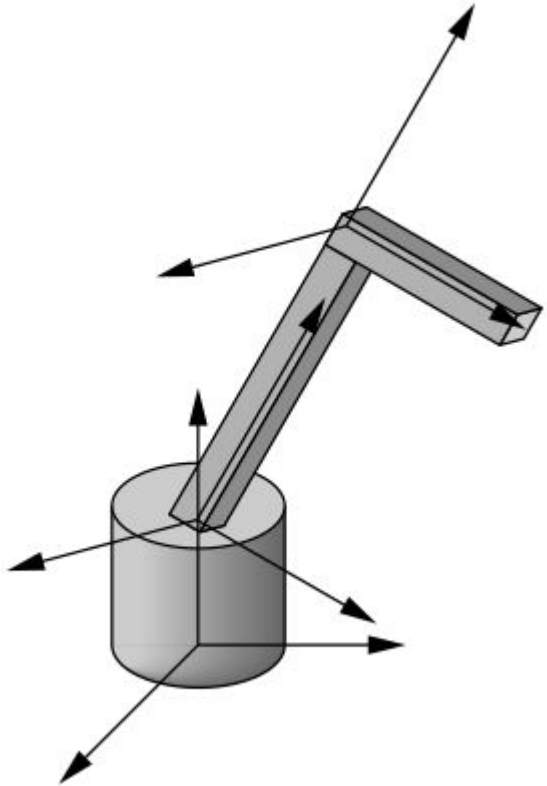
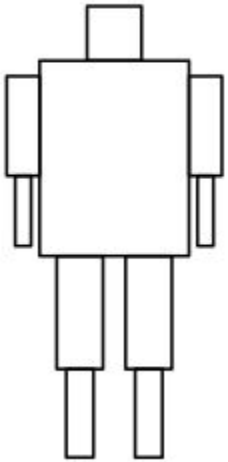
Hierarchical & Scene Graphs

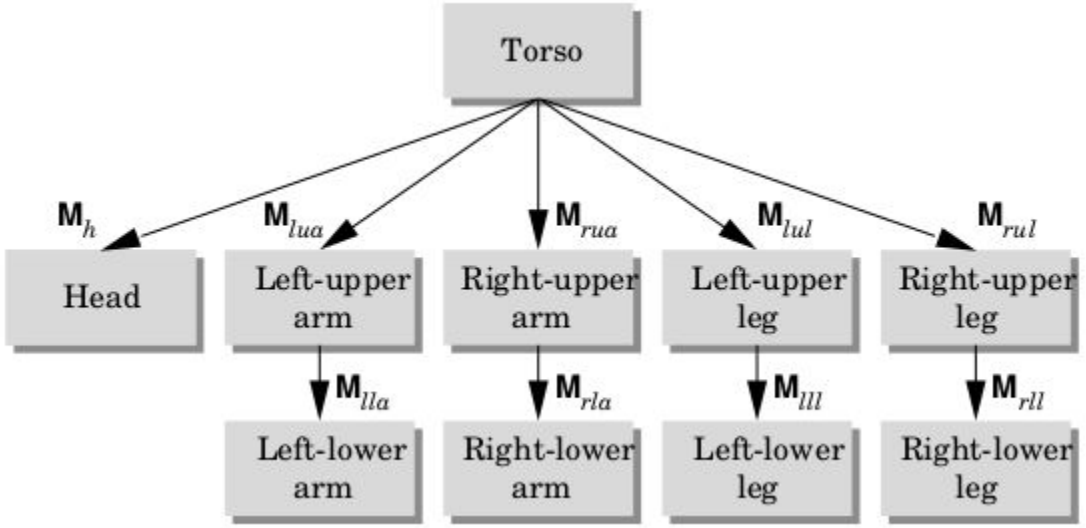
Robot Arm transformations:

R (Base) {
{ T R (lower arm)
{ T R (upper arm) } }

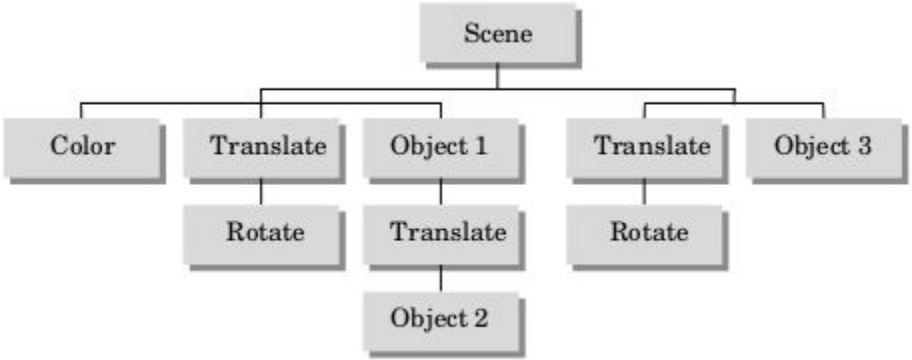


Hierarchical & Scene Graphs





Hierarchical & Scene Graphs



DAG = directed acyclic graph

depth first traversal: left child, right sibling

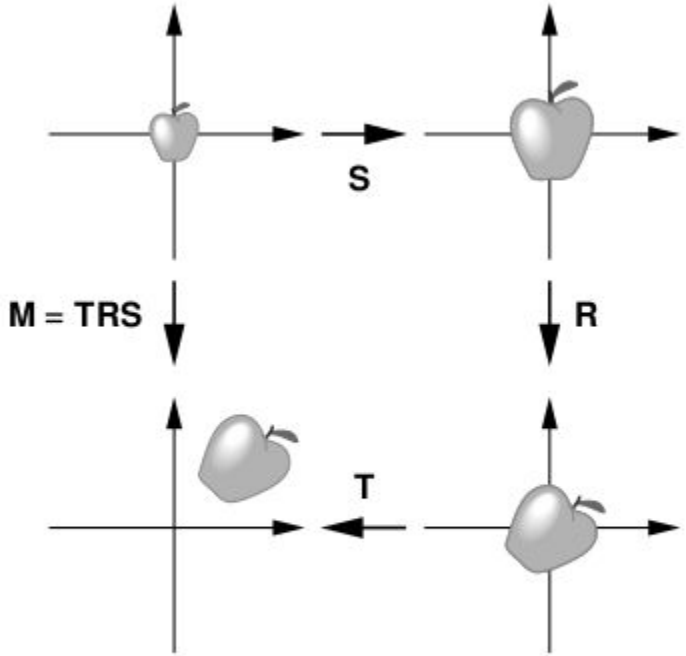
Data structure:

```
typedef struct treenode {
        GLfloat m[16]; // transformation
        void (*f)();  // figure
        struct treenode *sibling;
        struct treenode *child;
} treenode;
```

traverse:

```
glMultMatrixf(root->m);
root->f();
if(root->child!=NULL) traverse(root->child);
if(root->sibling!=NULL) traverse(root->sibling);
```

Hierarchical & Scene Graphs



$$\mathbf{T}ranslate * \mathbf{R}otate * \mathbf{S}cale * object$$

Two ways to view transformations.

- **object-centric**:
Define the object in its own model coordinate system, apply S, R, T in order.
That is, read T RS(v) from right to left and the code from bottom (=glVertex) up.

- **finite-state machine**:
Modify the ModelView matrix, i.e. form T RS then apply to the object v.
That is, read T RS(v) from left to right and the code from top (=glLoadIdentity) down.

The ordering of operations is important!