CS475m - Computer Graphics

Lecture 09 : Hierarchical Modelling

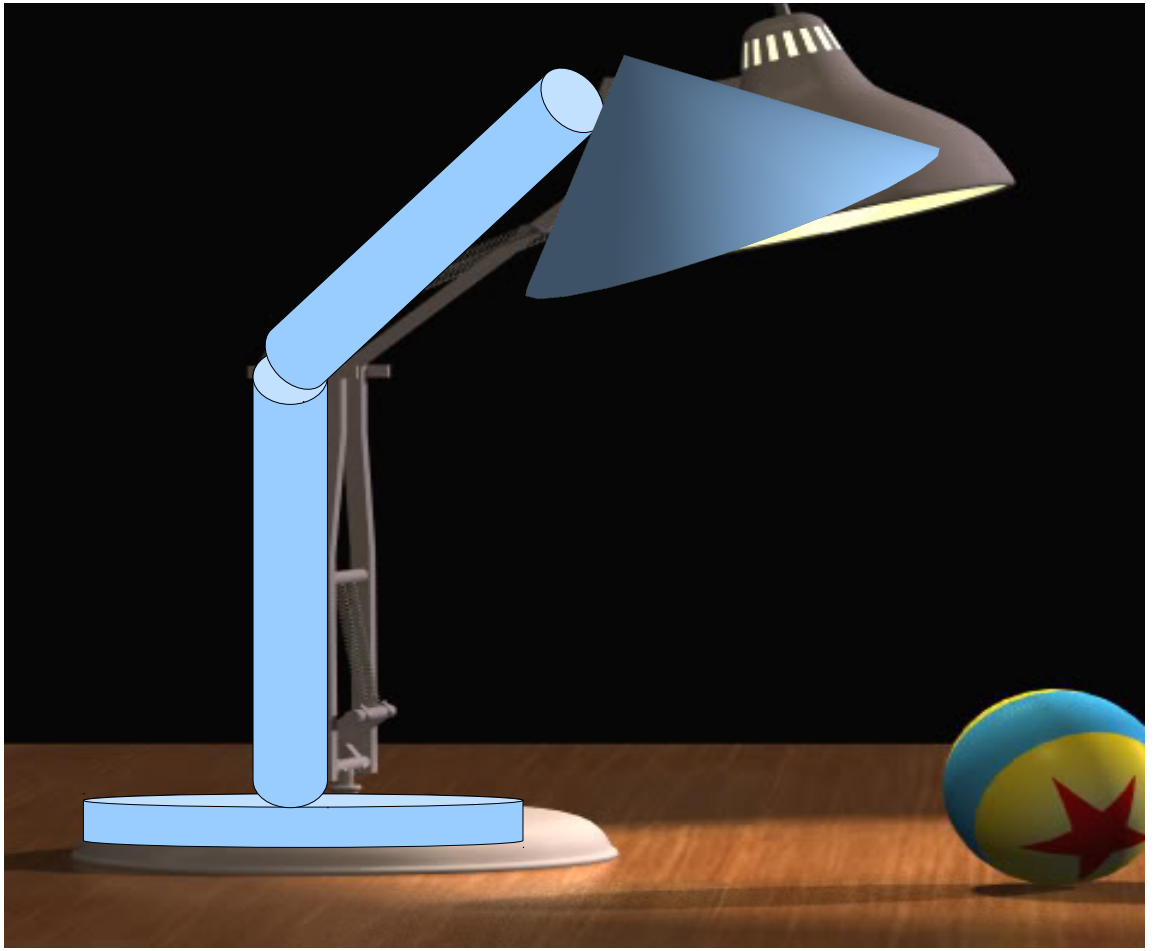
Modelling

- Modelling and Rendering

- Transformations

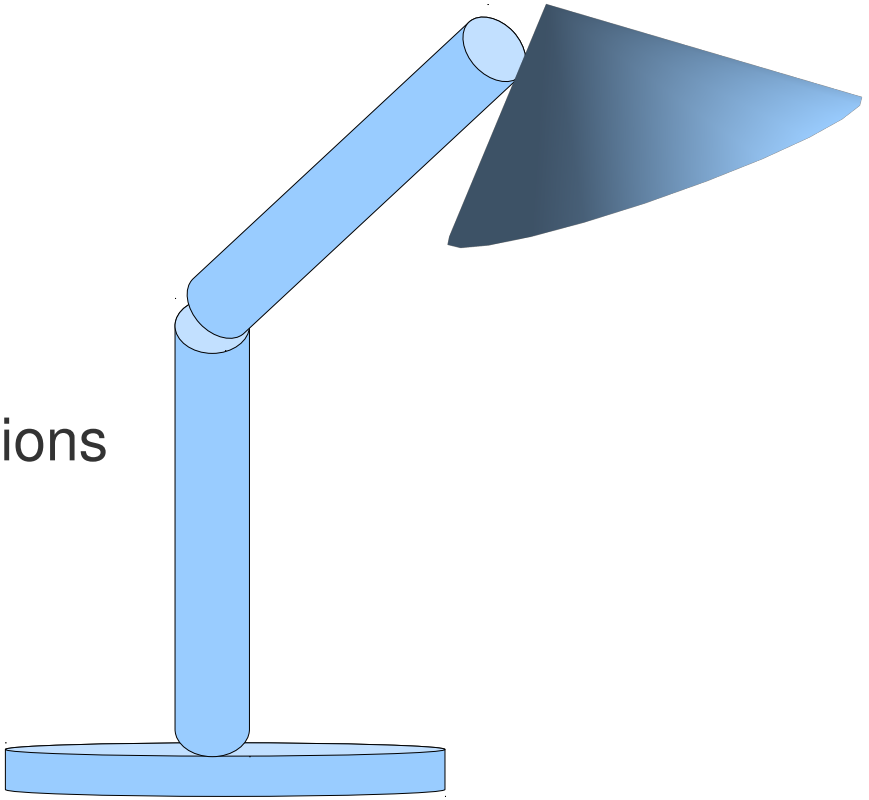
Modelling

- Modelling and Rendering
- Transformations



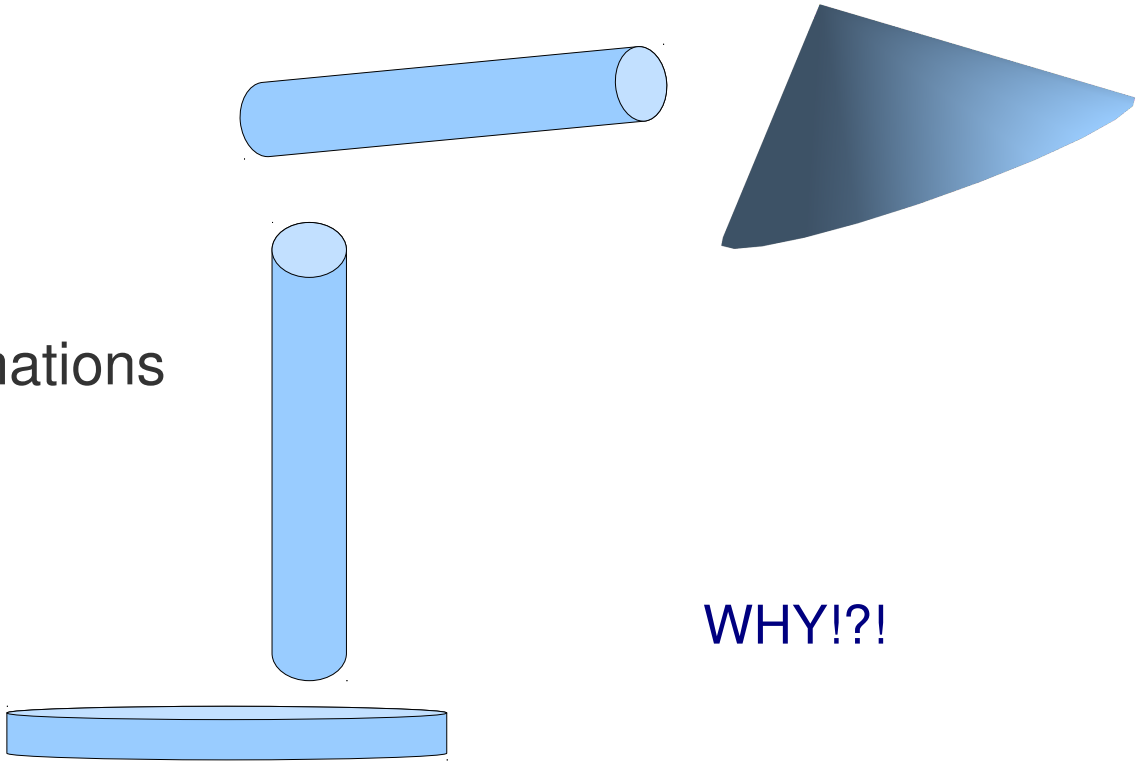
