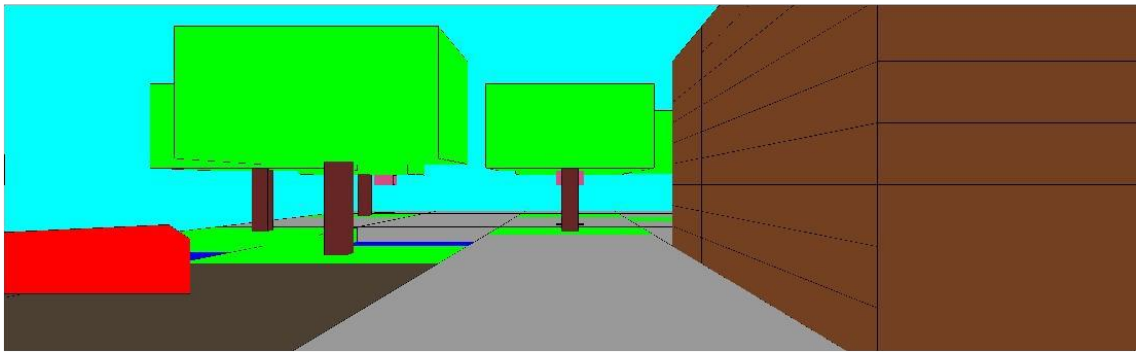
# INTELIGENCIA ARTIFICIAL

E.T.S. de Ingenierías Informática y de Telecomunicación

# Documentación

## Agentes Reactivos
**(y algo de deliberativos)**

## (Los extraños mundos de BelKan)



**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL**
**UNIVERSIDAD DE GRANADA**
Curso 2015-2016

El comportamiento del agente reactivo y un poco deliberativo implementado consta de:

## 1. ORIENTACIÓN

El primer objetivo es orientarse, es decir, saber dónde se encuentra el agente y una vez encontrado poder rastrear el mapa original oculto.

Para ello, primero debo encontrar dos puntos de referencia, representados por un cuadro amarillo en el mapa, por el carácter 'K' y emite un mensaje, en el cual nos facilita las coordenadas correctas al que pertenece.

Para ello utilizo la función "buscarPK()", en el que he implementado mediante un bucle que recorra su visión en la búsqueda de un punto de referencia, si en su visión del agente se encuentra un punto de referencia traza un recorrido a través de un switch y un vector de acciones hacia ese punto de referencia visualizado (siempre y cuando con la condición de que no tenga otro movimiento obligatorio que hacer y no haya encontrado aún dos puntos de referencia para orientarse).

### 1.2 ALMACENAR COORDENADAS

Para poder recordar o almacenar las coordenadas utilizo dos vectores de tipo entero, "posición_" para las coordenadas originales que recuperamos del mensaje emitido mediante la función "CapturaFilaColumnaPK" facilitada por el departamento que selecciona los valores enteros del mensaje, y un vector "posición_l" donde almaceno la posición x e y del mapa entorno que me encuentro.

También he utilizado un vector de tipo acción, en el que, si aún no he sido orientado, me faltan puntos de referencia por descubrir y en mi abanico de visión detecto un punto de referencia, pueda trazar un camino para ir hacia ese punto. Teniendo en cuenta que no haya obstáculos en ese camino trazado.

Una vez encontrado un punto de referencia, el siguiente punto debe ser distinto al anterior, por lo que en esta función también consta de una condición, de que las coordenadas del segundo punto de referencia no sean iguales al primero, en este caso sigue buscando en busca de un punto de referencia distinto.

### 1.3 ROTAR

Ya encontrados dos puntos de referencia, y almacenadas tanto las coordenadas originales del mapa oculto y las coordenadas del mapa entorno que me encuentro, podemos saber la orientación en la que nos encontramos. Nos podemos encontrar con 4 situaciones distintas: que estemos orientados boca abajo, boca arriba, o que nuestra derecha sea nuestra izquierda y viceversa.

Para este problema he creado una función "Rotar()" que su función es rotar la matriz de nuestro mapa entorno 90 grados a la derecha, que me facilita según en la situación que me encuentre hacer las llamadas necesarias a esta función para orientarme correctamente, sin olvidar que cada vez que llamo a esta función rotamos 90 grados a la derecha y tenemos que actualizar nuestra variable "orientación_" para que el agente acompañe la dirección a la hora de rotar.

Más detalladamente con la ayuda de una matriz auxiliar del mismo tamaño que la matriz entorno, vamos recorriendo las filas de abajo hacia arriba, para cambiar filas por columnas.

### 1.4 TRASLADAR

Una vez rotada la matriz entorno, ya tenemos la orientación, pero me encuentro ahora con el problema que cuando empezamos el juego, nuestra primera casilla siempre es la posición (99,99), y como sabemos nuestra matriz entorno es de tamaña 200x200, por este motivo nos vamos a encontrar nuestro agente explorando aproximadamente en el centro de la matriz "grande".

En resumen, he realizado una función que me traslada lo explorado mientras encuentro los puntos de referencia hacia la posición (0,0) de la misma. Entrando en detalle para realizar esta función, localizo la matriz solución de 100x100 dentro de mi matriz entorno 200x200, para hallar esto empleamos unos de los puntos de referencia encontrados y almacenados tanto las coordenadas originales obtenidas en el punto de referencia como las coordenadas locales. También hay que destacar en este paso, que dependiendo de la orientación que obtengo debo de utilizar las filas como columnas y las columnas como filas, y realizar el cálculo del inicio y fin de la matriz 100x100 dentro de la matriz 200x200.

