An interview with STB about stb_voxel_render.h

Q: I suppose you really like Minecraft?

A: Not really. I mean, I do own it and play it some, and I do watch YouTube videos of other people playing it once in a while, but I'm not saying it's that great.

But I do love voxels. I've been playing with voxel rendering since the mid-late 90's when we were still doing software rendering and thinking maybe polygons weren't the answer. Once GPUs came along that kind of died off, at least until Minecraft brought it back to attention.

Q: Do you expect people will make a lot of Minecraft clones with this?

A: I hope not!

For one thing, it's a terrible idea for the developer. Remember before Minecraft was on the Xbox 360, there were a ton of "indie" clones (some maybe making decent money even), but then the real Minecraft came out and just crushed them (as far as I know). It's just not something you really want to compete with.

The reason I made this library is because I'd like to see more games with Minecraft's *art style*, not necessary its *gameplay*.

I can understand the urge to clone the gameplay. When you have a world made of voxels/blocks, there are a few things that become incredibly easy to do that would otherwise be very hard (at least for an indie) to do in 3D. One thing is that procedural generation becomes much easier. Another is that destructible environments are easy. Another is that you have a world where your average user can build stuff that they find satisfactory.

Minecraft is at a sort of local maximum, a sweet spot, where it leverages all of those easy-to-dos. And so I'm sure it's hard to look at the space of 'games using voxels' and move away from that local maximum, to give up some of that. But I think that's what people should do.

Q: So what else can people do with stb_voxel_render?

A: All of those benefits I mentioned above are still valid even if you stay away from the sweet spot. You can make a 3D roguelike without player-creation/destruction that uses procedural generation. You could make a shooter with pre-designed maps but destructible environments.

And I'm sure there are other possible benefits to using voxels/blocks. Hopefully this will make it easier for people to explore the space.

The library has a pretty wide range of features to allow people to come up with some distinctive looks. For example, the art style of Continue?9876543210 was one of the inspirations for trying to make the multitexturing capabilities flexible. I'm terrible at art, so this isn't really something I can come up with myself, but I tried to put in flexible technology that could be used multiple ways.

One thing I did intentionally was try to make it possible to make nicer looking ground terrain, using the half-height slopes and "weird slopes". There are Minecraft mods with drivable cars and they just go up these blocky slopes and, like, what? So I wanted you to be able to make smoother terrain, either just for the look, or for vehicles etc. Also, you can spatially cross-fade between two ground textures for that classic bad dirt/grass transition that has shipped in plenty of professional games. Of course, you could just use a separate non-voxel ground renderer for all of this. But this way, you can seamlessly integrate everything else with it. E.g. in your authoring tool (or procedural generation) you can make smooth ground and then cut a sharp-edged hole in it for a building's basement or whatever.

Another thing you can do is work at a very different scale. In Minecraft, a person is just under 2 blocks tall. In Ace of Spades, a person is just under 3 blocks tall. Why not 4 or 6? Well, partly because you just need a lot more voxels; if a meter is 2 voxels in Mineraft and 4 voxels in your game, and you draw the same number of voxels due to hardware limits, then your game has half the view distance of Minecraft. Since stb_voxel_render is designed to keep the

meshes small and render efficiently, you can push the view distance out further than Minecraft--or use a similar view distance and a higher voxel resolution. You could also stop making infinite worlds and work at entirely different scales; where Minecraft is 1 voxel per meter, you could have 20 voxels per meter and make a small arena that's 50 meters wide and 5 meters tall.

Back when the voxel game Voxatron was announced, the weekend after the trailer came out I wrote my own little GPU-accelerated version of the engine and thought that was pretty cool. I've been tempted many times to extract that and release it as a library, but I don't want to steal Voxatron's thunder so I've avoided it. You could use this engine to do the same kind of thing, although it won't be as efficient as an engine dedicated to that style of thing would be.

Q: What one thing would you really like to see somebody do?

A: Before Unity, 3D has seemed deeply problematic in the indie space. Software like GameMaker has tried to support 3D but it seems like little of note has been done with it.

Minecraft has shown that people can build worlds with the Minecraft toolset far more easily than we've ever seen from those other tools. Obviously people have done great things with Unity, but those people are much closer to professional developers; typically they still need real 3D modelling and all of that stuff.

So what I'd really like to see is someone build some kind of voxel-game-construction-set. Start with stb_voxel_render, maybe expose all the flexibility of stb_voxel_render (so people can do different things). Thrown in lua or something else for scripting, make some kind of editor that feels at least as good as Minecraft and Infinifactory, and see where that gets you.

Q: Why'd you make this library?

A: Mainly as a way of releasing this technology I've been working on since 2011 and seemed unlikely to ever ship myself. In 2011 I was playing the voxel shooter Ace of Spades. One of the maps that we played on was a partial port of Broville (which is the first Minecraft map in stb_voxel_render release trailer). I'd made a bunch of procedural level generators for the game, and I started trying to make a city generator inspired by Broville.

But I realized it would be a lot of work, and of very little value (most of my maps didn't get much play because people preferred to play on maps where they could charge straight at the enemies and shoot them as fast as possible). So I wrote my own voxel engine and started working on a procedural city game. But I got bogged down after I finally got the road generator working and never got anywhere with building generation or gameplay.

stb_voxel_render is actually a complete rewrite from scratch, but it's based a lot on what I learned from that previous work.

Q: About the release video... how long did that take to edit?

A: About seven or eight hours. I had the first version done in maybe six or seven hours, but then I realized I'd left out one clip, and when I went back to add it I also gussied up a couple other moments in the video. But there was something basically identical to it that was done in around six.

Q: Ok, that's it. Thanks, me.

A: Thanks *me!*