PEC 3 – Aprendizaje en vídeojuegos

Presentación

En esta PEC trabajaremos con los conceptos relacionados con aprendizaje que se han visto en los materiales, especialmente con el aprendizaje por refuerzo.

Competencias

En este enunciado se trabajan las siguientes competencias generales de máster:

- Capacidad para aplicar el pensamiento creativo y generar nuevas soluciones.
- Capacidad para el aprendizaje autónomo.
- Capacidad para dominar las herramientas aplicables al desarrollo de videojuegos, según las tendencias tecnológicas.
- Capacidad para el uso efectivo de los lenguajes de programación y las metodologías para el desarrollo de videojuegos.
- Capacidad para analizar y diseñar los elementos y los principios de funcionamiento de un videojuego.
- Capacidad para diseñar los componentes asociados con la creación de una experiencia de juego.
- Capacidad para representar elementos visuales y sus interacciones de manera eficiente.
- Entender la relación existente entre la Inteligencia Artificial y los otros componentes del videojuego.
- Saber cómo mover los elementos en un mundo virtual, tanto de forma individual como colectiva.

Objetivos

Al acabar esta PEC el alumno sabrá aplicar aprendizaje por refuerzo a juegos sencillos, así como a modificar parámetros para adecuarlos al tipo de juego o agente que se desee conseguir.

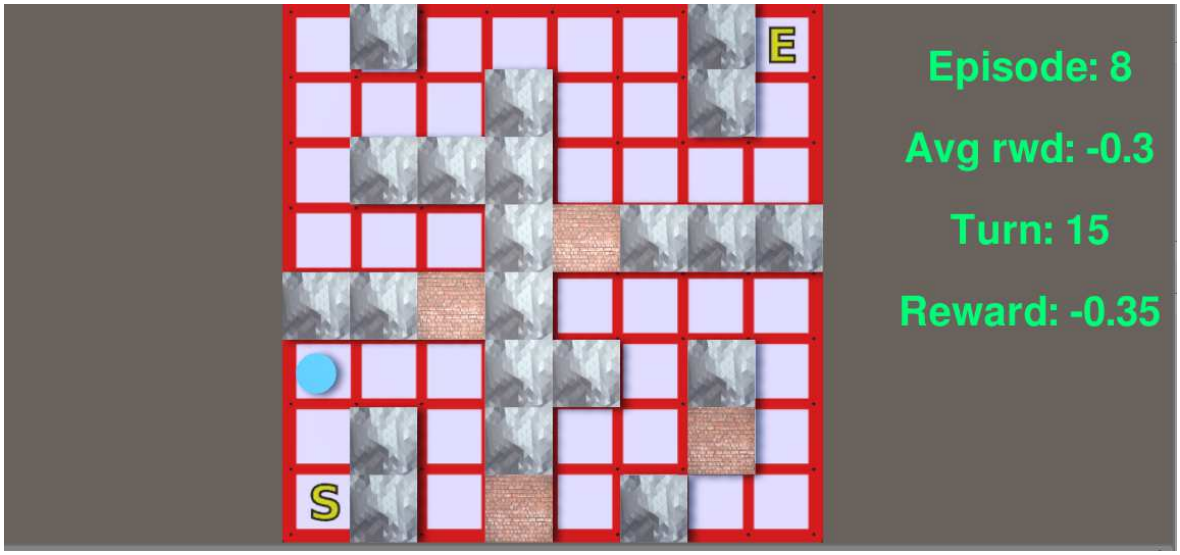
Descripción de la PEC a realizar

Actividades

Para desarrollar las actividades hay que leer los conceptos de aprendizaje por refuerzo (*reinforcement learning, RL*) de los materiales (pág. 62).

Instalación y ejecución del entorno de pruebas

El objetivo de esta actividad es la instalación del paquete de Unity titulado "Maze", que contiene una escena, los *scripts* y los objetos necesarios para el juego que se ve en la figura:



El juego consiste en conseguir que el jugador (cilindro azul), partiendo de la casilla S, llegue a la salida del laberinto, casilla E. El jugador puede:

- Moverse en las 4 direcciones cardinales, es decir no puede moverse en diagonal
- Caminar por las casillas libres
- Destruir las casillas de ladrillo, que entonces se convierten en casillas libres

Las casillas de piedra no se pueden destruir ni se puede caminar sobre ellas. Tampoco se puede salir del tablero, como es lógico.

El jugador es controlado por la IA, pero a diferencia de otros planteamientos de IA, en este caso no le damos ninguna indicación de cómo resolver el laberinto,

sino que simplemente vamos a dejar que la IA vaya aprendiendo con la práctica: jugando, y viendo cuál es el resultado de cada acción. Es lo que se conoce como **aprendizaje por refuerzo**.

