GLSL

OpenGL Shading Language

https://git.io/Jey3U

By Katherine, Sarah, Brandon, and Alex

What is GLSL?

- OpenGL Shading Language
- High level shading language
  - Graphics programming language
  - Allows for shading of graphics

History of GLSL

- Created by the OpenGL Architecture Review Board
- Introduced in 2004
  - First as extension of OpenGL, then formally included
  - Included in OpenGL 2.0 core
  - The first major revision
- Why was it created?
  - Direct control of graphics pipeline without ARB assembly language
- Good compatibility with web browsers/operating systems

Syntax

Getting Started with GLSL

- thebookofshaders.com
- Desktop applications: GLAD library
  - C++: #include <glad/glad.h>
  - Python: pip install glad
- Linux: glslviewer
  - sudo apt-get install git-core glslviewer
- Browser: Canvas with WebGL

GLSL Hello World

```
// an attribute will receive data from a buffer
attribute vec4 a_position;

// all shaders have a main function
void main() {

  // gl_Position is a special variable a vertex shader
  // is responsible for setting
  gl_Position = a_position;
}
```

```
out vec4 FragColor;

void main()
{
    FragColor = vec4(1.0f, 0.5f, 0.2f, 1.0f);
}
```

Syntax

- C-Style Language
  - For, if, switch, etc. all work the same way
  - But they’re not very efficient!!
  - Contains many of the same primitive variable types
- Recursion is NOT allowed
- `in` and `out` keywords
  - `void function(in float input, out float output, inout float inout)`
- No strings

Variable Types

- Basic
  - void
  - bool
  - int/uint
  - float
  - double
- Vectors
  - vec2, vec3, vec4
  - dvec, ivec, bvec, uvec
- Matrices
  - mat2, mat3, mat4
  - dmat2, dmat3, dmat4
  - mat2x3, mat2x4, mat3x2...

Uniforms

- GLSL needs data from the application passed to it
- Uniform and Attribute
  - Any information passed to the shader from the application
  - Can be buffers, textures, or any variable type
  - Uniform is a constant
    - e.g. a texture
  - Attribute is per vertex
    - e.g. position data

Examples

```
precision highp float;

void main()
{
  gl_FragColor = vec4(1,1,0, 1.0);
}
```

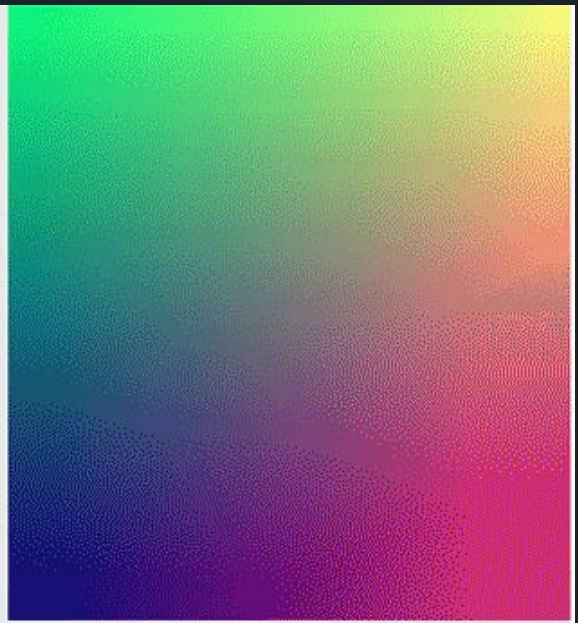
```
precision highp float;

void main()
{
  gl_FragColor = vec4(0.9,0,0.6, 1.0);
}
```

```
precision highp float;

uniform vec2 resolution;

void main()
{
  vec3 color;

  // Convert from pixels to coordinates
  float ndcx = (gl_FragCoord.y / resolution.x) - 1.0;

  if (ndcx > 0.33) {
    color = vec3(1,0,0);
  } else if (ndcx > -0.33) {
    color = vec3(0,1,0);
  } else {
    color = vec3(0,0,1);
  }
  gl_FragColor = vec4(color, 1.0);
}
```

