Práctica 2 DBA:

Find the lost Jedi

Ángel Gómez Ferrer
Raúl Rodríguez Pérez
Rubén Delgado Pareja
Agustín Mérida Gutiérrez

Índice

1. Estudio de los mundos y características (orografía, nivel máximo de vuelo...)

En una primera aproximación nos hemos enfocado en resolver los primeros mundos desde el mundo Abafar hasta Endor para asegurar una nota superior a 5, dedicando cierto tiempo al estudio de cada uno de ellos en cuanto a su orografía y altura mayoritariamente.

- En el estudio realizado sacamos las conclusiones del primer mundo **Abafar** en el cual en cuanto a la altura consideramos que no hay mucha variación de altura, y con bastante parecido al mundo de Dagobah de la práctica 0 y 1 en la que el agente TieFighter comenzará en el centro del mapa y el Jedi se encontrará en la parte superior izquierda del mapa.

- **Batuu** se puede ver que hay una franja de picos altos que deberá pasar por encima el agente ya que aparece relativamente cerca en la parte inferior izquierda para llegar al jedi con una posición cercana al centro por tanto podrá recorrer un trayecto en diagonal hasta el sobrepasando los picos de altura.

- En cuanto a **Chandrila** se puede ver que es un terreno con un par de superficies bastante altas, abrupto, con bastantes diferencias de altura en todo el mapa, en este el Jedi se encuentra en la parte superior derecha del mapa y la nave aparecerá en la esquina inferior derecha del mapa por lo cual tendrá que recorrerlo de lado a lado para capturar el objetivo.

- El caso de **Dathomir** es muy similar a Chandrila en cuanto al relieve aunque con mayor diferencia entre la altura mínima y máxima que podemos diferenciar, en la que Luke se encuentra en la parte inferior intermedia y la nave comienza en la esquina superior izquierda, prácticamente obligando al TieFighter a cruzar las partes más altas del mapa.

- El último mapa que se puede conseguir el objetivo con Direct-drive al igual que los anteriores es **Endor**, el cual tiene superficies extensas con diferentes alturas pero estables, en este mapa el Jedi se encuentra en la esquina inferior izquierda y la nave aparece en la parte superior derecha teniendo que hacer un recorrido en diagonal por todo el mapa para atrapar al objetivo.

- Para resumir sobre **Felucia, Hoth** y **Mandalore** ya que son realmente parecidos podemos decir que todos cuentan con diversas superficies las cual no podrá pasar nuestra nave, es decir tendrá que rodear esas zonas para poder llegar al objetivo, por ello en estos mundos ya no se podrá usar Direct-drive para superarlos, habrá que utilizar algoritmos como basic surround o right o left wall para ello, en cuanto a la altura de los mundos tienen en general alturas altas pero bastante uniformes en todo el mapa.

- Para terminar vamos a hablar sobre **Tatooine** y **Wobani** son mapas uno de ellos sin solución y otro muy dificil de conseguirla ya que no tienen sol y

cuentan además con un terreno muy variable y zonas que tendrá que rodear nuestra nave.

2. Sensores elegidos para resolver los mundos

Como sabemos, el objetivo de la práctica es el diseño e implementación de un agente que sea capaz de desplazarse y encontrar un objetivo en un mundo virtual 3D. La orografía de estos mundos 3D se desconoce totalmente, pero puede llegar a percibirse haciendo uso de los sensores que dispone el agente. Por ello, la elección y el uso de estos sensores es clave para conseguir los objetivos en los diferentes mundos, puesto que, con la información que nos proporcionan debemos evitar que el agente colisione con el suelo, con otro agente, con los límites del mundo o se quede sin energía.

Así que, una vez que realizamos el estudio de los mundos para conocer sus características, nuestro equipo realizó una puesta en común para elegir cuales son los sensores que utilizaremos para conseguir los objetivos. Dichos sensores, junto una breve explicación de su uso y el por qué pensamos que son necesarios, son los siguientes:

- **Alive**: Este sensor es necesario ya que define si el agente está vivo o no con el uso de un booleano. Por lo tanto su uso es imprescindible, ya que será utilizado para saber si el agente ha muerto tras colisionar con el suelo, con los límites del mundo...
- **Ontarget**: Este sensor define si el agente se encuentra exactamente en la posición del objetivo, por lo tanto, también es imprescindible su uso para conocer si se ha completado un mundo con éxito o no.
- **GPS**: Con este sensor podremos saber la posición del agente (con respecto a su posición en el mapa del dashboard gráfico) en un eje de coordenadas (x,y,z). Vimos necesario su uso ya que conocer las coordenadas del agente en un momento determinado nos podría servir para diversas funciones.
- **Compass**: Este define un valor que indica la orientación del agente. Crucial a la hora de realizar los desplazamientos del agente, pues seguramente necesitemos saber si tiene la orientación correcta antes de moverse.
- **Altitude**: El sensor define la altura a la que se encuentra el agente sobre el suelo. Vimos necesario su uso sobre todo para casos como el de querer recargar al tener poca batería, ya que es oportuno saber cual es la distancia al suelo para calcular si llegamos a tiempo a recargar o no.
- **Visual**: Define una malla 11x11 con el valor que indica la elevación del suelo alrededor del agente. Sensor necesario para poder actuar frente a un cambio en el terreno, por ejemplo, elevar la tie fighter para que no se choque.
- **Energy**: Sensor que define la energía que dispone el agente para realizar acciones. Este es uno también de los imprescindibles, ya que debemos evitar que dicha energía se agote antes de poder encontrar el objetivo.
- **Distance**: Este sensor nos indica la distancia a la que se encuentra nuestro objetivo (el Jedi), sin tener en cuenta la dirección. Su uso es más que evidente, ya que es una parte fundamental para conocer y establecer la ruta del agente para llegar al objetivo.
- **Angular**: Define el ángulo al que se encuentra el objetivo. Perfecto complemento para distance, tal y como se nos explica en el manual de referencia, ya que con los dos sensores podremos saber la distancia y la orientación en la que se encuentra el objetivo, para que nosotros podamos realizar las modificaciones/órdenes concretas en la tie fighter para que llegue al mismo.

Por otro lado, hemos descartado algunos de los sensores que podíamos elegir ya que lo veíamos innecesarios o simplemente pensamos que sus funciones no ayudarían para la resolución de los mundos:

- **Thermal**: Define una malla de 11x11 que rodea al agente hacia el objeto más cercano que desprenda calor, o sea, el objetivo. Creímos que con el uso de sensores como Distance y Angular ya tendríamos suficiente como para conocer la posición del objetivo, por lo que su uso sería innecesario.
- **Lidar**: Este sensor define una malla 11x11 definiendo la distancia del agente con respecto al suelo, teniendo en cuenta también si existen celdas que se encuentran por encima del agente, en cuyo caso se reflejarán con valores negativos. Una vez más, pensamos que con el uso de Altitude y Visual tenemos la suficiente información para conocer la altitud del suelo con respecto al agente y viceversa.
- **Payload**: Define el número de objetos almacenados en la bodega del agente. Sensor innecesario ya que el objetivo de la práctica es capturar al Jedi, no hay que recoger ningún objeto por el camino.

3. Gestión de la energía

El objetivo principal de esta práctica, como hemos comentado anteriormente, es localizar y capturar al Jedi en cada uno de los mundos. Este aunque es el objetivo principal, no es el único que tenemos. Existen algunos objetivos secundarios los cuales debemos cumplir para tener éxito en nuestra misión. Algunos ejemplos de estos pueden ser, conseguir que la tie fighter no colisione en ningún momento de su viaje, conseguir colocarse justo encima de la celda objetivo, y uno de los de vital importancia, gestionar el uso de la energía para poder realizar los distintos movimientos y/o acciones.

Para cada uno de los mundos, el agente empieza con un nivel de energía máximo de 3500 W. Dicha energía se va a ir consumiendo a medida que la tie fighter se desplace en cualquier dirección (consumiendo 1W por cada movimiento) o cada vez que realicemos una lectura de información para un sensor (consumiendo 1W por lectura también). Aún así, cabe destacar que tenemos acciones como la de recargar y capturar que no consumen energía, por lo que podemos realizarlas indefinidamente.

Si la Tie Fighter consume toda su energía antes de encontrar y capturar al Jedi perdido dentro del mundo, se habrá fracasado en la misión, ya que sin energía no podrá realizar ninguna acción para buscar el objetivo. Para evitar el fracaso, podemos realizar la acción de ‘Recharge’ que nos permite volver a recargar la energía de nuestra tie fighter de nuevo, restableciendo su valor a 3500 W. El único requisito que dispone esta acción es que la nave debe estar posada en el suelo, es decir, el valor del sensor ‘Altitude’ debe valer 0. Aún así, si intentamos realizar la recarga sin que la tie fighter esté posada en el suelo, Larva simplemente nos dará un warning, avisándonos de que no se ha realizado correctamente la acción pero que se puede continuar.

