

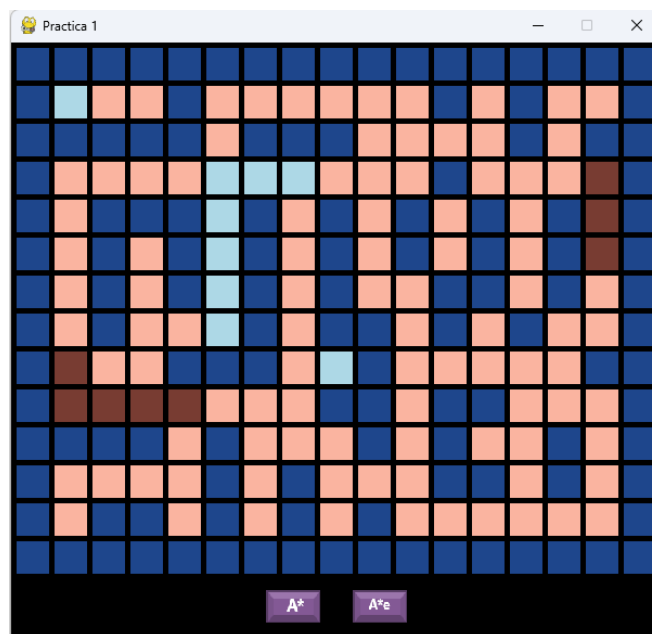
Práctica 1. Búsqueda heurística

Objetivos:

- Comprender el funcionamiento de la búsqueda heurística y en concreto del algoritmo A y del A^*_ϵ .
- Implementar los algoritmos A^* y A^*_ϵ y saber cómo seleccionar una heurística apropiada al problema.
- Realizar un análisis cuantitativo respecto al número de nodos explorados con este algoritmo.

Sesión 1: Introducción y entorno de trabajo

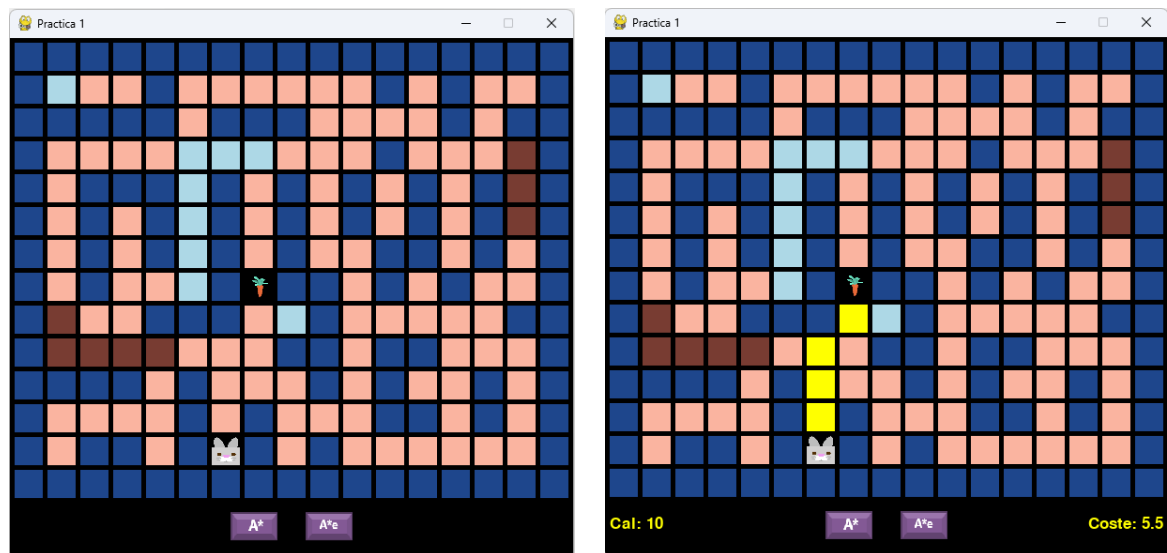
En esta primera práctica de la asignatura se debe desarrollar el algoritmo A^* para calcular el camino menos costoso entre dos casillas de un mapa. En nuestro mapa concreto tenemos una rejilla de celdas cuadradas de distintos colores con distinto significado en el que podemos indicar un origen y un destino.



