

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERÍA



**IMPLEMENTACION DE ALGORITMOS DE APRENDIZAJES DE
CAMINOS Y OPTIMIZACION EN TIEMPO DE RECORRIDO**

TRABAJO DE GRADUACIÓN PARA OPTAR AL TITULO DE
INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTADO POR:

EVER ABDUL FLORES SANDOVAL
CRUZ ANTONIO GALDAMEZ RIVERA

SOYAPANGO, EL SALVADOR, 2004

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

ESCUELA DE COMPUTACIÓN

**IMPLEMENTACION DE ALGORITMOS DE APRENDIZAJES DE
CAMINOS Y OPTIMIZACION EN TIEMPO DE RECORRIDO**

ING. EDUARDO RIVERA

Jurado 1

ING. JUAN CARLOS ZARATE

Jurado 2

ING. OSCAR DURAN VIZCARRA

Jurado 3

ING. JUAN CARLOS CRUZ

Asesor

UNIVERSIDAD DON BOSCO

RECTOR:
ING. FEDERICO MIGUEL HUGUET RIVERA

VICERECTOR ACADEMICO
PRESBITERO VICTOR BERMUDEZ YANES

SECRETARIO GENERAL:
LIC. MARIO RAFAEL OLMOS

DECANO FACULTAD DE INGENIERIA
ING. CARLOS GUILLERMO BRAN

AGRADECIMIENTOS

A ti *DIOS* te doy las gracias por darme la vida y las fuerzas necesarias para salir adelante en todo este tiempo y estar siempre a mi lado.

Por todo el amor que mis PADRES, HERMANA y demás familia me han brindado, los constantes consejos, el apoyo incondicional, la preocupación y el sacrificio que tuvieron para educarme y poder llegar donde me encuentro.

A mis AMIGOS que siempre estuvieron ahí, brindándome su amistad y sus consejos.

Ever Abdul Flores Sandoval

AGRADECIMIENTOS

Únicamente a DIOS.

Por bendecirme con la vida y con muchas personas que han sido de gran apoyo desde que estoy en este mundo, como lo son unos padres y una familia maravillosa, amigos que siempre pusieron sus manos en mi espalda, no para hacerme caer sino para impulsarme en salir a delante cuando me estaba deteniendo.

Cruz Antonio Galdámez Rivera

AGRADECIMIENTOS GRUPALES

Quisiéramos agradecer a muchas personas, pero no alcanzaríamos a mencionarlas todas:

A los evaluadores, Ing. Vizcarra, Ing. Rivera, Ing. Zárate, por concedernos su tiempo para poder realizar las defensas.

Al tutor, Lic. Coto, por la coordinación de los jurados y de local ya que esto no era nada fácil.

Al asesor, Ing. Juan Carlos Cruz por apoyarnos y brindarnos no solamente las horas de asesoría, sino su amistad.

A Paola de Cruz, por recibirnos en su casa, brindarnos hospitalidad y sobre todo su amistad, apoyo y dirección de exposición.

A Lic. Godoy, por su apoyo incondicional en redacción y verificación de estructura del documento así como también su amistad y apoyo.

A Jenniffer Martínez, por su dirección en el desarrollo de la estructura global del documento, por igual su amistad y apoyo.

INDICE

Introducción.....	II
Objetivos.....	1
Objetivo General	1
Objetivos Específicos	1
1. INFORMACION GENERAL	2
1.1 Inteligencia Artificial.....	2
1.2 Agentes Automatas	8
1.3 Búsqueda de Rutas y Memorización.....	13
2. ALGORITMOS DE INTELIGENCIA ARTIFICIAL.....	14
2.1 Algoritmo A*	14
Nodos	17
Pseudo-Código.....	17
Retornando Rutas Parciales.....	19
2.2 Algoritmo D*	22
Descripción del Algoritmo	24
Pseudo-Código.....	25
Pseudo-Código Optimizado.....	27
2.3 Aprendizaje aplicado Agente Automatas.....	28
Pseudo-Código.....	29
2.4 Áreas de Aplicación.....	30
3. COMPARACION ALGORITMO A* Y D* CON APRENDIZAJE.....	32
GLOSARIO DE TERMINOS	38
CONCLUSIONES	39
RECOMENDACIONES.....	41
ANEXOS.....	42
Manual de Usuario: Algoritmos IA.....	42
Manual Programador: Algoritmos IA.....	64
REFERENCIAS	78
INDICE HONOMÁSTICO.....	80

Introducción

La Inteligencia Artificial es una ciencia nueva que ha ido evolucionando a través del tiempo en diversas áreas, como es el desarrollo de algoritmos para la búsqueda de rutas que ayudan a desplazar un agente autómatas desde un punto inicial a un punto final.

En el presente documento se ha elaborado una recopilación de información general para comprender que es Inteligencia Artificial y sus áreas de aplicación como la robótica y los juegos.

Posteriormente, se presenta el algoritmo A*, conocido para encontrar la ruta más óptima de desplazamiento entre dos puntos. Dicho algoritmo hace uso de la planeación antes de iniciar su movimiento. Por el contrario, el algoritmo D* difiere del algoritmo A* al realizar la planeación a medida que sigue su trayecto, evitando obstáculos que han sido agregados después de iniciar su simulación. También se le ha agregado al algoritmo D* la facultad de ir aprendiendo cual es la ruta optima que debe seguir en caso de activar la simulación nuevamente y sus puntos de inicio y fin sean los mismos.

Finalmente, se ha incluido una manual de uso que facilitará la comprensión de la aplicación realizada.

Objetivos

Objetivo General

Explicar los conceptos generales de inteligencia artificial para comprender cual es su función en los algoritmos de búsqueda de rutas.

Objetivos Específicos

- Explicar los conceptos generales de inteligencia artificial.
- Comprender la definición y pseudo-código del algoritmo A*.
- Definir el concepto de aprendizaje y los distintos tipos.
- Describir y definir el algoritmo D* así como su pseudo-código.
- Mostrar las áreas de aplicación de los algoritmos de búsqueda de rutas.

1. INFORMACION GENERAL

1.1 Inteligencia Artificial

La Inteligencia Artificial (IA) es una de las disciplinas más nuevas. Nació en 1943 cuando Warren McCulloch y Walter Pitts propusieron un modelo de neurona del cerebro humano y animal. Formalmente se inicia en 1956 cuando se acuñó este término, aunque el estudio de la inteligencia contemplada como el razonamiento humano viene siendo estudiado por los filósofos hace más de 2 milenios.

Los primeros investigadores de esta innovadora ciencia, tomaron como base la neurona formalizada de McCulloch y postulaban que:

“El cerebro es un solucionador inteligente de problemas, de modo que imitemos al cerebro”.

Si se considera la enorme complejidad del mismo, esto era prácticamente imposible, debido también a varios aspectos como el hardware y el software de esa época, ya que no estaban a la altura para realizar semejantes proyectos.

En la Inteligencia Artificial hay conceptos diferentes, dependiendo del problema al cual se le busca solución, pero a grandes rasgos se mencionan los siguientes:

1. La concepción de IA como el intento de desarrollar una tecnología capaz de suministrar al ordenador capacidades de razonamiento o discernimiento similares, o aparentemente similares a las de la inteligencia humana.

2. La concepción de IA como investigación relativa a los mecanismos de inteligencia humana, que emplea el ordenador como herramienta de simulación para la validación de teorías.

El primer enfoque es por lo general el más práctico, se centra en los resultados obtenidos, en la utilidad, y no tanto en el método. En este enfoque se encuadran, por ejemplo, los Sistemas Expertos. Son temas claves en esta dirección la representación y la gestión del conocimiento. Los autores más representativos de este enfoque son McCarthy y Minsky.

El segundo enfoque está orientado a la creación de un sistema artificial que sea capaz de realizar los procesos cognitivos humanos. Desde este punto de vista no es tan importante la utilidad del sistema creado (qué hace), como lo es el método empleado (cómo lo hace). Como aspectos fundamentales de este enfoque se pueden señalar el aprendizaje y la adaptabilidad. Ambos presentan gran dificultad para ser incluidos en un sistema cognoscitivo artificial. Esta orientación es propia de Newell y Simon, de la Carnegie Mellon University.

Se presentan a continuación otros conceptos que pueden ser aplicados para comprender el término de Inteligencia Artificial:

- “La interesante tarea de lograr que las computadoras piensen... máquinas con mente, en su amplio sentido literal.” (Haugeland, 1985).
- “La automatización de actividades que se vinculan con procesos de pensamiento humano, actividades tales como la toma de decisiones, resolución de problemas, aprendizaje...” (Bellman, 1978).
- “El estudio de las facultades mentales mediante el uso de modelos computacionales.” (Charniak y McDermott, 1985).

- “El estudio de los cálculos que permiten, razonar y actuar.” (Winston, 1992).
- “Un campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales.” (Schalkoff, 1990).

Características de la Inteligencia Artificial

La inteligencia artificial presenta las siguientes características principales:

- a. Una característica fundamental que distingue a los métodos de Inteligencia Artificial de los métodos numéricos es el uso de símbolos no matemáticos, aunque no es suficiente para distinguirlo completamente. Otros tipos de programas como los compiladores y sistemas de bases de datos, también procesan símbolos y no se considera que usen técnicas de Inteligencia Artificial.
- b. El razonamiento basado en el conocimiento, implica que estos programas incorporan factores y relaciones del mundo real y del ámbito del conocimiento en que ellos operan.
- c. Dar solución a problemas con aplicabilidad de datos mal estructurados.
- d. Manipulación de información inexacta, incompleta o definida de una forma insuficiente.

La prueba de Turing (Alan Turing 1950) intenta ofrecer una definición de Inteligencia Artificial que se pueda evaluar.

Para que un ser o máquina se considere inteligente debe lograr engañar a un evaluador de que este ser o máquina se trata de un humano evaluando todas las actividades de tipo cognoscitivo que puede realizar el ser humano.

Si el diálogo que ocurre y el número de errores en la solución dada se acerca al número de errores ocurridos en la comunicación con un ser humano, se podrá estimar (según Turing) que estamos ante una máquina "inteligente".

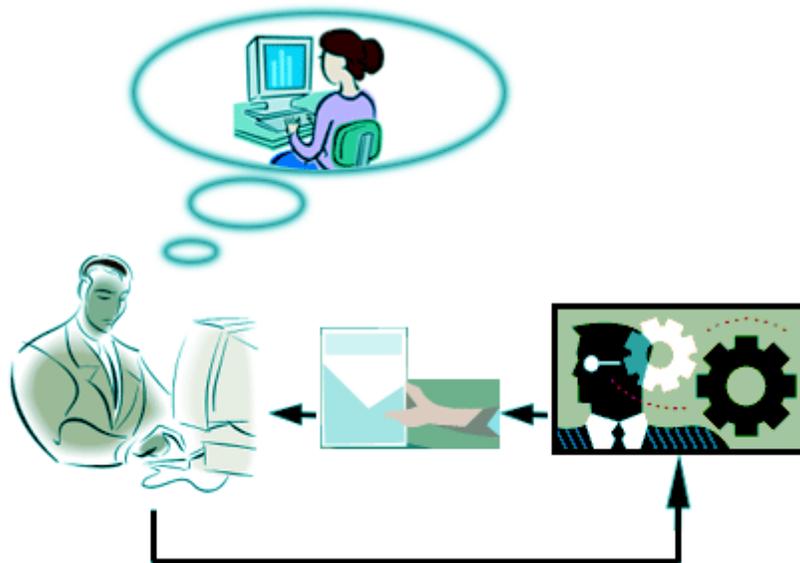


Fig. 1 Prueba de Turing

El trabajo que conlleva programar una computadora para pasar la prueba es considerable. La computadora debería ser capaz de lo siguiente:

- Procesar un *lenguaje natural*: para así poder establecer comunicación satisfactoria, sea en español, inglés o en cualquier otro idioma humano.

- Representar *el conocimiento*: para guardar toda la información que se le haya dado antes o durante el interrogatorio. Utilización de Base de Datos para receptar preguntas y luego almacenarlas.
- Razonar *automáticamente*: Utiliza la información guardada al responder preguntas y obtener nuevas conclusiones o tomar decisiones.
- *Aprender*: Con el propósito de adaptarse a nuevas circunstancias. El autoaprendizaje conlleva a la autoevaluación.

La prueba de Turing¹ puede ser realizada por la diversificación de procesos inteligentes que el computador puede realizar, analizar y la manera de responder o incluso recopilar información anteriormente dada para una reformulación (vista) y la interacción con la persona (robótica).

Técnicas de búsqueda de soluciones aplicando IA

Un aspecto importante de la hipótesis del sistema simbólico propuesto por Newell y Simon, es que los problemas resueltos por medio de la búsqueda entre varias alternativas, se basan en la aplicación del sentido común humano. Los humanos generalmente consideran un número de estrategias alternas que las guíen a la solución de problemas. De este modo, se han establecido diferentes alternativas o cursos de acción que conduzcan a la solución en dependencia de las características del espacio de estados del problema a resolver.

El espacio de estados se define como la representación de un problema o situación que abarca todas las posibles situaciones que se pueden presentar

¹ Ref. <http://www.loebner.net/Prizet/TuringArticle.html> y <http://www.turing.org.uk>

en la solución del mismo así como las relaciones que existen entre ellas. Está formado de nodos que describen situaciones particulares del problema y arcos que conectan pares de nodos y representan los movimientos legales o reglas que rigen el espacio de estados; ellos determinan si es posible pasar de una situación del problema a otra (Luger y Stubblefield, 1989).

