PRÁCTICA2 IA

ÍNDICE

Javier Castro Magro

Raúl Somavilla González

1. INTRODUCCIÓN

Este informe trata sobre la práctica que se ha desarrollado y que implica implementar técnicas de aprendizaje automático para crear un agente inteligente. Este proyecto trata sobre el uso de **q-learning**, un algoritmo de aprendizaje por refuerzo, para entrenar el comportamiento del agente. **El objetivo principal es que el agente llamado "*Agent*" debe aprender a huir del agente "*Player*".**

Se utiliza el motor de videojuegos ***unity*** y el lenguaje **C#**.

Para alcanzar el objetivo, se han desarrollado las clases y pruebas necesarias que le han permitido al agente aprender y ejecutar el comportamiento deseado.

Después de completar el desarrollo y el entrenamiento, el agente ha sido probado en **varios escenarios** de prueba. Esta parte es la más importante ya que es la forma de saber si el entrenamiento se está realizando de forma eficaz o no.

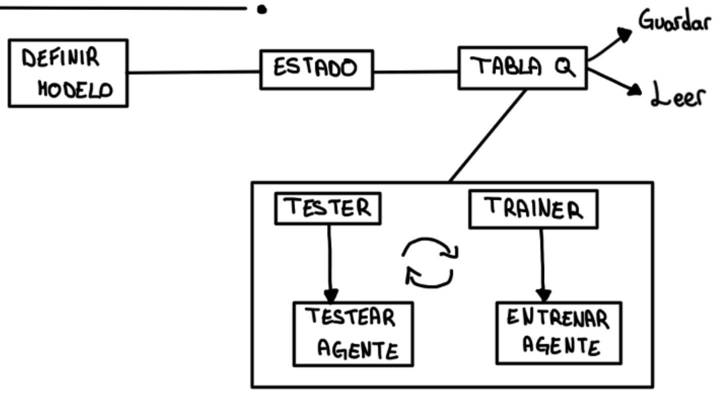
El informe resume los pasos necesarios para diseñar, implementar y probar un agente inteligente utilizando *Q-learning*, se van a destacar problemas y soluciones encontrados durante la fase de desarrollo.

2. PLANTEAMIENTO DE LA PRÁCTICA

Al igual que hicimos en la práctica anterior, antes de programar nada, se cogió "papel y boli" y se realizó un planteamiento genérico de la práctica con información y datos relevantes obtenidos de las sesiones de prácticas, tutorías e ideas planteadas con otros compañeros de clase.

A continuación, se muestra el **planteamiento** que realizamos:

Planteamiento IA



OBJETIVO: programar algoritmo de entrenamiento y completar tabla Q

- La tabla Q al comienzo completar con exploración.
  Posteriormente aprenderá en base a la experiencia que vaya obteniendo

⊕ PASOS = ⊕ NOTA

- Con la modificación de parámetros como alfa, gamma, epsilon... podemos ajustar velocidad a la que aprende, velocidad de la simulación...

- Clases: Trainer, Tester y State.

- TABLA Q:
  - Cada <u>fila</u> hace referencia a un estado — S0, S1, S2, ⋮, SN
  - Columnas: Norte, Este, Sur, Oeste → Un bool para cada uno, para determinar si hay muro.

Distancia Manhattan
- Para limitar el nº de estados → <u>discretizar</u>

{Pegado, Cerca, Medio, Lejos}   {Cerca, Medio, Lejos}   {Cerca, Lejos}

+ estados ———————— − estados

Dirección de donde viene el enemigo
× Posición del agente
nºs Posibles direcciones del player

7	0	1
6	×	2
5	4	3

sentido horario
0: N 1: NE 2: E 3: SE 4: S 5: SO 6: O 7: NO

Otra alternativa

	0	
3	×	1
	2	

0: N 1: E 2: S 3: O

Recompensas: tanto ⊕ como ⊖
Hacer pruebas para ir ajustando

ENTRENAMIENTO

Entrenar p.e. 20K episodios y guardar

Modificar escenario para que no se aprenda solo 1 ya que quedarán celdas vacías en la tabla.

Volver a entrenar 20K.

Importante ir haciendo copias de seguridad de las tablas para ir viendo los cambios y comprobar cuál es la mejor.

Ilustración 1 Planteamiento sobre papel

3. ¿CÓMO SE MANEJA EL PROGRAMA?

