Computer Graphics (CS 543)
Lecture 6a: Hierarchical 3D Models

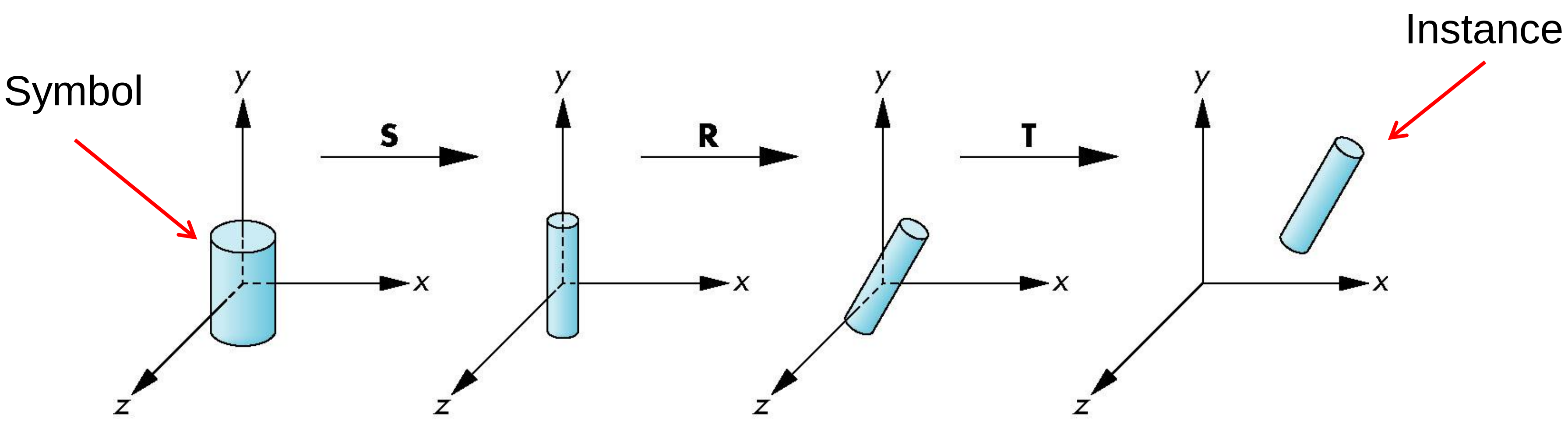
Prof Emmanuel Agu

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

Instance Transformation

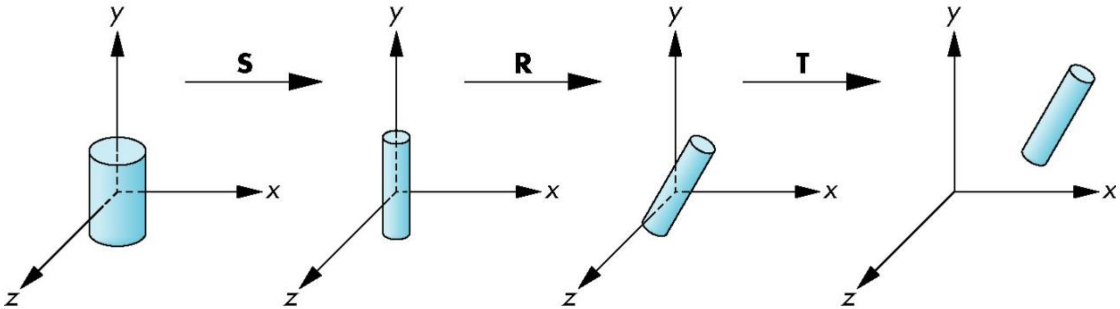
- Start with unique object (a *symbol*)
- Each appearance of object in model is an ***instance***
  - Then scale, orient, position (instance transformation)



Symbol-Instance Table

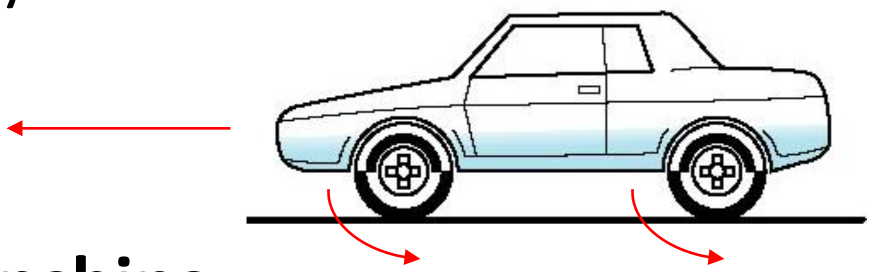
Approach 1: store **intances + instance transformations**

Symbol	Scale	Rotate	Translate
1	s_x, s_y, s_z	$\theta_x, \theta_y, \theta_z$	d_x, d_y, d_z
2			
3			
1			
1			
.			
.			



Problems with Symbol-Instance Table

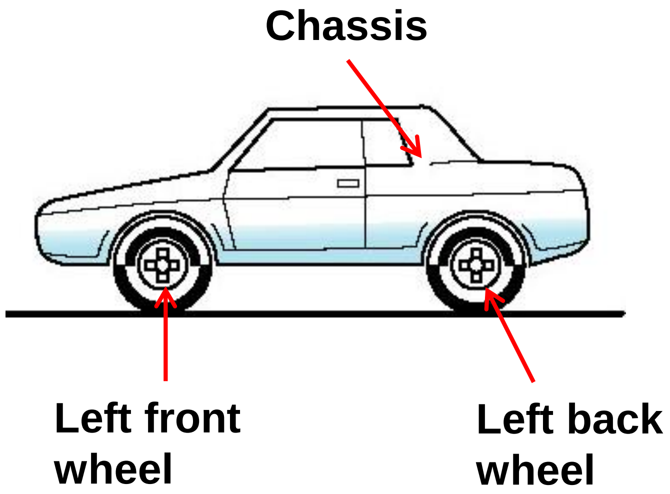
- Symbol-instance table does not show relationships between parts of model
- Consider model of car
  - Chassis (body) + 4 identical wheels
  - Two symbols



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

Structure Program Using Function Calls?

