Inteligencia Artificial

Seminario 1: Agentes reactivos

UNIVERSIDAD DE GRANADA

/ UGR / *decsai*

Curso 2023-2024

Índice i

Profesores

Introducción

Datos de los profesores

- Nuria Rodríguez Barroso / Juan Luis Suárez Díaz
- rbnuria@ugr.es / jlsuarezdiaz@ugr.es
- Tutorías: En cualquier horario (razonable) mediante:
  - Presencial en el edificio UGR-IA o CITIC (previo aviso).
  - Telegram: @RBNuria / @jlsuarezdiaz
  - Grupo de Telegram: `https://t.me/+awPZ7OUMk3Y1NTQ0`

Evaluación

Evaluación

La evaluación de las prácticas se hará teniendo en cuenta los siguientes criterios:

- Práctica 1 - agentes reactivos: 25 %
- Práctica 2 - resolución de prob. con agentes reactivos/deliberativos: 25 %
- Práctica 3 - resolución de juegos con técnicas de búsqueda: 25 %
- Examen de problemas: 25 %

Consideraciones adicionales:

- No es obligatorio entregar todas las prácticas.
- Es necesario alcanzar un 3 para poder hacer media con teoría.

Temporización

Temporización

Índice de sesiones (con posibles cambios):

Práctica	Sesión
práctica 1	7/8 de marzo - Presentación
	14/15 de marzo
	21/22 de marzo - Relación 1 problemas
	4/5 de abril
práctica 2	11/12 de abril - Presentación
	18/19 de abril
	25/26 de abril - Relación 2 problemas
	2/3 de mayo

Temporización (II)

Práctica	Sesión
práctica 3	9/10 de mayo - Presentación
	16/17 de mayo - Relación 3 problemas
	23/24 de mayo

Introducción

Introducción

Un agente reactivo es un proceso que toma decisiones en base a las condiciones actuales del ambiente en que se encuentra.

Flujo de Control

Tres subtareas se pueden distinguir en un agente reactivo: la de percepción, la de decisión y la de actuación.

Introducción - flujo de control

De forma esquemática, en cada una de las subtareas del flujo control las operaciones a realizar son:

Sistema sensorial		Actuadores
visión		
ubicación		
colisión		avanzar
reinicio		correr
batería		girar 45° a la derecha
tiempo		girar 90° a la izquierda
nivel		no hacer nada
Percibir	**Decidir**	**Actuar**

Introducción - flujo de control

Nuestro objetivo se centrará en la parte de decisión

Sistema sensorial	Comportamiento	Actuadores
visión	**Diseñar e implementar**	
ubicación	**un modelo de decisión**	
colisión	**para un agente reactivo**	avanzar
reinicio	**con el objetivo de recoger**	correr
batería	**la mayor información posible**	girar 45° a la derecha
tiempo	**sobre cómo es el mundo**	girar 90° a la izquierda
nivel	**que le rodea**	no hacer nada
Percibir	**Decidir**	**Actuar**

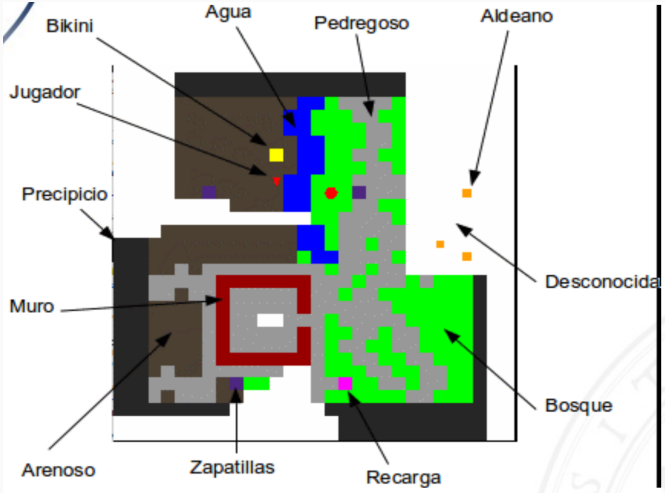
Presentación del juego

Presentación del juego: mundo

Las características del mundo en que se desarrolla el juego son:

