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)

Forward Kinematics: Where do the Pieces Move To?

Locations?

Positioning Part #1 With Respect to Ground

1. Rotate by $\theta_1$
2. Translate by $T_{1/G}$

Write it

$$[M_{1/G}] = [T_{1/G}] \cdot [R_{\theta_1}]$$

Say it

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.

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

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

mjb@cs.oregonstate.edu
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 $T_{G}$

\[
[M_{2/G}] = [T_{1/G}] \cdot [R_{\Theta_1}] \cdot [T_{21}] \cdot [R_{\Theta_2}]
\]

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 $T_{G}$

\[
[M_{3/G}] = [T_{1/G}] \cdot [R_{\Theta_1}] \cdot [T_{21}] \cdot [R_{\Theta_2}] \cdot [T_{32}] \cdot [R_{\Theta_3}]
\]

Sample Program, using OpenGL's Automatic Transformation Concatenation

```c
void DrawMechanism( float $\Theta_1$, float $\Theta_2$, float $\Theta_3$ )
{
    glPushMatrix( );
    gluLookAt( eyex, eyey, eyez, 
               centerx, centery, centerz, 
               upx, upy, upz );
    glTranslatef( LENGTH_1, 0., 0. );
    glRotatef( $\Theta_2$, 0., 0., 1. );
    glTranslatef( LENGTH_2, 0., 0. );
    glColor3f( 0., 0., 1. );
    DrawLinkThree( );
    glPopMatrix( );
}
```

Sample Program

1. Where in the window to display (pixels)
2. Viewing Info:
   - field of view:
   - aspect ratio:
   - near:
   - far:

```c
glViewport( 100, 100, 500, 500 );
glMatrixMode( GL_PROJECTION );
gluPerspective( 90., 1.0, 1., 10. );
glMatrixMode( GL_MODELVIEW );
done = FALSE;
while( ! done )
{
    << Determine $\Theta_1, \Theta_2, \Theta_3$ >>
    glLoadIdentity( );
    done = FALSE;
    while( ! done )
    {
        << Find eye position >>
        glBegin( GL_QUADS );
        glEnd( );
        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 full matrix separately using GLM Matrix class methods.

\[
[M_{3/2}] = [M_{1/2}] \cdot [M_{2/3}] \cdot [M_{3/2}]
\]

What If They Are Sliding Connections, Not Rotation Connections?

Sometimes, these are called Prismatic Constraints

\[
[M_{3/2}] = [M_{1/2}] \cdot [M_{2/3}] \cdot [M_{3/2}] = [T_{1/2}] \cdot [T_{2/3}] \cdot [T_{3/2}]
\]