Inteligencia Artificial
Práctica 4: Algoritmo A*

Verónica Esther Arriola Ríos
Pedro Rodríguez Zarazúa
Luis Alfredo Lizárraga Santos

Fecha de entrega: Miércoles 9 de Marzo de 2016

1. Objetivo

Que el alumno implemente y refuerce su comprensión del algoritmo A*

2. Introducción

Vamos a asumir que tenemos a alguien que quiere ir desde el punto A hasta el punto B. Asumamos también que un muro separa estos dos puntos. El ejemplo queda ilustrado en el gráfico siguiente, donde el recuadro verde es el punto de inicio A, el rojo es el punto de destino B y la zona azul el muro que hay entre ambos.

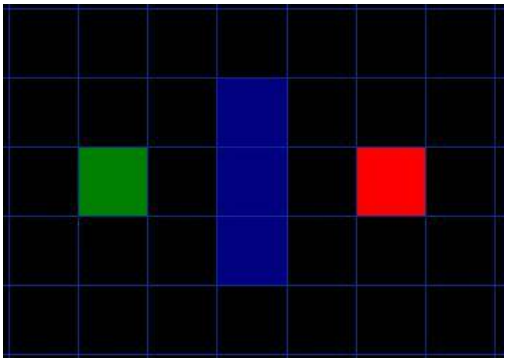

Figura 1

Notemos que hemos dividido nuestra área de búsqueda en una rejilla cuadrada. Simplificar el área, tal y como hemos hecho, es el primer paso en un pathfinding. Este particular método reduce nuestra área de búsqueda a una simple matriz bidimensional. Cada elemento de la matriz representa uno de los cuadrados de la rejilla, y su estado se almacena como transitable

o intransitable. El camino se elige calculando hacia qué cuadros deberíamos movernos para ir desde A hasta B. Una vez que el camino haya sido encontrado, el personaje de nuestro juego (o lo que sea) se moverá desde el centro de un cuadrado hacia el centro del siguiente hasta que el objetivo haya sido alcanzado.

A esos puntos centrales se les llama `nodos`.

2.1. Iniciando la búsqueda

Una vez que hemos simplificado nuestro área de búsqueda en un número de nodos asequible, tal y como hemos hecho con la rejilla de la figura anterior, el siguiente paso es dirigir una búsqueda para encontrar el camino más corto. En el pathfinding A*, lo hacemos empezando desde el punto A, comprobando los cuadros adyacentes y generalmente buscando hacia fuera hasta que encontremos nuestro destino.

Empezamos la búsqueda haciendo lo siguiente:

1. Empieza en el punto inicial A y añádelo a una **lista abierta** de cuadrados a tener en cuenta. La lista contiene los cuadrados que podrían formar parte del camino que queremos tomar, pero que quizás no lo hagan. Básicamente, esta es una lista de los cuadrados que necesitan ser comprobados.

2. Fíjate en todos los cuadrados alcanzables o transitables adyacentes al punto de inicio, ignorando cuadrados con muros, agua u otros terrenos prohibidos. Añádelos a la lista abierta también. Por cada uno de esos cuadrados, guarda el punto A como su **cuadrado padre**. El cuadrado padre es muy importante para trazar nuestro camino.

3. Saca el cuadro inicial A desde tu lista abierta y añádelo a una **lista cerrada** de cuadrados que no necesitan ser vistos de nuevo.

En este punto, se tendrá algo como la siguiente ilustración. En este diagrama, el cuadrado verde oscuro del centro es el cuadrado de inicio. Esta bordeado de azul claro para indicar que el cuadrado ha sido añadido a la lista cerrada. Todos los cuadros adyacentes están ahora en la lista abierta para ser comprobados. Cada uno tiene un puntero gris que señala a su padre, el cual es el cuadro inicial.

Después, elegimos uno de los cuadrados adyacentes de la lista abierta y más o menos repetimos el proceso anterior. Pero, ¿qué cuadro se debe elegir? Aquel que tenga costo $f(n)$ más bajo.