Significado de los colores:

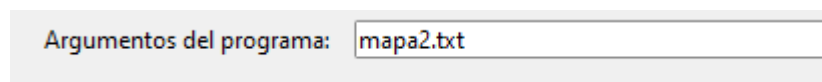
- Azul oscuro: celda no transitable
- Marrón claro: hierba
- Marrón oscuro: roca
- Azul claro: agua

Cuando se pulsa el botón izquierdo del ratón sobre una celda, ésta se convierte en el origen y aparece en esa celda un conejo, al pulsar con el botón derecho en una celda se convierte en el destino y aparece una zanahoria. Se debe implementar el algoritmo A* para calcular el camino de menor coste entre el conejo y la zanahoria. Cuando se ha calculado el camino las celdas correspondientes aparecerán coloreadas en amarillo y en la parte inferior de la ventana se mostrará el coste del camino y las calorías consumidas.



Funcionamiento del juego

Al ejecutar el juego se cargará por defecto mapa.txt. Si se quiere elegir otro mapa distinto, hay que pasarlo como argumento. Para ello, desde Thonny acceder a menú Visualización- Parámetros de ejecución del programa, en el cuadro de texto que aparece escribir el nombre del mapa que se quiere usar.



El mundo está formado por celdas cuadradas. Hay 8 movimientos posibles correspondientes a las 8 direcciones. Las celdas que forman el tablero pueden ser:

- Celdas por donde el conejo puede cruzar: hierba, agua y roca
- Celdas donde el conejo no pueden acceder: son las celdas de color azul oscuro.

Desarrollo

El entorno se ha desarrollado en Python usando el paquete `pygame`. Pygame facilita la creación de videojuegos en dos dimensiones y permite programar la parte multimedia (gráficos, sonido y manejo de eventos) de forma sencilla.

El entorno proporcionado está compuesto por 3 ficheros:

- Main: es el fichero que hay que ejecutar para lanzar el entorno. Tiene el bucle principal de manejo del juego, así como el desarrollo del interfaz gráfico
- Mapa: contiene el código que permite construir un objeto mapa que contiene la información del mapa a partir del fichero de texto que especifica el contenido del mapa
- Casilla: permite definir un objeto que especifica una posición a partir de una fila y una columna

Como herramienta de desarrollo se utilizará Thonny. Thonny es un IDE con los elementos básicos para crear aplicaciones en el lenguaje Python, es posible depurar y observar las variables en ejecución de forma sencilla.

