8. Hierarchical Modeling

Reading

Recommended:

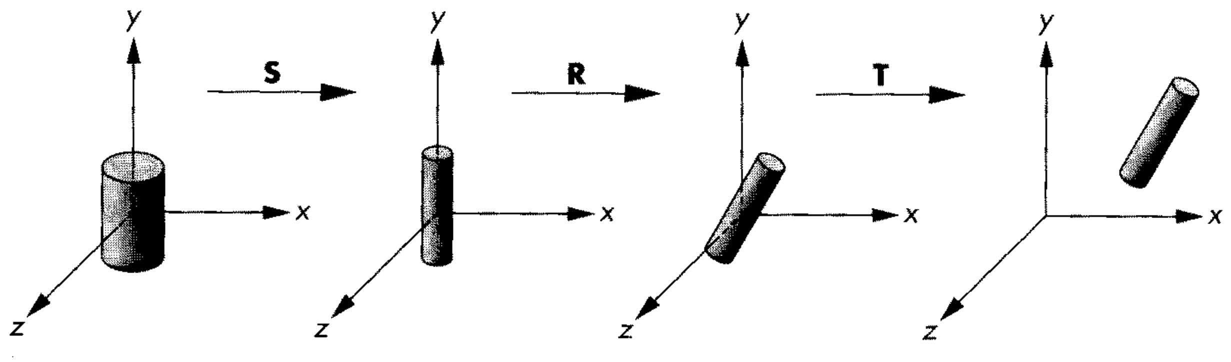
- Angel, Sections 8.1–8.6

Symbols and instances

Most graphics API's support a few geometric "primitives" — e.g.:

- spheres
- cubes
- cylinders

These "symbols" (or "masters") are "instanced" using an "instance transformation":



Instance transformation (Angel, fig 8.2)

Q: What is the matrix for the instance transformation above?

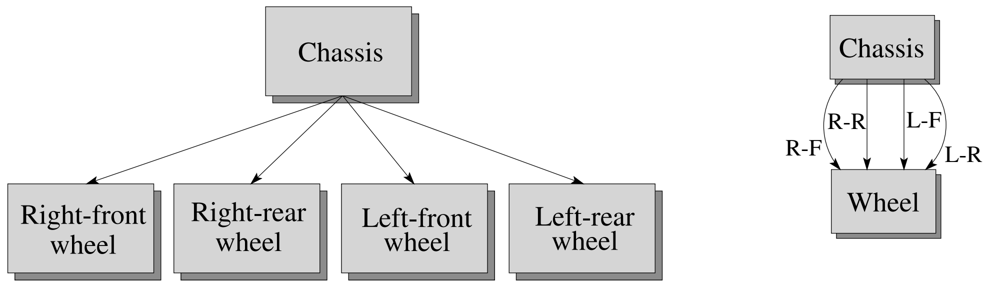
Instancing in OpenGL

In OpenGL, instancing is created by modifying the "model-view matrix":

