



ugr

Universidad
de Granada

SEMINARIO 1

Presentación Práctica 1

Agentes Reactivos

Inteligencia Artificial
Dpto. Ciencias de la Computación e
Inteligencia Artificial
ETSI Informática y de Telecomunicación
UNIVERSIDAD DE GRANADA
Curso 2016/2017



DECSAI

Índice

1. Introducción
2. Presentación del Problema
3. Presentación del Simulador
4. Implementación de un agente
5. Método de evaluación de la práctica

Índice

1. Introducción
2. Presentación del Problema
3. Presentación del Simulador
4. Implementación de un agente
5. Método de evaluación de la práctica

1. Introducción

- El objetivo de esta práctica consiste en el diseño e implementación de un agente reactivo que es capaz de:
 - *percibir el ambiente y*
 - *actuar de acuerdo a un comportamiento simple predefinido*
- Trabajaremos con un simulador software de la aspiradora inteligente basada en los ejemplos del libro *Stuart Russell, Peter Norvig, “Inteligencia Artificial: Un enfoque Moderno”*
- El simulador que utilizaremos fue inicialmente desarrollado por el profesor **Tsung-Che Chiang** de la NTNU (*Norwegian University of Science and Technology, Taiwan*)

1. Introducción

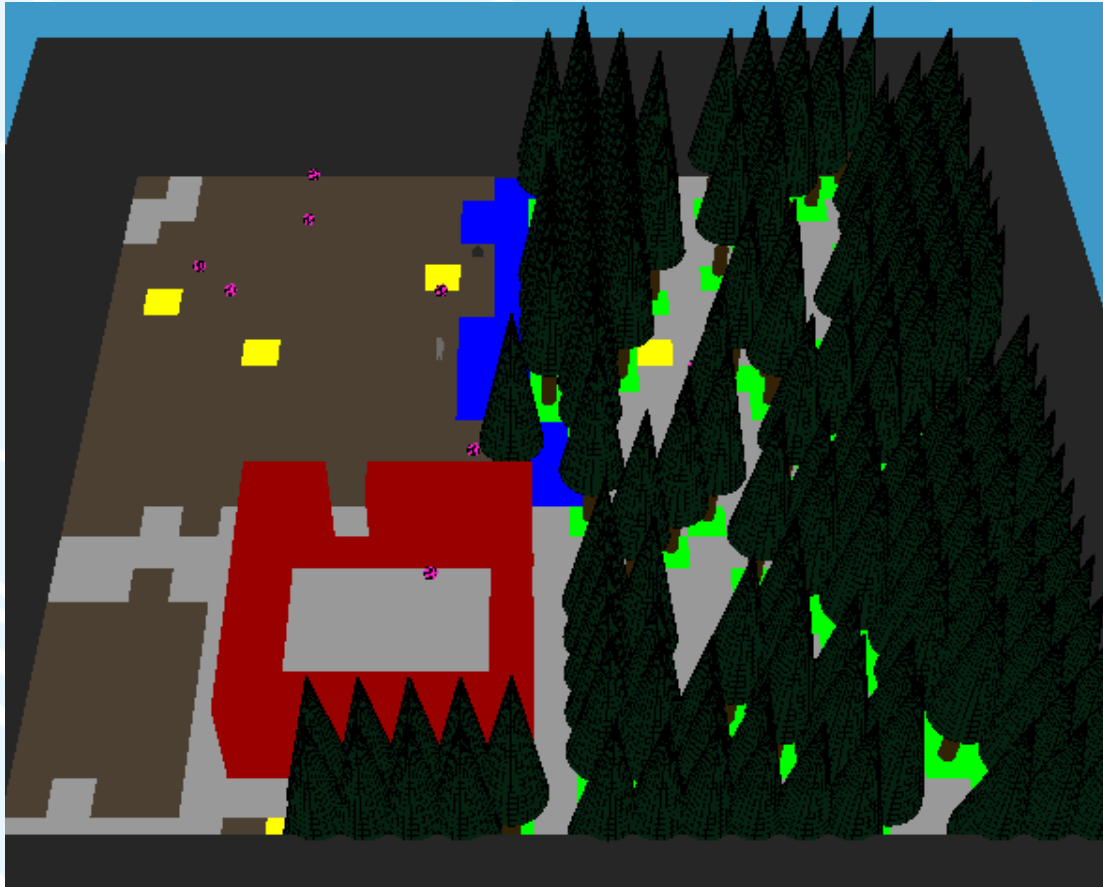
- Esta práctica cubre los siguientes objetivos docentes:
 - Entender la IA como conjunto de técnicas para el desarrollo de sistemas informáticos que exhiben comportamientos reactivos, deliberativos y/o adaptativos (sistemas inteligentes)
 - Conocer el concepto de agente inteligente y el ciclo de vida "percepción, decisión y actuación"
 - Comprender que el desarrollo de sistemas inteligentes pasa por el diseño de agentes capaces de representar conocimiento y resolver problemas y que puede orientarse a la construcción de sistemas bien completamente autónomos o bien que interactúen y ayuden a los humanos
 - Conocer distintas aplicaciones reales de la IA. Explorar y analizar soluciones actuales basadas en técnicas de IA.

Índice

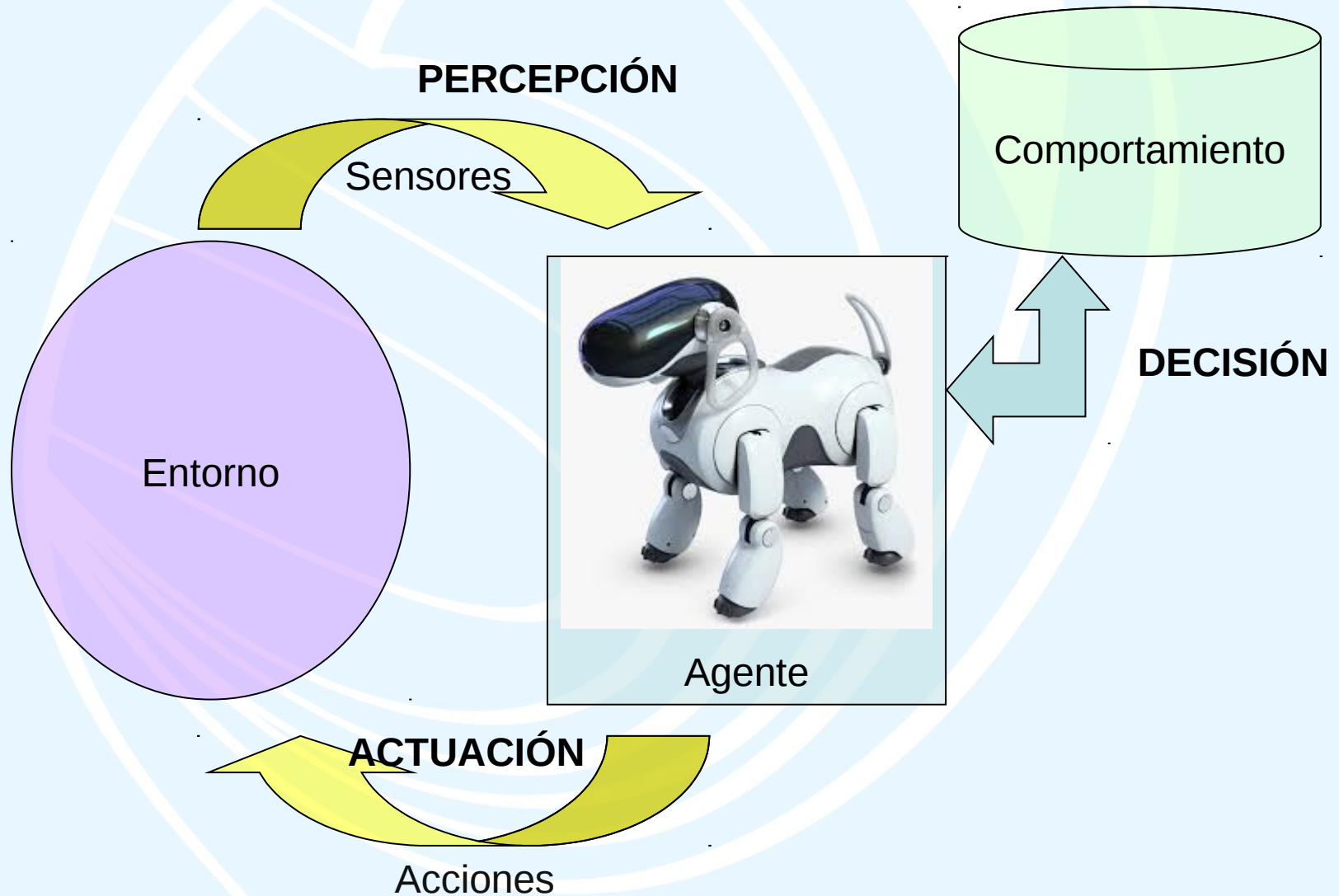
1. Introducción
2. Presentación del Problema
3. Presentación del Simulador
4. Implementación de un agente
5. Evaluación de la práctica

