Notes

- Finish up time integration methods today
- Assignment 1 is mostly out
  - Later today will make it compile etc.

Time scales

- [work out]
- For position dependence, characteristic time interval is
$$\Delta t = O\left(\frac{1}{\sqrt{K}}\right)$$
- For velocity dependence, characteristic time interval is
$$\Delta t = O\left(\frac{1}{D}\right)$$
- Note: matches symplectic Euler stability limits
  - If you care about resolving these time scales, there's not much point in going to implicit methods

Mixed Implicit/Explicit

- For some problems, that square root can mean velocity limit much stricter
- Or, we know we want to properly resolve the position-based oscillations, but don't care about damping
- Go explicit on position, implicit on velocity
  - Cuts the number of equations to solve in half
  - Often, a(x,v) is linear in v, though nonlinear in x; this way we avoid Newton iteration

Newmark Methods

- A general class of methods
$$x_{n+1} = x_n + \Delta t v_n + \tfrac{1}{2}\Delta t^2\left[(1-2\beta)a_n + 2\beta a_{n+1}\right]$$
$$v_{n+1} = v_n + \Delta t\left[(1-\gamma)a_n + \gamma a_{n+1}\right]$$
- Includes Trapezoidal Rule for example (β=1/4, γ=1/2)
- The other major member of the family is Central Differencing (β=0, γ=1/2)
  - This is mixed Implicit/Explicit

Central Differencing

- Rewrite it with intermediate velocity:
$$v_{n+\frac{1}{2}} = v_n + \tfrac{1}{2}\Delta t\, a(x_n, v_n)$$
$$x_{n+1} = x_n + \Delta t\, v_{n+\frac{1}{2}}$$
$$v_{n+1} = v_{n+\frac{1}{2}} + \tfrac{1}{2}\Delta t\, a(x_{n+1}, v_{n+1})$$
- Looks like a hybrid of:
  - Midpoint (for position), and
  - Trapezoidal Rule (for velocity - split into Forward and Backward Euler half steps)

Central: Performance

- Constant acceleration: great
  - 2nd order accurate
- Position dependence: good
  - Conditionally stable, no damping
- Velocity dependence: good
  - Stable, but only conditionally monotone
- Can we change the Trapezoidal Rule to Backward Euler and get unconditional monotonicity?

Staggered Implicit/Explicit

- Like the staggered Symplectic Euler, but use B.E. in velocity instead of F.E.:

$$v_{n+\frac{1}{2}} = v_{n-\frac{1}{2}} + \tfrac{1}{2}(t_{n+1} - t_{n-1})a\left(x_n, v_{n+\frac{1}{2}}\right)$$
$$x_{n+1} = x_n + \Delta t v_{n+\frac{1}{2}}$$

- Constant acceleration: great
- Position dependence: good (conditionally stable, no damping)
- Velocity dependence: great (unconditionally monotone)

Summary (2nd order)

- Depends a lot on the problem
  - What's important: gravity, position, velocity?
- Explicit methods from last class are probably bad
- Symplectic Euler is a great fully explicit method (particularly with staggering)
  - Switch to implicit velocity step for more stability, if damping time step limit is the bottleneck
- Implicit Compromise method
  - Fully stable, nice behaviour

Example Motions

Simple Velocity Fields

- Can superimpose (add) to get more complexity
- Constants: v(x)=constant
- Expansion/contraction: $v(x)=k(x-x_0)$
  - Maybe make k a function of distance $|x-x_0|$
- Rotation: $v(x) = \omega \times (x - x_0)$
  - Maybe scale by a function of distance $|x-x_0|$ or magnitude $|\omega \times (x - x_0)|$

Noise

- Common way to perturb fields that are too perfect and clean
- Noise (in graphics) =
  a smooth, non-periodic field with clear length-scale
- Read Perlin, "Improving Noise", SIGGRAPH'02
  - Hash grid points into an array of random slopes that define a cubic Hermite spline
- Can also use a Fourier construction
  - Band limited signal
  - Better, more control, but (possibly much) more expensive
  - FFT - check out www.fftw.org for one good implementation

Example Forces

- Gravity: $F_{gravity}=mg$ (a=g)
- If you want to do orbits

$$F_{gravity} = -GmM_0 \frac{x - x_0}{|x - x_0|^3}$$

- Note x_0 could be a fixed point (e.g. the Sun) or another particle
  - But make sure to add the opposite and equal force to the other particle if so!

