Alejandro Martínez
Cristian Lasso
José David Jiménez
Ana María Muñoz

MÉTODO DE LA INGENIERÍA

Fase 1. Identificación del problema.

Identificación de necesidades y síntomas:

- ➔ El consumo de memoria RAM es muy alto
- ➔ Los usuarios necesitan features para que la construcción en Minecraft sea mas cómoda
- ➔ El acceso a los materiales de construcción no es eficaz.

Definición del problema:

- ➔ Los usuarios de Minecraft necesitan un nuevo feature en el juego para la modalidad de construcción y el acceso a las herramientas necesarias, además de buscar la optimización de la memoria RAM.

Fase 2. Recopilación de la información necesaria.

Fuentes:

https://bit.ly/2kaM4Ye
https://cprog.cubava.cu/2018/03/25/tablas-hash/
http://web.dit.upm.es/~pepe/doc/adsw/tema1/Complejidad.pdf
https://www.profesionalreview.com/2018/11/01/memoria-ram/

- *Memoria RAM*

  Esta es una memoria de acceso remoto que generalmente va fijada a la placa base, es extraíble y se puede ampliar.
  La principal función de esta es cargar todas las órdenes que se ejecutan desde el procesador. los cuales pueden provenir del sistema operativo, discos duros, dispositivos de entrada o de salida.
  Si esta memoria no existiera las órdenes serian tomadas directamente por el disco duro haciendo que su consumo sea mayor y el tiempo de entrega mucho más tardío, por lo que es un componente necesario.

  - *Estructura de Datos (Stack)*

  La clase Stack es una clase de las llamadas de tipo LIFO (Last In - First Out). Las operaciones básicas son push (que introduce un elemento en la pila), pop (que saca un elemento de la pila), peek (consulta el primer elemento de la cima de la pila), empty (que comprueba si la pila está vacía) y search (que busca un determinado elemento dentro de la pila y devuelve su posición dentro de ella).

  - *Estructura de Datos (Queue)*

  Una cola es una colección de objetos que son insertados y removidos de acuerdo al principio primero en entrar, primero en salir, FIFO (first-in first-out). Esto es, los elementos pueden ser insertados en cualquier momento, pero solamente el elemento que ha estado en la cola más tiempo puede ser removido en cualquier momento.

El ADT cola soporta los siguientes dos métodos fundamentales:

- **enqueue(e):** inserta el elemento e en la parte posterior de la cola.
- **dequeue():** quita el elemento del frente de la cola y lo regresa; un error ocurre si la cola está vacía.

Adicionalmente, de igual modo como con el ADT pila, el ADT cola incluye los siguientes métodos de apoyo:

- **size():** regresa el número de elementos en la cola.
- **isEmpty():** regresa un booleano indicando si la cola está vacía.
- **front():** regresa el elemento del frente de la cola, sin removerlo; un error ocurre si la cola está vacía.

- *Estructura de Datos (Hash Table)*

Una tabla hash, matriz asociativa, 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.

- *Orden de complejidad de los algoritmos*

Un algoritmo será más eficiente comparado con otro, siempre que consuma menos recursos, como el tiempo y espacio de memoria necesarios para ejecutarlo. Así mismo, es posible hallar una ecuación que define la complejidad temporal del algoritmo y, de esta manera, también establecer el tipo de orden de complejidad que pertenece a cada uno. Por eso, es importante tener en cuenta que entre menor sea su orden, más eficiente será el algoritmo en el momento de su ejecución y menos espacio de memoria ocupará.

Siguiendo este orden, las funciones de complejidad algorítmica más habituales en las cuales el único factor del que dependen es el tamaño de la muestra de entrada n, ordenadas de mayor a menor eficiencia son:

O(1)	Orden constante
O(*log n*)	Orden logarítmico
O(*n*)	Orden lineal
O(*n log n*)	Orden cuasi-lineal

O(*n2*)	Orden cuadrático
O(*n3*)	Orden cúbico
O(*na*)	Orden polinómico
O(2*n*)	Orden exponencial
O(*n!*)	Orden factorial

Fase 3. Búsqueda de soluciones creativas.