2. Presentación del Problema

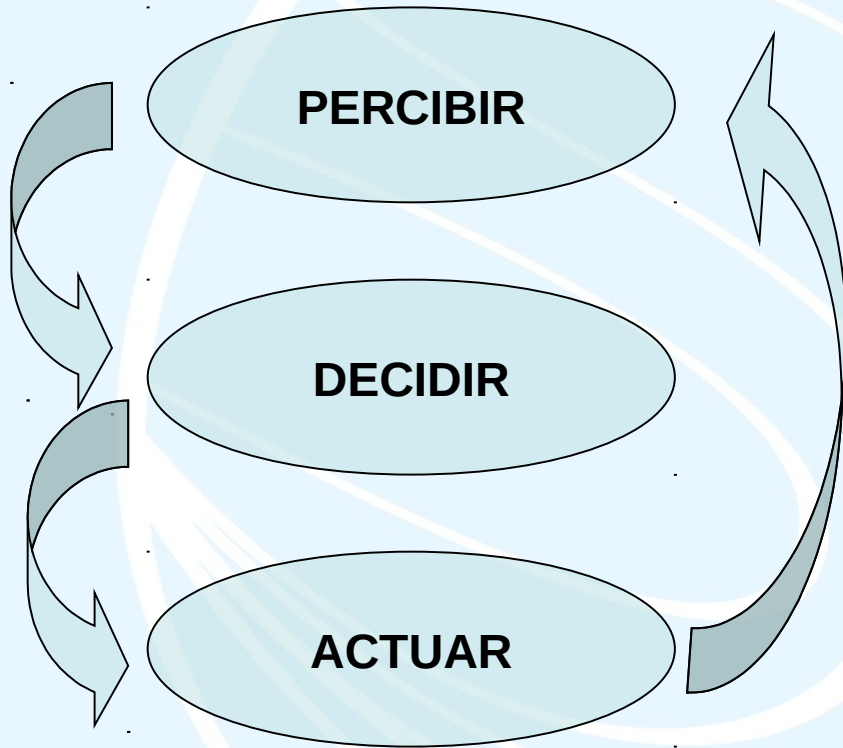
Los extraños mundos de BelKan



2. Presentación del Problema



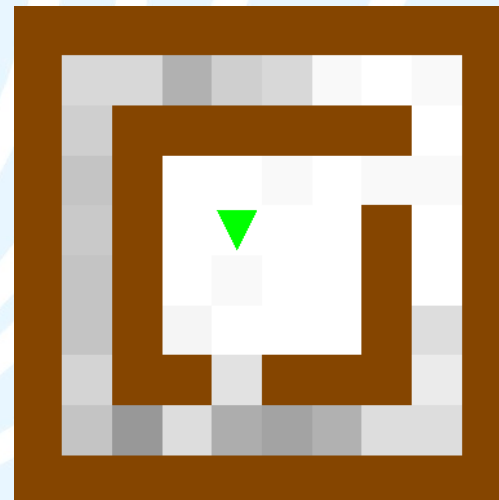
2. Presentación del Problema



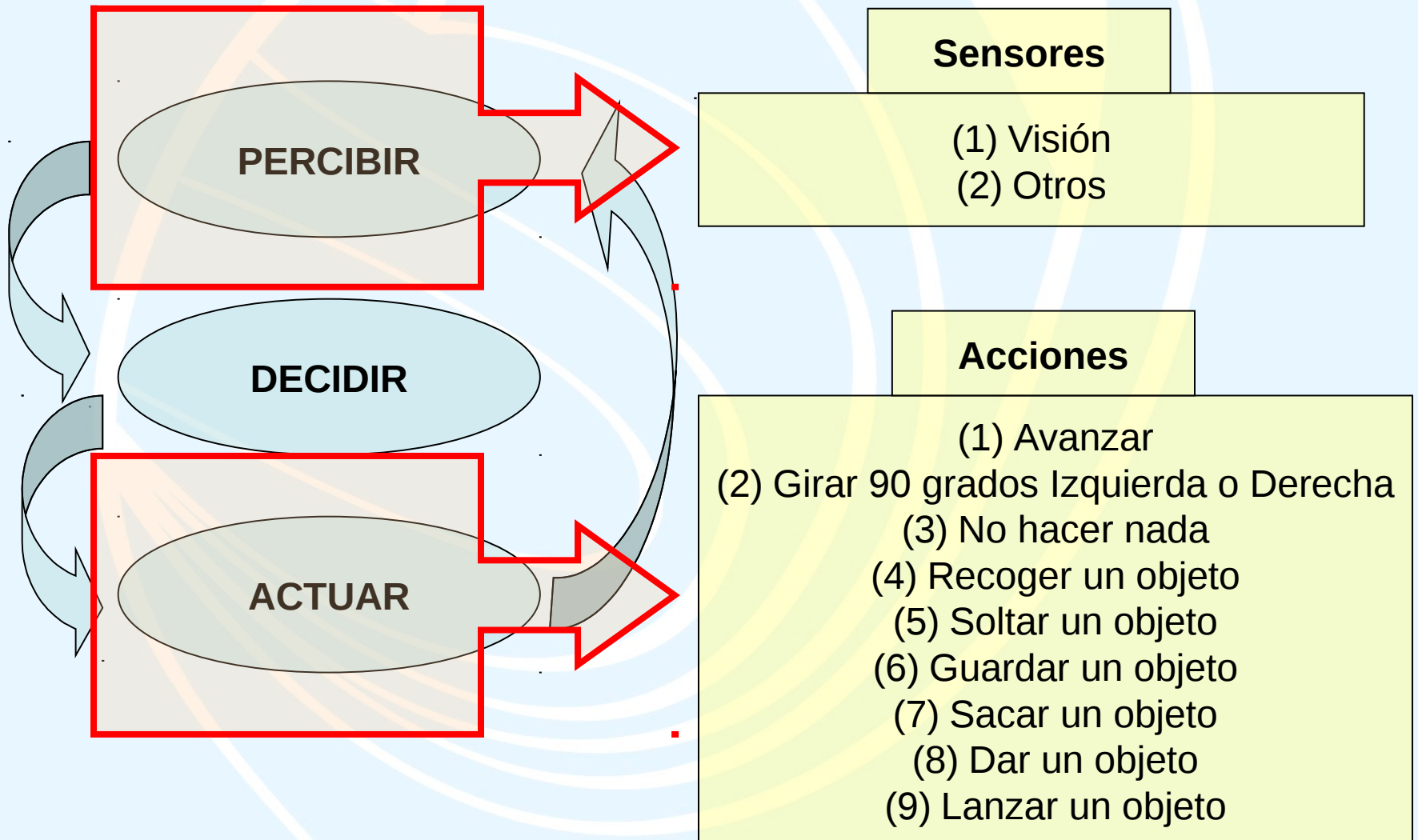
Controlador de ciclo cerrado

Trabajaremos en un mundo simulado:

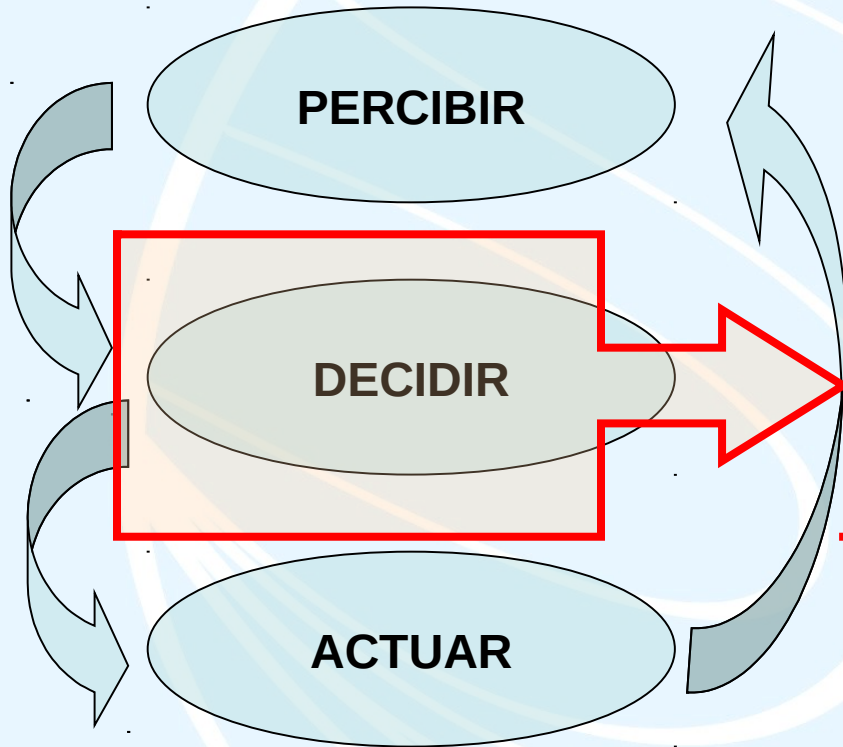
- Lo que es capaz de percibir el agente del entorno y,
- las acciones que el agente puede realizar.



2. Presentación del Problema



2. Presentación del Problema



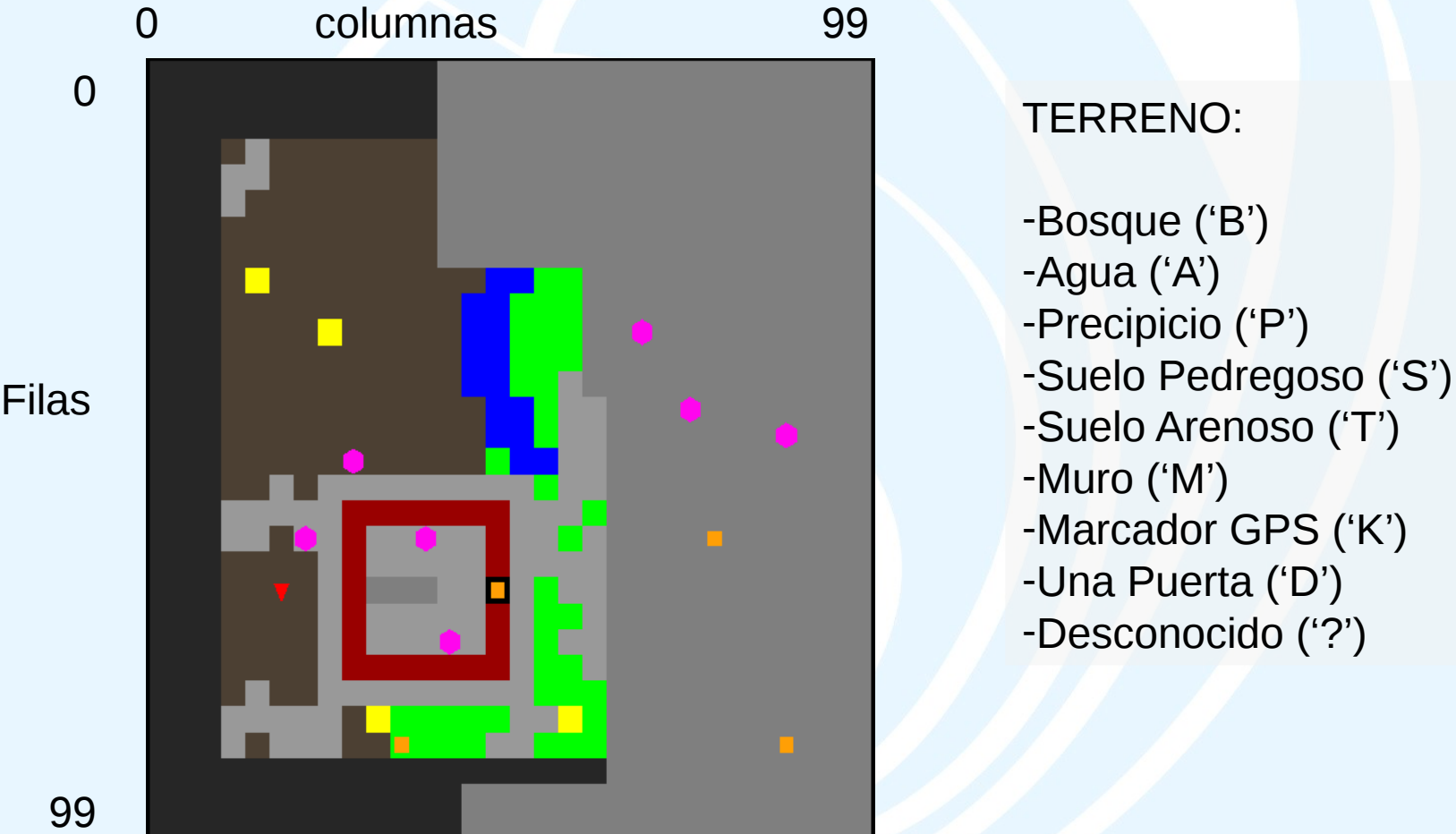
El objetivo de la práctica será:

Diseñar e implementar un modelo de decisión para este agente reactivo con las restricciones fijadas

El Juego

En este juego tomas el control de un agente que debe reconocer el mapa sobre el que se mueve, suponiendo que siempre parte de un punto aleatorio del mapa, de coordenadas desconocidas, pero siempre con orientación NORTE.

El Mundo



El perímetro del mundo siempre tiene 3 filas/columnas de precipicios

El Mundo

0 columnas 99

0

Filas

99



Objetos:

- 'a' Aldeanos
- 'd' Puerta Cerrada
- 'l' Lobos
- 'j' Jugador
- ' ' Nada
- ['0','3'] Objetos Útiles
- '?' Nada

Los Personajes

Pueden aparecer:

- Aldeanos: Solo estorban, pero no son agresivos.
- Lobos: Estos estorban y son peligrosos.

Los Objetos Útiles

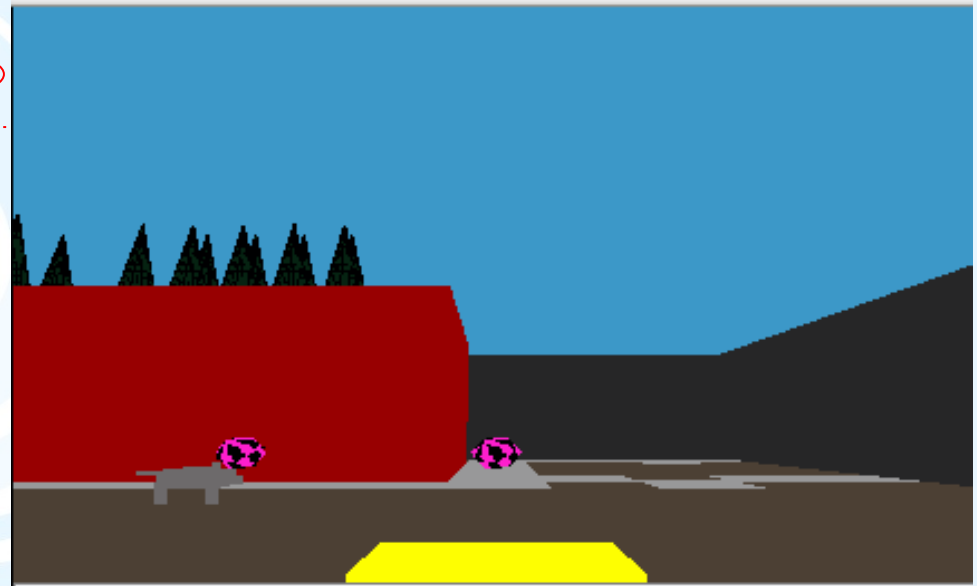
Son cuatro:

- Hueso ('0')
- Biquini ('1')
- Zapatillas ('2')
- Llaves ('3')

Los Sensores

```
Vida restante: 962
Objeto en uso _
Primer objeto de la mochila _
He colisionado? 0
Acabo de morir? 0
Fila y columna del PK (-1 , -1)
Ultima Accion: actFORWARD
Superficie:
j _ _ _ _ 1 _ _ _ _ _ _ _ _ _ _
Terreno:
T S S P T S S P P S S S S P P P
```

Indica los objetos que hay sobre la superficie y se trata de la misma forma que la visión del terreno.



