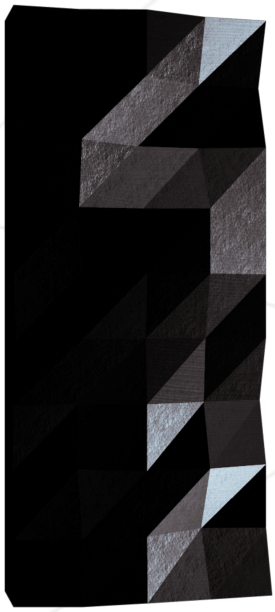


ft_minecraft

Pimp My World

Sylvain Lopez slopez@student.42lyon.fr
Ludovic Lemaire lulemair@student.42lyon.fr

Summary: ft_vox but harder

Version: 2.2

Contents

Chapter I

Preamble

Mojang Studios is a Swedish video game developer based in Stockholm. Founded by the independent game designer Markus Persson in 2009 as Mojang Specifications, it aimed to develop and release Persson's sandbox and survival video game, *Minecraft*. The studio inherited its name from a previous game project Persson had left two years earlier. Following Minecraft's release, Persson and Jakob Porsér incorporated the business as Mojang AB in late 2010, hiring Carl Manneh as the company's CEO. Early hires also included Daniel Kaplan and Jens Bergensten.

Minecraft became a massive success, eventually becoming the best-selling game of all time, driving Mojang's rapid growth. With Persson wanting to move on, he offered to sell his share in the company. In November 2014, Microsoft acquired Mojang through Xbox Game Studios (formerly Microsoft Studios). Persson, Porsér, and Manneh left Mojang, and Jonas Mårtensson stepped in as CEO. In May 2020, Mojang rebranded itself as Mojang Studios.

As of 2021, Mojang Studios employs approximately 600 people, with Jonas Mårtensson as CEO and Helen Chiang as studio head. Besides Minecraft, the studio developed Caller's Bane, a digital collectible card game, Crown and Council, a turn-based strategy game, and Minecraft Dungeons, a dungeon crawler. It also released smaller games as part of Humble Bundle game jams.

In 2011, Persson and Kaplan imagined a mix of Minecraft and Lego bricks, partnering with the Lego Group to create "Brickcraft," codenamed "Rex Kwon Do" (a nod to the movie Napoleon Dynamite). However, Mojang canceled the project after six months. Persson stated it was to focus on their own games, while Daniel Mathiasen from Lego blamed legal restrictions. Lego even considered acquiring Mojang but eventually decided against it, not realizing how big Minecraft would become.

Chapter II

Introduction

This project is the logical continuation of `ft_vox`. However, `ft_minecraft` aims to push things further, with a particular emphasis on Procedural Generation and the overall beauty of the rendered world.

You will also be diving into networking to enable multiplayer features. Let's face it, it's way more fun to build and destroy things with friends!

Chapter III

Objectives

This project focuses on two major aspects:

- **Advanced Procedural Generation:** Diverse biomes, lush vegetation, winding rivers, volumetric 3D clouds, and intricate cave systems filled with ores.
- **Advanced Rendering Effects:** Lighting, Shadows, Screen-Space Ambient Occlusion (SSAO), and much more.

