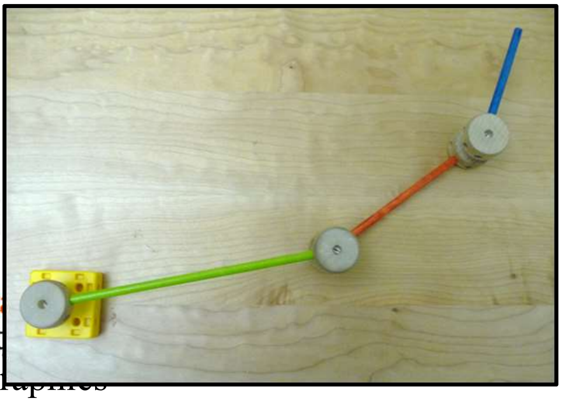
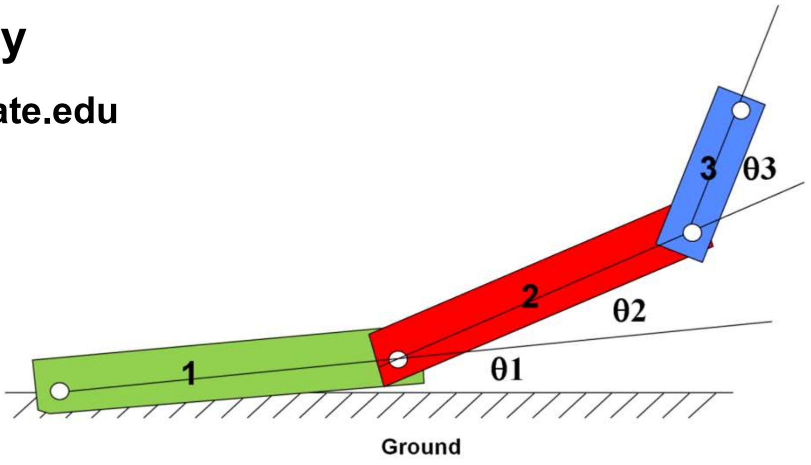
Forward Kinematics

