CG Practice 11

COLLEGE OF COMPUTING

HANYANG ERICA CAMPUS

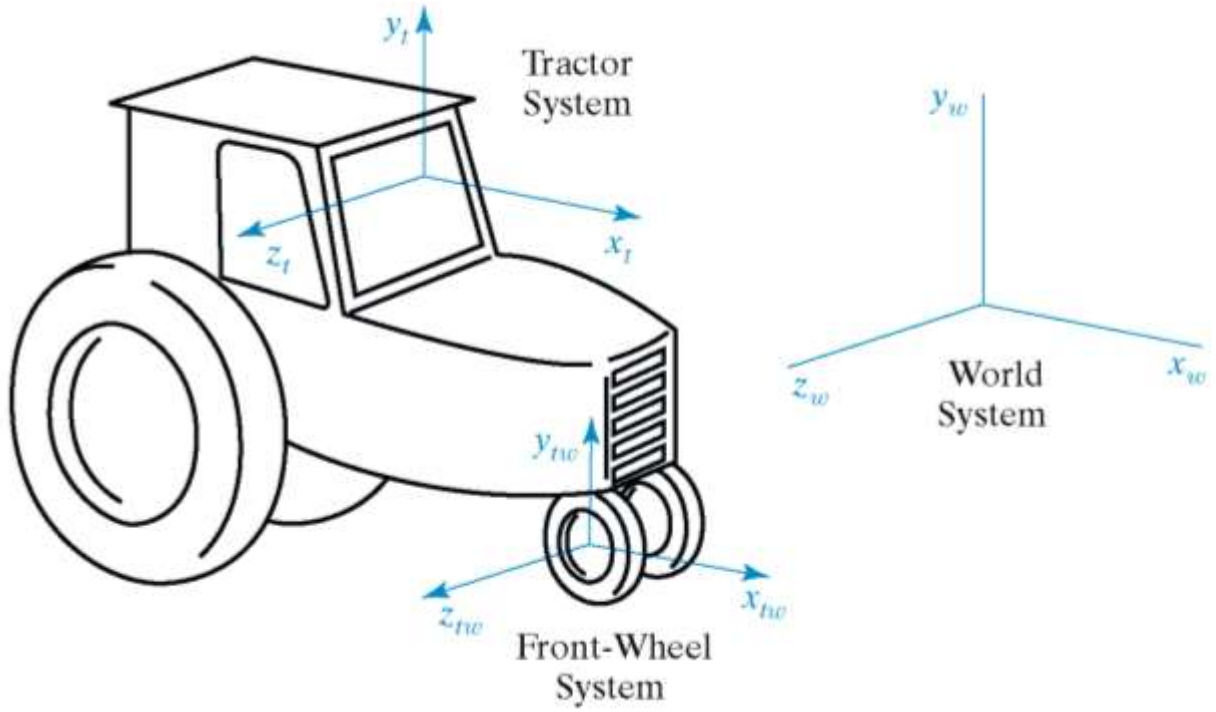
Q YOUN HONG (홍규연)

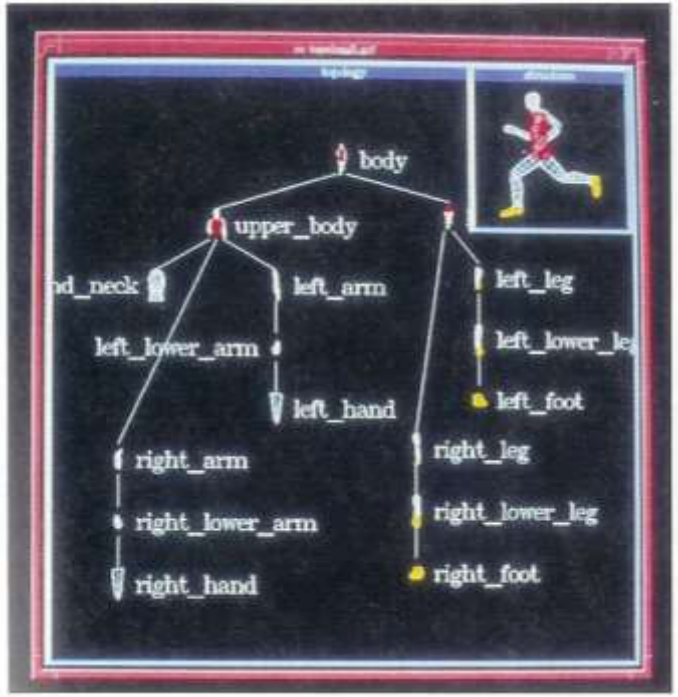
Hierarchical Modeling

Hierarchical Modeling

Hierarchical modeling:

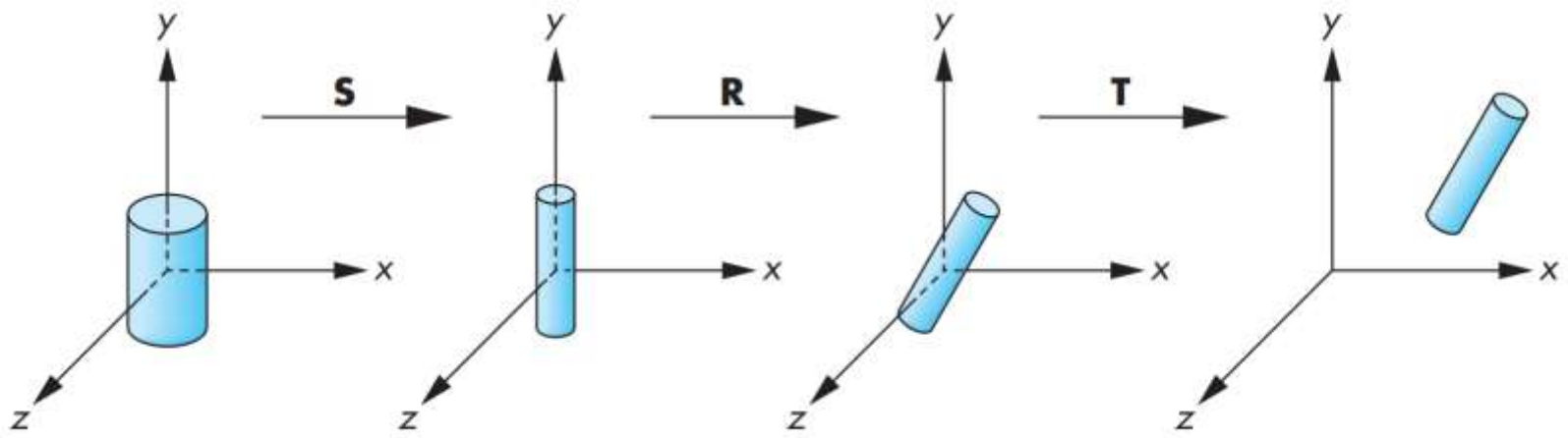
- 물체를 tree구조를 이용하여 계층적으로 표현하는 모델링
- 하위 물체의 특성(위치, 재질 등등)이 상위 물체에 nesting된 형태로 표현





Hierarchical Modeling

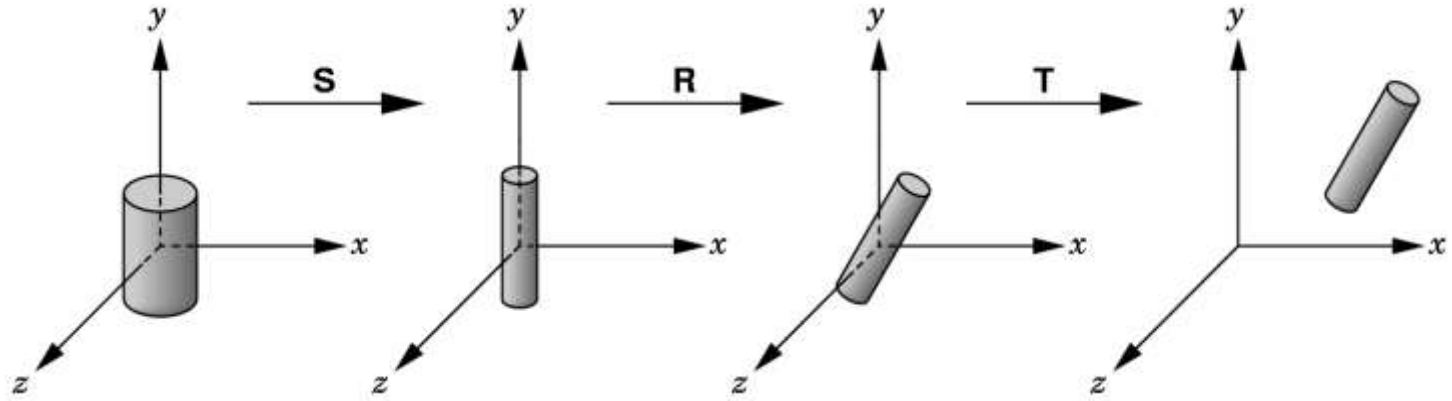
- Instance transformation:
  - 각 물체의 local model transform은 instance transformation으로 표현
  - 물체를 “standard”된 형태로 정의하고 instance transformation 적용
  - $M = TRS = Translate * Rotation * Scale$



