Nicolás Penagos – Mateo Loaiza – Juan José Calderón

Método de ingeniería

Algoritmos y Estructuras de Datos

Contenido

Identificación del problema

Contextualización

Minecraft es un videojuego de construcción muy famoso, de tipo mundo abierto o sandbox. El juego usa un modelo de construcción basado en bloques que representan diferentes materiales. Estos bloques son almacenados en un inventario de tal forma que el usuario los tenga siempre disponible. Sin embargo, actualmente el sistema que maneja el inventario se considera ineficiente. Por otra parte, los jugadores no están del todo satisfechos con el juego y han solicitado en varias ocasiones que se añadan funciones que mejoren su experiencia.

Problema

Se requiere implementar una solución a la ineficiencia con la que se maneja el inventario del juego, reduciendo el consumo de memoria que este genera. Adicionalmente, es necesario implementar dos nuevas funciones: Acceso rápido y Construcción. Lo anterior, para mejorar el funcionamiento del juego y satisfacer a las demandas de los jugadores.

Necesidades

I. Se debe proponer una forma eficiente para acceder a los bloques del sistema, resolviendo el problema del consumo de RAM por parte del juego.
II. Se requiere implementar las funciones de modalidad de *acceso rápido* y *construcción*, por medio del uso adecuado de estructuras de datos.
III. Se solicita proponer una visualización intuitiva para percibir los nuevos cambios del sistema y de esta forma atraer a nuevos usuarios al mundo de Minecraft.

Recopilación de información

Con el objetivo de tener más claro el problema, decidimos investigar un poco más frente a como funciona el juego, cuales estructuras son usadas, de que manera son usadas y como manejan los datos. Adicionalmente, quisimos saber un poco más sobre el contexto del juego y, de esta forma, entender las peticiones de los

- Valores de datos: Con esto, se hace referencia a distintos tipos de objetos y bloques manejados en el juego. “Los ID de bloques son usados para definir bloques que situados en el mundo y en el inventario (incluyendo objetos en cofres y dejados en el mundo). Los ID de objetos sólo son válidos para objetos. Cada espacio en el inventario tiene un número único.” (GAMEPEDIA, s.f.)
- Modos de juego de Minecraft:
    - Supervivencia:
        - “Players must collect resources, build structures, battle mobs, manage hunger, and explore the world in an effort to thrive and survive.” (GAMEPEDIA, s.f.) Aquí, los jugadores deben buscar recursos e irlos agregando a su inventario. Al ir consiguiendo cada vez más materiales, estos pueden ser usados para “craftear” diferentes objetos o incluso nuevos bloques de mejores materiales.
    - Creativo:
        - “Creative mode strips away the survival aspects of Minecraft and allows players to easily create and destroy structures and mechanisms.” (GAMEPEDIA, s.f.) En este caso no hay un límite de bloques para que el jugador pueda dejar volar su imaginación. De esta forma, el almacenamiento se maneja diferente, pues no hay que guardar el número de bloques que se tienen de cierto material, ya que este siempre estará disponible.

Búsqueda de soluciones creativas

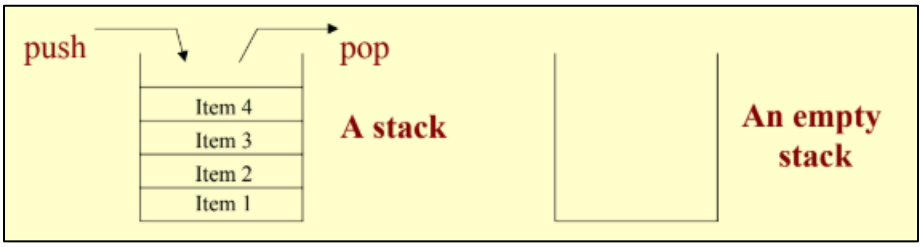
Requerimiento 1

Dada la necesidad de utilizar diferentes estructuras de datos para diferentes tipos de aplicaciones con el fin de disminuir la complejidad de los algoritmos y mejorar los tiempos de ejecución, se realizó una exploración acerca de los diferentes tipos de estructuras de datos disponibles con el fin de encontrar la que mejor se ajustara a las necesidades del caso. Se omitieron los tipos de datos tradicionales (Arreglo, ArrayList).

Alternativa 1. Stacks Data Structure

Stack o Pila es una estructura de datos linear con un número arbitrario de elementos que sigue un orden particular en cual se realizan las operaciones. Los procedimientos de acceso solamente permiten inserciones y eliminaciones al final de la secuencia (“top”).

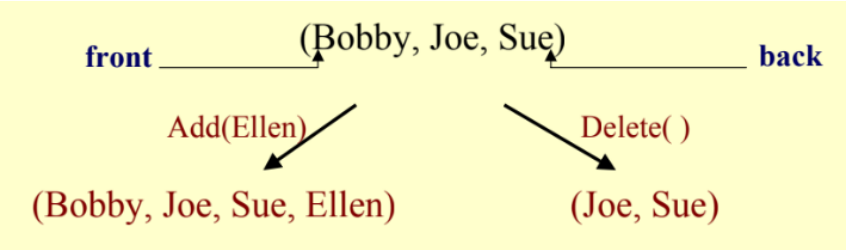
Frecuentemente es llamada last-in-first-out o lista LIFO por sus siglas en inglés.



