Using Minecraft To Store Data

Hampton Moore

March 2019

1 What is the goal?

The goal of this paper is to explore various routes of storing bits of data in Minecraft and find the most efficient way. All of these methods should be possible in vanilla Minecraft survival mode without any external tools or glitches. When calculating storage space I will be using bits per block and if required higher magnitudes like kilobytes per block. When comparing minecraft storage densities to real world storage densities one cubic block will be equal to one cubic meter.

1.1 Introduction to Bits

Throughout this paper various methods of storing bits will be used but one of the most important methods is using numbers to represent binary data. To calculate how many bits you take the highest possible number in that system or method can store then apple the Log function with a base of two, then floor the output to the closest integer. This shows how many bits can be represented. Note there can not be any gaps in the numbers as that would break binary counting and not allow full representation. For instance if a block has 5 different appearances, it can store 2 bits of data based on the appearance, $floor(log_2 5) = 2$. Appearance 1 would be 00, appearance 2 being 01, appearance 3 being 10, and appearance 4 being 11. Note the exclusion of the 5th possible appearance as there would need to be 8 appearances to add an extra binary bit.

1.2 Equations

Throughout this paper the calculations for storage density will be shown inline. These equations should be read left to right, ignoring normal orders of operations.

2 The Basics

The first and most obvious method is similar to a real world punch card. The existence of a block would count as a binary one while the lack of a block

counting as a binary zero, the main issue with this is the lack in density. With this method one achieves a storage density of only one bit per block. Now it is possible to get better densities with this method. I counted 298 survival placeable blocks in survival minecraft[1]. This number is closest to the number 2^8 or 256, meaing 8 bits can be encoded based on what block is placed. This increases the density to 8 bits per block

3 Storage Blocks

It is obvious that more blocks are going to need to be squeezed into each block, with chests being the obvious solution due to the fact that they can hold 27 blocks or items[8]. Using the same method discussed previously that brings the new density up to 216 bits per block. Chests are not the only storage block in the game though, since the introduction in shulker boxes into the game the player can now storage a whole lot more per block. Each shulker box, much like the chest, has 27 slots for storing blocks in it, these shulker boxes can then destroyed and placed inside the chest while still holding the items inside[2]. This means that each chest can now hold $8bits * 27slots * 27slots = 5832$ or 5832 bits per block, or a full 0.7 kilobytes.

4 Naming Blocks

Each block in Minecraft can also be named using an anvil, the max length for these names is 35 character, this gives another 35bits per block to work with[3]. This raises the density to an astonising $35bits(naming) * 8bits(blocktype) * 27slots(shulkerbox) * 27slots(chest) = 204120$ or 25.5 kilobytes per block, but even now more improvements can be made. As a reliable source on the characters allowed can not be found, it will be assumed they can only use a-z lowercase, A-Z uppercase, 0-9, dash (-), and space. This totals up to 64 different options allowing one to encode 6 bits based on the character, this means instead of 35 bits per block name, we now get $35characters * 6bitspercharacter = 210$ or 210 bits per block name. This boosts the density up to $210bits(naming) + 8bits(blocktype) * 27slots(shulkerbox) * 27slots(chest) = 158922$ or 19 kilobytes per block.

5 Switching Item Used

Currently the lowest number in the equation is the bits per block encoded. This is easily fixable if one switches to a different block/item. So what block/item allows to max customizability while still allowing to be placed in chests and be named.

5.1 Book And Quill

A Book and Quill can contain 12,800 characters so just writing ones and zeros would allow for megabytes to be written per block, but this simply removes the fun of the project and for this reason book and quills will not be used [4]. For those interested in the calculation the equation now becomes $210bits(name) + 12800bit(booktext) * 27slots(shulkerbox) * 27slots(chest) = 9331410$ or 1.2 Megabytes per block. This also is not the max number for Book and Quill as one can use the same A-z,0-9,dash and space method that was used to get just shy of 7MB of storage per block.