El programa se utiliza de la siguiente manera:

En primer lugar, si se quiere crear una nueva tabla Q, es necesario eliminar o modificar el nombre de la tabla "TablaQ". El código verifica si existe una tabla con ese nombre y, en caso afirmativo, continuará actualizándola durante el entrenamiento. Si no existe, creará una nueva tabla con los valores predeterminados establecidos.

En la escena de entrenamiento se pueden modificar parámetros que se encuentran en el gameobject vacío “QMindTrainer”. En él, se pueden ajustar los siguientes parámetros, los cuales serán detallados en el apartado [7.2 Valores empleados]:

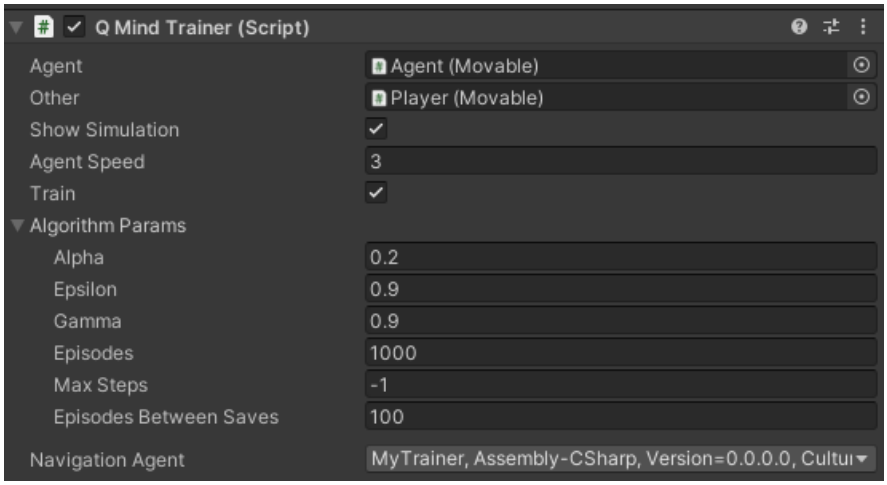


Ilustración 2 Parámetros de QMindTrainer

Una vez que se ha ejecutado el programa, dejas que el agente entrene un determinado número de episodios. Una vez entrenado pausas la simulación y se creará la tabla Q o en caso de que ya existiese una se mostrará actualizada.

Dando doble click en la tabla, se puede ver su contenido en programas que sean capaces de leer un documento .csv. Por ejemplo, en esta práctica se ha empleado **Excel**.

Una vez que verifiques los valores de los diferentes estados de la tabla, se va a la escena "TestPlayGround", donde se leerá la tabla Q y se hará una simulación leyendo los datos de la tabla. Aquí es donde puedes visualizar el resultado de tu entrenamiento y ver si tu agente consigue dar más o menos pasos.

4. MODELADO DEL ESTADO

VARIABLES EMPLEADAS

Las **posibles direcciones** que puede tomar el agente desde su posición actual: estas han sido definidas siguiendo el orden de las agujas del reloj (norte, este, sur, oeste).

Dirección actual: "direccion" indica la dirección en la que se encuentra actualmente el agente.

Distancia: "distancia" es la distancia manhattan entre la posición del agente y del player.

Estas variables se inicializan en el constructor de la clase "QState_L" y se utilizan para crear instancias de estado tanto en el agente ("MyTrainer" y "MyTester") como en la tabla Q.

MÉTODOS PRINCIPALES

Creación del estado: se crea un nuevo estado basado en la posición actual del agente, la posición del player y la información del mundo. Se calculan las posiciones adyacentes al agente y se verifica si son transitables o no.

Para calcular la **dirección del player con respecto al agente**, se han tenido en cuenta un total de 8 direcciones (N, NE, E, SE, S, SO, O, NO). Se planteó otra alternativa que se puede ver en [2. Planteamiento de la práctica] en el que solo se tenían en cuanta 4 direcciones. Esto reducía considerablemente el número de estados, por lo que

finalmente nos decantamos por la primera opción. A continuación, se muestran las diferentes direcciones:

NO	N	NE
O	X	E
SO	S	SE

Ilustración 4 Alternativa 1

	N	
O	X	E
	S	

Ilustración 3 Alternativa 2

Discretización de la distancia: se ha clasificado la distancia manhattan en un total de 4 niveles (que van de 0 a 3). Al principio de la práctica plantemos una discretización de dos niveles (cerca, lejos) o de tres (cerca, distancia media, lejos) pero esto hacía que hubiera muchos menos estados y nos estaba costando mucho más entrenar al agente.

