



Ray Tracing Pipeline Data Structure (RTPDS)



Oregon State
University

Mike Bailey

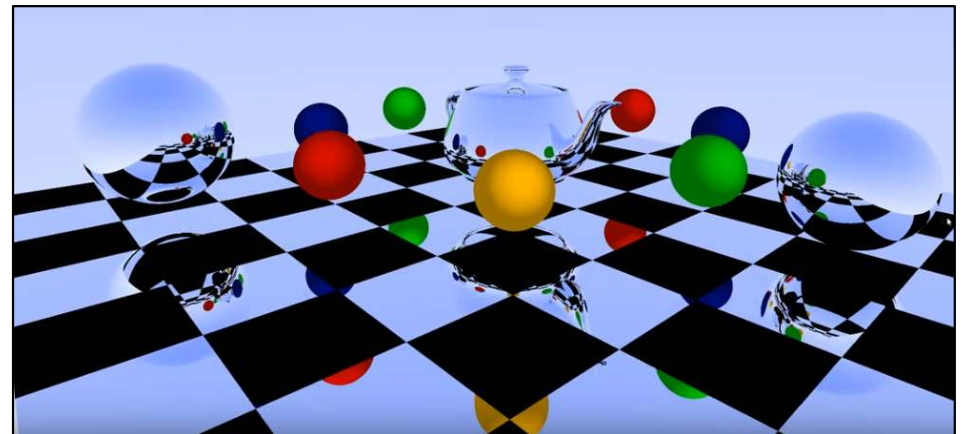
mjb@cs.oregonstate.edu



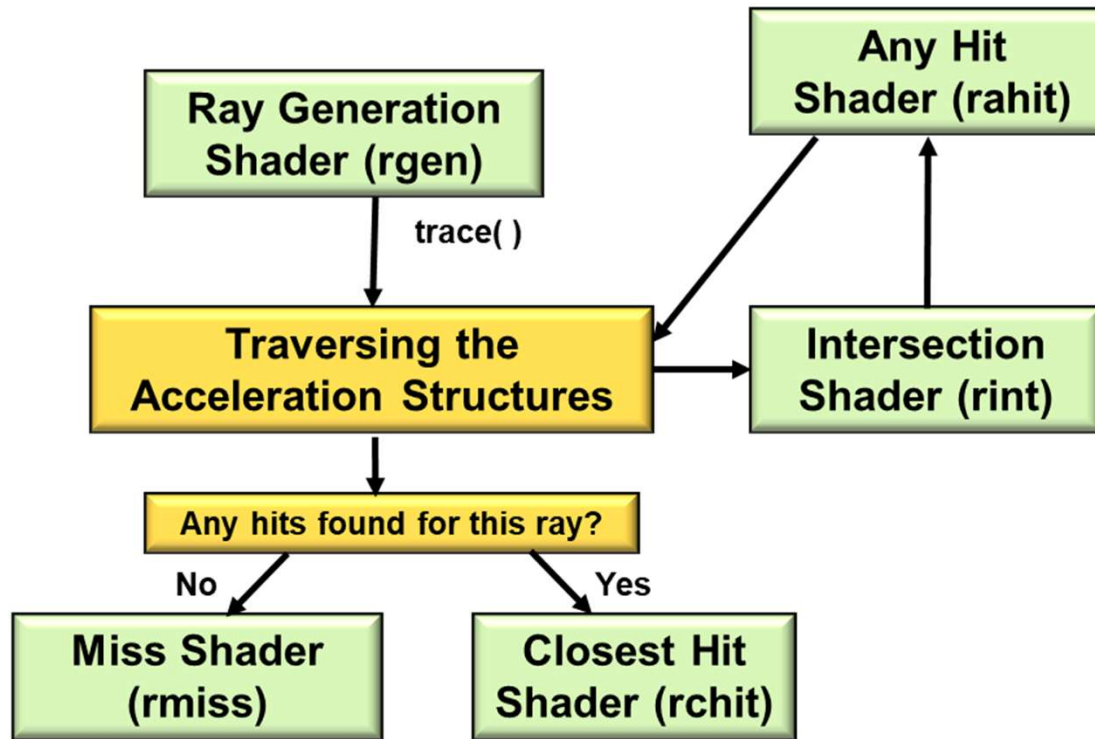
This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](https://creativecommons.org/licenses/by-nc-nd/4.0/)