Drag Forces

- Air drag: $F_{drag}=-Dv$
  - If there's a wind blowing with velocity v_w then $F_{drag}=-D(v-v_w)$
- D should be a function of the cross-section exposed to wind
  - Think paper, leaves, different sized objects, …
- Depends in a difficult way on shape too
  - Hack away!

Spring Forces

- Springs: $F_{spring}=-K(x-x_0)$
  - x_0 is the attachment point of the spring
  - Could be a fixed point in the scene
  - …or somewhere on a character's body
  - …or the mouse cursor
  - …or another particle (but please add equal and oppposite force!)

Nonzero Rest Length Spring

- Need to measure the "strain": the fraction the spring has stretched from its rest length L

$$F_{spring} = -K\left(\frac{|x-x_0|}{L}-1\right)\frac{x-x_0}{|x-x_0|}$$

Spring Damping

- Simple damping: $F_{damp}=-D(v-v_0)$
  - But this damps rotation too!
- Better spring damping:

  $F_{damp}=-D(v-v_0)\bullet u\ u$
  - Here u is $(x-x_0)/|x-x_0|$, the spring direction
- [work out 1d case]
- Critical damping

$$D = 2\sqrt{mK}$$

Collision and Contact

Collision and Contact

- We can integrate particles forward in time, have some ideas for velocity or force fields
- But what do we do when a particle hits an object?
- No simple answer, depends on problem as always
- General breakdown:
  - Interference vs. collision detection
  - What sort of collision response: (in)elastic, friction
  - Robustness: do we allow particles to actually be inside an object?

Interference vs. Collision

- Interference (=penetration)
  - Simply detect if particle has ended up inside object, push it out if so
  - Works fine if $v\Delta t < \frac{1}{2}w$ [w=object width]
  - Otherwise could miss interaction, or push dramatically the wrong way
  - The ground, thick objects and slow particles
- Collision
  - Check if particle trajectory intersects object
  - Can be more complicated, especially if object is moving too…
- For now, let's stick with the ground (y=0)

Repulsion Forces

- Simplest idea (conceptually)
  - Add a force repelling particles from objects when they get close (or when they penetrate)
  - Then just integrate: business as usual
  - Related to penalty method:
    instead of directly enforcing constraint (particles stay outside of objects), add forces to encourage constraint
- For the ground:
  - Frepulsion=-Ky when y<0 [think about gravity!]
  - …or -K(y-y0)-Dv when y<y0 [still not robust]
  - …or K(1/y-1/y0)-Dv when y<y0

Repulsion forces

- Difficult to tune:
  - Too large extent: visible artifact
  - Too small extent: particles jump straight through, not robust (or time step restriction)
  - Too strong: stiff time step restriction, or have to go with implicit method - but Newton will not converge if we guess past a singular repulsion force
  - Too weak: won't stop particles
- Rule-of-thumb: don't use them unless they really are part of physics
  - Magnetic field, aerodynamic effects, …

Collision and Contact

- Collision is when a particle hits an object
  - Instantaneous change of velocity (discontinuous)
- Contact is when particle stays on object surface for positive time
  - Velocity is continuous
  - Force is only discontinuous at start

Frictionless Collision Response

- At point of contact, find normal n
  - For ground, n=(0,1,0)
- Decompose velocity into
  - normal component v_N=(v•n)n and
  - tangential component v_T=v-v_N
- Normal response: $v_N^{after} = -\varepsilon v_N^{before}, \quad \varepsilon \in [0,1]$
  - ε=0 is fully inelastic
  - ε=1 is elastic
- Tangential response
  - Frictionless: $v_T^{after} = v_T^{before}$
- Then reassemble velocity v=v_N+v_T