Por lo que finalmente nos decantamos por los 4 niveles:

- Muy cerca: cuando la distancia manhattan está entre 0 y 1.
- Cerca: cuando la distancia está entre 1 y 2.
- Distancia media: entre 2 y 3.
- Lejos: Cuando se encuentra a más de tres casillas.

Esta discretización ha facilitado la construcción de una tabla Q manejable y efectiva, donde cada estado representa un conjunto discreto de condiciones del entorno.

Comparación y representación: se han implementado también métodos para comparar estados y para obtener una representación textual del estado. Esto se ha hecho para gestionar la igualdad de estados y para registrar información sobre los estados visitados.

5. RECOMPENSAS

El método “ActualizarValor” se encarga de calcular y actualizar el valor de una acción específica en un estado dado, teniendo en cuenta las posibles posiciones futuras del agente y del player. Las claves son:

Determinar la próxima posición del agente: según la acción tomada (norte, este, sur, oeste), se calcula la próxima posición del agente en el mundo.

Crear el nuevo estado: se crea un nuevo estado usando las posiciones calculadas del agente y del player.

Inicialización de la recompensa acumulada: se inicializa la variable “accumulatedReward” a 0. Esta acumulará las recompensas basadas en las siguientes condiciones:

- Si el enemigo atrapa al agente: se penaliza con -40.
- Si el agente se aleja del player: se recompensa con +20.
- Si el agente se mueve en dirección contraria al enemigo: se recompensa con +90.
- Si mantiene la distancia: se le da una pequeña recompensa de +10.
- Si se mueve hacia el enemigo: teníamos pensado penalizarlo, pero tras hacer pruebas al comienzo, daba resultados muy negativos, por lo que se ha dejado a 0.
- Si el player se acerca se penaliza con -10.

Actualización de variables: Las recompensas acumuladas se suman para obtener la recompensa de la acción tomada. Además, se actualiza “prizeSum” (suma total de recompensas) y “prizeNumber” (número total de recompensas) para calcular el promedio de las recompensas.

Con esto se obtienen los valores de la TablaQ para el estado actual y la acción específica.

6. MANEJO DE LAS TABLAS

Cuando se inicializa una tabla, primero se comprueba si existe una con el nombre "TablaQ". Se existe se lee y si no se crea una nueva.

Una vez que se termina el entrenamiento y se guarda una tabla, a esta se le establece un nombre significativo (por ejemplo, TablaQ_20kEpisodios_3500pasos).

Posteriormente, si se quiere crear una tabla nueva, no debe de existir una con el nombre "QTable", o bien establecer en el código el nombre que quieres que tenga (siempre y cuando no exista previamente)

```
if (File.Exists(folder + "TablaQ.csv"))
```

Para seleccionar una tabla para ser testeada, funciona de la misma manera, modificas la tabla a "TablaQ" para que sea leída línea a línea por:

```
using (var reader = new StreamReader(folder + "TablaQ.csv"))//
```

O bien se establece en el código el nombre de la tabla que quieres leer.

A continuación, se muestra un ejemplo de diferentes tablas con las que se inició el proyecto:

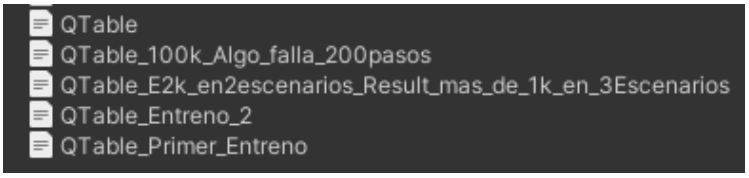


Ilustración 5 Primeras tablas del proyecto

7. ENTRENAMIENTO DEL AGENTE

7.1 DISEÑO DE MAPAS

Uno de los problemas a los que nos hemos enfrentado es que en la tabla Q había muchos estados que nunca se actualizaban (se quedaban a 0). Esto se debía a que los muros siempre estaban en la misma posición, por lo que se tomó la decisión de crear varios mapas diferentes para entrenar y testear.

Para diseñar y crear los escenarios, se ha hecho uso de unity y de Excel. Este último aprovechado para hacer uso de sus celdas.

Se cuenta con un grid de 20x20, además de un total de 9 muros de 1x5.