Chapter IV

General Instructions

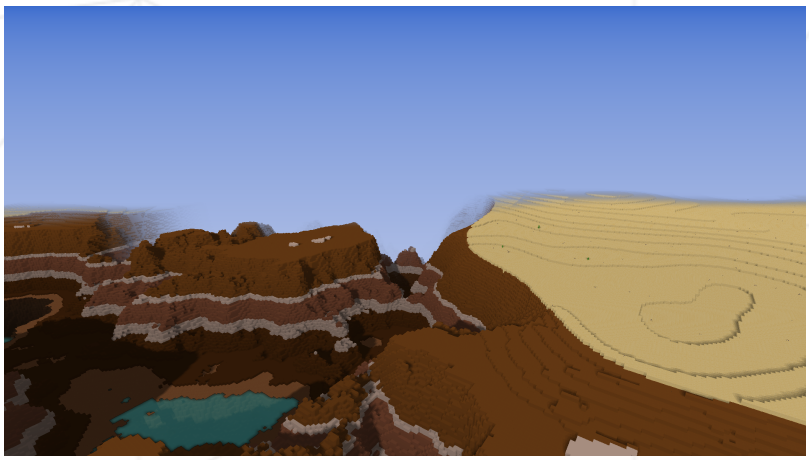
- You are free to use any programming language, but keep performance in mind. If you're unsure, consider `C`, `C++`, or `Rust`.
- For GPU calculations, you can use `Vulkan`, `Metal`, `WebGPU`, or `OpenGL/CL`.
  You cannot use a library that does the heavy lifting for you.
- You may use libraries for image loading, window management, audio, and mathematical operations (matrices, quaternions, vectors), but do not include them in your repository. Instead, write scripts to download and install them.
- Using pre-built libraries for terrain or biome generation is strictly **forbidden**. You must implement everything from scratch.
- The rendering must be consistently **smooth**, with a minimum of 25 FPS (on an i5 3.4 GHz, 8 GB RAM, Radeon Pro 570 4 GB, or equivalent specs).
- Any crash (uncaught exception, segmentation fault, abort, etc.) will be grounds for disqualification.
- Your program must run at 1080p or higher. Reducing the framebuffer resolution is not allowed.

Chapter V

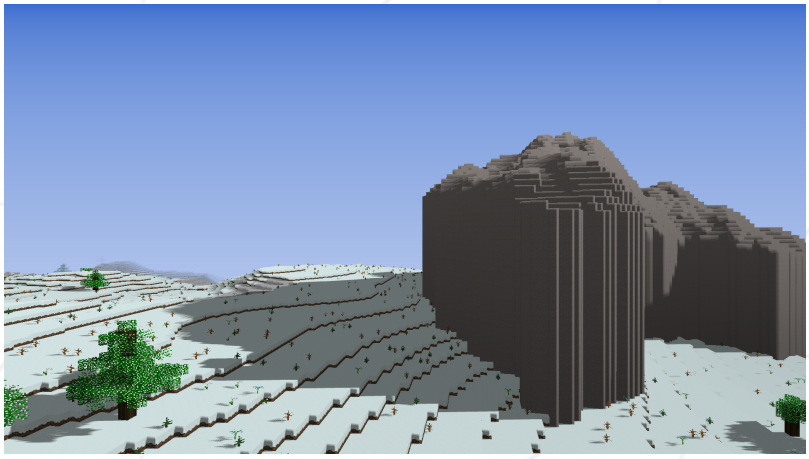
Mandatory Part

V.1 The World

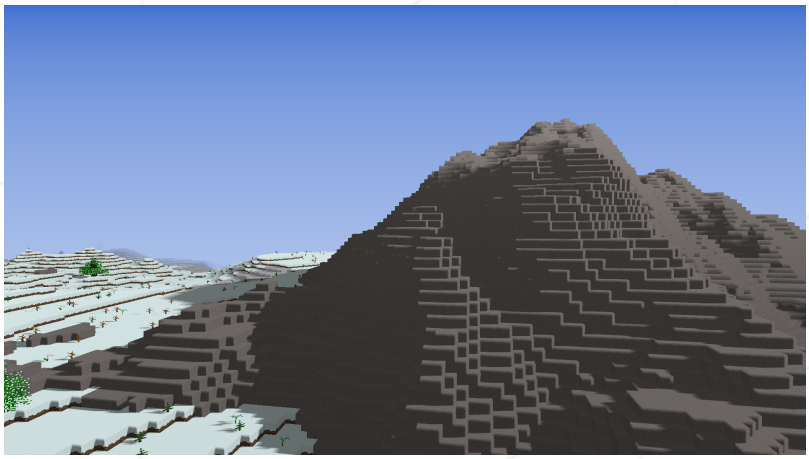
- The world must be generated on demand.
- You should be able to navigate through at least 5,000,000 cubes on the XZ plane. You are free to manage how surpassing this limit is handled: *(e.g., invisible walls, mirrored world, etc.)*.
- The terrain should not be uniform; you must implement different biomes like mountains, canyons, islands, etc.
- A minimum of 5 unique biomes is required: *(e.g., Mountain, Desert, Canyon, Swamp, Sequoia Forest, Island, Savanna, etc.)*.
- Each biome should have unique geography, elevation, vegetation, and distinct characteristics that make them feel truly unique.
- Biomes should transition smoothly and naturally without abrupt changes, as illustrated below:



Here are two biomes. A Canyon next to a Desert

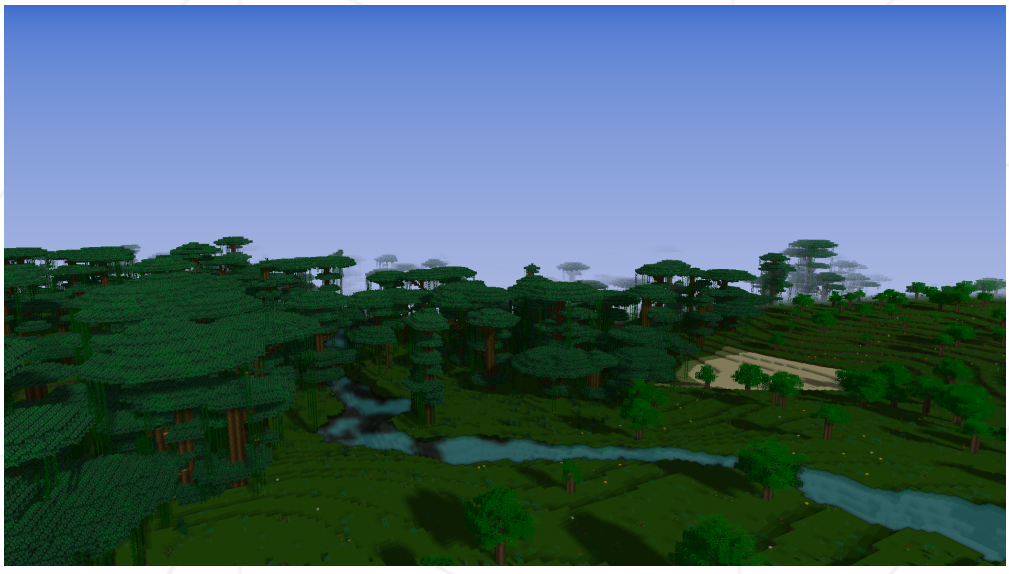


This is **wrong**



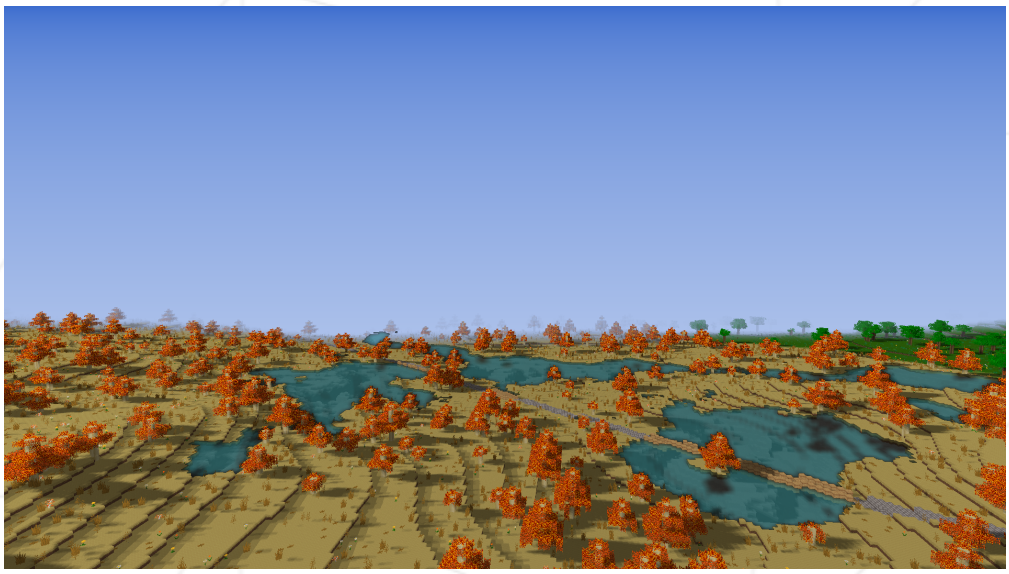
This is **right**

- There should be small plants, flowers, and mushrooms scattered throughout the world, as well as procedurally generated trees.
- Trees and leaves must not just be 3D models. Each tree should be generated with varying parameters like shape, height, width, and leaf density to ensure uniqueness.

