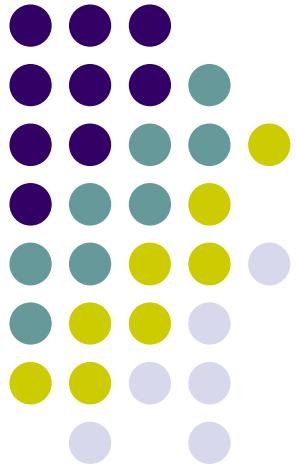
Computer Graphics (CS 4731)
Hierarchical Modeling

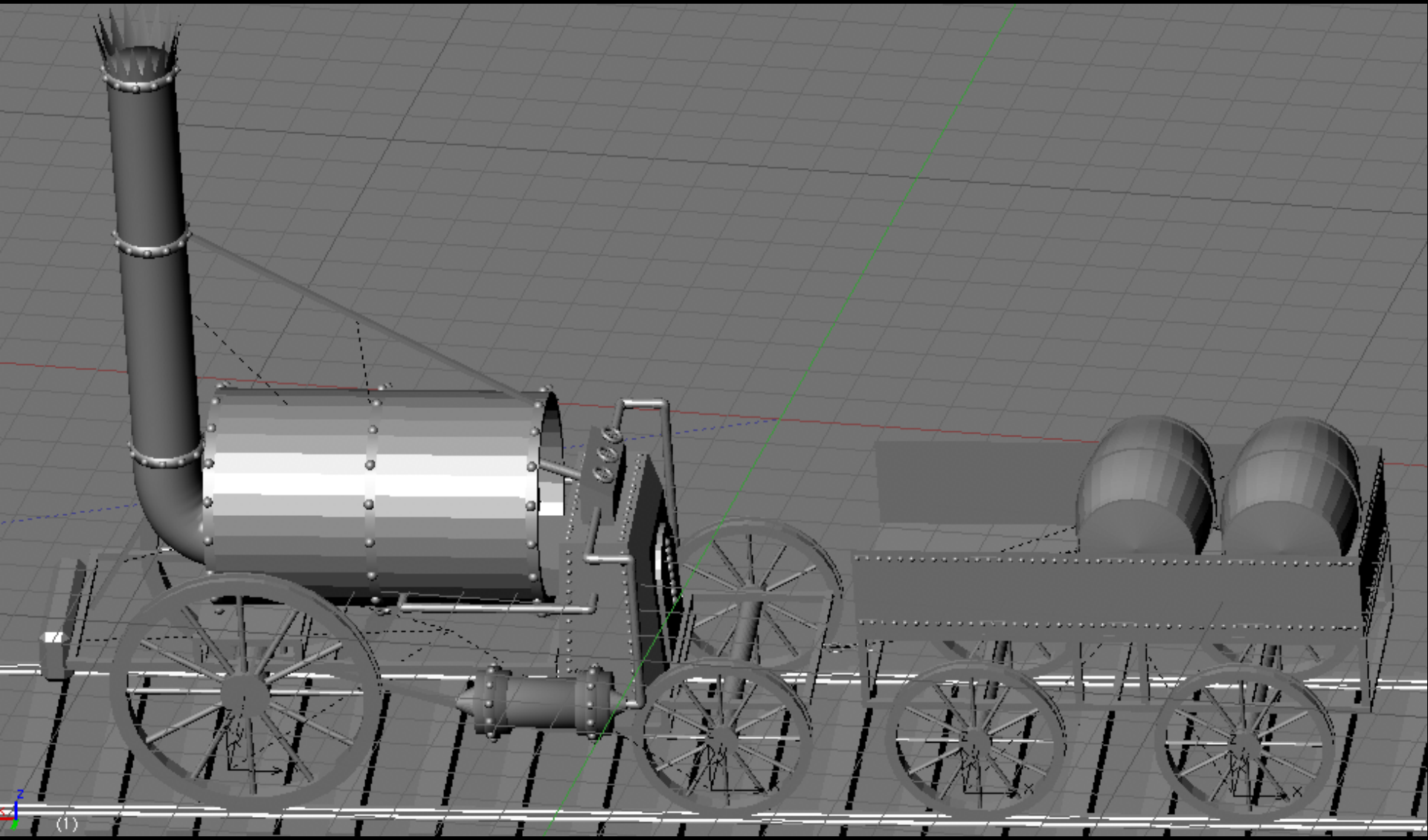
Joshua Cuneo

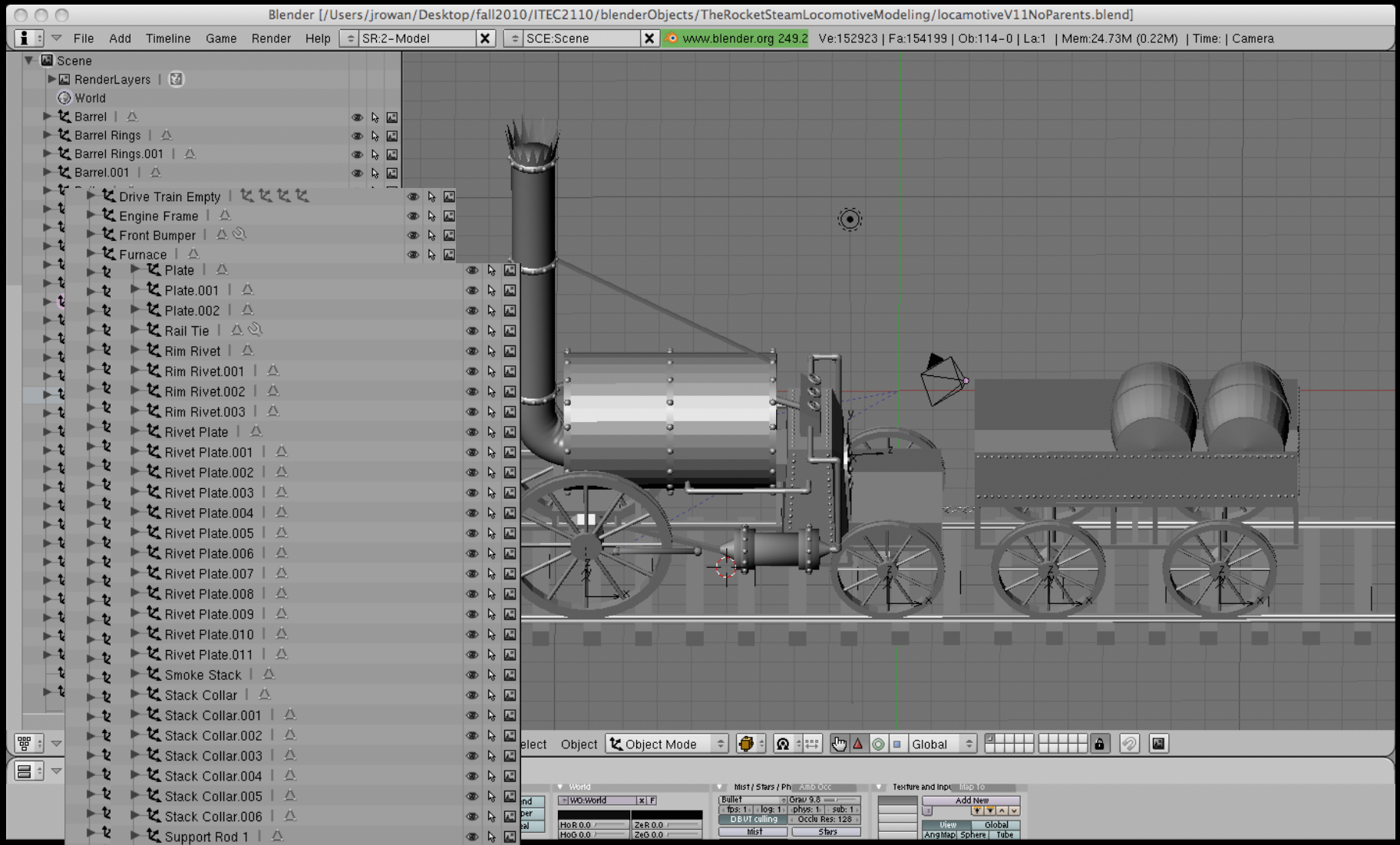
Computer Science Dept.

