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

PROYECTO N.1 JUEGO “BLOCKS"

201709196 – Allan Josué Rafael Morales

Resumen

El proyecto consiste en hacer uso del lenguaje de programación Python, para la creación de un mini-juego, esto se logra al graficar por medio del sistema. Un juego de dos personas, para lograr conseguir el mayor punteo y así ganar la partida, consta de un tablero en el cual deberán colocar bloques, mientras más logra colocar uno de los jugadores mayor será su puntaje.

Es posible modificar imágenes a discreción de los jugadores, actualizando la interfaz gráfica de forma automática y natural. Esto se logra por medio de la librería Tkinter, creando una interfaz grafica intuitiva para el usuario, agregando distintos colores a los bloques los cuales son los elementos más importantes del juego. Se utilizaron librerías como ElementTree y graphviz las cuales facilitan la lectura y construcción de los xml y las imágenes como grafos. El usuario podrá generar archivos xml para crear un reporte de la ultima partida, y volver a cargar esta cuando quiera para reanudarla.

Palabras clave

XML: Archivo el cual se estructura con etiquetas anidadas las cuales pueden contener atributos. Grafo: Figura referente a mapas de interconexión de datos. Librerías: Conjunto de archivos que se utilizan para facilitar la programación.

Abstract

The project consists of making use of the Python programming language, for the creation of a mini-game, this is achieved by graphing through the system. A game of two people, to achieve the highest score and thus win the game, consists of a board on which they must place blocks, the more one of the players manages to place the higher their score will be.

It is possible to modify images at the discretion of the players, updating the graphical interface automatically and naturally. This is achieved through the Tkinter library, creating an intuitive graphical interface for the user, adding different colors to the blocks, which are the most important elements of the game. Libraries such as ElementTree and graphviz were used which facilitate the reading and construction of xml and images as graphs. The user can generate xml files to create a report of the last game, and reload this whenever he wants to resume it.

Keywords

XML: File which is structured with nested tags which can contain attributes.

Graph: Figure referring to data interconnection maps.

Libraries: Set of files used to facilitate programming.

Introducción

Se busca disminuir el espacio en memoria que se utiliza con los datos manejados, haciendo que los datos se almacenen en una matriz dispersa para un óptimo manejo de datos.

Se intenta dejar en claro y de forma simple el manejo de diferentes datos para si manejo más optimo posible de las posiciones, y su operación simple para modificar los patrones dentro de la imagen.

Se implementan librerías para procesar y escribir archivos xml y png para los reportes respectivos de las partidas los cuales se van actualizando automáticamente después de cada modificación en la gráfica.

Además, se utilizó la librería Tkinter para la realización de la interfaz gráfica, la cual, aunque es una librería fácil de usar tiene ciertas complicaciones para operar imágenes y el manejo de métodos para los botones, así como la manera para actualizar estos después de cada modificación realizada por los usuarios jugadores al colocar piezas sobre un tablero.

Desarrollo del tema

Se implementaron TDA's los cuales almacenan todo el contenido obtenido de los archivos, se manejaron ciclos que permiten que el programa siga en ejecución después de realizar cualquier acción.

Se implemento la matriz con TDA's anidados para evitar el uso de vectores, definiendo otro TDA el cual contiene las matrices implementadas.

Para evitar la complejidad de las clases se implementaron varios métodos y funciones en cada TDA para hacerlos mas simples al momento de operar y de acceder a estos, además de esto se implementaron varios métodos dinámicos para la cual la operación de las imágenes sea mucho más optima.

Se accedió al contenido del xml con la librería ElementTree, obteniendo cada matriz continuando con la iteración en esta para obtener su valor y atributos contenida en esta.

Se implementó un método en el cual se crea una matriz NxM la cual se va llenando buscando la posición X y Y de cada dato en el TDA matriz, esto si el dato que se analiza corresponde a un carácter indicado de información, esto evita que haya datos innecesarios en la matriz, además de verificar que no se introduzca una posición fuera del rango del tablero generado por el usuario.

Cada método retorna una matriz ortogonal la cual se reemplaza por la matriz anteriormente generada, esto evitaría tener que modificar la matriz ortogonal solamente sustituyendo los elementos, evitando que se pueda corromper algún nodo o apuntador de dicha matriz.

Para creación del HTML se usó una librería de escritura que se encuentra incluida en el entorno virtual de Python, librería la cual facilita la escritura de este archivo, además de tener métodos específicos para la estructura correcta del archivo escrito. De otra manera se estaría escribiendo como un texto en el archivo, haciendo la comprensión de este poco amigable para el usuario.

