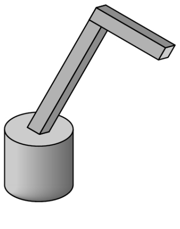
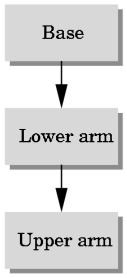
Hierarchy Tutorial

Hierarchical Models

- In many applications, the parts of a model depend on one another
    - If we move one part, it causes other parts to move
- Parts of such models can be arranged as a tree data structure e.g. a simple robot arm

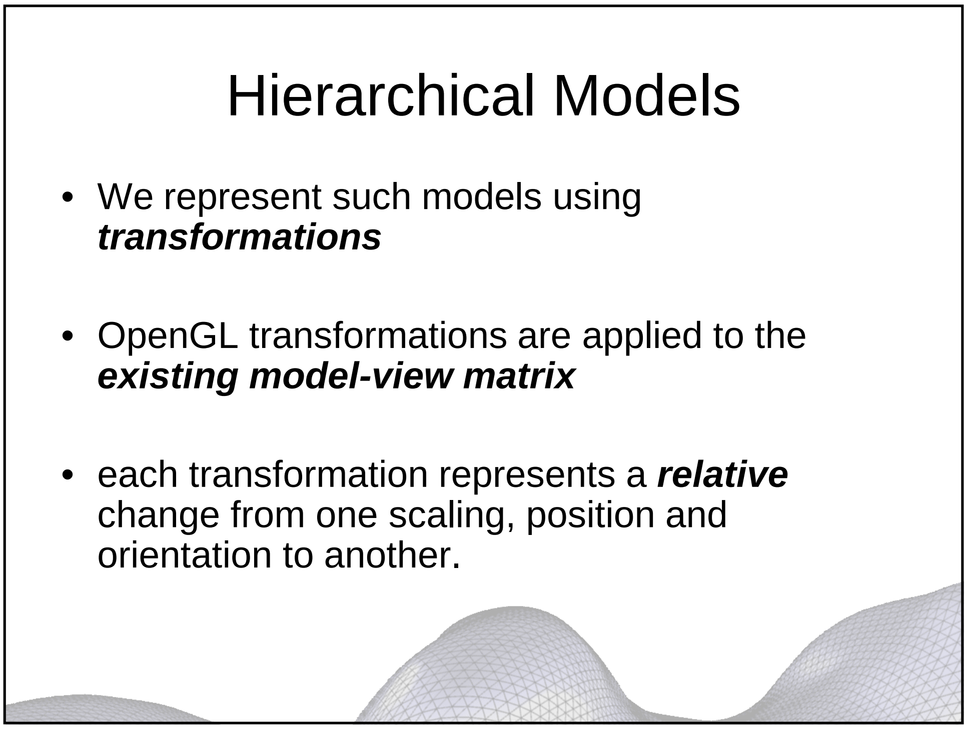




