

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

SKSL(AGSL)-shaders.skia.org

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 do
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)
```

```
1 //https://twitter.com/XorDev/status/1475524322785640455
2 void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
3     vec4 o = vec4(0.0);
4     vec2 p = vec2(0.0), c=p, u=fragCoord.xy*2.-iResolution.xy;
5     float a;
6     for (float i=0.0; i<4e2; i++) {
7         a = i/2e2-1.;
8         p = cos(i*2.4+iTime+vec2(0.0,11.0))*sqrt(1.-a*a);
9         c = u/iResolution.y+vec2(p.x,a)/(p.y+2.);
10        o += (cos(i+vec4(0.0,2.0,4.0,0.0))+1.)/dot(c,c)*(1.-p.y)/3e4;
11    }
12    fragColor = o;
13 }
14
```

GLSL-shadertoy.com

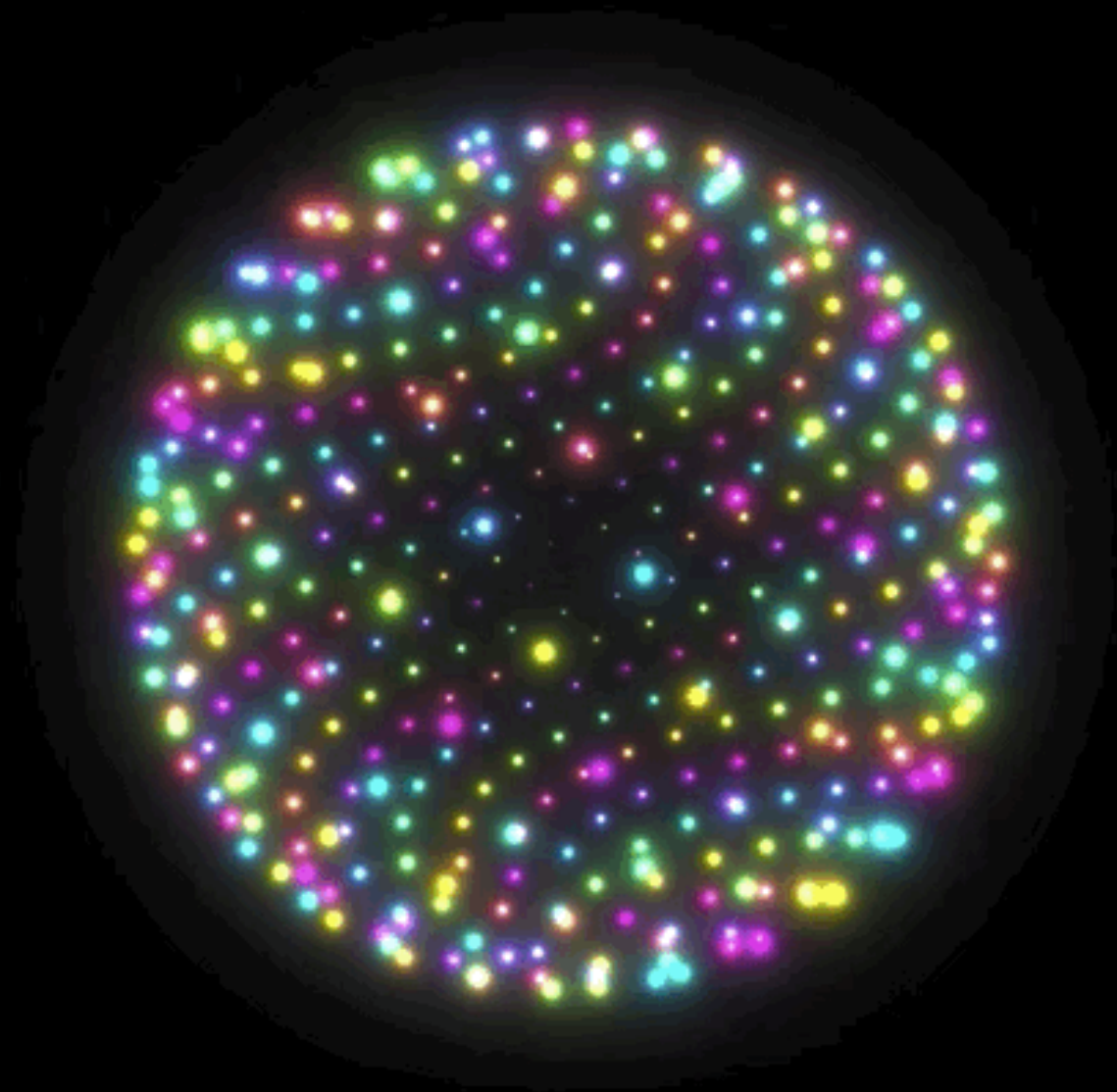
Shader Inputs

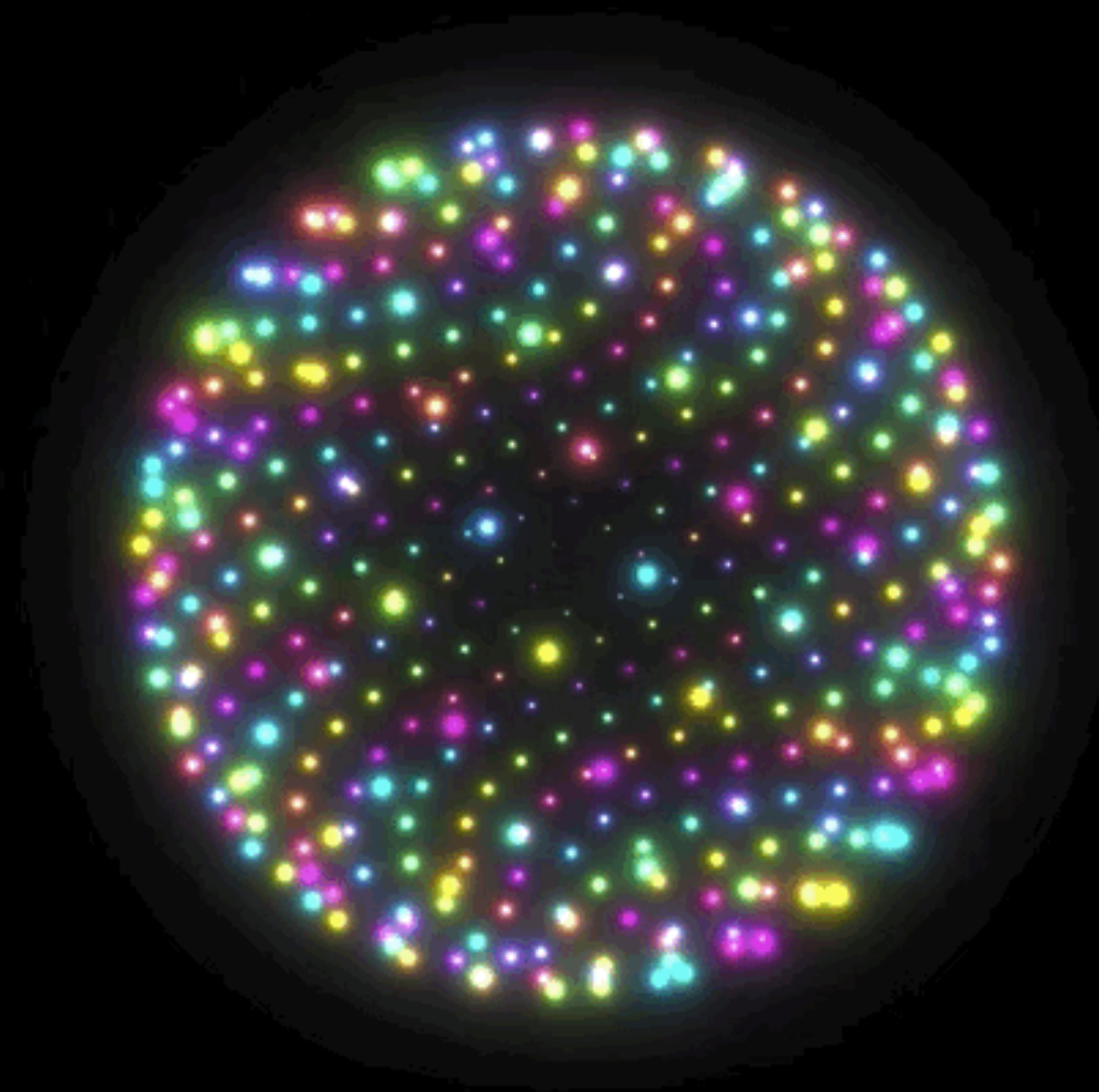
```
uniform float3 iResolution; // Viewport resolution (pixels)
uniform float iTime;        // Shader playback time (s)
uniform float4 iMouse;       // Mouse drag pos=.xy Click pos=.zw (pixels)
uniform float3 iImageResolution; // iImage1 resolution (pixels)
uniform shader iImage1;      // An input image.
```

