COS 426 – Precept 6

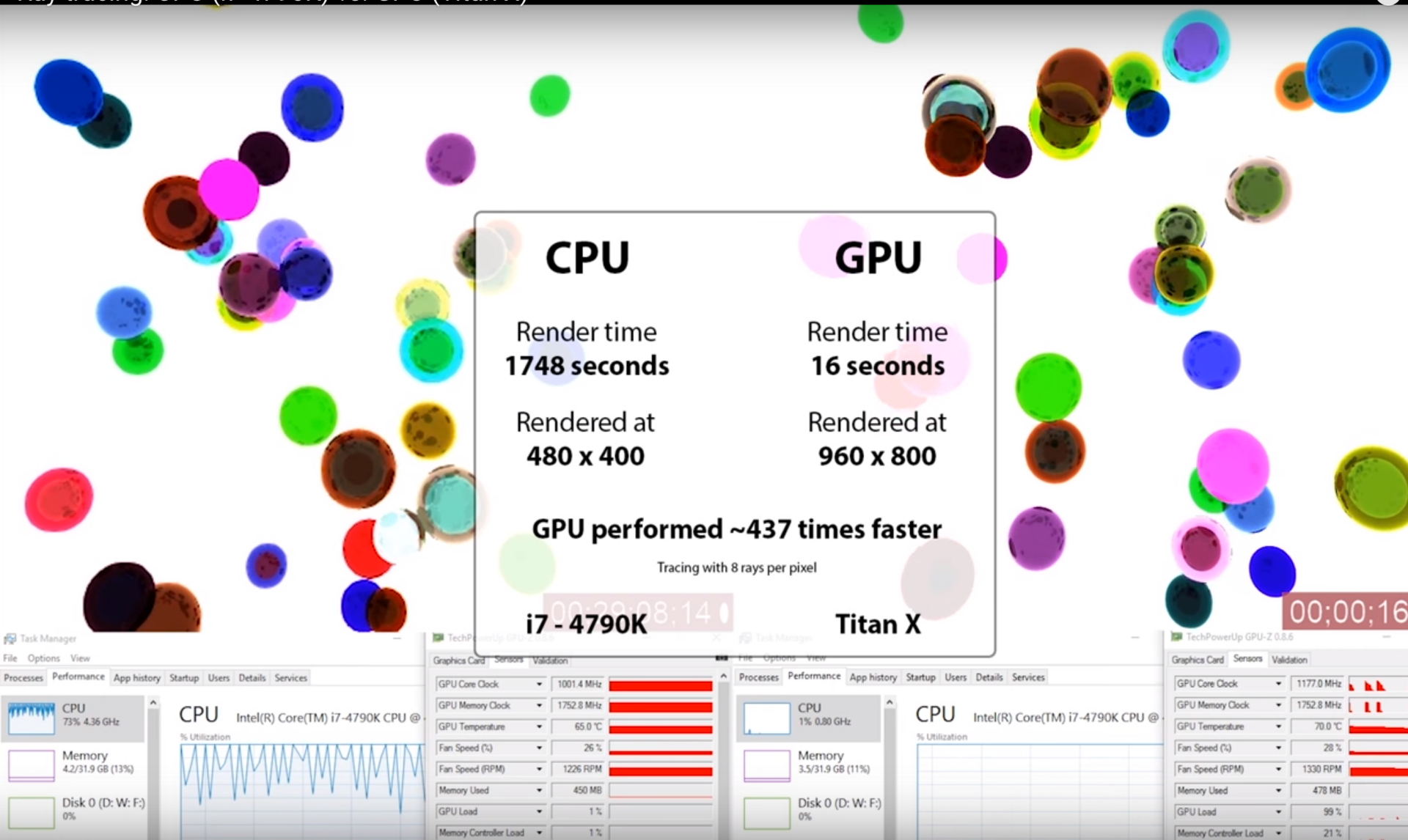
Raytracer: GLSL

Riley Simmons-Edler

GLSL?

open **G**raphics **L**ibrary **S**hading **L**anguage

Ray tracing: CPU (i7-4790K) vs. GPU (Titan X)



GLSL

+ Similar grammar as C
+ provide useful functions
  - min, max, sqrt
  - normalize, reflect, refract

For more information:
https://www.opengl.org/wiki/Core_Language_%28GLSL%29

Highly recommend you look at this!

