Robótica: Algoritmo de Denavit–Hartenberg. Caso de estudio SSRMS

por Eliseo Rivera

En el estudio de la Robótica, existe un algoritmo, llamado algoritmo de **Denavit–Hartenberg**, que nos ayuda a establecer los sistemas de referencia para cada uno de los eslabones con los que cuenta el robot.

Caso de estudio ***Space Station Remote Manipulator System*** **(SSRMS)**

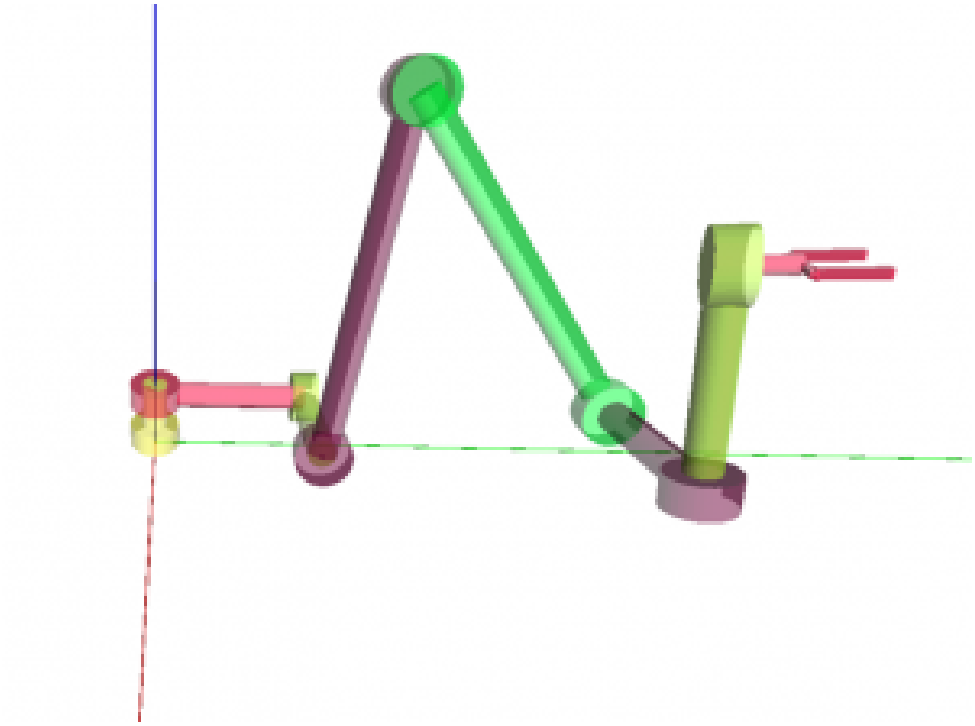

Figura 1. Configuración SSRMS

1. ALGORITMO

Paso 0.

Determinar el número de eslabones y el número de articulaciones. En nuestro caso se tiene que el número de eslabones es n+1, con n=7 y el número de articulaciones es n; por lo tanto hay 8 eslabones en este ejemplo. Para los eslabones, la numeración comienza en 0, el eslabón 0 es la base y el eslabón n=7 es el efector final. Las articulaciones comienzan a numerarse en 1.

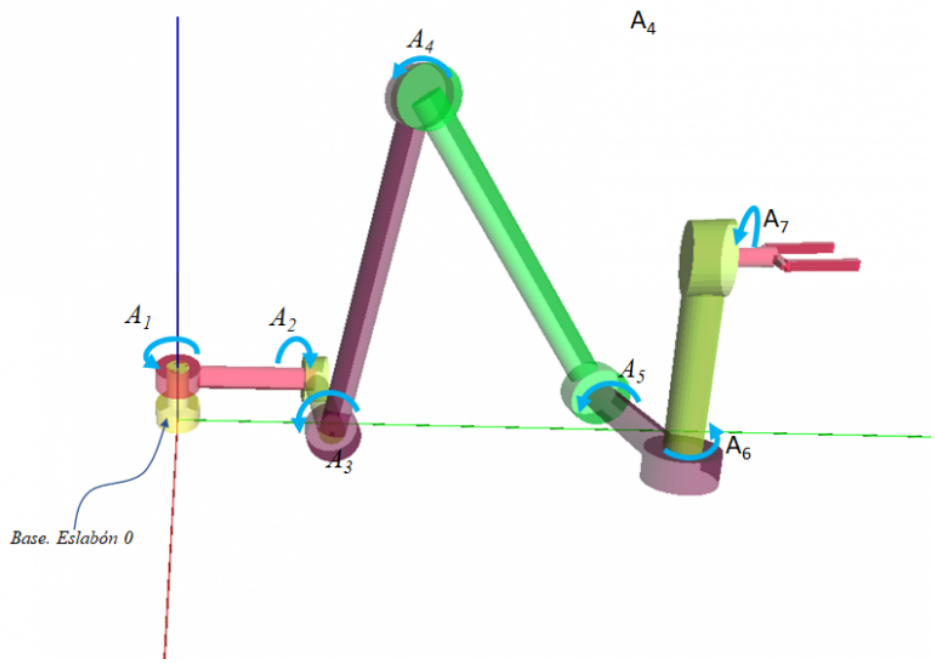

Figura 0. Idenficación de las articulaciones del robot. Siempre hay un eslabón más que el número de articulaciones.

Caso especial. Base (eslabón 0)

Determinar la dirección del eje $\mathbf{z_0}$.

El eje $\mathbf{z_0}$ se escoge de tal forma que este alineado (es decir además de paralelo de estar en la misma línea) con el eje de la articulación $\mathbf{A_1}$(figura 3), el origen del sistema de referencia $\mathbf{B_0}$ (base) se sitúa en cualquier punto del eje $\mathbf{z_0}$.

Los ejes $\mathbf{x_0}$, $\mathbf{y_0}$, $\mathbf{z_0}$ del sistema de referencia $\mathbf{B_0}$ situado en el eslabón $\mathbf{0}$ (base) son fijos (no rotan), se escogen de tal manera que sea un sistema que **obedece a la regla de la mano derecha** (figura 2).

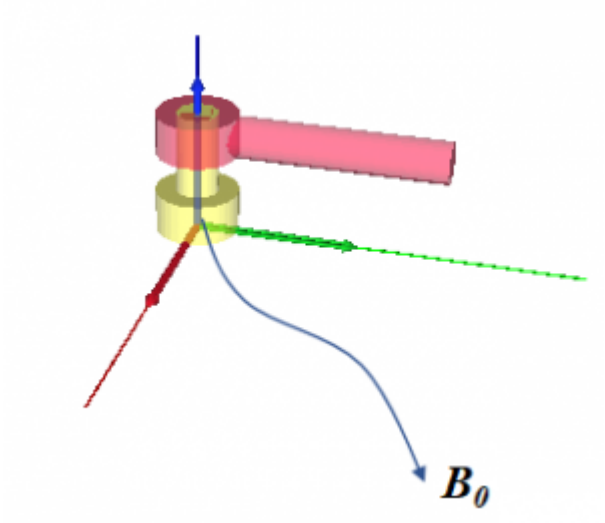

Figura 2. Eje z de eslabón 0 (base) que coincide con la dirección de la siguiente articulación

Figura 3. Articulación A1

Caso especial. Efector final (eslabón n)

Para el último eslabón la elección del sistema de referencia $\mathbf{B_n}$ el eje $\mathbf{x_n}$ **debe ser perpendicular al eje** $\mathbf{z_{n-1}}$; si la articulación es revoluta el eje $\mathbf{z_n}$ está alineado (coincide) con el eje $\mathbf{z_{n-1}}$ (ver figura 8).

Paso 1.

Para cada eslabón i=1,2,3...n-1 (en este ejemplo n-1=6, el i=0 es la base y para el efector final i=7, véase paso 0) hay tres pasos a realizar para elegir la dirección $\mathbf{z_i}$ y la dirección $\mathbf{x_i}$, con ello el eje $\mathbf{y_i}$ se elige simplemente de tal forma que el sistema de referencia $\mathbf{B_i}$ sea un sistema que obedece a la regla de la mano derecha (dextrógiro).