```
1 // Source: @XorDev https://twitter.com/XorDev/status/1475524322785
2 vec4 main(vec2 fragCoord) {
3     vec4 o = vec4(0.0);
4     vec2 p = vec2(0.0), c=p, u=fragCoord.xy*2.-iResolution.xy;
5     float a;
6     for (float i=0; i<4e2; i++) {
7         a = i/2e2-1.;
8         p = cos(i*2.4+iTime+vec2(0.0,11.0))*sqrt(1.-a*a);
9         c = u/iResolution.y+vec2(p.x,a)/(p.y+2.);
10        o += (cos(i+vec4(0.0,2.0,4.0,0.0))) + 1.) / dot(c,c) * (1.-p.y) / 3e4;
11    }
12    return o;
13 }
14
```







GLSL - shadertoy.com

SKSL (~AGSL) - shaders.skia.org

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 do
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)
```

```
1 //https://twitter.com/XorDev/status/1475524322785640455
2 void mainImage( out vec4 fragColor, in vec2 fragCoord ) {
3   vec4 o = vec4(0.0);
4   vec2 p = vec2(0.0), c=p, u=fragCoord.xy*2.-iResolution.xy;
5   float a;
6   for (float i=0.0; i<4e2; i++) {
7     a = i/2e2-1.;
8     p = cos(i*2.4+iTime+vec2(0.0,11.0))*sqrt(1.-a*a);
9     c = u/iResolution.y+vec2(p.x,a)/(p.y+2.);
10    o += (cos(i+vec4(0.0,2.0,4.0,0.0))+1.)/dot(c,c)*(1.-p.y)/3e4;
11  }
12  fragColor = o;
13 }
14
```

Shader Inputs

```
uniform float3 iResolution; // Viewport resolution (pixels)
uniform float iTime; // Shader playback time (s)
uniform float4 iMouse; // Mouse drag pos=.xy Click pos=.zw (pixels)
uniform float3 iImageResolution; // iImage1 resolution (pixels)
uniform shader iImage1; // An input image.
```

```
1 // Source: @XorDev https://twitter.com/XorDev/status/1475524322785
2 vec4 main(vec2 fragCoord) {
3   vec4 o = vec4(0.0);
4   vec2 p = vec2(0.0), c=p, u=fragCoord.xy*2.-iResolution.xy;
5   float a;
6   for (float i=0; i<4e2; i++) {
7     a = i/2e2-1.;
8     p = cos(i*2.4+iTime+vec2(0.0,11.0))*sqrt(1.-a*a);
9     c = u/iResolution.y+vec2(p.x,a)/(p.y+2.);
10    o += (cos(i+vec4(0.0,2.0,4.0,0.0))+1.)/dot(c,c)*(1.-p.y)/3e4;
11  }
12  return o;
13 }
14
```


// AGSL 

```
uniform float2 iResolution; // Viewport resolution (px)
uniform float iTime; // Shader playback time (s)
```

```
vec4 main(in float2 fragCoord) {
```

```
    // Normalized pixel coordinates (from 0 to 1)
```

```
    vec2 uv = fragCoord/iResolution.xy;
```

```
    // Time varying pixel color
```

```
    vec3 col = 0.8
```

```
    + 0.2*cos(iTime*2.0+uv.xxx*2.0+vec3(1,2,4));
```

```
    // Output to screen
```

```
    return vec4(col,1.0);
```

```
}
```