Instance Transformation

In legacy OpenGL, matrices are organized in matrix stacks

```
glMatrixMode(GL_MODELVIEW); // GL_MODEL_VIEW/GL_PROEJECTION
glLoadIdentity(); // M <- I
glTranslatef(...); // M <- M * M_trans
glRotatef(...); // M <- M * M_trans * M_rotate
glScalef(...); // M <- M * M_trans * M_rotate * M_scale
gluCylinder(...); // Apply M to cylinder defined in local frame
```



Hierarchical Modeling

In legacy OpenGL, instance transformation of hierarchical models are also managed with matrix stacks

=> Stack processing supported

- `glPushMatrix();` // Duplicate the current matrix at top
- `glPopMatrix();` // Remove the matrix at top

Hierarchical Modeling

In modern OpenGL (with GLSL), we build our own matrix stacks

- Option 1. use (linked-list type) STL containers, (i.e.) std::list, std::deque, std::stack

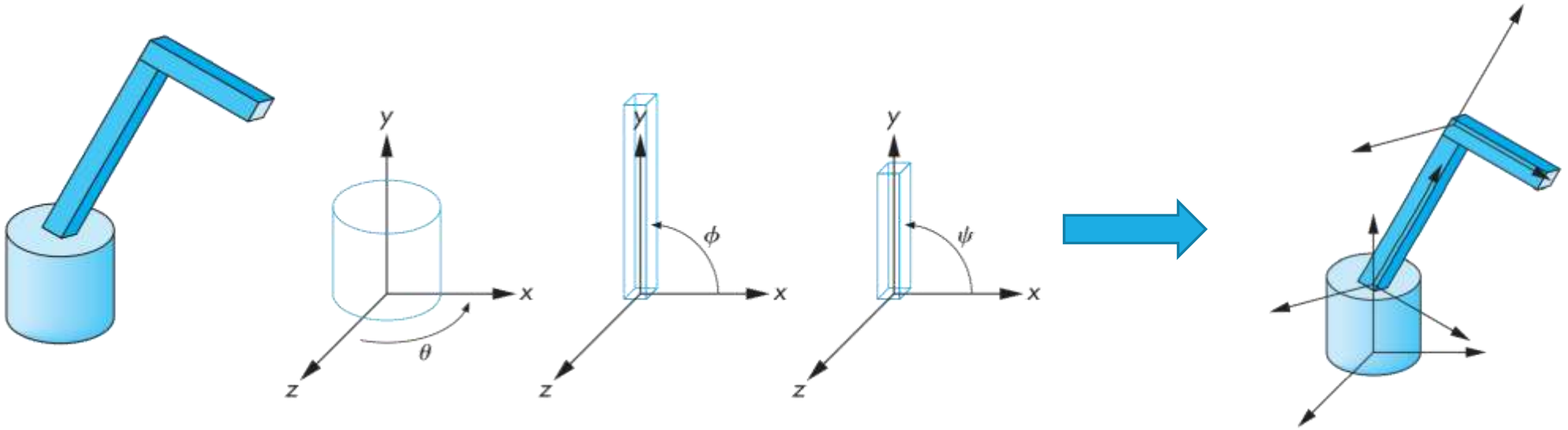
- Option 2. use a tree structure

Example 1. Robot Arm

Example: Robot Arm

Hierarchically modeled robot arm

- Robot arm = base + lower arm + upper arm
- lower arm's transformation(motion): depends on base's
- upper arm's transformation: depends on base's and lower arm's



Example: Robot Arm

- Base transformation: $R_y(\theta)$
- Lower arm transformation:

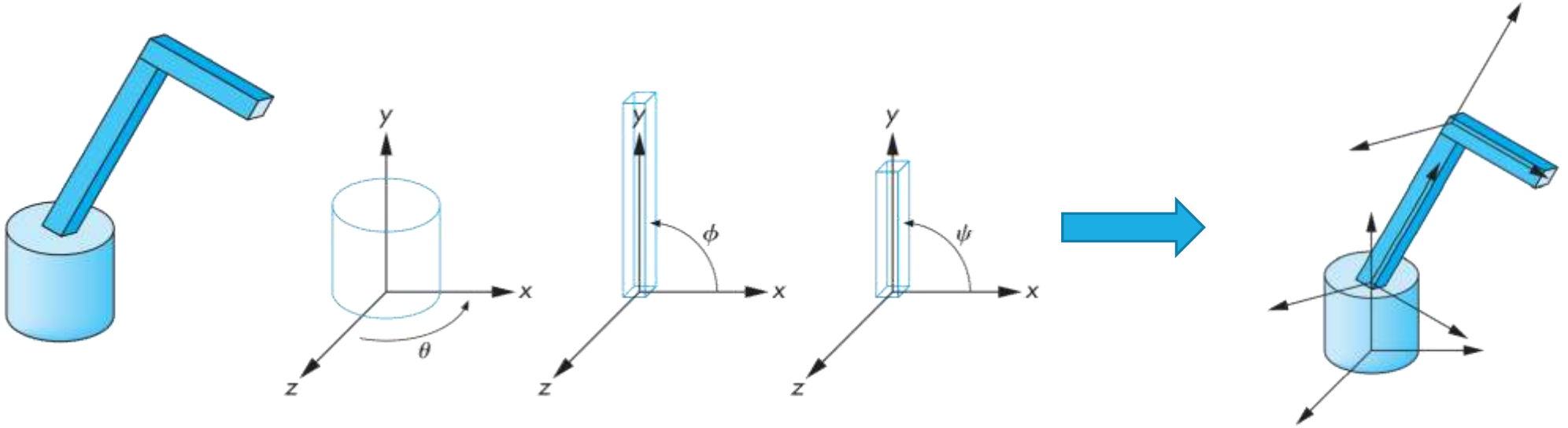
Lower arm의 원점을 lower arm의 parent frame(world frame)의 상대 적으로 위치

Lower arm의 local frame(원점: lower arm의 base의 상대적)에서의 transformation

$$R_y(\theta) * \left(T(0, h_1, 0) R_z(\phi)\right) = \left(R_y(\theta) T(0, h_1, 0)\right) R_z(\phi)$$

- Upper arm transformation:

$$R_y(\theta) * \left(T(0, h_1, 0) R_z(\phi) \left(T(0, h_2, 0) R_z(\psi)\right)\right) = \left(\mathrm{R_y}(\theta) T(0, h_1, 0) R_z(\phi) T(0, h_2, 0)\right) R_z(\psi)$$



Example: Robot Arm

- Parameters at the rest position

```
// Parameters controlling the size of the Robot's arm
const GLfloat BASE_HEIGHT      = 2.0;
const GLfloat BASE_WIDTH       = 5.0;
const GLfloat LOWER_ARM_HEIGHT = 5.0;
const GLfloat LOWER_ARM_WIDTH  = 0.5;
const GLfloat UPPER_ARM_HEIGHT = 5.0;
const GLfloat UPPER_ARM_WIDTH  = 0.5;
```

```
enum { Base = 0, LowerArm = 1, UpperArm = 2,
NumAngles = 3 };
int      Axis = Base;
GLfloat  Theta[NumAngles] = { 0.0 };
```

Example: Robot Arm

- Displaying a hierarchical model of the robot arm

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    // Accumulate ModelView Matrix as we traverse the tree
    model_view = RotateY(Theta[Base] );
    base();

    model_view *= ( Translate(0.0, BASE_HEIGHT, 0.0) *
    RotateZ(Theta[LowerArm]) );
    lower_arm();

    model_view *= ( Translate(0.0, LOWER_ARM_HEIGHT, 0.0) *
    RotateZ(Theta[UpperArm]) );
    upper_arm();

    glutSwapBuffers();
}
```

Step 1.

$M \leftarrow R_y(\theta)$

Example: Robot Arm

- Displaying a hierarchical model of the robot arm

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    // Accumulate ModelView Matrix as we traverse the tree
    model_view = RotateY(Theta[Base] );
    base();

    model_view *= ( Translate(0.0, BASE_HEIGHT, 0.0) *
    RotateZ(Theta[LowerArm]) );
    lower_arm();

    model_view *= ( Translate(0.0, LOWER_ARM_HEIGHT, 0.0
    RotateZ(Theta[UpperArm]) );
    upper_arm();

    glutSwapBuffers();
}
```

Step 1.

$M \leftarrow R_y(\theta)$

Step 2.

$M \leftarrow M * M_{base}$

Draw a base with M

```
void base()
{
    mat4 instance = (
      Translate( 0.0, 0.5 * BASE_HEIGHT, 0.0 ) *
      Scale( BASE_WIDTH, BASE_HEIGHT, BASE_WIDTH ) );

    glUniformMatrix4fv( ModelView, 1, GL_TRUE,
                        model_view * instance );

    glDrawArrays( GL_TRIANGLES, 0, NumVertices );
}
```