Modelling

- Modelling and Rendering
- Transformations
- Moving this model?
  - Change the transformations over time.



Modelling

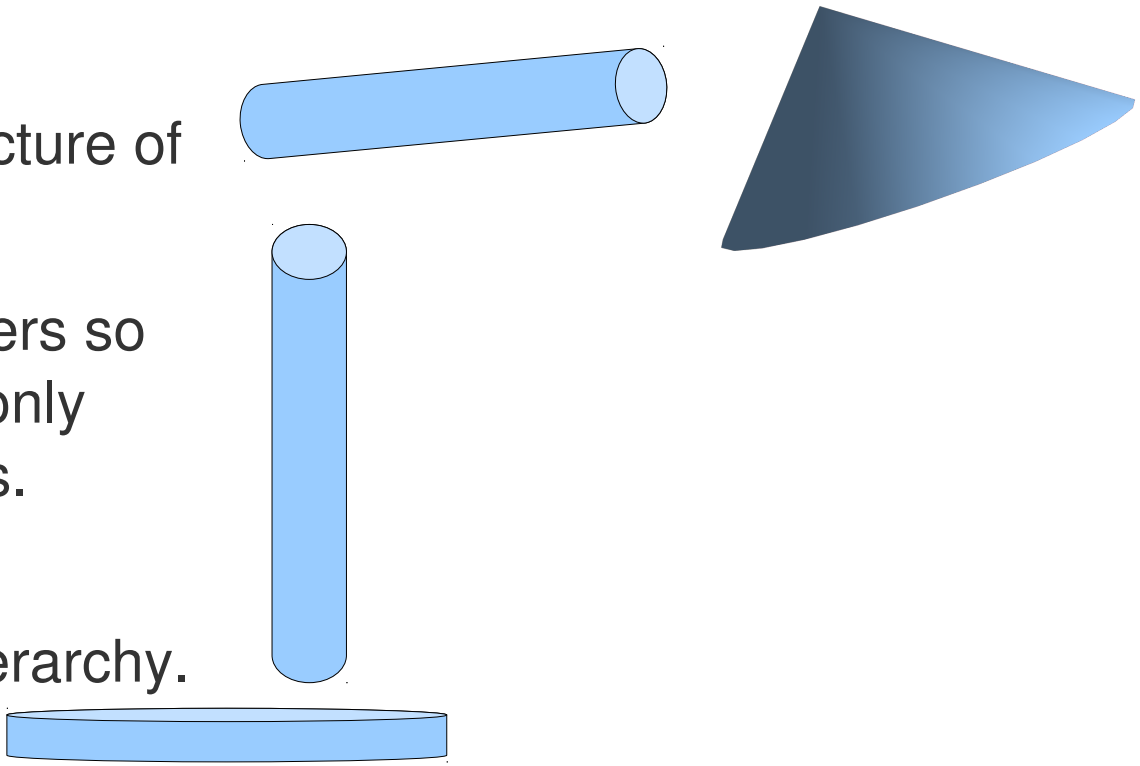
- Modelling and Rendering
- Transformations
- Moving this model?
  - Change the transformations over time.
  - Model falls apart!