Alternativa 2. Queue Data Structure

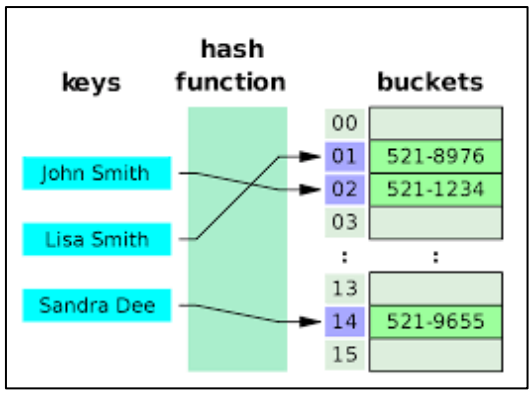
Queue o Fila es una estructura de datos linear con un número arbitrario de elementos junto con uno procedimientos de acceso. Dichos procedimientos, solo permiten adiciones en la parte trasera de la Fila (último elemento) y eliminaciones en el ítem que se encuentre al frente de la fila.

Frecuentemente es llamada first-in-first-out o lista FIFO por sus siglas en ingles.



Alternativa 3. Hash Table Data Structure

Esta estructura de datos utiliza una función especial llamada la función Hash para mapear una valor dado con una “llave” particular para brindar un acceso más rápido a los elementos que contiene.



Requerimiento 2

Se propone utilizar una estructura de datos tipo Stack o Queue anteriormente mencionadas paras implementar las dos nuevas funcionalidades, ya que ambas funcionalidades solo permiten adicionar bloques del mismo tipo si necesidad de ningún tipo de comprobación.

Requerimiento 3

Se propone el diseño de una interfaz gráfica intuitiva que tenga las siguientes características:

1. Permita almacenar solamente bloques del mismo tipo en la barra de acceso rápido.
2. Permita tener una n barras de acceso rápido para la modalidad de construcción.
3. Permita al usuario identificar por medio de imágenes que contengan miniaturas el tipo de bloque que se está usando.
4. Conserve la estética y estilo visual del videojuego para atraer a los usuarios.
5. Sea fácil de utilizar y contenga pocos botones para facilitar la tarea del usuario.

Transición de las Ideas a los Diseños Preliminares

Alternativas rechazadas

En primer lugar, se descarta utilizar la estructura de datos Queue para la implementación de las funcionalidades Construcción y Acceso Rápido. Esto se debe a que no hay necesidad de utilizar una jerarquía en entrada/salida de bloques. Con esto nos referimos a que podemos tratar el problema en su forma más simple, una sola entrada de objetos (bloques) del mismo tipo que se irán apilando. Por lo anterior, pudimos concluir que el uso de una Queue no aportaría mayormente a nuestra solución.

Alternativas aceptadas

- ✓ El uso de Hash Table es de vital importancia para reducir la complejidad y los tiempos de ejecución de los algoritmos utilizados para hacer referencia hacia el elemento deseado y así conectarlo de manera directa con el contador interno del sistema.
- ✓ Por simplicidad, se utilizará Stack como estructura de datos escogida para las implementación de las nuevas funcionalidades del sistema (Acceso Rápido / Construcción). Esto fue decidido en el momento en que comparamos que tanto nos aportaría usar Queue o Stack y, en resolución, decidimos que el problema no requiere del sistema FIFO de Queue, pues con el sistema LIFO de Stack podríamos manejar fácilmente los bloques.
- ✓ Se implementará una interfaz gráfica intuitiva de la manera como se plantea anteriormente debido a que no es una tarea sumamente tediosa y que, por el contrario, puede traer grandes beneficios para la experiencia de los usuarios.

Evaluación y Selección de la Mejor Solución

Criterios

1. Criterio A. Facilidad de interpretación
   [3] Sencillo, con leerlo se entiende
   [2] Moderado, necesito leerlo varias veces para entenderlo
   [1] Complicado, necesito leerlo varias veces durante mucho tiempo y buscar ayuda para entenderlo

2. Criterio B. Eficiencia
   [3] Constante
   [2] Linear
   [1] Exponencial

Evaluación

	Criterio A	Criterio B	**Total**
Alternativa 1. Stacks Data Structure	2	2	4
Alternativa 2. Queue Data Structure	2	2	4
Alternativa 3. Hash Table Data Structure	2	3	5

Selección

Las alternativas 3 y 1 son las escogidas. Con respecto a la alternativa 3, presenta conveniencia en eficiencia, lo que permitirá reducir tiempos de ejecución y reducir la complejidad de ejecución. El empate entre la alternativa 1 y 2 se decidió al saber que no existe una jerarquía en la entrada de los datos para lo cual el uso de una Queue de la alternativa 2 siendo equivalente a la alternativa 1 que maneja este problema de la misma manera y con mayor facilidad. Así que, las alternativas 1 y 3 serán las que van a ser usadas.

Preparación de Informes y Especificaciones

Diagrama de clases, Requerimientos, Diseño de pruebas y Diseño de TAD´s

Ver carpeta docs dentro del proyecto.