Skinning

CS418 Computer Graphics

John C. Hart

Simple Inverse Kinematics

- Given target point (x,y) in position space, what are the parameters (θ,ϕ) in configuration space that place the hand on the target point?
- Use Law of Cosines to find θ

$$d^2 = a^2 + b^2 - 2ab \cos \theta$$

$$\cos \theta = (a^2 + b^2 - d^2)/2ab$$

$$\cos \theta = (a^2 + b^2 - x^2 - y^2)/2ab$$

- And to find α

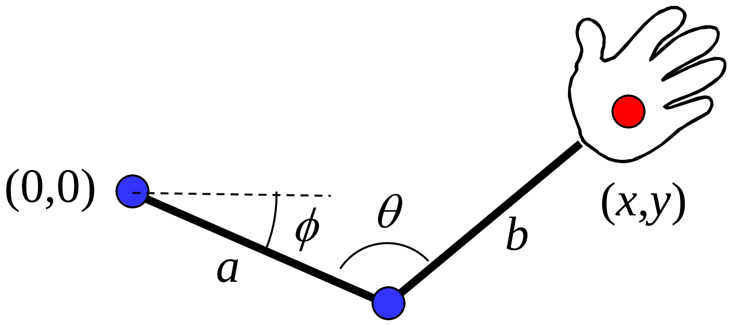
$$\cos \alpha = (a^2 + d^2 - b^2)/2ad$$

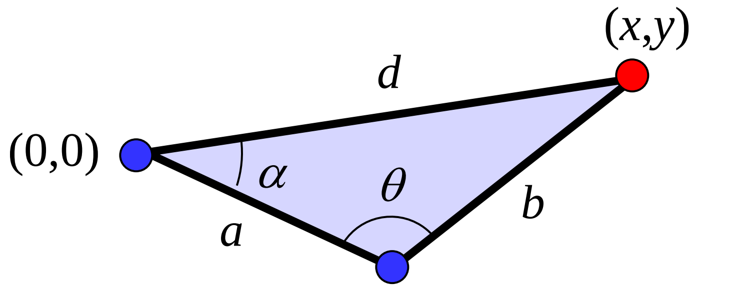
$$\cos \alpha = (a^2 + x^2 + y^2 - b^2)/2ad$$

