

Shader Inputs

```

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)

586     float tanHi = abs(mod(per*.5 + t + iTime, per) - per*.5);
587     vec3 tanHiCol = vec3(0, .2, 1)*(1./tanHi*.2);
588     sceneCol += tanHiCol;
589     */
590
591
592     //vec3 refCol = vec3(.5, .7, 1)*smoothstep(.2, 1., noise3D((sp + ref*2.)*2.)*.66 + nois
593     //sceneCol += refCol*.5;
594
595
596     // Shading.
597     sceneCol *= atten*shading*ao;
598
599     //sceneCol = vec3(ao);
600
601
602 }
603
604 // Blend the scene and the background with some very basic, 4-layered fog.
605 float mist = getMist(camPos, rd, light_pos, t);
606 vec3 sky = vec3(2.5, 1.75, .875)* mix(1., .72, mist)*(rd.y*.25 + 1.);
607 sceneCol = mix(sceneCol, sky, min(pow(t, 1.5)*.25/FAR, 1.));
608
609 // Clamp, perform rough gamma correction, then present the pixel to the screen.
610 fragColor = vec4(sqrt(clamp(sceneCol, 0., 1.)), 1.0);
611
612 }
```

3D Cellular Tiling created by Shan in 2016-04-17







Shader Inputs

```
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)

586 float tanHi = abs(mod(per*.5 + t + iTime, per) - per*.5);
587 vec3 tanHiCol = vec3(0, .2, 1)*(1./tanHi*.2);
588 sceneCol += tanHiCol;
589 */
590
591
592 //vec3 refCol = vec3(.5, .7, 1)*smoothstep(.2, 1., noise3D((sp + ref*2.)*2.)*.66 + nois
593 //sceneCol += refCol*.5;
594
595
596 // Shading.
597 sceneCol *= atten*shading*ao;
598
599 //sceneCol = vec3(ao);
600
601
602 }
603
604 // Blend the scene and the background with some very basic, 4-layered fog.
605 float mist = getMist(camPos, rd, light_pos, t);
606 vec3 sky = vec3(2.5, 1.75, .875)* mix(1., .72, mist)*(rd.y*.25 + 1.);
607 sceneCol = mix(sceneCol, sky, min(pow(t, 1.5)*.25/FAR, 1.));
608
609 // Clamp, perform rough gamma correction, then present the pixel to the screen.
610 fragColor = vec4(sqrt(clamp(sceneCol, 0., 1.)), 1.0);
611
612 }
```

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

On Android

- < Android 13 can use only pre-built shaders: **BitmapShader**, **LinearGradient**, etc
- with Android 13 can use programmable **RuntimeShaders** written in AGSL
- GPU level effects without direct OpenGL
- pass shaders as ***string*** into a **RuntimeShader** object!

```
val shader = RuntimeShader("""  
    // Shader code here in AGSL  
    """)  
  
// Make a Brush  
val brush = ShaderBrush(shader)  
  
// Canvas() / DrawScope  
onDraw = { value →  
    // Use it to paint anything!  
    drawRect(brush)  
}
```