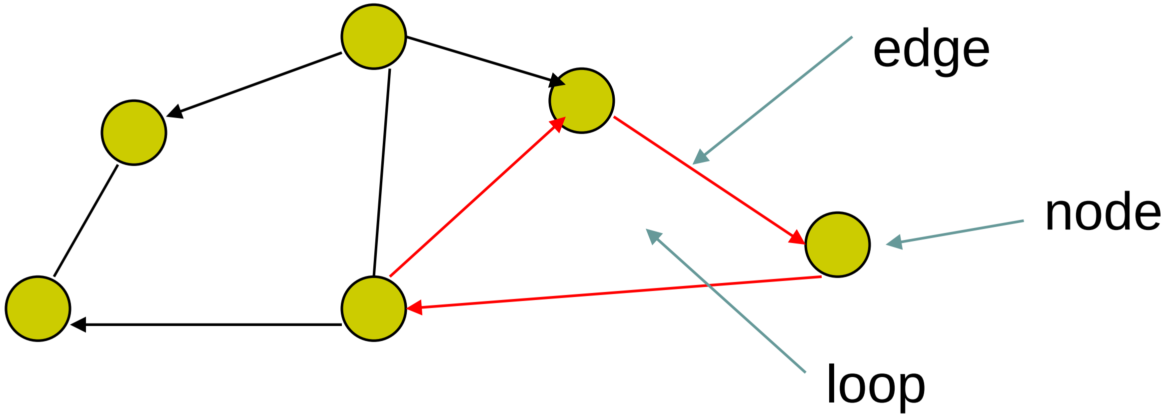
```
car(speed)
{
     chassis()
     wheel(right_front);
     wheel(left_front);
     wheel(right_rear);
     wheel(left_rear);
}
```