Las Propiedades Del Agente

Vida restante: 962

Objeto en uso _

Primer objeto de la mochila _

He colisionado? 0

Acabo de morir? 0

Fila y columna del PK (-1 , -1)

Ultima Accion: actFORWARD

Superficie:

j _ _ _ _ _ _ _ _ _ _

Terreno:

T S S P T S S P P S S S S P P P

Los objetos que va recogiendo pueden usarse. La forma correcta de uso debes aprenderla durante el juego.

En función del objeto, se entenderá o bien que lo lleva en la mano o bien que lo lleva puesto.

Sólo un objeto puede estar en uso en cada instante de simulación

Las Propiedades Del Agente

Vida restante: 962

Objeto en uso _

Primer objeto de la mochila _

He colisionado? 0

Acabo de morir? 0

Fila y columna del PK (-1 , -1)

Ultima Accion: actFORWARD

Superficie:

j _ _ _ _ _

Terreno:

T S S P T S S P P S S S P P P

El agente lleva una mochila que le permite guardar hasta 4 de los objetos que ha recogido.

Hay un sensor que le indica cual es el siguiente objeto que saldrá de la mochila.

Las Propiedades Del Agente

Vida restante: 962

Objeto en uso _

Primer objeto de la mochila _

He colisionado? 0

Acabo de morir? 0

Fila y columna del PK (-1 , -1)

Ultima Accion: actFORWARD

Superficie:

j _ _ _ _ _

Terreno:

T S S P T S S P P S S S P P P

El agente empieza con 1000 unidades de vida, y cada vez que se ejecuta un ciclo de simulación, pierde una unidad.

Cuando el agente llega a 0 unidades, desaparece del mapa perdiendo todas las pertenencias que tuviera en ese momento, y reaparece en un nuevo lugar del mapa (siempre con orientación NORTE), con 1000 nuevas unidades de vida.

OJO!! Bajo ciertas circunstancias puede perder unidades de vidas adicionales.

Las Propiedades Del Agente

```
Vida restante: 962  
Objeto en uso _  
Primer objeto de la mochila _  
He colisionado? 0  
Acabo de morir? 0  
Fila y columna del PK (-1 , -1)  
Ultima Accion: actFORWARD  
Superficie:  
j _ _ _ _ _ _ _ _ _ _  
Terreno:  
T S S P T S S P P S S S S P P P
```

El agente puede morir por otras circunstancias del juego. Cuando esto ocurre un sensor lo avisa.

Las Propiedades Del Agente

```
Vida restante: 962
Objeto en uso _
Primer objeto de la mochila _
He colisionado? 0
Acabo de morir? 0
Fila y columna del PK (-1 , -1)
Ultima Accion: actFORWARD
Superficie:
j _ _ _ _ _ _ _ _ _ _
Terreno:
T S S P T S S P P S S S P P P
```

El agente también puede ser bloqueado en su avance, o bien por un elemento del terreno, o bien por colisiones inesperadas con los otros agentes del juego.

Este sensor debe ser consultado para controlar estas circunstancias.

Las Propiedades Del Agente

```
Vida restante: 962
Objeto en uso _
Primer objeto de la mochila _
He colisionado? 0
Acabo de morir? 0
Fila y columna del PK (-1 , -1)
Ultima Accion: actFORWARD
Superficie:
j _ _ _ _ _
Terreno:
T S S P T S S P P S S S P P P
```

Quando el agente pasa por un punto PK (las casillas amarillas) se le proporciona información de su posición exacta en el mapa.

Un sensor se activa con los valores de fila y columna del agente cuando este hecho sucede.

Las Acciones

actFORWARD Avanzar una casilla en el sentido que marca su orientación.

actTURN_L Gira a la izquierda quedándose en la misma casilla.

actTURN_R Gira a la derecha quedándose en la misma casilla.

actIDLE No hace nada.

actPICKUP Recoge un objeto del mapa (debe estar mirando a la casilla en la que se encuentra el objeto).

actPUTDOWN Deja un objeto sobre el mapa (lo deja en la casilla siguiente según su orientación).

actPUSH Guarda un objeto en la mochila.

actPOP Saca un objeto de la mochila. (la mochila se comporta como una cola)

actGIVE Da un objeto a uno de los personajes.

actTHROW Lanza un objeto.

El Objetivo

El objetivo del juego es:

Definir un comportamiento reactivo para nuestro personaje que le permita reconocer el máximo porcentaje posible del mundo y que sepa orientarlo adecuadamente. Para realizar esta tarea tiene todo el tiempo de simulación y sólo al final es cuando este objetivo es evaluado.

Índice

1. Introducción
2. Presentación del Problema
3. Presentación del Simulador
4. Implementación de un agente
5. Evaluación de la práctica

3. Presentación del Simulador

- Compilación del simulador
- Ejecución del simulador

3. Presentación del Simulador

3.1. Compilación del Simulador

Nota: En esta presentación, asumimos que el sistema operativo **será Ubuntu 16.**

1. Cread una carpeta **“practica1”**
2. Descargar **practica1_ubuntu.tgz** desde la **web** de la asignatura y cópielo en la carpeta anterior.



(a) <http://decsai.ugr.es>

(b) Entrar en acceso identificado

(c) Elegir la asignatura “Inteligencia Artificial”

(d) Seleccionar “Material de la Asignatura”

(e) Seleccionar “Práctica 1”

(f) Descargar “Software Agente Reactivo”