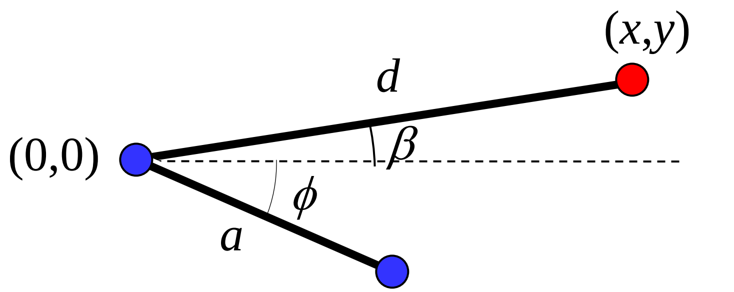
- Use arctangent to find β then ϕ

$$\beta = \text{atan2}(y,x)$$

$$\phi = \alpha - \beta$$

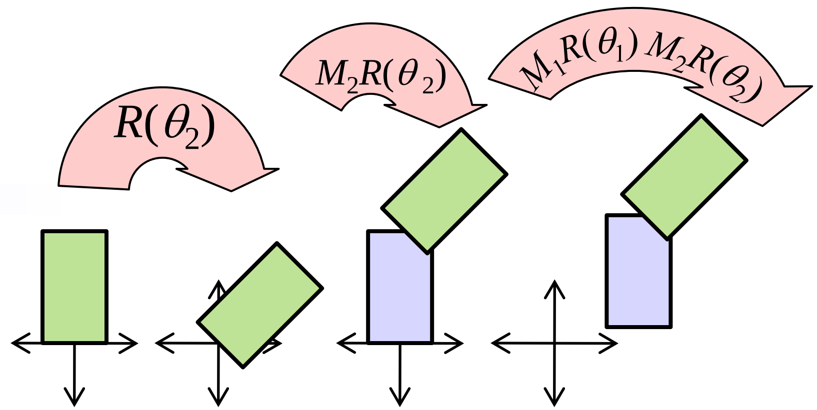


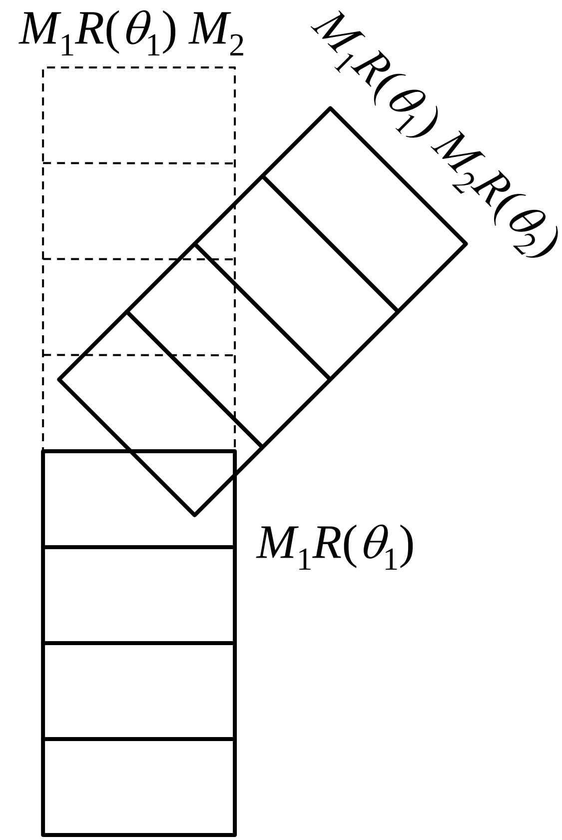




Skinning

- Elbow joints don’t look realistic because geometry detaches
- Transformation hierarchy:
  - $R(\theta_1)$ rotates upper-arm cylinder about its shoulder at the origin
  - M_1 moves upper-arm cylinder from the origin to its position in world coordinates
  - $R(\theta_2)$ rotates forearm cylinder about its elbow at the origin
  - M_2 moves forearm elbow from the origin to the end of the upper-arm cylinder when its shoulder is based at the origin
- When $\theta_2 \neq 0$ the elbow end of the upper-arm does not align with the elbow end of the forearm





Skinning

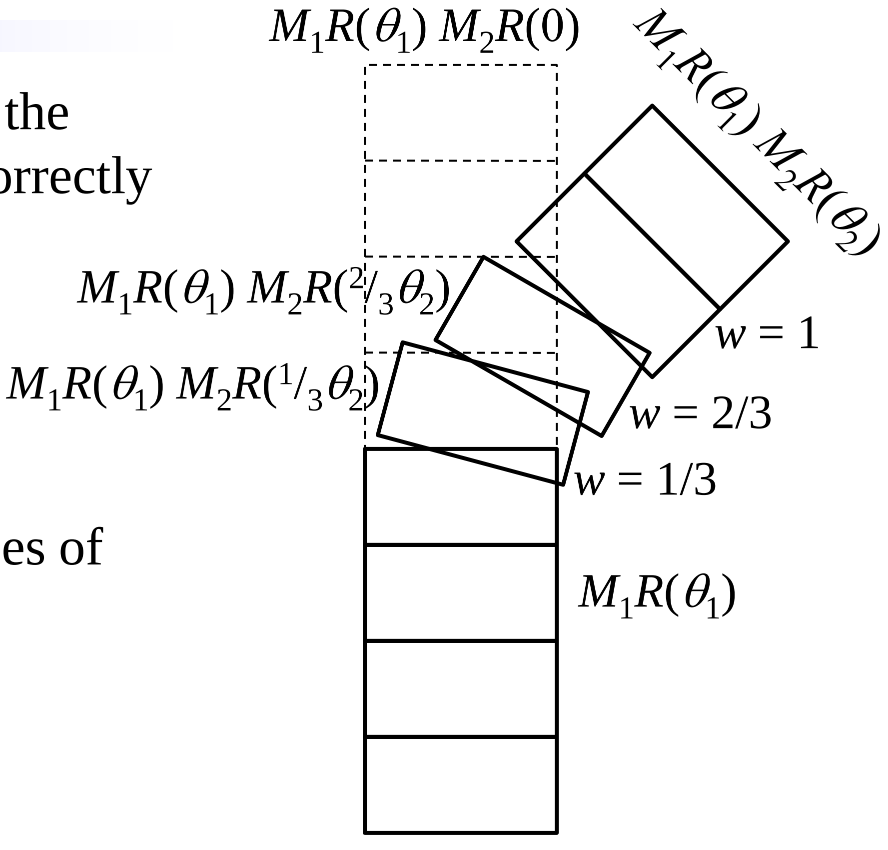
- Solution is to interpolate matrices from the undetached coordinate frame into the correctly oriented coordinate frame per-vertex
- Let