Base

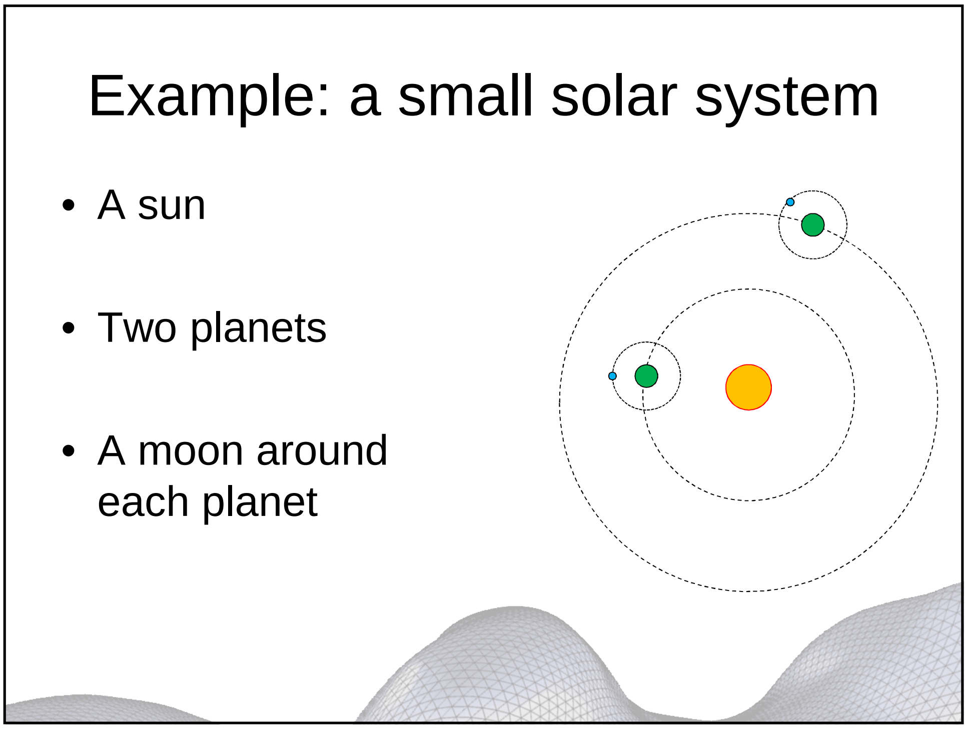
Lower arm

Upper arm



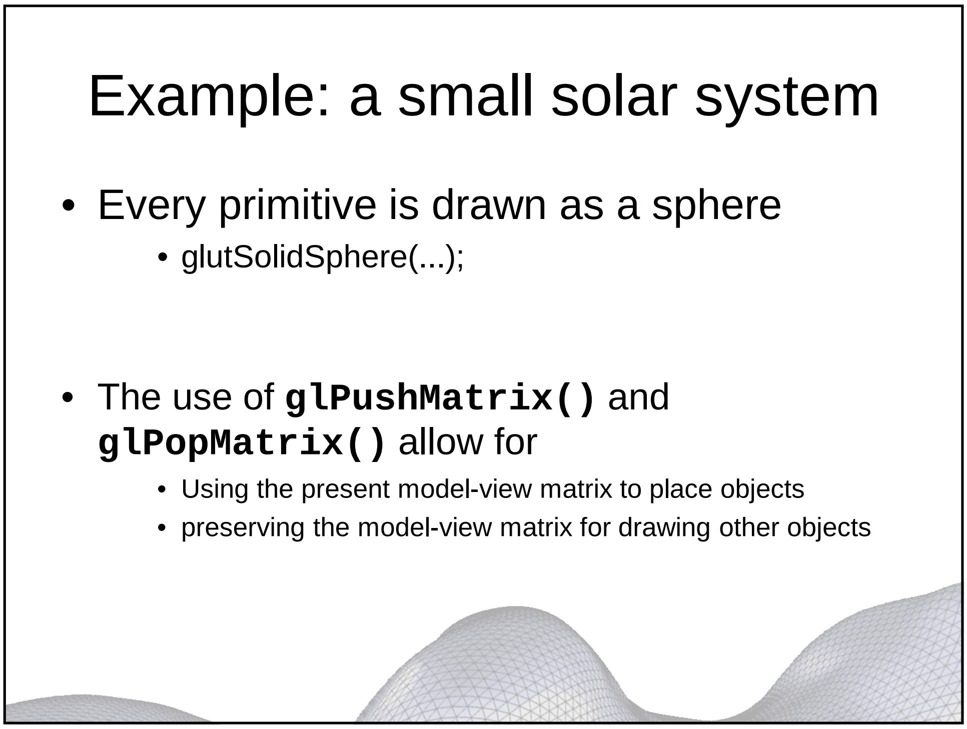
Hierarchical Models

- We represent such models using ***transformations***
- OpenGL transformations are applied to the ***existing model-view matrix***
- each transformation represents a ***relative*** change from one scaling, position and orientation to another.



