

**PRÁCTICA 1:**

PRÁCTICA DEL BLOQUE DE APRENDIZAJE AUTOMÁTICO

*David María Arribas*

*Alejandro Rodríguez Muñoz*

*Grado en diseño y desarrollo de videojuegos*

*Año académico: 2019/2020*

*Asignatura: Desarrollo de juegos con inteligencia artificial*

**Índice**

1. Descripción del algoritmo empleado para solucionar el problema

2. Características de diseño e implementación

3. Discusión sobre los resultados obtenidos.

**1. Descripción del algoritmo implementado**

Hemos implementado un algoritmo de q-learning (aprendizaje por refuerzo), se busca la mejor acción posible para el personaje desde su posición inicial hasta la meta final. Las acciones son moverse en las cuatro direcciones posibles: arriba, abajo, derecha e izquierda y no moverse, mientras que la meta final es que el personaje consiga llegar a la casilla objetivo del mapa.

Para realizar este proceso de forma exitosa, creamos una q-table, y en ella incluimos el estado actual (la posición en la que el personaje se encuentra), las diferentes acciones antes mencionadas y el q-value, que será el que define que acción es mejor en cada posición.

Una vez tenemos la tabla creada, el programa lee dicha tabla y va eligiendo las mejores acciones a realizar desde la casilla en la que se encuentra, esta es la que tiene un mayor q-value.

Estas tablas son exportadas fuera para que podamos abrirlas y revisarlas. También hemos implementado una barra de progreso que

**2. Características de diseño e implementación**

Entendido el funcionamiento general de nuestro algoritmo de q-learning, pasamos a ver cómo funciona más detalladamente nuestra implementación del algoritmo.

Todo el código que hemos implementado se encuentra dentro de la carpeta de Scripts, en la carpeta Practica1 y en los archivos QMind.cs y QTable.cs. A continuación, pasamos a describir el archivo QMind.cs.

Nos vamos al método Learn, método que define cómo el programa genera la tabla y calcula los q-values. Lo primeo que hacemos es crear una nueva q-table y crear las variables maxQValue y totalQValue. A continuación, comenzamos el bucle de aprendizaje, al que hemos asignado 1000 episodios. Para cada episodio tomamos en la variable initialState, una casilla aleatoria dentro del mapa y comenzamos a evaluar dicha casilla. Para ello comenzamos otro bucle que buscará un camino desde la casilla que hemos tomado. Este bucle termina cuando nuestro camino nos lleve bien a la meta o bien a una casilla no navegable. Dentro del bucle tomamos una acción de forma aleatoria desde la casilla en la que estamos y tomamos el valor q de la tabla de dicha acción en esa casilla ().

Después de calcular el valor q, procedemos a asignar el siguiente estado según la dirección tomada en la variable nextState, y usamos esta para calcular la recompensa de ese estado. Esto lo hacemos en el método GetReward, el cual funciona así: si la casilla es no transitable, la recompensa a devolver será de -1, si la casilla es transitable pero no es la meta, la recompensa será de 0 y si la casilla es la meta final, la recompensa que devolveremos será de 100.

A continuación, almacenamos el máximo valor q del próximo estado (siempre y cuando este no esté vacío) y procedemos a actualizar la q-table haciendo uso de esta fórmula y de los valores q obtenidos:

**(1 - alpha) \* Qactual + alpha \* (r + gamma \* MáximaQsiguiente)**

Y este es el valor que almacenamos en la q-table. Por último, actualizamos el totalQValue al cual sumamos el q-value que hemos obtenido y también el maxQValue, al cual le asignaremos el valor obtenido si este es mayor que el que había almacenado antes.

Antes de finalizar esta instancia del bucle asignamos el nextState a nuestro initialState para continuar el bucle desde el nuevo estado obtenido. Este bucle termina cuando nuestro camino nos lleve bien a la meta o bien a una casilla no navegable.

Al terminar este camino, terminamos un episodio, realizaremos hasta 1000 episodios antes de finalizar nuestra q-table.

El método Learn se ejecutará si nuestra q-table está vacía.

También podemos exportar las tablas al formato csv (que nos permite poder abrirlas y revisarlas con Microsoft Excel o similares) y leer tablas de dicho formato e importárselas al programa.

**3. Discusión sobre los resultados obtenidos**

Según hemos podido observar nuestra implementación cumple los requisitos pedidos, el algoritmo es capaz de encontrar el camino a la meta tras haber generado la tabla.