EJERCICIO 3. GEOMETRIA

OBJETIVO:

El objetivo de la práctica es familiarizarse con los conceptos básicos de geometría utilizando Matlab. En primer lugar, el alumno implementará las matrices de transformación básicas para posteriormente comprobar su funcionamiento estableciendo relaciones entre diferentes sistemas de coordenadas. El alumno también implementará las relaciones que existen entre ángulos de Euler tipo 3, quaterniones y matrices de rotación. Finalmente, se programará una animación que permita entender mejor los conceptos básicos de geometría.

ACTIVIDAD 1:

En primer lugar se programarán las matrices de rotación básicas que permitan establecer la relación de traslación y rotación básicas respecto de cada uno de los ejes entre dos sistemas de coordenadas. Cada una de estas funciones se implementará en un fichero diferente teniendo en cuenta que el nombre de los ficheros corresponde con el nombre de la función. Se trata de generar una toolbox de funciones a partir de los ficheros ya disponibles en Poliformat. El alumno debe arrancar Matlab y sustituir el código de las siguientes funciones, por un código que calcule la matriz de transformación correspondiente:

```
function T=rotaX(angulo)
% T = rotaX(angulo)
% Obtiene la matriz de rotación 'T' correspondiente a un giro definido por
% la variable 'angulo' respecto del eje X. 'angulo' esta medido en radianes

function T=rotaY(angulo)
% T = rotaY(angulo)
% Obtiene la matriz de rotación 'T' correspondiente a un giro definido por
% la variable 'angulo' respecto del eje Y. 'angulo' esta medido en radianes

function T=rotaZ(angulo)
% T = rotaZ(angulo)
% Obtiene la matriz de rotación 'T' correspondiente a un giro definido por
% la variable 'angulo' respecto del eje Z. 'angulo' esta medido en radianes

function T=tras(tx, ty, tz)
% T=tras(tx, ty, tz)
% Obtiene la matriz de traslación 'T' correspondiente a un desplazamiento
% definido por las coordenadas de las variable 'tx', 'ty', 'tz'
```

Para validar el correcto funcionamiento de las funciones programadas el alumno utilizará el script de *`main.m`*. Ejecutando este script en Matlab se genera un entorno 3D como el de la figura 1. En él se muestra un sistema de coordendas con origen de coordenadas en el punto (0, 0, 0) y otro con origen de coordenadas en el punto (-500, 500, 500). De la misma forma el alumno puede añadir nuevos sistemas de coordenadas para comprobar la funcionalidad de las funciones programadas. Para mostrar un sistema de coordenadas se utiliza la función dibujarSC.

```
function dibujarSC(matrizTransformacion, etiqueta, tamanyoEjes, grosorLinea)
% dibujarSC(matrizTransformacion, etiqueta, tamanyoEjes, grosorLinea)
%
% dibuja un sistema de coordenadas con la localización (posición y
% orientación) definida por la variable 'matrizTransformacion'
%
% etiqueta -> nombre que aparece en los subindices de los ejes
% tamanyoEjes -> longitud de los ejes del sistema de coordenadas
% grosorLinea -> grosor de los ejes del sistema de coordenadas
```

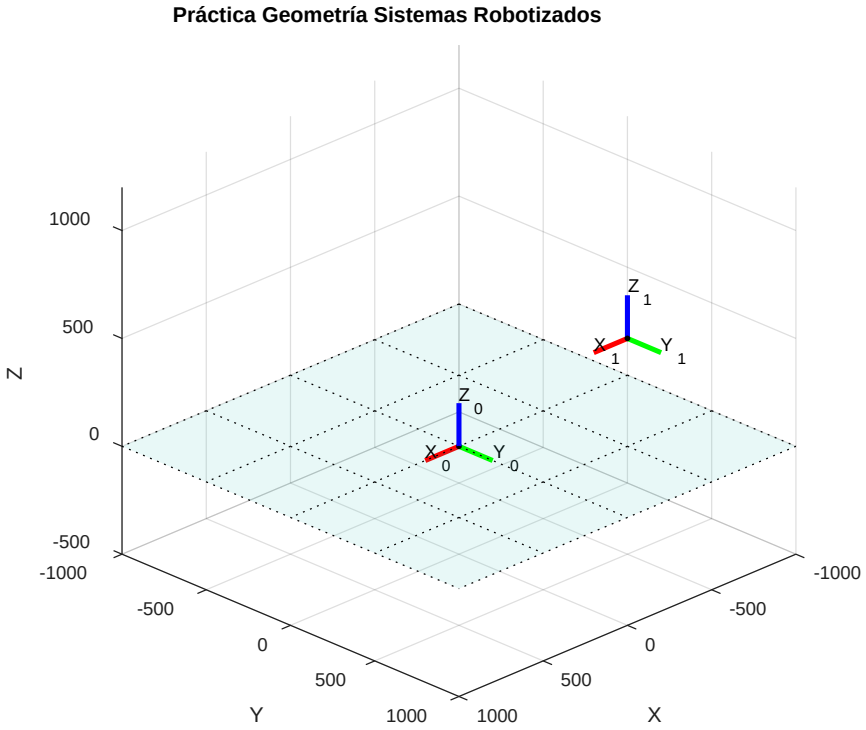

Figura 1. Entorno gráfico de simulación para visualizar resultados

ACTIVIDAD 2:

Con las funciones programadas en la actividad 1 se calcularán las matrices de transformación que permitan localizar los 6 sistemas de coordenadas que se muestran en la figura 2. El resultado debe ser parecido al mostrado en la figura 3 si se coloca el sistema de coordenadas 0 en el punto $[400, -350, 0]^T$. Para localizar los 6 sistemas de coordenadas en el espacio se deben calcular las siguientes matrices de trasformación. cálculo de los

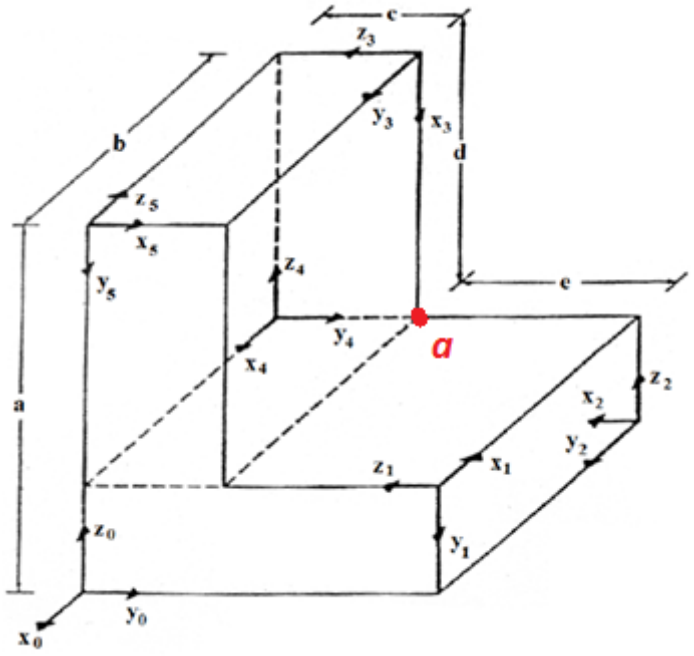

Figura 2. Localización de los ejes de la actividad 2. a = 700, b = 800, c = 300, d = 500, e = 400,

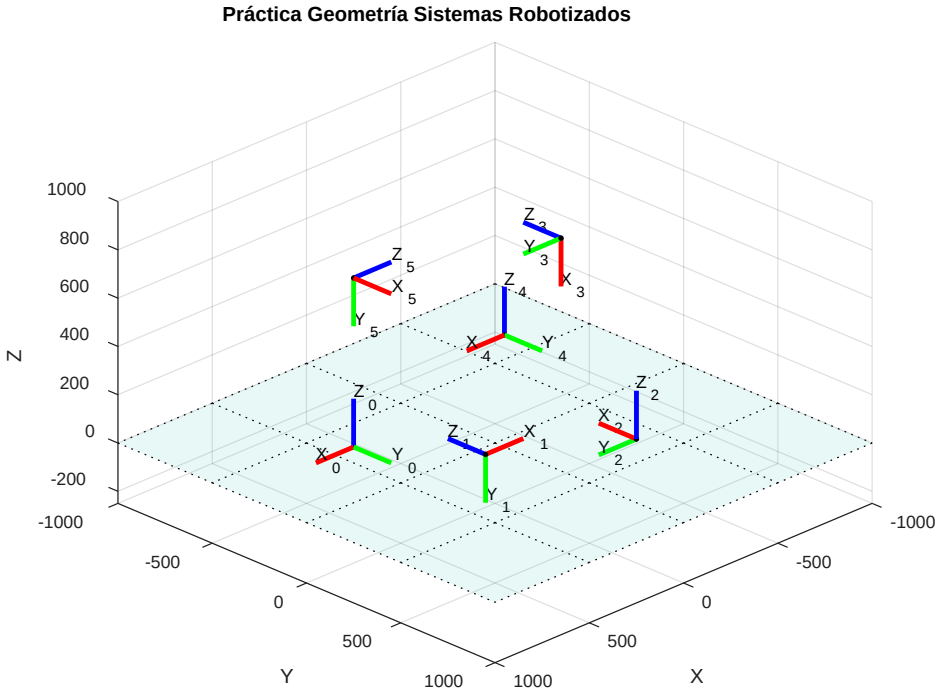

Figura 3. Localización de los ejes de la actividad 2 en el entorno Matlab.