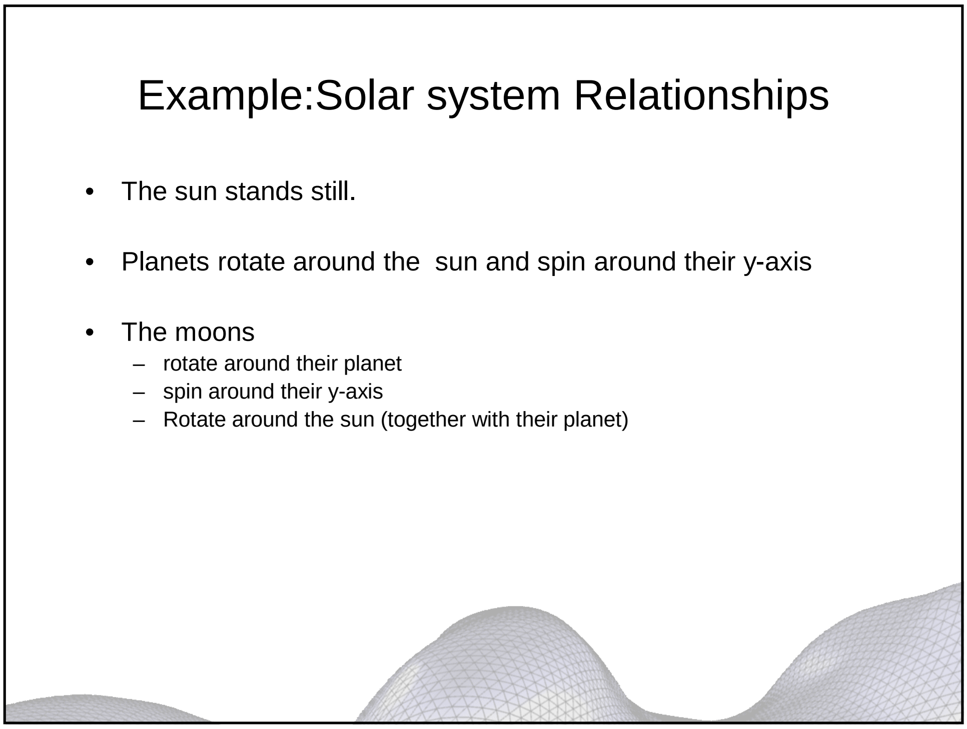
Example: a small solar system

- A sun
- Two planets
- A moon around each planet



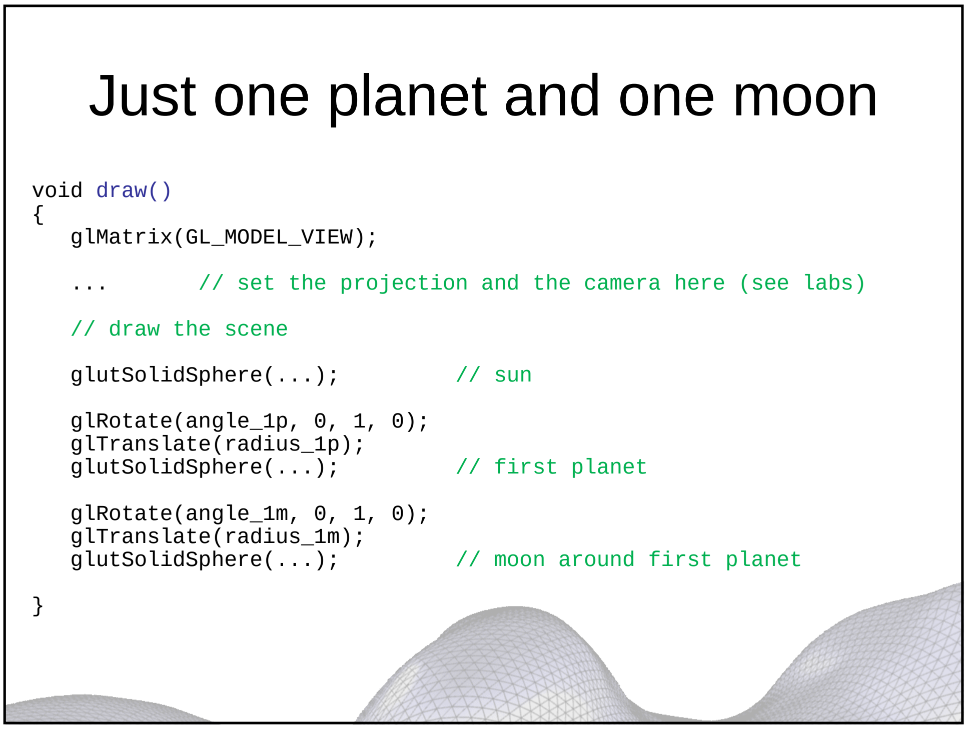
Example: a small solar system

- Every primitive is drawn as a sphere
  - glutSolidSphere(...);
- The use of **`glPushMatrix()`** and **`glPopMatrix()`** allow for
  - Using the present model-view matrix to place objects
  - preserving the model-view matrix for drawing other objects