Oregon State
University
Computer Graphics



Ray Trace Pipeline Shader Stages



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
  
```



Ray Trace Pipeline Data Structure

3

VkPipelineLayout
VkPipeline

RayTracePipelineLayout;
RayTracePipeline;

VkPipelineLayoutCreateInfo

```
    vplci.sType           = VK_STRUCTURE_TYPE_PIPELINE_LAYOUT_CREATE_INFO;
    vplci.pNext           = nullptr;
    vplci.flags           = 0;
    vplci.setLayoutCount  = 1;
    vplci.pSetLayouts     = &descriptorSetLayout;    // as usual
    vplci.pushConstantRangeCount = 0;                // as usual
    vplci.pPushConstantRanges = nullptr;             // as usual
```

```
result = vkCreatePipelineLayout( LogicalDevice, IN &vplci, nullptr, OUT &RayTracePipelineLayout);
```

VkRayTracingPipelineCreateInfo

```
    vrtpci.sType           = VK_STRUCTURE_TYPE_RAY_TRACING_PIPELINE_CREATE_INFO;
    vrtpci.pNext           = nullptr;
    vrtpci.flags           = 0;
    vrtpci.stageCount      = << # of shader stages in the ray-trace pipeline >>
    vrtpci.pStages         = << array of VkPipelineShaderStageCreateInfo >>
    vrtpci.groupCount      = << # of shader groups in the ray-trace pipeline >>
    vrtpci.pGroups         = << an array of VkRayTracingShaderGroupCreateInfo >>
    vrtpci.maxRecursionDepth = MAXRECURSIONS;        // at least 1, perhaps more
    vrtpci.layout           = RayTracePipelineLayout;
    vrtpci.pDynamicState    = nullptr;
    vrtpci.basePipelineHandle = VK_NULL_HANDLE;
    vrtpci.basePipelineIndex = 0;
```

```
result = vkCreateRayTracingPipelines( LogicalDevice, VK_NULL_HANDLE, VK_NULL_HANDLE, 1, IN &vrtpci,  
PALLOCATOR, OUT &RayTracePipeline);
```

```
VkPipelineShaderStageCreateInfo    vpssci[2];
    vpssci[0].sType = VK_STRUCTURE_TYPE_PIPELINE_SHADER_STAGE_CREATE_INFO;
    vpssci[0].pNext = nullptr;
    vpssci[0].flags = 0;
    vpssci[0].stage = VK_SHADER_STAGE_CLOSEST_HIT_VERTEX_BIT;
    vpssci[0].module = closestHitShader;
    vpssci[0].pName = "main";
    vpssci[0].pSpecializationInfo = (VkSpecializationInfo *)nullptr;

    vpssci[1].sType = VK_STRUCTURE_TYPE_PIPELINE_SHADER_STAGE_CREATE_INFO;
    vpssci[1].pNext = nullptr;
    vpssci[1].flags = 0;
    vpssci[1].stage = VK_SHADER_STAGE_MISS_BIT;
    vpssci[1].module = missShader;
    vpssci[1].pName = "main";
    vpssci[1].pSpecializationInfo = (VkSpecializationInfo *)nullptr;

    . . .
```



```
VkRayTracingShaderGroupCreateInfo          vrtsgci[2];          // one for each shader group
vrtsgci[0].sType = VK_STRUCTURE_TYPE_RAY_TRACING_SHADER_GROUP_CREATE_INFO;
vrtsgci[0].pNext = nullptr;
vrtsgci[0].type = VK_RAY_TRACING_SHADER_GROUP_TYPE_TRIANGLES_HIT_GROUP;
vrtsgci[0].generalShader = << index of Miss Shader in pStages >>;
vrtsgci[0].closestHitShader = << index of Closest Hit Shader in pStages >>;
vrtsgci[0].anyHitShader = << index of Any Hit Shader in pStages >>
vrtsgci[0].pShaderGroupCaptureReplayHandle = (VkSpecializationInfo *)nullptr;

...
```

If type is:

```
vrtsgci[0].type = VK_RAY_TRACING_SHADER_GROUP_TYPE_PROCEDURAL_HIT_GROUP;
```

Then need to provide an:

```
vrtsgci[0].intersectionShader = ...
```

