What are they?

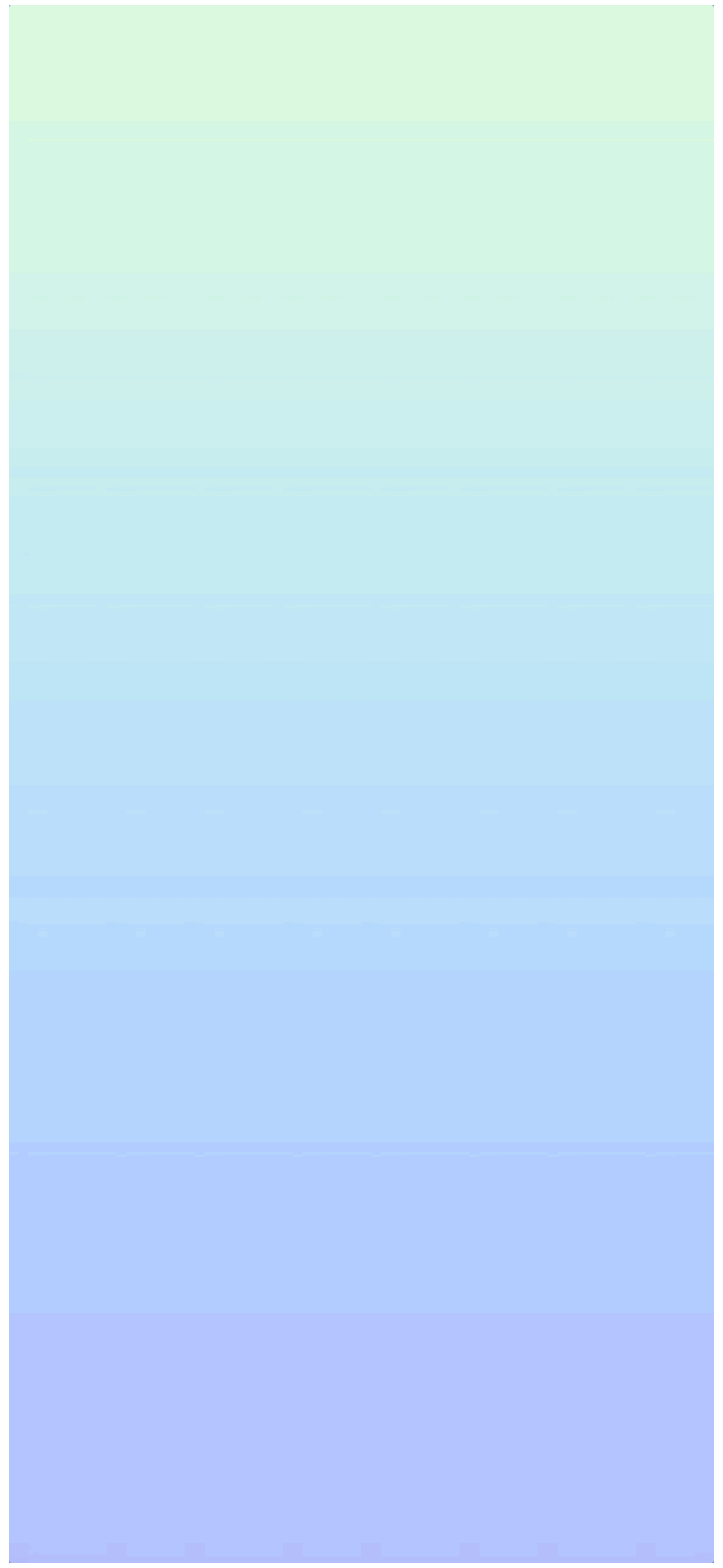
- programs mapping a pixel’s position to a color