3. Presentación del Simulador

3.1. Compilación del Simulador (Ubuntu)

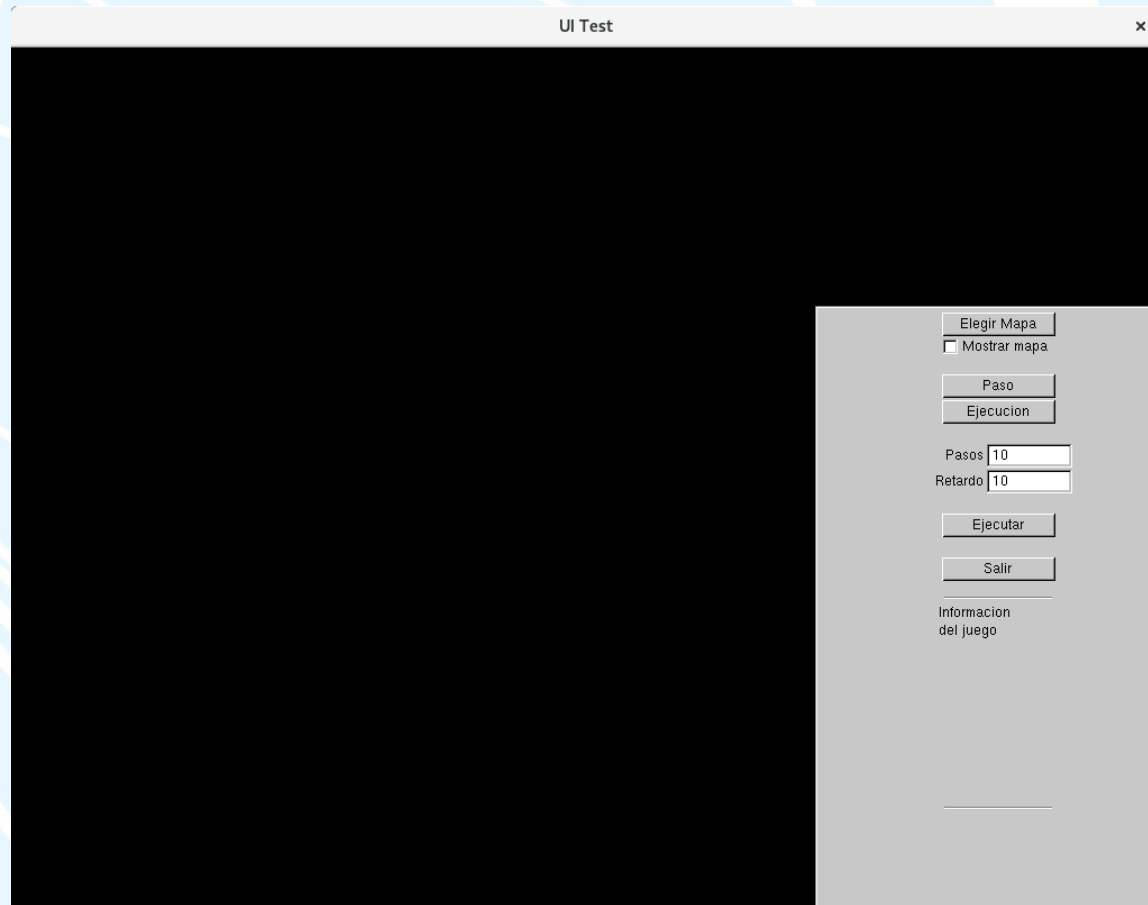
Una vez descargado y descomprimido el software hacemos lo siguiente:

1. Ejecutamos el fichero “./install.sh” que permite instalar la bibliotecas necesarias para poder compilar.
2. Ejecutamos el comando “cmake .” que nos creará el fichero “makefile”
3. Ejecutamos el comando “make”

3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

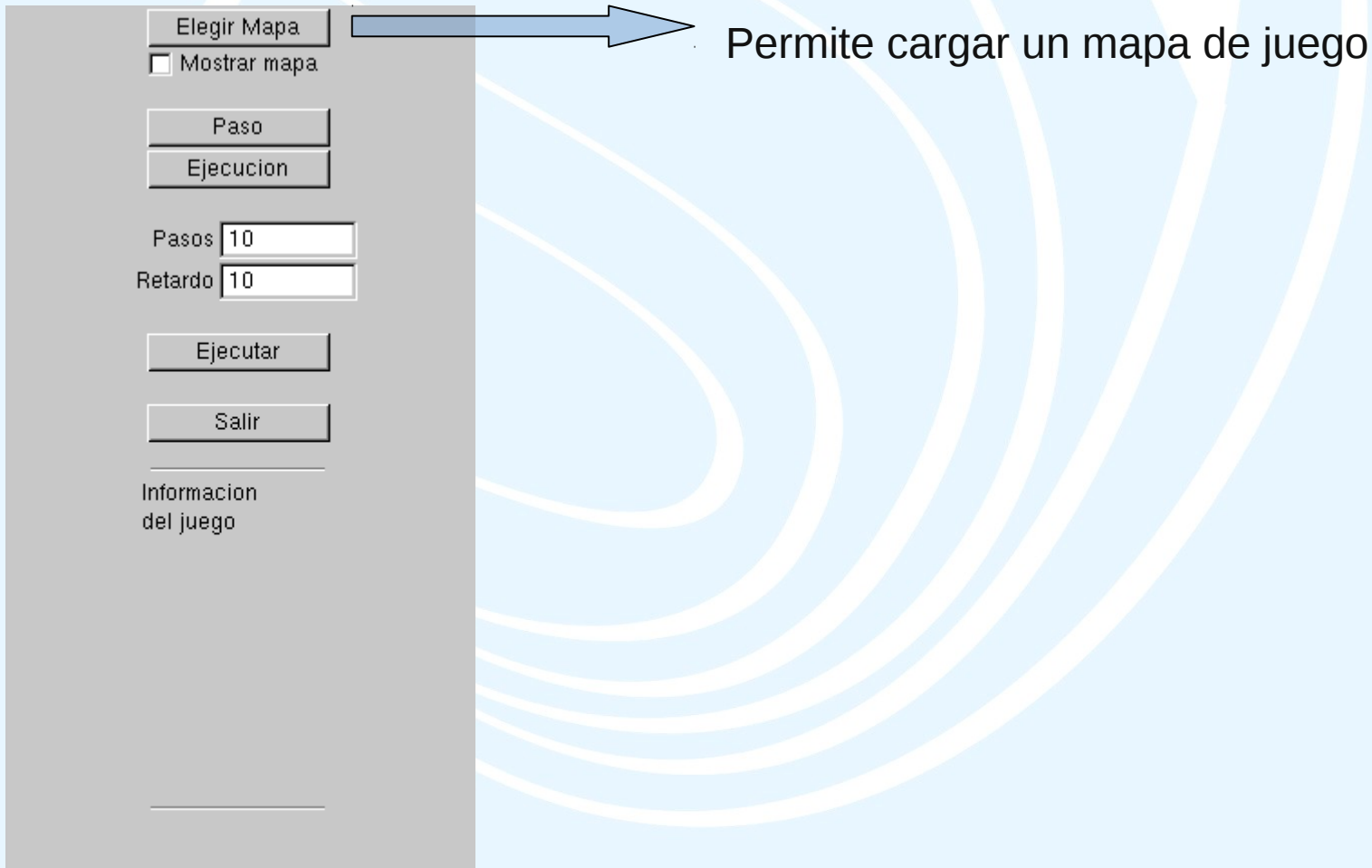
Ejecutamos el comando `“./Belkan”` y nos aparecera esto



3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

Las opciones del simulador son las siguientes:



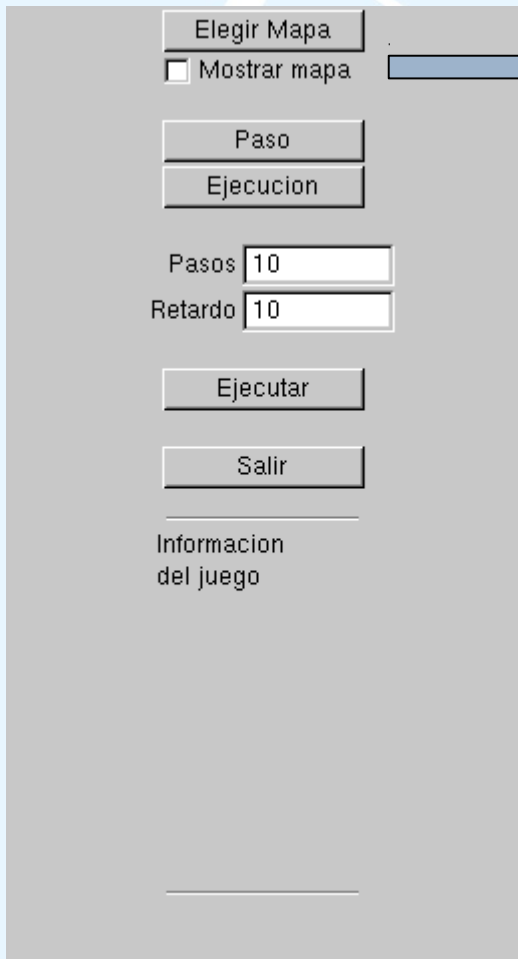
The image shows a screenshot of a simulator's control panel. On the left, there is a vertical stack of controls: a button labeled 'Elegir Mapa', a checkbox labeled 'Mostrar mapa', two buttons labeled 'Paso' and 'Ejecucion', two input fields labeled 'Pasos' and 'Retardo' both containing the value '10', a button labeled 'Ejecutar', and a button labeled 'Salir'. Below these is a section header 'Informacion del juego' followed by a horizontal line. A blue arrow points from the 'Elegir Mapa' button to the text 'Permite cargar un mapa de juego' on the right.