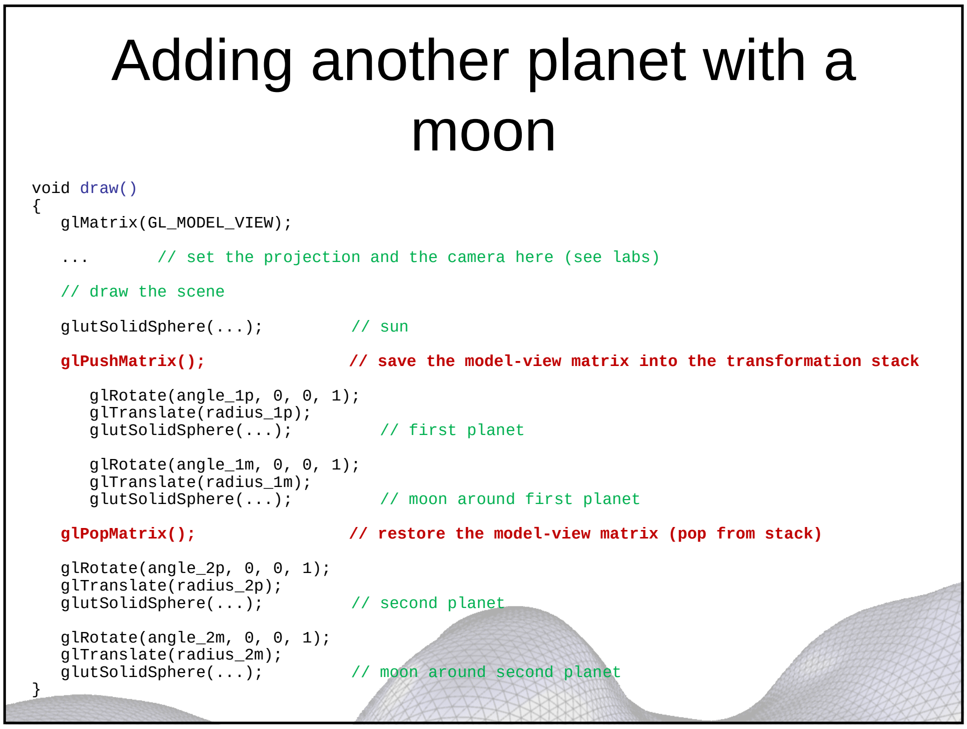
Example:Solar system Relationships

- The sun stands still.
- Planets rotate around the sun and spin around their y-axis
- The moons
  - rotate around their planet
  - spin around their y-axis
  - Rotate around the sun (together with their planet)



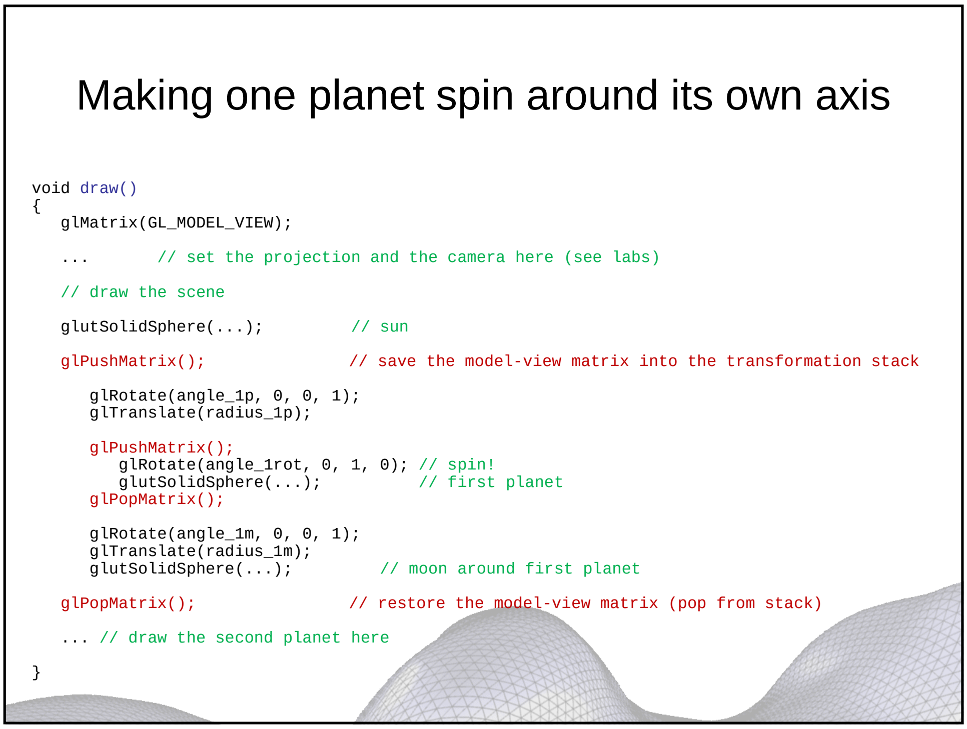
Just one planet and one moon

```
void draw()
{
   glMatrix(GL_MODEL_VIEW);

   ...       // set the projection and the camera here (see labs)

   // draw the scene

   glutSolidSphere(...);         // sun

   glRotate(angle_1p, 0, 1, 0);
   glTranslate(radius_1p);
   glutSolidSphere(...);         // first planet

   glRotate(angle_1m, 0, 1, 0);
   glTranslate(radius_1m);
   glutSolidSphere(...);         // moon around first planet

}
```



