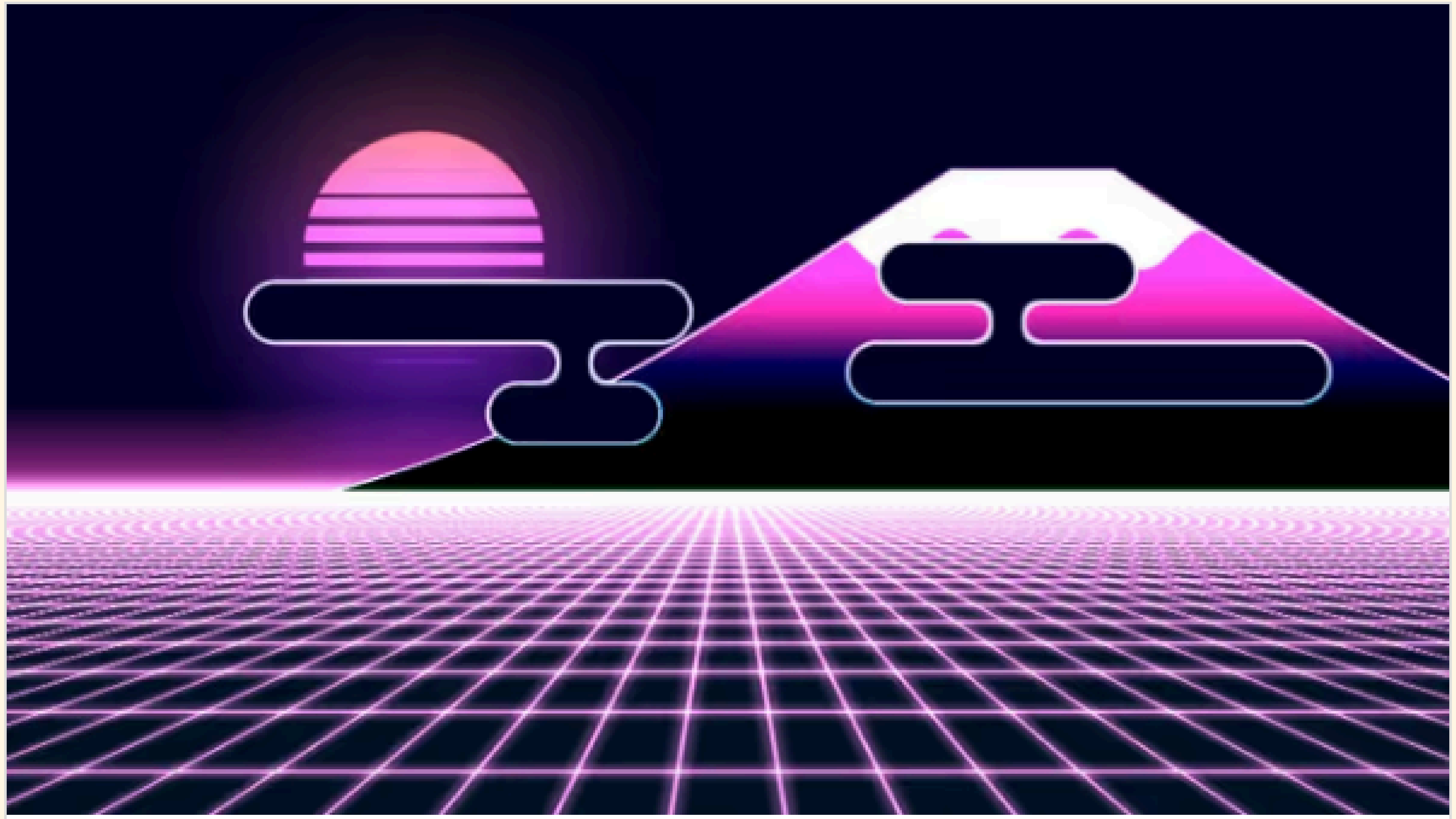
GLSL

OpenGL Shader Language

WHAT IS A SHADER?

A shader is code that will calculate positions and colors to determine the pixel color on a screen. In short, it is code to render an image.