Permite cargar un mapa de juego

3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

Las opciones del simulador son las siguientes:



Elegir Mapa

☐ Mostrar mapa

Paso

Ejecucion

Pasos

Retardo

Ejecutar

Salir

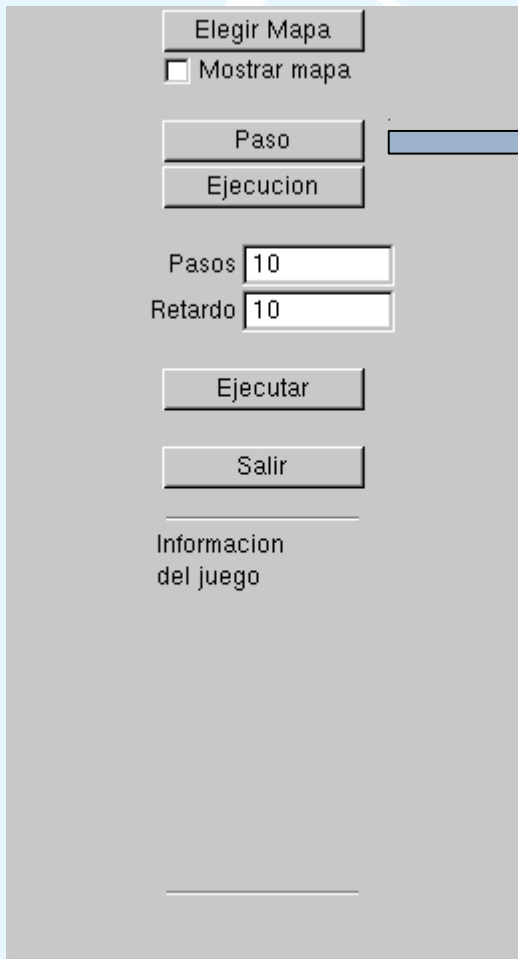
Informacion
del juego

Cambia entre mapa completo y mapa
actualmente explorado

3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

Las opciones del simulador son las siguientes:



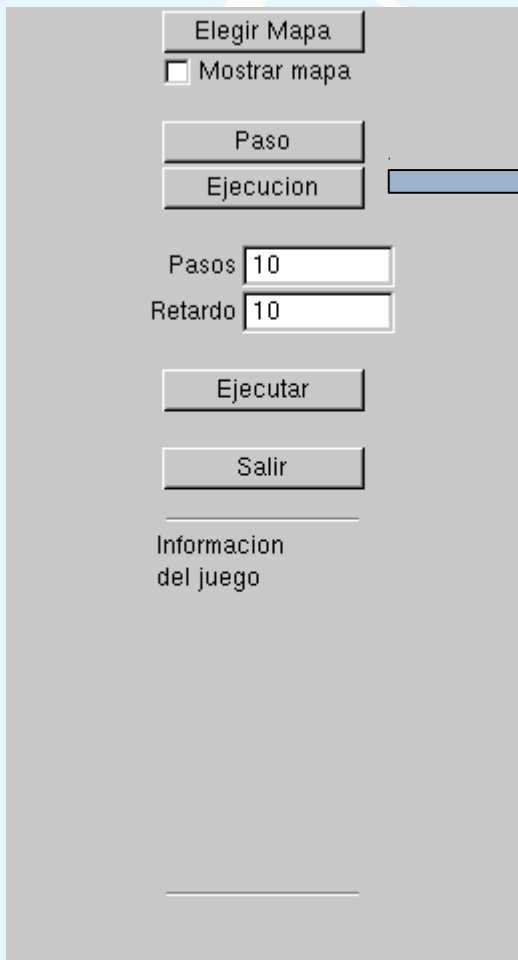
The image shows a control panel for a simulator. It includes a button labeled 'Elegir Mapa' at the top. Below it is a checkbox labeled 'Mostrar mapa' which is currently unchecked. There are two buttons, 'Paso' and 'Ejecucion', stacked vertically. Below these are two input fields: 'Pasos' with the value '10' and 'Retardo' with the value '10'. Further down are buttons for 'Ejecutar' and 'Salir'. At the bottom, there is a section header 'Informacion del juego' followed by a horizontal line.

Hace una ejecución de la simulación
paso a paso

3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

Las opciones del simulador son las siguientes:



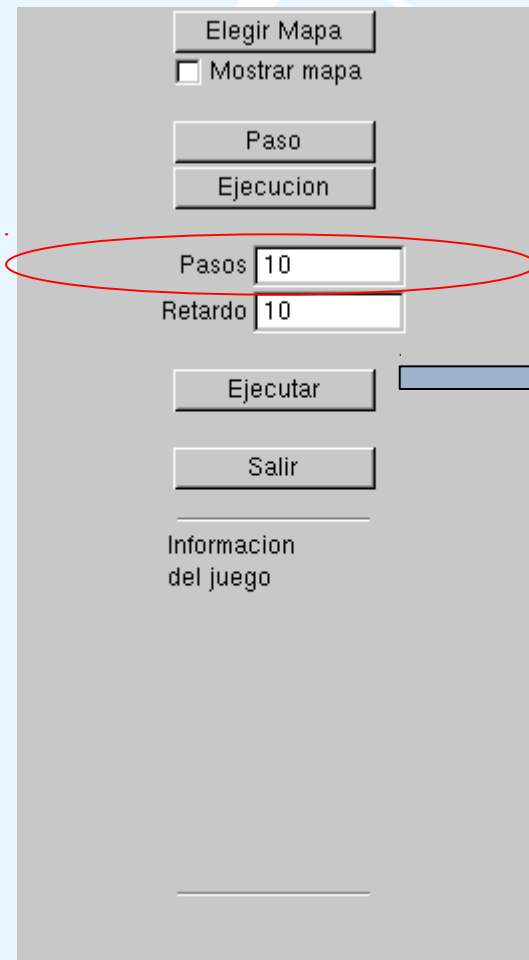
The image shows a vertical control panel for a simulator. It contains the following elements from top to bottom: a button labeled 'Elegir Mapa', a checkbox labeled 'Mostrar mapa' which is currently unchecked, a button labeled 'Paso', a button labeled 'Ejecucion', a text input field labeled 'Pasos' with the value '10', a text input field labeled 'Retardo' with the value '10', a button labeled 'Ejecutar', and a button labeled 'Salir'. Below these buttons is a section header 'Informacion del juego' followed by a horizontal line. A blue arrow points from the 'Ejecucion' button to the right, towards the text 'Realiza una ejecución completa de la simulación'.

Realiza una ejecución completa de la simulación

3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

Las opciones del simulador son las siguientes:



The image shows a screenshot of a simulator's control panel. It features several buttons and input fields. A red oval highlights the 'Pasos' input field, which contains the number '10'. A blue arrow points from this field towards the 'Ejecutar' button. Other visible elements include buttons for 'Elegir Mapa', 'Paso', 'Ejecucion', 'Ejecutar', and 'Salir', as well as checkboxes for 'Mostrar mapa' and 'Retardo' (set to 10). At the bottom, there is a section labeled 'Informacion del juego'.

