Inteligencia Artificial para Videojuegos

Práctica 2 : Control de unidades en juego RTS

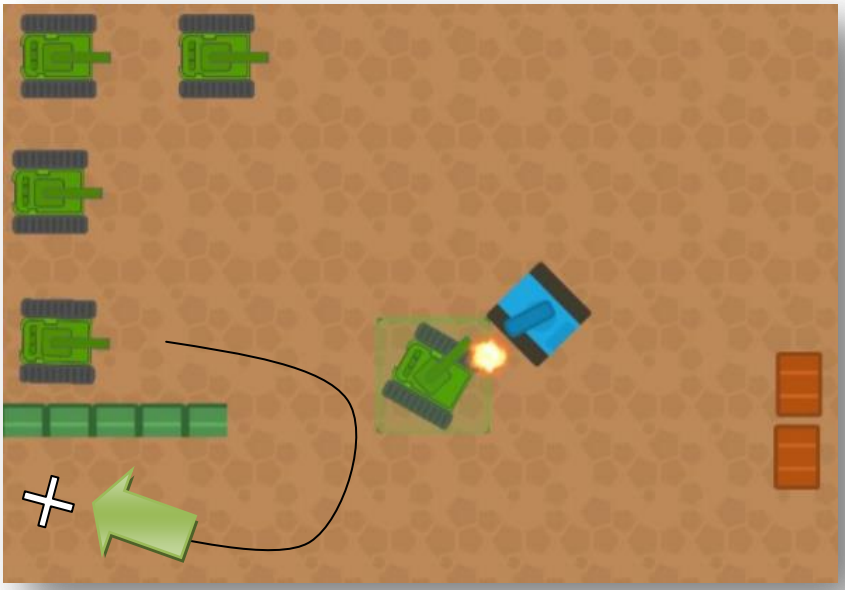
Grupo 07

Alejandro Ortega Álvarez

Borja Cano Álvarez

Problema a resolver

- La propuesta es representar en Unity un posible “trozo” de mapa de un juego RTS, donde se indican a ciertas unidades un punto de dicho mapa donde se las desea llevar para que se desplacen hacia esa posición por el camino más corto posible.



- Para ello se piden los siguientes requisitos:
    1. Un tablero de 10x10, con tres tipos de casillas: normales, embarradas y bloqueantes.
    2. Tres unidades posibles a las que indicar el camino
    3. Al clickear en una casilla, si no hay unidades seleccionadas, se cambia el tipo de casilla en este orden: *normal → embarrada → bloqueante → normal*
    4. Al clickear en una casilla, si hay una unidad seleccionada, aparecerá una cruz del color de la unidad en dicha casilla y la unidad comenzará el recorrido para llegar a la cruz por el camino más corto posible (y se deselecciona la unidad).
    5. Las casillas normales gastan 1 de movimiento, las embarradas 2, y las bloqueantes y los otros tanques impiden el movimiento por las casillas en las que se hallen.
- Adicionales:
    1. Botón para limpiar el terreno para que aparezca todo con casillas normales(como al inicio de la partida).
    2. Botón para general un mapa aleatorio.
    3. Botón para reiniciar la partida general nuevas posiciones aleatorias para los tanques.
    4. Texto de ayuda que indica el tanque actual seleccionado.

