Real-Time Rendering

186.140, 2021W, VU, 4.5ECTS

Vulkan Lecture Series, Episode 6:

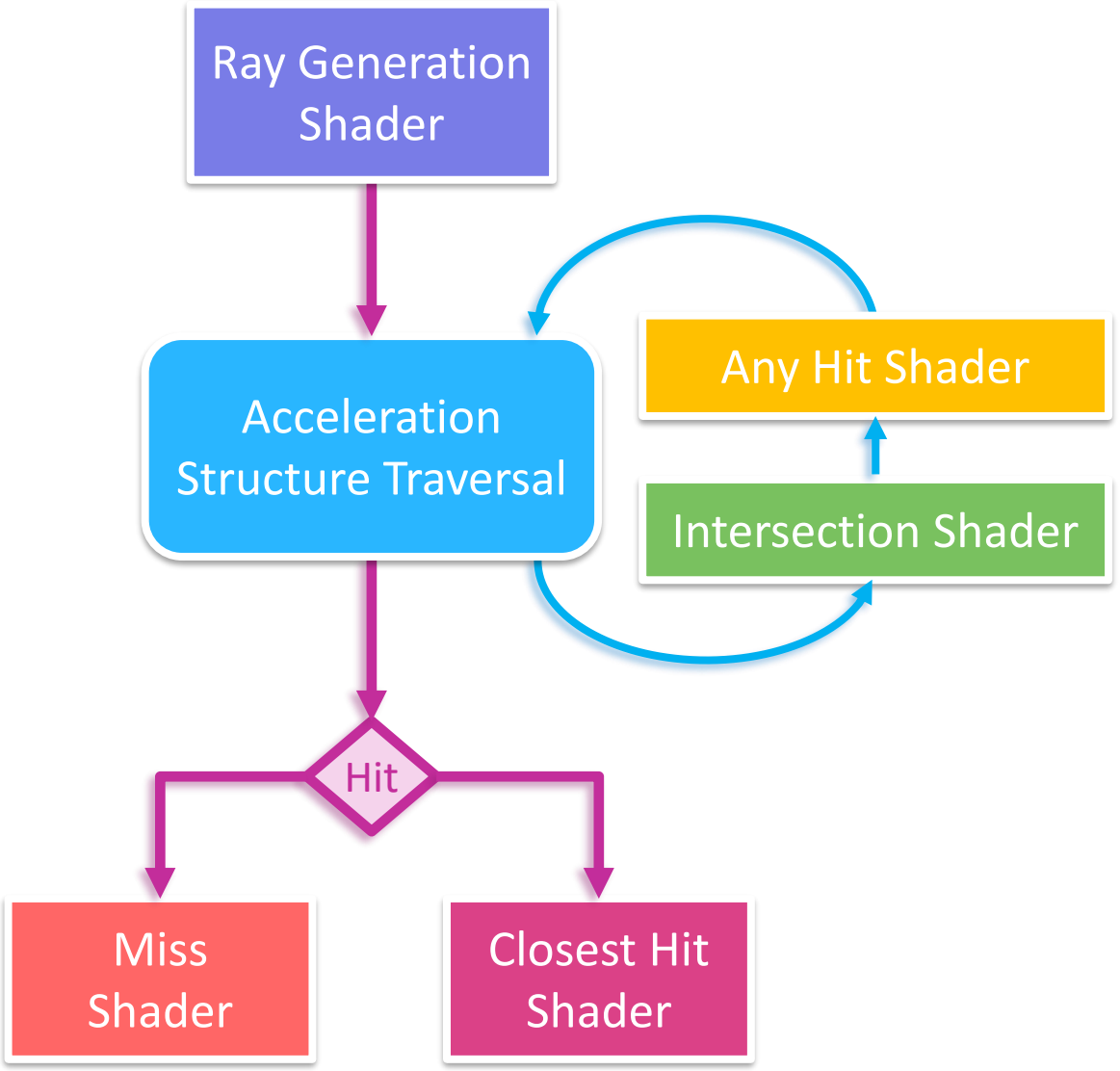
Real-Time Ray Tracing

Johannes Unterguggenberger

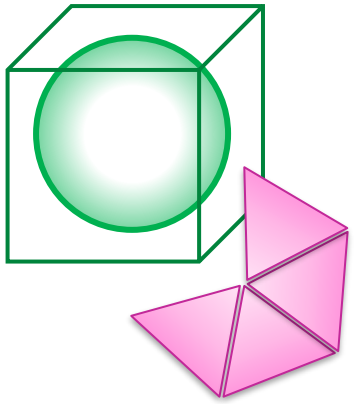
Institute of Visual Computing & Human-Centered Technology