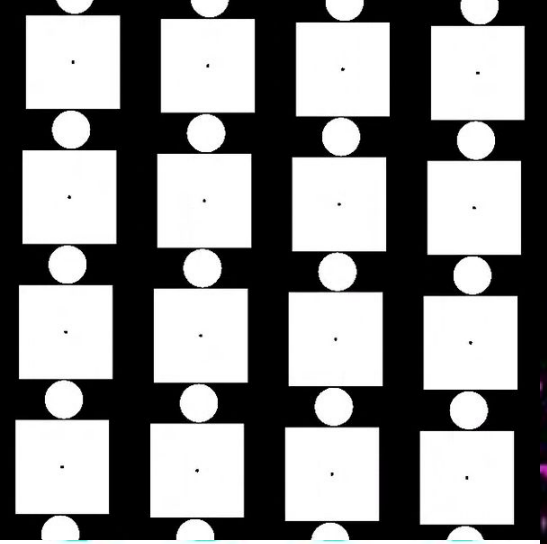
Shdr

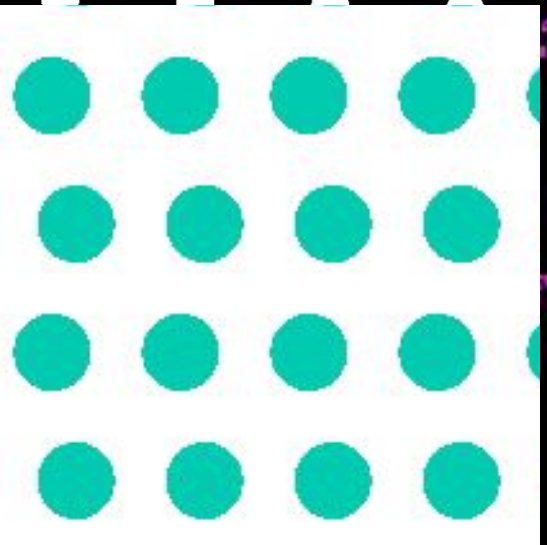
```
precision highp float;

uniform vec2 resolution;

void main()
{
  vec3 color;

  // Convert from pixels to coordinates
  float ndcx = (gl_FragCoord.x / resolution.x) - 1.0;

  if (ndcx > 0.33) {
    color = vec3(1,0,0);
  } else if (ndcx > -0.33) {
    color = vec3(0,1,0);
  } else {
    color = vec3(0,0,1);
  }
  gl_FragColor = vec4(color, 1.0);
}
```

