OpenCL Ray Tracer

Ray Tracing on the GPU

Xinyu Cheng, Jake Crouch, Spencer Murphy, and Dylan Sturgeon

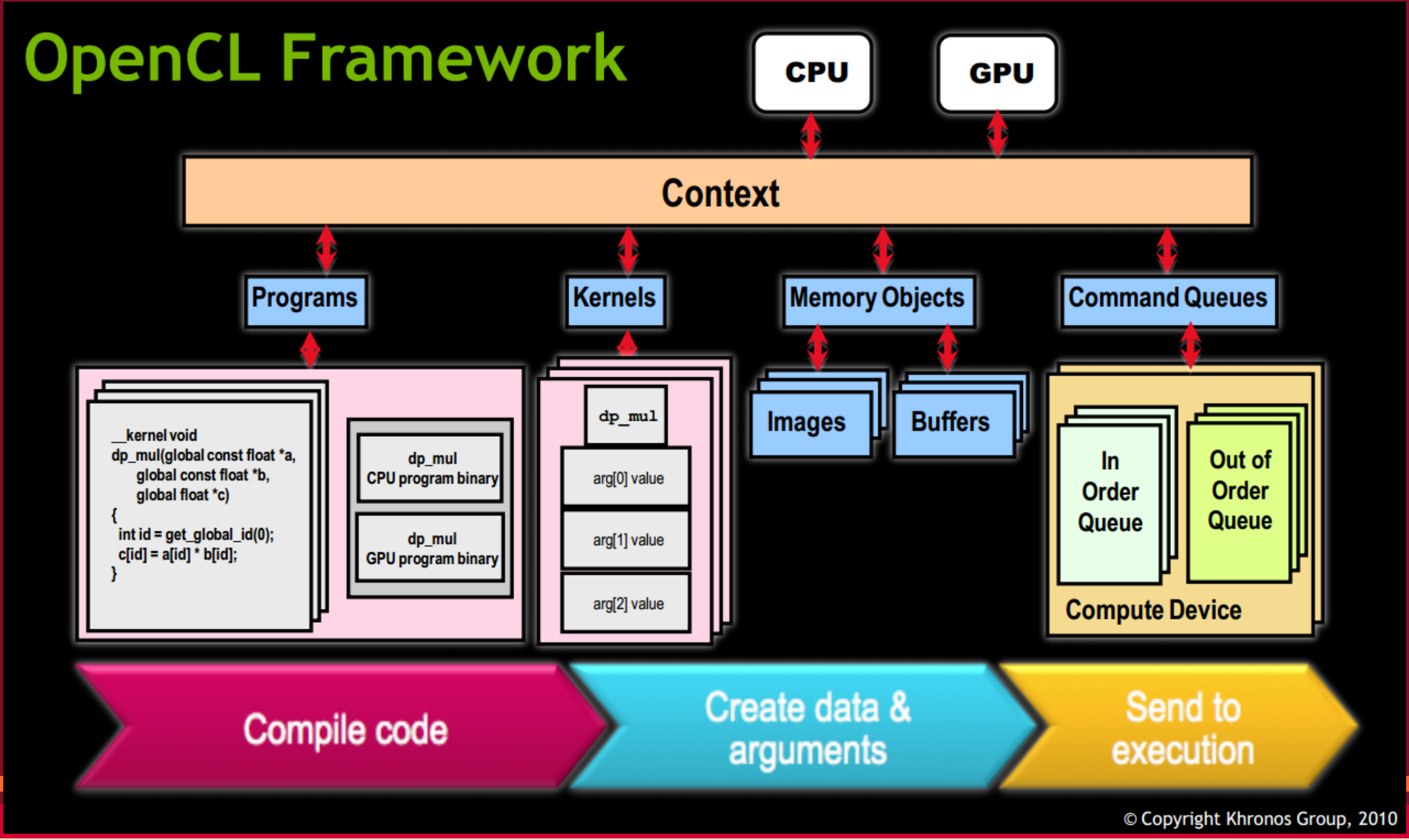
Roadmap

- OpenCL Research
- GPU Ray Tracing
- Results
- Future Work

OpenCL - OpenWhat?