5.2 Leather Armor

Leather armor comes in 4 forms allowing for a 2 bits to be encoded based on what item is used, but this is not the reason that leather armor was chosen. Each piece of leather armor can be dyed a total of 12,326,391 different colors, the closest exponent of 2 is 2^{23} or 8388608. This allows for 23 bits of data to be stored based on the color[7]. This means when added to the 2 bits for leather armor type 25 bits can be encoded per piece of leather armor excluding naming, this makes the new density $210bits(naming) + 25bits(leatherarmor) * 27slots(shulkerbox) * 27slots(chest) = 171315$ or 21.41 kilobytes per block.

6 Enchanting

Another reason for leather armor is it has the ability to be enchanted. There are six enchantments that each piece of leather armor can the this allows to 6 bits of data to be stored if the enchantment is on or not on the armor[5]. While this might not seem like much bit each bit added increases storage ability by a tenth of a kilobyte when factored in to chest multiplying. This changes the equation to $210bits(naming) + 25bits(leatherarmor) + 6bits(enchantments) * 27slots(shulkerbox) * 27slots(chest) = 175689$ or 21.96 kilobytes. Enchanting can go even further, if instead of thinking about the existence of an enchantment being the one or zero one can instead use the level the enchantment is at. Going enchantment by enchantment it starts out with Protection, this can be from level 1 to 4 allowing one to encoded 2 bits encoded almost like normal binary numbers (level 1 = 00, level 2 = 01, level 3 = 10, level 4 = 11). This brings possible bits now to 2. Next is curse of binding, vanishing, and mending. This all have a max value of 1 meaning we can keep using its existence or lack of existence to be a binary digit. This brings it up to 5 bits. Next there is thorns and unbreaking, both of these have a max level of 3. This will be encoded in the same way as Protection, but with the lack of an enchantment being 00, and level 3 being 11, this method adds 2 bits for both thorns and unbreaking. This means enchantments can now encode a total of 9 bits. This changes the density to $210bits(name) + 25bits(leatherarmor) + 9bits(enchantments) * 27slots(shulkerbox) * 27slots(chest) = 177876$ or 22.23 kilobytes per block.

7 Damage Values

Each piece of leather armor excluding the helmet has at least 64 damage, so we we can use the damage value to encode 6 bits of information, there is one bit lost for not using a helmet anymore but 24 bits is gained [6]. This changes the equation to $210bits(name)+24bits(leatherarmor)+24bits(leatherarmordamage)+9bits(enchantments)*27slots(shulkerbox)*27slots(chest) = 194643$ or 24.33 kilobytes of data per block.

8 Comparing To Real World: Selectron

The Selectron was one of the first digital storage mediums could store 4096 bits per unit[9]. The dimensions of a Selectron was 250mm tall with a diameter of 76mm to calculates out to a volume of 0.0011341 cubic meters, meaning a density of 3.6116102^6 bits per cubic meter, or 0.45 megabytes per cubic meter. Taking the density calculated in the section "Damage Values" the density of data storage in Minecraft is 194700 bits per cubic meter or 0.02434 megabytes per cubic meter. This is obviously a major difference and shows that even at the beginning of technology data could be stored much more efficiently. The one thing the Selectron loses to is the Book and Quill method, but that method was ignored due to lack of creativity.

References

[1] Minecraft Gamepedia: Block
https://minecraft.gamepedia.com/Block

[2] Minecraft Gamepedia: Shulker Box
https://minecraft.gamepedia.com/Shulker_Box

[3] Minecraft Gamepedia: Anvil
https://minecraft.gamepedia.com/Anvil

[4] Minecraft Gamepedia: Book and Quill
https://minecraft.gamepedia.com/Book_and_Quill

[5] Minecraft Gamepedia: Enchanting
https://minecraft.gamepedia.com/Enchanting

[6] Mineraft: Gamepedia: Armor
https://minecraft.gamepedia.com/Armor

[7] Minecraft Gamepedia: Dye
https://minecraft.gamepedia.com/Dye#Dyeing_armor

[8] Minecraft Gamepedia: Chest
https://minecraft.gamepedia.com/Chest

[9] Selectron
https://en.wikipedia.org/wiki/Selectron_tube