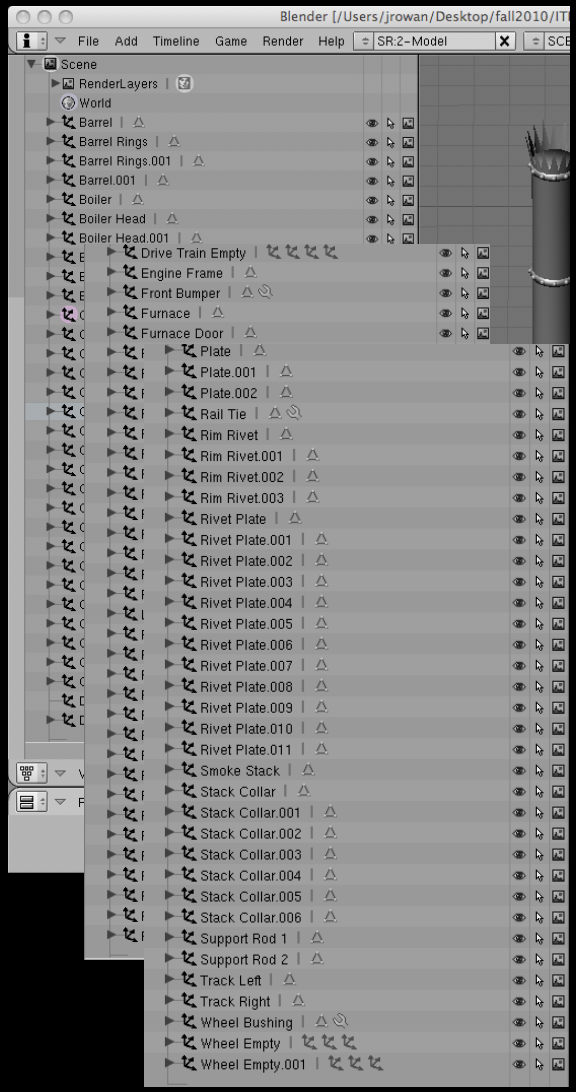
Worcester Polytechnic Institute (WPI)

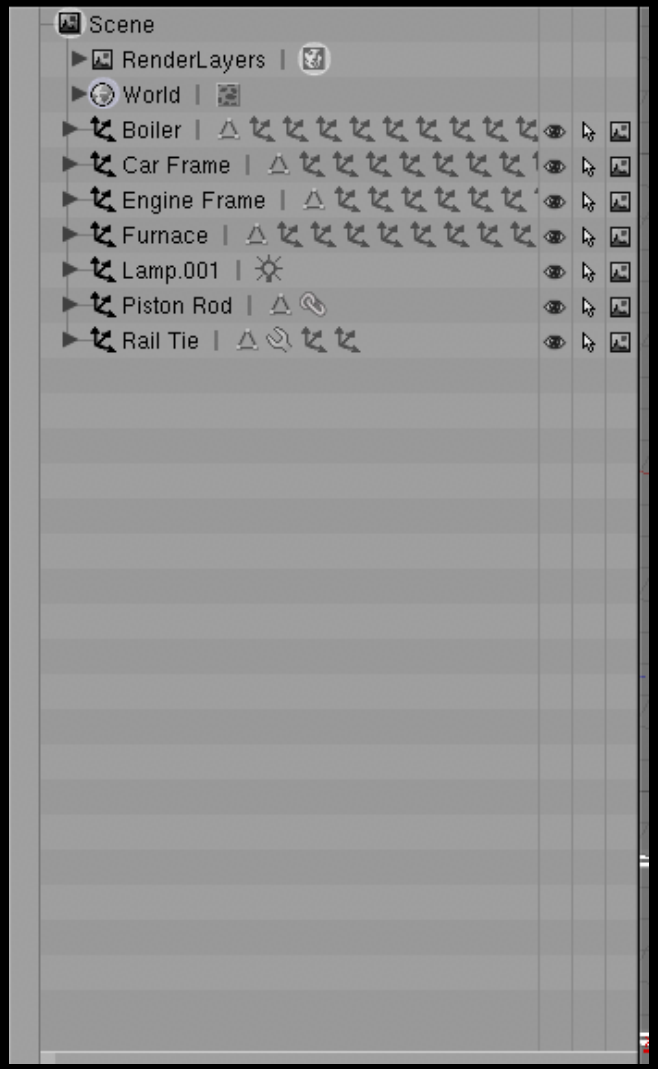


Some Slides Courtesy of Emmanuel Agu (WPI)

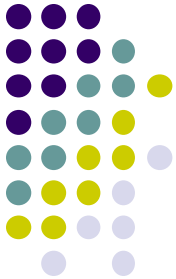

Z
(1)


Blender [/Users/jrowan/Desktop/fall2010/ITEC2110/blenderObjects/TheRocketSteamLocomotiveModeling/locamotiveV11NoParents.blend]
File Add Timeline Game Render Help SR:2-Model SCE:Scene www.blender.org 249.2 Ve:152923 | Fa:154199 | Ob:114-0 | La:1 | Mem:24.73M (0.22M) | Time: | Camera
Scene
RenderLayers
World
Barrel
Barrel Rings
Barrel Rings.001
Barrel.001
Drive Train Empty
Engine Frame
Front Bumper
Furnace
Plate
Plate.001
Plate.002
Rail Tie
Rim Rivet
Rim Rivet.001
Rim Rivet.002
Rim Rivet.003
Rivet Plate
Rivet Plate.001
Rivet Plate.002
Rivet Plate.003
Rivet Plate.004
Rivet Plate.005
Rivet Plate.006
Rivet Plate.007
Rivet Plate.008
Rivet Plate.009
Rivet Plate.010
Rivet Plate.011
Smoke Stack
Stack Collar
Stack Collar.001
Stack Collar.002
Stack Collar.003
Stack Collar.004
Stack Collar.005
Stack Collar.006
Support Rod 1
Support Rod 2
Track Left
Track Right
Wheel Bushing
Wheel Empty
Wheel Empty.001
Object Object Mode Global

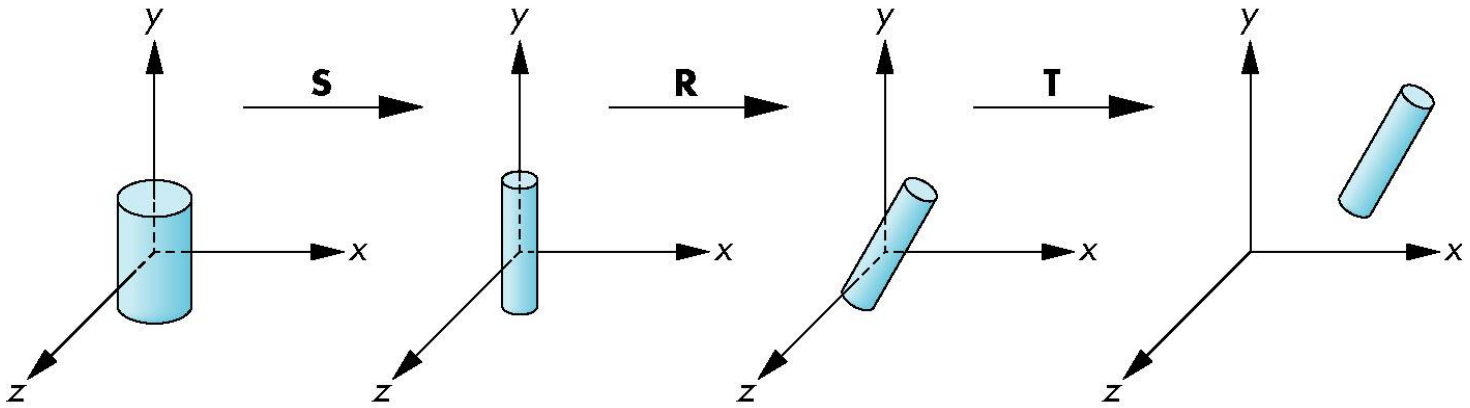

Blender [/Users/jrowan/Desktop/fall2010/IT
File Add Timeline Game Render Help SR:2-Model
Scene
RenderLayers
World
Barrel
Barrel Rings
Barrel Rings.001
Barrel.001
Boiler
Boiler Head
Boiler Head.001
Drive Train Empty
Engine Frame
Front Bumper
Furnace
Furnace Door
Plate
Plate.001
Plate.002
Rail Tie
Rim Rivet
Rim Rivet.001
Rim Rivet.002
Rim Rivet.003
Rivet Plate
Rivet Plate.001
Rivet Plate.002
Rivet Plate.003
Rivet Plate.004
Rivet Plate.005
Rivet Plate.006
Rivet Plate.007
Rivet Plate.008
Rivet Plate.009
Rivet Plate.010
Rivet Plate.011
Smoke Stack
Stack Collar
Stack Collar.001
Stack Collar.002
Stack Collar.003
Stack Collar.004
Stack Collar.005
Stack Collar.006
Support Rod 1
Support Rod 2
Track Left
Track Right
Wheel Bushing
Wheel Empty
Wheel Empty.001


