



# Vulkan Ray Tracing – 5 New Shader Types!



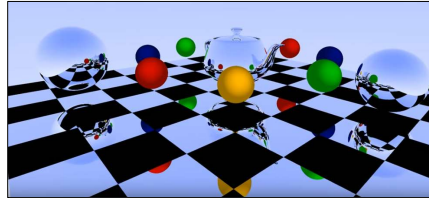
Oregon State University

Mike Bailey

mjb@cs.oregonstate.edu



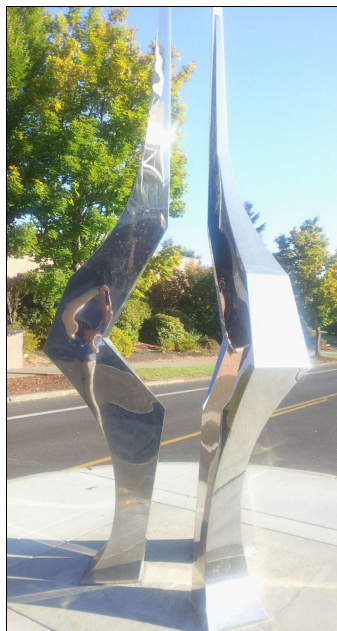
This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)



VulkanRayTracing.pptx

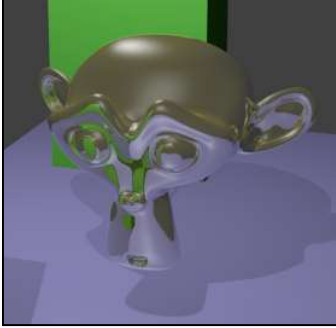
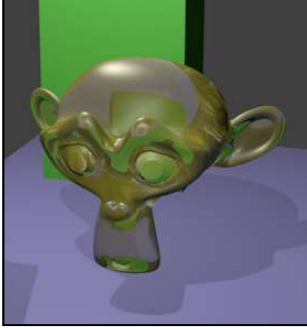
mjb – March 5, 2023