$$M_{\text{straight}} = M_1 R(\theta_1)\, M_2 R(0)$$

$$M_{\text{bent}} = M_1 R(\theta_1)\, M_2 R(\theta_2)$$

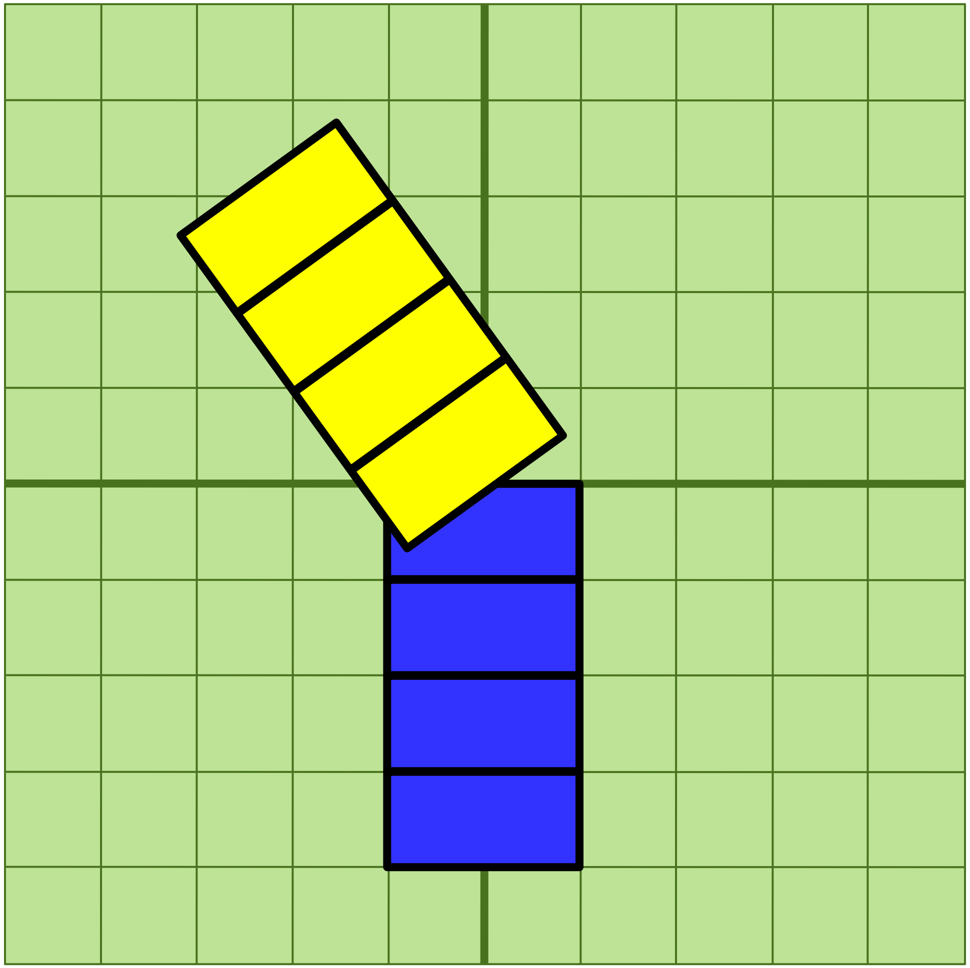
- Distribute ("paint") weights *w* on vertices of forearm cylinder
  - $w = 0$ at elbow end
  - $w = 1$ after elbow
- Transform vertices using