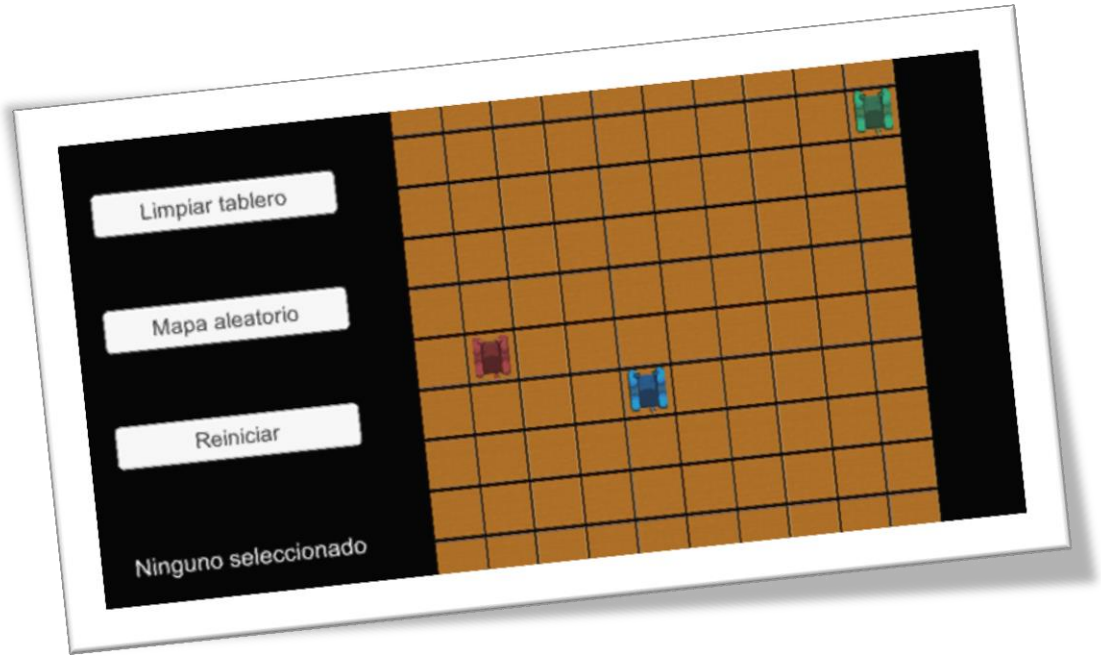
Cómo lo hemos hecho

GameObjects

- Los GameObjects que componen el tablero son los de “casilla”. Sólo está compuesto de un SpriteRenderer para la imagen, un BoxCollider que cuadre con su zona de selección y su Script Index para saber su información y qué ocurre cuando se clickea.

- Los GameObjects de los tanques se pasan a sí mismos por referencia al ser clickeados al GameManager, que será el que junte la información de éstos con la información de las casillas para poder realizar el recorrido.

- Los demás GameObjects del juego son el GameManager (explicado el Script más abajo), el propio Tablero que junta las casillas, el texto y los botones y un fondo con un collider para que se deseleccione unidad automáticamente al cliquear fuera del tablero.



(juego en estado inicial)

Scripts

- Script Selección: para cada tanque, detecta si se cliquea en él con OnMouseDown() y le pasa la información al GameManager.

- Script Index : para cada casilla, tiene un número que determina su identidad y que además coincide con su valor de movimiento (1 = normal, 2 = barro, 0 = roca). Si se cliquea en una casilla, dependiendo de la selección, cambiará su index o pasará información al GameManager de su posición para indicar el destino del tanque.

- Script Fondo: para deseleccionar si se hace click fuera del tablero.

- Script Algoritmo: explicado más detalladamente abajo. Comprueba si existe una ruta desde la posición del tanque a la posición de su destino para poder pasarla.

- Script Move: llamado por el GM, se le pasa una ruta al tanque para que éste se vaya moviendo por el tablero atravesando dicha ruta.

- Script GameManager: encargado de sincronizar y conectar todas las conexiones del juego.Contiene los siguientes métodos:

    - El Start() crea un tablero con casillas normales y llama a createGame().
    - CreateGame() genera una posición aleatoria para los tanques asegurándose de que no sean las mismas y limpia el tablero (es el método que llama el botón reiniciar).
    - Limpiar() pasa el index de todas las casillas a 1 (normales) y deselecciona unidad.
    - MapaAleatorio() recorre todo el tablero y mediante un número random va dándole nuevos index a cada casilla (la casilla normal tiene el triple de probabilidad de salir para que sea la más abundante). Las casillas donde se sitúan en ese momento los tanques siempre serán normales.
    - Seleccionar() y SeleccionarVacio() para cambiar el objeto que haya seleccionado.
    - PonerCruz() se llama desde el OnMouseDown() de una casilla cuando se haya pinchado y haya un tanque seleccionado, y no sea una casilla de roca. Crea la cruz (destino) correspondiente en la posición de la casilla y llama al algoritmo para que vea si hay una ruta posible entre la posición del tanque y la del destino y en caso de que sí, la traza.