Example: Robot Arm

- Displaying a hierarchical model of the robot arm

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    // Accumulate ModelView Matrix as we traverse the tree
    model_view = RotateY(Theta[Base] );
    base();

    model_view *= ( Translate(0.0, BASE_HEIGHT, 0.0) *
    RotateZ(Theta[LowerArm]) );
    lower_arm();

    model_view *= ( Translate(0.0, LOWER_ARM_HEIGHT, 0.0) *
    RotateZ(Theta[UpperArm]) );
    upper_arm();

    glutSwapBuffers();
}
```

Step 1.

$$M \leftarrow R_y(\theta)$$

Step 2.

$M \leftarrow M * M_{base}$

Draw a base with M

Step 3.

$M \leftarrow M * M_{lower}$

Draw the lower arm with M

Example: Robot Arm

- Displaying a hierarchical model of the robot arm

```
void display( void )
{
    glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );

    // Accumulate ModelView Matrix as we traverse the tree
    model_view = RotateY(Theta[Base] );
    base();

    model_view *= ( Translate(0.0, BASE_HEIGHT, 0.0) *
    RotateZ(Theta[LowerArm]) );
    lower_arm();

    model_view *= ( Translate(0.0, LOWER_ARM_HEIGHT, 0.0) *
    RotateZ(Theta[UpperArm]) );
    upper_arm();

    glutSwapBuffers();
}
```

Step 1.

$$M \leftarrow R_y(\theta)$$

Step 2.

$M \leftarrow M * M_{base}$

Draw a base with M

Step 3.

$M \leftarrow M * M_{lower}$

Draw the lower arm with M

Step 4.

$M \leftarrow M * M_{upper}$

Draw the upper arm with M

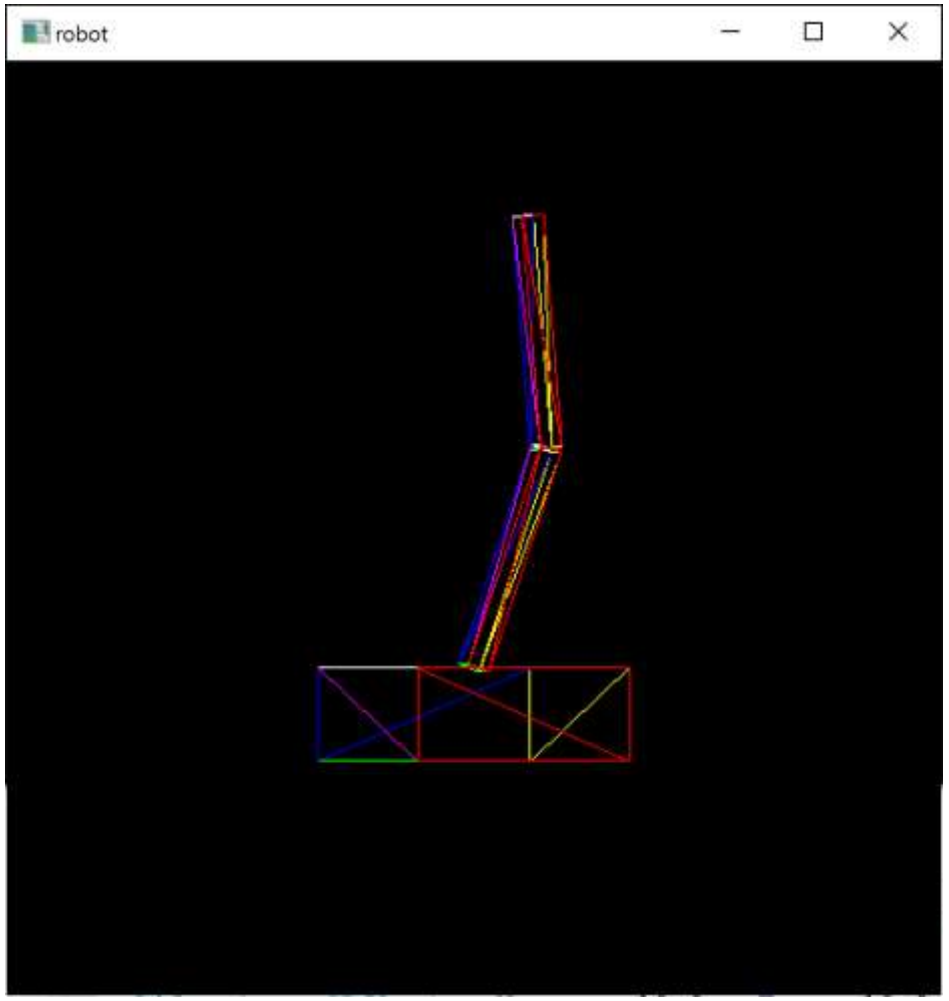
Example: Robot Arm

- Add user-defined pop-up menus to the window

```
In main():
glutCreateMenu( menu );
// Set the menu values to the relevant rotation axis values (or Quit)
glutAddMenuEntry( "base", Base );
glutAddMenuEntry( "lower arm", LowerArm );
glutAddMenuEntry( "upper arm", UpperArm );
glutAddMenuEntry( "quit", Quit );
glutAttachMenu( GLUT_MIDDLE_BUTTON );
```

