Universidad de Granada

ESCUELA TECNICA SUPERIOR DE INGENIERÍA INFORMÁTICA Y TELECOMUNICACIONES

INTELIGENCIA ARTIFICIAL

Documentacion

Documentacion a LaTeX

Hecho por:
Francisco Javier Fuentes Barragán

Índice

MEMORIA

Francisco Javier Fuentes Barragán

10 de mayo de 2016

1. Descripción de la práctica

En esta práctica vamos a programar un agente reactivo, el cual estará suelto en un entorno que debe descubrir. Este entorno viene determinado por una matriz de 100x100 en el que se representa tanto el terreno como los objetos y personajes con los que podemos interactuar. Nuestro agente tiene una vision tubular ,representada internamente por un vector de 10 elementos, con la cual percibe el entorno.

$$\begin{pmatrix} ? & ? & ? & ? & ? \\ 5 & 6 & 7 & 8 & 9 \\ ? & 2 & 3 & 4 & ? \\ ? & ? & 1 & ? & ? \\ ? & ? & 0 & ? & ? \end{pmatrix}$$

El objetivo de esta práctica es descubrir el máximo terreno posible dentro del mapa, y la máxima puntuación final.
La puntuación final viene determinada por la interaccion de nuestro agente con los personajes.

2. Algoritmo de movimiento

Nuestro agente va a seguir un algoritmo básico de movimiento, el cual combinado con demas objetos encontrados en el entorno es mas que suficiente para descubrir la mayor parte del mapa entorno en el que nos encontramos.
El algoritmo que seguimos es el "$AlgoritmodePulgarcito$"o"$HanselyGretel$", depende del cuento que prefiera cada uno. Consiste en ir soltando "$migasdepan$" en el terreno. Estoa nivel interno, lo representamos como una matriz de enteros, inicializada a 0 la cual vamos incrementando según vamos pasando por un punto y, lejos de querer volver a nuestra casa como en el cuento, lo que haremos será ir por el camino que menos migas de pan tenga.

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 4 & 5 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

```
while pasos < 20000 do
    if arriba ≤ izquierda and arriba ≤ derecha then
        Go arriba;
    end
    else if izquierda < derecha then
        Go izquierda;
    end
    else
        Go derecha;
    end
end
```

Algoritmo 1: Algoritmo de decisión

3. orientación

Dentro del entrono nos encontramos con una serie de puntos, visualmente de color amarillo, e internamente representados con la letra k los cuales son GPS. En nuestro mundo, la matriz es de 100x100 pero realmente nosotros estamos en una matriz de 200x200, es decir nuestras coordenadas no se ajustan a las dadas por la solución. Y no solo eso, nuestra matriz puede estar rotada, de tal forma que no sepamos realmente donde nos encontramos ni siquiera con nuestra variable de orientacón. Supongamos que esta es nuestra matriz original:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Nuestra matriz se puede encontrar rotada de tal forma que la veamos a la izquierda, derecha o boca abajo:

$$\begin{pmatrix} 3 & 6 & 9 \\ 2 & 5 & 8 \\ 1 & 4 & 7 \end{pmatrix} \begin{pmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{pmatrix} \begin{pmatrix} 7 & 4 & 1 \\ 8 & 5 & 2 \\ 9 & 6 & 3 \end{pmatrix}$$

Para poder orientarnos debemos localizar en al mapa dos gps para guardar tanto mis coordenadas reales como las de la matriz en la que me encentro. Una vez hecho esto calculamos f1, f2, c1 y c2 los cuales se calculan asi:

$f1 = fila_{primer\_punto\_original} - fila_{segundo\_punto\_original}$

$f2 = fila_{primer\_punto\_de\_nuestra\_matriz} - fila_{segundo\_punto\_de\_nuestra\_matriz}$

$c1 = columna_{primer\_punto\_original} - columna_{segundo\_punto\_original}$

$c2 = columna_{primer\_punto\_de\_nuestra\_matriz} - columna_{segundo\_punto\_de\_nuestra\_matriz}$

El algoritmo a seguir para orientarse será el siguiente:

```
if f1 == f2 then
    Estoy bien orientado;
    Algoritmo de copia basica;
end
else if f1 == −f2 then
    Estoy Boca abajo;
    Algoritmo de copia invertida;
end
else if f1 == c2 then
    Estoy girado a la izquierda;
    Algoritmo de rotar a la derecha;
end
else if f1 == -c2 then
    Estoy girado a la derecha;
    Algoritmo de rotar a la izquierda;
end
Algoritmo para copiar
```

