# Memoria Práctica 3-IAV

Francisco López-Bleda y Manuel Hernández

## Resumen

Si algo demuestra que un IA existe es que interactúa con su entorno, en esta ocasión usaremos lo creado en la práctica 2 pero ahora la máquina evaluará riesgos y beneficios para resolver un crimen, ahora lo importante no es el tiempo que tarda en encontrar el camino, si no que se resuelva el problema de la forma más rápida posible, para no perder tiempo encontrando el camino.

## Enunciado

Las condiciones del taller eran las siguientes

### El agente

- Tiene que ser capaz de vigilar o explorar de forma segura.
- Comienza en la casa, y desconoce como es el mapa.
- Es de noche, solo sabe como es las casillas por donde ha pasado, y las va descubriendo.
- El barro le ayudará a evitar los precipicios.
- La sangre le ayudará a encontrar el cuerpo.
- El cadáver y la sangre le ayudará a encontrar el cuchillo.
- Tiene que encontrar el arma y la ubicación.
- El coste de la solución es el número de pasos.
- Si el detective se cae a un agujero muere.

### El entorno

- Las únicas casillas que pueden coexistir son sangre con barro.
- El arma tiene que estar distancia 2 del cuerpo.
- Por cada barranco sus cuatro casillas vecinas son barro.
- Un botón de reinicio.

### Restricciones

- Separad el agente de Unity, que sea una cosa aparte
- Usad lógica proposicional.
- Reutilizar el pathfinding del 2º taller
- Mostrar métricas internas

## Resultado

En este apartado analizamos y comparamos lo obtenido con el enunciado propuesto, por el orden de las secciones.

## El agente

A falta de un agente hemos hecho 5, que luego explicaremos, primero contaremos las bases del agente.

Nuestros agentes están entrenados de disciplina de catalogar lo que no conocen, es decir, a la hora de medir el riesgo de las casillas fronterizas el riesgo y el beneficio se distribuye por las casillas vecinas que el desconoce, esto es en lo que se basa el agente para evaluar a sus vecinos, respecto a peligro se refiere:

```
if (infoMapa[pos.y, pos.x]._Terreno == Tile.T_Terreno.T_GRAVA)
{
    if (infoMapa[vecino.y, vecino.x]._Terreno == Tile.T_Terreno.T_DESCONOCIDO && !infoMapa[vecino.y, vecino.x].noPrecipicio)
    {
        infoMapa[vecino.y, vecino.x].probPrecipicio += 1000 / nVecinos;
    }
}
else if (infoMapa[pos.y, pos.x]._Terreno == Tile.T_Terreno.T_CESPED)
{
    infoMapa[vecino.y, vecino.x].noPrecipicio = true;
    infoMapa[vecino.y, vecino.x].probPrecipicio = 0;

}
```

Esto en el caso de beneficio:

```
if (infoMapa[pos.y, pos.x]._Contenido == Tile.T_Contenido.C_SANGRE)
{
    if (infoMapa[vecino.y, vecino.x]._Contenido == Tile.T_Contenido.C_DESCONOCIDO)
        infoMapa[vecino.y, vecino.x].probCuerpo -= 500 / nVecinos;
    if (fiambreFound)
    {
        infoMapa[vecino.y, vecino.x].probCuerpo = 0;


    }
}
else if (infoMapa[pos.y, pos.x]._Contenido == Tile.T_Contenido.C_CUCHILLO)
{
    if (infoMapa[vecino.y, vecino.x]._Contenido == Tile.T_Contenido.C_DESCONOCIDO )
        infoMapa[vecino.y, vecino.x].probCuerpo -= 250 / nVecinos;
    if (fiambreFound)
    {
        infoMapa[vecino.y, vecino.x].probCuerpo = 0;


    }
}
```

El número de vecinos es siempre los vecinos que él no conoce. como se puede apreciar el tienen más en cuenta si algo es peligroso, se podría decir que juegan conservador, porque el fracaso no es una opción, el riesgo/beneficio de la casilla viene dado por las suma de probabilidad que haya cuerpo sumado a la probabilidad de que haya un barranco. Nuestro agente va llevando una lista con todas las casillas fronterizas, que va actualizando.

Ahora me dispongo a explicar nuestros agentes.

**Agente Normal**
Este es el clásico, un hombre metódico, que va recorriendo de forma ordenada el mapa sin asumir riesgos, escoge siempre la casilla que menos riesgo tenga pero siempre la más cercana, sin pegarse a los bordes del mapa para no perder tiempo, raro es cuando se ve obligado a dar un paso en casilla conocida. Solo falla cuando el caso es imposible.

**Agente Aleatorio**
Este agente no se sabe cómo llegó aquí, su lógica se basa en pura apetencia, de forma erratica sin evaluar riesgos, he aquí su algoritmo,

```
Vector2Int bestOption = frontera[Random.Range(0,frontera.Count)];
GetComponent<PathFinder>().CalculatePath(bestOption);
```

se explica solo, coge cualquier posición, de la frontera va, y a lo que ocurra, este falla mucho, porque no tiene en cuenta el riesgo.

