Sokoban

Tarea 1

En esta primera tarea el objetivo es representar el nivel e identificar la función de cada caracter. Para ello se ha desarrollado una clase Map, donde se encapsulan los elementos de la cadena introducida por teclado.
Los tres módulos que se han desarrollado en esta primera tarea son `sokoban.py`, `map.py` y `parser.py`. Añadiendo la opcion 'T1' en el comando de ejecución de sokoban, muestra por pantalla la posición de cada elemento del juego.

* sokoban.py
  Este es el módulo principal, en él se recogen los parámetros introducidos por teclado y se realizan las llamadas correspondientes a las opciones elegidas.

* parser.py
  Es el encargado de parsear las opciones y parametros para encapsularlos en un objeto parser. También es el encargado de mostrar un mensaje de error en caso de no introducir bien las opciones.

* map.py
  El módulo que contiene la clase Map, dónde se desarrollan las funcionalidades del proyecto.

    - Map

        - `__init__`: Es el método constructor, requiere del parámetro "level" para inicializarse, al crear un objeto Map se inicializan los atributos `level`, `ID`, `rows`, `columns`, `wallList`,`player`, `boxList`, `targetList` y `map`.

        - `define_grid`: Cuenta el número de filas y columnas que tiene el nivel y retorna un grid inicializado con todas las posiciones a ' '. Cada caracter encontrado hasta el salto de línea (\n) es una columna, por lo tanto el número de columnas (NC) se incrementa en 1; por cada salto de línea se incrementa en 1 el número de filas (NR), también en cada salto se compara el número de columnas contado en esa iteración con el almacenado, en caso de ser mayor, se almacena en la variable maxNC. De esta manera se define un grid para no dejar ningún carácter fuera de límites.

        - `make_level`: En primera instancia llama al método `define_grid`, posteriormente itera sobre la cadena 'level', diferenciando los caracteres, asignandoles una posición (i,j) en el grid y almacenando sus posiciones en los atributos del objeto. Por último almacena el resultado en el atributo map y genera un identificador del nivel.

        - `show_map_elements`: Imprime por pantalla el identificador del nivel y las posiciones de los diferentes elementos que lo componen.

Tarea 2

El objetivo de esta tarea es identificar la función sucesor del jugador y de las cajas. Las nuevas funcionalidades se añaden en `map.py` y se descubren añadiendo la opción 'T2S' y 'T2T' en el argumento task del juego.

T2S

La acción T2S obtiene la función sucesor del nivel, para ello se han desarrollado `successors`, `get_index`, `get_moves` y `show_successors`.

* `successors`, es el método que retorna la lista final con los movimientos sucesores del nivel, diferenciando entre mayúsculas y minúsculas, cajas y jugador, respectivamente la dirección del movimiento U (Up, Arriba), R (Right, Derecha), D (Down, Abajo), L (Left, Izquierda). Cada elemento de la lista que retorna es una tupla: <direccion,nuevo_estado,coste>.

* `get_index`, por parámetros se le pasa una posición (i,j) y comprueba en la lista de las cajas el igual al parámetro, devolviendo el índice.

* `get_moves`, con la posición del jugador este método obtiene todos sus movimientos posibles. Si encuentra una caja adyacente, comprueba también si puede ser movida. Este método retorna una lista con elementos <<u|U|r|R|d|D|l|L>, posicion_sucesor(i,j) , posicion_actual(i,j)>

* `show_successors`, muestra por pantalla los sucesores del nivel.

T2T

La meta de esta tarea es mostrar el resultado de la función OBJETIVO para el nivel. Se ha desarrollado el método `objective`.

* `objective`, comprueba que no exista ningún objetivo en el nivel, en caso de que exista alguna caja ('$') se considerará que el objetivo no está cumplido. Muestra por pantalla el resultado.