Algoritmo 2: Algoritmo de orientación

```
for fila = 0; fila < 200 fila++ do
    for col = 0; col < 200; col++ do
        aux[fila][col] = entorno[fila][col];
    end
end
```

Algoritmo 3: Algoritmo de copia basica

```
segundo_punto_f = 199 − segundo_punto_f;
segundo_punto_c = 199 − segundo_punto_c;
for fila = 0; fila < 200 fila++ do
    for col = 0; col < 200; col++ do
        aux[fila][col] = entorno[199 − fila][199 − col];
    end
end
```

Algoritmo 4: Algoritmo de copia invertida

```
swap(segundo_punto_f, segundo_punto_c);
segundo_punto_c = 199 − segundo_punto_c;
for fila = 0; fila < 200 fila++ do
    for col = 0; col < 200; col++ do
        aux[fila][col] = entorno[199 − col][fila];
    end
end
```

Algoritmo 5: Algoritmo de rotar a la derecha

```
swap(segundo_punto_f, segundo_punto_c);
segundo_punto_f = 199 − segundo_punto_f;
for i|0 : 99 do
    for j|0 : 99 do
        solucion[i][j] = aux[segundo_punto_f − punto_original_f −
         i][segundo_punto_c − punto_original_c − j];
    end
end
```

Algoritmo 6: Algoritmo para copiar

4. Recoger objetos

En el mundo nos podemos encontrar con diversos objetos ls cuales podemos recoger para posteriormente usarlos a nuestro beneficio. Como dije antes, nuestro algoritmo de movimiento es muy básico, pero existen dos objetos gracias a los cuales se vuelve muy potente. Esos dos objetos son las botas y el bañador. En nuestro entorno hay zonas en las que no podemos entrar, muros, bosque, agua. Pues con estos dos objetos seremos capaces de acceder a la zona de los bosques (botas), y a la zona de agua (bañador). El algoritmo a seguir seria;

```
if bosque and botas then
    actFOWARD;
end
else if agua and bañador then
    actFOWARD;
end
else if bosque and botas-en-mochila then
    find botas;
end
else if agua and bañador-en-mochila then
    find bañador;
end
else
    Algoritmo de movimiento normal;
end
```

Algoritmo 7: Algoritmo busqueda con objetos

5. Interacción con los personajes

A parte de objetos tambien podemos encontrarnos con varios personajes a lo largo del juego. Los personajes pueden ser princesas, princesos, brujas, nuestro querido profesor de IA y al actor Leonardo Di Caprio, que supongo que se habrá perdido o algo. En nuestro entorno, a parte de los objetos que usamos para movernos, botas, bañador, se encuentran objetos que estos personajes desean, el princeso quiere la espada, el profesor de IA quiere el algoritmo, Leonardo ansía el oscar etc.. Si recogemos esos objetos y se los entregamos conseguiremos puntuacion que se sumara al final en la nota. Nuestro agente se encargará de entregarle el oscar a dicaprio y el algoritmo al profesor. el procedimiento a seguir

es igual que con las botas.

```
if dicaprio and oscar then
    actGIVE;
end
else if profesor and alg then
    actGIVE;
end
else if dicaprio and oscar-en-mochila then
    find oscar;
end
else if profesor and alg-en-mochila then
    find alg;
end
else
    Algoritmo de movimiento normal;
end
```

Algoritmo 8: Algoritmo para entregar objetos

6. Muerte

Este es un entorno hostil, en el que nos podemos encontrar amenazas, la mas significativa es un oso que al vernos nos atacara y nos matará al instante, sin que a Leonardo le de lugar a enfrentarse a él. Esto hace que debamos tener en cuenta en todo momento la posible muerte de nuestro personaje. Al morir se borrara todo y otra vez tendremos que volver a orientarnos. ¿Como podemos hacer que la muerte no sea tan perjudicial? En mi caso, con una variable booleana "*respawn*" la cual al morir se activa y salva el mapa solución. Gracias a esta variable, al la hora de volver a orientarnos nos dirá si ya teniamos solución guardada anteriormente para evitar sobreescribirla.