De esta forma, la solución al problema se establece como un algoritmo de búsqueda que analiza los nodos del espacio de estados y se representa por el conjunto definido de la siguiente forma $[N, A, I, D]$ (Luger y Stubblefield, 1989) donde:

- N es el conjunto de nodos del espacio de estados. Estos corresponden a los estados en el proceso de solución del problema.
- A es el conjunto de arcos o ligas entre nodos. Corresponden a los pasos en el proceso de solución del problema.
- I es un subconjunto no vacío de N que contiene el o los estados iniciales del problema.
- D es un subconjunto no vacío de N que contiene el o los estados finales o la solución al problema, los cuales pueden ser obtenidos usando una propiedad medible de los estados encontrados durante la búsqueda o una propiedad de la ruta recorrida durante la búsqueda.

La función de un algoritmo de búsqueda es encontrar una trayectoria que conduzca a una solución del problema por medio del espacio de estados.

Los humanos no solo usan la búsqueda exhaustiva, es decir, también resuelven los problemas basados en la aplicación de reglas de juicio que guíen la búsqueda por aquellas porciones del espacio de estados que

parezcan “prometedoras”. Estas reglas son conocidas como heurísticas. Una heurística es una estrategia de búsqueda selectiva en el espacio de un problema y guía la búsqueda a lo largo de las líneas que tienen una alta probabilidad de éxito mientras que descartan aquellas trayectorias que no la ofrecen (Luger y Stubblefield, 1989).

Las heurísticas no son infalibles, ya que no siempre garantizan una solución óptima al problema, pero una buena heurística puede y debe aproximarse lo más que se pueda la mayoría de las veces a ella. Lo más importante es que emplea conocimiento relacionado con la naturaleza del problema para encontrar una solución de manera eficiente.

1.2 Agentes Automatas

Definición

Es todo aquello que se considera puede percibir el ambiente en el cual se desenvuelve, que responde, que actúa y en el cual puede ser capaz de autoevaluarse, aprender por si mismo, y cambiar su acción de acuerdo al ambiente en el cual se encuentra interactuando.

Los agentes autónomos han nacido de diferentes disciplinas y sus principales contribuidores son:

- Inteligencia Artificial.
- Programación Orientada en Objetos y Sistemas Concurrentes basados en objetos.
- Diseños de interfaces computadoras-hombres.

Inteligencia Artificial

El mayor contribuidor sin duda para los agentes autómatas es la Inteligencia Artificial (IA), porque se ha enfocado últimamente en la construcción de artefactos inteligentes los cuales actúan y sienten el ambiente en el que se están desarrollando.

En 1970, la mayor parte de investigaciones de IA estaban dirigidas a la planeación de IA lo cual está relacionado directamente con los agentes autónomos porque estos necesitan saber que acción realizar, ya que estos son agentes que poseen un sistema que realiza diversas actividades en un ambiente. Típicamente este tipo de sistema debería tener los siguientes componentes:

- Un modelo simbólico del ambiente del agente.
- Especificaciones simbólicas de las acciones que el agente puede realizar en términos de PDA (Precondiciones, decisiones, agregar).
- Un algoritmo de planeación, el cual toma una representación del ambiente, una serie de acciones específicas a realizar, una representación de la meta a desplazarse las cuales utiliza para producir cual será el plan de acción a seguir para llegar a la meta.

En 1980 muchos investigadores cuestionaron el uso de un modelo símbolo debido a que esta forma de operar de los agentes autómatas no contemplaba su uso en escenarios reales los cuales podían variar de un momento a otro. Rodney Brooks quien fue el mayor crítico conocido del modelo simbólico presento varios informes (Comportamiento IA, IA reactivo, IA Situado²) en los

² Ref. <http://www.csc.liv.ac.uk/~mjw/pubs/jaamas98.pdf>

cuales decía que la inteligencia y el comportamiento no es un atributo que se podía probar a través de teoremas o sistemas expertos, sino que era un producto de la interacción del agente con el ambiente.

Como parte de su investigación Brooks creó una arquitectura asumida, una arquitectura que el agente controla en la cual no toma símbolos representativos, sino que estaba basada en el comportamiento al momento de realizar todas las tareas. Cada uno de los comportamientos es una máquina de estados finita que continuamente está percibiendo su ambiente para determinar cual es su respuesta a ese comportamiento. En algunas implementaciones se representa como “situación → acción”, el cual representa una acción a cada estado que puede poseer el agente.

Algunas desventajas que presenta esta arquitectura son:

- Si los agentes no emplean modelos de sus ambientes, deben poseer información suficiente disponible sobre sus ambientes locales para determinar a partir de esto la mejor acción.
- Debido a que la información de su ambiente la toman localmente no se puede determinar que haría el agente autómatas con información no local.
- Es difícil implementar el aprendizaje a través de la experiencia y mejorar su comportamiento en el tiempo.

En 1990 los investigadores encontraron que este modelo podía ser implementado en algunos campos y con ciertos problemas en otro tipo. A partir de esto comenzaron a desarrollar una arquitectura *híbrida* la cual contiene los mejores aspectos del modelo simbólico y de la arquitectura asumida, las cuales se encontraban a través de software de capas. Dichas capas pueden ser arreglos verticales (solamente una capa tiene acceso a los

sensores del agente) u horizontales (todas las capas tienen acceso a los sensores para que puedan dar o enviar una respuesta (acción) de salida. Ver Fig. 2

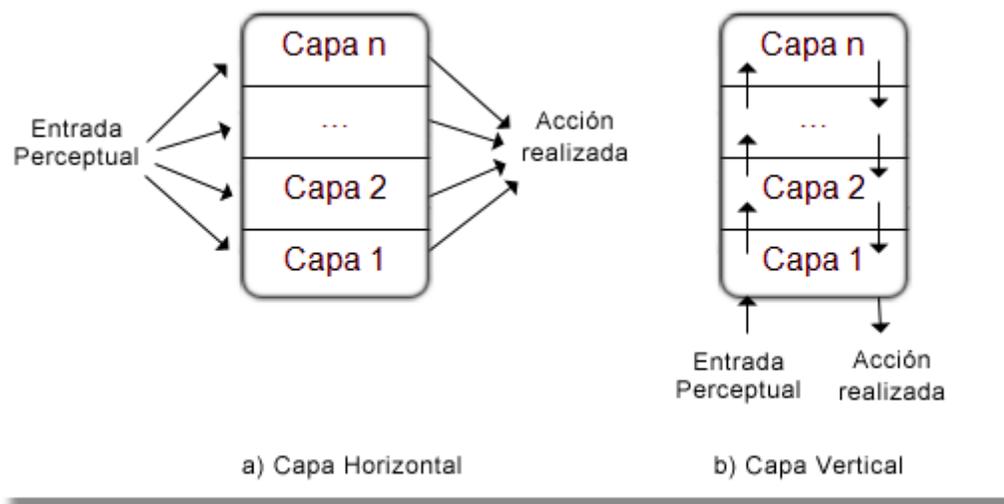


Fig. 2 Arquitectura de Capas Agente Autómata

En forma similar a la arquitectura de Brooks, se trabaja con capas las cuales son ordenadas en forma jerárquica, con los diferentes niveles trabajando con la información que adquieres del ambiente. A partir de esto, muchas nuevas arquitecturas encontraron que solo trabajar con 3 capas era suficiente, porque el nivel más bajo de la jerarquía es una capa reactiva, ya que decide que hacer a partir de la información que los sensores le envían. La capa del centro trabaja con la información que los sensores le envían para decidir en base al conocimiento que posee del ambiente en el cual el agente encuentra interactuando (modelo simbólico). El nivel superior trabaja con los niveles sociales del ambiente porque posee un nivel conocimiento social visto.

Una tradición final en el área de las arquitecturas de los agentes autónomos, es la práctica de agentes razonantes, los cuales se basan en el modelado o en la teoría del razonamiento humano; es decir, se usa un razonamiento pragmático para decidir que hacer.

Objetos y Sistemas Concurrentes de objetos.

La relación que guardan los agentes con los objetos, es que los objetos han sido definidos como entidades en la computación los cuales se encapsulan en estados, realizan acciones, métodos del estado, y se comunican enviando mensajes.

Mientras que en ambos (agentes y objetos) guardan similitudes, también se encuentra una diferencia y es la autonomía de cada uno de ellos, porque los objetos deciden que hacer en base a los objetos que los han invocado, mientras que los agentes deciden que hacer en base a las peticiones de los mismos agentes que los invocaron. Esta distinción puede entenderse de la siguiente forma: “Los objetos hacen algo de gratis, mientras que los agentes lo hacen por dinero”.

Otra diferencia importante entre objetos y agentes es la flexibilidad en su comportamiento autónomo (reactivo, preactivo, social).

Una diferencia más que puede ser encontrada es que cada agente posee su propia línea de control, mientras que en los objetos existe una sola línea la cual controla todo el sistema.

Interfaces Computadoras-Hombre

Se conocen como todas aquellas interfaces las cuales son capaces de tomar la iniciativa en realizar una actividad sin esperar una serie de instrucciones que le indiquen que hacer, conocidos como *asistentes expertos*.

Uno de los investigadores ha sido Nicholas Negroponte en su libro “Being Digital”, en el cual dice:

“El ‘agente’ contesta el teléfono, reconoce al que llama, te interrumpe cuando sea apropiado y hasta pueda inclusive decir una mentira blanca para ayudarte. El mismo agente se encuentra bien entrenado en buscar los mejores momentos con respecto e idiosincrasia.”³

“Si usted posee a alguien que lo conoce bien y con quien comparte la mayor parte de información, esta persona puede actuar en su lugar efectivamente. Por ejemplo, si su secretaria se enferma, no habría diferencia si le envían a Albert Einstein para cubrir a su secretaria, porque no interesa su IQ, sino el grado de conocimiento y el modo de usar ese conocimiento en ayudarlo a cumplir sus intereses lo que realmente importa”.⁴

1.3 Búsqueda de Rutas y Memorización

La Búsqueda de Rutas⁵ o algoritmos de rutas, son todos aquellos algoritmos de computadoras que encuentran cual es el camino desde un lugar hasta otro (usualmente llamado punto inicial hasta un punto final). La mayor parte de algoritmos, trabajan con cuadrículas de datos, llamadas mapas, los cuales poseen un valor definido de costo para cada nodo que conforma la cuadrícula. Los algoritmos buscan dentro del mapa en el que se están desplazando todos aquellos nodos que poseen un total de costo menor o distancia (lo hacen moviéndose en las cuatro direcciones cardinales, y en algunos casos, también en sus diagonales). Una de las características principales que tienen los algoritmos es que buscarán la mejor ruta para movilizarse y tratarán de usar la menor cantidad posible de recursos con los que cuentan.

³ Negroponete, N. Being Digital, Hodder and Stoughton, 1995, p. 150

⁴ Negroponete, N. op. cit. p. 151

⁵ El término búsqueda de rutas también es conocido como “Pathfinding”.

2. ALGORITMOS DE INTELIGENCIA ARTIFICIAL

2.1 Algoritmo A*

El algoritmo A*⁶, es un algoritmo el cual encontrará una ruta entre 2 puntos dentro de un mapa. Actualmente, existen muchos algoritmos los cuales se utilizan para encontrar rutas, pero el algoritmo A* encontrará la ruta más corta, si es que existe una, y lo hará de la forma relativamente más rápida.

El algoritmo A* es un algoritmo directo, el cual busca por la mejor ruta a explorar, y en algunas ocasiones revisa las rutas alternas para encontrar la mejor alternativa.

El algoritmo A* atraviesa un mapa creando nodos que corresponden a las posiciones que va explorando, los cuales se van guardando para llevar un registro de la búsqueda. Adicionalmente, a guardar la información del lugar donde se encuentra cada nodo, cada uno de los nodos posee tres atributos comúnmente llamados *f*, *g* y *h*, los cuales se refieren a los valores de *conveniencia*, *meta* y *heurística*, respectivamente. A partir de esto, se forma la siguiente ecuación:

$$f = g + h$$

Ec. 1

donde,

- o *g*: Es el costo para llegar desde el nodo inicial hasta ese nodo. Existen muchas rutas para llegar desde el nodo inicial hasta esa parte del mapa, pero ese nodo representa una sola ruta de cómo llegar.

⁶ El algoritmo A* es conocido también como A asterisco o Astar.

- h : Es un costo estimado para llegar desde ese nodo hasta la meta. La variable h hace referencia a la heurística, el cual es una forma de adivinar educadamente cual será el valor para llegar hasta la meta.

h puede ser estimado de distintas formas. Entre los métodos más utilizados se encuentran:

i. Método Manhattan:

Su principal ventaja es que normalmente encuentra el objetivo más rápido, aunque esto conlleva una mayor sobrecarga de h en comparación con g y por lo tanto, reducirá cualquier pequeño modificador que se desee añadir al cálculo.

La ecuación utilizada para la obtención de h es:

$$h = v * (\text{abs}(X_{\text{actual}} - X_{\text{destino}}) + \text{abs}(Y_{\text{actual}} - Y_{\text{destino}}))$$

Ec. 2

v , es el valor del costo el cual se recomienda sean valores enteros para evitar que sea menos lenta la búsqueda de la ruta.

ii. Atajo Diagonal:

Este método equilibra los valores de h y g , lo cual permite considerar pequeños modificadores como una penalización de giro o modificaciones en el mapa. Este método es un poco más lento que el método Manhattan.

$$XDistancia = \text{abs}(\text{abs}(X_{\text{actual}} - X_{\text{destino}}))$$

$$YDistancia = \text{abs}(\text{abs}(Y_{\text{actual}} - Y_{\text{destino}}))$$

SI $XDistancia > YDistancia$ ENTONCES

$$h = 14 * YDistancia + 10 * (XDistancia - YDistancia)$$

DE LO CONTRARIO

$$h = 14 * xDistancia + 10 * (yDistancia - xDistancia)$$

FIN

$$h = \max(\text{abs}(A.x - \text{goal}.x), \text{abs}(A.y - \text{goal}.y))$$

Ec. 3

- *f*: Es la suma de *g* y *h*. Representa la mejor aproximación de costo (adivinada) que pasa por un nodo dado. Mientras más bajo sea el valor de *f*, es mejor la ruta por ese nodo.

