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?

Positioning Part #1 With Respect to Ground

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

Write it

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

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.
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

$$M_{2/G} = \begin{bmatrix} T_1[G] \cdot R_{\Theta_1} \cdot T_{2/1} \cdot R_{\Theta_2} \\ M_{1/G} \cdot M_{2/1} \end{bmatrix}$$

Say it

$$M_{3/G} = \begin{bmatrix} T_1[G] \cdot R_{\Theta_1} \cdot T_{2/1} \cdot R_{\Theta_2} \cdot T_{3/2} \cdot R_{\Theta_3} \\ M_{1/G} \cdot M_{2/1} \cdot M_{3/2} \end{bmatrix}$$

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 $T_1/G$

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( );
}
```

```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();
}
```

Sample Program

```c
Sample Program
```
Sample Program

```cpp
void drawMechanism(float angle1, float angle2, float angle3)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glmRotatef(angle1, 0.0f, 0.0f, 1.0f);
    glmRotatef(angle2, 0.0f, 0.0f, 1.0f);
    glmRotatef(angle3, 0.0f, 1.0f, 0.0f);
    glmTranslatef(LENGTH_1, 0.0f, 0.0f);
    glmTranslatef(LENGTH_2, 0.0f, 0.0f);
    DrawLinkOne();
    DrawLinkTwo();
    DrawLinkThree();
    glPopMatrix();
}
```

In your Forward Kinematics project, you won’t be allowed to do this.

You will need to create each full matrix separately using your own Matrix class methods.