WHY!?!

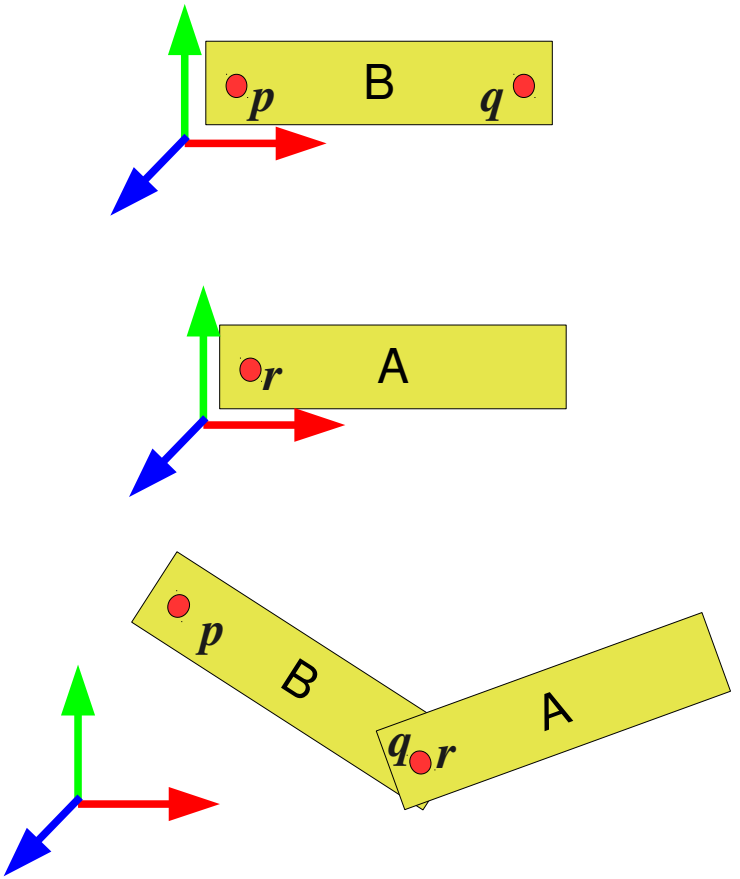
Modelling

- The object we are modelling is ***constrained*** but the model does not know that.
- We need:
  - To represent the structure of the model.
  - A handle on parameters so that we can move only through valid poses.
- So we structure our transformations into a hierarchy.



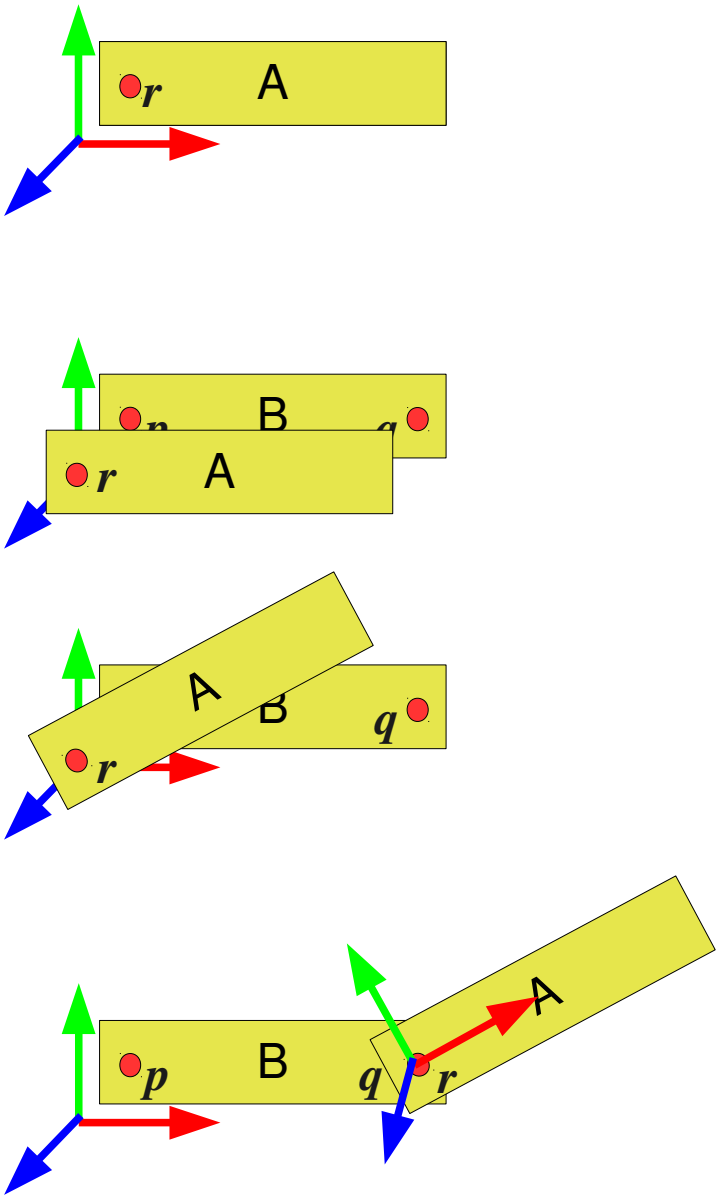
Modelling

- Modelling a two-link arm
  - Rigid Links
  - Hinge Joints
  - Upper arm link B has two joints p and q (shoulder and elbow)
  - Lower arm link A has one joint, r
  - Attach point q on B to r on A.
  - Parameters to control –
    - shoulder position T
    - shoulder angle θ (A and B together rotate about p)
    - elbow angle φ (A rotates about r, and stays attached to B at q)