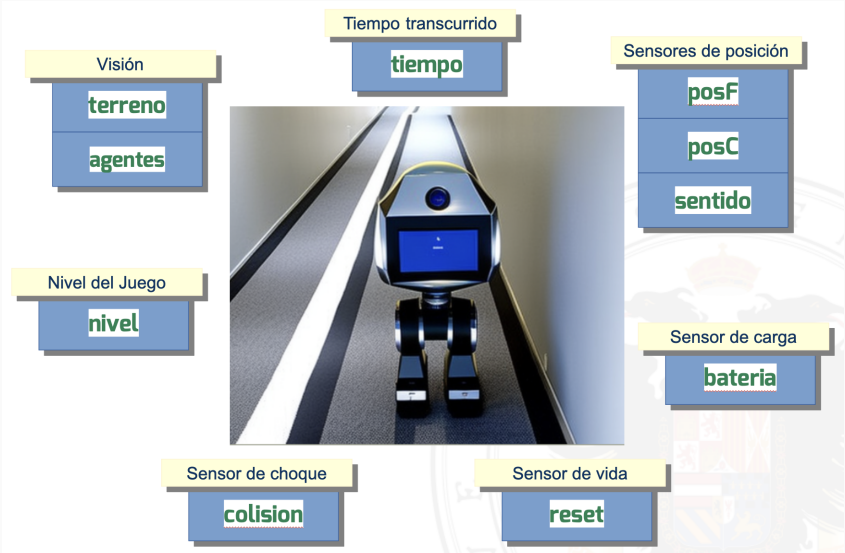
- es un tablero de celdas, con tamaño máximo de 100 filas y 100 columnas.
- cada celda es de un determinado tipo de terreno y puede contener además objetos y personajes.
- las leyes del movimiento indican que:
  - el agente que controlamos se reinicia si cae en un precipicio
  - no puede atravesar muros ni ocupar la posición de otros personajes
  - el choque con un lobo también produce el reinicio

Presentación del juego: mundo



	‘B’	Árboles
	‘A’	Agua
	‘P’	Precipicios
	‘S’	Suelo pedregoso
	‘T’	Suelo arenoso
	‘M’	Muros
	‘K’	Bikini
	‘D’	Zapatillas
	‘X’	Recarga
	‘G’	Posicionamiento
	‘?’	Casillas desconocida

Presentación del juego: sistema sensorial

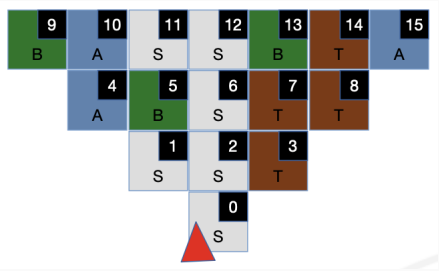


Presentación del juego: sistema sensorial

El sistema de visión ofrece información sobre:

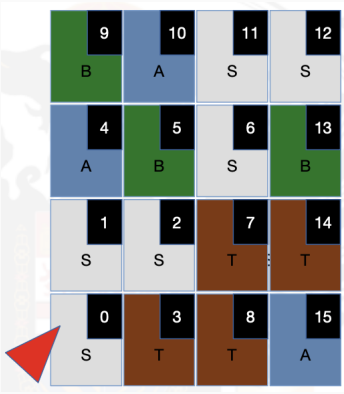
- **terreno**: tipo de terreno de las celdas que percibe el agente (A: agua; B: bosque; T: terreno pedregoso; S: terreno arenoso; ...)
- **agentes**: presencia de otros agentes en las celdas sobre las que hay visión (a: aldeano; l: lobo; _: desocupado)

Presentación del juego: sistema sensorial



norte, sur

este, oeste



noreste, sureste

suroeste, noroeste

Presentación del juego: acciones

Las acciones posibles son:

- **actWALK**: permite avanzar a la siguiente casilla, siguiendo la orientación actual. Para que la operación se finalice con éxito es necesario que la casilla de destino sea transitable para el agente.
- **actRUN**: permite avanzar dos casillas del mapa siguiendo la orientación actual. Para que la operación se finalice con éxito es necesario tanto que la casilla destino como la anterior al destino sean transitables. La casilla intermedia solo sirve de paso y no se recoge ningún objeto que pudiera haber en ella.

Presentación del juego: acciones

- **actTURN_SR**: le permite mantenerse en la misma casilla y girar a la derecha 45 $^{\circ}$ teniendo en cuenta su orientación.
- **actTURN_L**: le permite mantenerse en la misma casilla y girar a la izquierda 90 $^{\circ}$ teniendo en cuenta su orientación.
- **actIDLE**: no hace nada.

