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.

PROYECTO 2 IPC2 “CHAPIN WARRIORS APP”

201900042 – Rodrigo Alejandro Hernández de León

Resumen

El ensayo presenta la explicación del desarrollo de la aplicación utilizando una interfaz gráfica para que sea más fácil de utilizar para el usuario, donde presenta una simulación de las misiones de extracción de recursos, combate contra unidades militares, y rescate de unidades civiles de una ciudad seleccionada por el usuario, donde estos mismos datos son obtenidos por medio de la carga de archivos XML donde tiene la información de la ciudad y acerca de los robots que se tienen disponibles para las misiones donde pueden ser de tipo Chapin Fighter y Chapin Rescue y se declara como una misión completada cuando en el recorrido del robot encuentra a la unidad seleccionada y sin que las unidades civiles hayan destruido las unidades del robot Chapin Fighter, y muestra el recorrido de la misión hecha junto con los datos para la lectura del mapa de la ciudad y se pueda comprender los caminos a tomar para poder cumplir con las misiones de los robots.

Palabras clave

XML, robot, ciudad, recursos, entradas, unidades civiles, unidades militares, Python, Graphviz.

Abstract

The essay presents the explication of the development of the application using a graphical interface to make it easier to use for the user, where it presents a simulation of the missions of resource extraction, combat against military units, and rescue of civil units of a city selected by the user, where these same data are obtained by means of the load of XML files where it has the information of the city and about the robots that are available for the missions where they can be of type Chapin Fighter and Chapin Rescue and it is declared as a completed mission when in the route of the robot it finds the selected unit and without the civil units have destroyed the units of the robot Chapin Fighter, and shows the route of the mission done together with the data for the reading of the map of the city and it is possible to understand the ways to take to be able to fulfill the missions of the robots.

Keywords

XML, robot, city, resources, entries, civil units, military units, Python, Graphviz.

Introducción

Se desarrollo un programa el cual simula las misiones de la empresa Chapin Warriors donde por medio de un archivo XML se cargan las configuraciones para que la misión se desarrolle, teniendo en cuenta que existen diferentes ciudades y dos tipos de robot para realizar la misión como Chapin Fighter que realiza la extracción de recursos de la ciudad y combate contra las unidades militares, y Chapin Rescue que realiza las misiones de búsqueda de unidades civiles para rescatarlas de la ciudad y si es exitosa la misión entonces procede a mostrar el camino del robot para cumplir su misión. Para lograr todo esto se necesitó de la librería Tkinter con el lenguaje de programación Python, el uso de memoria dinámica con la matriz dispersa para cumplir con dar la solución que necesita la empresa de Chapin Warriors.

Desarrollo del tema

La aplicación consiste en simular las misiones de rescate y misiones de extracción de recursos para una ciudad seleccionada y que haya sido cargada al sistema de configuración para poder encontrar los caminos mas viables para completar las misiones.

Se ha optado por el uso del lenguaje de programación Python debido a que es multiparadigma y multiplataforma ya que su aplicación es sencilla de utilizar y se tuvo como complemento la ayuda de la librería de Tkinter para desarrollar una ventana interactiva al usuario.

Para la elaboración de la solución se provee un archivo con extensión y estructura XML, el cual provee los datos para la configuración de las misiones que quisiera realizar, donde dentro contiene la lista de ciudades y la forma en que esta constituida la ciudad, también contiene información de las unidades militares de la ciudad y por ultimo contiene los robots disponibles en la empresa para que puedan ser utilizados en su exploración. Para poder cumplir con las necesidades de las configuraciones en el programa se es necesario que cuando cargue el archivo tenga la siguiente estructura:

```
<?xml version="1.0"?>
<configuracion>
    <listaCiudades>
        <ciudad>
            <nombre filas="valorEntero" columnas="valorEntero">valorAlfanumerico</nombre>
            <fila numero="valorEntero">"***  **R****E***C**"</fila>
            ...
            <unidadMilitar fila="valorEntero" columna="valorEntero">valorEntero</unidadMilitar>
            ...
        </ciudad>
        ...
    </listaCiudades>
    <robots>
        <robot>
            <nombre tipo="ChapinFighter" capacidad="valorEntero">valorAlfanumerico</nombre>
        </robot>
        <robot>
            <nombre tipo="ChapinRescue">valorAlfanumerico</nombre>
        </robot>
        ...
    </robots>
</configuracion>
```

Figura 1. Estructura del archivo XML de entrada.