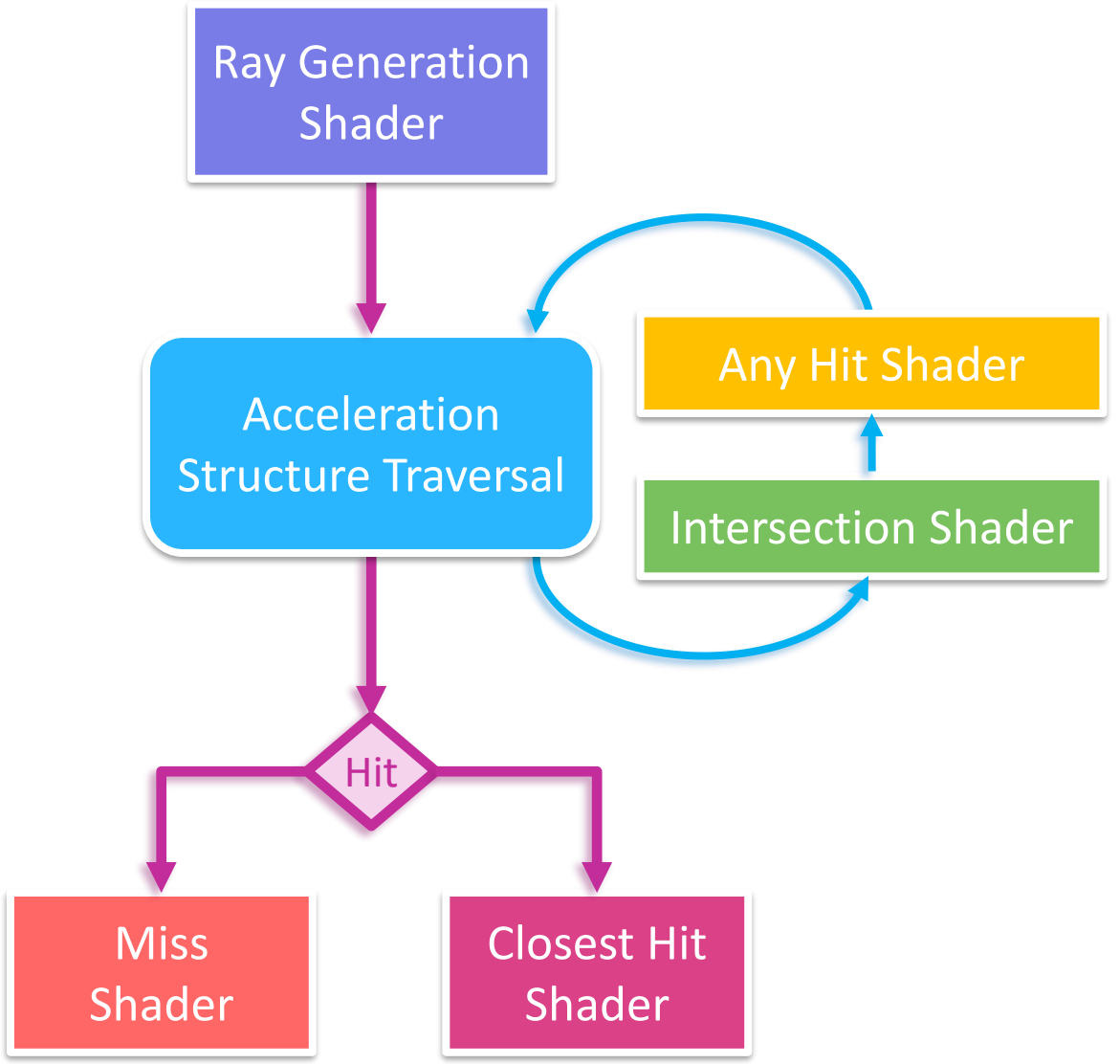
TU Wien, Austria

Ray Tracing Pipeline

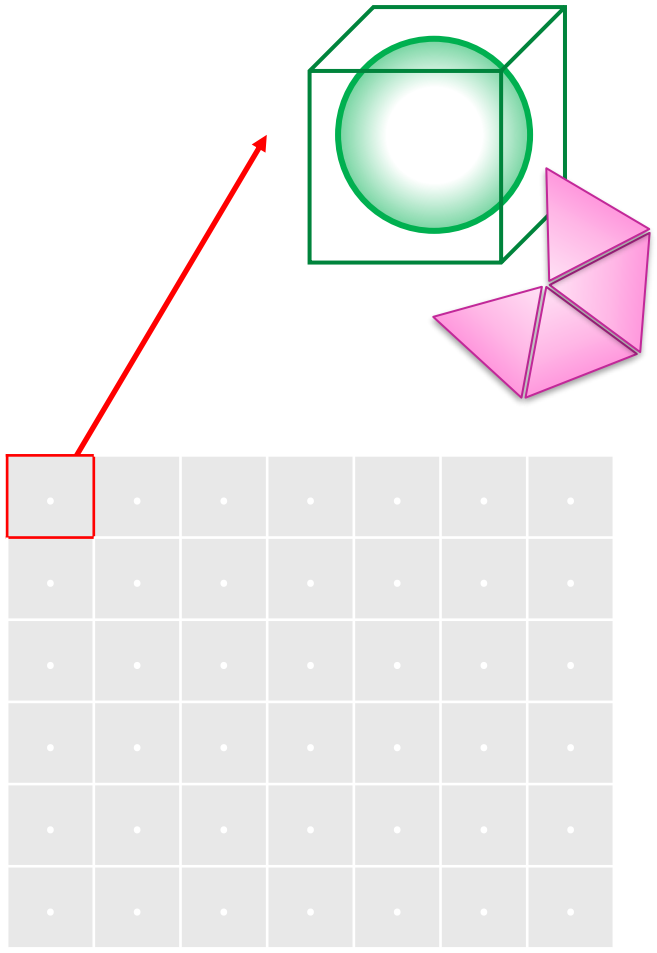


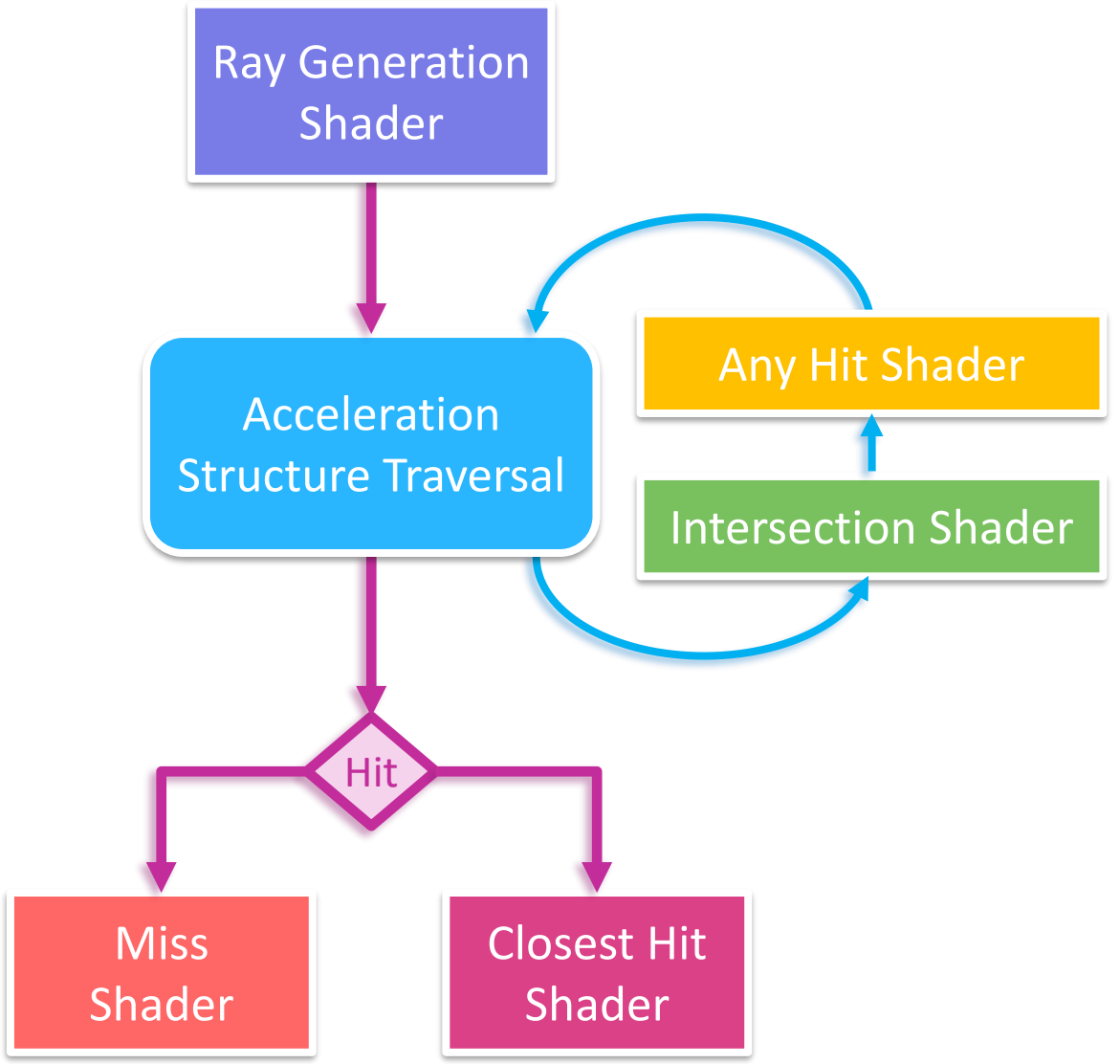
Ray Tracing Pipeline



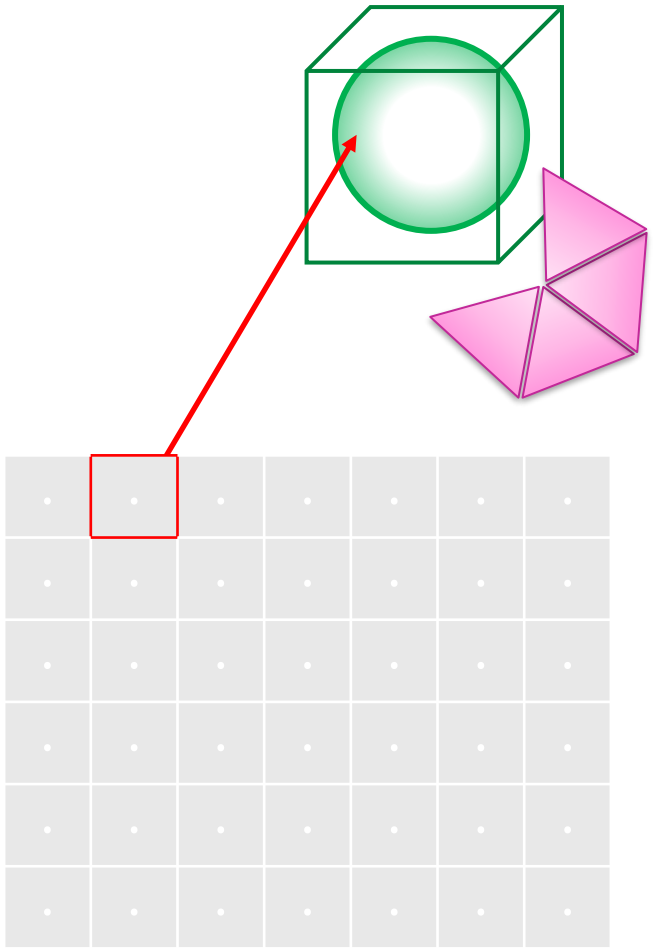


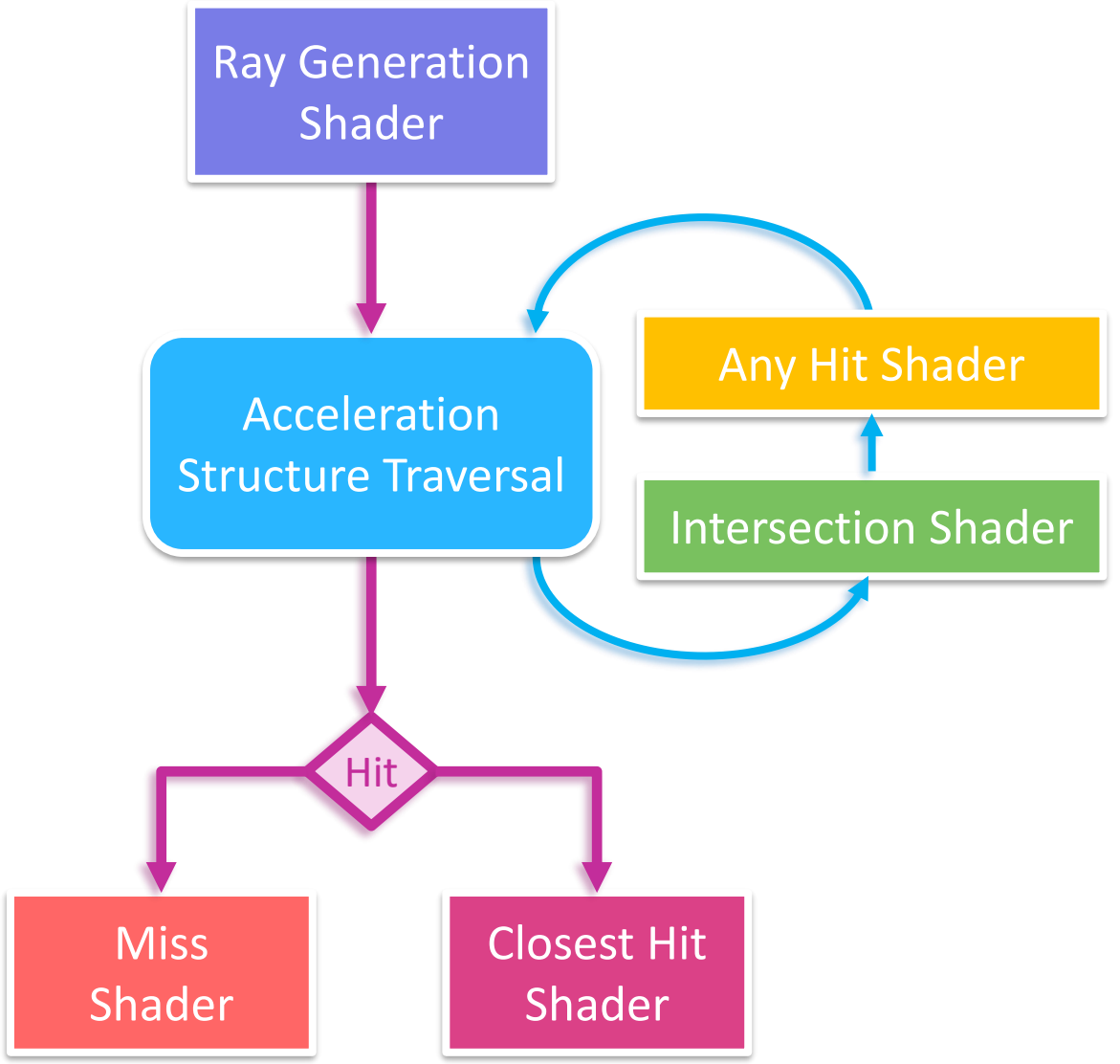
Ray Tracing Pipeline



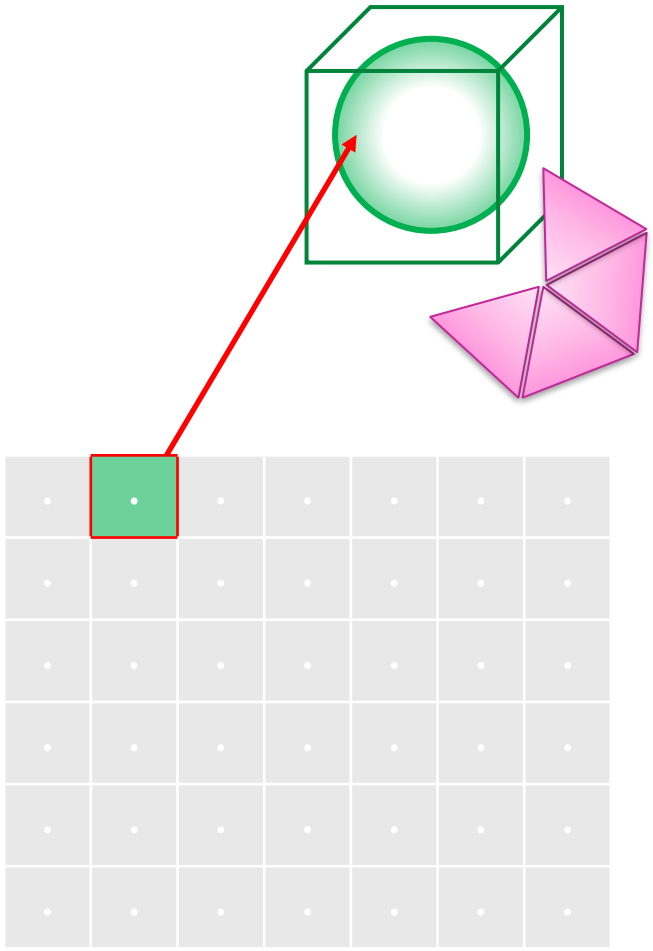


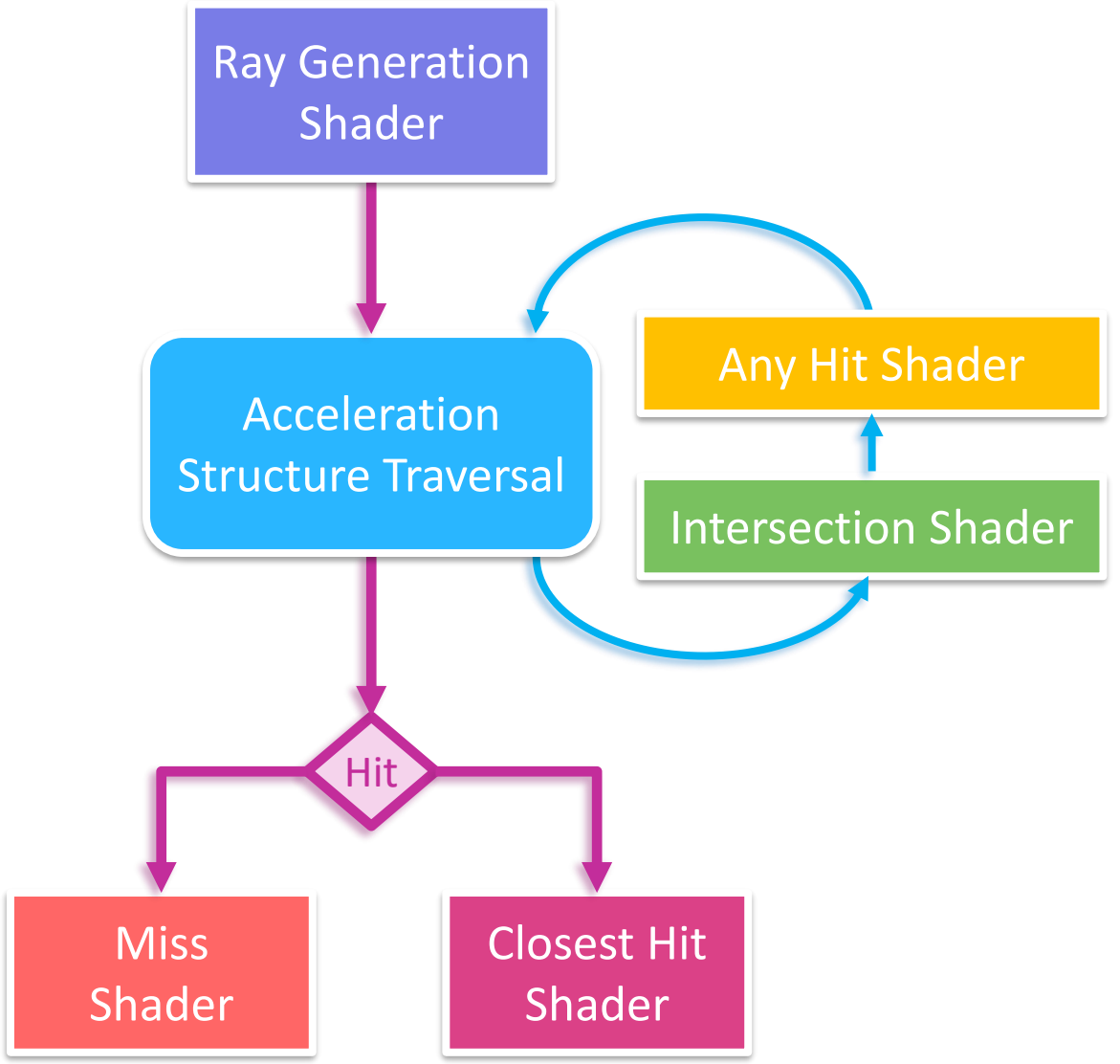
Ray Tracing Pipeline



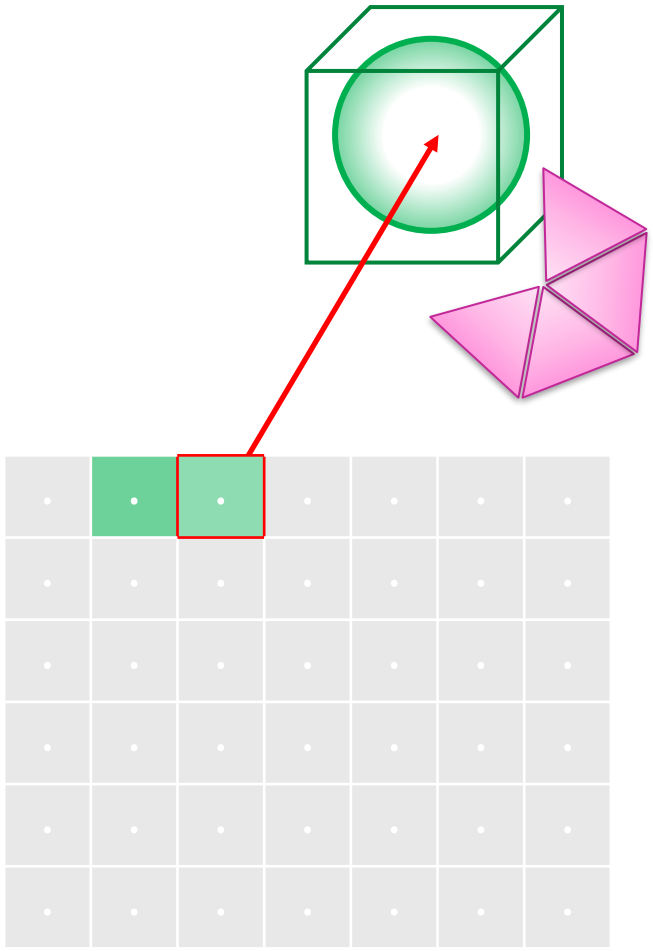


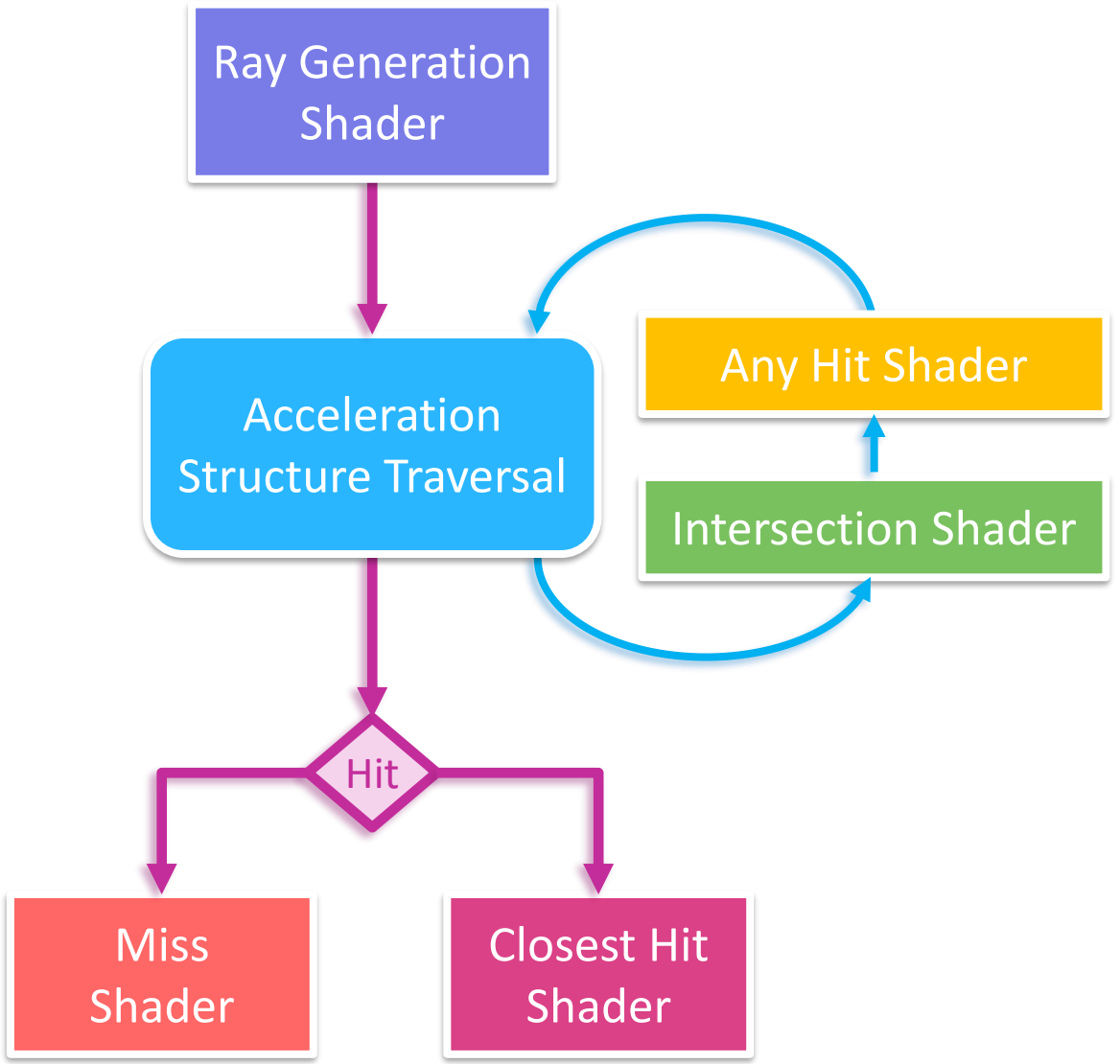
Ray Tracing Pipeline



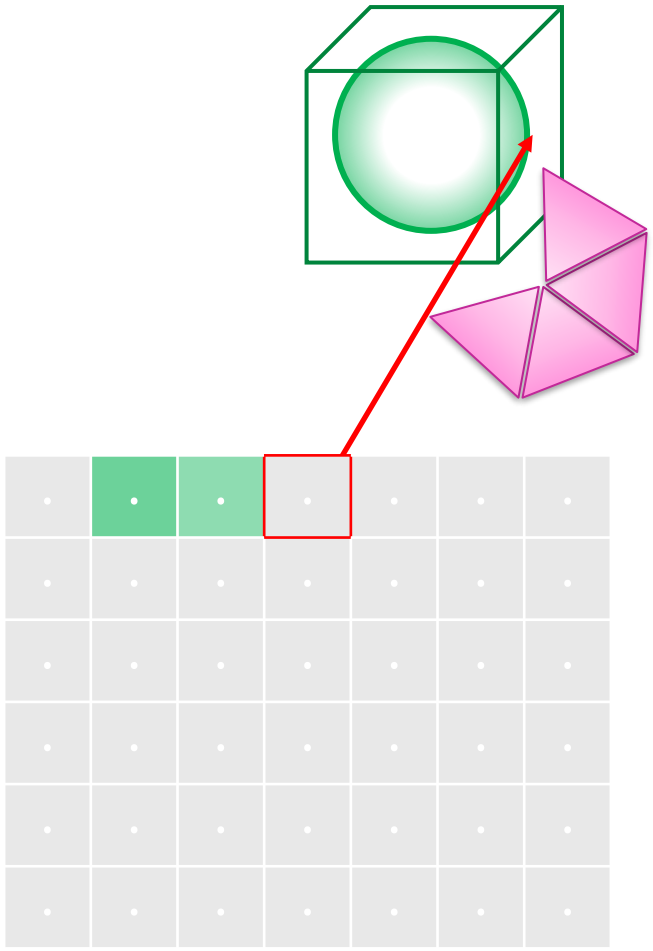


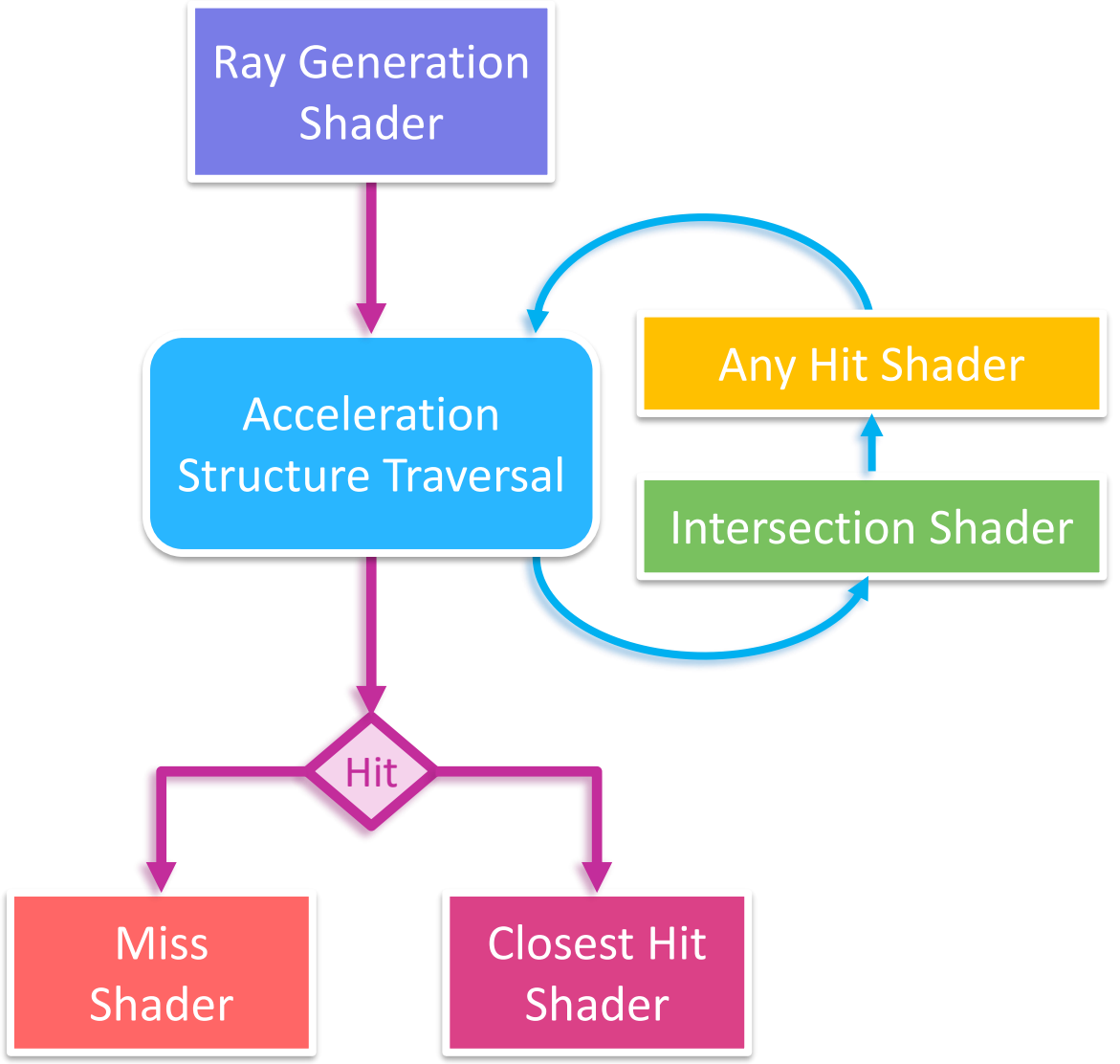
Ray Tracing Pipeline



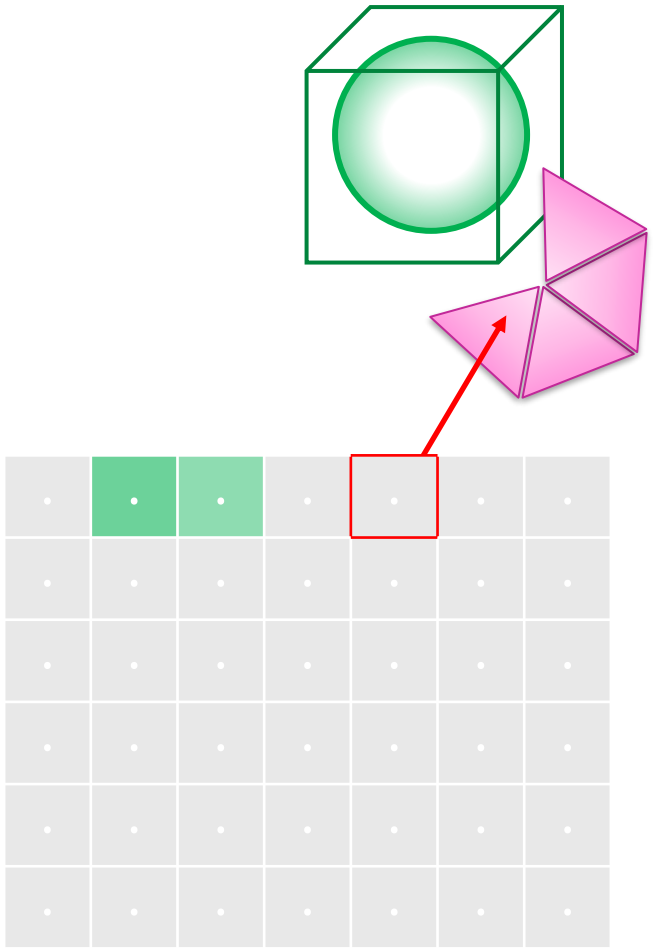


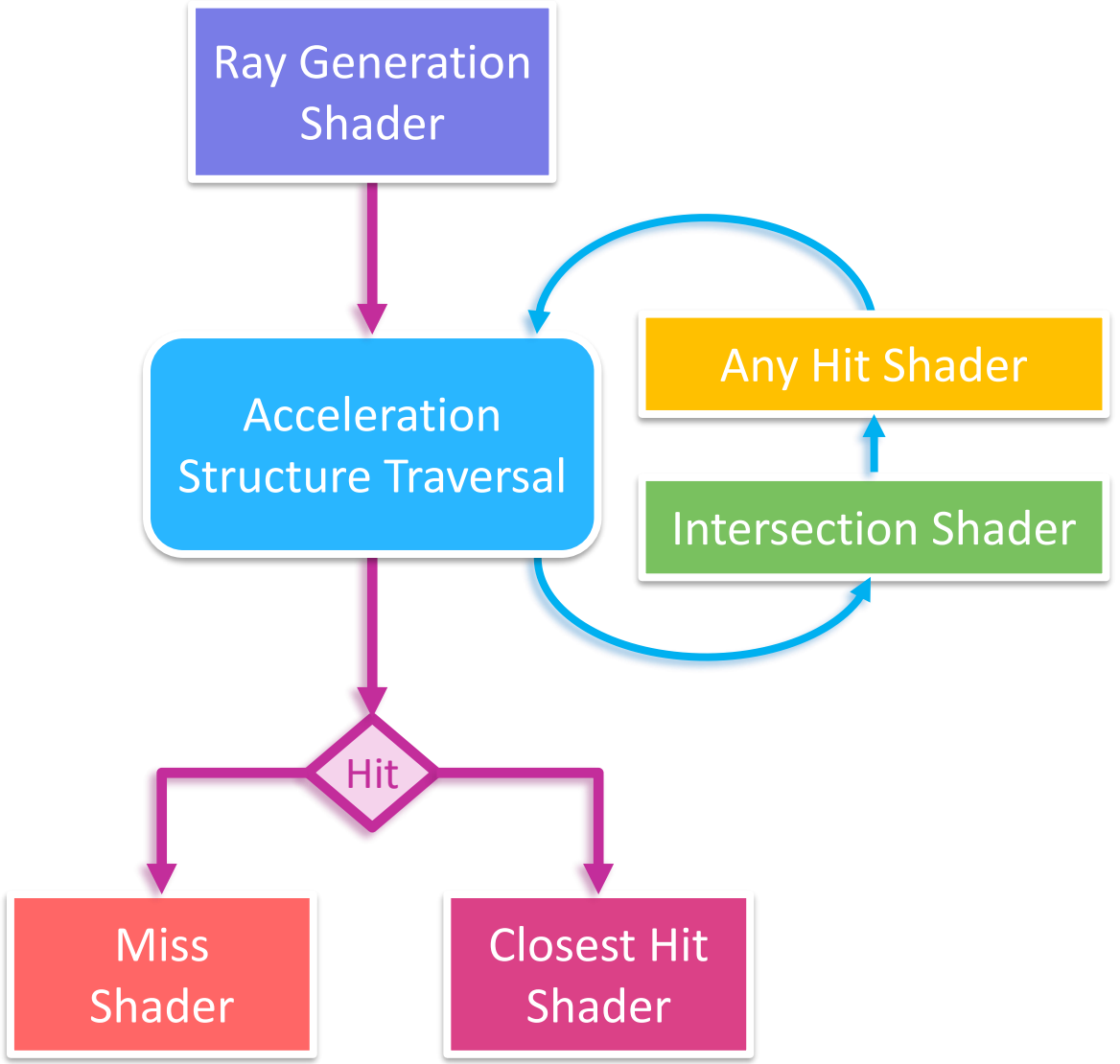
Ray Tracing Pipeline



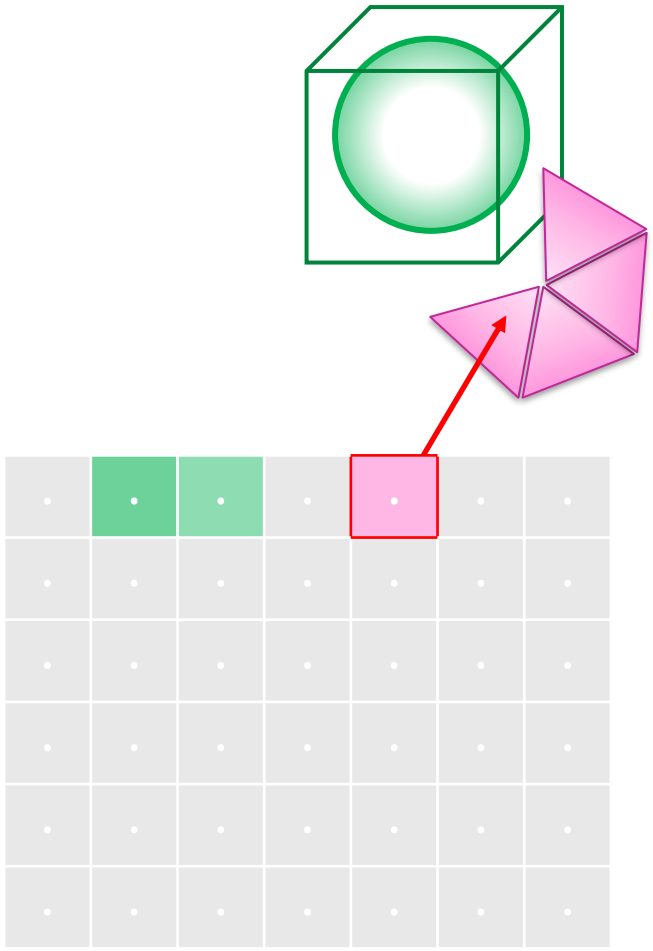


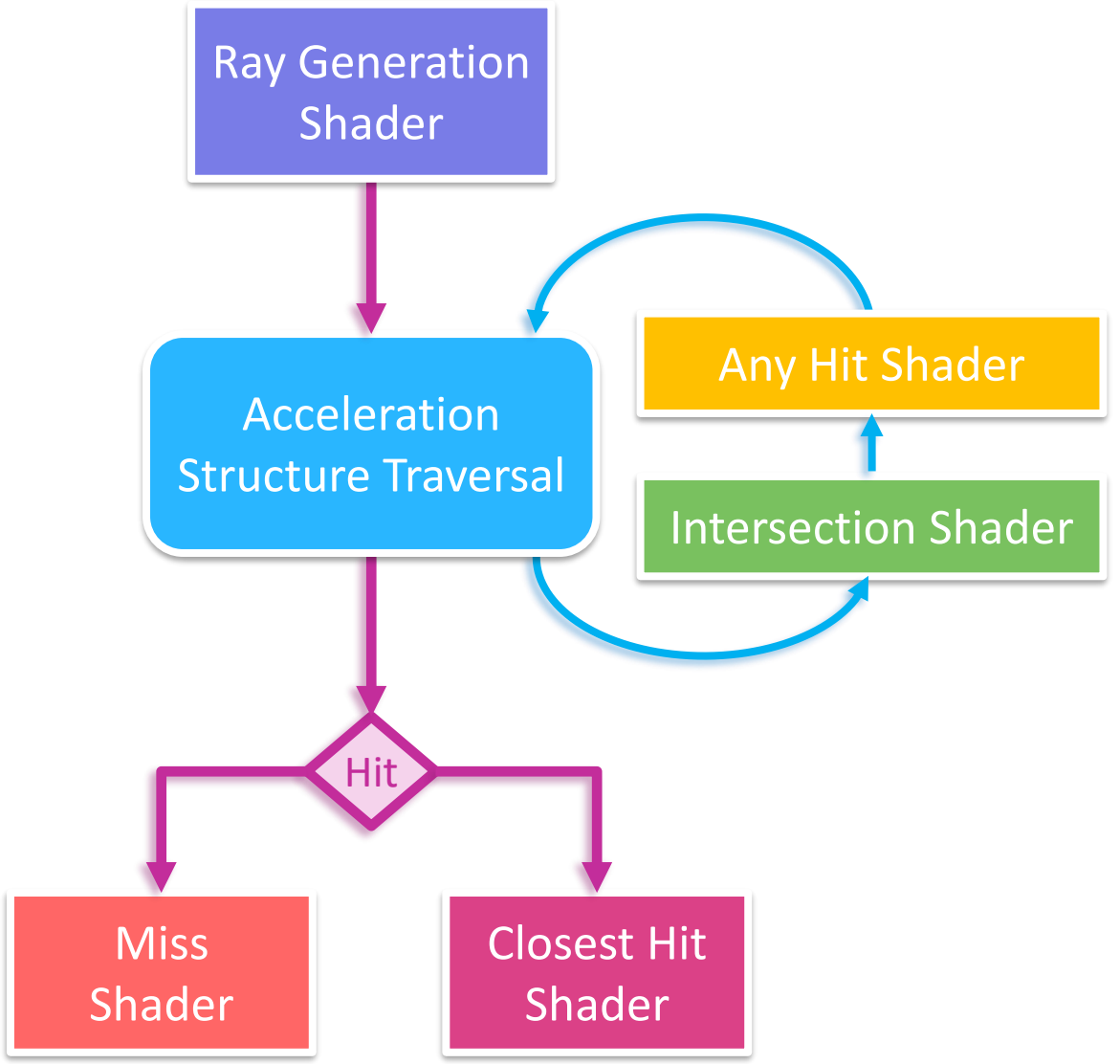
Ray Tracing Pipeline



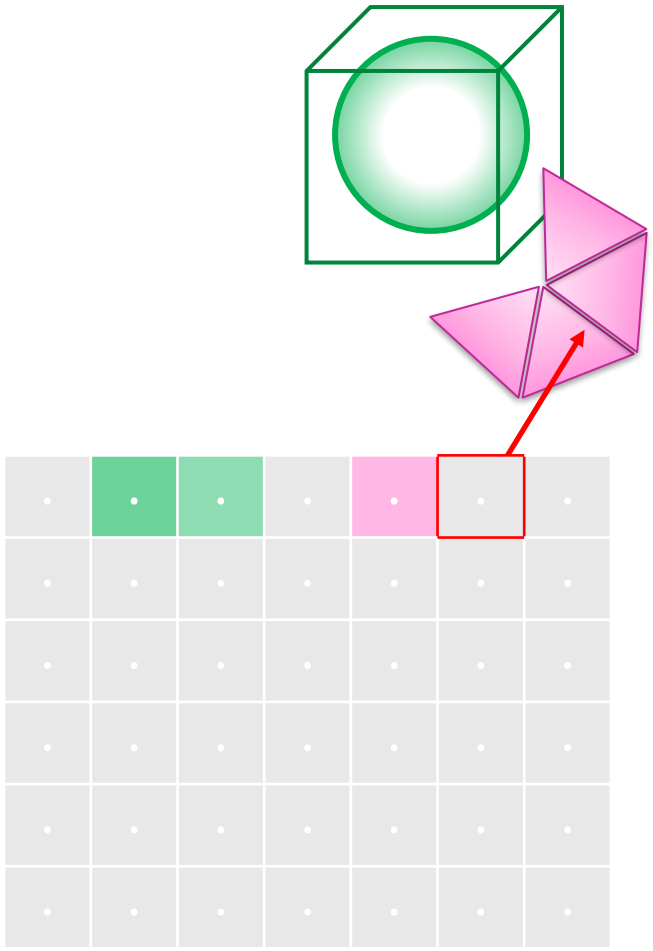


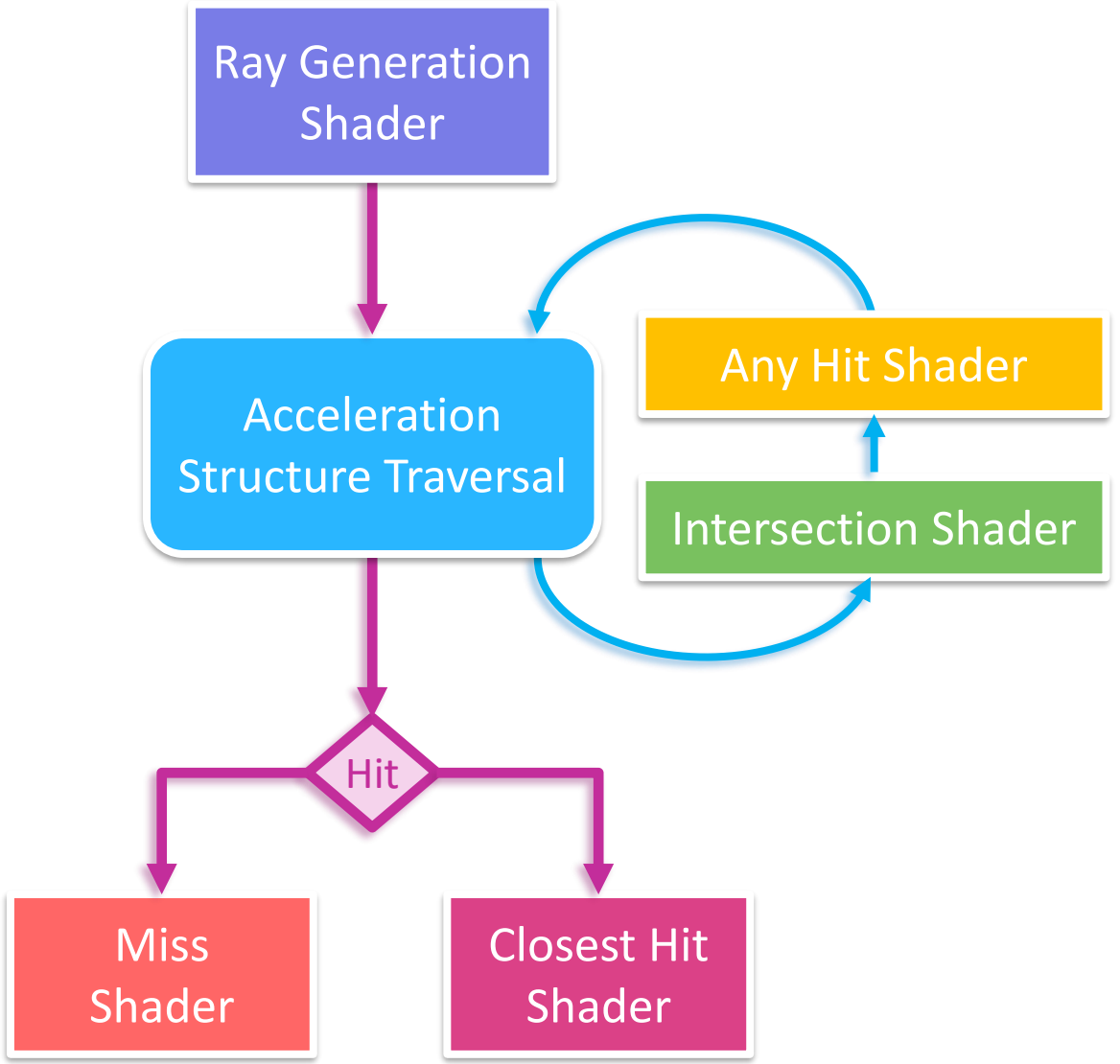
Ray Tracing Pipeline



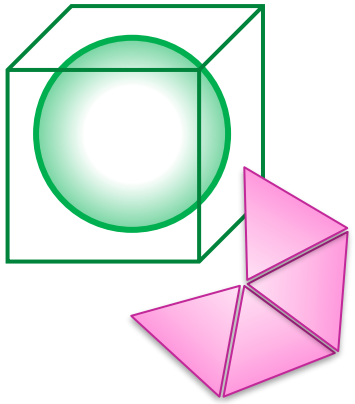


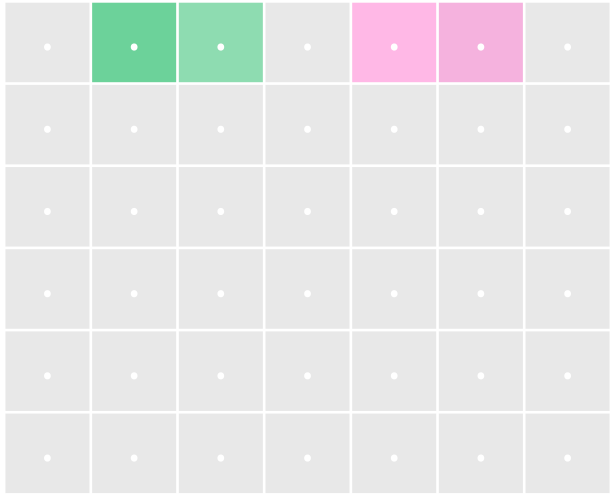
Ray Tracing Pipeline

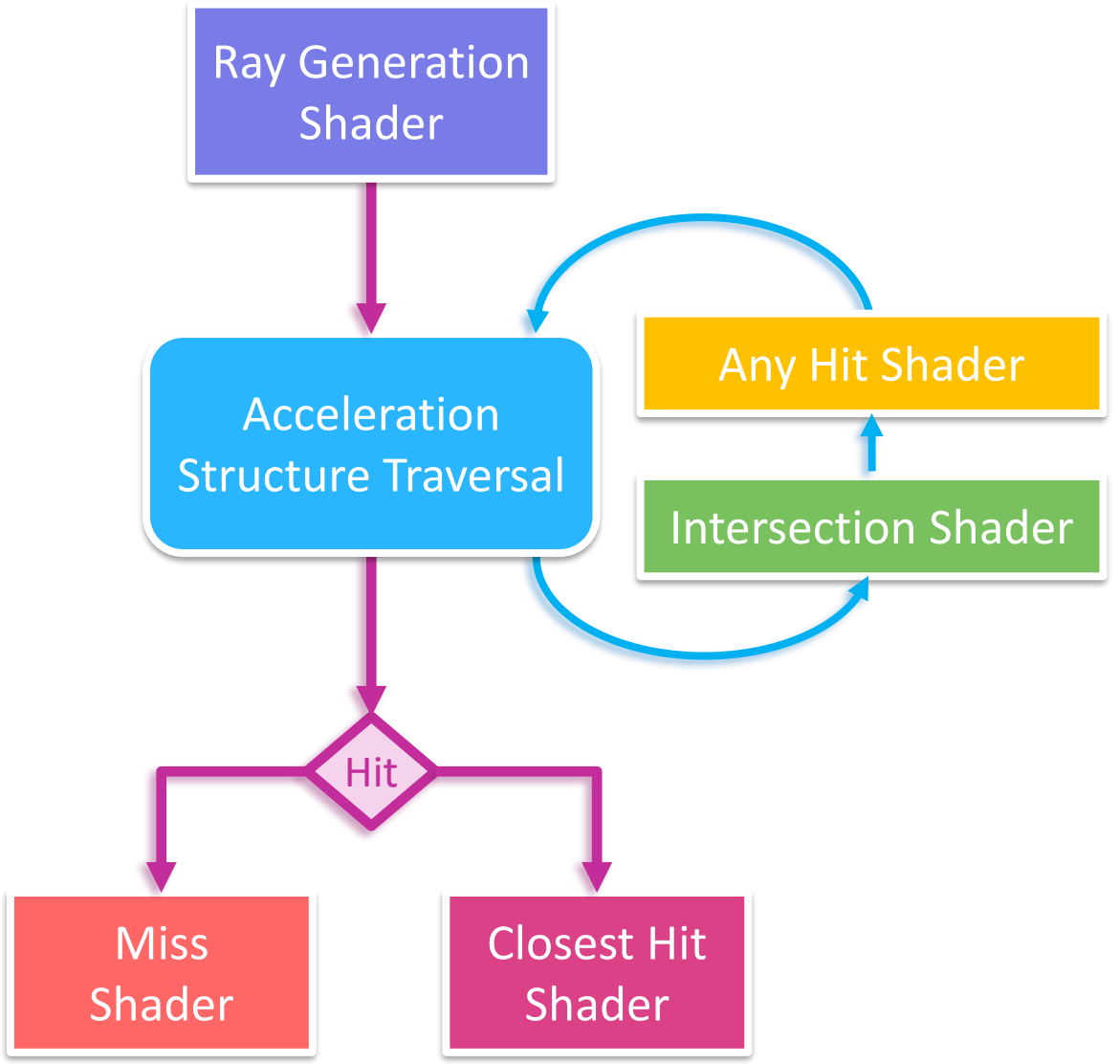




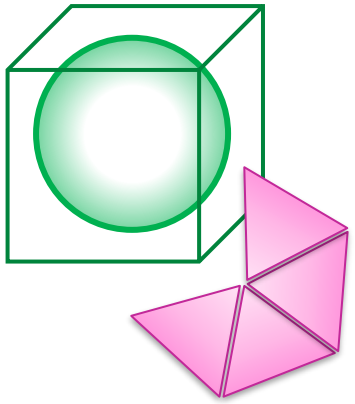
Ray Tracing Pipeline

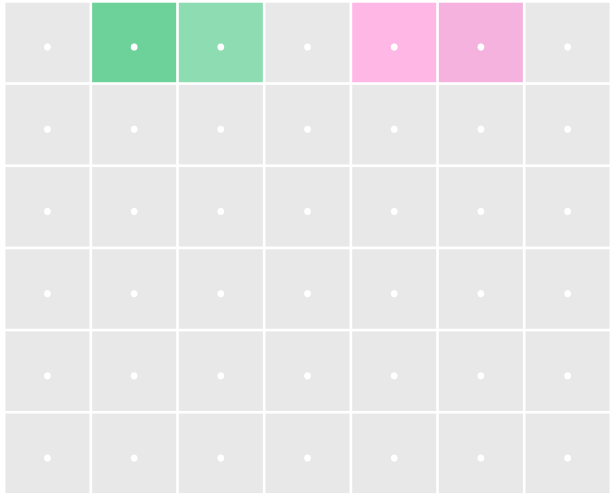


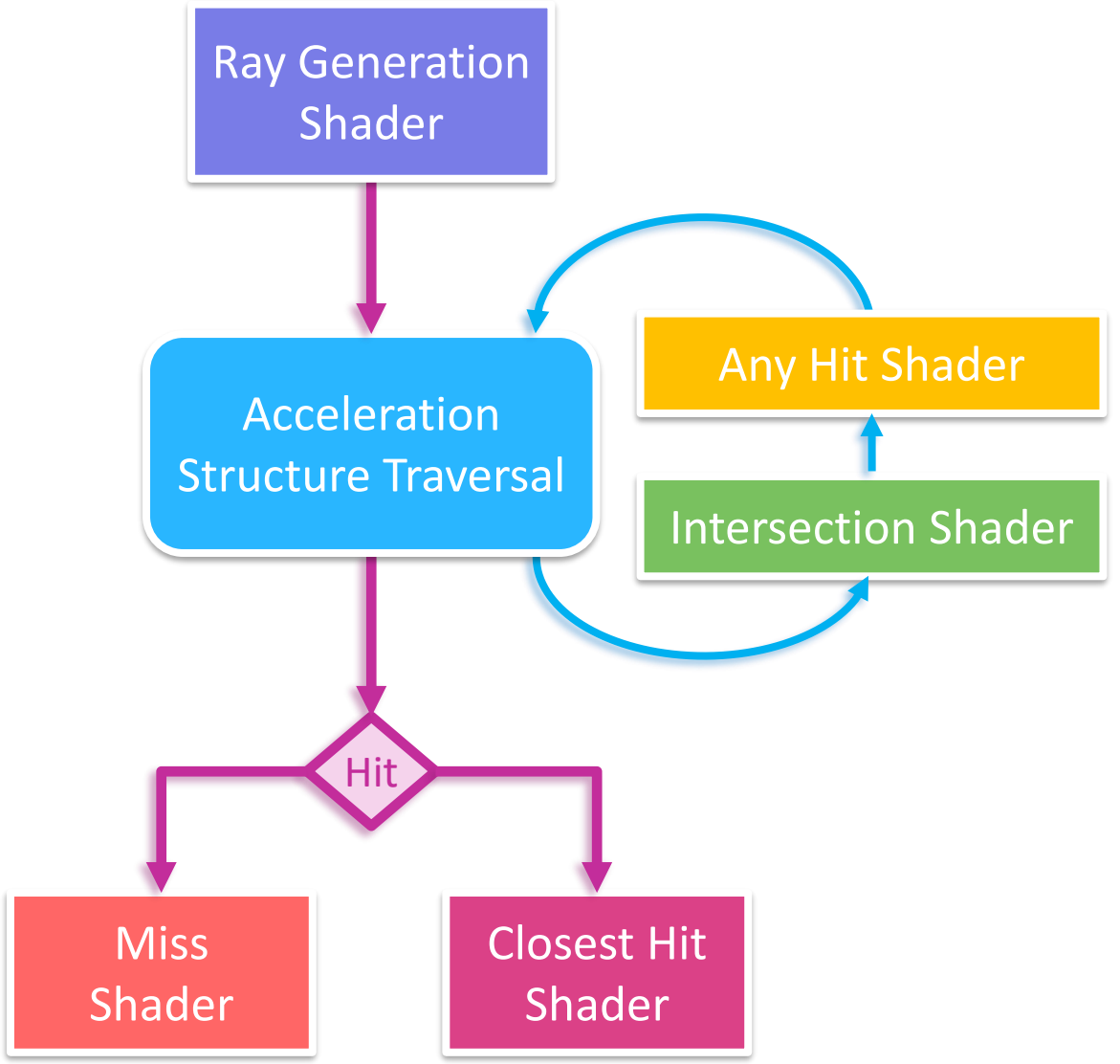




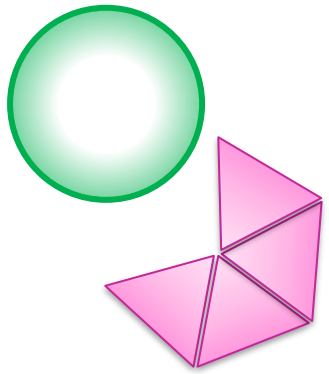
Ray Tracing Pipeline







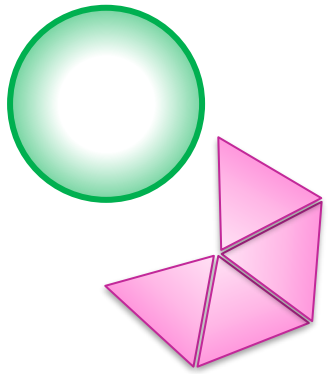
Ray Tracing Pipeline



Acceleration Structure Traversal

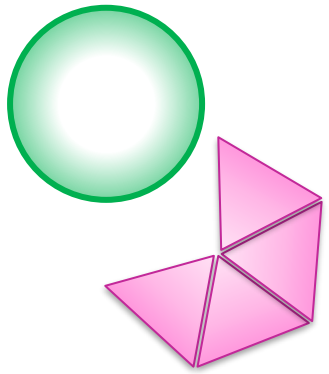
Acceleration Structures

Acceleration Structures



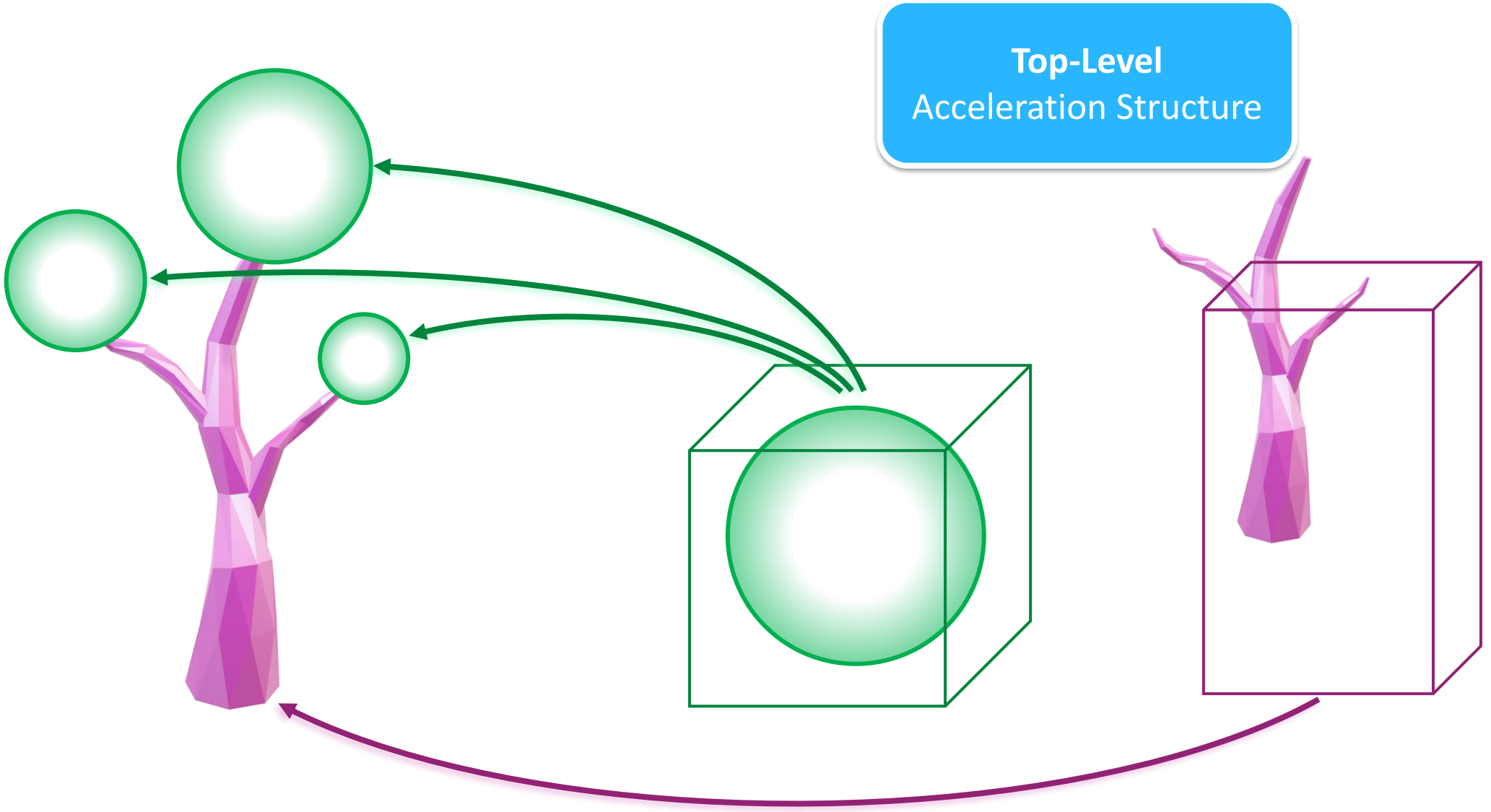
Acceleration Structure Traversal

Acceleration Structures

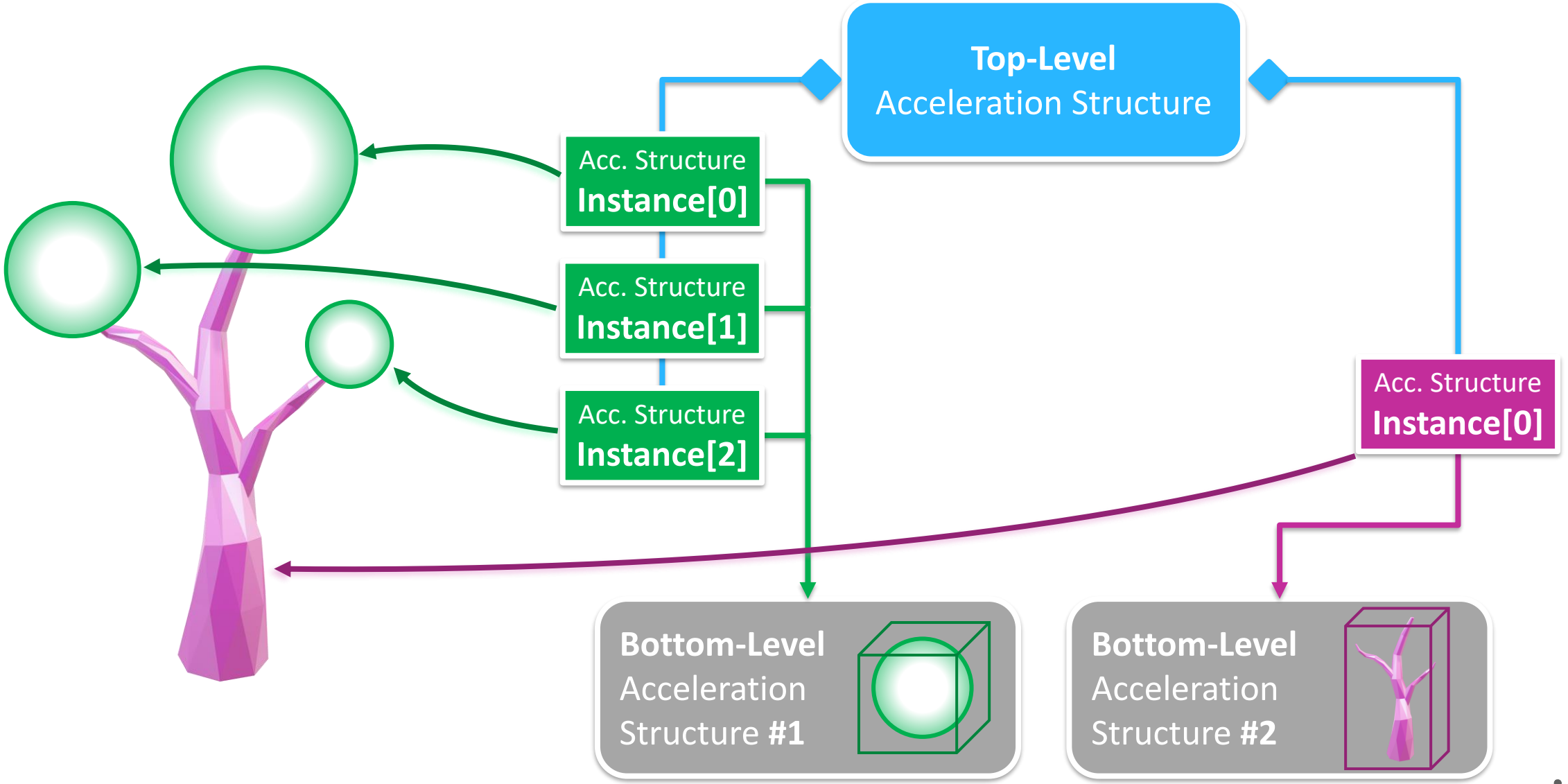


Top-Level
Acceleration Structure

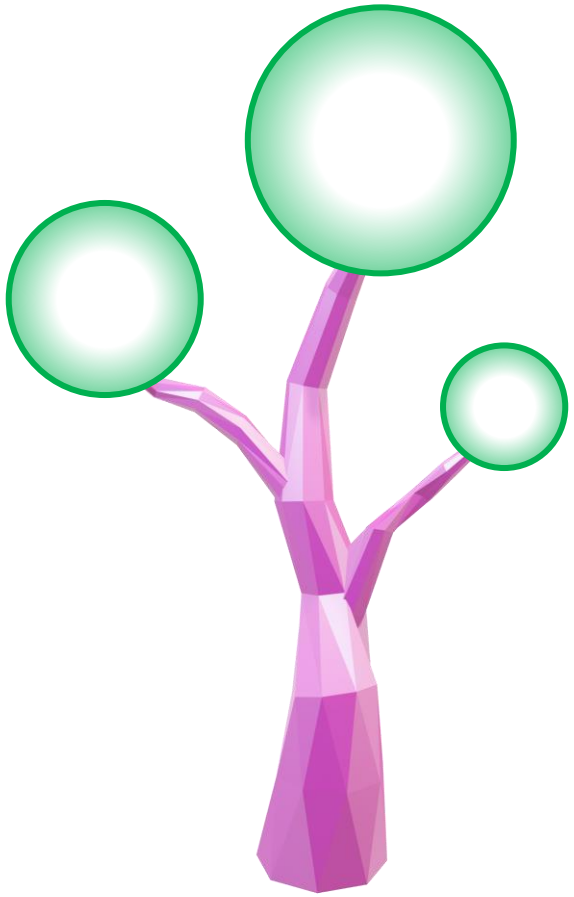
Acceleration Structures



Acceleration Structures



Acceleration Structures



Top-Level
Acceleration Structure

Acc. Structure
Instance[0]

Acc. Structure
Instance[1]

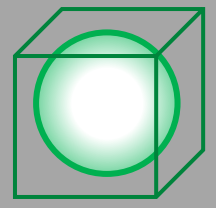
Acc. Structure
Instance[2]

Data per Instance

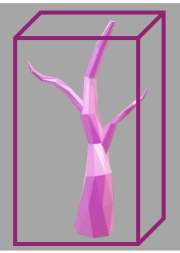
- Transformation Matrix
- Mask (include or exclude from tracing, e.g.: shadow pass)
- Flags (e.g.: winding order, back face culling on/off)
- Offset (for shader selection)
- Instance Custom Index (accessible in shaders)

Acc. Structure
Instance[0]

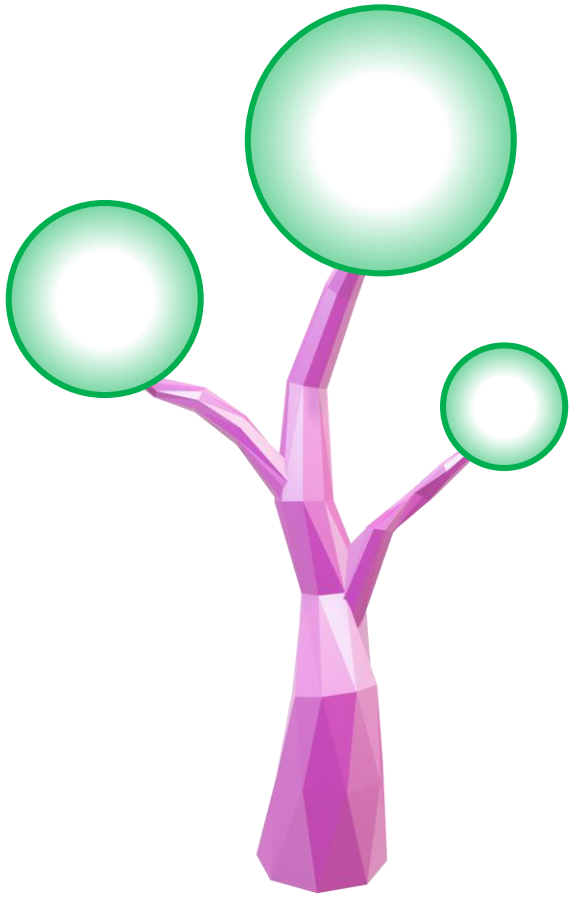
Bottom-Level
Acceleration
Structure **#1**



Bottom-Level
Acceleration
Structure **#2**



Acceleration Structures



Top-Level
Acceleration Structure

Acc. Structure
Instance[0]

Acc. Structure
Instance[1]

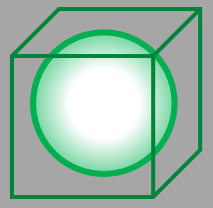
Acc. Structure
Instance[2]

Data per Instance

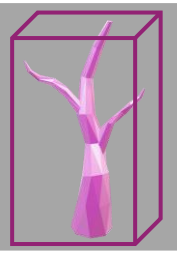
- Transformation Matrix
- Mask (include or exclude from tracing, e.g.: shadow pass)
