MÉTODO DE LA INGENIERÍA - LAB 2

Elías Estupiñan
Felipe Gómez
Óscar Riascos

Fase 1. Identificación del Problema

- Descripción del contexto del problema

  Minecraft es uno de los videojuegos más populares y exitosos de la última década. Perteneciente al género del Sandbox o mundo abierto, permite al jugador sumergirse en un mundo de posibilidades infinitas, dejando vía libre para la imaginación para crear su propia historia de fantasía.

  Siguiendo con lo anterior y, como es bien sabido, este mundo "dominado" por los bloques de elementos que, podemos encontrar en la naturaleza, permite crear desde objetos tan simples como una mesa, hasta objetos muchisimos mas complejos como pequeñas computadoras funcionales.

  Seguidamente, esa "libertad" que da Minecraft representa un desafío para los creadores del juego, ya que tienen que proporcionarles a los usuarios las mejores herramientas para que estos desarrollen todo su potencial creativo en la construcción por bloques, aplicando esto tanto a nivel gráfico (interfaces gráficas, plugins) como a nivel funcional (optimización). Sin embargo, hemos llegado a un punto donde la "demanda" de construcción por parte de los jugadores es tal, que, las estrategias pensadas desde un principio por los desarrolladores, quedaron obsoletas.

  En conclusión, por un lado, necesitamos encontrar una manera de optimizar los tiempos del inventario a la hora de acceder a este, con algoritmos más eficientes. Por otro lado, agregar nuevas funcionalidades al programa para que los usuarios puedan acceder a los bloques y así agilizar el tiempo de construcción dentro del juego.

- Definición del problema

  Minecraft a nivel funcional, usa un algoritmo que resulta ser muy ineficiente en el proceso de identificación de los bloques de construcción que se agregan al inventario. En cuanto a nivel operacional, resulta ser demasiado engorroso y lento en el proceso de construcción.

- Requerimientos funcionales

  - ❏ R1: Buscar e identificar de manera eficiente un bloque en el inventario.
  - ❏ R2: Crear una barra de acceso rápido específica para cada tipo de bloque.
  - ❏ R3: Agregar un bloque a la barra de acceso rápido.
  - ❏ R4: Alternar de manera intuitiva entre las distintas barras de acceso rápido.
  - ❏ R5: Eliminar un bloque del inventario.

- Requerimientos no funcionales

  - ❏ Diseño intuitivo
  - ❏ Fácil implementación
  - ❏ Manejo fácil e intuitivo

Fase 2. Recopilación de información.

- Definiciones

  - Bloque:
    Trozo grande y sin labrar de un material compacto y duro.

  - Inventario:
    El inventario es una relación detallada, ordenada de los elementos que se obtienen durante el juego, por ejemplo: los bloques, la comida, las armas etc.

  - Ventajas del inventario:
    Detalla y especifica las características de cada uno de los elementos que el jugador a almacenado.
    Ordena y agrupa los elementos del patrimonio en sus casillas correspondientes y la cantidad de elementos en la posición correspondiente.
    Puedo acceder fácilmente a los elementos guardados en el.

  - Tabla hash:
    Una tabla hash, matriz asociativa, hashing, mapa hash, tabla de dispersión o tabla fragmentada es una estructura de datos que asocia *llaves* o *claves* con *valores*. La operación principal que soporta de manera eficiente es la *búsqueda*: permite el acceso a los elementos (teléfono y dirección, por ejemplo) almacenados a partir de una clave generada (usando el nombre o número de cuenta, por ejemplo). Funciona transformando la clave con una función hash en un hash, un número que identifica la posición (*casilla* o *cubeta*) donde la tabla hash localiza el valor deseado.

❑ Función de resumen.

