Universidad
Rey Juan Carlos

MEMORIA IA PRACTICA 1

DESARROLLO DE JUEGOS CON INTELIGENCIA ARTIFICIAL
Diseño y Desarrollo de Videojuegos
Campus Madrid-Quintana
2021-2022

Trabajo realizado por:
Javier Picado Hijón y Pablo Álvarez de Lara Fernández (Grupo 15)

Índice

1. Introducción

Este documento sirve como memoria del desarrollo realizado para la práctica uno de la asignatura de *Desarrollo de Juegos con Inteligencia Artificial*.

En esta se presentará primeramente una descripción del algoritmo implementado para el método de búsqueda del agente inteligente (el personaje). Ahí se contará tanto el algoritmo que se ha escogido para la búsqueda que realizará el agente, como el por qué de esta decisión. Además, se añadirá una explicación de cómo se ha implementado el algoritmo dentro del entorno de Unity3D. A continuación, se comentarán de manera detallada las decisiones de diseño que se han tomado en la práctica, así como los problemas que han surgido durante el trabajo en esta y la resolución a la que se ha llegado para estos. Finalmente, se incluirá también una breve conclusión en la que se comentarán ciertos aspectos notables e ideas a las que se llegaron tras realizar el trabajo.

2. Descripción del algoritmo

Para esta práctica se pedía implementar un algoritmo de búsqueda de caminos offLine, con algún tipo de heurística, que permitiese al personaje llegar hasta un objetivo fijo (goal). Por ello, se ha decidido implementar un algoritmo de A* ya que es uno de los algoritmos más robustos, completos y sencillos de implementar. Además se ha usado la distancia Manhattan como heurística, que al ser optimista (ya que el movimiento en el mundo está modelizado en casillas con posibles jugadas únicamente verticales y horizontales), hace que el algoritmo sea también óptimo.

Para empezar la implementación, se creó la clase nodo, que contiene los atributos miCelda, que contiene la información de la celda a la que se asocia, un nodo padre, un valor de h*, que guarda la distancia Manhattan al goal, una g, que almacena la profundidad acumulada hasta ese nodo del grafo, una dirección que indica hacia donde se tiene que mover el personaje desde el nodo padre para llegar a él, y un f* que sirve como heurística para el algoritmo. Además esta clase cuenta con un constructor y los métodos de acceso pertinentes.

Clase nodo:

```
namespace Assets.Scripts.DataStructures{
    public class Nodo
    {

        private Nodo padre; // nodo padre
        private CellInfo miCelda; // info de la celda
        private int hEstrella; // distancia manhattan
        private float g; // camino recorrido
        private Locomotion.MoveDirection movimiento; // direccion de movimiento
        private float fEstrella; // (h* + g)

        //  CONSTRUCTOR
        public Nodo(CellInfo infoCelda){
                this.miCelda = infoCelda;
            }
}
```

El atributo h* de la clase nodo, como se ha explicado anteriormente, almacena la distancia Manhattan que hay desde el nodo actual hasta la casilla goal. Para ello se realiza la suma de los valores absolutos de las diferencias que hay entre las filas y las columnas de ambos nodos. A estos elementos se accede por medio de los atributos RowId y ColumnId, respectivamente, de las casillas asociadas.

Algoritmo para el cálculo de h*:

```
private int hEstrella(Nodo ex, CellInfo[] goals){

                // estrella da el valor del h* del nodo ex respecto al goal posicionado en el índice 0 del array goals
                // este número se calcula haciendo la distancia manhattan
                int estrella=(System.Math.Abs(goals[0].ColumnId - ex.getCelda().ColumnId))+(System.Math.Abs(goals[0].RowId - ex.getCelda().RowId));

                // devolvemos el valor de h*
                return estrella;
        }
```

Por otro lado, el atributo f* también debía ser calculado para poder ordenar la lista abierta. Para ello se implementó un setter para f* al que se le pasa como atributo una suma la h* del nodo en el que se encuentra el personaje y el coste del camino que lleva hasta ese momento, la g. Esta operación se realiza durante la parte del código que extiende los hijos del nodo actual, para así poder ordenar la nueva lista abierta en función de este atributo.

Algoritmo para el cálculo de f*:

```
nuevoVecino.sethEstrella(hEstrella(nuevoVecino, goals));
                        nuevoVecino.setPadre(nodoPrimero);
                        nuevoVecino.setG((nuevoVecino.getPadre().getG()) + nuevoVecino.getCelda().WalkCost);

nuevoVecino.setfEstrella(nuevoVecino.getG()+nuevoVecino.gethEstrella());
```

3. Características de diseño e implementación

A continuación, se van a comentar los aspectos más relevantes del diseño y la implementación realizados, en base a las situaciones que se nos han planteado a lo largo del desarrollo de la práctica.

Uno de los principales obstáculos con el que nos encontramos es que al principio se nos olvidó establecer el valor de g dentro de nuestra función f*. esto hacía que el personaje se chocara con las paredes y no completará de manera correcta el camino que debería seguir el A*. Para arreglarlo, nos dimos cuenta del atributo .walkcost que tenían las celdas. Este nos permitía acceder al valor de camino que poseían las celdas, subiendo a un valor extremadamente alto si no eran accesibles.

La otra mayor dificultad que nos ralentizó el proceso fue un fallo por no entender muy bien el proceso del programa. A la hora de empezar el while del método search, debíamos devolver el primer elemento de la lista open, pero se nos olvidó, haciendo que el camino fuese erróneo. Tenía una solución muy fácil: crear un nodo auxiliar que recogiese los datos del primer nodo de la lista open y luego borrar el nodo de la lista, así podemos continuar operando con sus datos aunque se haya borrado el nodo.

4. Conclusión

Con esta práctica hemos podido ver el cómo se puede implementar un algoritmo como A* de manera bastante fácil. Al principio, nos costó un poco entender cómo funcionaban los métodos del juego, qué hacían sus atributos y cómo acceder a ellos. Al final, mediante una extensa lectura de los scripts implementados, dibujamos un plan para proceder a la implementación del ejercicio de la práctica. Gracias a esto, hemos afianzado nuestro conocimiento sobre el algoritmo A* para el futuro.

Además, esta práctica nos ha permitido aprender cómo trabajar en el entorno de Unity, y, siendo un motor bastante utilizado en la industria de los videojuegos, creemos que nos va a ayudar mucho de cara al mundo laboral.

En resumen, el mayor problema de todos ha sido comprender cómo funcionaba el programa por dentro para poder utilizar sus datos a la hora de implementar el algoritmo. La implementación de este, sin embargo, ha resultado bastante sencilla y directa una vez se ha entendido el entorno.