- they run per pixel on the screen, in parallel

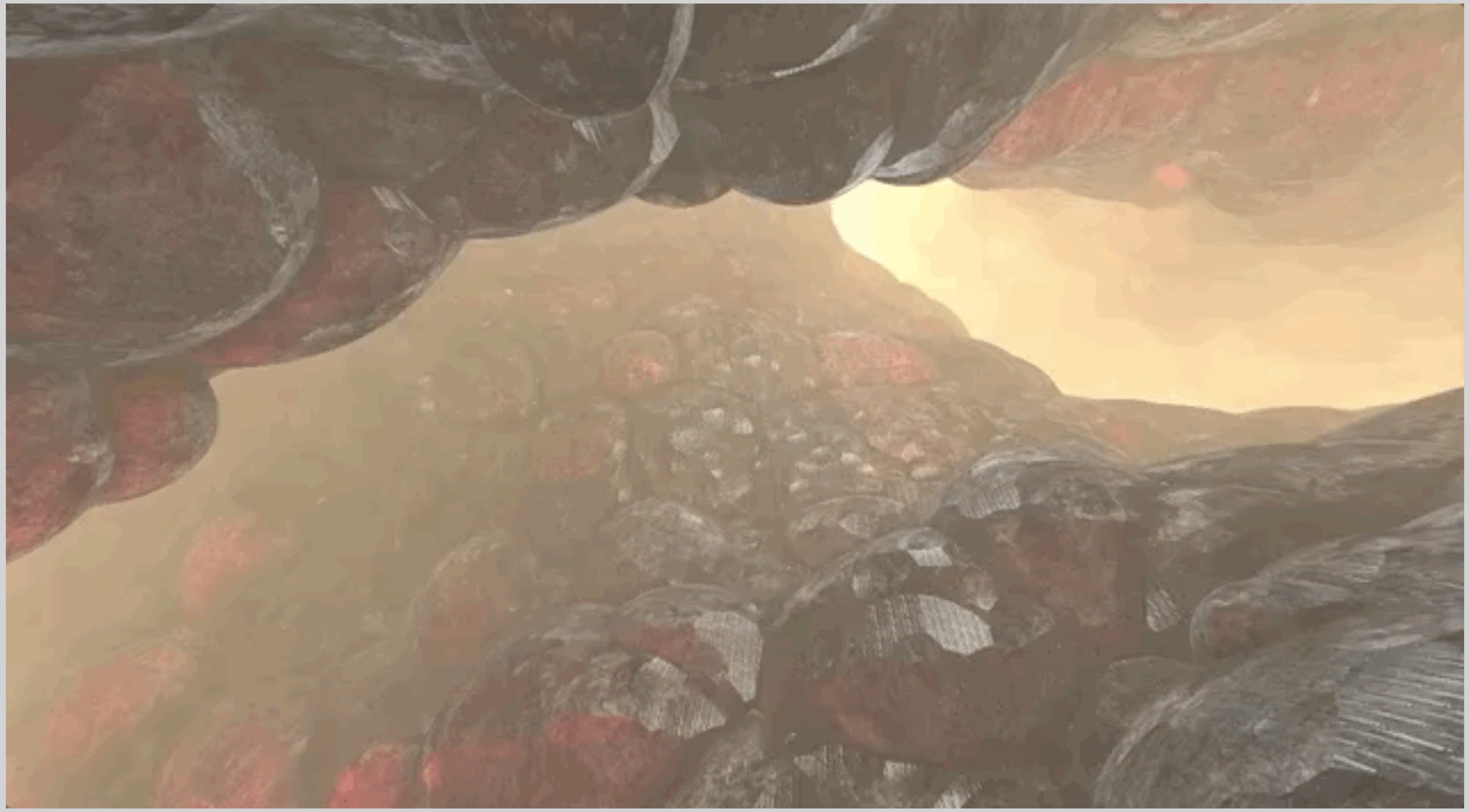
- they only have info about “the current” pixel and its position