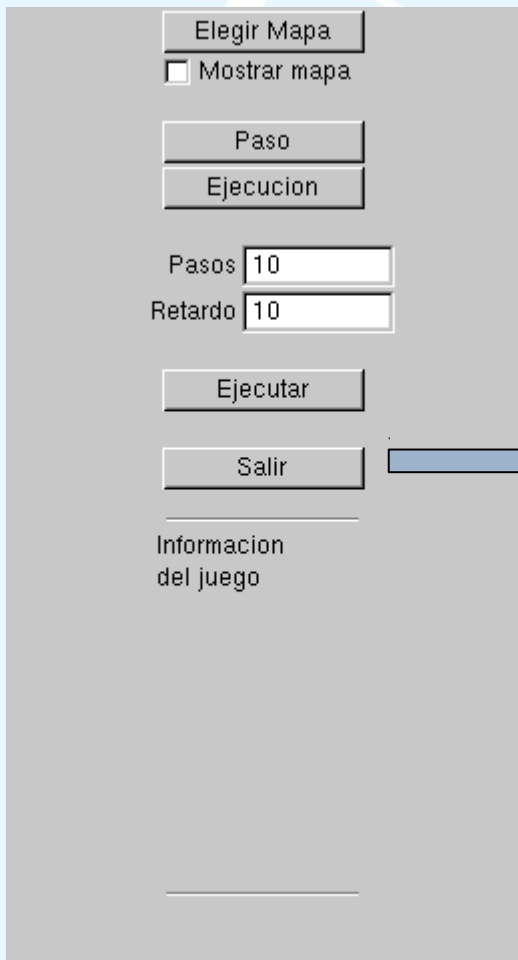
Elegir Mapa
<input type="checkbox"/> Mostrar mapa
Paso
Ejecucion
Pasos 10
Retardo 10
Ejecutar
Salir
Informacion del juego

Realiza una ejecución con un número de pasos fijo. El número de pasos viene definido en el campo Pasos

3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

Las opciones del simulador son las siguientes:



Elegir Mapa

☐ Mostrar mapa

Paso

Ejecucion

Pasos

Retardo

Ejecutar

Salir

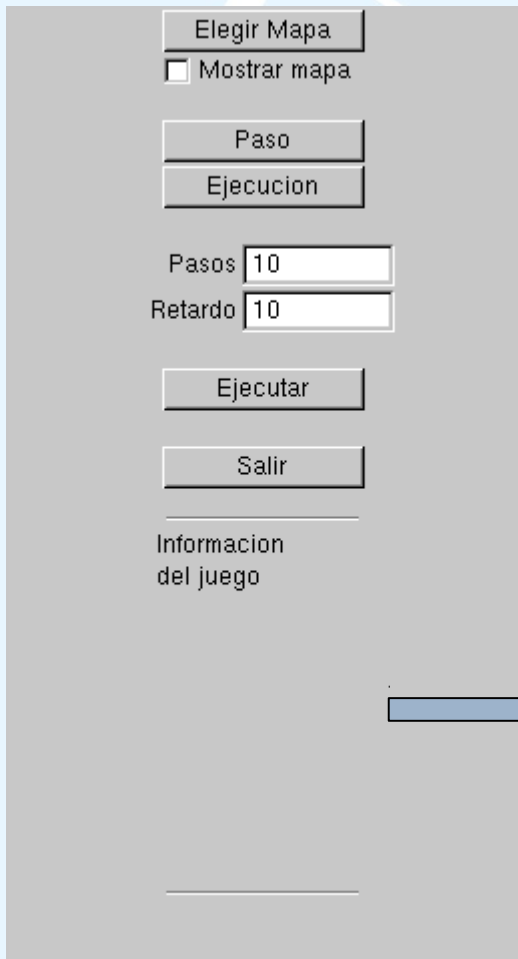
Informacion del juego

Salir del Juego

3. Presentación del Simulador

3.2. Ejecución del Simulador (Ubuntu)

Las opciones del simulador son las siguientes:



The image shows a control panel for a simulator. It includes several buttons and input fields. At the top, there is a button labeled 'Elegir Mapa'. Below it is a checkbox labeled 'Mostrar mapa'. Further down are buttons for 'Paso' and 'Ejecucion'. Below these are two input fields: 'Pasos' with the value '10' and 'Retardo' with the value '10'. Below the input fields are buttons for 'Ejecutar' and 'Salir'. At the bottom, there is a label 'Informacion del juego' followed by a horizontal line.

Información de los sensores cuya descripción ya hemos visto antes.

Índice

1. Introducción
2. Presentación del Problema
3. Presentación del Simulador
4. Implementación de un agente
5. Método de evaluación de la práctica

4. Implementación de un agente

1. Ficheros Relevantes del simulador
2. Métodos y variables del agente
3. Modificando el comportamiento del agente: un ejemplo ilustrativo.

4. Implementación de un agente

4.1. Ficheros Relevantes

Sólo hay 2 ficheros relevantes para la práctica: “jugador.cpp” y “jugador.hpp”.

Estos dos archivos se encuentran en la carpeta “Comportamientos_Jugador”

El resto de archivos que se adjuntan con la práctica no se pueden modificar y se han incluido para poder hacer la compilación.


La compilación genera dos ejecutables:

1. Belkan (simulador con entorno gráfico)
2. BelkanSG (simulador sin entorno gráfico)

4. Implementación de un agente

4.2. Métodos y variables del agente

```
1 #ifndef COMPORTAMIENTOJUGADOR_H
2 #define COMPORTAMIENTOJUGADOR_H
3
4 #include "comportamientos/comportamiento.hpp"
5 using namespace std;
6
7 class ComportamientoJugador : public Comportamiento{
8
9 public:
10     ComportamientoJugador(unsigned int size) : Comportamiento(size){
11     }
12
13     ComportamientoJugador(const ComportamientoJugador & comport) : Comportamiento(comport){}
14     ~ComportamientoJugador(){}
15
16     Action think(Sensores sensores);
17
18     int interact(Action accion, int valor);
19
20
21     ComportamientoJugador * clone(){return new ComportamientoJugador(*this);}
22
23
24 private:
25 };
26
27
28
29 #endif
```

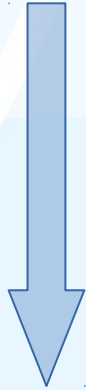


Constructor de la Clase. Aquí se realizará la inicialización de las variables de estado

4. Implementación de un agente

4.2. Métodos y variables del agente

```
1 #ifndef COMPORTAMIENTOJUGADOR_H
2 #define COMPORTAMIENTOJUGADOR_H
3
4 #include "comportamientos/comportamiento.hpp"
5 using namespace std;
6
7 class ComportamientoJugador : public Comportamiento{
8
9     public:
10         ComportamientoJugador(unsigned int size) : Comportamiento(size){
11         }
12
13         ComportamientoJugador(const ComportamientoJugador & comport) : Comportamiento(comport){}
14         ~ComportamientoJugador(){}
15
16         Action think(Sensores sensores);
17
18         int interact(Action accion, int valor);
19
20
21         ComportamientoJugador * clone(){return new ComportamientoJugador(*this);}
22
23     private:
24 };
25
26 #endif
```



El método “think” es el encargado de definir el comportamiento del agente. Recibe como entrada los sensores y devuelve la siguiente acción a realizar.

4. Implementación de un agente

4.2. Métodos y variables del agente

```
1 #include "../Comportamientos_Jugador/jugador.hpp"
2 #include <iostream>
3 using namespace std;
4
5
6
7
8
9 Action ComportamientoJugador::think(Sensores sensores){
10
11     Action accion = actIDLE;
12
13     // En esta matriz de tamaño 100x100 hay que escribir el mapa solución
14     // mapaResultado[fila][columna] = lo que hay en fila columna
15
16
17     cout << "Terreno: ";
18     for (int i=0; i<sensores.terreno.size(); i++)
19         cout << sensores.terreno[i];
20     cout << endl;
21
22     cout << "Superficie: ";
23     for (int i=0; i<sensores.superficie.size(); i++)
24         cout << sensores.superficie[i];
25     cout << endl;
26
27     cout << "Colisión: " << sensores.colision << endl;
28     cout << "Mochila: " << sensores.mochila << endl;
29     cout << "Reset: " << sensores.reset << endl;
30     cout << "Vida: " << sensores.vida << endl;
31     cout << "objetoActivo: " << sensores.objetoActivo << endl;
32     cout << endl;
33
34     return accion;
35 }
36
37 int ComportamientoJugador::interact(Action accion, int valor){
38     return false;
39 }
```

Sensores es de tipo registro y contiene los siguientes campos:

- terreno: un vector de “unsigned char” con la codificación del terreno que ve el agente.
- superficie: un vector de “unsigned char” con la codificación de los objetos u otras entidades que están en el mapa.
- colision: de tipo “bool” que indica si se ha producido una colisión.