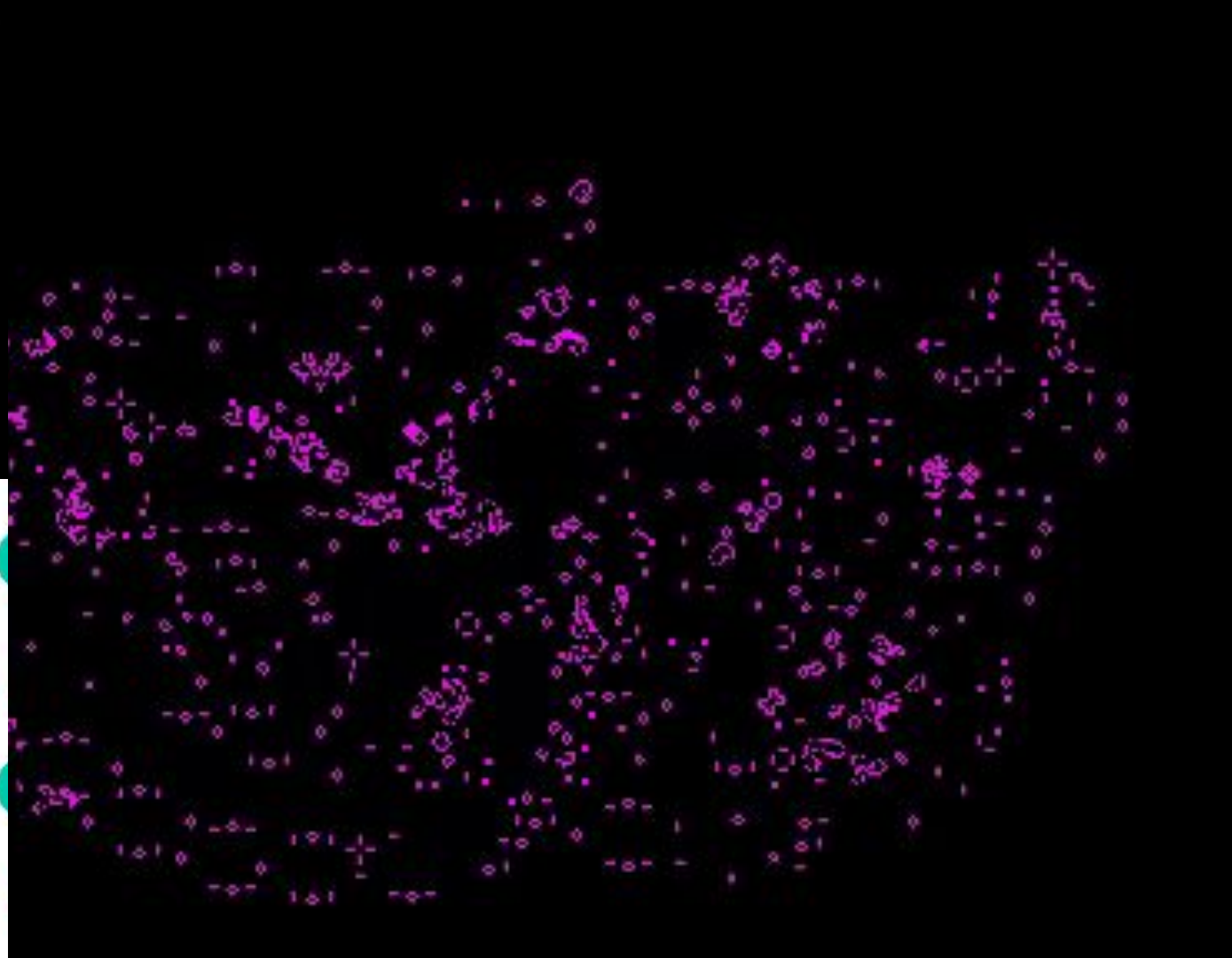
```
#ifdef GL_ES
precision mediump float;
#endif

uniform vec2 u_resolution;
uniform vec2 u_mouse;
uniform float u_time;

void main()
{
    vec2 st = gl_FragCoord.xy/u_resolution.xy;
    st.x *= u_resolution.x/u_resolution.y;

    vec3 color = vec3(0.);
    color = vec3(st.x,st.y,abs(sin(u_time)));

    gl_FragColor = vec4(color,1.0);
}