- Flags (e.g.: winding order, back face culling on/off)
- Offset (for shader selection)
- Instance Custom Index (accessible in shaders)

Acc. Structure
Instance[0]

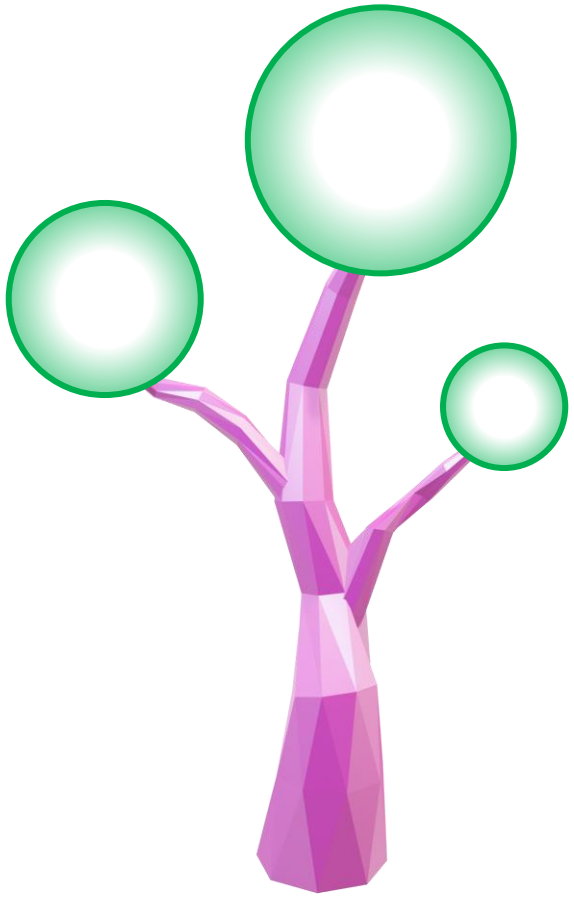
Bottom-Level
Acceleration
Structure **#1**



Bottom-Level
Acceleration
Structure **#2**



Acceleration Structures



Top-Level
Acceleration Structure

Acc. Structure
Instance[0]

Acc. Structure
Instance[1]

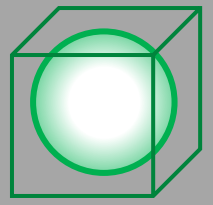
Acc. Structure
Instance[2]

Data per Instance

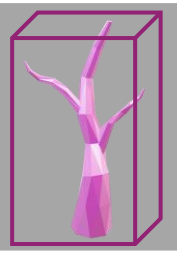
- Transformation Matrix
- Mask (include or exclude from tracing, e.g.: shadow pass)
- Flags (e.g.: winding order, back face culling on/off)
- Offset (for shader selection)
- Instance Custom Index (accessible in shaders)

Acc. Structure
Instance[0]

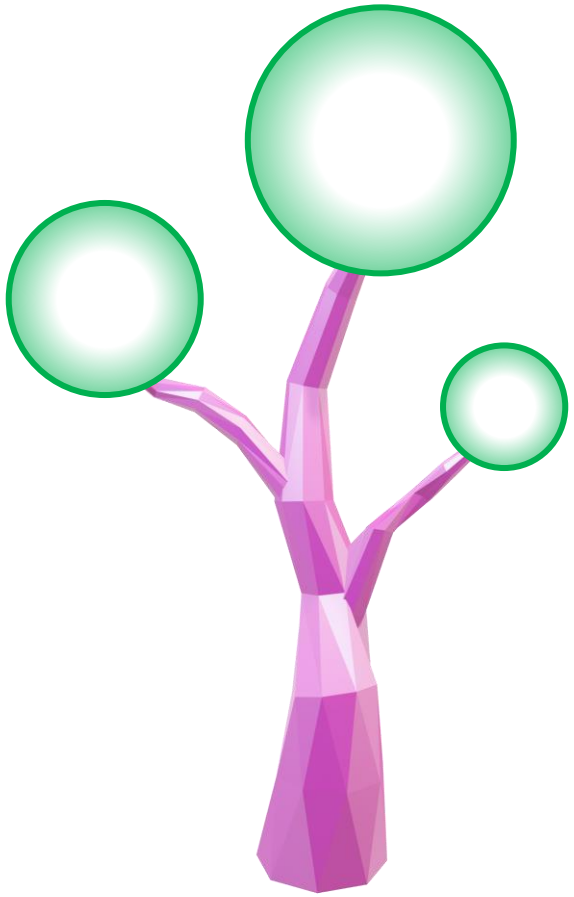
Bottom-Level
Acceleration
Structure **#1**



Bottom-Level
Acceleration
Structure **#2**



Acceleration Structures



Top-Level
Acceleration Structure

Acc. Structure
Instance[0]

Acc. Structure
Instance[1]

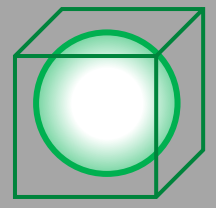
Acc. Structure
Instance[2]

Data per Instance

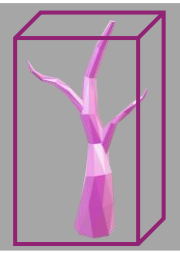
- Transformation Matrix
- Mask (include or exclude from tracing, e.g.: shadow pass)
- Flags (e.g.: winding order, back face culling on/off)
- Offset (for shader selection)
- Instance Custom Index (accessible in shaders)

Acc. Structure
Instance[0]

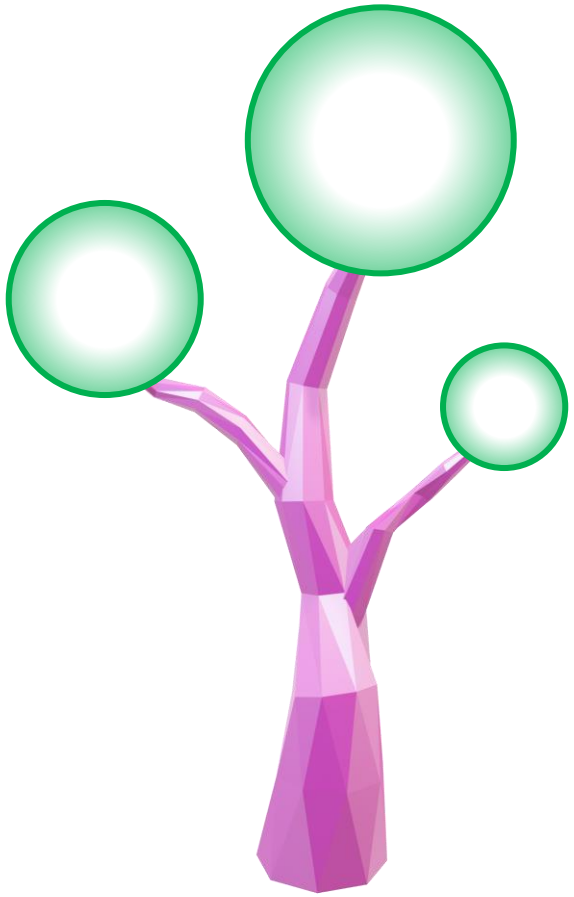
Bottom-Level
Acceleration
Structure **#1**



Bottom-Level
Acceleration
Structure **#2**



Acceleration Structures



Top-Level
Acceleration Structure

Acc. Structure
Instance[0]

Acc. Structure
Instance[1]

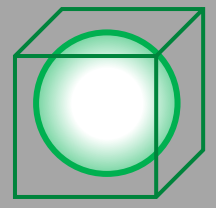
Acc. Structure
Instance[2]

Data per Instance

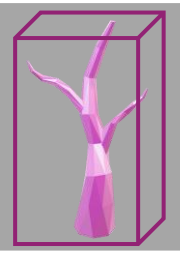
- Transformation Matrix
- Mask (include or exclude from tracing, e.g.: shadow pass)
- Flags (e.g.: winding order, back face culling on/off)
- Offset (for shader selection)
- Instance Custom Index (accessible in shaders)

Acc. Structure
Instance[0]

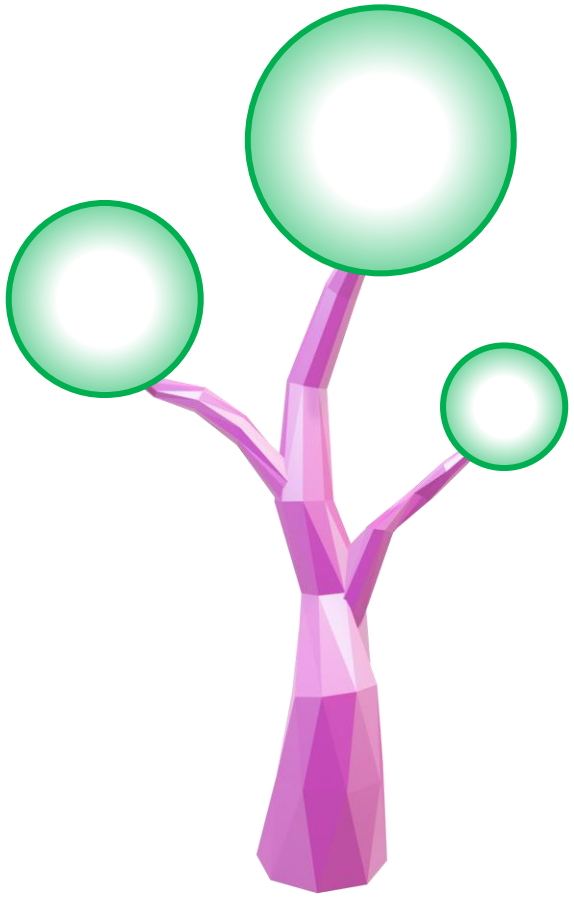
Bottom-Level
Acceleration
Structure **#1**



Bottom-Level
Acceleration
Structure **#2**



Acceleration Structures



Top-Level
Acceleration Structure

Acc. Structure
Instance[0]

Acc. Structure
Instance[1]

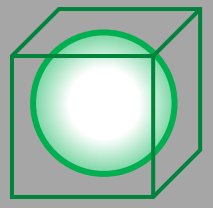
Acc. Structure
Instance[2]

Data per Instance

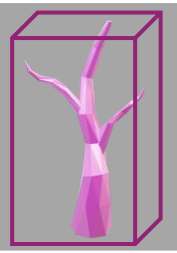
- Transformation Matrix
- Mask (include or exclude from tracing, e.g.: shadow pass)
- Flags (e.g.: winding order, back face culling on/off)
- Offset (for shader selection)
- Instance Custom Index (accessible in shaders)

Acc. Structure
Instance[0]

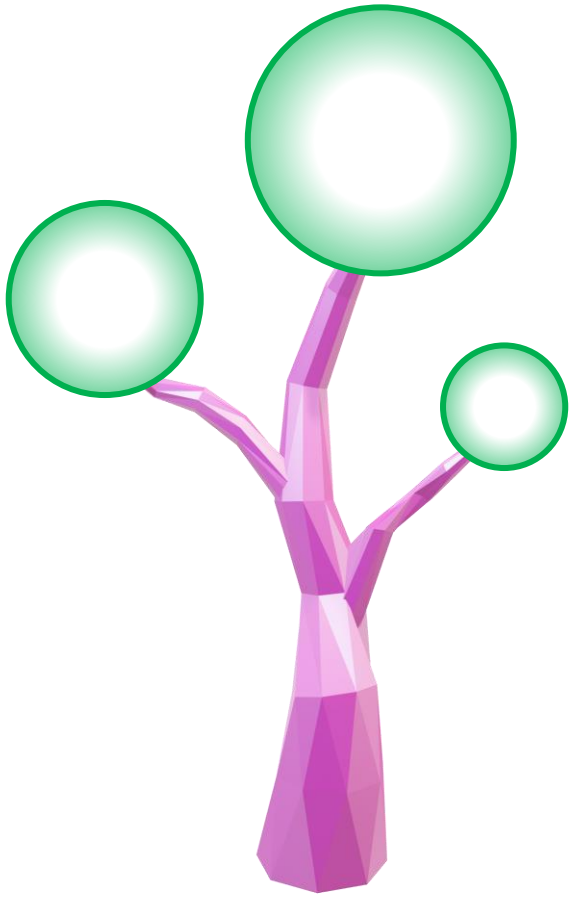
Bottom-Level
Acceleration
Structure **#1**

