Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, Vacaciones 2021.

PROYECTO 1 - BLOKUS

202000605 – Camilo Ernesto Sincal Sipac

Resumen

Blokus es un juego de mesa en el cual los jugadores colocan distintas piezas en ciertos campos de un tablero, el proyecto consistió en el desarrollo de su versión digital mediante el lenguaje de Python e implementando tipos de datos abstractos para la creación de una matriz dispersa y finalmente para el registro de memoria se usó XML y HTML.

La aplicación de una matriz dispersa tuvo la finalidad de tener un control eficiente de memoria, dado que esta trabaja con memoria dinámica, con lo cual la dicha memoria se va reservando y ocupando durante el desarrollo del juego.

Como se mencionó anteriormente para el registro de la información se implementaron los lenguajes de etiqueta XML y HTML, siendo la primera utilizada para el guardado y cargado de partidas y el último para el control de toda la información vital de la partida en juego, este se vio complementado con el uso de Graphviz para diagramar el estado de la matriz anteriormente mencionada.

Palabras clave
Matriz, XML, HTML, Dinámica

Abstract

Blokus is a board game in which the players place different pieces in a board, the project consisted in the develop of its digital version using Python, and implementing abstract data types to create a sparse matrix, and finally for the information record it was used XML and HTML.

The finality of the application of a sparse matrix was for an efficient control of memory, because it Works with Dynamic memory, whence it is being used in real time while playing.

As mentioned before for the information record it was implemented XML and HTML. The first one was used to save and charge different games and the second one for the control of vital information, this last one was complemented with Graphviz to diagram the state of the sparse matrix.

Keywords
Matrix, XML, HTML, Dynamic

Introducción

Un tipo de dato abstracto es un conjunto de datos que mediante su abstracción proporciona un conjunto de operaciones que obedece el comportamiento de estas abstracciones conocidas como estructuras de datos.

Siendo estas últimas estructuras que permiten organizar y controlar una gran cantidad y tipos de datos de manera eficiente y rápida sin perder rendimiento.

Estos conceptos unidos al encapsulamiento que ofrece la programación orientada a objetos y el manejo de clases permiten ocultar la el tipo de dato abstracto utilizado, de tal manera que el usuario únicamente pueda acceder a los datos mediante las operaciones definidas a dicha estructura.

La finalidad del proyecto fue comprender los conceptos anteriormente mencionados mediante la creación de una estructura conocida como matriz dispersa que permita el control de memoria de manera dinámica.

La finalidad de este ensayo es demostrar dicha comprensión en base a los algoritmos desarrollados para lograr la funcionalidad del juego.

Desarrollo del tema

a. Definición de nodos

Un nodo es un elemento que forma parte de una estructura de datos, estas cuentas varios campos, donde al menos uno de ellos guarda un apuntador o referencia de otro nodo, de tal manera que a partir del primer nodo se pueda desplazar al nodo que referencia

b. Matriz dispersa con nodos cabeceras

Una matriz dispersa consiste en una estructura que no contiene valores asignados a ninguna de sus posiciones o nodos, dado que estas se irán reservando y creando conforme el usuario las solicite, para su implementación en el desarrollo del juego cada nodo o valor correspondió a cada parte de una pieza, por lo cual un conjunto ordenado de estos valores forma una pieza completa.

Para la creación de la matriz se implementó el método de cabeceras, este consiste en la creación de dos listas doblemente enlazadas que nacen de un nodo principal para formar los ejes Y y X.

Ambas cabeceras tienen la función de ubicar distintos nodos o valores en posiciones especificadas por un usuario, teniendo en cuenta que independientemente de estas coordenadas estos nodos se ubicaran continuamente, aunque las coordenadas no sean continuas, por lo cual podrían ser visualizadas de la siguiente manera:

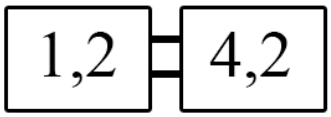

Figura 1. Nodos de una matriz dispersa.