Scene
RenderLayers
World
Boiler
Car Frame
Engine Frame
Furnace
Lamp.001
Piston Rod
Rail Tie

Current Transformation Approach

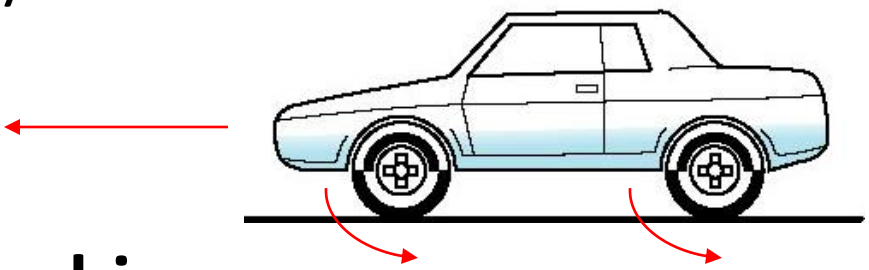


- Start with unique object
- Must scale, orient, position that object



Problem

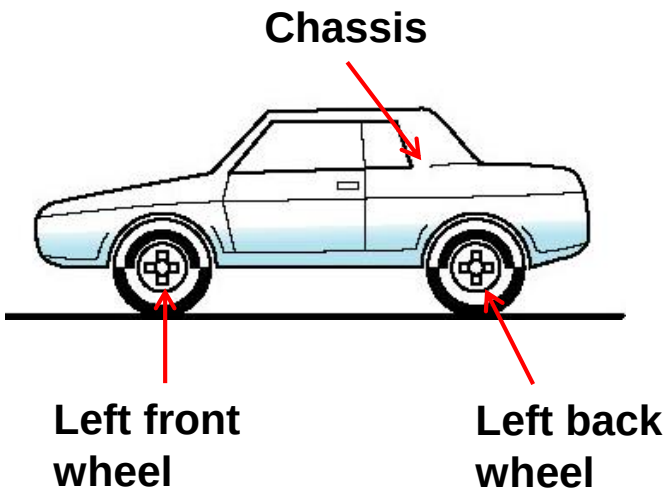
- This approach does not show relationships between parts of model
- Consider model of car
  - Chassis (body) + 4 identical wheels
  - Two symbols



- **Relationships:**
  - Wheels connected to chassis
  - Chassis motion determined by rotational speed of wheels

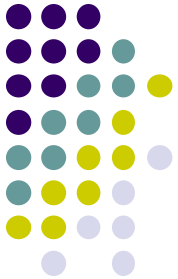
Structure Program Using Function Calls?

```
car(speed)
{
      chassis()
      wheel(right_front);
      wheel(left_front);
      wheel(right_rear);
      wheel(left_rear);
}
```

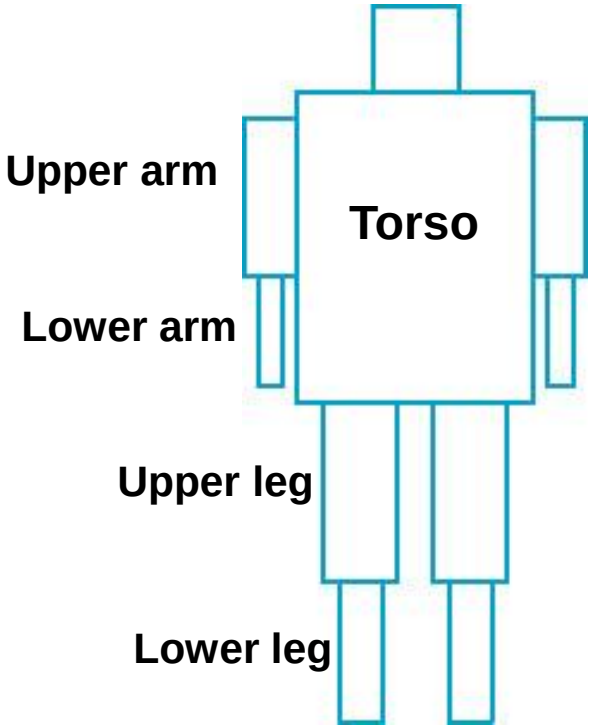


- Fails to show relationships between parts

Hierarchical Modeling

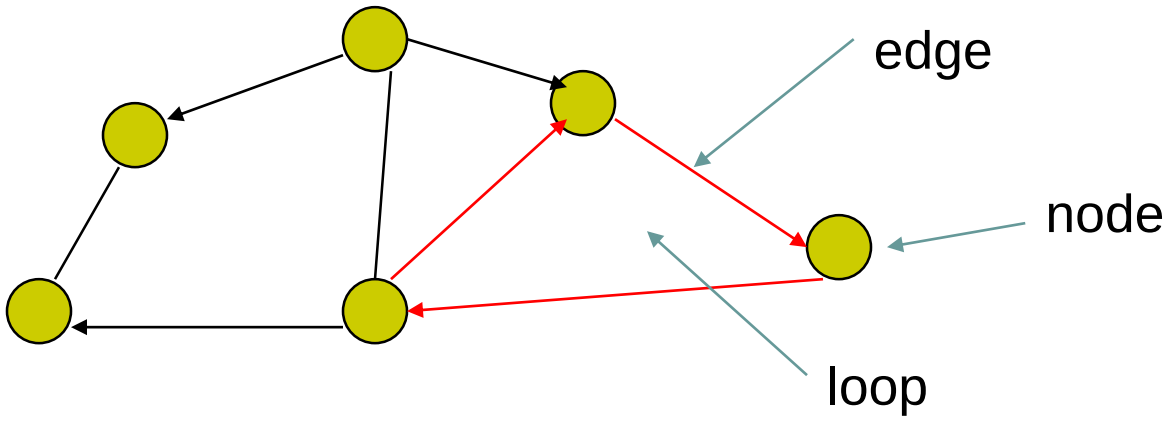


- For large objects with many parts, need to transform **groups** of objects
- Need better tools



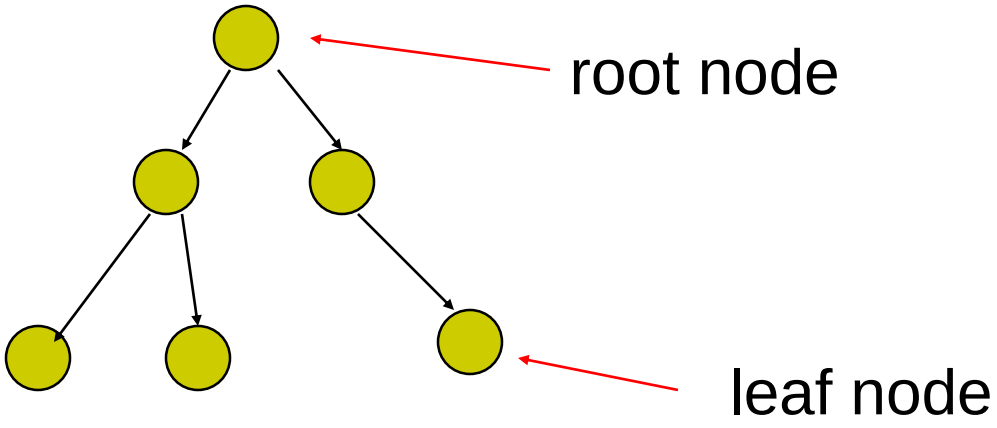
Graphs

- Set of ***nodes*** + ***edges (links)***
- **Edge** connects a pair of nodes
  - Directed or undirected
- ***Cycle*****:** directed path that is a loop