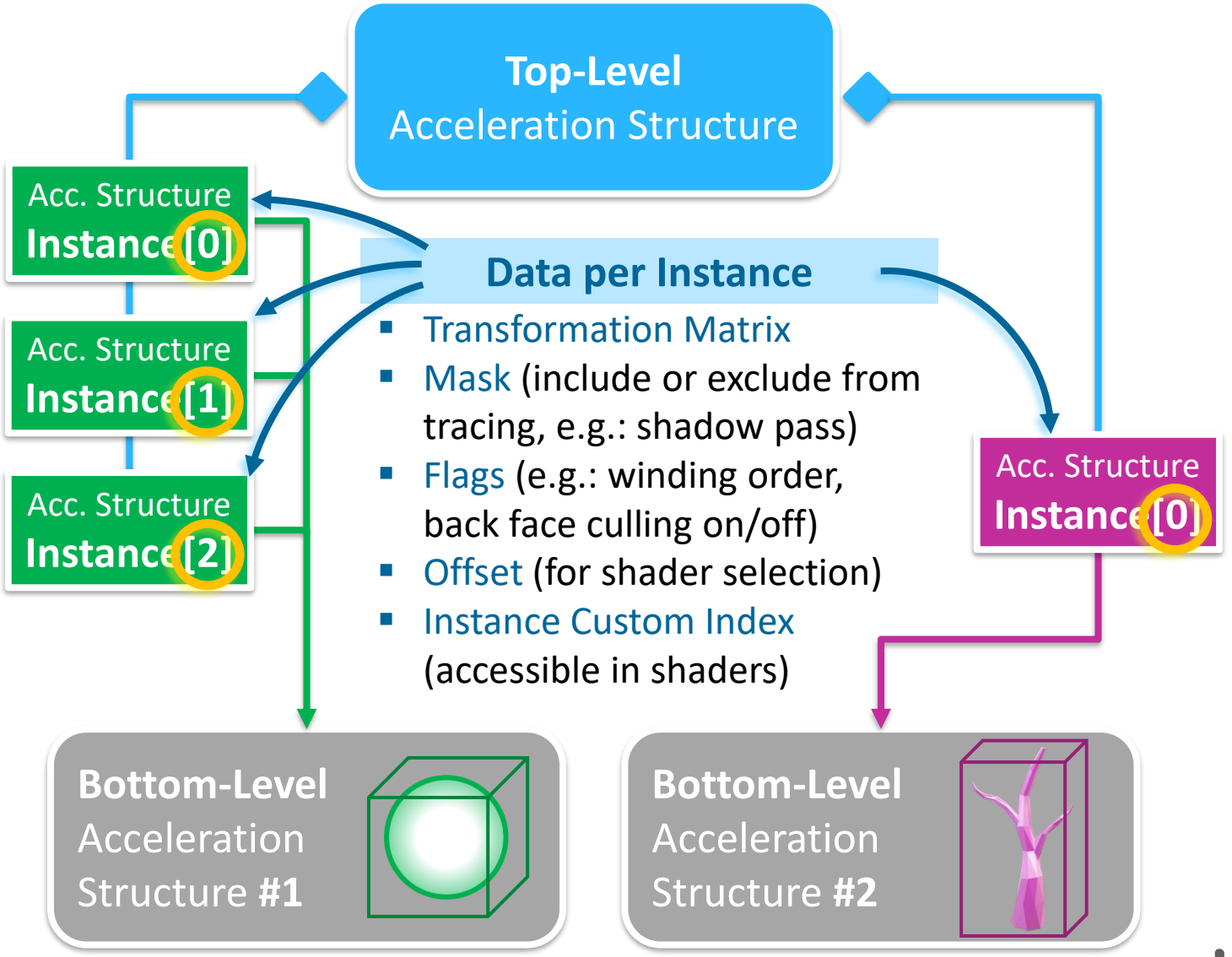


Bottom-Level
Acceleration
Structure **#2**



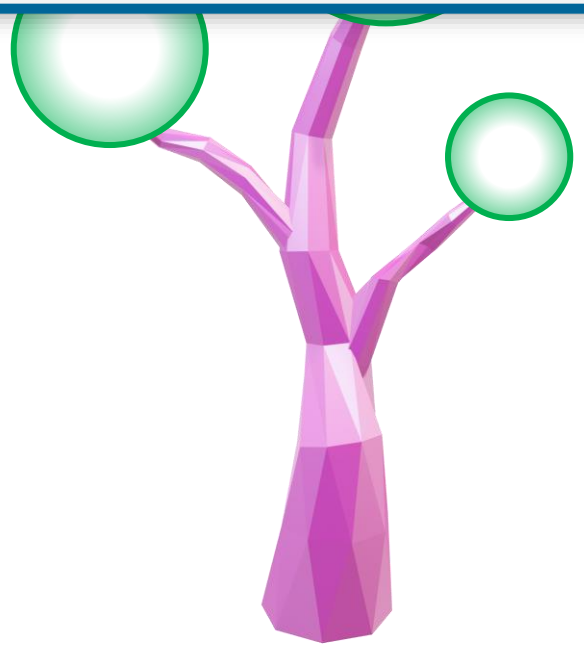
Acceleration Structures





Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR           transform;
    uint32_t                       instanceCustomIndex:24;
    uint32_t                       mask:8;
    uint32_t                       instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR     flags:8;
    uint64_t                       accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;
```



Top-Level
Acceleration Structure

Data per Instance

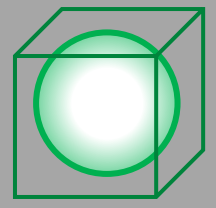
- Transformation Matrix
- Mask (include or exclude from tracing, e.g.: shadow pass)
- Flags (e.g.: winding order, back face culling on/off)
- Offset (for shader selection)
- Instance Custom Index (accessible in shaders)

Acc. Structure **Instance[0]**

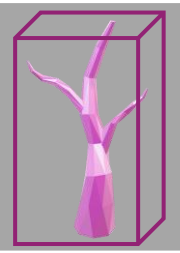
Acc. Structure **Instance[1]**

Acc. Structure **Instance[2]**

Bottom-Level Acceleration Structure **#1**

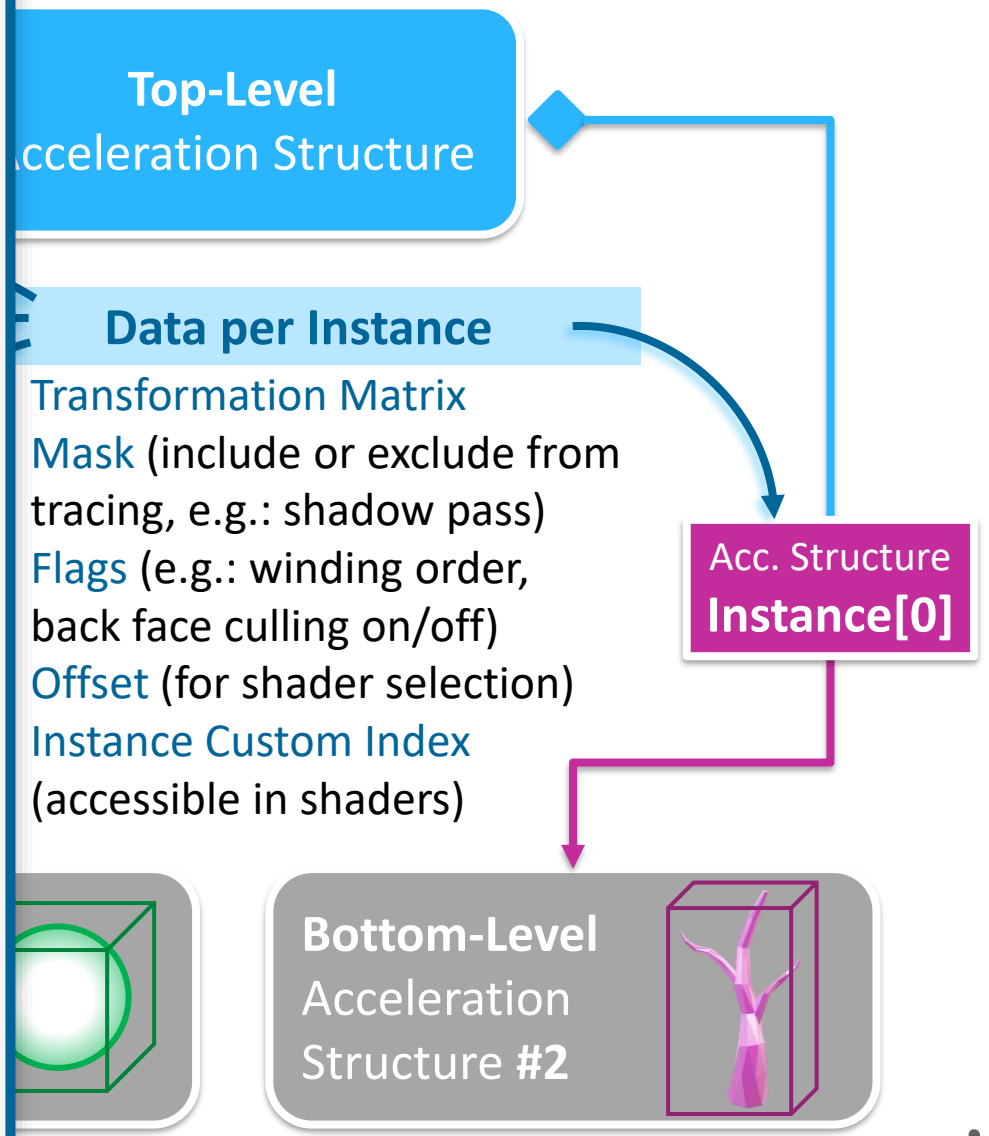


Bottom-Level Acceleration Structure **#2**



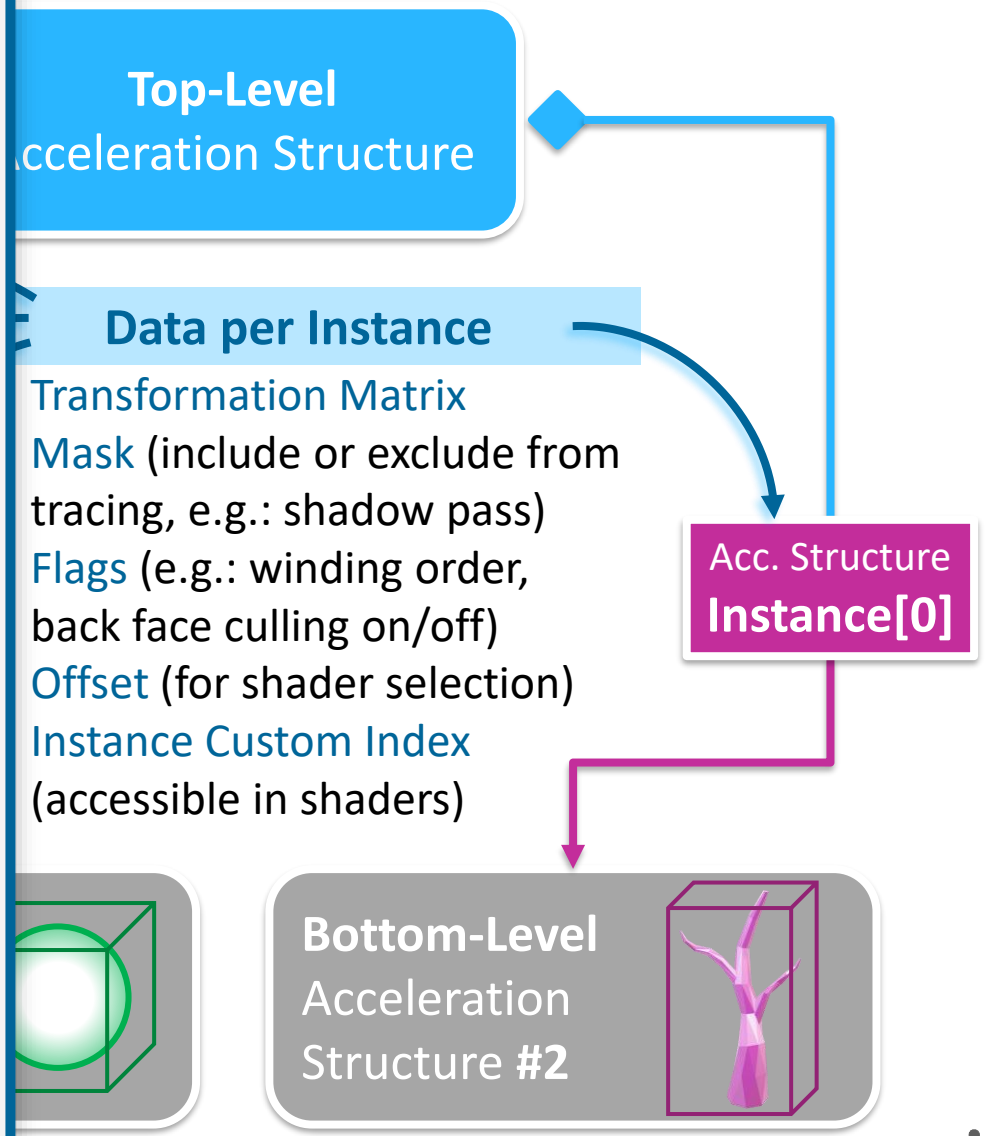
Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;


// Configure an instance:

uint64_t blasAddress = vkGetAccelerationStructureDeviceAddressKHR(...);

VkTransformMatrixKHR instanceTransform = {
    1.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 1.0f, 0.0f
};

VkAccelerationStructureInstanceKHR instance = {};
instance.transform = instanceTransform;
instance.instanceCustomIndex = 0;
instance.mask = 0xFF;
instance.instanceShaderBindingTableRecordOffset = 0;
instance.flags = VK_GEOMETRY_INSTANCE_TRIANGLE_FACING_CULL_DISABLE_BIT_KHR;
instance.accelerationStructureReference = blasAddress;
```



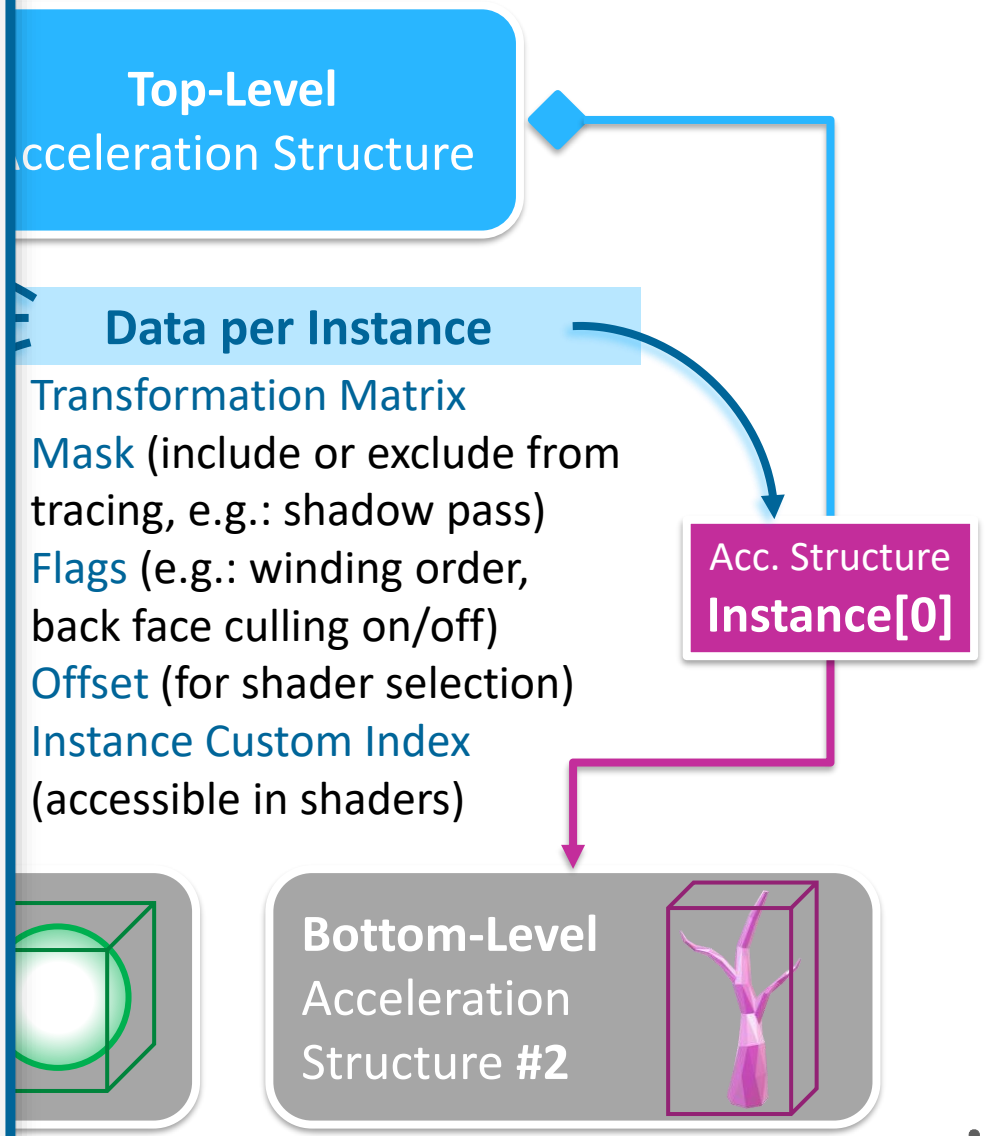
Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;


// Configure an instance:

uint64_t blasAddress = vkGetAccelerationStructureDeviceAddressKHR(...);

VkTransformMatrixKHR instanceTransform = {
    1.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 1.0f, 0.0f
};

VkAccelerationStructureInstanceKHR instance = {};
instance.transform = instanceTransform;
instance.instanceCustomIndex = 0;
instance.mask = 0xFF;
instance.instanceShaderBindingTableRecordOffset = 0;
instance.flags = VK_GEOMETRY_INSTANCE_TRIANGLE_FACING_CULL_DISABLE_BIT_KHR;
instance.accelerationStructureReference = blasAddress;
```



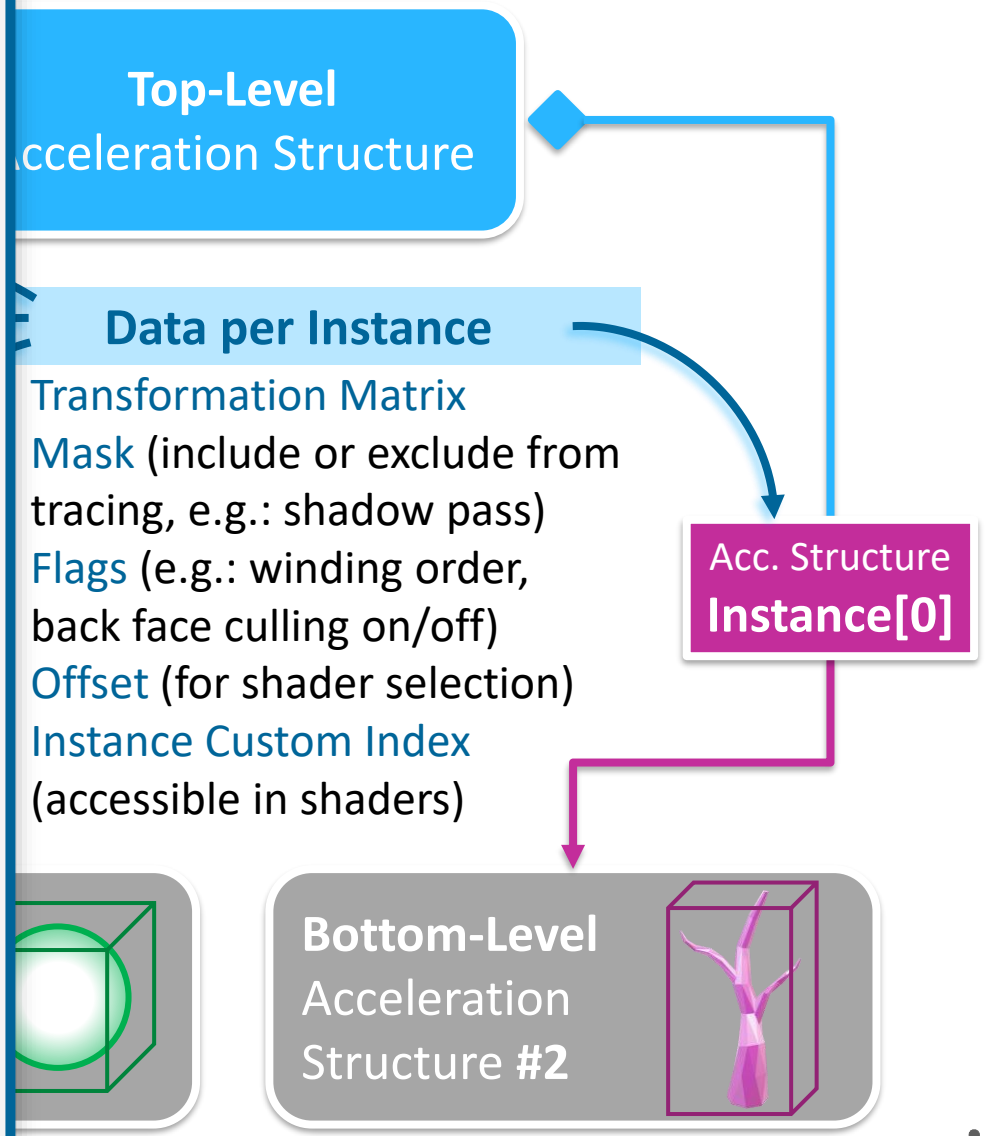
Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;


// Configure an instance:

uint64_t blasAddress = vkGetAccelerationStructureDeviceAddressKHR(...);

VkTransformMatrixKHR instanceTransform = {
    1.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 1.0f, 0.0f
};

VkAccelerationStructureInstanceKHR instance = {};
instance.transform = instanceTransform;
instance.instanceCustomIndex = 0;
instance.mask = 0xFF;
instance.instanceShaderBindingTableRecordOffset = 0;
instance.flags = VK_GEOMETRY_INSTANCE_TRIANGLE_FACING_CULL_DISABLE_BIT_KHR;
instance.accelerationStructureReference = blasAddress;
```



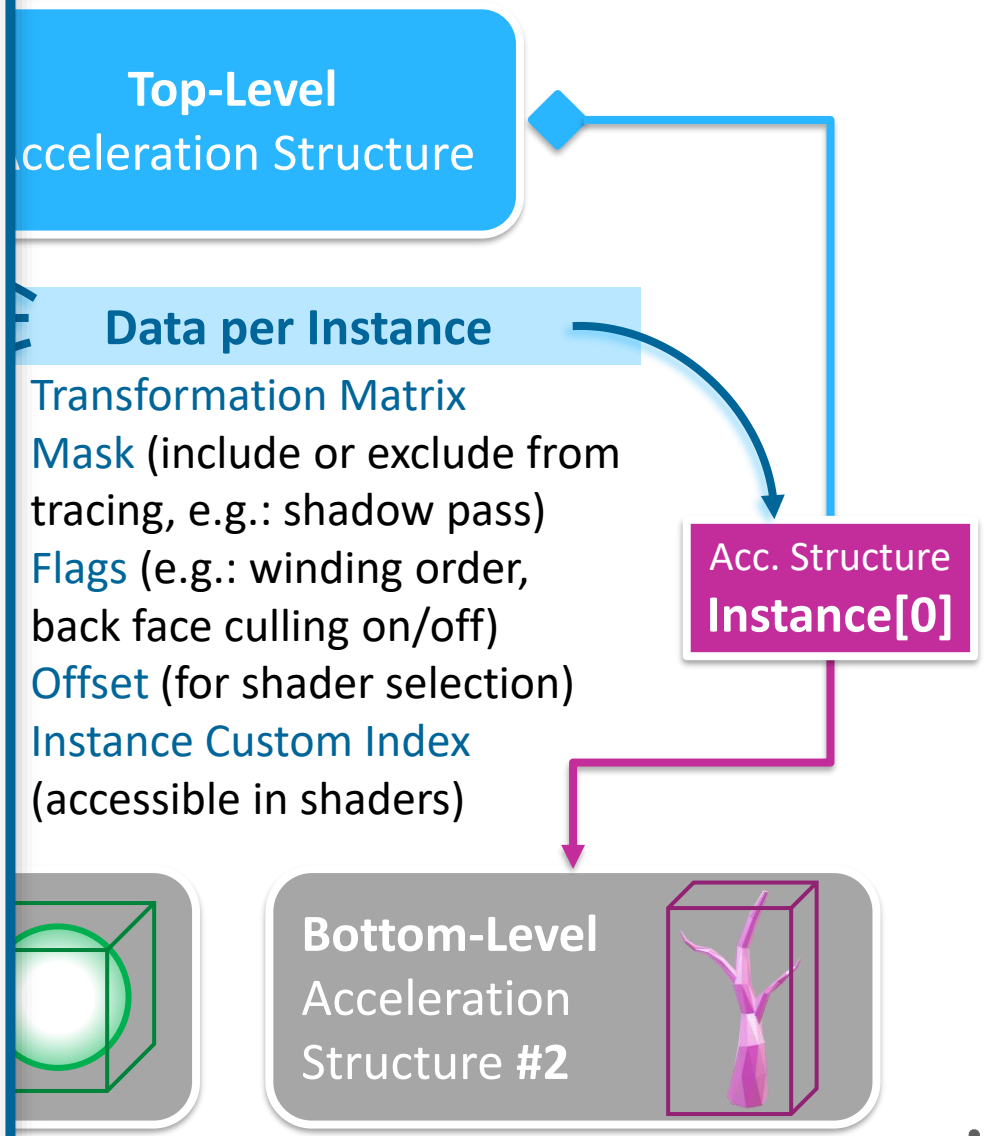
Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;


// Configure an instance:

uint64_t blasAddress = vkGetAccelerationStructureDeviceAddressKHR(...);

VkTransformMatrixKHR instanceTransform = {
    1.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 1.0f, 0.0f
};

VkAccelerationStructureInstanceKHR instance = {};
instance.transform = instanceTransform;
instance.instanceCustomIndex = 0;
instance.mask = 0xFF;
instance.instanceShaderBindingTableRecordOffset = 0;
instance.flags = VK_GEOMETRY_INSTANCE_TRIANGLE_FACING_CULL_DISABLE_BIT_KHR;
instance.accelerationStructureReference = blasAddress;
```



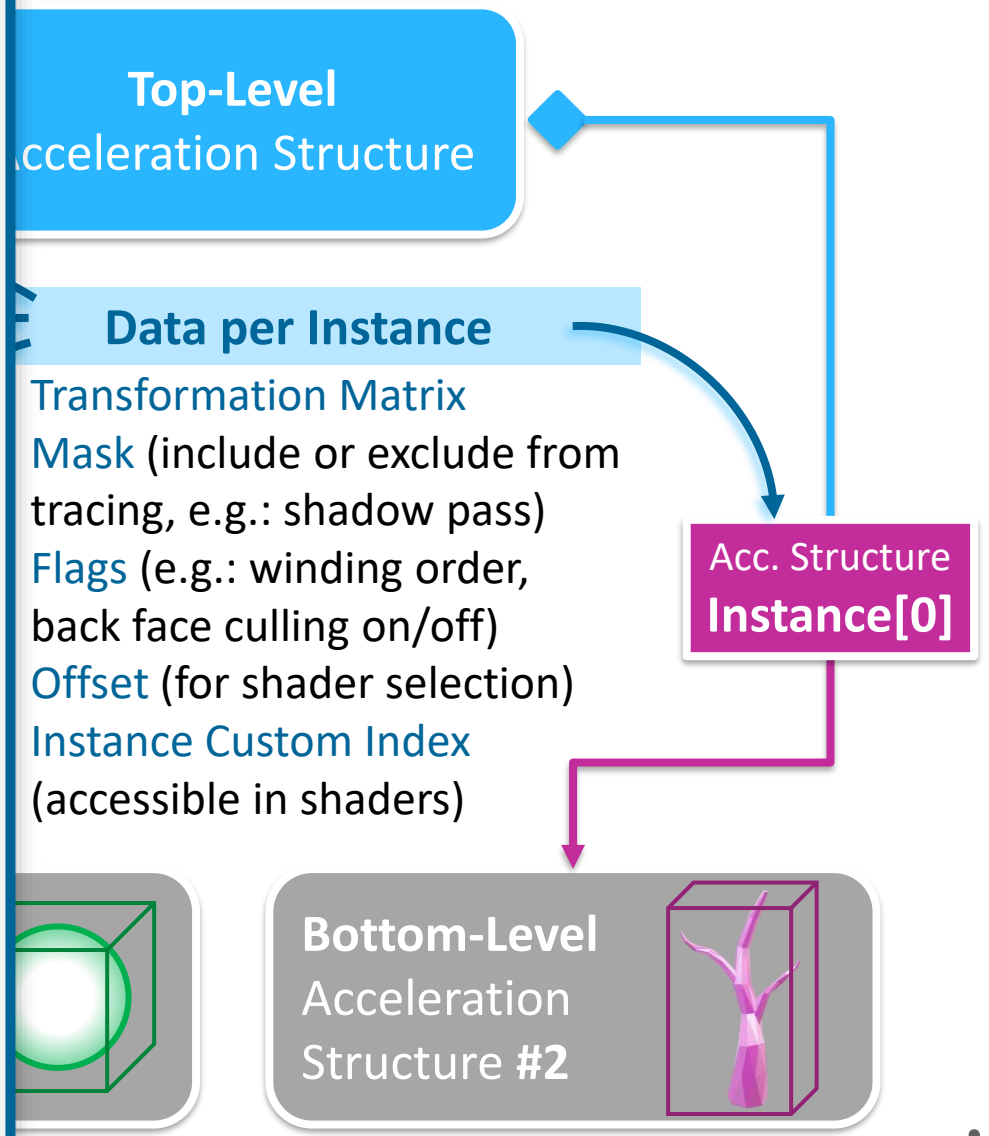
Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;


// Configure an instance:

uint64_t blasAddress = vkGetAccelerationStructureDeviceAddressKHR(...);

VkTransformMatrixKHR instanceTransform = {
    1.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 1.0f, 0.0f
};

VkAccelerationStructureInstanceKHR instance = {};
instance.transform = instanceTransform;
instance.instanceCustomIndex = 0;
instance.mask = 0xFF;
instance.instanceShaderBindingTableRecordOffset = 0;
instance.flags = VK_GEOMETRY_INSTANCE_TRIANGLE_FACING_CULL_DISABLE_BIT_KHR;
instance.accelerationStructureReference = blasAddress;
```



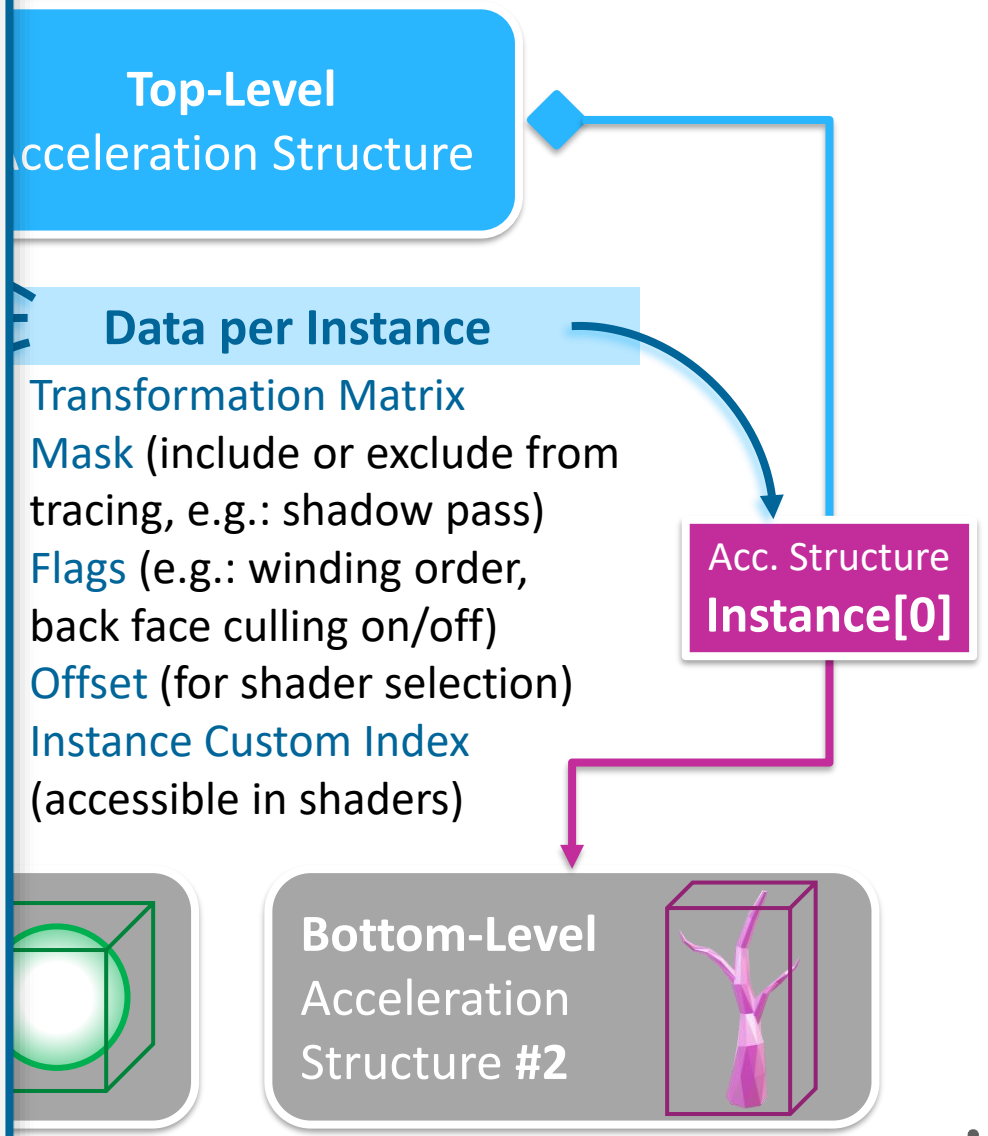
Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;


// Configure an instance:

uint64_t blasAddress = vkGetAccelerationStructureDeviceAddressKHR(...);

VkTransformMatrixKHR instanceTransform = {
    1.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 1.0f, 0.0f
};

VkAccelerationStructureInstanceKHR instance = {};
instance.transform = instanceTransform;
instance.instanceCustomIndex = 0;
instance.mask = 0xFF;
instance.instanceShaderBindingTableRecordOffset = 0;
instance.flags = VK_GEOMETRY_INSTANCE_TRIANGLE_FACING_CULL_DISABLE_BIT_KHR;
instance.accelerationStructureReference = blasAddress;
```



Acceleration Structures

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;


// Configure an instance:

uint64_t blasAddress = vkGetAccelerationStructureDeviceAddressKHR(...);

VkTransformMatrixKHR instanceTransform = {
    1.0f, 0.0f, 0.0f, 0.0f,
    0.0f, 1.0f, 0.0f, 0.0f,
    0.0f, 0.0f, 1.0f, 0.0f
};

VkAccelerationStructureInstanceKHR instance = {};
instance.transform = instanceTransform;
instance.instanceCustomIndex = 0;
instance.mask = 0xFF;
instance.instanceShaderBindingTableRecordOffset = 0;
instance.flags = VK_GEOMETRY_INSTANCE_TRIANGLE_FACING_CULL_DISABLE_BIT_KHR;
instance.accelerationStructureReference = blasAddress;
```

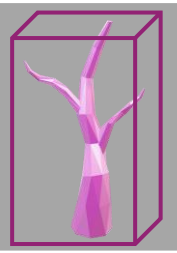


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

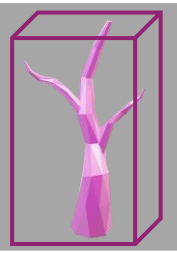


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

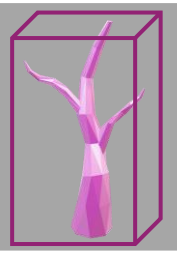


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

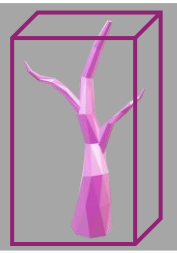


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

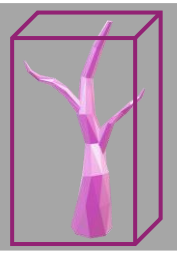


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

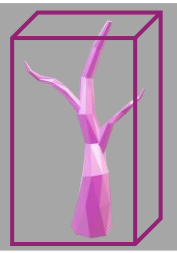


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

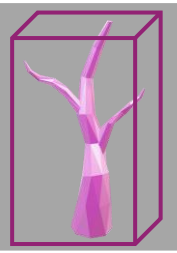


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

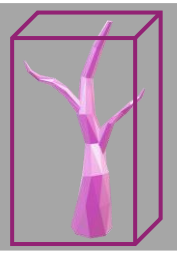


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

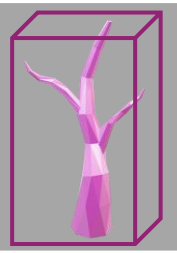


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

Bottom-Level Acceleration Structure **#2**

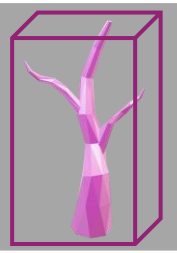


Acceleration Structures

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;
```

```
struct MyVertexData {
  float position[3];
  float normal[3];
  float texCoord[2];
};
```

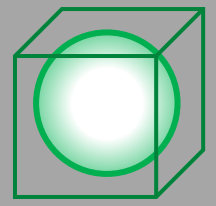
Bottom-Level Acceleration Structure **#2**



Acceleration Structures

```
VkAccelerationStructureGeometryKHR aabbGeometry = {};
aabbGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
aabbGeometry.geometryType = VK_GEOMETRY_TYPE_AABBS_KHR;
aabbGeometry.geometry.aabbs.sType =
VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_AABBS_DATA_KHR;
aabbGeometry.geometry.aabbs.data = aabbsBufferDeviceAddress;
aabbGeometry.geometry.aabbs.stride = sizeof(VkAabbPositionsKHR);
```

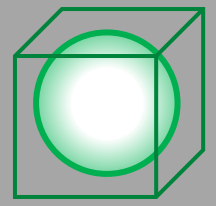
Bottom-Level Acceleration Structure **#1**



Acceleration Structures

```
VkAccelerationStructureGeometryKHR aabbGeometry = {};
aabbGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
aabbGeometry.geometryType = VK_GEOMETRY_TYPE_AABBS_KHR;
aabbGeometry.geometry.aabbs.sType =
VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_AABBS_DATA_KHR;
aabbGeometry.geometry.aabbs.data = aabbsBufferDeviceAddress;
aabbGeometry.geometry.aabbs.stride = sizeof(VkAabbPositionsKHR);
```

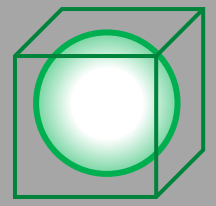
Bottom-Level Acceleration Structure **#1**



Acceleration Structures