El propósito de *f*, *g* y *h* es de cuantificar que tan prometedora es una ruta a través de ese nodo.

El algoritmo A* utiliza dos listas, una lista Abierta y una lista Cerrada. La lista Abierta son todos aquellos nodos que aún no han sido explorados, por lo tanto, la lista cerrada consiste en todos aquellos nodos que ya han sido explorados.

Para que un nodo sea considerado como explorado, el algoritmo debe haber explorado cada uno de los nodos con los que se encuentra conectado, haber calculado sus valores de *f*, *g* y *h* y haberlos colocado en la lista Abierta para ser explorados posteriormente.

Las listas Abierta y Cerrada son necesarias porque los nodos no son únicos (por ejemplo, si se comienza en el nodo (0,0) y se mueve al nodo (0,1), es válido que este regrese al nodo (0,0)), y así se poseerá un registro de todos aquellos nodos que ya han sido explorados. Por tal razón, los nodos representan el estado y el progreso de la búsqueda realizada.

Nodos

Son las estructuras de datos que representan las posiciones dentro del mapa. Cada uno de los nodos posee información importante sobre su posición, si el nodo puede ser transitable, etc.

Tipos de Nodos:

- Imposibles de pasar: El valor que poseen hace imposible para el algoritmo encontrar una ruta a través de él.
- Difícil de pasar: El paso a través de este tipo de nodos es posible ya que poseen un costo accesible para el objeto móvil. En muchas ocasiones se requiere de tiempo por parte del objeto para que se pueda pasar a través de ellos.
- Dificultad intermedia de pasar: Se puede transitar a través de ellos y el costo que esto representa es mínimo, aunque siempre representan cierta dificultad.
- Fácil de pasar: Es posible pasar a través de ellos sin tener ningún tipo de dificultad.

Pseudo-Código

- a. Sea P = al punto de inicio.
- b. Asignar los valores de f , g y h a P .
- c. Agregar P a la lista abierta.
- d. Mientras la Lista Abierta no se encuentre vacía.

- i. Tomar el mejor nodo que se encuentra en la lista abierta. Este nodo es el que posee un menor costo.
- ii. Salir del proceso si ya se llegó al punto final.
- iii. Agregar a la lista abierta todos aquellos nodos adyacentes que no se encuentren en ninguna de las listas (abierta o cerrada) luego de haber calculado sus valores de f , g y h .
- iv. Borrar el “mejor” nodo de la lista abierta y agregarlo a la lista cerrada.
- e. Si la lista Abierta no se encuentra vacía.
 - i. Utilizar la lista cerrada desde el nodo de llegada para encontrar la ruta.
- f. Si la lista abierta esta vacía, no es posible encontrar una ruta.

Para cada uno de los nodos analizados por el algoritmo A^* dentro del mapa se encuentran los siguientes estados posibles:

- a. Limpio: Este nodo todavía no ha sido explorado, por lo que aún no se ha evaluado si esta bloqueando o no el camino del agente autónoma y se tiene garantía de que este nodo aún no existe dentro de ninguna lista del algoritmo.
- b. Pasable: Este es transitable (pasable) para el algoritmo A^* . Si esta bandera se encuentra, es porque se ha examinado el nodo anteriormente y no se posee un estado de bloqueado para ese nodo.
- c. Bloqueado: El nodo esta bloqueado para el algoritmo A^* porque anteriormente se ha examinado y no se posee un estado de pasable.
- d. Abierto: El nodo se encuentra en la lista abierta.
- e. Cerrado: El nodo se encuentra en la lista cerrada.

Para el manejo de la lista abierta, existen 2 métodos comúnmente usados:

- Estructura de pila ordenada, el cual permite remover nodos de una forma rápida, pero posee una inserción lenta.
- Lista desordenada la cual permite una rápida inserción, pero es un método lento de remover nodos.

Para el manejo de la lista cerrada, se puede hacer uso de una lista ordenada, la cual posee todos aquellos nodos que ya han sido explorados.

Retornando Rutas Parciales

Cuando el algoritmo A* falla en encontrar la ruta de destino, es porque no existe o porque es imposible de llegar, aunque el algoritmo usualmente regresa una ruta parcial. La ruta parcial puede ser aquella que posee el costo total menor o la que posee un costo mínimo estimado para llegar al nodo de final. Por tal razón, la ruta formada es un fragmento de la ruta más corta para llegar al destino.

Ejemplo Algoritmo A*:

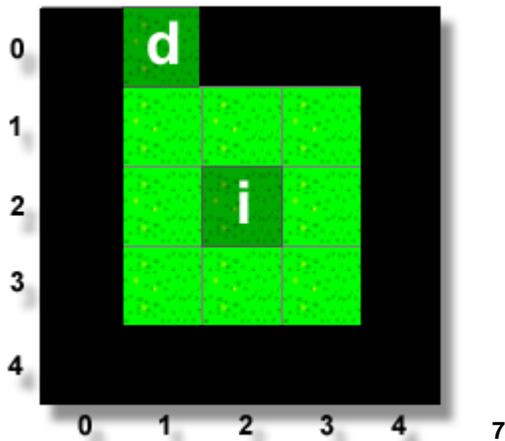


Fig. 3 Mapa simple.

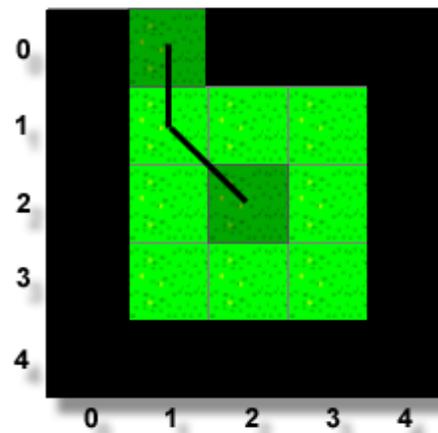


Fig. 4 Ruta Solución.

El punto central es la posición inicial (*i*), y la posición final está representada por *d*. Se asignan los valores de *f*, *g* y *h* los cuales son simples de asignar. El valor de *g* es cero porque no representa ningún costo para pasar sobre punto inicial *i*. Para calcular *h* (heurística) se hará uso del método Atajo Diagonal, el cual utiliza el valor máximo de la diferencia de la posición actual del agente autónomo hasta la posición final *d* haciendo uso de las coordenadas *x* y *y*. Iniciando la simulación de la siguiente manera:

Se inicializan las variables:

$$A.x = 2$$

$$A.y = 2$$

$$\text{goal}.x = 0$$

$$\text{goal}.y = 0$$

Donde:

A.x = Posición actual del agente autónomo en la posición coordenada x

⁷ *d* representa el destino de la simulación e *i* representa el inicio de la simulación.

A.y = Posición actual del agente autómatas en la posición coordenada y

goal.x = Posición final que busca tener el agente autómatas en la posición coordenada x

goal.y = Posición final que busca tener el agente autómatas en la posición coordenada y

Aplicando la ecuación 3.

$$h = \max (\text{abs} (A.x-\text{goal}.x), \text{abs} (A.y-\text{goal}.y))$$

$$h = \max (\text{abs} (2 - 0), \text{abs} (2 -0))$$

$$h = 2^8$$

Al poseer h un valor de 2 y g un valor de 0, f será la suma de g más h .

$$f = g + h$$

Por lo que da como resultado luego de aplicar la ecuación 1 se tiene:

$$f = 0 + 2$$

$$f = 2$$

Generar los nodos hijos es sencillo en vista que todos los nodos son validos (se puede desplazar en las ocho direcciones) y todos los nuevos nodos serán agregados a la lista abierta. El valor de g de cada nodo hijo será 1, ya que es el costo de llegar al nodo padre que se encuentra en la posición cero. El valor de h , será diferente para cada nodo, pero se puede observar que la posición (1,1) es el que posee el menor valor ya que se encuentra cerca del nodo final. Por tal razón, el nodo hijo (1,1) es el siguiente nodo a explorar.

Posteriormente, hay que analizar los nodos adyacentes al nodo (1,1), los cuales son (1,0) (1,2) (2,1) y (2,2). A partir de estos se debe determinar

⁸ h toma el valor mayor del resultado de ambas operaciones. Para el ejemplo, ambos valores son iguales por lo que es indiferente el valor que se toma. Max: significa obtener el valor máximo del resultado de las 2 diferencias realizadas. Abs obtiene el valor absoluto.

cuales nodos se encuentran en la lista abierta o la lista cerrada y cuales son considerados como nuevos nodos. Para el caso cuando se analice el nodo (2,2) se encontrará en la lista cerrada ya que es el punto de inicio del análisis. El nodo (1,2) y (2,1) se encuentran ya en la lista abierta porque son adyacentes también al punto de inicio y aún no han sido explorados, y se considera como nodo nuevo a la posición (1,0).

Después de asignar los valores de f, g y h a cada uno de los nuevos nodos, será evidente que el nodo (1,0) poseerá el menor valor, y en la próxima iteración del algoritmo A*, se descubrirá que es el nodo de destino.

2.2 Algoritmo D*

El nombre D* fue escogido porque se asemeja al algoritmo A*, excepto que es dinámico porque recalcula los costos de movilización a medida se encuentra un cambio durante la búsqueda de una ruta desde un punto de partida hasta un punto final.

Definiciones

El objetivo primordial que busca este algoritmo es movilizar un objeto autónoma desde un punto inicial dentro del mapa hasta un punto final, evitando obstáculos y minimizando el tiempo recorrido. Para lograr esto, cada uno de los obstáculos posee un valor (costo) que denota cuál es la ubicación del objeto autónoma. El objeto autónoma inicia en un estado en particular y este comienza a moverse en el mapa hasta que este alcanza el punto final, que será denotado por G. Cada uno de los estados por los que el objeto móvil pasa (denotados por X), poseen un puntero hacia atrás para el siguiente estado Y, el cual esta denotado por $b(X) = Y$.

D* utiliza punteros hacia atrás para representar las rutas hacia el punto final. El costo de moverse a través de una ruta desde el estado Y hasta el estado X es representado por un número positivo el cual está dado por la función de la representación de la ruta $c(X,Y)$.

Si no existe una ruta desde Y hasta X, entonces $c(X,Y)$ es una función indefinida. Para que los estados X y Y sean considerados vecinos (se encuentren adyacentes) es que el espacio de la función $c(X,Y)$ o $c(Y,X)$ está definida.

Al igual que el algoritmo A*, el algoritmo D* posee una lista abierta en la cual se encuentran todos aquellos estados o nodos que aún no han sido explorados. La lista abierta, se utiliza para propagar la información de todos aquellos cambios de rutas que pueden existir y también calcula los costos para todos aquellos nodos o estados que existan o se encuentren adyacentes a este.

Cada uno de los estados X posee una sub-función (etiqueta).

- i. $r(X) = \text{NUEVO}$, si X nunca se ha encontrado en la lista Abierta.
- ii. $r(X) = \text{Abierto}$, si X se encuentra en la lista Abierta.
- iii. $r(X) = \text{CERRADO}$, si X no se encuentra en la lista Abierta.

Por tal razón, para cada uno de los estados X, el algoritmo D* mantiene un estimado de la suma de los costos X a G haciendo uso de la función de costo $h(G,X)$. Dadas estas condiciones, este estimado es equivalente al costo óptimo (mínimo) desde el estado (nodo) X hasta G, esto es por la función $o(G,X)$. Para cada uno de los estados X en la lista Abierta, existe una función principal, $k(G,X)$, la cual está definida por ser igual al mínimo de $h(G,X)$ antes de cualquier modificación y todos los valores asumidos por

$h(G,X)$ desde que X fue colocado en la lista Abierta. La función principal clasifica al estado X en la lista Abierta en los siguiente tipos: un estado Superior(Raise) si $k(G,X) < h(G,X)$, y un estado Bajo si $k(G,X) = h(G,X)$. El algoritmo D^* utiliza los estados Superiores de la lista Abierta para propagar información acerca de los incrementos en costo de las rutas, y utiliza el estado Bajo para propagar las reducciones en los costos de los rutas. Este proceso de propagación de información se produce a medida se remueven los estados de la lista Abierta, por lo que cada vez que se remueve un estado, se pasan los cambios de costo a los estados adyacentes.

Todos los estados que se encuentran en la lista Abierta, se encuentran ordenados por los valores de la función principal. El parámetro k_{\min} está definido para ser $\min(k(X))$ para todos aquellos X que $r(X) = \text{Abierto}$. El parámetro k_{\min} es importante para el algoritmo D^* , debido a que todas aquellas rutas que posean un costo menor o igual al valor de k_{\min} son óptimas, mientras que si sus valores son mayores, no son rutas óptimas.

Descripción del Algoritmo

Este algoritmo consiste principalmente de 2 funciones, las cuales son ESTADO-PROCESO y MODIFICAR-COSTO.