Contact Friction

- Some normal force is keeping v_N=0
- Coulomb's law ("dry" friction)
  - If sliding, then kinetic friction:

$$F_{friction} = -\mu_k |F_{normal}| \frac{v_T}{|v_T|}$$

  - If static (v_T=0) then stay static as long as

$$|F_{friction}| \le \mu_s |F_{normal}|$$

- "Wet" friction = damping

$$F_{friction} = -D|F_{normal}| v_T$$

Collision Friction

- Impulse assumption:
  - Collision takes place over a very small time interval (with very large forces)
  - **Assume** forces don't vary significantly over that interval---then can replace forces in friction laws with impulses
  - This is a little controversial, and for articulated rigid bodies can be demonstrably false
  - But nevertheless…
  - Normal impulse is just $m\Delta v_N = m(1+\varepsilon)v_N$
  - Tangential impulse is $m\Delta v_T$

Wet Collision Friction

- So replacing force with impulse:

$$m\Delta v_T = -D|m\Delta v_N| v_T$$

- Divide through by m, use $v_T^{after} = v_T^{before} + \Delta v_T$

$$\begin{aligned} v_T^{after} &= v_T^{before} - D|\Delta v_N| v_T^{before} \\ &= (1 - D|\Delta v_N|) v_T^{before} \end{aligned}$$

- Clearly could have monotonicity/stability issue
- Fix by capping at $v_T=0$, or better approximation for time interval e.g. $v_T^{after} = e^{-D|\Delta v_N|} v_T^{before}$

Dry Collision Friction

- Coulomb friction: assume $\mu_s = \mu_k$
  - (though in general, $\mu_s \geq \mu_k$)
- Sliding: $m\Delta v_T = -\mu|m\Delta v_N| \frac{v_T^{before}}{|v_T^{before}|}$
- Static: $|m\Delta v_T| \leq \mu|m\Delta v_N|$
- Divide through by m to find change in tangential velocity

Simplifying…

- Use $v_T^{after} = v_T^{before} + \Delta v_T$
- Static case is $v_T^{after} = 0 \Rightarrow \Delta v_T = -v_T^{before}$ when $|v_T^{before}| \leq \mu|\Delta v_N|$
- Sliding case is

$$v_T^{after} = v_T^{before} - \mu|\Delta v_N| \frac{v_T^{before}}{|v_T^{before}|}$$

- Common quantities!

Dry Collision Friction Formula

- Combine into a max
  - First case is static where v_T drops to zero if inequality is obeyed
  - Second case is sliding, where v_T reduced in magnitude (but doesn't change signed direction)

$$v_T^{after} = \max\left(0, 1 - \frac{\mu|\Delta v_N|}{|v_T^{before}|}\right) v_T^{before}$$

Where are we?

- So we now have a simplified physics model for
  - Frictionless, dry friction, and wet friction collision
  - Some idea of what contact is
- So now let's start on numerical methods to simulate this

"Exact" Collisions

- For very simple systems (linear or maybe parabolic trajectories, polygonal objects)
  - Find exact collision time (solve equations)
  - Advance particle to collision time
  - Apply formula to change velocity
    (usually dry friction, unless there is lubricant)
  - Keep advancing particle until end of frame or next collision
- Can extend to more general cases with conservative ETA's, or root-finding techniques
- **Expensive** for lots of coupled particles!

Fixed collision time stepping

- Even "exact" collisions are not so accurate in general
  - [hit or miss example]
- So instead fix $\Delta t_{collision}$ and don't worry about exact collision times
  - Could be one frame, or 1/8th of a frame, or …
- Instead just need to know did a collision happen during $\Delta t_{collision}$
  - If so, process it with formulas

Relationship with regular time integration

- Forgetting collisions, advance from x(t) to $x(t+\Delta t_{collision})$
  - Could use just one time step, or subdivide into lots of small time steps
- We approximate velocity (for collision processing) as constant over time step:

$$v = \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

- If no collisions, forget this average v, and keep going with underlying integration

Numerical Implementation 1

- Get candidate $x(t+\Delta t)$
- Check to see if $x(t+\Delta t)$ is inside object (interference)
- If so
  - Get normal n at $t+\Delta t$
  - Get new velocity v from collision response formulas and average v
  - Replay $x(t+\Delta t)=x(t)+\Delta t v$