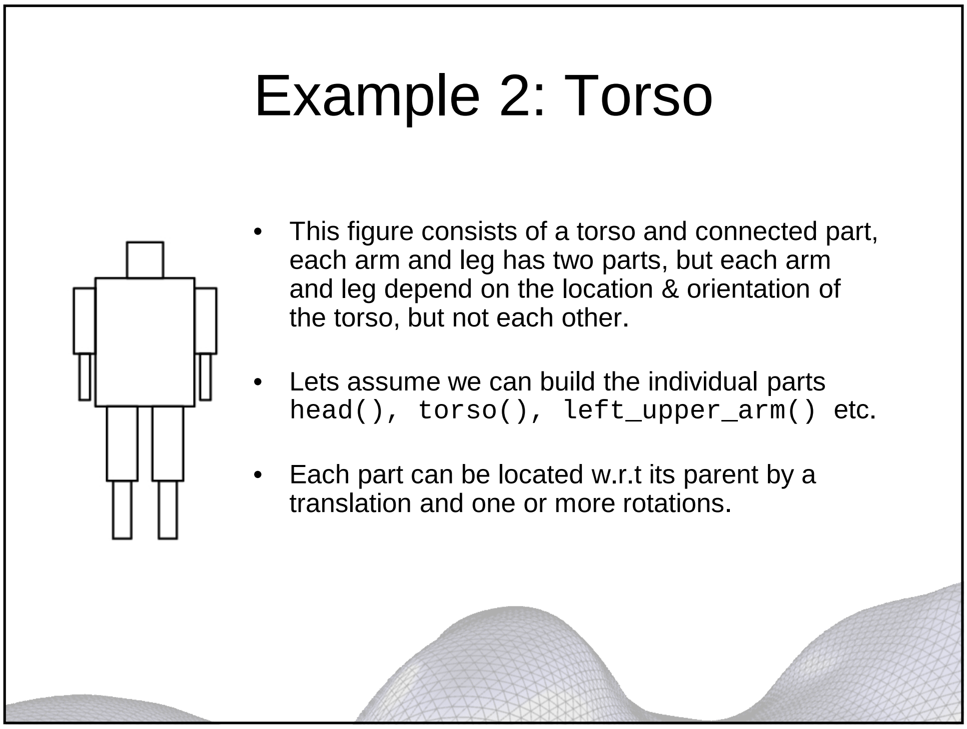
Adding another planet with a moon

```
void draw()
{
   glMatrix(GL_MODEL_VIEW);

   ...       // set the projection and the camera here (see labs)

   // draw the scene

   glutSolidSphere(...);         // sun

   glPushMatrix();               // save the model-view matrix into the transformation stack

      glRotate(angle_1p, 0, 0, 1);
      glTranslate(radius_1p);
      glutSolidSphere(...);         // first planet

      glRotate(angle_1m, 0, 0, 1);
      glTranslate(radius_1m);
      glutSolidSphere(...);         // moon around first planet

   glPopMatrix();                // restore the model-view matrix (pop from stack)

   glRotate(angle_2p, 0, 0, 1);
   glTranslate(radius_2p);
   glutSolidSphere(...);         // second planet

   glRotate(angle_2m, 0, 0, 1);
   glTranslate(radius_2m);
   glutSolidSphere(...);         // moon around second planet
}
```



