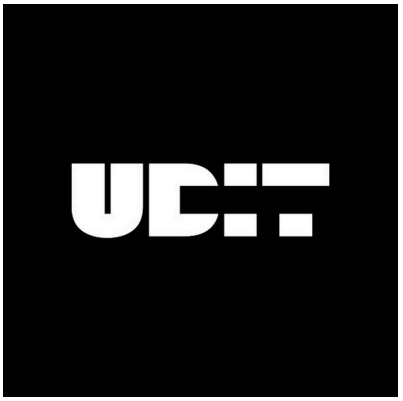
DISEÑO Y DESARROLLO DE VIDEOJUEGOS

Q-Learning y SARSA

Hecho por Iñaki Cezón Ortega y
Ignacio Tapia Marfil

Promoción: 2024/2025

Fecha de entrega: 15/05/2025



Índice

Resumen Ejecutivo

El entregable es un programa que implementa un agente que intentará salir de un laberinto por el mejor camino posible. Se han implementado los métodos de Q-Learning y SARSA como algoritmos de aprendizaje por refuerzo.

Diseño

El programa es un ejecutable de consola del sistema que imprime un mapa por pantalla importado de un archivo de texto. En el mapa aparecerán el personaje, muros, enemigos, tesoros y las salidas como posibles elementos.

La estructura del código se divide en “Controller”, “Enemy”, “Entity”, “FileReader”, “main”, “Map”, “MovementComponent”, “Player”, “Q-Learning”, “RandomController”, “Render”, “Sarsa” y “Treasure”.

Explicación de clases

A continuación desglosamos las partes implementadas en el código:

- Controller: Clase padre que contiene funciones propias de los dos algoritmos Q-Learning y Sarsa. Las funciones comunes que define es el “LoadQTable” y el “SaveQTable” y el “DoAction”.
- Q-Learning: Clase hija de “Controller” donde se implementan las funciones mencionadas en Controller.
- Sarsa: Clase hija de “Controller” donde se implementan las funciones mencionadas en Controller.
- RandomController: Clase hija de “Controller” que implementa la función “DoAction” para que ejecute acciones aleatorias.
- Entity: Clase padre que define las entidades posibles del mapa (Player, Enemy, Treasure). Define la función “Update” y gestiona la posición x, y además de si la entidad está activa o inactiva y el símbolo asociado a cada entidad.
- Enemy: Clase hija de “Entity” que inicializa el símbolo (&) de la entidad cuando se crea.
- Player: Clase hija de “Entity” que inicializa el símbolo (@) de la entidad cuando se crea. También implementa una función “Update” que gestiona el encuentro del jugador con tesoros o enemigos y una función de reset. Si encuentra un enemigo hay un 50% de probabilidad de matarlo y que sume puntos o de morir y que ese individuo muera y pase al siguiente.
- Treasure: Clase hija de “Entity” que inicializa el símbolo ($) de la entidad cuando se crea.

- FileReader: Clase que lee el mapa deseado y lo almacena en un vector de dos dimensiones. También almacena en un vector una estructura de entidades que contiene el tipo de entidad, la fila y la columna en la que se encuentran.
- Render: Clase que contiene una función “RenderScene” que lee las casillas del mapa y las posiciones de las entidades y escribe en un array de dos dimensiones los elementos del mapa que se devuelve como referencia para uso futuro.
- Map: Clase que contiene una estructura sobre el tipo de tile (Floor, Wall, Goal, Start) y que gestiona un vector de dos dimensiones para simular los tiles del laberinto.
- MovementController: Clase que regula si una entidad puede realizar un movimiento o no y en caso de poder realizar el movimiento modifica la posición de la entidad.
- main: Clase base del programa, contiene una función de “CreateEntity” que llama a “EntityCharger” de “FileReader” y añade todas las entidades a un vector. Tiene una función “Reset” que reinicia los valores de los algoritmos y entidades. Tiene una función “SelectMap” que busca todos los mapas en una carpeta (resources) que estén nombrados siguiendo la estructura (Map_01) y luego pide que mapa se desea. También tiene una función “Menú” para sacar por pantalla el menú y finalmente una función “Game” que gestiona el proceso de aprendizaje del algoritmo. Se carga el mapa y empiezan las generaciones. Una vez terminan las generaciones, el programa termina y se almacenan las tablas Q en un fichero en la carpeta (resources).

Análisis de Resultados

Para el análisis de resultados vamos a estudiar 2 mapas cargados desde un fichero de texto. El número de generaciones máximas está definido en 299 y el número de pasos posibles es de 5000. Posteriormente en el apartado de conclusiones se analizaran los resultados obtenidos del experimento.