```
VkAccelerationStructureGeometryKHR aabbGeometry = {};
aabbGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
aabbGeometry.geometryType = VK_GEOMETRY_TYPE_AABBS_KHR;
aabbGeometry.geometry.aabbs.sType =
VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_AABBS_DATA_KHR;
aabbGeometry.geometry.aabbs.data = aabbsBufferDeviceAddress;
aabbGeometry.geometry.aabbs.stride = sizeof(VkAabbPositionsKHR);
```

Bottom-Level Acceleration Structure **#1**

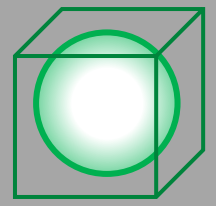


Acceleration Structures

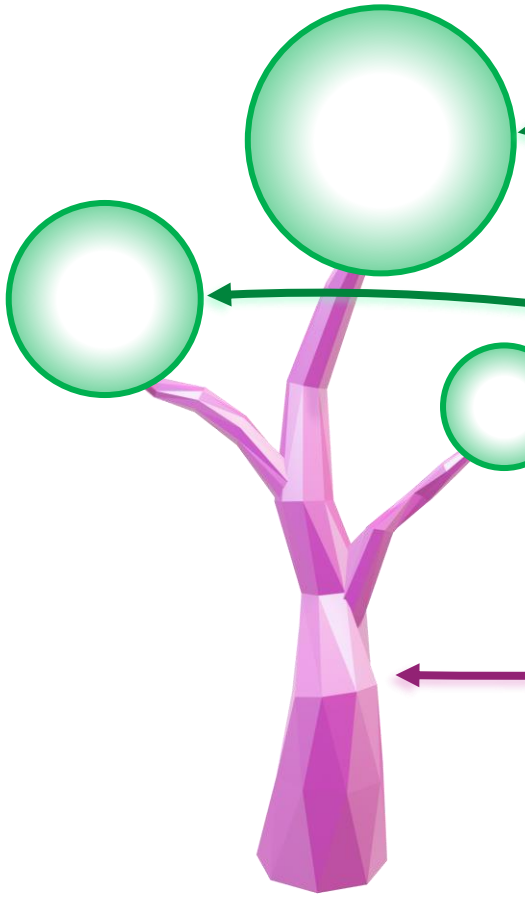
```
VkAccelerationStructureGeometryKHR aabbGeometry = {};
aabbGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
aabbGeometry.geometryType = VK_GEOMETRY_TYPE_AABBS_KHR;
aabbGeometry.geometry.aabbs.sType =
VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_AABBS_DATA_KHR;
aabbGeometry.geometry.aabbs.data = aabbsBufferDeviceAddress;
aabbGeometry.geometry.aabbs.stride = sizeof(VkAabbPositionsKHR);
```

```
typedef struct VkAabbPositionsKHR {
    float minX;
    float minY;
    float minZ;
    float maxX;
    float maxY;
    float maxZ;
} VkAabbPositionsKHR;
```

Bottom-Level Acceleration Structure **#1**



Acceleration Structures



Top-Level Acceleration Structure

`vkCmdBuildAccelerationStructuresKHR`

Acc. Structure **Instance[0]**

Acc. Structure **Instance[1]**

Acc. Structure **Instance[2]**

Acc. Structure **Instance[0]**

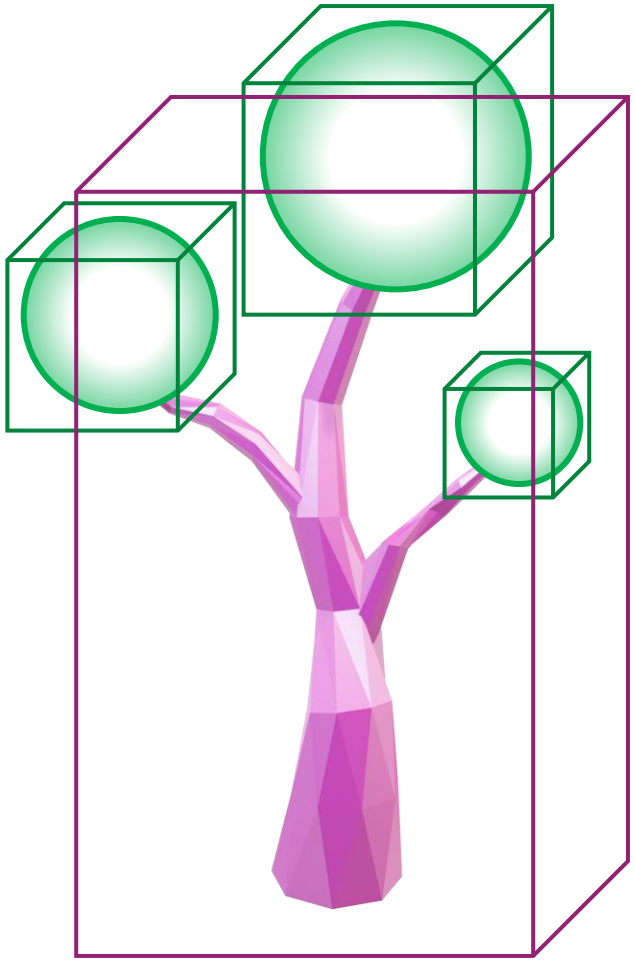
Bottom-Level Acceleration Structure **#1**

`vkCmdBuildAccelerationStructuresKHR`

Bottom-Level Acceleration Structure **#2**

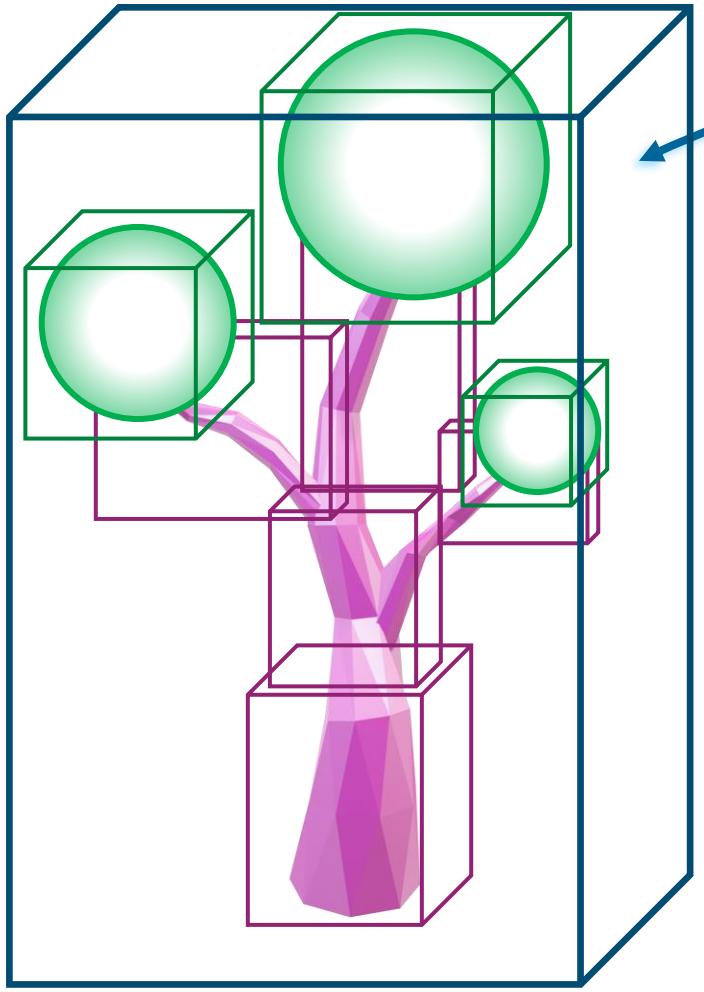
`vkCmdBuildAccelerationStructuresKHR`

Bounding Volume Hierarchy Traversal



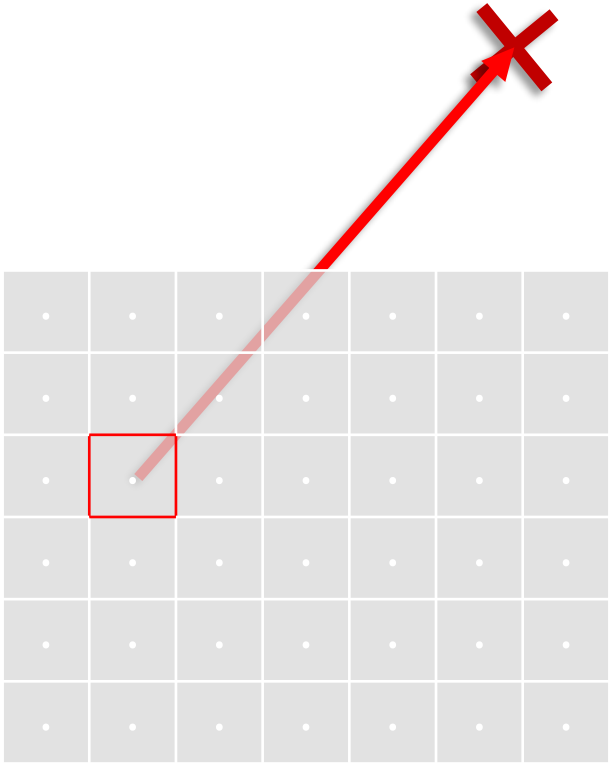
Top-Level
Acceleration Structure

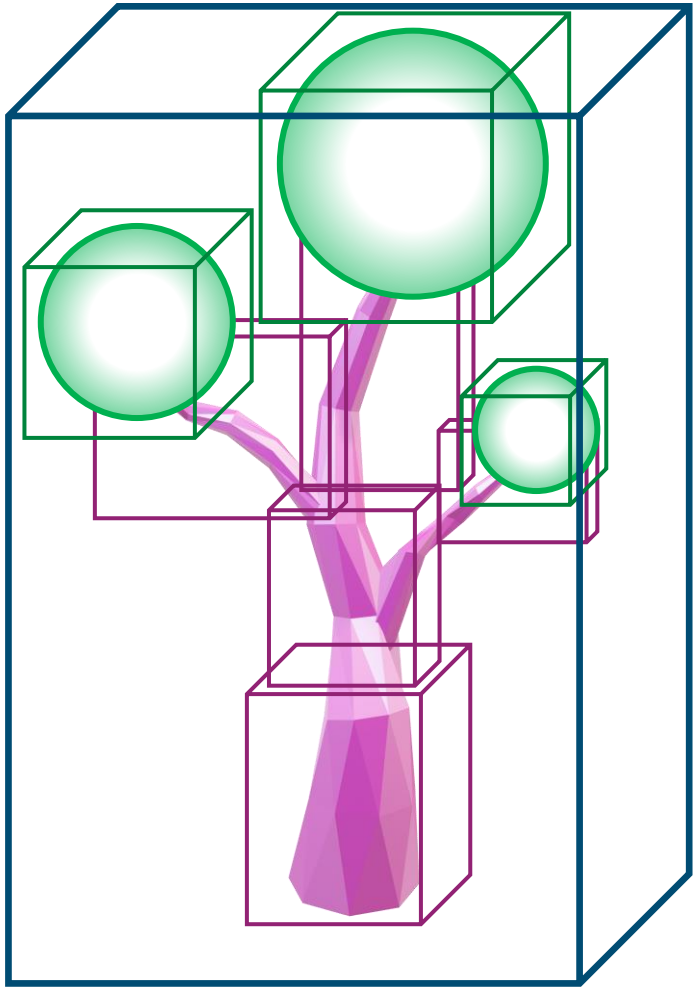
Bounding Volume Hierarchy Traversal



Top-Level
Acceleration Structure

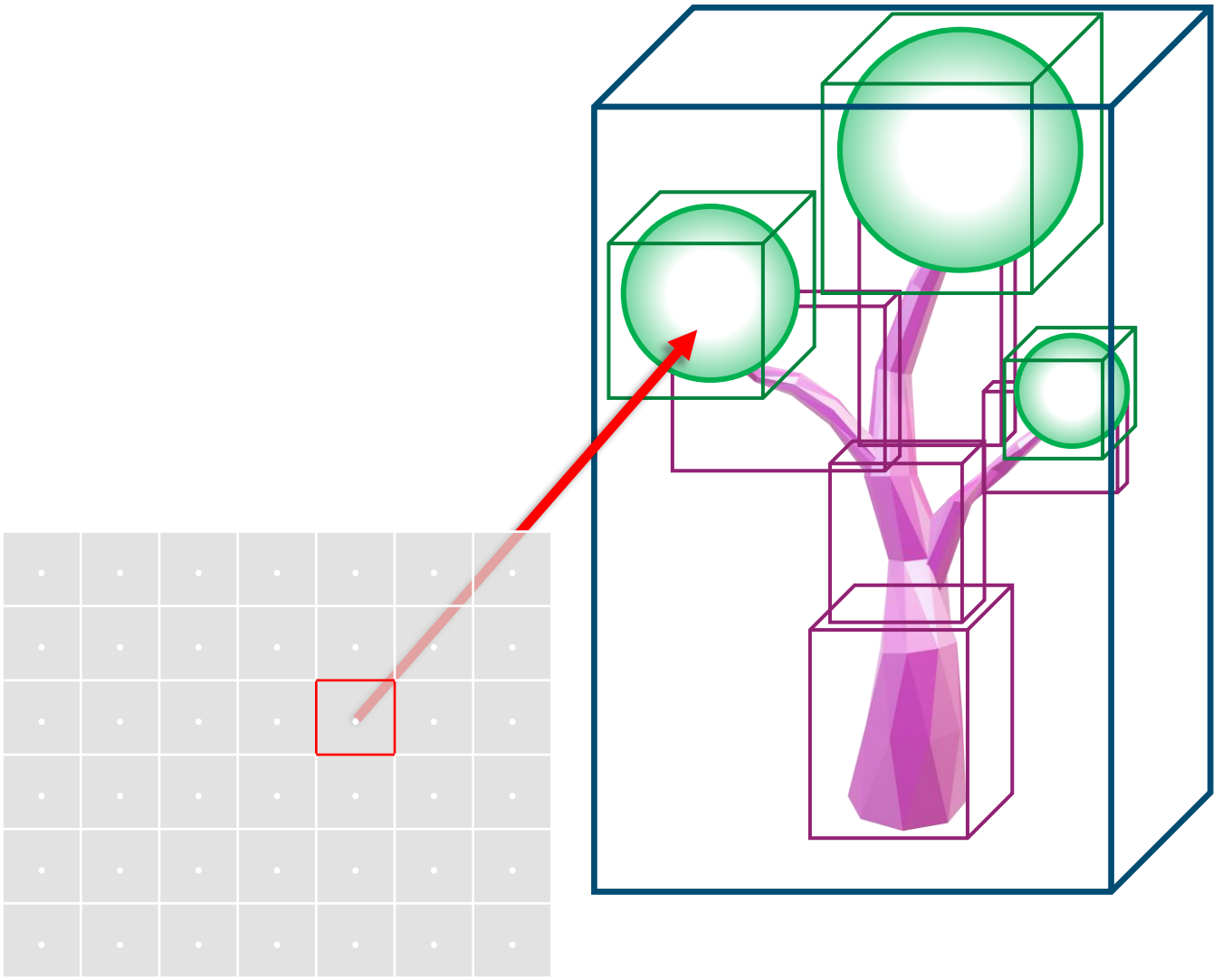
Bounding Volume Hierarchy Traversal





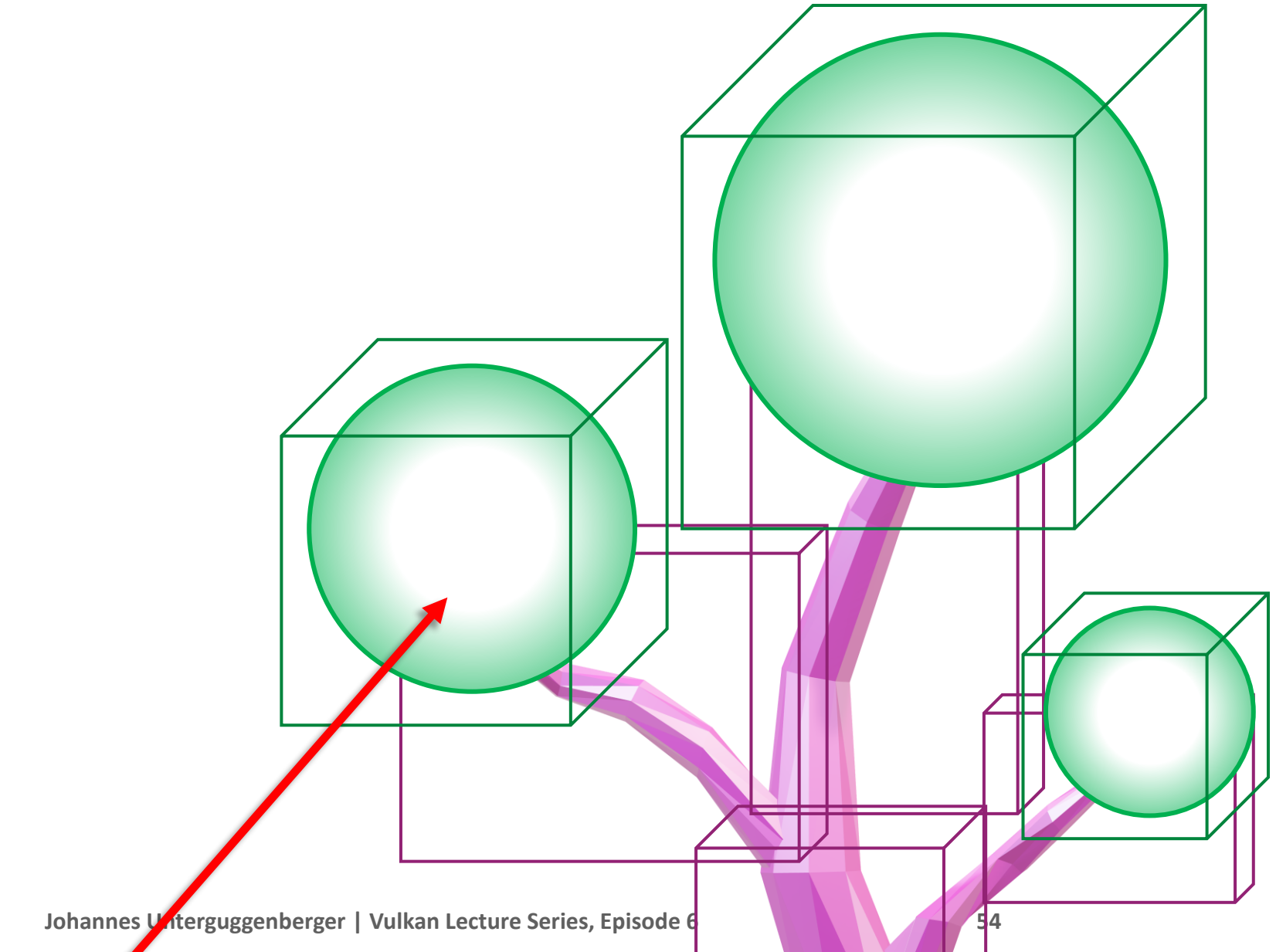
Top-Level
Acceleration Structure

Bounding Volume Hierarchy Traversal



Top-Level
Acceleration Structure

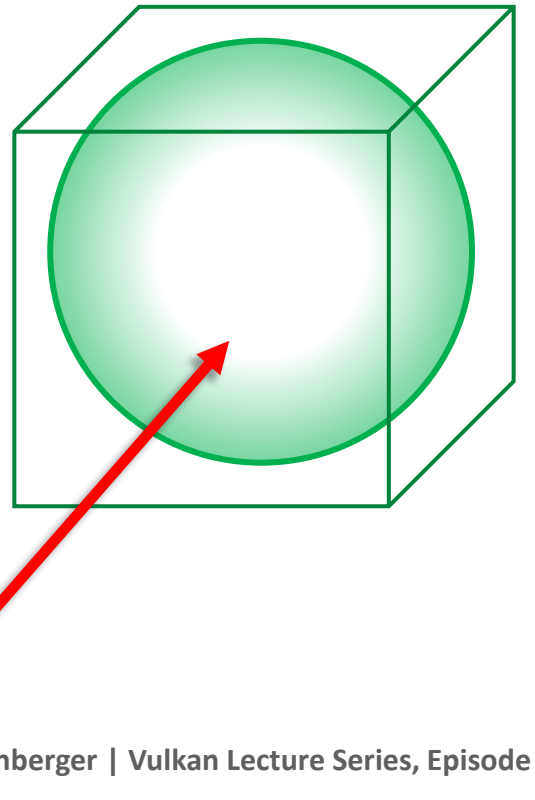
Bounding Volume Hierarchy Traversal



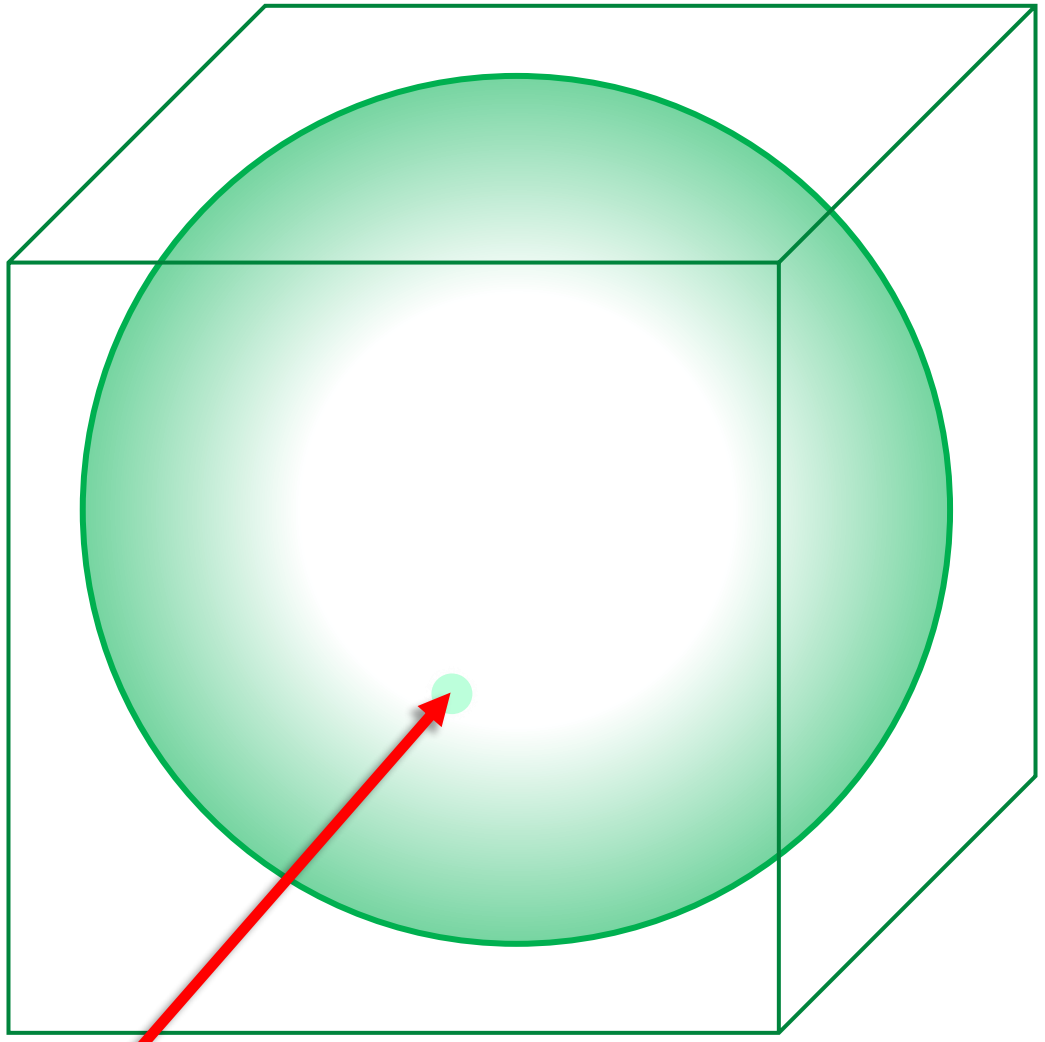
Top-Level
Acceleration Structure

Bounding Volume Hierarchy Traversal

Top-Level
Acceleration Structure



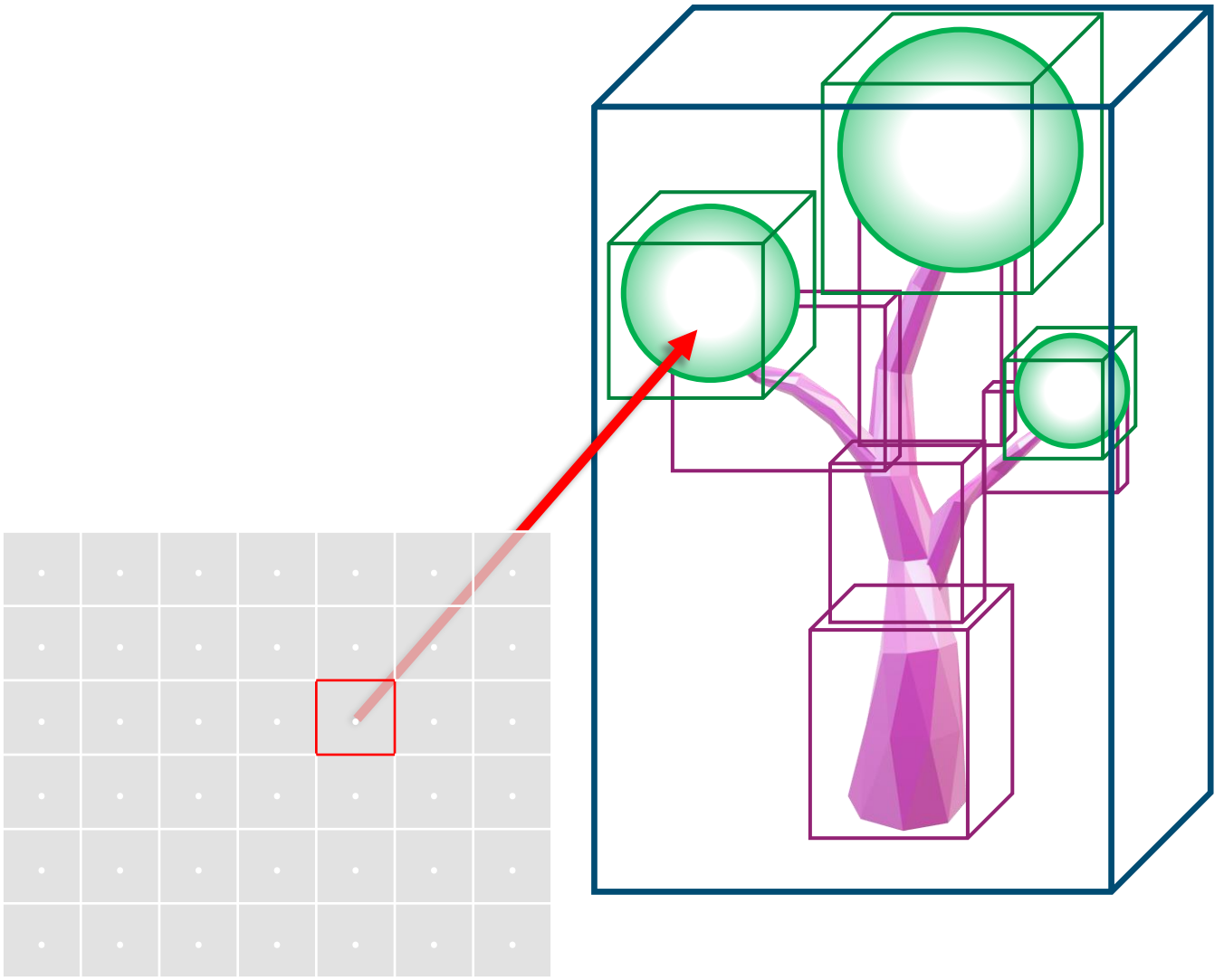
Bounding Volume Hierarchy Traversal



Top-Level
Acceleration Structure

Intersection Shader

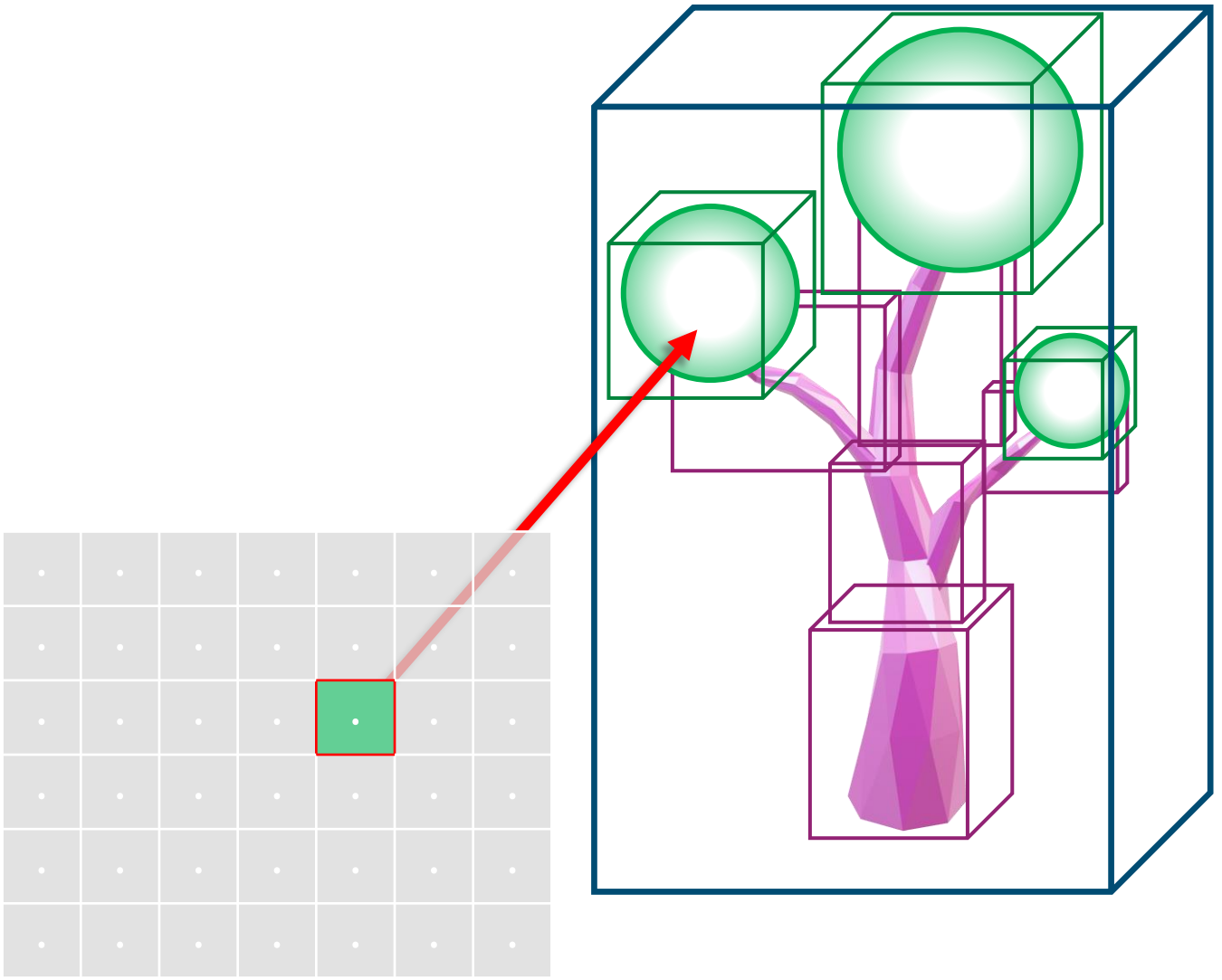
Bounding Volume Hierarchy Traversal



Top-Level
Acceleration Structure

Closest Hit Shader

Bounding Volume Hierarchy Traversal



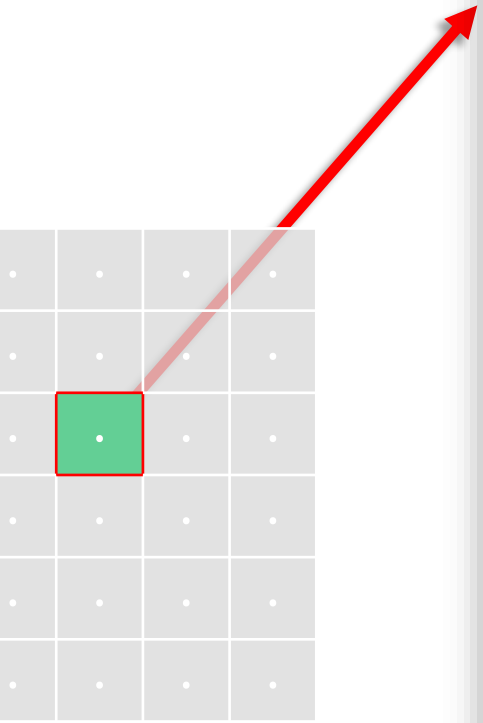
Top-Level
Acceleration Structure

Tracing One Ray

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():

Top-Level
Acceleration Structure



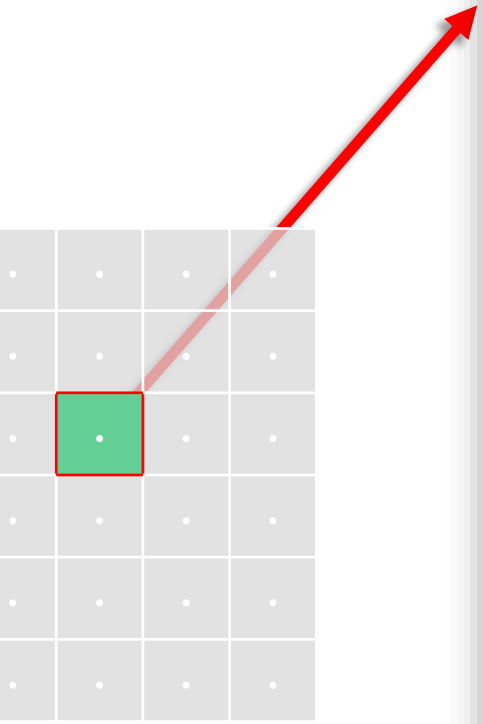
Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():

Top-Level Acceleration Structure

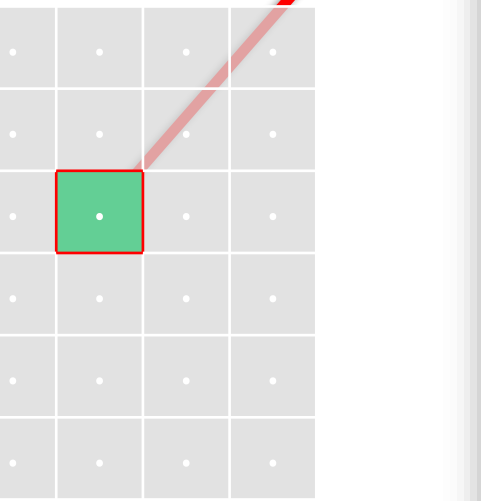


Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



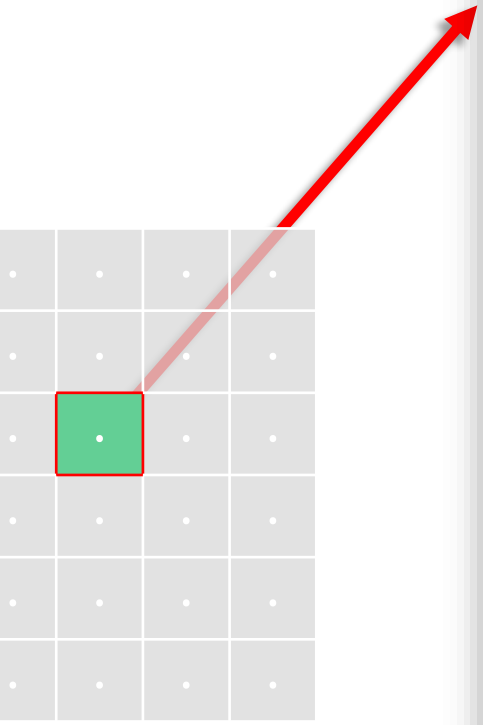
Top-Level
Acceleration Structure

Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



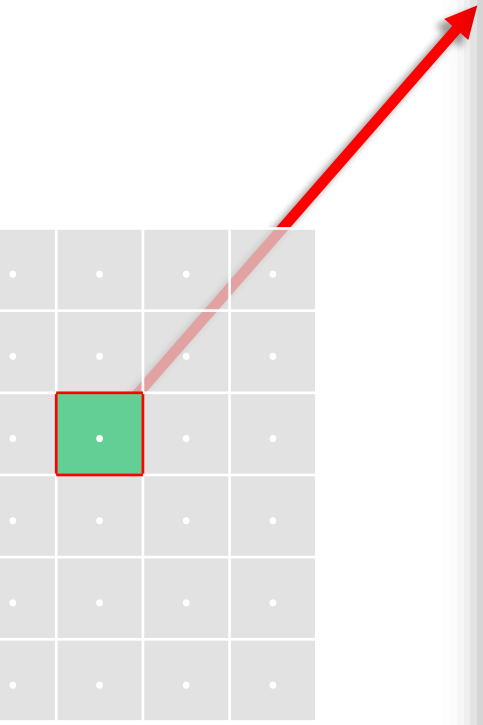
Top-Level
Acceleration Structure

Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



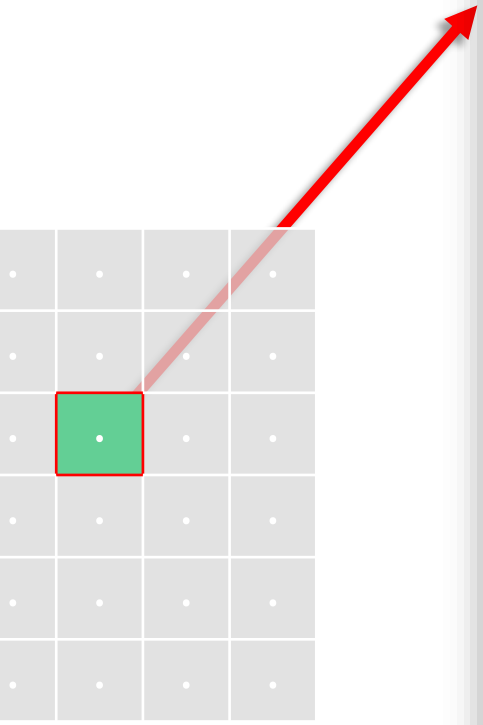
Top-Level
Acceleration Structure

Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



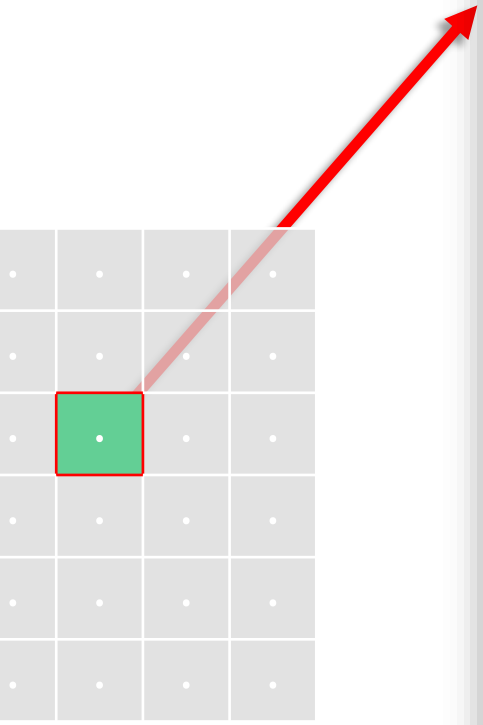
Top-Level
Acceleration Structure

Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



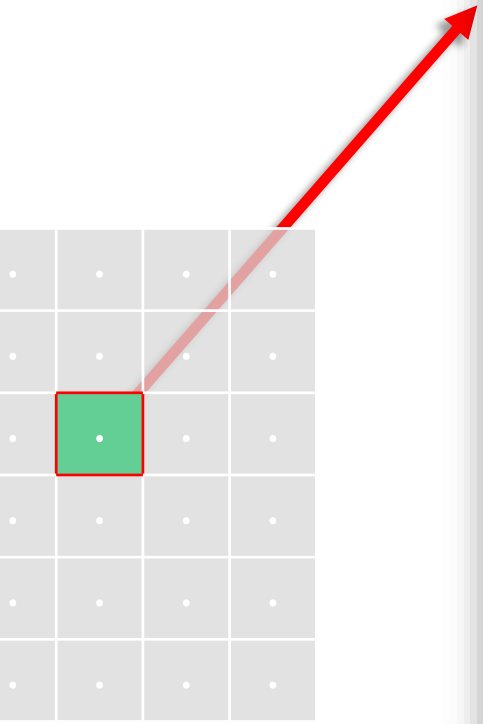
Top-Level
Acceleration Structure

Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



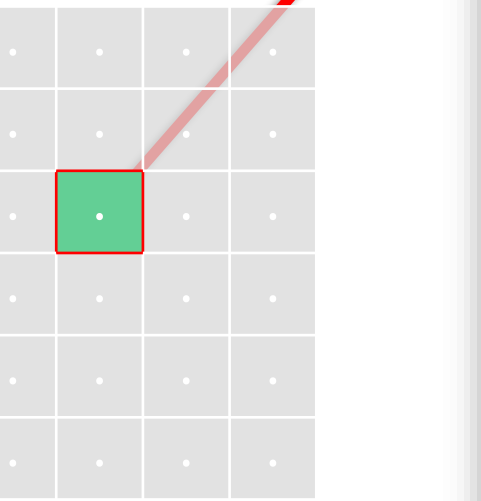
Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Top-Level Acceleration Structure

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



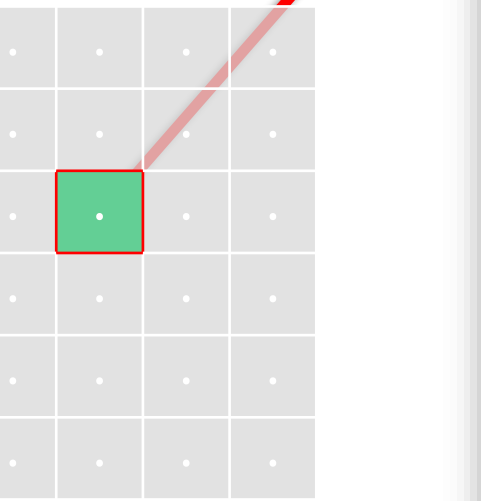
Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

Top-Level Acceleration Structure

Tracing One Ray

Start ONE ray in a shader with traceRayEXT():



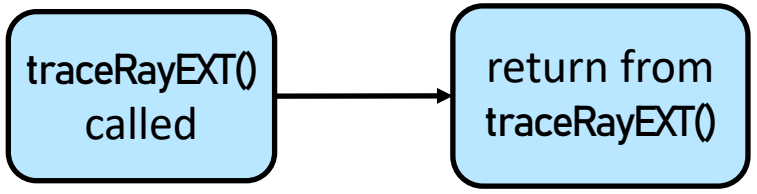
Top-Level
Acceleration Structure

Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT topLevelAS;
layout(set = 0, binding = 1, rgba8) uniform image2D image;
layout(location = 0) rayPayloadEXT vec3 hitValue;

void main() {
    // ...
    // Compute ray parameters like origin and direction
    // ...
    traceRayEXT(topLevelAS,
        gl_RayFlagsOpaqueEXT /*ray flags*/, 0xff /*culling mask*/,
        0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
        origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/,
        0 /*location of payload*/);
    imageStore(image, ivec2(gl_LaunchIDEXT.xy), vec4(hitValue, 0.0));
}
```