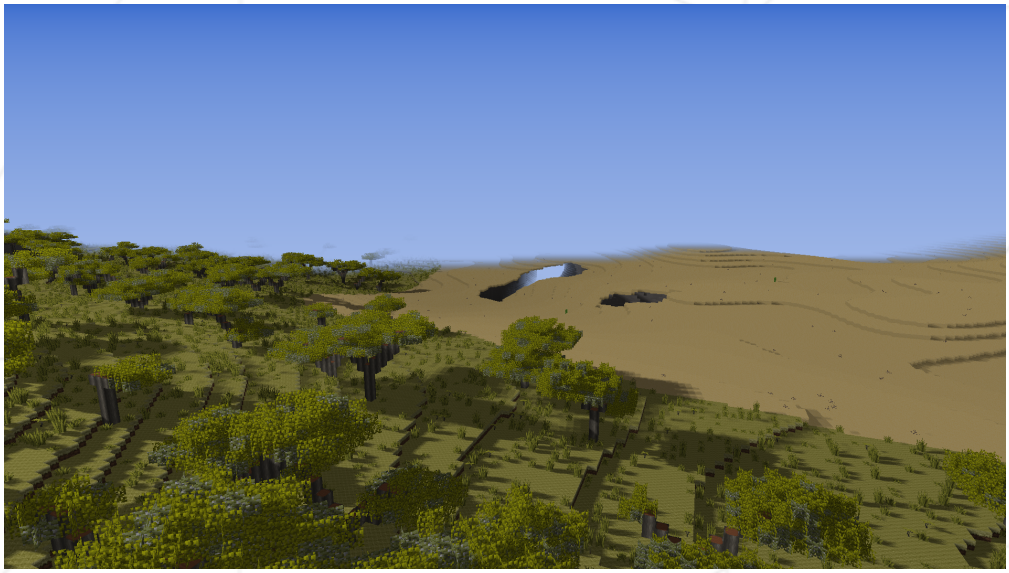


Jungle trees alongside Hill trees, each unique in shape and size.

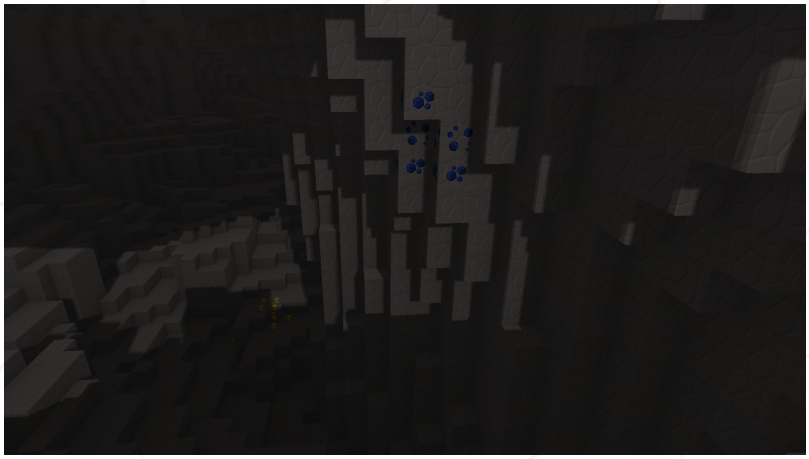
- There must be lakes and rivers meandering across the world, as well as natural cave entrances visible from the surface.
- These caves should feature realistic formations (wormhole style) and contain clusters of rare ores like gold and diamonds, not just simple noise-based distribution.