Fuente: Elaboración Propia, 2022.

donde:

configuracion = es la etiqueta padre.

listaCiudades = es la etiqueta que indica que existen ciudades dentro de la configuración.

ciudad = es la etiqueta que indica la ciudad con la que cuenta la lista de Ciudades.

nombre = es la etiqueta que indica el nombre de la ciudad.

filas = es el atributo que indica la cantidad de filas de la ciudad.

columnas = es el atributo que indica la cantidad de columnas de la ciudad.

fila = es la etiqueta que indica el contenido de la ciudad en esa fila donde:

‘E’ = representa un punto de entrada.

‘C’ = representa una unidad civil.

‘R’ = representa un recurso.

‘*’ = representa una celda intransitable.

‘ ‘ = representa una celda transitable.

unidadMilitar = es la etiqueta que indica la existencia de una unidad militar en la ciudad y se indica la capacidad de tropas de la unidad militar.

fila = es el atributo que indica la posición en fila que se encuentra la unidad militar.

columna = es el atributo que indica la posición en columna que se encuentra la unidad militar.

robots = es la etiqueta que indica todos los robots disponibles en el sistema.

robot = es la etiqueta que indica dentro las propiedades del robot.

nombre = es la etiqueta que indica el nombre del robot.

tipo = es el atributo que indica el tipo de robot que puede ser ChapinFighter o ChapinRescue.

capacidad = es el atributo que indica la capacidad de tropas del robot (únicamente aplica con los robots de tipo ChapinFighter).

Obtenido este formato, el usuario puede configurar su misión por medio de lo que contiene el archivo, el cual su contenido se guarda en las siguientes estructuras:

1. ListaCiudades: Aquí se almacena todas las ciudades cargadas del archivo XML.

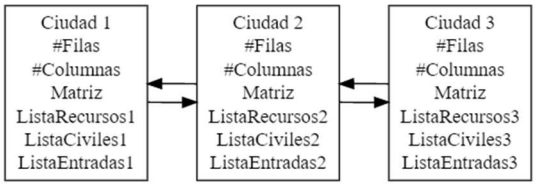

Figura 2. Representación grafica de la Lista Doblemente Enlazada de Ciudades.

Fuente: Elaboración Propia, 2022.

2. ListaEntradas: Aquí se almacenan todos los nodos donde sean las entradas de la ciudad.

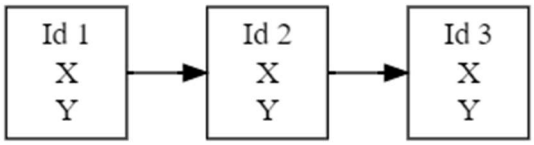

Figura 3. Representación gráfica de la Lista Enlazada Simple de Entradas de la ciudad.

Fuente: Elaboración Propia, 2022.

3. ListaRecursos: Aquí se almacenan todos los nodos que sean de tipo Recurso.

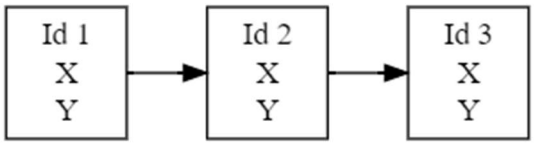

Figura 4. Representación gráfica de la Lista Enlazada Simple de Recursos de la ciudad.

Fuente: Elaboración Propia, 2022.