Algoritmo

En cuanto al algoritmo utilizado se trata de una implementación del algoritmo A*.

El algoritmo consiste en buscar los nodos más prometedores entre todos los disponibles, consiguiendo una respuesta óptima.

Para ello el algoritmo costa de los siguientes componentes.

- Node: son los nodos a explorar. Almacenan la siguiente información:
    - Una referencia al su padre
    - Su posición en el tablero
    - Su índice en el array booleano de marcados
    - El valor propio del tipo de terreno.
- Lista open: la lista de los nodos candidatos.
- Lista close: la lista de los nodos guardados.
- Array marcados: es un array de booleanos, cada posición representa un nodo, si ha sido visitado anteriormente o no.

¿Cómo funciona el algoritmo?

Creamos dos nodos: uno para el comienzo del camino y otro para el final.

Añadimos el comienzo al lista de open y comenzamos el bucle:

1. Si la lista de open está vacía, reportamos error.

2. Cogemos el nodo más prometedor de la lista (el que tiene menor coste potencial)

3. Si el nodo es igual a la meta, **hemos encontrado el mejor camino.** Añadimos el nodo al closed y se devuelve ella lista, la cual será el recorrido a seguir.

4. Si no es la meta, creamos todos los nodos vecinos al nodo que estamos mirando calculando para cada uno sus costes.

5. Por cada nuevo nodo creado se realizan los siguientes pasos:

5.1. Si el nodo es un nodo nuevo (false en el array de marcados) Se añade el nuevo nodo en la lista de open y se marca.

5.2. Si el índice del nodo ya ha sido marcado en el array de booleanos se comprueba si tiene coste menor que el nodo que ya teníamos antes, de ser así se sustituye.

6. Se añade el nodo que habíamos utilizado al principio de la iteración a la lista de cerrados y se vuelve al punto 1.

Definiciones:

Coste real: es el coste del camino por el cual se ha llegado a un nodo en concreto. Es importante sumar el coste del nodo concreto(puesto que tenemos nodos con distintos costes (casillas embarradas)).

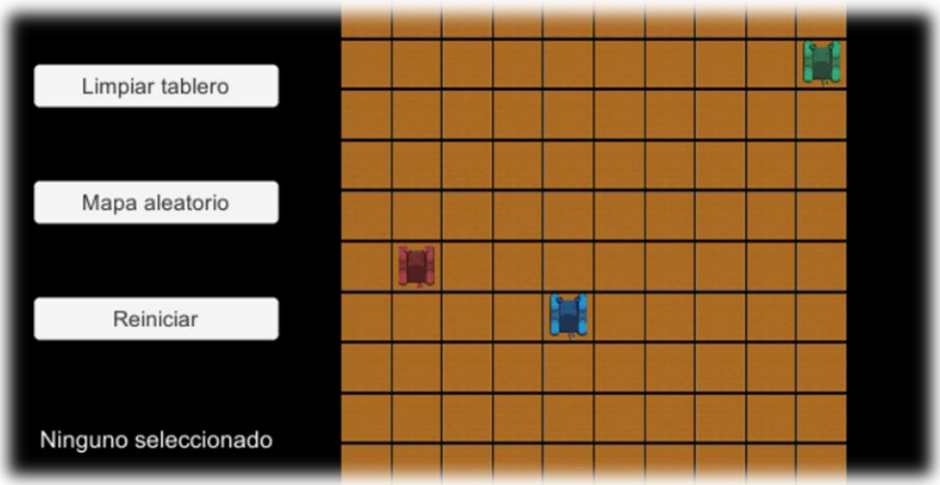
Coste estimado: es el coste real de ese nodo más el coste que se estima (por heurística) que tiene el nodo concreto hasta llegar a la meta.