Ejecución del proyecto

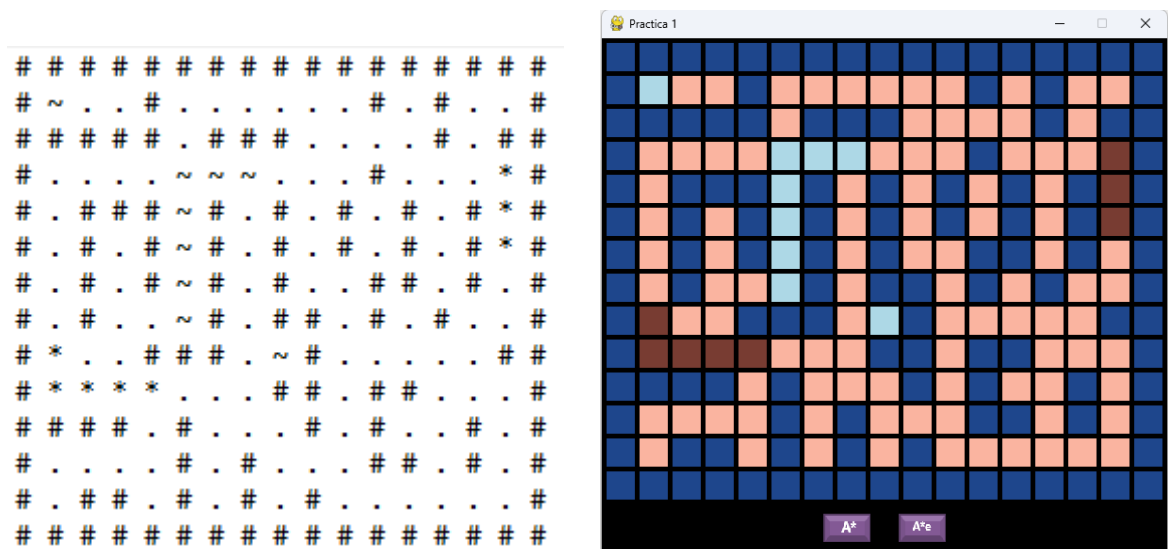
Ejecutar el fichero main.py

Definición del mundo

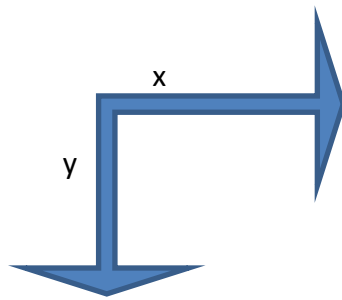
La especificación del mundo se realiza en un fichero de texto. En este fichero de texto se utilizarán unos caracteres definidos:

- '#': celda de bloque
- ' ': celda de hierba
- '~': celda de agua
- '*': celda de roca

A continuación, se muestra un ejemplo de mundo. En la parte de la izquierda se puede ver el contenido del fichero y en la de la derecha al aspecto gráfico del mismo.

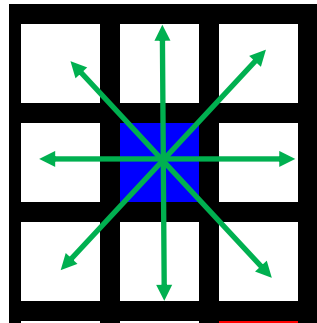


El sistema de coordenadas se representa en la siguiente figura:



Movimientos posibles desde una celda

El mundo en que se desarrolla el juego está formado por celdas cuadradas. Hay como máximo 8 posibles movimientos ya que a las celdas pintadas de rojo no se puede ir.



Tarea a realizar en esta primera sesión:

- Prueba y analiza el entorno. Averigua cómo acceder a una celda del mapa y cómo se pueden generar los distintos movimientos válidos que se pueden hacer desde una posición.

Sesiones 2 y 3: Diseño del algoritmo de búsqueda A* (Con $h=0$)

El algoritmo A*

Este algoritmo es uno de los más utilizados para encontrar un camino o ruta entre dos puntos, con la característica que, si se cumplen unas condiciones, el camino encontrado será el camino de menor coste entre los dos puntos y además, si existe, siempre encontrará ese camino (algoritmo completo).

Un problema clásico dentro del mundo de los videojuegos es dotar a los personajes controlados por la máquina de la capacidad de moverse por el entorno de una manera cada vez más realista, bien para perseguir al jugador o para competir por un objetivo común.

En nuestro caso concreto vamos a utilizar el algoritmo A* para que el conejo encuentre el camino de coste mínimo para llegar hasta la celda con la zanahoria.

A* es un algoritmo que utiliza una función de **coste** para calcular cuánto cuesta llegar de una celda a otra. Cada movimiento conlleva un coste que depende del tipo de movimiento. Vamos a considerar los siguientes valores:

- horizontal: 1
- vertical: 1
- diagonal: 1.5

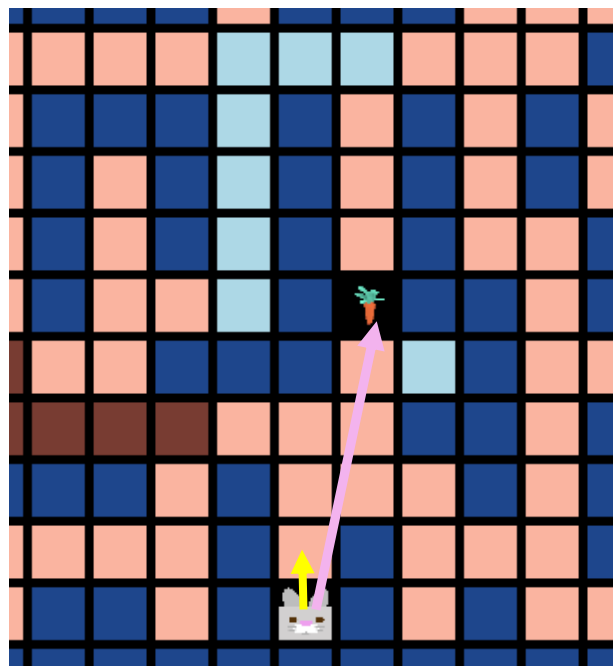
El **gasto en calorías** que supone llegar al destino se va a simular teniendo en cuenta que el tipo de terreno conlleva un gasto distinto de calorías:

- Hierba: 2
- Agua: 4
- Roca: 6

La **función de evaluación**, $f(n)=g(n)+h(n)$, que emplea A* está compuesta por dos partes:

- g: coste de ir desde la posición inicial hasta la posición actual
- h: estimación optimista de llegar desde la posición actual hasta el objetivo

La figura siguiente ilustra el significado de estas dos partes. Si queremos calcular el valor de $f(n)$ para la casilla que implica moverse una casilla hacia arriba, la flecha amarilla representa el valor de $g(n)$, que en este caso valdría 1 puesto que es un movimiento vertical, la línea rosa se corresponde con $h(n)$ que habría que estimar empleando alguna función de distancia entre ambas celdas.



La función heurística es una estimación optimista de lo que le costará al conejo llegar hasta el objetivo.

$$h(x,y) \sim \leq \text{distancia al objetivo}$$

De forma breve, lo que hace el algoritmo es, partiendo de la posición inicial del conejo, elegir el movimiento a la celda que le supone un camino más corto para llegar al destino. Para ello tiene que estudiar todos los posibles movimientos que puede hacer y calcular el coste de cada uno de esos caminos. Para cada posible movimiento, analizará a su vez los movimientos que se podrían hacer y así sucesivamente hasta considerar un movimiento que le lleve al objetivo (celda con la zanahoria). Elegirá el camino que suponga un coste menor.

A continuación, se muestra el pseudocódigo del algoritmo A*.

Pseudocódigo

Alg A*

listaInterior = vacío

listaFrontera = inicio

mientras listaFrontera no esté vacía

 n = obtener nodo de listaFrontera con menor $f(n) = g(n) + h(n)$

 si n es meta devolver

 reconstruir camino desde la meta al inicio siguiendo los punteros

 Salir

 sino

 listaFrontera.del(n)

 listaInterior.add(n)

 para cada hijo m de n que no esté en lista interior

$g'(m) = n.g + c(n, m)$ //g del nodo a explorar m

 si m no está en listaFrontera

 almacenar la f, g y h del nodo en (m.f, m.g, m.h)

 m.padre = n

 listaFrontera.add(m)

 sino si $g'(m)$ es mejor que m.g //Verificamos si el nuevo camino es mejor

 m.padre = n

 recalcular f y g del nodo m

 fsi

 fpara

 fmientras

 Error, no se encuentra solución

Falg

Se empleará como primera heurística, $h = 0$. Es decir, que la función heurística evaluara con 0 cada celda. Con la cual obtendríamos resultados de una búsqueda con coste uniforme.

Detalles de implementación

En el código suministrado hay una matriz llamada camino que se inicializa con '.'. Esta matriz se utiliza por el entorno para mostrar el camino que calcula el A*.

De igual forma, hay una variable llamada coste inicializada a -1 que muestra el coste del camino y una variable llamada cal inicializada a 0 que indica el gasto en calorías de dicho camino.

El A* debe:

- actualizar la matriz camino con algún símbolo distinto de '.' en las celdas que forman el camino para que el entorno la pueda dibujar.
- asignar valor a la variable coste que indica el coste del camino de origen a destino y que el entorno muestra en la parte inferior de la ventana.
- asignar valor a la variable cal con el gasto en calorías del camino determinado por el A*

