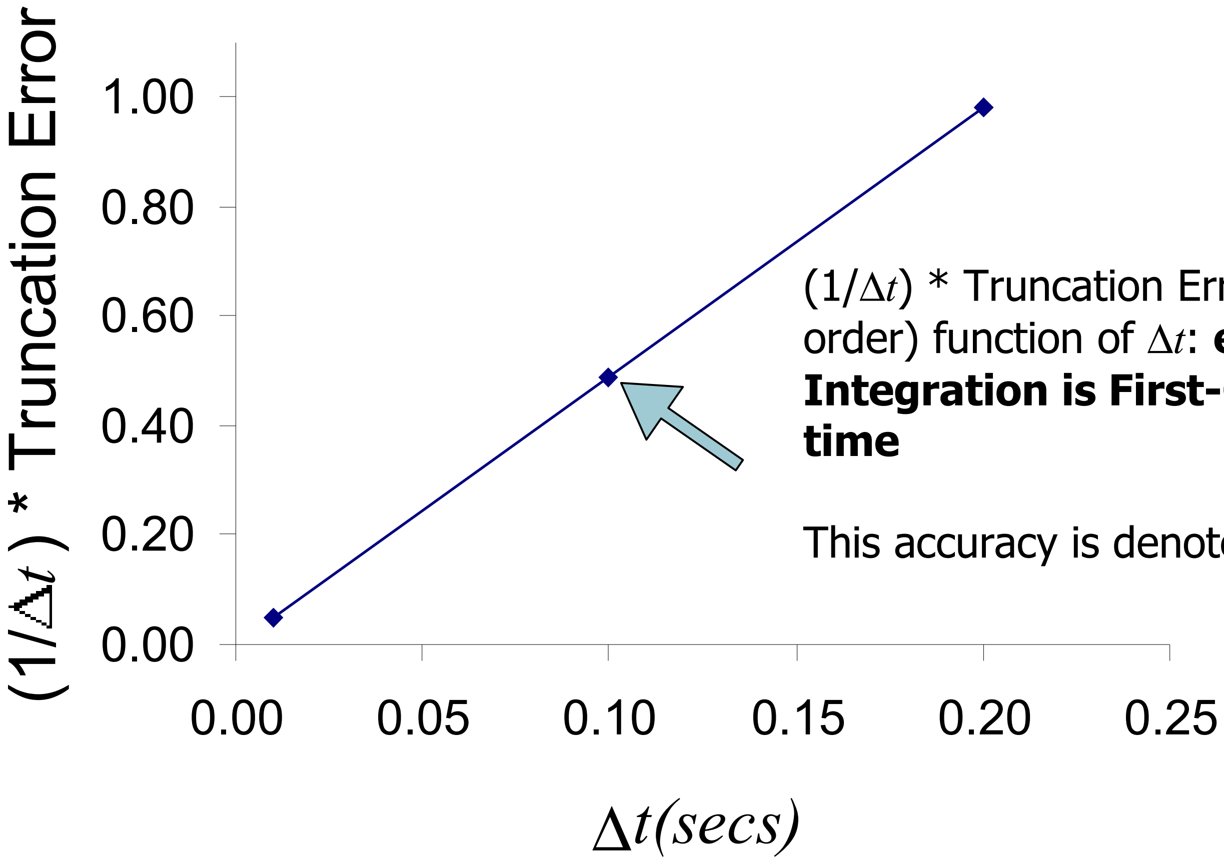
Truncation Error

$$\text{Truncation Error}(\Delta t = 0.2s) = \begin{bmatrix} 11.5362 \\ 0.0 \\ 7.800 \end{bmatrix}_{\text{numerical}} - \begin{bmatrix} 11.5362 \\ 0.0 \\ 7.6038 \end{bmatrix}_{\text{exact}} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.1962 \end{bmatrix}$$

$$\text{Truncation Error}(\Delta t = 0.1s) = \begin{bmatrix} 10.1536 \\ 0.0 \\ 4.9000 \end{bmatrix}_{\text{numerical}} - \begin{bmatrix} 10.1536 \\ 0.0 \\ 4.8510 \end{bmatrix}_{\text{exact}} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.049 \end{bmatrix}$$

$$\text{Truncation Error}(\Delta t = 0.01s) = \begin{bmatrix} 10.0768 \\ 0.0 \\ 2.2900 \end{bmatrix}_{\text{numerical}} - \begin{bmatrix} 10.0768 \\ 0.0 \\ 2.2895 \end{bmatrix}_{\text{exact}} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0005 \end{bmatrix}$$

Explicit Euler Integration – Truncation Error



(1/Δt) * Truncation Error is a linear (first-order) function of Δt: **explicit Euler Integration is First-Order-Accurate in time**

This accuracy is denoted by "O(Δt)"

Explicit Euler Integration - Computing Solution Over Time

- The solution proceeds step-by-step, each time integrating from the prior state

Time	Position (m)			Linear Momentum (kg-m/s)			Force (N)			Velocity (m/s)		
	p_x	p_y	p_z	mV_x	mV_y	mV_z	F_x	F_y	F_z	V_x	V_y	V_z
5.00	10.00	0.00	2.00	19.20	0.00	72.50	0.00	0.00	-24.53	7.68	0.00	29.00
5.20	11.54	0.00	7.80	19.20	0.00	67.60	0.00	0.00	-24.53	7.68	0.00	27.04
5.40	13.07	0.00	13.21	19.20	0.00	62.69	0.00	0.00	-24.53	7.68	0.00	25.08
5.60	14.61	0.00	18.22	19.20	0.00	57.79	0.00	0.00	-24.53	7.68	0.00	23.11
⋮		⋮			⋮			⋮			⋮	
10.40	51.48	0.00	20.87	19.20	0.00	-59.93	0.00	0.00	-24.53	7.68	0.00	-23.97