**Agente Despistado**
A diferencia de Aleatorio, este coge una casilla aleatoria de una lista de casillas con el mismo riesgo. si no tiene estímulos, sea barro o sangre, se mueve de forma rara, puede llegar a tardar demasiado, es como un TDH, que está medicado. Solo falla cuando el caso es imposible.

**Agente Mediana**
Va en diagonal hasta la otra esquina, es más o menos efectivo, tiene en cuenta los peligros y los beneficios.Solo falla cuando el caso es imposible.

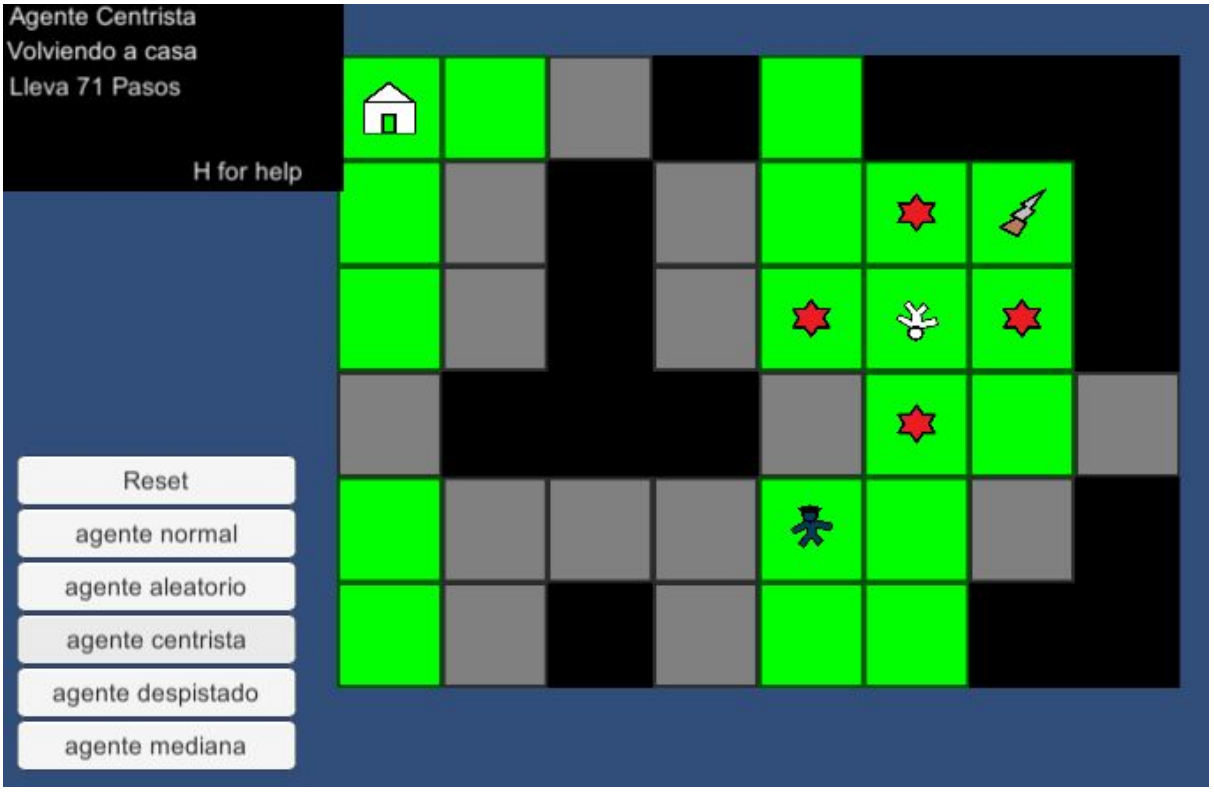
**Agente Centrista**
Intenta ir lo más rápido posible al centro del bosque, para ir patrullando hasta encontrar el cuerpo, tiene en cuenta los peligros y los beneficios. Este es el que más le gusta a Fran. Solo falla cuando el caso es imposible.

### El entorno
Todo esto se ha seguido, y se han añadido 5 botones para ejecutar 5 tipos diferentes de agentes, para poder ver como lo hacen, un mapa de 8x6, con hasta 6 barrancos

### Restricciones
Se realiza todo con éxito.

## Layout

El layout de la práctica es una implementación simple para comprobar el uso de los algoritmos. Se basa en el uso del Grid, las casillas desconocidas están en negro. También se renderiza un texto con la cantidad de pasos que lleva el agente, y el tipo de agente que es.




## Conclusiones

Las conclusiones son las siguientes:

### Quien arriesga pierde

Que el agente sea arriesgado es estúpido, en este caso arriesgarse es de forma absoluta por lo tanto frente al riesgo se debe jugar conservador en cuanto se ve un peligro, porque es más importante no fallar y tardar, que fracasar rápido, damos más importancia a cómo se recorre las casillas que tenemos disponibles y que sean seguras.

### Más estímulos mejor respuesta

Es muy importante que IA's de este tipo tengan estímulos varios, para poder realizar una respuesta mejor, sobre todo si el mapa se va haciendo más grande, podría darse el caso si el mapa es demasiado grande y no hay estimuló la búsqueda sea larga, estaría bien que para poder facilitar la búsqueda poner huellas o algo más,

## Referencias


- Apuntes de Métodos Algorítmicos para la Resolución de Problemas.