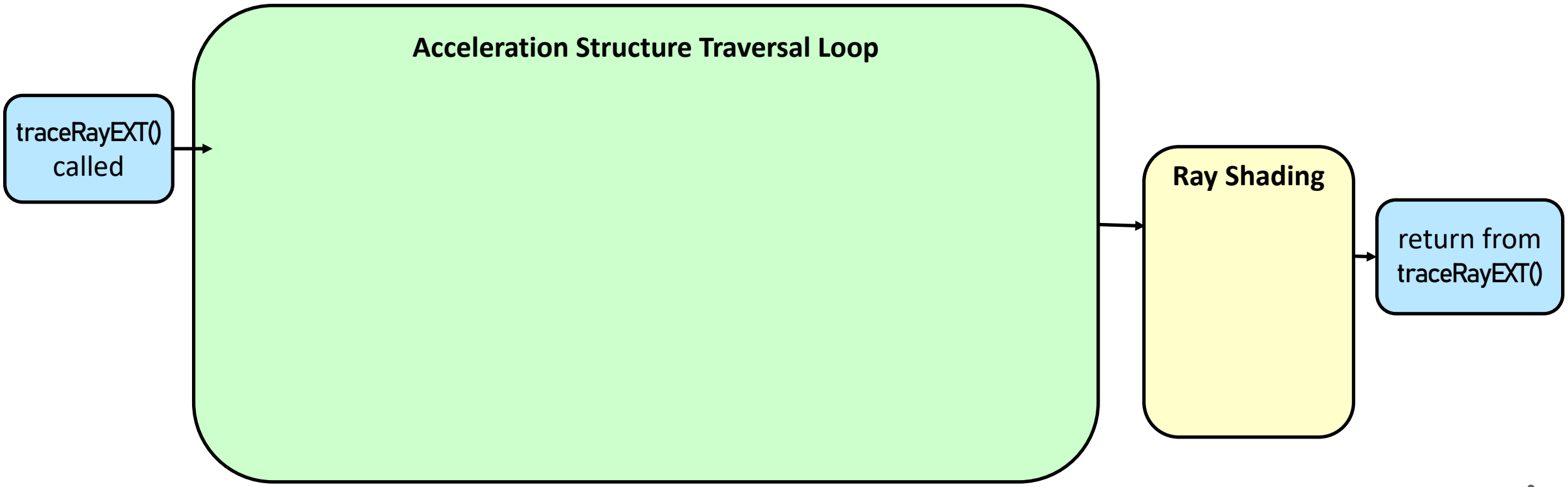
Ray Tracing Pipeline in Depth

- Ray Generation Shader
- Per pixel: traceRayEXT()



Ray Tracing Pipeline in Depth

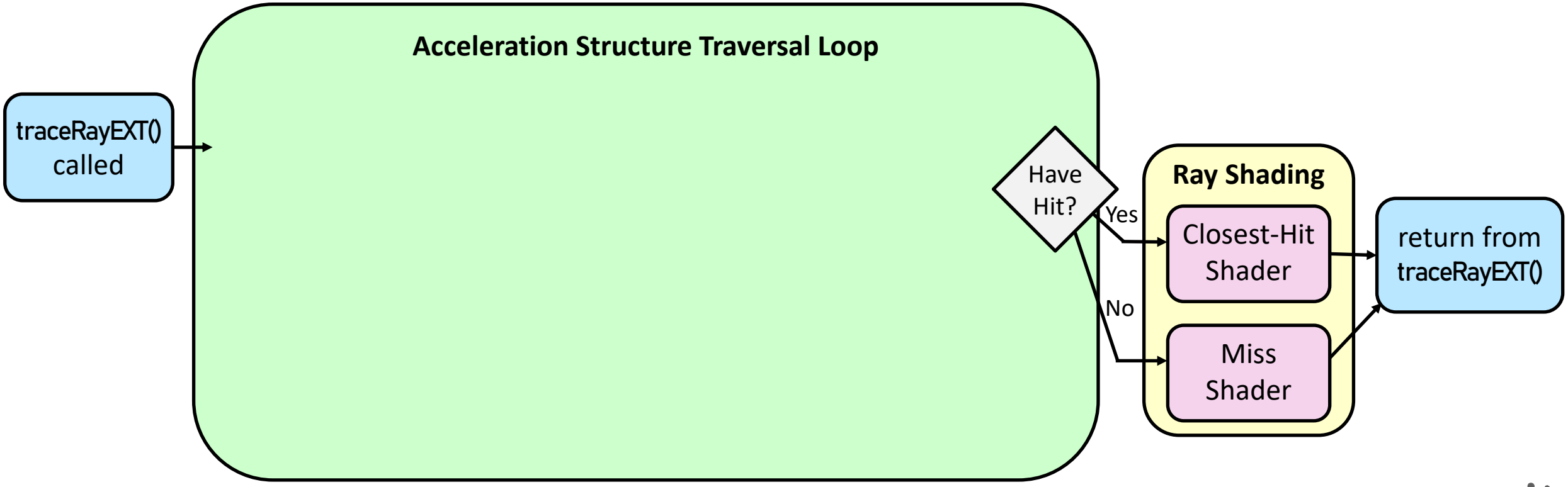
- Ray Generation Shader
- Per pixel: traceRayEXT()



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

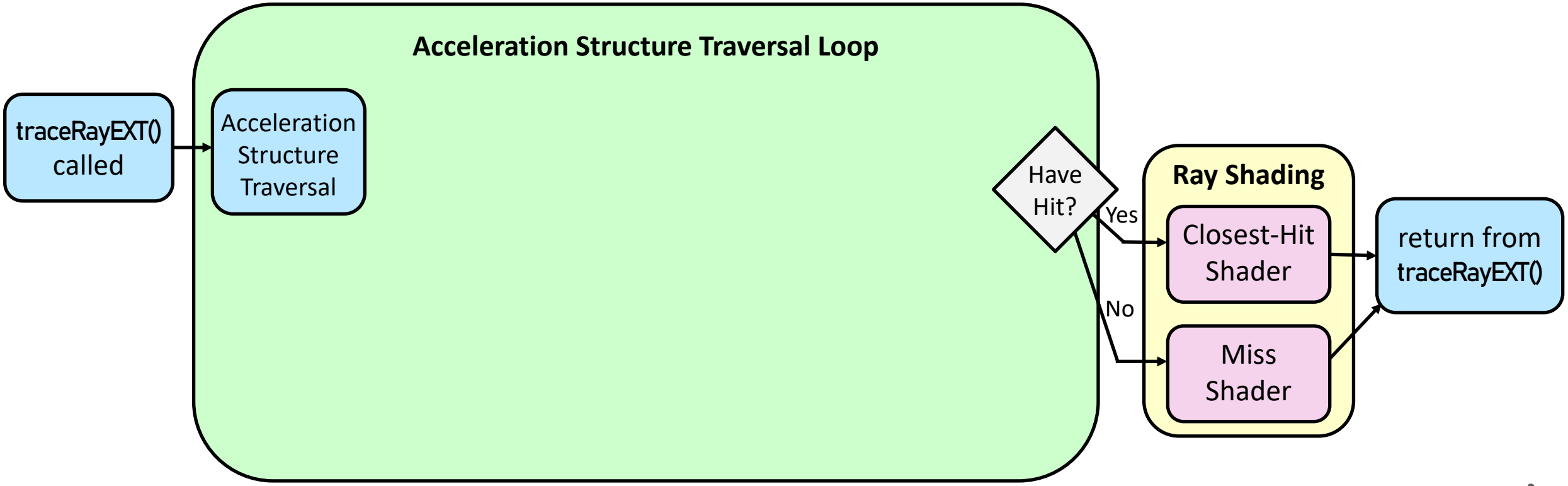
- Shading in Closest-Hit or in Miss Shader
  - If we had a valid hit => Closest-Hit Shader
  - If we didn't => Miss Shader



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

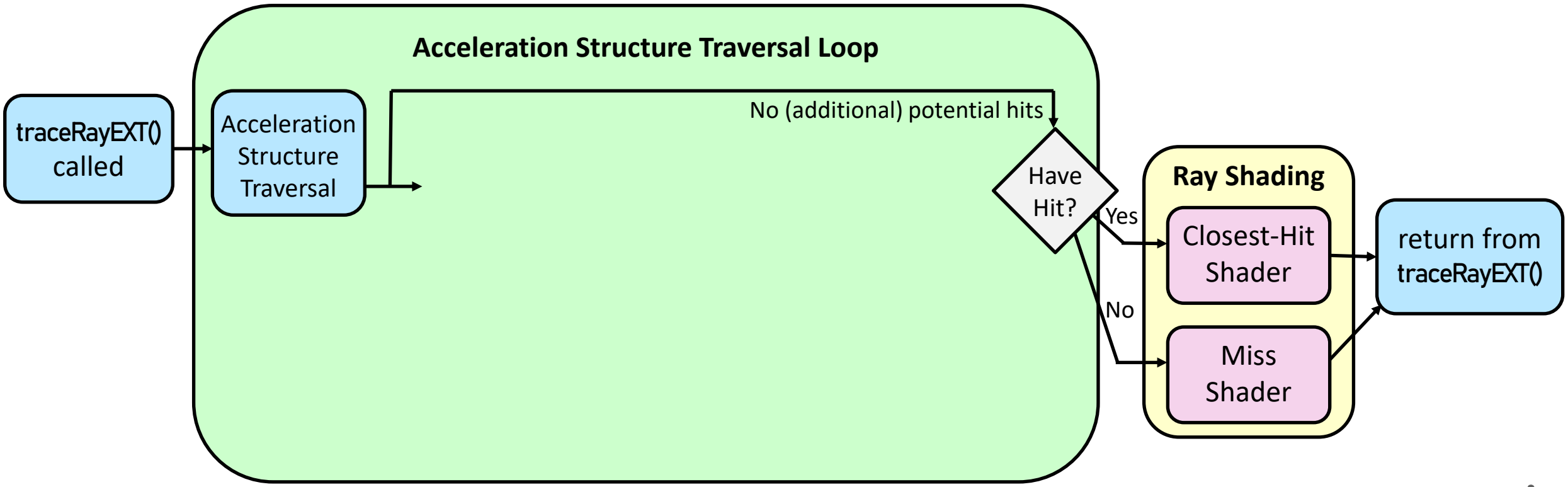
- Traverse the acceleration structure
  - Opaque process, configurable
  - Vendor-specific implementation



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

- No hit => acceleration structure traversal ends

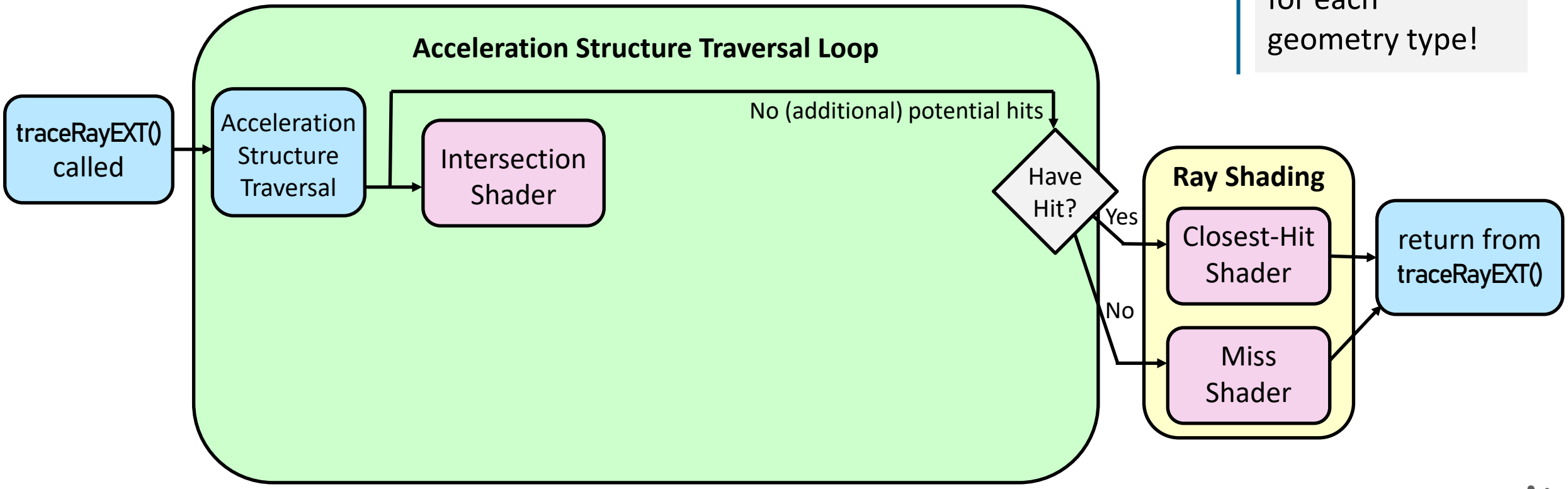


Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

- For potential intersections, Intersection Shader is invoked
  - Specific to a particular geometry type (spheres, Bezier patches, …)
  - Default, optimized intersection for triangles

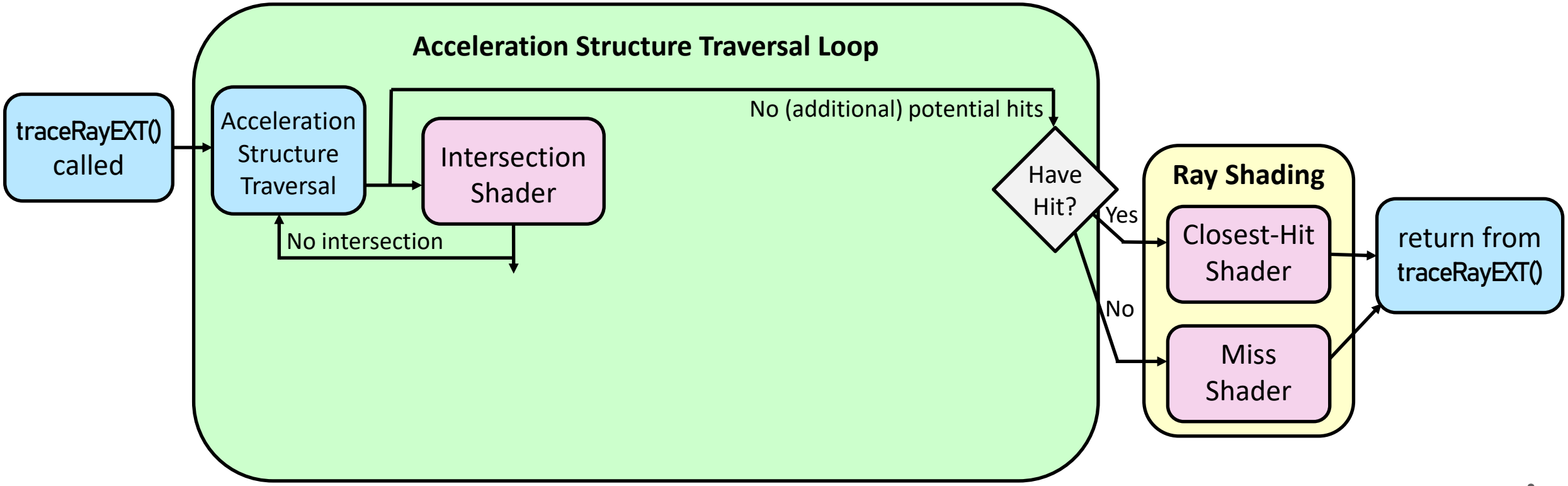
Different shader for each geometry type!



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

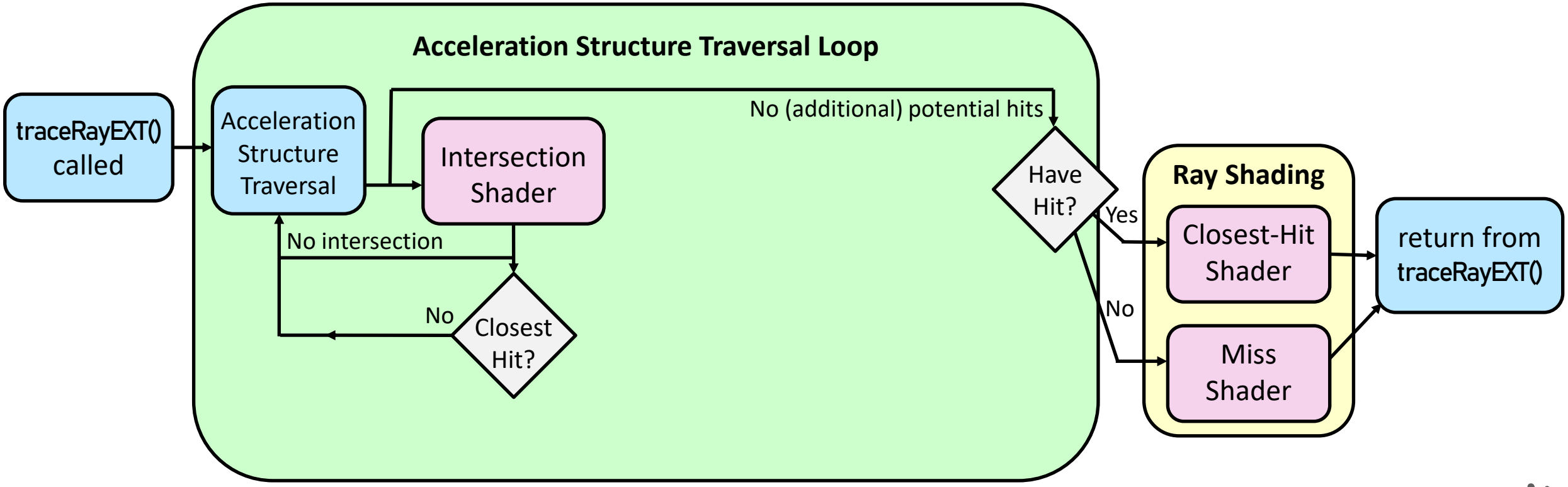
- No intersection from Intersection Shader?
- => Continue traversing the acceleration structure!



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

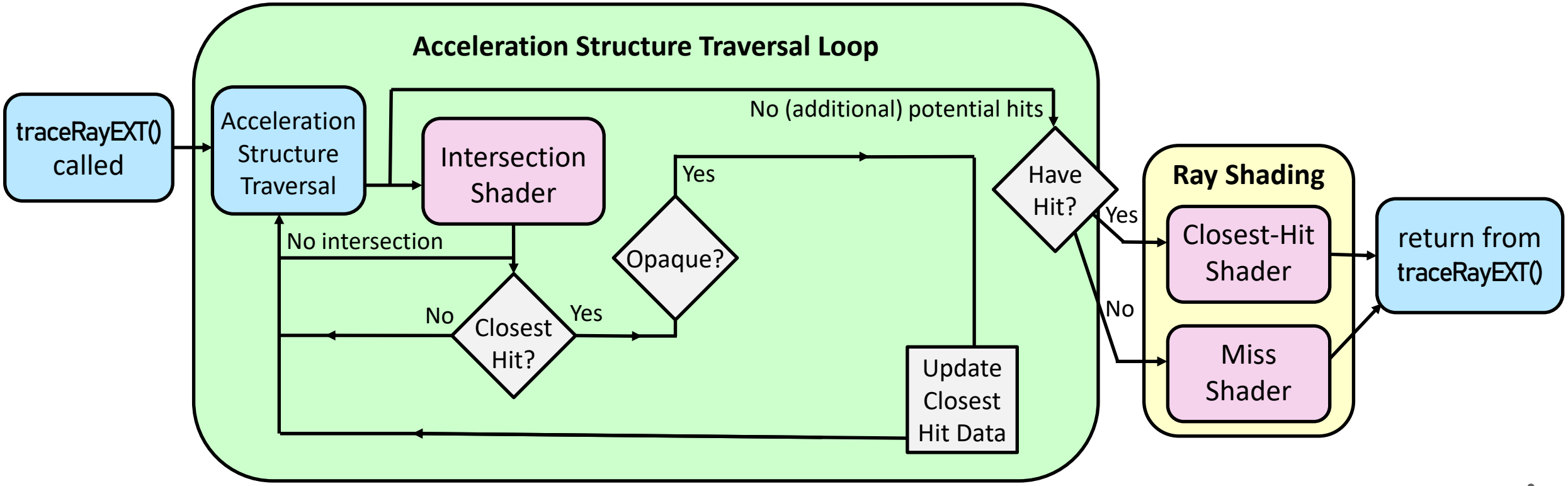
- We have detected an intersection, but:
- The intersection is not the closest hit (so far)
- => Continue traversing the acceleration structure!