Para empezar a trabajar:

- Cread un proyecto nuevo e importad el paquete *Maze*.
- Comprobad que el juego funciona correctamente.
- El objeto *GameManager* ofrece una serie de parámetros de configuración del juego; probad a reducir el valor "Turn Duration" para que el juego vaya más rápido y se aprecie antes el aprendizaje conseguido.

Aplicación de RL en este juego

En primer lugar, habréis observado que el juego ya incluye el código para hacer el RL, y de hecho si lo ejecutáis notaréis que el agente va aprendiendo a recorrer el laberinto y alcanzar la salida. El objetivo de esta PEC no es que programéis código de aprendizaje sino que entendáis el código dado, cómo se ha aplicado al juego, cómo se puede configurar para conseguir otros comportamientos, y finalmente que apliquéis algún cambio al proyecto.

En el aprendizaje por refuerzo se suele dividir el aprendizaje en episodios de entrenamiento (*episodes*), lo que en el juego corresponde con las "partidas", que simplemente son secuencias de movimientos hasta que el jugador alcanza la salida o bien ha utilizado 100 turnos (movimientos del jugador).

El agente solo puede realizar cuatro acciones: moverse hacia la izquierda, hacia la derecha, hacia arriba o hacia abajo. No puede quedarse quieto. Las posibles acciones y sus recompensas son:

- Moverse a una casilla libre: -0.05
- Moverse a una casilla de ladrillo (y por tanto romperla): -0.2
- Intentar moverse a una casilla de piedra (no se ejecuta el movimiento, simplemente choca): -0.7
- Intentar salir de los límites del tablero (tampoco se ejecuta): -0.8
- Alcanzar la salida: +10

El código del juego consta de dos *scripts*: uno que gestiona el juego y coordina los diferentes elementos, y otro que gestiona el aprendizaje que lleva a cabo el agente. El código tiene abundantes comentarios para que podáis seguir lo que hace.

Recordad que en RL el agente va aprendiendo la recompensa que obtendrá si, en el estado S del juego, ejecuta la acción A. Poco a poco irá viendo cuál es la acción más fructífera en cada estado.

El estado de este juego se define por:

1. La posición del jugador. Hay 8x8=64 casillas, por tanto 64 posiciones diferentes. Se podría optimizar ya que no es posible colocarse sobre algunas casillas (piedra)
2. El estado intacto/destruido de los bloques de ladrillo. Hay 4 bloques de ladrillo y cada uno tiene dos estados posibles, por tanto hay $2^4 = 16$ combinaciones posibles

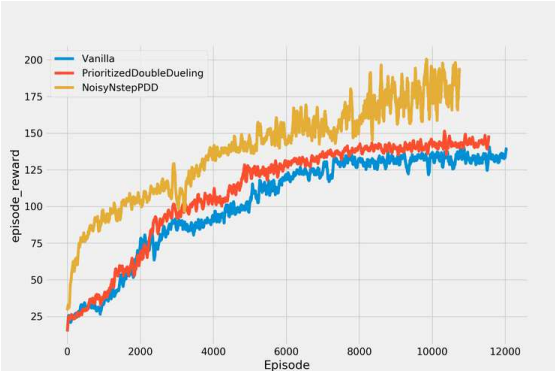
Los estados posibles del juego son todas las combinaciones de ambos factores, por tanto hay 64x16=1024 estados posibles del juego (algunos no se darán nunca pero no hay problema, solo que usan memoria).

Así, la tabla de aprendizaje (tabla Q) es una tabla de dimensiones estados x acciones, es decir de 1024 x 4 elementos, en la que se va almacenando la recompensa obtenida al tomar la acción *j* (columna) cuando el juego se encontraba en el estado *i* (fila).

De esa manera, poco a poco el agente va eligiendo las acciones mejor recompensadas dado el estado actual.

Actividad 1: monitorización del aprendizaje

La principal herramienta para entender lo que ocurre durante el proceso de entrenamiento son las curvas de aprendizaje, en las que se muestra la recompensa obtenida en cada episodio, y así se puede ver si el agente va mejorando, si se ha estancado, etc.



Realiza las siguientes tareas:

a) Modifica Maze para poder visualizar la curva de aprendizaje del agente. Elige el número de episodios. Recuerda que puedes acelerar los turnos para poder ejecutar más episodios. Sugerencia: guarda los pares #episodio, recompensa en un archivo y a continuación utiliza una hoja de cálculo para leerlos y generar la gráfica.

b) Analiza la curva obtenida. Puedes ejecutar varias veces el programa desde cero y comparar los resultados.

c) ¿Hay subidas y bajadas en la recompensa a lo largo de los episodios? Si es así, ¿a qué crees que es debido?