Determinar la dirección de los ejes $\mathbf{z_i}$ con i=1,2,3...n-1.

El eje $\mathbf{z_i}$ se escoge de tal forma que esté alineado (**en la misma línea**) con el eje de la articulación $\mathbf{A_{i+1}}$.

Cada eje $\mathbf{z_i}$ está montado sobre el eslabón *i.*

Para el eslabón 1, según el robot SSRMS de ejemplo en estudio, a continuación se muestra la configuración del eslabón 1 con el eslabón 2, aún sin representar el eje $\mathbf{z_1}$.

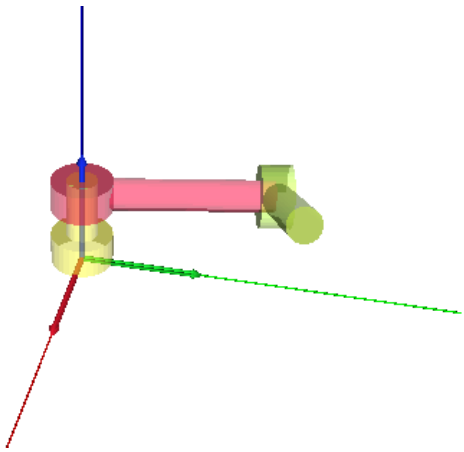

Figura 4. Aún no se ha especificado la dirección del eje z1 montando en el eslabón 1.

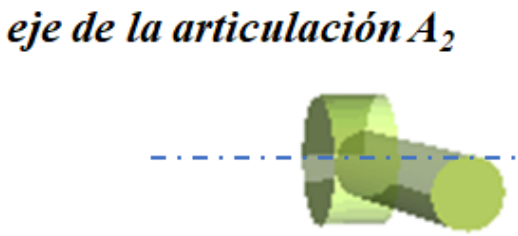

Figura 5. Eslabón 2

De acuerdo con la figura 5, el eje z del eslabón 1 queda como se muestra en la figura 6, es decir, de tal manera que al ensamblar el robot el eje $\mathbf{z_1}$ esté alineado con el eje de la articulación $\mathbf{A_2}$

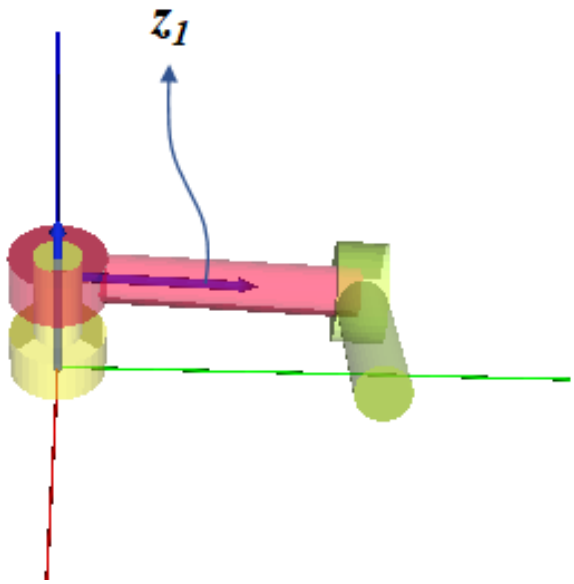

Figura 6. Ilustración de la elección correcta del eje z del eslabón 1.

Continuando de esta manera el conjunto de ejes $\mathbf{z_i}$ con i=0,1,2,3...n-1, se ilustra en la figura 7.

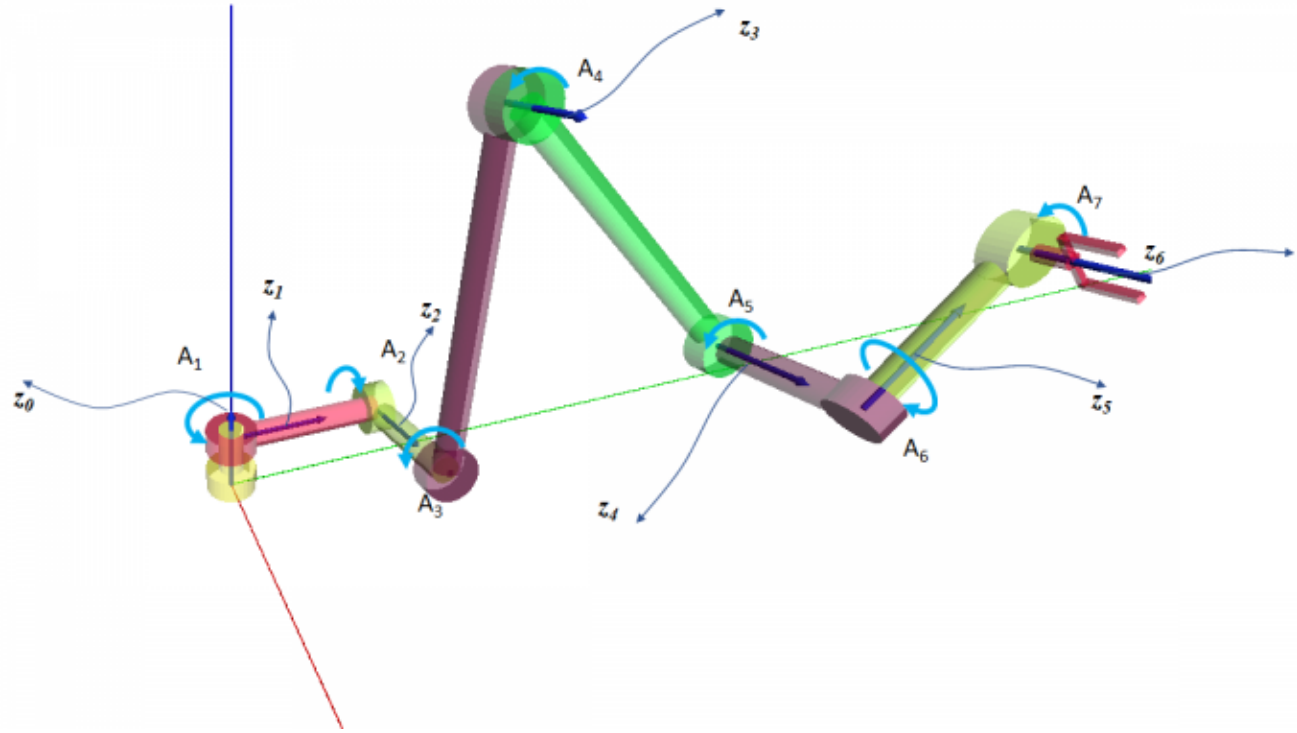

Figura 7. Ejes zi alineado con eje de articulación Ai+1

Paso 2.

Determinar la dirección de los ejes $\mathbf{x_i}$ con i=1,2,3...n-1.

Caso 1.

Si los ejes $\mathbf{z_i}$ y $\mathbf{z_{i-1}}$ se **intersecan** , la dirección del eje $\mathbf{x_i}$ esta dada por la dirección del vector $\mathbf{x_i} = \mathbf{z_i} \times \mathbf{z_{i-1}}$.

Para el caso 1, donde ocurre tal intersercción se coloca el origen del sistema de referencia $\mathbf{B_i}$.

De la figura 8, observe que este caso se cumple para los ejes $\mathbf{x_i}$ con i=1,2,5,n-1.

