Memoria Práctica 3

Integrate path-planning and task-planning

2020

David García Ortiz

UAH

25/05/2020

# Introducción

El objetivo de esta práctica es el de comparar los algoritmos de Dijkstra, A\* y Theta\*; aplicar heurística al algoritmo Theta\* para así comprobar los cambios y el impacto que tiene aplicar dicha heurística. Para las simulaciones se usará R2P2. Finalmente, realizará la integración de path-planning y task-planning.   
Esta práctica se ha realizado de manera individual, aunque si bien se ha intercambiado información con algunos compañeros para resolver dudas en ciertos puntos de esta.

# Instalación

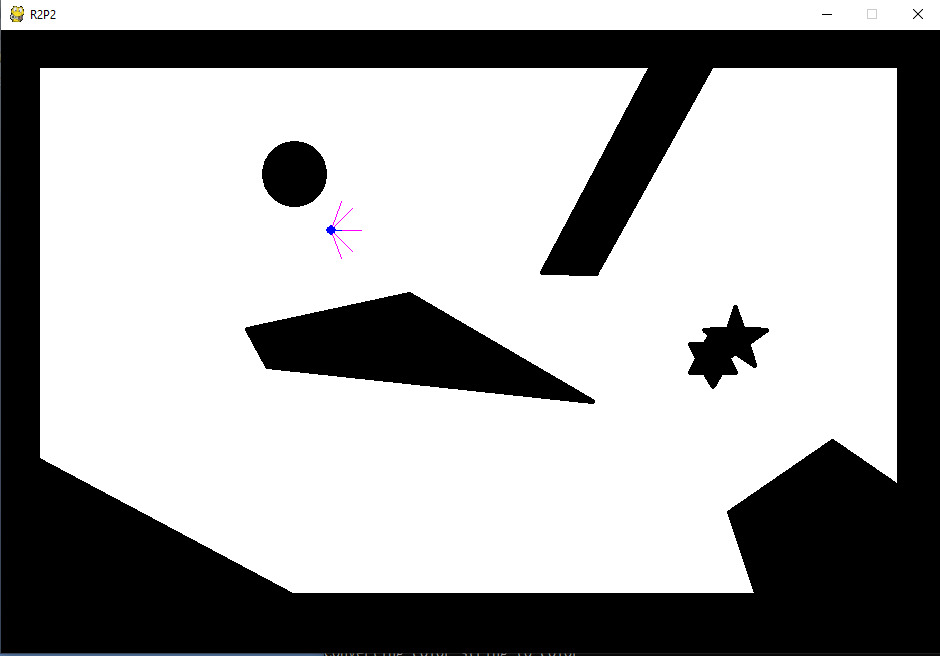
Una vez se ha instalado R2P2, Anaconda, y se ha creado el respectivo enviroment con Conda, y tras realizar las convenientes actualizaciones a las versiones de python y de pip que se indican en la guía, ya se tiene todo a punto para ejecutar el simulador.

# Ejercicio1

La primera prueba la haremos con el siguiente comando desde la consola de comandos de Conda en la carpeta de r2p2:



El resultado de ejecutar este comando será el siguiente:



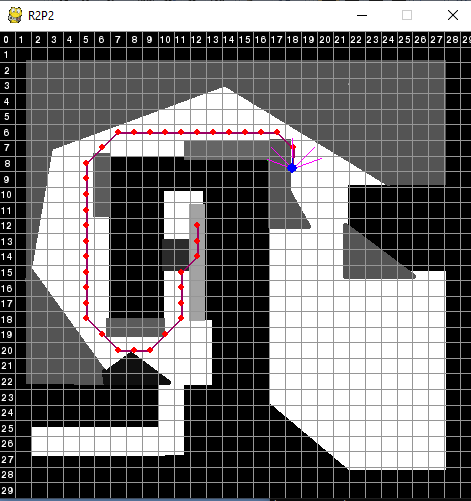
Se trata de un simulador de un robot con sensores, el cual podemos controlar con las flechas del teclado.

# Ejercicio2

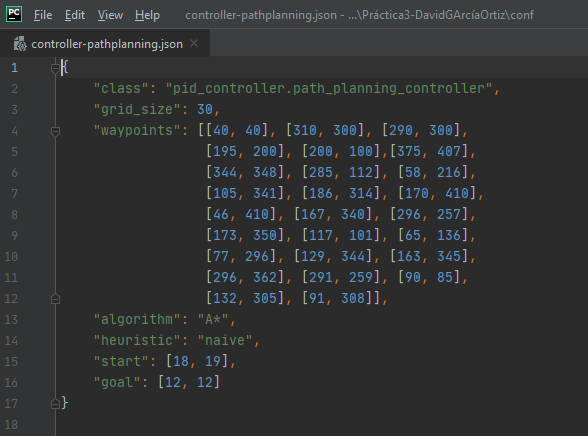
A continuación, se ejecutará el escenario Path-planning como se indica en la práctica, con el siguiente comando:



El resultado será el siguiente:



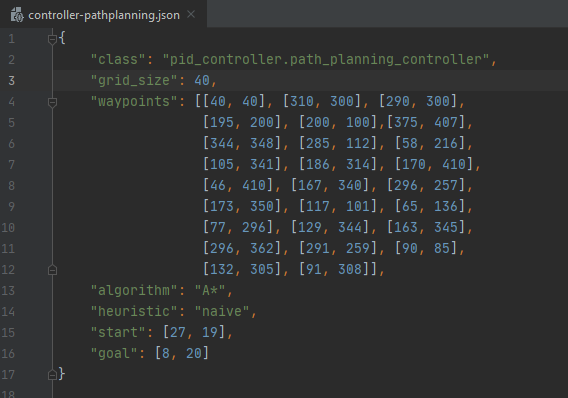
El robot tratará de seguir una ruta preestablecida desde el punto inicial hasta el final, sorteando los obstáculos. El punto de inicio y de fin viene establecido dentro de la clase controlador que hemos seleccionado:



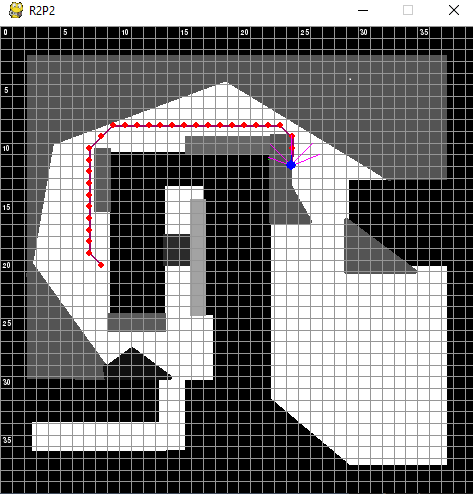
Para la visualización y modificación de los archivos de estará utilizando a lo largo de toda la práctica en entorno de Python: *PyCharm.*

# Ejercicio 3

Como se pide, se cambiará el punto de origen de (18,19) a (27,19), así como la meta se cambiará de (12,12) a (8,20). También se ampliará el tamaño de la rejilla a 40. El resultado del .json es el siguiente:



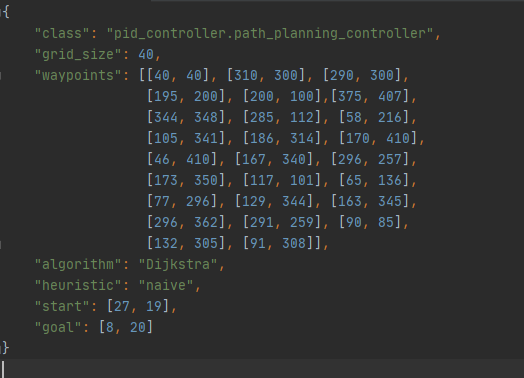
Al ejecutar de nuevo el Path-planning obtendremos lo siguiente:

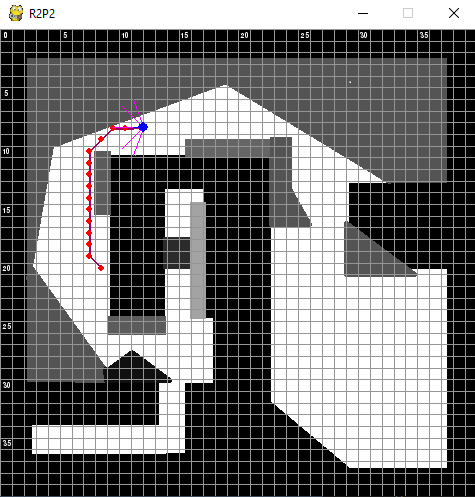


Como se puede apreciar en la imagen, se han producido los cambios indicados.