La función ESTADO-PROCESO es usada para buscar la ruta más óptima en base a su costo para desplazarse hasta un punto, y la función MODIFICADOR- COSTO es usada para cambiar la función del costo de la trayectoria $c(^{\circ})$ e ingresar todos aquellos estados modificados a la lista Abierta. Inicialmente, $t(^{\circ})$ se encuentra en Nuevo para cada uno de los estados que corresponden el mapa, $h(G)$ se encuentra en cero, y G es colocada en la lista abierta. La primera función Estado-Proceso se continua

ejecutando hasta que el estado X ha sido cambiando a la lista cerrada, o el valor de retorno que ha enviado ha sido -1, lo cual significa que no ha sido encontrada una ruta o no se encuentra. Hasta este punto el objeto autómata revisa los estados adyacentes al estado principal hasta que descubre el punto de llegada o encuentra un error en la función que muestra la trayectoria $c(\circ)$. Por ejemplo, porque se ha encontrado un obstáculo el cual no permite llegar hasta el destino. En este punto la segunda función Estado-Proceso es llamada, la cual se utiliza para corregir el valor de $c(\circ)$ y colocar todos aquellos estados que han sido afectados en la lista Abierta. El objeto autómata posee un estado Y, el cual se utiliza para descubrir cualquier error en $c(\circ)$. Hasta que la función Estado-Proceso regresa un valor $k_{\min} \geq h(Y)$, se propagan los cambios de costo al estado Y hasta que se da la igualdad en la función $h(Y) = o(Y)$. Es posible que hasta este punto se hayan encontrado nuevas rutas y que hasta hayan sido construidas, esto ayuda al objeto autómata a seguir revisando los estados adyacentes hasta que se encuentra una ruta que llegue hasta la meta propuesta.

Pseudo-Código

Función Estado-Proceso

X = Estado-Min ()

SI X = NULO ENTONCES REGRESAR -1

$k_{old} = \text{GET} - \text{KMIN} (); \text{BORRAR}(X)$

SI $k_{old} < h(X)$ ENTONCES

 PARA CADA Estado Adyacente Y de X:

 SI $h(Y) \leq k_{old}$ y $h(X) > h(Y) + c(Y,X)$ ENTONCES

$b(X) = Y$

$h(X) = h(Y) + c(Y,X)$

 FIN SI

SI $k_{old} = h(X)$ ENTONCES

PARA CADA Estado Adyacente Y de X:

SI $t(Y) = \text{NUEVO}$ o

($b(Y) = X$ y $h(Y) \neq h(X) + c(X,Y)$) o

($b(Y) \neq X$ y $h(Y) > h(X) + c(X,Y)$) ENTONCES

$b(Y) = X$

AGREGAR (Y, $h(X) + c(X,Y)$)

DE OTRA FORMA

PARA CADA Estados Adyacentes Y de X:

SI $t(Y) = \text{NUEVO}$ o

($b(Y) = X$ y $h(Y) \neq h(X) + c(X,Y)$) ENTONCES

$b(Y) = X$

INSERTAR (Y, $h(X) + c(X,Y)$)

ENTONCES

SI $b(Y) \neq X$ y $h(Y) > h(X) + c(X,Y)$ ENTONCES

INSERTAR (X, $h(X)$)

ENTONCES

SI $b(Y) \neq X$ y $h(X) > h(Y) + c(Y,X)$ y

$t(Y) = \text{CERRADO}$ y $h(Y) > k_{old}$ ENTONCES

INSERTAR (Y, $h(Y)$)

RETORNAR EL VALOR GET-KMIN

En la función MODIFICAR-COSTO, la función que actualiza la ruta seleccionada se actualiza con los valores que han sufrido algún cambio. Dado que los valores para los estados de la ruta Y cambian, X es colocado en la lista Abierta. Cuando el estado X se expande haciendo uso de la función Estado-Proceso, este recalcula un nuevo valor para $h(Y) = h(X) + c(X,Y)$ y coloca Y en la lista Abierta. Adicionalmente, el estado Y actualiza los costos para los estados descendientes de él.

Función Modificador-Costo (X,Y,cval)

```
C(X.Y) = cval  
SI t(X) = CERRADO ENTONCES  
    INSERTAR (X,h(X))  
RETORNAR GET-KMIN ()
```

Luego de la implementación del pseudo-código anterior se encontró la necesidad de modificarlo ampliamente porque este solo evaluaba si un nodo representa un obstáculo o no para el agente autónomo. Por tal razón, se procedió a modificar el pseudo-código para hacer uso de los valores de costo g y heurística h de cada nodo y así obtener una mejor ruta de forma más rápida.

Pseudo-Código Optimizado

1. S = Punto de Inicio.
2. P = S
3. Hacer mientras P diferente -1
 - a. Asignar f , g y h a P.
 - b. Asignar f , g y h a nodos adyacente a P.
 - c. P1 = Nodo adyacente a P con menor f
 - d. Si P1 se encuentra en lista de nodos analizados
 - i. Si P1 es diferente a P entonces
 1. Borrar P1 lista nodos analizados
 2. Asignar costo nodo no pasable a P
 - e. Si P1 no existe en lista de nodos analizados agregarlo.
 - f. Si P1 es igual al nodo salida entonces
 - i. Salir
 - g. Dibujar nodo P1

- h. $P=P1$
- 4. Si $P1 = -1$ Entonces no existe camino y agente autómata regreso al punto de partida.

2.3 Aprendizaje aplicado Agente Autómatas

El aprendizaje es un proceso inteligente, repetitivo, eficaz y eficiente, porque permite resolver problemas cambiantes, detectar y corregir todo aquel conocimiento que ha sido adquirido previamente y que es erróneo, resolver problemas desconocidos en entornos inaccesibles.

Tipos de Aprendizaje:

- a. Implementación directa, el aprendizaje es programado haciendo uso del software, hardware o ambos.
- b. Aprendizaje mediante instrucciones, se posee una fuente de la cual se obtienen los datos o información a aprender. Por ejemplo, libros, maestros, etc.
- c. Aprendizaje por deducción, donde las estructuras del conocimiento se van obteniendo en base a la experiencia adquirida.
- d. Aprendizaje por analogía, en el cual se da una extensión del conocimiento de un dominio a otro.

- e. Aprendizaje por inducción, el cual puede ser adquirido mediante ejemplos o por observación, en el cual se descubren relaciones a partir de ejemplos sin deducción.

Pseudo-Código

El pseudo-código siguiente ha sido aplicado al algoritmo D^* al momento de analizar cada uno de los nodos por lo que se desplaza para llegar al nodo final haciendo uso de un tipo de aprendizaje por deducción.

1. $P =$ Nodo inicio
2. $G =$ Nodo llegada
3. $PA =$ Nodo inicio aprendido
4. $GA =$ Nodo llegada aprendido
5. Si aprendizaje existe entonces
 - a. Si $P = PA$ y $G = GA$ entonces
 - i. Inicializar contador en cero.
 - ii. Hacer mientras lista aprendizaje posea información diferente a -1
 1. Si costo lista aprendizaje en posición contador es igual al costo del nodo del mapa actual en la misma posición Entonces
 - a. $P =$ valor lista aprendizaje en la posición contador
 - b. Dibujar agente autómatas en posición P
 - c. Si $P = G$ entonces
 - i. Salir
 - d. Si P es diferente de G
 - i. Incrementar contador
 - e. Incrementar contador
 2. Si costo lista aprendizaje es posición contador es diferente al costo del nodo del mapa actual

- a. Salir
6. Analizar ruta nuevamente

2.4 Áreas de Aplicación

Las primeras aplicaciones en esta área estuvieron enfocadas a desarrollar algoritmos para juegos.

Actualmente, la Inteligencia Artificial es una rama de la teoría de la computación que incluye áreas tales como el razonamiento automático, la demostración de teoremas, los sistemas expertos, el procesamiento de lenguaje natural, robótica, lenguajes y ambientes de Inteligencia Artificial, aprendizaje, redes neuronales, algoritmos genéticos, por mencionar sólo algunas.

En general, las áreas de aplicación de la Inteligencia Artificial tienen características similares, entre las que se pueden mencionar las siguientes (Luger y Stubblefield, 1989):

1. Aplicación de razonamiento simbólico mediante modelos computacionales.
2. Aplicación de técnicas de búsqueda a problemas de IA en lugar de soluciones algorítmicas.
3. Análisis de características cualitativas del problema para plantear su solución.

4. Utilización del significado semántico como la forma sintáctica de la información.
5. Manipulación de grandes cantidades de conocimiento específico para la solución de problemas.
6. Aplicación de conocimiento de meta-nivel para tener un control más sofisticado de estrategias de solución de problemas

Todos estos puntos han sido utilizados para elaborar una serie de proyectos donde se han creado robots y otros equipos los cuales han ido recorriendo en tiempo real una superficie física (mapa) y evitan obstáculos para alcanzar una meta, tomando en cuenta factores de costo y tiempo para un desplazamiento óptimo. Equipos como el Spirit (2003), Pathfinder (1997), utilizados para exploraciones en otro planeta, han hecho uso de de la Inteligencia Artificial y Algoritmos de búsquedas de rutas como el D^* y A^* para llevar a cabo una misión.

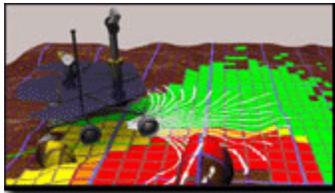


Fig. 5. Spirit, simulación en computador



Fig. 6. Pathfinder, pruebas en terreno

3. COMPARACION ALGORITMO A* Y D* CON APRENDIZAJE

A continuación se encuentran las simulaciones comparativas entre el algoritmo A* y el algoritmo D* con aprendizaje en equipos con que poseen diferentes especificaciones haciendo uso de distintos mapas para obtener el tiempo de ejecución de ambos algoritmos.

En las tablas comparativas presentadas, se tienen las columnas que muestran las especificaciones técnicas de cada equipo utilizado y los tiempos en segundos que se tardó el algoritmo A* y D* con aprendizaje.

Los diferencias de tiempos de recorridos para el agente autómeta que representa al algoritmo A* y D* con aprendizaje puede ser imperceptible dependiendo de las dimensiones y cantidad de obstáculos que posea un mapa. Al mismo tiempo es necesario tomar en cuenta las capacidades de procesamiento de datos que tenga el computador en el cual se efectúa la simulación.

Si el agente autómeta con el algoritmo D* se mueve en un mapa en el cual no realiza ninguna exploración (Fig. 7) en nodos que no le conducen a su punto final pero que igual son explorados (Fig. 8) para llegar al punto final el tiempo de recorrido de ambos algoritmos varia en una pequeña proporción (Tabla 1, 0.01 Seg de diferencia en algunos casos); sin embargo cuando el algoritmo D* recorre nodos que no son tomados dentro el aprendizaje (Fig. 8), el tiempo de recorrido es mayor que el algoritmo A* pero se reduce considerablemente cuando se aplica el aprendizaje(Tabla 2). Si los dos algoritmos recorren un mapa sin que existan modificaciones en el (Fig. 9) en tiempo real, el tiempo de recorrido del algoritmo D* puede o no ser menor que el del algoritmo A*

dependiendo del número de obstáculos que se tengan. Por tal razón el tiempo de recorrido del Algoritmo D* aplicándole aprendizaje siempre será menor que cuando se recorre la primera vez y que va aprendiendo la ruta (Tabla1, Tabla2, Tabla3). Si se modifica el mapa en tiempo real (Fig. 10) el tiempo que se obtiene (Tabla4) es el que toma a los agentes autómatas llegar a su destino a partir de haber realizado la modificación para poder validar que el en este caso el agente autómata D* será más rápido porque no tendrá que iniciar sus cálculos desde el punto de partida y percibir los cambios realizados para llegar a su punto final.

Partiendo de la hipótesis que el algoritmo D* sería mejorado, más rápido y eficiente si se le aplicaba un aprendizaje, comparado con el tiempo que le toma al algoritmo A* en llegar a su punto final y que el tiempo de desplazamiento del autómata D* se reduciría considerablemente cuando se modifica el mapa en tiempo real. Las pruebas realizadas en diferentes computadores muestran que el algoritmo D* si fue mejorado y que es más rápido y eficiente que el A* (Tabla 3), pero esto depende del diseño del mapa, porque para mapas pequeños y obstáculos apilados (Fig. 8) el tiempo del algoritmo D* siempre fue mayor que el A* aun aplicando aprendizaje (Tabla 2). También el tiempo de recorrido para el agente autómata D* sí se reduce considerablemente al compararlo con el autómata A* al modificar un mapa en tiempo real (Tabla 4).

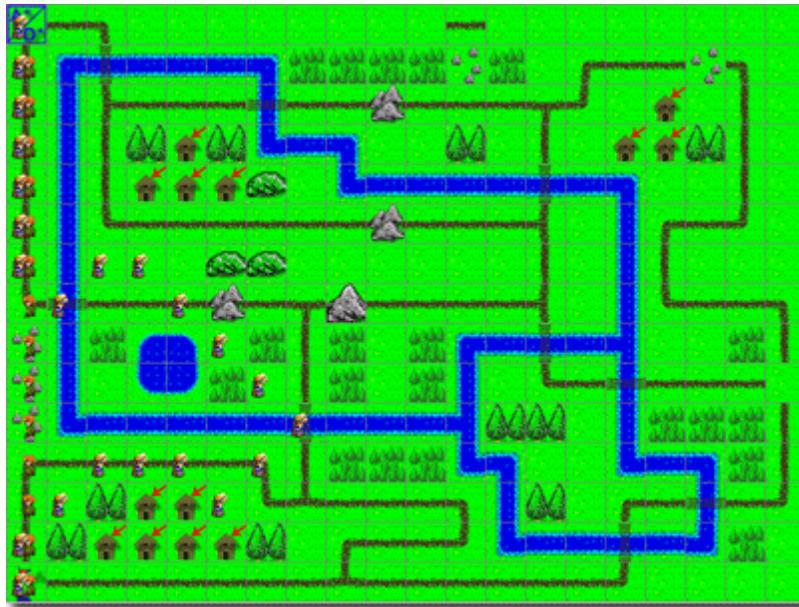


Fig. 7 Mapa simulación comparativa. Nombre mapa: MapaBig

Descripción de equipo	A*	D*	D* con aprendizaje
Procesador: 200 MHz Memoria: 32 Mb RAM Video: 2 MB RAM Compartida	23.69	18.61	18.60
Procesador: 500 MHz Memoria: 32 Mb RAM Video: 4 MB RAM Compartida	23.24	18.43	18.37
Procesador: 1000 MHz Memoria: 256 Mb RAM Video: 2 MB RAM	21.06	17.02	17.01
Procesador: 2400 MHz Memoria: 512 Mb RAM Video: 256 MB RAM	21.08	17.03	17.02
Procesador: 2800 MHz Memoria: 1000 Mb RAM Video: 1000 MB RAM Compartida	21.01	17.00	16.99

Tabla 1⁹

⁹ Todos los valores numéricos representan el tiempo en segundos que la simulación de los algoritmos han tardado en desplazarse.



