skinning 23.1.2, but superseded

- in our robot, we will draw each limb as its own cuboid
- in games, we might start with the character as a complicated triangle mesh “skin”
  - we want to animate the skin by moving some underlying “bones”
    - maybe do this smoothly at joints
- demo 9/d3d/pallete

rigging

- start with mesh in a natural “rest pose”.
  - we will cover meshes later.
- each each vertex is described using object coordinates, i.e., $\tilde{p} = \tilde{o}^i c$
- artist designs a geometric skeleton and fits it to the mesh.
- each vertex is associated to one bone by the artist.
- in our robot example, let us add an “r” subscript to to mean the initial rest pose matrices.
- define the cumulative matrix for from the object frame to the bone frame, $N_r := S_rL_rB$
- this matrix expresses the frame relationship: $\tilde{b}_i = \tilde{o}^i N_r$.
- consider some vertex, with input object-coordinates $c$, that has been associated with the lower-arm bone.
  - We can write this point as $\tilde{p} = \tilde{o}^i c = \tilde{b}_i^r N_r^{-1}c$.

animate

- manipulate the skeleton, by updating some of its matrices to new settings, say $S_n$, and $L_n$ where the subscript “n” means “new”.
- define the “new” cumulative matrix for this bone, $N_n := S_nL_nB$
  - which expresses the relation: $\tilde{b}_n^i = \tilde{o}^i N_n$.
- frame has updated as $\tilde{b}_n^i \Rightarrow \tilde{b}_n^r$.
- to move the point $\tilde{p}$ in a rigid fashion along with this frame, then we need to update it using
  \[
  \tilde{b}_n^r N_r^{-1}c \Rightarrow \tilde{b}_n^r N_r^{-1}c = \tilde{o}^i N_n N_r^{-1}c
  \]
  - giving us our MVM

soft skinning

- allow the animator to associate a vertex to more than one bone. We then apply the above computation to each vertex, for each of its bones, and then blend the results together.
- we allow the animator to set, for each vertex, an array of weights $w_i$, summing to one,
  - specify how much the motion of each bone should affect this vertex.
- during animation, we compute the eye coordinates for the vertex as
  \[
  \sum_i w_i E^{-1}O(N_n)_i(N_r)_i^{-1}c
  \] \hspace{1cm} (1)
  where the $(N)_i$ are the cumulative matrices for bone $i$.
- can be implemented in a vertex shader
  - need to pass an array of MVM matrices.