PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN
IIC2132 – ESTRUCTURA Y REPRESENTACIÓN DE DATOS

Tarea 3: Concurso de navegación

Alumno: GERMÁN VICENCIO L.

30 de Octubre de 2009

ALGORITMO:

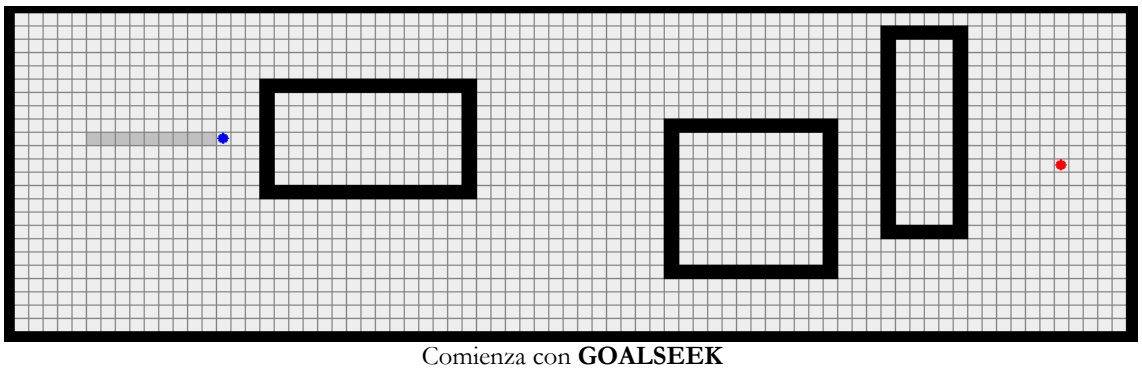
La idea es que el robot tiene dos estados: WALLFOLLOW y GOALSEEK.

Se comienza con WALLFOLLOW, caminando siempre en dirección al goal.

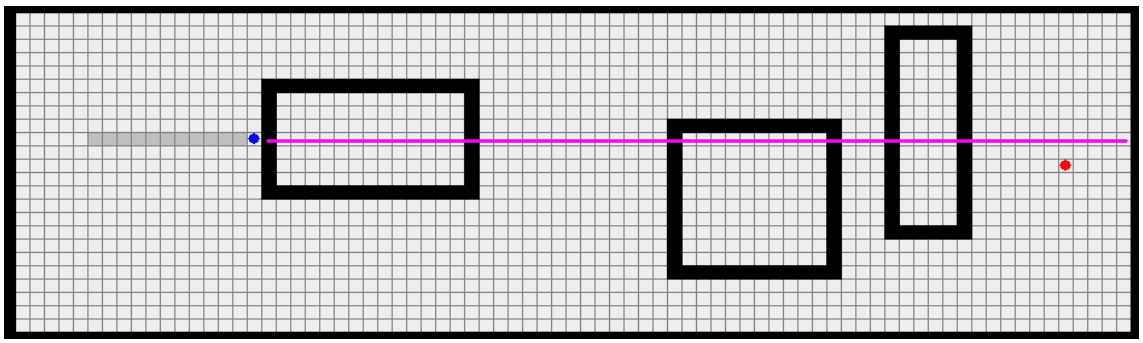
Cuando choca con un obstáculo, se cambia al estado GOALSEEK y se traza[1] una linea recta desde ese punto en dirección al goal.

En este estado el robot rodea el obstáculo[2]. Si se topa nuevamente con la línea trazada, vuelve al estado GOALSEEK.

Ejemplo:



Comienza con **GOALSEEK**

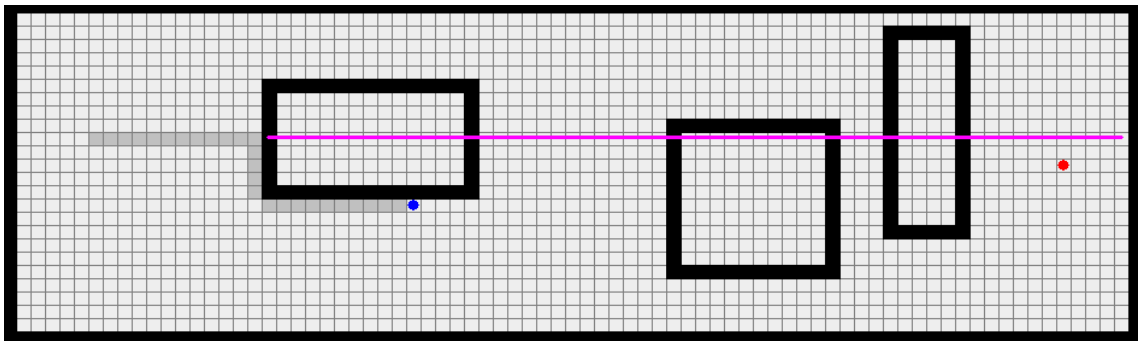


Choca con obstáculo. Cambia a **WALLFOLLOW**.
Traza una linea recta en dirección al goal.

