COS 426 – Precept 6

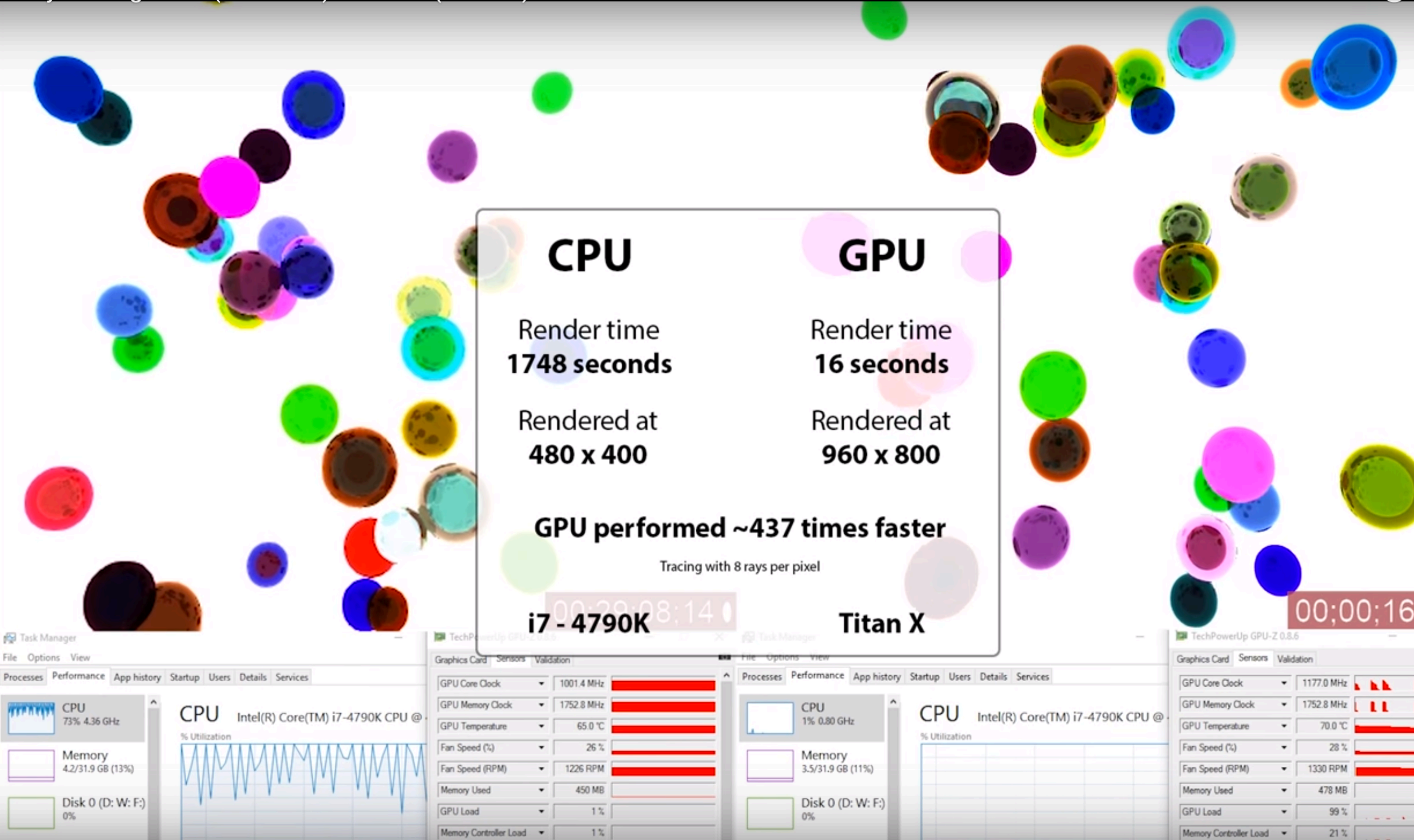
Raytracer: GLSL

Huiwen Chang

GLSL?

