Notes

- Electronic Arts Speaker Series:
  - September 23rd (Thursday) 2-3pm in Neville Scarfe room 203
  - Dr. Zoran Popovic - controlling animation
- Textbook reference for arc-length parameterization:
  - Section 3.2

Arc Length

- Arc length is just the length of a curve
  - Think of wrapping a tape measure along the curve
- Definition:

$$s(u) = \int_0^u \left| \frac{d\vec{x}}{du} \right| du$$

  - Where x(u) is the 3d position of the curve at parameter value u
    - Really three curves: X(u), Y(u), Z(u)
  - Recall how to measure vector norm:

$$\left| \frac{d\vec{x}}{du} \right| = \sqrt{\left(\frac{dX}{du}\right)^2 + \left(\frac{dY}{du}\right)^2 + \left(\frac{dZ}{du}\right)^2}$$

Inverse Map

- The question we really want to answer, though, is what value of u gives us a specific length s along the curve?
  - I.e. invert the arc length function s(u)
  - Let's call this u(s)
- Then timing curve is s(t), which feeds into u(s), which feeds into motion curve x(u):
  - Position at time t is x(u(s(t)))
- Question remains: how to calculate u(s)?

Numerical Inversion

- Analytic approach is hopeless
  - Even analytically solving the integral s(u) is hard, solving for u in terms of s is crazy
- Numerical approach works fine
- Use approximate evaluation of s(u) to get a table of values
  - Cut up curve into small line segments, add up their lengths
- Then interpolate a smooth curve through the values (Catmull-Rom)
  - Use table of s values as knots, associated u values as control point values

Aside: Solving Equations (1D)

- [NOTE: irrelevant for arc-length]
- Say we're solving s(u)=L
- Traditionally rewrite as s(u)-L=0 to put it in the form f(u)=0
- Bisection Search:
  - Begin by finding $u_{low}<u_{high}$ so that $f(u_{low})$ and $f(u_{high})$ have different signs
  - Get $u_{mid}=(u_{low}+u_{high})/2$ and look at $f(u_{mid})$
  - Pick new low/high as either low/mid or mid/high (whichever interval has the zero)
  - Repeat until f(u) small enough, or $u_{high}-u_{low}$ small enough

Secant Search

- Bisection is guaranteed to converge linearly - but that can be slow
- Secant works for smooth f by being smarter about u_{mid} guess
  - Assume f is close to linear between previous guesses u_{low} and u_{high}
  - Solve for where that line crosses zero:
    $u_{mid}=(f(u_{hi})u_{lo}-f(u_{lo})\ u_{hi})/(f(u_{hi})-f(u_{lo}))$
- Usually the fastest method for smooth f

Kinematics

Kinematics

- The study of how things move
- Usually boils down to describing the motion of articulated rigid figures
  - Things made up of rigid "links" attached to each other at movable joints (articulation)
- There are many mathematical approaches to this description
- Text: sections 2.2 and 4.2
  (and more advanced: 6.1 and 6.2)

Links

- The actual geometry of each link is irrelevant for kinematics
  - But typically corresponds to a bone, or a solid piece of metal, or …
- All that matters is that they have an attached coordinate system
  - That is an "origin" -- the base point for doing measurements relative to the link -- and a set of basis vectors -- three orthogonal directions relative to the link

Joints

- A joint connects two links
  - Attachment point has to be specified in both coordinate systems
  - Often simplified by making sure attachment is the origin of one of the links
  - Also specify how basis vectors of the two are related (rotation)
- A joint can have up to six degrees of freedom (DOF) - three translations, three rotations
- For animation, usually just concerned with rotations (for robotics, maybe not): revolute
  - That is, how are the basis vectors of the second link rotated from the basis vectors of the first?

Reality Check

- This boils down to how we model real joints in the body
- E.g. from a distance:
  - Elbow has 1 DOF: the angle the forearm makes with the upper arm (rotation in plane)
  - Wrist has 3 DOF
  - …
- Up close, life can be much more complex: no simple attachment

Joint Decomposition

- Often write multiple-DOF joints as a sequence of 1-DOF joints connecting imaginary links between them
- D-H notation
  - For each joint-link sequence, specify displacement and angles in standard format

Hierarchical Skeletons

- Usually a character (or a robot) has no loops --- link/joint graph is a tree
- Pick an arbitrary link to be the root
- Settle on its coordinate system with respect to the world
  - 6 DOF: position + orientation (more on this later)
- For attached links, get translation and rotation from root's coordinate system to neighbours' coordinate systems
- Go through the tree to the tips ("end-effectors") concatenating transformations
- This is "Forward Kinematics" (FK)

Forward Kinematics

- A simple layered approach
  - Get the root link moving first (e.g. the pelvis)
  - Fix the pose outward, link by link (the back and the legs next, then the head, the arms, the hands, the fingers, ...)
- Great for certain types of motion
  - General acting, moving in free space, ...
- Problems when interacting with other objects
  - How do you make sure fingers are exactly in contact with a doorknob throughout a scene, as the arm moves, as the body moves, ...
  - Even simpler: how do you make sure the feet stay planted on the ground? (avoiding "foot-skate")

Inverse Kinematics (IK)

- Keep the same tree structure,
  fix the root,
  get the body into roughly the right pose
- Specify a target position+orientation for an end-effector
- Solve system of equations for the joint angles that achieve the target
- This is really just a way for the computer to help you set the joint angles to what you want

IK issues 1

- Too many joints: there are many solutions, how do you pick the best one?
  - Often add constraints---e.g. keep the arm in the specified plane so it doesn't rotate wildly
  - Minimize some quantity---e.g. the sum of squares of differences from original rough pose (or previous frame)
  - Open research---go to Zoran Popovic's talk for "Style-based Inverse Kinematics", SIGGRAPH 2004

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
  - Penalize bad joint angles, huge differences from initial pose