Christian Flor - A00355624, Daniel Villota - A00356255, Natalia Gonzalez - A00354849

Minecraft

Context of the Problem.

Minecraft[1] is a very famous construction video game, of the open world or sandbox type. The developers of this game have identified some problems that players have when building using blocks, as well as other development issues, therefore, they are looking for an optimal solution so that the performance is the best.

Solution development.

To solve the previous situation, the Engineering Method was chosen to develop the solution, following a systematic approach in accordance with the problematic situation.

Based on the description of the Engineering Method of the book "Introduction to Engineering" by Paul Wright, it presents the following flow chart, whose steps we will follow in the development of the solution.
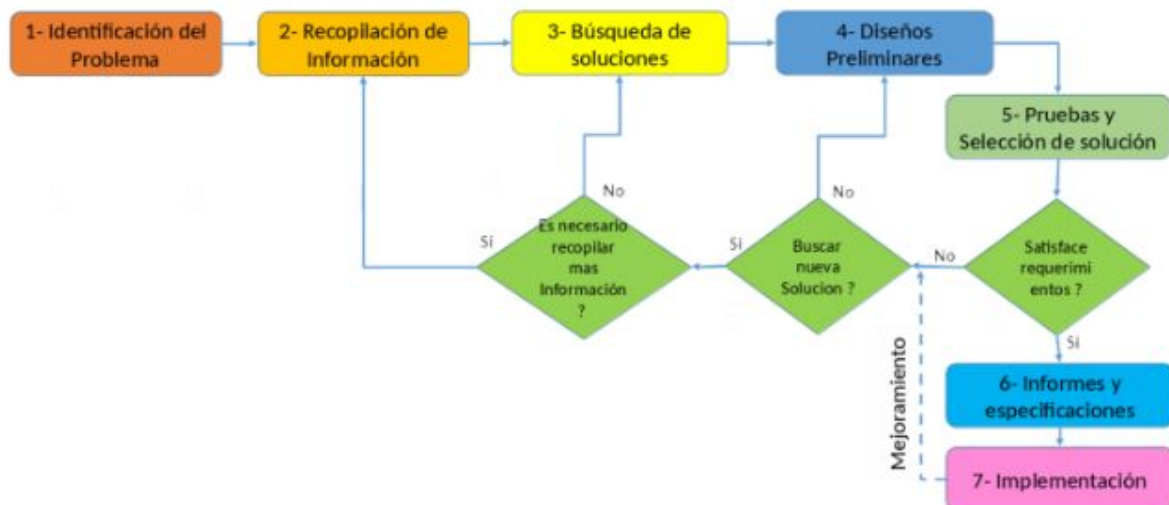
Step 1. Problem Identification

The specific needs of the problematic situation as well as its symptoms and conditions under which it must be resolved are specifically recognized.

Identification of needs and symptoms

➢ They have a slow algorithm to verify the type of block that is used to then add it to the inventory. This makes the consumption of RAM while using the game to be very high.

---

[1] Official Web site https://www.minecraft.net/es-es/

➢ Minecraft users have made requests to add features in the next version to have a new mode of quick access and construction in a more efficient way.
➢ The solution of the problem must be optimal so that the performance is the best.
➢ The construction mode in the game should allow the use of n different quick access bars. In this way the players would increase their productivity when they're building.

Definition of the problem

The minecraft game requires the development of new features to have quick access in order to build efficiently, also to improve the construction mode that allows using n fast access bars that allows to increase the productivity and finally implement a new algorithm for verifying the type of block in an inventory to reduce the consumption of RAM and make this process to be faster.

Specification of Functional Requirements. (in terms of input / output)

Name	R.F# 1. Allow to group blocks in a stack
Summary	The system will allow to group blocks of the same type and distribute them in the inventory
Input	
Blocks of the same type	
Output	
Grouped blocks	

Name	R.F# 2. Fill quick access bars with blocks
Summary	The system will have n quick access bars, each bar have slots which contain a only one kind of block. The user decides how many blocks are going to be placed in the bar. Everytime that a new type of block is added, a new bar is created.
Input	
Block, quantity	
Resultados	
Bar with the needed blocks	

Name	R.F# 4. Add a new block to the inventory
Resumen	The system will allow to add to the inventory a new block to the inventory
Input	
Block type, quantity	
Output	
Inventory with the added block	

Step 2. Information Collection

To have a total clarity in the concepts involved and how to play Minecraft, a search is made of the data structures that we can implement to improve the performance of the game, and to efficiently solve the problem posed. It is important to perform this search in recognized and reliable sources to know which elements are part of the problem and which are not.

