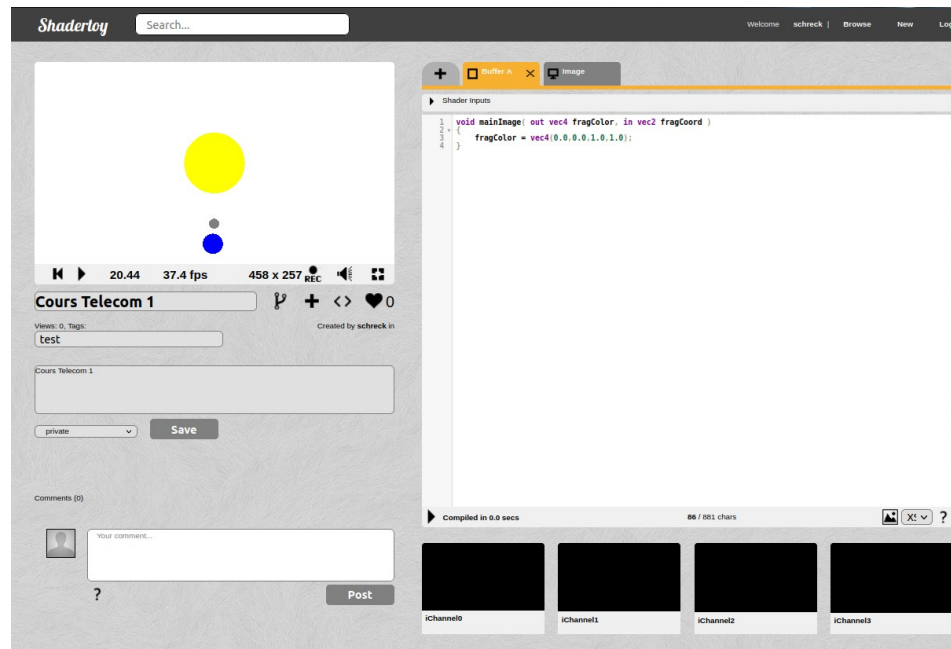


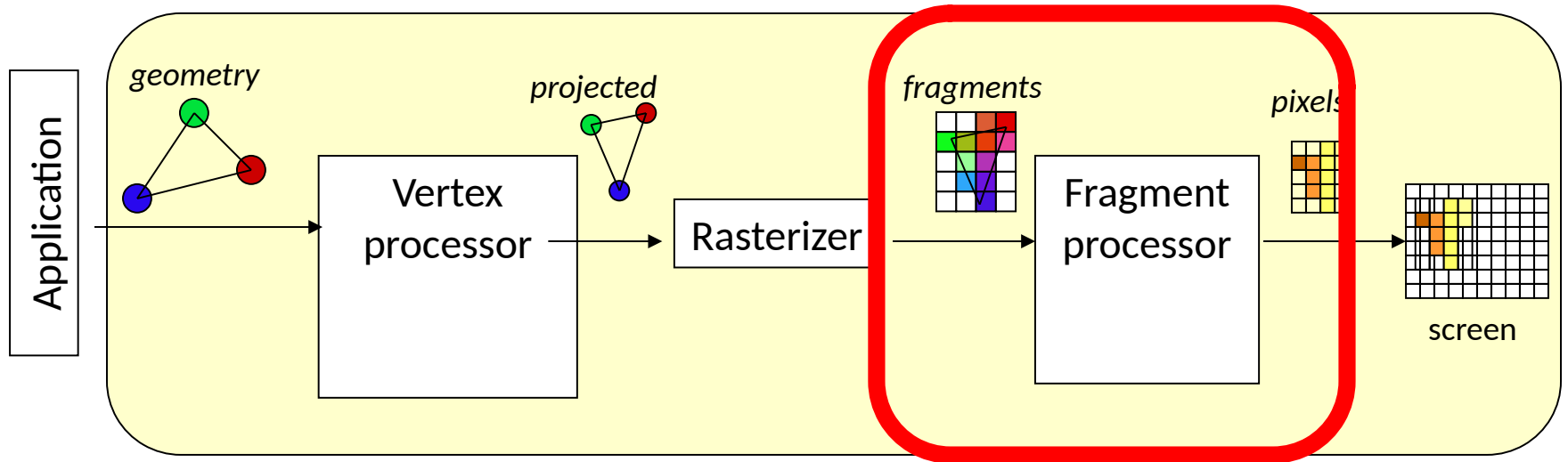
Intro Shadertoy

Fragment shader: language GLSL

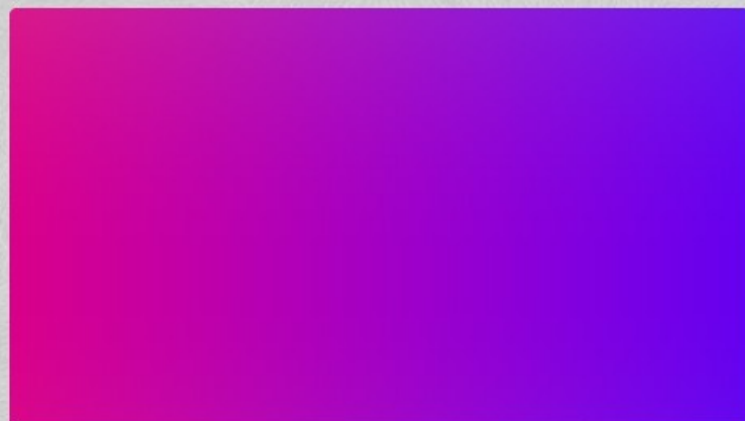
<https://www.shadertoy.com/>
<https://shaderoo.org>



GPU pipeline in a nutshell



Shadertoy: fragment shader



Name of your shader

Tags, comma separated. For example:

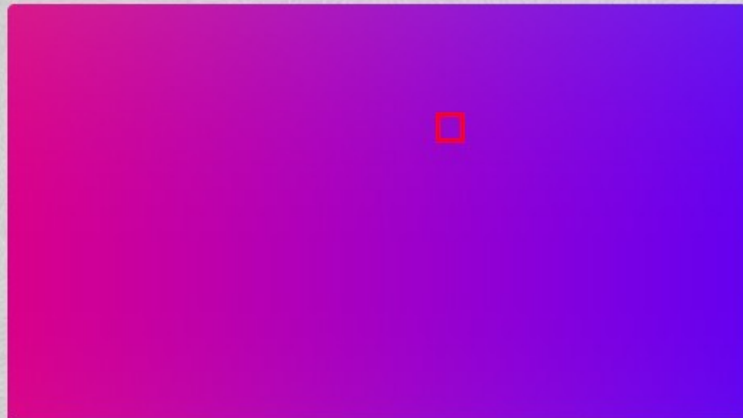
Describe your shader

private

Submit

```
+ Image
Shader Inputs
1 void mainImage( out vec4 fragColor, in vec2 fragCoord )
2 {
3     // Normalized pixel coordinates (from 0 to 1)
4     vec2 uv = fragCoord/iResolution.xy;
5     // Time varying pixel color
6     vec3 col = 0.5 + 0.5*cos(iTime+uv.yyx+vec3(0,2,4));
7     // Output to screen
8     fragColor = vec4(col,1.0);
9 }
10
11
Compiled in 0.0 secs 158 chars X? ?
```