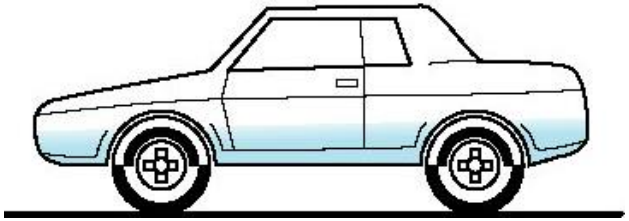


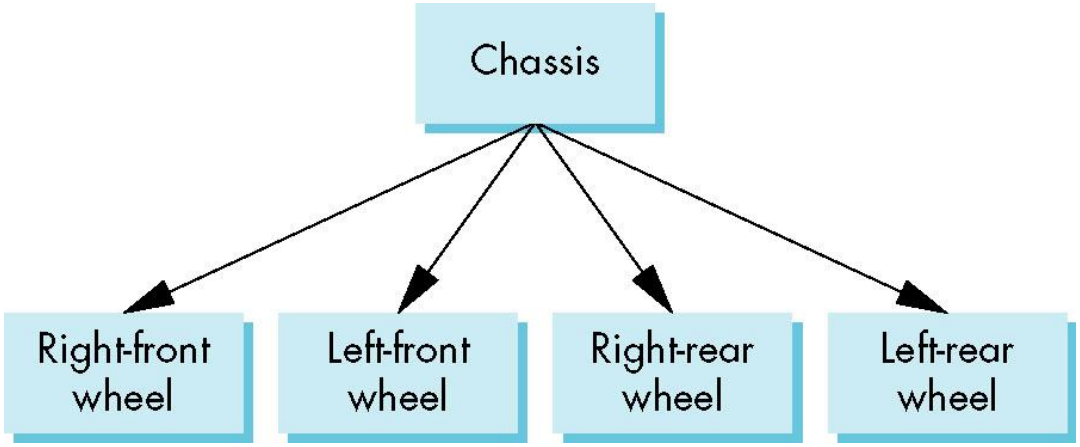
Tree

- Graph in which each node (except root) has exactly one parent node
  - A parent may have multiple children
  - Leaf node: no children



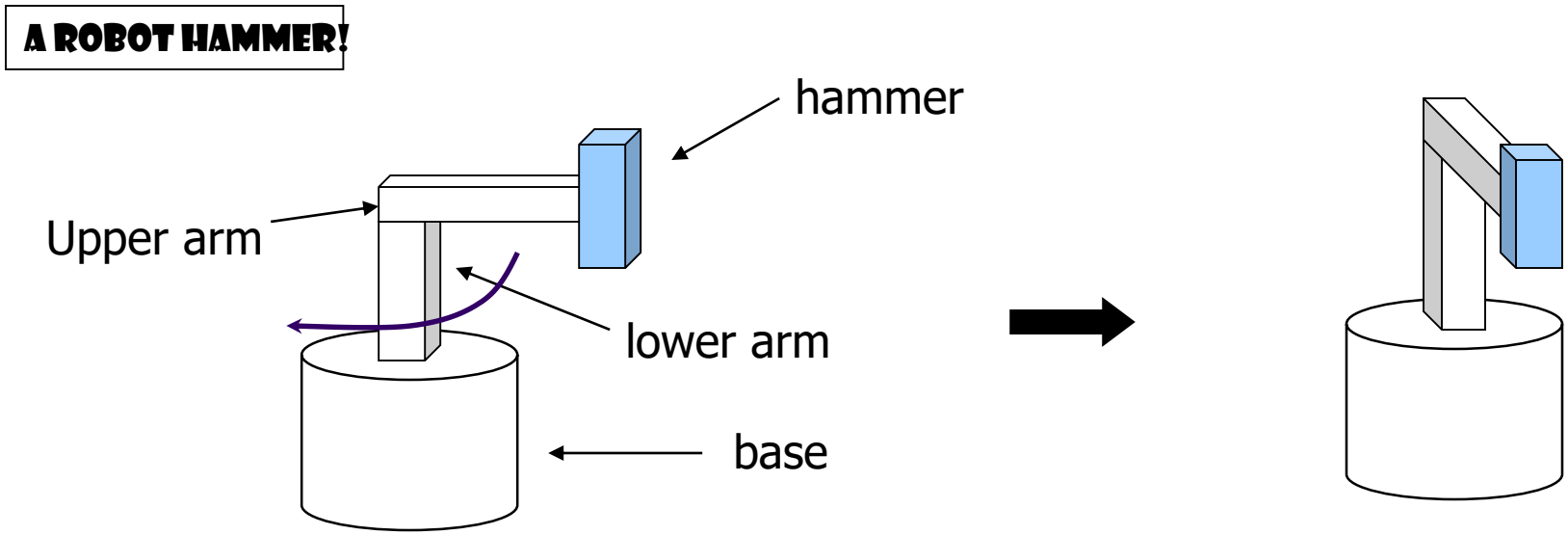
Tree Model of Car





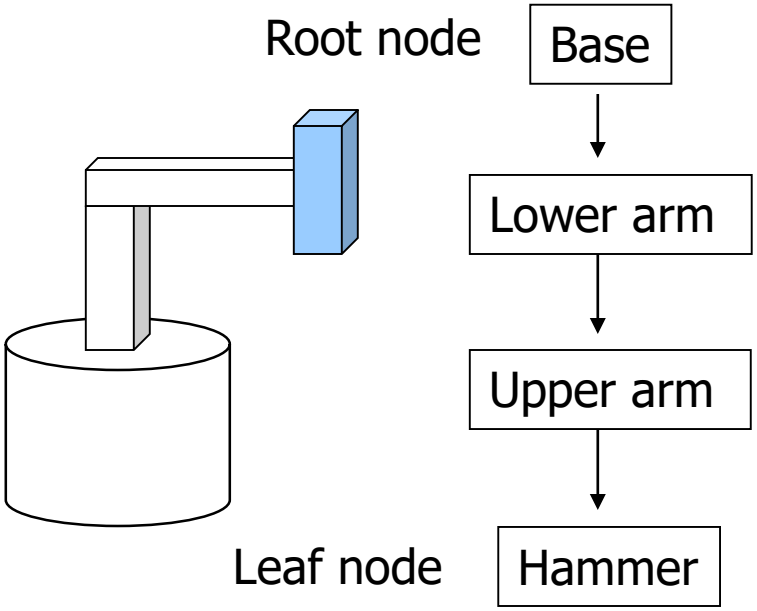
Hierarchical Transforms

- **Robot arm:** Many small **connected** parts
- Attributes (position, orientation, etc) depend on each other



Hierarchical Transforms

- Object dependency description using tree structure

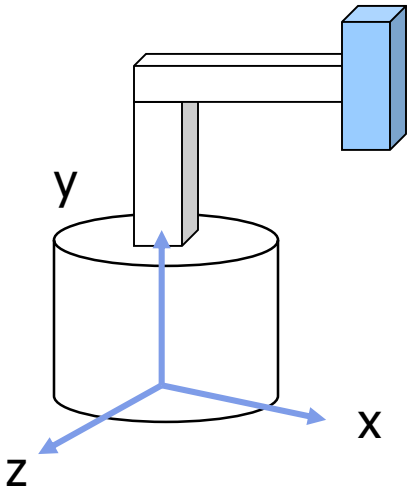


Object position and orientation can be affected by its parent, grand-parent, grand-grand-parent … nodes

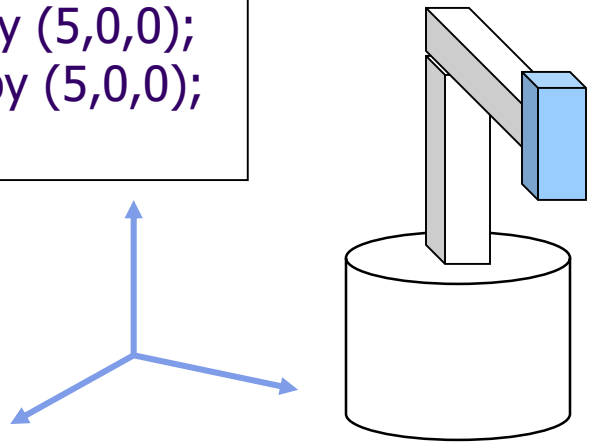
Hierarchical representation is known as a **Scene Graph**