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

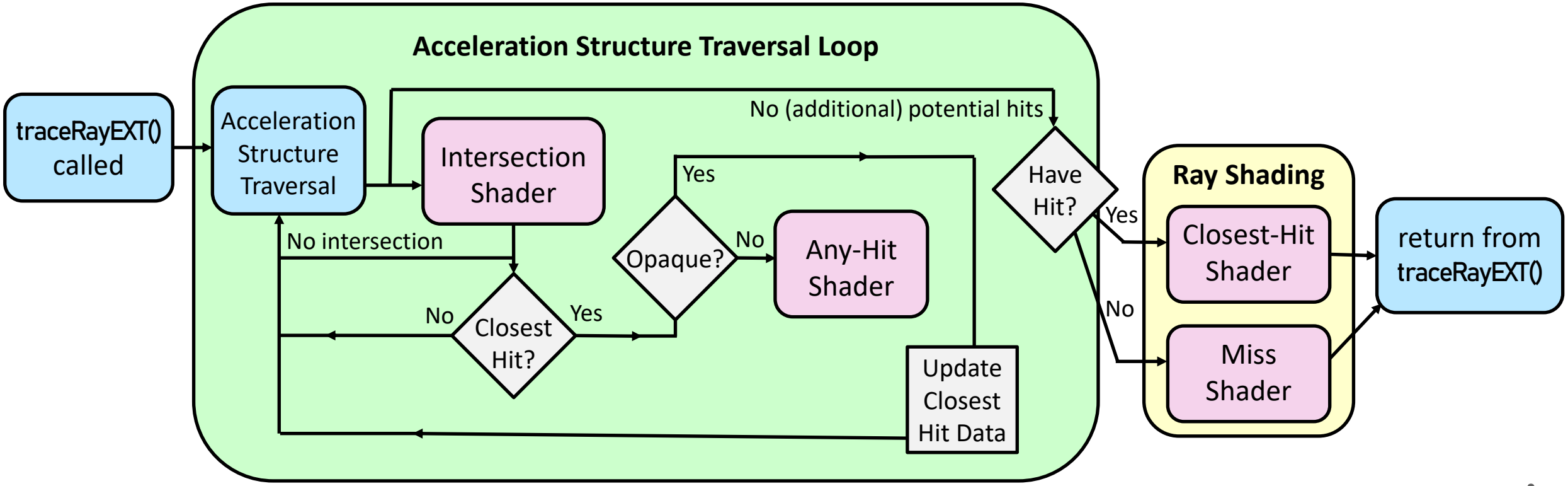
- Hit detected AND it is opaque
  - The new hit is our closest hit (so far!) => Update the closest hit data
  - Continue traversing the acceleration structure (search for closer hit)



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

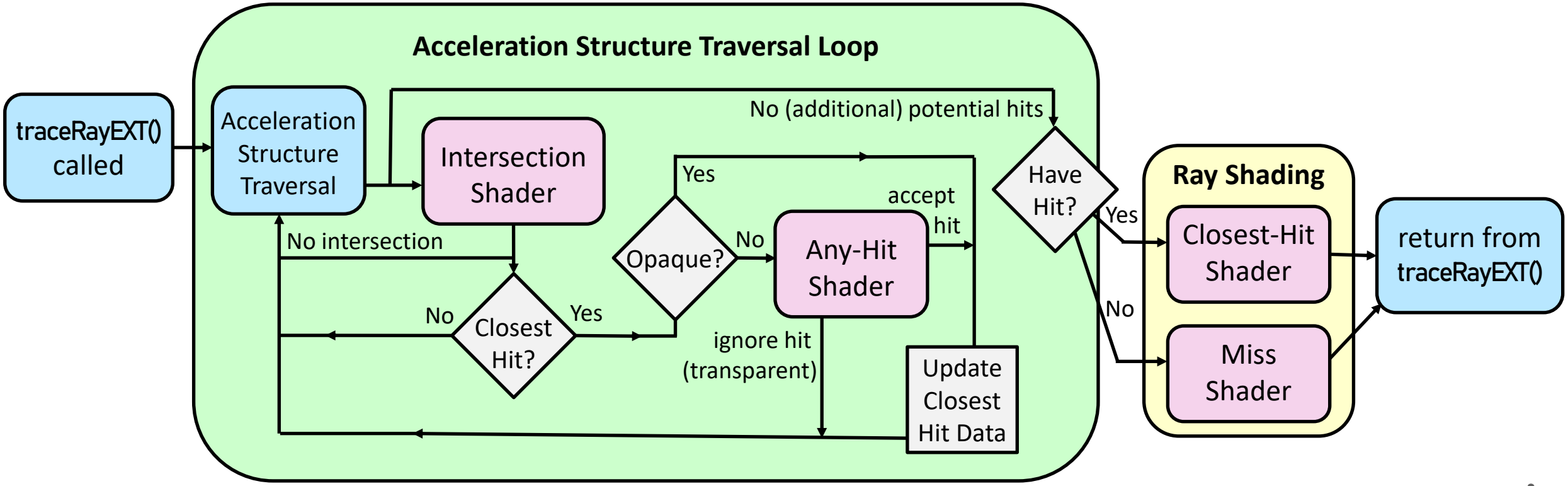
- Hit detected AND it is not opaque
  - => run the Any-Hit Shader



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

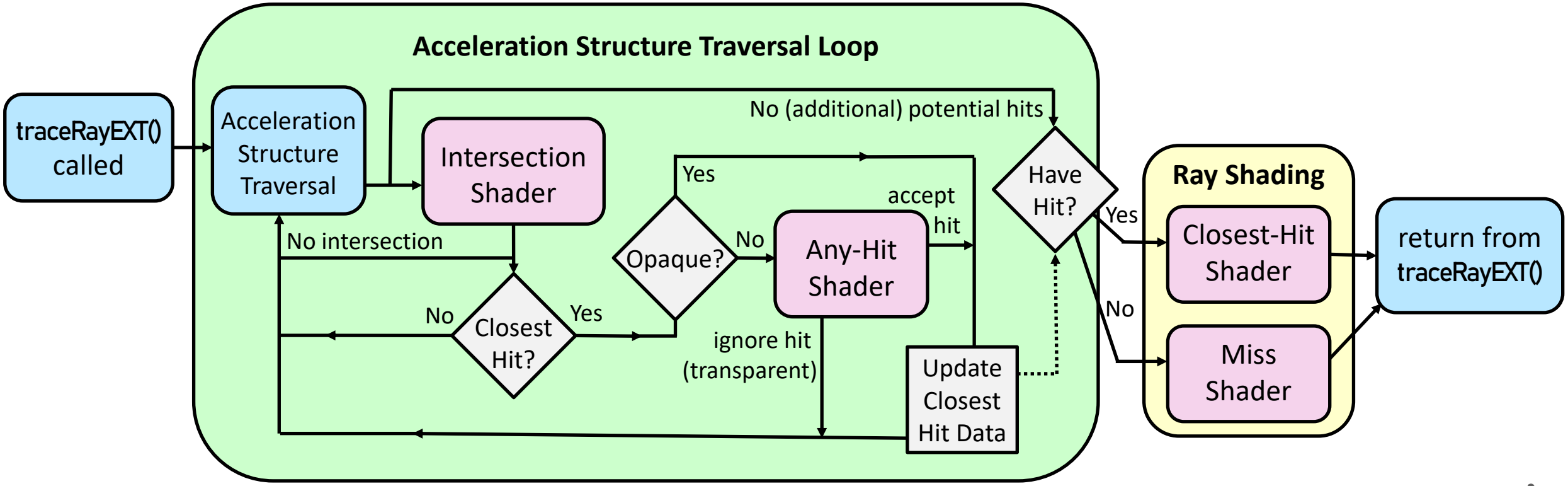
- In hit shader, accept or ignore the hit
  - Accept hit => Current hit becomes the new closest hit
  - In any case, continue traversing AS to look for closer intersections



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

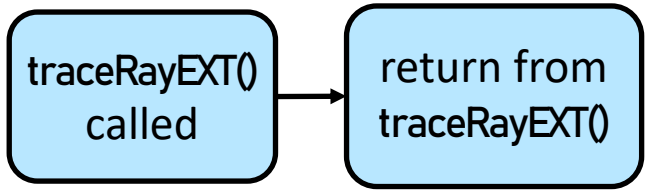
- Continue traversing AS until no closer hits discovered
  - Had a valid hit along the ray? => Shade via the Closest-Hit Shader
  - Didn't have valid hits? => Shade via the Miss Shader



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

Ray Tracing Pipeline in Depth

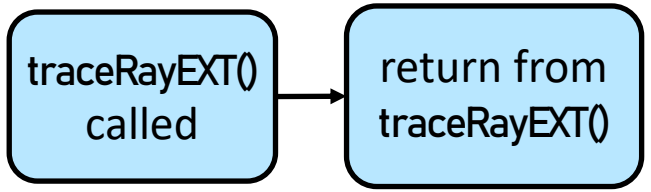
- Ray Generation Shader
- Per pixel: traceRayEXT()



Tracing All the Rays

Host-Side Code

- Ray Generation Shader
- Per pixel: traceRayEXT()



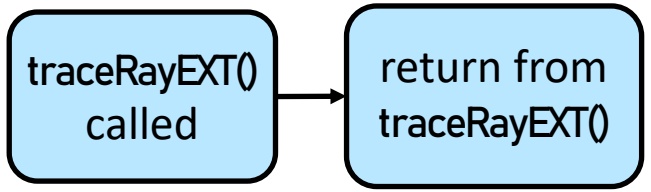
Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT as;
// ...
void main() {
  // ...
  traceRayEXT(as,
    gl_RayFlagsOpaqueEXT, 0xff,
    0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
    origin, 0.1, direction, 100.0,
    0 /*location of payload*/);
  // ...
}
```

```
vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,     &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

Host-Side Code

- Ray Generation Shader
- Per pixel: traceRayEXT()



Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT as;
// ...
void main() {
  // ...
  traceRayEXT(as,
    gl_RayFlagsOpaqueEXT, 0xff,
    0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
    origin, 0.1, direction, 100.0,
    0 /*location of payload*/);
  // ...
}
```

```
vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,     &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

Shader Binding Table

Shader Binding Table

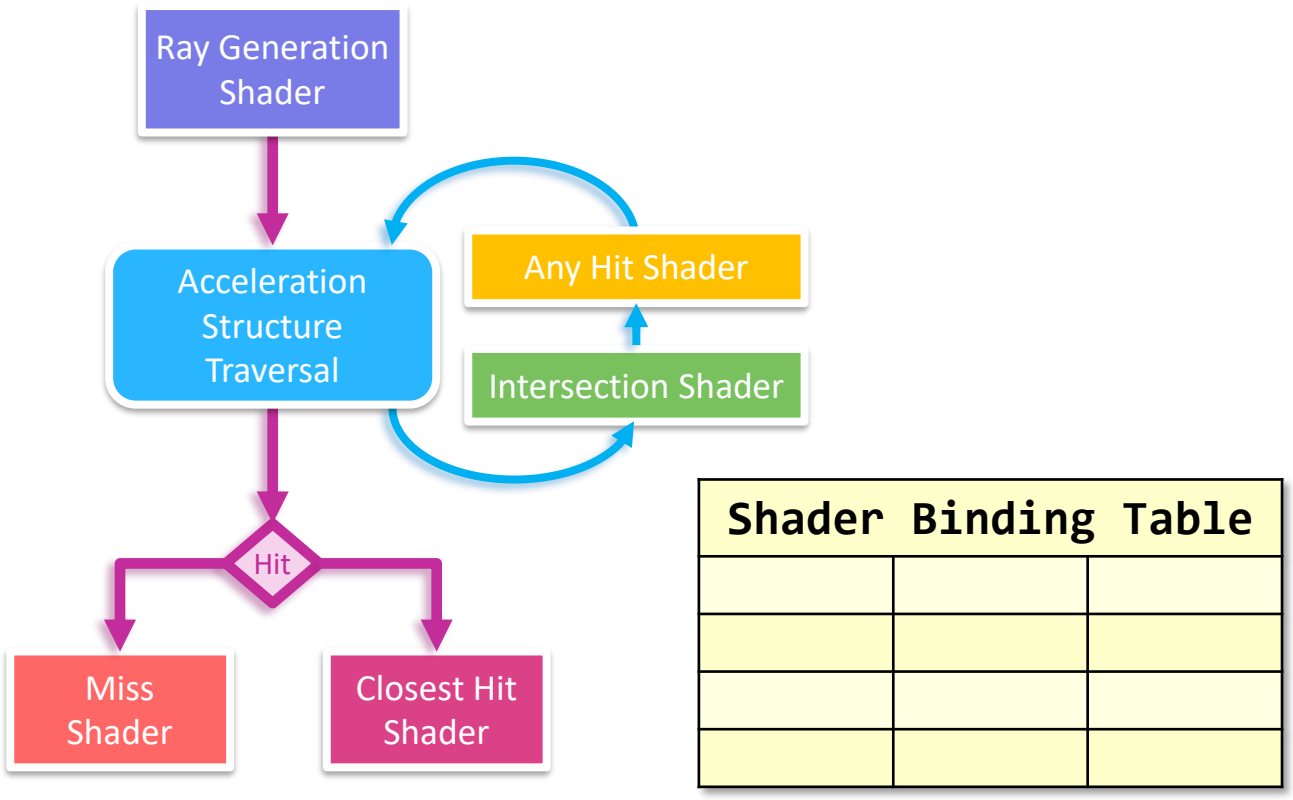
Shader Binding Table		

Ray Generation Shader (GLSL)

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT as;
// ...
void main() {
  // ...
  traceRayEXT(as,
    gl_RayFlagsOpaqueEXT, 0xff,
    0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
    origin, 0.1, direction, 100.0,
    0 /*location of payload*/);
  // ...
}
```

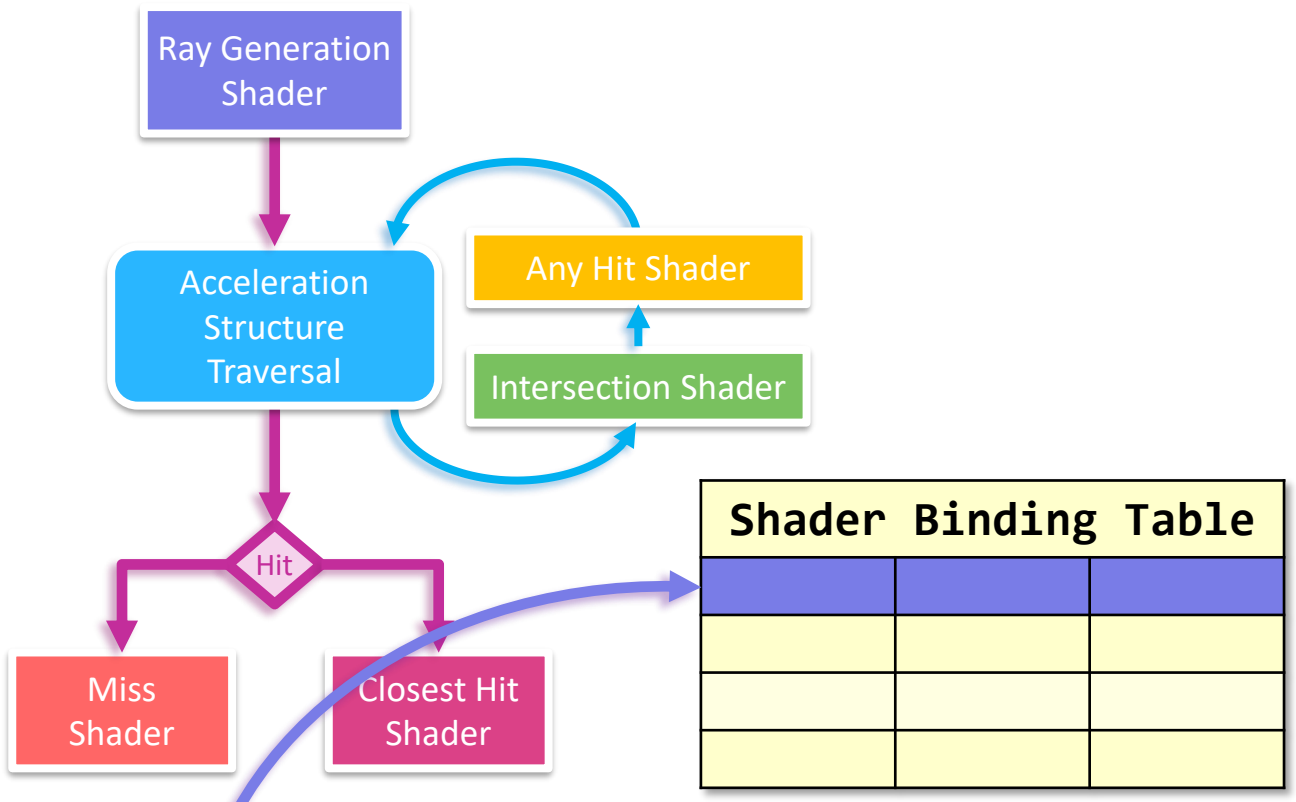
```
vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,     &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

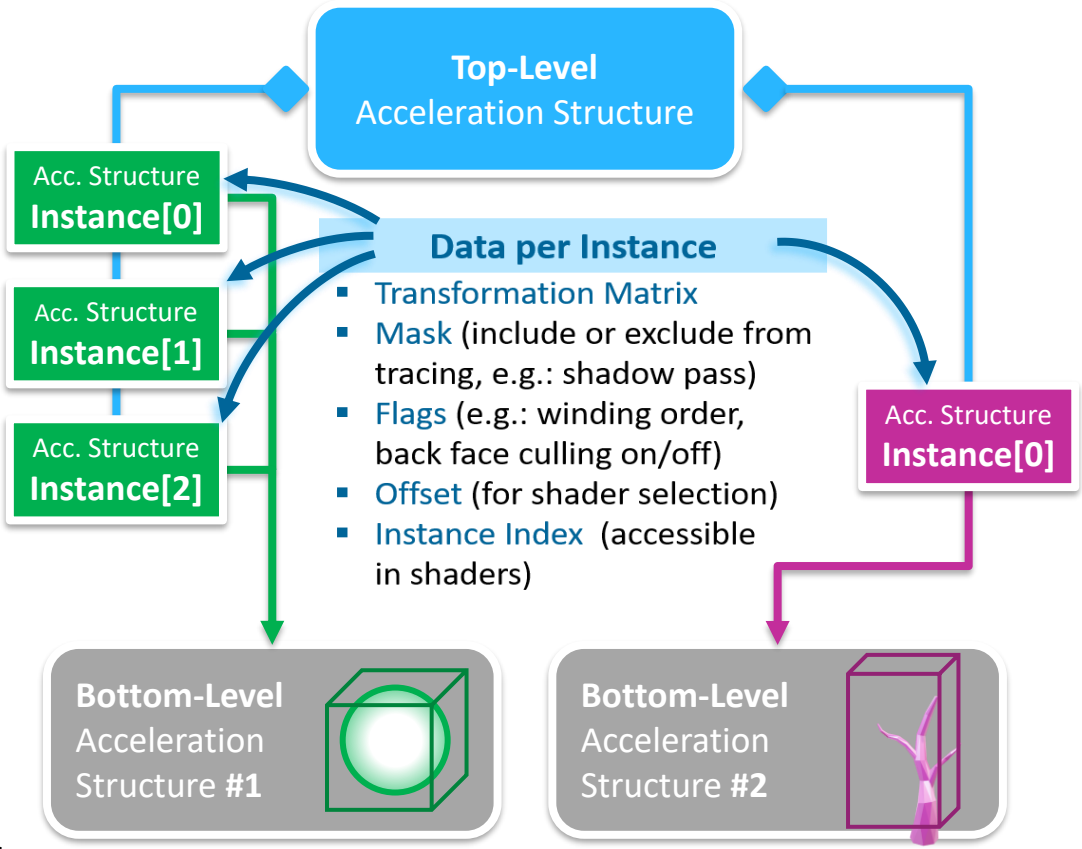
Shader Binding Table




```
vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,     &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

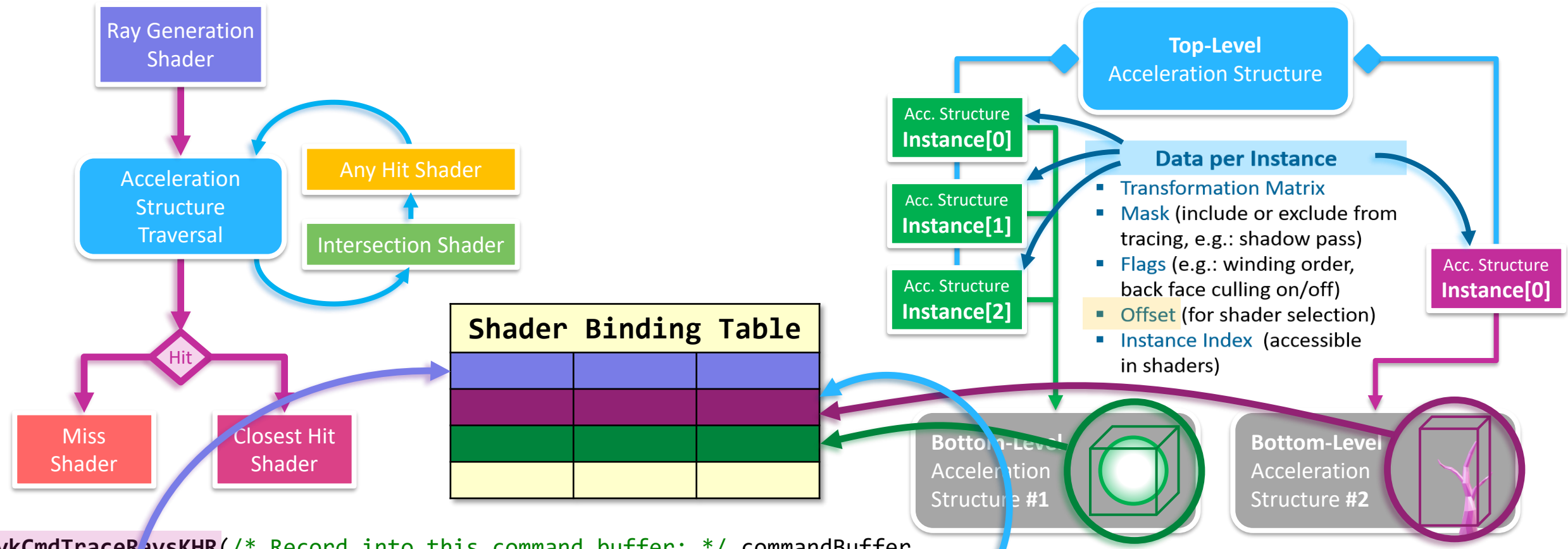
Shader Binding Table





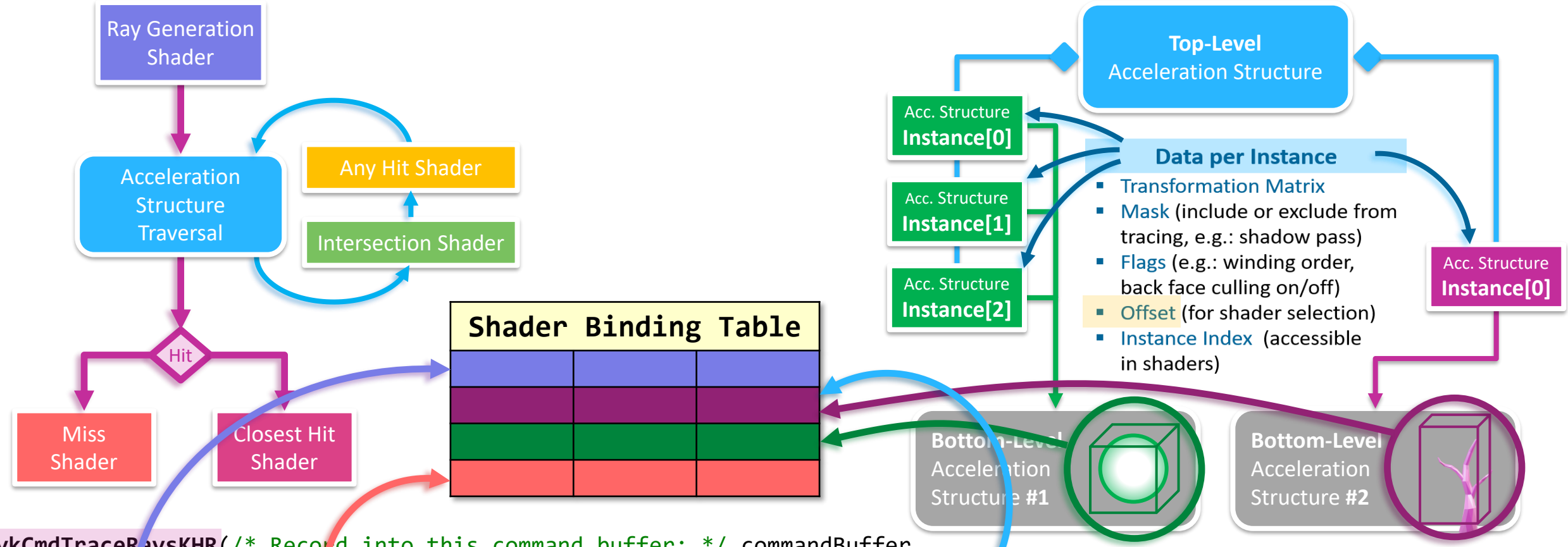

```
vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,    &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

Shader Binding Table




```
vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,    &missShaderStartAddr,  &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

Shader Binding Table




```
vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,  miss shaders,  and  hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,  &missShaderStartAddr,  &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

Shader Binding Table

```
VkStridedDeviceAddressRegionKHR rayGenOffsets{ rayGenBufferDeviceAddress };
rayGenOffsets.stride = 32;        // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
rayGenOffsets.size = rayGenOffsets.stride * numberOfRayGenShaders;

VkStridedDeviceAddressRegionKHR missShaderOffsets{ rayMissBufferDeviceAddress };
missShaderOffsets.stride = 32;    // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
missShaderOffsets.size = missShaderOffsets.stride * numberOfMissShaders;

VkStridedDeviceAddressRegionKHR hitGroupsOffsets{ rayHitBufferDeviceAddress };
hitGroupsOffsets.stride = 32;     // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
hitGroupsOffsets.size = hitGroupsOffsets.stride * numberOfHitGroups;

VkStridedDeviceAddressRegionKHR callableShaderOffsets{};

vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,     &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```


Shader Binding Table

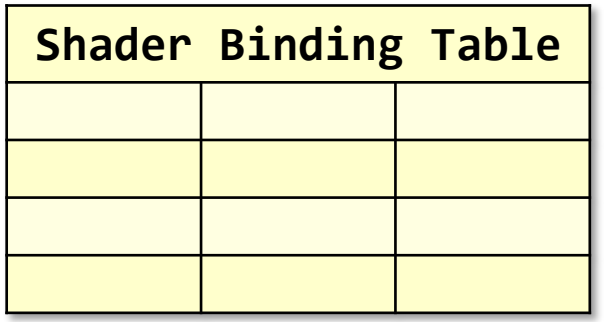
```
VkStridedDeviceAddressRegionKHR rayGenOffsets{ rayGenBufferDeviceAddress };
rayGenOffsets.stride = 32;       // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
rayGenOffsets.size = rayGenOffsets.stride * numberOfRayGenShaders;

VkStridedDeviceAddressRegionKHR missShaderOffsets{ rayMissBufferDeviceAddress };
missShaderOffsets.stride = 32;   // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
missShaderOffsets.size = missShaderOffsets.stride * numberOfMissShaders;

VkStridedDeviceAddressRegionKHR hitGroupsOffsets{ rayHitBufferDeviceAddress };
hitGroupsOffsets.stride = 32;    // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
hitGroupsOffsets.size = hitGroupsOffsets.stride * numberOfHitGroups;

VkStridedDeviceAddressRegionKHR callableShaderOffsets{};

vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,     &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```


Shader Binding Table

```
VkStridedDeviceAddressRegionKHR rayGenOffsets{ rayGenBufferDeviceAddress };
rayGenOffsets.stride = 32;        // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
rayGenOffsets.size = rayGenOffsets.stride * numberOfRayGenShaders;

VkStridedDeviceAddressRegionKHR missShaderOffsets{ rayMissBufferDeviceAddress };
missShaderOffsets.stride = 32;    // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
missShaderOffsets.size = missShaderOffsets.stride * numberOfMissShaders;

VkStridedDeviceAddressRegionKHR hitGroupsOffsets{ rayHitBufferDeviceAddress };
hitGroupsOffsets.stride = 32;     // <-- TODO: query via VkPhysicalDeviceRayTracingPipelinePropertiesKHR
hitGroupsOffsets.size = hitGroupsOffsets.stride * numberOfHitGroups;

VkStridedDeviceAddressRegionKHR callableShaderOffsets{};

vkCmdTraceRaysKHR(/* Record into this command buffer: */ commandBuffer,
    // Pass all the offsets for:
    // ray gen. shaders,   miss shaders,     and   hit groups (i.e. tuples of [closest hit], [any hit], [intersection])
    &rayGenStartAddr,     &missShaderStartAddr,   &hitGroupsStartAddr,
    &callableShadersStartAddr,
    // Specify ray tracing resolution of 1920 x 1080 x 1:
    1920, 1080, 1);
```

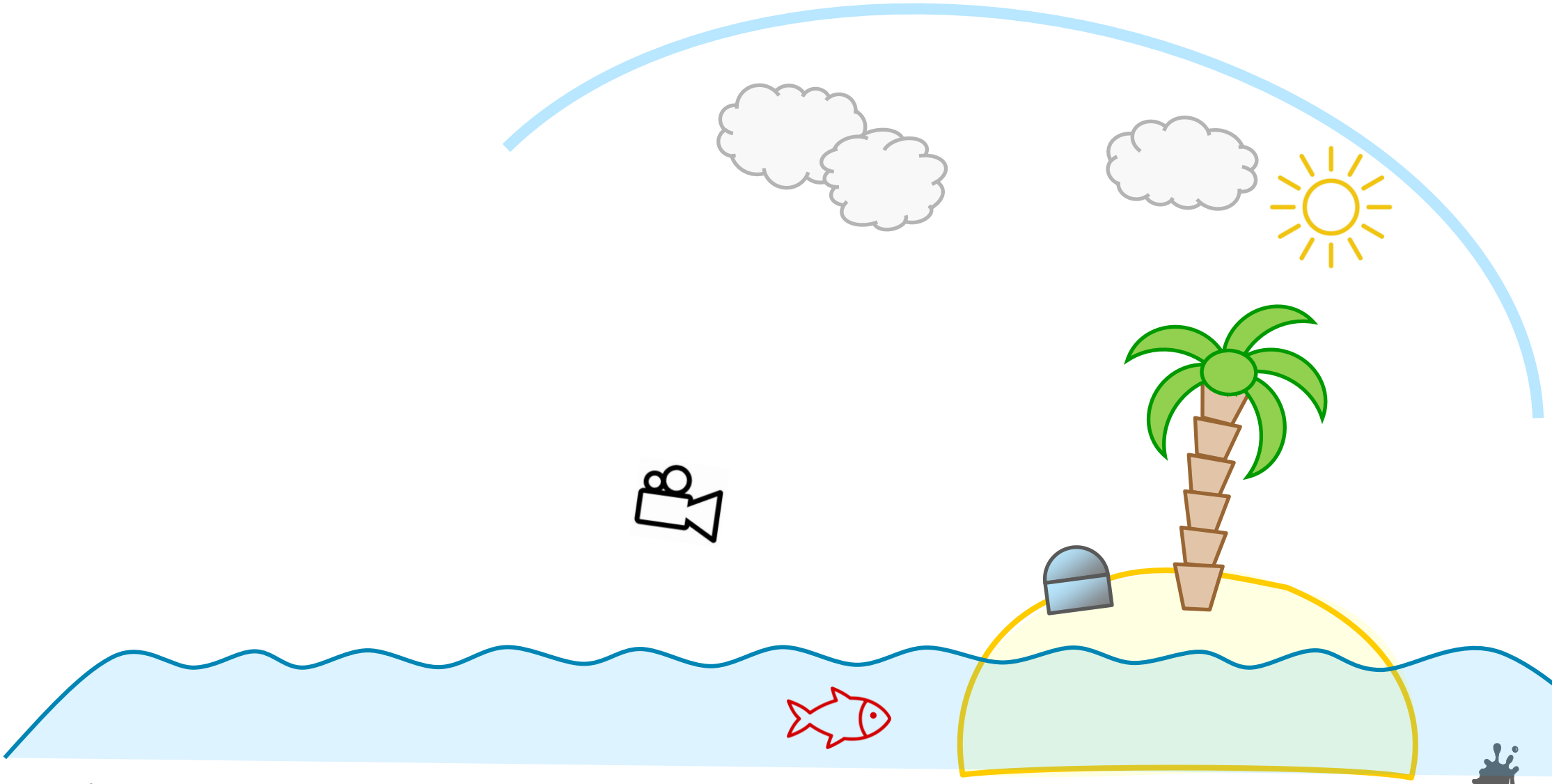

Shader Binding Table

- Shader Binding Table is a buffer
  - VK_BUFFER_USAGE_SHADER_BINDING_TABLE_BIT_KHR
  - VK_BUFFER_USAGE_SHADER_DEVICE_ADDRESS_BIT
  - Addresses are passed to vkCmdTraceRaysKHR()



- **One** shader binding table **entry** refers to 1, 2, or 3 shaders ("**Shader Group**")
  - E.g.: One specific **ray generation** shader (**"General Group"**)
  - E.g.: One specific **miss** shader (**"General Group"**)
  - E.g.: A group of shaders for when geometry was **hit** ("**Hit Group**")
    - E.g.: Closest hit shader only for opaque triangle mesh (**"Triangle Hit Group"**)
    - E.g.: Closest hit + any hit for a transparent triangle mesh (^ same ^)
    - E.g.: Closest hit + any hit + intersection for a **"Procedural Hit Group"**