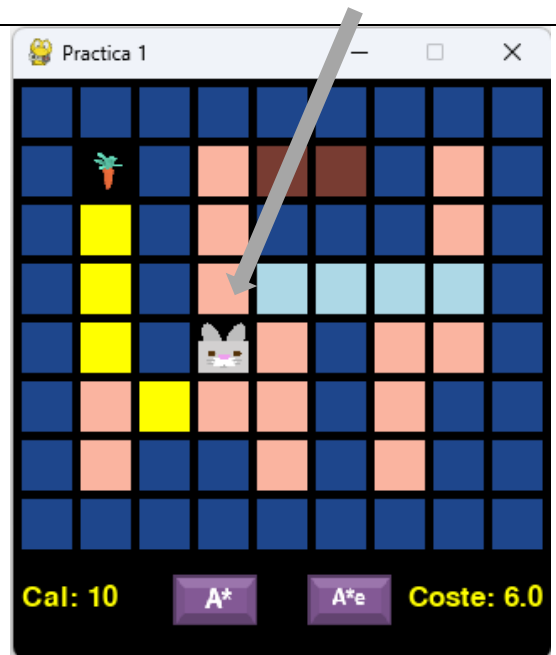
La matriz camino se inicializa con '.' en todas las celdas, esta tarea la hace la función `inic (mapa)`. El entorno muestra en color amarillo las celdas de la matriz camino que contienen un carácter distinto de '.'. La siguiente figura muestra un mapa con el camino detectado. En la parte de la izquierda se puede observar el orden en el que han sido generados los estados en el algoritmo A*, el valor 0 se corresponde con la posición del conejo y el valor 25 con la posición destino. Se puede ver que todas las celdas coloreadas en amarillo tienen en la parte de la izquierda un valor distinto de -1, sin embargo, hay celdas que tienen valor distinto de -1 pero no corresponden al camino ya que el algoritmo explora posibilidades que luego rechaza, por ejemplo, el valor 3 en la matriz de estados explorados se corresponde a una celda que ha sido explorada pero no forma parte del camino (se marca con una flecha gris en la imagen). Mirando la figura se puede ver que se han generado 26 estados (incluyendo el estado inicial) pero sólo 6 forman parte del camino.

Orden de generación de los estados

```
-1 -1 -1 -1 -1 -1 -1 -1 -1
-1 25 -1 11 15 18 -1 24 -1
-1 21 -1 7 -1 -1 -1 23 -1
-1 16 -1 3 6 9 14 19 -1
-1 13 -1 0 2 -1 17 22 -1
-1 8 5 1 4 -1 20 -1 -1
-1 12 -1 -1 10 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1
```

Matriz camino

```
. . . . . . . . .
. X . . . . . . .
. X . . . . . . .
. X . . . . . . .
. X . . . . . . .
. . X . . . . . .
. . . . . . . . .
. . . . . . . . .
```



Tarea a realizar en esta sesión:

- Implementa el algoritmo A* empleando como heurística $h=0$

Sesión 4 y 5: Incorporando heurísticas. Hito intermedio de entrega

Como se ha comentado anteriormente, la función heurística es una estimación optimista de cómo de lejos está el objetivo al que se desea llegar. Esta función lo que hace es asociar a cada celda un valor que evalúa lo prometedor que es esa celda para llegar al objetivo, normalmente es una estimación de lo próximo que se encuentra el objetivo. El algoritmo A* irá explorando los nodos en función de lo prometedores que sean, es decir, irá seleccionando el nodo que menor coste ($g+h$) tenga.

Según la función heurística que implementemos el algoritmo A* puede obtener resultados diferentes. Si la función heurística **es admisible**, el algoritmo A* siempre obtendrá el camino óptimo, si este existe. Aunque dependiendo de lo 'buena' que sea esa función heurística, el algoritmo será más o menos eficiente y explorará más o menos nodos. Si la función heurística **no es admisible**, puede ser que el algoritmo A* encuentre un camino, pero que este no sea el óptimo.

Una heurística es admisible si nunca sobrestima el coste de alcanzar el objetivo. Es decir, si el valor que da para cada celda es siempre menor o igual que el coste mínimo para alcanzar el objetivo.

$$h(n) \leq h^*(n)$$

Siendo $h^*(n)$ el coste mínimo real de la celda n al objetivo.

Por todo esto, la función heurística es una de las piezas clave dentro del algoritmo A*. Según el tipo de problema al que nos enfrentemos, la heurística óptima para ese problema será diferente. Si no se conoce cuál es la heurística óptima para el problema al cual nos enfrentamos, probaremos diferentes heurísticas y las analizaremos viendo si el camino que obtiene es el óptimo (el de menor coste) y el número de nodos explorados.