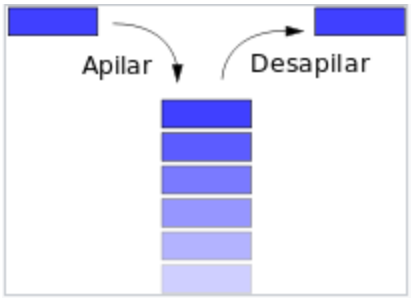
Los **hash** o funciones de resumen son algoritmos que consiguen crear a partir de una entrada (ya sea un texto, una contraseña o un archivo, por ejemplo) una salida alfanumérica de longitud normalmente fija que representa un resumen de toda la información que se le ha dado (es decir, a partir de los datos de la entrada crea una cadena que *solo* puede volverse a crear con esos mismos datos).

❑ Pila.

Una pila (stack en inglés) es una lista ordenada o estructura de datos que permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo LIFO (del inglés Last In, First Out, «último en entrar, primero en salir») . Esta estructura se aplica en multitud de supuestos en el área de informática debido a su simplicidad y capacidad de dar respuesta a numerosos procesos.

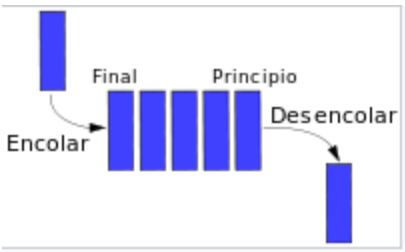
Para el manejo de los datos cuenta con dos operaciones básicas: apilar (push), que coloca un objeto en la pila, y su operación inversa, retirar (o desapilar, pop), que retira el último elemento apilado.

En cada momento sólo se tiene acceso a la parte superior de la pila, es decir, al último objeto apilado (denominado TOS, Top of Stack en inglés). La operación retirar permite la obtención de este elemento, que es retirado de la pila permitiendo el acceso al anterior (apilado con anterioridad), que pasa a ser el último, el nuevo TOS.



❑ Cola.

Una cola (también llamada fila) es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pull por el otro. También se le llama estructura FIFO (del inglés First In First Out), debido a que el primer elemento en entrar será también el primero en salir.



❑ Barra de acceso rápido:

La barra de acceso rápido es una barra que podemos personalizar con los elementos que deseamos utilizar en un momento dado, logrando así minimizar el tiempo de búsqueda de un objeto que esté almacenado en un inventario o lugar donde se encuentra contenido.

❑ Recursión:

Se denominan funciones recursivas a aquellas que se llaman a sí mismas. Por ejemplo un número factorial el cual se representa de esta forma:

$$n!=n\cdot(n-1)\cdot(n-2)...2\cdot1$$

- En seudocódigo seria:

```
procedure factorial(n: n ∈ □⁺)
if n = 0 then
        factorial(n) := 1
else
        factorial(n):= n*factorial(n-1)
```

- Iterativo:

  Una iteración significa repetir varias veces un proceso con la intención de alcanzar una meta deseada, objetivo o resultado. Cada repetición del proceso también se le denomina una "iteración", y los resultados de una iteración se utilizan como punto de partida para la siguiente iteración.

  Por ejemplo hallar el factorial de un número:

```
long factorial(int n){
    long resultado=1;
    for(int i=1; i<=n; i++){
    resultado*=i;
    }
return resultado;
}
```

- Estructura lineal:

  Los Arreglos, Listas, Pilas y Colas son estructuras de datos lineales. Esto significa que los elementos de las mismas se almacenan y/o acceden uno detrás del otro.

  Por ejemplo, en un arreglo, a la hora de añadir un elemento, el mismo se insertará al final de los elementos ya insertados, o entre dos existentes, pero siempre respetando que los elementos se encuentren y administren uno tras otro; esto es, su linealidad.

- Estructuras no lineal:

  Los árboles y grafos son ejemplos de estructuras de datos no lineales. Esto significa que sus elementos no se almacenan uno tras otro, o en su defecto, la forma de administrar los mismos no es lineal. Por ejemplo, en un árbol los elementos se insertan como hijos de otros, creando una jerarquía no lineal.

❑ Elementos clave

- ❑ Una estructura que permita el acceso de manera rápido y eficiente a un grupo de datos.
- ❑ Barras de acceso rápido a los bloques que se necesiten usar de manera continua o repetitiva.
- ❑ Transición rápida e intuitiva a las diferentes barras de acceso rápido.

