PROYECTO 2 – PATHFINDING ROBOT

201807085 – Cristian Alexander Mejía Cahuec

Resumen

La empresa Chapín Warriors, S. A. ha desarrollado equipos automatizados para rescatar civiles y extraer recursos de las ciudades que se encuentran inmersas en conflictos bélicos. Con el fin de realizar las misiones de rescate y extracción, Chapín Warriors, S. A. ha construido drones autónomos e invisibles para los radares llamados ChapinEyes. Los ChapinEyes sobrevuelan las ciudades y construyen un mapa bidimensional de la misma, este mapa bidimensional consiste en una malla de celdas, donde cada celda es identificada como un camino, un punto de entrada, una unidad de defensa, una unidad civil, un recurso o una celda intransitable.
Para esto se hará uso de un algoritmo que permitirá hacer el recorrido del robot hasta que encuentre el objetivo.

Palabras clave

TDA: es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico. Un TDA es una abstracción que permite modelar las características de un elemento en particular.

XML: es un lenguaje de marcado similar a HTML.

Nodo: es un registro que contiene un dato de interés y al menos un puntero para referenciar (apuntar) a otro nodo.

Matriz Dispersa: Nodos entrelazados entre sí que contienen apuntadores que apuntan hacia arriba, abajo, izquierda y derecha.

Pila: Una cola es una estructura de datos donde el último elemento en entrar es el primero en salir (Last In, First Out)

Abstract

The company Chapín Warriors, S.A. has developed automated equipment to rescue civilians and extract resources from cities that are immersed in armed conflicts. In order to carry out rescue and extraction missions, Chapín Warriors, S.A. has built autonomous drones that are invisible to radars called ChapinEyes. The ChapinEyes fly over the cities and build a two-dimensional map of it, this two-dimensional map consists of a grid of cells, where each cell is identified as a road, an entry point, a defense unit, a civil unit, a resource or an impassable cell.

For this, an algorithm will be used that will allow the robot to travel until it finds the objective.

Keywords

TDA: is a set of data or objects created in a personalized way by a programmer for a specific purpose. A TDA is an abstraction that allows modeling the characteristics of a particular element.

XML: It is a markup language similar to HTML.

Node: is a record that contains a data of interest and at least one pointer to reference (point) to another node.

Sparse Matrix: Interlocking nodes containing pointers pointing up, down, left, and right.

Stack: A queue is a data structure where the last element to enter is the first to leave (Last In, First Out)

Introducción

En este proyecto se trata de la elaboración de un algoritmo que permita realizar con éxito una misión de rescate de civiles o una de extracción de recursos.

Para las misiones de rescate se utilizará un robot ChapinRescue que buscará el camino hasta encontrar a la unidad civil seleccionado, estos robots no poseen capacidad de combate.

Para las misiones de Extracción de recursos se utilizará un robot ChapinFighter que buscará una unidad de recursos dentro del mapa seleccionado, estos robots se deberán enfrentar a unidades militares durante su trayecto reduciendo así su capacidad de combate.

TDA

Un Tipo de dato abstracto (en adelante TDA) es un conjunto de datos u objetos al cual se le asocian operaciones. El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en como están implementadas dichas operaciones. Esto quiere decir que un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.

El paradigma de orientación a objetos permite el encapsulamiento de los datos y las operaciones mediante la definición de clases e interfaces, lo cual permite ocultar la manera en cómo ha sido implementado el TDA y solo permite el acceso a los datos a través de las operaciones provistas por la interfaz.

Memoria dinámica

Se refiere a aquella memoria que no puede ser definida ya que no se conoce o no se tiene idea del número de la variable a considerarse, la solución a este problema es la memoria dinámica que permite solicitar memoria en tiempo de ejecución, por lo que cuanta más memoria se necesite, más se solicita al sistema operativo. El sistema operativo maneja la memoria gracias al uso de punteros, por la misma naturaleza del proceso nos impide conocer el tamaño de la memoria necesaria en el momento de compilar.

Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definir la cantidad de memoria a utilizar en el momento de compilar el programa.

XML

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

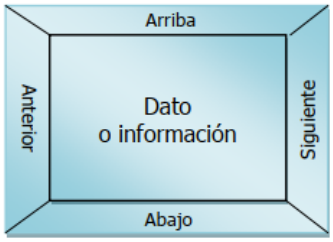
El lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. El lenguaje XML proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado.

