MEMORIA

*PRÁCTICA ALGORITMO A**

EMILIO ÁLVAREZ PIÑEIRO

DANIEL SERRANO TORRES

Descripción

En primera instancia se ha decidido elegir *Python* como el lenguaje de programación con el cual se codificará el algoritmo A*. Ya que dicho lenguaje es realmente eficiente en términos de estructuras de datos, la versión utilizada para el desarrollo es 2.7.6 (dado que esta es la que hay instalada en los laboratorios de la facultad).

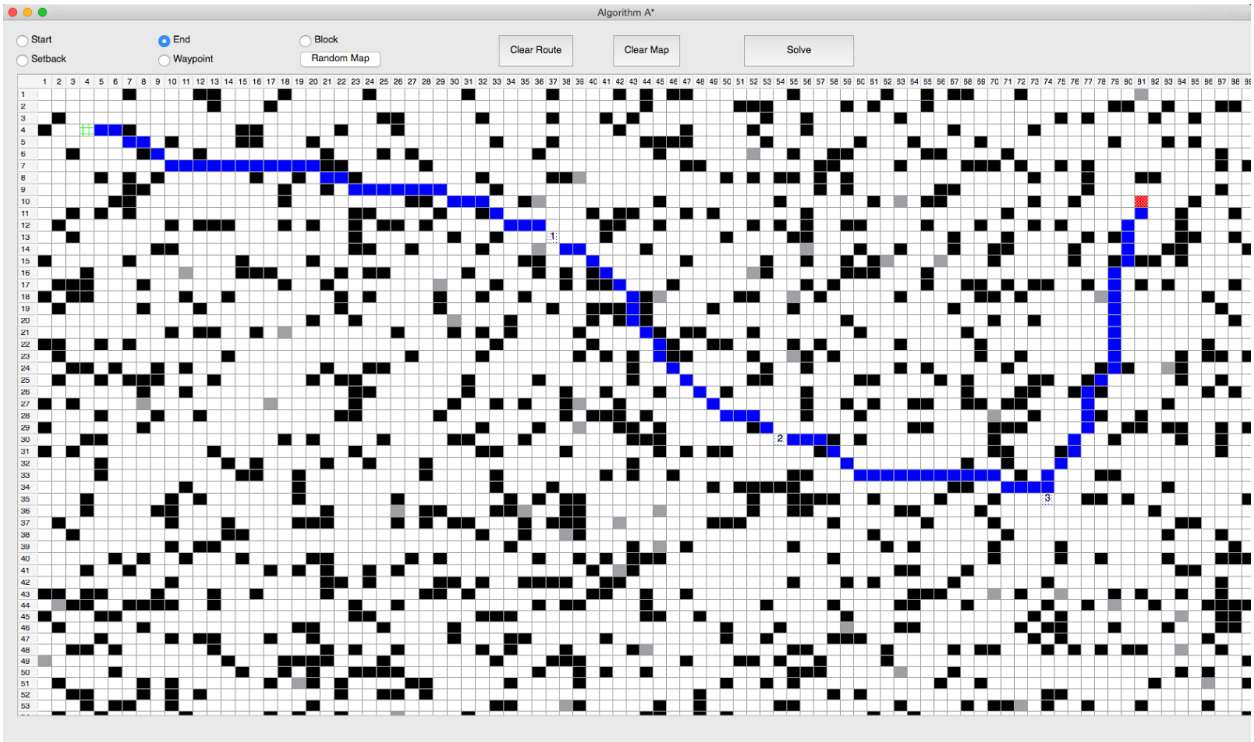
Diseño

El diseño del código se ha realizado pensando en que la fuente de información puede ser externa y diversa, por tanto el algoritmo A* implementado opera con una estructura de datos *mapa* la cual debe responder a una *API* (un conjunto de métodos) esperada por el programa, dicho de otra manera, una implementación de programación genérica.

Se dispone de una clase Mapa que contiene la información de todos los nodos del entorno del cual se va a resolver el problema de búsqueda de un camino óptimo, una clase Nodo que representa cada uno de los nodos del mapa y mantiene información, y la clase A_estrella que es la encargada de resolver el problema.

La práctica dispone de un fichero Python (**constants.py**) que provee ciertos ajustes fijos para la ejecución de la aplicación, tales como: tamaño del mapa, uso del algoritmo con o sin re-enlaces, mostrar trazas de depuración por la consola, etc…

La aplicación ha sido desarrollada junto a una interfaz construida en *Qt* que permite configurar el mapa a resolver por el algoritmo, así como ver de forma fácil la representación de la solución.



Ampliaciones

1. Resolución del problema del camino óptimo realizando re-enlaces.
2. Cálculo real de la distancia recorrida desde el inicio (g real).
3. Interfaz gráfica.
4. Implementación de waypoints.
5. Implementación de celdas con penalizaciones.
6. Implementación de resolución de waypoints mediante multiprocesamiento.