Shadertoy

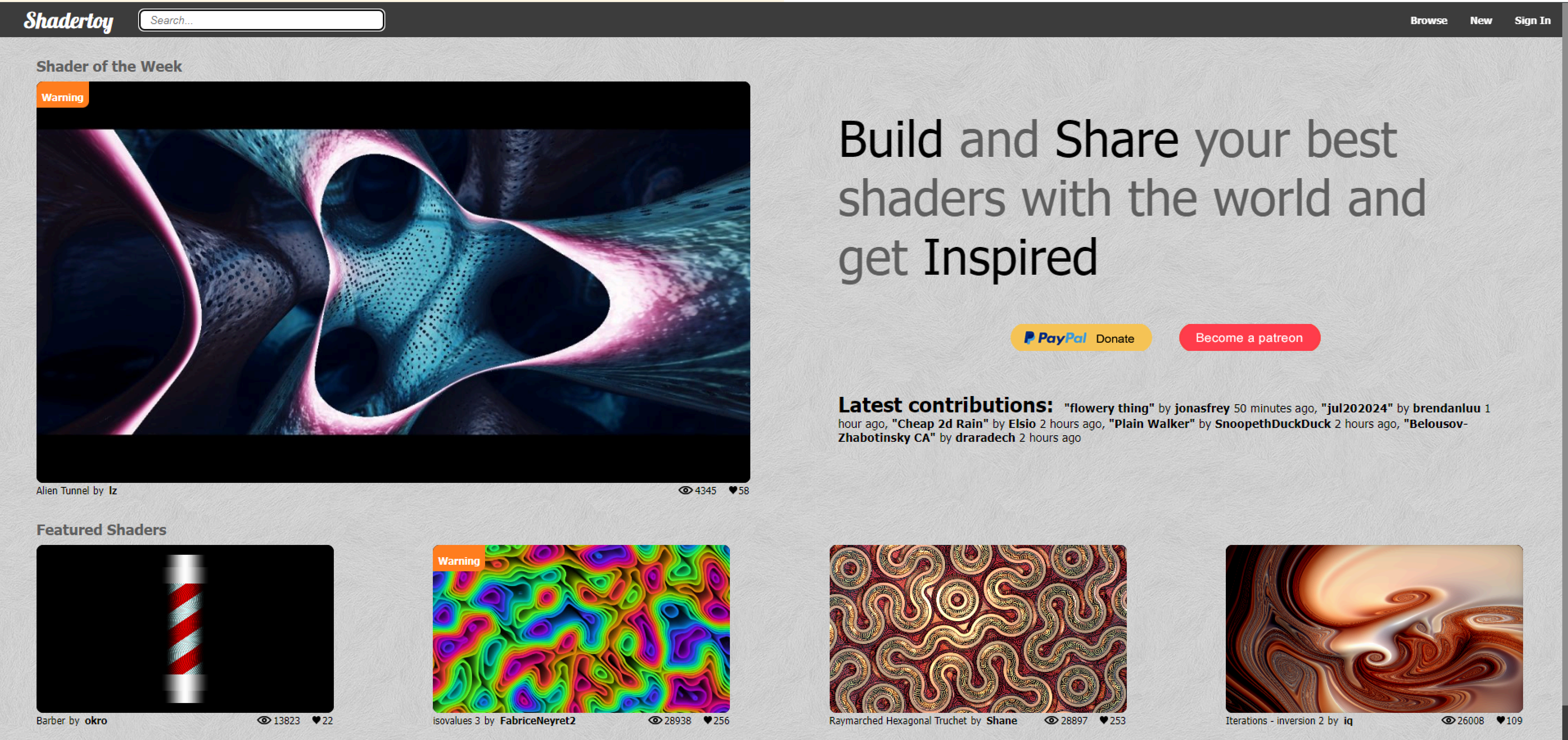
Build shaders, share them, and learn from the best community.

shadertoy.com /

Click me to learn more!

LETS MAKE A SHADER!