Definitions

Source:
https://minecraft-es.gamepedia.com/Minecraft_Wiki

Minecraft inventory organization

If you want to divide a large number of objects or blocks from an inventory, you can right click on it to divide it into two parts. If the number of objects in the group is odd, then you will receive the largest part and the smallest part will remain in the inventory. For example, if the stack has 7 elements, dividing them you will get 4 items and the inventory will remain 3. This leaves half of the stack in its original position and allows you to move the other half. On the contrary, while holding a stack of items and right clicking on an inventory space, a single object will be placed anywhere in the inventory where the mouse is.

One way to quickly organize and sort objects is by holding down ⇧ Shift and clicking on the stack. In this way, while the chest is open, the object or stack will automatically go to the chest, if ⇧ Shift is held down and an item is clicked inside the open chest, the stack or item will automatically enter the inventory. If you do it only with the inventory open and the item is in your quick access bar, ⇧ Shift and a click will move the stack to the storage inventory (the 9x3 grid that is located above the quick access bar). If you do and there is a block or object where you want to move the battery, the elements will be replaced instead.

Blocks

The blocks are the basic units of any minecraft structure. They can be put and destroyed.

Data Structures to be used in the project:

- **Stack**: It's a collection of elements, that follows the LIFO order. LIFO stands for Last In First Out, which means element which is inserted most recently will be removed first.

- **Queue**: It's a data structure that follows the FIFO principle. FIFO means First In First Out i.e the element added first in the queue will be the one to be removed first. Elements are always added to the back and removed from the front.

Source: https://www.hackerearth.com/practice/notes/stacks-and-queues/

- **Hashtable**: It's a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be found.

Source: https://en.wikipedia.org/wiki/Hash_table

Each of the members of the project installed a free version of the official Minecraft website to understand how the inventory of this game was specifically, this was done with the purpose of having a clearer idea of what we have to carry out without lose the basic idea of game developers.
Source:
https://www.minecraft.net/es-es/download/

Step 3. Search for creative solutions

It's necessary to achieve an efficient way to access the blocks in the system, so it has been found out that a hash table might be useful for this task, because with a good implementation of it, searching for an item in the average case takes the constant time of O(1). It is a big improvement to the algorithm actually used that looks for every block in the system till the needed one is found. This algorithm takes O(n) time in the worst case.

Every block is organized into a tridimensional axis divided in spaces of one cubic meter (1m³). There cannot be more than one block in one space.

Blocks and fluids constitute the environment in the game, the can be picked up and used in a large sort of ways.

(Illustration #1)

As it can be seen in the Illustration #1, there are several kinds of blocks in the minecraft world, but we are only going to focus on the naturalish created blocks (they are specified in the [Minecraft Wiki](), also can be seen in the Illustration #2).

★ Add blocks to the inventory
Alternative 1.
There will be a main mini inventory in which all the 15 construction objects specified in illustration 2 are found, which the user can use to obtain all the necessary blocks to build. To add them to your personal inventory you must click for each block you want.

This alternative can be implemented using:
- A matrix, useful for storing each object
- An array, because the number of objects is already defined
- A Hash table, because you can assign a key to each element and perform a quick search when necessary.

Icono	Dec.	Hex.	Nombre	Bloque
	0	0	minecraft:air	Air
	1	1	minecraft:stone	Stone [S B]
	2	2	minecraft:grass	Grass Block
	3	3	minecraft:dirt	Dirt [S B]
	4	4	minecraft:cobblestone	Cobblestone
	8	8	minecraft:flowing_water	Water [S]
	39	27	minecraft:brown_mushroom	Brown Mushroom
	40	28	minecraft:red_mushroom	Red Mushroom
	49	31	minecraft:obsidian	Obsidian
	51	33	minecraft:fire	Fire [S]
	78	4E	minecraft:snow_layer	Snow [S B]
	79	4F	minecraft:ice	Ice
	81	51	minecraft:cactus	Cactus [S]
	83	53	minecraft:reeds	Sugar Cane [I S]
	106	6A	minecraft:vine	Vines [S]
	110	6E	minecraft:mycelium	Mycelium

(Ilustration #2)

Alternative 2.
From the main mini inventory (proposed in alternative 1), the user will be shown the types of blocks that exist and when clicking on the type he wants, he must enter the amount he wants to add to his personal inventory with a maximum number of 9 * 64, since the personal inventory will have 9 spaces available with a maximum of up to 64 objects grouped for each one.

For this alternative you can implement the options of alternative 1 and additionally:
- A stack, which is responsible for saving the selected objects from the inventory
- A queue, useful for the selected inventory items to be added and used in the selected order

★ Add blocks to the quick access bar
It is clear that the quick access bar will be located at the bottom of the screen to allow easy access to all the objects you own. This bar shows only one type of object at a time.

Alternative 1.
Use the same blocks in the inventory avoiding to create new blocks.

Alternative 2.

The user is asked for what type of blocks he wants and what is the desired amount of blocks (n). If n is greater than 64, the blocks are going to be put in floor(n/64) slots and if m = nMod64 is greater than 0, the other m blocks are going to be located in the next free slot.

In this case you need a structure that is responsible for grouping the objects in each block with the maximum amount of 64. For this it is possible to use:
- Stacks
- Lists
- Arrays
- Queues

★ Add different quick access bars
Alternative 1.
To see more bars with different types of objects it must click on an arrow to the right of the bar, to find the following bar. The main advantage of this proposal is that the user will have full access to their blocks and the different bars.

In this alternative it is possible to use:
- An array queue, which is responsible for displaying one arrangement at a time, which corresponds to the same type of blocks.
- A list containing the blocks sorted by each type

Alternative 2.
A new bar will be implemented which will be created as a list with the information provided by the user saying how many fast access bars you want and from each of these choose the type and quantity

Playability by default
In addition to the ideas presented above, it is thought that by default we have initialized blocks read by text files, in which the user must limit himself by not being able to add more, in the inventory he will only have that and with those he must play. For quick access it would also be done with predetermined amount by default so that the user is only limited to what he has and is recursive when playing.

Step 4. Transition of ideas formulation to preliminary designs

In this step we discard unfeasible options.

Considering the phase of searching for creative solutions, we discard the option to limit the user since all the blocks that were initially would be default quantities that will be available for the use of the game, but will only have these blocks to interact with the quick access bar and the inventory, therefore we discard all the ideas raised

that it is suggested that everything be predetermined, thus being, as a result, the proposals remain that the user can choose the amount of blocks that he wants so that he interacts more with the game and does not get bored of this. In addition the idea is to keep the idea is to keep the base of the game without modifications.

Step 5. Evaluation and Selection

★ (A) Add blocks to the inventory
In this possibility we discard the alternative (2) due to the obligation to implement an algorithm to properly organize the blocks in the inventory, in addition to knowing that we can restrict the entry, the user will always tend to place large quantities. Finally, taking into account that two input variables must be analyzed, this will require more RAM consumption.

The careful review of the other alternatives leads us to the following:

Alternative 1.
- For the user it can be boring and delayed to add the blocks one by one, but it will be more effective and it can be considered that when clicking, two or three blocks are added to the inventory to improve the user experience.
- Ease of grouping objects.

Alternative 2.
- The program will have to implement an algorithm if the amount entered by the user is greater than 64, to consistently distribute the blocks with the appropriate quantities.
- There will be more use of RAM since you will have to analyze the variable of the type of object you want to add, plus the amount that the user wants to add to the inventory

(B) Add blocks to the quick access bar
In this possibility we select the alternative (2) because the user can have more blocks to build and can also use the inventory blocks.

Alternative 1.
- It has the advantage of transferring the blocks desired by the user in the inventory and then added to the features bar.
- Faster when you just select the items that exist in the inventory
- Some inventory blocks may be grouped which make it easy to add to the inventory.

Alternative 2.
- In this alternative, although it is considered to create new building blocks, a mathematical equation is used that helps us to group the created blocks in an optimal way.
- The inventory does not move but we could add inventory items to this quick access bar.

★ **(C) Add different quick access bars**
Alternative 1 is selected for best user experience.
Alternative 1.
- In this alternative it will be easy to add new bars below, from the first one since the user is only asked the type of block and the quantity, the new bar will be created following the same idea selected from the features bar mentioned above
- Improve the user experience by creating more bars.

Alternative 2.
- The user is given more freedom to decide how many quick access bars he wants but at the same time he is restricted since this can only be done once.
- The user can type a large number of quick access bars.

Step 6. Preparation of Reports and Specifications

2. Class diagram
3. Test case design
4. Spatial and temporal complexity analysis

4. Pseudocode of the most relevant algorithms

Algorithm flow chart	Pseudo Code Algorithm

Step 7. Design implementation
The solution to the problem is in the following repository:
https://github.com/DanielVillota16/minecraft-features.git