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

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



CPU	GPU
Render time **1748 seconds**	Render time **16 seconds**
Rendered at **480 x 400**	Rendered at **960 x 800**

GPU performed ~437 times faster

Tracing with 8 rays per pixel

i7 - 4790K	**Titan X**

00;29;08;14

00;00;16

Task Manager

File Options View

Processes Performance App history Startup Users Details Services

CPU 73% 4.36 GHz

Memory 4.2/31.9 GB (13%)

Disk 0 (D: W: F:) 0%

CPU Intel(R) Core(TM) i7-4790K CPU @

% Utilization

TechPowerUp GPU-Z 0.8.6

Graphics Card Sensors Validation

GPU Core Clock	1001.4 MHz
GPU Memory Clock	1752.8 MHz
GPU Temperature	65.0 °C
Fan Speed (%)	26 %
Fan Speed (RPM)	1226 RPM
Memory Used	450 MB
GPU Load	1 %
Memory Controller Load	1 %

Task Manager

File Options View

Processes Performance App history Startup Users Details Services

CPU 1% 0.80 GHz

Memory 3.5/31.9 GB (11%)

Disk 0 (D: W: F:) 0%

CPU Intel(R) Core(TM) i7-4790K CPU @

% Utilization

TechPowerUp GPU-Z 0.8.6

Graphics Card Sensors Validation

GPU Core Clock	1177.0 MHz
GPU Memory Clock	1752.8 MHz
GPU Temperature	70.0 °C
Fan Speed (%)	28 %
Fan Speed (RPM)	1330 RPM
Memory Used	478 MB
GPU Load	99 %
Memory Controller Load	21 %

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

GLSL shaders:

- Vertex shaders

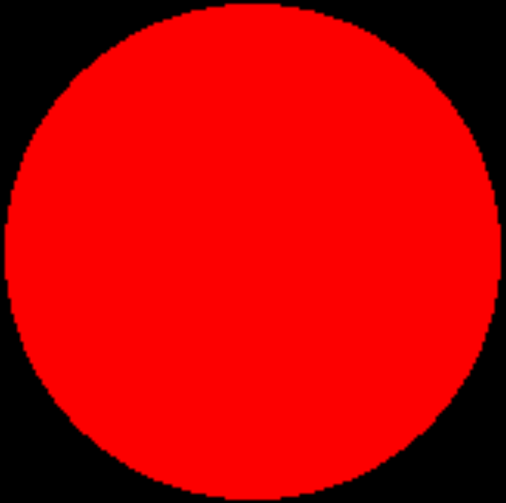
```
<script id="2d-vertex-shader" type="x-shader/x-vertex”>

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


  void main() {
    float normalizedX = gl_FragCoord.x - width/2.0;
    float normalizedY = gl_FragCoord.y - height/2.0;

    if (sqrt(normalizedX*normalizedX + normalizedY*normalizedY) < 0.0){
        gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);
    } else {
        gl_FragColor = vec4(0.0, 0.0, 0.0, 1.0);
    }
  }
```



Loop

Using loop in this way:

```
#define MAX_OBJECTS 5
uniform int numObjects;
for (int i=0; i<MAX_OBJECTS; i++) {
            if ( i>= numObjects ) break;
}
```

Tips

1. Array index - using constant or loop variable

```
int u = 5;
for (int i=0; i<MAX_OBJECTS; i++) {
    object[i]   OK!
     object[3]   OK!
    object[u]   NO!
    if( u == 5 )
        object[u]
```

Tips

1. Array index - using constant or loop variable

```
int u = 5;
for (int i=0; i<MAX_OBJECTS; i++) {
    object[i]   OK!
     object[3]   OK!
    object[u]   NO!
    if( u == 5 )
        object[u]    NO!
```

Tips

function parameter

```
in(copy in), out(copy out)
-void sqr( float x, out float res ) { res = x*x; }
-float sqr (float x) { return x*x;}
```

Tips

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;
```

Tips

EPS

```
if (a!==0)
```

```
if( a < -EPS || a > EPS )
```