Sarsa Map 1	Learning Phase				
Learning Rate	0.7				
Discount Rate	0.6				
Epsilon:	0.6				
Goal Reward	1000				
Movement Reward	-1				

Collision Reward	-10				
Kill Reward	100				
Treasure Reward	200				
Die Reward	-1000				
GoBack Reward	-5				
Generations	299	41	71	120	291
Maze Completed	280	23	52	101	272
Enemies Killed	16	15	16	16	16
Treasures Caught	25	17	20	22	25

Q-Learning Map 1	Learning Phase				
Learning Rate	0.7				
Discount Rate	0.6				
Epsilon:	0.6				
Goal Reward	1000				
Movement Reward	-1				
Collision Reward	-10				
Kill Reward	100				
Treasure Reward	200				

Die Reward	-1000				
GoBack Reward	-5				
Generations	299	39	106	165	256
Maze Completed	288	31	95	154	245
Enemies Killed	20	18	20	20	20
Treasures Caught	30	22	29	30	30

Sarsa Map 2	Learning Phase 1				
Learning Rate	0.7				
Discount Rate	0.6				
Epsilon:	0.6				
Goal Reward	1000				
Movement Reward	-1				
Collision Reward	-10				
Kill Reward	100				
Treasure Reward	200				
Die Reward	-1000				
GoBack Reward	-5				
Generations	299	45	100	156	230
Maze Completed	42	4	9	40	42

Reaches Goal		Yes	No	No	No
Steps		4766	1062	3301	4656
Enemies Killed	47	1	0	0	0
Treasures Caught	162	0	0	0	0

Sarsa Map 2	Learning Phase 2				
Learning Rate	0.7				
Discount Rate	0.6				
Epsilon:	0.6				
Goal Reward	1000				
Movement Reward	-1				
Collision Reward	-10				
Kill Reward	100				
Treasure Reward	200				
Die Reward	-1000				
GoBack Reward	-5				
Generations	299	45	100	156	230
Maze Completed	65	11	24	65	65
Reaches Goals		No	No	No	No

Steps		3606	1943	2831	2336
Enemies Killed	37	0	0	0	0
Treasures Caught	112	0	0	0	0

Q-Learning Map 2	Learning Phase 1				
Learning Rate	0.7				
Discount Rate	0.6				
Epsilon:	0.6				
Goal Reward	1000				
Movement Reward	-1				
Collision Reward	-10				
Kill Reward	100				
Treasure Reward	200				
Die Reward	-1000				
GoBack Reward	-5				
Generations	299	45	100	200	260
Maze Completed	141	12	29	48	103
Reaches Goal		No	No	Yes	Yes
Steps		224	105	120	106
Enemies Killed	83	0	0	0	0

Treasures Caught	271	0	0	1	1

Q-Learning Map 2	Learning Phase 2				
Learning Rate	0.7				
Discount Rate	0.6				
Epsilon:	0.6				
Goal Reward	1000				
Movement Reward	-1				
Collision Reward	-10				
Kill Reward	100				
Treasure Reward	200				
Die Reward	-1000				
GoBack Reward	-5				
Generations	299	45	100	151	248
Maze Completed	211	16	39	65	161
Reaches Goals		No	Yes	Yes	Yes
Steps	104	1101	706	114	96
Enemies Killed	42	0	0	0	0
Treasures Caught	72	0	0	0	0

Conclusiones

A la hora del desarrollo de los algoritmos, para optimizar y afinar un poco más los algoritmos, hemos considerado implementar penalizaciones si chocase con un muro y si deshace un movimiento (retrocede una casilla), además de una penalización continua de movimiento.

Respecto a ambos algoritmos en el primer mapa, tanto Sarsa como Q-Learning, hemos podido comprobar que ambos encuentran la solución más corta y en pocas generaciones, pero no hemos logrado que tenga en cuenta los valores cuando encuentra tesoros o enemigos de tal forma que por ejemplo en el primer mapa, el camino final del aprendizaje se realiza en 12 pasos sin coger tesoros ni encontrar enemigos. Sin embargo hay un camino claro con mejor resultado que sería un camino de 14 pasos en el que coge un tesoro y solo aumenta dos pasos el recorrido. Éste camino está bloqueado por un enemigo más adelante lo cual hace que el valor baje, pero seguiría siendo mejor resultado coger el tesoro y volver hacia atrás.

Comparando los algoritmos en el segundo mapa, se puede apreciar claramente que el algoritmo Sarsa no encuentra un camino hasta la meta y se queda “atascado” después del primer entrenamiento y también después del segundo entrenamiento sin una mejora notable. En el caso de Q-Learning, en el primer entrenamiento logra encontrar un camino hasta la salida que además recoge un tesoro. Al ejecutar el algoritmo en este momento sin aprendizaje, completa el algoritmo a la perfección en 106 pasos recogiendo 1 tesoro y sin enfrentar enemigos. Tras esto se realiza un segundo entrenamiento a partir del anterior y se logra que el algoritmo realice el laberinto en 95 pasos sin recoger tesoros ni enfrentar enemigos. Esto demuestra que al volver a entrenarlo una segunda vez, ha preferido realizar el camino en menos pasos a sacrificar 11 pasos por coger un tesoro.

Con estos resultados podemos ver que ambos realizan las funciones básicas y casi aproximadas al objetivo final. No se ha logrado tener en cuenta el valor positivo de los tesoros aunque sí aparenta que se ha tenido en cuenta el valor negativo que aportan los enemigos.