Forward Kinematics:

You Start with Separate Pieces, all Defined in their Own Local Coordinate System.

Hook the Pieces Together, Change Parameters, Things Move
(All Children Understand This)

Where do the Pieces Move To?
Locations?
Positioning Part #1 With Respect to Ground

1. Rotate by $\Theta_1$
2. Translate by $T_{1/G}$

$$M_{1/G} = [T_{1/G}] * [R_{\Theta_1}]$$

Why Do We Say it Right-to-Left?

It's because in the matrix notes, we adopted the convention that the coordinates are multiplied on the right side of the matrix:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

So the right-most transformation in the sequence multiplies the $(x,y,z,1)$ first and the left-most transformation multiplies it last.

Positioning Part #2 With Respect to Ground

1. Rotate by $\Theta_2$
2. Translate the length of part 1
3. Rotate by $\Theta_1$
4. Translate by $T_{1/G}$

$$M_{2/G} = [T_{1/G}] * [R_{\Theta_1}] * [T_{2/G}] * [R_{\Theta_2}]$$

$$M_{2/G} = [M_{1/G}] * [M_{2/G}]$$

Positioning Part #3 With Respect to Ground

1. Rotate by $\Theta_3$
2. Translate the length of part 2
3. Rotate by $\Theta_2$
4. Translate the length of part 1
5. Rotate by $\Theta_1$
6. Translate by $T_{1/G}$

$$M_{3/G} = [T_{1/G}] * [R_{\Theta_1}] * [T_{2/G}] * [R_{\Theta_2}] * [T_{3/G}] * [R_{\Theta_3}]$$

$$M_{3/G} = [M_{1/G}] * [M_{2/G}] * [M_{3/G}]$$
Sample Program

```c
DrawLinkOne() {
    glColor3f(1., 0., 0.); // red, green blue
    glBegin(GL_QUADS);
    glVertex2f(-BUTT, -THICKNESS/2);
    glVertex2f(LENGTH_1, -THICKNESS/2);
    glVertex2f(LENGTH_1, THICKNESS/2);
    glVertex2f(-BUTT, THICKNESS/2);
    glEnd();
}
```

```
Sample Program, using OpenGL's Automatic Transformation Concatenation

```c
DrawMechanism( float \( \theta_1 \), float \( \theta_2 \), float \( \theta_3 \)) {
    glPushMatrix();
    glRotatef(\( \theta_1 \), 0., 0., 1.);
    glColor3f(1., 0., 0.);
    DrawLinkOne();
    glTranslatef(LENGTH_1, 0., 0.);
    glRotatef(\( \theta_2 \), 0., 0., 1.);
    glColor3f(0., 1., 0.);
    DrawLinkTwo();
    glTranslatef(LENGTH_2, 0., 0.);
    glRotatef(\( \theta_3 \), 0., 0., 1.);
    glColor3f(0., 0., 1.);
    DrawLinkThree();
    glPopMatrix();
}
```

Sample Program

```c
glViewport(100, 100, 500, 500);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90., 1.0, 1.0, 10.);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
done = FALSE;
while(!done) {
    << Determine \( \theta_1, \theta_2, \theta_3 >>
    glPushMatrix();
    gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz);
    DrawMechanism(\( \theta_1 \), \( \theta_2 \), \( \theta_3 \));
    glPopMatrix();
}
```

Sample Program

```c
glViewport(100, 100, 500, 500);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(90., 1.0, 1.0, 10.);
eglMatrixMode(GL_MODELVIEW);
eglLoadIdentity();
done = FALSE;
while(!done) {
    << Determine \( \theta_1, \theta_2, \theta_3 >>
    glPushMatrix();
    gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz);
    DrawMechanism(\( \theta_1 \), \( \theta_2 \), \( \theta_3 \));
    glPopMatrix();
}
```
In Project #4, you won’t be able to do this. You will need to create each full matrix separately using your own Matrix class methods.