- Open Computing Language
  - Standard for Parallel Programming
  - Maintained by Khronos Group
  - Open and royalty free
- Implemented by most modern hardware
  - All Intel/AMD CPUs
  - All NVidia/AMD GPUs

OpenCL Framework



CPU

GPU

Context

Programs

Kernels

Memory Objects

Command Queues

```
__kernel void
dp_mul(global const float *a,
      global const float *b,
      global float *c)
{
  int id = get_global_id(0);
  c[id] = a[id] * b[id];
}
```

dp_mul
CPU program binary

dp_mul
GPU program binary

dp_mul

arg[0] value

arg[1] value

arg[2] value

Images

Buffers

In Order Queue

Out of Order Queue

Compute Device

Compile code

Create data & arguments

Send to execution

OpenCL - C Code

Derivative of ISO C99

Additional Features

- Vector Types (float3, float4, etc.)
- Address Space qualifiers
- Synchronization
- Built-in functions

OpenCL - Code Restrictions

Pointers to functions

Pointers as an argument

Recursion

Dynamic memory

OpenCL - Kernels

Build a kernel from file
Create memory objects
Pass arguments and execute kernel
Read back memory objects

OpenCL - Command Queues

Commands Enqueued

- Execute Asynchronously (optionally block)
- In-order or Out-of-order (in-order by default)