4. Implementación de un agente

4.2. Métodos y variables del agente

```
1 #include "../Comportamientos_Jugador/jugador.hpp"
2 #include <iostream>
3 using namespace std;
4
5
6
7
8
9 Action ComportamientoJugador::think(Sensores sensores){
10
11     Action accion = actIDLE;
12
13     // En esta matriz de tamaño 100x100 hay que escribir el mapa solución
14     // mapaResultado[fila][columna] = lo que hay en fila columna
15
16
17     cout << "Terreno: ";
18     for (int i=0; i<sensores.terreno.size(); i++)
19         cout << sensores.terreno[i];
20     cout << endl;
21
22     cout << "Superficie: ";
23     for (int i=0; i<sensores.superficie.size(); i++)
24         cout << sensores.superficie[i];
25     cout << endl;
26
27     cout << "Colisión: " << sensores.colision << endl;
28     cout << "Mochila: " << sensores.mochila << endl;
29     cout << "Reset: " << sensores.reset << endl;
30     cout << "Vida: " << sensores.vida << endl;
31     cout << "objetoActivo: " << sensores.objetoActivo << endl;
32     cout << endl;
33
34     return accion;
35 }
36
37 int ComportamientoJugador::interact(Action accion, int valor){
38     return false;
39 }
```

- mochila: de tipo "unsigned char" que mantiene el siguiente objeto en salir.

- reset: de tipo "bool" que indica si has sido reiniciado en el mapa.

- mensajeF: de tipo "int" indicando la fila en la que se encuentra el agente (siempre que sea distinto de -1)

- mensajeC: de tipo "int" igual que el anterior, pero indicando la columna.

4. Implementación de un agente

4.2. Métodos y variables del agente

```
1 #include "../Comportamientos_Jugador/jugador.hpp"
2 #include <iostream>
3 using namespace std;
4
5
6
7
8
9 Action ComportamientoJugador::think(Sensores sensores){
10
11     Action accion = actIDLE;
12
13     // En esta matriz de tamaño 100x100 hay que escribir el mapa solución
14     // mapaResultado[fila][columna] = lo que hay en fila columna
15
16
17     cout << "Terreno: ";
18     for (int i=0; i<sensores.terreno.size(); i++)
19         cout << sensores.terreno[i];
20     cout << endl;
21
22     cout << "Superficie: ";
23     for (int i=0; i<sensores.superficie.size(); i++)
24         cout << sensores.superficie[i];
25     cout << endl;
26
27     cout << "Colisión: " << sensores.colision << endl;
28     cout << "Mochila: " << sensores.mochila << endl;
29     cout << "Reset: " << sensores.reset << endl;
30     cout << "Vida: " << sensores.vida << endl;
31     cout << "objetoActivo: " << sensores.objetoActivo << endl;
32     cout << endl;
33
34     return accion;
35 }
36
37 int ComportamientoJugador::interact(Action accion, int valor){
38     return false;
39 }
```

- objetoActivo: de tipo "unsigned char" indica el objeto que tiene el agente en la mano o puesto en este momento.

4. Implementación de un agente

4.2. Métodos y variables del agente

```
1 #include "../Comportamientos_Jugador/jugador.hpp"
2 #include <iostream>
3 using namespace std;
4
5
6
7
8
9 Action ComportamientoJugador::think(Sensores sensores){
10
11     Action accion = actIDLE;
12
13     // En esta matriz de tamaño 100x100 hay que escribir el mapa solucion
14     // mapaResultado[filas][columnas] = lo que hay en fila columna
15
16
17     cout << "Terreno: ";
18     for (int i=0; i<sensores.terreno.size(); i++)
19         cout << sensores.terreno[i];
20     cout << endl;
21
22     cout << "Superficie: ";
23     for (int i=0; i<sensores.superficie.size(); i++)
24         cout << sensores.superficie[i];
25     cout << endl;
26
27     cout << "Colisión: " << sensores.colision << endl;
28     cout << "Mochila: " << sensores.mochila << endl;
29     cout << "Reset: " << sensores.reset << endl;
30     cout << "Vida: " << sensores.vida << endl;
31     cout << "objetoActivo: " << sensores.objetoActivo << endl;
32     cout << endl;
33
34     return accion;
35 }
36
37 int ComportamientoJugador::interact(Action accion, int valor){
38     return false;
39 }
```

En la matriz llamada `mapaResultado` de tamaño 100x100 se almacena el mapa que se devolverá como solución.

4. Implementación de un agente

4.3. Modificando el comportamiento del agente: ejemplo ilustrativo 1.

Supongamos que queremos hacer que el agente avance.

```
1 #include "../Comportamientos_Jugador/jugador.hpp"
2 #include <iostream>
3 using namespace std;
4
5
6
7
8
9 Action ComportamientoJugador::think(Sensores sensores){
10
11     Action accion = actIDLE;
12
13     // En esta matriz de tamaño 100x100 hay que escribir el mapa solución
14     // mapaResultado[fila][columna] = lo que hay en fila columna
15
16
17     cout << "Terreno: ";
18     for (int i=0; i<sensores.terreno.size(); i++)
19         cout << sensores.terreno[i];
20     cout << endl;
21
22     cout << "Superficie: ";
23     for (int i=0; i<sensores.superficie.size(); i++)
24         cout << sensores.superficie[i];
25     cout << endl;
26
27     cout << "Colisión: " << sensores.colision << endl;
28     cout << "Mochila: " << sensores.mochila << endl;
29     cout << "Reset: " << sensores.reset << endl;
30     cout << "Vida: " << sensores.vida << endl;
31     cout << "objetoActivo: " << sensores.objetoActivo << endl;
32     cout << endl;
33
34     accion = accFORWARD;
35
36     return accion;
37 }
38
39 int ComportamientoJugador::interact(Action accion, int valor){
40     return false;
41 }
```

Asignamos avanzar
(actFORWARD) a la
variable accion

4. Implementación de un agente

4.3. Modificando el comportamiento del agente: ejemplo ilustrativo 3.

*Los árboles parece que son un obstáculo para el agente.
Intentemos evitarlos....*

Sustituir la sentencia anterior por la siguiente:

```
if (sensores.terreno[2]== 'B'){ //Si lo que tengo en frente es un árbol  
    accion = actTURN_L;          // Gira a la izquierda  
}  
else{  
    accion = actFORWARD;        // Si no avanza  
}
```


Índice

1. Introducción
2. Presentación del Problema
3. Presentación del Simulador
4. Implementación de un agente
5. Evaluación de la práctica

5. Evaluación de la práctica

1. ¿Qué hay que entregar?
2. ¿Qué debe contener la memoria de la práctica?
3. ¿Cómo se evalúa la práctica?
4. ¿Dónde y cuándo se entrega?

5. Evaluación de la práctica

¿Qué hay que entregar?

Un único archivo comprimido zip que llamado “***practica1.zip***” contenga sin carpetas los tres ficheros siguientes:

- La memoria de la práctica (en formato pdf)
- Los ficheros “***jugador.cpp***” y “***jugador.hpp***” propuestos como solución.

No ficheros ejecutables

5. Evaluación de la práctica

¿Qué debe contener la memoria de la práctica?

Descripción de los comportamientos implementados en la propuesta y como se combinan entre ellos.

Documento 5 páginas máximo

5. Evaluación de la práctica

¿Cómo se evalúa?

La capacidad de exploración del comportamiento definido para el agente en un conjunto de mapas. Para ello, se probará el agente sobre una serie de mapas y alcanzará una puntuación en base al porcentaje de descubrimiento medio.

Por otro lado, la puntuación anterior se matizará con la complejidad introducida por el estudiante. Este último aspecto se evalúa en una sesión de defensa de prácticas.

5. Evaluación de la práctica

¿Dónde y cuándo se entrega?

- Se entrega en la aplicación de gestión de prácticas de la asignatura decsai.ugr.es → Entrega de Prácticas

Grupos	Fecha límite
A3, B3, C3 y D2	22 de Marzo hasta las 23:00 horas
A1, B1 y Doble Grado	23 de Marzo hasta las 23:00 horas
C1	20 de Marzo hasta las 23:00 horas
A2, B2, C2 y D1	21 de Marzo hasta las 23:00 horas