- Fails to show relationships between parts
- Explore graph representation

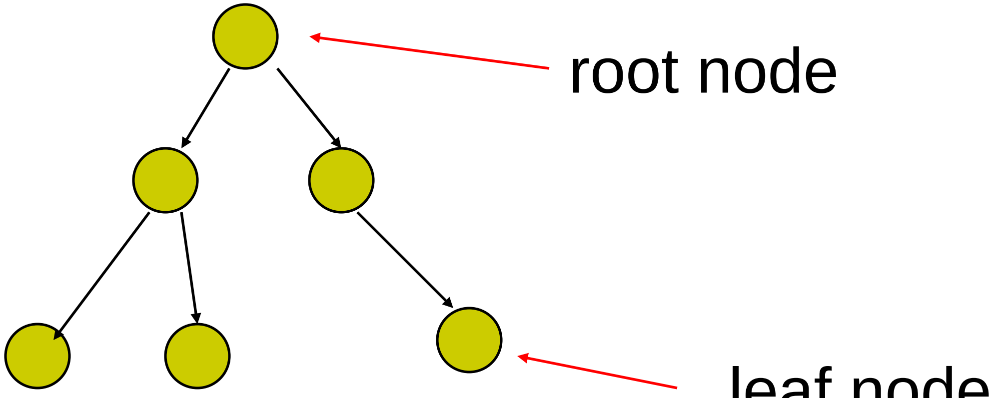
Graphs

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

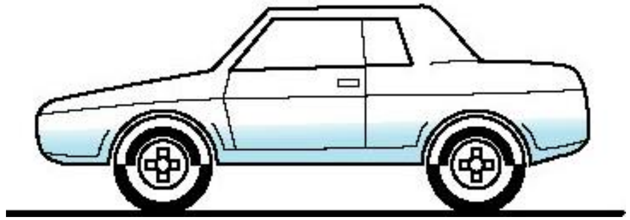


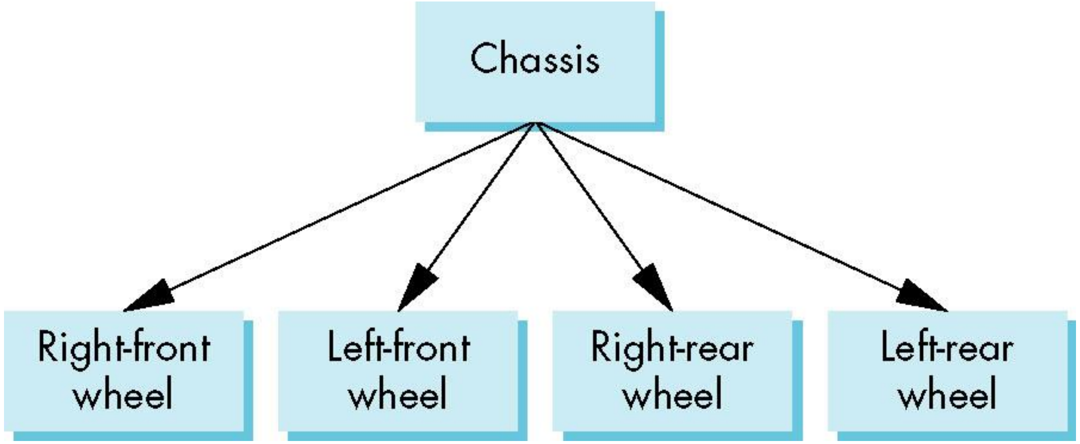
Tree

- 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

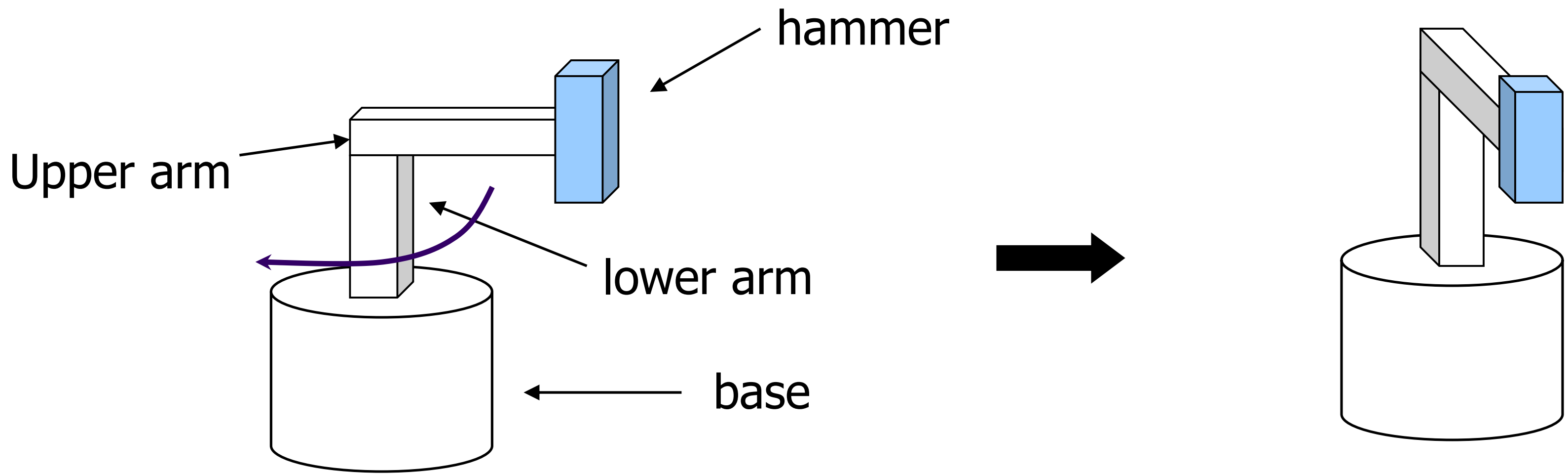




Hierarchical Transforms

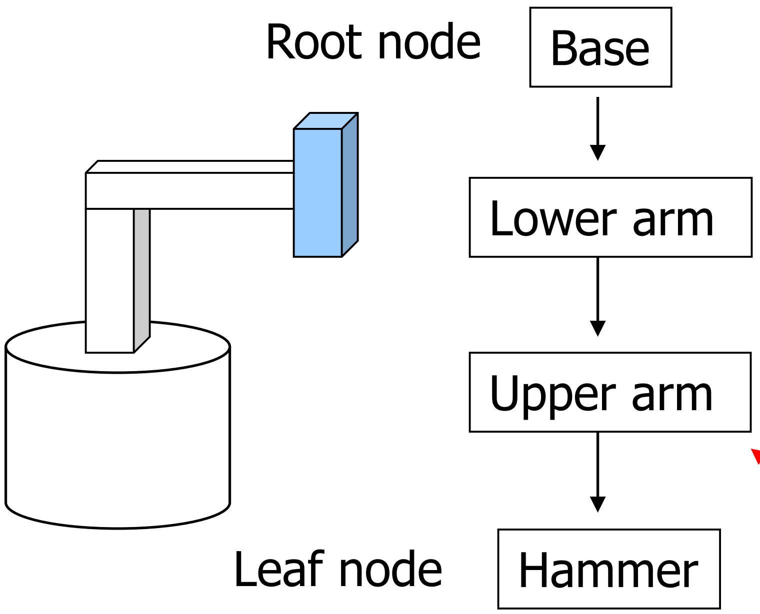
- **Robot arm:** Many small **connected** parts
- Attributes of parts (position, orientation, etc) depend on each other

A ROBOT HAMMER!



Hierarchical Transforms

- Object dependency description using tree structure

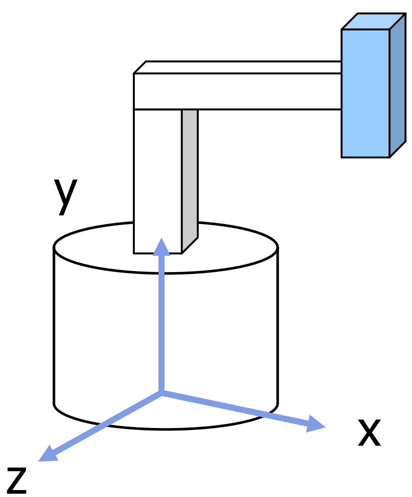


Object position and orientation can be affected by its parent, grand-parent, grand-grand-parent … nodes

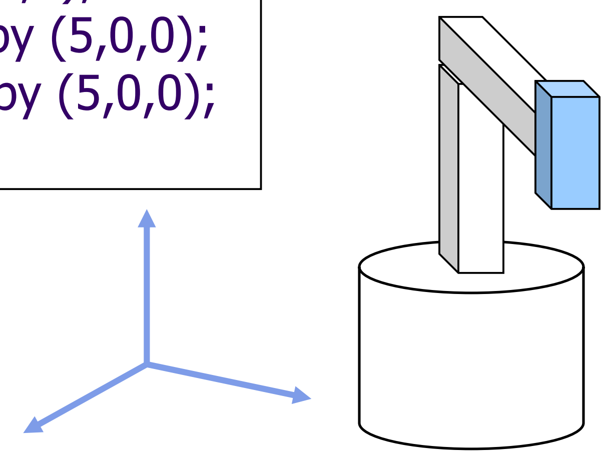
Hierarchical representation is known as a **Scene Graph**