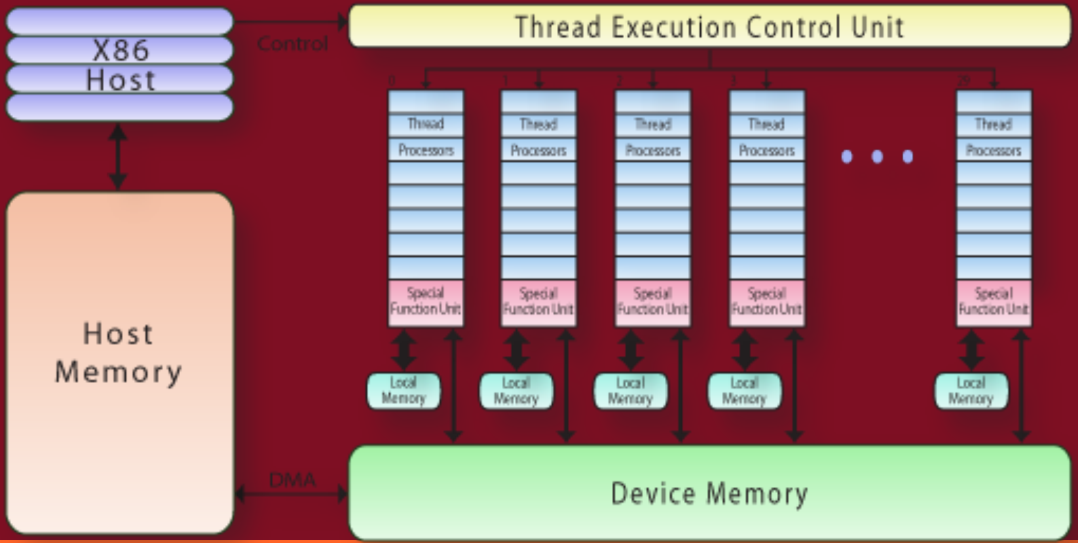
Command Queue

- Commands scheduled to execute on a device

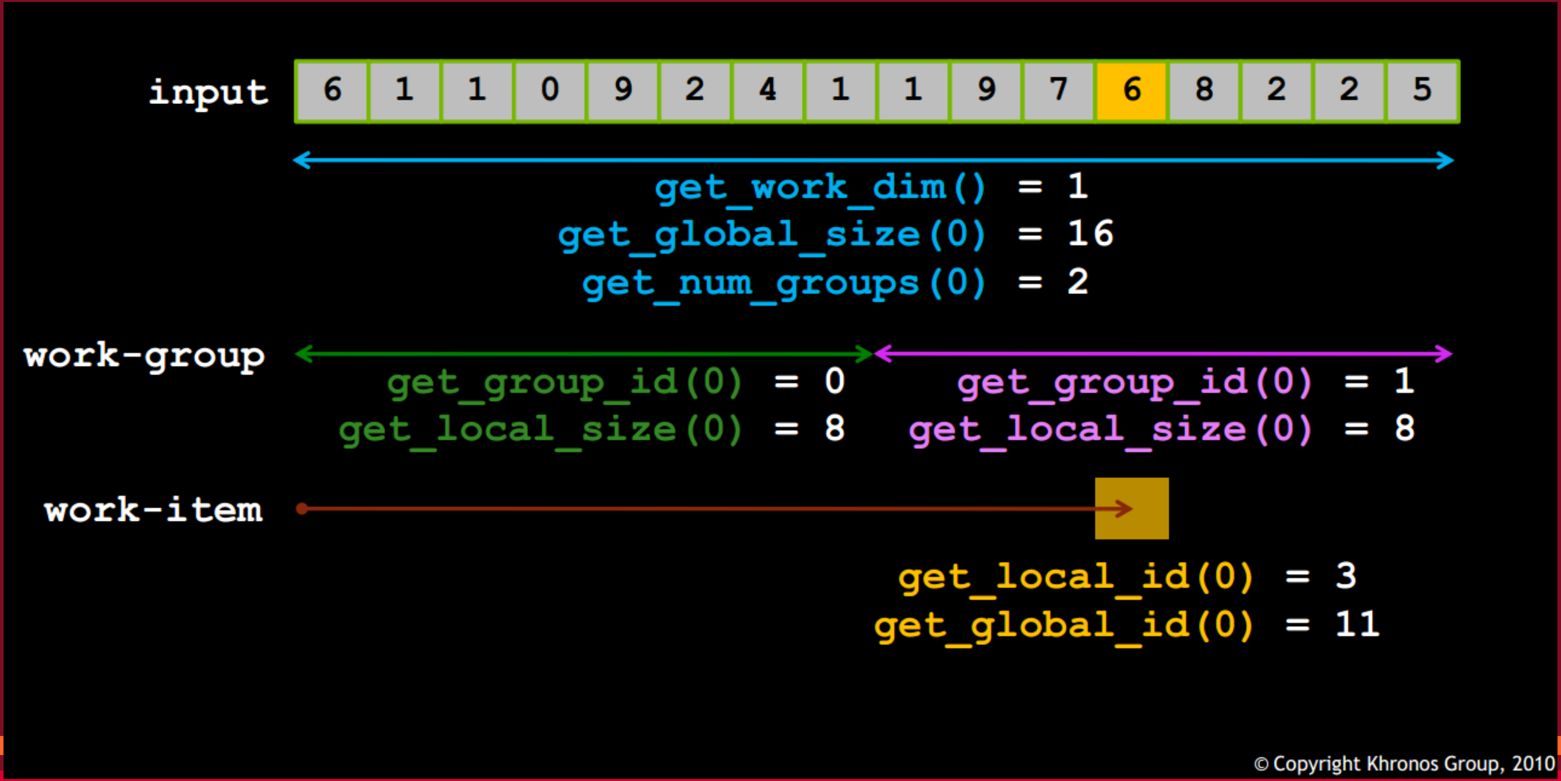
OpenCL - Kernel Execution

Work-items execute kernels in lock step

Work-group - collection of work-items



OpenCL - Work Group Example



input															
6	1	1	0	9	2	4	1	1	9	7	6	8	2	2	5

```
get_work_dim()    = 1
get_global_size(0) = 16
get_num_groups(0)  = 2
```

work-group

```
get_group_id(0)   = 0     get_group_id(0)   = 1
get_local_size(0) = 8     get_local_size(0) = 8
```

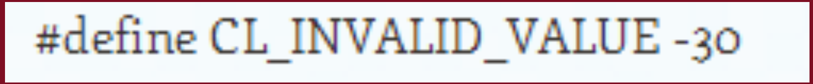
work-item

```
get_local_id(0)  = 3
get_global_id(0) = 11
```

OpenCL - Errors

- Any operation can (and probably will) fail!
  - Operations set or return an error code
  - Error codes map to *useful* descriptions

Lesson: *Always* check for errors.



```
#define CL_INVALID_VALUE -30
```

Thanks Khronos...