Fase 3. Búsqueda de Soluciones Creativas

Utilizando el método de lluvia de ideas, se expondrán a continuación las posibles soluciones. Se implementará en forma de software la más efectiva y eficiente en su respectiva tarea. Adicionalmente se han clasificado en tres partes las funcionalidades que se tienen que implementar, esto, para identificar el caso que desarrolle mejor los problemas que presenta el videojuego.

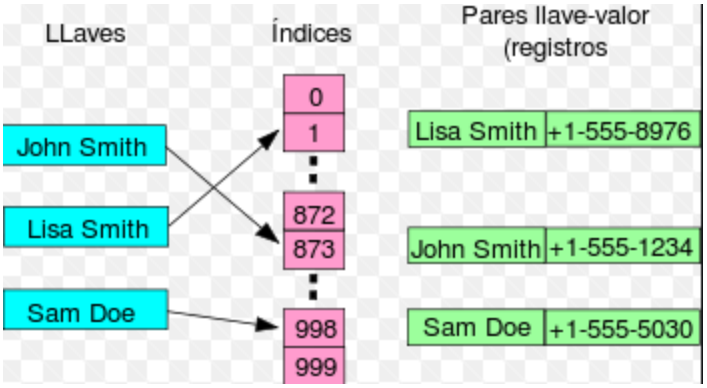
Soluciones para el manejo eficiente del inventario de bloques

- *Alternativa 1: Recorrer de manera iterativa las posiciones de una matriz contenedora de los bloques.*

  Esta alternativa permite recorrer la matriz de forma iterativa hasta encontrar el elemento deseado. Por tanto, si se busca un elemento en la estructura contenedora y esté no se encuentra, la iteración se realizaría $□^2$veces, es decir que recorrería toda la matriz sin sentido alguno.

- *Alternativa 2: Acceso a los bloques utilizando una estructura Hash Table.*

  Tabla Hash. Es una estructura de datos no lineal cuyo propósito final se centra en llevar a cabo las acciones básicas (inserción, eliminación y búsqueda de elementos) en el menor tiempo posible, mejorando las cotas de rendimiento respecto a un gran número de estructuras.



- *Alternativa 3: Recorrer de manera recursiva a los posiciones una matriz contenedora de los bloques.*

  Permite recorrer las posiciones de la matriz de forma recursiva.

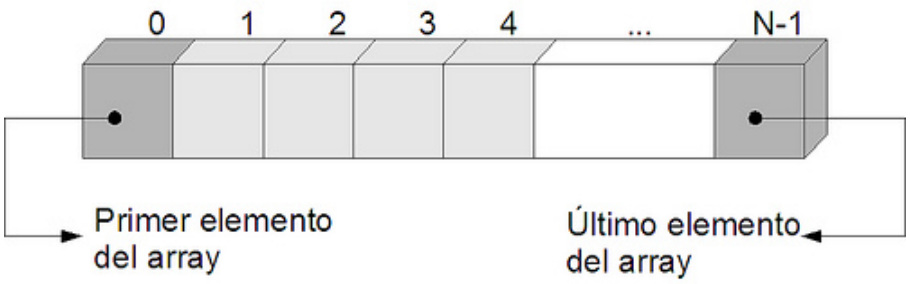
- *Alternativa 4: Arrastrar uno por uno los elementos, del mismo tipo, del inventario a la barra de acceso rápido.*

  Permite recorrer en reiteradas ocasiones la matriz en busca del elemento que el usuario desea agregar a la barra de acceso rápido. Por tanto, si dicho elemento está ubicados en diferentes posiciones el programa internamente buscará el mismo elemento tantas veces como el usuario lo arrastre.

Soluciones para el manejo eficiente de la barra de acceso rápido

- *Alternativa 1: Un array de bloques del mismo tipo.*

  Un array, es un tipo de dato estructurado que permite almacenar un conjunto de datos homogéneo, es decir, todos ellos del mismo tipo y relacionados. Cada uno de los elementos que componen un array puede ser de tipo simple como caracteres, entero o real, o de tipo compuesto o estructurado como son vectores, estructuras, listas...