Fuente: elaboración propia

Por lo cual al recorrer una matriz dispersa esta podrá acceder de la posición uno a la posición cuatro sin necesidad de recorrer espacios vacíos o nulos entre los dos valores como se muestra en la Figura 1.

c. Creación de piezas por el usuario

La creación de cada pieza del juego tuvo como base dos funciones, donde cada una dependía de otra función para cumplir con lo que se necesitaba, la primera de estas últimas mencionadas tiene la funcionalidad de verificar si existe un nodo en las coordenadas indicadas por el usuario, para esto primero se verifica que estas coordenadas se encuentren dentro del tablero y luego se recorre la cabecera X de la matriz hasta llegar a la coordenada indicada para verificar si esta tiene algún enlace, en caso de no tenerla este nuevo nodo se concatena inmediatamente al nodo de la posición en X indicada y con el valor de la cabecera Y correspondiente. Sin embargo, en caso de que ya exista un valor relacionado a la cabecera en X se recorre hacia abajo hasta que se encuentre un nodo con el valor de Y o hasta que no existan más nodos enlazados.

En caso de que se encuentre un nodo con el valor de Y solicitado se indica que esa posición no está disponible, caso contrario se retorna un valor verdadero que informa que no existe ningún dato en esas coordenadas solicitadas y por ende hay espacio para poder ingresar un nuevo nodo.

Esta función complementa a otra que se encarga de recibir como parámetro las posiciones de origen de la figura que se desea colocar, para posteriormente moverse a todas las posiciones que en conjunto constituyen a la pieza deseada, cada posición se verifica y guarda en una variable y en caso de que todas tengan un valor verdadero la función retorna un valor verdadero que posteriormente servirá para poder colocar la pieza.

La otra función complementaria es similar a la primera, con la diferencia de que esta recibe un nuevo parámetro, el cual es el identificador del jugador que desea colocar la pieza, esta función se encarga de determinar si un par de coordenadas están ocupadas por un nodo del usuario que desea colocar la pieza mediante la comparación del dato almacenado en el nodo. Esta complementa a otra función que de manera similar al del comprobante de espacios vacíos recibe las coordenadas iniciales del usuario, para posteriormente recorrer todos los lados adyacentes comprobando su disponibilidad con la función complementaria y guardando dicho resultado en distintas variables, de tal manera que al culminar con el recorrido si todos los valores son verdaderos se retorna otro valor verdadero para indicar que no existen piezas del mismo jugador a los lados de la figura.

Cada una de estas funciones se ve involucrada en una nueva función, la cual se encarga de recibir las coordenadas iniciales, el identificador de la figura a insertar, así como el jugador que desea insertar la pieza. Esta función llama a las dos anteriormente mencionadas para comprobar los lados y el espacio que ocupa la pieza, en caso de que ambos sean verdaderos se insertan los nodos en la matriz dispersa al igual que en la tabla que funciona como interfaz gráfica con el color correspondiente con al jugador que ingreso la pieza.

d. Interfaz Gráfica

Para la presentación gráfica de la interfaz se implementó el modulo de pyQT5 junto al drag and drop QtDesigner, con la combinación de ambos se representó el tablero de juego mediante una tabla la cual al igual que toda la parte de la interfaz únicamente funciona para representar visualmente las operaciones y los nodos de la matriz dispersa, recibiendo las posiciones que estas guardan para su

ubicación en la tabla logrando de esta manera establecer el color elegido por dicho jugador.

De igual manera junto a esta tabla se colocaron un conjunto de botones y etiquetas que funcionan para guiar al usuario en la creación de la tabla y la inserción de cada pieza.

e. Guardado y Cargado de partidas

Para el grabado de la información de cada partida se utilizó XML, un metalenguaje que mediante etiquetas permite dividir en diferentes secciones un grupo de información.