$$M(w) = (1 - w)\, M_{\text{straight}} + w\, M_{\text{bent}}$$



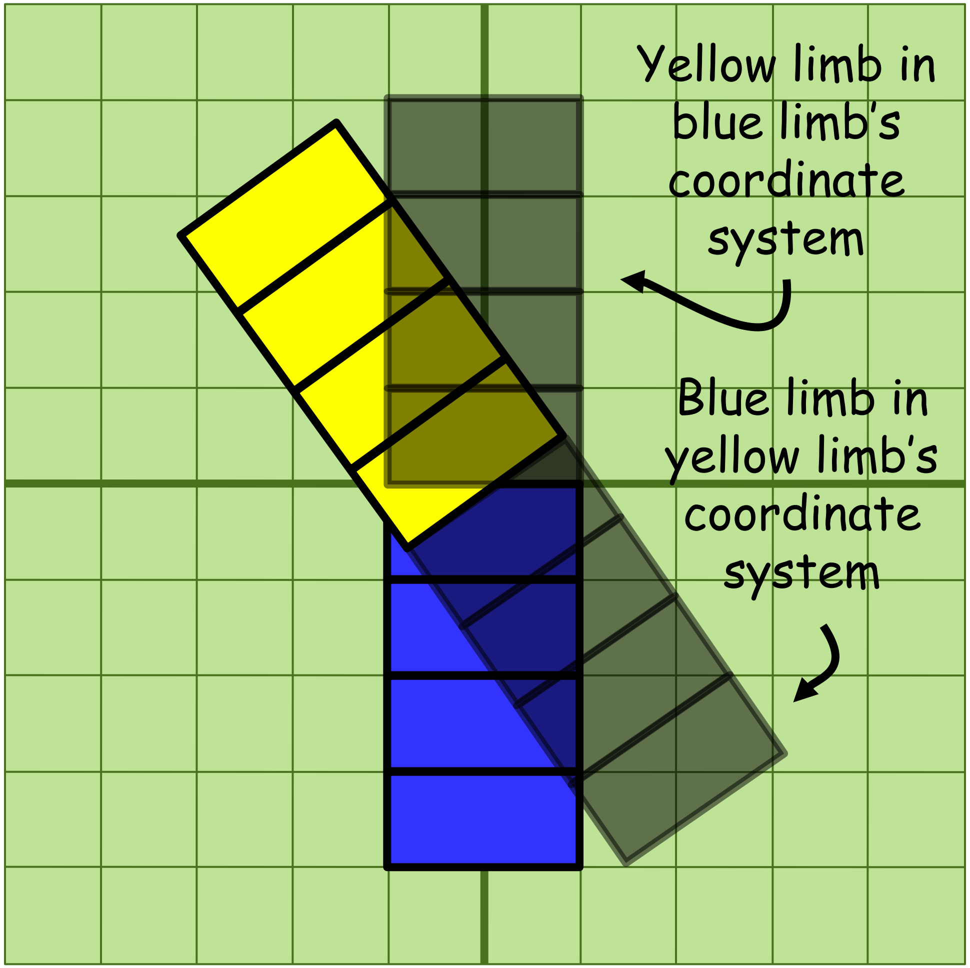
Build an Elbow

```
glPushMatrix();
glColor3f(0,0,1);
glTranslatef(0,-2,0);
drawquadstrip();
glPopMatrix();

glPushMatrix();
glColor3f(1,1,0);
glRotatef(elbow,0,0,1);
glTranslate(0,0,2);
drawquadstrip();
glPopMatrix();
```