Transformations

- Two ways to specify transformations:
  - **(1) Absolute transformation:** each part transformed independently (relative to origin)



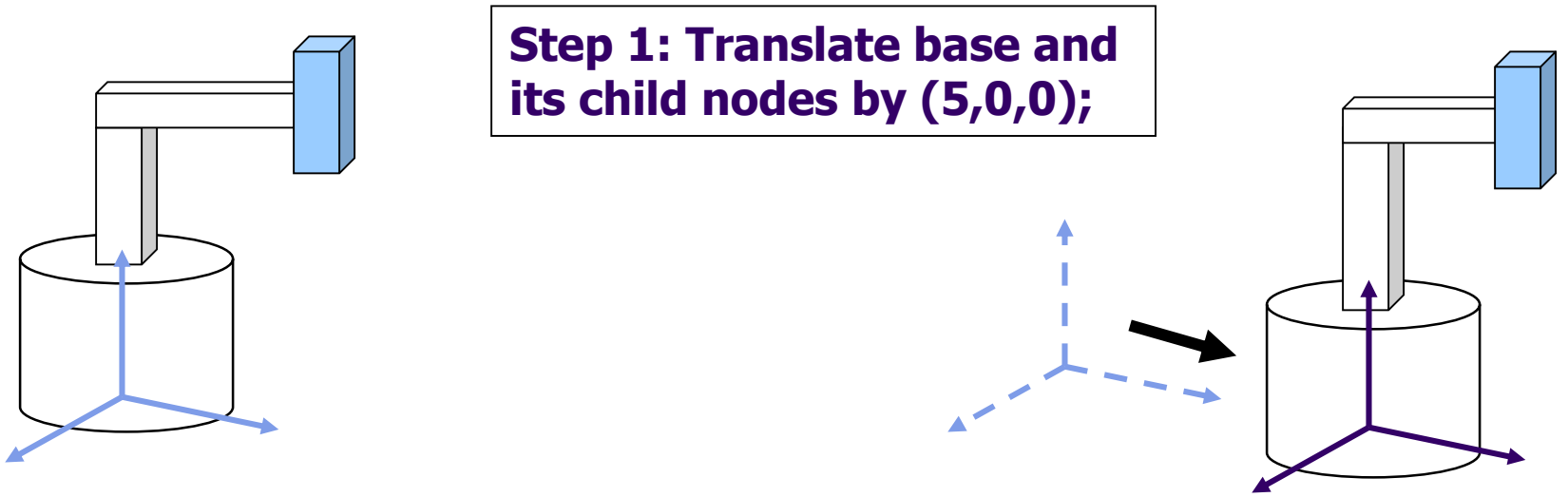

```
Translate the base by (5,0,0);
Translate the lower arm by (5,0,0);
Translate the upper arm by (5,0,0);
...
```



Relative Transformation

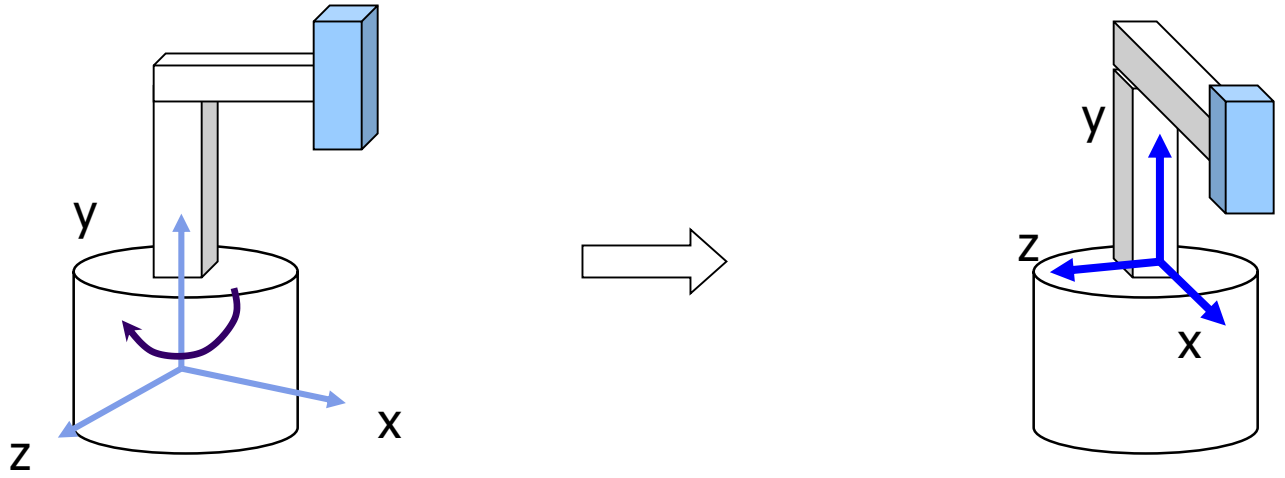
A better (and easier) way:

(2) **Relative transformation:** Specify transformation for each object relative to its parent



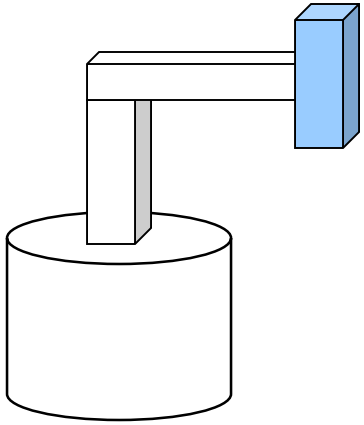
Relative Transformation

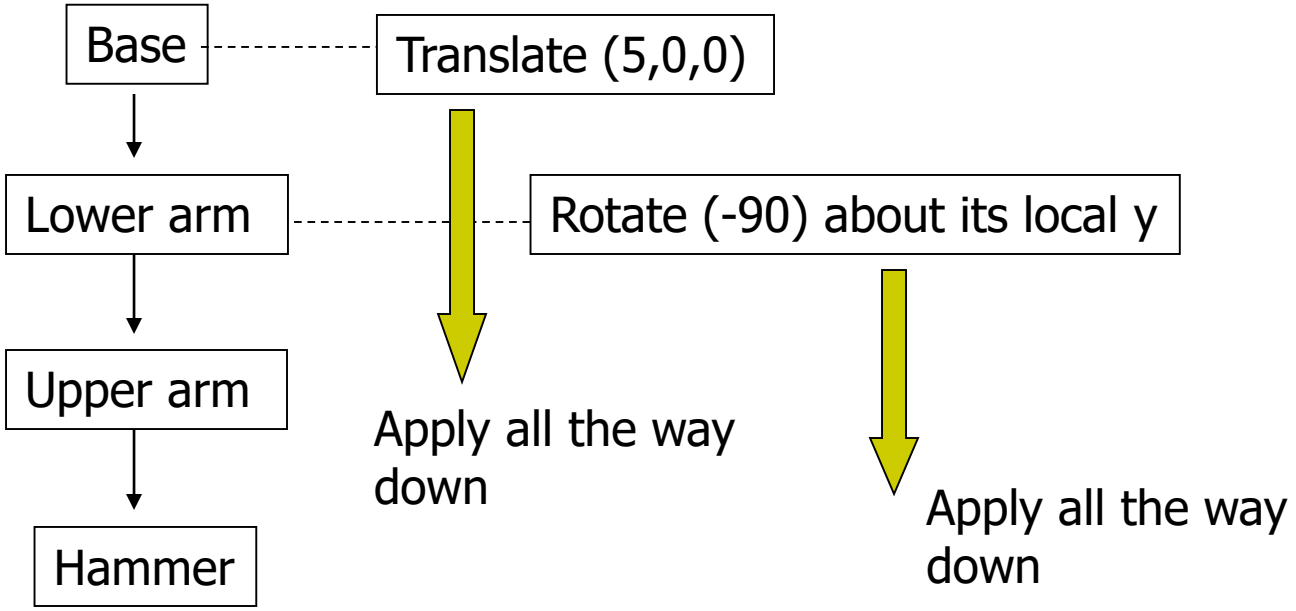
Step 2: Rotate the lower arm and all its descendants relative to the base's local y axis by -90 degree