Shadertoy: fragment shader



⏮ ⏭ 7.07 34.2 fps 458 x 257 REC 🔊 🖥

Name of your shader

Tags, comma separated. For example: fractal, noise

Describe your shader

private

Submit

+ Image

Shader Inputs

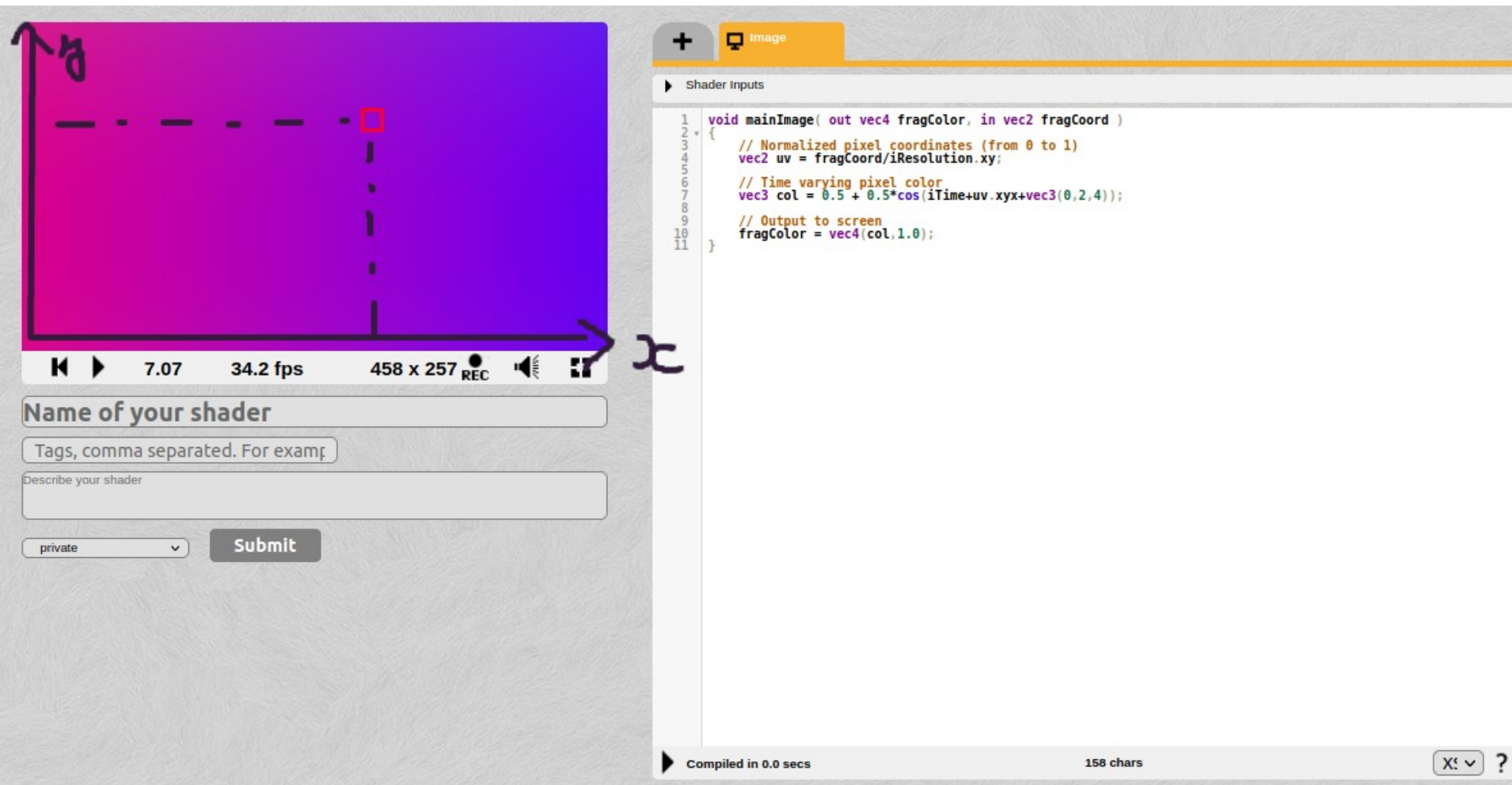
```
1 void mainImage( out vec4 fragColor, in vec2 fragCoord )
2 {
3   // Normalized pixel coordinates (from 0 to 1)
4   vec2 uv = fragCoord/iResolution.xy;
5
6   // Time varying pixel color
7   vec3 col = 0.5 + 0.5*cos(iTime+uv.yyx+vec3(0,2,4));
8
9   // Output to screen
10  fragColor = vec4(col,1.0);
11 }
```

Compiled in 0.0 secs

158 chars

X! v ?

