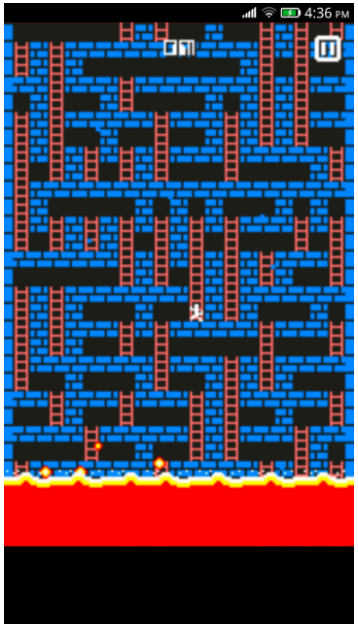
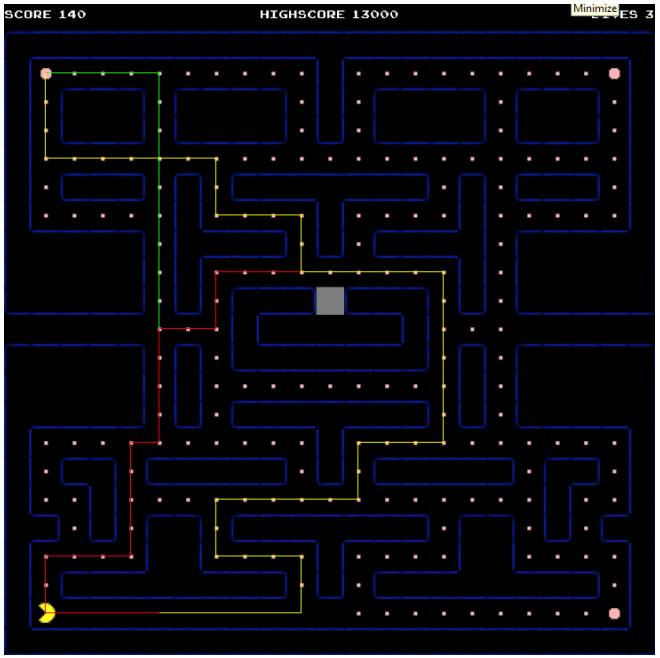
Índice:

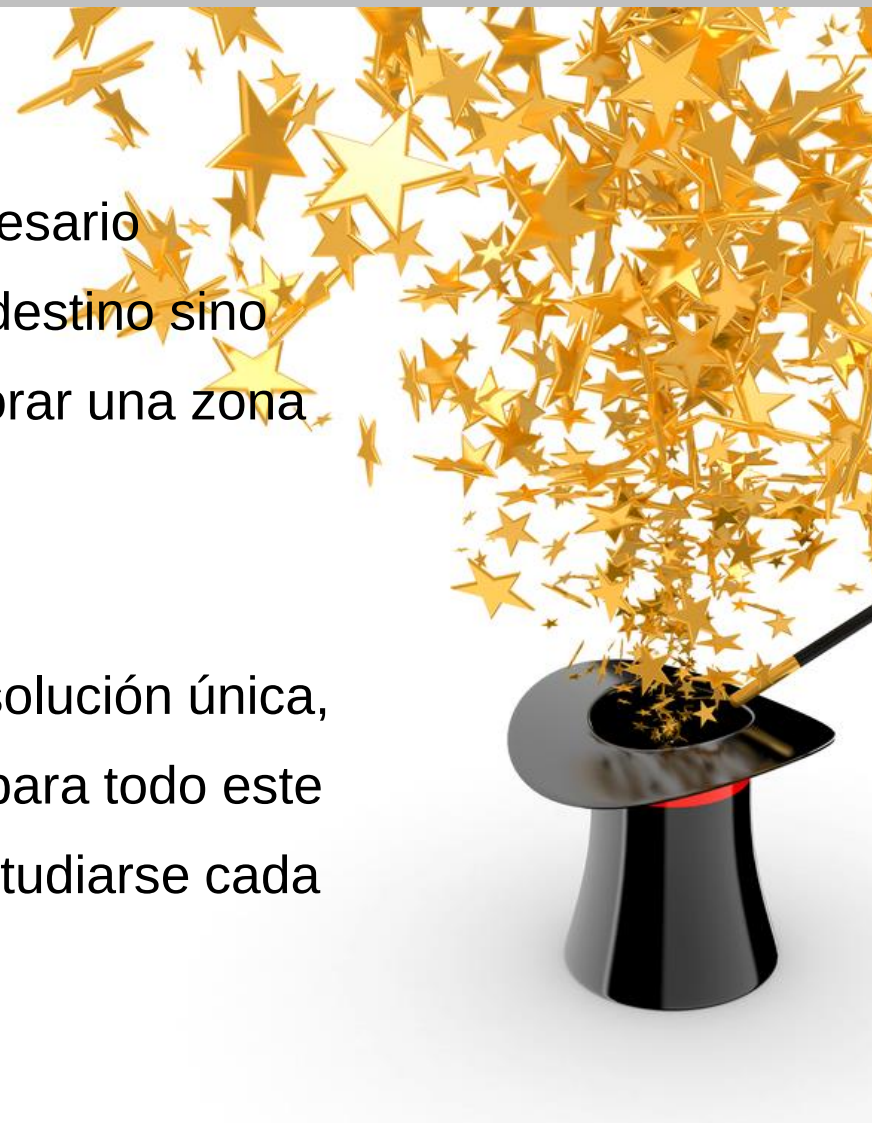
2.2 Búsqueda de caminos (*pathfinding o path planning*)

– Los problemas de búsqueda de caminos pueden ser muy diferentes según los juegos:



- ¿Destino fijo o móvil?
- ¿Con obstáculos o sin ellos?
- ¿Se mueven los obstáculos?
- ¿Qué tipo de terreno tenemos?
- ¿Es siempre la distancia más corta, el mejor camino?
- ¿Es mejor un camino más largo por una calle que uno más corto por una montaña?



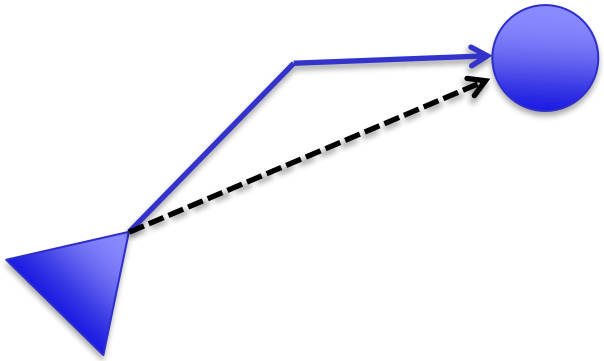


- Incluso, a lo mejor no es necesario alcanzar necesariamente el destino sino cambiar de ubicación o explorar una zona de forma inteligente.