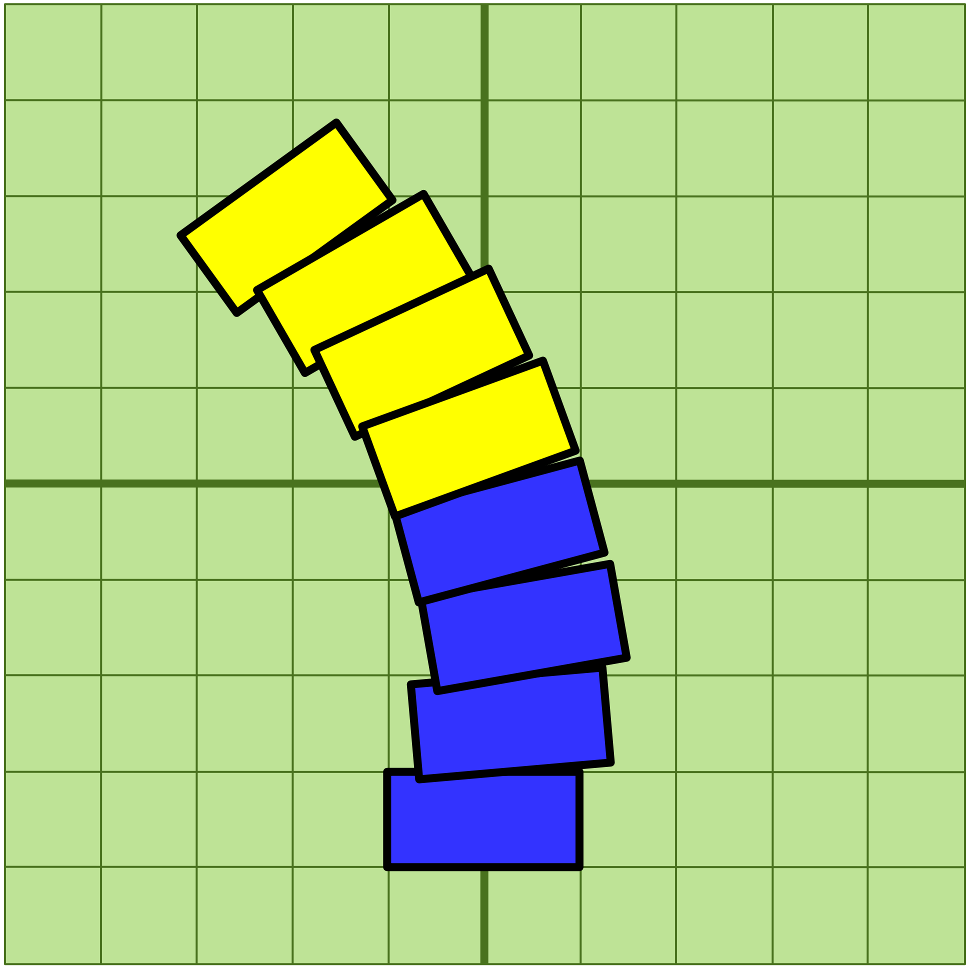
Two Coordinate Systems

```
glPushMatrix();
glColor3f(0,0,1);
glTranslatef(0,-2,0);
drawquadstrip();
glColor3f(1,1,0,.5)
glTranslatef(0,4,0);
drawquadstrip();
glPopMatrix();

glPushMatrix();
glRotatef(elbow,0,0,1);
glColor3f(1,1,0);
glTranslatef(0,0,2);
drawquadstrip();
glColor3f(0,0,1,.5);
glTranslatef(0,0,-4);
drawquadstrip();
glPopMatrix();
```