Heurística: Es una función que intenta estimar el coste de una solución. Para esta ocasión hemos utilizado la distancia Manhattan.

Juego final

-Repasamos el funcionamiento final del simulador.

Ésta es la vista nada más iniciar el juego:



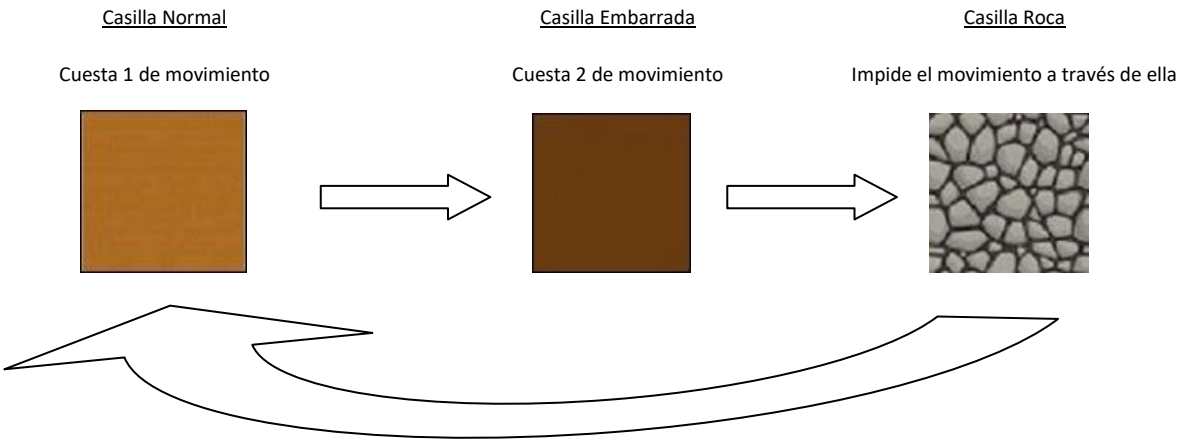
Cómo jugar

Al cliquear tanques

-Se selecciona el tanque que corresponda

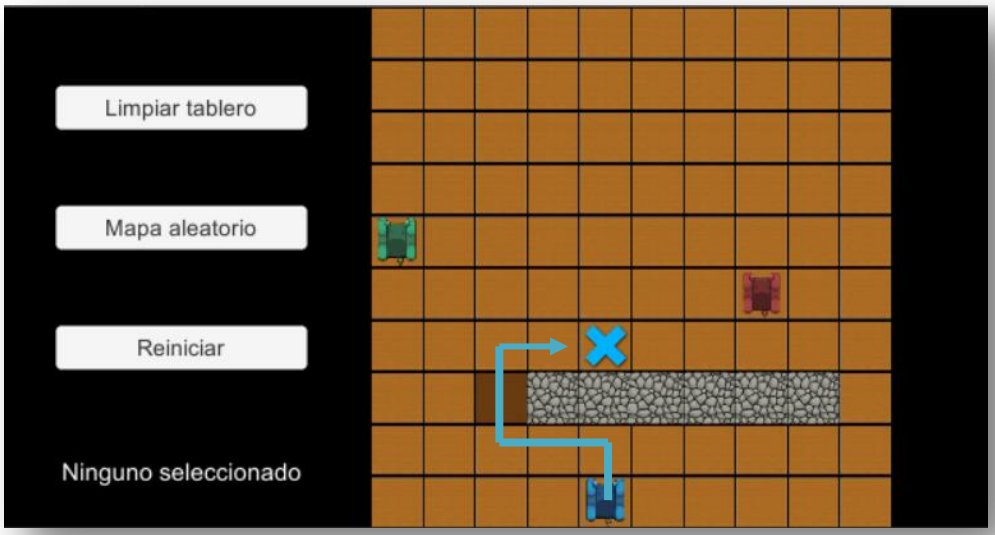
Al cliquear casillas

-Si NO hay tanque seleccionado, se cambia el tipo de casilla:



-Si hay un tanque seleccionado se crea una ruta desde la posición del tanque hacia la casilla que se haya seleccionado por el camino más corto. También se deselecciona la unidad.

No se trazará una ruta si se cliquea en otro tanque, una casilla de roca o fuera del mapa.



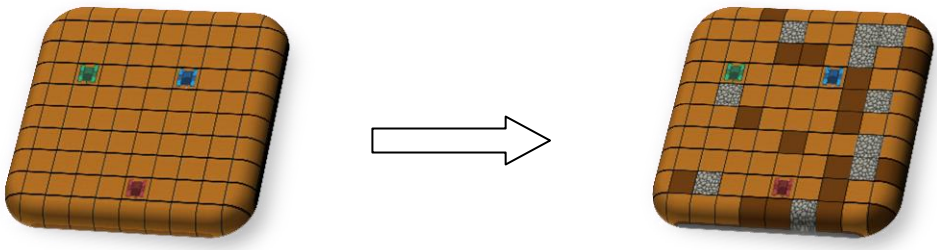
Al cliquear fuera del mapa o

en una casilla de roca mientras hay tanque seleccionado

-Se deselecciona el tanque.

Botones

- Limpiar tablero : convierte todas las casillas a normales, manteniendo la selección actual.
- Mapa aleatorio : genera un mapa aleatorio siempre distinto del anterior.



- Reiniciar: limpia el mapa, da una nueva posición aleatoria a los tanques y deselecciona unidad.

Problemas

A parte del bug anterior, surgieron los siguientes problemas en el desarrollo de la práctica:

- Al implementar el algoritmo, primero tuvimos problemas con sintaxis de C#, al pasarlo desde JavaScript.
- Una vez solucionado, surgió otro problema ya que, aunque sacaba el recorrido correctamente esquivando obstáculos, no tenía en cuenta las casillas embarradas ya que en cuanto encontraba una solución válida acababa el algoritmo, lo solucionamos buscando más de un resultado para los caminos y, comparándolos, nos quedamos con el de mejor coste.
- Otro problema importante que nos surgió fue al implementar el recorrido del tanque. Al utilizar valores discretos como posiciones en el mundo y añadirle una velocidad al tanque para ponerle en movimiento, esto originaba que nunca cuadraran los valores del transform position con la posición discreta de la casilla a la que tenía que llegar. Para solucionarlo, se nos ocurrió hacer un casting implícito a int del transform position, de esta forma redondeaba y cogía el valor entero de la posición del tanque. Esto funciona parcialmente, ya que por ejemplo hace redondeos extraños al llegar a la posición 0 y a veces al girar en X redondea a la siguiente casilla en Y, por lo que se sale del camino que tiene que seguir.
- Otro problema fue que no nos iba correctamente el OnCollisionEnter() de los tanques, por lo que el tema de las “colisiones” con las casillas directamente lo detectamos mirando la casilla del tablero. Con los tanques se ha intentado también (están las líneas comentadas) pero al no poder hacerlo con collisions y hacerlo por casillas, directamente se paraba el tanque al pasar por el lado de otro y eso era peor aún.

Referencias y apoyos del desarrollo de la práctica:

- *Apuntes del Campus Virtual*
- *Problema de los Sensores, MARP Ejercicio 12 evaluación*
- *Google Imágenes*
- *http://buildnewgames.com/astar/ (algoritmo)*

Repositorio de Github: `https://github.com/alex97ortega/IA.git`