Análisis Laberinto
Daniel Pestana, Cristian Sanchez, Sebastian Cataño

a. ¿Cómo se está representando cada nodo del terreno a recorrer en la aplicación interactiva?

- Cada Nodo se representa como una celda única, un tile o teja que puede tener un vecino arriba, abajo o a los lados.

b. ¿Cuál es la estructura de datos que se está usando para almacenar cada nodo del terreno?

- A partir del código de Node.cs, se guardan todos los nodos en un Diccionario con un Vector2int y cada nodo.

c. ¿Cómo se definen el punto de inicio y el punto final del recorrido?

- El _startingPoint es el nodo inicial privado declarado al principio y se define en la función de BFS al ponerlo en la cola y empezar desde ese la exploración de nodos vecinos, por otro lado se puede encontrar al crear el path o camino a partir de ese mismo punto inicial.
- El punto final o _endingPoint es otro nodo privado definido al principio como variable y se compara con el searchingpoint para saber si hemos llegado al final, además para crear un nuevo camino desde el nodo previo

d. ¿Qué ventaja tiene usar la estructura de datos anterior para este problema?

- La búsqueda en Anchura o BFS es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo, comenzando en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo), para luego explorar todos los vecinos de este nodo. Y es muy necesario para este problema ya que se usa cuando resulta crítico elegir el mejor camino posible en cada momento del recorrido. Además el árbol es muy profundo y las soluciones son raras, y el BFS es usualmente usado para encontrar el camino más corto entre dos nodos.

e. Escribe los pasos que componen el algoritmo BFS en la solución del problema.

- Lo primero que se hace es poner el origen o raíz en la cola (queue), mientras aún haya ítems por procesar en la cola, se saca el ítem que estamos procesando de la cola (ya lo hemos visitado), y cogemos todos los vecinos de esta. Por cada vecino que aún no se ha visitado del nodo con el que se está trabajando, se meten en la cola y son marcados como visitados.
- En la cola se van añadiendo todos los vecinos procesados y el algoritmo acaba cuando ya no quedan nodos por procesar.

- Específicamente se inicia cargando todos los nodos en LoadAllBlocks, luego se usa el BFS para sortearlos desde el nodo inicial al final, y finalmente se emplea CreatePath creando el camino más rápido luego de visitar los nodos. Lo que devuelve un camino

f. ¿Para qué se usa una cola en el algoritmo BFS de la aplicación interactiva?

- Se usa para guardar todos lo nodos al visitarlos y poder cruzar por ellos, teniendo a la raíz de primera
-

g. ¿Cómo se determina si un nodo ya fue explorado?

- Es necesario guardarlo en la cola y marcarlo como _isExplored = true

h. ¿Cómo se determina qué nodos serán explorados?

- Como hay ítems por procesar al primero que visitamos se le cogen todos sus vecinos, los cuales son determinados para ser explorados y metidos en la cola como visitados.

i. ¿Qué estructura de datos se está utilizando para almacenar los nodos que componen la ruta calculada desde el punto de inicio al punto final?

- Para guardar el camino atravesado final se emplea una simple lista que ocupa nodos uno después de otro.

j. ¿Qué ventaja tiene usar la estructura de datos de la pregunta anterior?

- Lo bueno es que, a diferencia de los arrays, no dependen de un tamaño fijo y determinado al principio, por lo que se le pueden agregar más elementos a medida que crece.

k. Al crear el path ¿Por qué es necesario usar el método Reverse()?

- Por que como se guardan de la misma forma que el queue, está al revés, la raíz o inicial está al final y el endingpoint al principio, así que hay que revertirlo

l. Explica cómo funciona la aplicación interactiva luego de dar click en play.

- El programa luego de correr el código sigue el camino trazado como opción más rápida y con cada paso pausa para realizar el BFS y poder avanzar en el path.

m. ¿Qué es una Corrutina y cómo y para qué se está usando en la aplicación interactiva?

- Los métodos normales van de principio a fin, debe acabar un proceso para iniciar el siguiente y no pueden parar. Mientras que las corrutinas sirven para detenerse en cualquier momento específico de la ejecución, para pasado un tiempo reanudar desde el punto exacto desde donde lo dejó, y que se siga ejecutando de manera normal.
- Las corrutinas deben ser de tipo IEnumerator y retornar el valor homólogo.
- Se usan en el método Movement para moverse una casilla, esperar a que se realice el proceso de BFS y proseguir con la transformación de posición.