Un archivo XML se divide en dos partes: prolog y body. La parte prolog consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios. La parte del body se compone de dos partes: estructural y de contenido (presente en los textos simples).

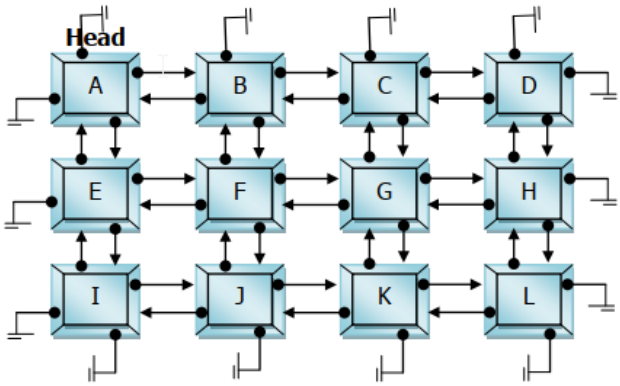
El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web. Tanto es así que hay sistemas destinados a ayudar en la definición de lenguajes basados en XML, así como APIs que ayudan en el procesamiento de datos XML - que no deben confundirse con HTML.

Lista Ortogonal o Matriz Dispersa

La característica principal de una lista ortogonal lineal es que los enlaces de los últimos nodos apuntan hacia un valor nulo. El nodo de la lista ortogonal debe contener como mínimo cinco campos: uno para almacenar la información y cuatro para guardar la dirección de memoria hacia el siguiente, anterior, arriba y abajo.



Unidos todos los nodos tomaría la siguiente forma:



Algoritmo A*

A* es un algoritmo de búsqueda inteligente o informada que busca el camino más corto desde un estado inicial al estado meta a través de un espacio de problema, usando una heurística (Método para aumentar el conocimiento) óptima, ignorando los casos que no cumplan con el camino más óptimo.

Para ellos durante cada movimiento se hace una estimación para determinar cuál de los vecinos de nuestro nodo actual es el siguiente al que debemos movernos. Para esto nos apoyamos de la siguiente fórmula:

$$F(n) = G(n) + H(n)$$

Donde:

G(n): es el costo de las movidas realizadas, se calcula en base a la distancia del nodo actual hasta el nodo vecino.

H(n): es la función heurística. Representa el costo o distancia estimada e ideal desde nuestro nodo actual hasta nuestro nodo de destino si no hubiera ningún obstáculo.

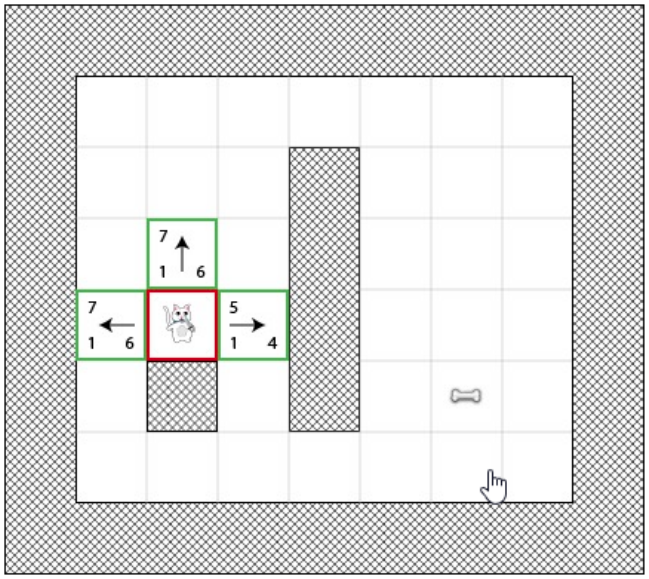
F(n): siendo la suma de G y de H que luego se comparara que con el cálculo en los otros nodos para determinar cuál de los 4 es el menor.