En este proyecto se indicó que el registro de las piezas colocadas consistiría en un texto que tendría valores de unos y dos para indicar quien colocó la pieza y guiones como representantes de los espacios disponibles.

Para el guardado de la partida se creó una función que recorría el nodo cabecero en Y, de tal manera que por cada uno de estos iniciaba un recorrido hacia la derecha para evaluar los nodos introducidos.

Dado que la matriz dispersa no contiene valores que indiquen directamente cuantos espacios vacíos hay al principio al final o entre nodos, la función verifica si el primer nodo corresponde al de la primera posición en X y en caso de que no lo sea toma el valor de la posición en X asignada al nodo y le resta una unidad para luego entrar en un bucle agregando la cantidad de espacios que sean necesarios según la resta.

Posteriormente se sigue avanzando en la lista de nodos al mismo tiempo que se realizan restas entre sus posiciones para determinar si existen espacios entre esto. Esto se logra verificando si la resta es mayor que uno, dado que esa es la condición que indica que deben existir espacios entre nodos.

Finalmente verifica si la posición en X del último nodo corresponde al final de la cabecera del mismo eje, en caso de no cumplir esta condición se realiza una resta entre la posición del nodo evaluado y el valor de la última posición de la cabecera para determinar cuántos espacios existen y luego escribirlos.

Para la lectura de las partidas se utilizó la librería de xml.dom, la cual permite determinar el texto que hay entre etiquetas específicas.

Dado que cada fila se escribe luego de un espacio en blanco se utilizó un split para separar cada fila, posteriormente se recorre cada fila y el valor de cada uno de los valores de texto, y al encontrar un valor numérico se guardan los iteradores correspondientes a fila y columna utilizado en el bucle para crear un nodo en esa posición y posteriormente representarlo en la tabla.

Conclusiones

Dado que el manejo y control de la memoria de distintos dispositivos forma parte vital de su rendimiento el implementar estructuras que permitieran un control de memoria dinámica tiene un gran peso positivo, ya que logra generar mejores resultados que el uso de memoria estática.

De igual manera es importante mencionar que el uso de XML permite un control sencillo y limpio a la vez que eficiente de información básica, pero no tan segura si uno se dirige al apartado de seguridad.

Anexos

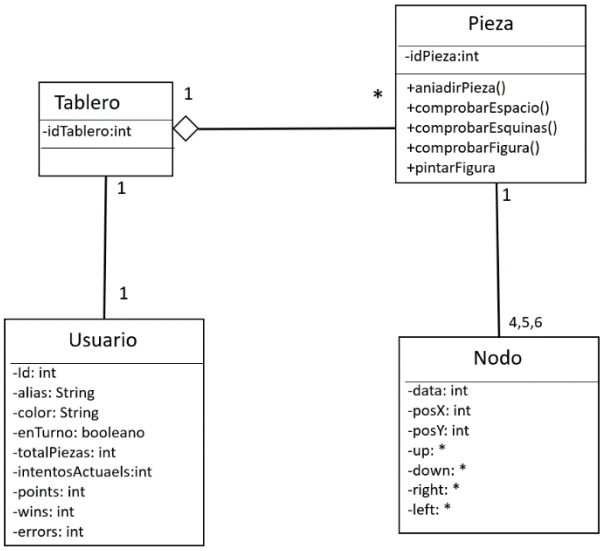

*Figura 2.*Diagrama de clases.

Fuente: elaboración propia

Referencias bibliográficas

Anónimo (Indefinido). *Tipos de datos abstracto*

C. Hektor, (2018) *Primeros pasos en PyQt5 y QtDesigner: Programas gráficos con Python.*

G. Yennifer, (2020). *Estructuras de datos: Descripción, ejemplos y más.* Tecnoinformática

V. F Miguel, (2015) *Matrices Dispersas.* CIMAT