# Ejercicio 4

Si en el campo de “algorithm” del controlador cambiamos “A\*” por “Dijkstra” el simulador se ejecutará teniendo en cuenta este algoritmo:





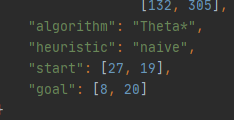
# Ejercicio 5

Como el algoritmo Theta\* no está implementado, se implementará sirviéndonos del pseudocódigo que se nos ofrece en la práctica. Ya que es muy similar al algoritmo de A\*, se copiará ese archivo y se realizarán las respectivas modificaciones sobre este, como ha sido añadir la función *LineOfSight*, modificar la función *aStar* y adecuarla a la nueva función *tStar*, donde se usa *LineOfSight*, además de añadir al final la función de *check*, para devolver el color del grid.

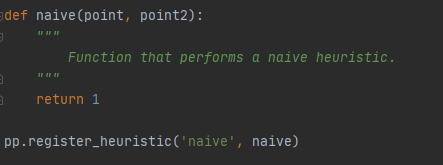
Este archivo se encontrará en la carpeta de r2p2 bajo el nombre de *thetaStar*.

# Ejercicio 6

La heurística elegida por defecto la podemos observar de nuevo en el .json del controlador de Path-planning. En este caso la heurística por defecto es “Naive”.



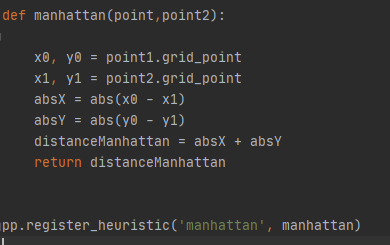
Podemos ver en qué consiste esta heurística si accedemos al archivo “heuristics.py” de la carpeta r2p2:



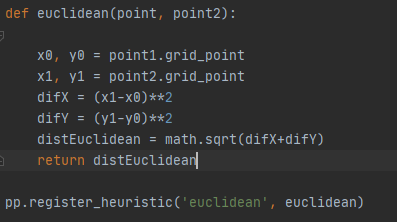
Como se aprecia, devuelve como coste entre dos puntos siempre un 1.

# Ejercicio 7

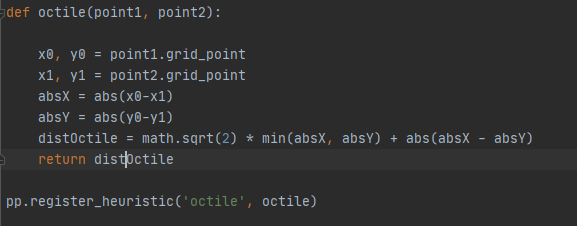
En este ejercicio se implementarán las heurísticas de *Manhattan*, *Octile* y *Euclidean* para los algoritmos *A\** y *Theta\**. Para ello modificaremos el archivo de heurísticas indicado en el ejercicio anterior.  
Se empezará implementando la heurística de *Manhattan*, cuyo resultado será el siguiente:



La siguiente en implementarse será la heurística *Euclidean*, cuyo resultado es el siguiente:



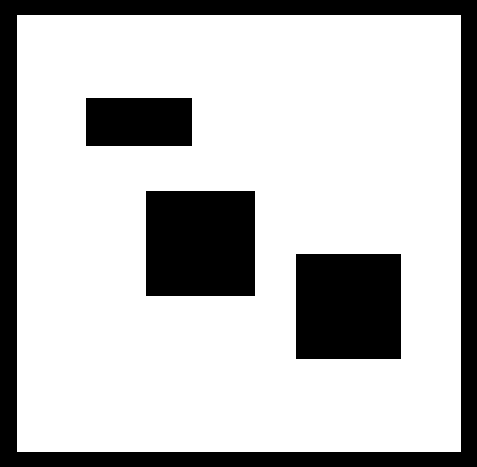
La ultima en ser implementada, y la cual se deberá añadir a dicho archivo, será la heurística *Octile*. El resultado de su implementación será el siguiente:



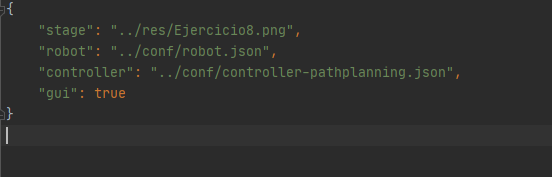
No se adjuntarán capturas de su funcionamiento en este ejercicio, ya que se utilizarán todas y se adjuntaran las imágenes correspondientes en el siguiente ejercicio:

# Ejercicio8

Se pide que se cree un escenario en concreto. En mi caso, lo he realizado a paint de manera aproximada, y el resultado ha sido el siguiente:



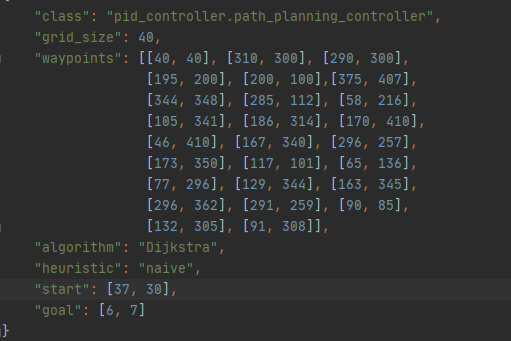
Se puede encontrar en la carpeta “res” bajo el nombre de “Ejercicio8”. Para utilizarlo, cambiaremos el fichero “scenario-pathplanning.json” y elegiremos nuestra imagen:



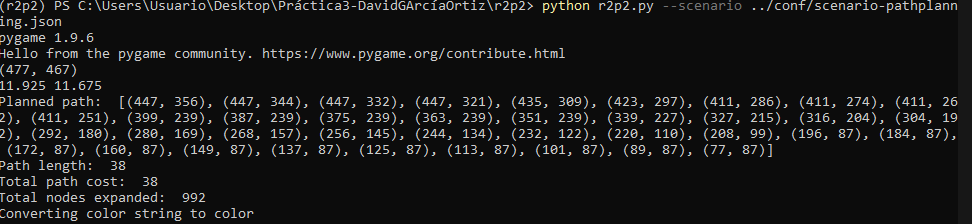
A continuación se procederá aprobar las diferentes heurísticas con los diferentes algoritmos en el nuevo escenario que acabamos de crear.

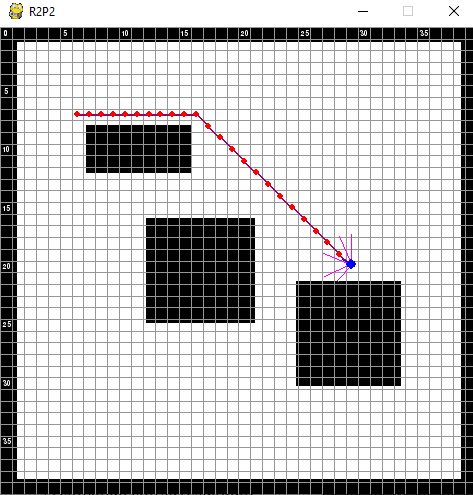
## Algoritmo de Dijkstra heurística naive

Para esta ejecución, dejamos el controller tal que así:

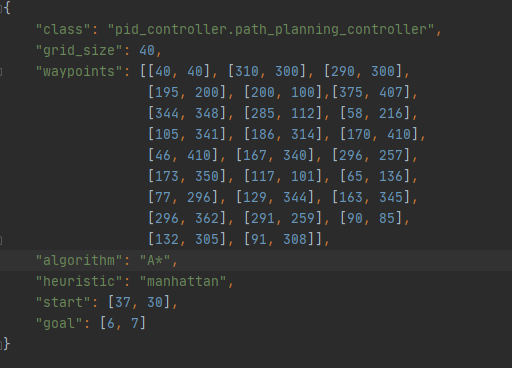


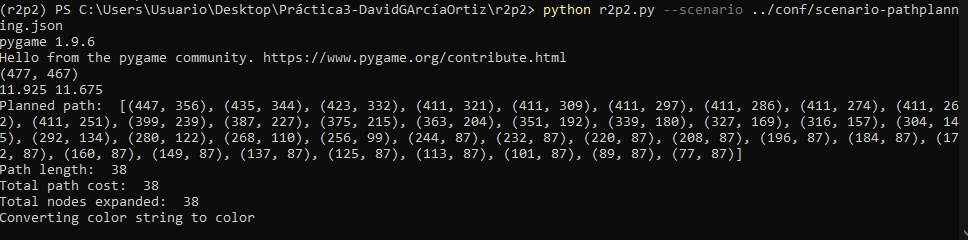
El comando con su resultado:

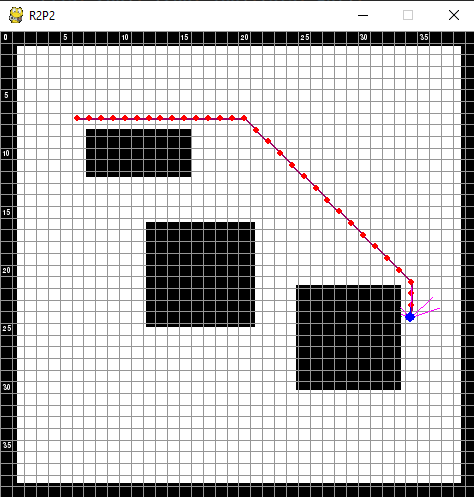




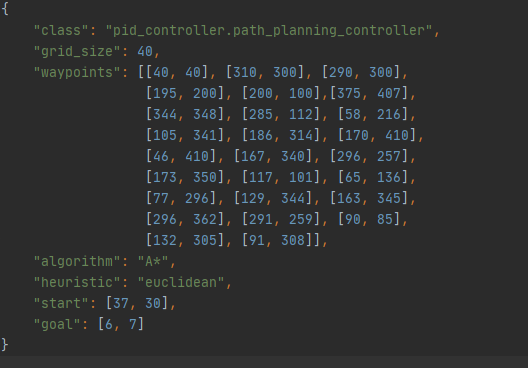
## Algoritmo A\* heurística Manhattan

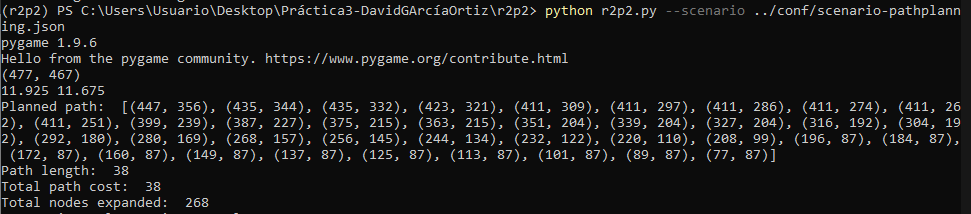


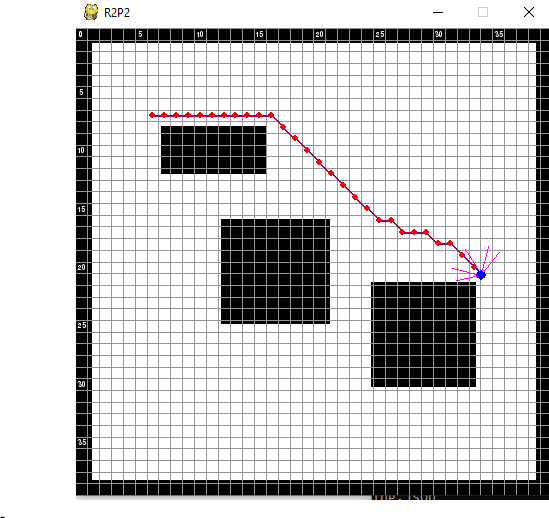




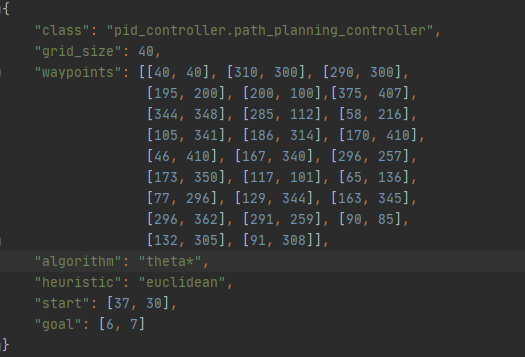
## Algoritmo A\* heurística Euclidean

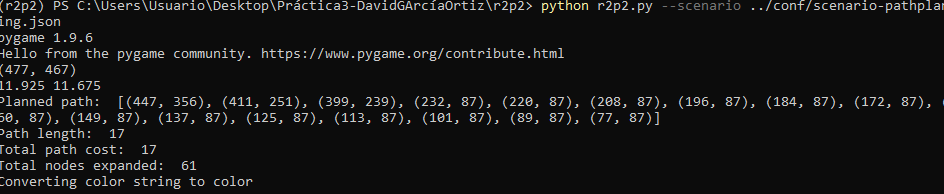


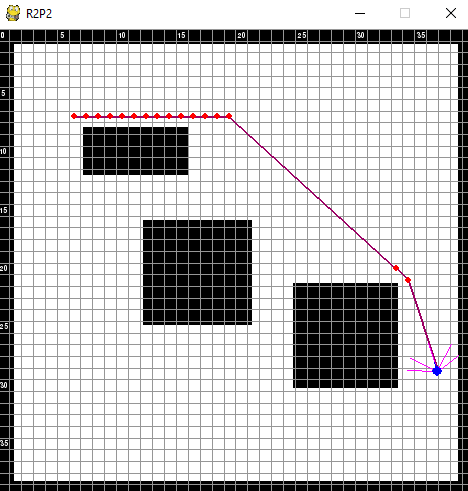




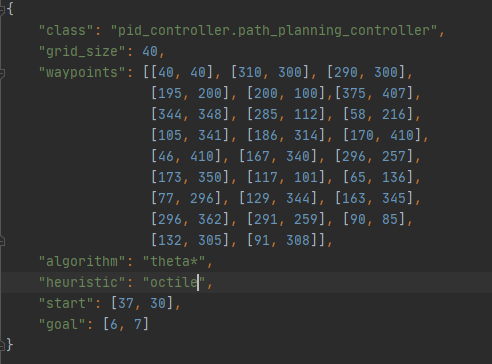
## Algoritmo Theta\* heurística Euclidean

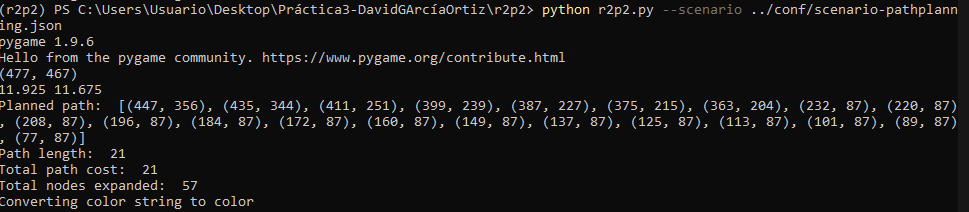


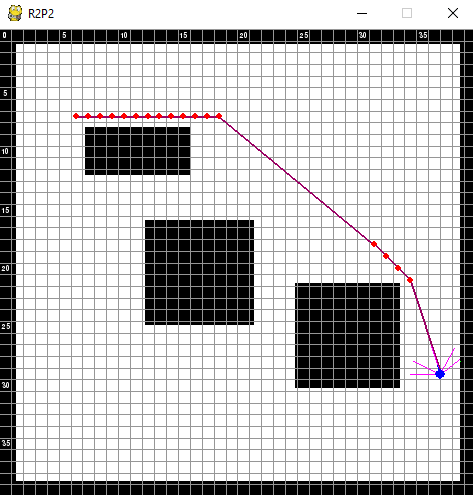




## Algoritmo Theta\* heurística Octile







Con este número de pruebas creo que es suficiente para tener unas ideas básicas de cómu funciona cada algoritmo con cada heurística y cuál es el rendimiento de estos.

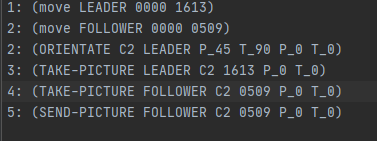
A continuación se pasará con la segunda parte de la práctica, la cual se trata de la integración.

# Integración

Para este apartado se pide modificar el dominio del ejercicio “Cooperation” de la practica anterior de tal manera que las coordenadas se tomen como un único punto.