Let’s start by making an account on ShaderToy.
https://www.shadertoy.com/



Shader of the Week

Alien Tunnel by Iz 4345 58

Build and Share your best shaders with the world and get Inspired

Latest contributions: **"flowery thing"** by **jonasfrey** 50 minutes ago, **"jul202024"** by **brendanluu** 1 hour ago, **"Cheap 2d Rain"** by **Elsio** 2 hours ago, **"Plain Walker"** by **SnoopethDuckDuck** 2 hours ago, **"Belousov-Zhabotinsky CA"** by **draradech** 2 hours ago

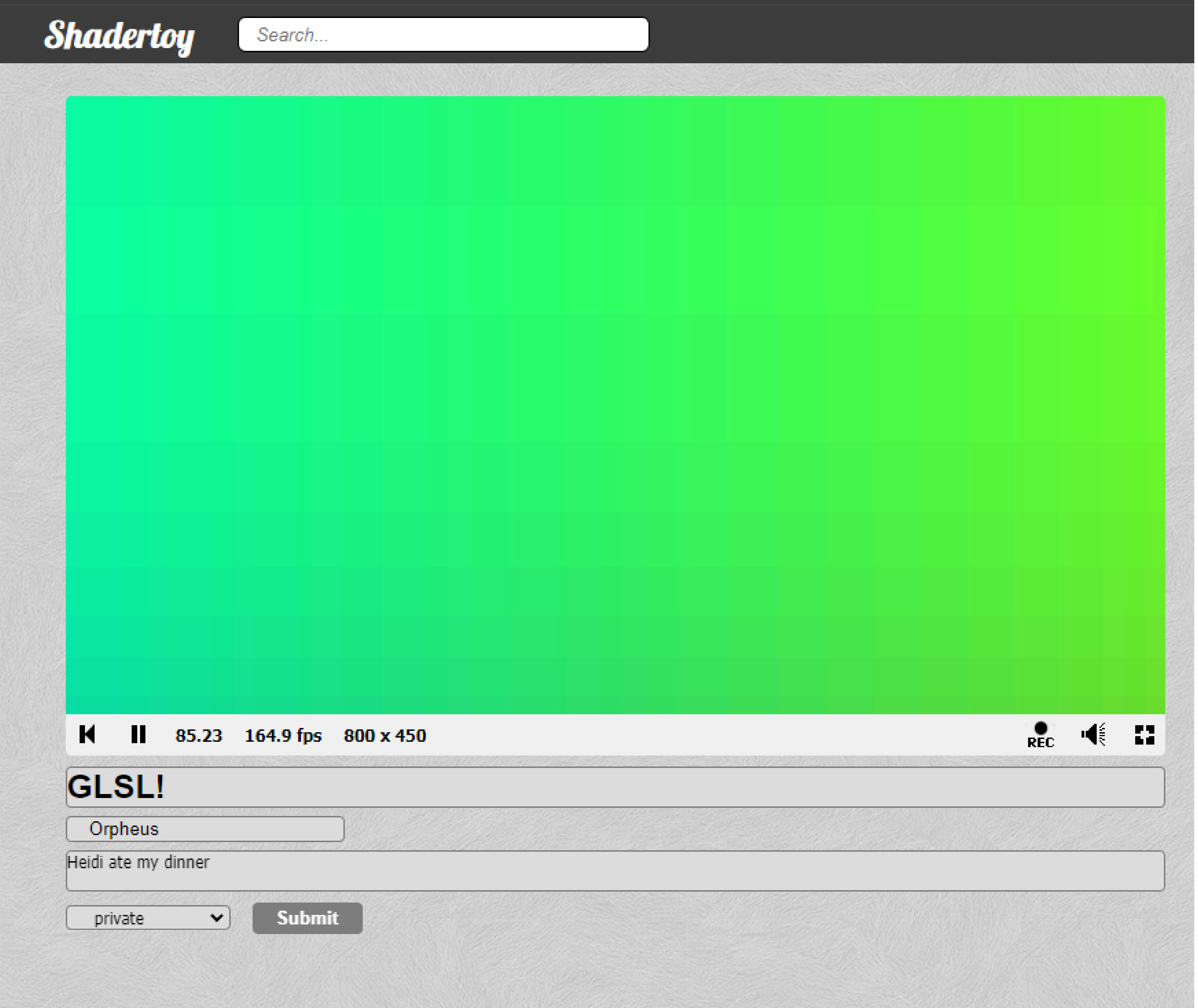
Featured Shaders

Barber by **okro** 13823 22

isovalues 3 by **FabriceNeyret2** 28938 256

Raymarched Hexagonal Truchet by **Shane** 28897 253

Iterations - inversion 2 by **iq** 26008 109



GLSL!