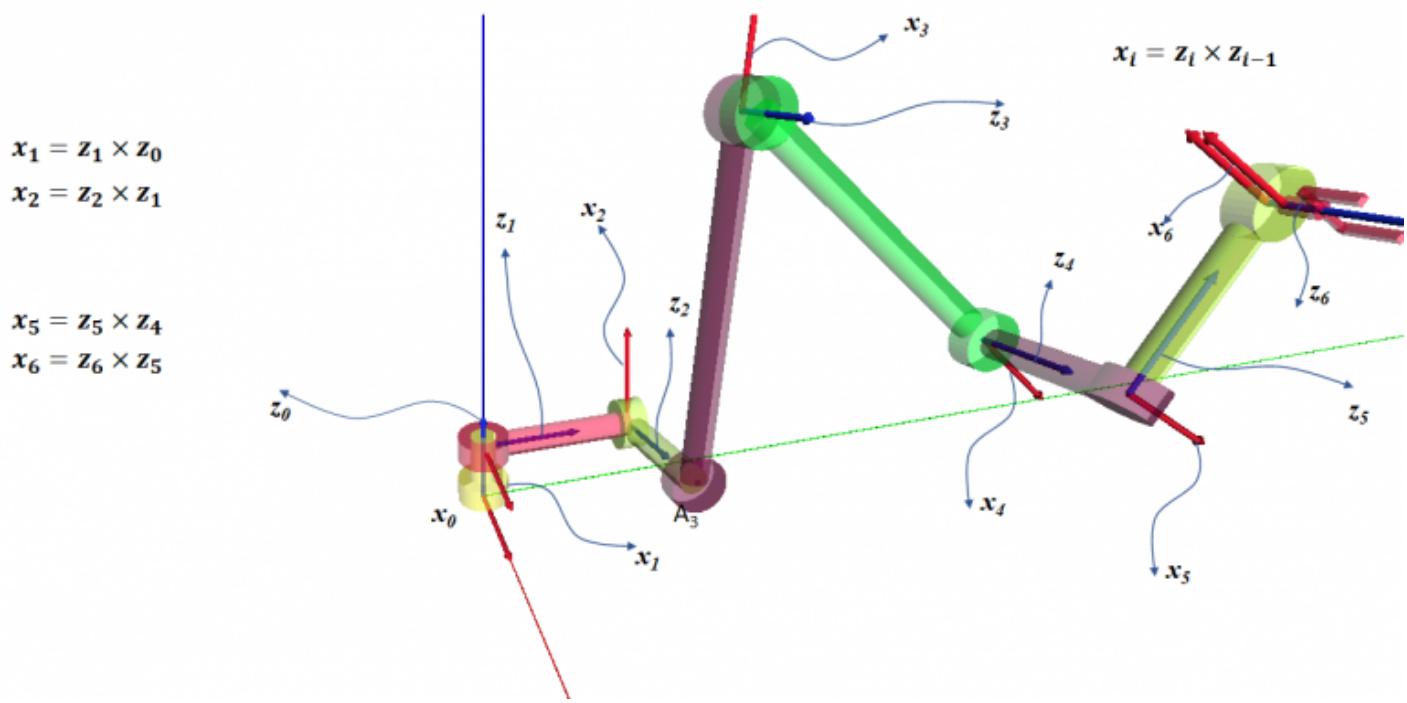

Figura 8.

Caso 2.

Si los ejes $\mathbf{z_i}$ y $\mathbf{z_{i-1}}$ son paralelos, hay un número infinito de normales comunes, escoja alguna de ellas (la más adecuada o cómoda), la dirección del eje $\mathbf{x_i}$ esta dada por la dirección de la normal común que eligió, ésta se dirige del eje $\mathbf{z_{i-1}}$ al eje $\mathbf{z_i}$.

De la figura 8, observe que los ejes $\mathbf{z_2}$ y $\mathbf{z_3}$, así como los ejes $\mathbf{z_3}$ y $\mathbf{z_4}$son paralelos y por lo tanto esto implica que este caso aplica para definir la dirección de los ejes $\mathbf{x_3}$ y $\mathbf{x_4}$.

Caso 3.

Si los ejes $\mathbf{z_i}$ y $\mathbf{z_{i-1}}$ no son paralelos ni se intersecan , la dirección del eje $\mathbf{x_i}$ esta dada por la dirección de la normal común entre dichos ejes, ésta se dirige del eje $\mathbf{z_{i-1}}$ al eje $\mathbf{z_i}$, en el robot SSRMS no ocurre este caso.

Para el caso 2 y 3 el origen del sistema de referencia $\mathbf{B_i}$ se elige donde ocurre la intersección de la normal común de los ejes $\mathbf{z_i}$ y $\mathbf{z_{i-1}}$ con el eje de la articulación $\mathbf{A_{i+1}}$.

En la figura 9 se ilustra la ubicación de los orígenes para cada uno de los sistemas de referencias $\mathbf{B_i}$ con i=1,2,...,n-1; observe que el origen $\mathbf{B_3}$ se ubica en la intersección del eje de la articulación $\mathbf{A_4}$ con la normal común entre el eje $\mathbf{z_2}$ y $\mathbf{z_3}$. Análogamente, el origen $\mathbf{B_4}$ se ubica en la intersección del eje de la articulación $\mathbf{A_5}$ con la normal común entre el eje $\mathbf{z_3}$ y $\mathbf{z_4}$.

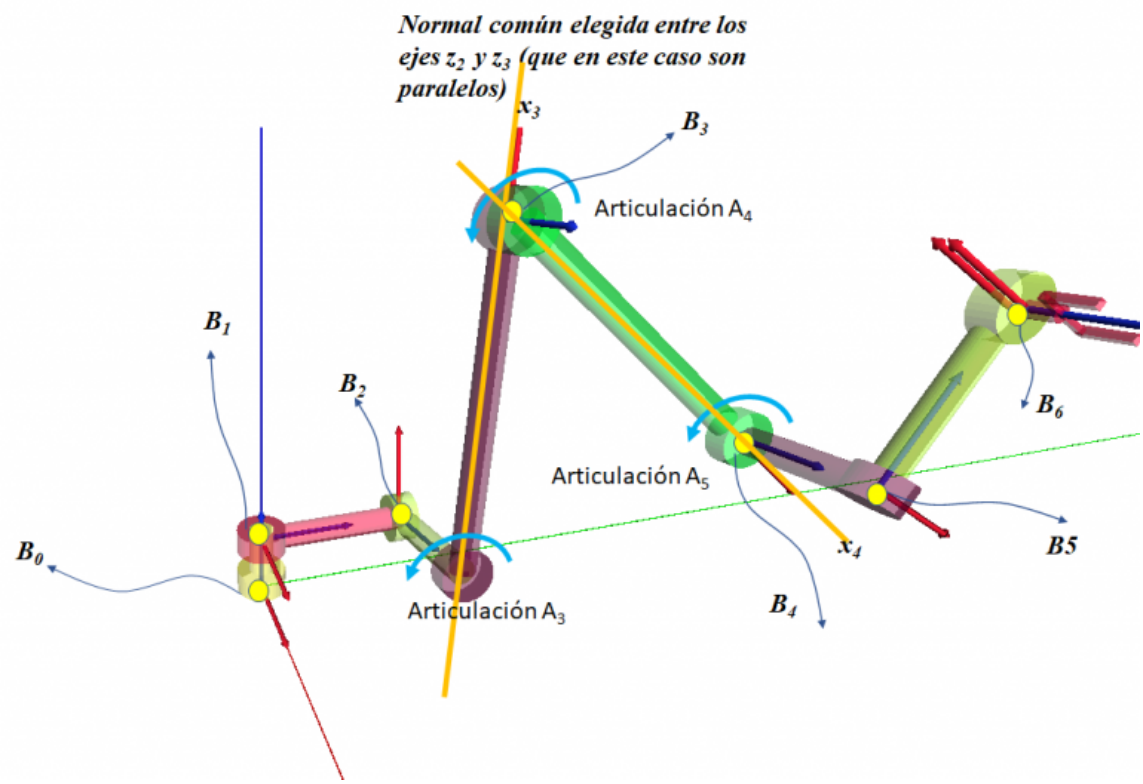

Figura 9. La elección de los orígenes se ha realizado según el caso correspondiente. Observe que no siempre se coloca un origen en cada articulación.

Con la elección de los origenes B_i así como de los ejes z_i *y* x_i *(el eje* y_i *se encuentra fácilmente, ver paso 3) ya se puede comenzar a diseñar y a orientar cada pieza en el sistema de referencia del modelado (sistema de referencia local).*

Durante el diseño de la pieza, ***debe conocerce a priori la dirección del eje de la articulación siguiente***, de esta manera el eje z en el sistema de referencia local (donde se modela la pieza), una vez que se ensambla el robot, **debe estar alineado** con el eje de la articulación siguiente (para la base ver figura 10 y para eslabón 1 ver figura 12).