GPU Ray Tracing - Work in CPU & GPU

- Works in CPU

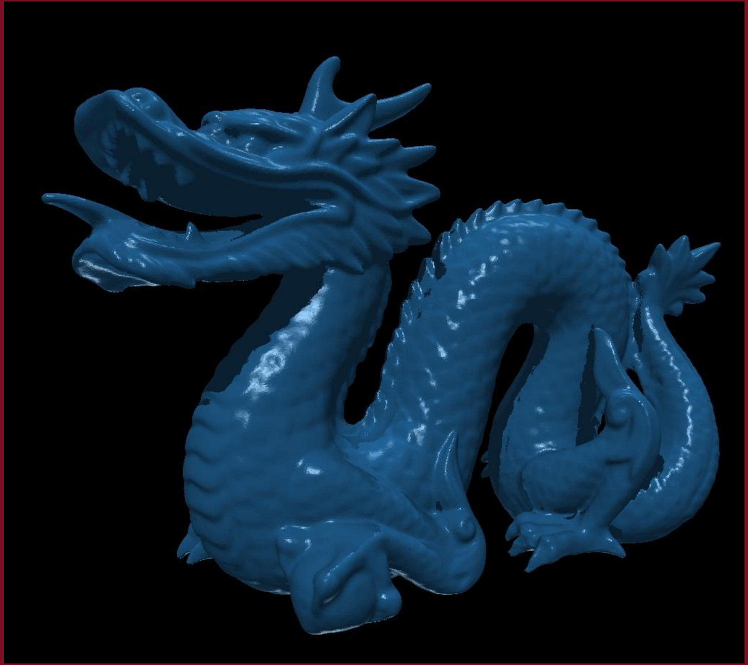
  Load objects and materials.

  Build camera, scene and tree.

- Works in GPU

  Run intersections.

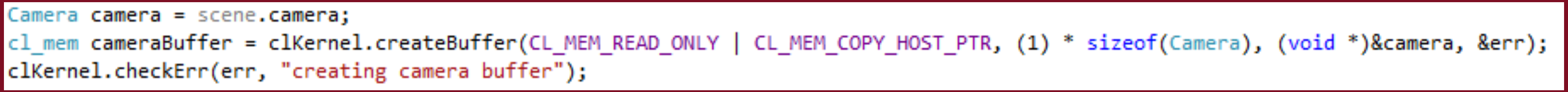
  Calculate color.



GPU Ray Tracing - Data Passing

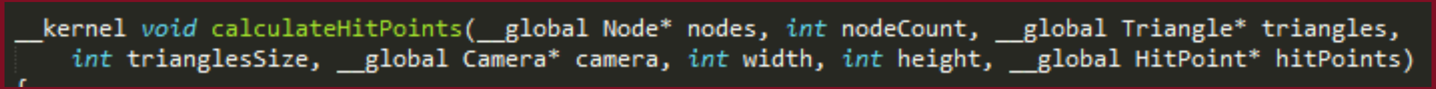
Basic Idea for Data Passing

In CPU, we create buffer for input and output of GPU functions. The buffer data can be used in the device functions.

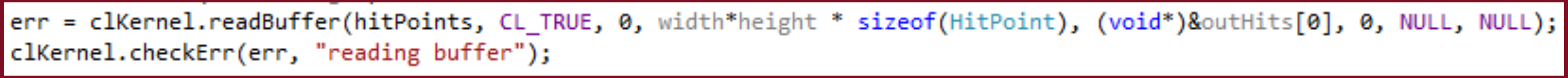


```
Camera camera = scene.camera;
cl_mem cameraBuffer = clKernel.createBuffer(CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR, (1) * sizeof(Camera), (void *)&camera, &err);
clKernel.checkErr(err, "creating camera buffer");
```

In GPU, we create the corresponding functions to receive input buffers. Then do the calculation and put results into output buffers.



```
__kernel void calculateHitPoints(__global Node* nodes, int nodeCount, __global Triangle* triangles,
    int trianglesSize, __global Camera* camera, int width, int height, __global HitPoint* hitPoints)
{
```



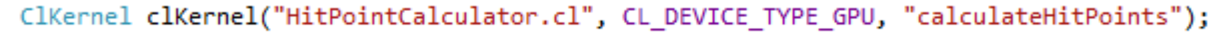
```
err = clKernel.readBuffer(hitPoints, CL_TRUE, 0, width*height * sizeof(HitPoint), (void*)&outHits[0], 0, NULL, NULL);
clKernel.checkErr(err, "reading buffer");
```

GPU Ray Tracing - Multi Kernel

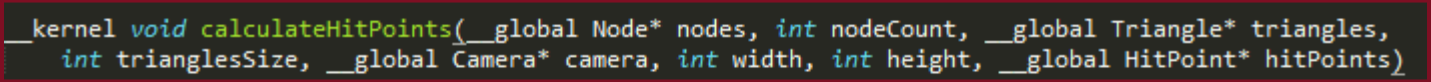
Two kernels:

Intersection Kernel and Color Kernel.