2.2. Puntuando el camino

La clave para determinar que cuadrados usaremos para resolver el camino está en la siguiente ecuación:

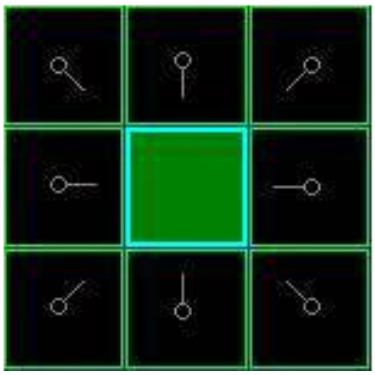

Figura 2

$$f(n) = g(n) + h(n)$$

donde:

- $g(n)$ es el costo del movimiento para ir desde el punto inicial A a un cierto cuadro de la rejilla (n), siguiendo el camino generado para llegar ahí.
- $h(n)$ es el costo del movimiento estimado para ir desde ese cuadro de la rejilla (n) hasta el destino final, el punto B. Esto se conoce como la heurística. Aquí se verá una forma de calcular la heurística, pero no es la única.

Nuestro camino se genera por ir repetidamente a través de nuestra lista abierta y eligiendo el cuadrado con la puntuación $f(n)$ más baja. Este proceso se describirá con más detalle un poco más adelante. Primero veamos más de cerca cómo calculamos esta puntuación.

Tal y como está descrito arriba, $g(n)$ es el costo del movimiento para ir desde el punto de inicio a un cuadro n usando el camino generado para llegar ahí. En esta práctica asignaremos un costo de 10 a cada cuadro vertical u horizontal hacia el que nos movamos, y un costo de 14 para un movimiento diagonal. Usamos estos números ya que es más simple poner 14 que $\sqrt{2^2} \times 10$ y porque usar números enteros es mucho más rápido para la computadora. Pronto descubrirás que el pathfinding puede ser muy lento si no usas atajos como este.

Ahora que hemos calculado el costo $g(n)$ mediante un camino específico hasta cierto cuadro, la forma de resolver el costo $g(n)$ del cuadro es tomar el costo $g(n)$ de su padre, y luego añadirle 10 o 14 dependiendo de si está en diagonal u ortogonal con respecto al cuadro padre.

$h(n)$ puede ser estimado de diferentes maneras. El método que usaremos aquí se llama el método Manhattan, donde calculas el número total de cuadros movidos horizontalmente y verticalmente para alcanzar el cuadrado destino desde el cuadro actual, sin hacer uso de

movimientos diagonales. Luego multiplicamos el total por 10. Se llama método Manhattan porque es como calcular el número de manzanas que hay desde un lugar a otro, donde no puedes acortar atravesando en diagonal una manzana. Cabe señalar que cuando calculamos $h(n)$, ignoramos cualquier obstáculo que intervenga. Es una estimación de la distancia que queda, no de la distancia actual, es por eso que se llama heurística.

$f(n)$ se calcula sumando $g(n)$ y $h(n)$. El resultado del primer paso en nuestra búsqueda puede ver se en la ilustración inferior. Las puntuaciones $f(n)$, $g(n)$ y $h(n)$ están escritas en cada cuadrado. En el cuadro inmediatamente a la derecha de cuadro inicial, la $f(n)$ está impresa arriba a la izquierda, la $g(n)$ abajo a la izquierda y la $g(n)$ abajo a la derecha.

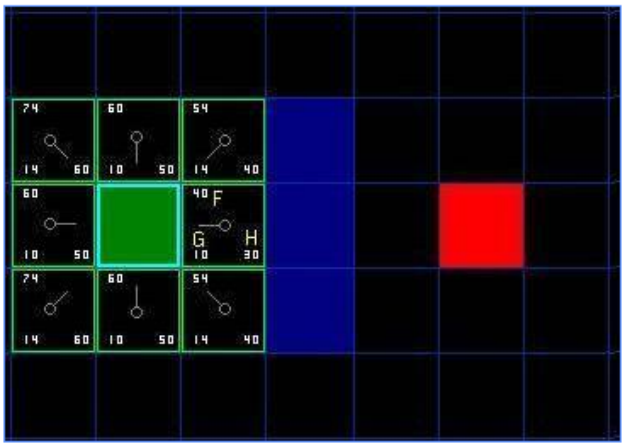

Figura 3

Así pues, vamos a mirar algunos ejemplos de estos cuadros. En el cuadrado con letras, $g(n) = 10$. Esto es debido a que está solo a un cuadro del cuadrado inicial en dirección horizontal. Los cuadrados inmediatamente encima, abajo y a la izquierda del cuadrado inicial; tienen todos el mismo valor $g(n)$ de 10. Los cuadros diagonales tienen un valor $g(n)$ de 14.