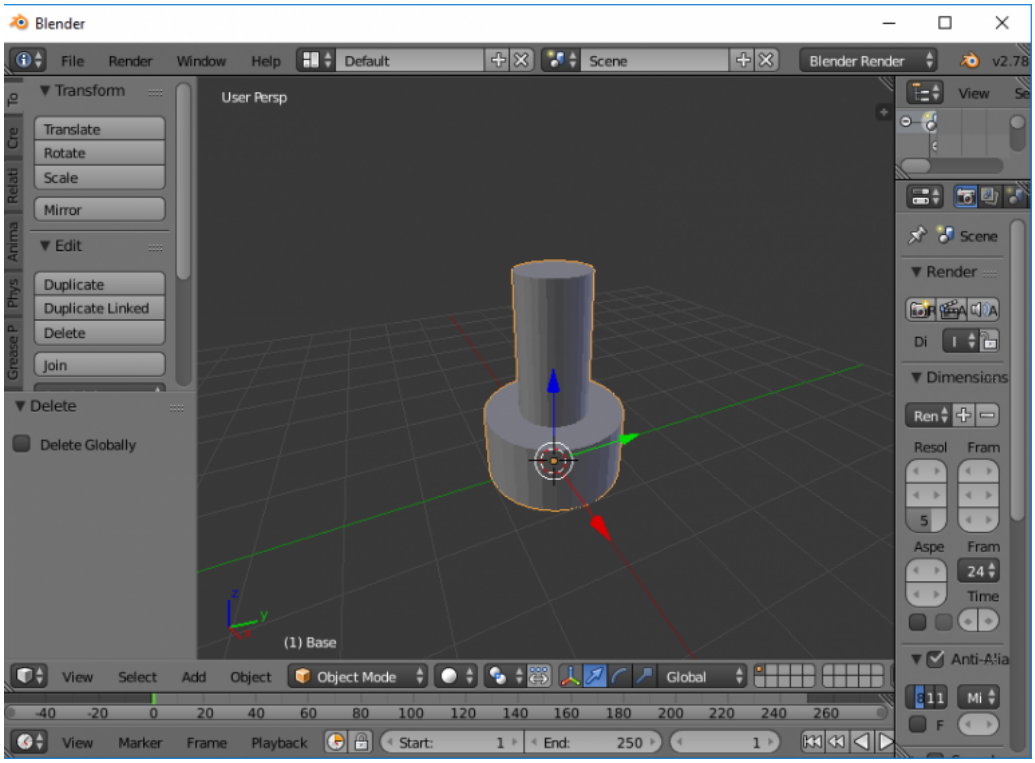

Figura 10. Diseño correcto de la base, observe el sistema de referencia local (rojo eje X, verde eje Y, azul eje Z). Al esamblar el robot el eje z de este eslabón 0 coincide con el eje de la articulación A1

Si durante el diseño del eslabón 1 se tiene la orientación que se muestra en la figura 11, se observa que el eje z, al ensamblar el robot (figura 6), no coincide con el eje de la articulación 2 que se muestra en la figura 5.

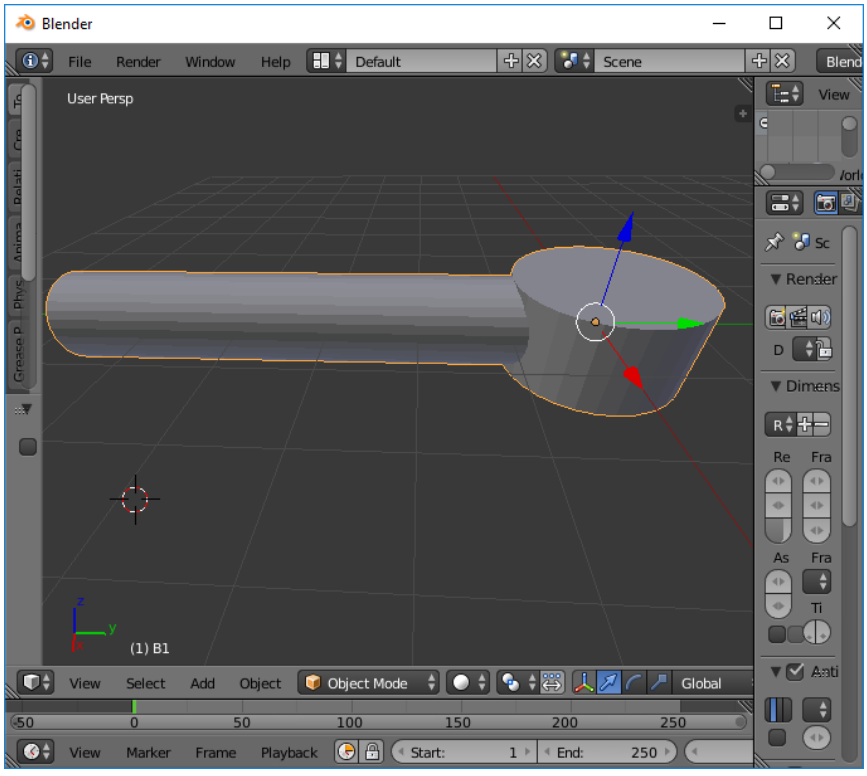

Figura 11. Diseño incorrecto de eslabón 1.

Lo correcto es el diseño que se muestra en la figura 12.

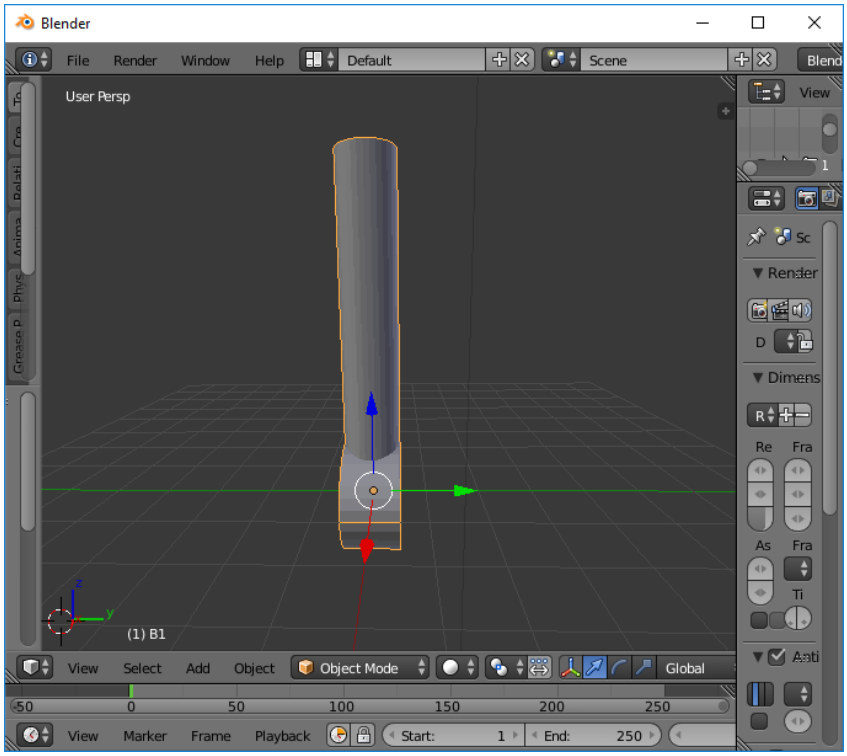

Figura 12. Diseño correcto de eslabón 1. El eje z coincide con el eje de la articulación A2 al ensamblar el robot.

De esta forma, al esamblar el robot, el eje z local del eslabón 1 coincide con la dirección el eje de la articulación $\mathbf{A_2}$ como se ilustra en la figura 6 y el eje x queda como ya se había establecido (figura 8). Continuando este procedimiento para cada eslabon, el sistema de referencia local en cada eslabón debe ser como se muestra en la figura 13.

