Forward Kinematics

Mike Bailey
mjb@cs.oregonstate.edu
Oregon State University
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)
Forward Kinematics: Where do the Pieces Move To?

Locations?
Positioning Part #1 With Respect to Ground

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

\[ [M_{1/G}] = [T_{1/G}] \times [R_{\theta_1}] \]
Why Do We Say it Right-to-Left?

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

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

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
1
\end{bmatrix} = \left[ \begin{array}{cccc}
A & B & C & D \\
E & F & G & H \\
I & J & K & L \\
0 & 0 & 0 & 1 \\
\end{array} \right] \cdot \begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}
\]

So the right-most transformation in the sequence multiplies the \((x,y,z,1)\) \textit{first} and the left-most transformation multiplies it \textit{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 $T_{1/G}$

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

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

\[
\begin{bmatrix}
M_{3/G}
\end{bmatrix} = \begin{bmatrix}
T_{1/G}
\end{bmatrix} \ast \begin{bmatrix}
R_{\Theta_1}
\end{bmatrix} \ast \begin{bmatrix}
T_{2/1}
\end{bmatrix} \ast \begin{bmatrix}
R_{\Theta_2}
\end{bmatrix} \ast \begin{bmatrix}
T_{3/2}
\end{bmatrix} \ast \begin{bmatrix}
R_{\Theta_3}
\end{bmatrix}
\]
Sample Program, using OpenGL’s Automatic Transformation Concatenation

```
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

```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( );
}
```

\[
\begin{align*}
[M_{1/G}] &= [T_{1/G}] \cdot [R_{\theta_1}] \\
[M_{2/G}] &= [T_{1/G}] \cdot [R_{\theta_1}] \cdot [T_{2/1}] \cdot [R_{\theta_2}] \\
[M_{3/G}] &= [T_{1/G}] \cdot [R_{\theta_1}] \cdot [T_{2/1}] \cdot [R_{\theta_2}] \cdot [T_{3/2}] \cdot [R_{\theta_3}]
\end{align*}
\]
Sample Program

```c
glViewport( 100, 100, 500, 500 );

glMatrixMode( GL_PROJECTION );
gluPerspective( 90., 1.0, 1., 10. );

glMatrixMode( GL_MODELVIEW );
gluLookAt( eyex, eyey, eyez,
           centerx, centery, centerz,
           upx, upy, upz );

<< Determine $\theta_1, \theta_2, \theta_3$ >>
gluLookAt( eyex, eyey, eyez,
           centerx, centery, centerz,
           upx, upy, upz );

DrawMechanism( $\theta_1, \theta_2, \theta_3$ );
glPopMatrix();
```

Where in the window to display (pixels)

Viewing Info:
- field of view angle, x:y aspect ratio, near, far

Whatever interaction is being used to get the eye position

Set the eye position
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.