- cant access neighboring pixels

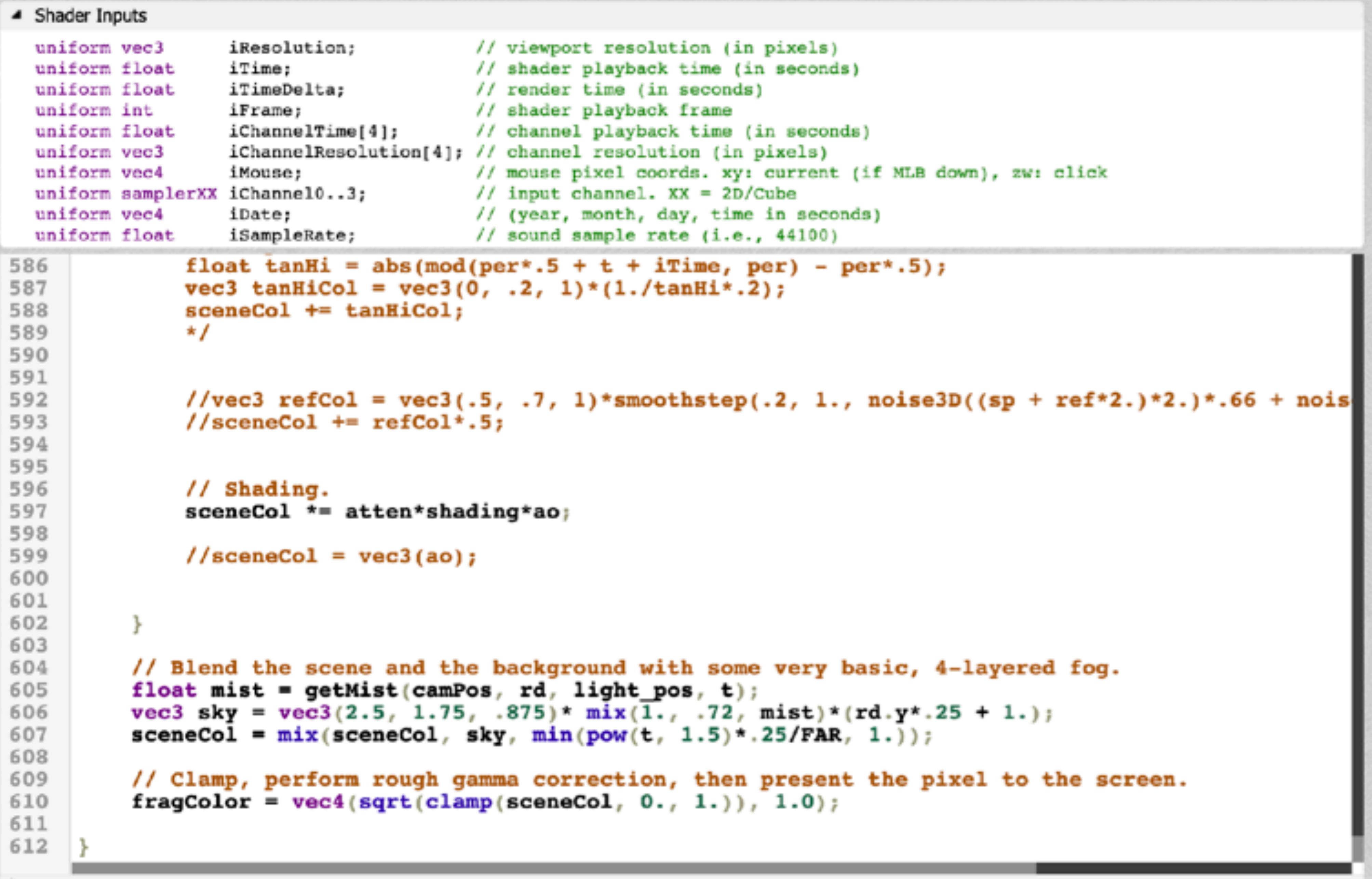
What are they?



- programs mapping a pixel’s position to a color
- they run per pixel on the screen, in parallel
- they only have info about “the current” pixel and its position
- cant access neighboring pixels



3D Cellular Tiling *Created by* ***Shane*** *in 2016-04-17*



```
uniform vec3      iResolution;           // viewport resolution (in pixels)
uniform float     iTime;                 // shader playback time (in seconds)
uniform float     iTimeDelta;            // render time (in seconds)
uniform int       iFrame;                // shader playback frame
uniform float     iChannelTime[4];       // channel playback time (in seconds)
uniform vec3      iChannelResolution[4]; // channel resolution (in pixels)
uniform vec4      iMouse;                // mouse pixel coords. xy: current (if MLB down), zw: click
uniform samplerXX iChannel0..3;          // input channel. XX = 2D/Cube
uniform vec4      iDate;                 // (year, month, day, time in seconds)
uniform float     iSampleRate;           // sound sample rate (i.e., 44100)
```

```
        float tanHi = abs(mod(per*.5 + t + iTime, per) - per*.5);
        vec3 tanHiCol = vec3(0, .2, 1)*(1./tanHi*.2);
        sceneCol += tanHiCol;
        */


        //vec3 refCol = vec3(.5, .7, 1)*smoothstep(.2, 1., noise3D((sp + ref*2.)*2.)*.66 + nois
        //sceneCol += refCol*.5;


        // Shading.
        sceneCol *= atten*shading*ao;

        //sceneCol = vec3(ao);


    }

    // Blend the scene and the background with some very basic, 4-layered fog.
    float mist = getMist(camPos, rd, light_pos, t);
    vec3 sky = vec3(2.5, 1.75, .875)* mix(1., .72, mist)*(rd.y*.25 + 1.);
    sceneCol = mix(sceneCol, sky, min(pow(t, 1.5)*.25/FAR, 1.));

    // Clamp, perform rough gamma correction, then present the pixel to the screen.
    fragColor = vec4(sqrt(clamp(sceneCol, 0., 1.)), 1.0);

}
```