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef( ... );
glRotatef( ... );
glScalef( ... );
cylinder();
```

Hierarchical modeling

Hierarchical models can be composed of instances using trees or DAGS:



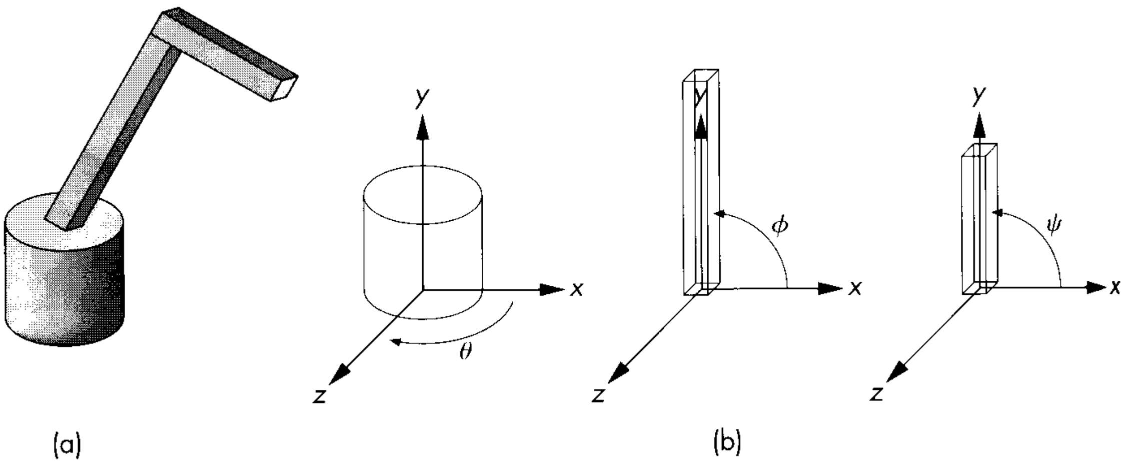
Two representations of a simple car

- Edges contain geometric transformations
- Nodes contain geometry (and possibly drawing attributes)

Example: A robot arm

Consider this robot arm with 3 degrees of freedom:

1. Base rotates about its vertical axis by θ
2. Lower arm rotates in its xy-plane by ϕ
3. Upper arm rotates in its xy-plane by ψ



A robot arm (Angel, fig. 8.8)

Q: What is the tree structure of the robot?

Q: What matrix do we use to transform the base?

Q: What matrix for the lower arm?

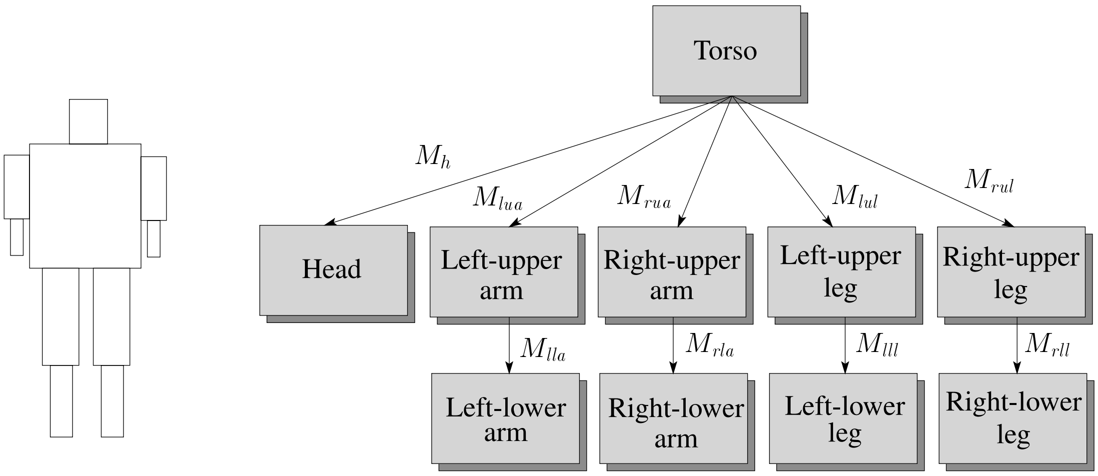
Q: What matrix for the upper arm?

Robot arm implementation

The robot arm can be displayed by altering the model-view matrix incrementally:

```
display()
{
   glRotatef(theta, 0., 1., 0.);
   base();
   glTranslatef(0., h1, 0.);
   glRotatef(phi, 0., 0., 1.);
   lower_arm();
   glTranslatef(0., h2, 0.);
   glRotatef(psi, 0., 0., 1.);
   upper_arm();
}
```

A more complex example: Human figure



Human figure

Q: What's the most sensible way to traverse this tree?

Human figure implementation

The traversal can be implemented by pushing the model-view matrix onto a stack:

```
figure()
{
   torso();
   glPushMatrix();
      glTranslate
      glRotate3
      head();
   glPopMatrix();
   glPushMatrix();
      glTranslate
      glRotate3
      left_upper_leg();
      glTranslate
      glRotate3
      left_lower_leg();
   glPopMatrix();
   glPushMatrix();
      .
      .
}
```

Animation

The above examples are called "articulated models":

- rigid parts
- connected by joints

They can be animated by specifying the joint angles (or other display parameters) as functions of time.

(Note that each one can be changed independently.)

Kinematics and dynamics

Definitions:

- "Kinematics": How the positions of the parts vary as a function of joint angles.

- "Dynamics": How the positions of the parts vary as a function of applied forces.

Questions:

Q: What do the terms "inverse kinematics" and "inverse dynamics" mean?

Q: Why are these problems more difficult?

Key-frame animation

One way to get around these problems is to use "key-frame animation":

- Each joint specified at various "key frames" (not necessarily the same as other joints)
- System does the interpolation or "inbetweening"

Doing this well requires:

- A way of smoothly interpolating key frames: "splines"
- A good interactive system
- A lot of skill on the part of the animator!

Scene graphs

The idea of hierarchical modeling can be extended to an entire scene, encompassing:

- many different objects
- lights
- camera position

Called a "scene tree" or a "scene graph."

Summary

Here's what you should take away from this lecture:

1. All the terms in quotations.
2. How primitives can be instanced and composed to create hierarchical models using geometric transformations.
3. How this notion can be extended to entire scenes.
4. How keyframe animation works.