Shadertoy: fragment shader

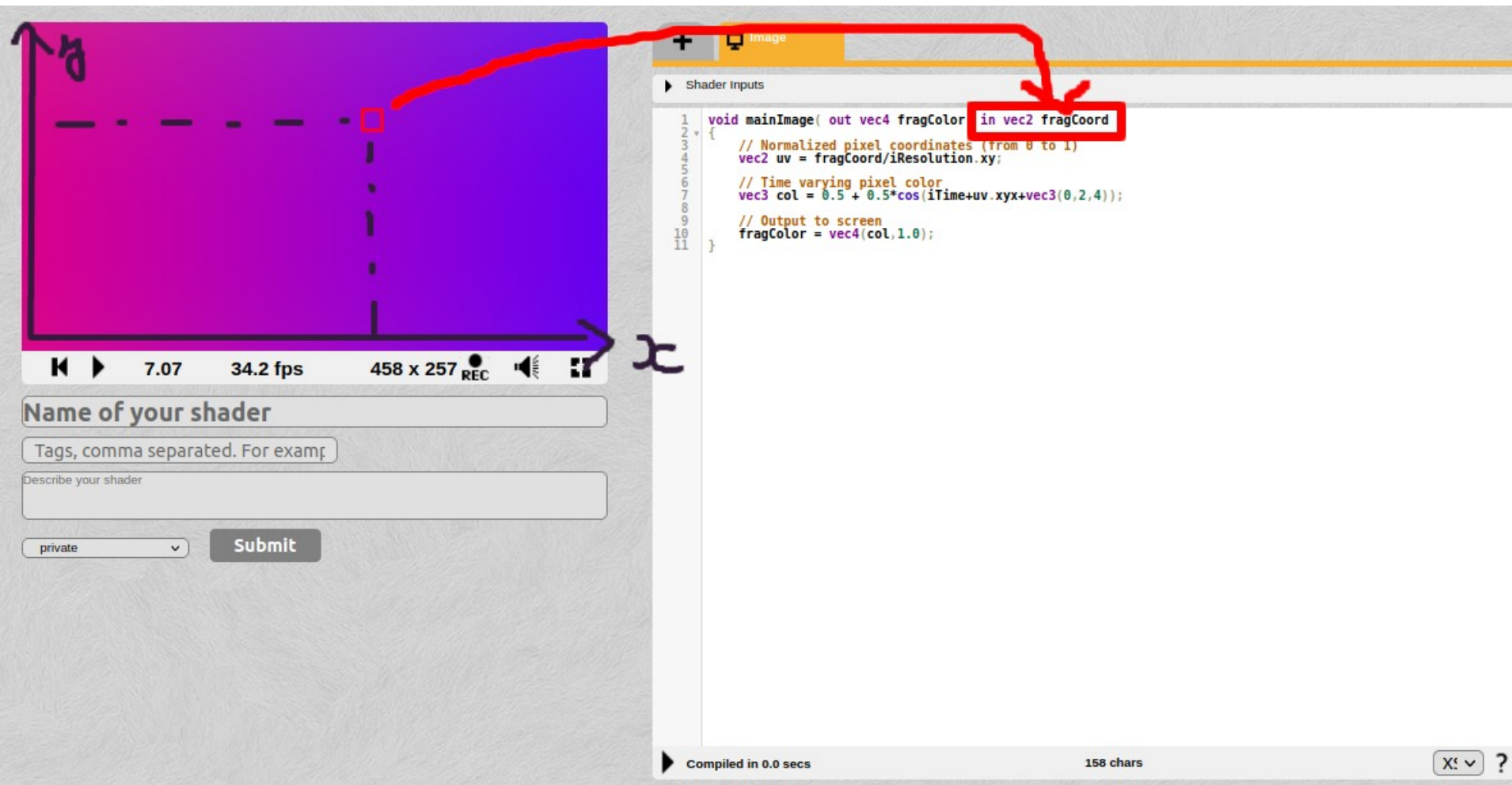


The screenshot displays the Shadertoy web application interface. On the left, a preview window shows a 2D color gradient from magenta on the left to blue on the right. A red square highlights a specific pixel on the right side of the gradient. Hand-drawn black arrows point from the red square to the 'x' and 'y' axes of a coordinate system drawn on the left side of the preview. Below the preview, there is a form for submitting the shader, including fields for 'Name of your shader', 'Tags, comma separated. For example', and 'Describe your shader'. A 'Submit' button is located at the bottom right of the form. To the right of the preview, the 'Shader Inputs' tab is active, showing the following GLSL code:

```
1 void mainImage( out vec4 fragColor, in vec2 fragCoord )
2 {
3     // Normalized pixel coordinates (from 0 to 1)
4     vec2 uv = fragCoord/iResolution.xy;
5
6     // Time varying pixel color
7     vec3 col = 0.5 + 0.5*cos(iTime+uv.yyx+vec3(0,2,4));
8
9     // Output to screen
10    fragColor = vec4(col,1.0);
11 }
```

At the bottom of the interface, a status bar indicates 'Compiled in 0.0 secs' and '158 chars'. There are also icons for closing the window and a help icon.