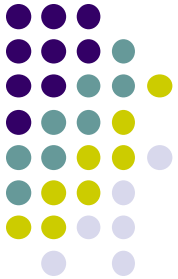
Relative Transformation

- Relative transformation using scene graph





Hierarchical Modeling



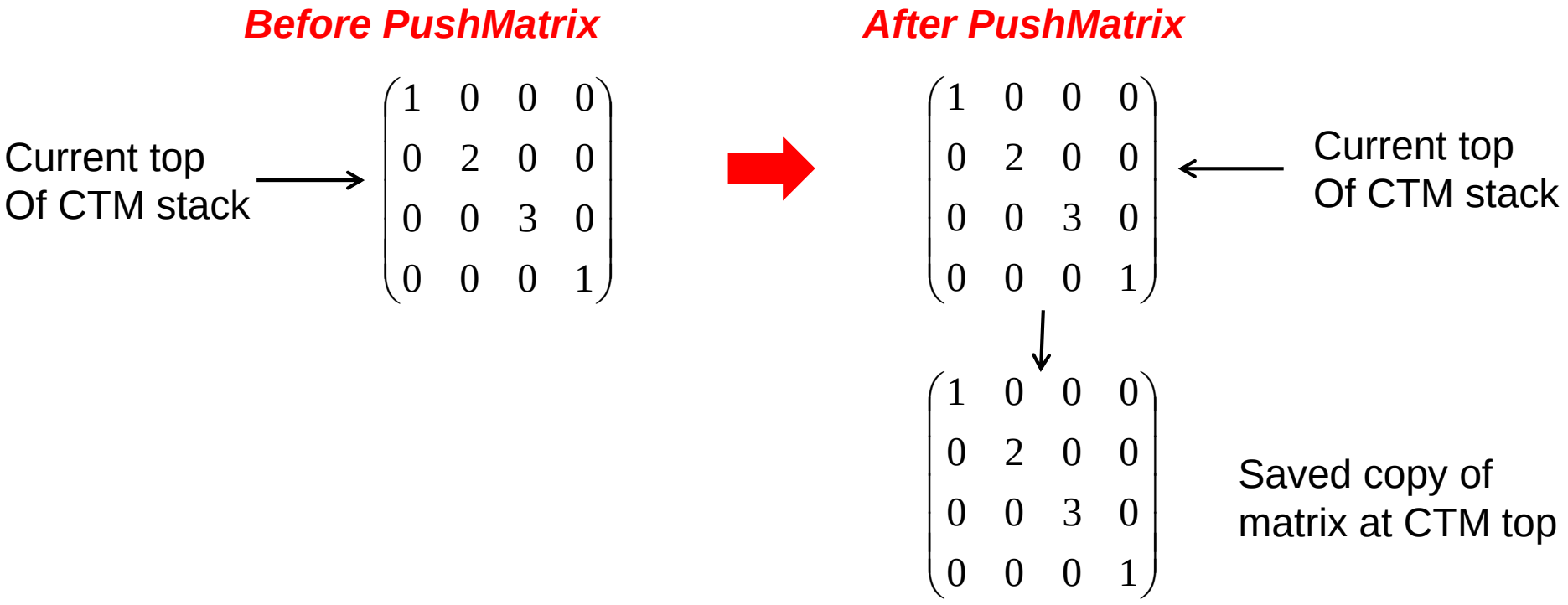
- Previous CTM had 1 level
- **Hierarchical modeling:** extend CTM to stack with multiple levels using linked list
- Manipulate stack levels using 2 operations
  - pushMatrix
  - popMatrix

Current top
Of CTM stack

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

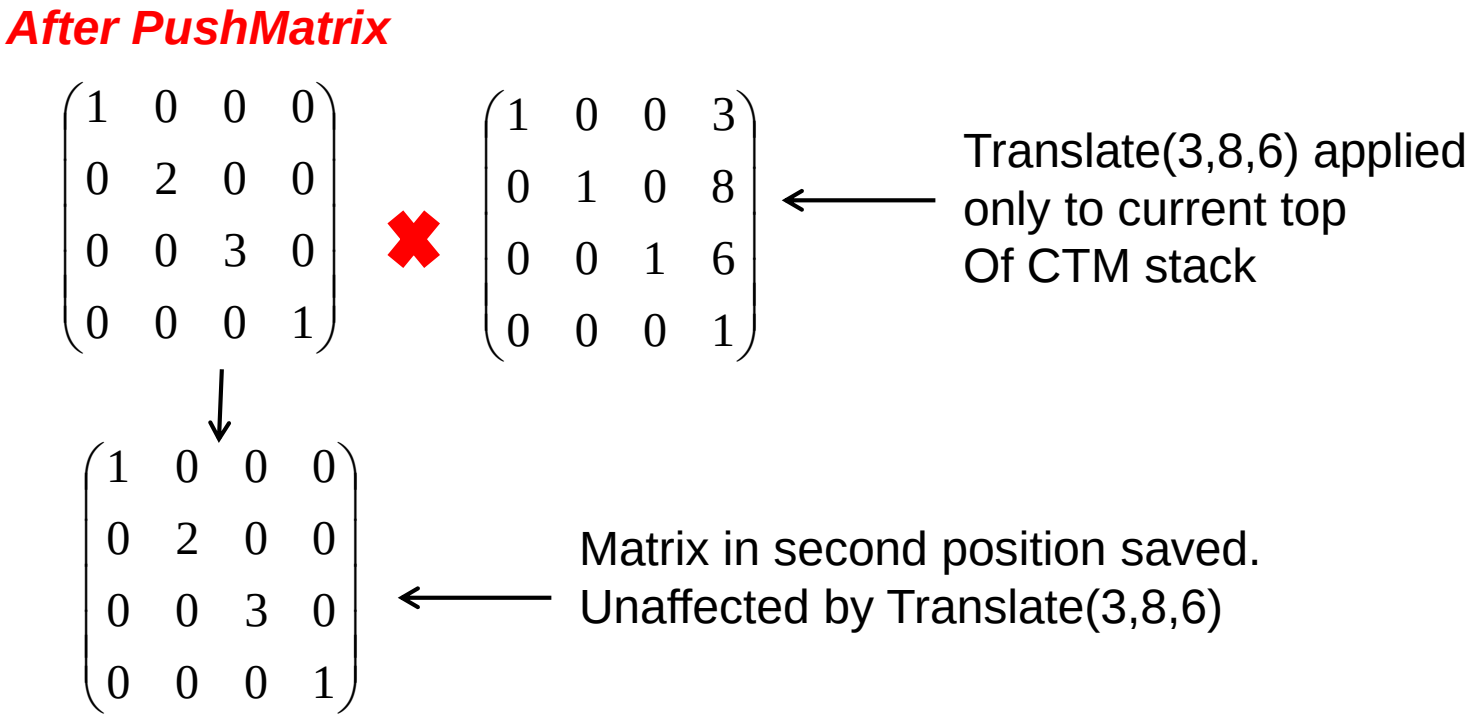
PushMatrix

- **PushMatrix( ):** Save current modelview matrix (CTM) in stack
- Positions 1 & 2 in linked list **are same** after PushMatrix



PushMatrix

- Further Rotate, Scale, Translate affect only top matrix
- E.g. **`ctm = ctm * Translate (3,8,6)`**



PushMatrix

- Further Rotate, Scale, Translate affect only top matrix
- E.g. **`ctm = ctm * Translate (3,8,6)`**

After PushMatrix

$$\begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & 2 & 0 & 16 \\ 0 & 0 & 3 & 18 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

