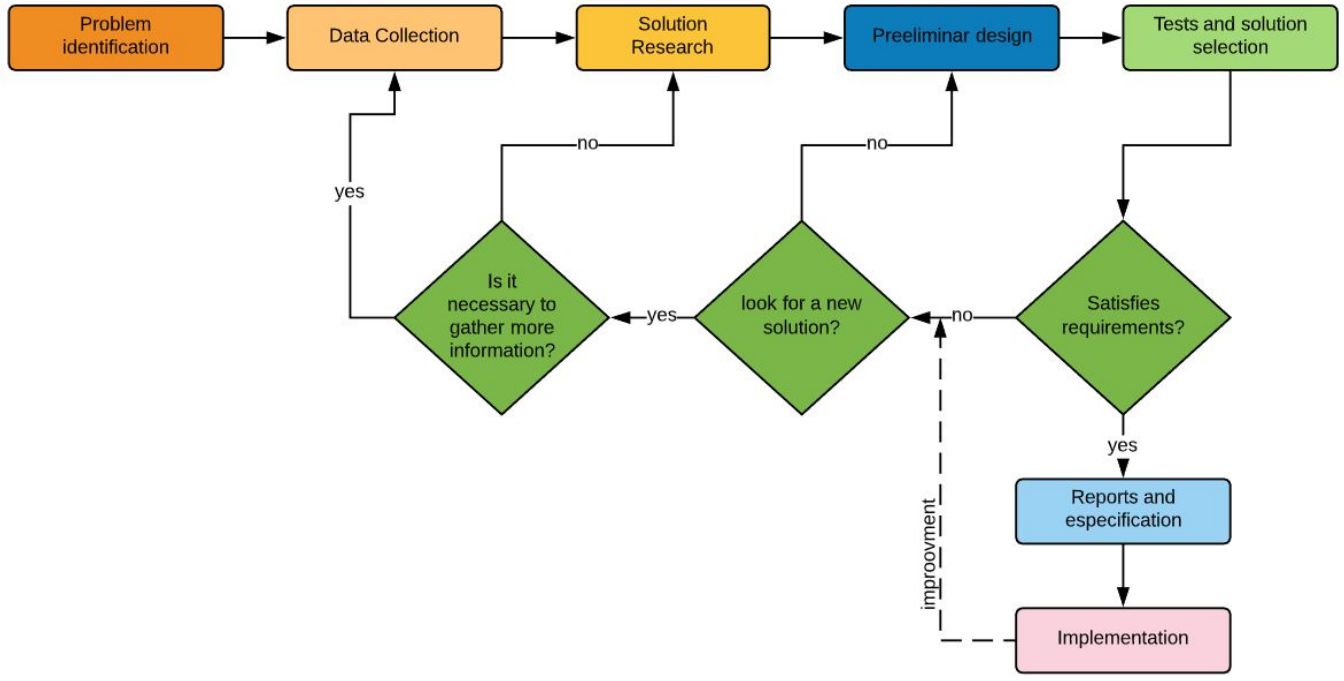
A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

Minecraft

Engineering method

Problematic context

In order to solve the previous situation we choose to use the Engineering method to develop following a systematic focus according to the planted problem situation.
Based on the description of the engineering method from the book “Introduction to Engineering” from Paul Wright, it’s been defined the following flow diagram, which steps will be following during the development of the solution



Step 1: Identification of the problem

Identification of symptoms and needs

- The algorithm used to verify the blocks, runs one by one the total blocks in the system, and if they are of the same type, groups them with the blocks that are present in the inventory.
- Minecraft requires the implementation of a more efficient algorithm that allows block verification more efficiently and consumes less system memory.
- Minecraft does not currently have a quick selection menu to sort the blocks in the inventory.

A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

- A quick access menu should be developed that allows stacking blocks of the same type in the quick access bar, this varies by clicking an arrow next to the quick access menu, when arriving at the nth block, the next type of block would be again the first quick access menu.
- It is necessary to implement the appropriate use of data structures for the project, such as Queue, Stack and Hash Tables.

Definition of the problem

Minecraft requires the implementation of a better block verification algorithm and a new quick access menu to the blocks in the inventory.

Step 2: Data collection

Definitions

Font:
https://drive.google.com/file/d/1CzDT9IXYyiBQobtfgpcDB8ibexFOgPx0/view?usp=sharing
https://sites.google.com/a/espe.edu.ec/programacion-ii/home/tablas-hash
https://sites.google.com/a/espe.edu.ec/programacion-ii/home/colas
https://sites.google.com/a/espe.edu.ec/programacion-ii/home/pila
https://minecraft.gamepedia.com/Inventory
https://www.taringa.net/+info/id-de-objetos-y-bloques-minecraft_v227e

Hash Tables

It is a data structure that associates keys or keys with values. The main operation that supports efficiently is the search: it allows access to the elements stored from a generated key. It works by transforming the key with a hash function into a hash, a number that identifies the position (slot) where the hash table locates the desired value.

Hash Functions

Stack

A Stack in simple words is a place where data is stored, as in an Array, but a Stack has a philosophy of data entry and exit, this philosophy is the LIFO (Last In First Out, in Spanish, last to enter , first out). This data structure has many applications due to its simplicity.

Queue

A list behaves like a queue if the insertions are made at the end and the extractions are made in front of the list. They are also called FIFO lists (First In First Out - first to enter first to exit).

Minecraft inventory

The inventory consists of 4 armor slots, 27 storage slots, 9 hotbar slots, and an off-hand slot. Items in the hotbar slots can be selected during play using the keyboard (keys 1–9) or mousewheel, and placed or wielded with the mouse buttons.

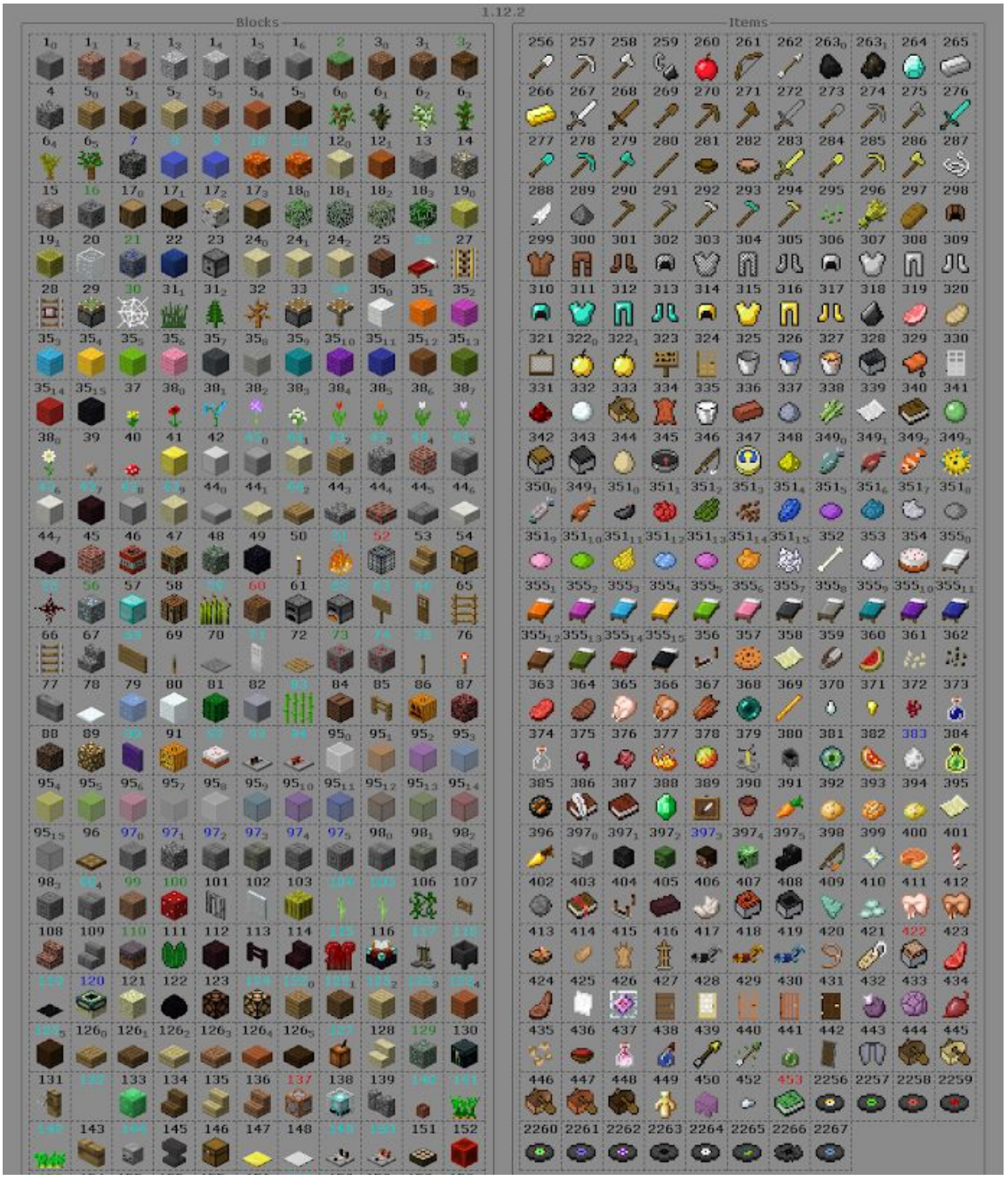
A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

Most items can stack up to a maximum of 64 in one slot. Some items cannot stack, notably tools (with the exception of clocks and compasses), armor, and potions. Certain items, such as snowballs, empty buckets, eggs, signs, banners, and ender pearls can only stack up to 16.

In Survival or Adventure mode, there is also a 2×2 crafting grid, which allows any recipe that fits to be crafted, but does not offer item storage.

Pressing use on a block with an inventory, such as a chest, crafting table, or a furnace opens its GUI and allow items to be transferred between the 27 main slots, the 9 hotbar slots, and the block's GUI.

In Minecraft each block / object has a specific ID, so we will take that ID as a key to execute the functions of the hash table, the object IDs are shown in the following image:



A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

Step 3: Solution research

Choice 1: using a hash table with a stack

Use a hash function that it's going to be in charge of sorting and adding the objects into a stack using its name as a key, the first hash function will place the object in a respective row of the table, every position of the hash table is going to have a stack that can hold a max of 64 objects when the stack is full is going to count as a collision and it will use a second hash function that will place it in the position next to it(next to it but not in the next row)

Choice 2: Using a hash table with a queue

Use a hash function that it's going to be in charge of sorting and adding the objects into a queue using its name as a key, the first hash function will place the object in a respective row of the table, every position of the hash table is going to have a queue that can hold a max of 64 objects when the queue is full is going to count as a collision and it will use a second hash function that will place it in the position next to it(next to it but not in the next row)

Choice 3: Using binary search to identify the type of block

It is the most efficient method of searching a list that is previously sorted, at the beginning the key searched is compared to the central element of the list, if these are the same there ends the search and displays the position of the key if not, then Either the top or bottom of the list should be searched in a similar way depending on the value of the key. The list can be sorted by insertion or any other sorting method, naturally (A-Z) by block name.

Choice 4: Using Hash table (Open Addressing) to identify the type of block

Hashing is an important Data Structure which is designed to use a special function called the Hash function which is used to map a given value with a particular key for faster access of elements. The efficiency of mapping depends of the efficiency of the hash function used. For a collision we can use double hashing: is a collision resolving technique in Open Addressed Hash tables. Double hashing uses the idea of applying a second hash function to key when a collision occurs.

Function Hash

$$h(k) = (h_1(k) + h_2(k) * i) \bmod m$$

Where:

$k =$ ID of the block we are looking for.

i = Number of collisions that occur.

m = Hash table size.

mod = module operation.

$$A = (\sqrt{5} - 1)/2$$

$$h_1(k) = \downarrow m * (k * A \bmod 1) \downarrow \bmod m$$

$$h_2(k) = \downarrow (k \bmod m) \downarrow$$

A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

Choice 5: Using the name of an block as a key to identify the unknown new block.

The use of a hash table to identify a type of a block is very efficient, but one problem is how to selectionate the key to search. The option is using a system of identification of strings to a key whose each character of the String parameter will have a value $(a = 0,\ b = 1,\ c = 3, \ldots, x = 24, y = 25, z = 26)$ for each word it will be an operation $((valueOfCharacter * 5^{0} + valueOfCharapter * 5^{1} + \ldots + valueOfCharapter * 5^{n}))$, in this way we can identify easily in the hash table of blocks the unknown element with $\theta(1)$ (the value search of an element in a hash table), and the next part of the problem who is add up the identify block with the actual elements in the inventory, if the position of actual inventory is full, add the rest to the next position, if the next position is occupied use a system of collisions to find a slot, and to finish if there is not a slot, the block can't be added.
we will use double hashing in the two hash tables(the identify hash table and the inventory hash table)

Function Hash

$$h(k) = (h_1(k) + h_2(k) * i) \bmod m$$

Where:

i = Number of collisions that occur.
m = Hash table size.
mod = module operation.
$A = (\sqrt{5} - 1)/2$

$$h_1(k) = \downarrow m * (k * A \bmod 1) \downarrow \bmod m$$
$$h_2(k) = \downarrow (k \bmod m) \downarrow$$

Choice 6: Using the predetermined tag of each element in minecraft as a key

The design of minecraft have a tag in all the elements in the game, that allows use the tag as a key of a hash table to identify the unknowns blocks with his respective tags. doing the program more effective, the next part of the problem who is add up the identify block with the actual elements in the inventory, if the position of actual inventory is full, add the rest to the next position, if the next position is occupied use a system of collisions to find a slot, and to finish if there is not a slot, the block can't be added.
we will use double hashing in the two hash tables(the identify hash table and the inventory hash table)

Function Hash

$$h(k) = (h_1(k) + h_2(k) * i) \bmod m$$

Where:

A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

i = Number of collisions that occur.

m = Hash table size.

mod = module operation.

$A = (\sqrt{5} - 1)/2$

$$h_1(k) = \downarrow m * (k * A \bmod 1) \downarrow \bmod m$$

$$h_2(k) = \downarrow (k \bmod m) \downarrow$$

Step 4: Preliminary design

Choice 1: using a hash table with a stack

- the conversion from text to hexadecimal or integers is an impractical way of getting the key for each object
- the implementation of two data structures when one is inside the other means a considerable increase in time processing for the machine

Choice 2: using a hash table with a queue

- the conversion from text to hexadecimal or integers is an impractical way of getting the key for each object
- the implementation of two data structures when one is inside the other means a considerable increase in time processing for the machine
- the queue is not best approach for this type of problem because it mean more relations between classes and that mean a heavier program

Choice 3: Using binary search to identify the type of block

- Implementing binary search requires that the list with which we are going to work is organized, so, if it is not, we will have to do so, consuming resources that we want to save.
- Although the number of iterations is considerably reduced, to do a linear search, the fact of having to organize the list before completely discards this alternative.

Choice 4: Using Hash table (Open Addressing) to identify the type of block

- The hash table method allows us to search very efficiently, since each function results in a different value for each key, and in case two values give the same result, we only follow the steps I would do if I got a collision until you find the value.
- How we know the ID of the block we are looking for, it is only to search the hash table for that ID through h (k) where k = ID and we will know where it is stored.

Choice 5: Using the name of an block as a key to identify the unknown new block

- This way is creative to identify the unknown blocks but the method of identify a block will be inefficient because the number of characters in the word could be extensive, and the value of each character could be very large. for this reason this alternative is discarded

A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

Choice 6: Using the predetermined tag of each element in minecraft as a key

- Minecraft have already a tag in each block of the game, far that reason the implementation of an hashTable will very easy, because the keys are already established and there will be not collisions because all the tags are different form each one.

Step 5: Solution Selection

Criterion A: Accuracy of the solution.

[2] Exactly
[1] Approximate

Criterion B: Efficiency in the algorithm time of execution.

[4] Constant
[3] Greater than constant
[2] Logarithmic
[1] Linear

Criterion C: Ease of algorithmic implementation.

[3] Practical.the algorithm is suitable for different cases
[2] Kind of practical. the algorithm only works for this case
[1] impractical, the algorithm does not works

Criterion D: Completeness of the solution

[3] All and more
[2] More than one, but not all
[1] only one or none

Criterion E: Implementation given time.

[3] Leftover
[2] Optimal
[1] Scarce

Choice	Criterion A	Criterion B	Criterion C	Criterion D	Criterion E	Total
Choice 1: using a hash table with a stack	Exactly [2]	Linear [1]	Kind of practical [2]	more than one but not all [2]	Optimal [2]	9
Choice 2: Using a hash table with a queue	Exactly [2]	Linear [1]	Kind of practical [2]	more than one but not all [2]	Optimal [2]	9
Choice 3: Using binary search to	Exactly [2]	Logarithmic [2]	Practical.the algorithm is	All and more [3]	Optimal [2]	12

A00354573 Juan David Lectamo Caicedo
A00354217 Jairo Emilio Rodriguez Viveros
A00295744 Johann Andrei Ocampo Torres

identify the type of block			suitable for different cases [3]			
Choice 4: Using Hash table (Open Addressing) to identify the type of block	Exactly [2]	Greater than constant [3]	Practical.the algorithm is suitable for different cases [3]	All and more [3]	Optimal [2]	13
Choice 5: Using the name of an block as a key to identify the unknown new block.	Exactly [2]	Linear [1]	Practical.the algorithm is suitable for different cases [3]	All and more [3]	Scarce [1]	10
Choice 6: Using the predetermined tag of each element in minecraft as a key	Exactly [2]	Greater than constant [3]	Practical.the algorithm is suitable for different cases [3]	All and more [3]	Scarce [1]	12