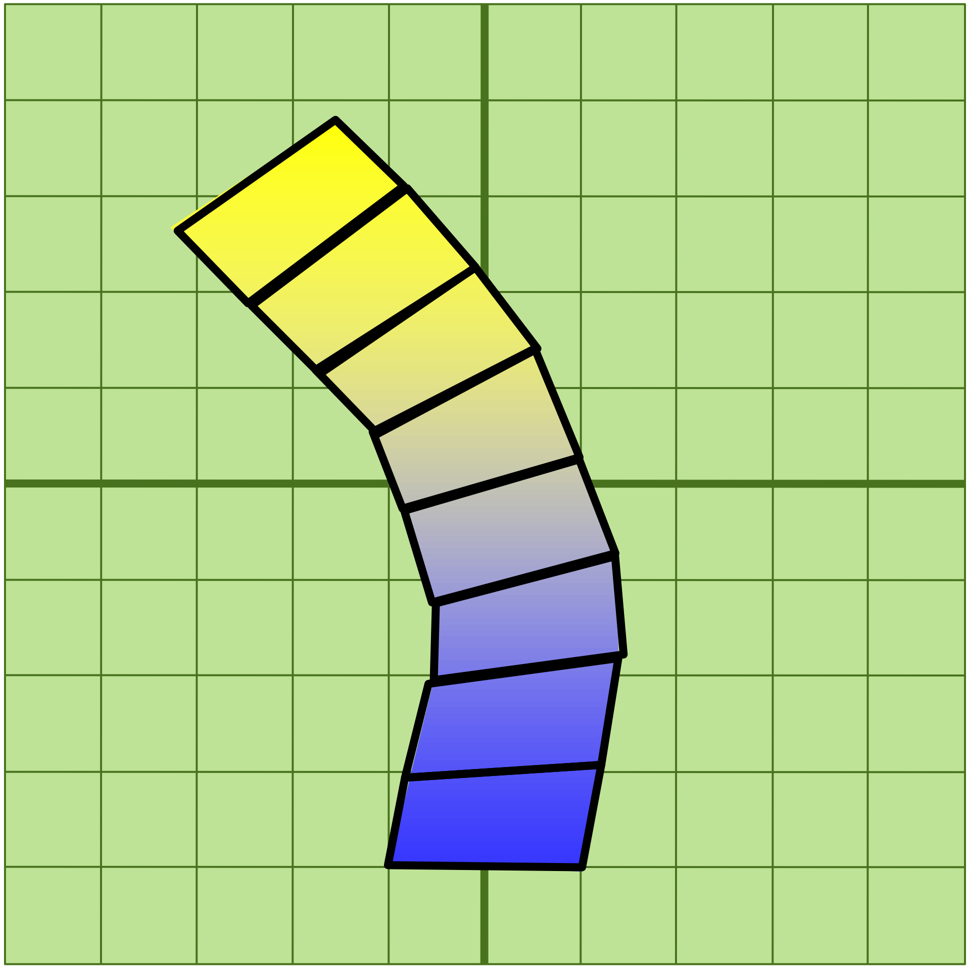
Interpolate the Transformations

```
for (i = 0; i < 8; i++) {
    weight = i/7.0;
    glPushMatrix();
    glRotatef(weight*elbow,0,0,1);
    glTranslate3f(0,0,-3.5+i);
    drawquad();
    glPopMatrix();
}
```