Actividad 2: límite de turnos

En la versión de Maze proporcionada hay un máximo de turnos por episodio, de forma que si el agente no alcanza la salida en ese número de turnos se acaba el episodio con la recompensa que tuviera hasta ese momento.

Elimina ese límite y analiza el efecto en el aprendizaje. Ten en cuenta las curvas de aprendizaje y también el tiempo de ejecución requerido, ya que sin límite de turnos cada episodio puede durar mucho más.

Indica cuál de las dos estrategias consideras más adecuada.

Continúa la práctica usando el número de turnos máximo.

Actividad 3: ladrillos

Como primer cambio en el juego y para profundizar en el cálculo del espacio de estados, realiza las siguientes tareas:

a) Añade otro bloque de tipo ladrillo en la posición que desees. Ajusta el cálculo del número de estados posibles y el tamaño de la tabla Q.
b) Comprueba que el agente resuelve esta versión del laberinto.
c) Compara la curva de aprendizaje actual con la obtenida en la actividad 1.
d) ¿Cómo cambiaría el tamaño de la tabla Q a medida que se añadieran nuevos bloques de ladrillo? ¿Y si se añadieran bloques de roca?

Actividad 4: enemigo

Se introduce un nuevo elemento: un enemigo que se mueve aleatoriamente (un paso cada turno), de manera que si el jugador y el enemigo chocan (están en la misma casilla), el jugador muere, termina el episodio y recibe una recompensa negativa (a determinar por vosotros).

El enemigo no puede destruir bloques de ladrillo y solo se puede mover por las casillas libres.

Para resolver esta actividad debes:

- Programar el enemigo
- Explicar cómo cambia el cálculo de estados y modificar el juego para que lo tenga en cuenta
- Entrenar el agente en este nuevo entorno y observar el aprendizaje. Analiza las diferencias respecto al entorno anterior

Incluye un vídeo del resultado.

Actividad 5: visualización

En esta actividad visualizaremos la tabla Q de forma intuitiva. Para conseguirlo simplificaremos un poco el juego. Realiza las siguientes tareas:

a) Elimina los bloques de ladrillo del juego (deja casillas vacías en su lugar)
b) Modifica la codificación de estados para ajustarla a las nuevas reglas
c) Sobre cada casilla dibuja una flecha o equivalente que indique la dirección preferida por el agente si se encontrara en esa casilla, es decir la dirección (acción) con valor más alto en la tabla Q en la fila correspondiente a esa

casilla. Estas flechas irán cambiando a medida que el agente aprenda, así que se tendrían que actualizar (cada turno o cada episodio)

Incluye un paquete de Unity con tu proyecto y un vídeo del resultado.

Recursos

Manual de teoría de la asignatura, en especial el apartado 4.1.

Paquete de Unity con el juego Maze.

Criterios de valoración

Las actividades de la PEC tendrán la siguiente valoración asociada:

Actividad 1:	2 puntos
Actividad 2:	1 punto
Actividad 3:	2 puntos
Actividad 4:	2 puntos
Actividad 5:	3 puntos

Formato y fecha de entrega

La PEC se ha de entregar antes del próximo 10 de enero (a las 24h).

La solución que hay que entregar consiste en un informe en formato PDF usando este enunciado como plantilla y, para cada actividad en la que se pida: un vídeo demostrativo donde se vea el uso de lo que pide el enunciado y una carpeta con la implementación. El informe ha de contener la descripción y el código relacionado a lo que pide el enunciado. Todos estos ficheros se han de comprimir en un archivo ZIP.

Adjuntad el fichero al apartado de **Entrega y registro de EC (REC)**. El nombre del fichero debe ser ApellidosNombre_IA_PEC3.zip.

Para dudas y aclaraciones sobre el enunciado, dirigíos al consultor responsable de vuestra aula.

Nota: Propiedad intelectual

A menudo es inevitable, al producir una obra multimedia, hacer uso de recursos creados por terceras personas. Es por tanto comprensible hacerlo en el marco de una práctica de los estudios del Máster de Ingeniería Informática, siempre que esto se documente claramente y no suponga plagio en la práctica.

Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se presentará junto con ella un documento en el que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y el su estatus legal: si la obra está protegida por copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL ...). El estudiante deberá asegurarse de que la licencia que sea no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente deberá asumir que la obra está protegida por copyright.

Deberán, además, adjuntar los archivos originales cuando las obras utilizadas sean digitales, y su código fuente si corresponde.