```
void menu( int option ) {
    if ( option == Quit )
        exit( EXIT_SUCCESS );
    else {
        Axis = option;
}
```

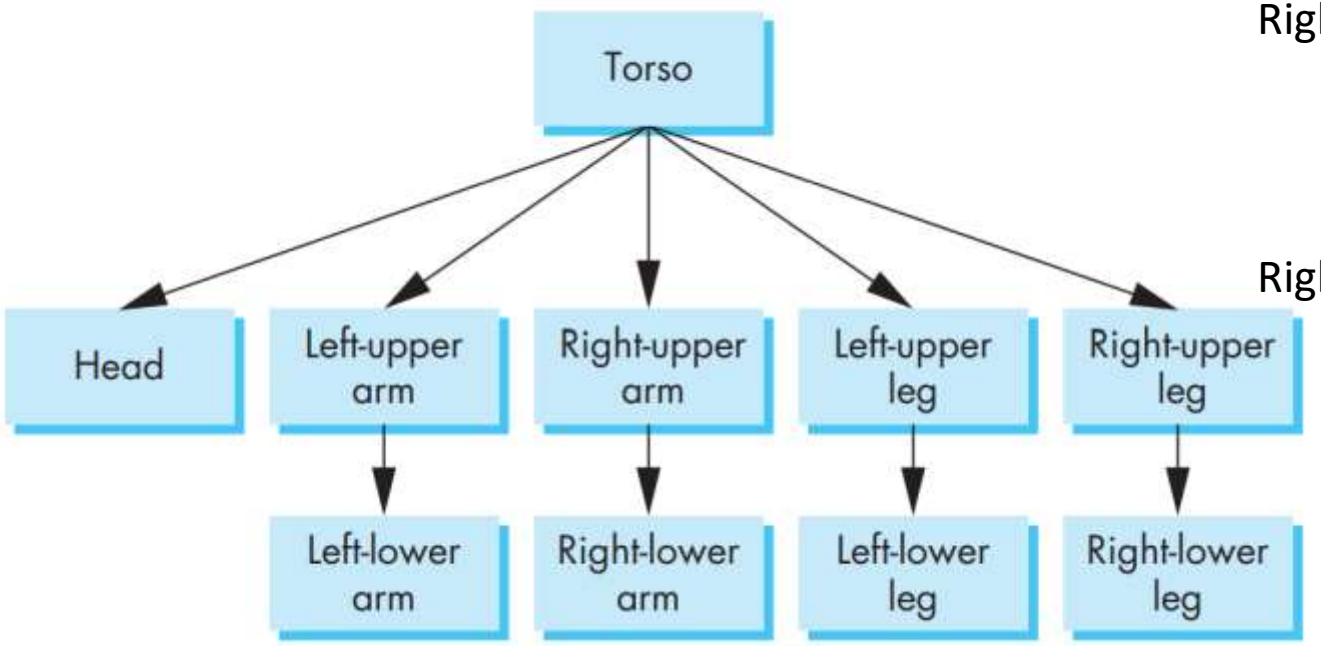
Example: Robot Arm

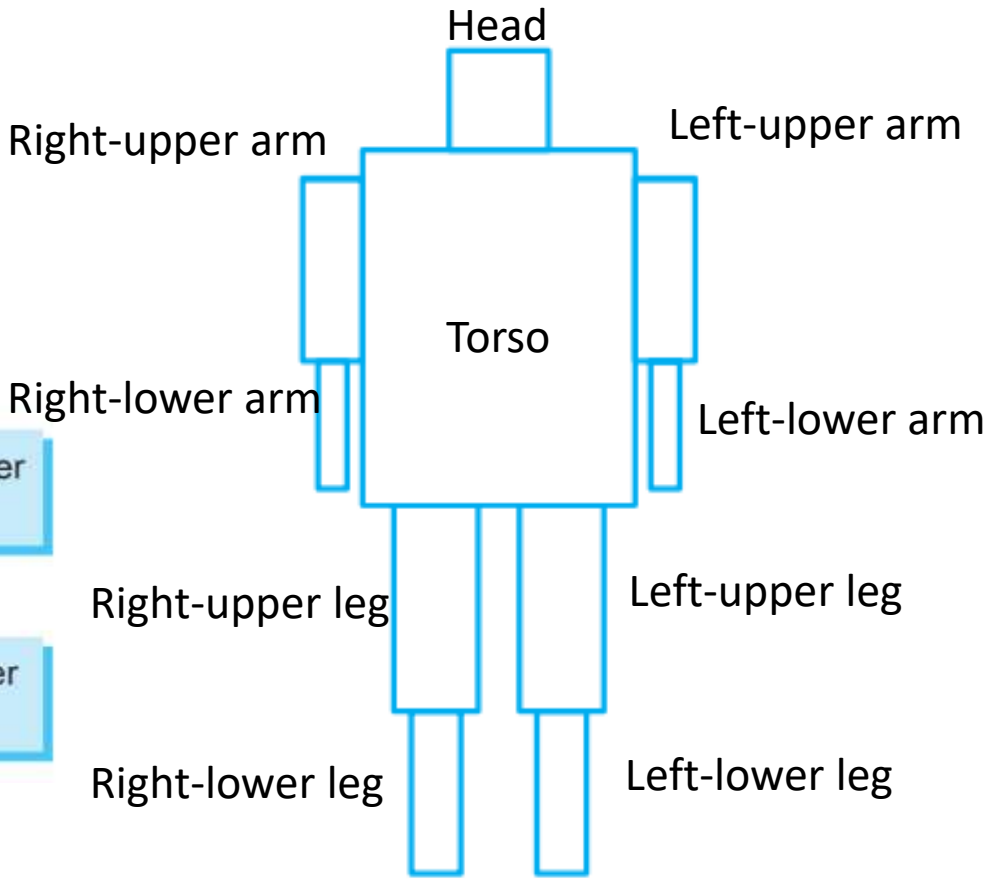


Example: Robot (Full)

Example: Robot (Full)

