PIRATE CHASE

Algoritmos y estructuras de datos

Tarea integradora #3

Consolidación de los objetivos aprendidos durante el curso de algoritmos y estructuras de datos

Giovanni Mosquera Diazgranados | A00365672
Juan Pablo Sanin | A00296776
Juan Camilo Zorrilla Calvache | A00365972
Juan Sebastián Rodriguez | A00365843

Universidad Icesi

Enunciado:

Es el año 2021, y la compañía Nintendo quiere crear un prototipo para su nuevo juego Pirate Chase, desea que el prototipo simule las funcionalidades del juego en una escala más pequeña. Esta nueva saga dentro de la gran empresa trata de un pirata bueno que será nombrado por el jugador y un pirata malvado llamado Morgan. El objetivo es llegar al paraíso donde hay un cofre especial antes que Morgan pasando por diversas islas. El pirata bueno cuenta con una energía inicial que se verá afectada dependiendo del camino que tome para llegar al paraíso. Los trayectos entre isla tendrán un valor de energía que cuestan energía. Nintendo desea que haya niveles de dificultad que afecten la velocidad de Morgan y el tiempo que tiene el jugador para decidir a qué isla ir, además solicita que en las islas puedan haber distintos poderes, recarga de energía, ralentizador de Morgan, y aumento de tiempo para tomar una decisión. El jugador debe perder si Morgan llega primero, si se queda sin energía o se le acaba el tiempo para tomar la decisión y gana cuando llega al paraíso. Nintendo solicita en lo posible que las islas y el valor de los trayectos sean generados aleatoriamente. Además espera que el estilo gráfico vaya concorde a su marca y con pixel art.

Método de la ingeniería

Contexto Problemático:

Nintendo desea un prototipo para su nuevo juego Pirate Chase donde se implementan estructuras de datos de la manera más adecuada y que resuelva el problema en su totalidad.

Desarrollo de la solución:

Para resolver la problemática anterior se decidió implementar el Método de la Ingeniería para llevar a cabo la solución siguiendo las fases descritas en el *Resumen del capítulo 5 del libro Introduction to Engineering. Paul H. Wright. 3rd ed. Editorial John Wiley & Sons, Inc. 2002*. Brindado por el profesor Andrés Aristizabal en la programación del curso de Algoritmos y Estructuras de Datos.

Son 7 Fases que se llevarán a cabo:

Fase 1: Identificación del Problema
Fase 2: Recopilación de la Información Necesaria
Fase 3: Búsqueda de Soluciones Creativas
Fase 4: Transición de la Formulación de Ideas a los Diseños Preliminares
Fase 5: Evaluación y Selección de la Mejor Solución
Fase 6: Preparación de Informes y Especificaciones
Fase 7: Implementación del Diseño

Fase 1: Identificación del Problema

Para desarrollar esta fase se deben identificar necesidades reales del cliente.

Necesidades y Restricciones:

- *Se requiere la implementación de grafos.*
- *Es necesario implementar dos algoritmos de recorridos sobre grafos.*
- *Los grafos podrán ser ponderados o no ponderados.*
- *Se pide realizar 2 versiones de grafos, de tal forma que el programa pueda variar entre las versiones sin dañar su funcionamiento.*
- *La solución requiere una interfaz gráfica.*
- *Requerimientos solicitados en el enunciado:*
    - *El juego debe tener personaje principal*
    - *El juego debe tener personaje enemigo*
    - *El juego debe tener islas con valores entre ellas*
    - *El juego debe tener poderes*
    - *El juego debe tener dificultades*
    - *El juego permitirá ingresar el nombre del jugador*

Definición del Problema:

Nintendo requiere el desarrollo de un juego implementando estructuras de datos adecuadamente.

Fase 2: Recopilación de la información necesaria

Para poder empezar a pensar en cómo solucionar el problema decidimos indagar sobre qué elementos podríamos utilizar:

- **Definición de grafos:** Teoría de grafos Diagrama de un grafo con 6 vértices y 7 aristas. En matemáticas y en ciencias de la computación, la teoría de grafos (también llamada teoría de las gráficas) estudia las propiedades de los grafos (también llamadas gráficas). Un grafo es un conjunto, no vacío, de objetos llamados vértices (o nodos) y una selección de pares de vértices, llamados aristas (edges en inglés) que pueden ser orientados o no. Típicamente, un grafo se representa mediante una serie de puntos (los vértices) conectados por líneas (las aristas). (Tomado de: http://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf)

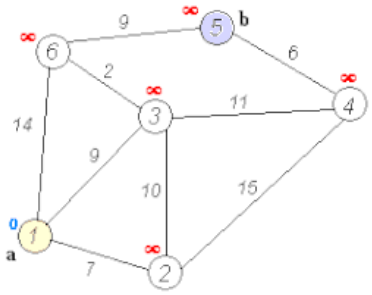
- **Vértice:** Los vértices constituyen uno de los dos elementos que forman un grafo. Como ocurre con el resto de las ramas de las matemáticas, a la Teoría de Grafos no le interesa saber qué son los vértices. Diferentes situaciones en las que pueden identificarse objetos y relaciones que satisfagan la definición de grafo pueden verse como grafos y así aplicar la Teoría de Grafos en ellos. (Tomado de: http://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf)