Making one planet spin around its own axis

```
void draw()
{
   glMatrix(GL_MODEL_VIEW);

   ...       // set the projection and the camera here (see labs)

   // draw the scene

   glutSolidSphere(...);         // sun

   glPushMatrix();               // save the model-view matrix into the transformation stack

      glRotate(angle_1p, 0, 0, 1);
      glTranslate(radius_1p);

      glPushMatrix();
         glRotate(angle_1rot, 0, 1, 0); // spin!
         glutSolidSphere(...);          // first planet
      glPopMatrix();

      glRotate(angle_1m, 0, 0, 1);
      glTranslate(radius_1m);
      glutSolidSphere(...);         // moon around first planet

   glPopMatrix();                // restore the model-view matrix (pop from stack)

   ... // draw the second planet here

}
```



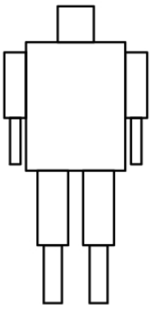
Example 2: Torso

- This figure consists of a torso and connected part, each arm and leg has two parts, but each arm and leg depend on the location & orientation of the torso, but not each other.

- Lets assume we can build the individual parts `head(), torso(), left_upper_arm()` etc.

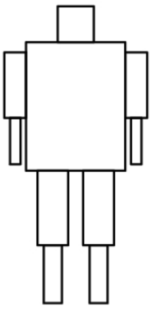
- Each part can be located w.r.t its parent by a translation and one or more rotations.

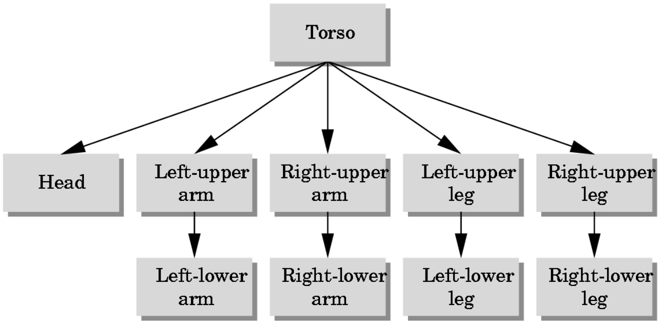
Example 2: Torso



- The display callback must traverse the tree i.e. visit every node, drawing the object for that node, using the correct model-view matrix
- A standard pre-order traversal (that travels down the left of the tree, visiting each node) is used