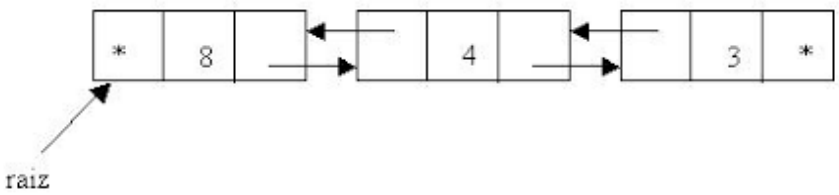


A los datos almacenados en un array se les denomina elementos; al número de elementos de un array se les denomina tamaño o rango del vector.

Para acceder a los elementos individuales de un array se emplea un índice que será un número entero no negativo que indicará la posición del elemento dentro del array. Para referirse a una posición particular o elemento dentro del array, especificamos el nombre del array y el número de posición del elemento particular dentro del mismo, el índice.

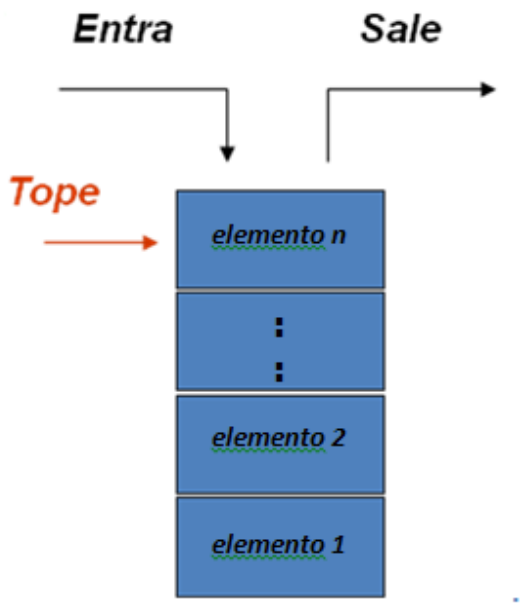
- *Alternativa 2: Una lista simplemente enlazada de bloques del mismo tipo.*

Se definen como un conjunto de nodos uno detrás de otro, del cual siempre se puede conocer al nodo inicial y al final, de cada nodo de la lista, se conoce un contenido, que es la información que almacena dentro puede ser de cualquier tipo de dato un sucesor único excepto el último nodo de la lista.



- *Alternativa 3: Una pila de bloques del mismo tipo.*

  Una Pila es una colección ordenada de elementos en la que se pueden insertar y suprimir por un extremo, llamado tope. Por tal razón se conoce como una estructura de datos LIFO (last-in, first-out).

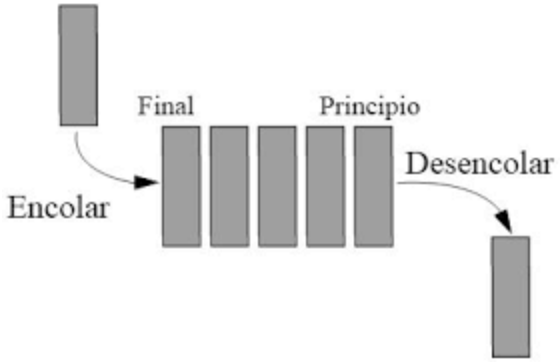


  Para nuestra solución, la pila permite acceder rápidamente a los bloques.