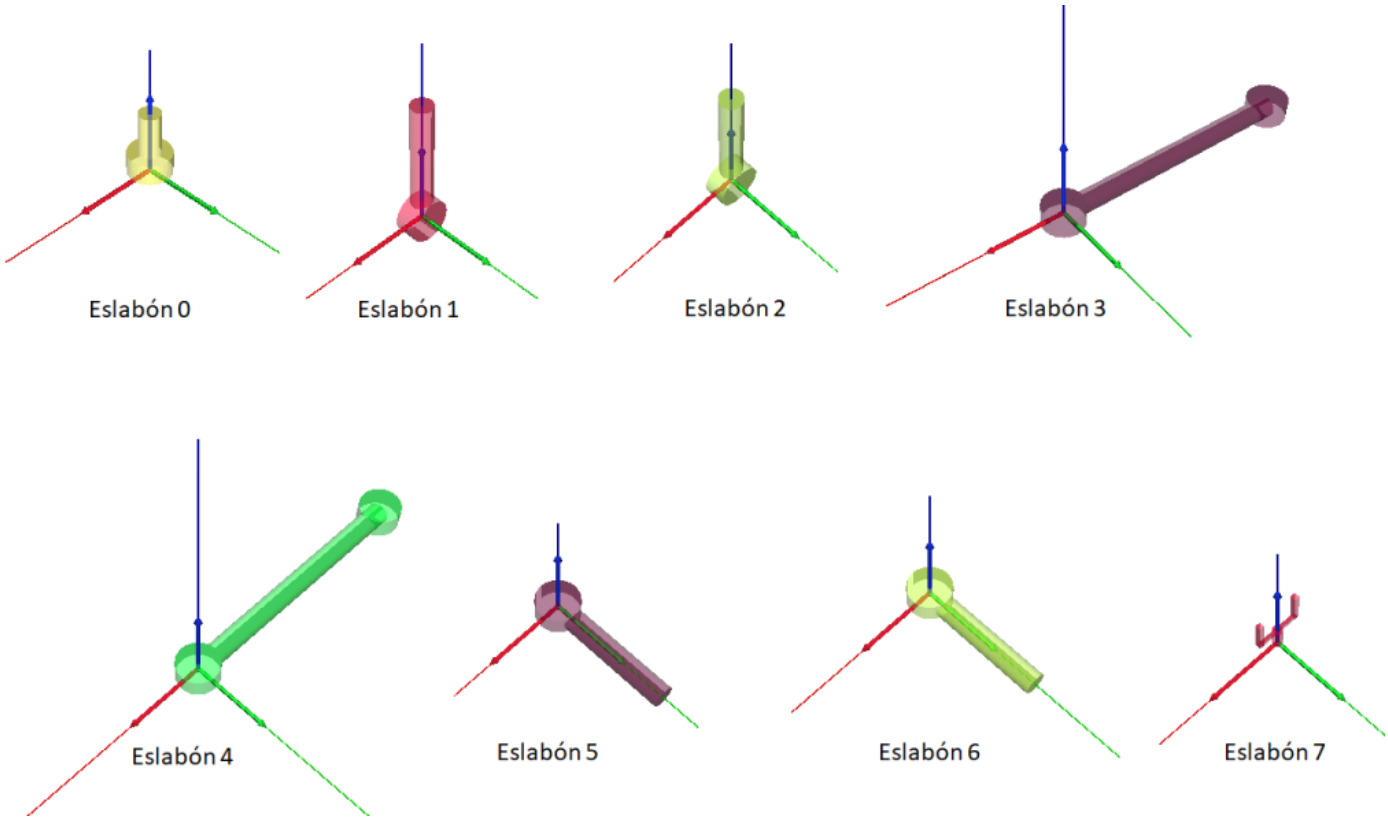

Figura 13. Sistema de referecia local apropiado, intente ensamblar el robot cumpliendo con loas pasos anteriores y note que el resultado es como el de la figura 14. No es necesario hacer todo el ensamble mentalmente, hágalo con cada par

de eslabones contiguos, recuerde que el eje z de cada eslabón es el eje de rotación alrededor del cual rota el siguiente eslabón

Paso 3. $\mathbf{y_i = z_i \times x_i}$.

Determinar la dirección de los ejes $\mathbf{y_i}$.

El eje $\mathbf{y_i}$ *está dado por* $\mathbf{y_i = z_i \times x_i}$.

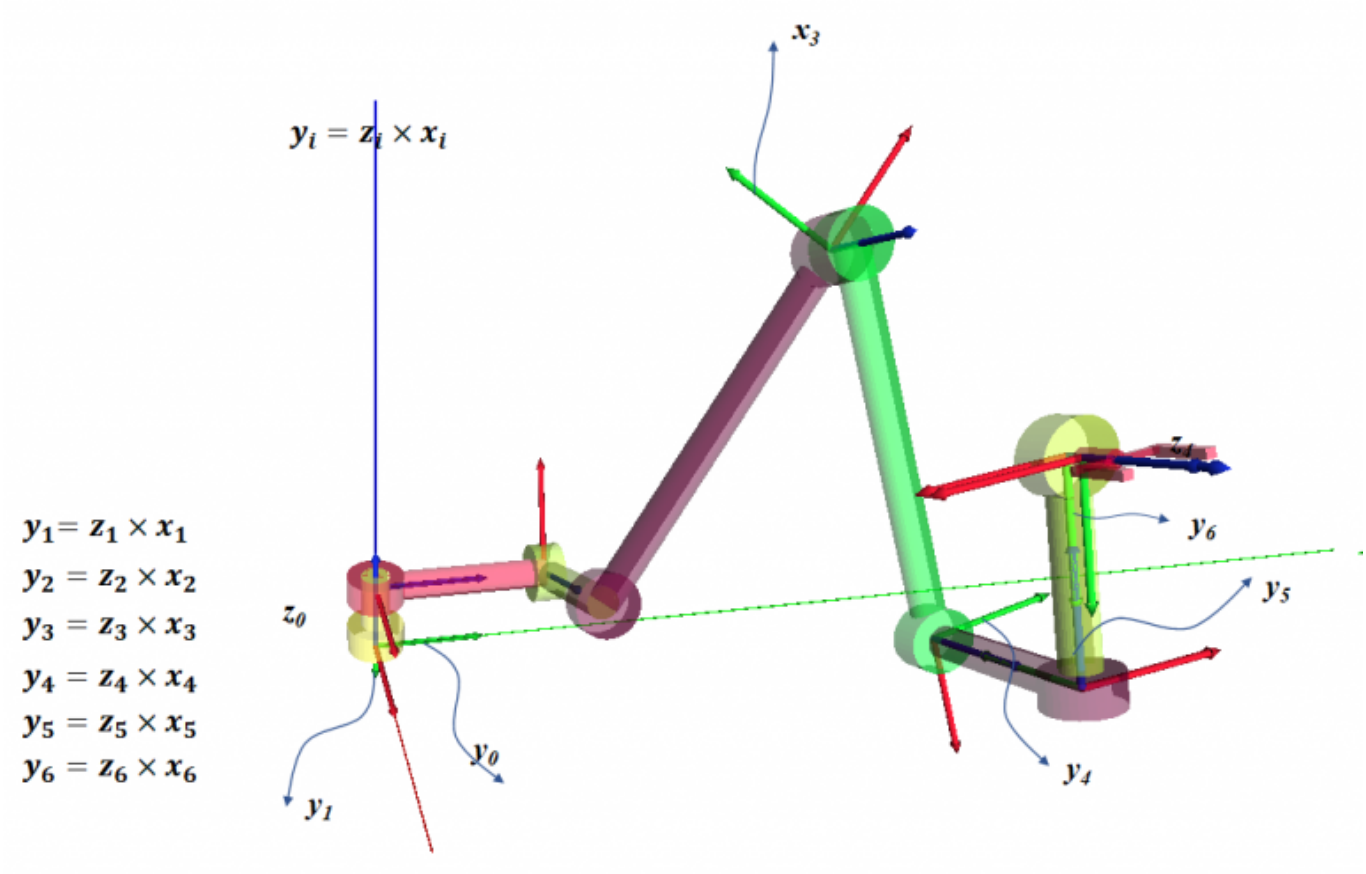

Figura 14. Elección de la dirección de los ejes y para formar sistemas que obdezcan a la regla de la mano derecha.