Example 2: Torso





Torso

Head

Left-upper arm

Right-upper arm

Left-upper leg

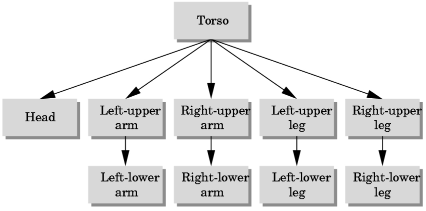
Right-upper leg

Left-lower arm

Right-lower arm

Left-lower leg

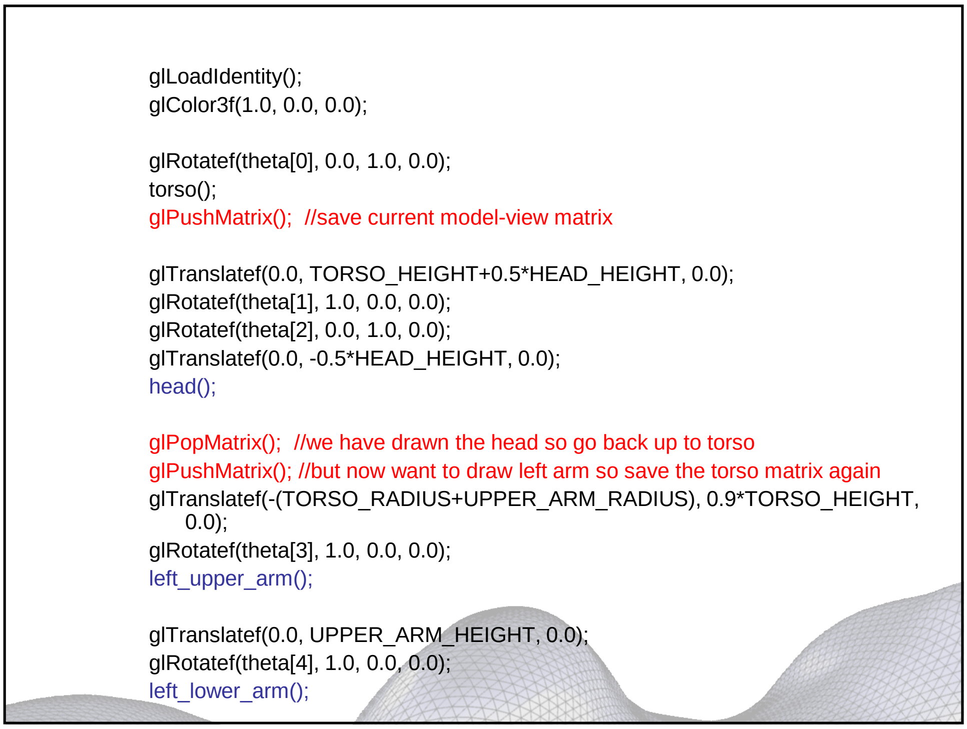
Right-lower leg



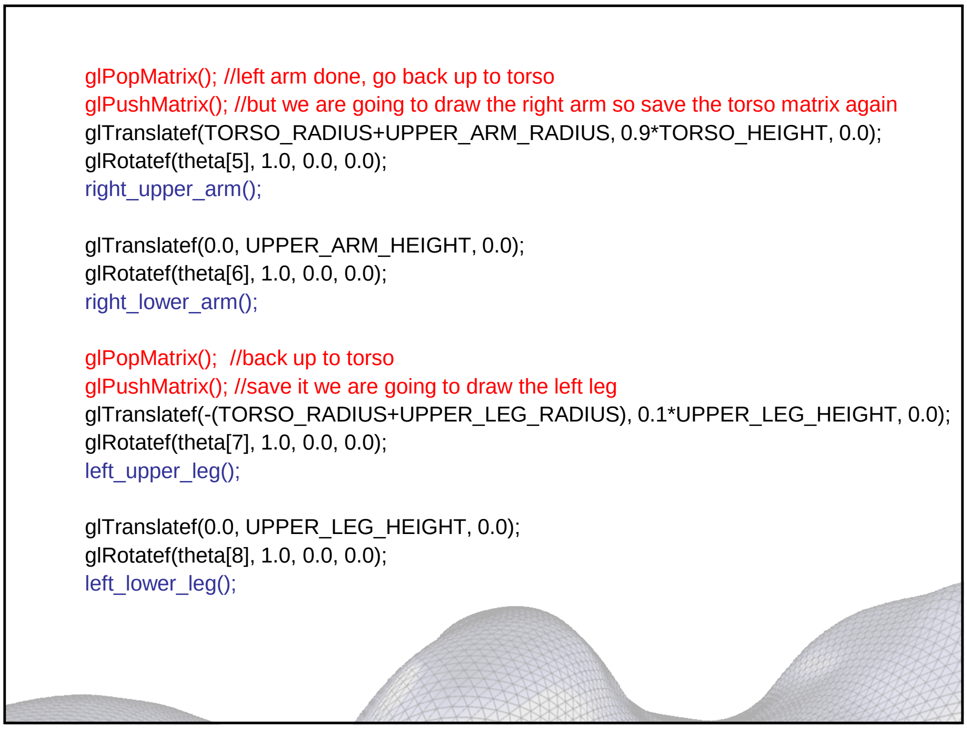
- First draw torso. It only has one angle associated with it that allows it to rotate about y.
- Then we go to the head, however note we have to come back up to the torso to get to the arms and legs
- Any matrix that we apply to draw the head is not required for the arms or legs.
- Rather than recompute the matrix that we apply to the torso node we can save it on the stack with a glPushMatrix().
- We can then go to the node for the head, changing the model-view matrix as necessary to draw the head.
- When we come back up to the torso node, we recover the model-view matrix with a glPopMatrix()
- We have to come back up the the torso after dealing with the left arm so we must go to a glPushMatrix() immediately after the pop to keep a copy of the same model-view matrix

Simple!

