Hierarchical Modeling

Spring 2016

Seungyong Lee
POSTECH

Overview

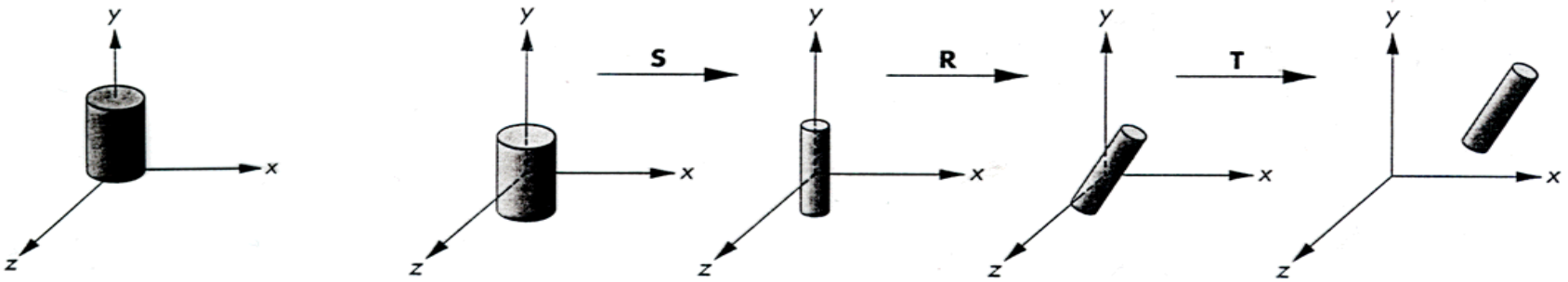
- Hierarchical model
  - model consisting of several components
  - how to represent, draw, and animate?
- Scene graph
  - totality of objects and relations that describes a scene
- Chapter summary
  - hierarchical models
  - modeling transformations
  - transformations of a hierarchical model
  - scene graphs

Overview (2)

- Related materials
  - Angel: Sections 8.1 - 8.9
  - H&B: Chapter 7

Hierarchical Models

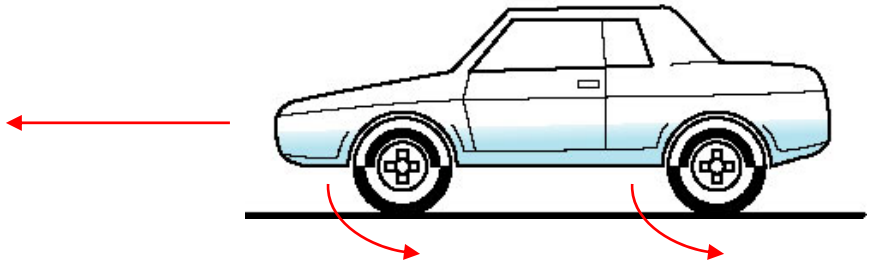
- Models and instances
  - model frame (modeling coordinates)
  - instance transformation (instancing in world coordinates)