4. ListaRobots: Aquí se almacenan todos los robots que han sido cargados en el archivo XML.

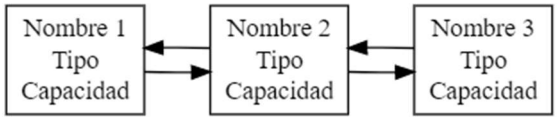

Figura 5. Representación grafica de la Lista Doblemente Enlazada de Robots.

Fuente: Elaboración Propia, 2022.

5. ListaUCiviles: Aquí se almacenan todos los nodos que sean de tipo Unidad Civil.

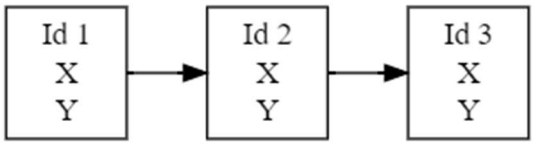

Figura 6. Representación gráfica de la Lista Enlazada Simple de Unidades Civiles de la ciudad.

Fuente: Elaboración Propia, 2022.

6. MatrizDisperza: Esta matriz está diseñada para almacenar el diseño del mapa de la ciudad. Para hacer esta matriz se necesitó de

2 listas de tipo cabecera junto con la creación de nodos celda para insertarlos en la matriz.

Con las estructuras ya creadas, se realizaron los siguientes algoritmos:

a. Procesar Archivo: Este algoritmo se encarga inicialmente de leer los archivos XML por medio de la librería Element Tree y verifica su el nombre de la ciudad esta registrado en el sistema y al no estar registrado en el sistema entonces crea la ciudad, de lo contrario sobrescribe el nodo perteneciente a ese nombre de ciudad. Tambien se agregan los nodos a sus respectivas listas y se almacenan los nodos de cada ciudad descrita.
b. Misión de Rescate: Este algoritmo se encuentra en la clase de la lista de ciudades donde en este mismo comienza creando una matriz auxiliar sin afectar la matriz original, y hace la verificación del entorno de cada nodo, tomando en cuenta si existen espacios transitables para poder pasar o de lo contrario tome otros caminos y si llega a un tope entonces se manda un mensaje que el robot no logró completar la misión de rescate.
c. Misión de Extracción de Recursos: Este algoritmo se encuentra en la clase de la lista de ciudades y se encarga inicialmente de crear una matriz auxiliar para trazar las rutas y verificar en donde inicia su misión y su nodo destino. Luego por medio de un ciclo while va verificando si el robot tiene paso o se encuentra con una unidad militar, en dado caso que se encuentre una unidad militar se resta la capacidad de unidades del robot con la capacidad de unidades de la unidad civil y si sale un numero negativo entonces se sale del ciclo y muestra un mensaje de misión fallida, de lo contrario el robot continua su camino en busca del nodo del recurso destino y poder completar la misión.
d. Graficar Matriz: Este algoritmo se encarga de utilizar la herramienta de Graphviz para graficar la matriz de la ciudad para que pueda ser mostrada en la ventana de la interfaz grafica.
e. Graficar Matriz de Rescate. Este algoritmo se encarga de graficar la matriz de la ciudad junto con el recorrido trazado por el robot, agregando también los datos de la misión de rescate.
f. Graficar Matriz de Extracción de Recursos: Este algoritmo se encarga de graficar la matriz de la ciudad junto con el recorrido hecho por el robot junto con los datos de la mision y la cantidad de capacidad de combate inicial que tenia el robot y su cantidad de combate final al terminar la misión de rescate.

Los algoritmos de extracción de recursos y de rescate contienen un pequeño fallo de poder ser obsoletos en algunos caminos para encontrar el nodo destino así que no es capaz de realizar todo tipo de misiones. Agregando que se intentó modular un nuevo algoritmo por medio del uso de pilas pero no se logró completar dicho algoritmo.

Conclusiones

Se cumplió con poder crear una interfaz amigable al usuario en donde pueda realizar todo tipo de configuraciones para que las misiones se cumplan sin tener inconvenientes con la misma.