Intersection Kernel:



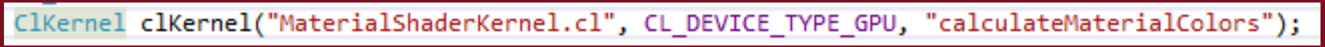
```
ClKernel clKernel("HitPointCalculator.cl", CL_DEVICE_TYPE_GPU, "calculateHitPoints");
```



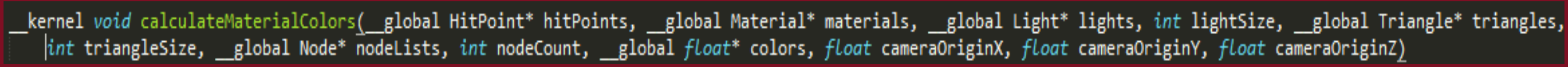
```
__kernel void calculateHitPoints(__global Node* nodes, int nodeCount, __global Triangle* triangles,
    int trianglesSize, __global Camera* camera, int width, int height, __global HitPoint* hitPoints)
```

For intersection, we pass tree nodes, triangles, camera and scene resolution to the kernel function. The output will be put into hitPoints, which is a list of hit points we get from all intersections.

Color Kernel:



```
ClKernel clKernel("MaterialShaderKernel.cl", CL_DEVICE_TYPE_GPU, "calculateMaterialColors");
```

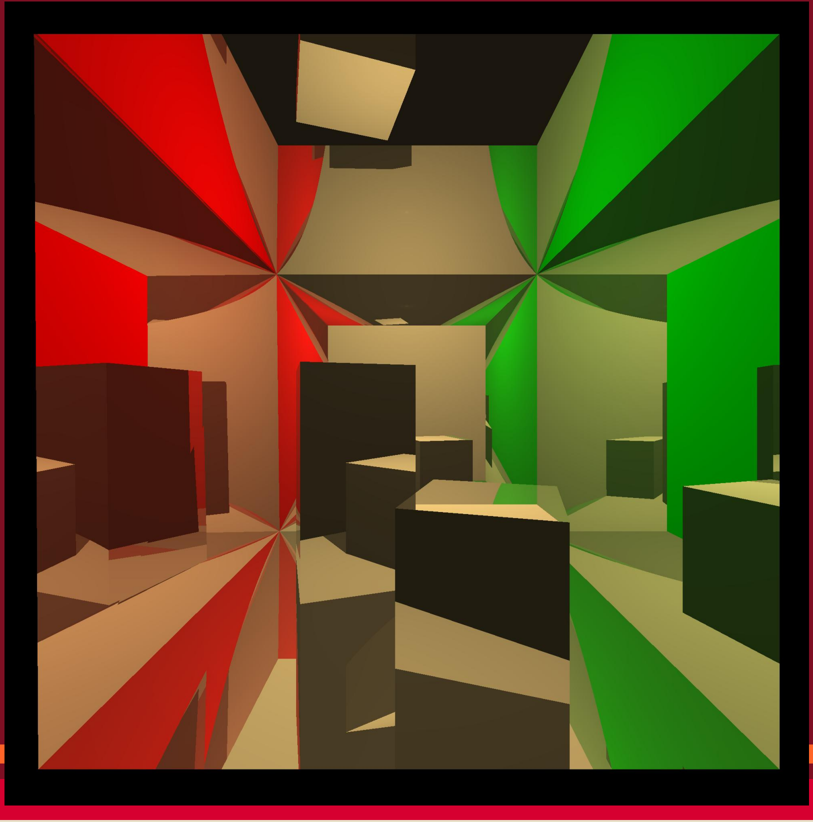


```
__kernel void calculateMaterialColors(__global HitPoint* hitPoints, __global Material* materials, __global Light* lights, int lightSize, __global Triangle* triangles,
    int triangleSize, __global Node* nodeLists, int nodeCount, __global float* colors, float cameraOriginX, float cameraOriginY, float cameraOriginZ)
```

For color calculation, we pass the hitPoints, materials, lights, triangles and nodelists to the GPU. The output is the colors.