Manual de uso

Requisitos

Se requiere que el sistema tenga instalado Python 2.7 (https://www.python.org/downloads) y la librería gráfica PyQt4.

Generar ejecutable

Para generar el ejecutable entregado se ha usado Py2exe (http://www.py2exe.org/).
Una vez instalado, introduce en un terminal de Windows:
>> python deployToEXE.py py2exe
El ejecutable y las librerías se incluyen en la carpeta **dist/**

Activar el multiprocesamiento

Para activar el uso de procesos concurrentes en la resolución del algoritmo, en el archivo **constants.py** la propiedad **allow_multiprocessing()** tiene que devolver **True**. Para que los cambios tengan efecto se tendrá que generar el ejecutable de nuevo.
El uso de varios procesos acorta el tiempo de ejecución del algoritmo.

Activar el re-enlace

El programa da la oportunidad que el problema se resuelva con re-enlace de nodos o sin él. Para activar esta opción en el archivo **constants.py** el valor de la propiedad **algorimthReLink**() tiene que devolver **True**. Para que los cambios tengan efecto se tendrá que generar el ejecutable de nuevo.

Consideraciones generales

El ejecutable entregado está compilado con la opción de multiprocesamiento desactivada y la opción de re-enlace de nodos activada.

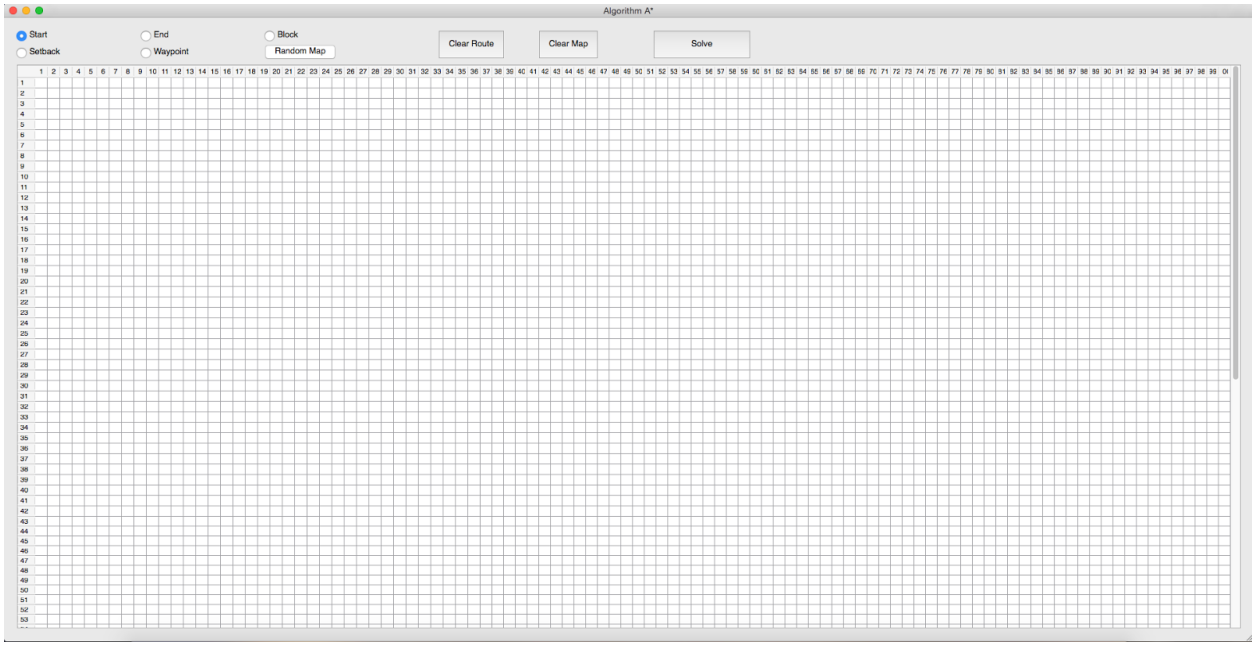
Además en el archivo **constants.py** se puede configurar el tamaño del tablero, el color de cada tipo de celdas y el valor de la penalización de las celdas del tipo Setback.

Inicio

- Windows: Solo será necesario ejecutar el fichero **ui_main.exe.**
- Linux: Descomprima el fichero A_estrella_unix.zip, entre en la carpeta que se ha descomprimido y a continuación ejecute en un terminal: python ui_main.py.
- OS X: Descomprima el fichero A_estrella_unix.zip, entre en la carpeta que se ha descomprimido y a continuación ejecute en un terminal: python ui_mani.py.

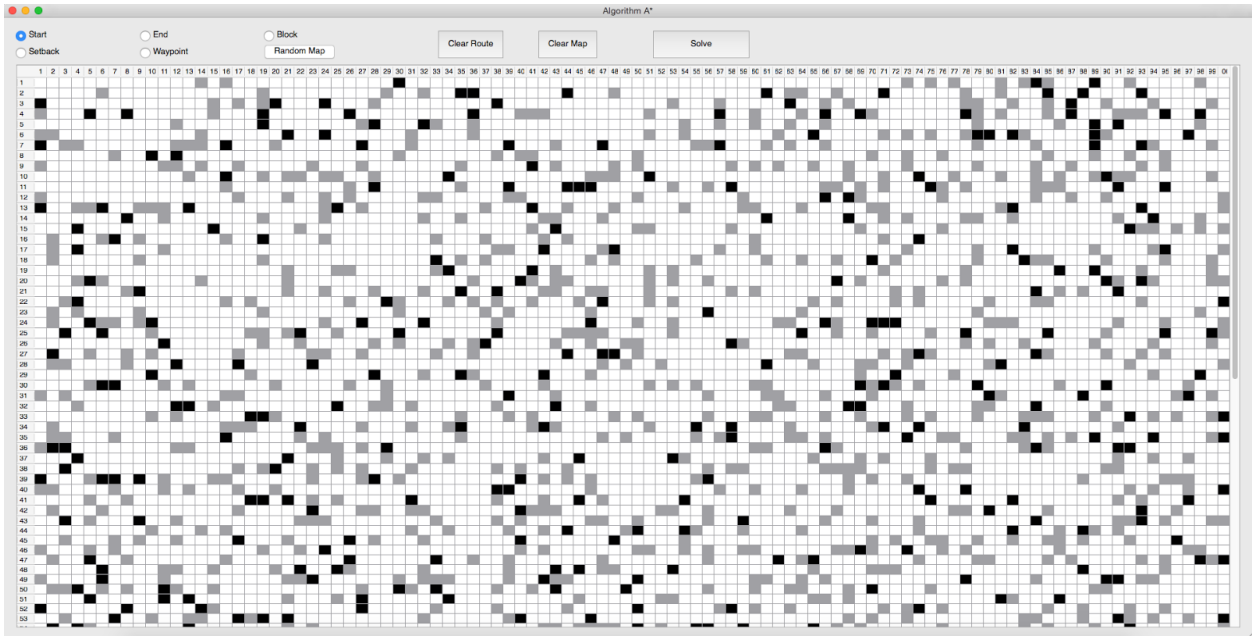
Uso

Una vez iniciada la aplicación se mostrará la siguiente ventana.