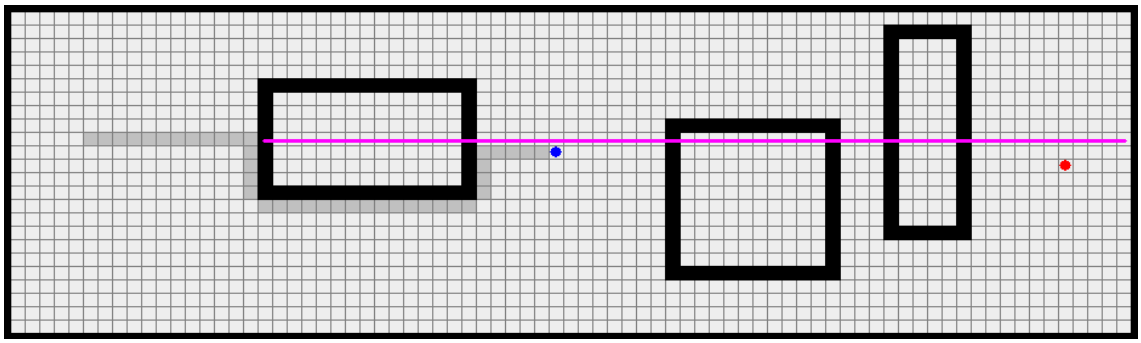
---

[1] Al principio el algoritmo que hice fue "**a partir de la posición del choque, pintar path hasta el final del mapa privado**". Pero con este algoritmo hay un problema (testtucker), cuya solución era hacer el algoritmo así: "**a partir de la posición del choque, pintar path hasta el final del mapa privado solo en las posiciones Unvisited**"

[2] Por la izquierda o derecha dependiendo de dónde está el goal para él.
Si esque el goal está ortogonal (N, S, W, E) no se puede deducir mucho (acá se hace **random** la decisión)
Si es que el goal está al NW o SE, entonces conviene rodear por la izquierda. Análogo para SW y NE.

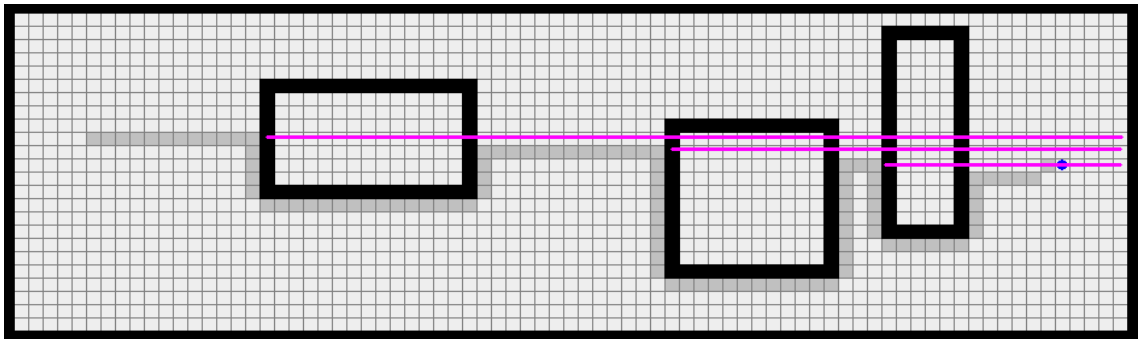


En **WALLFOLLOW** se rodea el obstáculo



Cuando cruza denuevo la línea vuelve al estado **GOALSEEK**

(En realidad, cuando detecta la línea *"adyacente y no obstáculo"*, esto porque el algoritmo para rodear obstáculos que hice se salta las esquinas y podría, entonces, saltarse el goal path)



Repite este proceso hasta llegar al goal (en cada choque redefine la línea).

RESULTADOS:

El algoritmo es completo: Siempre llega al objetivo. Pero varía la eficiencia dependiendo del mapa y de la decisión "rodear por la derecha o por la izquierda", por ejemplo en test4 es muy eficiente, pero en test3 depende de la decisión.

Optimizaciones o ideas que no tuve tiempo de probar:

- En vez de hacer el movimiento random cuando no sabe qué decidir, se podría gastar unos cuantos movimientos en reconocer los dos lados y con eso ver qué dirección tomar, por ejemplo:
  Adelante tengo 3 bloques (diagonal izq, front y diagonal derecha), podría ir a la izquierda y ver a dónde apunta el goalDirection, volver, repetir con la derecha, volver, y finalmente tomar una decisión.
- Ponerle cierta tolerancia de **alejamiento**: Si se está alejando demasiado en el modo WALLFOLLOW, quizás el robot debería devolverse y probar por el otro lado (guardándo un flag "2nd and last try" eso sí, donde ya no se aplique la tolerancia)
  Edit: Al final lo implementé pero, poniendo poca tolerancia (20) ayuda en algunos mapas (por ejemplo en test2), pero empeora otros (testtucker, donde debe alejarse del goal por artos pasos). Asi que para la entrega lo deje con tolerancia 300, cosa que nunca se aplique.

NOTA:
Alguien podría argumentar que cambiarse al modo GOALSEEK **antes** de cruzar denuevo la línea (por ejemplo cuando pueda ir al goal denuevo) sería más rápido, pero eso es fácil de hacerlo fallar.