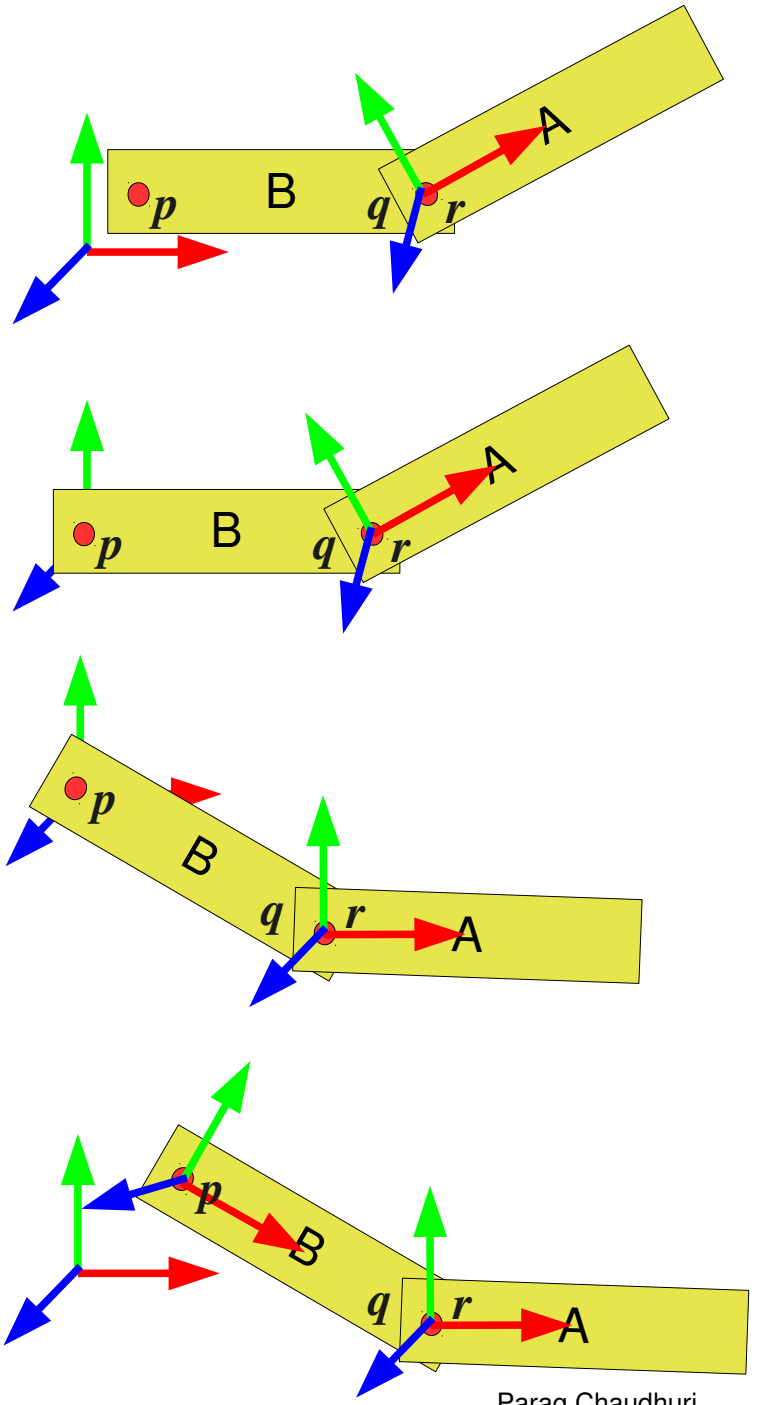
Modelling

- Modelling a two-link arm
  - Start with A and B in their original positions
  - Apply only to A
    - Translate by -r
    - Rotate by φ about the origin.
    - Translate by q, bringing r and q together.
    - We can now consider q as the origin of the lower arm link, and regard A as being in this coordinate system.



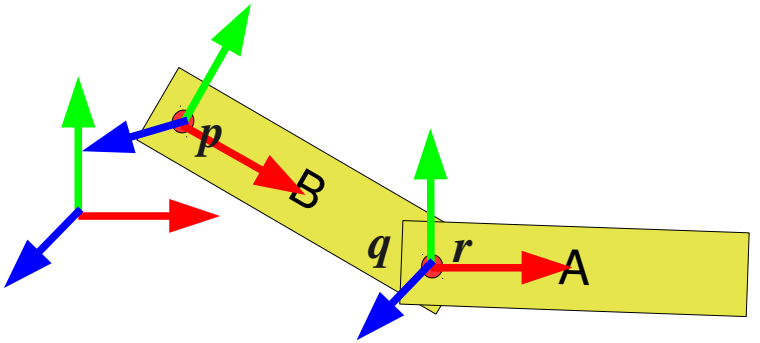
Modelling

- Modelling a two-link arm
  - Now the transformations apply to both A and B
    - Translate by -p
    - Rotate by θ about the origin.
    - Translate by T to place the two link arm at the proper position.



Modelling

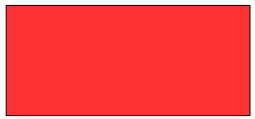
- Modelling a two-link arm
  - Complicated?
  - Remember the sequence of transformations and parameters
  - Re-apply all transformations in same sequence when parameters change
- Note:
  - θ ,φ, and T are parameters – we change these to animate the model
  - p,q and r are structural constraints. If we change them – model falls apart.



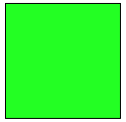
Hierarchical Modelling

- Store the modelling sequence in a hierarchy
  - Leaves have the geometry.
  - Internal nodes have transformations.
  - Transformations apply to everything under them – start at the bottom and work you way up.