GPU Ray Tracing - No Recursion

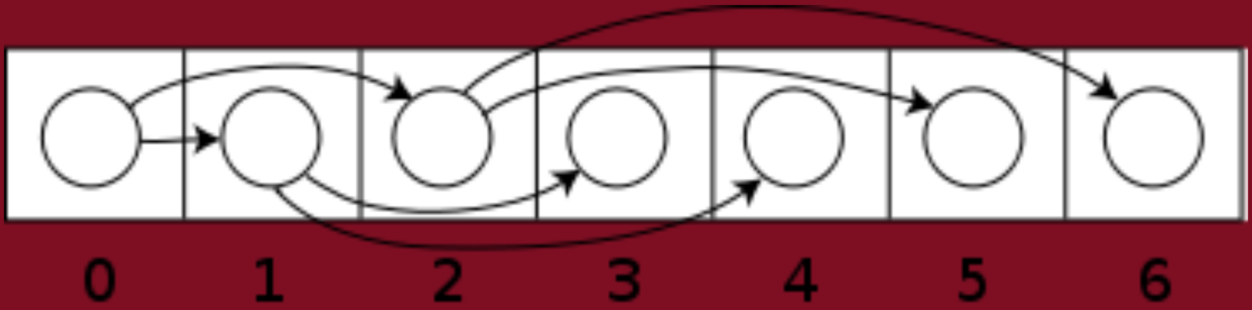
Recursion does not work in GPU
Create a Stack for tree traversal
Use while loop for reflections



We made fractals too!
But only in 3D...

GPU Ray Tracing - Flat Trees

Can’t pass pointers into a kernel
Flat Tree - Tree as list of nodes
Traverse using list indices