Se aplicó una solución por medio de memoria dinámica para aplicar sus principios y conceptos de

matriz dispersa jugando con nodos y crear posibles caminos para ir al nodo destino.

Es importante tener los principios de conocimientos de listas para poder crear una matriz dispersa que es muy útil a la hora de manejar problemas de este estilo o de distintos problemas relacionados a la estructura de datos.

Es importante poder utilizar una interfaz grafica para que el usuario pueda ver e interactuar de forma mas sencilla las configuraciones del sistema.

Referencias bibliográficas

USAC, T.d.(7 de Marzo de 2022). *Proyecto 2, UEDi.* Obtenido de *https://uedi.ingenieria.usac.edu.gt/campus/pluginfile.php/283008/mod_resource/content/0/%5BIPC2%5DProyecto%20_2_1S2022-v2.pdf*

Anexos

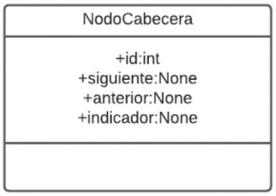

Figura 7. Modelo del tipo de dato NodoCabecera.

Fuente: Elaboración Propia, 2022.

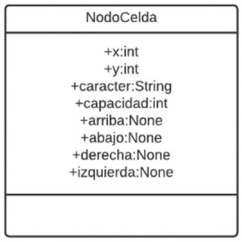

Figura 8. Modelo del tipo de dato NodoCelda.

Fuente: Elaboración Propia, 2022.

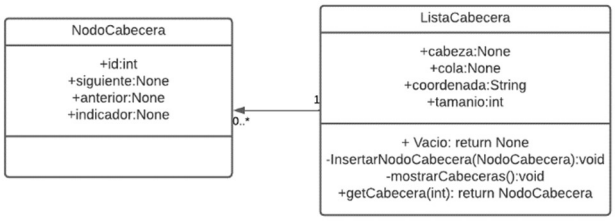

Figura 9. Modelo del tipo de dato ListaCabecera.

Fuente: Elaboración Propia, 2022.

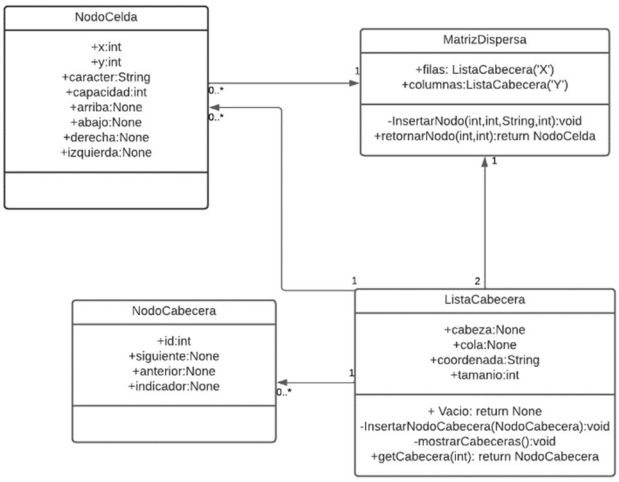

Figura 10. Modelo del tipo de dato MatrizDispersa.

Fuente: Elaboración Propia, 2022.

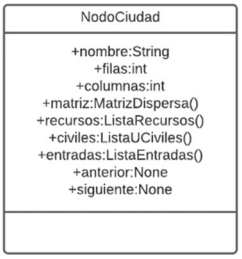

Figura 11. Modelo del tipo de dato NodoCiudad.

Fuente: Elaboración Propia, 2022.

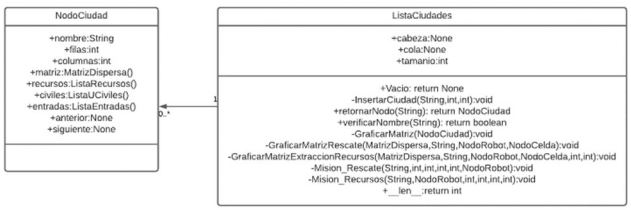

Figura 12. Modelo del tipo de dato ListaCiudades.