← Translate(3,8,6) applied only to current top Of CTM stack

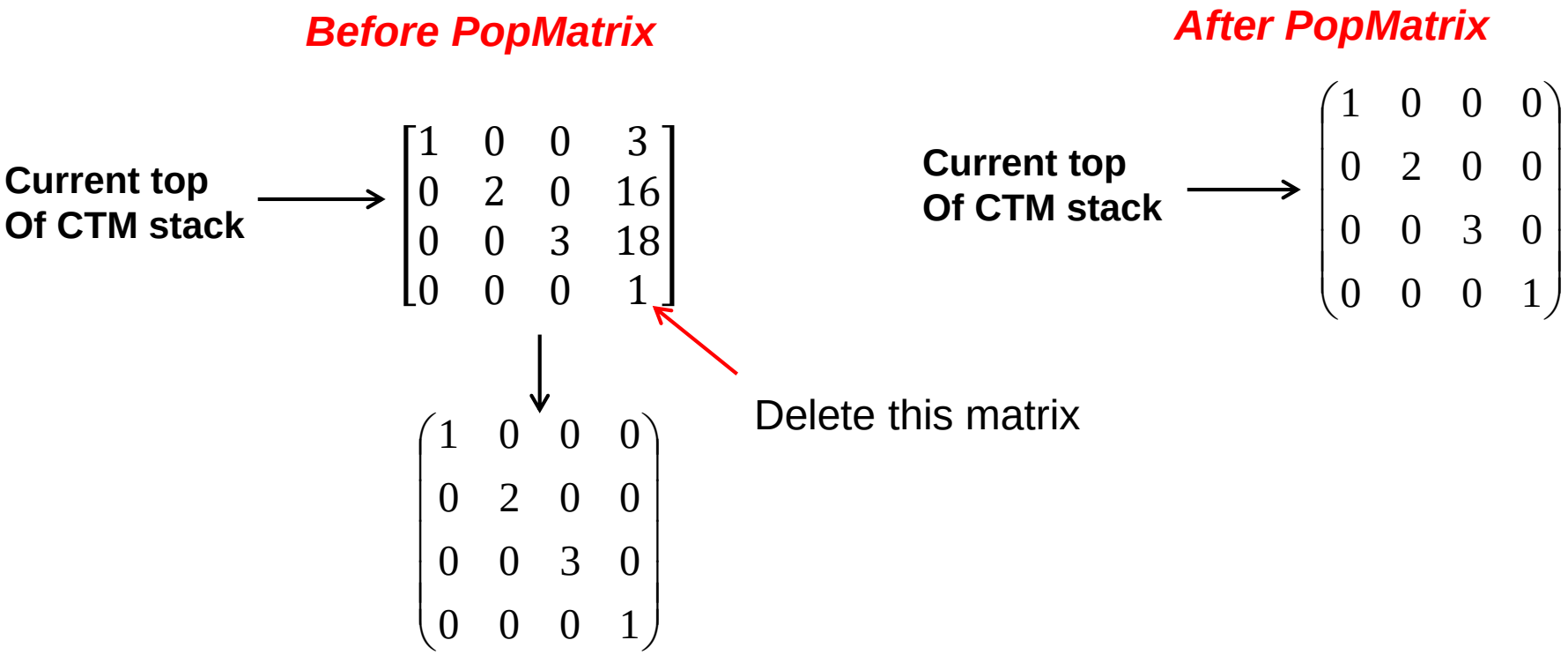
↓

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

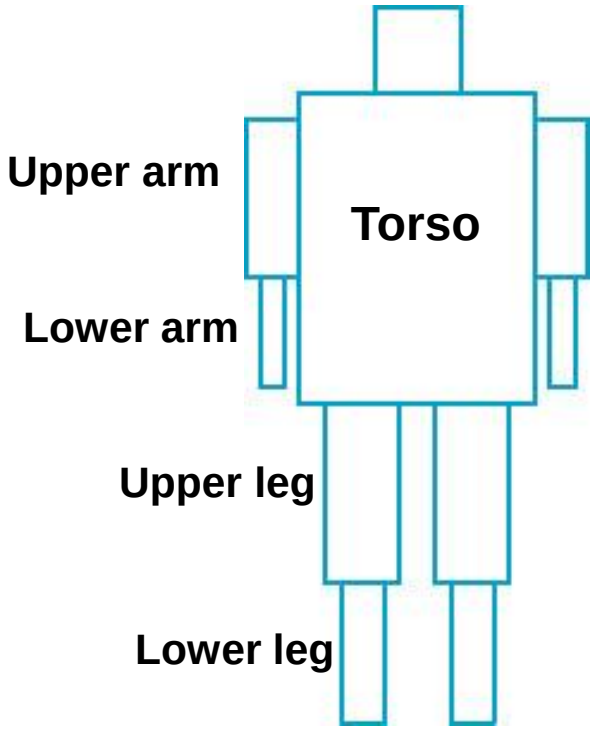
← Matrix in second position saved. Unaffected by Translate(3,8,6)

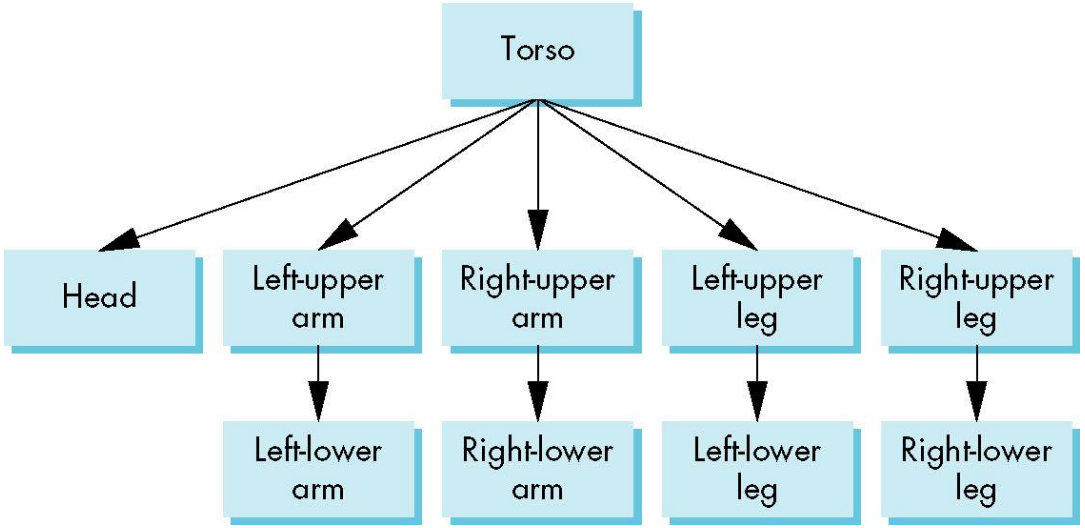
PopMatrix

- **PopMatrix( ):** Delete position 1 matrix, position 2 matrix becomes top



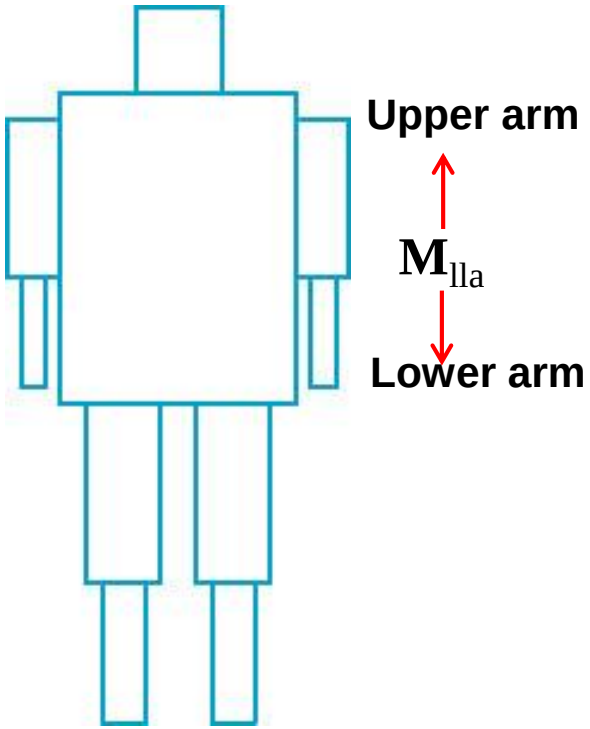
Humanoid Figure





Building the Model

- Draw each part as a function
  - `torso()`
  - `left_upper_arm(), etc`
- **Transform Matrices:** transform of node wrt its parent
  - $\mathbf{M}_{lla}$ positions left lower arm with respect to left upper arm
- Stack based traversal (push, pop)



