PROYECTO 1: JUEGO BLOQUES

201904066 – Falkin Jarok Jarquin Roblero

Resumen

El proyecto es la simulación de un juego en cual su objetivo es colocar piezas en un tablero mxn representado por una matriz ortogonal.
El juego cuenta con un total de 6 piezas diferentes, y la posibilidad de elegir entre 4 colores para el jugador.

La matriz ortogonal se llevó a cabo por medio de 3 pasos, creación de un nodo raíz, creación de cabeceras y el posterior llenado de las cabeceras.

El juego fue desarrollado en el lenguaje de programación Python y posee una interfaz gráfica para su interacción usando la librería Tkinter.

Palabras clave

- Matriz ortogonal.
- Panel.
- Nodo.
- Lista.

Abstract

The project is the simulation of a game in which your objective is to place pieces on a mxn board represented by an orthogonal matrix.
The game has a total of 6 different pieces, and the possibility of choosing between 4 colors for the player.

The orthogonal matrix was carried out by means of 3 steps, creation of a root node, creation of headers and the subsequent filling of the headers.

The game was developed in the Python programming language and has a graphical interface for interaction using the Tkinter library.

Keywords

- Orthogonal matrix.
- Panel.
- Node.
- List.

Introducción

En la siguiente documentación se presentan aspectos de desarrollo sobre la aplicación bloques; se detallan las estructuras de datos utilizadas, su desarrollo, programación y posteriormente graficación.

Así también diagramas que presentan, detalles relevantes e incluso una breve descripción de uso de la aplicación.

Desarrollo del tema

Matriz ortogonal.

Se llevó a cabo por medio de 3 tipos de nodos.

Nodos cabecera, Nodo matriz y nodo ortogonal.

El nodo ortogonal era el que representaba la matriz, posee un nombre y dos apuntadores a listas cabeceras.

El nodo cabecera simulan los ejes de le matriz. Posee únicamente la coordenada y los apuntadores a los siguientes nodos cabecera.

El nodo matriz representa los valores guardados dentro de la matriz, el cual posee ya dos coordenadas (x,y) y el valor que almacena.

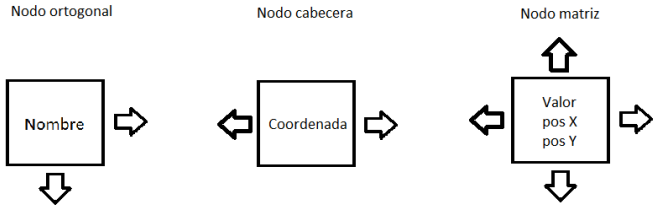

Figura 1. Tipos de nodos.

Fuente: elaboración propia

Con los nodos cabeceras se crearon **listas cabecera.** Para este caso se usaron dos listas para simular el eje X y el eje Y.

Figura 2. Lista cabecera.

Fuente: elaboración propia

Estos ejes se conectan al nodo ortogonal y crean la estructura base para la matriz.

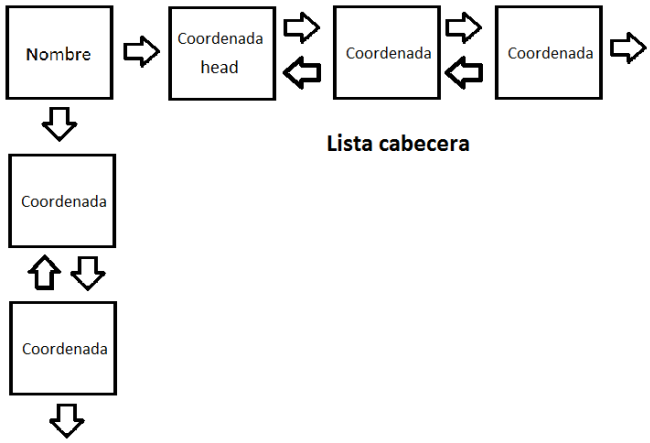

Figura 3. Estructura base de la Matriz.

Fuente: elaboración propia

Ahora solo queda llenar esta estructura.

Ahora la pregunta que surge es, ¿Cómo lo hago?, ya que son 4 apuntadores para cada nodo y cada uno podría apuntar a un nodo cabecera o a otro nodo matriz.

A continuación, presento una solución desarrollada por mi cuenta. La cual da solución al llenado de una matriz ortogonal.

Paso 1: Definir un tamaño.

Lo que necesitamos es saber cuántas columnas y cuantas filas usará nuestra matriz ortogonal (mxn).

Al tener estos números creamos la estructura de la *figura 3*, con las filas y columnas necesarias.

Paso 2: Columnas

Por cada nodo de la cabecera x creamos una lista de nodos matriz anidados. La cantidad de nodos debe ser igual al número de filas registrado.

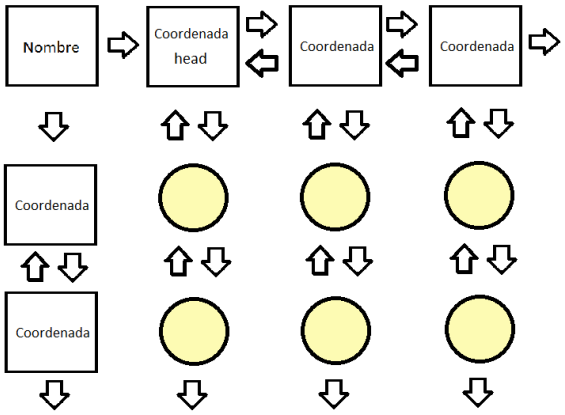

Figura 4. Columnas de nodos matriz.