Fuente: Elaboración Propia, 2022.

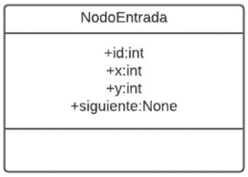

Figura 13. Modelo del tipo de dato NodoEntrada.

Fuente: Elaboración Propia, 2022.

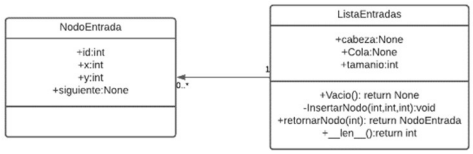

Figura 14. Modelo del tipo de dato ListaEntradas.

Fuente: Elaboración Propia, 2022.

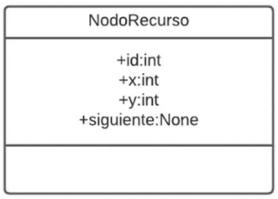

Figura 14. Modelo del tipo de dato NodoRecurso.

Fuente: Elaboración Propia, 2022.

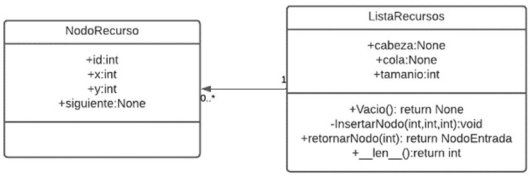

Figura 15. Modelo del tipo de dato ListaRecursos.

Fuente: Elaboración Propia, 2022.

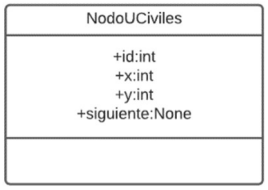

Figura 16. Modelo del tipo de dato NodoUCiviles.

Fuente: Elaboración Propia, 2022.

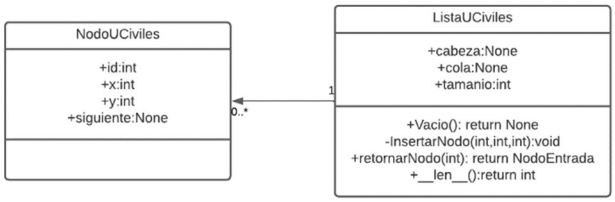

Figura 17. Modelo del tipo de dato ListaUCiviles.

Fuente: Elaboración Propia, 2022.

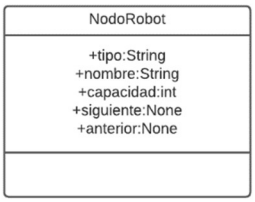

Figura 18. Modelo del tipo de dato NodoRobot.

Fuente: Elaboración Propia, 2022.

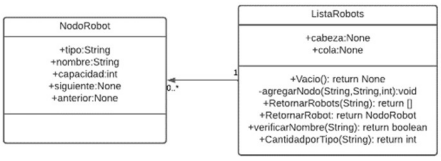

Figura 19. Modelo del tipo de dato ListaRobots.

Fuente: Elaboración Propia, 2022.

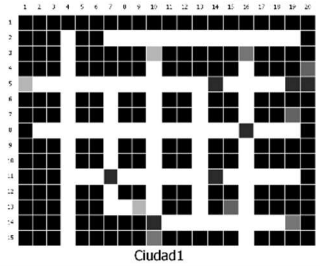

Figura 20. Mapa Generado de una ciudad.

Fuente: Elaboración Propia, 2022.

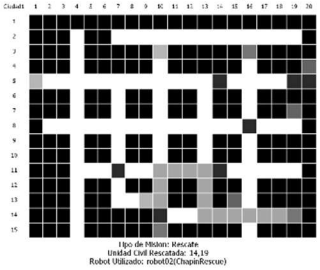

Figura 20. Ejemplo del recorrido de una misión de Rescate.

Fuente: Elaboración Propia, 2022.