A continuación, se pueden identificar los mismos tanto dentro de unity como en Excel (el escenario que se muestra es el proporcionado en el proyecto base):

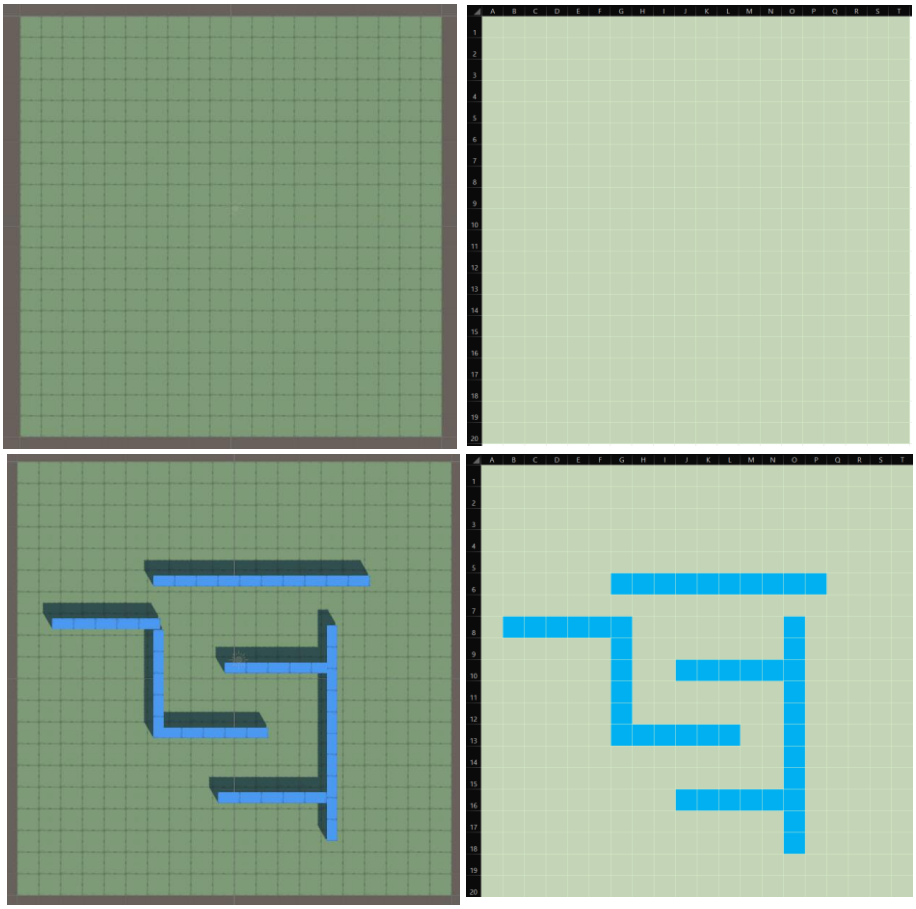


Ilustración 6 Escenarios base

El cambio de escenario también evitará que el agente se aprenda una serie de movimientos que repita todo el rato, haciendo su tarea de aprendizaje más difícil. Hemos observado que este comportamiento tiende a hacerlo en zonas amplias como las que se han marcado en **verde** (en estas pruebas el zombie no estaba muy entrenado):

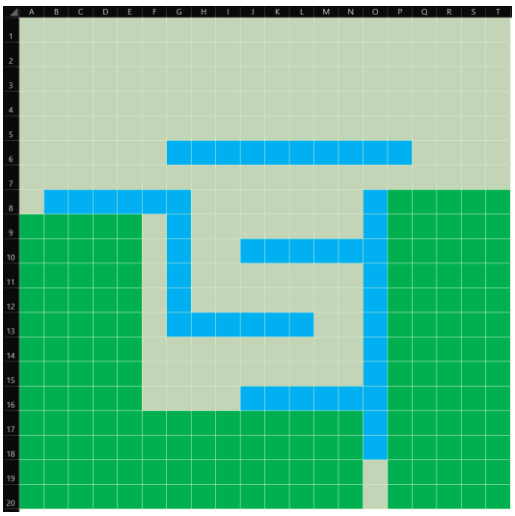


Ilustración 7 Zonas en las que ha repetido patrón de movimiento

Con poco entrenamiento, también se han detectado puntos débiles, en los que el agente tiende a tomar malas decisiones o zonas como las del centro a las que tiende a no acceder. A continuación, se muestran esas zonas marcadas en **rojo**:

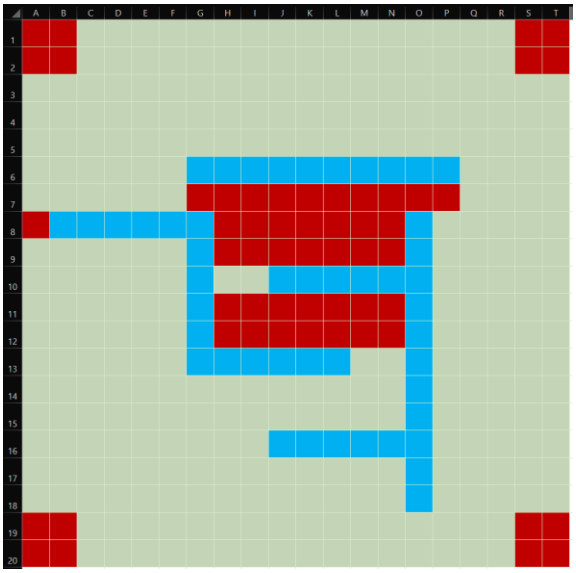


Ilustración 8 Puntos débiles

Tras haber comentado esos puntos débiles y tendencias del agente, se van a mostrar los mapas que se han diseñado tanto para entrenar como para testear. Se diseñaron primero en Excel y luego se crearon las escenas correspondientes en unity.