Hierarchically model a full body of a robot

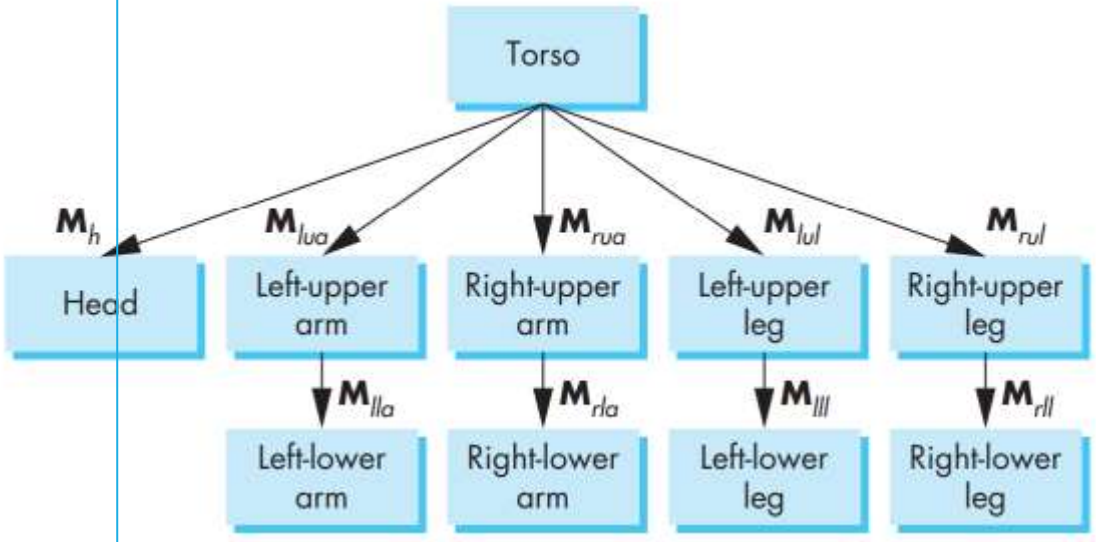




Exampl: Robot(Full)

- Manage transformation matrices with a matrix stack

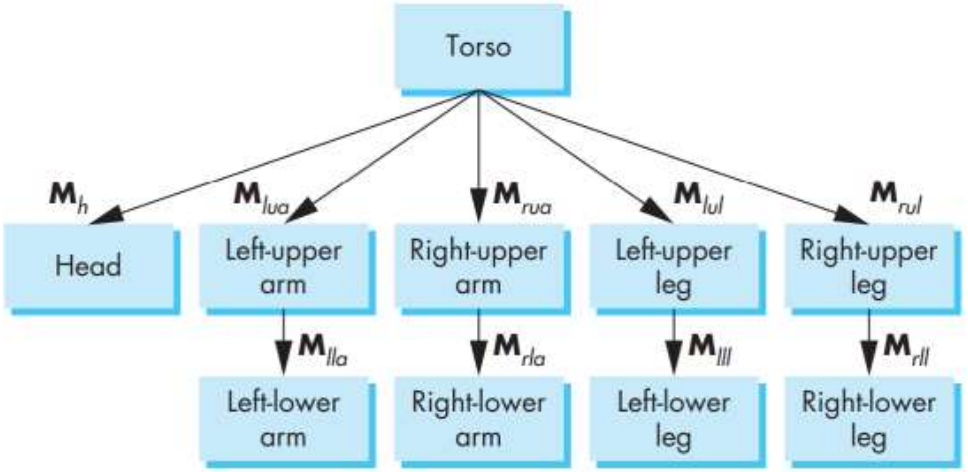
```
class MatrixStack {
    std::stack<mat4> _matrices;