QUESTIONES:

1.1. ¿Qué movimientos has realizado para calcular para las matrices de transformación ${}^0T_1$, ${}^0T_2$, ${}^0T_3$, ${}^0T_4$, ${}^0T_5$?

1.2. Comprueba si es cierto que ${}^0T_1 \cdot {}^1T_2 = {}^0T_5 \cdot {}^5T_4 \cdot {}^4T_3 \cdot {}^3T_2$ ¿es cierta esta igualdad? Justifica tu respuesta

1.3. Conociendo la matriz de transformación ${}^0T_2$ ¿cuáles son las coordenadas del punto ***a*** respecto del sistema de coordenadas 2 (***a₂***), si las coordenadas del punto ***a*** respecto del sistema de coordenadas 0 son $\boldsymbol{a_0} = [-b, c, a-d]^T$? Justifica tu respuesta

1.4. Conociendo la matriz de transformación ${}^0T_2$ ¿cuáles son las coordenadas del vector director $\boldsymbol{x_2}$ del sistema de coordenadas 2 respecto del sistema de coordenadas 0?

1.5. Conociendo la matriz de transformación ${}^0T_2$ ¿cuáles son las coordenadas del vector director $\boldsymbol{z_0}$ del sistema de coordenadas 0 respecto del sistema de coordenadas 2?

ACTIVIDAD 3:

En esta actividad se van a programar las funciones que permitirán obtener la matriz de transformación a partir de los angulos de Euler tipo III y los quaterniones, y viceversa. Se debe implementar y comprobar el código de las siguientes funciones:

```
function R = euler2rot(rX, rY, rZ)
% R = euler2rot(rX, rY, rZ)
%  Convierte los angulos de Euler III (roll, pitch, yaw) expresados en radianes
%  en una matriz de rotation (3x3)
% R -> matriz de rotación
% rX-> giro en el eje X
% rY-> giro en el eje Y
% rZ-> giro en el eje Z
```

```
function [rX, rY, rZ] = rot2euler(R)
% [rX, rY, rZ] = rot2euler(R)
%  Convierte una matriz de rotation (3x3) en los angulos de Euler III (roll,
%  pitch, yaw) expresados en radianes
% rX-> giro en el eje X
% rY-> giro en el eje Y
% rZ-> giro en el eje Z
% R -> matriz de rotación

function q = rot2quat(R)
% q = rot2quat(R)
%  Convierte una matriz de rotation (3x3) en el quaternión correspondiente
% rX-> giro en el eje X
% rY-> giro en el eje Y
% rZ-> giro en el eje Z
% R -> matriz de rotación

function R = quat2rot(q)
% R = quat2rot(q)
%  Convierte un quaternion en una matriz de rotation (3x3)
% R -> matriz de rotación
% q -> quatenion con 4 elementos

function quaternion = aVect2quat(vector, angulo)
% quaternion = aVect2quat(vector, angulo)
% Cálculo del quaternión a parti de vector del eje de giro y el ángulo de giro en
% radianes
% quaternion = aVect2quat(vector, angulo)
% vector -> vector de 3 elementos con las coordenadas del vector director del
% giro
% angulo -> ángulo de giro
% quaternion -> vector de 4 elementos con el quaternion equivalente

function [vector, angulo] = quat2aVect(quaternion)
% [vector, angulo] = quat2aVect(quaternion)
% Cálculo del de vector del eje de giro y el ángulo de giro en radianes a partir
% del quaternión
% quaternion -> vector de 4 elementos con el quaternion equivalente
% vector -> vector de 3 elementos con las coordenadas del vector director del
% giro
% angulo -> ángulo de giro en radianes
```

QUESTIONES:

1.6. ¿Cuál es el vector director y el ángulo de giro que permite hacer el cambio del sistema de coordenadas 0 al 2 con un unico eje de giro? Justifica tu respuesta