2. Descripción de las transformaciones

Es natural que durante el diseño se deseen conocer las dimensiones de cada eslabón, así también la forma deben rotar los sistemas de referencia de cada eslabón para que las orientaciones resultantes cumplan con la convención DH; esto es precisamente una tarea que debe realizarse antes de realizar la simulación o animación; existen un conjunto de parámetros que caracterizan la configuración DH de un robot y son precisamente los que permiten realizar un ensamble que cumpla con el algoritmo DH, dichos parámetros que conforman un sistema de coordenadas, llamado sistema de coordenadas DH, son los que debemos medir para poder construir la simulación que ejecute una cinemática directa del robot.

Para referencia se ilustran las medidas (sin unidades) de los eslabones.

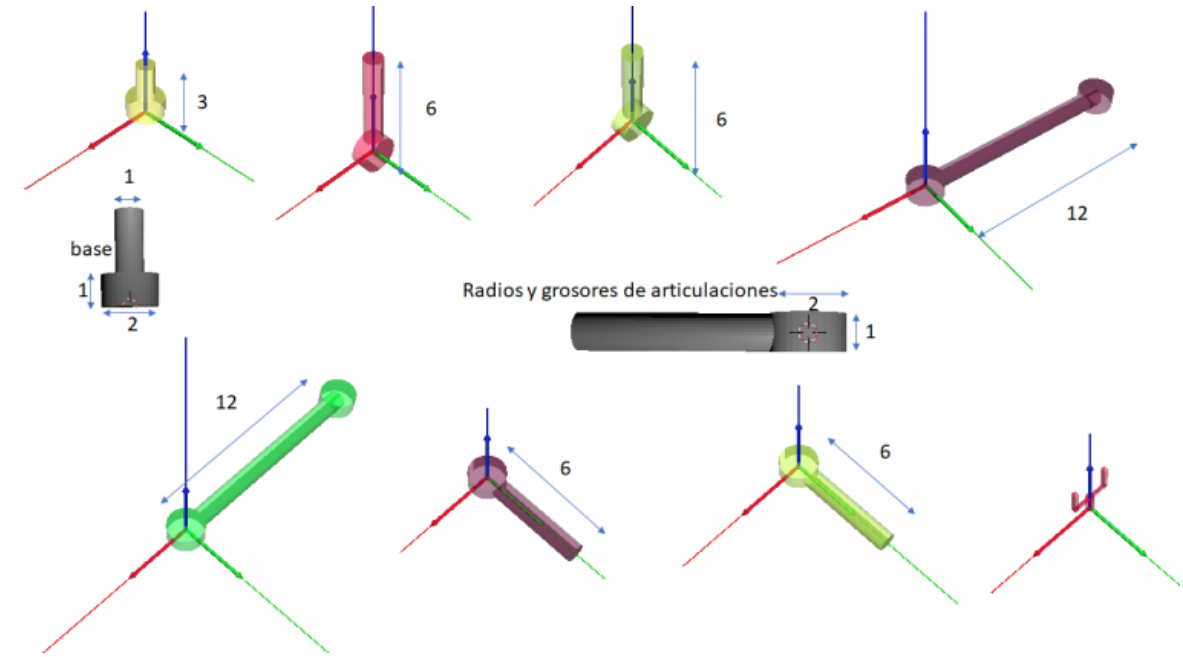

Figura 15. Dimensiones de los eslabones

Para referencia nos apoyamos en la figuras 15 y 16.

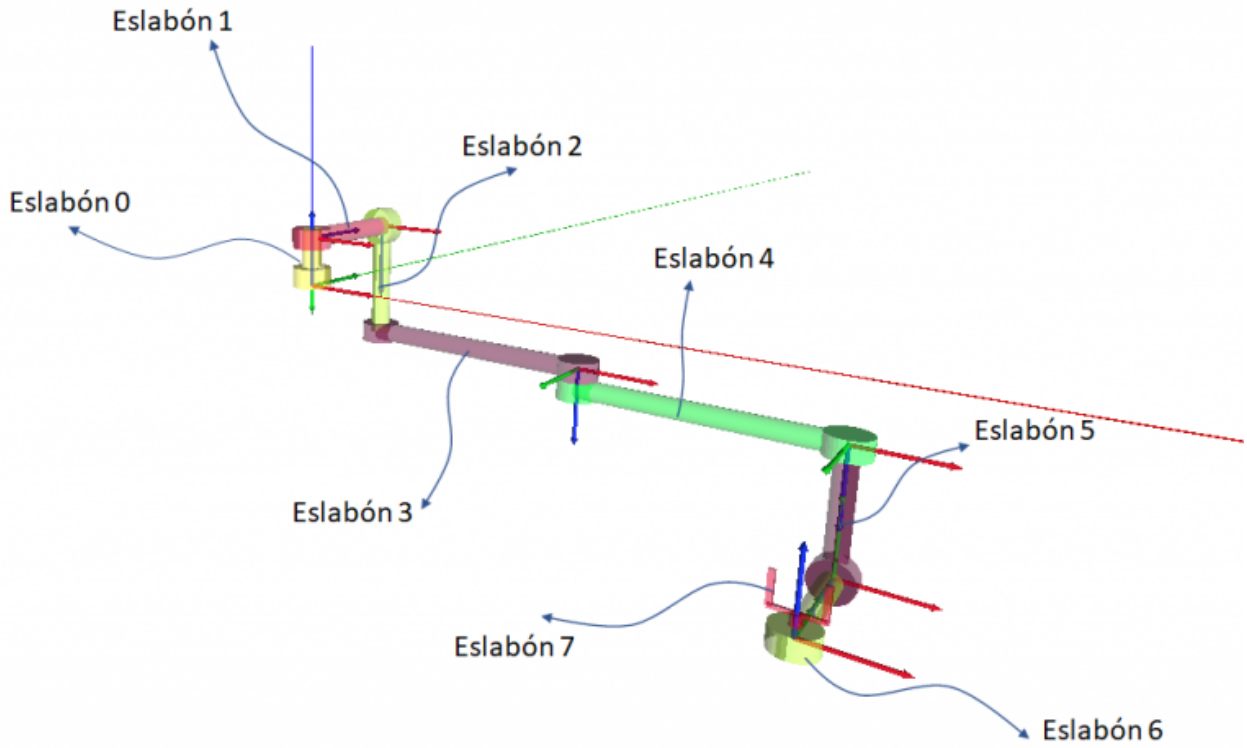

Figura 16. Configuración DH

Para describir las transformaciones que sufre cada eslabón, nos concetraremos en los ejes de cada eslabón **en forma individual y en el eslabón siguiente** que se muestran en la **figura 16** (no en figura 14, ya que esa figura representa al robot cuando ya se han realizado rotaciones alrededor de algunos de los ejes z), analizamos los cambios de orientación entre los sistemas de referencia de un eslabón y su sucesor.**Lo interesante de la convención DH es que el cambio de orientación de un sistema de referencia está dado por el producto de dos transformaciones homogéneas que tienen la forma simplificada**

$$\mathrm{T}_x\,[\alpha, x] = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathrm{T}_z\,[\theta, z)] = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura 17. Forma de las transformaciones homogéneas que experimenta cada eslabón en una configuración DH

y dado que en nuestro caso de estudio hay 8 eslabones la posición $(\mathbf{r})$respecto del sistema de referencia global de un punto $(\mathbf{r_P})$ medido en el sistema de referencia del efector final está dado por

$$\mathbf{T(r)=T_0*T_1*T_2*T_3*T_4*T_5*T_6*T_7(r_P)},$$

donde cada $\mathbf{T_i}$ con i=1,2,3,4,5,6,7 está dada por el producto de dos transformaciones homogéneas que tienen la forma

$$\mathbf{T_i = T_{z_{i-1}}}\,[\theta_i, z] * \mathbf{T_{x_{i-1}}}\,[\alpha_i, x]$$

En código fuente se expresa por

```
void Robot::DefinirTHz(float dtheta, vector3d d){

THz.aij[0][0]=cos(dtheta);
THz.aij[0][1]=-sin(dtheta);
THz.aij[0][2]=0;
THz.aij[0][3]=d.x;

THz.aij[1][0]=sin(dtheta);
THz.aij[1][1]=cos(dtheta);
THz.aij[1][2]=0;
THz.aij[1][3]=d.y;

THz.aij[2][0]=0;
THz.aij[2][1]=0;
THz.aij[2][2]=1;
THz.aij[2][3]=d.z;

THz.aij[3][0]=0;
THz.aij[3][1]=0;
THz.aij[3][2]=0;
THz.aij[3][3]=1;

}
```

```
void Robot::DefinirTHx(float dtheta, vector3d d){

THx.aij[0][0]=1;
THx.aij[0][1]=0;
THx.aij[0][2]=0;
THx.aij[0][3]=d.x;

THx.aij[1][0]=0;
THx.aij[1][1]=cos(dtheta);
THx.aij[1][2]=-sin(dtheta);
THx.aij[1][3]=d.y;

THx.aij[2][0]=0;
THx.aij[2][1]=sin(dtheta);
THx.aij[2][2]=cos(dtheta);
THx.aij[2][3]=d.z;

THx.aij[3][0]=0;
THx.aij[3][1]=0;
THx.aij[3][2]=0;
THx.aij[3][3]=1;

}
```

.

A continuación se enuncia la transformación que sufre cada eslabón para cumplir con la convención DH.

Se sabe que la base (**eslabón 0**) no sufre ninguna rotación ni traslación, en otras palabras experimenta la transformación identidad.

Eslabón 1. Sufrió una rotación de -90 (el signo es porque es contrario a las manecillas del reloj) alrededor del eje $\mathbf{x_0}$ y además una traslación de 2.5 unidades en la dirección del eje $\mathbf{z_0}$.

Por lo tanto la transformación que sufre el sistema de referencia $\mathbf{B_1}$ se expresa por $\mathbf{T}_{\mathbf{x_0}}[-90, 0]$, si al resultado, es decir el sistema $\mathbf{B_1}$ reorientado, lo deseamos rotar respecto del eje $\mathbf{z_0}$ un ángulo θ, precisamente para tener control de articulación $\mathbf{A_1}$, la transformación a aplicar es $\mathbf{T}_{\mathbf{z_0}}[\theta, 2.5]$, de esta forma cambiando el ángulo θ se observará una rotación del sistema $\mathbf{B_1}$ alrededor del eje $\mathbf{z_0}$.

La transformación total para el sistema el sistema $\mathbf{B_1}$ es por tanto

$$\mathbf{T_1}=\mathbf{T}_{\mathbf{z_0}}[\theta, 2.5]* \mathbf{T}_{\mathbf{x_0}}[-90, 0].$$

Y a nivel de código se expresa:

```
AplicarTHz(0,{0,0,2.5}); //b1
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b1
THList.push_back(THx);
```

Aunque el código muestra que $\theta = 0$ en las líneas

```
case 97:
```

```
Miclase->SSRMS.theta1=Miclase->SSRMS.theta1+dtheta;
Miclase->SSRMS.AplicarTHz(Miclase->SSRMS.theta1, {0,0,2.5}); //b1
Miclase->SSRMS.THList[2]=Miclase->SSRMS.THz;
```

se ve claramente que la matriz $\mathbf{T}_{\mathbf{z_1}}[\theta, 2.5]$ se está modificando (ya que cambia la variable theta1) al presionar en el teclado de la pc el número 1 (su código es el número 97), el número 2 en THList[2] se refiere a la matriz tercera de la lista de matrices agregadas a la memoria.

Eslabón 2. Sufrió una rotación de -90 alrededor del eje $\mathbf{x_1}$ y además una traslación de 6 unidades en la dirección del eje $\mathbf{z_1}$.

Por lo tanto la transformación que sufre el sistema de referencia $\mathbf{B_2}$ se expresa por $\mathbf{T}_{\mathbf{x_1}}[-90, 0]$, en forma similar, si deseamos rotar el sistema resultante $\mathbf{B_2}$ respecto del eje $\mathbf{z_1}$ un ángulo θ, precisamente para tener control de la articulación $\mathbf{A_2}$, la transformación a aplicar es $\mathbf{T}_{\mathbf{z_1}}[\theta, 6]$, de esta forma cambiando el ángulo θ se observará una rotación del sistema $\mathbf{B_2}$ alrededor del eje $\mathbf{z_1}$.

La transformación total para el sistema $\mathbf{B_2}$ es por tanto

$$\mathbf{T_2}=\mathbf{T_{z_1}}[\theta,6]*\mathbf{T_{x_1}}[-90,0].$$

Y a nivel de código se expresa:

```
AplicarTHz(0,{0,0,6}); //b2
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b2
THList.push_back(THx);
```

Aunque el código muestra que $\theta = 0$ en las líneas

```
case 100:

 Miclase->SSRMS.theta2=Miclase->SSRMS.theta2+dtheta;
 Miclase->SSRMS.AplicarTHz( Miclase->SSRMS.theta2, {0,0,6}); //b2
 Miclase->SSRMS.THList[4]=Miclase->SSRMS.THz;
```

se ve claramente que la matriz $\mathbf{T_{z_2}}[\theta,6]$ se está modificando (ya que cambia la variable theta2) al presionar en el teclado de la pc el número 4 (su código es el número 100), el número 4 en THList[4] se refiere a la matriz quinta de la lista de matrices agregadas a la memoria.

Eslabón 3. Únicamente sufrió una traslación al punto 12i+0j+6k respecto del sistema de referencia $\mathbf{B_2}$. Es por ello que los ejes $\mathbf{z_3}$ y $\mathbf{z_2}$ son paralelos.

La transformación total para el sistema $\mathbf{B_3}$ es por tanto

$$\mathbf{T_3}=\mathbf{T_{z_2}}[\theta,6]*\mathbf{T_{x_2}}[0,12].$$

Y a nivel de código se expresa:

```
AplicarTHz(0,{0,0,6}); //b3
THList.push_back(THz);
AplicarTHx(0,{12,0,0}); //b3
THList.push_back(THx);
```

Aunque el código muestra que $\theta = 0$ en las líneas

```
case 103:

 Miclase->SSRMS.theta3=Miclase->SSRMS.theta3+dtheta;
 Miclase->SSRMS.AplicarTHz( Miclase->SSRMS.theta3, {0,0,6});
 Miclase->SSRMS.THList[6]=Miclase->SSRMS.THz;
```

se ve claramente que la matriz $\mathbf{T}_{z_3}[\theta, 6]$ se está modificando al presionar en el teclado de la pc el número 7 (su código es el número 103), el número 6 en THList[6] se refiere a la matriz séptima de la lista de matrices agregadas a la memoria.

Eslabón 4. Únicamente sufrió una traslación al punto 12i+0j+1k respecto del sistema de referencia $\mathbf{B_3}$. Es por ello que los ejes $\mathbf{z_4}$ y $\mathbf{z_3}$ son paralelos.

La transformación total para el sistema $\mathbf{B_4}$ a aplicar es por tanto

$$\mathbf{T_4}=\mathbf{T}_{z_3}[\theta, 1]* \mathbf{T}_{x_3}[0, 12].$$

Y a nivel de código se expresa:

```
AplicarTHz(0,{0,0,1}); //b4
THList.push_back(THz);
AplicarTHx(0,{12,0,0}); //b4
THList.push_back(THx);
```

La explicación para el código $\theta = 0$ que se muestra es análoga

```
case 'R':
Miclase->SSRMS.theta4=Miclase->SSRMS.theta4+dtheta;
Miclase->SSRMS.AplicarTHz( Miclase->SSRMS.theta4, {0,0,1}); //b4
Miclase->SSRMS.THList[8]=Miclase->SSRMS.THz;
```

Eslabón 5. Sufrió una rotación de -90 alrededor del eje $\mathbf{x_4}$ y además una traslación de 6 unidades en la dirección del eje $\mathbf{z_4}$ (punto 0i+0j+6k).

La transformación total para el sistema $\mathbf{B_5}$ es por tanto

$$\mathbf{T_5}=\mathbf{T}_{z_4}[\theta, 6]* \mathbf{T}_{x_4}[-90, 0].$$

Y a nivel de código se expresa:

```
AplicarTHz(0,{0,0,6}); //b5
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b5
THList.push_back(THx);
```

La explicación para el código $\theta = 0$ que se muestra es análoga

```
case 'F':
Miclase->SSRMS.theta5=Miclase->SSRMS.theta5+dtheta;
Miclase->SSRMS.AplicarTHz( Miclase->SSRMS.theta5, {0,0,6}); //b5
Miclase->SSRMS.THList[10]=Miclase->SSRMS.THz;
break;
```

Eslabón 6. Sufrió una rotación de -90 alrededor del eje $\mathbf{x_5}$ y además una traslación de 6 unidades en la dirección del eje $\mathbf{z_5}$ (punto 0i+0j+6k).

La transformación total para el sistema $\mathbf{B_6}$ es por tanto

$$\mathbf{T_6}=\mathbf{T}_{\mathbf{z_5}}[\theta, 6]* \mathbf{T}_{\mathbf{x_5}}[-90, 0].$$

Y a nivel de código se expresa:

```
AplicarTHz(0,{0,0,6}); //b6
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b6
THList.push_back(THx);
```

La explicación para el código $\theta = 0$ que se muestra es análoga

```
case 'V':
 Miclase->SSRMS.theta6=Miclase->SSRMS.theta6+dtheta;
 Miclase->SSRMS.AplicarTHz( Miclase->SSRMS.theta6, {0,0,6}); //b6
 Miclase->SSRMS.THList[12]=Miclase->SSRMS.THz;
 break;
```

Eslabón 7. No sufre ni rotación ni traslación.

La transformación total para el sistema $\mathbf{B_7}$ es por tanto

$$\mathbf{T_7}=\mathbf{T}_{\mathbf{z_6}}[\theta, 0]* \mathbf{T}_{\mathbf{x_6}}[0, 0].$$

Y a nivel de código se expresa:

```
AplicarTHz(0,{0,0,0.0}); //gripe
THList.push_back(THz);
AplicarTHx(0,{0,0,0}); //gripe
THList.push_back(THx);
```

La explicación para el código $\theta = 0$ que se muestra es análoga

```
case 'G':
 Miclase->SSRMS.theta7=Miclase->SSRMS.theta7-dtheta;
 Miclase->SSRMS.AplicarTHz( Miclase->SSRMS.theta7, {0,0,0}); //b7, efector final
 Miclase->SSRMS.THList[14]=Miclase->SSRMS.THz;
 break;
```

La transformación total, que finalmente da como resultado la posición en el sistema de referencia global de un punto que se mide en el sistema de referencia local del efector final es

$$\mathbf{T}=\mathbf{I}*\mathbf{T}_{z_0}[\theta,2.5]*\mathbf{T}_{x_0}[-90,0]*\mathbf{T}_{z_1}[\theta,6]*\mathbf{T}_{x_1}[-90,0]*\mathbf{T}_{z_2}[\theta,6]*\mathbf{T}_{x_2}[0,12]*\mathbf{T}_{z_3}[\theta,1]*\mathbf{T}_{x_3}[0,12]*\mathbf{T}_{z_4}[\theta,6]*\mathbf{T}_{x_4}[-90,0]*\mathbf{T}_{z_5}[\theta,6]*\mathbf{T}_{x_5}[-90,0]*\mathbf{T}_{z_6}[\theta,0]*\mathbf{T}_{x_6}[0,0],$$

la matriz $\mathbf{I}$ se refiera a la transformación que sufre la base, la cual es la matriz identidad. En el código fuente este se resultado se expresa como sigue, se encuentra en el método void Robot::renderizar()

```
for (int m=0;m<modelos.size();m++){
      model=modelos[m];
      TH=TH* THList[2*m+0]*THList[2*m+1];
```

El método para el control del robot completo puede revisarlo completo en el método LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam) en el archivo principal main.cpp.

Ejercicio. Realice la configuración DH para el robot PUMA

Puede descargar gratuitamente el software de animación en tiempo real disponible gratuitamente https://github.com/EliseoRivera/Denavit-Hartenberg. Si desea saber más sobre el robot PUMA puede consultarlo en https://es.wikipedia.org/wiki/PUMA_(robot). El diseño de las piezas se realizará en Blender o algún otro software de diseño. Los puntos a calificar son

1.- Bosquejo del robot (a mano) para establecer inmediatamente el número de articulaciones y la configuración DH. Seguir los pasos 0, 1,2,3.

2.- Diseño de los modelos de tal forma que cumplan con lo explicado en las figuras 10, 11,12 y 13. De esta manera no debe modificar el modelo3D dentro del código, es decir, su código en el método inicializar deberá ser similar al siguiente

```
void Robot::inicializar(){
base=new modelo3D();
b1=new modelo3D();
b2=new modelo3D();
b3=new modelo3D();
b4=new modelo3D();
b5=new modelo3D();
b6=new modelo3D();
gripe=new modelo3D();

base->leer("base.STL");
b1->leer("b1.STL");
b2->leer("b2.STL");
b3->leer("b3.STL");
b4->leer("b4.STL");
b5->leer("b5.STL");
b6->leer("b6.STL");
```

```
gripe->leer("gripe.STL");

modelos.push_back(base);
modelos.push_back(b1);
modelos.push_back(b2);
modelos.push_back(b3);
modelos.push_back(b4);
modelos.push_back(b5);
modelos.push_back(b6);
modelos.push_back(gripe);

}
```

3.- Descripción de las transformaciones. Como se explica en la sección DESCRIPCIÓN DE LAS TRANSFORMACIONES, tanto conceptualmente y modificar el código fuente para renderizar su modelo. Solo debe haber transformaciones de la forma

$$\mathbf{T_i} = \mathbf{T}_{\mathbf{z_{i-1}}}[\theta_i, z] * \mathbf{T}_{\mathbf{x_{i-1}}}[\alpha_i, x],$$

que pueden introducirse en el método

```
void Robot::configurarTH(){
AplicarTHz(0,{0,0,0}); //base
THList.push_back(THz);
AplicarTHx(0,{0,0,0}); //base
THList.push_back(THx);

AplicarTHz(0,{0,0,2.5}); //b1
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b1
THList.push_back(THx);

AplicarTHz(0,{0,0,6}); //b2
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b2
THList.push_back(THx);

AplicarTHz(0,{0,0,6}); //b3
THList.push_back(THz);
AplicarTHx(0,{12,0,0}); //b3
THList.push_back(THx);

AplicarTHz(0,{0,0,1}); //b4
THList.push_back(THz);
AplicarTHx(0,{12,0,0}); //b4
THList.push_back(THx);

AplicarTHz(0,{0,0,6}); //b5
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b5
THList.push_back(THx);

AplicarTHz(0,{0,0,6}); //b6
THList.push_back(THz);
AplicarTHx(-90,{0,0,0}); //b6
```

```
THList.push_back(THx);

AplicarTHz(0,{0,0,0.0}); //gripe
THList.push_back(THz);
AplicarTHx(0,{0,0,0}); //gripe
THList.push_back(THx);

}
```

así también debe cambiar el método del control del robot dependiendo del número de eslabones, las modificaciones únicamente deben ser de la forma

```
Miclase->SSRMS.theta4=Miclase->SSRMS.theta4+dtheta;
Miclase->SSRMS.AplicarTHz( Miclase->SSRMS.theta4, {0,0,1}); //b4
Miclase->SSRMS.THList[8]=Miclase->SSRMS.THz;
```

Éxito!