Soluciones para el manejo eficiente de la lista de barras de acceso rápido

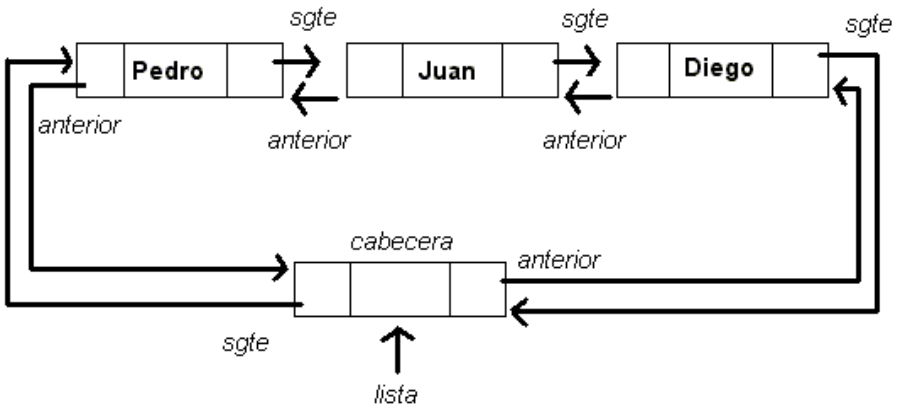
- *Alternativa 1: Una cola de barras de acceso rápido.*

  La cola es una colección ordenada de elementos de la que se pueden borrar elementos en un extremo (llamado el frente de la cola) o insertarlos por el otro (llamado el final de la cola). El primer elemento insertado en la cola, es el primero en ser eliminado; razón por la que a las colas se les conoce como una estructura de datos FIFO (por first-in, first-out)



- *Alternativa 2: Una lista circular simplemente enlazada de barras de acceso.*

  Una lista circular es una lista lineal en la que el último nodo apunta al primero. Las listas circulares evitan excepciones en las operaciones que se realicen sobre ellas. Cada nodo siempre tiene uno anterior y uno siguiente.



Pero, ¿Por qué utilizar una estructura Circular?

En una lista simplemente enlazada, el movimiento siempre fluirá desde la cabeza en dirección hacia el final de la lista, entonces ¿qué ocurre cuando desde el último nodo se necesita operar con el primero?, este es el punto diferencial de una estructura abierta y una cerrada.

En una lista circular:

- No existe algún elemento que apunte a NULL
- Se integra una estructura tipo anillo
- Solo hay una cabeza
- La cabeza siempre será el siguiente enlace para algún nodo
- Se pueden llegar a crear recorridos en bucles infinitos

Fase 4. Transición de las Ideas a los Diseños Preliminares

Antes de avanzar en el análisis y profundizar en las opciones que se seleccionaron previamente, primero se deberá descartar algunas de las ideas anteriormente descritas en la fase 3. Para esto, se realizará la misma clasificación de la fase anterior y se suprimen las ideas, inviables por algún motivo, de cada estás.

Soluciones a descartar para el manejo eficiente del inventario de bloques

- Alternativa 1: Recorrer de manera iterativa las posiciones de una matriz contenedora de los bloques. Se descarta esta opción, ya que en el peor de los casos recorrería toda la matriz en busca de un bloque en específico ocasionando lo cual tendría $\square^2$ recorridos hasta llegar a elemento deseado.

- Alternativa 3: Recorrer de manera recursiva a los posiciones una matriz contenedora de los bloques. Se suprime esta alternativa, puesto que presenta características similares a la opción uno, anexando que puede llegar a utilizar grandes cantidades de memoria en un instante, pues implementa una pila cuyo tamaño crece linealmente con el número de

recursiones necesarias en el algoritmo. Si los datos en cada paso es muy grande, podemos requerir grandes cantidades de memoria.

- Alternativa 4: Arrastrar uno por uno los elementos, del mismo tipo, del inventario a la barra de acceso rápido. Este caso se descarta de manera inmediata, dado que al arrastrar uno por uno los elementos del mismo tipo, el programa internamente estaría recorriendo la matriz tantas veces como el usuario arrastre los objetos, ocasionando gran cantidades de memoria y otros recursos del aparato electrónico, por esta razón, se usará otra alternativa que permitirá arrastrar todos los elementos del mismo tipo sin necesidad de recorrer varias veces la matriz.

Soluciones a descartar para el manejo eficiente de la barra de acceso rápido

- Alternativa 1: Un array de bloques del mismo tipo. Se descarta dado que, en cada posición del array, sólo se puede guardar un elemento, es decir, solo se podría guardar un solo bloque de un solo elemento en una sola casilla de la barra de acceso rápido. lo cual no servirá para hacer del programa mucho más eficiente en las tareas que se quieren mejorar.