## Analog Ray Tracing Example ☺

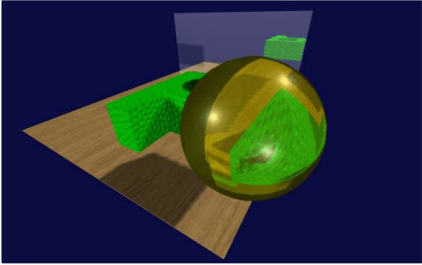


mjb – March 5, 2023


### Digital Ray Tracing Examples



Blender

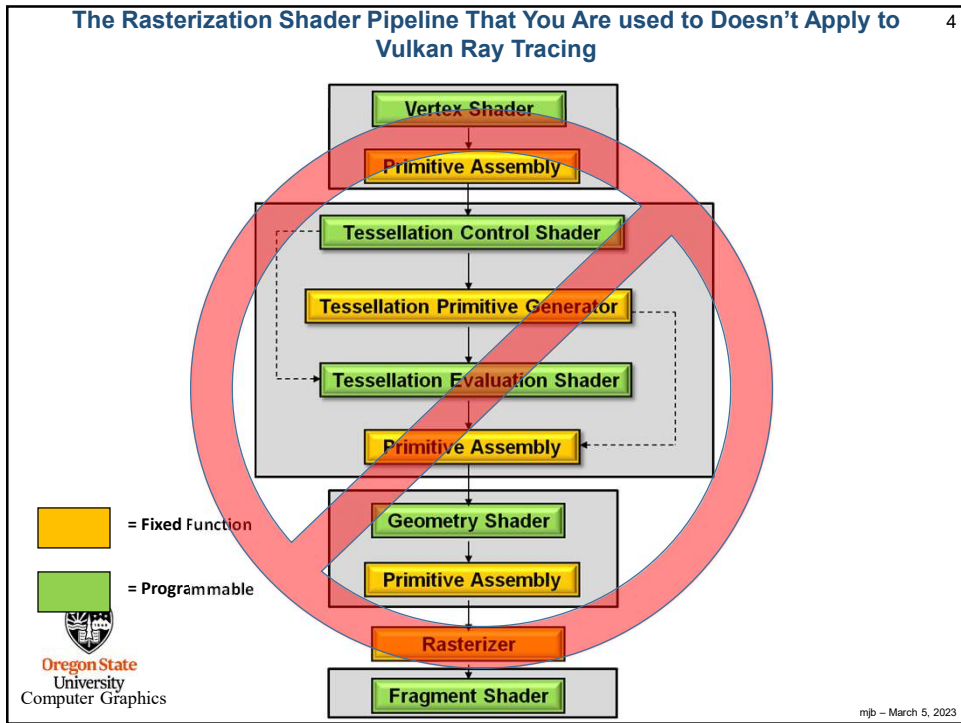


IronCad

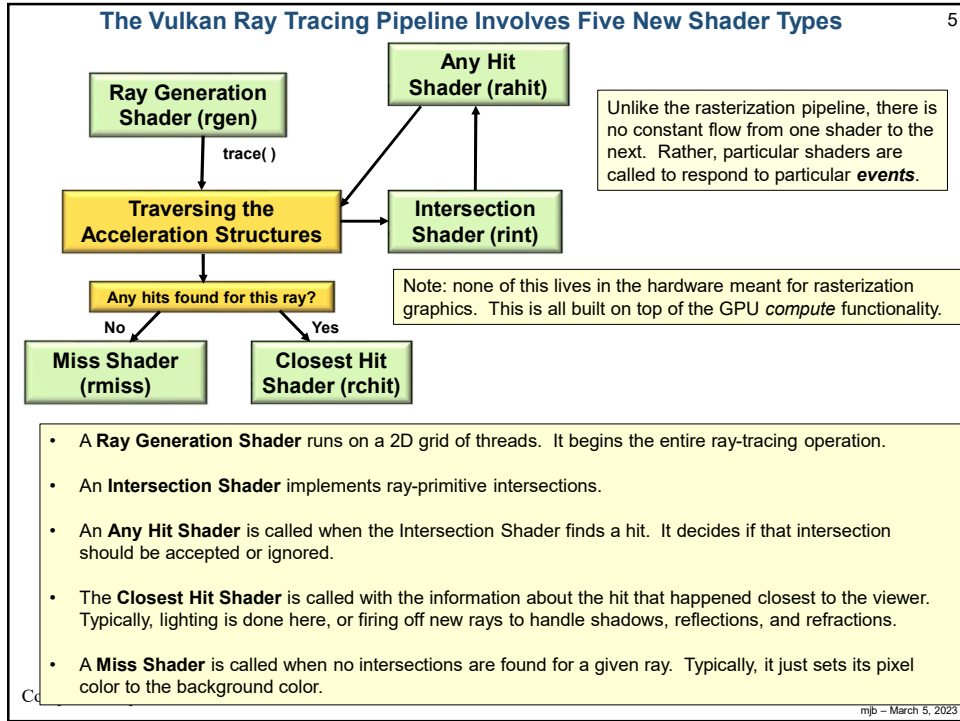


mjb - March 5, 2023

3



4



5

### Example: The Ray Intersection Process for a Sphere

6

Sphere equation:  $(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2 = R^2$   
 Ray equation:  $(x,y,z) = (x_0,y_0,z_0) + t*(dx,dy,dz)$

Plugging  $(x,y,z)$  from the second equation into the first equation and multiplying-through and simplifying gives:

$$At^2 + Bt + C = 0 \quad \rightarrow \quad t_1, t_2 = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Solve for  $t_1, t_2$  and analyze the solution like this:

1. If both  $t_1$  and  $t_2$  are complex (i.e., have an imaginary component), then the ray missed the sphere completely.
2. If both  $t_1$  and  $t_2$  are real and identical, then the ray brushed the sphere at a tangent point.
3. If both  $t_1$  and  $t_2$  are real and different, then the ray entered and exited the sphere.

In Vulkan terms:  
`gl_WorldRayOrigin = (x0,y0,z0)`  
`gl_Hit = t`  
`gl_WorldRayDirection = (dx,dy,dz)`

Computer Graphics

mjb - March 5, 2023

6

**Example: The Ray Intersection Process for a Cube** 7


Plane equation:  $Ax + By + Cz + D = 0$   
 Ray equation:  $(x,y,z) = (x_0,y_0,z_0) + t^*(dx,dy,dz)$

Plugging  $(x,y,z)$  from the second equation into the first equation and multiplying through and simplifying gives:

$Qt + R = 0$   
 Solve for  $t = -R/Q$

A cube is actually the intersection of 6 half-space planes (just 4 are shown here). Each of these will produce its own  $t$  intersection value. Treat them as pairs:  $(t_{x1}, t_{x2})$ ,  $(t_{y1}, t_{y2})$ ,  $(t_{z1}, t_{z2})$

The ultimate cube entry and exit values are:

  
**Oregon State**  
 University  
 Computer Graphics

$$t_{\min} = \max(\min(t_{x1}, t_{x2}), \min(t_{y1}, t_{y2}), \min(t_{z1}, t_{z2}))$$

$$t_{\max} = \min(\max(t_{x1}, t_{x2}), \max(t_{y1}, t_{y2}), \max(t_{z1}, t_{z2}))$$