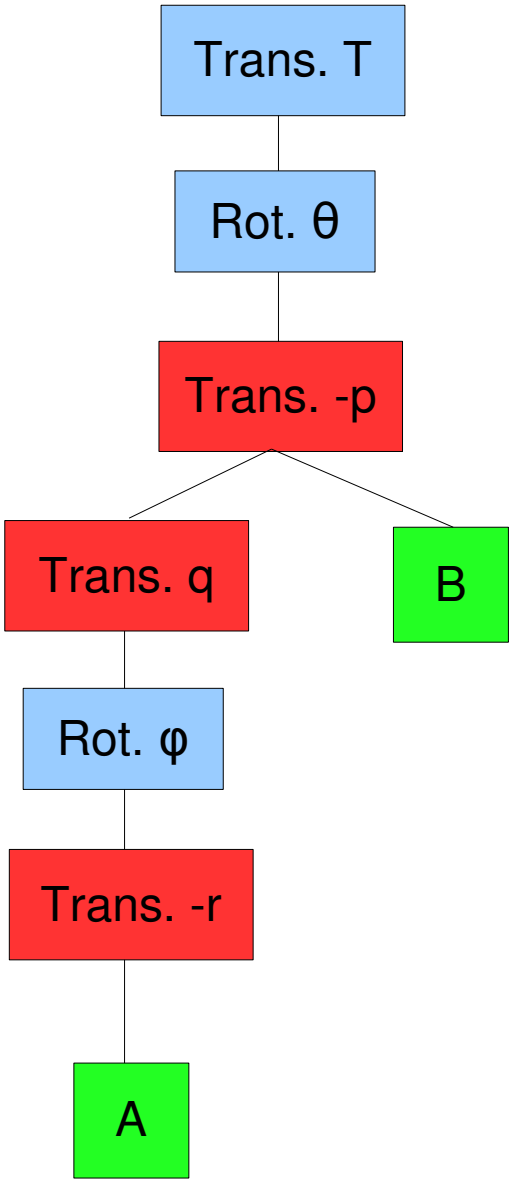
Parameters



Structural constraints



Geometric Primitive

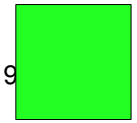


Hierarchical Modelling

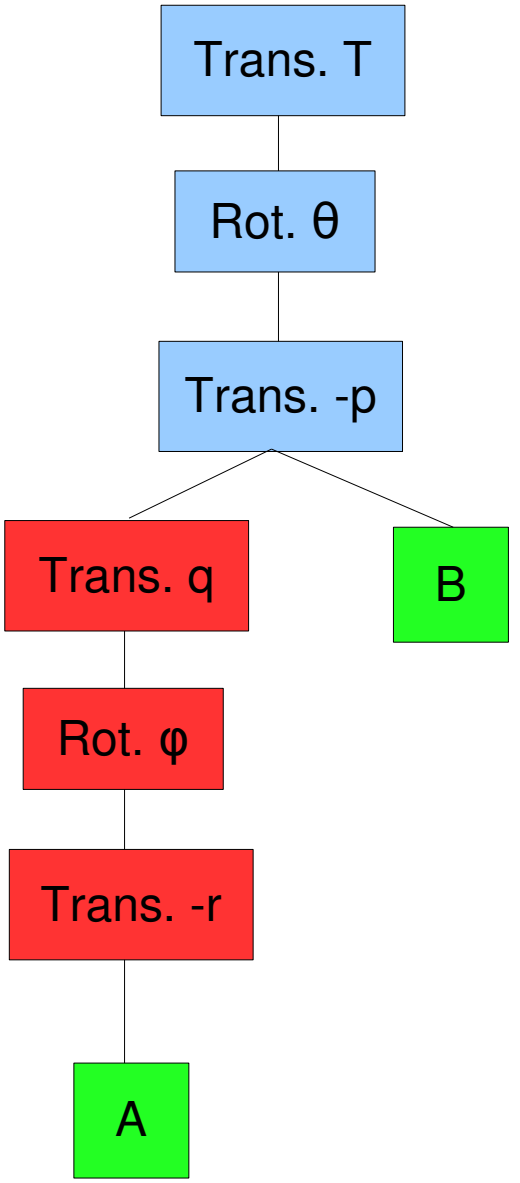
- Another view
  - The shoulder coordinate transformation moves everything below it w.r.t. the shoulder:
    - B
    - A and its transformation
  - The elbow coordinate transform moves A with respect to the shoulder coordinate transform.