```

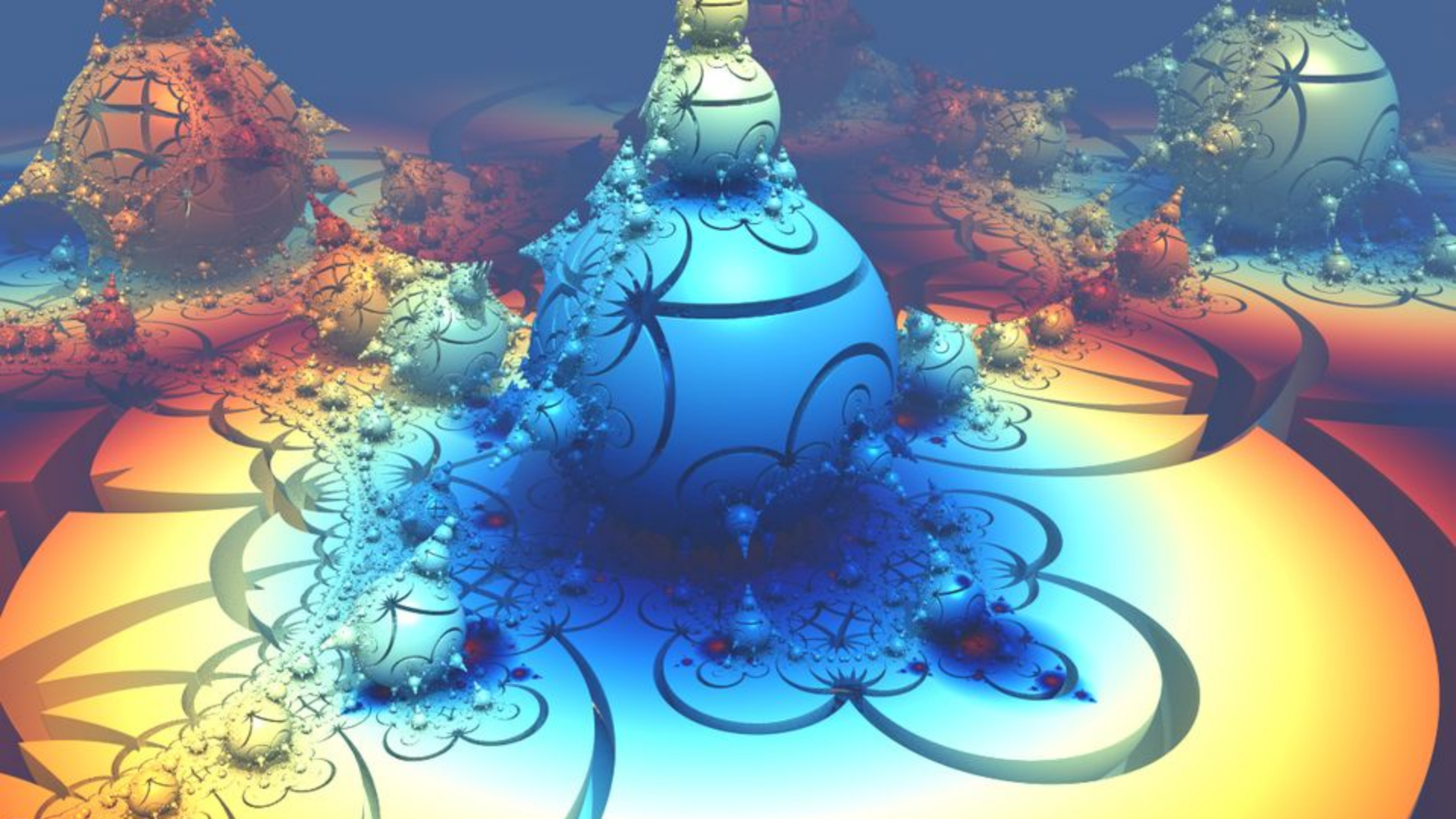


More Advanced Examples









When and why you should use GLSL

General Reasons

- Useful for graphics
  - Domain Specific Language
  - Shouldn’t be compared to Java, Python, and other general languages
- More efficient computation
  - Runs in parallel per pixel
- Gain experience in shader programming
  - Useful for those who want to be game programmers, graphics developers, technical artists, etc.

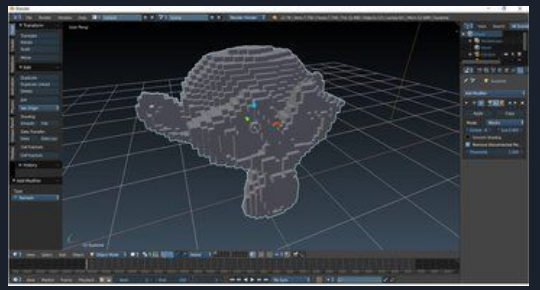
The Competition

- Has two main rivals:
  - SPIR-V
    - Allows many shading languages to compile to it
    - Individual shaders will either work on Vulkan or OpenGL
    - Requires more specification
  - C for Graphics (Cg)
    - Allows for precompiling of shaders
    - No longer being supported
- GLSL
  - The standard of OpenGL
  - Potential driver bugs, especially on ATI compilers

Where is it used?

Google Earth

Adobe Photoshop



https://www.youtube.com/watch?v=aS5UU498Bf8

MINECRAFT

FORTNITE

Interactive Demo!!!!

History:

- https://www.khronos.org/opengl/wiki/History_of_OpenGL
- https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.1.20.pdf
- https://en.wikipedia.org/wiki/OpenGL_Shading_Language

Syntax:

- https://www.khronos.org/registry/OpenGL/specs/gl/GLSLangSpec.4.50.pdf
- http://graphics.cs.wisc.edu/WP/cs559-sp2016/2016/03/08/glsl-shader-examples/
- https://learnopengl.com/Getting-started/Shaders

Whatever Brandon's Thing Is

- Nothing? Really? That's pretty sad, Brandon...

Why you'd use GLSL:

- https://www.khronos.org/opengl/wiki/Selecting_a_Shading_Language#ARB_assembly
- http://nehe.gamedev.net/article/glsl_an_introduction/25007/
- https://en.wikibooks.org/wiki/GLSL_Programming/Introduction

References