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.

Robots de rescate Chapin Warriors S.A.

202001471 – Luis René Yaquian Recinos

Resumen

La aplicación consiste en encontrar un camino para poder rescatar o extraer recursos, dependiendo del tipo de misión se deberá de seleccionar el robot deseado para que se pueda cumplir la misión, dicha aplicación se espera que tenga una vigencia a largo plazo, pero siempre como cualquier algoritmo se puede optimizar para que sea más eficiente y así lograr encontrar el camino más optimo y seguro. Esa aplicación tendría un contexto internacional para que se puedan solucionar conflictos.

Para la aplicación en nivel técnico se pueden implementar muchas nuevas funcionalidades ya que se podría implementar que los ChapinFighter puedan evitar las pelas y si es necesario si enfrentarse, dichos cambios pueden realizarse dependiendo de las necesidades y los recursos que se deseen obtener, esto podrá evitar enfrentamientos ya así lograr rescatar a las personas y extraer los recursos de la mejor manera.

Palabras clave
Técnico, Optimización, Enfrentarse, Funcionalidades

Abstract

The application consists of finding a way to be able to rescue or extract resources, depending on the type of mission, the robot must be selected so that the mission can be fulfilled, the application is expected to have a long-term validity, but always like any Algorithm can be optimized to be more efficient and find the most optimal and safe path. That application would have an international context so that conflicts can be resolved.

For the application at the technical level, many new functionalities can be implemented since it could be implemented that the ChapinFighter can avoid the fights and if necessary if they face each other, these changes can be made depending on the needs and the resources that are desired to obtain, this will be able to avoid confrontations and thus manage to rescue people and extract resources in the best way

Keywords

Technical,Optimization,Confront,Features

Introducción

Se tiene la problemática que se tienen posiciones de rescate, caminos transitables, posiciones de inicio y posiciones de pela, se tienen distintos tipos de robots dependiendo de la misión que se desea ejecutar.
Para resolver dicho problema se crea un algoritmo para poder seleccionar la ciudad y el tipo de robot para poder cumplir la misión y se debe de mostrar el recorrido del robot y mostrarlo de manera gráfica el recorrido.

Desarrollo del tema

Para la solución de este problema se implementaron nodos y una matriz ortogonal la cual tiene como definición “una estructura de datos que implementa una tabla con memoria dinámica se puede buscar o recorrer por uno de los dos aspectos de orden”. Y se utilizaron listas de listas las cuales se definen como “una lista o secuencia es un dato abstracto que representa una secuencia ordenada de valores, donde el mismo valor puede ocurrir más de una vez. Un caso de una lista es una representación computacional del concepto matemático de una secuencia finita”. Y se implementó el algoritmo Dijkstra para poder encontrar el camino.

Para la apertura del archivo .xml se debe de importar xml.etree.ElemenrTree, para poder acceder al sistema se importó os y lo más importante para poder crear grafos se debe de importar Graphviz.

Se crearon distintas clases: Por ejemplo, se creó la clase crearIndiceVertical y crearIndiceHorizontal que era la que iba a recorrer la información que iba a ingresar por medio del archivo .xml. Se creó una clase insertarDato para poder ir ingresando los datos leídos del xml.

Se creó un tmp para poder utilizarlo cuando se desea obtener la información de este, al principio el tmp nos ayudará a extraer los datos y así poder ir intercambiando en las clases, agregando los datos en nodos y poder crear sub-nodos para poder acceder a los caminos ya sean transitables, intransitables, entrada, civil y recurso.

Los vecinos se calculan desde la posición inicial desde el punto de origen del terreno y el terreno que queremos recorrer. La función algoritmo se basa en el objeto tipo terreno.

En la calse nodo se agrega

-Visitando: Que nos indica que si un nodo ya fue tomado en cuenta por otro nodo.
-Acumulado: Es la que nos indica la cantidad de energía para llegar al nodo actual
-Padre: Es la encargada en indicarnos en el nodo al cual procede.

Se crearon métodos para poder extraer los datos y poder implementarlos como una matriz Ortogonal y así lograr recorrerla para poder extraer la posición y así poder cambiarla como es deseada.

Se implementó una lista enlazada para poder navegar entre los nodos y así poder acceder a los datos deseados.Se implemento el .siguiente y el .anterior para poder recorrer los datos, al inicio y al final se debe de tener un None ya que los datos del nodo al inicio no existe un dato anterior y al finalizar de recorrer el nodo no existirá nada ya que es una lista doblemente enlazada.

Al implementar el algoritmo de Dijkstra se debe de:
-Etiquetar: Se toma el nodo inicial y desde este nodo que se tomó se coloca el resto de las etiquetas para los nodos.
Se tiene [0,None](0) donde el primer cero es acumulado o es el gasto de energía que se hace para llegar a ese punto, el None representa el nodo padre, como se mencionó anteriormente es el que nos muestra de cual proceden. Y el ultimo digito nos muestra las iteraciones para llegar al punto.