La fórmula que se utiliza para G(n) y H(n) es la siguiente:

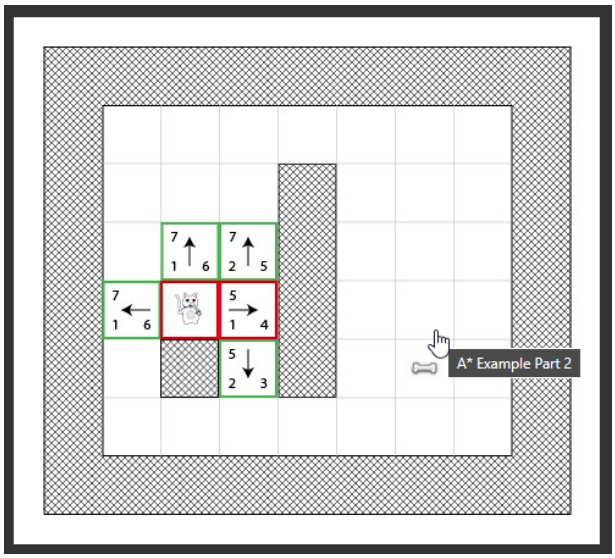
G(n) = |X2-X1| + |Y2-Y1|
H(n) = |X2-X1| + |Y2-Y1|

También conocida como la fórmula de distancia de Manhattan para medir la distancia entre dos puntos en un plano cartesiano.

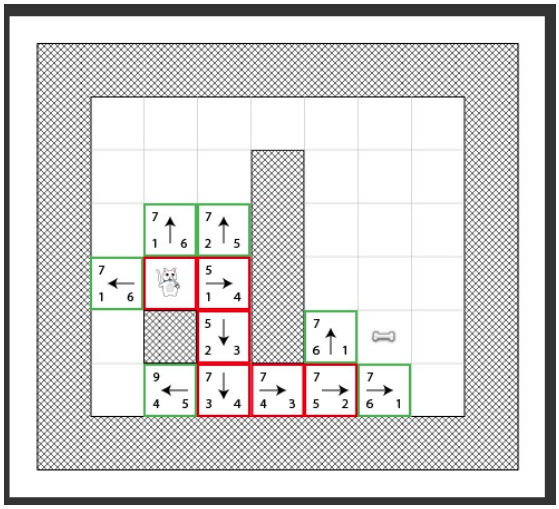
La manera en que el algoritmo funciona es la siguiente: Supongamos que queremos partir de un punto A hacia un punto B.



Para ello utilizamos las formulas anteriormente mencionadas y para cada nodo vecino hacemos el cálculo, en este ejemplo G(n) da como resultado 1 para todos los nodos ya que están a 1 posición de distancia del nodo actual mientras que para H(n) varía ya que algunos están más cerca o más alejados del nodo vecino.
La suma de ambos servirá luego para comparar las 3 direcciones ignorando los obstáculos para determinar hacia qué dirección debemos dirigirnos.



Continuando con este proceso hasta llegar al final, pudiera darse el caso en el dos o más nodos tengan el mismo valor por lo que podremos elegir cualquiera de ellos y continuar de esta manera hasta llegar al final.