Transformations

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



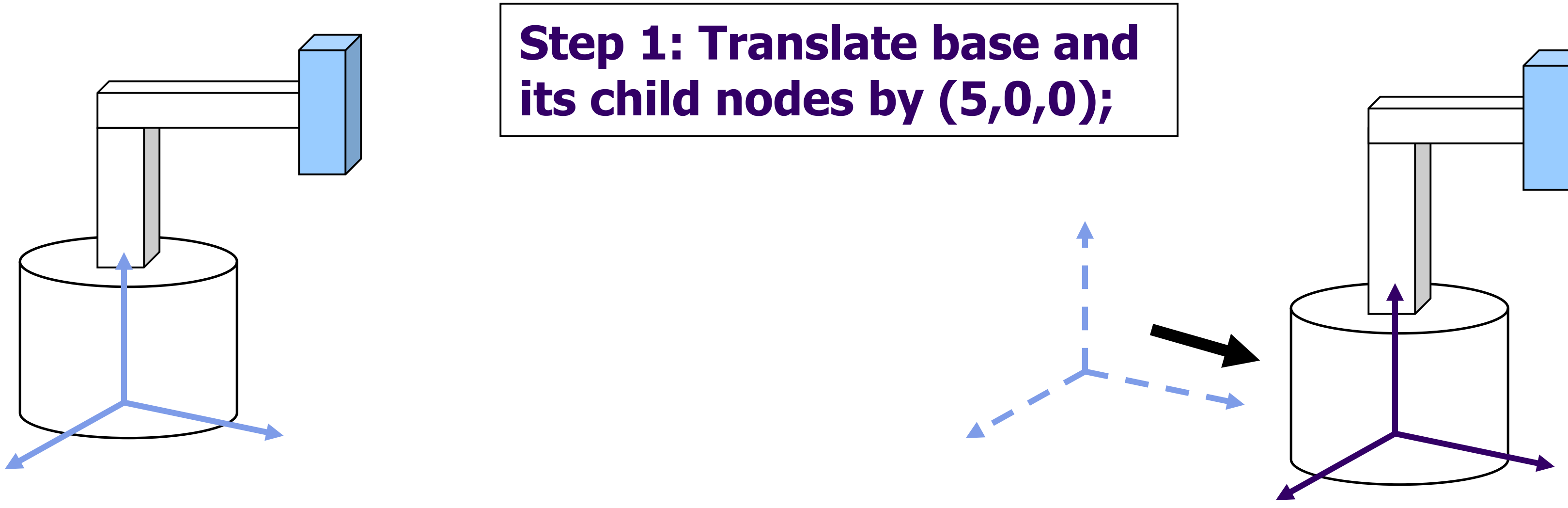
Translate the base by (5,0,0);
Translate the lower arm by (5,0,0);
Translate the upper arm by (5,0,0);
…



Relative Transformation

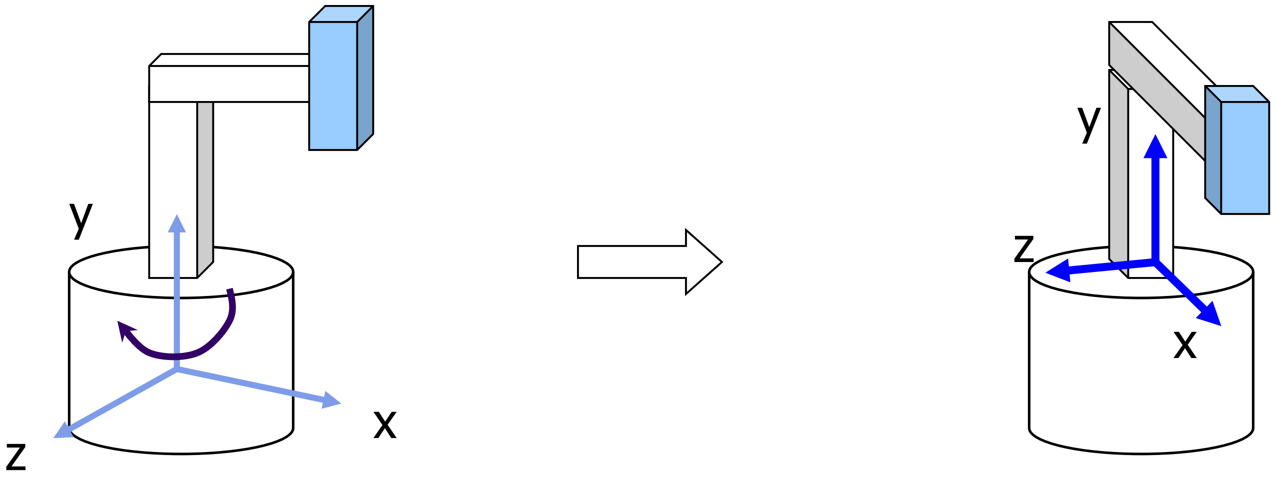
A better (and easier) way:

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



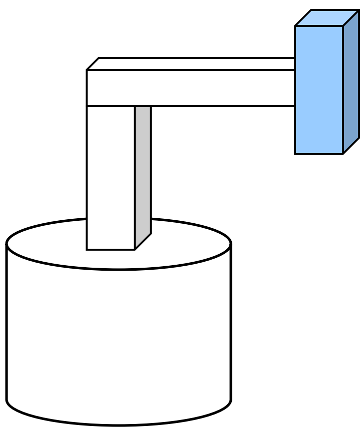
Relative Transformation

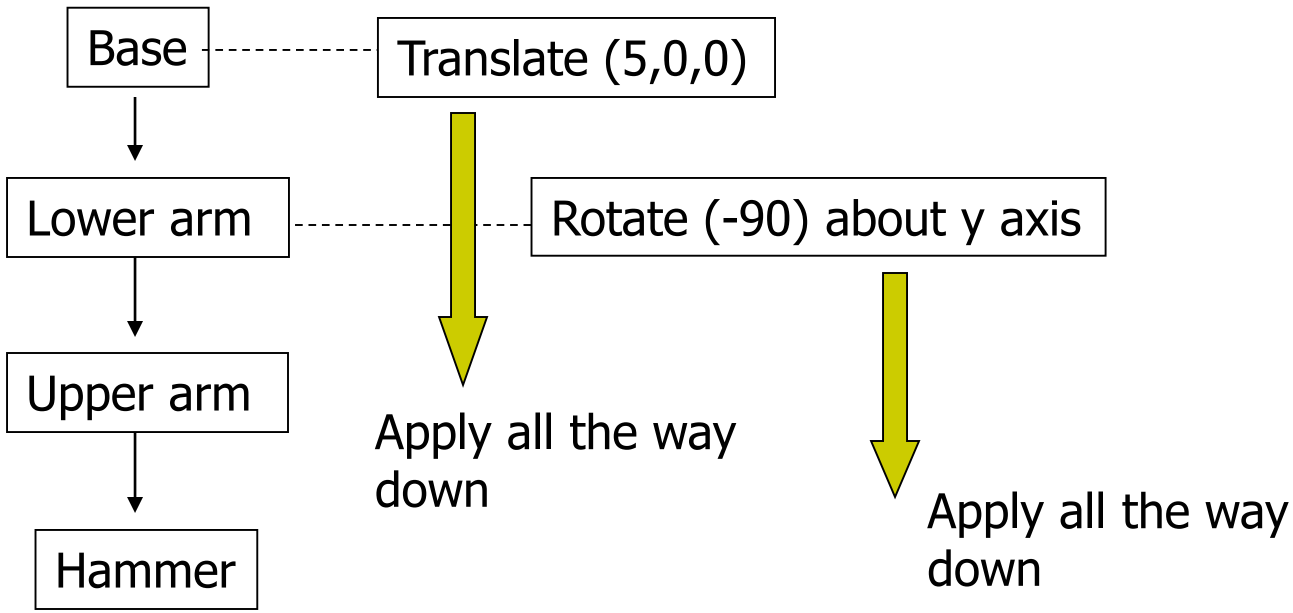
Step 2: Rotate the lower arm and all its descendants by -90 degrees, relative to the base's local y axis



Relative Transformation

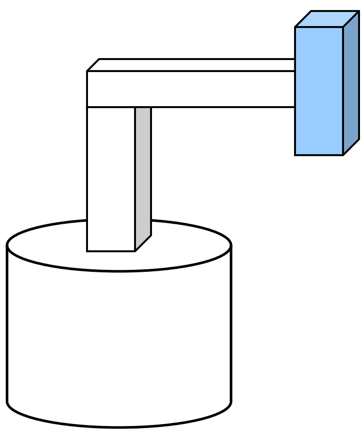
- Relative transformation using scene graph

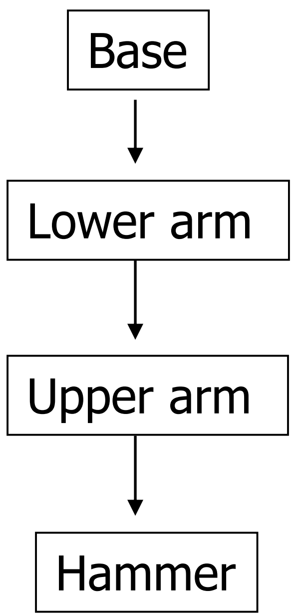




Hierarchical Transforms Using OpenGL

- Translate base and all its descendants by (5,0,0)
- Rotate lower arm and its descendants by -90 degree about local y

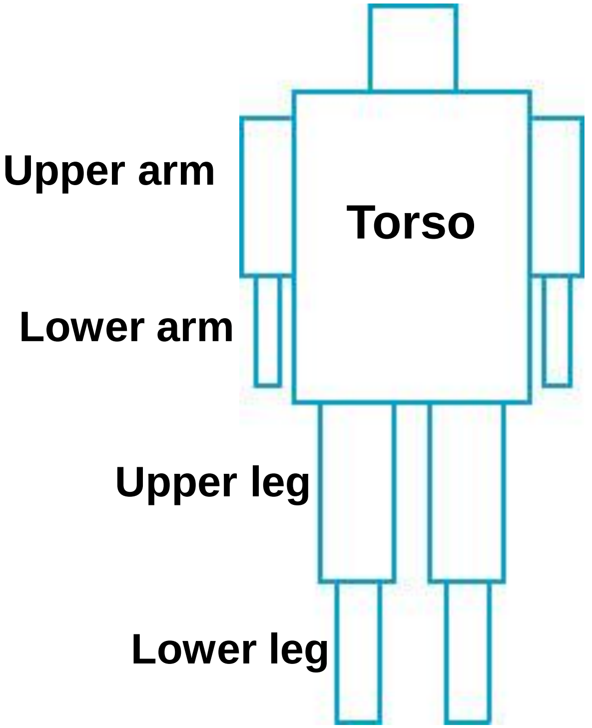





```
ctm = LoadIdentity();

… // setup your camera

ctm = ctm * Translatef(5,0,0);

Draw_base();

ctm = ctm * Rotatef(-90, 0, 1, 0);

Draw_lower _arm();
Draw_upper_arm();
Draw_hammer();
```

Hierarchical Modeling

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



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

PushMatrix

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

Before PushMatrix

Current top
Of CTM stack →

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

After PushMatrix

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

← Current top
Of CTM stack

↓

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

Saved copy of
matrix at CTM top

PushMatrix

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

After PushMatrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

← 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. Unchanged by Translate(3,8,6)

PopMatrix

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

Before PopMatrix

Current top Of CTM stack →

$$\begin{pmatrix} 1 & 5 & 4 & 0 \\ 0 & 2 & 2 & 0 \\ 0 & 6 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