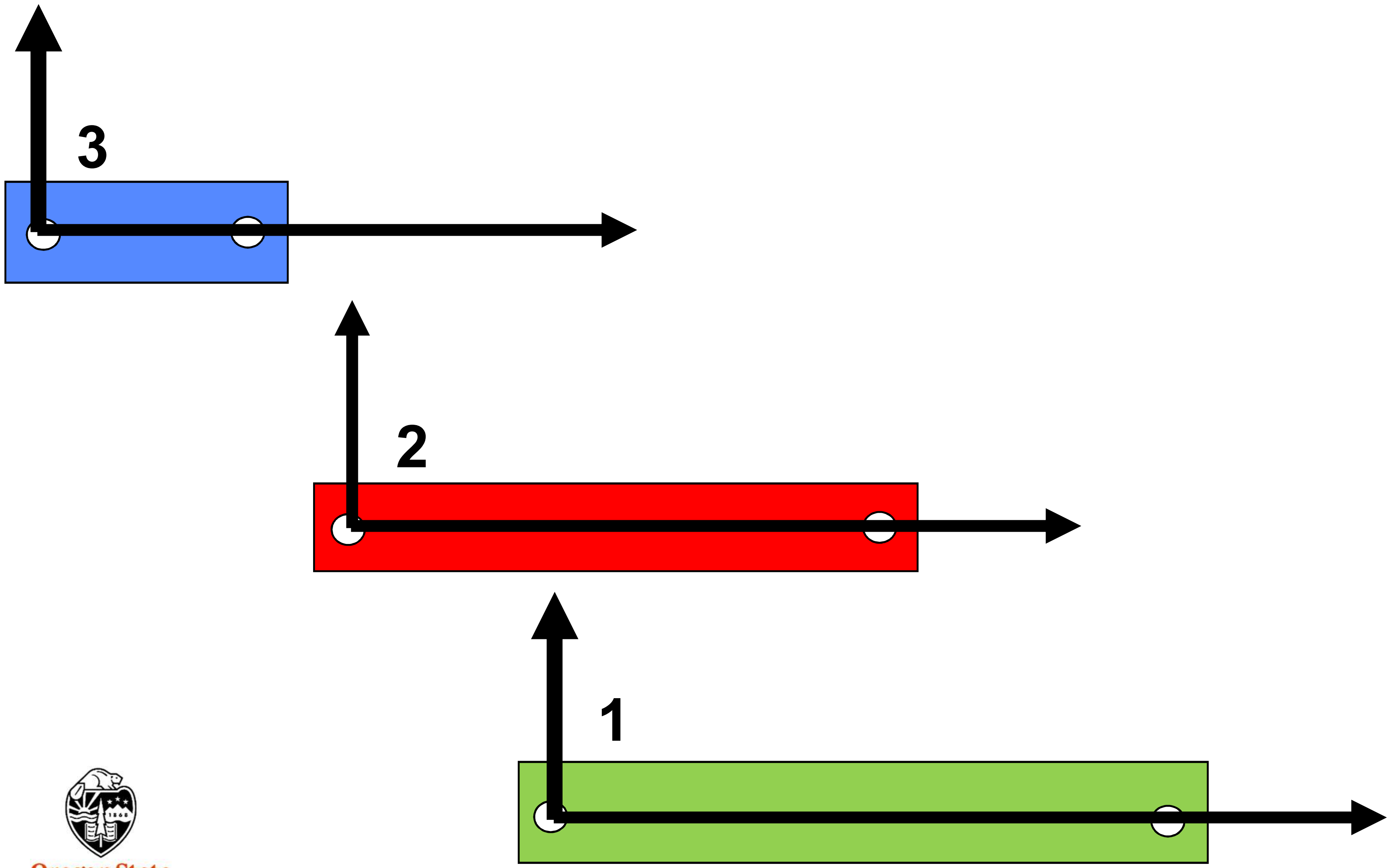
Mike Bailey

mjb@cs.oregonstate.edu



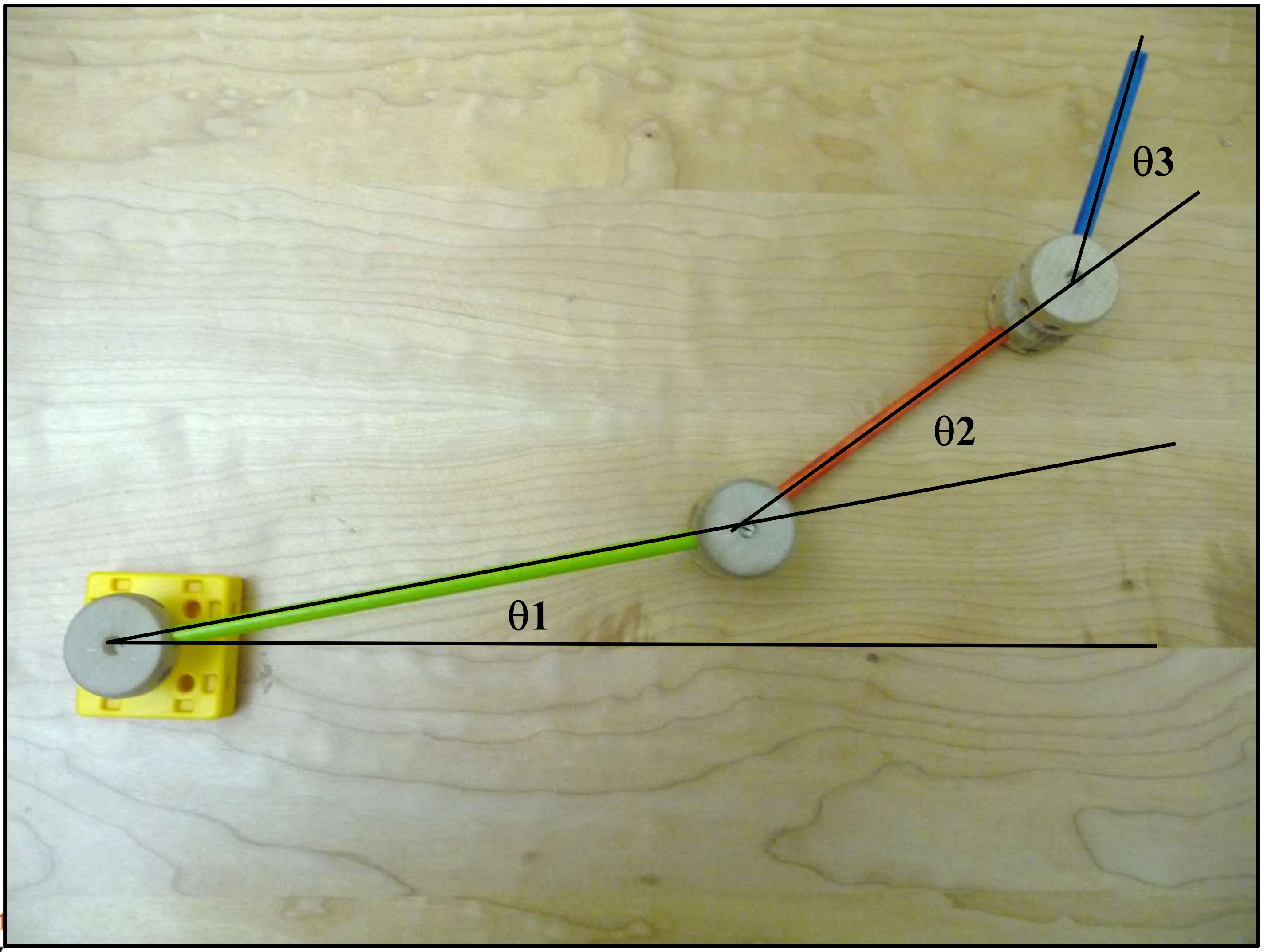


Forward Kinematics: You Start with Separate Pieces, all Defined in their Own Local Coordinate System