- Alternativa 2: Una lista simplemente enlazada de bloques del mismo tipo. Se descarta ya que, por un lado, la listas simplemente enlazadas ocupan mucha más memoria que un array o ArrayList del mismo tamaño. Por otro lado, surgiría el mismo problema que en la alternativa 1.

Soluciones a descartar para el manejo eficiente de la lista de barras de acceso rápido

- Alternativa 2: Una lista circular simplemente enlazada de barras de acceso. Se descarta debido al almacenamiento extra necesario para las referencias a otros nodos de la lista, lo que conlleva al consumo extra de memoria RAM, lo cual, a su vez, es el principal problema que se quiere resolver. Siguiendo con lo anterior, las listas enlazadas en general, sólo permiten acceso secuencial con lo cual llevaría mucho más tiempo que utilizando otras alternativas.

Fase 5. Evaluación y Selección de la Mejor Solución

De acuerdo con el minucioso análisis que se realizó en las etapas preliminares de diseño, se ha llegado a la conclusión que las mejores alternativas en cada uno de los siguientes ámbitos serían:

1. Primeramente para el manejo eficiente del inventario de bloques se optó por la **alternativa 2: acceso a los bloques utilizando una estructura Hash Table.** En vista de que permite realizar las acciones básicas (inserción, eliminación y búsqueda de elementos) en el menor tiempo posible.
   Además, de lo anterior, por ser una estructura no lineal, sus elementos se almacenan en un arreglo, que es una estructura lineal, pero la administración de dichos elementos no se realiza de forma lineal, permitiendo así que se convierta en una estructura que tiene como fin minimizar el tiempo dedicado a sus principales acciones tales como insertar, eliminar y buscar.
   De igual manera, una Tabla Hash es una estructura que puede almacenar una gran cantidad de información y lograr para sus acciones principales en tiempo de ejecución $\Theta(1)$, ya que incorporan una función de hash que brinda a la acción de buscar de manera instantánea y constante. Esto supera las cotas comunes de las listas enlazadas, cuyos tiempos oscilan aproximadamente por $\Theta(N)$.

2. Soluciones para el manejo eficiente de la barra de acceso rápido

   Se tomará la **alternativa 3: una pila de bloques del mismo tipo,** ya que esta nos permite apilar rápidamente los elementos cuando nos desplazamos desde el inventario a la barra de acceso rápido para posteriormente usarlo más adelante.

3. Soluciones para el manejo eficiente de la lista de barras de acceso rápido

   Se eligió la **alternativa 1: una cola de barras de acceso rápido**, por la razón de que nos permite movernos de manera eficaz entre las diferentes barras de acceso rápido. Igualmente, permite añadir un nuevo elemento más sin límite, con la ventaja de que solo se pueden insertar en un extremo, lo cual, a su vez, se manifiesta en un uso menor de operaciones y/o recursos.

Referencias

- http://estr-datos-omar.blogspot.com/2011/10/listas-circulares.html
- https://javiergarciaescobedo.es/programacion-en-java/31-arrays/123-arrays
- http://decsai.ugr.es/~jfv/ed1/c/cdrom/cap5/f_cap52.htm
- https://users.dcc.uchile.cl/~bebustos/apuntes/cc30a/Estructuras/
- http://www.utm.mx/~dtorres/cursos/estructuradedatos/Tema1-Pilas.pdf
- http://www.utm.mx/~dtorres/cursos/estructuradedatos/Tema2-Colas.pdf
- http://www.itnuevolaredo.edu.mx/takeyas/Apuntes/Administracion_Archivos/Apuntes/Hashing.PDF
- https://www.genbeta.com/desarrollo/que-son-y-para-que-sirven-los-hash-funciones-de-resumen-y-firmas-digitales
- https://es.wikipedia.org/wiki/Pila_(inform%C3%A1tica)