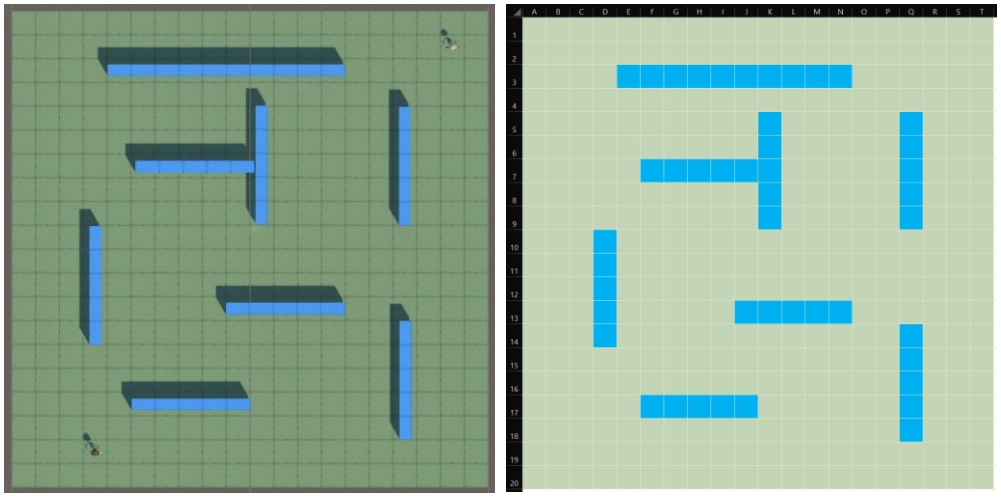


Ilustración 9 Mapa 1

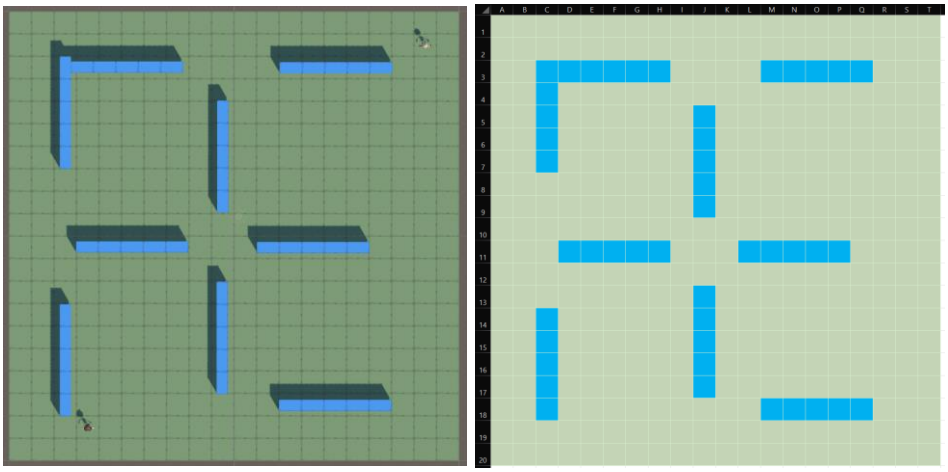


Ilustración 10 Mapa 2

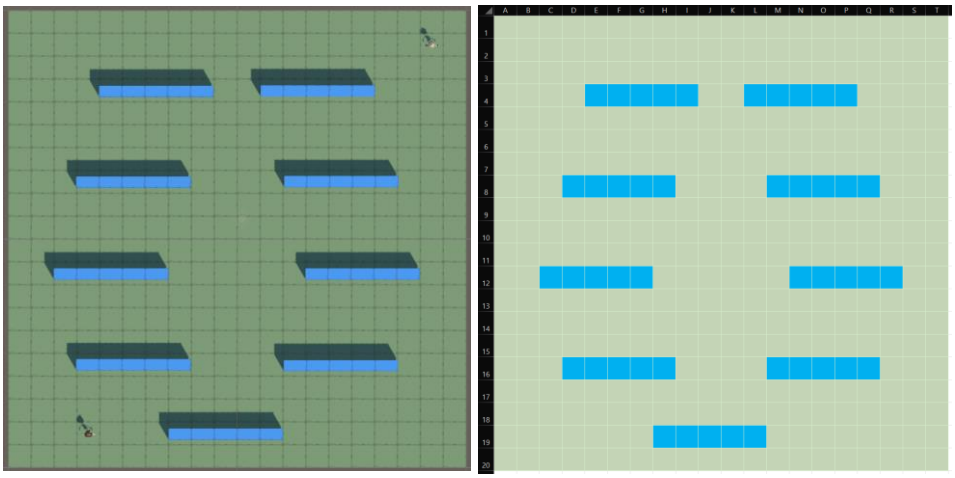


Ilustración 11 Mapa 3

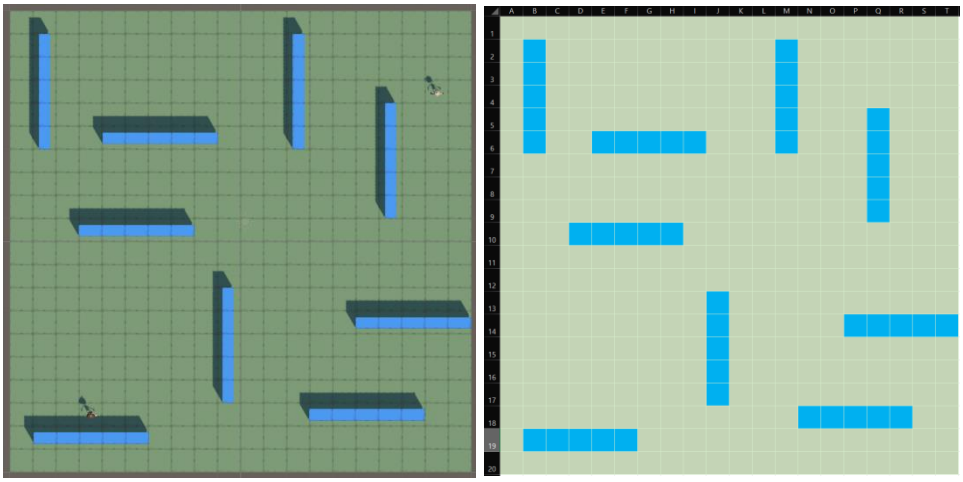


Ilustración 12 Mapa 4

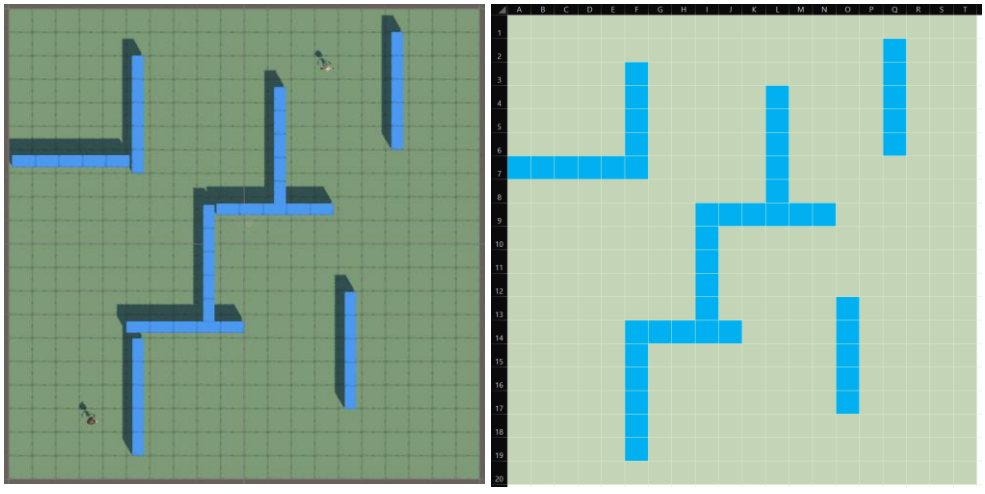


Ilustración 13 Mapa 5

7.2 PARAMETRIZACIÓN Y VALORES EMPLEADOS

En este apartado, se va a desarrollar la evolución del aprendizaje del agente y cómo se han ido actualizando los parámetros.

Para actualizar el valor de la tablaQ se ha empleado la fórmula de Q-learning, donde se actualiza el valor de la acción en el estado actual.

$$Q(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha \cdot \left(R + \gamma \cdot \max_{a'} Q(s',a')\right)$$

Donde:

Q(s, a) es el valor actual de la acción en el estado.

α es la tasa de aprendizaje.

Épsilon: para indicar si se quiere que el agente explore por libre o que tenga en cuenta lo ya aprendido.

γ es el factor de descuento.

maxa'Q(s',a') es el valor máximo de la acción en el próximo estado.

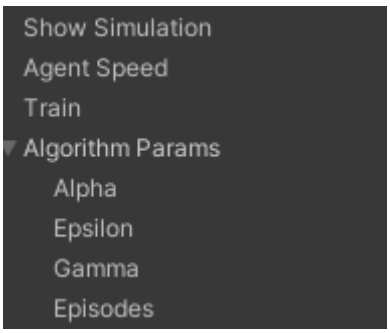


Ilustración 14 Parámetros de la escena

Un problema que se tuvo al comienzo es que teníamos marcada siempre el flag de "Show simulation", por lo que los episodios avanzaban de forma muy lenta. Pudimos comprobar lo que se comentó en el apartado anterior de los puntos débiles del agente en los mapas y los patrones que tendía a realizar.