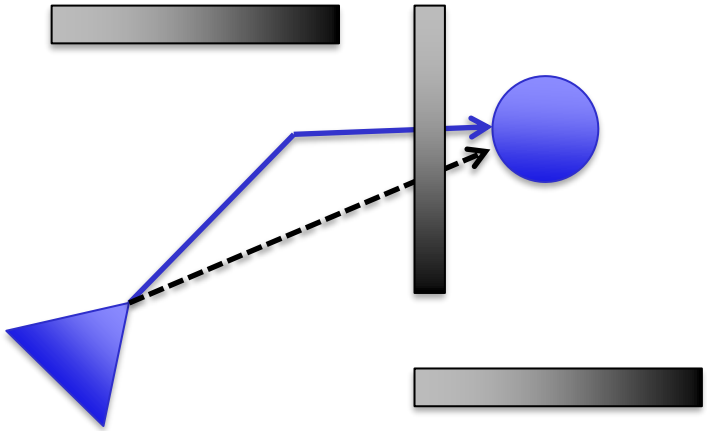
- Por todo ello, no existe una solución única, es decir, un algoritmo único para todo este tipo de problemas, y debe estudiarse cada caso.

Video demostrativo de *pahfinding* de un grupo de caracteres

- Búsqueda de caminos (continuación)
  - La búsqueda de caminos es el proceso de realizar el movimiento de un personaje desde una posición determinada a otra.
  - Las soluciones más básicas (que estarían basadas en incrementar/decrementar las coordenadas en un bucle (movimientos rectilíneos verticales y horizontales, o diagonales de 45°)), producirían un movimiento muy poco natural.

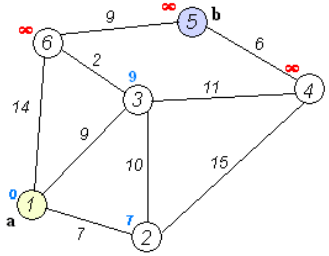


- Búsqueda de caminos (continuación)
  - Si hay obstáculos, el problema cambia sustancialmente:



- Búsqueda de caminos (continuación)
  - Se trata pues de problemas de navegación en un escenario, que se representan como una grafo, en el que se desea ir desde un nodo (inicial) a otro nodo (final) con el menor coste posible (entre nodos hay un peso/coste: distancia, dificultad, obstáculos, etc). Similar al problema de ir de una ciudad a otra dado un mapa, con muchos caminos posibles, acaso ciudades intermedias, zonas inalcanzables, etc.
  - A priori, podrían servir técnicas muy sencillas como los algoritmos voraces, pero éstos podrían terminar en soluciones de coste muy elevado; o por algoritmos de escalada que usen el coste real, que por no usar una opción peor puntual, no puedan encontrar una de coste más bajo.

- Búsqueda de caminos (continuación)

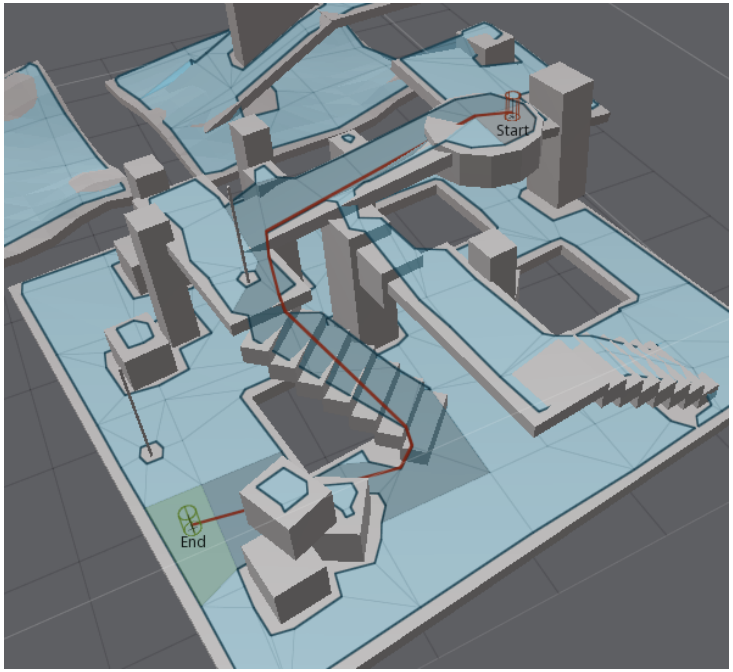


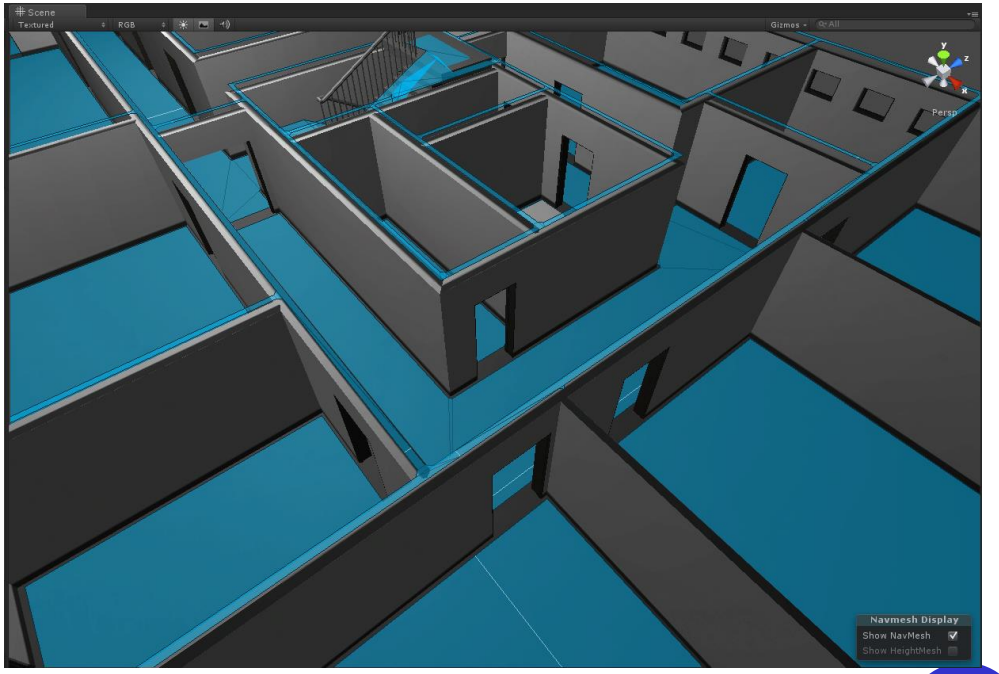
  - Genéricamente, se resolverían adaptando algoritmos tales como los clásicos Dijkstra *(no originalmente diseñado para ello)*, escalada (*hill climbing*), primero el mejor, y sobretodo A* (el cual hace uso de heurística para mejorar la eficiencia)… todos ellos conocidos por el estudio de asignaturas previas en la titulación.
  - Dentro de ellos, los mejores, son aquellos que usan valores heurísticos (h) (evaluación de lo prometedor de un nodo) y coste real (g) (desde el inicial hasta él) combinados, como lo hace por ejemplo A*:

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

- Búsqueda de caminos (continuación)
  - Se use la técnica que se use, en videojuegos ésta debe:
    - Ejecutarse con gran rapidez
    - Reiteradamente
    - Tener en cuenta el modelo del agente: puede limitar en qué cosas o cómo puede hacerlas para ser realista (no es lo mismo un ave, que un perro)
  - En perfiles de terrenos en 3D, el grafo se puede generar automáticamente, o estar pre-generado en el juego

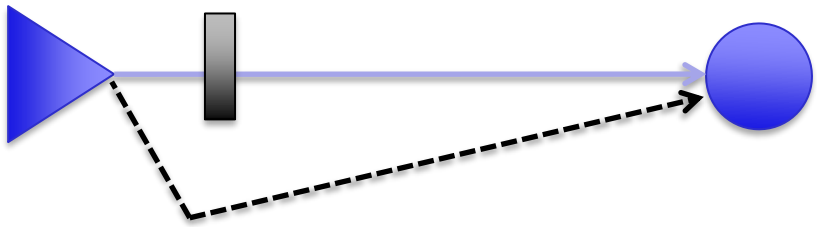
- Búsqueda de caminos (continuación)
  - Las áreas transitables (y no transitables) deben estar definidas: con frecuencia se crean **redes de polígonos** (***navmesh***) de modo que es transitable el adyacente en la red



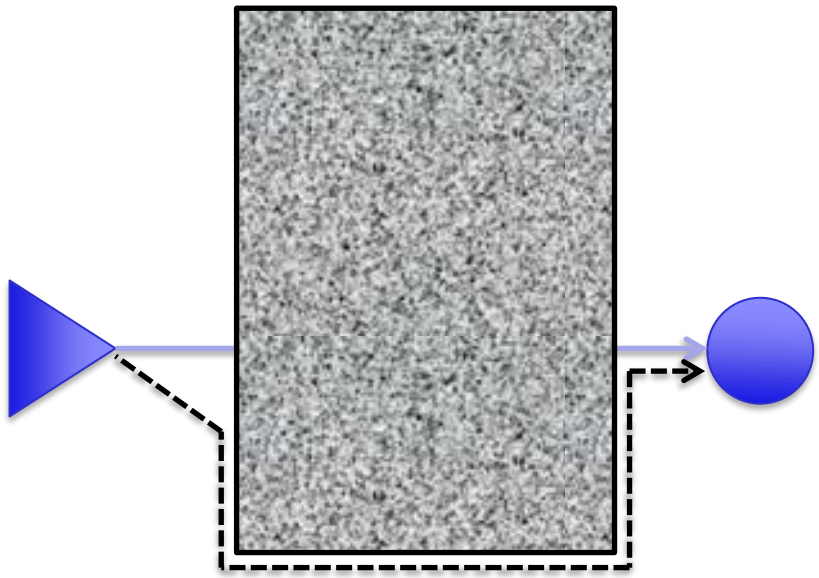


- Búsqueda de caminos (continuación)
  - Sin embargo, para aplicaciones sencillas, puede haber modelos o heurísticas que resulten menos complejos y más naturales que los clásicos citados anteriormente:
    - *Movimientos aleatorios para evitar obstáculos*
    - *Avanzar alrededor de los obstáculos*
    - *Método de las migas de pan*
    - *…*

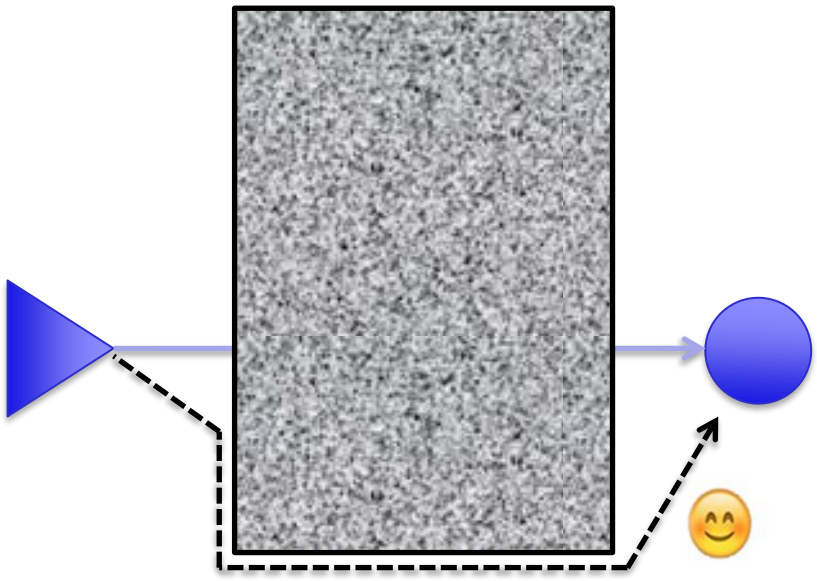
- Búsqueda de caminos (continuación)
  - *Movimientos aleatorios para evitar obstáculos:* Si los obstáculos son "pequeños" (árboles, piedras, etc.), se pueden esquivar realizando un cambio de dirección lateral aleatorio y temporal en el avance.



- Búsqueda de caminos (continuación)
  - *Avanzar alrededor de los obstáculos:* Si los obstáculos son “grandes” (casa, etc.), se pueden salvar siguiendo su frontera…

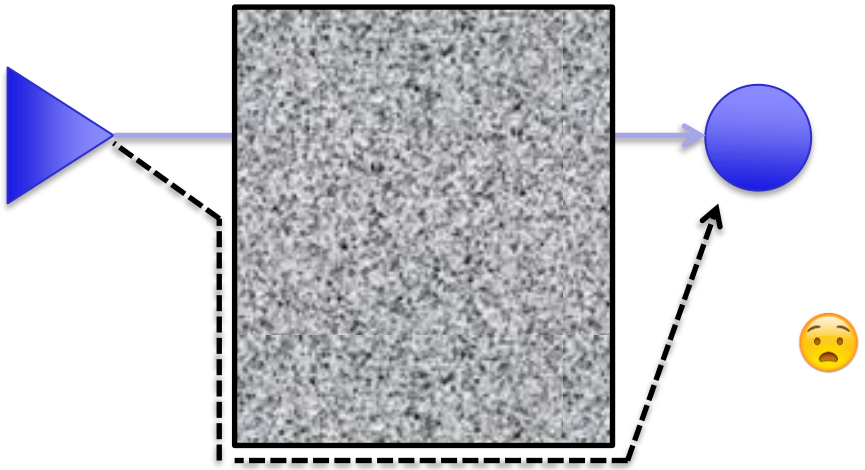


- Búsqueda de caminos (continuación)
  - *Avanzar alrededor de los obstáculos:* Si los obstáculos son “grandes” (casa, etc.), se pueden salvar siguiendo su frontera hasta que sea alcanzable de nuevo el destino sin el obstáculo.