Fig. 8 Mapa simulación comparativa. Nombre mapa: DStarOnly

Descripción de equipo	A*	D*	D* con aprendizaje
Procesador: 200 MHz Memoria: 32 Mb RAM Video: 2 MB RAM Compartida	6.26	14.57	9.35
Procesador: 500 MHz Memoria: 32 Mb RAM Video: 4 MB RAM Compartida	6.25	14.55	9.35
Procesador: 1000 MHz Memoria: 256 Mb RAM Video: 2 MB RAM	6.01	14.01	9.01
Procesador: 2400 MHz Memoria: 512 Mb RAM Video: 256 MB RAM	6.01	14.01	9.01
Procesador: 2800 MHz Memoria: 1000 Mb RAM Video: 1000 MB RAM Compartida	6.03	14.00	9.0

Tabla 2⁹



Fig. 9 Mapa simulación comparativa. Nombre mapa: PresentacionD

Descripción de equipo	A*	D*	D* con aprendizaje
Procesador: 200 MHz Memoria: 32 Mb RAM Video: 2 MB RAM Compartida	11.11	10.89	10.89
Procesador: 500 MHz Memoria: 32 Mb RAM Video: 4 Mb RAM Compartida	11.71	11.37	11.35
Procesador: 1000 MHz Memoria: 256 Mb RAM Video: 2 Mb RAM	9.02	9.01	9.01
Procesador: 2400 MHz Memoria: 512 Mb RAM Video: 256 Mb RAM	9.02	9.01	8.99
Procesador: 2800 MHz Memoria: 1000 Mb RAM Video: 1000 Mb RAM Compartida	9.0	9.0	8.95

Tabla 3⁹



Fig. 10 Mapa simulación comparativa. Nombre mapa: PresentacionD (Modificación del mapa en tiempo de ejecución)

Descripción de equipo	A*	D*	D* con aprendizaje
Procesador: 200 MHz Memoria: 32 Mb RAM Video: 2 Mb RAM Compartida	20.03	9.73	9.69
Procesador: 500 MHz Memoria: 32 Mb RAM Video: 4 Mb RAM Compartida	12.96	13.73	12.62
Procesador: 1000 MHz Memoria: 256 Mb RAM Video: 2 Mb RAM	10.02	9.02	10.01
Procesador: 2400 MHz Memoria: 512 Mb RAM Video: 256 Mb RAM	10.02	7.01	10.01
Procesador: 2800 MHz Memoria: 1000 Mb RAM Video: 1000 Mb RAM Compartida	8.0	8.0	9.9

Tabla 4 ⁹

GLOSARIO DE TERMINOS

Adaptabilidad:	Ajustar su actuación en el ambiente en el cual está interactuando.
Cognoscitivo:	De lo que es capaz de conocer.
Costo:	Es el valor que se ha generado al hacer uso de una ruta. Es decir representa el tiempo que se ha utilizado para desplazarse desde la posición inicial hasta la posición final.
Estados:	Se refiere al nombre dado en el algoritmo D* al término de nodos.
Distancia (o heurística):	Representa la factibilidad que un nodo posee de ser explorado.
Mapa:	Es el espacio utilizado por un algoritmo para desplazarse. El mapa puede ser creado haciendo uso de hexágonos, cuadrado.
Nodos:	Son las estructuras de datos que representan las posiciones dentro del mapa. Cada uno de los nodos posee información importante sobre su posición, si el nodo puede ser transitable, etc.
Pragmático:	Práctico, opuesto a teórico o especulativo.

CONCLUSIONES

Los algoritmos A* y D* fueron implementados para encontrar una ruta óptima desde un punto de inicio hasta uno final y así poder comparar el comportamiento de cada uno de ellos, obteniendo de esta manera, una visión clara de la aplicación de estos en un mapa que contenía obstáculos con diferentes costos, que representan la dificultad que los agentes autómatas encontrarían al querer pasar por ellos. De esta forma un costo en el tiempo es lo que tarda un agente autómata en moverse dentro de ese ambiente en busca de su objetivo.

La comparación de ambos algoritmos arrojó en las simulaciones en un computador que es más eficiente y rápido el algoritmo A* debido a que primero analiza el mapa obteniendo una ruta a seguir y hasta que tenga definida la ruta inicia su movimiento, si existen cambios en el mapa debe comenzar de nuevo todo el proceso para tomar en cuenta esa variación en el ambiente. Sin embargo, el algoritmo D* es más lento por su capacidad de evaluar la ruta a tomar cuando se va desplazando hasta alcanzar su objetivo; pero es mucho más rápido si existen cambios en el mapa, debido a que no tiene que iniciar todo el proceso sino solamente actualizar los costos que lleva almacenado en memoria para que en sea tomada en cuenta en el movimiento que esta a punto de realizar.

Las rutas encontradas por los agentes autómatas no siempre son las más óptimas en base al desplazamiento, sino en base al costos que representa moverse de un punto a otro encontrando siempre una ruta al punto final si esta existe.

De la desventaja que representa para el algoritmo D* frente al A* en la rapidez para encontrar una ruta al punto final (si no existe cambio alguno en el mapa al momento de ejecución), se tiene la aplicación del aprendizaje al algoritmo D*

brindando con esto una forma más rápida de moverse, porque cuando se corre de nuevo la simulación el algoritmo A* realiza de nuevo los cálculos, mientras que el algoritmo D* ya tiene almacenada cual fue la ruta que previamente encontró.

Mediante las pruebas realizadas se observó que el tiempo que el algoritmo D* con aprendizaje tarda en desplazarse desde su punto inicial hasta su punto final es menor a cuando se simula por primera vez el algoritmo, debido a que en esa ejecución, el algoritmo esta aprendiendo que ruta es la que puede utilizar posteriormente si el punto inicial y final es el mismo.

Luego de revisar la información proporcionada se observa que algoritmo D* con aprendizaje puede ser aplicado en distribución de paquetes sobre una red de área local donde se necesita encontrar la ruta más corta para desplazar un paquete desde un computador hasta otro teniendo la capacidad de moverse haciendo de uso del conocimiento adquirido reduciendo el tiempo de desplazamiento. Al mismo tiempo será capaz de reevaluar el camino a seguir si existe un cambio dentro de la red (concentradores dañados, cables cortados, etc.). Un simulador de tráfico o creador de rutas en calles urbanas es otra área donde se pueden aplicar los algoritmos de búsqueda de rutas (D*), debido a que permitiría predecir el compartimiento del tráfico cuando una calle es cerrada, o existe algún tipo de incidente, brindando siempre una ruta alterna de desplazamiento hasta el punto de llegada deseado.

RECOMENDACIONES

Se recomienda realizar lo siguiente para obtener mejores resultados en la evaluación del algoritmo D* o brindarle nuevas capacidades a la aplicación presentada:

1. Ampliar el rango de análisis de nodos para obtener una mayor rapidez en encontrar la ruta; es decir, en lugar de analizar los ocho nodos adyacentes, se pueden analizar veinticuatro nodos.
2. Aplicar un reconocedor de mapas para así dar al agente autómatas la capacidad de leerlo previamente para poder desplazarse analizando siempre por si existe una variación.
3. Creación de un agente autómatas mecánico incorporándole la aplicación para desplazarse, modificando la lectura de los datos haciendo uso de sensores los cuales darán a conocer los obstáculos y lugar dentro del mapa.
4. Proveer al agente autómatas la capacidad de detección de colisiones.

ANEXOS

Manual de Usuario: Algoritmos IA.

Panel de Simulador

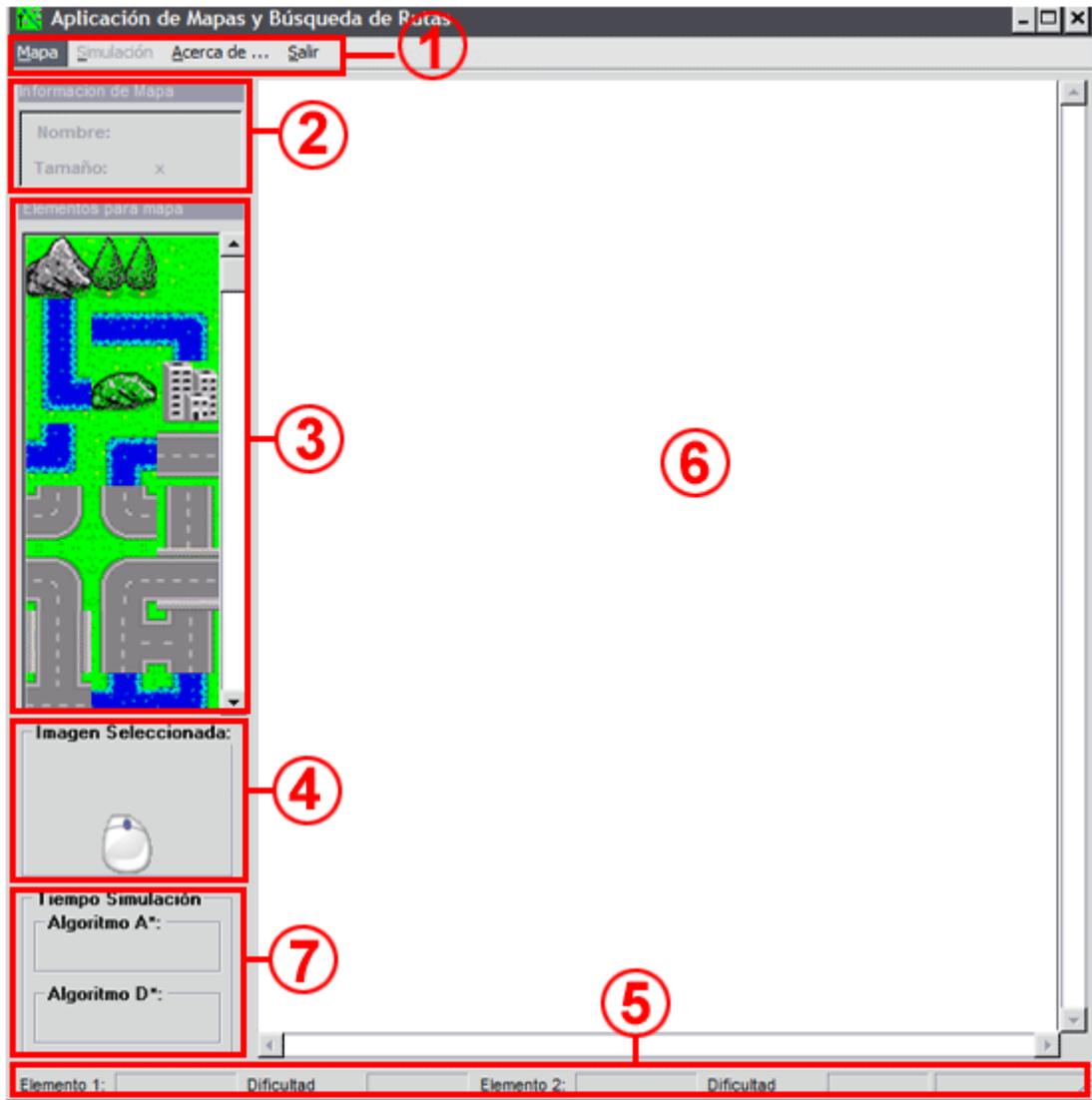


Fig. 11 Partes que constituyen el simulador.

Partes que constituyen el simulador:

1. Menú
2. Información de Mapa
3. Elementos para mapa
4. Elementos seleccionados
5. Barra de estado
6. Área de mapa
7. Tiempo transcurrido en simulación

1. Menú

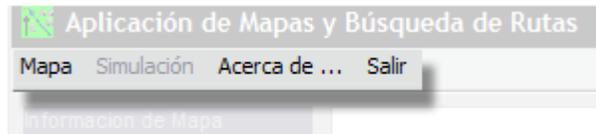


Fig. 12 Opciones de menú principal.

Menú principal de simulador:

1. Mapa
2. Simulación
3. Acerca de ...
4. Salir

1.1 Menú Mapa

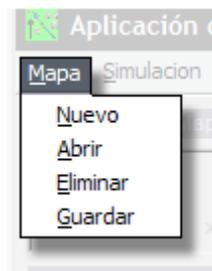


Fig. 13 Submenú mapas

Dentro del menú mapa se encuentran las opciones para crear, abrir y eliminar un

mapa, así como también, la opción para guardar el mapa o los cambios realizados en él.

1.1.1 Menú Nuevo Mapa

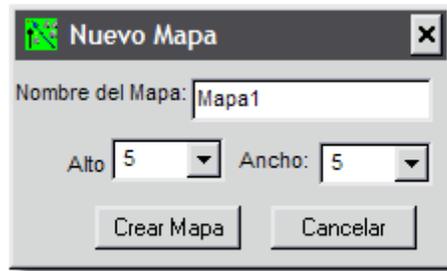


Fig. 14 Creación de nuevo mapa con sus dimensiones.

Este permite definir los parámetros iniciales que debe contener el mapa que se este creando.