Orpheus

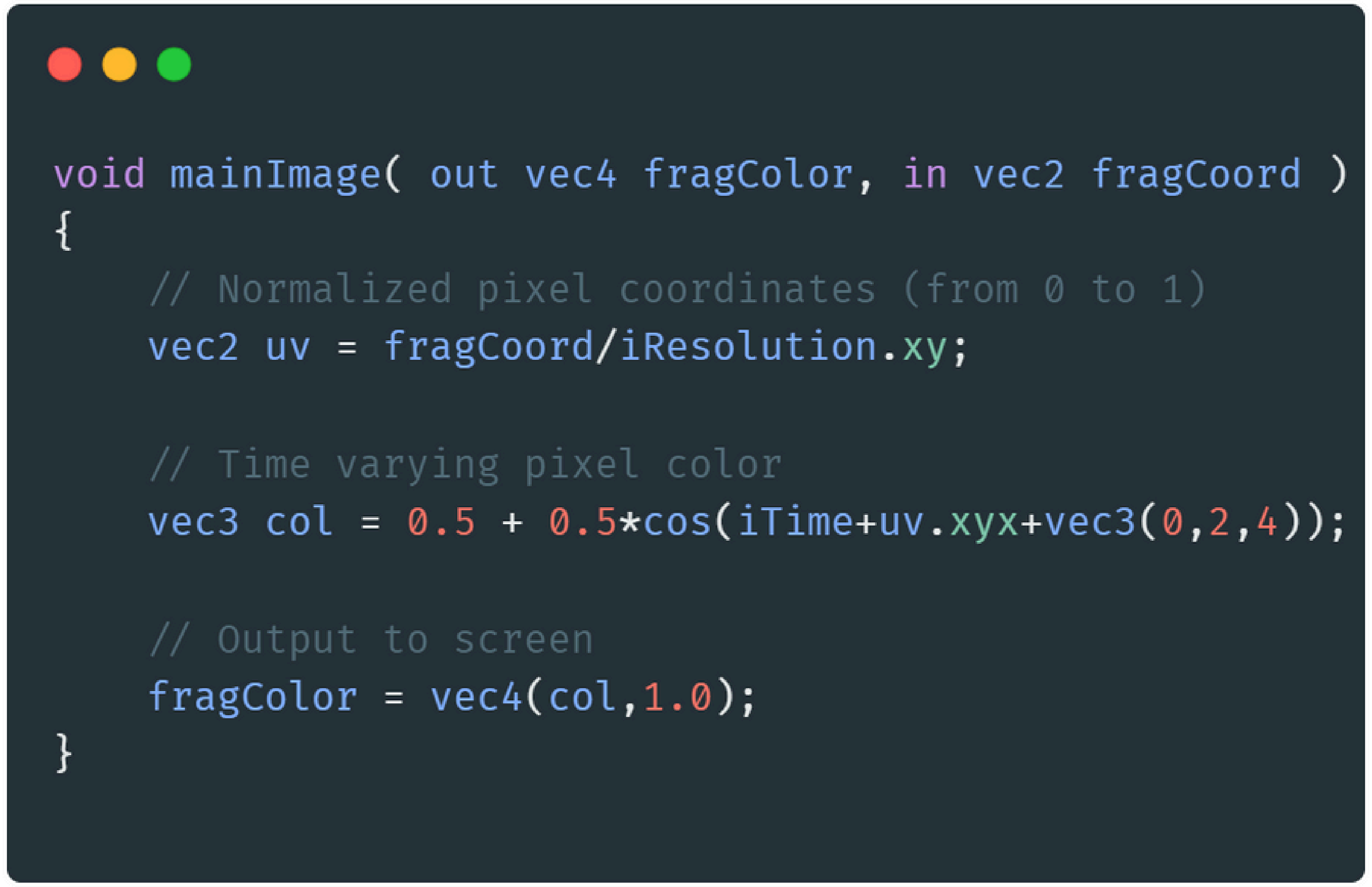
Heidi ate my dinner

private

Create a new shader, give it a name, tag, and description! (Be creative…)

THIS IS WHAT YOU WILL SEE!

When dealing with shaders, we think in pixels. In this case, every pixel is calculated as a different color!