← Delete this matrix

↓

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

After PopMatrix

Current top Of CTM stack →

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

PopMatrix and PushMatrix Illustration

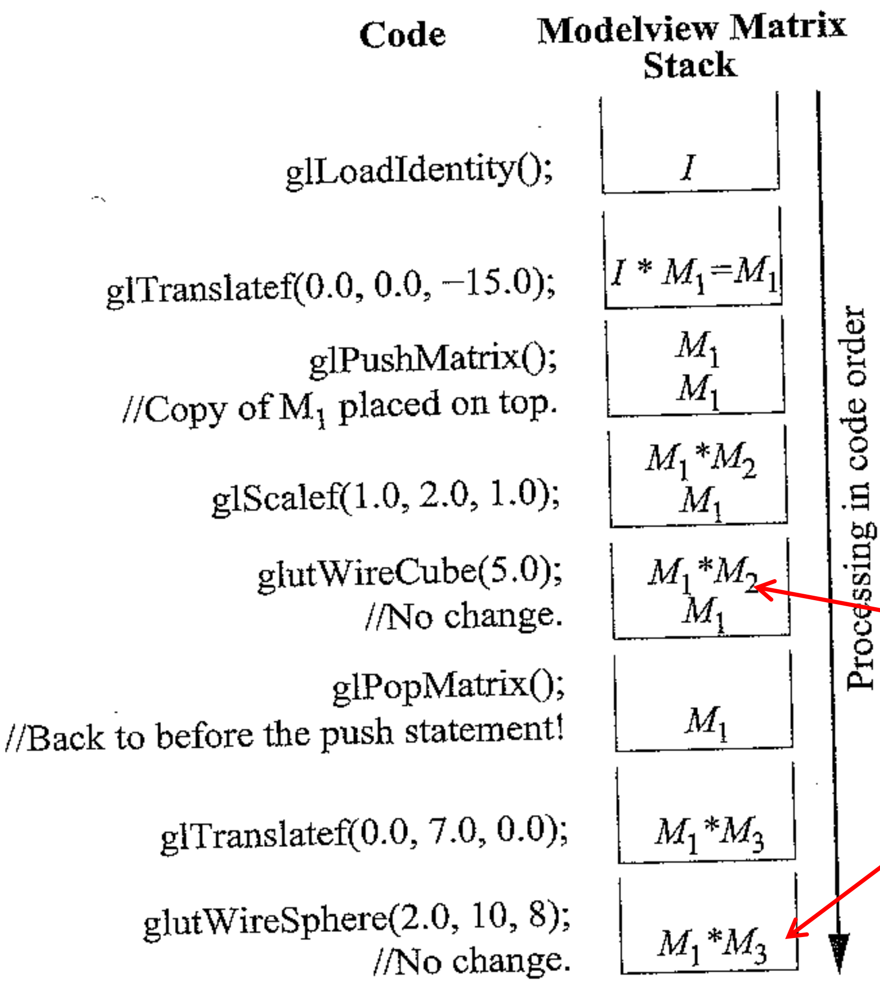


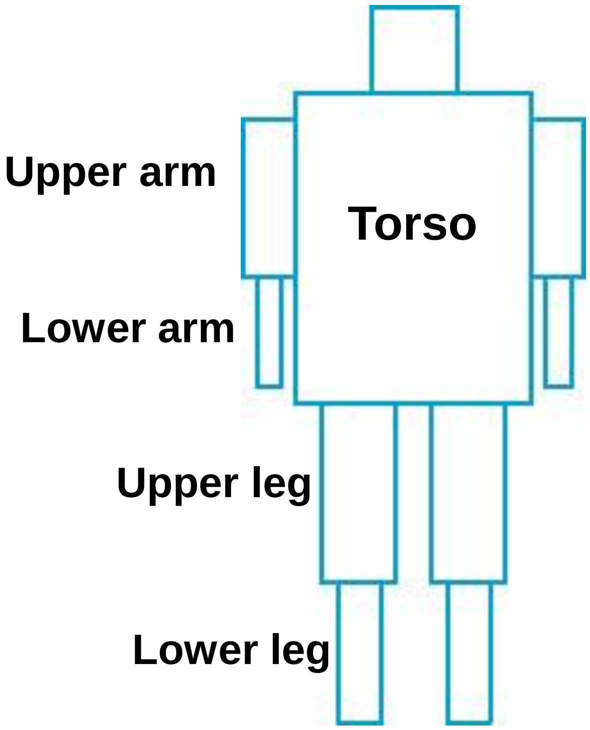
Figure 4.19: Transitions of the modelview matrix stack.

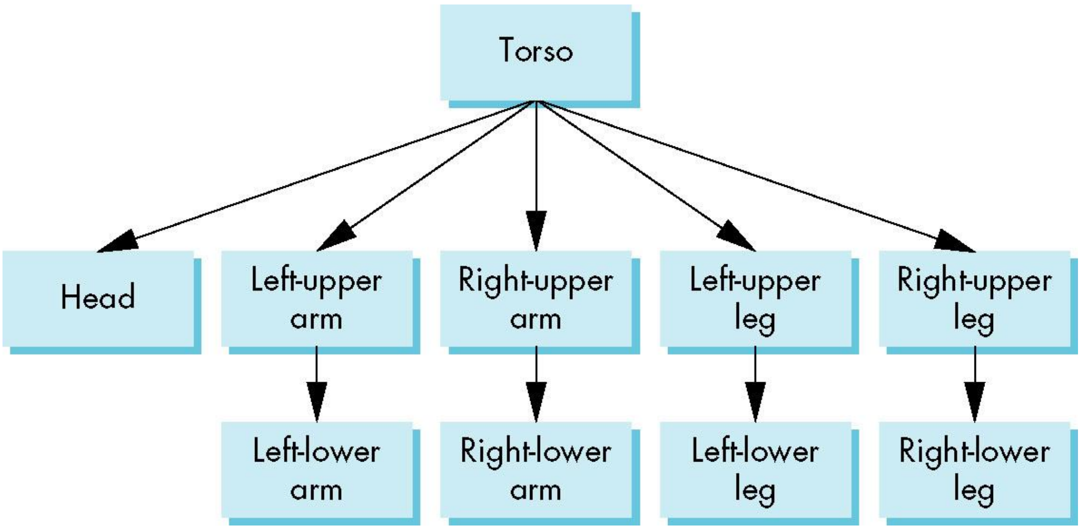
- **Note:** Diagram uses old glTranslate, glScale, etc commands. **Deprecated!!**
- We want same behavior though