1.1.1.1 Menú Nuevo Mapa Dimensiones

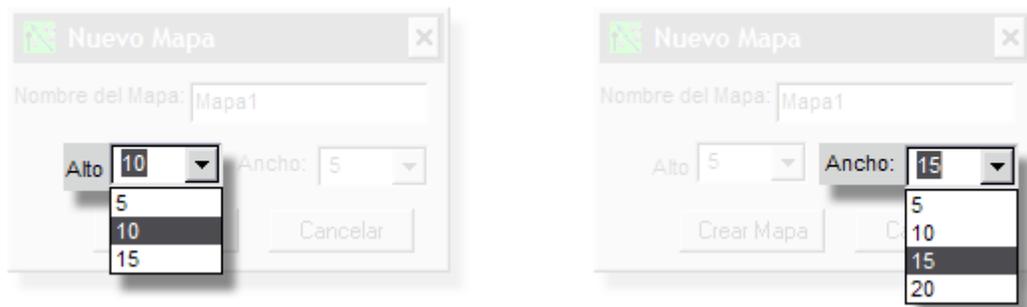


Fig. 15 Especificación de dimensiones del mapa.

Las dimensiones del mapa pueden ser seleccionadas en base a un alto y un ancho, cuyos máximos son 15 y 20.

1.1.1.2 Menú Nuevo Mapa Error 1

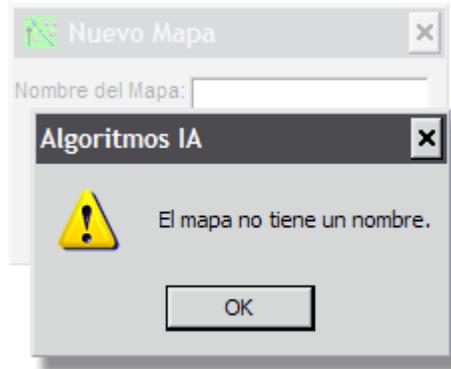


Fig. 16 Error porque no se especificó nombre de mapa.

Al crear un mapa debe asignársele un nombre, debido a que esto facilita los cambios al diseño realizado.

1.1.1.3 Menú Nuevo Mapa Error 2

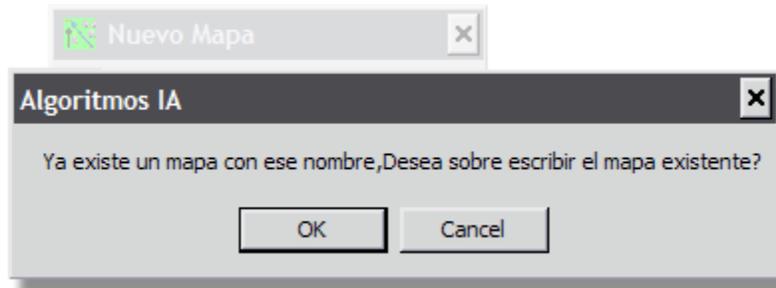


Fig. 17 Existe un mapa con el mismo nombre y pregunta si se desea sobrescribir o no.

Al colocar el nombre de un mapa ya existente el simulador preguntará si se quiere reemplazar. Debe tomarse en cuenta que si se reemplaza un mapa ya existente no será posible recuperarlo posteriormente.

1.1.2. Abrir Mapa

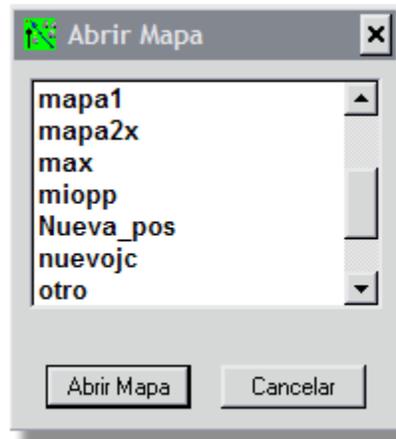


Fig. 18 Selección de un mapa previamente creado para su visualización.

Para abrir un mapa guardado anteriormente, se debe seleccionar un mapa del listado mostrado.

1.1.2.1 Abrir Mapa Error



Fig. 19 Debe seleccionar un mapa para poder utilizar la opción de abrir mapa.

Cuando no se ha seleccionado ningún nombre de mapa, se desplegará un mensaje de error indicando que se debe seleccionar antes un mapa que ya ha sido guardado previamente.

1.1.3. Eliminar Mapa

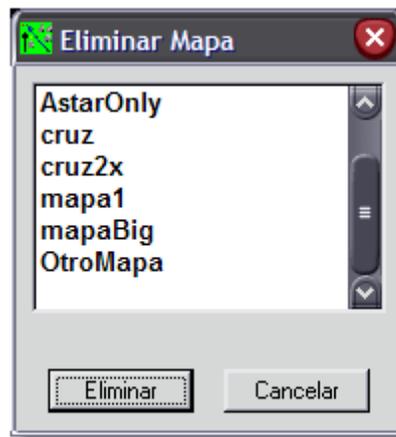


Fig. 20 Ventana de selección de para eliminarlo.

El usuario tiene la capacidad de poder eliminar un mapa que ya no desee tener almacenado.

1.1.3.1 Eliminar Mapa

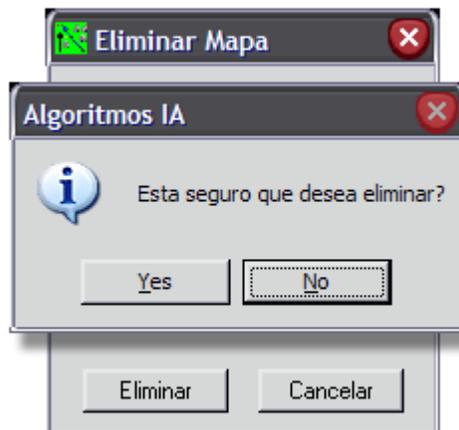


Fig. 21 Mensaje de confirmación para eliminar un mapa.

Es importante tomar en cuenta que si Elimina un mapa no será posible recuperarlo posteriormente. Si esta seguro de eliminarlo presione Yes (Si).

1.1.3.2 Eliminar Mapa Confirmación

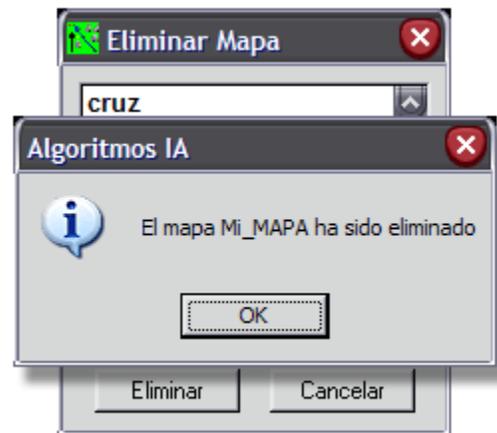


Fig. 22 Mensaje de confirmación que el mapa ha sido eliminado.

Cuando el mapa sea eliminando recibirá un mensaje indicando que el mapa ha sido eliminado satisfactoriamente.

1.1.3.3 Eliminar Mapa Error 1

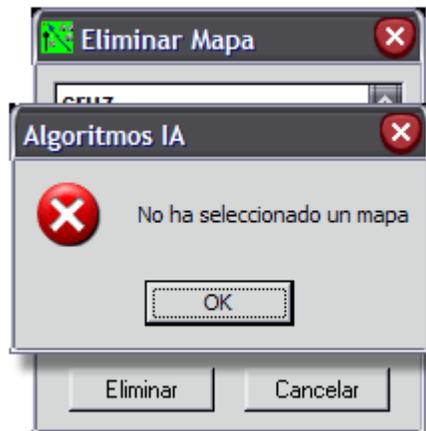


Fig. 23 Debe seleccionarse un mapa para su eliminación.

Cuando no se ha seleccionado un mapa de la lista, se presentará un mensaje indicando que aún no se ha seleccionado un mapa para ser eliminado.

1.1.4 Guardar Mapa

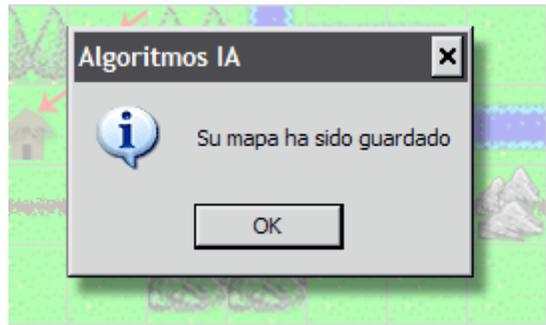


Fig. 24 Mensaje indicando que ha sido guardado el mapa.

El mapa puede ser guardado en el momento que se crea conveniente, en este caso cuando guarde el mapa desde el menú mapa (Fig. 13) aparecerá un mensaje indicando que se ha guardado satisfactoriamente.

1.2 Simulación

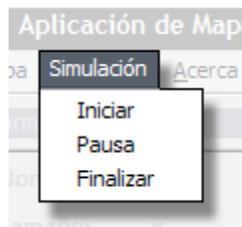


Fig. 25 Opciones del menú simulación.

Esta opción del menú permitirá detener, pausar o iniciar la simulación en la cual se verá el funcionamiento de los algoritmos ya implementados. Mientras no se haya cargado un mapa esta opción estará deshabilitada.

1.3 Acerca de ...

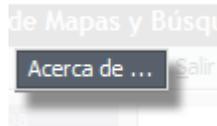


Fig. 26 Información acerca de la aplicación.

Muestra la información acerca del simulador.

1.3.1 Acerca de Información

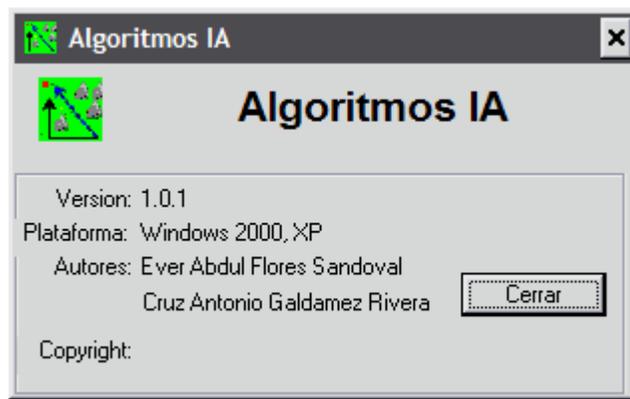


Fig. 27 Información acerca de la aplicación.

Permite visualizar la versión, plataforma en la que se puede ejecutar, autores y derechos (Copyright) del simulador de Algoritmos AI.

1.5 Salir



Fig. 28 Opción de abandonar el simulador.

Abandonar el Simulador.

1.5.1 Confirmación

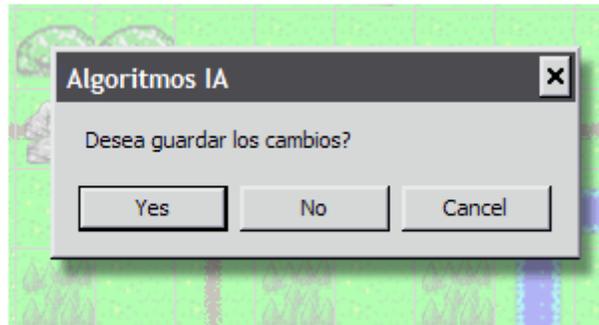


Fig. 29 Mensaje de confirmación para salir del simulador.

Si se quiere abandonar el simulador y tiene un mapa activo, le preguntará si desea guardar los cambios realizados al mapa, si presiona "Yes (Si)", guardará el mapa activo y abandonará el simulador; si presiona "No", abandonará el simulador y no se guardarán los cambios que se hallan realizado en el mapa activo, si no se quiere abandonar el simulador entonces presione "Cancel (Cancelar)".

2. Panel Información de Mapa

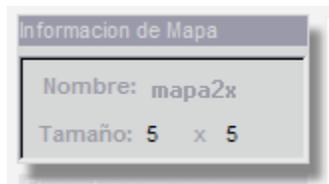


Fig. 30 Información mapa activo y sus dimensiones.

Muestra el nombre del mapa activo y las dimensiones que este tiene.

3. Panel Elementos de Mapa



Fig. 31 Panel con elementos utilizados para construcción de un mapa.

Contiene los elementos que se usan para construir un mapa. Puede seleccionar dos elementos (imágenes) que estarán disponibles (Fig. 32) para ser colocados dentro del mapa. Los elementos seleccionados pueden ser identificados en la lista en borde negro.

4. Imágenes seleccionadas



Fig. 32 Imágenes seleccionadas con el botón derecho e izquierdo del ratón.

Muestra cada uno de los elementos seleccionados del Panel de Elementos del Mapa (Fig. 31). Haciendo uso del botón izquierdo y derecho del ratón pueden seleccionarse dos imágenes simultáneamente las cuales se utilizan al momento de dibujar el mapa.

Para liberar las imágenes que fueron seleccionadas con los botones del ratón (Botón derecho o izquierdo), basta con dar doble clic sobre la imagen que se quiere liberar (Fig. 32).

5. Barra de estado

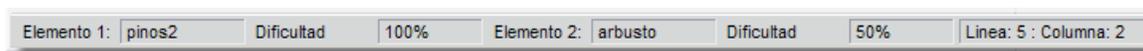


Fig. 33 La barra de estado muestra información de imágenes seleccionadas con el ratón.

En la barra de estado es posible visualizar el nombre y los porcentajes de dificultad que representa cada uno de los elementos (imágenes) seleccionados (Fig. 32).

También, muestra en la esquina inferior derecha el número de fila y columna en la que se encuentra el puntero del ratón.

6. Construyendo un mapa sencillo: Paso 1

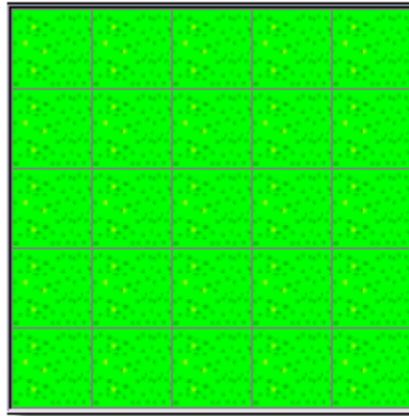


Fig. 34 Mapa mostrado luego de su creación.