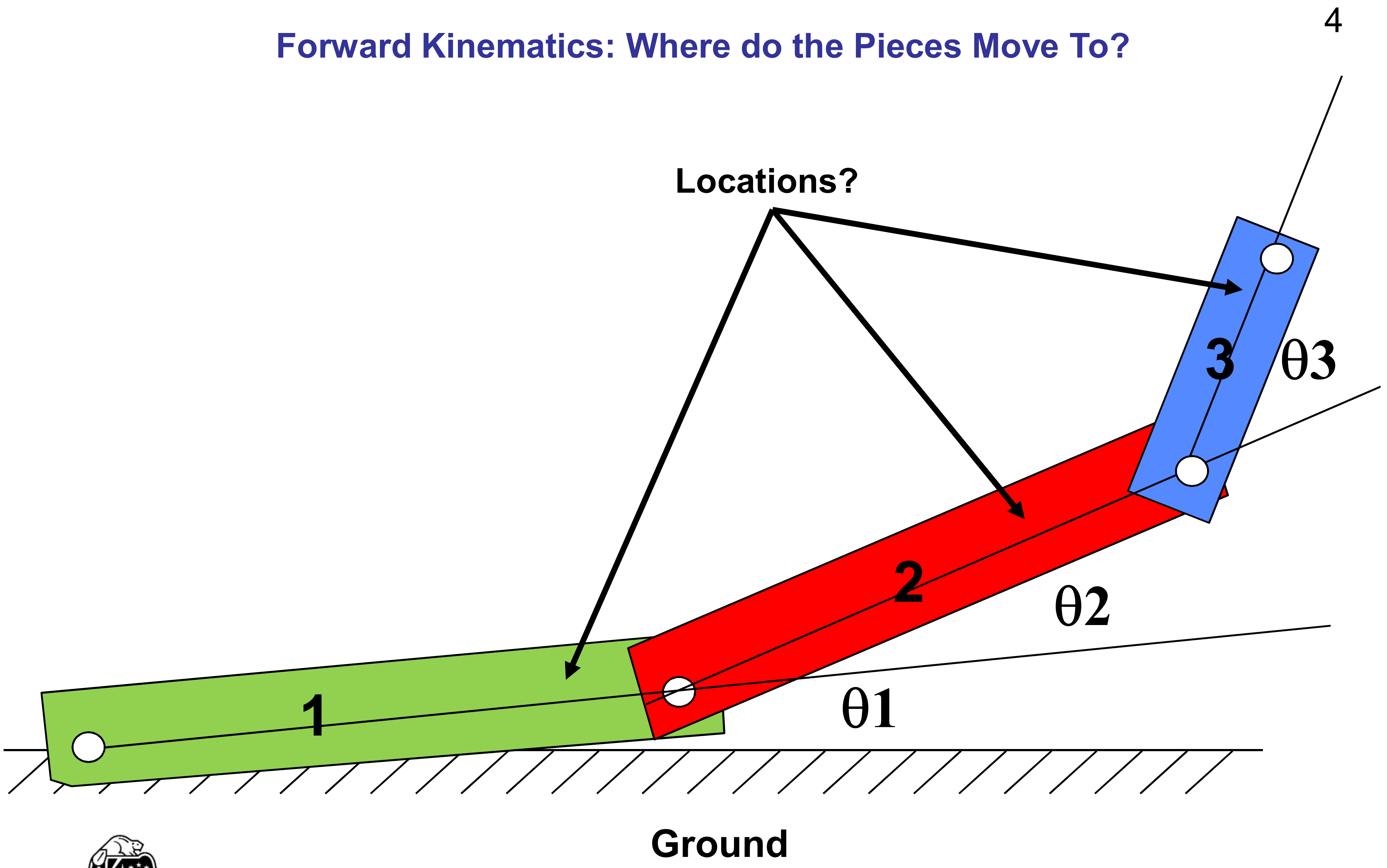


Oregon State University Computer Graphics

Forward Kinematics:
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 Θ1
2. Translate by $T_{1/G}$

Write it

$\longrightarrow$

$$\left[M_{1/G}\right] = \left[T_{1/G}\right] * \left[R_{\theta 1}\right]$$

Say it

$\longleftarrow$

Why Do We Say it Right-to-Left?

$$\xrightarrow{\text{Write it}}$$

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

$$\xleftarrow{\text{Say it}}$$

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

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = \begin{bmatrix} A & B & C & D \\ E & F & G & H \\ I & J & K & L \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

$$\begin{Bmatrix} x' \\ y' \\ z' \\ 1 \end{Bmatrix} = [M_{1/G}] \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} = [T_{1/G}] * [R_{\theta 1}] * \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix}$$

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

Positioning Part #2 With Respect to Ground

1. Rotate by Θ2
2. Translate the length of part 1
3. Rotate by Θ 1
4. Translate by $T_{1/G}$

Write it

$$\xrightarrow{\hspace{6cm}}$$

$$[M_{2/G}] = \underbrace{[T_{1/G}]*[R_{\theta 1}]}*\underbrace{[T_{2/1}]*[R_{\theta 2}]}$$

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

$$\xleftarrow{\hspace{6cm}}$$

Say it

Positioning Part #3 With Respect to Ground

1. Rotate by Θ3
2. Translate the length of part 2
3. Rotate by Θ2
4. Translate the length of part 1
5. Rotate by Θ1
6. Translate by $T_{1/G}$