Lake and river, with a road/bridge passing through *(road/bridge isn't required)*



natural cave entrance



There must be clusters of ores just like in minecraft. Not a simple probability on each block.

- Monsters (like creepers or zombies) should spawn and chase you when you get close.
- 3D clouds should float across the world. They can either be represented as blocks (purely visual with no interaction) or as shaders.
- You should be able to pick up blocks after destroying them (just like in Minecraft) and place them wherever you want.
- Destroyed or placed blocks must be persistent.
  This means that if you unload a chunk and then reload it later, the modifications should still be present.

V.2 Graphic rendering

You are required to implement everything that was included in ft_vox, but with significant improvements.

Differences from ft_vox:

- Minimum render distance is increased from 160 to 260.
- You may use a sky shader instead of a skybox if desired.

Additionally, you are now expected to implement:

- Directional lighting
- Shadows
- Screen Space Ambient Occlusion (SSAO)
- Transparent water surfaces
- Far distance fog for better immersion

Be careful, your FPS should never drop below 25.

To achieve smooth rendering, you should balance the workload evenly between the CPU and GPU.

V.3 Camera

Movement is essential.
The keyboard should allow forward, backward, strafe right, and strafe left movement, relative to the camera's orientation.

You should also implement:

- Jump and sprint actions
- 360-degree mouse control on the Y-axis, with the ability to look up and down
- Walking speed of approximately 1 cube per second, and 2 cubes per second when sprinting
- A toggleable fly-mode, with running speed multiplied by 20 when flying

V.4 Sounds

The world must be immersive, which means:

- Each biome should have its own unique ambient music, with smooth transitions between them.
- Both players and monsters must have sounds for actions like walking, attacking, and swimming.
- The sound volume should dynamically adjust based on distance from the source.

V.5 Multiplayer and server

Exploring such a vast world alone would be a shame; that's why ft_minecraft must be multiplayer-ready.

Your server should allow at least four players to join simultaneously.
Players should be visible in the world, performing any actions such as walking, attacking, destroying blocks, and even getting killed by monsters.

All modifications to the world (block placement, block destruction) must be synchronized across all players and persistent even after reloading. Entity states (like monsters) should also be synchronized.

You are free to decide how the server-side is managed, either:

- Procedurally generate the world on the server and dispatch it to clients.
- Have each client generate the world and synchronize modifications with other clients.

Both approaches have their pros and cons. Think carefully before making a choice.

V.6 Interface

Players must have access to basic information:

- FPS, triangles, cube, and chunk counts must be displayed on-screen with a key toggle.
- A list of all connected players should also be available with a key toggle.

V.7 Other

Since we are moving closer to a real game, some basic functionalities are required:

- A simple gravity system that handles block collisions (excluding water).
- The ability to swim and dive, with optional slowed movement underwater.
- Visual rendering adaptations for underwater exploration (color filters, reduced visibility).
- Basic animations for walking and attacking, Minecraft-like in simplicity.

Chapter VI

Bonus Part

The possibilities are endless! Here are some suggestions:

- Procedurally generated villages.
- Crafting system.
- Realistic water simulation (dynamic flow and spreading).
- Growing plants (from seeds to maturity).
- A bow and arrow system similar to Minecraft.
- Nether portal that teleports you to another dimension.
- Cross-platform support (Windows, Mac, Linux).
- Stereo sound implementation.
- An online map interface (like Minecraft's Dynmap).

Chapter VII

Submission And Peer Evaluation

As usual, submit your work to your Git repository. Only the content available in your repository will be evaluated.

You must push all assets necessary for the project to run on the school server, within a reasonable size limit.
If your assets exceed 42 MB, you must provide a script to download or manually copy them.