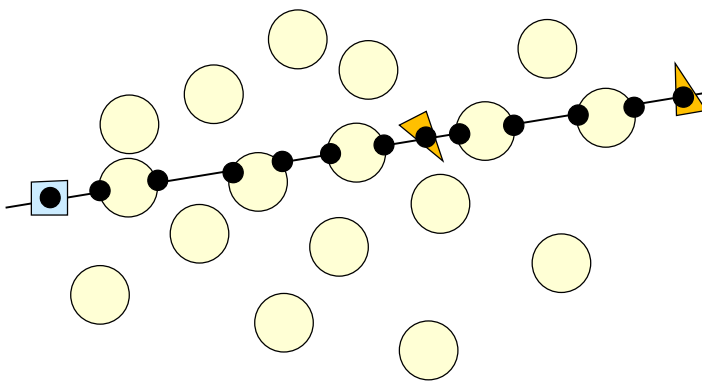
}


This algorithm works for all convex solids (e.g., cylinder, cone)

mjb - March 5, 2023

7

**In a Raytracing, each ray typically hits a lot of Things** 8



  
**Oregon State**  
 University  
 Computer Graphics

mjb - March 5, 2023

8

### Acceleration Structures

- A Bottom-level Acceleration Structure (BLAS) reads the vertex data from vertex and index VkBuffers to determine bounding boxes.
- You can also supply your own bounding box information to a BLAS.
- A Top-level Acceleration Structure (TLAS) holds transformations and pointers to multiple BLASes.
- The BLAS is essentially used as a Model Coordinate bounding box, while the TLAS is used as a World Coordinate bounding box.

Oregon State University  
Computer Graphics

... March 5, 2023

9

### Ray Generation Shader

Gets each individual ray going and writes the final color to the pixel

```

layout( location = 1 ) rayPayload myPayload
{
    vec4 color;
};

void main()
{
    trace( topLevel, ... 1 )
    imageStore( framebuffer, gl_GlobalInvocationID.xy, color );
}
    
```

A "payload" is information that keeps getting passed through the processing of an individual ray. Different stages can add to it. It is finally consumed at the very end, in this case by writing *color* into the pixel being worked on.

Oregon State University  
Computer Graphics

mjb - March 5, 2023

10

### A New Built-in GLSL Function

```

void trace
(
    VkAccelerationStructure    topLevel,    // TLAS
    uint                       rayFlags,
    uint                       cullMask,
    uint                       sbtRecordOffset,
    uint                       sbtRecordStride,
    uint                       missIndex,
    vec3                       origin,      // x0, y0, z0
    float                       tmin,       // minimum t to allow (near)
    vec3                       direction,  // dx, dy, dz
    float                       tmax,       // maximum t to allow (far)
    int                         payload
);
        
```

In Vulkan terms (these are built-ins accessible from GLSL):  
**gl\_WorldRayOrigin** = (x<sub>0</sub>,y<sub>0</sub>,z<sub>0</sub>)  
**gl\_Hit** = t  
**gl\_WorldRayDirection** = (dx,dy,dz)

Oregon State University Computer Graphics mjb - March 5, 2023

11

### Sample Intersection Shader Code

```

hitAttribute vec3 attribs
void main()
{
    SpherePrimitive sph = spheres[ gl_PrimitiveID ];
    vec3 orig = gl_WorldRayOrigin;
    vec3 dir = normalize( gl_WorldRayDirection );
    ...
    float discr = b*b - 4.*a*c;
    if( discr < 0. )
        return;

    float tmp = ( -b - sqrt(discr) ) / ( 2.*a );
    if( gl_RayTmin < tmp && tmp < gl_RayTmax )
    {
        vec3 p = orig + tmp * dir;
        attribs = p;
        reportIntersection( tmp, 0 );
        return;
    }
    tmp = ( -b + sqrt(discr) ) / ( 2.*a );
    if( gl_RayTmin < tmp && tmp < gl_RayTmax )
    {
        vec3 p = orig + tmp * dir;
        attribs = p;
        reportIntersection( tmp, 0 );
        return;
    }
}
        
```

Intersect a ray with an arbitrary 3D object.  
 Passes data to the **Any Hit** shader.  
 There is a built-in ray-triangle **Intersection Shader**.

```

graph TD
    RGS[Ray Generation Shader (rgen)] -- trace() --> TAS[Traversing the Acceleration Structures]
    TAS --> RIS[Intersection Shader (rint)]
    TAS --> AHS[Any Hit Shader (rahit)]
    AHS --> RIS
    AHS --> MS[Miss Shader (miss)]
    AHS --> CHS[Closest Hit Shader (rchit)]
    
```