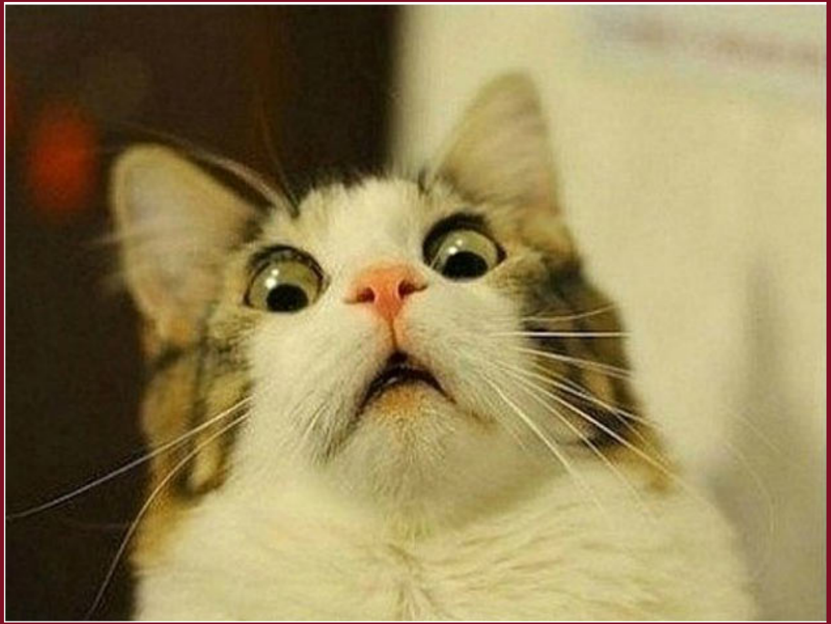


Results

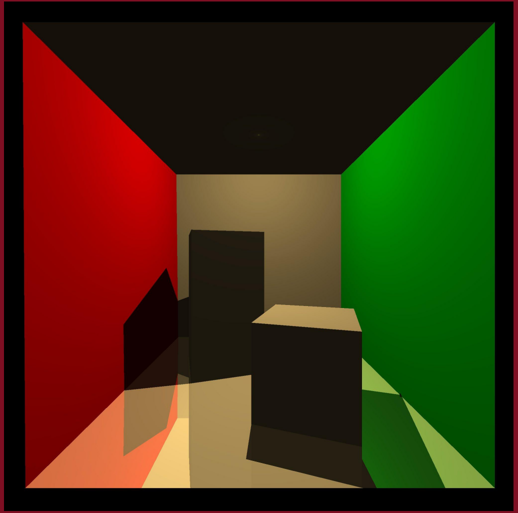
It's fast

But could be faster…

Approximately 10x fps of original ray tracer

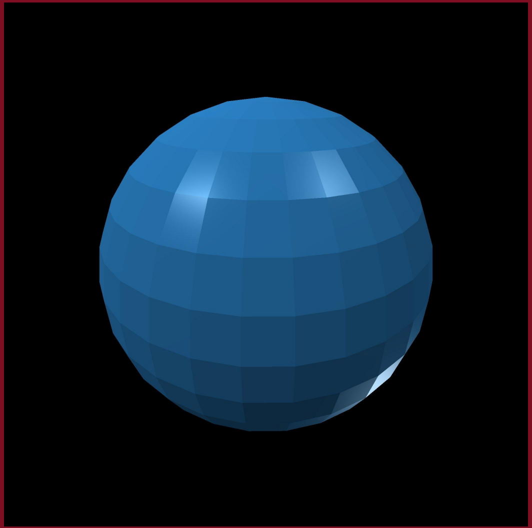


Results - Cornell Box



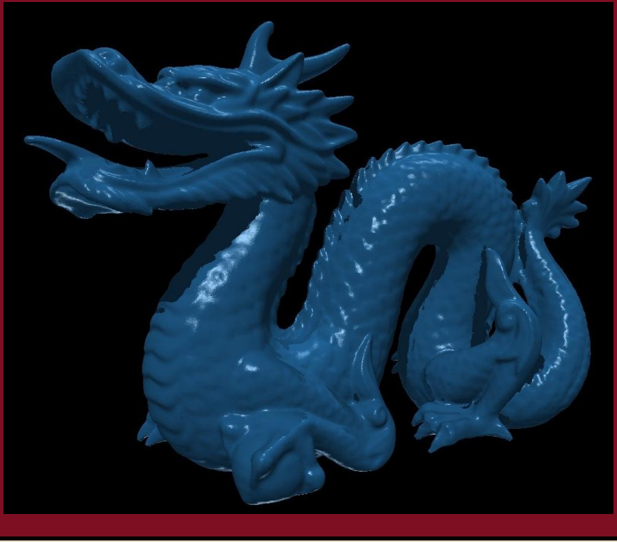
OpenCL - GPU					C++ (CPU)				
Resolution	256	512	1024	2048	Resolution	256	512	1024	2048
Run Time (ms)	6	13	45	183	Run Time (ms)	121	471	1737	7240
FPS	166.7	76.9	22.2	5.5	FPS	8.36	2.12	0.58	0.14

Results - Blue Sphere



OpenCL - GPU					C++ (CPU)				
Resolution	256	512	1024	2048	Resolution	256	512	1024	2048
Run Time (ms)	10	29	99	333	Run Time (ms)	97	375	1459	5521
FPS	100	34.5	10.1	3.00	FPS	10.3	2.67	0.69	0.18

Results - Dragon



OpenCL - GPU					C++ (CPU)				
Resolution	256	512	1024	2048	Resolution	256	512	1024	2048
Run Time (ms)	34	118	468	1580	Run Time (ms)	185	648	2311	8323
FPS	29.4	8.47	2.14	0.63	FPS	5.41	1.54	0.432	0.120

Future Work

Optimizations

- Our tree used spatial splitting - there are better algorithms
- Improve memory reference locality
  - Use less global memory
  - Kernels are faster when referencing contiguous chunks

Future Work (cont.)

Animations

- Providing movement to the scene.
- Can move the camera to create a 360 degree view of the static scene but would like to add additional translations.

Textures

- Allow for drawing images instead of only grabbing visual information from the material

Questions?

Sample Questions

- What is this Ray Tracer thing?
- Does the real world application of Ray Tracing use the GPU or the CPU?
- Where’s the Millennium Falcon?