Shoulder coordinate transform

Elbow coordinate transform

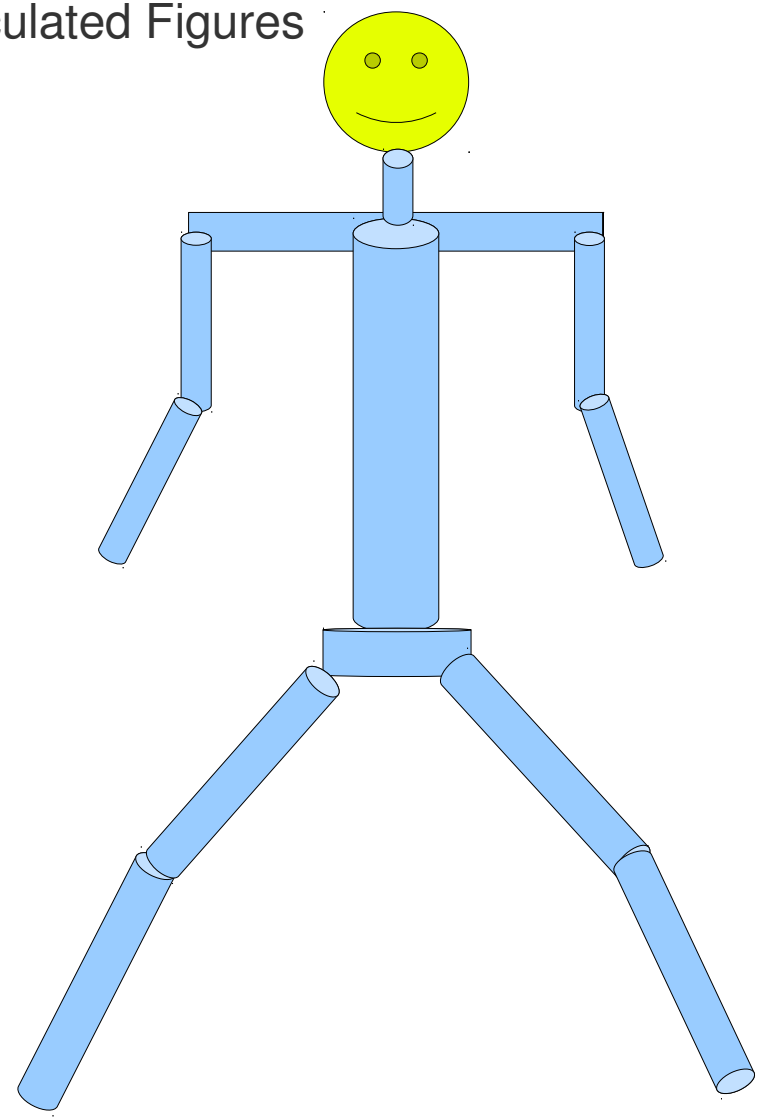


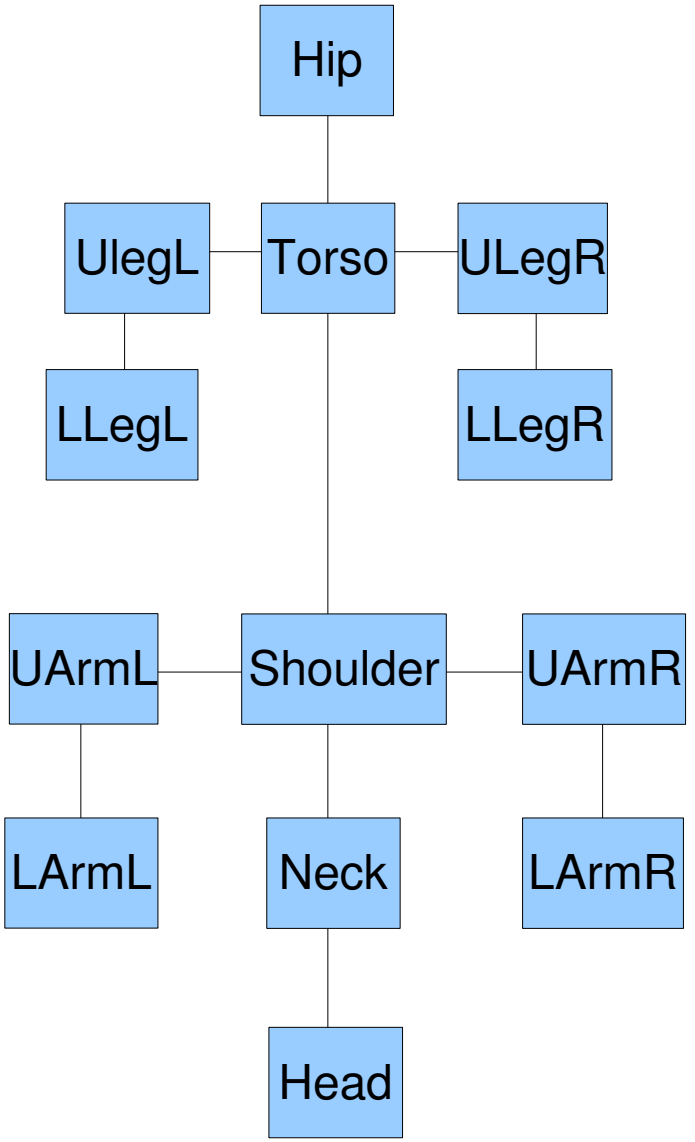
Geometric Primitive



Hierarchical Modelling

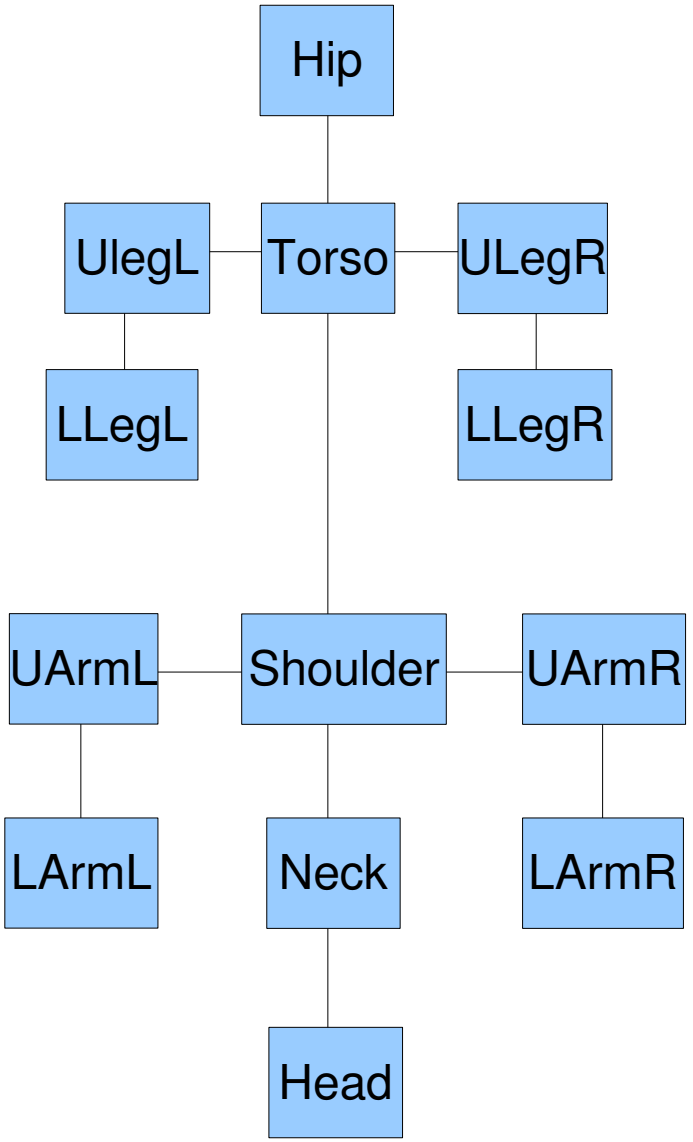
- Articulated Figures





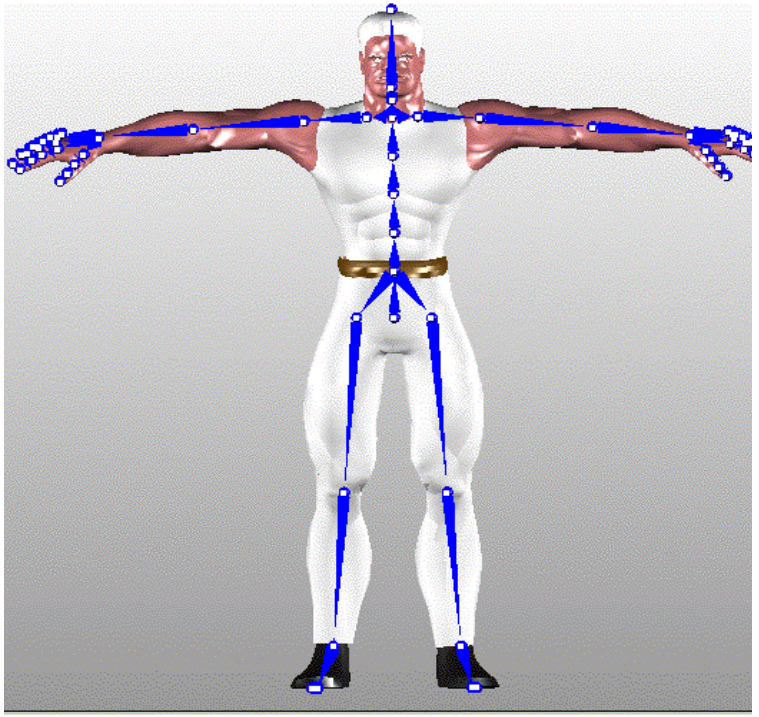
Hierarchical Modelling

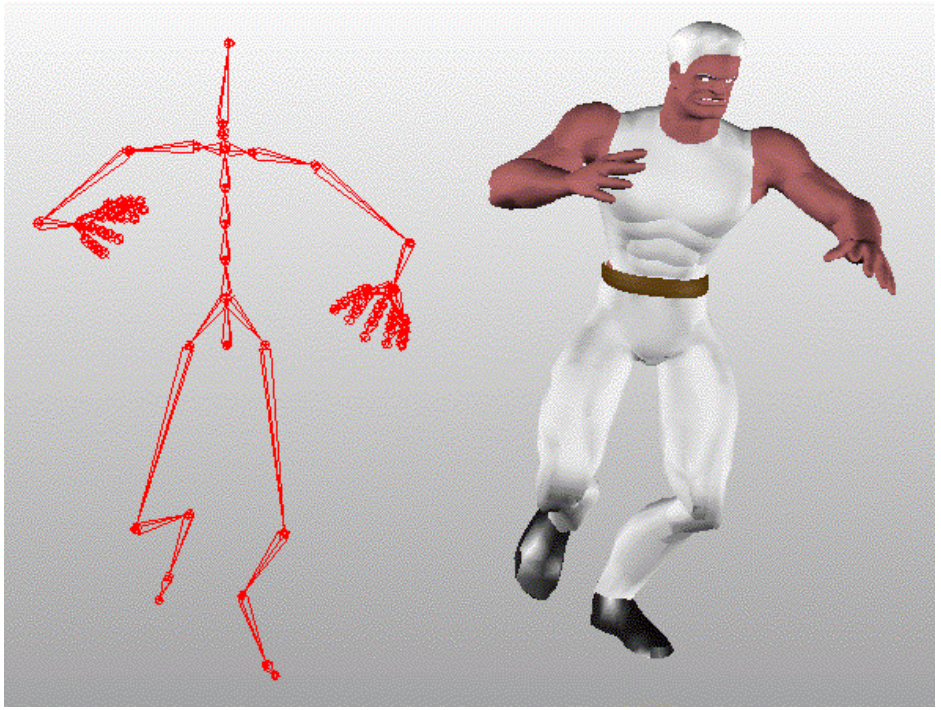
- Articulated Figures
  - Each node represents the geometry, rotation parameters and structural transformations.
  - Root can we anywhere – here it is at the hip.
  - A realistic human is much more complex
  - Difficult to control so many DoF's (later problem)
  - A **D**irected **A**cyclic **G**raph
  - Not necessarily a tree, as geometry can be transformed instances of each other