Presentación del juego: coste de las acciones

Cada acción realizada por el agente tiene un coste en tiempo de simulación y en consumo de batería:

- **tiempo**: todas las acciones consumen un instante independientemente de la acción que se realice y del terreno donde se encuentre el jugador.
- cada simulación tiene una duración de 3000 instantes de tiempo.

Presentación del juego: coste de las acciones

El consumo de batería depende de la acción y del terreno en que se realiza y también de si se dispone o no de determinados objetos.

Hay que considerar que no hay sensor para detectar si el agente dispone o no de objetos (bikini, zapatillas), por lo que habrá que crear variables de estado para disponer de esta información.

Presentación del juego: coste de las acciones

actWALK

tipo de casilla	gasto normal	gasto reducido
A	50	5 (con bikini)
B	25	5 (con zapatillas)
T	2	2
resto de casillas	1	1

actRUN

tipo de casilla	gasto normal	gasto reducido
A	500	10 (con bikini)
B	250	15 (con zapatillas)
T	3	3
resto de casillas	1	1

Presentación del juego: coste de las acciones

actTURN_SR

tipo de casilla	gasto normal	gasto reducido
A	50	5 (con bikini)
B	10	1 (con zapatillas)
T	2	2
resto de casillas	1	1

actTURN_L

tipo de casilla	gasto normal	gasto reducido
A	50	5 (con bikini)
B	20	1 (con zapatillas)
T	2	2
resto de casillas	1	1

Presentación del juego: objetivo

El objetivo consiste en definir un comportamiento reactivo para nuestro personaje con la idea de:

- obtener una orientación correcta
- reconocer el máximo porcentaje posible del mundo
- evitar situaciones de reinicio del agente para no perder información ya adquirida

Para ello se dispone del tiempo de simulación. El agente se enfrenta a 4 niveles de dificultad para probar el comportamiento definido.

Presentación del juego: objetivo

Los niveles del juego se describen a continuación:

- **nivel 0**: todo el sistema sensorial funciona correctamente.
- **nivel 1**: no funcionan los sensores de posicionamiento (los sensores **posF** y **posC** devuelven el valor -1 y **sentido** devuelve el valor norte. Para situarse en el mapa el agente debe alcanzar una casilla de **Posicionamiento** para conseguir valores correctos de los sensores.
- **nivel 2**: al igual que en el nivel anterior, pero hay activos agentes de **aldeanos y lobos**.
- **nivel 3**: igual que en el nivel 2, con **aldeanos** y **lobos** moviéndose por el mundo, pero ahora no se conoce inicialmente la orientación del agente. Además, el sensor de terreno no ofrece información sobre el estado de las casillas 6, 11, 12 y 13.

Presentación del simulador

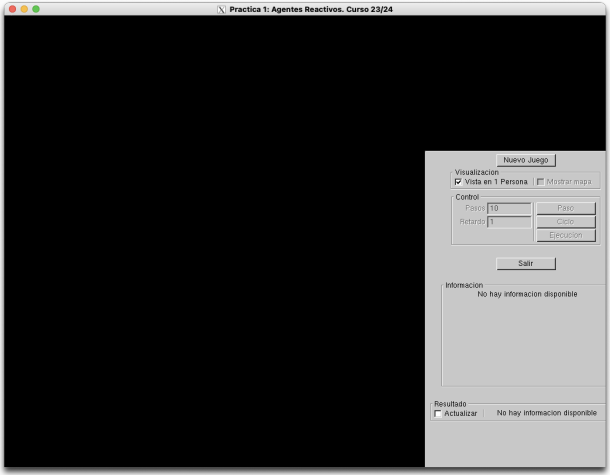
Presentación del simulador

El software se proporciona para el sistema operativo **linux** en el repositorio de **GitHub** `https://github.com/ugr-ccia-IA/practica1`.

- existen dos tipos de simuladores: uno con interfaz gráfica (ejecutable **practica1**) y otro en modo *batch* sin interfaz (ejecutable **practica1G**)
- todos los detalles de la instalación, uso y detalles de las variables necesarias para el desarrollo se encuentran en el guion de prácticas también disponible en el repositorio de **GitHub** (en la parte de instalación)

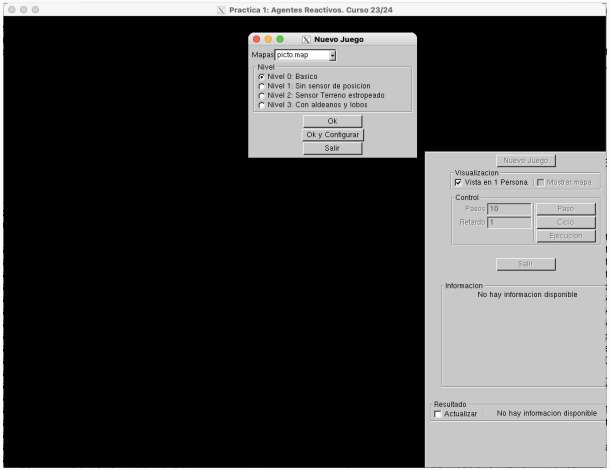
Presentación del simulador

La ventana inicial del simulador es la siguiente:



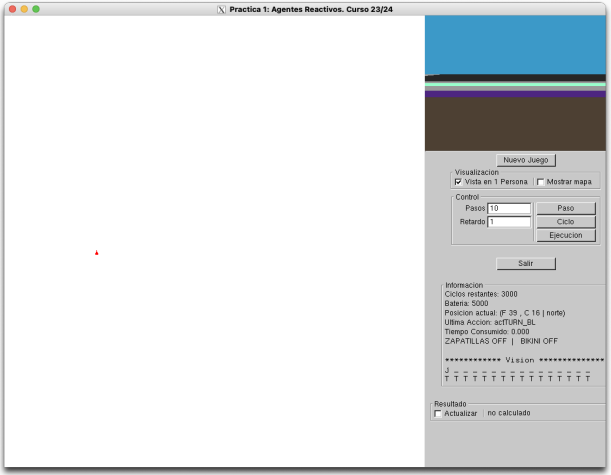
Presentación del simulador

Al pulsar sobre **Nuevo Juego** aparece un diálogo para selección del mapa:



Presentación del simulador

Tras seleccionar el mapa y el nivel el simulador quedaría de la siguiente forma:



Presentación del simulador

El sistema **batch**, sin interfaz, se lanzaría de la siguiente forma:

```
./practica1SG mapas/mapas30.map 1 0 4 5 1
```

donde los argumentos son:

- fichero con el mapa a usar
- semilla para el generador de números aleatorios
- nivel de dificultad: 0, 1, 2, 3
- fila origen
- columna origen
- orientación inicial: 0 (norte), 1 (noreste), 2 (este), ... 7 (noroeste)

Se incluye esta opción para acelerar la ejecución completa de la simulación y para facilitar la depuración de errores.

Presentación del simulador

Al final de la ejecución se proporciona información sobre:

- tiempo consumido
- batería restante al final
- número de colisiones sufridas
- número de reinicios
- porcentaje del mapa descubierto

Presentación del simulador

También se puede usar una versión combinada del sistema **batch** con el entorno gráfico:

```
./practica1 mapas/mapas30.map 1 0 4 5 1
```

- el orden de los parámetros es el mismo
- es una forma simple de fijar las condiciones del simulador sin tener que navegar por el sistema de ventanas
- los resultados que se obtienen deben ser los mismos que los producidos por la versión **batch**

Implementación de un agente

Implementación de un agente

Hay que tener en cuenta las siguientes observaciones:

- solo hay dos archivos 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 se pueden modificar y se han incluido para poder hacer la compilación
- la compilación genera dos ejecutables:
  - **practica1**: simulador con entorno gráfico
  - **practica1SG**: simulador sin entorno gráfico

Implementación de un agente

```
1  #ifndef COMPORTAMIENTOJUGADOR_H
2  #define COMPORTAMIENTOJUGADOR_H
3  #include "comportamientos/comportamiento.hpp"
4  using namespace std;
5
6  class ComportamientoJugador : public Comportamiento{
7    public:
8      ComportamientoJugador(unsigned int size) : Comportamiento(size){
9          // constructor de la clase
10         // dar aqui valor inicial a las variables de estado
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   private:
21   // Declarar aquí las variables de estado
22 };
23
24 #endif
```

Implementación de un agente

En el código aparecen:

- constructor de la clase
- constructor de copia
- destructor
- método **think**: método donde definir el comportamiento del agente. La implementación se hace en el archivo **jugador.cpp**

Implementación de un agente i

```
#include "../Comportamientos_Jugador/jugador.hpp"
#include <iostream>
using namespace std;

Action ComportamientoJugador::think(Sensores sensores) {
   Action accion = actIDLE;

   cout << "Posicion: fila " << sensores.posF << " columna " << sensores.posC << " ";
   switch(sensores.sentido){
      case 0: cout << "Norte" << endl; break;
      case 1: cout << "Noreste" << endl; break;
      case 2: cout << "Este" << endl; break;
      case 3: cout << "Sureste" << endl; break;
      case 4: cout << "Sur" << endl; break;
      case 5: cout << "Suroeste" << endl; break;
      case 6: cout << "Oeste" << endl; break;
      case 7: cout << "Noroeste" << endl; break;
   }
   cout << "Terreno: ";
   for(int i=0; i < sensores.terreno.size(); i++)
      cout << sensores.terreno[i];
```

Implementación de un agente ii

```
22      cout << endl;
23
24      cout << "Agentes: ";
25      for(int i=0; i < sensores.agentes.size(); i++)
26         cout << sensores.agentes[i];
27      cout << endl;
28
29      cout << "Colision: " << sensores.colision << endl;
30      cout << "Reset: " << sensores.reset << endl;
31      cout << "Vida: " << sensores.vida << endl;
32      cout << endl;
33
34      // determinar la siguiente accion a realizar
35      // y devolverla
36      return accion;
37  }
38
39  int ComportamientoJugador::interact(Action accion, int valor) {
40      return false;
41  }
```

Implementación de un agente

A tener en cuenta:

- **think** es un método que toma como entrada el estado sensorial del agente y devuelve como salida la siguiente acción a realizar
- por tanto, es el método encargado de decidir el comportamiento del agente
- en su versión inicial simplemente muestra la forma en que hay que invocar a los distintos sensores del agente
- existe una matriz llamada **mapaResultado** donde se ha de colocar lo que se ha descubierto del mapa
- se recomienda hacer el tutorial que se adjunta como material adicional a la práctica

Método de evaluación de la práctica

Método de evaluación de la práctica

En esta práctica:

- se pide desarrollar un programa (modificando el código de los fichero del simulador **jugador.cpp** y **jugador.hpp**) con el comportamiento requerido para el agente
- estos dos ficheros se entregan mediante la plataforma web de la asignatura, en un fichero llamado **practica1.zip**, que no contendrá ni carpetas ni subcarpetas
- el archivo **zip** deberá contener solo el código fuente de estos dos ficheros con la solución del alumno

Método de evaluación de la práctica

La forma de evaluación es la siguiente:

- se tomarán 3 mapas semejantes a los que se proporcionan como material de la práctica
- sobre cada mapa se aplicará el agente propuesto por el estudiante para los 4 niveles (del 0 al 3)
- sobre cada nivel y mapa se realizará una simulación; llamaremos s_i al valor devuelto por el software de porcentaje de mapa descubierto
- si la simulación no termina correctamente por alguna razón (normalmente por un error de segmentación y que en general denominaremos **core**) el valor asignado será $s_i = -0.1$

Método de evaluación de la práctica

- la nota asociada a cada mapa se calcula como:

$$nota = s_0 p_0 + s_1 p_1 + s_2 p_2 + s_3 p_3$$

siendo p_i la puntuación asociada al nivel i y que se describe en la siguiente tabla:

nivel	0	1	2	3
puntuación	2	3	2	3

siendo s_i el grado de descubrimiento del mapa descubierto en el nivel i. Este valor se calcula mediante la siguiente fórmula:

$$s_i = min\,[1.0, (aciertos - errores)/(totaldecasillas \times 0.85)]$$

Método de evaluación de la práctica

Para el cálculo de s_i se tiene en cuenta que cada casilla situada en **mapaResultado** que no coincida con el mapa original resta un acierto.

La nota final es la media artimética de las obtenidas en los 3 mapas.

Método de evaluación de la práctica

Fechas importantes:

- fecha límite de entrega: 7 de abril a las 23.00 horas
- cuestionario de evaluación: disponible desde el 8 de abril al 10 de abril a las 23.00 horas

Muy importante:

- **esta práctica es individual**
- **en caso de detectar práctica copiadas, los involucrados (tanto el que se copia como el que se dejó copiar) tendrán suspensa la asignatura**.