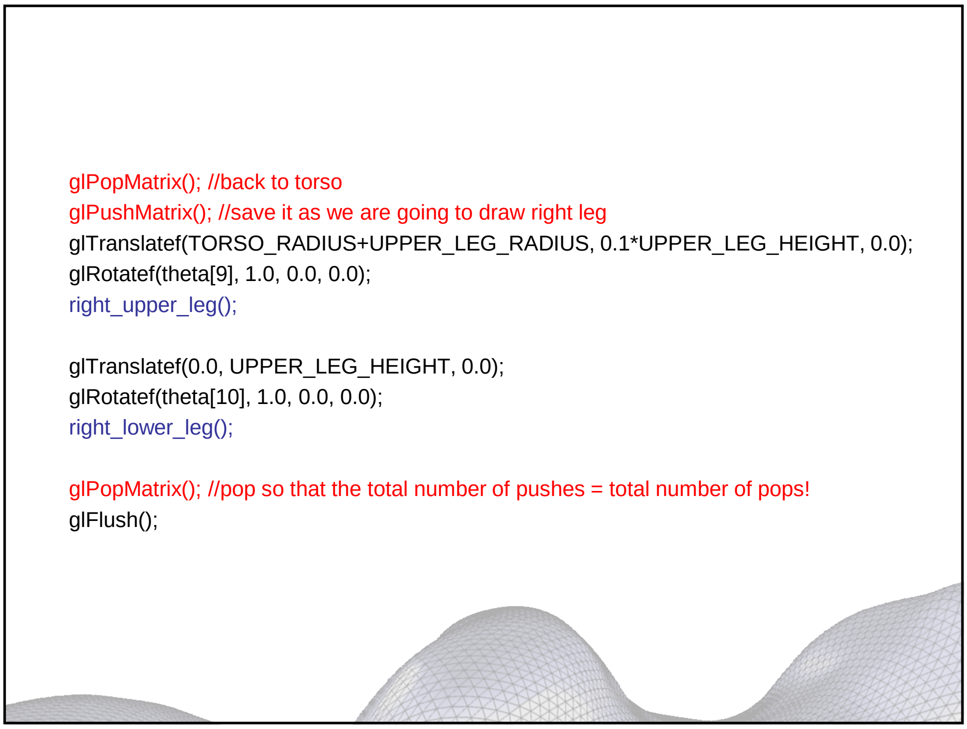
- Although it appears convoluting, the rule is simple – every time we go to the left at a node with another unvisited right child we do a push; everytime we return to the node we do a pop.
- Note we must do a pop at the end so the total number of pushes and pops is the same



```
glLoadIdentity();
glColor3f(1.0, 0.0, 0.0);

glRotatef(theta[0], 0.0, 1.0, 0.0);
torso();
glPushMatrix();  //save current model-view matrix

glTranslatef(0.0, TORSO_HEIGHT+0.5*HEAD_HEIGHT, 0.0);
glRotatef(theta[1], 1.0, 0.0, 0.0);
glRotatef(theta[2], 0.0, 1.0, 0.0);
glTranslatef(0.0, -0.5*HEAD_HEIGHT, 0.0);
head();

glPopMatrix();  //we have drawn the head so go back up to torso
glPushMatrix(); //but now want to draw left arm so save the torso matrix again
glTranslatef(-(TORSO_RADIUS+UPPER_ARM_RADIUS), 0.9*TORSO_HEIGHT,
    0.0);
glRotatef(theta[3], 1.0, 0.0, 0.0);
left_upper_arm();

glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);
glRotatef(theta[4], 1.0, 0.0, 0.0);
left_lower_arm();
```



```
glPopMatrix(); //left arm done, go back up to torso
glPushMatrix(); //but we are going to draw the right arm so save the torso matrix again
glTranslatef(TORSO_RADIUS+UPPER_ARM_RADIUS, 0.9*TORSO_HEIGHT, 0.0);
glRotatef(theta[5], 1.0, 0.0, 0.0);
right_upper_arm();

glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);
glRotatef(theta[6], 1.0, 0.0, 0.0);
right_lower_arm();

glPopMatrix();  //back up to torso
glPushMatrix(); //save it we are going to draw the left leg
glTranslatef(-(TORSO_RADIUS+UPPER_LEG_RADIUS), 0.1*UPPER_LEG_HEIGHT, 0.0);
glRotatef(theta[7], 1.0, 0.0, 0.0);
left_upper_leg();

glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
glRotatef(theta[8], 1.0, 0.0, 0.0);
left_lower_leg();
```



```
glPopMatrix(); //back to torso
glPushMatrix(); //save it as we are going to draw right leg
glTranslatef(TORSO_RADIUS+UPPER_LEG_RADIUS, 0.1*UPPER_LEG_HEIGHT, 0.0);
glRotatef(theta[9], 1.0, 0.0, 0.0);
right_upper_leg();

glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
glRotatef(theta[10], 1.0, 0.0, 0.0);
right_lower_leg();

glPopMatrix(); //pop so that the total number of pushes = total number of pops!
glFlush();
```