The University of New Mexico

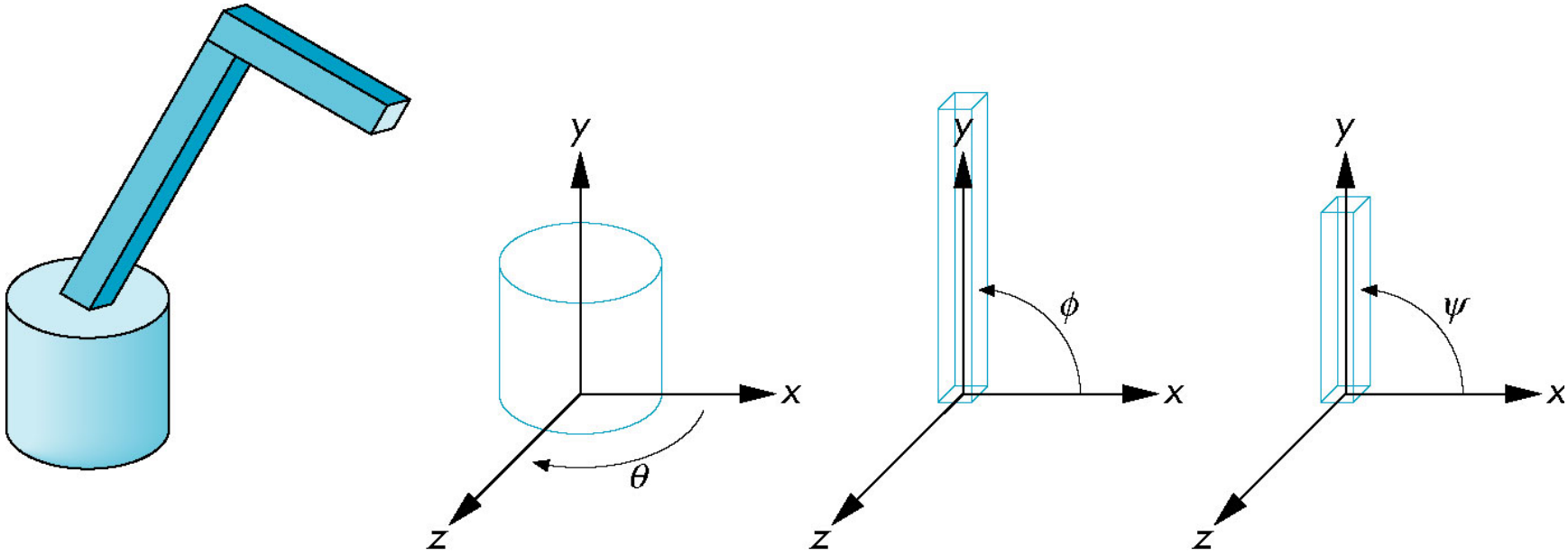
Car Model

- Consider a model of a car
  - Chassis + 4 identical wheels
  - Two models
  - Five instances (1 + 4 instances)
  - Different transformation for each instance



The University of New Mexico

Robot Arm

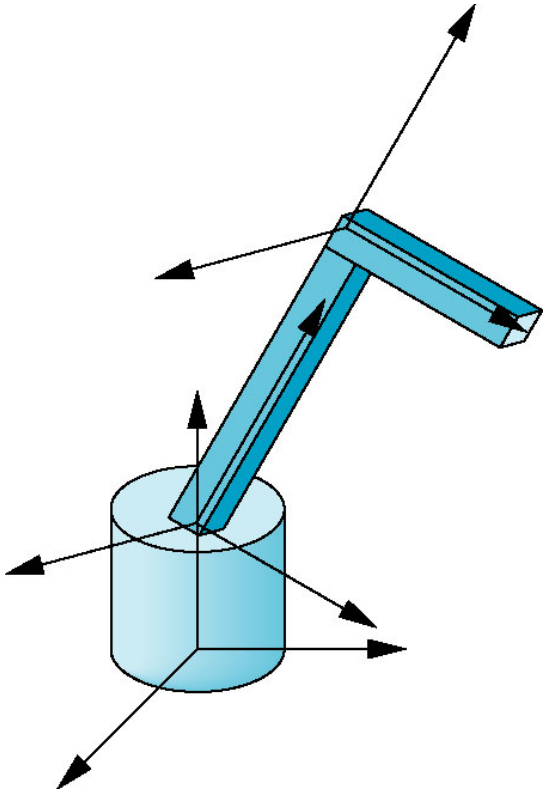


robot arm

parts in their own coodinate systems

Articulated Models

- Robot arm is an example of an *articulated model*
  - Parts connected at joints
  - Can specify state of model by giving all joint angles



Relationships in Robot Arm

- Base rotates independently
  - Single angle determines position
- Lower arm attached to base
  - Its position depends on rotation of base
  - Must also translate relative to base and rotate about connecting joint
- Upper arm attached to lower arm
  - Its position depends on both base and lower arm
  - Must translate relative to lower arm and rotate about joint connecting to lower arm

Required Matrices

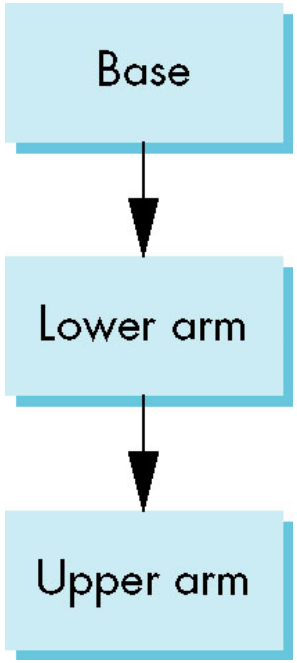
- Rotation of base: $\mathbf{R}_b$
  - Apply $\mathbf{M} = \mathbf{R}_b$ to base
- Translate lower arm relative to base: $\mathbf{T}_{lu}$
- Rotate lower arm around joint: $\mathbf{R}_{lu}$
  - Apply $\mathbf{M} = \mathbf{R}_b \mathbf{T}_{lu} \mathbf{R}_{lu}$ to lower arm
- Translate upper arm relative to upper arm: $\mathbf{T}_{uu}$
- Rotate upper arm around joint: $\mathbf{R}_{uu}$
  - Apply $\mathbf{M} = \mathbf{R}_b \mathbf{T}_{lu} \mathbf{R}_{lu} \mathbf{T}_{uu} \mathbf{R}_{uu}$ to upper arm

OpenGL Code for Robot

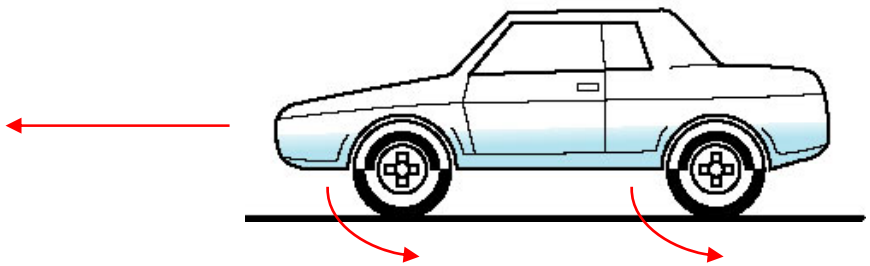
```
mat4 ctm;
robot_arm()
{
    ctm = RotateY(theta);
    base();
    ctm *= Translate(0.0, h1, 0.0);
    ctm *= RotateZ(phi);
    lower_arm();
    ctm *= Translate(0.0, h2, 0.0);
    ctm *= RotateZ(psi);
    upper_arm();
}
```