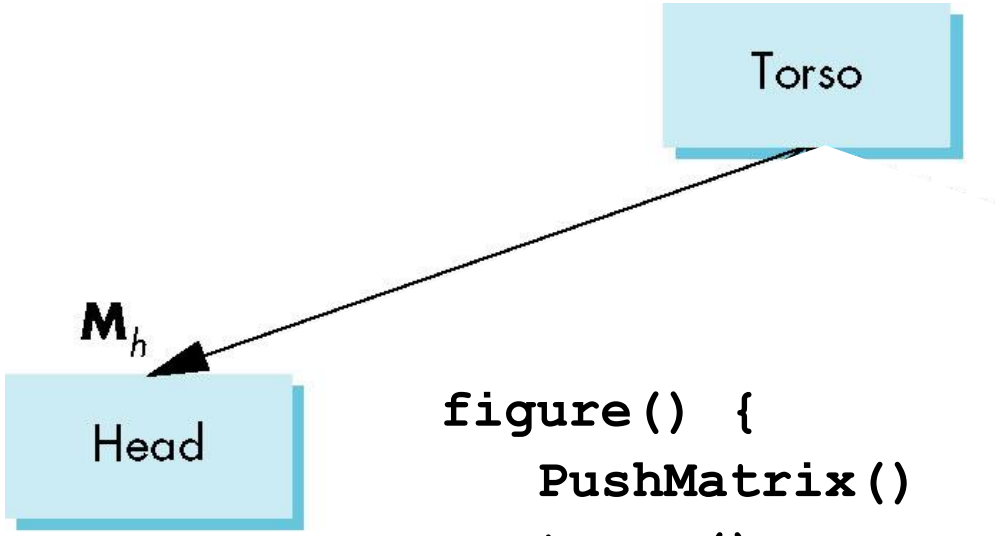
Draw Humanoid using Stack


```
figure() {
    PushMatrix()
    torso();
```

save present model-view matrix

draw torso

Draw Humanoid using Stack

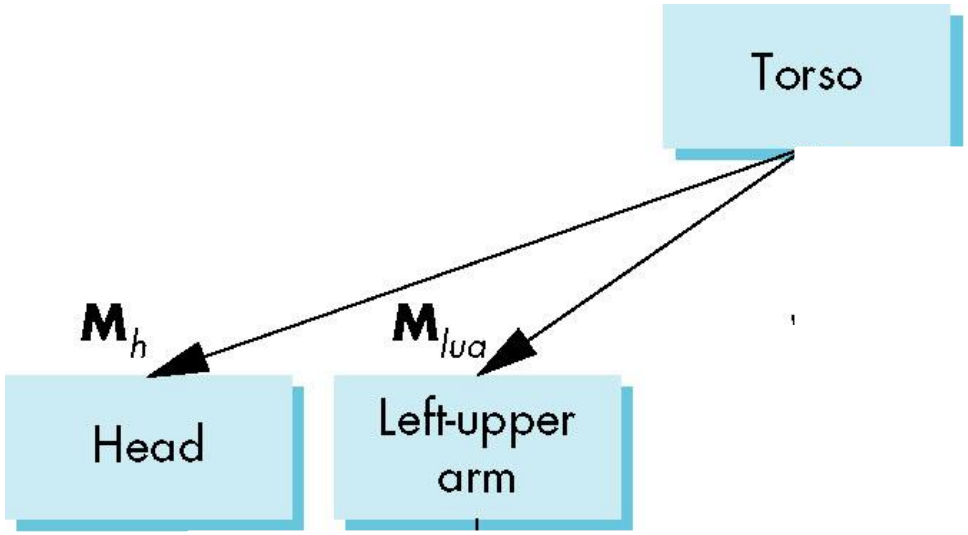



```
figure() {
    PushMatrix()
    torso();
    Rotate (…);
    head();
```

(M_h) Transformation of head Relative to torso

draw head

Draw Humanoid using Stack



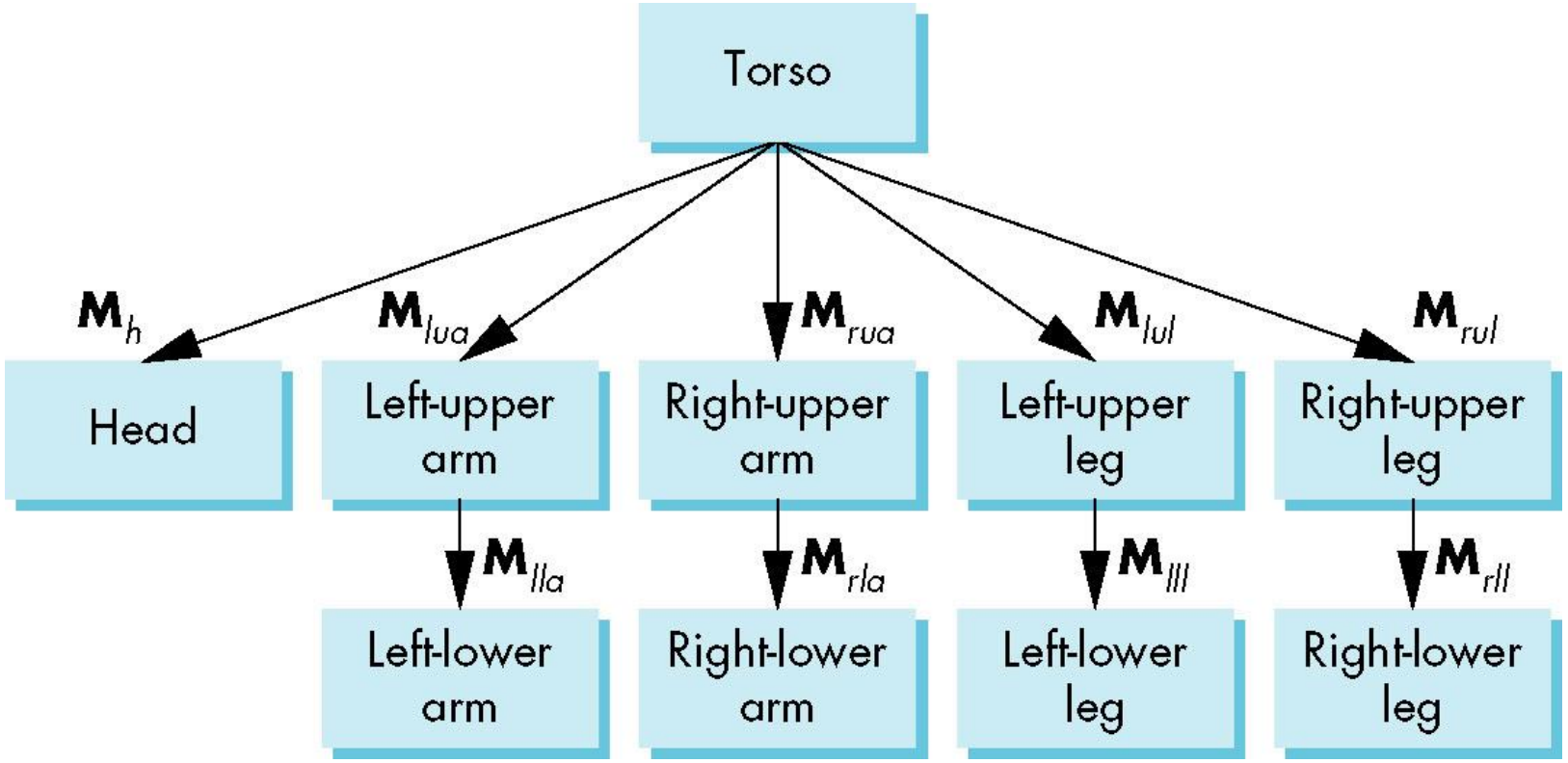
Go back to torso matrix, and save it again

(M_{lua}) Transformation(s) of left upper arm relative to torso

draw left-upper arm

```
PushMatrix()
torso();
Rotate (…);
head();
PopMatrix();
PushMatrix();
Translate(…);
Rotate(…);
left_upper_arm();
……..
// rest of code()
```

Complete Humanoid Tree with Matrices



Scene graph of Humanoid Robot