A Very Quick Introduction of OpenGL Shading Language (GLSL)

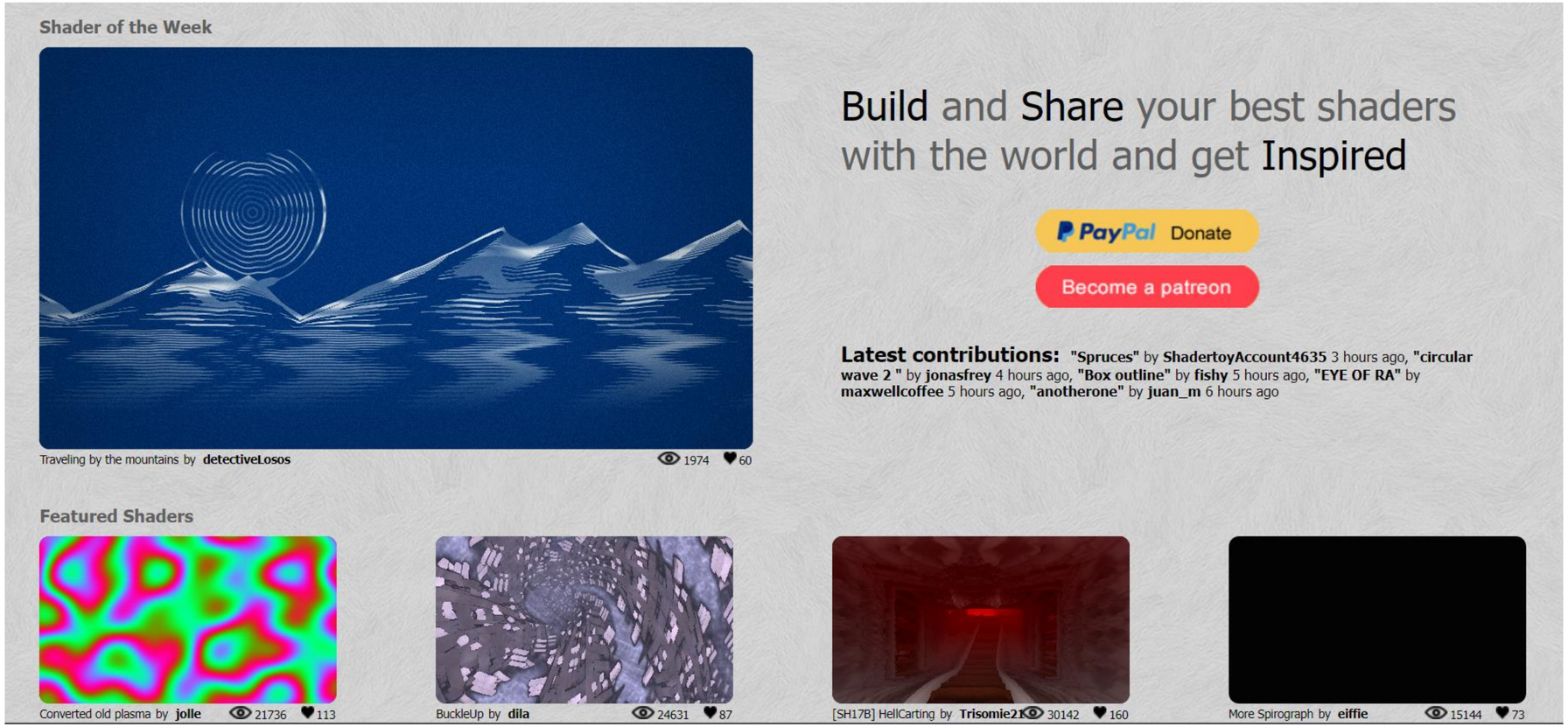
Bo Zhu

School of Interactive Computing

Georgia Institute of Technology

Many slides from Ed Angel and Dave Shreiner for SIGGRAPH 2013 course notes

ShaderToy



https://www.shadertoy.com/

GLSL Data Types

- Scalar types: `float, int, bool`
- Vector types: `vec2, vec3, vec4`
  `ivec2, ivec3, ivec4`
  `bvec2, bvec3, bvec4`
- Matrix types: `mat2, mat3, mat4`
- Texture sampling: `sampler1D, sampler2D, sampler3D, samplerCube`
- C++ Style Constructors

```
vec3 a = vec3(1.0, 2.0, 3.0);
```

Operators

- Standard C/C++ arithmetic and logic operators
  - and: `&&`
  - or: `||`
  - not: `!`
  - equality: `==`
  - inequality: `!=`
  - conditional operator: `? :`

  (Read the GLSL reference for more)

- Overloaded operators for matrix and vector operations

```
mat4 m;
vec4 a, b, c;

b = a*m;
c = m*a;
```

Components and Swizzling

- Access vector components using either:
  - [ ] (c-style array indexing, with index number starting from 0)
  - `xyzw`, `rgba` or `stpq` (named components)

- For example:

```
vec3 v;
v[1], v.y, v.g, v.t
```

 - all refer to the same element

- Component swizzling:

```
vec3 a, b;
a.xy = b.yx;
```

Qualifiers

- `in`, `out`
  - Copy vertex attributes and other variable into and out of shaders

```
in  vec2 texCoord;
out vec4 color;
```

- `uniform`
  - shader-constant variable from application

```
uniform float time;
uniform vec4 rotation;
```

Functions

- Built in
  - Arithmetic: `sqrt`, `pow`, `abs`
  - Trigonometric: `sin`, `asin`
    - `Take angle as put`
  - Graphical: `length`, `reflect`
- (Read the GLSL reference for more)

Built-in Variables

- `gl_Position`
  - (required) output position from vertex shader
- `gl_FragCoord`
  - input fragment position
- `gl_FragDepth`
  - input depth value in fragment shader

A Simple Vertex Shader

```
#version 430

in vec4 vPosition;
in vec4 vColor;

out vec4 color;

void main()
{
    color = vColor;
    gl_Position = vPosition;
}
```

The Simplest Fragment Shader

```
#version 430

in vec4 color;

out vec4 fColor; // fragment's final color

void main()
{
   fColor = color;
}
```

Starter Code

- Live Demo
- Practice: Let's start!