Tarea 2

Jorge Alfonso Cárdenas Treviño

Noviembre 2023

Para implementar el algoritmo de Tangent Bug manejamos una clase de obtáculos que cuenta con los métodos

- **insideObstacle:** verifica si un punto dado se encuentra dentro del obstáculo.
- **createVertices:** crea los vértices para visualizar el obstáculo.
- **createIndices:** crea los índices para el orden en el que se tienen que unir los vértices del obstáculo.

Dentro de esta clase tenemos las subclases de triángulo y de círculo. También definimos una clase obstacleWorld que guarda todos los vértices e índices para poder dibujar nuestros obstáculos.

Gráficamente mostramos los objetos de nuestra simulación de la siguiente manera

- **Círculo rojo:** robot.
- **Círculo amarillo:** rango de visión del robot.
- **Círculo gris:** punto al cual el robot se quiere mover.
- **Figuras negras:** obstáculos.
- **Círculo verde:** meta.

Para detectar los obstáculos, nuestro robot recurre a lanzar rayos apoyándose de las funciones **raycast** y **sphereCast** que lanzan un rayo en una dirección dada y en un círculo alrededor de nuestro robot, respectivamente.

El movimiento del robot se rige de la siguiente manera:

1. Si la distancia a la meta es menor a la velocidad de movimiento del robot, se ha llegado a la meta.
2. Si no se está siguiendo la frontera y la dirección a la meta no está obstruida, se mueve hacia la meta.
3. De otro modo, seguir la discontinuidad más cercana a la meta.
4. Si se empieza a alejar de la meta, seguir la frontera hasta que dreach sea menor a dfollowed o que se haya alcanzado el punto inicial en el que se empezó a seguir la frontera.
5. Si se alcanza el punto inicial al seguir la frontera, no hay solución.

Incluimos 3 videos de distintas simulaciones que muestran el funcionamiento del TangentBug.