Forward Kinematics:
You Start with Separate Pieces, all Defined in their Own Local Coordinate System
Forward Kinematics:
Hook the Pieces Together, Change Parameters, Things Move
(All Children Understand This)

Locations?

Ground
Positioning Part #1 With Respect to Ground

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

$$\begin{align*}
[M_{1/G}] &= [T_{1/G}] \cdot [R_{\Theta_1}] \\
&= [T_{1/G}] \cdot [R_{\Theta_1}]
\end{align*}$$

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:

$$\begin{align*}
\begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} &= M_{v/G} \begin{bmatrix}
x \\
y \\
z
\end{bmatrix} = T_{1/G} \cdot [R_{\Theta_1}] \begin{bmatrix}
x \\
y \\
z
\end{bmatrix} \\
&= \begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
x' \\
y' \\
z'
\end{bmatrix} = \begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
\end{align*}$$

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 $T_{1/G}$

Write it

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

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

Say it

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 $T_{1/G}$

Write it

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

\[
M_{3/G} = M_{1/G} * M_{2/1} * M_{3/2}
\]

Say it
Sample Program

```
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
Sample Program

\[
[M_{3/6}] = [M_{1/6}] \times [M_{2/1}] \times [M_{3/2}]
\]

```c
DrawMechanism(float \theta_1, float \theta_2, float \theta_3)
{
    glPushMatrix();
    glTranslatef(X1, Y1, Z1);
    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., 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();
}
```
In your Forward Kinematics project, you won’t be allowed to do this.

You will need to create each \( \mathbf{M}_{\text{ic}} \) matrix separately using GLM Matrix class methods.

\[
\begin{align*}
\mathbf{M}_{1/2} &= \mathbf{T}_{1/2} \times \mathbf{R}_{1/1} \\
\mathbf{M}_{2/3} &= \mathbf{T}_{2/3} \times \mathbf{R}_{2/2} \\
\mathbf{M}_{3/4} &= \mathbf{T}_{3/4} \times \mathbf{R}_{3/3} \\
\end{align*}
\]

What If They Are Sliding Connections, Not Rotation Connections?

Sometimes, these are called **Prismatic Constraints**.

\[
\mathbf{M}_{3/G} = \mathbf{M}_{1/G} \times \mathbf{M}_{2/1} \times \mathbf{M}_{3/2} = \mathbf{T}_{1/G} \times \mathbf{T}_{2/1} \times \mathbf{T}_{3/2}
\]