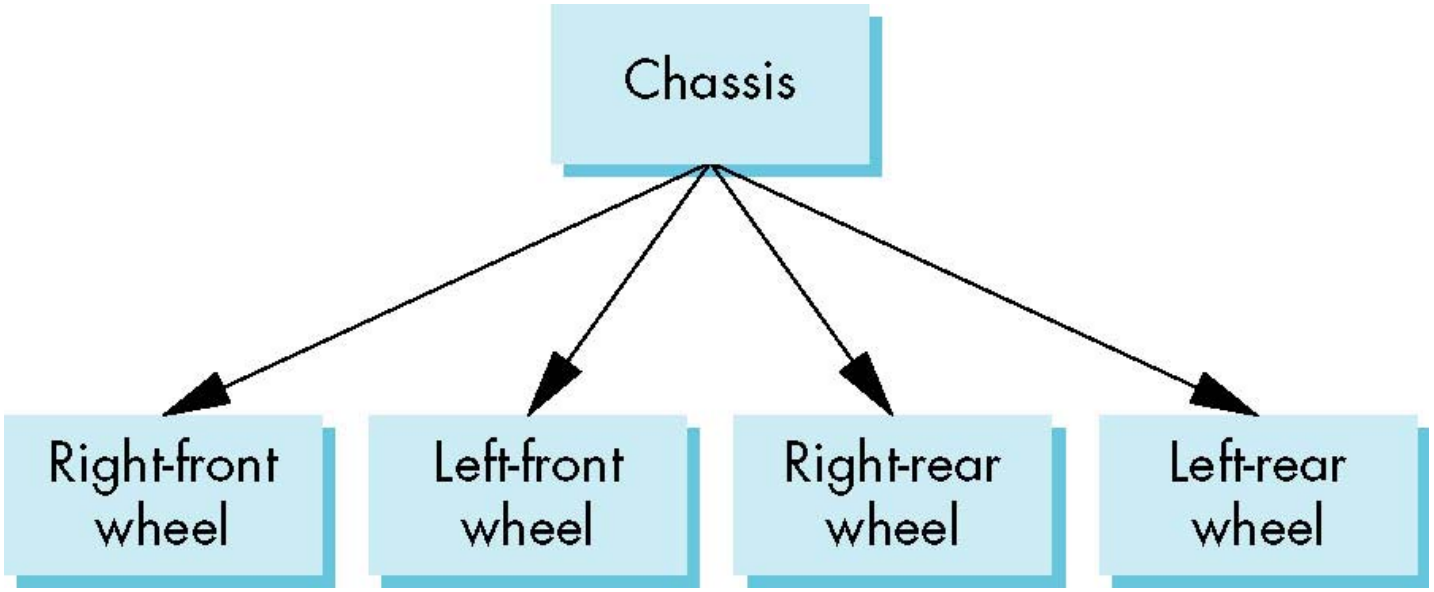
Tree Model of Robot

- OpenGL code shows relationships between parts of model
  - Can change “look” of parts easily without altering relationships
- A tree model



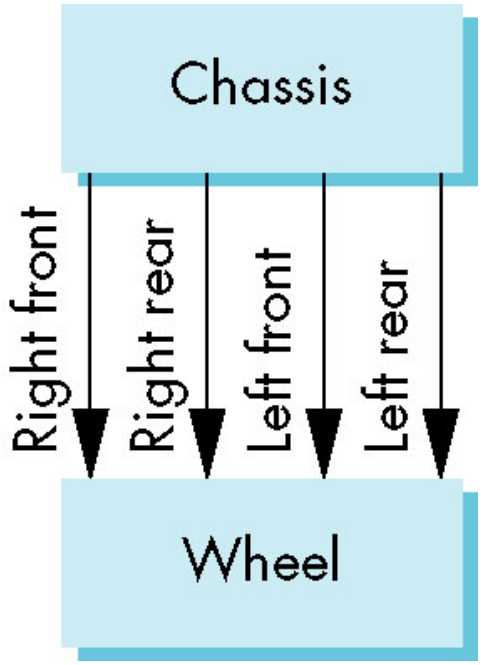
Tree Model of Car





DAG Model of Car

- If we use the fact that all the wheels are identical, we get a *directed acyclic graph*
  - Not much different than dealing with a tree