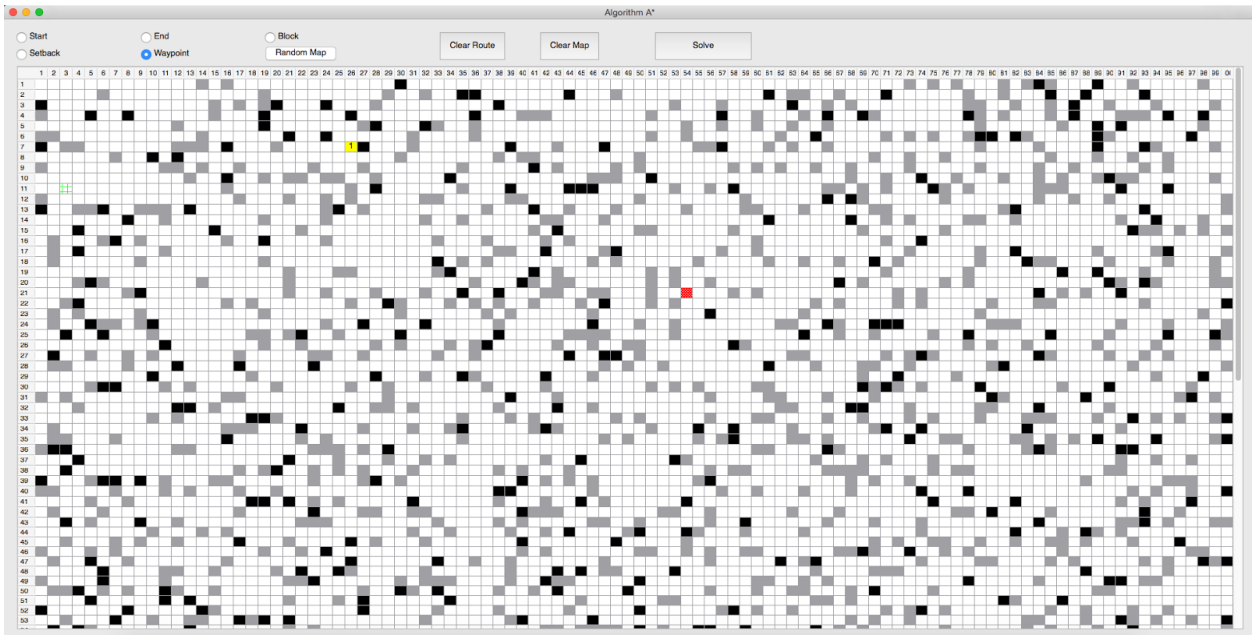


En la parte superior izquierda se permite seleccionar el tipo de elemento que quiere utilizar para confeccionar el mapa a solucionar, ya sean: obstáculos, elementos penalizadores, el inicio, la meta y waypoints.

Por otro lado, si no precisa de la generación personalizada de un mapa puede pulsar el botón *random map* y generará uno aleatorio (no comprenderá el inicio, la meta ni waypoints).

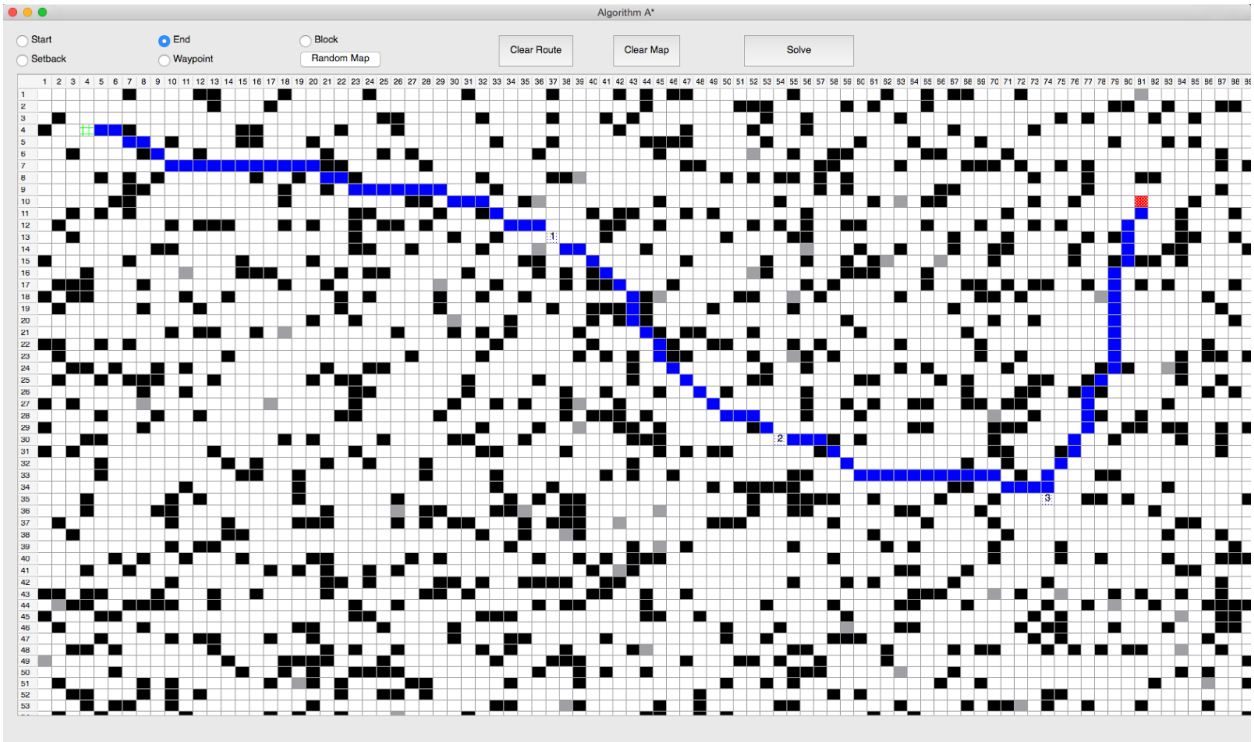


Un a vez generado el mapa aleatoriamente deberá, con el radio button *Start* seleccionado, definir el inico en el mapa pinchando sobre cualquier casilla ocupada o libre; de la misma forma debe hacerse para la meta y/o para los waypoints. Resultado algo similar a la siguiente figura.



Se puede observar que el inicio queda representado por una malla verde, la meta por una malla roja, y los waypoints por una casilla amarilla con el número del waypoint.

Llegados a este punto solo hay que pulsar en *Solve* para que la aplicación resuelva el problema planteado.



El resto de opciones: *Clear Route* sirve para borrar la ruta sin quitar el mapa, pudiendo definir así nuevo inicio y meta; en el caso de *Clear Map* se limpiará el mapa y ruta para permitir generar uno completamente nuevo.