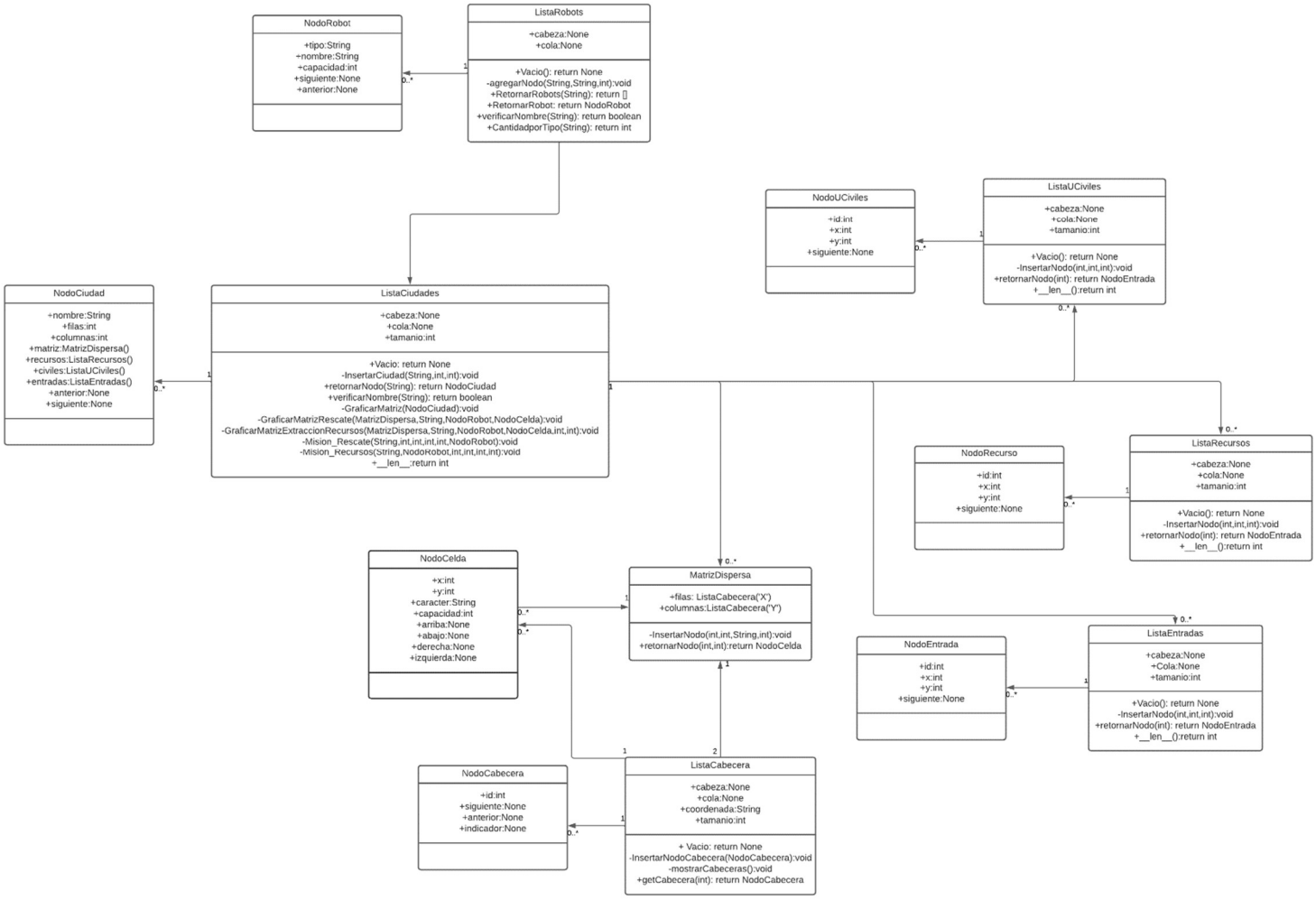

Figura 22. Diagrama de Clases del Sistema.

Fuente: Elaboración Propia, 2022.