Posibles heurísticas:

- **Distancia Manhattan.** Esta distancia es la suma de las diferencias absolutas de las coordenadas de la celda origen y de la celda destino. Es decir, la distancia entre la celda 1 y la celda 2, sería:

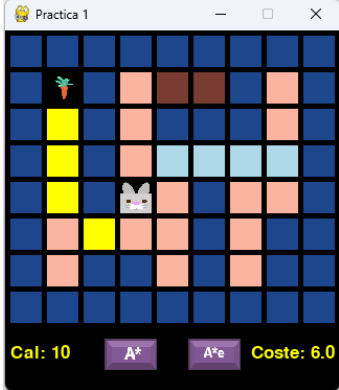
$$|x_2 - x_1| + |y_2 - y_1|$$

- **Distancia Euclídea.** Define la línea recta entre dos puntos. Se deduce a partir del Teorema de Pitágoras y se calcularía la distancia entre la celda 1 y la celda 2 como:

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- **Otras funciones de distancia**

La utilización de una buena heurística es un factor importante. La siguiente figura representa la cantidad de nodos explorados utilizando $h=0$ (en la parte de la izquierda) y utilizando otra heurística en la parte de la derecha. Aunque $h=0$ es una heurística admisible, hay una gran diferencia en el número de nodos explorados.

h=0		h= una función de distancia
<pre> -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 25 -1 11 15 18 -1 24 -1 -1 21 -1 7 -1 -1 -1 23 -1 -1 16 -1 3 6 9 14 19 -1 -1 13 -1 0 2 -1 17 22 -1 -1 8 5 1 4 -1 20 -1 -1 -1 12 -1 -1 10 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 </pre>		<pre> -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 11 -1 3 -1 -1 -1 -1 -1 -1 10 -1 2 -1 -1 -1 -1 -1 -1 9 -1 1 4 -1 -1 -1 -1 -1 8 -1 0 5 -1 -1 -1 -1 -1 -1 7 6 -1 </pre>

Al probar las heurísticas, veremos que algunas de ellas serán admisibles (siempre obtendrán el camino óptimo) y puede ser que otras no sean admisibles (puede ser que el camino que encuentren no sea óptimo).

Tarea a realizar en esta sesión:

- Implementa varias funciones de distancia para emplearlas como heurística del A* y analiza si son admisibles o no
- En la entrega del hito deja la heurística que te parezca más adecuada

Sesión 6: A^*_ϵ : relajación de la restricción de optimalidad

Esta sesión se dedicará a implementar el algoritmo de estimación del coste de búsqueda. Este algoritmo es un A* que incorpora una lista adicional llamada lista focal. Esta lista es una sublista de Lista_Frontera que contiene únicamente aquellos nodos cuya $f(n)$ no excede del mejor valor de cualquier $f(n)$ dentro de la Lista_Frontera por un factor $(1+\epsilon)$:

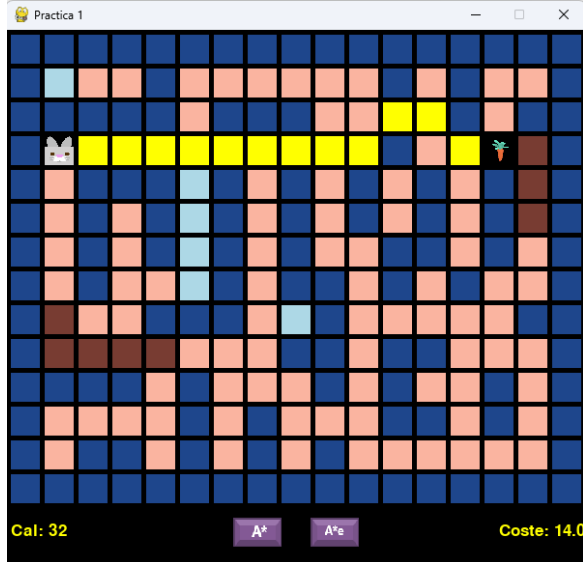
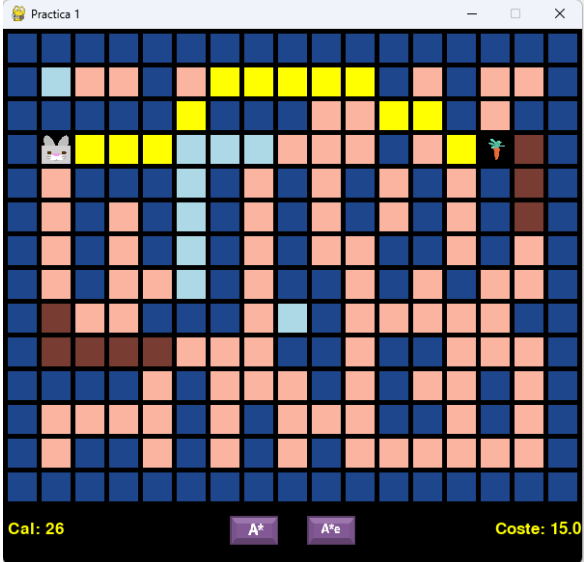
$$L_f = \{n: f(n) \leq (1+\epsilon) \min(f(m))\}$$