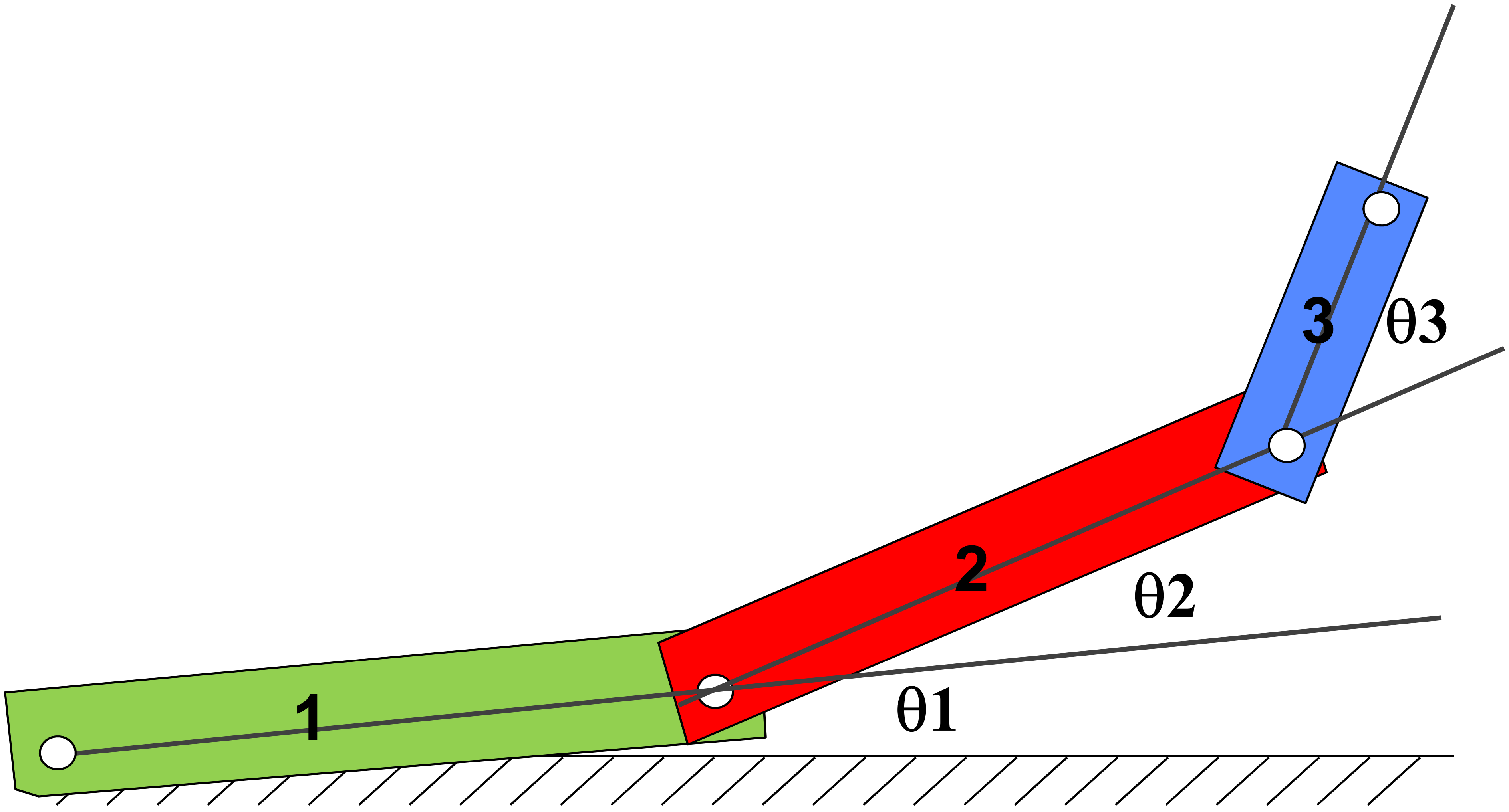
Write it $\longrightarrow$

$$[M_{3/G}] = \underbrace{[T_{1/G}]*[R_{\theta 1}]}_{[M_{1/G}]} * \underbrace{[T_{2/1}]*[R_{\theta 2}]}_{[M_{2/1}]} * \underbrace{[T_{3/2}]*[R_{\theta 3}]}_{[M_{3/2}]}$$