- **Aristas dirigidas y no dirigidas:** En algunos casos es necesario asignar un sentido a las aristas, por ejemplo, si se quiere representar la red de las calles de una ciudad con sus direcciones únicas. El conjunto de aristas será ahora un subconjunto de todos los posibles pares ordenados de vértices, con (a, b) ≠ (b, a). Los grafos que contienen aristas dirigidas se denominan grafos orientados, como el siguiente: Las aristas no orientadas se consideran bidireccionales para efectos prácticos (equivale a decir que existen dos aristas orientadas entre los nodos, cada una en un sentido). En el grafo anterior se ha utilizado una arista que tiene sus dos extremos idénticos: es un lazo (o bucle), y aparece también una arista bidireccional, y corresponde a dos aristas orientadas.
  (Tomado de: http://www.unipamplona.edu.co/unipamplona/portalIG/home_23/recursos/general/11072012/grafo3.pdf)

- **El algoritmo de Warshall:** El algoritmo de Warshall es un ejemplo de algoritmo booleano. A partir de una tabla inicial compuesta de 0`s (no hay correspondencia inicial en el grafo) y 1`s (hay una correspondencia, llamase “flecha”, entre nodos), obtiene una nueva matriz denominada “Matriz de Clausura Transitiva” en la que se muestran todas las posibles uniones entre nodos, directa o indirectamente. Es decir, si de “A” a “B” no hay una “flecha”, es posible que si haya de “A” a “C” y luego de “C” a “B”. Luego, este resultado se verá volcado en la matriz final.
  (Tomado de: https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall#El_algoritmo_de_Warshall)

- **Algoritmo de Dijkstra:** El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista. Su nombre alude a Edsger Dijkstra, científico de la computación de los Países Bajos que lo concibió en 1956 y lo publicó por primera vez en 1959.

  Tiempo de ejecución: $O(|E| + |V| \log |V|)$ (Peor caso)



  (Tomado de: Algoritmo de Dijkstra - Wikipedia, la enciclopedia libre)

- **Algoritmo de Kruskal:** Es un algoritmo de la teoría de grafos para encontrar un árbol recubridor mínimo en un grafo conexo y ponderado. Es decir, busca un subconjunto de aristas que, formando un árbol, incluyen todos los vértices y donde el valor de la suma de todas las aristas del árbol es el mínimo. Si el grafo no es conexo, entonces busca un bosque expandido mínimo (un árbol expandido mínimo para cada componente conexa). Este algoritmo toma su nombre de Joseph Kruskal, quien lo publicó por primera vez en 1956.12. Otros algoritmos que sirven para hallar el árbol de expansión mínima o árbol

recubridor mínimo son el algoritmo de Prim, el algoritmo del borrador inverso y el algoritmo de Boruvka.
(Tomado de: https://es.wikipedia.org/wiki/Algoritmo_de_Kruskal)

- **Algoritmo de Prim:** El algoritmo de Prim es un algoritmo perteneciente a la teoría de los grafos para encontrar un árbol recubridor mínimo en un grafo conexo, no dirigido y cuyas aristas están etiquetadas.

  En otras palabras, el algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol recubridor mínimo para uno de los componentes conexos que forman dicho grafo no conexo.

  El algoritmo fue diseñado en 1930 por el matemático Vojtech Jarnik y luego de manera independiente por el científico computacional Robert C. Prim en 1957 y redescubierto por Dijkstra en 1959. Por esta razón, el algoritmo es también conocido como algoritmo DJP o algoritmo de Jarnik.
  (Tomado de: https://es.wikipedia.org/wiki/Algoritmo_de_Prim)

- **DFS:** Depth First Traversal (or Search) for a graph is similar to Depth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, a node may be visited twice. To avoid processing a node more than once, use a boolean visited array.
  (Tomado de: https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/)

- **BFS:** Breadth First Traversal (or Search) for a graph is similar to Breadth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex.
  (Tomado de: https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/)

Fase 3: Búsqueda de las Soluciones Creativas

Para esta fase decidimos cada uno proponer una idea que podría solucionar el problema, implementamos la técnica lluvia de ideas donde nos limitamos a que debían ser implementaciones de estructuras de datos.

Utilizamos una plantilla de miro para facilitar esta actividad, donde cada uno generará 2 ideas que veían viables.

Fase 4: Transmisión de la Formulación de Ideas a los Diseños Preliminares

Falta

Fase 5: Evaluación y Selección de la mejor Solución
Falta

Miro con fases 3-5
https://miro.com/welcomeonboard/x8EyW3kTwQ9mpvXR65qfHGUuhEdTxOF5XjtsFyJT3HFSoT9zgQGoVRUr5oq7yo60

Fase 6: Preparación de Informes y Especificaciones
Falta

Fase 7: Implementación del Diseño
Falta

//Falta TAD