Firing Rays

Firing Rays First Takes Place in the Ray Generation Shader

New shader stage names:

- VK_SHADER_STAGE_RAYGEN_BIT
- VK_SHADER_STAGE_ANY_HIT_BIT
- VK_SHADER_STAGE_CLOSEST_HIT_BIT
- VK_SHADER_STAGE_MISS_BIT
- VK_SHADER_STAGE_INTERSECTION_BIT
- VK_SHADER_STAGE_CALLABLE_BIT
The Trigger comes from the Command Buffer: 
vkCmdBindPipeline( ) and vkCmdTraceRays( )

```c
vkCmdBindPipeline( CommandBuffer, VK_PIPELINE_BIND_POINT_RAYTRACING, RayTracePipeline );
vkCmdTraceRays( CommandBuffer,
    raygenShaderBindingTable
    missShaderBindingTable, 
    hitShaderBindingTable,
    callableShaderBindingTable,
    width,
    height,
    1 // depth
);
```

What Is a Shader Binding Table (SBT)?

When a ray hits a piece of geometry in the scene, the system must figure out what set of shaders need to be called to handle intersections and shading calculations.

This set of shaders is called the Shader Binding Table (SBT).
That causes the Raygen Shaders to make Numerous Calls to traceRay(

```cpp
traceRay(
    TopLevelAccelerationStructure,
    gl_RayFlagsOpaque,    // ray flags
    0xff                   // the culling mask
    sbtOffset,             // used to lookup the hit group in the SBT
    sbtStride,             // used to lookup the hit group in the SBT
    missIndex,             // which miss shader to call in the shader group
    eyePosition,           // the vec3 ray origin
    tmin,                  // minimum t to allow for an intersection
    rayDir,                // the ray direction
    tmax,                  // maximum t to allow for an intersection
    0                      // location number holding the payload
);
```

layout( location=0, rayPayload vec4 payload );  // color

```cpp
imageStore( imageIndex, ivec2(gl_LaunchID), payload );
```

The Ray that Gets Fired

```cpp
traceRay(
    TopLevelAccelerationStructure,
    gl_RayFlagsOpaque,    // ray flags
    0xff                   // the culling mask
    sbtOffset,             // used to lookup the hit group in the SBT
    sbtStride,             // used to lookup the hit group in the SBT
    missIndex,             // which miss shader to call in the shader group
    eyePosition,           // the vec3 ray origin
    tmin,                  // minimum t to allow for an intersection
    rayDir,                // the ray direction
    tmax,                  // maximum t to allow for an intersection
    0                      // location number holding the payload
);
```

float tmin = 0.01;
float tmax = 1000.;
vec3 rayDir = compute_ray_dir( gl_LaunchID, gl_LaunchSize );
Computing the Ray Direction

```
vec3 RayDirection( uvec3 launchID, uvec3 launchSize )
{
  float x = -1. + ( 2. * float(launchID.x) + 0.5 ) / float(launchSize.x); // [-1.,+1.]
  float y = -1. + ( 2. * float(launchID.y) + 0.5 ) / float(launchSize.y); // [-1.,+1.]
  y = -y;
  vec4 ecDirecton = inverseModelViewProjection * vec4( x, y, 0., 1. );
  return normalize( ecDirecton.xyz );
}
```

A Closest Hit Shader can also make calls to traceRay( )

Miss Shader (`mmiss`)

Closest Hit Shader (`rchit`)

Reflections

Normal Vector

Shadows

Refractions

Normal Vector
Shadows, Reflections, and Refractions