public:
    void push(const mat4& m) {
        _matrices.push(m);
    }
    mat4& pop(void) {
        mat4 ret = _matrices.top();
        _matrices.pop();
        return ret;
    }
};

// Shader transformation matrices
MatrixStack mvstack;
mat4  model_view;
```



Example: Robot (Full)

Drawing the hierarchical robot model: draw in *pre-order*

```
void display(void) {
    …
    model_view = RotateY(theta[Torso]);
    torso();

    mvstack.push(model_view);
    model_view *= ( Translate( 0.0, TORSO_HEIGHT + 0.5 * HEAD_HEIGHT, 0.0 ) *
    RotateX( theta[Head1] ) *
    RotateY( theta[Head2] ) *
    Translate( 0.0, -0.5 * HEAD_HEIGHT, 0.0 ) );
    head();
    model_view = mvstack.pop();
        …
```

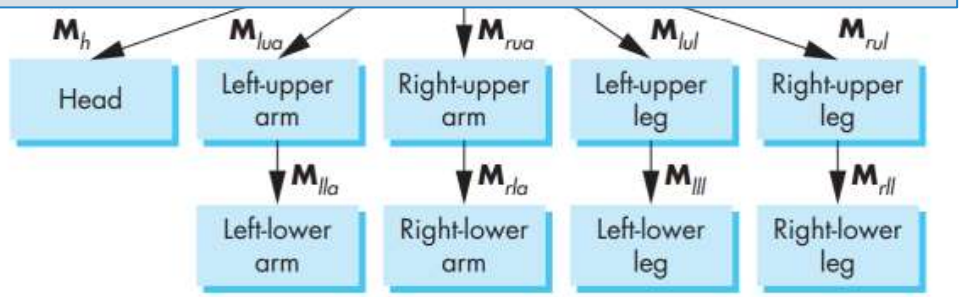


Example: Robot (Full)

Drawing the hierarchical robot model: draw in *pre-order*

```
void display(void) {
    …
    model_view = RotateY(t
    torso();

    mvstack.push(model_vie
    model_view *= ( Transl
    RotateX( theta[Head1]
    RotateY( theta[Head2]
    Translate( 0.0, -0.5 *
    head();
    model_view = mvstack.pop();
        …
```

```
void torso()
{
    mvstack.push( model_view );

    mat4 instance = ( Translate( 0.0, 0.5 * TORSO_HEIGHT, 0.0 ) *
      Scale( TORSO_WIDTH, TORSO_HEIGHT, TORSO_WIDTH ) );

    glUniformMatrix4fv( ModelView, 1, GL_TRUE, model_view * instance );
    glDrawArrays( GL_TRIANGLES, 0, NumVertices );

    model_view = mvstack.pop();
}
```



Example: Robot (Full)

Parameters at the rest position

```
const GLfloat TORSO_HEIGHT = 5.0;
const GLfloat TORSO_WIDTH = 1.0;
const GLfloat UPPER_ARM_HEIGHT = 3.0;
const GLfloat UPPER_ARM_WIDTH = 0.5;
const GLfloat LOWER_ARM_HEIGHT = 2.0;
const GLfloat LOWER_ARM_WIDTH = 0.5;
const GLfloat UPPER_LEG_HEIGHT = 3.0;
const GLfloat UPPER_LEG_WIDTH = 0.5;
const GLfloat LOWER_LEG_HEIGHT = 2.0;
const GLfloat LOWER_LEG_WIDTH = 0.5;
const GLfloat HEAD_HEIGHT = 1.5;
const GLfloat HEAD_WIDTH = 1.0;
```

```
// Joint angles with initial values
Glfloat theta[NumJointAngles] = {
    0.0,    // Torso
    0.0,    // Head1
    0.0,    // Head2
    0.0,    // RightUpperArm
    0.0,    // RightLowerArm
    0.0,    // LeftUpperArm
    0.0,    // LeftLowerArm
    180.0,  // RightUpperLeg
    0.0,    // RightLowerLeg
    180.0,  // LeftUpperLeg
    0.0     // LeftLowerLeg
};

GLint angle = Head2;
```

Execution Result