Las puntuaciones $h(n)$ se calculan estimando la distancia Manhattan hasta el cuadrado rojo objetivo, moviéndose solo horizontal y verticalmente e ignorando el muro que está en el camino. Usando este método, el cuadro de la derecha del inicial, está a 3 cuadros del cuadrado rojo con una puntuación $h(n)$ de 30. El cuadrado está a solo 4 cuadros de distancia (recuerda que solo nos movemos en horizontal y vertical) con una puntuación $h(n)$ de 40. Probablemente comprendas como se calculan las puntuaciones $h(n)$ para los demás cuadros.

2.3. Continuando la búsqueda

Para continuar la búsqueda, simplemente elegimos la puntuación F más baja de todos aquellos que estén en la lista abierta. Después hacemos lo siguiente con el cuadro seleccionado:

1. Sácalo de la lista abierta y añádelo a la lista cerrada.

2. Comprueba todos los cuadrados adyacentes, ignorando aquellos que estén en la lista cerrada o que sean intransitables terrenos con muros, agua, o cualquier terreno prohibido), añade los cuadros a la lista abierta si no están ya en esa lista. Haz al cuadro seleccionado el **padre** de los nuevos cuadros.

3. Si el cuadro adyacente ya está en la lista abierta, comprueba si el camino a ese cuadro es mejor que este. Si no es así, no hagas nada. Por otro lado, si el coste $g(n)$ del nuevo camino es más bajo, cambia el padre del cuadro adyacente al cuadro seleccionado (en el diagrama superior, cambia la dirección del puntero para que señale al cuadro seleccionado). Finalmente, recalcula la $f(n)$ y la $g(n)$ de ese cuadrado.

3. Desarrollo e implementación

La práctica consiste de implementar el algoritmo A* y generar una visualización de éste resolviendo un problema.
Ustedes pueden modificar el mundo del problema, para que puedan ver cómo se comporta A* con distintas localizaciones de obstáculos y de nodos de inicio y fin.

3.1. Implementación

Se volverá a trabajar con `Processing`.

Se debe programar lo referente al algoritmo A* y la función de heurística.
También necesitarán implementar (o usar alguna biblioteca o clase ya implementada de `Java`) sus listas abierta y cerrada, ya que se necesitan para poder hacer la implementación de A* vista en la sección anterior.

Los métodos que necesitarán implementar son los siguientes:

1. `calculaHeuristica(Mosaico meta)`

2. `expandeNodoSiguiente()`

Cada método se encuentra especificado dentro del archivo `AEstrella.java`.

4. Requisitos y resultados

Para evaluar y calificar la práctica es necesario que se implementen todos los métodos mencionados e indicados en el código, respetando implementar sólo lo que se pide (para evitar comportamientos extraños de la simulación). Es completamente válido utilizar bibliotecas adicionales si lo consideran necesario, así como la creación y uso de sus propios métodos auxiliares si lo desean.

Si crean métodos auxiliares, no olviden documentar cual es su función.

5. Notas adicionales.

La práctica es individual, anexen a su código un archivo readme.txt con su nombre completo, número de cuenta, número de la práctica y cualquier observación o notas adicionales (posibles errores, complicaciones, opiniones, críticas de la práctica o del laboratorio, cualquier comentario relativo a la práctica).

Compriman la práctica en un solo archivo (.zip, .rar, .tar.gz) con la siguiente estructura:

- ApellidopaternoNombreNúmerodepráctica.zip (por ejemplo: LizarragaLuis04.zip)
  - AEstrella
    - src
      - AEstrella.java
  - readme.txt

La práctica se entregará en la página del curso en la plataforma AVE Ciencias.
O por medio de correo electrónico a `luislizarraga@ciencias.unam.mx` con asunto `Práctica04[IA 2016-2]`

La fecha de entrega es hasta el día miércoles 9 de Marzo a las 23:59:59 hrs.

Para quienes hagan la práctica con netbeans, pongan el directorio del proyecto dentro de la carpeta `src`

Referencias

[1] Patrick Lester *A* Pathfinding para Principiantes* 2003 : http://www.policyalmanac.org/games/articulo1.htm .