A^*_ϵ opera de forma idéntica al algoritmo A* salvo que selecciona aquel nodo de Lista_focal con menor valor de $H_f(n)$, una segunda heurística.

Para este problema vamos a **utilizar como heurística focal el concepto de gasto en calorías**.

Tarea a realizar en esta sesión:

- Implementa el algoritmo A^*_ϵ

Solución con A^*	Solución con A^*_ϵ
	

Sesión 7: Documentación y pruebas. Hito final

Esta última sesión está dedicada a terminar las pruebas y finalizar la documentación que se deberá haber ido elaborando de manera continua durante todo el desarrollo de la práctica.

La documentación es la parte más importante de la práctica (60%). **Como mínimo** debe contener:

- Explicación y **traza de un problema pequeño** donde se observe el funcionamiento del algoritmo A^* .
- Análisis comparativo de las **distintas heurísticas** implementadas analizando el número de nodos.
- Pruebas de A^*_ϵ
- con distintos valores de ϵ
- Análisis comparativo de los algoritmos A^* y A^*_ϵ
- Varios mundos de prueba con diseños que muestren distintas casuísticas del problema: problema sin solución, varios caminos, etc.

Se valora el empleo de gráficas para ilustrar el análisis.

Entrega de la práctica

La fecha límite de entrega de la práctica es el **3 de noviembre de 2024** a las 23:55h. La entrega se realizará a través de Moodle.

Formato de entrega del proyecto para el hito 2 (entrega final)

La entrega debe consistir en un **fichero comprimido zip** con tres carpetas:

- /Fuente: todos los ficheros .py y elementos gráficos que constituyen la práctica
- /Mundos: mundos de prueba utilizados
- /Doc: fichero pdf con la documentación

!!!AVISO IMPORTANTE!!!

No cumplir cualquiera de las normas de formato/entrega puede suponer un suspenso en la práctica.

Recordad que **las prácticas son INDIVIDUALES** y NO se pueden hacer en parejas o grupos.

Cualquier código copiado supondrá un suspenso de la práctica para todas las personas implicadas en la copia y, como indica el Reglamento para la Evaluación de Aprendizajes de la Universidad de Alicante (BOUA 9/12/2015) y el documento de Actuación ante copia en pruebas de evaluación de la EPS, se informará a la dirección de la Escuela Politécnica Superior para la toma de medidas oportunas.

Plan de entrega por hitos

Durante el periodo de ejecución de la práctica se realizarán dos hitos de entrega. Es obligatorio cumplir las fechas de las entregas correspondientes:

Hito	Entrega	Fecha tope
1 (intermedio)	Todos los ficheros .py y elementos gráficos que constituyen la práctica. No es necesario entregar documentación	20 de octubre
2 (final)	Práctica completa con varias heurísticas implementadas siguiendo la estructura del formato de entrega	3 de noviembre

- La no entrega del hito 1 en la fecha prevista supone una penalización del 20%.
- Para la entrega final se dejará el programa empleando la heurística más adecuada. Las otras heurísticas analizadas aparecerán en el código, aunque no se utilicen.

La nota de la práctica sufrirá una penalización de dos puntos si no se cumple rigurosamente con los requisitos de la entrega (tanto en la estructura de los ficheros entregados como en la salida que debe generar la práctica)