Al crear un nuevo mapa (Fig. 14) debe asignársele un nombre (Fig. 14) y las dimensiones que tendrá (Fig. 15). Después de esto el mapa tendrá las dimensiones especificadas sin ningún elemento en él (Fig. 34).

6.1 Construyendo un mapa sencillo: Paso 2

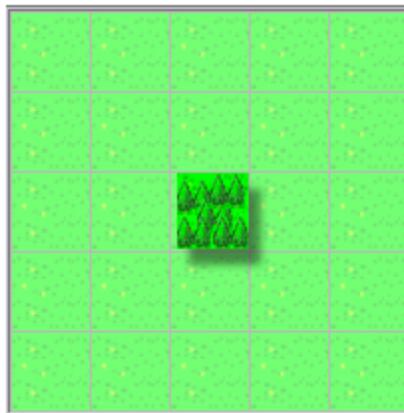


Fig. 35 Elemento colocado en el mapa.

Para iniciar el mapa, se deben seleccionar los elementos (imágenes) que poseerá (Fig. 31). Esto se hace dando clic en la parte del mapa en la cual se

desea colocar la imagen.

6.2 Construyendo un mapa sencillo: Paso 3

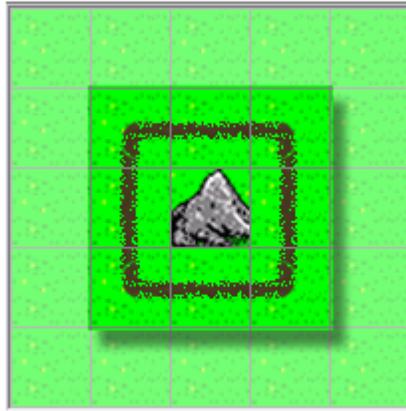


Fig. 36 Construcción de mapa en forma de caminos, ríos, etc.

Luego, se puede seleccionar las imágenes necesarias de tal manera que, al colocarlas juntas dentro del mapa puedan formar caminos, calles, ríos, bosques, etc.

6.3 Construyendo un mapa



Fig. 37 Imagen luego de finalizar un mapa.

Luego de terminar el mapa, este puede ser guardado (Fig. 24) para hacer modificaciones posteriores o ser usado para la simulación.

6.4 Elementos para Simulación

6.4.1 Punto Inicio Algoritmo A*



Fig. 38 Elemento que identifica el punto de inicio del algoritmo A*.

Indica el punto de inicio que tendrá el autómata 1 dentro del mapa, el cual representa la implementación del algoritmo A*.

*6.4.2 Punto Inicio Algoritmo D**



Fig. 39 Elemento que identifica el punto de inicio del algoritmo D*.

Indica el punto de inicio que tendrá el autómata 2 dentro del mapa, el cual representa la implementación del algoritmo D*.

6.4.2 Punto Inicio de Algoritmos A y D* compartido*



Fig. 40 Elemento que identifica el punto de inicio del algoritmo A* y D* compartido.

Indica un punto de inicio compartido que puede poseer el autómata 1 y 2 dentro del mapa para implementar los algoritmos A* y D*.

6.4.2 Punto de Destino de Algoritmos A y D**



Fig. 41 Elemento que identifica el punto de llegada de los algoritmos A* y D*.

Indica dentro del mapa el objetivo (punto de destino) que persiguen los agentes autómatas en la implementación de los algoritmos de búsqueda de rutas A* y D*.

6.5 Simulación del Algoritmo A*

6.5.1 Elementos Indispensables

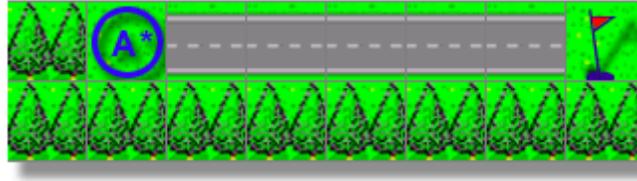


Fig. 42 Imagen de mapa para desplazar el algoritmo A*.

Para que se pueda iniciar la simulación (Fig. 25) es necesario abrir un mapa (Fig. 18) que se halla creado y guardado previamente, luego es necesario incluir en el elementos *Punto Inicio Algoritmo A** (Fig. 38) y *Punto de Destino de Algoritmos A* y D** (Fig. 41)

6.5.2 Simulación Error 1

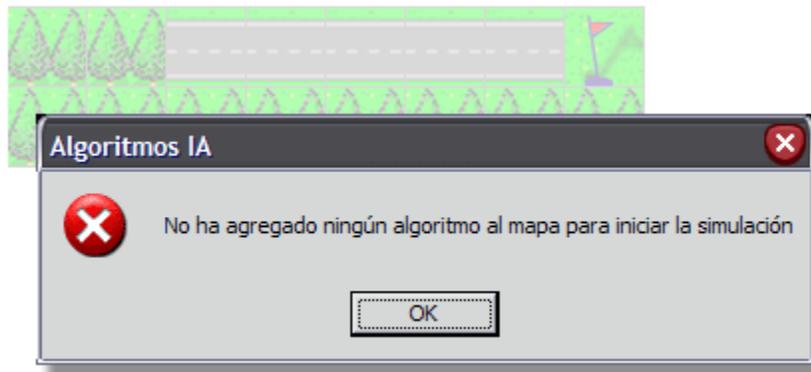


Fig. 43 No se ha indicado el punto de inicio para la simulación.

Si aún no se ha especificado el punto de inicio dentro del mapa, se desplegará un mensaje que no se ha indicado el punto de partida.

6.5.3 Simulación Error 2

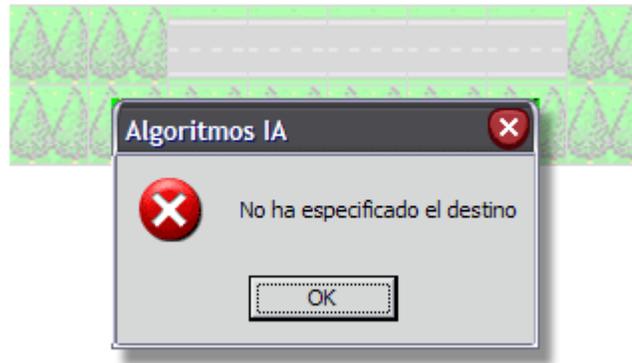


Fig. 44 No se ha indicado el punto de finalización para la simulación.

En caso de no agregar el punto al cual se desea llegar, no se podrá efectuar la simulación.

6.5.4 Simulación

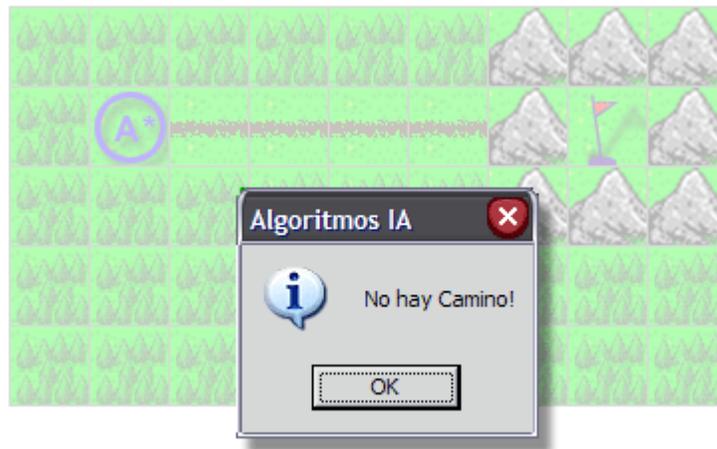


Fig. 45 No existe un camino para desplazarse desde el punto de inicio hasta el punto final.

Si al iniciar la simulación no existe un camino que pueda conducir al agente autómatas hasta la meta deseada, se recibe el mensaje “No hay Camino!”, el cual indica que no existe un camino libre de obstáculos por lo que no se puede llegar a la meta deseada.

6.5.5 Simulación

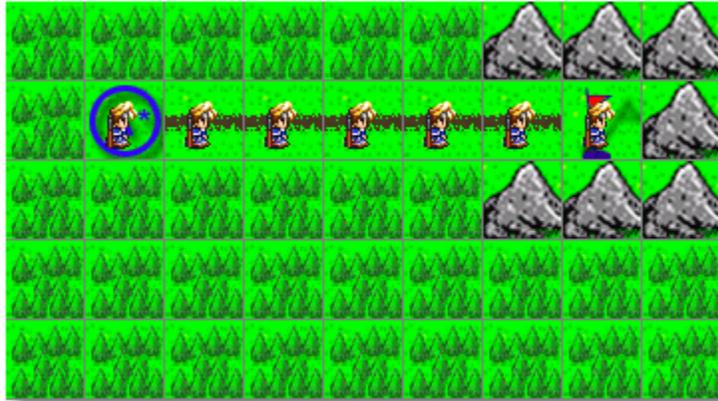


Fig. 46 Desplazamiento de un agente autónomo desde un punto de inicio hasta el punto final.

Al iniciar la simulación, el agente autónomo analizará el mapa y luego de conocer la ruta que lo lleva hasta el punto final se desplazará. A medida que este avance se mostrará cual ha sido el recorrido realizado.

6.5.6 Simulación



Fig. 47 Desplazamiento de agente autónomo en un mapa con mayor dificultad.

En la figura 47 el agente autónomo ha encontrado la mejor ruta para desplazarse la cual es rodeando las piedras que se están a la mitad del camino.

6.6. Simulación algoritmo D*



Fig. 48 Imagen de mapa para desplazar al algoritmo D*

Al igual que en el algoritmo A*, antes de iniciar la simulación es necesario haber colocado el punto de inicio del algoritmo D* y el punto de llegada. Luego de seleccionar en el menú el inicio de la simulación, el algoritmo D* iniciará su desplazamiento para llegar hasta el nodo de llegada.

En la figura 48 se muestra como el algoritmo D* analizó los nodos hasta que encontró la ruta hasta el punto de llegada.

6.7 Simulación algoritmo A* y D*



Fig. 49 Imagen de mapa para desplazar al algoritmo A* y D*

Al momento de la simulación es posible colocar en el mapa un mismo punto de

inicio y fin para los algoritmos A* y D*. En la figura 49 se muestra la ruta seguida por cada algoritmo.

6.7.1 Simulación algoritmo A* y D*



Fig. 50 Imagen de mapa para desplazar al algoritmo A* y D*

En la figura 50 también se encuentra otra simulación de los algoritmos A* y D* iniciando desde el mismo punto. .

6.7.2 Simulación algoritmo A* y D*



Fig. 51 Imagen de mapa para desplazar al algoritmo A* y D*

En la figura 51 también se encuentra otra simulación de los algoritmos A* y D*

iniciando desde diferentes puntos. Como se observa en la figura ambos algoritmos han encontrado una ruta diferente para desplazarse.

7. Tiempo Transcurrido en Simulación



Fig. 52 Tiempo de la simulación de los algoritmos A* y D*

Luego de la simulación de los algoritmos se muestra cual ha sido el tiempo en milisegundos que se han tardado en desplazarse desde el punto inicial hasta el punto final.

Manual Programador: Algoritmos IA.

Formulario Principal frmMain

Initialise()

Se encarga de inicializar las dimensiones de cada nodo en su ancho y su alto. También llama la función de cargar las imágenes prediseñadas y el panel en el que se muestran.

Tilelist()

Se encarga de cargar todas las imágenes que han sido configuradas para ser usadas en la creación de un mapa. También se encuentra las imágenes para representar el inicio de los algoritmos A*, D* y el punto de llegada. Todos los valores los obtiene de la base de datos tiles.mdb en la cual se encuentran los campos de identificación, nombre, costo de cada nodo.

DisplayTilePalette()

Se encarga de mostrar cada una de las imágenes en el panel que se muestra a la derecha de la ventana.

NewMap()

Dibuja las dimensiones que han sido especificadas para el mapa luego de definir su nombre.

DrawTile(ByVal TileNumber As Integer, intX As Integer, intY As Integer)

Dibuja la imagen que ha sido seleccionada del panel de imágenes. Toma los por referencia el número de imagen a colocar y su posición xy.

Parámetros:

TileNumber: Valor que referencia el numero de imagen que se encuentra en el panel.

intX: Posición en X en la cual se debe dibujar la imagen.

IntY: Posición en Y en la cual se debe dibujar la imagen.

DrawMap()

Carga las imágenes que posee un mapa que ha sido abierto para su simulación colocándolas en la posición en la cual fueron creadas. Al mismo tiempo se encarga de cargar la imagen predeterminada para un nuevo mapa en base a sus dimensiones.

ImgTile_MouseDown(index As Integer, button As Integer, Shift As Integer, X As Single, Y As Single)

Muestra la información de una imagen que ha sido seleccionada en la barra de estado (nombre, costo, etc.). También se encarga de mostrar en el recuadro de imagen seleccionada una vista previa de la selección hecha con el botón izquierdo o derecho del ratón.

Parámetros:

- Index: Número de imagen que han sido seleccionada.
- Button: Botón izquierdo o derecho del ratón que se utilizó para seleccionar una imagen.
- Shift: Se utiliza para determinar si se ha presionada la tecla ctrl, shift o alt del teclado.
- X: Posee posición X de la imagen.
- Y: Posee posición Y de la imagen.

PicMap MouseDown(button As Integer, Shift As Integer, X As Single, Y As Single)

Hace la llamada de la función de arrastre del ratón.

Parámetros:

- Index: Número de imagen que han sido seleccionada.
- Button: Botón izquierdo o derecho del ratón que se utilizó para seleccionar una imagen.
- Shift: Se utiliza para determinar si se ha presionada la tecla ctrl, shift o alt del teclado.
- X: Posee posición X de la imagen.
- Y: Posee posición Y de la imagen.

PicMap MouseMove(button As Integer, Shift As Integer, X As Single, Y As Single)

Posee la información de la posición en la que una imagen será colocada y cual de los botones del ratón ha sido seleccionada.

Se utiliza para hacer una validación de lo siguiente:

- No pueden existir mas de dos puntos de inicio para el algoritmo A*.
- No pueden existir mas de dos puntos de inicio para el algoritmo D*.
- No pueden existir mas de dos puntos de inicio para el algoritmo A* y D* cuando estos comparten el mismo punto de partida.
- No pueden existir mas de dos puntos de finalización de los algoritmos

Al momento de revisar la información anterior se da cualquiera de los casos, se sobrescribe la posición en la que se encontraba esa imagen con otra predeterminada.

Parámetros

- Button: Botón del ratón que ha sido utilizado para seleccionar una imagen.
- Shift: Se utiliza para determinar si se ha presionada la tecla ctrl, shift o alt del teclado.
- X: Posición en X en la que se desea colocar la imagen.
- Y: Posición en Y en la que se desea colocar la imagen.

PathStart()

Inicia la simulación para encontrar la ruta para desplazar al agente de un punto de inicio hasta un punto de fin. Avalúa que algoritmo de búsqueda de caminos simulará y en base a esta información hace la llamada respectiva de las funciones.

AstarPathStart()

Se encarga de llamar a la función del algoritmo A* para encontrar la ruta óptima para desplazar al agente autómeta desde su punto de inicio hasta el punto final. También inicializa el control para determinar el tiempo que le tomo al algoritmo A* desplazarse.

AstarTimer Timer()

Se encarga de mostrar cual es la ruta que el algoritmo A* ha encontrado para desplazarse desde su punto inicial hasta su punto final, evaluando el costo de cada nodo por el que va pasando porque ese es el tiempo que el agente autómeta mostrado debe quedarse antes de moverse al siguiente nodo.

DstarPathStart()

Se encarga de llamar temporizador para evaluar el camino a seguir por el algoritmo D*. Al mismo tiempo evalúa si existe un aprendizaje que pueda ser aplicado luego de determinar si el punto de inicio y fin corresponden a la última simulación del algoritmo. También se encarga de inicializar el control para determinar el tiempo que el algoritmo D* utilizó para desplazarse.

DstarTimer Timer()

Se encarga de seleccionar el mejor nodo a seguir para encontrar el camino hasta el nodo de llegada, mostrando la imagen del agente autómeta para comprender cual es la ruta escogida por el algoritmo D*.

También en el caso de encontrar una ruta óptima para desplazarse, va adquiriendo un aprendizaje el cual es guardado en una lista. En caso de simular nuevamente el algoritmo D*, evaluará si el aprendizaje adquirido anteriormente puede ser aplicado; en caso de existir algún cambio en la ruta, reevaluará todo el camino a partir de último nodo analizado.

Módulo modMap

SaveMapTiles(ByVal StrMapName As String)

Se utiliza para guardar todos aquellos cambios que fueron realizados al mapa.

Parámetros:

StrMapName: Posee el nombre la base de datos que será guardada.

LoadMapTiles(ByVal StrMapName As String)

Función que carga toda la información de todos los mapas que han sido guardados anteriormente. Entre los datos que se encuentran están las dimensiones (alto y ancho) del mapa, tipo de imagen que posee.

Parámetros:

StrMapName: Posee el nombre la base de datos que será guardada.

Módulo modDatabase

CreateMDB(ByVal strFileName As String, ByVal strPath As String)

Crea la base de datos que se utiliza para guardar la información del mapa. La creación de las tablas, campos e índices que contiene se hace llamando a las funciones que están dentro del mismo modulo.

Parámetros

StrFileName: Nombre asignado a la base de datos.

StrPath: Ruta en la cual se guardará la base de datos.

CreateTables()

Crea las tablas que contiene la base de datos en la que se guarda la información de los mapas.

CreateKeys()

Crea la llave primaria y foránea que poseen las tablas.

Módulo modTiles

tMapTile

Estructura que posee información acerca de cada uno de los nodos que conforman el mapa. Contiene datos de imagen, ubicación, costo y sección de la imagen.

MapTile

Estructura de datos heredada del tipo tMapTile.

tTile

Estructura que posee información de todas las imágenes disponibles para crear un mapa, establecer punto de inicio de algoritmos de búsqueda de caminos y su punto de llegada.

Tile

Estructura de datos heredada del tipo tTile.

Módulo mdBitBlt

BitBlt Lib "GDI32" (ByVal hDCDest As Long, ByVal XDest As Long, ByVal YDest As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hDCSrc As Long, ByVal XSrc As Long, ByVal YSrc As Long, ByVal dwRop As Long)

Se encarga de mostrar la imagen del agente autómatas que representa al desplazamiento del algoritmo A* y D* dentro del mapa.

Parámetros:

hDestDC: Destino en el que se dibujara una imagen.

X: Posición en X en la que debe dibujarse la imagen.

Y: Posición en Y en la que debe dibujarse la imagen.

nWidth: Ancho de la imagen a ser mostrada.

nHeight: Alto de la imagen a ser mostrada.
hSrcDC: Origen de la imagen a mostrar.
xSrc: Posición izquierda del origen de la imagen.
ySrc: Posición superior del origen de la imagen.
dwRop: Define como copiar la imagen de origen en el área definida para su destino.

Puede poseer los siguientes valores:

- SRCCOPY: Copiar la imagen tal como es.
- SRCPAINT: Copiar la imagen excepto lo que es negro.
- SRCAND: Copiar la imagen excepto lo que es blanco.

Módulo modAstar

FindPath(EntNode As Long, ExtNode As Long)

Se utiliza para buscar una ruta desde un punto inicial hasta un punto final. Analiza cada uno de los nodos calculando la conveniencia de cada nodo del mapa. Se auxilia de de varias funciones entre las que se encuentran H que encuentra la heurística, AddtoOpenList que agrega a una lista abierta todos los nodos.

Parámetros:

EntNode: Especifica la posición de inicio del algoritmo A*.
ExtNode: Se encuentra la posición del nodo de llegada para el algoritmo A*.

H(Pos As Long, Dest As Long)

Calcula la heurística de un nodo, es decir, un costo estimado para llegar desde ese nodo hasta la meta en base al método de Atajo Diagonal.

Parámetros:

Pos: Posición nodo actual en el cual se encuentra.

Dest: Posición nodo de destino.

AddToOpenList(Adding As Long, Dest As Long)

Agrega a la lista abierta todos aquellos nodos que aún no han sido explorados.

Parámetros:

Adding: Nodo que será agregado a la lista abierta.

Dest: Nodo que posee el punto de llegada del algoritmo.

RemoveFromOpenList()

Remover de una lista abierta todos aquellos nodos que ya han sido analizados y que han sido agregados a la lista cerrada.

GetAdjNode(N As Long, Dir As Long)

Toma un nodo adyacente para ser analizado.

Parámetros:

N: Contiene el nodo actual, es decir, el nodo en el cual se encuentra el algoritmo A* en un momento determinado.

Dir: Posición del nodo que deberá ser analizado.

ReArrangeHeap(ByVal Changed As Long)

Se encarga de ordenar la lista abierta para que en la parte abierta siempre quede el nodo con la menor conveniencia "f", así este será el próximo a ser analizado.

Parámetro:

Changed: Nodo adyacente que será ordenado dentro de la lista abierta.

FillSections()

Asigna la sección en la cual se encuentra cada nodo.

DrawPath()

Ordena el arreglo que contiene la ruta que el algoritmo A* ha encontrado para desplazarse hasta el punto de llega y activa el temporizador para dibujar la ruta en el mapa.

Módulo modDstar

FindPath_D(EntNode_D As Long, ExtNode_D As Long)

Busca cual es el nodo adyacente con la menor conveniencia para desplazarse. Se auxilia de la función H_D para calcular la heurística de los nodos analizados que han sido obtenidos con la función GetAdjNode_D.

Parámetros:

EntNode_D: Nodo en el que se encuentra actualmente el algoritmo D*.

ExtNode_D: Representa el nodo al que debe desplazarse algoritmo D*.

H_D(Pos As Long, Dest As Long)

Calcula la heurística de un nodo, es decir, un costo estimado para llegar desde el nodo que se esta analizando hasta la meta.

Parámetros:

Pos: Posición nodo actual en el cual se encuentra.

Dest: Posición nodo de destino.

GetAdjNode_D(N As Long, Dir As Long)

Toma un nodo adyacente para ser analizado.

Parámetros:

N: Contiene el nodo actual, es decir, el nodo en el cual se encuentra el algoritmo D* en un momento determinado.

Dir: Posición del nodo que deberá ser analizado.

Check List(Item As Long)

Se utiliza para verificar si el nodo que posee la menor conveniencia se encuentra ya en la lista de nodos que han sido analizados anteriormente.

Parámetros:

Item: Nodo que se esta analizando.

Insert Item(Item As Long)

Se utiliza para agregar a la lista de nodos analizados, el nodo que presenta la menor conveniencia. Al mismo tiempo también se agrega a la lista de memorización.

Parámetros:

Item: Nodo que presenta la menor conveniencia a ser agregado a la lista.

Delete Item(IPosition As Long)

Borra el nodo que presenta la menor conveniencia de la lista de nodos analizados si este ya ha sido revisado anteriormente. Al mismo tiempo también se utiliza para borrar de la lista de memorización el nodo analizado y todos los nodos que se encuentran en las posiciones siguientes.

Parámetros:

IPosition: Posición dentro de la lista que se borrará.

REFERENCIAS

Libros

- Buckland, Mat. AI Techniques for Game Programming (with CD-ROM); Press Hall, Inc. Ohio. 2002.
- Rabin, Steve. AI Game Programming Wisdom (with CD-ROM); Charles River Media, Inc. Massachusetts. 2002.
- Rabin, Steve. AI Game Programming Wisdom 2 (with CD-ROM); Charles River Media, Inc. Massachusetts. 2002.
- Luger and Stubblefield. "Artificial Intelligence and the Design of Expert Systems". The Benjamin/Cummings Publishing Company, Inc. California. 1989.
- Rusell S. and Norving P. "Inteligencia Artificial: Un Enfoque Moderno". Prentice Hall. México. 1996.
- N. Negroponte. Being Digital. Hodder and Stoughton. New York. 1995

Sitios Web

- GamaSutra
Descargado de <http://www.gamasutra.com>
el 15 de septiembre de 2004.

- GameDev.net:
Descargado de <http://www.gamedev.net/>
el 15 de Febrero de 2004.
- The Game Programming and Design Search Engine
Descargado de <http://www.gdse.com>
el 15 de Febrero de 2004.
- Descargado de http://www.frc.ri.cmu.edu/~axs/dynamic_plan.html
el 15 de Febrero de 2004.
- flipCode - Game Development News & Resources
Descargado de <http://www.flipcode.com/>
el 15 de Febrero de 2004.
- GarageGames
Descargado el <http://www.garagegames.com>
el 15 de Febrero de 2004.
- Descargado de <http://www.csc.liv.ac.uk/~mjw/pubs/jaamas98.pdf>
el 12 de Febrero de 2004.
- Descargado de <http://www.ecs.soton.ac.uk/~nrj/download-files/KE-REVIEW-95.ps>
el 12 de Febrero de 2004.
- Descargado de <http://www.loebner.net/Prizef/TuringArticle.html>
el 17 de Febrero de 2004.
- Descargado de <http://www.turing.org.uk>
el 17 de Febrero de 2004.

INDICE HONOMÁSTICO

A

A*, 14-16, 17-19

definición de, 14

ejemplo algoritmo, 20

Adaptabilidad, 3

definición de, 32

Agentes Automatas

definición de, 8

Aprender, 6, 8, 28

Aprendizaje, 3, 10

definición de, 28

tipos de, 28

mediante instrucciones, 28

por Analogía, 28

por deducción, 28

por Inducción, 29

Atajo Diagonal, 15, 20, 67

B

Bloqueado, 28

Búsqueda, 13, 37, 65

técnicas de, 6

Búsqueda Rutas, 13

definición de, 13

C

Conocimiento, 3, 6, 8, 11, 31

Conveniencia, 14, 66, 70-71

Costo, 13-17

definición de, 32

D

D*, 22

descripción del algoritmo, 22-23

objetivo del, 22

punto inicio algoritmo, 51

E

Espacio de estados, 6-7

definición de, 6

H

Heurística, 8, 14-15, 20, 27, 66, 69

definición de una, 8, 32

I

Implementación Directa, 28

Inteligencia Artificial, 8-9

características, 4

características de sus áreas de

aplicación, 30

conceptos de, 2-3

definición de, 2

Interfaces computadora-hombre, 12

definición de, 8

L

Lista Abierta, 16, 21, 23, 67-68

definición de, 16

métodos para el manejo de, 19

Lista Cerrada, 16, 18-19, 67

definición de, 16

Lista Desordenada, 19

Lista Nueva, 23

M

Manual de Usuario, 36

Mapa, 13-15, 17-18, 20, 22, 24, 29, 31

definición de, 32

Memorización, 13, 70-71

Método Manhattan, 15

Modelo Simbólico, 19-11

N

Nodos, 7, 13-14, 16, 18-19, 21-23,

27, 29

definición de, 17, 32

tipos de, 17

O

Objetos, 8, 12

Orientación de Newell, 3, 6

P

Prueba de Turing, 4-6

Pseudo-Código A*, 17

Pseudo-Código D*, 25, 27

R

Razonar automáticamente, 6

Ruta(s), 13-14, 19, 23, 25, 31, 33, 51

S

Simulación, 3, 20, 31, 34, 37, 43, 50,

52-54

Sistemas Concurrentes, 8, 12