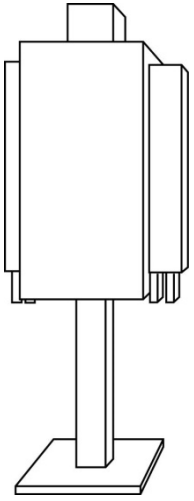


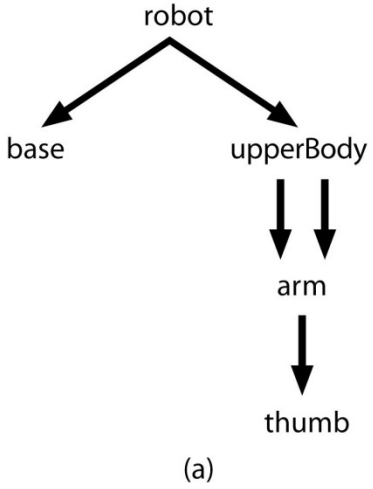
Modeling with Trees

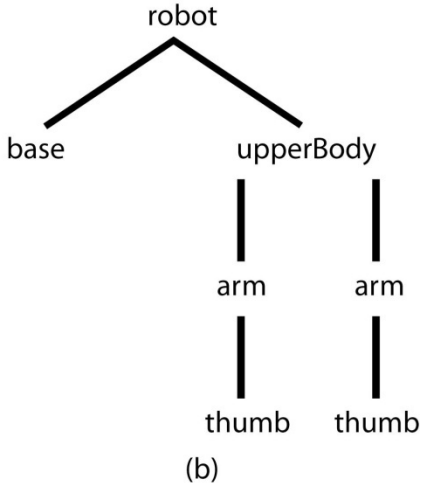
- Must decide what information to place in nodes and what to put in edges
- Nodes
  - What to draw
  - Pointers to children
- Edges
  - May have information on incremental changes to transformation matrices (can also store in nodes)

Hierarchical Models

- Hierarchical models
  - components are used as building blocks to create higher-level entities
  - directed acyclic graph (DAG)
  - parameter passing (transformations) in object hierarchy







Hierarchical Models

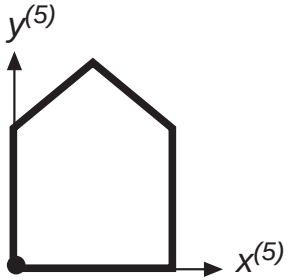
- Need to deal with multiple children
  - How do we represent a more general tree?
  - How do we traverse such a data structure?
- Animation
  - How to animate a hierarchical model?

Modeling Transformation

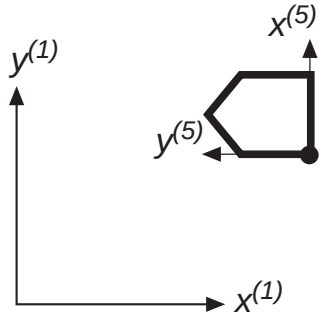
- Transformation from model to world coordinates
  - modeling transformation matrix
  - transformation for a single object instance
    - composition of basic transformations
  - transformation for a component object in the hierarchy
    - concatenation of local transformations
    - the end effect is cumulative
  - coordinate system transformation from the model to the world

Modeling Transformation (2)

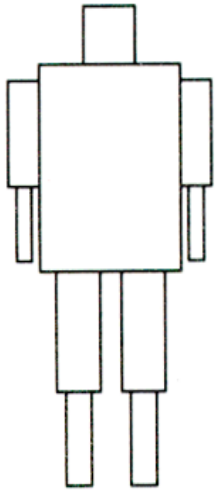
- Example

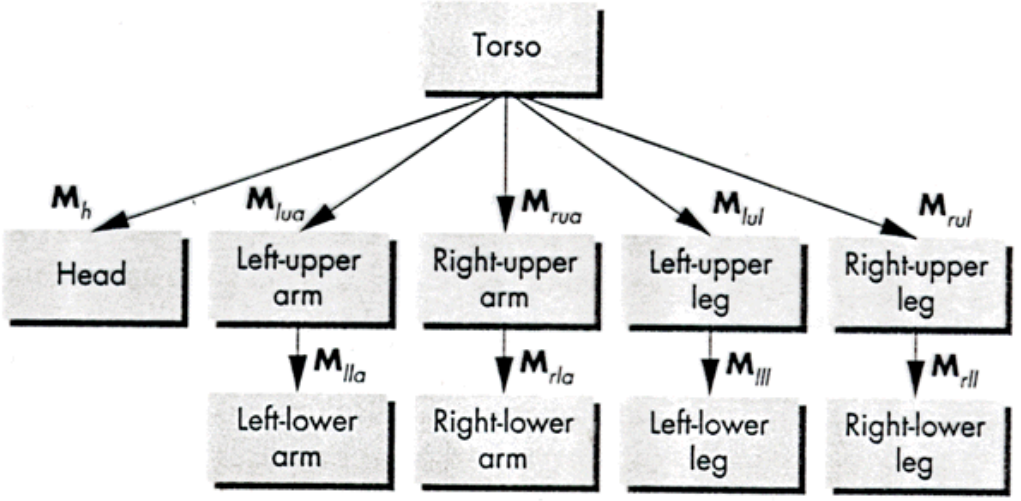


(a)



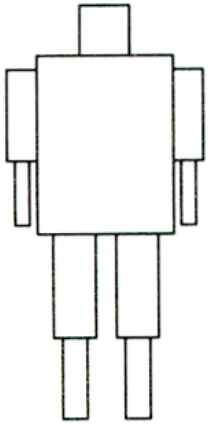
(b)



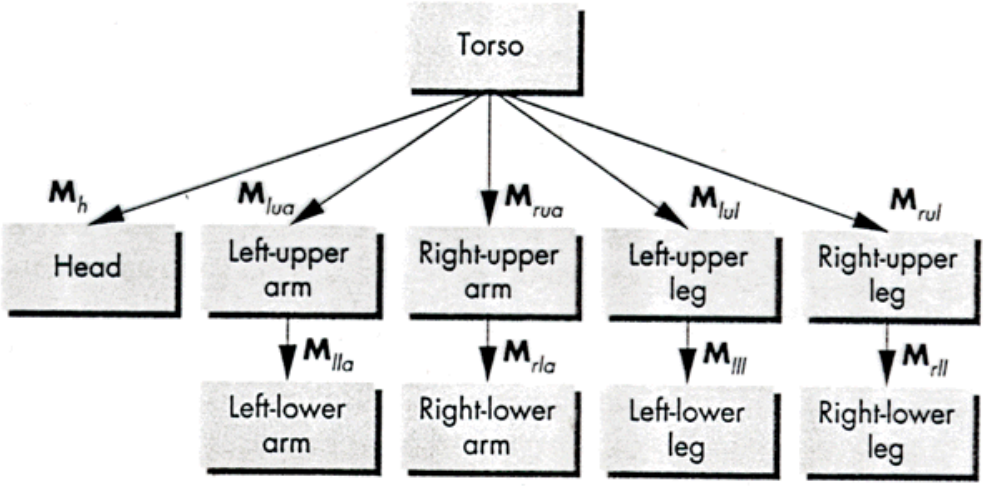


Stack-based Traversal

- How to implement in OpenGL?
  - model-view matrix stack
  - glPushMatrix();
  - glPopMatrix();
- Sample program

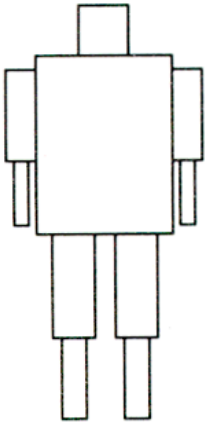


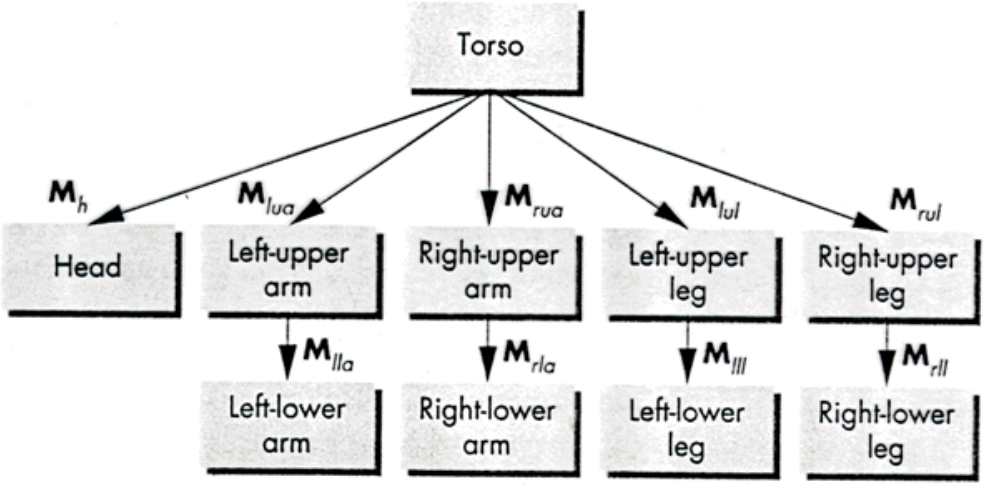
```
figure()
{
        torso();
        glPushMatrix();
        glTranslate
        glRotate3
        head();
```



Stack-based Traversal (2)

```
glPopMatrix();
glPushMatrix();
glTranslate
glRotate3
left_upper_arm();
glPushMatrix();
glTranslate
glRotate3
left_lower_arm();
glPopMatrix();
glPopMatrix();
glPushMatrix();
glTranslate
glRotate3
right_upper_arm();
...
}
```





Traversal Code

```
figure() {
    PushMatrix()
    torso();
    Rotate (…);
    head();
    PopMatrix();
    PushMatrix();
    Translate(…);
    Rotate(…);
    left_upper_arm();
    PopMatrix();
    PushMatrix();
```

save present model-view matrix

update model-view matrix for head

recover original model-view matrix

save it again

update model-view matrix for left upper arm

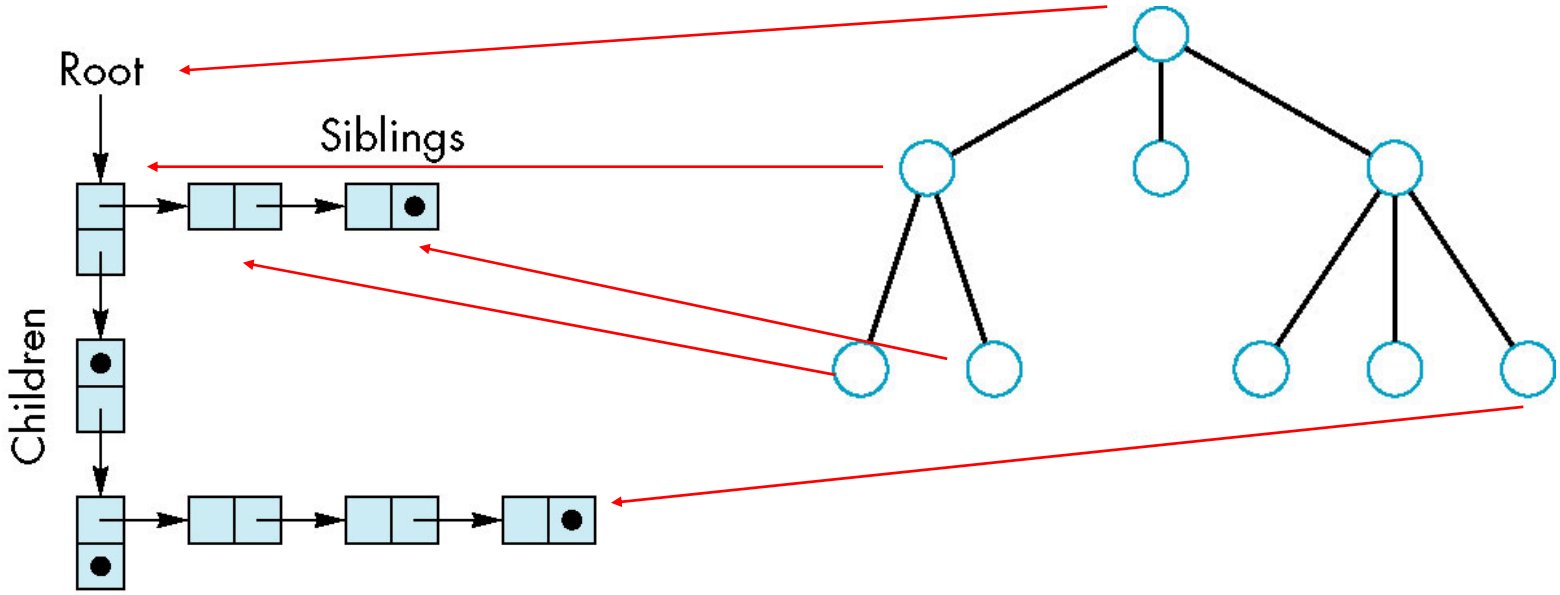
recover and save original model-view matrix again

rest of code

General Tree Data Structure

- Need a data structure to represent tree and an algorithm to traverse the tree
- We will use a *left-child right sibling* structure
  - Uses linked lists
  - Each node in data structure is two pointers
  - Left: next node
  - Right: linked list of children

Left-Child Right-Sibling Tree



Tree node Structure

- At each node we need to store
  - Pointer to sibling
  - Pointer to child
  - Pointer to a function that draws the object represented by the node
  - Homogeneous coordinate matrix to multiply on the right of the current model-view matrix
    - Represents changes going from parent to node
    - In OpenGL this matrix is a 1D array storing matrix by columns

C Definition of treenode

```
typedef struct treenode
{
   mat4 m;
   void (*f)();
   struct treenode *sibling;
   struct treenode *child;
} treenode;
```

torso and head nodes

```
treenode torso_node, head_node, lua_node, … ;

torso_node.m = RotateY(theta[0]);
torso_node.f = torso;
torso_node.sibling = NULL;
torso_node.child =  &head_node;

head_node.m = translate(0.0,
     TORSO_HEIGHT+0.5*HEAD_HEIGHT, 0.0)
     * RotateX(theta[1])*RotateY(theta[2]);
head_node.f = head;
head_node.sibling = &lua_node;
head_node.child = NULL;
```

Preorder Traversal

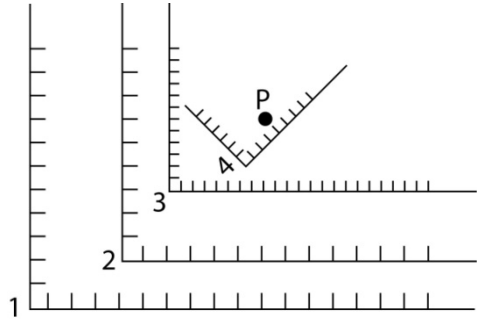
```
void traverse(treenode* root)
{
   if(root==NULL) return;
   mvstack.push(model_view);
   model_view = model_view*root->m;
   root->f();
   if(root->child!=NULL)
     traverse(root->child);
   model_view = mvstack.pop();
   if(root->sibling!=NULL)
     traverse(root->sibling);
}
```

Transformations of a Hierarchical Model

- Geometric transformation of a single object
  - model definition in the modeling coordinates
  - motion in the world is defined by a transformation
  - change of modeling transformation
- Transformation of an object in a hierarchy
  - change of local transformation from the object coordinates to its parent coordinate system
  - effects on the descendent nodes as well
  - the end effect is cumulative

Transformations of a Hierarchical Model (2)

- Transformation from coordinate system j to i: $M_{i \leftarrow j}$
  - $P^{(i)} = M_{i \leftarrow j} P^{(j)}$
  - $M_{i \leftarrow k} = M_{i \leftarrow j} M_{j \leftarrow k}$
  - $M_{i \leftarrow j} = M_{j \leftarrow i}^{-1}$



Transformations of a Hierarchical Model (3)

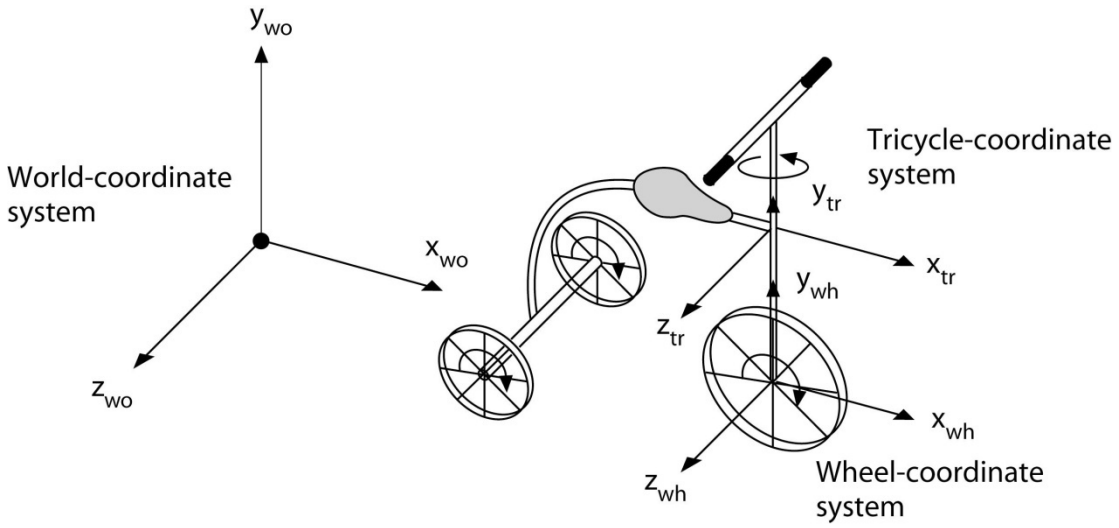
- Single object example (rotating wheel)
  - a wheel of a tricyle is in its coordinate system
  - the wheel is rotating in the world coordinate system

$$P^{(wo)} = M_{wo \leftarrow wh} P^{(wh)}$$

$$P'^{(wh)} = Q^{(wh)} P^{(wh)} = T(\alpha r, 0,0) R_z(\alpha) P^{(wh)}$$

$$P'^{(wo)} = M'_{wo \leftarrow wh} P^{(wh)} = M_{wo \leftarrow wh} P'^{(wh)} = M_{wo \leftarrow wh} Q^{(wh)} P^{(wh)}$$

$$M'_{wo \leftarrow wh} = M_{wo \leftarrow wh} Q^{(wh)}$$



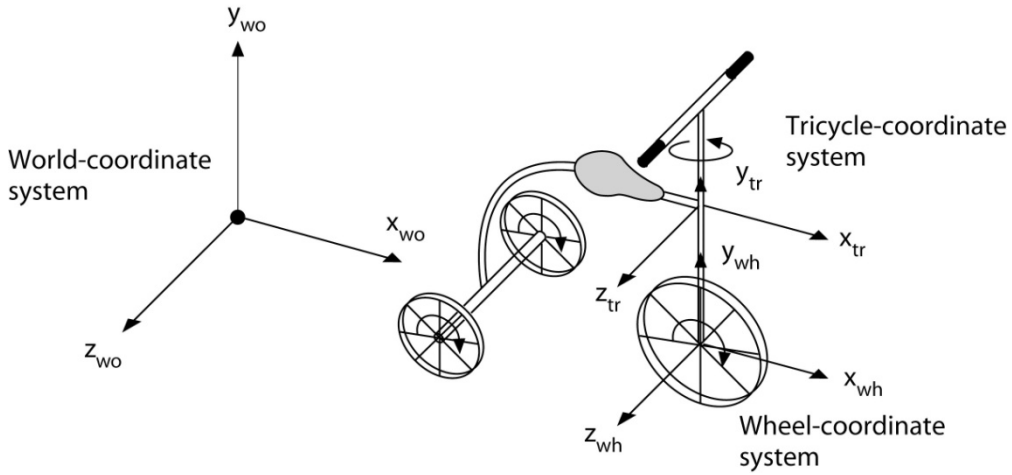
Transformations of a Hierarchical Model (4)

- Hierarchical example
  - drawing a moving tricycle with rotating wheels?
  - separate modeling of the tricycle body and a wheel
  - hierarchical coordinate system transformations

$$\boldsymbol{P}^{(wo)} = \boldsymbol{M}_{wo \leftarrow wh} \boldsymbol{P}^{(wh)} = \boldsymbol{M}_{wo \leftarrow tr} \boldsymbol{M}_{tr \leftarrow wh} \boldsymbol{P}^{(wh)}$$

$$\boldsymbol{P}'^{(wo)} = \boldsymbol{M}'_{wo \leftarrow wh} \boldsymbol{P}^{(wh)} = \boldsymbol{M}'_{wo \leftarrow tr} \boldsymbol{M}'_{tr \leftarrow wh} \boldsymbol{P}^{(wh)}$$

- What about multiple tricycles and a tricycle pushing another one?



Scene Graphs

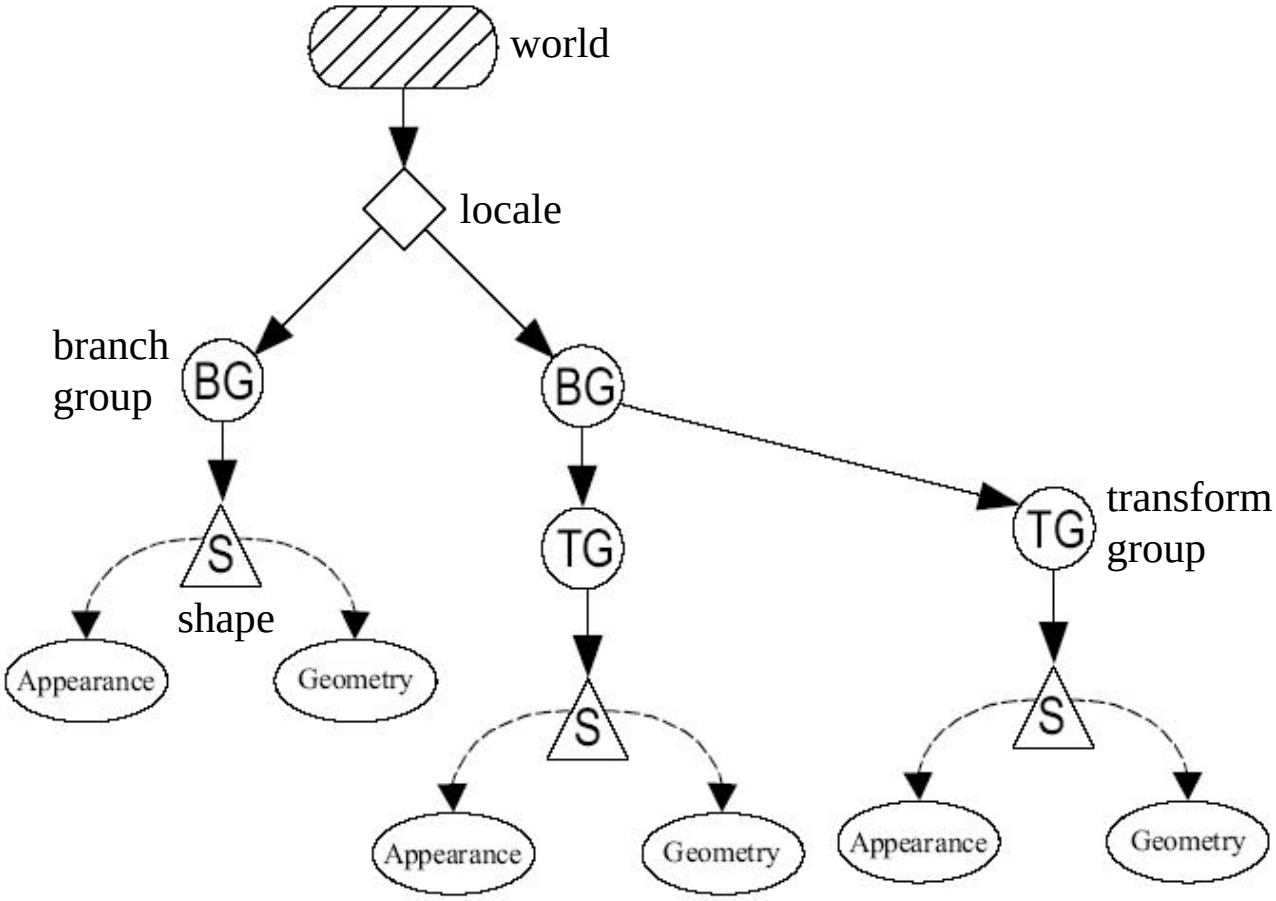
- What is a scene graph?
  - totality of objects and relations that describes a scene
  - traversed to generate an image or animation with proper rendering routines
- Nodes
  - geometric objects
    - texts, polygons, curves and surfaces, solids, …
  - cameras, lights
  - shapes, attributes, textures, binding
  - transformations, manipulators
  - time

Scene Graphs (2)

- Graph (tree) structure
  - relations between nodes
  - group information
- Action
  - graph traversal
    - rendering, bounding box computation, …
  - picking, draggers, manipulators
  - animation
- Scene graph database
  - Open Inventor, Java3D, and JSR-184
  - not supported in OpenGL

Scene Graphs (3)

- Java3D scene graph example



Summary

- Hierarchical models
- Modeling transformations
- Transformations of a hierarchical model
- Scene graphs
  - Inventor, Java3D
  - VRML
  - Open Scene Graph