Shadertoy: fragment shader

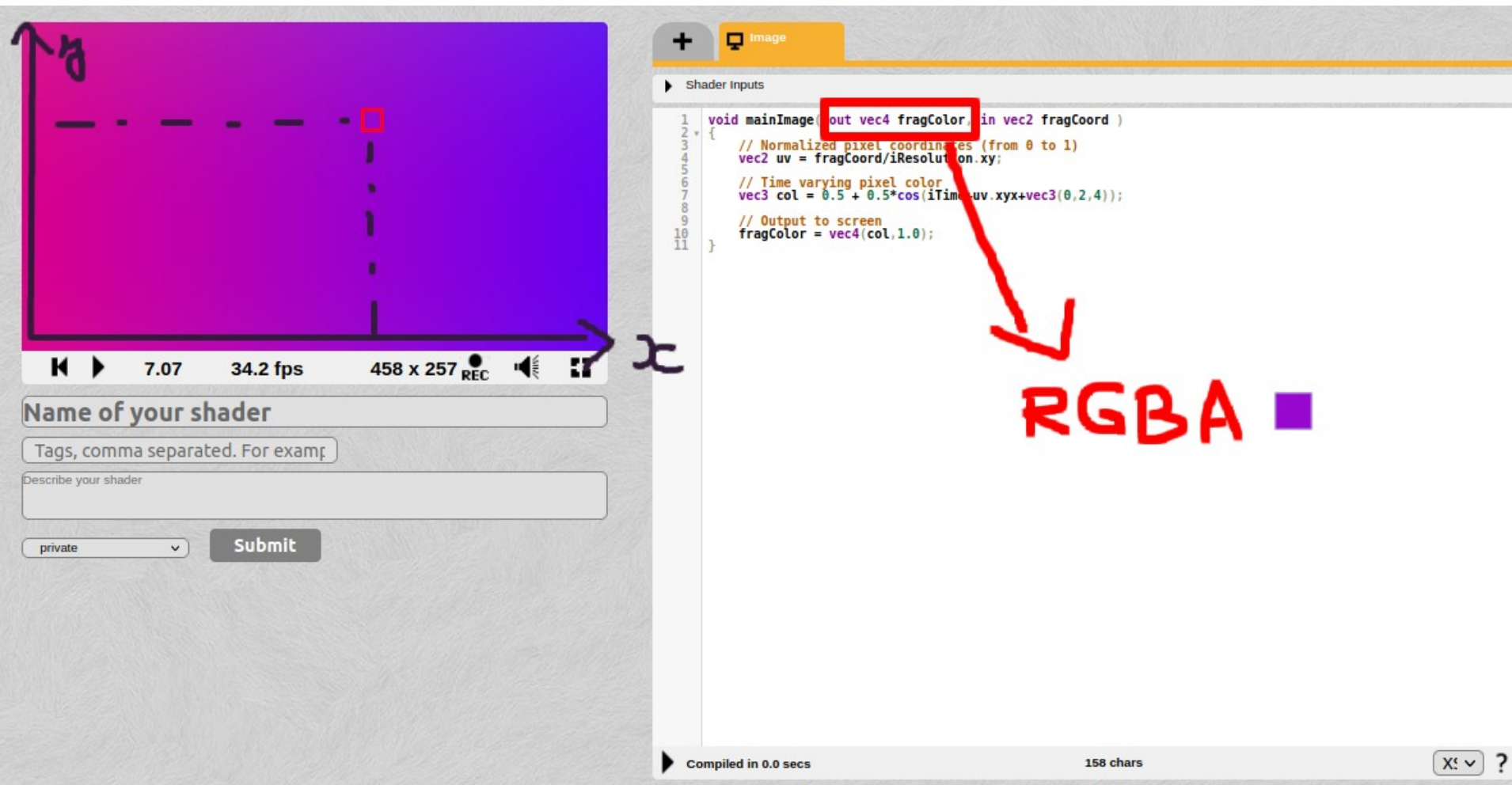


The screenshot displays the Shadertoy web application. On the left, a 2D plot shows a color gradient from purple to blue, with a red square highlighting a specific pixel. A red arrow points from this square to the 'in vec2 fragCoord' parameter in the shader code on the right. The plot also features a dashed horizontal line and a vertical line, with a handwritten 'y' and 'x' indicating the coordinate system. Below the plot, there are input fields for 'Name of your shader', 'Tags, comma separated. For example', and 'Describe your shader', along with a 'Submit' button and a privacy dropdown set to 'private'. The shader code on the right is as follows:

```
1 void mainImage( out vec4 fragColor, in vec2 fragCoord
2 {
3     // Normalized pixel coordinates (from 0 to 1)
4     vec2 uv = fragCoord/iResolution.xy;
5
6     // Time varying pixel color
7     vec3 col = 0.5 + 0.5*cos(iTime+uv.yyx+vec3(0,2,4));
8
9     // Output to screen
10    fragColor = vec4(col,1.0);
11 }
```

The bottom status bar indicates 'Compiled in 0.0 secs' and '158 chars'.

Shadertoy: fragment shader



The screenshot displays the Shadertoy web application. On the left, a 2D plot shows a color gradient from purple to blue, with a red square highlighting a specific pixel. Below the plot, there are input fields for the shader's name, tags, and description, along with a 'Submit' button. On the right, the 'Shader Inputs' panel shows the following GLSL code:

```
1 void mainImage(out vec4 fragColor, in vec2 fragCoord)
2 {
3     // Normalized pixel coordinates (from 0 to 1)
4     vec2 uv = fragCoord/iResolution.xy;
5
6     // Time varying pixel color
7     vec3 col = 0.5 + 0.5*cos(iTime-uv.yyx+vec3(0,2,4));
8
9     // Output to screen
10    fragColor = vec4(col,1.0);
11 }
```

A red arrow points from the `fragColor` variable in the code to the text **RGBA** and a small purple square, indicating that the output is a 4-component color vector (Red, Green, Blue, Alpha).

Shadertoy: fragment shader



<https://www.youtube.com/watch?v=8--5LwHRhjk>

<https://www.shadertoy.com/view/WsSBzh>