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?
### Positioning Part #1 With Respect to Ground

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

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

### 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 $T_{1/G}$

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

### 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 $T_{1/G}$

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

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

\[
[M_{3/G}] = [M_{1/G}] \cdot [M_{2/1}] \cdot [M_{3/2}]
\]
**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
Sample Program, using OpenGL's Automatic Transformation Concatenation

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 );
gluPerspective( 90., 1.0, 1., 10. );
glMatrixMode( GL_MODELVIEW );
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
// transformMechanism (float 01, float 02, float 03)
{
    GLfloat
        matrix = [0];
    glPushMatrix();
    glLoadIdentity();
    glBegin( GL_LINES );
    glVertex2f( 0.0, 0.0 );
    glVertex2f( 1.0, 0.0 );
    glEnd();
    glPushMatrix();
    glLoadIdentity();
    glBegin( GL_LINES );
    glVertex2f( LEN 1, 0.0 );
    glVertex2f( LEN 2, 0.0 );
    glEnd();
    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.