Fuente: elaboración propia

Paso 3: Conectar la primera columna con las cabeceras Y.

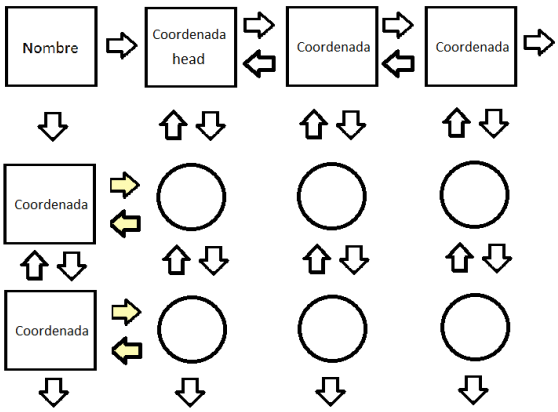

Figura 5. Apuntadores primera columna y cabecera Y.

Fuente: elaboración propia

Paso 4: conectar columnas

Por último para conectar columnas lo que hacemos es ir recorriendo cada nodo cabecera x con su siguiente nodo. De modo que usamos dos variables auxiliares.

Y en cada iteración o columna, también vamos recorriéndolos hacía abajo, y conectando su apuntador anterior y siguiente.

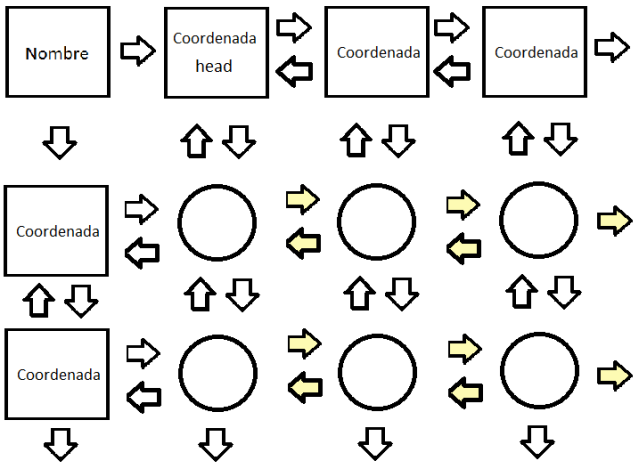

Figura 6. Matriz ortogonal completa.

Fuente: elaboración propia

Colocar piezas

Para colocar piezas se crearon dos funciones para la clase matriz ortogonal. “obtener_nodo” y “colocar_pieza”.

“obtener_nodo” es una función que nos permite traer un nodo por medio de sus coordenadas (x,y).

“colocar_pieza” por otro lado, es una función más compleja. Para funcionar requiere 4 valores, la coordenada en X y coordenada en Y donde se colocará el nodo principal de la figura, un valor que representa la figura y un color en le que se pintará la figura.

Las figuras se manejan por el código:

- Letra L = 1
- Letra L inversa = 2
- Barra horizontal = 3
- Barra vertical = 4
- Cuadrado = 5
- Piramide = 6

Y el color debe ser colocado en ingles y en minúsculas, de lo contrario la función notifica en consola y no cambia el dato.

La función funciona haciendo un recorrido por los nodos necesarios. Si encuentra un nodo NONE, o un nodo con valor diferente de “nulo” (con color), entonces lanza un error y no agrega la pieza.

Por el contrario, si el camino está libre, vuelve a tomar el nodo principal dado y ahora sí, va pintando uno por uno en el orden predeterminado.

A continuación, presento los recorridos que hacen algunas figuras:

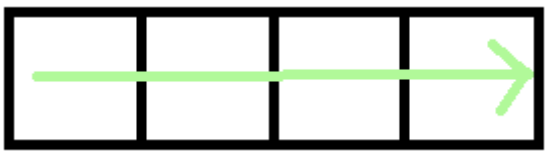

Figura 7. Recorrido de un rectángulo horizontal

Fuente: elaboración propia

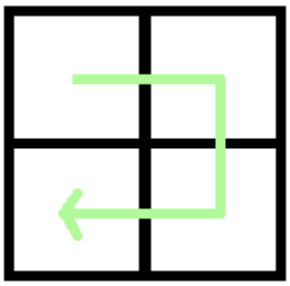

Figura 8. Recorrido de un cuadrado

Fuente: elaboración propia

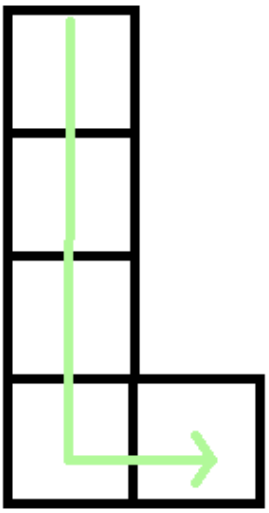

Figura 9. Recorrido de una L

Fuente: elaboración propia

Conclusiones

- Se aprendió sobre el manejo de memoria dinámica con clases nodos.
- Se desarrolló un algoritmo propio para llenar una matriz ortogonal.

- Se logró mostrar por medio del programa graphviz cómo es y cómo se conecta una matriz ortogonal de nxm

Comentario:

De este proyecto podemos obtener una visión más amplia de cómo trabajan los ordenadores, estamos acostumbrados a que la mayoría de las cosas ya estén hechas, ya existe documentación, tutoriales, herramientas, y claro, no inventaremos nuevamente la rueda. Pero saber cómo algo funciona, cómo está hecho, cómo se implementa. Permite desarrollar una mayor capacidad de abstracción, nos da más herramientas para enfrentar distintos problemas que se nos presentan.