

# Converting ShaderToy to GLSL

ShaderToy is a platform for creating and sharing shaders in GLSL, but it includes some unique functions and features that are specific to its environment. If you want to port ShaderToy shaders to plain GLSL or another environment, you'll need to understand these specific functions and how to replace or adapt them.

Here are some common ShaderToy-specific functions and their equivalents or alternatives in standard GLSL:

1. **iGlobalTime**: This variable represents the time elapsed since the shader started running.

- **In GLSL**: You'll need to pass the elapsed time as a uniform variable. For example:

glsl

Copy code

```
uniform float uTime; // Use this uniform in your shader
```

2. **iResolution**: This variable holds the resolution of the output image.

- **In GLSL**: You can pass the resolution as a uniform variable:

glsl

Copy code

```
uniform vec2 uResolution; // Use this uniform in your shader
```

3. **iMouse**: This variable contains the position and state of the mouse.

- **In GLSL**: You'll need to pass the mouse position as a uniform variable:

glsl

Copy code

```
uniform vec4 uMouse; // x, y, z, and w can represent mouse position and state
```

4. **iChannel0**, **iChannel1**, **etc.**: These variables represent textures that you can sample from.

- **In GLSL**: You'll need to pass these textures as uniform sampler2D variables:

glsl

Copy code

```
uniform sampler2D uTexture0; // Use this uniform in your shader
```

5. **mainImage(out vec4 fragColor, in vec2 fragCoord)**: This is the main function in ShaderToy shaders, where **fragColor** is the output color and **fragCoord** is the coordinate of the current fragment.

- **In GLSL**: In a typical GLSL setup, you might use a similar fragment shader function, but the signature and setup might vary depending on your framework. For example, in a typical OpenGL setup, the fragment shader might look like this:

glsl

Copy code

```
void main() {  
    // Your code here  
}
```

6. **fragCoord.xy**: This variable provides the coordinates of the current fragment.
- **In GLSL**: You can pass these coordinates as a varying variable or compute them based on the screen dimensions:

glsl

Copy code

```
vec2 fragCoord = gl_FragCoord.xy; // In a GLSL shader
```

When porting shaders, make sure to adapt any dependencies on the ShaderToy environment to the specifics of your target environment. This usually involves setting up appropriate uniform variables, handling texture inputs and outputs, and ensuring that your fragment shader logic aligns with the rendering pipeline you're using.

---

Revision #3

Created 13 September 2024 23:42:42 by victor

Updated 13 September 2024 23:45:57 by victor