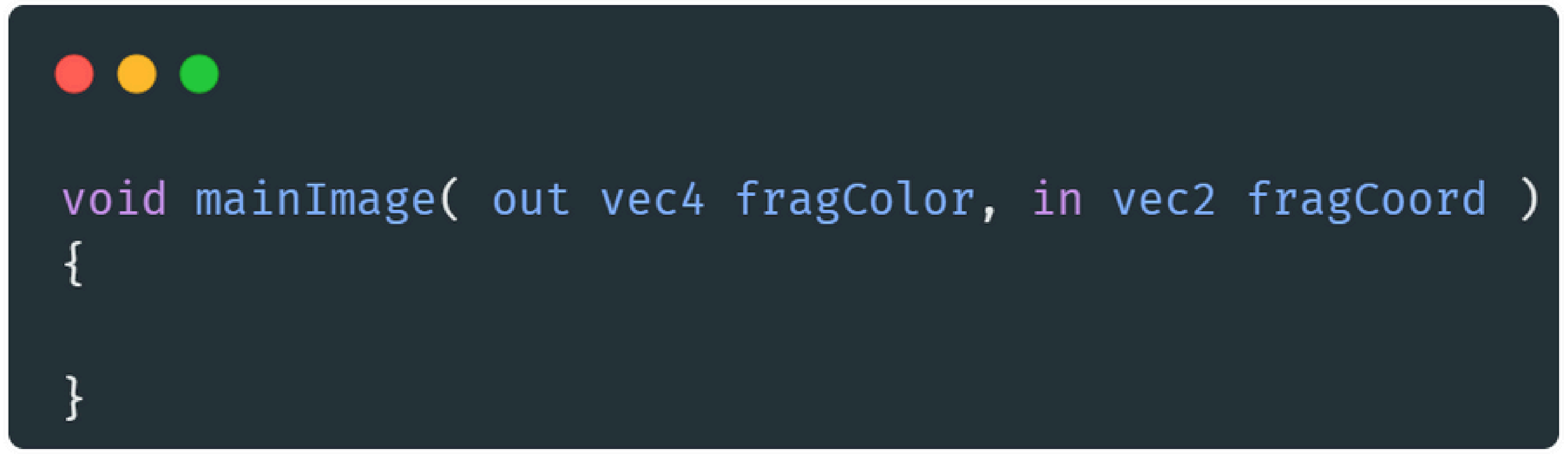


```glsl
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    // Normalized pixel coordinates (from 0 to 1)
    vec2 uv = fragCoord/iResolution.xy;

    // Time varying pixel color
    vec3 col = 0.5 + 0.5*cos(iTime+uv.xyx+vec3(0,2,4));

    // Output to screen
    fragColor = vec4(col,1.0);
}
```

LET'S START FROM THE BEGINNING

Delete everything inside void mainImage!

There are a couple parameters that we can see, vec4 is a vector with 4 different components, and vec2 is a vector with 2 different components. We can see that there is an input of fragCoord, and an output of fragColor. In this case, we are stating that the position of the fragment is the input, and we are going to write code to show the output of that fragment or pixel.



```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{

}
```

INITIALIZE THE COORDINATE PLANE

Here, we are creating a new vector, with 2 components named uv. As we know, XYZ are axes of 3D coordinate systems, but in our case we will be dealing with 2D, with U being horizontal (X) and V being vertical (Y).

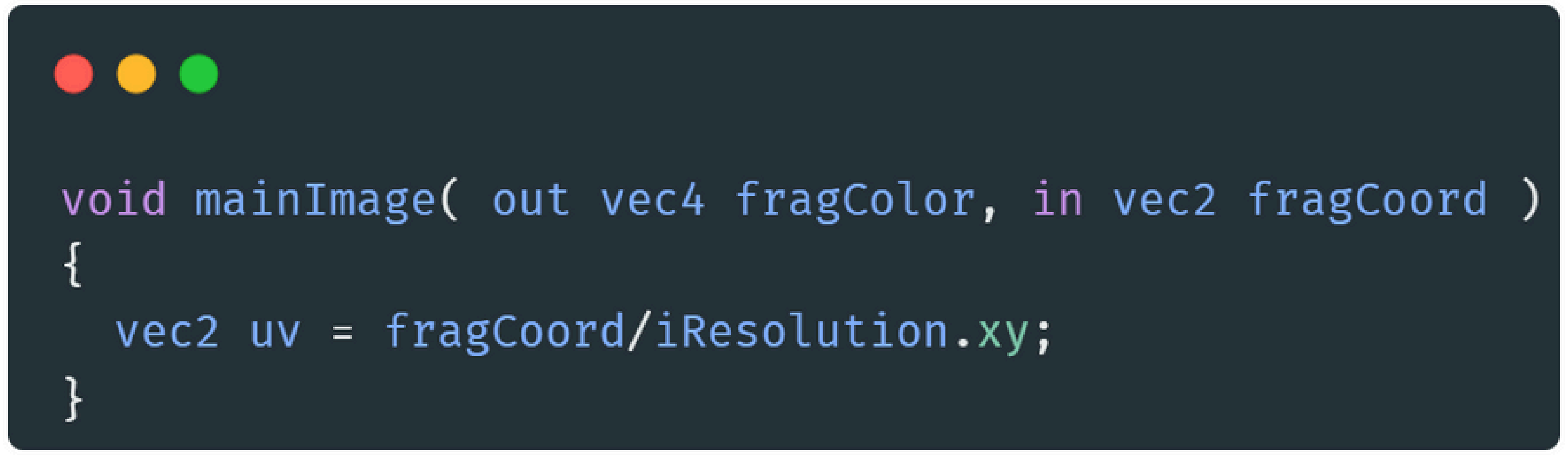
Since everyone has different screen resolutions, we are going to initialize the vector with a coordinate plane.

`fragCoord/iResolution.xy;`

This states that the bottom left corner will be (0,0), and the top right corner will be (1,1).

Copy this code into mainImage!

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



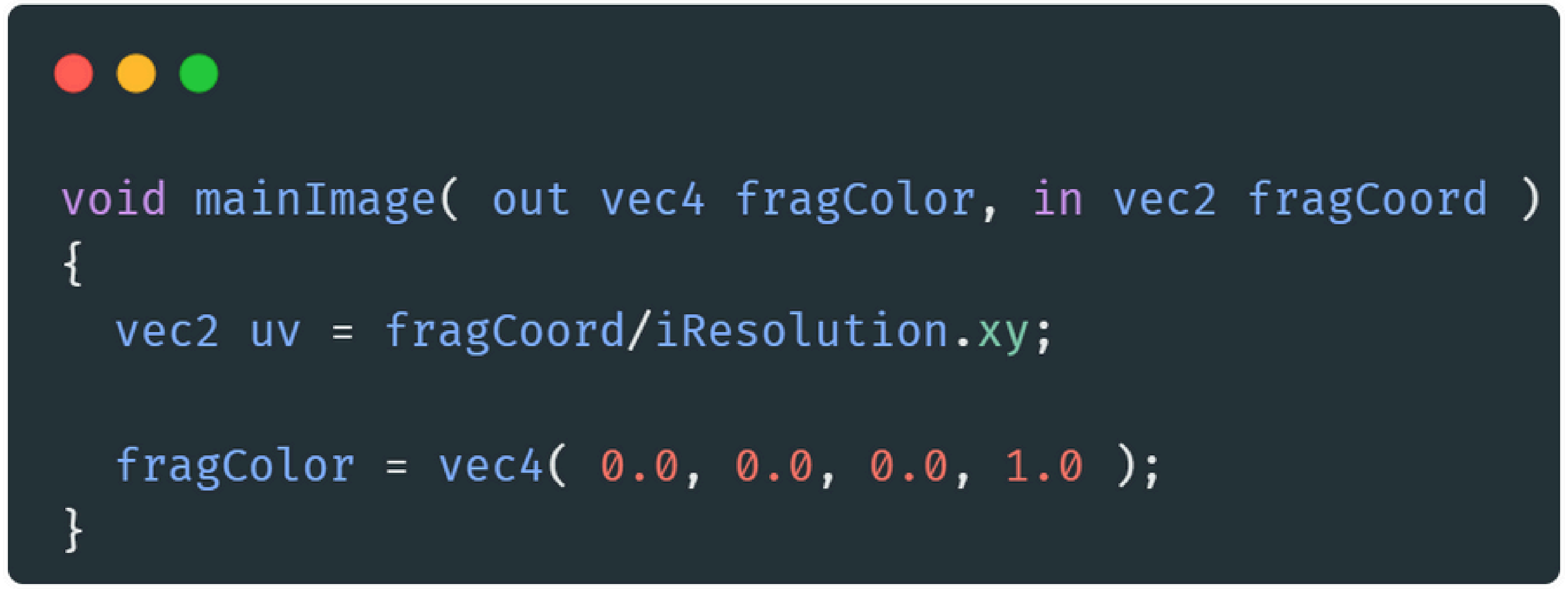
```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
  vec2 uv = fragCoord/iResolution.xy;
}
```

DISPLAYING COLORS

fragColor is a vec4 vector, meaning it consists of 4 components. These components are (R,G,B,a)! In our case, we will not worry about the last value. Try changing these values and run the code. What do you see?

Copy this code into mainImage!

```
fragColor = vec4( 0.0, 0.0, 0.0, 1.0 );
```



```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
  vec2 uv = fragCoord/iResolution.xy;

  fragColor = vec4( 0.0, 0.0, 0.0, 1.0 );
}
```

DISPLAYING COLORS

Lets try to create a gradient!

How do you think we could increment the color as it goes higher?

Hint: as uv.x gets higher, RGB gets higher

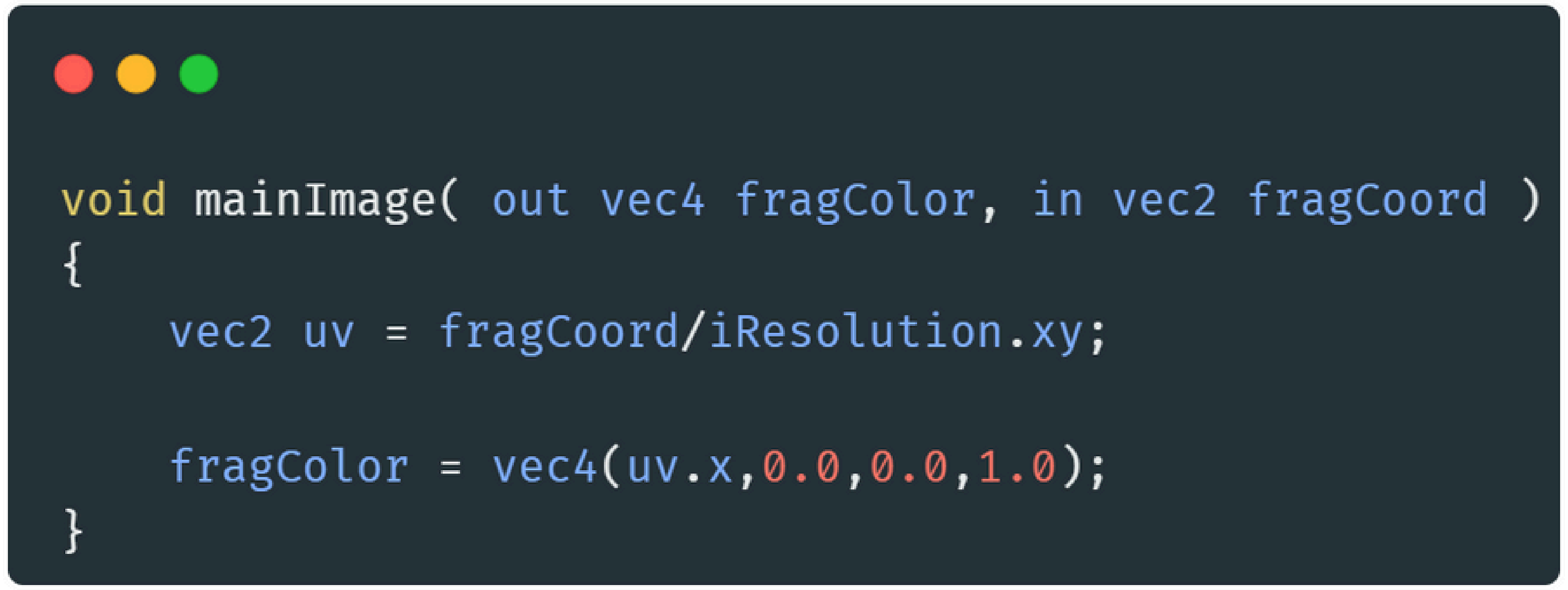
The solution to this is to simply place uv.x or uv.y in the fragColor vector!

As you can see, the higher the x value, the higher the red value is!

We can do multiple gradients upon multiple axes as such:

```
fragColor = vec4(uv.x,0.0,uv.y,1.0);
```

It is visible that there is a blue gradient going up, and a red gradient going right.



```
void mainImage( out vec4 fragColor, in vec2 fragCoord )
{
    vec2 uv = fragCoord/iResolution.xy;

    fragColor = vec4(uv.x,0.0,0.0,1.0);
}
```