### 1.5 GUARDAR PARTIDA

Una vez solucionado el problema anterior, ya he podido trasladar la matriz entorno a la posición (0,0) y aprovecho para guardar la partida, es decir, creo una nueva función "guardarPartida()", la cual se encarga de lo que tengo explorado en la matriz entorno la copio en la matriz solución, que esto será lo que a partir de ahora nos dará puntuación final.

A igual que esta función he creado la función "recuperarPartida()" que hace el proceso contrario, pero hablaré de ella más adelante.

### 1.6 RETINTAR BORDES

Aquí llega el momento de las fullerías, hasta este momento he conseguido, encontrar dos puntos de referencia, rotar la matriz entorno los grados necesarios para la correcta orientación, traslado la matriz entorno al origen de la misma (0,0), y almacenarla en el mapa solución. El mapa solución sé que consta de unos márgenes que equivalen a precipicio de tamaño 3 filas en el margen izquierdo y derecho, y 3 columnas en el margen superior e inferior. Por lo que sin tener que explorar estos márgenes lo adelanto con esta función.

También con la experiencia de horas y horas con esta práctica he obtenido que en el mapa oculto hay unos "muros" cerrados que solo se puede entrar encontrado en el mapa un objeto "llave" que pueda abrir la puerta para entrar a estos "muros". Pero sé que dentro de estos muros solo se encuentra terrero pedregoso representado por "S", por lo que incorporo dentro de esta función, que el mapa solución en vez de inicializarlo en blanco "?", lo inicializo en este suelo "S", facilitando que, si no encuentro la llave o no puedo explorar el interior de estos muros, eso que me llevo.

## 2. CALCULAR MOVIMIENTO

Aquí implemento en esta función el comportamiento reactivo del agente. Utilizando el llamado "camino de pulgarcito" explicado en clase.

He creado una nueva matriz global de tipo enteros de tamaño 200x200, en la que mediante una variable también global llamada "ciclo" inicializada en 0, la voy incrementando en cada turno o paso del agente, y en la posición que se encuentra voy almacenando el valor de la variable "ciclo" (tiempo[x_][y_]).

Gracias a esta matriz, voy comprobando según la orientación que se encuentre el agente, su casilla de la izquierda, casilla derecha, y la casilla frente al agente. Una vez la comprobación anterior, si la menos es la casilla izquierda, el agente gira a su izquierda. Si la menor es la derecha, el agente gira a su derecha. Y en caso de empate o que todas las casillas tengan el mismo valor, el agente siempre seguirá hacia adelante.

Tengo que destacar que antes de realizar esta decisión, realizo una llamada a la función "actualizarObstaculo()" que la describo en el siguiente apartado.

### 2.1 ACTUALIZAR OBSTACULO

Esta función solo se encarga después de cada movimiento del agente comprobar que tiene justo delante, en cualquier caso, que fuera un obstáculo (bosque, muro, personaje, precipicio…) da el valor a esa casilla correspondiente en la matriz de tiempo realizada en la función anterior, un valor suficientemente alto para que marque esa casilla como obstáculo para el agente y no la atraviese.

## 3. RECUPERAR PARTIDA

Un objetivo a destacar es que, si el agente pierde sus vidas o sobrepasa un precipicio, es reiniciado con consecuencia de perder los objetos, volver a situarse en el mapa en una localización aleatoria y perdiendo de nuevo la orientación. Por este motivo, cuando el agente "vuelve a la vida" comienza de nuevo a explorar y a realizar todas las funciones anteriores descritas. La única diferencia que una vez que el agente vuelva a encontrar dos puntos de referencia y es orientado, todo lo explorado en el mapa "antes de morir" se queda almacenado en su memoria, por lo cual, esta función recupera la información del mapa solución a su mapa entorno.

## 4. ACTUALIZAR MAPA SOLUCIÓN

Cuando he llegado hasta el punto 1.6 de este documento, la actualización del mapa solución a medida que se desarrolla la partida del juego, la realizo a través de la función "ActualizarInformacion()" ya implementada en la versión del juego.

## 5. OBJETIVOS

El primer objetivo de esta práctica ya está realizado, ahora voy a detallar el segundo objetivo de esta práctica que consta de realizar objetivos, es decir, buscar objetos en el mundo, recogerlos y entregarlos al personaje correspondiente.

Aquí tengo lo mínimo implementado por lo que será breve.

### 5.1 DAR OBJETOS

Esta parte de cumplir objetivos, he realizado la función más sencilla y básica. Me baso en aprovechar mientras el agente va explorando el mundo si se encuentra algún objeto lo coja, en el caso de encontrar más de un objeto lo almacena en la mochila, que tiene capacidad para

almacenar 4 objetos más un objeto en uso, mientras halla capacidad sigue cogiendo objetos. La idea tan simple es que, cuando el agente tenga un objeto o más, y explorando el mapa se encuentra con un personaje, le corresponda o no le facilita todos los objetos que dispone el agente. Con esto si tiene suerte conseguirá algunos puntos.

Entrando en detalle, el agente explora y en cuanto encuentra un objeto justo en su posición lo coge del mundo, con una variable global “n_objetos” puedo saber cuántos objetos tiene disponible el agente. Cuando sigue explorando y se encuentra con un personaje, realizo la llamada a la función “darObjetos()”, operación parecida a la función “buscarPK()”, dependiendo del número de objetos que tenga disponible el agente, con un vector de acciones declaramos el conjunto de acciones que realiza el agente, por ejemplo si tiene dos objetos, ya sea uno en uso y otro en la mochila, un conjunto seria 1-dar el objeto, 2-sacar el siguiente objeto de la mochila, 3-dar el objeto. Otro ejemplo con los mismos dos objetos pero este caso en la mochila el conjunto seria, 1-coger el objeto de la mochila, 2-dar el objeto, 3-coger el otro objeto de la mochila, 4-dar el objeto.