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

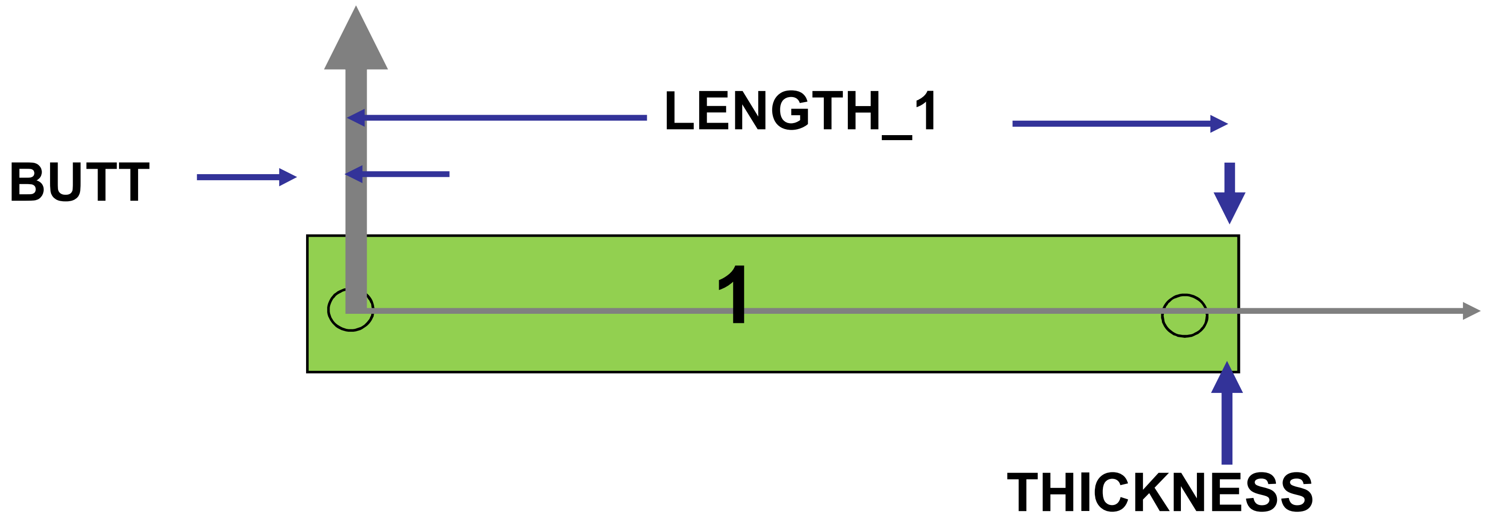
$\longleftarrow$ Say it

Sample Program



Ground

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

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

Write it

```
DrawMechanism( float θ1, float θ2, float θ3 )
{
        glPushMatrix( );
                gl_Translatef( X1, Y1, Z1 );
                glRotatef( θ1,  0., 0., 1. );
                glColor3f( 1., 0., 0. );
                DrawLinkOne( );

                glTranslatef( LENGTH_1, 0., 0. );
                glRotatef( θ2,  0., 0., 1. );
                glColor3f( 0., 1., 0. );
                DrawLinkTwo( );

                glTranslatef( LENGTH_2, 0., 0. );
                glRotatef( θ3,  0., 0., 1. );
                glColor3f( 0., 0., 1. );
                DrawLinkThree( );
        glPopMatrix( );
}
```

Say it

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

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

$[M_{3/G}] = [T_{1/G}] * [R_{\theta 1}] * [T_{2/1}] * [R_{\theta 2}] * [T_{3/2}] * [R_{\theta 3}]$

Sample Program

Where in the window to display (pixels)

3D Viewing Info: field of view angle, x:y aspect ratio, near, far

Set the eye position

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

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

glMatrixMode( GL_MODELVIEW );
glLoadIdentity( );

done = FALSE;
while( ! done )
{
        <<   Determine θ1, θ2, θ3   >>
        glPushMatrix();
        gluLookAt( eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz );
        DrawMechanism( θ1, θ2, θ3 );
        glPopMatrix();
}
```

Sample Program