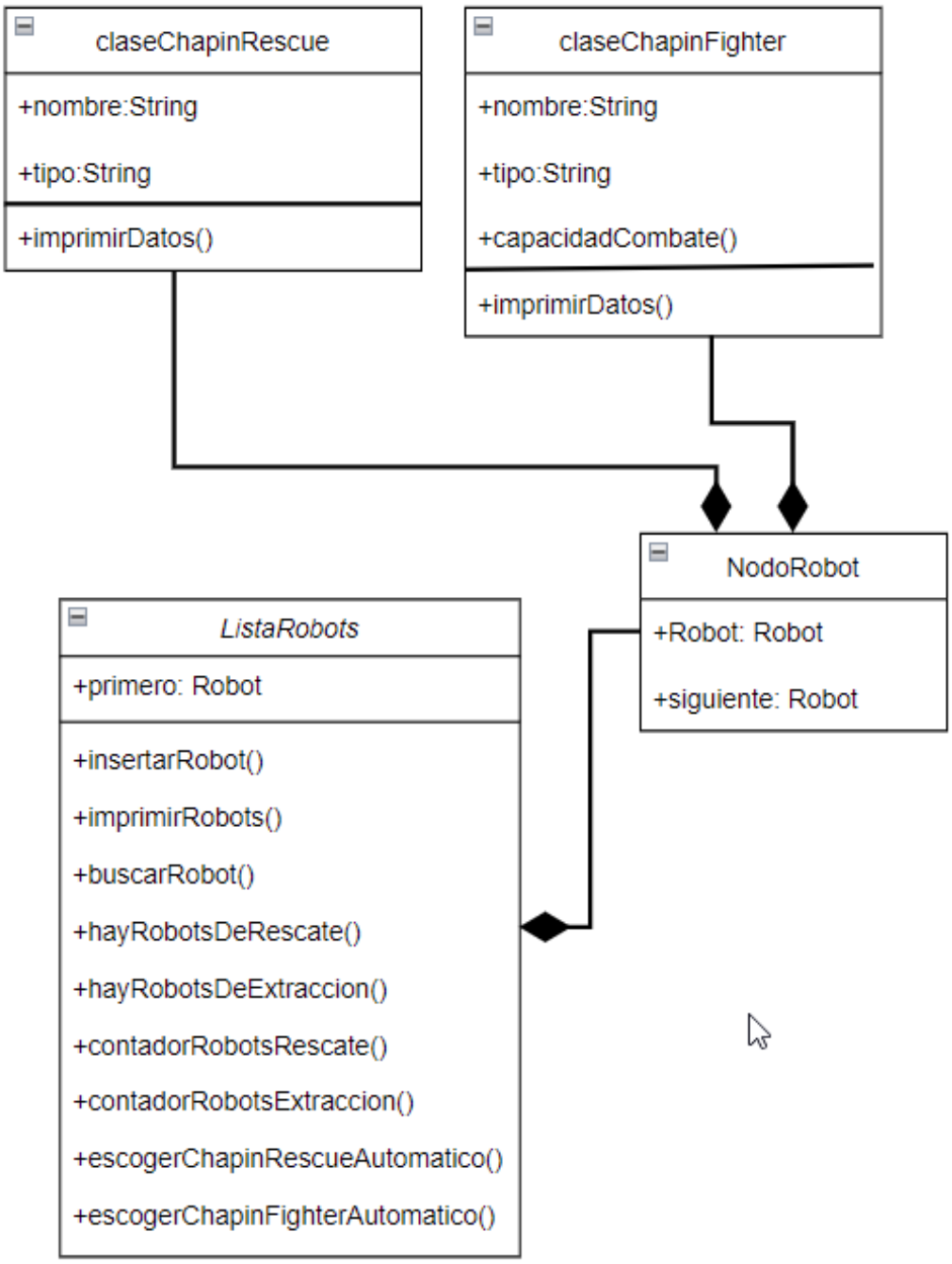
Conclusiones

- Para este proyecto se hicieron uso de estructuras de datos abstractas (TDA) utilizando el paradigma de la programación orientada a objetos, que permite hacer estructuras personalizadas que facilitan la resolución de este problema, estructurando las distintas partes del programa como módulos que se unen entre sí y son fáciles de modificar.
- Se utilizó como solución al problema el algoritmo A* que permite al robot encontrar el camino más óptimo hasta su destino.
- Este tipo de algoritmos tienen muchos usos en muchos campos del área de la informática, para dar solución a problemas complejos como este en el que se requiere que la solución sea la más óptima.

Referencias bibliográficas

- Problemas resueltos de listas / Mónica Adriana Carreño León 2010.
- CAIRÓ O., GUARDATI S., Estructura de Datos, México, Mc Graw-Hill, 2001.
- JOYANES A. LUIS, Algoritmos y Estructuras de Datos: Una Perspectiva en C., España, Mc Graw-Hill, 2005.
- Introduction to A* Pathfinding: Ray Wenderlich. Sep 29 - 2011.

Anexos


claseChapinRescue
+nombre:String
+tipo:String
+imprimirDatos()
claseChapinFighter
+nombre:String
+tipo:String
+capacidadCombate()
+imprimirDatos()
NodoRobot
+Robot: Robot
+siguiente: Robot
ListaRobots
+primero: Robot
+insertarRobot()
+imprimirRobots()
+buscarRobot()
+hayRobotsDeRescate()
+hayRobotsDeExtraccion()
+contadorRobotsRescate()
+contadorRobotsExtraccion()
+escogerChapinRescueAutomatico()
+escogerChapinFighterAutomatico()

Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2022.