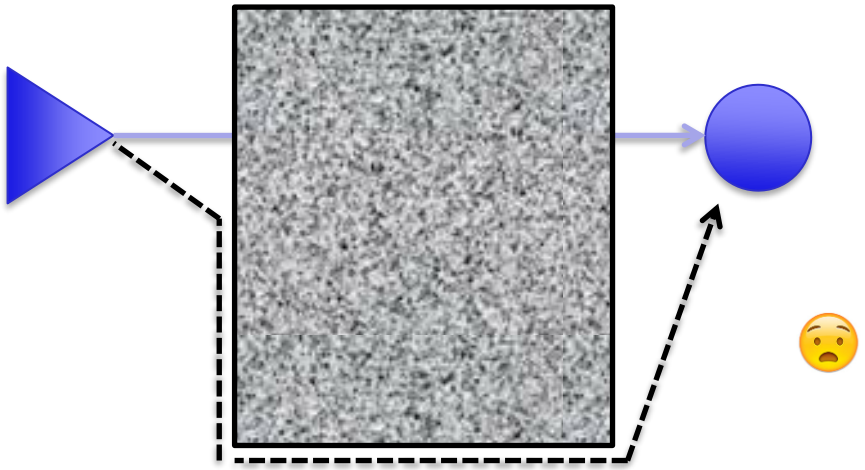
- Búsqueda de caminos (continuación)
  - *Avanzar alrededor de los obstáculos:* Si los obstáculos son “grandes” (casa, etc.), se pueden salvar siguiendo su frontera hasta que sea alcanzable de nuevo el destino sin el obstáculo.

… la dirección elegida para el seguimiento del borde del obstáculo puede tener relevancia ...



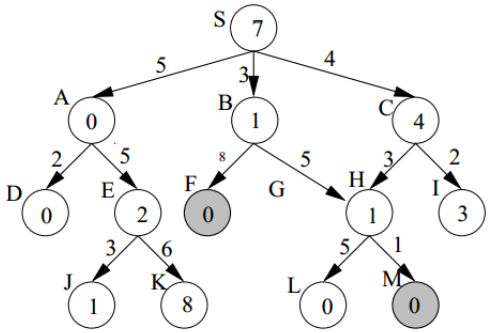
- Búsqueda de caminos (continuación)
  - *Avanzar alrededor de los obstáculos:* Si los obstáculos son “grandes” (casa, etc.), se pueden salvar siguiendo su frontera hasta que sea alcanzable de nuevo el destino sin el obstáculo.

Varias soluciones, Por ej.: Se podría optar por medir la distancia total recorrida con cada solución



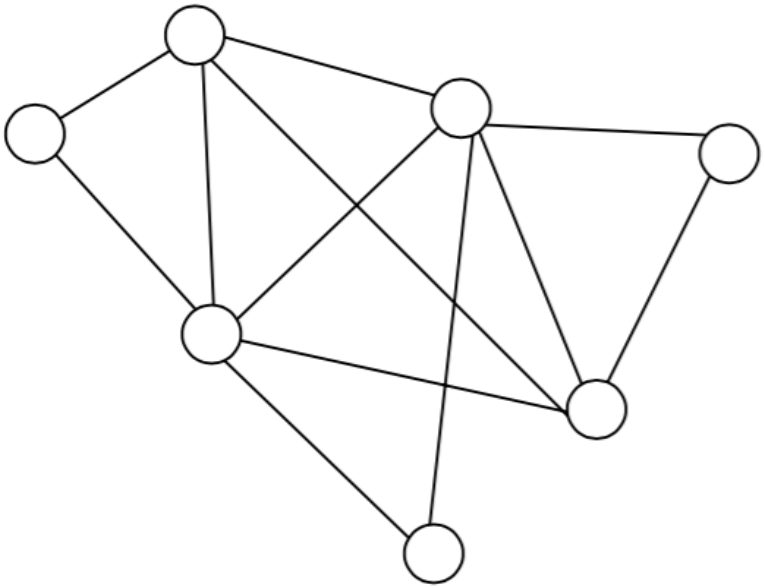
- Búsqueda de caminos (continuación)
  - *Método de las migas de pan (breadcrumb pathfinding):* Consiste en que el jugador nos de la información de cómo recorrer el terreno y evitar los obstáculos, es decir, almacenaremos sus caminos para poder seguirle o trazar en un futuro (otras partidas) caminos que él ya ha realizado.
    - Ahorra muchos cálculos.
    - Obtiene rutas inteligentes incluso para grupos.
    - Si no tenemos el camino, podemos esperar con movimientos locales o aleatorios a que pase por primera vez el jugador y nos deje las *migas de pan*.
    - Las coordenadas de los caminos se pueden almacenar en una estructura de datos dinámica que se vuelca a disco tras la partida

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado:
    - Cuando se trata de establecer caminos con cierta complejidad (muchos obstáculos, diversas habitaciones, etc), se recurre a otro tipo de algoritmos.
    - El más usado es A* *(Dijkstra lo es menos, aunque es más simple)*
      - Relativamente sencillo
      - Razonablemente eficiente: Ojo si son muchos a la vez
      - Encuentra el mejor camino entre dos puntos



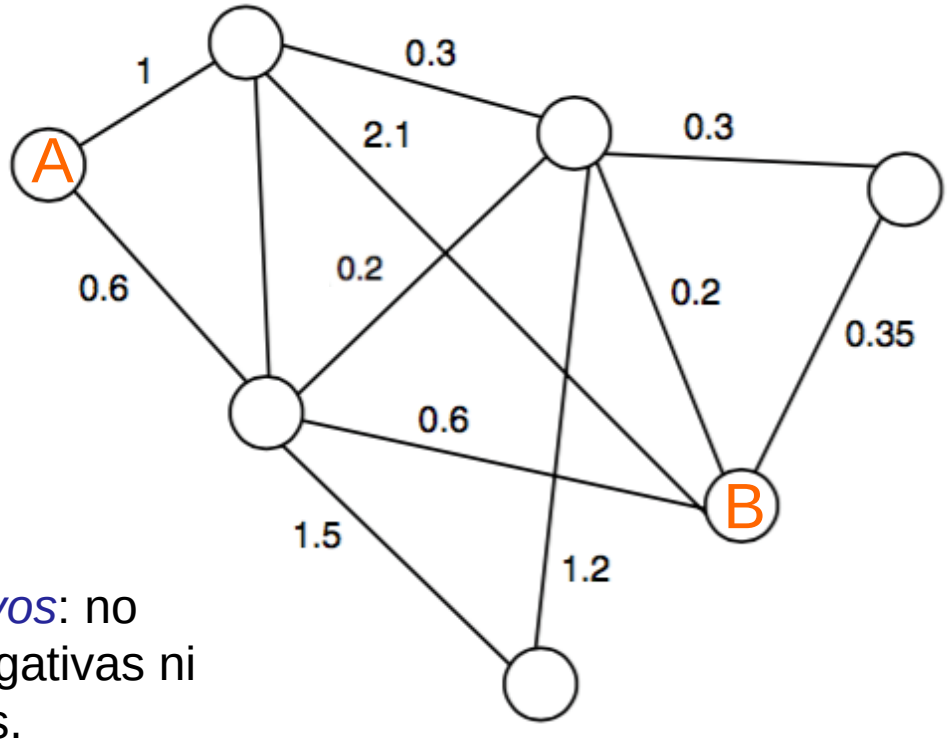
- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Para utilizarlo es necesario definir el área de búsqueda: establecer nodos en la escena
    - Escenarios muy abiertos, con infinidad de lugares alcanzables por el personaje no son apropiados para A*, salvo que reduzcamos los mismos, y establezcamos otros métodos (sencillos) para llegar a esos nodos. Ejemplo:
      - En un lugar con muchos pasillos y habitaciones, se establecerían los nodos de cada habitación dentro de cada una y cercano a la puerta, y en los pasillos.