Apply matrix at top of CTM to vertices of object created

Ref: Computer Graphics Through OpenGL by Guha

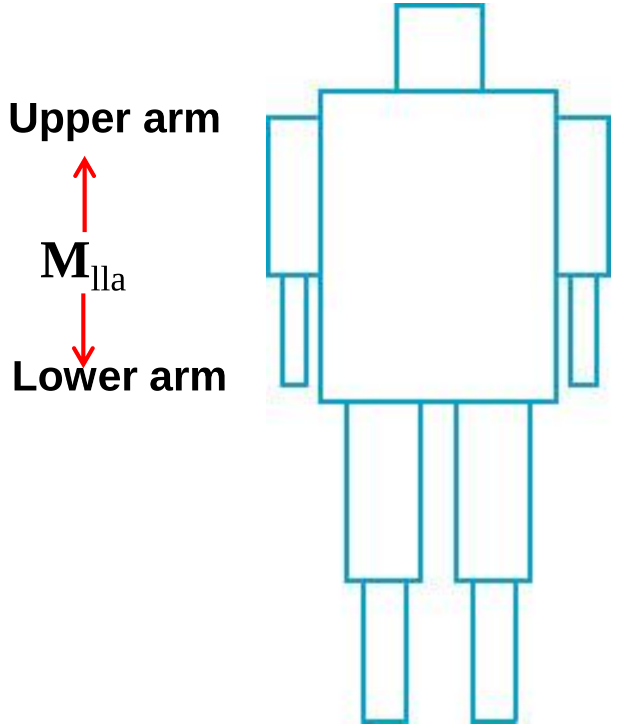
Humanoid Figure





Building the Model

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



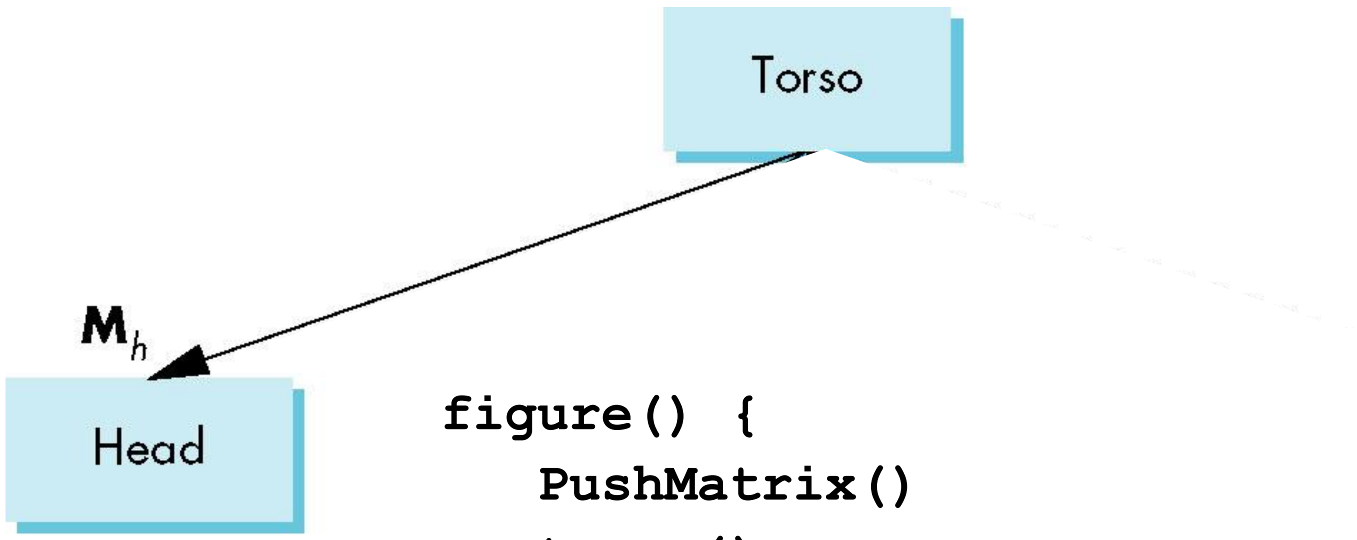
Draw Humanoid using Stack


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

save present model-view matrix

draw torso

Draw Humanoid using Stack

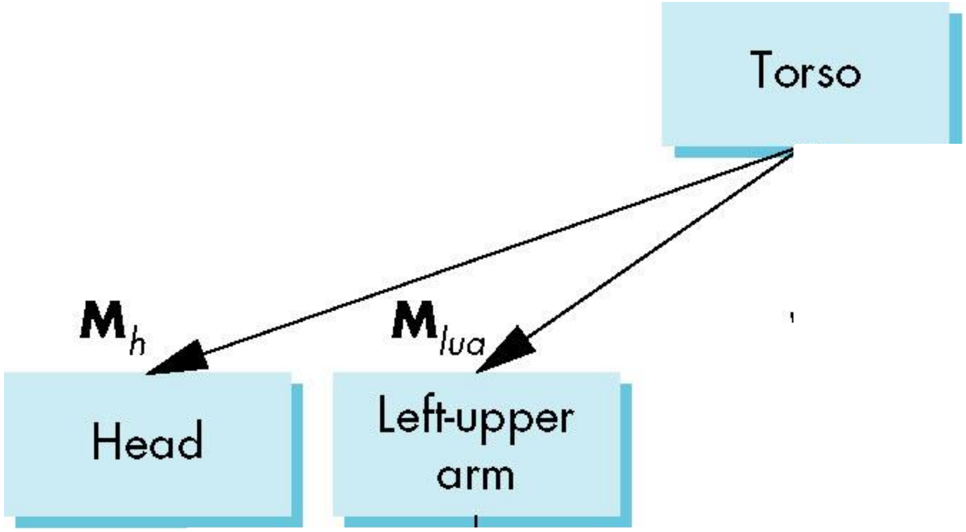



```
figure() {
    PushMatrix()
    torso();
    Rotate (…);
    head();
```

(M_h) Transformation of head Relative to torso (points to `Rotate (…);`)

draw head (points to `head();`)

Draw Humanoid using Stack



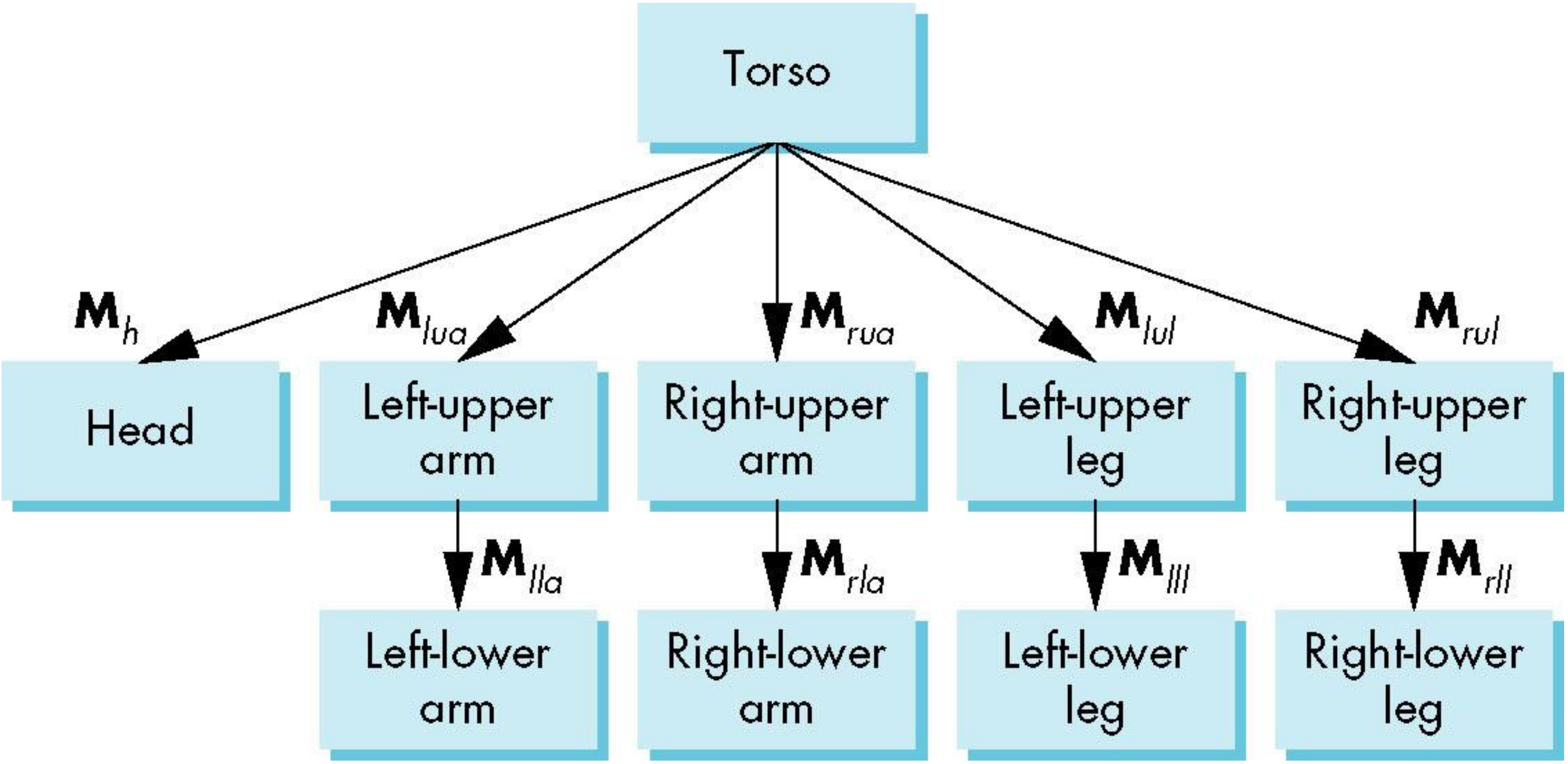
Go back to torso matrix, and save it again

(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()
```

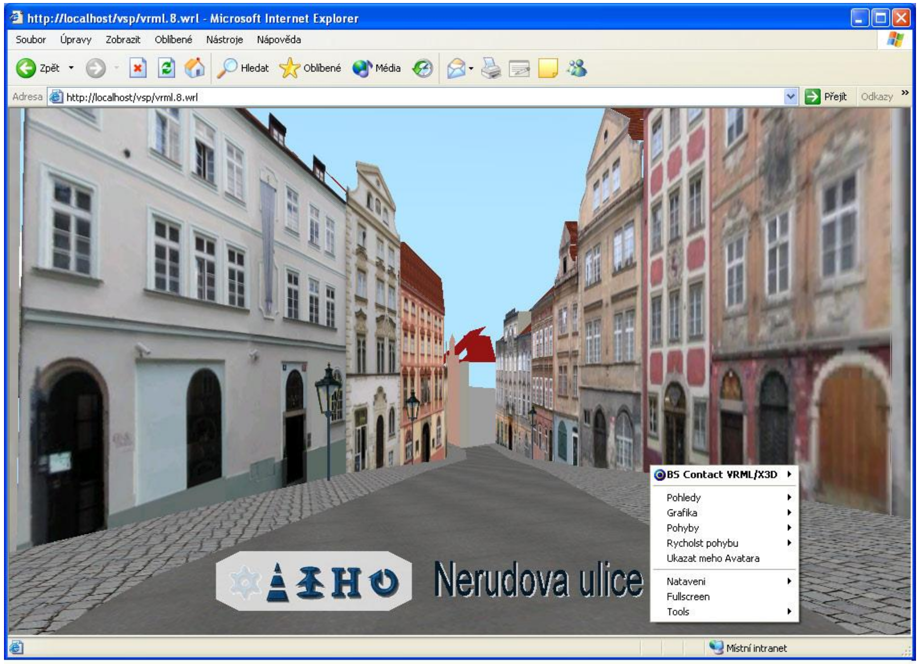
Complete Humanoid Tree with Matrices



Scene graph of Humanoid Robot

VRML

- Scene graph introduced by SGI Open Inventor
- Used in many graphics applications (Maya, etc)
- Virtual Reality Markup Language
  - Scene graph representation of virtual worlds on Web
  - Scene parts can be distributed across multiple web servers
  - Implemented using OpenGL



References

- Angel and Shreiner, Interactive Computer Graphics (6^{th} edition), Chapter 8