Notes

- Errors last class:
  - Simple wrist model has 3DOF, not 2 (wave)
  - Finding $s(u)$ numerically: construct the spline from the table, with knots at the $s$ values and control values from the $u$'s
    - Then you don’t need secant search
- Electronic Arts Speaker Series:
  - September 23rd (Thursday) 2-3pm in Neville Scarfe room 203
  - Dr. Zoran Popovic - controlling animation
  - Tomorrow office hours may start late

More notes

- ROOM CHANGE:
  - Hugh Dempster Pavilion 101 (new building just behind CICSR)
  - We will meet there on Friday
- Assignment 1 is (partially) out
  - I will give you more code as soon as I have written it!
  - Get from the website: read as soon as you can and start asking questions if it’s unclear

Recall: Inverse Kinematics (IK)

- We have a hierarchical skeleton
- Fix part of it, solve for joint angles (or displacements) to put some part into given position/orientation
- Obvious application in robotics
- Used in graphics as an efficient UI for animating grasping, contact, foot plants, etc.
- Frequently not well posed

IK issues 2

- Target impossible (or just barely possible) --- not enough degrees of freedom
  - Try to get as close as possible to the target in some sense
  - Regularize by penalizing bad joint angles, huge differences from initial pose
    - So if pose is only just possible, via some weird unrealistic angles, we don’t do it
How do you do IK?

- Simple skeletons: analytic solutions may exist
  - Look up formulas from papers
  - Not fun, not flexible
- More generally: numerical solution of several simultaneous equations
  - Newton’s method
  - Or related optimization algorithms
  - Half the battle is just formulating the equations!

Newton’s Method

- Solve $F(x)=0$ where $F$ and $x$ are vectors
- Start with close initial guess $x_0$
- Assume that solution is nearby, and that the function is close to linear nearby:
  - $F(x)=F(x_0)+VF(x_0)(x-x_0)$
  - $VF$ is called the Jacobian
- Then solve approximate linear equations for next guess $x_i$:
  - $F(x_i)+VF(x_i)(x_i-x_0)=0$
- Keep iterating until $|F(x_{k+1})|$ is small enough or $|x_{k+1}-x_k|$ is small enough
  - $F(x_k)+VF(x_k)(x_{k+1}-x_k)=0$

Too many DOF

- If $VF$ isn’t square (or isn’t invertible)
  - simple solution is use pseudo-inverse
  - More on this in Applied Math Reference, coming soon to the website
- This solves the system in a least-squares sense
  - Comes as close as possible to answer
  - Picks smallest solution possible

Impossible goal

- Newton’s method may well go wild if goal is impossible
- Need to limit how much change to any joint angle
- The usual Newton step is
  - $F(x_k)+VF(x_k)(x_{k+1}-x_k)=0$
- Rewrite as:
  - Solve $J\Delta=-F(x_k)$ then update $x_{k+1}=x_k+\Delta$
- We can fix the bad behaviour by limiting the update: $x_{k+1}=x_k+\alpha\Delta$
- Choose $\alpha<1$ so that angles don’t change a lot
### More sophisticated IK

- Preceding algorithms may not be robust enough
- What if we need a true minimization?
- What if we have more constraints?
  - E.g. arm stays in given plane, skeleton doesn’t intersect itself or other geometry
- Need to go to optimization algorithms

### Character Rigging

- A “rig” is a model together with a UI for posing it
- At its simplest, a skeleton with joint angles available for motion curves
- May simplify DOF by enforcing relationships between joints
  - E.g. hand and fingers
- May define standard poses (especially for facial expressions!) that can be mixed together
  - Then can set sliders to, say, 70% happy, 20% surprised, ...
  - Take weighted linear combination of pose angles

### Breaking Rigs

- Who said animated figures had to have rigid parts?
  - Remember animation principles: stretch & squash, exaggeration, etc.
- Often attractive to break up a rigid skeleton into separate parts (e.g. torso, arms, legs, head)
  - Allow connecting links to change dimension as needed
  - Kinematics only done on a small part - artist doesn’t need to worry about effect on whole (local vs. global control)
- “If it ain’t broken, then fix it”

### What’s left?

- We now have the basics of animation
  - Except the biggest issue: user interface
- Plan for next while:
  - Rendering animations
  - (Semi-)automatic animation
    - Dynamics for rigid bodies
    - Particle systems
    - Skinning, morphing, blending
    - Motion capture
    - Motion control
Rendering for Film

Compositing

- The action of combining multiple “layers” -- parts of each frame -- into the final shot
  - E.g. background + actors + vfx
  - For vfx-intensive shots, there could be dozens of separate layers
- Handling each layer separately
  - makes the problem simpler,
  - allows better division of labour,
  - and gives flexibility in putting the elements together at the end

Atop

- The simplest (useful) and most common form of compositing: put one image “atop” another
  - Image 1 (RGB) on top of image 2 (RGB)
- For each pixel in the final composite, need to know what RGB value to take
  - Where image 1 is opaque, choose RGB₁
  - Where image 1 is “empty”, choose RGB₂
  - Where image 1 is partly transparent, or where image 1 only covers part of the pixel?