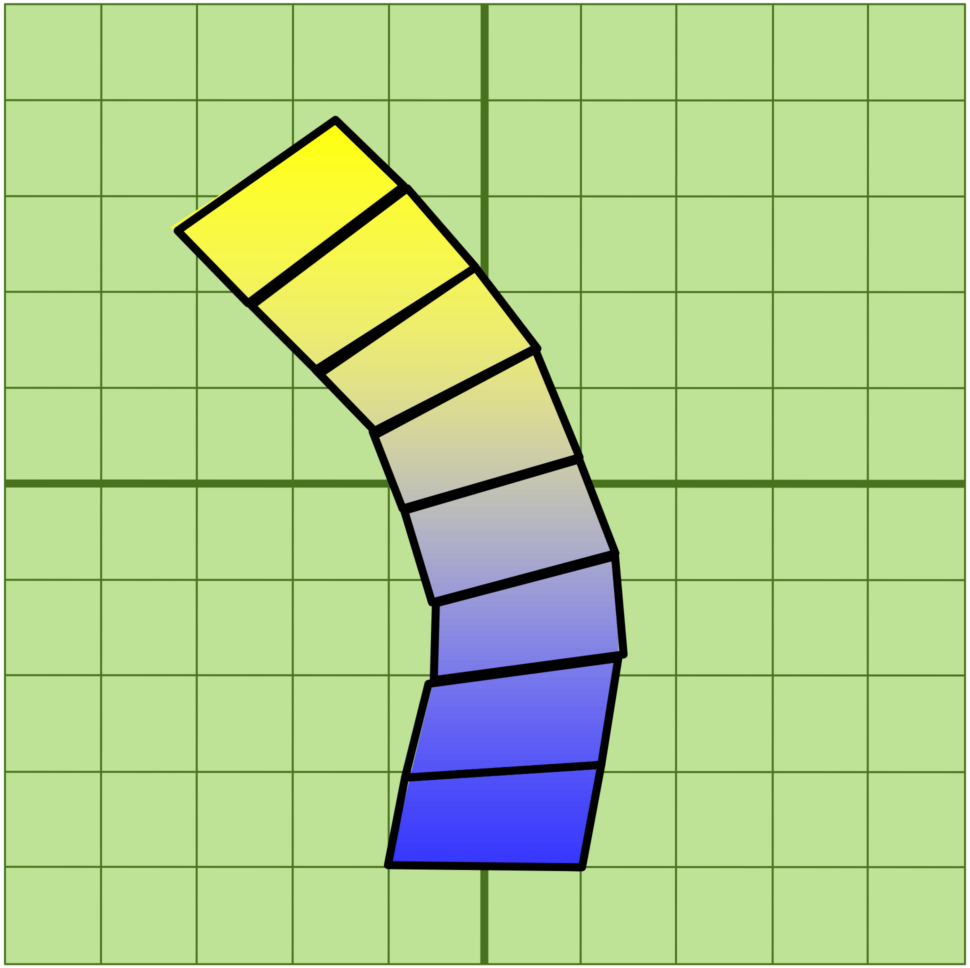
Interpolate the Vertices

```
glBegin(GL_QUAD_STRIP);
for (i = 0; i <= 8; i++) {
    weight = i/8.0;
    glColor3f(weight,weight,1-weight);
    glPushMatrix();
    glRotatef(weight*elbow,0,0,1);
    glVertex2f(-1,-4.+i);
    glVertex2f(1,-4.+i);
    glPopMatrix();
}
glEnd(/*GL_QUAD_STRIP*/);
```