Oregon State University Computer Graphics mjb - March 5, 2023

12


### Sample Miss Shader Code

Handles a ray that doesn't hit *any* objects

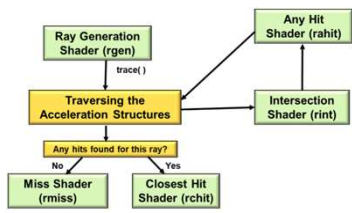
```

Layout( location=0 ) rayPayload
{
    vec4 color;
} myPayload;

void
main( )
{
    myPayload.color = vec4( 0., 0., 0., 1. );
}
    
```



Oregon State  
University  
Computer Graphics



```

graph TD
    rgen[Ray Generation Shader (rgen)] -- trace() --> tra[Traversing the Acceleration Structures]
    tra --> rint[Intersection Shader (rint)]
    rint --> ra[Any Hit Shader (rahit)]
    ra -- Yes --> tra
    tra --> q[Any hits found for this ray?]
    q -- No --> rmiss[Miss Shader (rmiss)]
    q -- Yes --> rchit[Closest Hit Shader (rchit)]
    
```

mjb - March 5, 2023

13

### Sample Any Hit Shader Code


Handle a ray that hits *anything*.  
Store information on each hit.  
Can reject a hit.

```

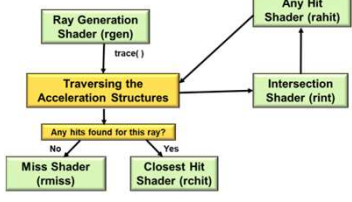
layout( binding = 4, set = 0 ) buffer outputProperties
{
    float outputValues[ ];
} outputData;

layout(location = 0) rayPayloadIn uint outputId;
layout(location = 1) rayPayloadIn uint hitCounter;
hitAttribute vec3 attribs;

void
main( )
{
    outputData.outputValues[ outputId + hitCounter ] = gl_PrimitiveID;
    hitCounter = hitCounter + 1;
}
    
```



Oregon State  
University  
Computer Graphics



```

graph TD
    rgen[Ray Generation Shader (rgen)] -- trace() --> tra[Traversing the Acceleration Structures]
    tra --> rint[Intersection Shader (rint)]
    rint --> ra[Any Hit Shader (rahit)]
    ra -- Yes --> tra
    tra --> q[Any hits found for this ray?]
    q -- No --> rmiss[Miss Shader (rmiss)]
    q -- Yes --> rchit[Closest Hit Shader (rchit)]
    
```

mjb - March 5, 2023

14

### Sample Closest Hit Shader

Handle the intersection closest to the viewer.  
 Collects data from calls to the Any Hit shader.  
 This shader can spawn more rays to handle shadows, reflections, and refractions.

```

uniform sampler2D uTexUnit;

rayPayload myPayload
{
    vec4 color;
};

void main()
{
    vec3 stp = gl_WorldRayOrigin + gl_Hit * gl_WorldRayDirection;
    color = texture( uTexUnit, stp ); // material properties lookup
}
    
```

In Vulkan terms:  
**gl\_WorldRayOrigin** = (x<sub>0</sub>, y<sub>0</sub>, z<sub>0</sub>)  
**gl\_Hit** = t  
**gl\_WorldRayDirection** = (dx, dy, dz)

```

graph TD
    rgen[Ray Generation Shader (rgen)] -- traceNV() --> tra[Traversing the Acceleration Structures]
    tra --> rint[Intersection Shader (rint)]
    tra --> q[Any hits found for this ray?]
    q -- No --> rmiss[Miss Shader (rmiss)]
    q -- Yes --> rhit[Closest Hit Shader (rhit)]
    rhit --> rahit[Any Hit Shader (rahit)]
    
```

mjb - March 5, 2023

15

### Other New Built-in Functions

void **terminateRay**( );

void **ignoreIntersection**( );

}

Loosely equivalent to “discard”

void **reportIntersection**( float hit, uint hitKind );

Oregon State University  
Computer Graphics

mjb - March 5, 2023

16




17

### The Trigger comes from the Command Buffer: vkCmdBindPipeline( ) and vkCmdTraceRays( )

```
vkCmdBindPipeline( CommandBuffer, VK_PIPELINE_BIND_POINT_RAY_TRACING, RaytracePipeline );
```

```
vkCmdTraceRays( CommandBuffer,
    raygenShaderBindingTableBuffer,
    raygenShaderBindingOffset,
    missShaderBindingTableBuffer,
    missShaderBindingOffset,
    missShaderBindingStride,
    hitShaderBindingTableBuffer,
    hitShaderBindingOffset,
    hitShaderBindingStride,
    callableShaderBindingTableBuffer,
    callableShaderBindingOffset,
    callableShaderBindingStride
    width,
    height,
    depth );
```



mjb - March 5, 2023

17

Check This Out!

18



<https://www.youtube.com/watch?v=QL7sXc2iNJ8>




mjb - March 5, 2023

18