PRÁCTICA 4

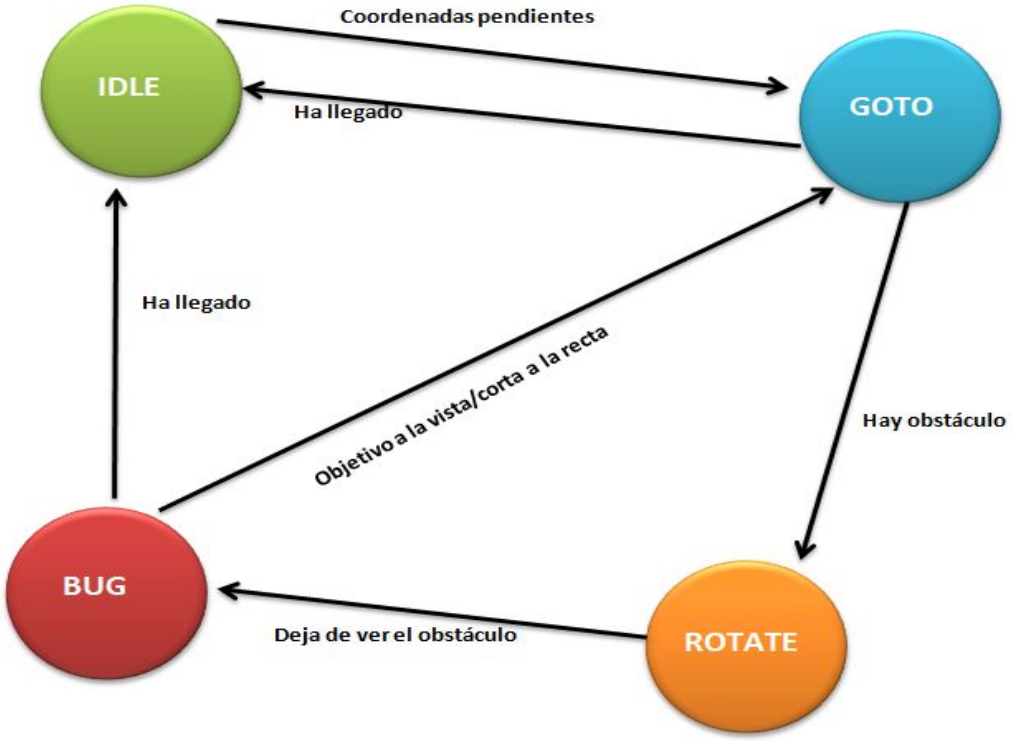
ROBÓTICA

Trabajo realizado por:
Carmen Carrero Hurtado
Andrés Fernández Pérez

INTRODUCCIÓN:

En esta entrega se nos pedía extender el componente de la práctica anterior para que cuando pulsaramos sobre cualquier parte del tablero el robot se moviera hasta allí, esta vez evitando los obstáculos. Para esto último utilizamos el algoritmo bug, inspirando en el movimiento de los insectos. Este algoritmo consiste en que cuando el robot está cerca del objeto, lo rodea siempre en la misma dirección hasta que ve de nuevo el objetivo.

Lo primero que hicimos antes de programar fue crear una máquina de estados sobre el papel, que nos serviría para controlar la lógica del componente. Esta máquina de estados es la siguiente:



CÓDIGO DESARROLLADO:

Tradujimos la máquina de estados del papel al código de la siguiente manera: (hay algunos cambios de estado que se encuentran dentro de los propios métodos, que veremos en detalle más adelante)

```
switch(state) {

     case State::IDLE:

          if (coord.getPendiente())
               state = State::GOTO;
          break;
```

```
        case State::GOTO:
              gotoTarget();
              break;

        case State::BUG:
              bug();
              break;

        case State::ROTATE:
              rotar();
              break;
        }
```

En un principio, el robot se encuentra en estado IDLE hasta que el usuario clicka sobre cualquier zona del mapa. En ese momento, se calcula la recta entre dos puntos (dónde se encuentra el robot al principio y donde tiene que llegar) y se almacena en tres variables llamadas A, B y C, correspondiendo con la ecuación de la recta (Ax + By + C = 0). Esto es importante ya que en un momento dado el robot comprobará si desde la posición actual corta dicha recta, ya que eso significa que se encuentra en la línea hacia las coordenadas objetivo, por lo que lo único que tendrá que hacer será avanzar "siguiendo la línea" y no seguir rodeando el obstáculo.

Volviendo a la máquina de estados, hemos dicho que el robot se encuentra en IDLE y como tiene coordenadas pendientes, la condición del if se cumple y cambia su estado por GOTO, entrando en el método gotoTarget().
En dicho método se hace una primera comprobación para ver si el robot tiene algún obstáculo cerca (nosotros hemos puesto ese umbral en 250 mm de distancia); si esto se cumple, el robot cambia su estado a ROTATE y se para, saliendo también del método. Si no tiene ningún obstáculo próximo el robot avanza hacia las coordenadas objetivas, calculando siempre la distancia hasta las mismas. Finalmente, si la distancia es menor que 250 ha llegado al objetivo, por lo que se para y cambiamos el estado a IDLE, esperando nuevas coordenadas.

Hemos mencionado que si se encuentra con un objeto, cambia el estado a ROTATE. Vamos a suponer que así es por lo que ahora tenemos al robot en dicho estado, entrado en el método correspondiente. Este estado, junto con el estado BUG completan el algoritmo con el nombre del último; el método rotar() lo que hace es que gira el robot sobre sí mismo hasta que deja de ver el obstáculo, y el método bug() lo que hace es que una vez que ha dejado de ver el obstáculo, lo bordea. Nosotros lo hemos programado para que siempre gire hacia la izquierda y si la distancia es mayor o igual que 450, entonces es que ha dejado de ver el objeto, por lo que se para y pasa al estado BUG.

Como hemos comentado antes, el estado BUG lo que hace es rodear el objeto. Cambia de estado a GOTO si se produce una de las dos condiciones, o bien tiene las coordenadas a la vista, o hay un momento que el robot corta la recta mencionada al principio. Si no se cumple ninguna de las dos opciones, el robot avanza rodeando el objeto. El robot para cuando se llega al objetivo.