GLSL shaders:

- Vertex shaders

```
<script id="2d-vertex-shader" type="x-shader/x-vertex”>

attribute vec2 a_position;
  void main() {
    gl_Position = vec4(a_position, 0, 1);
  }

</script>
```

GLSL shaders:

- Fragment shaders:

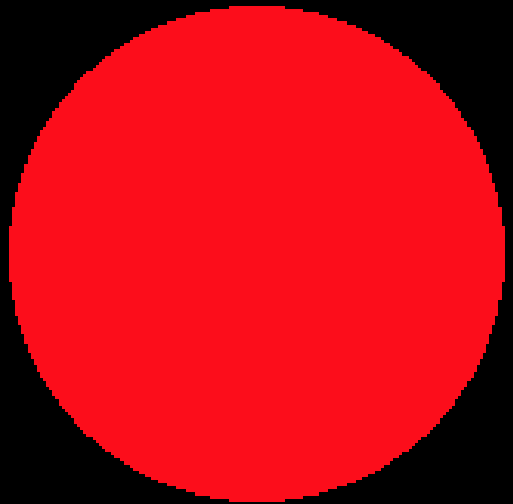
```
<script id="2d-fragment-shader" type="x-shader/x-
fragment”>

void main() {
    gl_FragColor = vec4(gl_FragCoord.x / canvas_width,
gl_FragCoord.y / canvas_height, 0, 1);
  }

</script>
```

```
<script id="2d-fragment-shader"
type="x-shader/x-fragment”>

  #ifdef
GL_FRAGMENT_PRECISION_HIGH
    precision highp float;
  #else
    precision mediump float;
  #endif
```



Uniform Versus Non-Uniform Variables

```
in vec3 fromPrevious;
in uvec2 fromRange;

const int foo = 5;
const uvec2 range = uvec2(2, 5);
uniform vec2 pairs;

uniform sampler2d tex;

void main()
{
  foo; //constant expressions are dynamically uniform.

  uint value = 21; //'value' is dynamically uniform.
  value = range.x; //still dynamically uniform.
  value = range.y + fromRange.y; //not dynamically uniform; current contents come from a non-dynamically uniform source.
  value = 4; //dynamically uniform again.
  if(fromPrevious.y < 3.14)
    value = 12;
  value; //NOT dynamically uniform. Current contents depend on 'fromPrevious', an input variable.

  float number = abs(pairs.x); //'number' is dynamically uniform.
  number = sin(pairs.y); //still dynamically uniform.
  number = cos(fromPrevious.x); //not dynamically uniform.

  vec4 colors = texture(tex, pairs.xy); //dynamically uniform, even though it comes from a texture.
                                        //It uses the same texture coordinate, thus getting the same texel every time.
  colors = texture(tex, fromPrevious.xy); //not dynamically uniform.
```

Uniform= Identical value for each fragment shader call (for each pixel)

“in” and “out” variables

```
void MyFunction(in float inputValue, out int outputValue, inout float inAndOutValue)
{
  inputValue = 0.0;
  outputValue = int(inAndOutValue + inputValue);
  inAndOutValue = 3.0;
}

void main()
{
  float in1 = 10.5;
  int out1 = 5;
  float out2 = 10.0;
  MyFunction(in1, out1, out2);
}
```

What are the values of in1, out1, and out2 after we return?

Tips

NO RECURSION

Recursive

```
#define MAX_RECURSION 10
function f(float x, int depth) {
    if( depth >= MAX_RECURSION) return 0;
   return 0.3 + 0.8 * f(x+1, depth+1)
}
function g() { return f(0,0) }
```

Avoid Recursion

```
#define MAX_RECURSION 10
function g() {
   float x = 0.0, weight = 1.0, res = 0.0;
   for (int i=0;i < MAX_RECURSION;i++ ) {
        res = res + weight * 0.3;
        weight = weight * 0.8;
        x = x + 1.0;                    //Not x = x + 1
   }
   return res;
```

Questions?