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Estos nodos son el primer paso para crear un *grafo dirigido no negativo con pesos*
    - Los arcos entre los nodos por tanto representan que es posible alcanzar esos lugares: son contiguos o están conectados



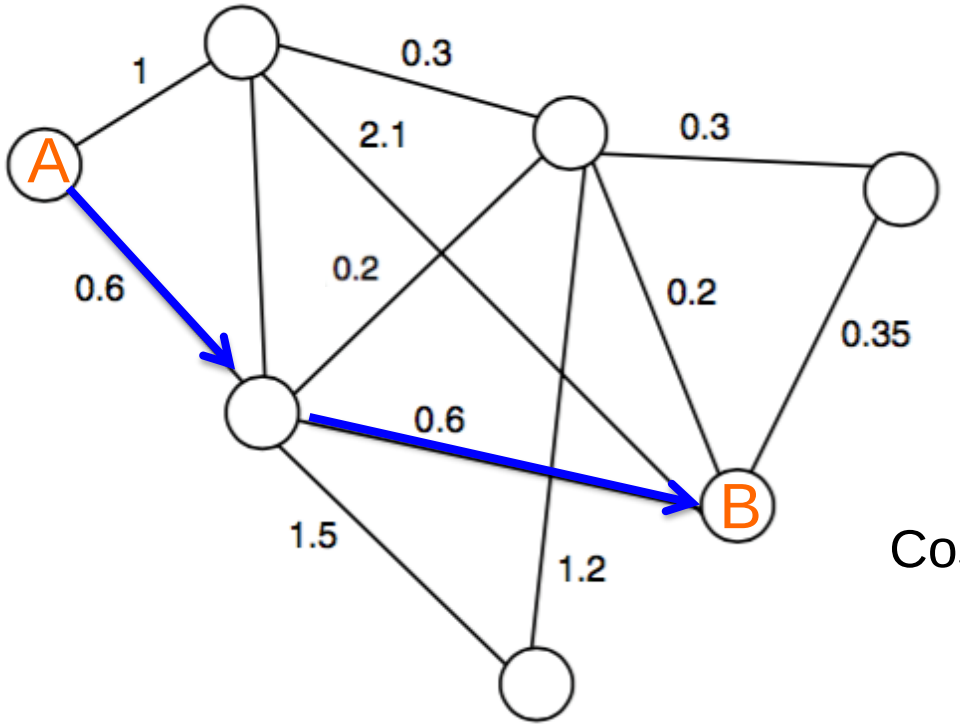
- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Se añaden pesos al grafo si es necesario (es lo habitual) emplear costes para esos desplazamientos, es decir, distancia por ejemplo, (*o acaso tiempo necesario para hacer ese desplazamiento (quizás dos habitaciones contiguas están muy próximas, pero sus puertas están alejadas y se tardaría mucho tiempo en llegar de una a la otra)*).
    - Es posible combinar (sumar o agregar) en los pesos también distintos conceptos, que lleven a un coste total a tener en cuenta.

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - El coste total de ir de uno a otro, sería la suma de costes:



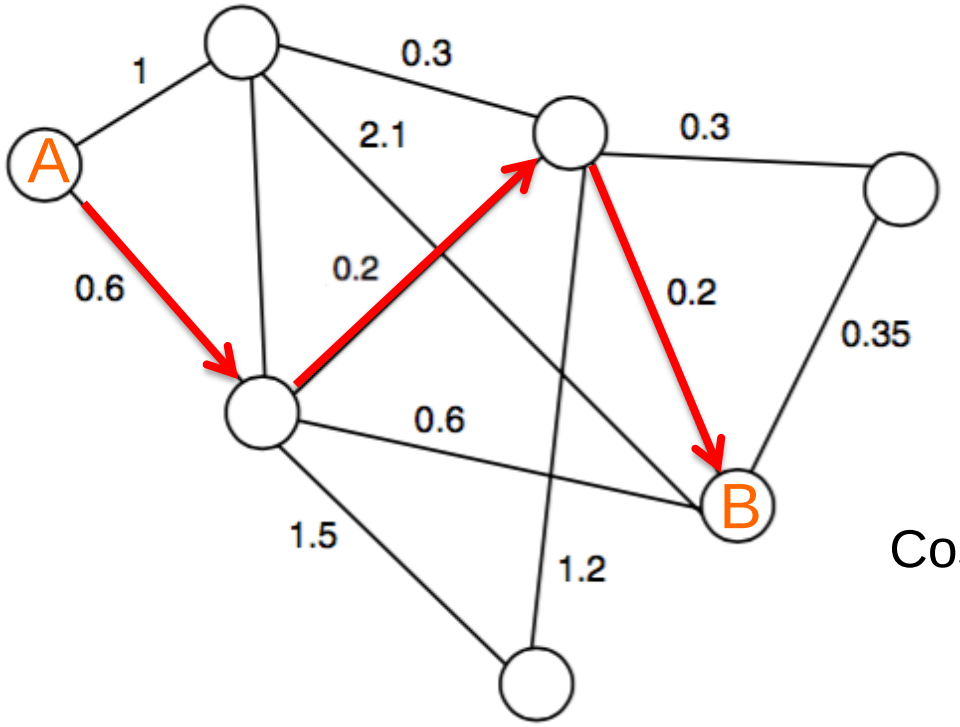
Costes *no negativos*: no hay distancias negativas ni tiempos negativos.

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - El coste total de ir de uno a otro, sería la suma de costes:



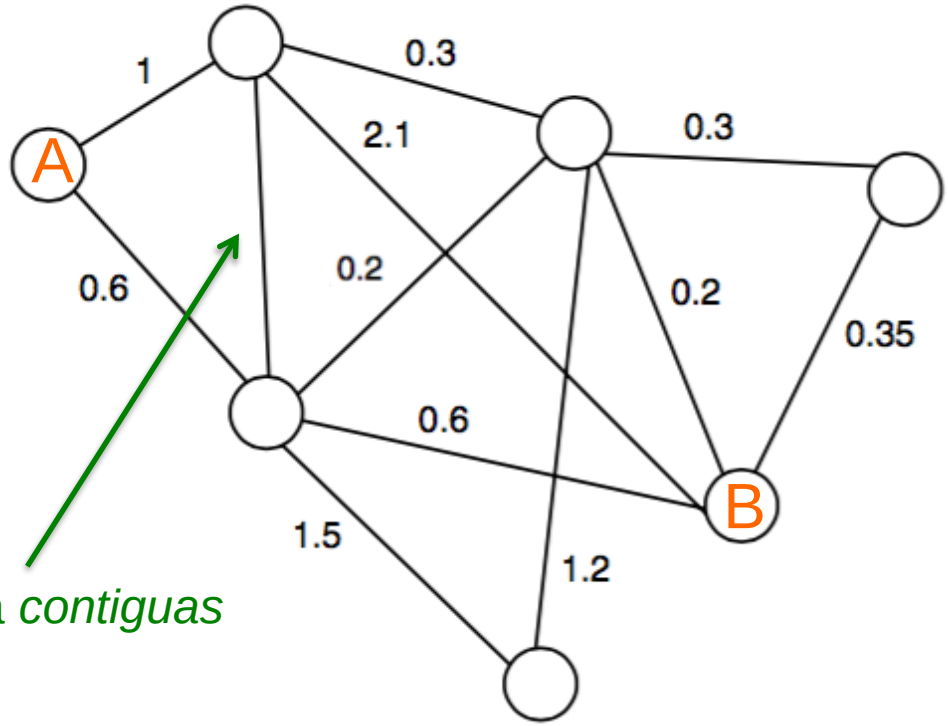
Coste 0.6+0.6 = 1.2

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - El coste total de ir de uno a otro, sería la suma de costes:



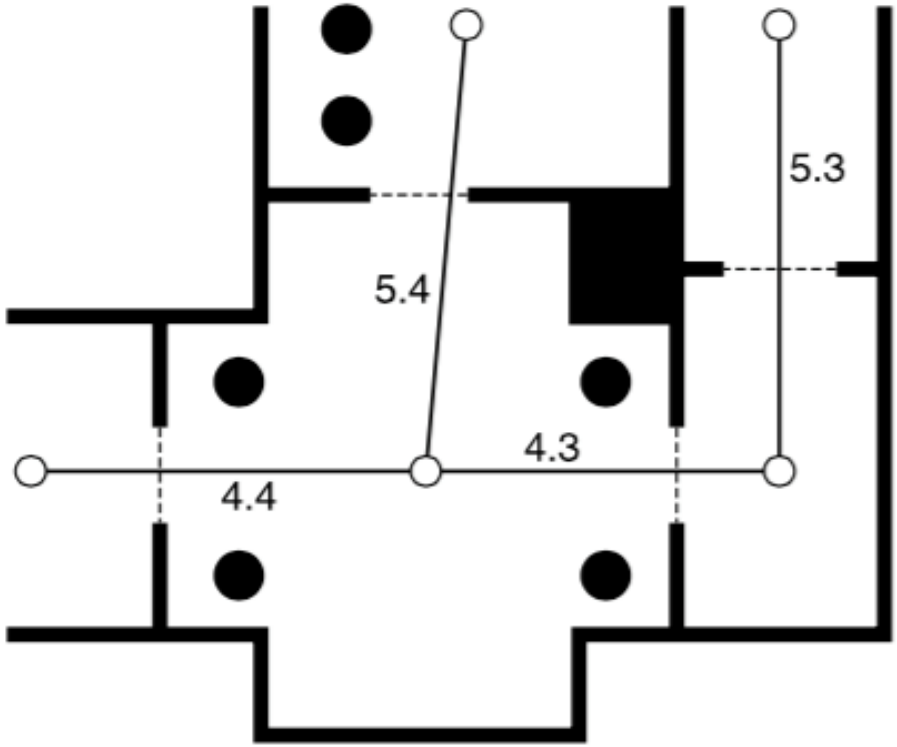
Coste 0.6+0.2+0.2= 1

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - El coste total de ir de uno a otro, sería la suma de costes:



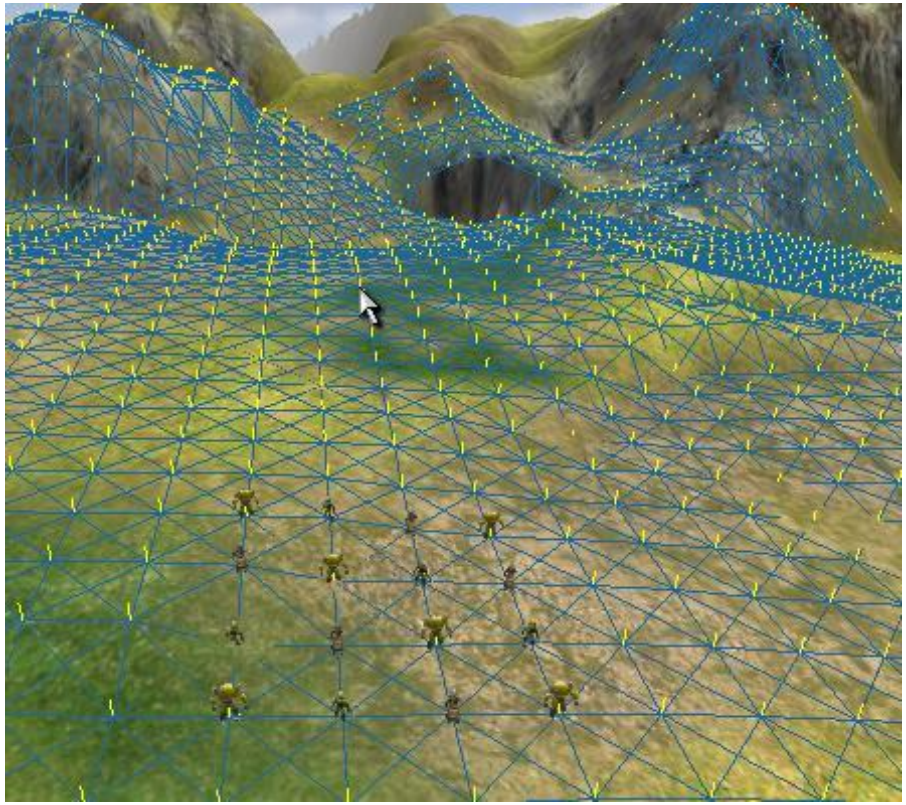
Coste ***0*** significa *contiguas*

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Ejemplo de un grafo superpuesto con el terreno:

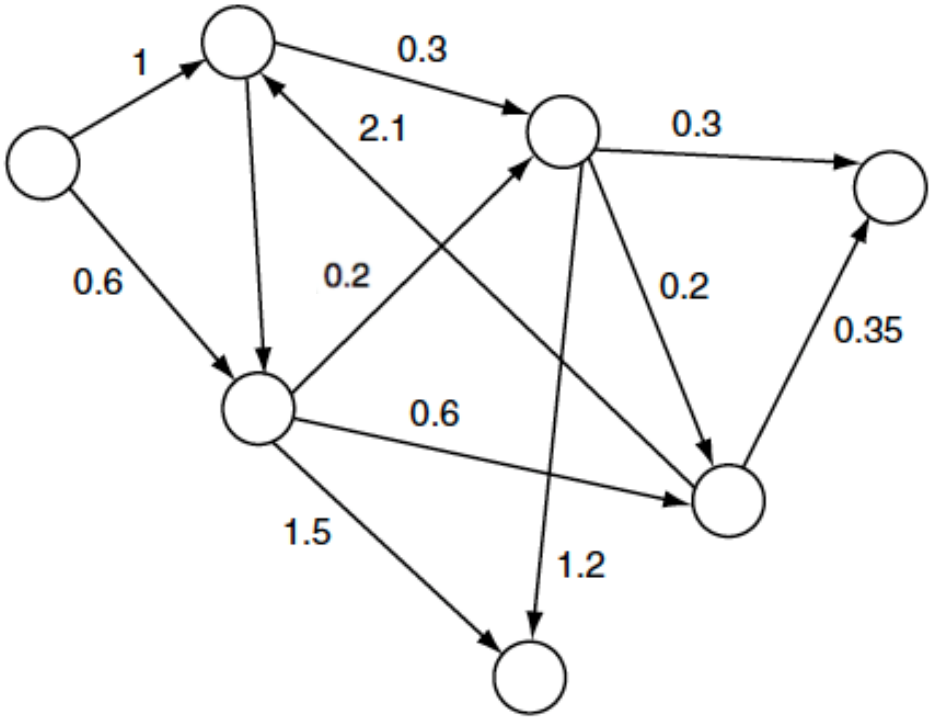


- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Ejemplo de un grafo superpuesto con el terreno:

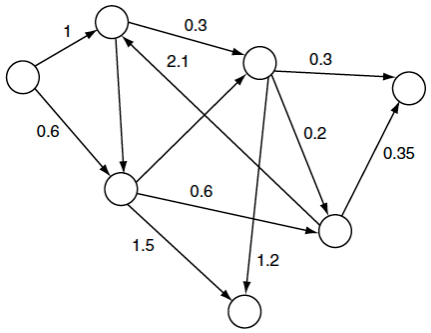
- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Ejemplo de un grafo superpuesto con el terreno:



- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Grafos Dirigidos:



- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Grafos Dirigidos:
      - Los grafos **NO dirigidos** se usan cuando:
        - » Desde un nodo A, es posible alcanzar un nodo B, y desde un nodo B es posible alcanzar un nodo A, es decir, ida y vuelta, y al mismo coste.
      - Sin embargo, también existe interés en los **dirigidos** en situaciones como estas:
        - » Cuando el personaje puede ir de A a B pero no de B hasta A:
          - » Es el caso típico por ejemplo de un descenso/caída semicontrolada que no admite escalar/subir de vuelta.
        - » Cuando el coste de ir de A hasta B es diferente de ir a B hasta A:
          - » Es el caso de subir o bajar una escalera, pues subir tiene mayor costo que bajar.



- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Estructuras de datos:
      - Es una cuestión vinculada a la implementación de A* que tengamos.
      - Genéricamente, conceptualmente podríamos decir que se trata de una tabla de *conexiones*, y en cada conexión se almacena el coste (entero o real) de la misma. (Puede haber conexiones sin coste: 0; y también puntos no conectados: se usa número muy alto determinado para indicar que no hay conexión).
        - » La tabla será simétrica (uso de una sola diagonal) si no es dirigido, y no simétrica en caso contrario.
        - » Antes de lanzar el algoritmo A*, hay que tener la estructura de datos con los costes ya cargados: precálculo.

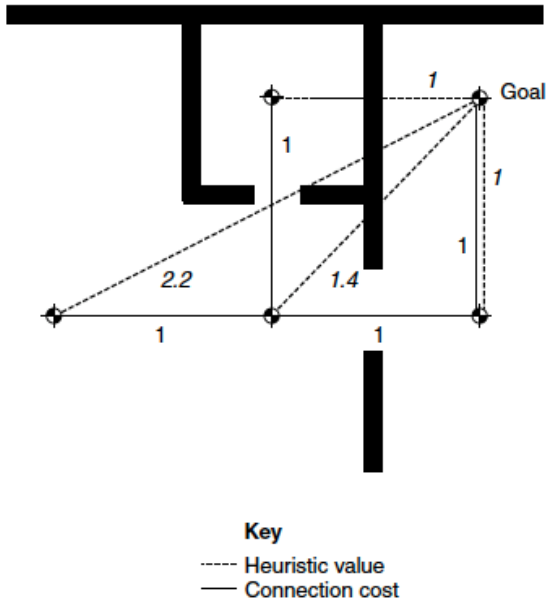
- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Estructuras de datos:
      - Sería posible utilizar implementaciones que no tuviesen precalculados los costes sino que los computasen cuando son necesarios:
        - » P.e.: En situaciones en los que los costes pudiesen depender de variables relacionadas con el estado del juego, etc.
      - Para un número de nodos elevado, la estructura de tabla debería plantearse hacerla dinámica para ahorrar en consumo de memoria.

    - El Algoritmo A* no se estudiará en esta asignatura, por haber sido objeto de amplio estudio previo en la titulación.

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Eficiencia y Eficacia de A*:
      - A* encuentra el mejor camino bajo ciertas condiciones.
      - La elección de la implementación será determinante.
      - La función heurística es clave: una función que devuelva exactamente la mínima distancia entre dos nodos dirigirá al algoritmo A* directo a la solución correcta con eficiencia ***O(p)*** siendo *p* el número de pasos del camino.
      - Sin embargo, encontrar la función heurística perfecta es en sí mismo, el problema del *pathfinding.*
      - Por tanto, en realidad, en función de si es optimista o pesimista, el algoritmo se puede comportar de formas ligeramente diferentes.

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Eficiencia y Eficacia de A*:
      - Comportamiento en función de la heurística:
        - » Si la heurística ofrece menor valor que el real (subestima), el algoritmo explorará más nodos por la zona próxima (se parecerá más a Dijkstra). Puede encontrar el mejor camino.
        - » Si la heurística ofrece mayor valor que el real (sobreestima), el algoritmo A*, tenderá a darnos caminos con pocos nodos intermedios aunque su valor global sea mayor. Puede **NO** encontrar el menor camino.
      - **Conclusión:** Por tanto, *si queremos el mejor camino será preferible una heurística que subestime ligeramente a una que sobreestime* *(mejor que explore de más a que lo haga de menos).*

- Búsqueda de caminos (continuación)
  - Pathfinding avanzado: Algoritmo A*
    - Eficiencia y Eficacia de A*:
      - Ejemplos de funciones heurísticas:
        - » La distancia Euclídea:
          - » Nos garantiza que es una subestimación