Una vez que desactivamos ese flag, conseguimos entrenar a un ritmo aproximado de 20.000 episodios cada 40-45 minutos.

A continuación, y en el siguiente apartado se va a poder apreciar el desarrollo de aprendizaje y los cambios que se han ido realizando a la hora de entrenar.

PRIMERA RONDA DE ENTRENOS

La primera ronda de entrenamiento va a consistir en realizar **20.000 episodios en los 6 mapas (120.000 entrenos en total)**. En el apartado [7.3 Resultados] se podrá ver el rendimiento a la hora de testearlo.

Los parámetros establecidos para este entrenamiento son los siguientes:

Ilustración 15 Parámetros primera ronda

SEGUNDA RONDA DE ENTRENOS

Para esta segunda ronda de entrenamiento, se han ajustado los parámetros para que aprenda un poco menos, pero que no realice una exploración tan exagerada, sino que tenga un poco más en cuenta lo que ya ha aprendido.

En esta ocasión en vez de 20.000 episodios va a ser entrenado **10.000 episodios** en cada mapa, lo que hará un total de **60.000 episodios.** A continuación, se muestran los nuevos parámetros establecidos en la escena.

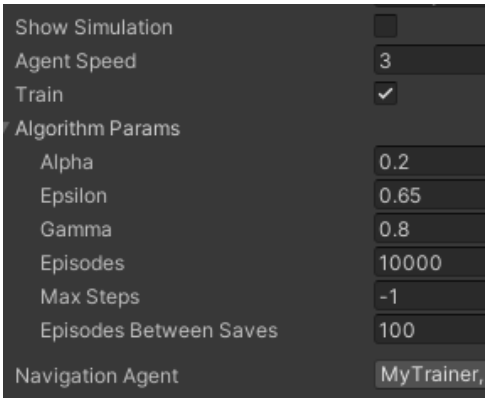


Ilustración 16 Parámetros segunda ronda

7.3 RESULTADOS

A continuación, se muestran los resultados de las distintas rondas de entrenamiento, mostrando el número de pasos que consigue dar en cada uno de los mapas.