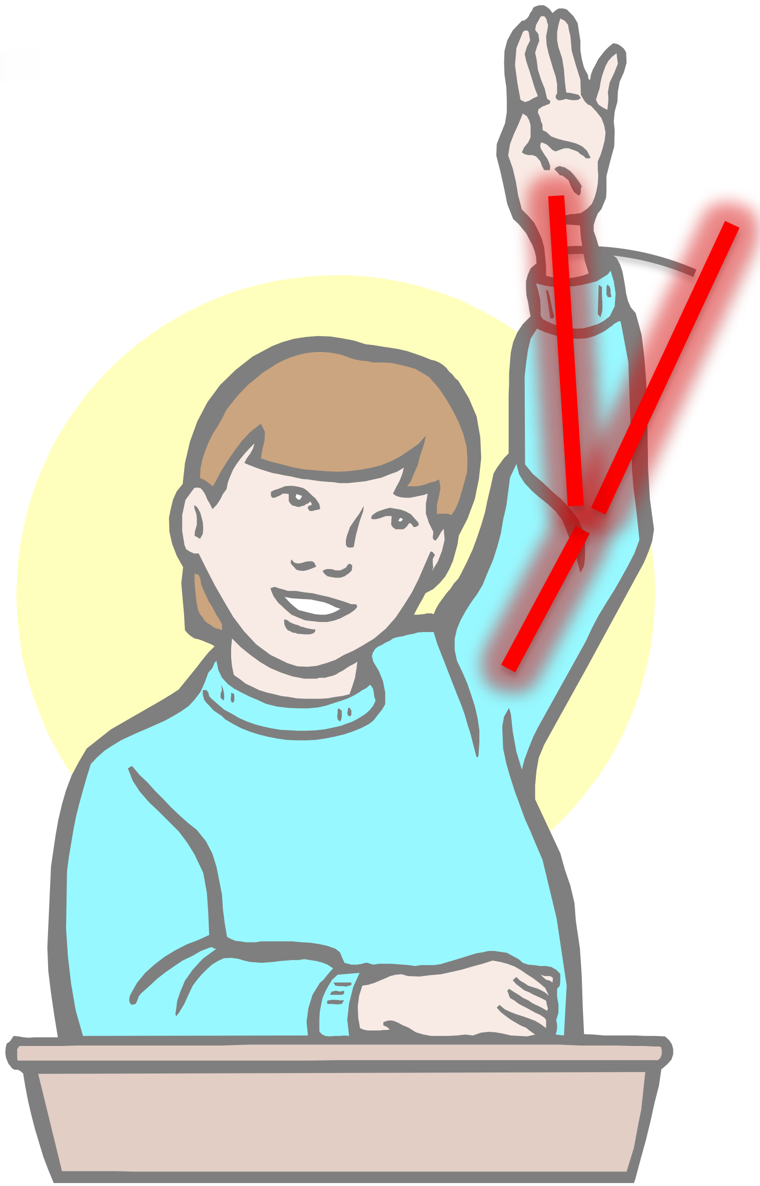
Interpolate the Matrices

```
glLoadIdentity();
glGetMatrixf(A);
glRotatef(elbow,0,0,1);
glGetMatrixf(B);

glBegin(GL_QUAD_STRIP);
for (i = 0; i <= 8; i++) {
    weight = i/8.0;
    glColor3f(weight,weight,1-weight);
    C = (1-weight)*A + weight*B;
    glLoadIdentity();
    glMultMatrix(C);
    glVertex2f(-1,-4.+i);
    glVertex2f(1,-4.+i);
}
glEnd(/*GL_QUAD_STRIP*/);
```



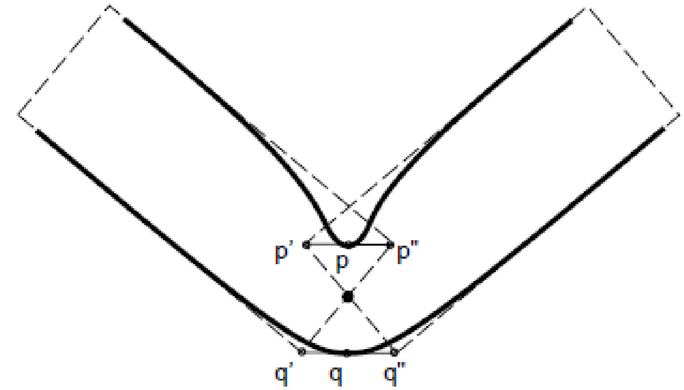
Matrix Palette Skinning

- Each vertex has one or more weight attributes associated with it
- Each weight determines the effect of each transformation matrix
- “Bones” – effect of each transformation is described by motion on bone from canonical position
- Weights can be painted on a meshed model to control effect of underlying bone transformations (e.g. chests, faces)



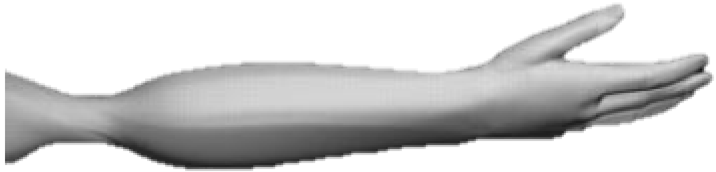
Interpolating Matrices

- Skinning interpolates matrices by interpolating their elements
- Identical to interpolating vertex positions after transformation
- We've already seen problems with interpolating rotation matrices
- Works well enough for rotations with small angles
- Rotations with large angles needs additional processing (e.g. polar decomposition)
- Quaternions provide a better way to interpolate rotations…



$$(aA + bB)p = a(Ap) + b(Bp)$$

a,b = weights
A,B = matrices
p = vertex position



From: J. P. Lewis, Matt Cordner, and Nickson Fong. "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation." Proc. SIGGRAPH 2000