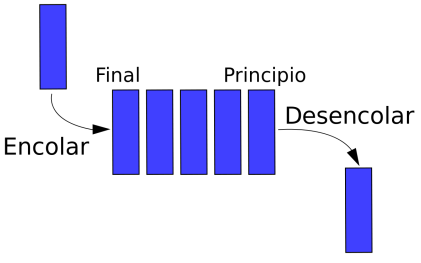
- Agregar los tipos de materiales dentro de una cola.
- Los bloques de cada material se agregan dentro de una pila.
- Utilizar el hashing doble para el proceso de inserción de bloques.
- Establecer un límite de 10 tipos de bloques de Minecraft para la solución.
- Asignar una llave para cada uno de los tipos de bloques que el usuario guardará.
- Utilizar 2 pilas para el manejo de la barra de materiales.
- Implementar la función cuadrática a la hora de insertar en la estructura de datos tabla HASH
- Permitir que el usuario escoja el tipo de material que necesita y que se busque automáticamente en la cola la barra que lo contiene.
- Implementar los 3 tipos de estructuras de datos (tabla HASH, Stack, Queue)

Fase 4. Transición de las Ideas a los Diseños Preliminares.

Alternativa 1. Implementar las 3 estructuras de datos, centrándonos en utilizar aquella que mejor se ajuste para cada parte puntual de la solución del problema.[1]

En primer lugar, una de las principales implementaciones en los algoritmos sería utilizar la estructura Stack para agrupar los bloques de un mismo tipo pues al ser todos iguales se pueden colocar en una pila sin afectar su uso pues no hace diferencia sacar el primer bloque de la pila o el quinto.

Del mismo modo, se utilizará la estructura de datos Qeue para tener la colección de barras de acceso rápido debido a que, es necesario para el usuario que al llegar a la última barra que contenga cierto tipo de material, tenga la posibilidad de que cuando haga clic en siguiente, lo lleve de vuelta a la primera barra que contiene otro tipo de material y, así, la estructura Qeue



[1] *Teniendo en cuenta que uno de los principales factores a mejorar dentro del juego es la organización de los bloques en el inventario, así como también la petición de los usuarios de una modalidad de barra de acceso rápido.*

puede facilitar el proceso con su forma FIFO utilizandola como si fuese una lista circular simple.

Y por último, la estructura de datos Hashtable para insertar los tipos de bloques en su correspondiente lugar dándole a cada tipo de bloque una key que lo llevará a su respectivo sitio.

Alternativa 2. Implementar 2 tipos de estructuras de datos. Una para optimizar la memoria en uso (Tabla Hash), y la otra para la modalidad de acceso rápido a la hora de construir (Stack).

En el primer caso, se busca utilizar una tabla Hash para poder disminuir el tiempo y espacio que requiere otro tipo de estructura en cuanto a la búsqueda de algún objeto en particular, por lo tanto, se cumpliría con uno de los objetivos que sería tener una complejidad temporal que permita una eficiencia algorítmica lo suficientemente satisfactoria para el usuario.

Por otro lado, se tiene la modalidad de acceso rápido para realizar construcciones, la cual se realizaría por medio de la estructura de datos Stack, que contaría con cada barra de acceso diferenciada por un tipo específico de material que pertenezca a Minecraft.

Fase 5: Evaluación y selección de la mejor solución.

Criterios. A partir de las alternativas planteadas y la definición del problema, se plantean los siguientes criterios para escoger la solución más apropiada:

- **Criterio 1:** Facilidad de implementación. Se refiere a la posibilidad que hay de terminar la implementación a un plazo corto de tiempo. Esto puede ser:
    - [3] Muy probable.
    - [2] Probable
    - [1] Improbable.
- **Criterio 2:** Eficiencia. Se refiere al tiempo requerido por el algoritmo para terminar su ejecución. El cual puede ser:
    - [3] Bajo.
    - [2] Aceptable.
    - [1] Alto.
- **Criterio 3:** Consumo de memoria. Se refiere a la cantidad de almacenamiento que implementa el algoritmo para su debido funcionamiento. Puede ser:
    - [3] Baja.
    - [2] Normal.
    - [1] Excesiva.
- **Criterio 4:** Accesibilidad. Se refiere a la dificultad de acceder a algún dato de la estructura
    - [3] Fácil.
    - [2] Media.

Alejandro Martínez
Cristian Lasso
José David Jiménez
Ana María Muñoz

- [1] Difícil.

Evaluación:

	Criterio 1	**Criterio 2**	**Criterio 3**	**Criterio 4**	**Total**
Alternativa 1.	3	2	3	3	11
Alternativa 2.	2	2	2	3	9

Selección:
De acuerdo a los criterios planteados y la obtención de los resultados finales, se considera como mejor opción de diseño de solución la Alternativa 1. Teniendo en cuenta que es importante fortalecer los criterios en los que se presente mayor dificultad y, además, implementar la solución con las modificaciones que sean necesarias para lograr el objetivo final, manteniendo la esencia de esa Alternativa.