533D: Animation Physics

- http://www.cs.ubc.ca/~rbridson/courses/533d-winter-2005
- Course schedule
  - Slides online, but you need to take notes too!
- Reading
  - Relevant animation papers as we go
- Assignments + Final Project information
- Resources

Contacting Me

- Robert Bridson
  - CICSR 189 (moving to CS2 in reading week)
  - Drop by, or make an appointment
  - 604-822-1993 (or just 21993)
  - email rbridson@cs.ubc.ca
  - Newsgroup ubc.courses.cpsc.533b
- I always like feedback!

Evaluation

- 4 assignments (60%)
  - See the web for details + when they are due
  - Mostly programming, with a little analysis (writing)
- Also a final project (40%)
  - Details will come later, but basically you need to either significantly extend an assignment or animate something else - talk to me about topics
  - Present in final class - informal talk, show movies
- Late: without a good reason, 20% off per day
  - For final project starts after final class
  - For assignments starts morning after due

Why?

- Natural phenomena: passive motion
- Film/TV: difficult with traditional techniques
  - When you control every detail of the motion, it's hard to make it look like it's not being controlled!
- Games: difficult to handle everything convincingly with prescripted motion
- Computer power is increasing, audience expectations are increasing, artist power isn't: need more automatic methods

- Directly simulate the underlying physics to get realistic motion

Topics

- Particle Systems
  - the basics
- Deformable Bodies
  - e.g. cloth and flesh
- Constrained Dynamics
  - e.g. rigid bodies
- Fluids
  - e.g. water

Particle Systems

Particle Systems

- Read:
  Reeves, “Particle Systems…”, SIGGRAPH’83
  Sims, “Particle animation and rendering using data parallel computation", SIGGRAPH '90
- Some phenomena is most naturally described as many small particles
  - Rain, snow, dust, sparks, gravel, …
- Others are difficult to get a handle on
  - Fire, water, grass, …

Particle Basics

- Each particle has a position
  - Maybe orientation, age, colour, velocity, temperature, radius, …
  - Call the state x
- Seeded randomly somewhere at start
  - Maybe some created each frame
- Move (evolve state x) each frame according to some formula
- Eventually die when some condition met

Example

- Sparks from a campfire
- Every frame (1/24 s) add 2-3 particles
  - Position randomly in fire
  - Initialize temperature randomly
- Move in specified turbulent smoke flow
  - Also decrease temperature
- Render as a glowing dot (blackbody radiation from temperature)
- Kill when too cold to glow visibly

Rendering

- We won’t talk much about rendering in this course, but most important for particles
- The real strength of the idea of particle systems: how to render
  - Could just be coloured dots
  - Or could be shards of glass, or animated sprites (e.g. fire), or deforming blobs of water, or blades of grass, or birds in flight, or …

First Order Motion

First Order Motion

- For each particle, have a simple 1st order differential equation:

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

- Analytic solutions hopeless
- Need to solve this numerically forward in time from x(t=0) to x(frame1), x(frame2), x(frame3), …
  - May be convenient to solve at some intermediate times between frames too

Forward Euler

- Simplest method:

$$\frac{x_{n+1} - x_n}{\Delta t} = v(x_n, t_n)$$

Or:

$$x_{n+1} = x_n + \Delta t\, v(x_n, t_n)$$

- Can show it's first order accurate:
  - Error accumulated by a fixed time is O(Δt)
- Thus it converges to the right answer
  - Do we care?

Aside on Error

- General idea - want error to be small
  - Obvious approach: make Δt small
  - But then need more time steps - expensive
- Also note - O(1) error made in modeling
  - Even if numerical error was 0, still wrong!
  - In science, need to validate against experiments
  - In graphics, the experiment is showing it to an audience: **does it look real?**
- So numerical error can be huge, as long as your solution has the right qualitative look

Forward Euler Stability

- Big problem with Forward Euler: it's not very stable
- Example: $dx/dt = -x, \quad x(0) = 1$
- Real solution e^{-t} smoothly decays to zero, always positive
- Run Forward Euler with Δt=11
  - x=1, -10, 100, -1000, 10000, …
  - Instead of 1, $1.7*10^{-5}$, $2.8*10^{-10}$, …

Linear Analysis

- Approximate

$$v(x,t) \approx v(x^*,t^*) + \frac{\partial v}{\partial x} \cdot (x - x^*) + \frac{\partial v}{\partial t} \cdot (t - t^*)$$

- Ignore all but the middle term (the one that could cause blow-up)

$$dx/dt = Ax$$

- Look at x parallel to eigenvector of A: the "test equation" $dx/dt = \lambda x$