```
DrawMechanism( float θ1, float θ2, float θ3 )
{
      glPushMatrix( );
            glRotatef( θ1,  0., 0., 1. );
            glColor3f( 1., 0., 0. );
            DrawLinkOne( );

            glTranslatef( LENGTH_1, 0., 0. );
            glRotatef( θ2,  0., 0., 1. );
            glColor3f( 0., 1., 0. );
            DrawLinkTwo( );

            glTranslatef( LENGTH_2, 0., 0. );
            glRotatef( θ3,  0., 0., 1. );
            glColor3f( 0., 0., 1. );
            DrawLinkThree( );
      glPopMatrix( );
}
```

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

You will need to create each M[$_{i/G}$] matrix separately using **GLM** Matrix class methods.

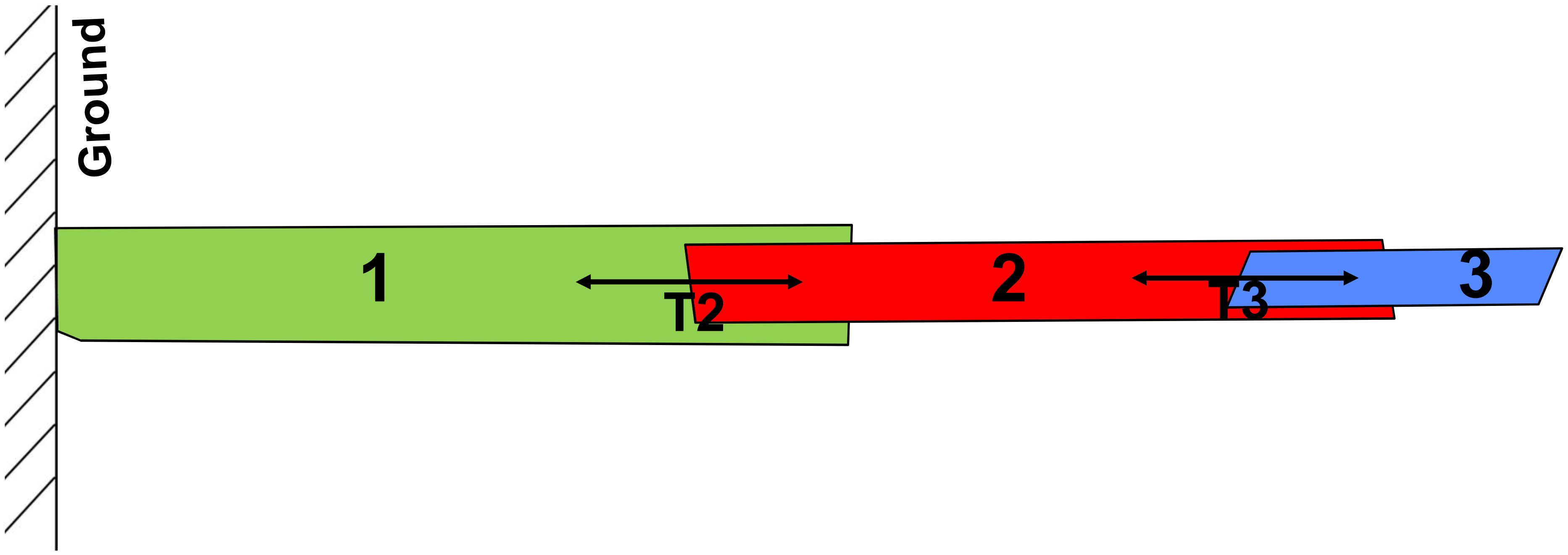
$$[M_{1/G}] = [T_{1/G}] * [R_{\theta 1}]$$

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

$$[M_{3/G}] = [T_{1/G}] * [R_{\theta 1}] * [T_{2/1}] * [R_{\theta 2}] * [T_{3/2}] * [R_{\theta 3}]$$

What If They Are *Sliding* Connections, Not Rotation Connections?

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



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