1.7. Cuales son los ángulos de Euler III que permiten realizar el giro del sistema de coordenadas 0 al 1 de la figura 4.

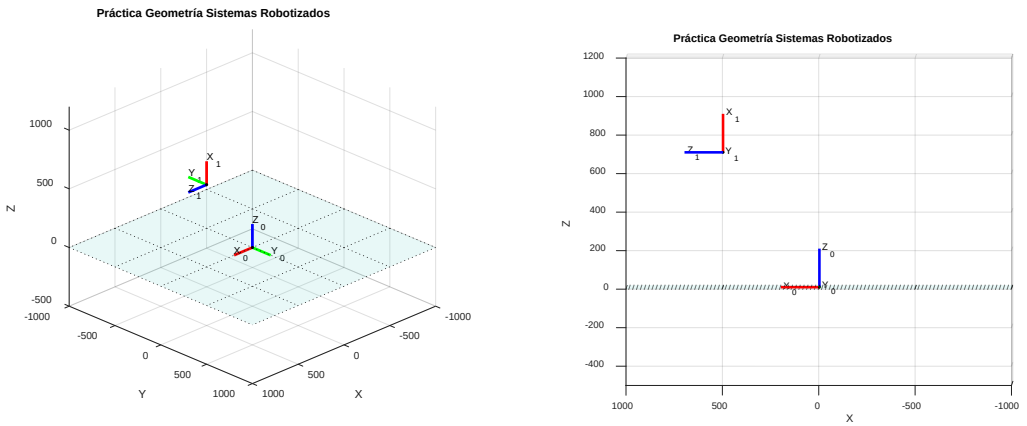

Figura 4. Localización de los ejes para la cuestión. 1.7

ACTIVIDAD 4 (opcional):

En esta actividad se pretende animar un conjunto de puntos que representa un coche de rally. El movimiento que se pretende conseguir es que siga un 8 tal y como se puede ver el video colgado en poliformat. Ejecutando el script del fichero `actividad4.m` aparece en entorno de simulación junto con el coche realizando un pequeño movimiento lineal hacia adelante.

Con la función `crearEntorno(figura, amplitudX, amplitudY, amplitudZ)`se programa el entorno en una figura de graficos de Matlab con las amplitudes en los tres ejes definidas por las variables *amplitudX*, *amplitudY*, *amplitudZ*.
Con la función `[objetoH, nubePuntos] = cargarObjeto(nombreFichero, color, sistemaCoordenadas, tamanyoEjes)` se carga una nube de puntos de un fichero, los cuales se pintan de color determinado por la variable *color* que contiene los valores RGB de 0 a1 en un vector de 3 posiciones. La variable *sistemaCoordenadas*, es un booleano con valores '*true*' o '*false*' que permite definir si se pintan el sistema de coordenadas asociado al objeto, la variable *tamanyoEjes* define la longitud de los ejes que se pintan. Como resultado de la función se obtiene un manejador del objeto y una matriz con la nube de puntos que define el objeto. Para cambiar la localización del objeto en el entorno se debe multiplicar la nube de puntos por una matriz de transformación que contenga el movimiento deseado y utilizar la función `actualizarObjeto(objetoH, T, nubePuntos)`. Esta función necesita el manejador del objeto, la matriz de transformación T para redibujar el sistema de coordenadas en el caso de mostrarse y la nueva nube de puntos.

Cambiar los parámetros *color* y *sistemaCoordenadas*, y visualizar los efectos que produce. Programar el movimiento a partir de las funciones matemáticas que definen el movimiento. Tener en cuenta que el coche se levanta en las curvas y siempre está encima de la carretera, tal y como se muestra en la figura 5.

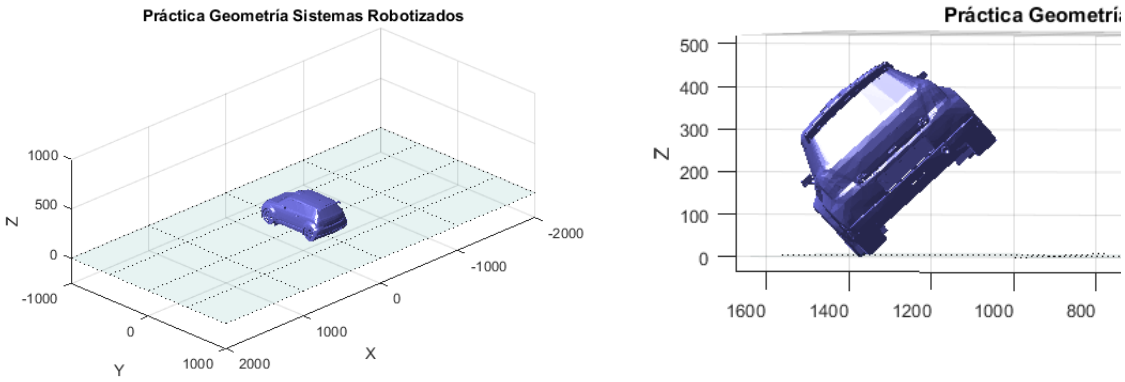

Figura 5. Posición inicial del objeto y en uno de los laterales de la curva. El coche no sobrepasa la línea del suelo