The Test Equation

- Get a rough, hazy, heuristic picture of the stability of a method
- Note that eigenvalue λ can be complex
- But, assume that for real physics
  - Things don't blow up without bound
  - Thus **real** part of eigenvalue λ is ≤ 0
- Beware!
  - Nonlinear effects can cause instability
  - Even with linear problems, what follows assumes constant time steps - varying (but supposedly stable) steps can induce instability
    - see J. P. Wright, "Numerical instability due to varying time steps…", JCP 1998

Using the Test Equation

- Forward Euler on test equation is

$$x_{n+1} = x_n + \Delta t \lambda x_n$$

- Solving gives

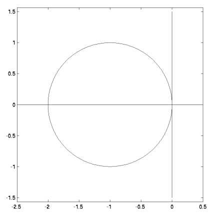
$$x_n = (1 + \lambda \Delta t)^n x_0$$

- So for stability, need

$$|1 + \lambda \Delta t| < 1$$

Stability Region

- Can plot all the values of $\lambda\Delta t$ on the complex plane where F.E. is stable:



Real Eigenvalue

- Say eigenvalue is real (and negative)
  - Corresponds to a damping motion, smoothly coming to a halt
- Then need:

$$\Delta t < \frac{2}{|\lambda|}$$

- Is this bad?
  - If eigenvalue is big, could mean small time steps
  - But, maybe we really need to capture that time scale anyways, so no big deal

Imaginary Eigenvalue

- If eigenvalue is pure imaginary…
  - Oscillatory or rotational motion
- Cannot make Δt small enough
- Forward Euler unconditionally unstable for these kinds of problems!
- Need to look at other methods

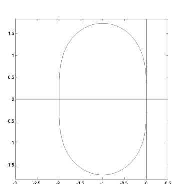
Runge-Kutta Methods

- Also “explicit”
  - next x is an explicit function of previous
- But evaluate v at a few locations to get a better estimate of next x
- E.g. midpoint method (one of RK2)

$$x_{n+1/2} = x_n + \tfrac{1}{2}\Delta t v(x_n, t_n)$$
$$x_{n+1} = x_n + \Delta t v\left(x_{n+1/2}, t_{n+1/2}\right)$$

Midpoint RK2

- Second order: error is $O(\Delta t^2)$ **when smooth**
- Larger stability region:



- But still not stable on imaginary axis: no point

Modified Euler

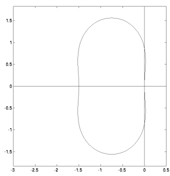
- (Not an official name)
- Lose second-order accuracy, get stability on imaginary axis:

$$x_{n+\alpha} = x_n + \alpha\Delta t v(x_n, t_n)$$
$$x_{n+1} = x_n + \Delta t v(x_{n+\alpha}, t_{n+\alpha})$$

- Parameter α between 0.5 and 1 gives trade-off between imaginary axis and real axis

Modified Euler (2)

- Stability region for α=2/3

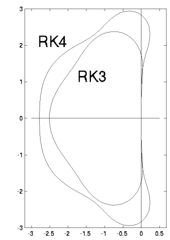


υ Great! But twice the cost of Forward Euler

- Can you get more stability per v-evaluation?

Higher Order Runge-Kutta

- RK3 and up naturally include part of the imaginary axis



TVD-RK3

- RK3 useful because it can be written as a combination of Forward Euler steps and averaging: can guarantee stuff!

$$\begin{aligned}
\tilde{x}_{n+1} &= x_n + \Delta t v(x_n, t_n) \\
\tilde{x}_{n+2} &= \tilde{x}_{n+1} + \Delta t v(\tilde{x}_{n+1}, t_{n+1}) \\
\tilde{x}_{n+1/2} &= \tfrac{3}{4} x_n + \tfrac{1}{4} \tilde{x}_{n+2} \\
\tilde{x}_{n+3/2} &= \tilde{x}_{n+1/2} + \Delta t v\left(\tilde{x}_{n+1/2}, t_{n+1/2}\right) \\
x_{n+1} &= \tfrac{1}{3} x_n + \tfrac{2}{3} \tilde{x}_{n+3/2}
\end{aligned}$$

RK4

- Often most bang for the buck

$$\begin{aligned}
v_1 &= v(x_n, t_n) \\
v_2 &= v\left(x_n + \tfrac{1}{2}\Delta t v_1, t_{n+1/2}\right) \\
v_3 &= v\left(x_n + \tfrac{1}{2}\Delta t v_2, t_{n+1/2}\right) \\
v_4 &= v(x_n + \Delta t v_3, t_{n+1}) \\
x_{n+1} &= x_n + \Delta t\left(\tfrac{1}{6} v_1 + \tfrac{2}{6} v_2 + \tfrac{2}{6} v_3 + \tfrac{1}{6} v_4\right)
\end{aligned}$$

Selecting Time Steps

Selecting Time Steps

- Hack: try until it looks like it works
- Stability based:
  - Figure out a bound on magnitude of Jacobian
  - Scale back by a fudge factor (e.g. 0.9, 0.5)
    - Try until it looks like it works… (remember all the dubious assumptions we made for linear stability analysis!)
    - Why is this better than just hacking around in the first place?
- Adaptive error based:
  - Usually not worth the trouble in graphics

Time Stepping

- Sometimes can pick constant Δt
  - One frame, or 1/8th of a frame, or …
- Often need to allow for variable Δt
  - Changing stability limit due to changing Jacobian
  - Difficulty in Newton converging
  - …
- But prefer to land at the exact frame time
  - So clamp Δt so you can't overshoot the frame

Example Time Stepping Algorithm

- Set done = false
- While not done
  - Find good Δt
  - If $t+\Delta t \geq t_{frame}$
    - Set $\Delta t = t_{frame}-t$
    - Set done = true
  - Else if $t+1.5\Delta t \geq t_{frame}$
    - Set $\Delta t = 0.5(t_{frame}-t)$
  - …process time step…
  - Set $t = t+\Delta t$
- Write out frame data, continue to next frame

Implicit Methods

Large Time Steps

- Look at the test equation $\frac{dx}{dt} = \lambda x$
- Exact solution is

$$x(t_{n+1}) = e^{\lambda \Delta t} x(t_n) = \left(1 + \lambda \Delta t + \tfrac{1}{2}(\lambda \Delta t)^2 + \ldots\right) x(t_n)$$

- Explicit methods approximate this with polynomials (e.g. Taylor)
- Polynomials must blow up as t gets big
  - Hence explicit methods have stability limit
- We may want a different kind of approximation that drops to zero as Δt gets big
  - Avoid having a small stability limit when error says it should be fine to take large steps ("stiffness")

Simplest stable approximation

- Instead use $e^{\lambda \Delta t} \approx \frac{1}{1-\lambda \Delta t}$
- That is, $x_{n+1} = \frac{1}{1-\lambda \Delta t} x_n$
- Rewriting: $x_{n+1} = x_n + \Delta t\, \lambda x_{n+1}$
- This is an "implicit" method: the next x is an **implicit** function of the previous x
  - Need to solve equations to figure it out

Backward Euler

- The simplest implicit method:

$$x_{n+1} = x_n + \Delta t\, v\left(x_{n+1}, t_{n+1}\right)$$

- First order accurate
- Test equation shows stable when $|1-\lambda \Delta t| > 1$
- This includes everything except a circle in the positive real-part half-plane
- It's stable even when the physics is unstable!
- This is the biggest problem: damps out motion unrealistically

Aside: Solving Systems

- If v is linear in x, just a system of linear equations
  - If very small, use determinant formula
  - If small, use LAPACK
  - If large, life gets more interesting…
- If v is **mildly** nonlinear, can approximate with linear equations ("semi-implicit")

$$\begin{aligned} x_{n+1} &= x_n + \Delta t\, v(x_{n+1}) \\ &\approx x_n + \Delta t\left(v(x_n) + \frac{\partial v(x_n)}{\partial x}(x_{n+1} - x_n)\right) \end{aligned}$$

Newton's Method

- For more strongly nonlinear v, need to iterate:
  - Start with guess x_n for x_{n+1} (for example)
  - Linearize around current guess, solve linear system for next guess
  - Repeat, until close enough to solved
- Note: Newton's method is **great** when it works, but it might not work
  - If it doesn't, can reduce time step size to make equations easier to solve, and try again
  - Maybe use higher power optimization methods (e.g. at least use line search)

Newton's Method: B.E.

- Start with $x^0=x_n$ (simplest guess for x_{n+1})
- For k=1, 2, … find $x^{k+1}=x^k+\Delta x$ by solving

$$x^{k+1} = x_n + \Delta t\left(v(x^k) + \frac{\partial v(x^k)}{\partial x}(x^{k+1} - x^k)\right)$$

$$\Rightarrow \left(I - \Delta t \frac{\partial v(x^k)}{\partial x}\right)\Delta x = x_n + \Delta t\, v(x^k) - x^k$$

- To include line-search for more robustness, change update to $x^{k+1}=x^k+\alpha\Delta x$ and choose $0 < \alpha \le 1$ that minimizes $\left\| x_n + \Delta t\, v\left(x^{k+1}, t_{n+1}\right) - x^{k+1} \right\|$
- Stop when right-hand side is small enough, set $x_{n+1}=x^k$

Trapezoidal Rule

- Can improve by going to second order:

$$x_{n+1} = x_n + \Delta t\left(\tfrac{1}{2} v(x_n, t_n) + \tfrac{1}{2} v(x_{n+1}, t_{n+1})\right)$$

- This is actually just a half step of F.E., followed by a half step of B.E.
  - F.E. is under-stable, B.E. is over-stable, the combination is **just right**
- Stability region is the left half of the plane: **exactly** the same as the physics!
- Really good for pure rotation (doesn't amplify or damp)