PRIMER ENTRENO

Tras los primeros 20.000 episodios en el mapa por defecto, ha tenido un rendimiento muy bueno. En alguno de los mapas ha tenido un rendimiento malo, como por ejemplo en el 4 ya que hay caminos con una sola salida en los que si entra podría fallar fácilmente.

También se ha comprobado que el lugar desde el que empiezan la simulación tanto el player como el agente puede influir en que se den unos estados u otros y tenga un mejor o peor rendimiento (por ejemplo, en uno de los mapas se ha ido directo a una esquina y ha dado menos de 100 pasos, pero cambiando sus posiciones, ha seguido un camino de otros estados y ha superado los 1.000 pasos).

Se van a realizar los 20.000 entrenos en el resto de los mapas y a continuación se muestran los resultados en cada uno de ellos:

Cabe destacar que en cada uno de los diferentes mapas tanto el agente como el player han empezado en posiciones distintas.

Escenario base: Tendencia a realizar un bucle en la parte de la derecha y arriba. Ha superado los 5.000 pasos.

Mapa 1: Ha seguido un patrón muy similar al anterior. Supera los 5.000 pasos.

Mapa 2: Al comienzo ha avanzado por diferentes zonas del mapa sorteando correctamente los muros. Posteriormente, se ha puesto a dar vueltas en círculos alrededor del mapa hasta los 1.500 pasos. Y al final ha vuelto a entrar en el bucle de los dos mapas anteriores. Ha superado los 5.000 pasos.

Mapa 3: Vueltas alrededor del mapa manteniendo una buena distancia con el player hasta los 500 pasos. A partir de ahí ha vuelto a encontrar el bucle y ha seguido hasta los 5.000 pasos.

Mapa 4: Primer intento 96 pasos, ha intentado hacer el bucle y ha sido pillado en la casilla de arriba a la derecha. Se ha procedido ha cambiar las posiciones del agente y del player y se ha probado de nuevo.

Tras el cambio de posiciones ha vuelto a dar solo 64 pasos. Ha sido pillado en el espacio de 1x1 entre el muro y el vacío de la esquina superior izquierda. Se ha vuelto a ejecutar una tercera vez, en la que ha estado recorriendo diferentes partes del mapa hasta llegar a los 1.000 pasos y a partir de ahí, ha seguido un patrón en forma de L en la esquina superior derecha. Ha superado los 5.000 pasos.

Mapa 5: Al primer intento ha superado los 5.000 pasos. Al comienzo ha estado recorriendo todo el mapa, incluso en ocasiones entrando en la zona superior izquierda. Luego ha recorrido un patrón circular entre muros hasta los 500 pasos aproximadamente y a partir de ahí ha vuelto al bucle de la derecha.

RESUMEN DEL PRIMER ENTRENO

Resultado muy **satisfactorio**, dado que solo ha fallado en dos ocasiones y solo en uno de los mapas.

Se ha detectado una tendencia a repetir un patrón en la parte de la derecha subiendo y bajando todo el rato, pero se han hecho pruebas bloqueando esa zona y ha sido capaz de continuar por otras zonas e incluso encontrando otros patrones igual de sólidos y eficaces.

SEGUNDO ENTRENO

Escenario base: A conseguido dar los 5.000 pasos cruzando primero el laberinto y luego se ha mantenido dando vueltas por las zonas amplias.

Mapa 1: Tendencia desde el comienzo a dar vueltas por las zonas exteriores. Ha alcanzado los 5.000 pasos.

Mapa 2 y 3: 5.000 pasos dando vueltas por el exterior.

Mapa 4: Se rompe con el patrón de ir por el exterior gracias al posicionamiento de los muros. Aunque ha acabado realizando un patrón usando tanto zonas exteriores como más interiores del escenario. También ha alcanzado los 5.000 pasos.

Mapa 5: Ha conseguido dar los 5.000 incluso accediendo a la zona complicada de este mapa (esquina superior izquierda).

RESUMEN DEL SEGUNDO ENTRENO

De nuevo, se ha obtenido un resultado **satisfactorio**, esta vez superando los 6 mapas con 5.000 pasos a la primera.

El bucle de la zona derecha no lo ha realizado, por el contrario, sí ha tenido la tendencia de realizar uno por la zona exterior del mapa. Este comportamiento se debe a que la mayoría de los mapas no cuentan con muros en estas zonas.

De todas formas, se han hecho pequeñas pruebas moviendo muros para romper ese esquema y el agente consigue encontrar otro patrón con el que superar la prueba.