En la creación del grafo de la matriz que se genera por medio de la agregación de nodos, se utilizó la librería graphviz, esta facilita la creación de dichos nodos y del correcto manejo de celdas o posiciones en el tablero que es visible para el usuario en todo momento. Esto se implementa de manera simple y proporciona la vista grafica de las matrices en tiempo real dentro de la aplicación.

Matriz_1	1	2	3	4	5	6	7	8
1								
2		*	*	*		*	*	*
3			*			*	*	*
4			*			*		
5		*	*	*		*		
6								
7		*	*	*		*	*	*
8		*						*
9		*	*	*		*	*	*

Figura 1. Grafo de una matriz.

Fuente: elaboración propia.

Se optó por realizar la mayoría de los métodos utilizados, en clases separadas y dinámicas lo cual facilitó en gran medida la complejidad de operaciones en las imágenes. Esto evita que surjan problemas de referenciación de nodos y confusión en la matriz creada.

Se realizó un gráfico con la ayuda de graphviz, utilizando su entorno para graficar nodos, debido a la estructura en la que se trabajo cada lista y cada nodo, por lo tanto se menciona que siempre habrá una mejor manera que optimizaría la manera en escribir el código que grafica dicho grafo.

Se utilizó programación orientada a objetos para facilitar el almacenamiento de datos y para simplificar los bloques de código.

Conclusiones

Se intento crear un programa lo más simple posible, con una forma ligera de procesamiento de nodos dentro de una matriz ortogonal, procurando la correcta utilización ideando una metodología de programación orientada a objetos.

Se evidenció que existen muchos problemas al no referenciar los nodos de forma correcta haciendo que estos sean un problema al realizar ciertos ciclos dentro de los métodos, además de problemas al momento de hacer referencias a widgets de Tkinter, dificultando el hecho de no recibir errores tan precisos los cuales son un tanto difíciles de identificar si no se conoce el funcionamiento correcto de estos.

Se recomienda utilizar mensajes de traza frecuentemente para obtener referencias de ejecución de ciertos valores y verificar que lo que se esta operando sea un valor y no una dirección de memoria.

Se optó por realizar gran parte de los métodos en clases especificar, y se procuró la creación de un código limpio en el cual se validaron las excepciones de forma simple y general.

Se espera que se verifiquen primeramente sitios con documentación oficial y en el idioma de origen para obtener información detallada y precisa del funcionamiento de librerías y del lenguaje mismo antes de buscar en sitios externos.

Se espera que el código sea implementado correctamente, también deberá estar correctamente versionado y comentado para no confundir diferentes segmentos de código que puedan ser similares.

Se recomienda realizar pruebas constantes para tener un punto de referencia para posibles fallas durante la construcción de la solución, esto evita la complejidad de búsqueda de errores, además evitara el uso excesivo de herramientas como el debugger del IDE utilizado, en este caso Pycharm.

Aunque siempre puede ser mejor optimizado el código y de mejor forma, se implemento lo que se creyó en su momento los adecuados y necesarios, pero siempre existe la posibilidad de hacerlo mejor.

Se recomienda hacer la documentación necesaria que ayude al fácil desarrollo de la solución, tales como diagramas de flujo, diagramas de clases, diagramas de entidades, etc. Siendo el principal problema con cierto grado de complejidad, se busco la forma más simple de solucionarlo.

Se utilizó el software de control de versiones GIT, el cual además se complementó con una integración de github en el IDE Pycharm, facilitando en gran medida la realización de commits, cuales tuvieron como objetivo principal mejorar el desarrollo de la solución.

Finalmente, en github se utilizo el manejo de control de releases para la publicación de versiones funcionales del programa, esto también ayuda a la comunidad y el propio desarrollador, o desarrolladores los cuales pueden obtener una retroalimentación de comentarios de los que han revisado el programa.

Referencias bibliográficas

Graphviz. (s. f.). *The DOT Language. graphviz.org.* Recuperado 15 de junio de 2021, de https://www.graphviz.org/doc/info/lang.html

Python. (s. f.). *tkinter — Python interface to Tcl/Tk.* Python.org. Recuperado 15 de junio de 2021, de https://docs.python.org/3/library/tkinter.htm

python.org. (s. f.). *xml.etree.ElementTree — The ElementTree XML API — Python 3.9.2 documentation. python.* Recuperado 15 de junio de 2021, de https://docs.python.org/3/library/xml.etree.elementtree.html

pypi. (2020, 24 diciembre). *Graphviz.* Recuperado 15 de junio de 2021, de, https://pypi.org/project/graphviz/#description