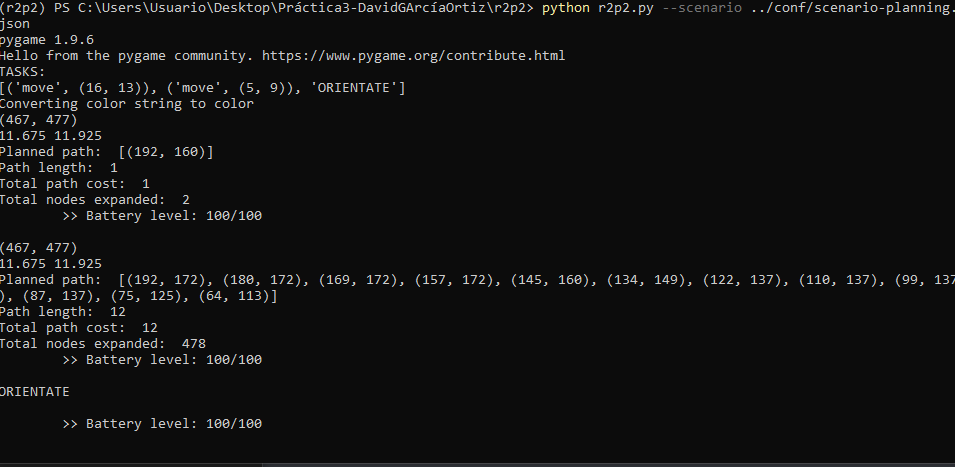
En mi caso, ya tenía el ejercicio modelado de esa forma, por lo que directamente cogeré la solución generada y la guardaré en un fichero llamado “planning.txt” y se introducirá en la carpeta “res”. También, se cambiará el escenario del ejercicio y se utilizará la imagen “Sandbox” ya que no tiene obstáculos y servirá mejor para observar el movimiento.

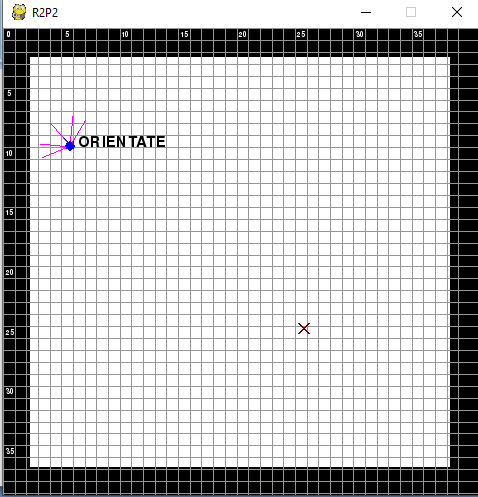
El resultado de la ejecución del dominio “Cooperation”, transformándolo en el formato requerido para la ejecución, es el siguiente:



Se realizan las modificaciones pertinentes en el fichero de “pddl\_executor.py” y se indica que se quiere realizar con el algoritmo Dijkstra y con la heurística naive.

Se procede a ejecutar el comando:





Se ha ejecutado correctamente.

Sin embargo, con otra prueba, en concreto utilizando algoritmo A\* y heurística octile, no encuentra camino posible, dando lugar al siguiente resultado al ejecutar:



# What else?

Finalmente, se nos plantea como cuestión el si es suficiente todo lo visto a lo largo de esta práctica como para lograr un control autónomo del robot, y en caso negativo, como se podría mejorar y que se necesitaría.

Por la experiencia que he adquirido en otras asignaturas como Sistemas de Control Inteligente y otras asignaturas en cuyo temario se recogían asuntos de robótica, puedo sacar las siguientes conclusiones:

* El robot en sí con el que se trabaja en esta práctica tiene un carácter muy simple. No tiene demasiados sensores u otras herramientas que le permitan interactuar con el entorno, como podrían ser sensores tanto de luz como de ultrasonido, de proximidad, de presión, etc. Con elementos como estos se podrían realizar entrenamientos que preparasen al robot de cara a toda variedad de situaciones, lo cual enlaza con el siguiente punto.
* Sería interesante añadir al robot un sistema de aprendizaje (machine learning) de manera que mediante la repetición y entrenamiento (tanto manual como autónoma) el robot “aprenda” a reaccionar a todos los estímulos que sus herramientas le hagan posible percibir.
* Con las herramientas adecuadas y sensores tanto de altura como de profundidad, se podría añadir una tercera dimensión al mapeado, introduciendo obstáculos como rampas, caídas, y otros tipos de terrenos para los que el robot de esta práctica no esta preparado.

En conclusión, esta práctica nos muestra las bases del campo de Path-planning, el cual no deja de ser solo una de las muchas piezas de las que se debería formar un robot que realmente aspire a ser autómo.