Por lo tanto, con toda esta información solo nos quedaba decidir sobre cuándo realizar la acción de recargar. Esta decisión es importante, ya que podría darse el caso de que, si es muy conservadora, haya que recargar muchas veces, con las subidas y bajadas que eso conlleva, o si es muy arriesgada, que no dé tiempo a aterrizar y se quede sin energía antes. Por esto decidimos seguir una estrategia de cambios mínimos de altura, es

decir, a lo largo del transcurso del viaje de la nave, está solo subirá de altura si es estrictamente necesario. Y una vez, se haya alcanzado el límite de energía impuesto por nuestro criterio, dicha nave bajará hasta posarse en el suelo para recargar y volver a retomar el proceso de nuevo. Hemos elegido esta estrategia, no solo porque era la que veíamos más intuitiva, sino porque de las estrategias explicadas era la que lograba un menor consumo de energía.

Cabe mencionar que en nuestro algoritmo implementado para resolver los mundos, hemos desarrollado una sección del código en donde realizamos una recarga cada cierto periodo de tiempo (aunque la nave no se encuentre en el suelo). Esto lo implementamos para conseguir unos de los milestone que se pedían, que era realizar acciones que pasen cada cierto tiempo. Además vimos oportuno utilizar la acción de recargar para conseguir este milestone, ya que como hemos explicado anteriormente, no hay problema en realizar la acción sin la nave posada, ya que lo único que ocurrirá será un warning pero las demás acciones podrán continuar sin problema.

4. Heurísticas utilizadas

Tal y como hemos comentado anteriormente, para la resolución de los 5 primeros mundos (desde Abafar a Endor), nuestro equipo ha implementado una solución basada en la heurística “Direct Drive”. Dicha estrategia consistía en, de primeras, comprobar que la tie Fighter tuviera la orientación correcta, es decir, que estuviera apuntando al lugar en donde se encontraba el Jedi objetivo. Si dicha orientación no era correcta, la modificamos. Una vez que la nave está bien posicionada, hacemos un reconocimiento de las casillas que tenemos alrededor para decidir cuál es nuestra próxima casilla. Si la casilla a la que apunta la nave tiene una altura menor o igual que la actual, simplemente nos desplazamos hacia esa casilla. Si por el contrario, la casilla a la que apunta tiene una altura mayor, realizamos la acción de elevarnos para poder avanzar por dicha trayectoria. Teniendo en cuenta esta secuencia de actividades (e incluyendo las funciones que hemos comentado en el apartado anterior de la gestión de energía), nuestro tie fighter era capaz de superar sin problemas estos primeros 5 mundos.

Una vez comenzamos a probar los siguientes mundos (desde Felucia hasta Wobani) nos dimos cuenta de que con lo que habíamos hecho no era suficiente, ya que no conseguimos lograr los objetivos de dichos mundos. En una primera instancia, mirando las transparencias de teoría, optamos por implementar una de las heurísticas Left Wall o Right Wall. Estas consistían en que si encontraba un obstáculo el cual no se pudiera superar por encima, se tomara la decisión de bordear dicho obstáculo, yendo a la izda o dcha según lo que hubiéramos decidido. En una primera aproximación que hicimos, conseguimos resolver algunos de los mundos como Felucia y Hoth, pero Mandalore se resistía. No sabíamos muy bien el motivo de por qué en Mandalore no conseguía llegar al objetivo, y tras una reunión por parte del equipo, decidimos utilizar memoria como una opción alternativa ya que lo que teníamos no era suficiente. Por lo que, finalmente teníamos entre manos una algoritmo que seguía las bases del direct drive, es decir, establecer la trayectoria hacia el objetivo y seguirla, pero con la peculiaridad de que si se encontraba un obstáculo que no pudiera volar por encima, lo rodearía (modificando su trayectoria), y además nos aseguraríamos de que dicho rodeo fuera correcto, almacenando unas k casillas para que no volviera por donde ha venido y se quedara atrapado en un ciclo. Por lo tanto, podemos decir que tenemos un algoritmo que pilla o toma referencias de varias heurísticas (direct drive, right/(left wall y basic surround). Con esta nueva implementación del algoritmo conseguimos sobrepasar

esos problemas que obtuvimos anteriormente y resolver los mundos restantes hasta llegar a Tattoine y Wobani. En estos dos casos, el primero de ellos no tenía solución puesto que la nave se ponía a dar vueltas sin llegar nunca a encontrar el objetivo. Por lo que podríamos decir que nuestro algoritmo se comporta de manera “correcta” en dicho mapa. Por último, el mapa de Wobani lo conseguimos resolver pero modificando el algoritmo específicamente para dicho mapa, ya que con el número k de casillas almacenadas que habíamos elegido no lo lograba superar. Por lo tanto podemos decir que es el único mundo que nos faltó por resolver con nuestro algoritmo.

5. Diagrama de actividad