Conclusiones

Se utilizó una matriz ortogonal para podernos desplazar entre los datos, ya que se necesita acceder a la posición de arriba, abajo, izquierda, derecha y así poder movernos entre la matriz.

Se creó una clase matriz ortogonal, la que será la encargada de extraer la información y poder ir insertando los índices Horizontales y Verticales. Para poder insertar estos índices se debe de recorrer el eje X e ir insertando los datos de manera Vertical el cual sería el eje Y.

Se pudo crear con listas enlazadas todo el proyecto, pero como se mencionó anteriormente necesitamos acceder a todas las posiciones se decidió utilizar la lista Ortogonal.

Referencias bibliográficas

Autor: Studios Cat| Sitio: Youtube | Título: IO Tutoriales - 02 Algoritmo de DIJKSTRA Fecha: Dic. 2013 | URL: https://www.youtube.com/watch?v=LLx0QVMZVkk&t=1s&ab_channel=StudiosCAT

Autor: Juan Chalita | Sitio: Definición ABC | Título: Implementando una lista doblemente ligada en Python.| Fecha: Nov. 2020 | URL: https://a01153884.medium.com/implementando-una-lista-doblemente-ligada-en-python-b8def2b8df73

Autores: C.onclase. | Sitio: Universidad de Granada| Título: Estructuras de datos | Fecha:| URL: http://c.conclase.net/edd/?cap=005#:%7E:text=Una%20lista%20doblemente%20enlazada%20es,siguiente%2C%20y%20otro%20al%20anterior.

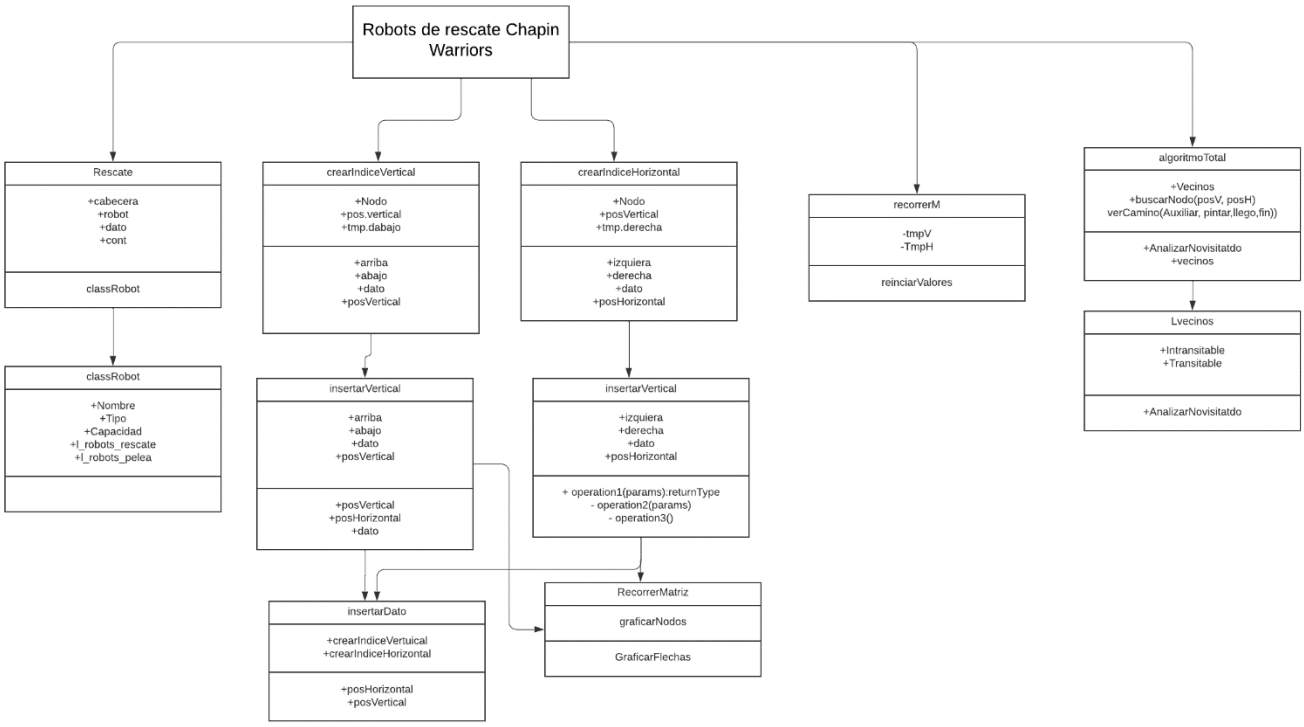

Figura 1. Diagrama de clases del programa.

Fuente: elaboración propia