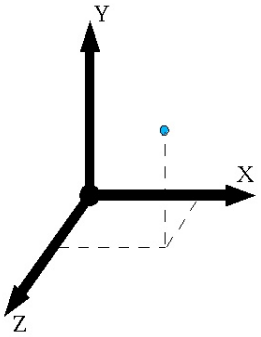
TP Nº 2: Ray Casting

- Para los siguientes ejercicios, recomendamos hacer uso de la librería *vecmath* de *Java* (https://vecmath.dev.java.net/) para la manipulación de vectores y matrices.
- Para el cálculo de las intersecciones entre objetos, recomendamos consultar el siguiente link: http://www.realtimerendering.com/intersections.html.
- Recomendamos utilizar un sistema de coordenadas de mano derecha:



Ejercicio 1

Definir las clases primitivas necesarias para:

a) La manipulación de un rayo en el espacio. Recordar que un rayo tiene un punto de origen y una dirección.
b) La representación de triángulos, cuadriláteros y esferas. Implementar algún algoritmo de intersección de rayos con estas primitivas. Analizar si la representación elegida para los elementos de la escena es la más conveniente. Cambiarla si fuera necesario.
c) Representar la escena, que pueda contener las primitivas mencionadas anteriormente.

Ejercicio 2

Definir una escena que contenga una esfera de radio 4 en el origen de coordenadas, y comprobar la colisión con N rayos que tengan su origen en el punto (0, 0, 10) y que pasen por los puntos (n - (N/2), 0, 0), donde n es el número de ese rayo y N la cantidad total de rayos a disparar. Por cada rayo, mostrar la coordenada más cercana donde intersecta con la esfera (si hay intersección) o un mensaje indicando que no hubo colisión.

Ejercicio 3

Modificar el ejercicio anterior para que los rayos pasen por puntos aleatorios dentro del plano X, Y (o sea, con Z = 0), partiendo desde el punto de origen ya definido.

Ejercicio 4

Definir una escena con N triángulos ubicados en forma aleatoria con todos sus puntos en el plano X-Y, con -10 <= x, y <= 10. Disparar rayos desde el punto (0, 0, 10) hacia todos los puntos del plano X, Y con coordenadas enteras ‘x’ e ‘y’ comprendidas entre [-10, 10]. Mostrar para cada rayo la lista de triángulos con los que intersecta.