FAQ

How to run it?

There's no GUI. Find a directory with Minecraft Anvil files (.mca). Copy a Minecraft "terrain.png" into that directory (do a google image search). Run from that directory.

How accurate is this as a Minecraft viewer?

Not very. Many Minecraft blocks are not handled correctly:

- No redstone, rails, or other "flat" blocks
- No signs, doors, fences, carpets, or other complicated geometry
- Stairs are turned into ramps
- Upper slabs turn into lower slabs
- Wood types only for blocks, not stairs, slabs, etc
- Colored glass becomes regular glass
- Glass panes become glass blocks
- Water is opaque
- Water level is incorrect
- No biome coloration
- Cactus is not shrunk, shows holes
- Chests are not shrunk
- Double-chests draw as two chests
- Pumpkins etc. are not rotated properly
- Torches are drawn hackily, do not attach to walls
- Incorrect textures for blocks that postdate terrain.png
- Transparent textures have black fringes due to non-premultiplied-alpha
- Skylight and block light are combined in a single value
- Only blocks at y=1..255 are shown (not y=0)
- If a 32x32x256 "quad-chunk" needs more than 800K quads, isn't handled (very unlikely)

Some of these are due to engine limitations, and some of these are because I didn't make the effort since my goal was to make a demo for stb_voxel_render.h, not to make a proper Minecraft viewer.

Could this be turned into a proper Minecraft viewer?

Yes and no. Yes, you could do it, but no, it wouldn't really resemble this code that much anymore.

You could certainly use this engine to render the parts of Minecraft it works for, but many of the things it doesn't handle it can't handle at all (stairs, water, fences, carpets, etc) because it uses low-precision coordinates to store voxel data.

You would have to render all of the stuff it doesn't handle through another rendering path. In a game (not a viewer) you would need such a path for movable entities like doors and carts anyway, so possibly handling other things that way wouldn't be so bad.

Rails, ladders, and redstone lines could be implemented by using tex2 to overlay those effects, but you can't rotate tex1 and tex2 independently, so there may be cases where the underlying texture needs a different rotation from the overlaid texture, which would require separate rendering. Handling redstone's brightness being different from underlying block's brightness would require separate rendering.

You can use the face-color effect to do biome coloration, but the change won't be smooth the way it is in Minecraft.

Why isn't building the mesh data faster?

Partly because converting from minecraft data is expensive.

Here is the approximate breakdown of an older version of this executable and lib that did the building single-threaded.

- 25% loading & parsing minecraft files (4/5ths of this is my crappy zlib)
- 18% converting from minecraft blockids & lighting to stb blockids & lighting
- 10% reordering from data[z][y][x] (minecraft-style) to data[y][x][z] (stb-style)
- 40% building mesh data
- 7% uploading mesh data to OpenGL

I did do significant optimizations after the above, so the final breakdown is different, but it should give you some sense of the costs.