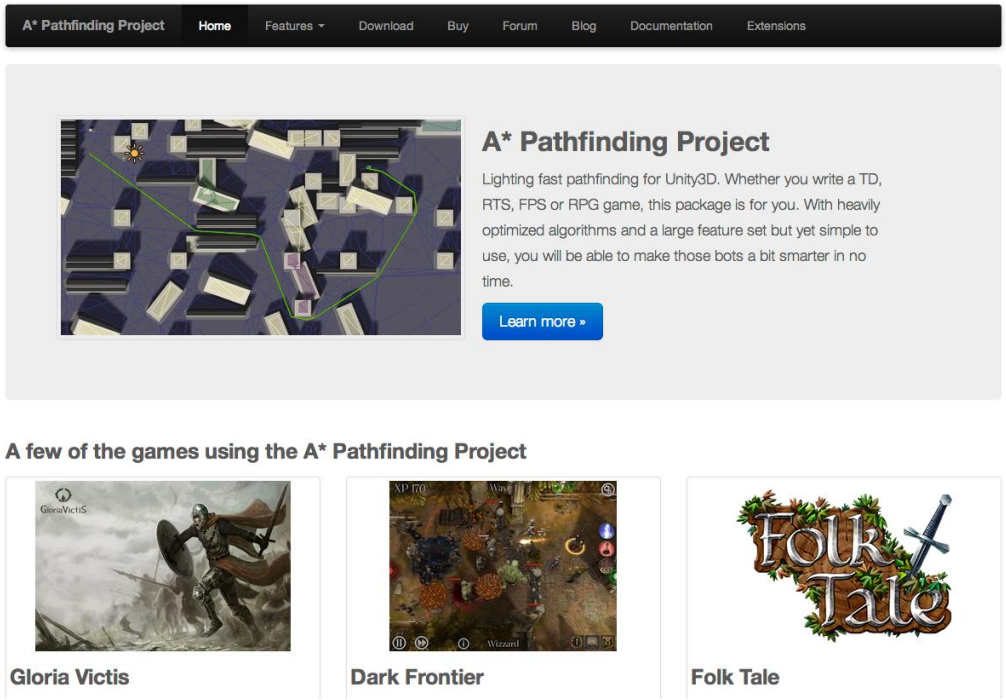


Bastante típica en juegos FPS

- Búsqueda de caminos (continuación)
  - Comparativa: Dijkstra vs A* vs Bi-Directional BFS (anchura)

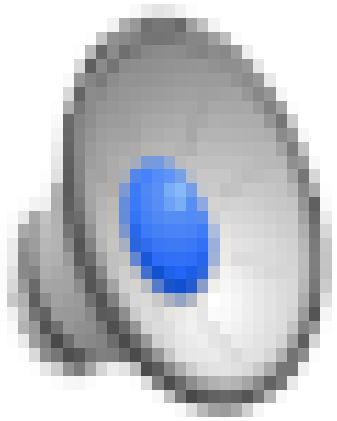
Este video muestra dinámicamente los nodos explorados para encontrar el camino, por diferentes algoritmos clásicos y variantes (funciones heurísticas en el caso del A*) de ellos.

- Búsqueda de caminos (continuación)
  - Relevancia y actualidad del problema:
    - El *pathfinding* en Unity
      - http://arongranberg.com/astar/

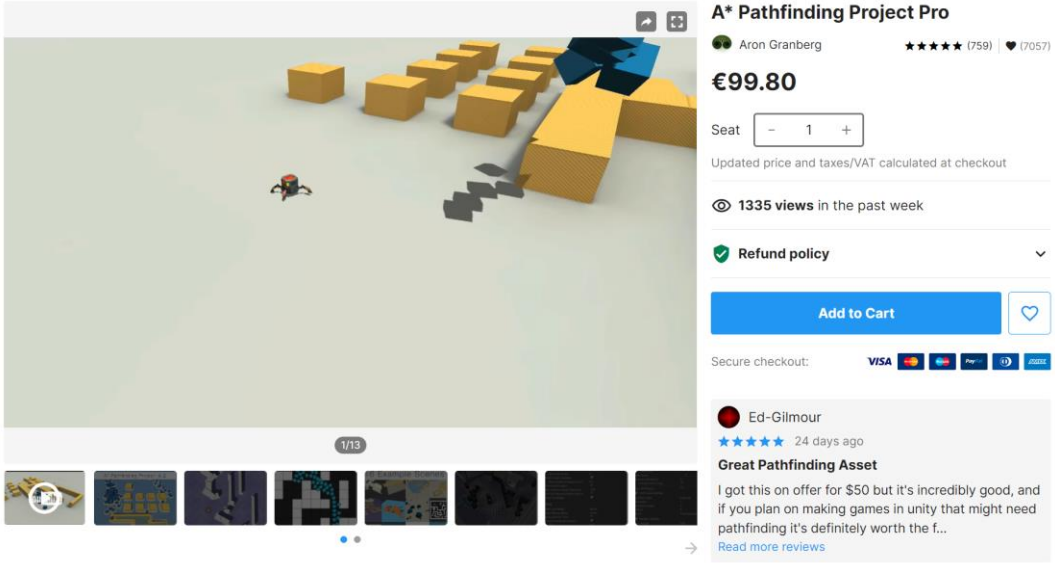


Se descarga el paquete desde *downloads* y se importa desde Unity 3D para utilizarlo

- Búsqueda de caminos (continuación)
  - Relevancia y actualidad del problema:
    - El *pathfinding* en Unity

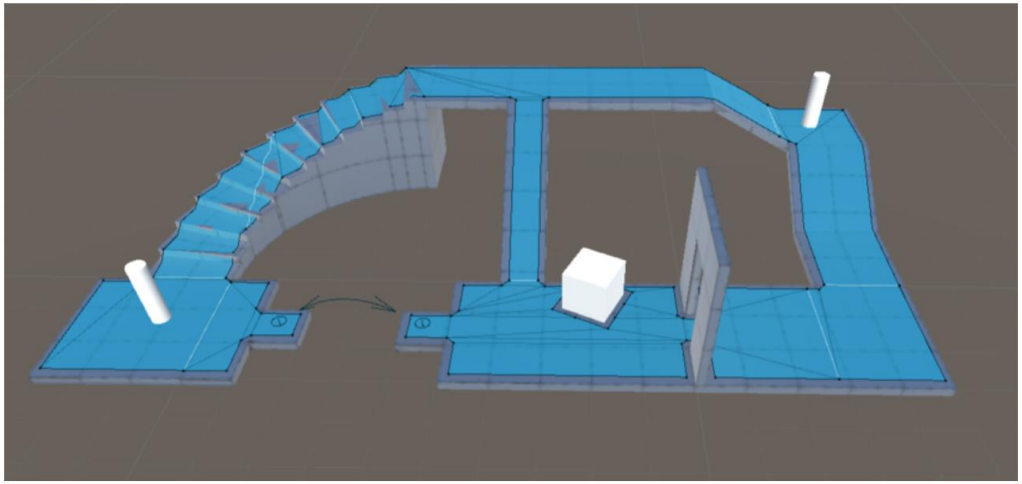


- Búsqueda de caminos (continuación)
  - Relevancia y actualidad del problema:
    - El *pathfinding* en Unity
      - https://assetstore.unity.com/packages/tools/ai/a-pathfinding-project-pro-87744



- Búsqueda de caminos (continuación)
  - Relevancia y actualidad del problema:
    - El *pathfinding* en Unity
      - https://docs.unity3d.com/Manual/Navigation.html

Navigation and Pathfinding



The navigation system allows you to create characters that can intelligently move around the game world, using navigation meshes that are created automatically from your **Scene** geometry. Dynamic obstacles allow you to alter the navigation of the characters at runtime, while OffMesh links let you build specific actions like opening doors or jumping down from a ledge. This section describes Unity's navigation and pathfinding systems in detail.

- Actividad presencial entregable:

Pathfinding en Unity

¿Cómo usaría PathFinding en su Videojuego?

Índice: