uc3m

Enunciado del proyecto final

CONTROL DE UN DRON DE RECONOCIMIENTO

Curso Inteligencia Artificial
Grado en Ingeniería Informática
Curso 2017-18

Esquema y funcionamiento general

El software entregado consta de los siguientes elementos:

1. Librería SIMPLE-AI, implementa los algoritmos de búsqueda
2. Librería GAME-AI, el componente gráfico, consta de varios ficheros que no deben modificarse
3. Clase GAMEPROBLEM, implementada por el alumno

Para ejecutar el juego hay que teclear `python startGame.py`.
El componente gráfico interactúa con el componente del alumno de la siguiente secuencia:

1. En la inicialización del juego, se crea un objeto de clase *gameProblem*
2. A continuación, se invoca *setup()*, donde el alumno tendrá que especificar el estado inicial, final y algoritmo de búsqueda
3. Se realiza la búsqueda invocando el algoritmo especificado y obtiene una solución o no. Si no hay solución, se detiene el proceso. En esta fase se generan estadísticas del proceso de búsqueda.
4. Para visualizar la solución en funcionamiento, el componente gráfico muestra el movimiento del agente en el mapa, haciendo uso para ello de las acciones predefinidas: **North**,**East**,**South**,**West**. De este modo se puede observar la trayectoria del agente sobre el mapa. Cualquier otra acción de la solución es ignorada por el componente gráfico.

La salida se debe capturar a fichero para analizar las estadísticas:

```
Total length of solution: 47
Total cost of solution: 51
visited nodes: 8001
iterations: 8001
max fringe size: 554
```

Variable *config.configuration*: Inicialización del juego

El fichero *config.py* contiene la definición de una variable *configuration*, con varios campos que definen el tablero y otros parámetros del juego (Tabla 2). El alumno puede modificar la especificación de las casillas, incluyendo si lo desea otras nuevas, utilizando los campos predefinidos (ver Tabla 1).

```
"plains": {
    "img": "game/graphics/terrains/plains100.png",
    "id":  "plains",
    "marker": '.',
    "num": 0,
    "attributes":
         {"agent":None }
    },
"plains-traversed": {
    "img": "game/graphics/terrains/plainsTraversed100.png",
    "id":  "plains-traversed",
    "marker": '.',
    "num": 0,
    "attributes":
         {"agent":None }
    },
```

Cada casilla tiene dos versiones, cuyos atributos deben ser idénticos, ya que únicamente se usan para conmutar el color de la casilla cuando el agente pasa por encima. En cada caso, los campos significan:

La casilla "básica" se define en el campo "basicTile" y no se usa el campo número: el tablero contiene casillas de este tipo en todas las posiciones restantes tras colocar las casillas de otros tipos.

La clase *gameProblem* contiene el método `self.getAttribute(posicion,clave)`, que permite acceder a los atributos de una casilla.

Tabla 1: Descripción de los campos de la configuración de casilla

Campo	Tipo	Descripción
img	String	Fichero que contiene la imagen usada para la casilla
id	String	Identificador de la casilla en las variables MAP y POSITIONS.
marker	Char	Caracter utilizado para simbolizar la casilla al volcar o leer el mapa a fichero
num	Int	Número de casillas de este tipo presentes en el mapa
attributes	Dict.	Diccionario de atributos, debe contener al menos el atributo "agent" siempre inicializado a `None`

Clase *gameProblem*: Documentación de las variables

El objeto *gameProblem* contiene las siguientes variables internas, que se pueden usar en los métodos del alumno. Estas variables se inicializan de forma automática en el código proporcionado con la clase (Tabla 3).

A efectos de la implementación de la búsqueda, las coordenadas `(x,y)` se interpretan como primer índice y segundo índice de la variable MAP, que contiene un elemento por casilla, y corresponden con la coordenada X e Y tradicionales sobre la representación gráficas, en la que la casilla `(0,0)` se muestra en la esquina inferior izquierda.

Métodos a codificar por el alumno

El alumno debe codificar las siguientes funciones, que aparecen marcadas como "Modificables por el alumno", en el fichero *gameProblem*. Corresponden con las funciones necesarias para la búsqueda en SIMPLE-AI. Los tipos de los datos deben ser los siguientes:

- `state`, codifica un estado, debe ser inmutable (strings,números,tuplas)
- `action`, codifica una acción, debe ser inmutable (strings,números,tuplas)
- `c`, coste, debe ser un valor numérico mayor que 0
- `h`, heurística, debe ser un valor numérico mayor o igual a 0

```
def actions(self, state):
   l = []
   return l # Lista de acciones

def result(self, state, action):
   s = ()
   return s # estado (inmutable, p.ej. una tupla)

def is_goal(self, state):
   b = False
   return b # Booleano

def cost(self, state, action, state2):
   c = 1.0
   return c # Coste (>=0)

def heuristic(self, state):
   h = 0.0
   return h # Heuristica en el estado state (>=0)

def setup (self):
    # Estado,EstadoObjetivo y funcion de busqueda
    return initial_state,final_state,algorithm
```

Estas funciones pueden (y deben) usar las variables del objeto. Recuérdese que para acceder a las mismas hay que usar la sintaxis: `self.` (ej: `self.INITIAL_STATE`).

Se proporciona también un método `self.getAttribute(posicion,clave)` que devuelve el valor de uno de los atributos de una casilla, identificado por su clave, según está definido en el fichero de configuración.

Tabla 2: Descripción de los campos de la variable *configuration*

Campo	Tipo	Descripción
"text_size"	Int.	Tamaño del texto
"tile_size"	Int.	Tamaño de las casillas
"type"	"random"	Genera un mapa aleatorio
	"read"	Carga mapa desde fichero
"seed"	Int.	Semilla del generador de números aleatorios
"file"	String	Fichero para cargar o salvar mapa
"save"	Boolean	True, salva mapa a fichero
"showState"	Boolean	True muestra la acción y estado en cada paso
"map_size"	Tupla (Int,Int)	Tamaño del mapa (columnas, filas)
"delay"	Float	Velocidad de representación (0.1)
"debugMap"	Boolean	True, vuelca la variable MAP
"basicTile"	String (key)	Casilla por defecto usada en el tablero (debe estar declarada)
"agentInit"	Tupla	Posición inicial del agente (X, Y)
"agentBaseTile"	Clave	Casilla de base del agente
"agentType"	Clave	Tipo de agente (no modificar)
"agentMarker"	Caracter	Marcador del agente en el mapa salvado
"debug"		No modificar
"hazards"		No modificar

Tabla 3: Descripción de las variables del objeto *gameProblem*

Variable	Tipo	Descripción
MAP	Lista	Mapa del tablero. El contenido de cada casilla es una lista:
	[0]	Tipo de casilla
	[1]	Número de la casilla
	[2]	Atributos de la casilla, cargados desde el fichero de configuración
POSITIONS	Diccionario	Contiene claves para todas los tipos de casilla del tablero
	key	Tipo de casilla
	value	Lista de posiciones de las casillas del tipo
CONFIG	Diccionario	Configuración leída del fichero
AGENT_START	Tupla	Posición inicial del agente (viene del fichero de configuración)
	[0]	Posición X
	[1]	Posición Y
INITIAL_STATE	Def.Alumno (Inmutable)	Estado inicial, se carga en la inicialización
GOAL	Def.Alumno (Inmutable)	Estado objetivo, se carga en la inicialización y se comprueba en *is_goal()*