Kyle Quinones

GDSTRUC_OTIE3

DECONSTRUCTION PAPER

This deconstruction paper will show the direct application of data structures and algorithms in video games. How they affect and support a more efficient gameplay, world management, and real-time processing. As todays modern world video games have evolved in a more complex and expanding world featuring interactive entities, the use of AI for AI behaviors, and multiplayer environments, optimized and efficient systems are essential. There are two key points to focus on data structures provides a way to organize and store data in a more efficient way, while algorithms on the other hand are created to solve problems like pathfinding, procedural generation, inventory management, and network synchronization. By examining their roles in game development and analyzing the game that is intended fro me to work on which is Minecraft, we can better understand how these data structures and algorithms help support the player for better gameplay experience.

Data structures as previously mentioned helps manage the game data or information more efficiently like handling memory allocation, world segmentation, event storage, and collision detection. Open-world games use structures like quadtrees, octrees, and hash maps to reduce memory use and improve the performance of gameplay, hash maps often provide fast lookups in inventories and arrays or lists that support animations and temporary objects. Another one is object pooling it works in a way that it reuses entities like bullets or particles to save resources. Algorithms on the other hand works on run time challenges like pathfinding(e.g A* thats for Minecraft) guides AI navigation, procedural generation (Perlin Noise, Simple Noise, Cellular Automata) for world building, for scalable worlds, physics algorithm handle collisions and interactions, and you cant leave out networking algorithms that support better and smoother multiplayer experiences. With these data structure and algorithms it enables and efficient, complex , and interactive gameplay.

Minecraft is one of the great example of a video game that demonstrates data structure and algorithms. Like its procedurally generated terrain cannot be stored fully in memory, so it opted to divide the world into chunks specifically 16x16 block sections that extends vertically from bedrock to the build limit. These chunks are are marked indexed using hash maps and stored in region files, which allows the game to load only nearby chunks as the player moves. This

approach is ingeious as it reduces memory usage and ensures smooth performance even in large, dynamic environments. Its terrain generation is unique as it relies on layered noise algorithms such as Perlin noise and Simplex Noise, with this it determines the elevation, biome placement, cave formation, ore distribution and many more actually. Entities such as villagers, zombies, and animals use A* pathfinding to navigate the block-based world efficiently. The crafting system relies on structured recipe data stored in arrays and maps for fast validation, with each recipe containing inputs, outputs, and pattern information. In multiplayer, the game employs server tick updates at 20 TPS and network packet optimizations such as delta compression and client-side prediction to ensure synchronized gameplay.

Example:

1.Chunk System (Data Structure)

- Stores the world in 16×16 block segments.
- Likely implemented using hash maps keyed by chunk coordinates.
- Saves chunks in 32×32 region files for fast loading and unloading.
- Reduces memory usage and improves world-streaming performance.

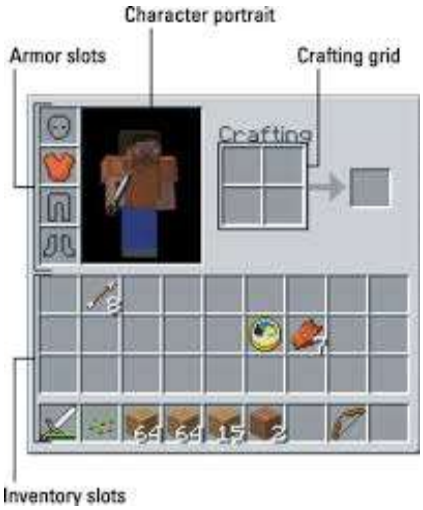
2.Inventory and Crafting Recipe Structures (Data Structure)

- Inventories use array-based slot systems.
- Crafting recipes stored in maps/dictionaries for quick lookup.

Recipe objects contain:

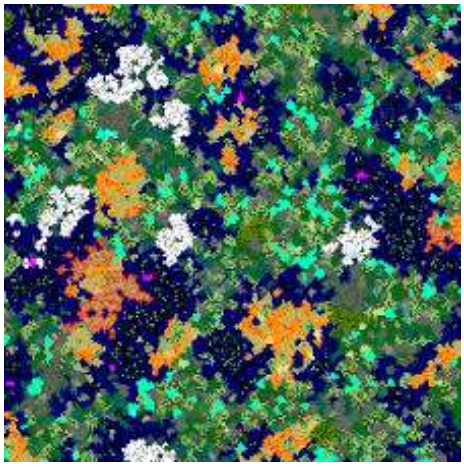
- input[] materials

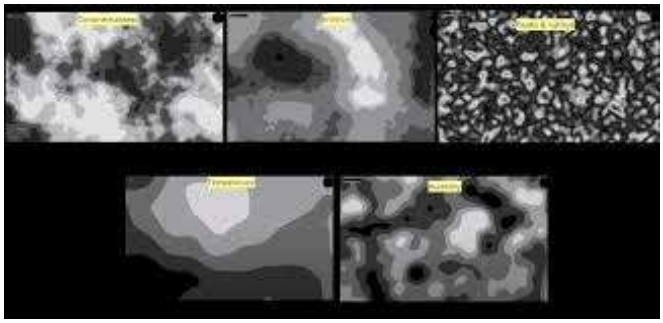
- output item
- pattern[] for shaped recipes
- Ensures efficient crafting validation.



3.Procedural Terrain Generation Algorithms (Algorithm)

- Uses Perlin Noise and Simplex Noise for landscapes.
- Multi-noise samplers determine biome temperature, humidity, and terrain erosion.
- Cellular automata influence cave and ore placement.
- Deterministic seed ensures reproducible world generation.





4. A* Pathfinding for Mobs (Algorithm)

- Grid-based movement calculations for navigating blocks.
- Uses Manhattan distance as the heuristic.
- Performs bounding box collision checks for each step.
- Limits recalculation frequency to reduce CPU load.

REFERENCES:

Bukkit Forum user. (n.d.). Lib: A pathfinding algorithm. Bukkit. https://bukkit.org/threads/lib-a-pathfinding-algorithm.129786

Supercent. (2023, May 22). Data structures for game developers. Medium. https://medium.com/supercent-blog/data-structures-for-game-developers-2d2e0adcc931

Penton, R. (n.d.). Data structure for game programmers [PDF]. https://cdn.preterhuman.net/texts/math/Data_Structure_And_Algorithms/Data%20Structure%20For%20Game%20Programers%20-%20Ron%20Penton.pdf

Minecraft Wiki. (n.d.). Java Edition level format. https://minecraft.wiki/w/Java_Edition_level_format

Minecraft Fandom. (n.d.). Development resources. https://minecraft.fandom.com/wiki/Development_resources

Minecraft Wiki. (n.d.). Map item format. https://minecraft.wiki/w/Map_item_format

Smooney, D. J. (2011, December 6). Modifying Minecraft: The source code. WordPress. https://darrenjsmooney.wordpress.com/2011/12/06/modifying-minecraft-the-source-code

Minecraft Fandom. (n.d.). NBT format. https://minecraft.fandom.com/wiki/NBT_format?utm_source=chatgpt.com

Minecraft Wiki. (n.d.). Chunk. https://minecraft.wiki/w/Chunk

Minecraft Wiki. (n.d.). Noise generator. https://minecraft.wiki/w/Noise_generator