Hierarchical Modelling

- Articulated Figures
  - Character Rigging and skinning

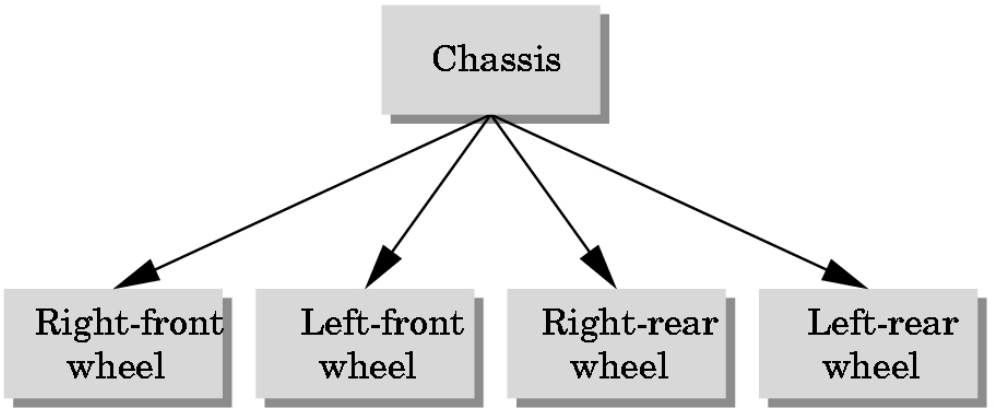


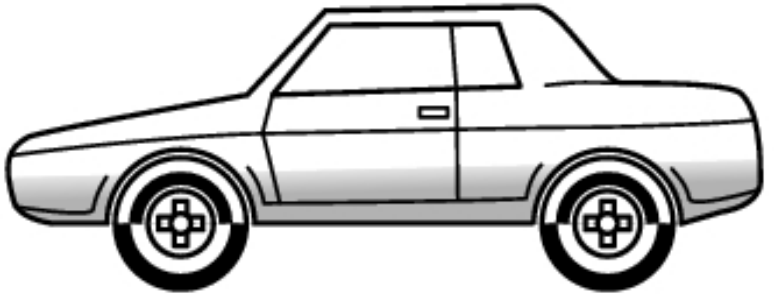


http://www.okino.com/conv/skinning.htm

Hierarchical Modelling

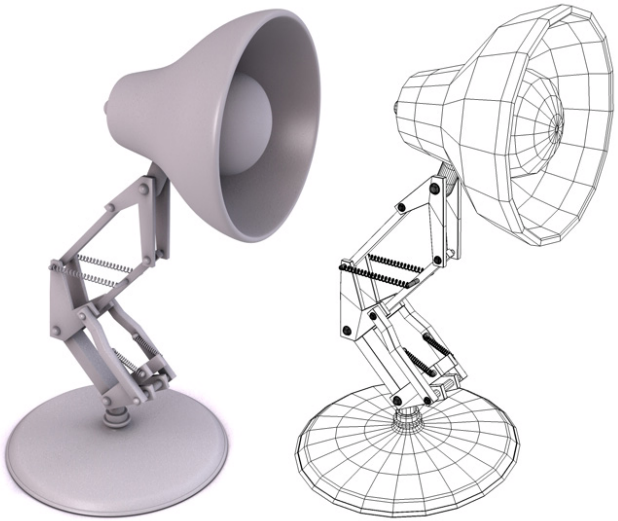
- We can model a lot of things this way

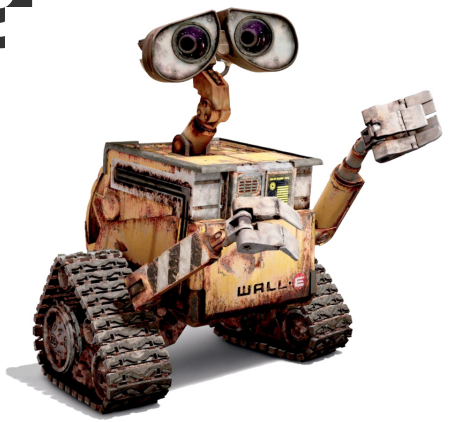




Hierarchical Modelling

- We can model a lot of things this way





Wall-E, PIXAR Animation Studios, 2008

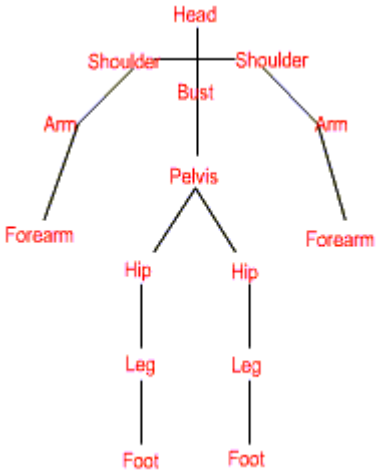
http://thechainring.com/complete-bicycles-frames-and-forks/complete-unicycles/nimbus-red-24-unicycle/

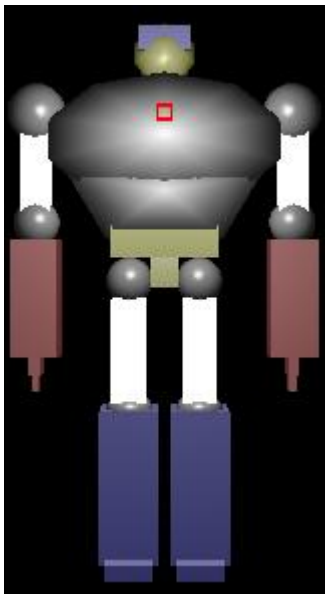
Hierarchical Modelling

- Doing this in OpenGL
  - Use the Matrix Stack
  - Current matrix is automatically product of everything already on the stack
  - This is the matrix on top of the stack
- Recursive algorithm
  - Load Identity Matrix
  - For each internal node
    - Push new matrix into stack
    - Concatenate transformations onto current matrix.
    - Recursively descend tree
    - Pop matrix off stack
  - For each leaf node
    - Draw the geometry using the current transformation

Hierarchical Modelling

- Doing this in OpenGL
- Using Display Lists - deferred mode rendering





http://www.gamedev.net/reference/articles/article1267.asp