Scheduling in OptiX™

The NVIDIA ray tracing engine

Austin Robison
The OptiX Engine

- A General Purpose Ray Tracing API
  - Rendering, baking, collision detection, A.I. queries, etc.
  - Modern shader-centric, stateless and bindless design
  - Is not a renderer but can implement many types of renderers

- Highly Programmable
  - Shading with arbitrary ray payloads
  - Ray generation/framebuffer operations (cameras, data unpacking, etc.)
  - Programmable intersection (triangles, NURBS, implicit surfaces, etc.)

- Easy to Program
  - Write single ray code (no exposed ray packets)
  - No need to rewrite programs to target different hardware
  - Acceleration structures are abstracted by the API
Programmable Operations

<table>
<thead>
<tr>
<th>Rasterization</th>
<th>Ray Tracing</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Fragment</td>
<td>• Closest Hit</td>
</tr>
<tr>
<td>• Vertex</td>
<td>• Any Hit</td>
</tr>
<tr>
<td>• Geometry</td>
<td>• Intersection</td>
</tr>
<tr>
<td></td>
<td>• Selector</td>
</tr>
<tr>
<td></td>
<td>• Ray Generation</td>
</tr>
<tr>
<td></td>
<td>• Miss</td>
</tr>
<tr>
<td></td>
<td>• Exception</td>
</tr>
</tbody>
</table>

The ensemble of programs defines the rendering algorithm
(or collision detection algorithm, or sound propagation algorithm, etc.)
Closest Hit Programs: called once after traversal has found the closest intersection
- Used for traditional surface shading
- Deferred shading

Any Hit Programs: called during traversal for each potentially closest intersection
- Transparency without traversal restart (can read textures): `rtIgnoreIntersection()`
- Terminate shadow rays that encounter opaque objects: `rtTerminateRay()`

Both can be used for shading by modifying per ray state
struct PerRayData_radiance
{
    float3 result;
    float importance;
    int depth;
};

rtDeclareVariable(float3, eye);
rtDeclareVariable(float3, U);
rtDeclareVariable(float3, V);
rtDeclareVariable(float3, W);
rtBuffer<float4, 2> output_buffer;
rtDeclareVariable(
    rtNode, top_object);
rtDeclareVariable(
    unsigned int, radiance_ray_type);

rtDeclareSemanticVariable(
    rtRayIndex, rayIndex);

RT_PROGRAM void pinhole_camera()
{
    ...
    float2 d = make_float2(index) / make_float2(screen) * 2.f - 1.f;
    float3 ray_origin = eye;
    float3 ray_direction = normalize(d.x*U + d.y*V + W);
    Ray ray = make_ray(ray_origin, ray_direction, radiance_ray_type, scene_epsilon, RT_DEFAULT_MAX);
    PerRayData_radiance prd;
    prd.importance = 1.f;
    prd.depth = 0;
    rtTrace(top_object, ray, prd);
    output_buffer[index] = prd.result;
}
Execution Model on GT200 Class HW

- **Continuations**
  - Execution is a state machine, presented as recursion
  - Software managed local stack
  - Accomplished through PTX recompilation

- **Persistent Warps**
  - Launch just enough threads to fill the machine
  - Each warp, upon termination of its rays, gets new work

- **Two level load balancing**
  - Balance work between SMs and their persistent warps
  - Balance work between GPUs
Pinhole Camera

Traversal

Phong Shader
while(true):
    switch(state):
        case 0:
            ...code for state0...
        case 1:
            ...code for state1...
        case 2:
            ...code for state2...
        ...
Call to rtTrace()
All threads enter traversal, hit the Phong material
All cast secondary rays via rtTrace()
Back to traversal, some rays hit again and some miss

1 2

3 2

4 2

5 2
We now have divergence in the warp’s execution. What is the minimum number of steps to state 5?
while(true):
    schedule = schedule_state()
    if(state == schedule):
        switch(state):
            case 0:
                ...code for state0...
            case 1:
                ...code for state1...
            case 2:
                ...code for state2...
            ...

4 State Transitions to Finish
3 State Transitions to Finish
Per-pixel Render Time

Warp Synchronous

Prioritized
Frame rate: 1.25x
Warp Divergence

Warp Synchronous

Prioritized
States executed: 0.74x
Warp Synchronous State History
Prioritized State History
Warp Executions (Lower is Better)

- **Warp Synchronous**
- **Prioritized**
OptiX SDK Release
Available to registered developers in early fall from
http://www.nvidia.com
Go See Steve Parker’s talk

**Wednesday at 2:45**

SIGGRAPH Room 294

for more API information

and a short tutorial
Questions?

arobison@nvidia.com

http://www.nvidia.com