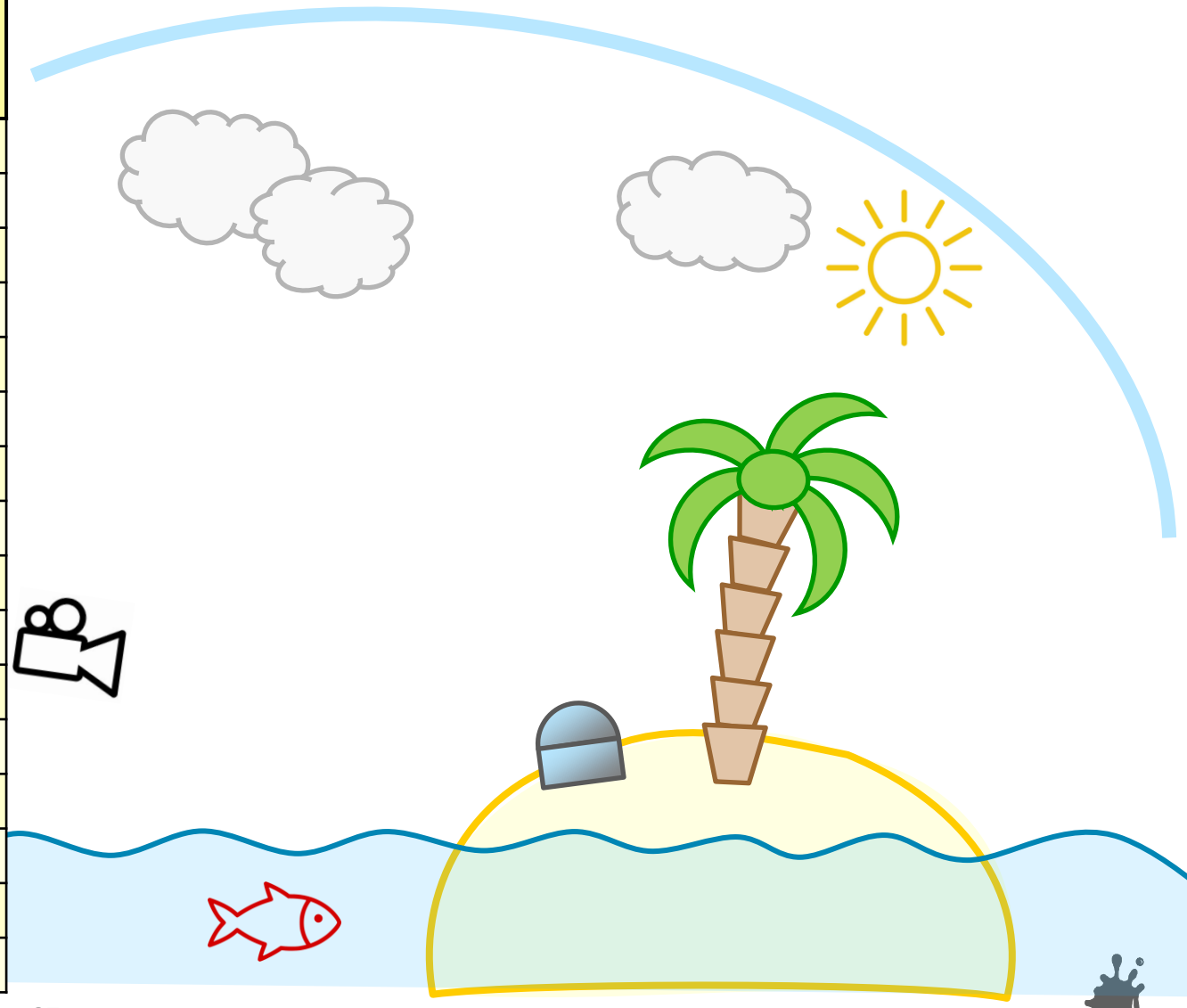
Shader Binding Table Example



Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

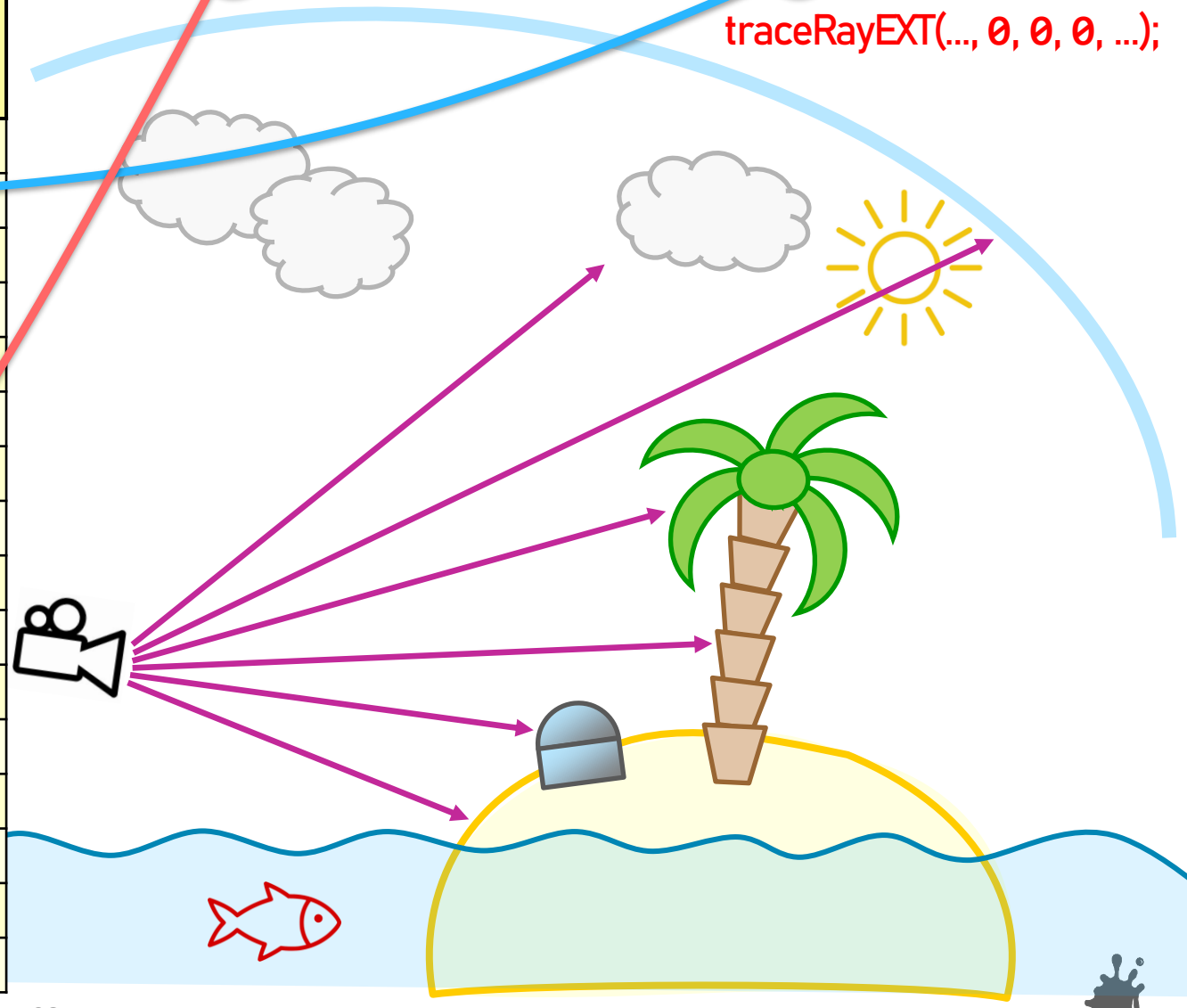


Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

traceRayEXT(..., 0, 0, 0, ...);

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

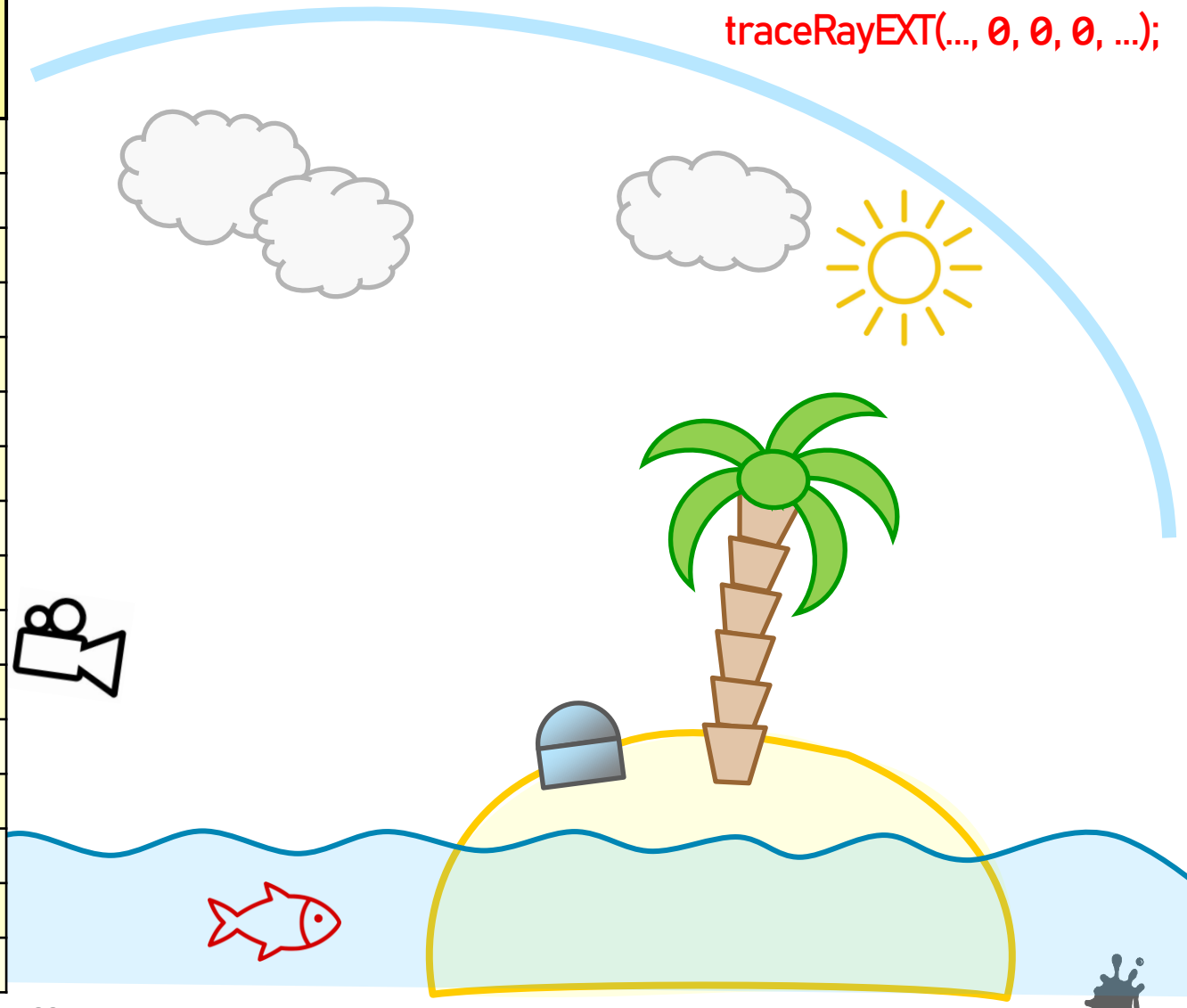


Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

traceRayEXT(..., 0, 0, 0, ...);

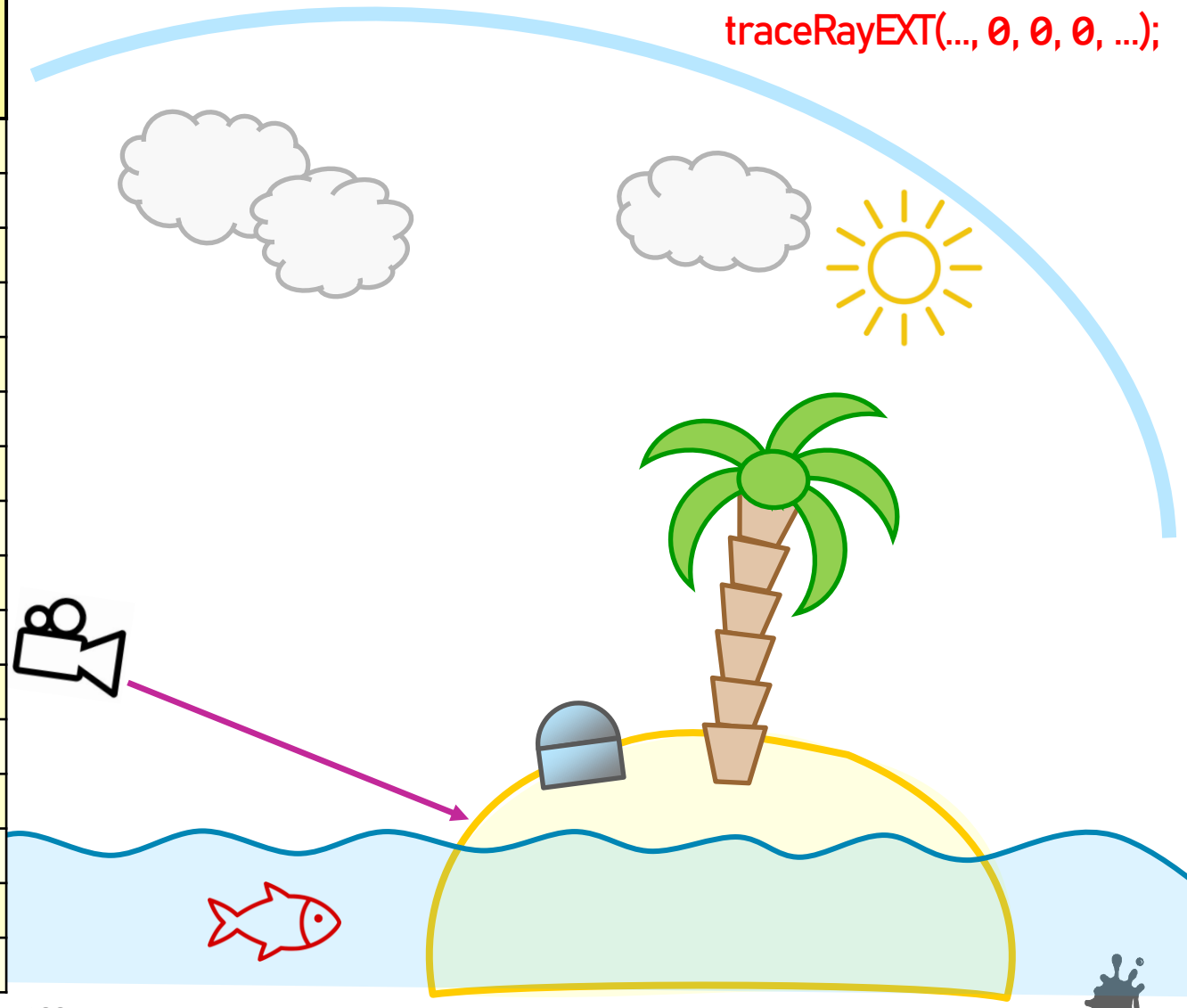


Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

traceRayEXT(..., 0, 0, 0, ...);

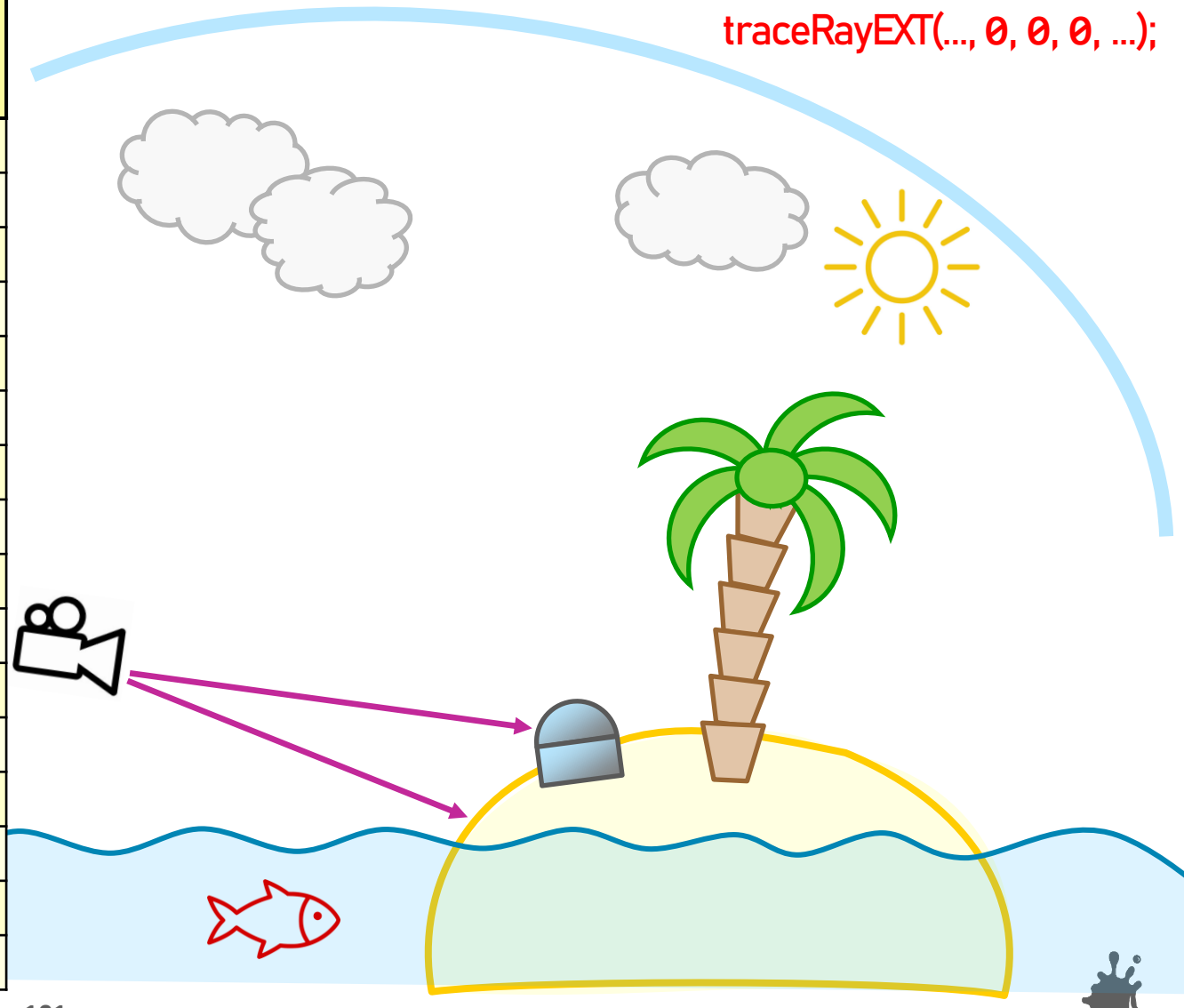


Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

traceRayEXT(..., 0, 0, 0, ...);

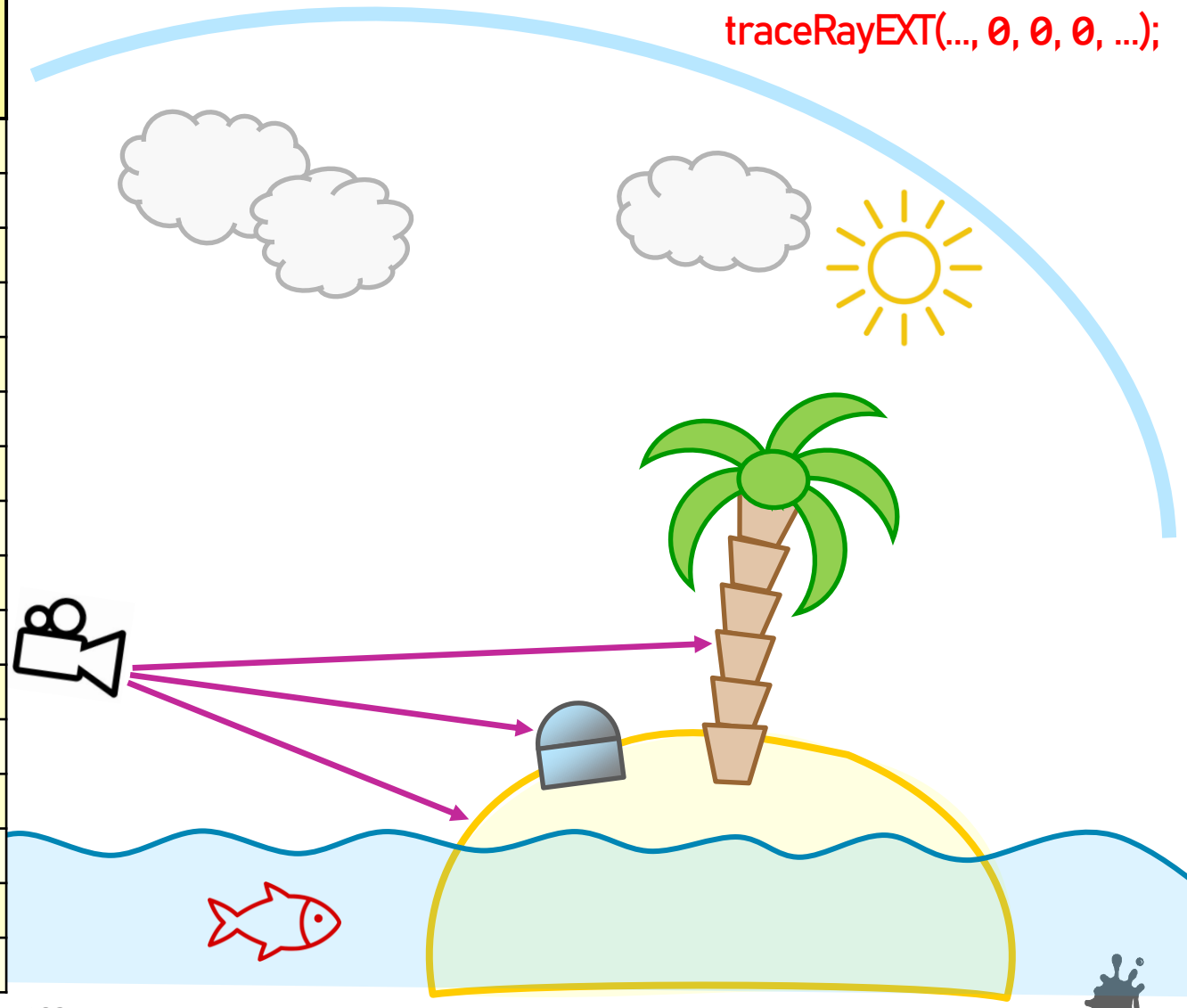


Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

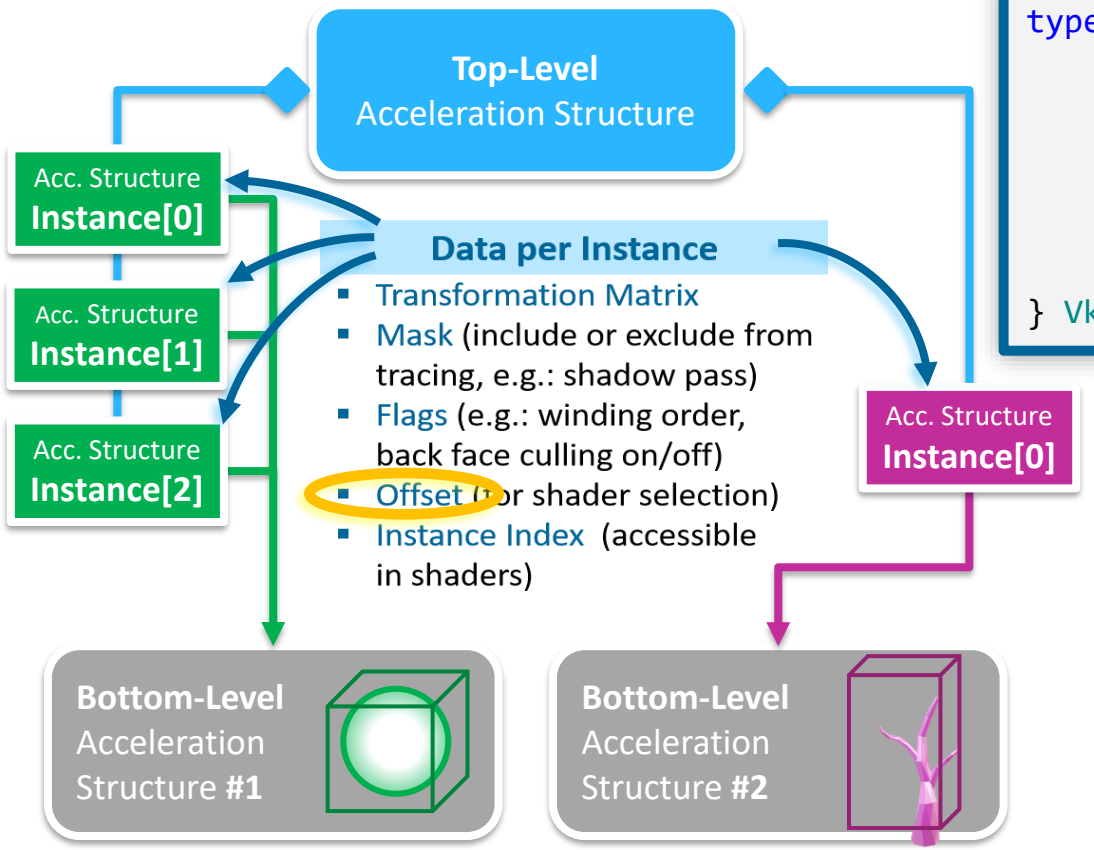
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

traceRayEXT(..., 0, 0, 0, ...);

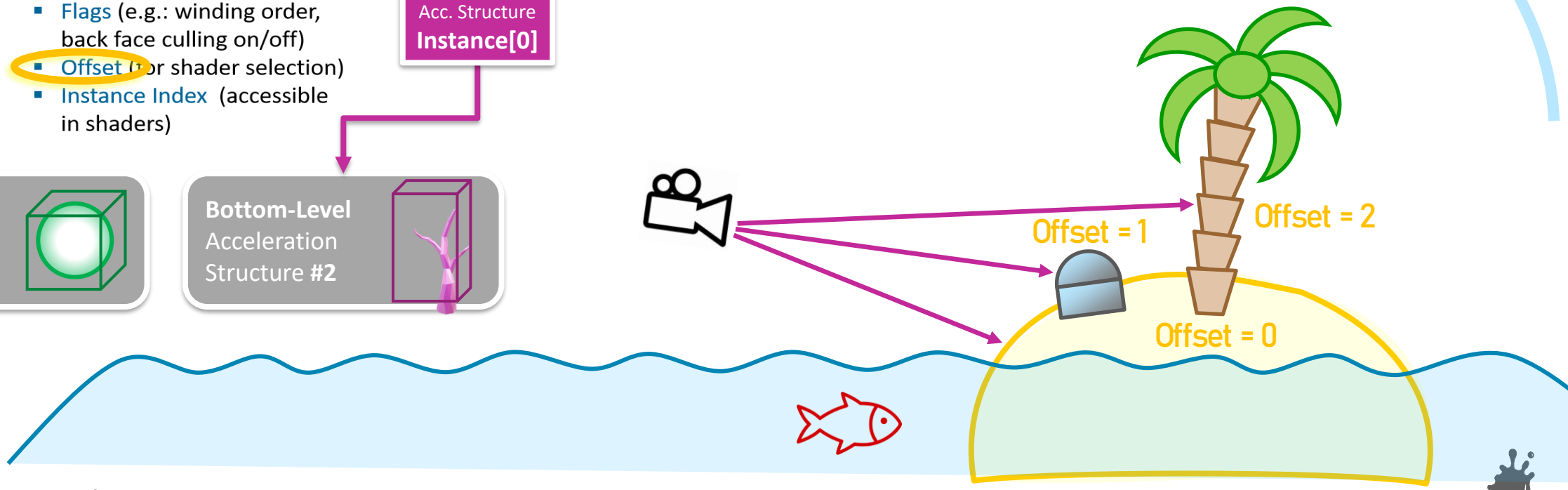


Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;



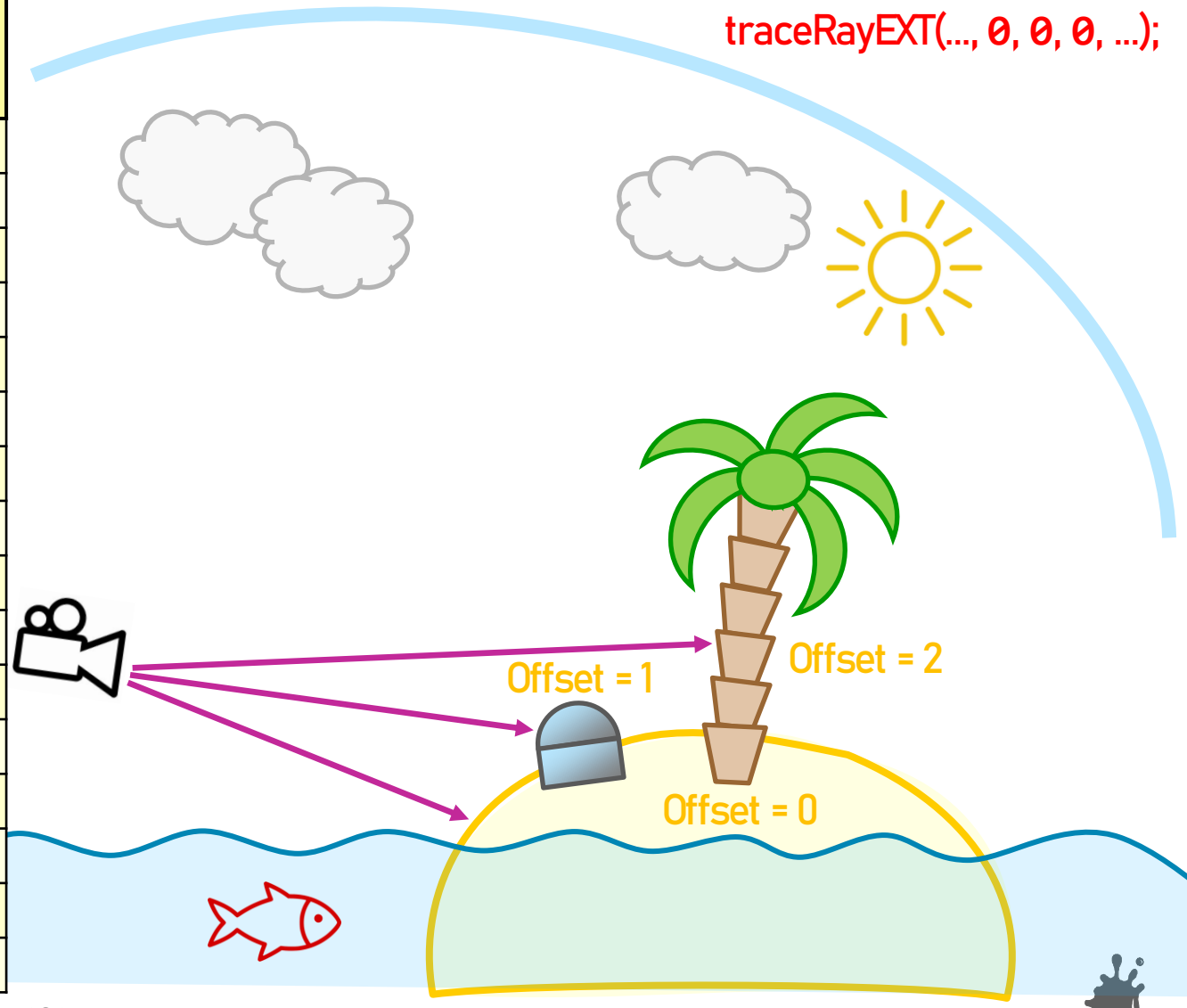

```
typedef struct VkAccelerationStructureInstanceKHR {
    VkTransformMatrixKHR          transform;
    uint32_t                      instanceCustomIndex:24;
    uint32_t                      mask:8;
    uint32_t                      instanceShaderBindingTableRecordOffset:24;
    VkGeometryInstanceFlagsKHR    flags:8;
    uint64_t                      accelerationStructureReference;
} VkAccelerationStructureInstanceKHR;
```



Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

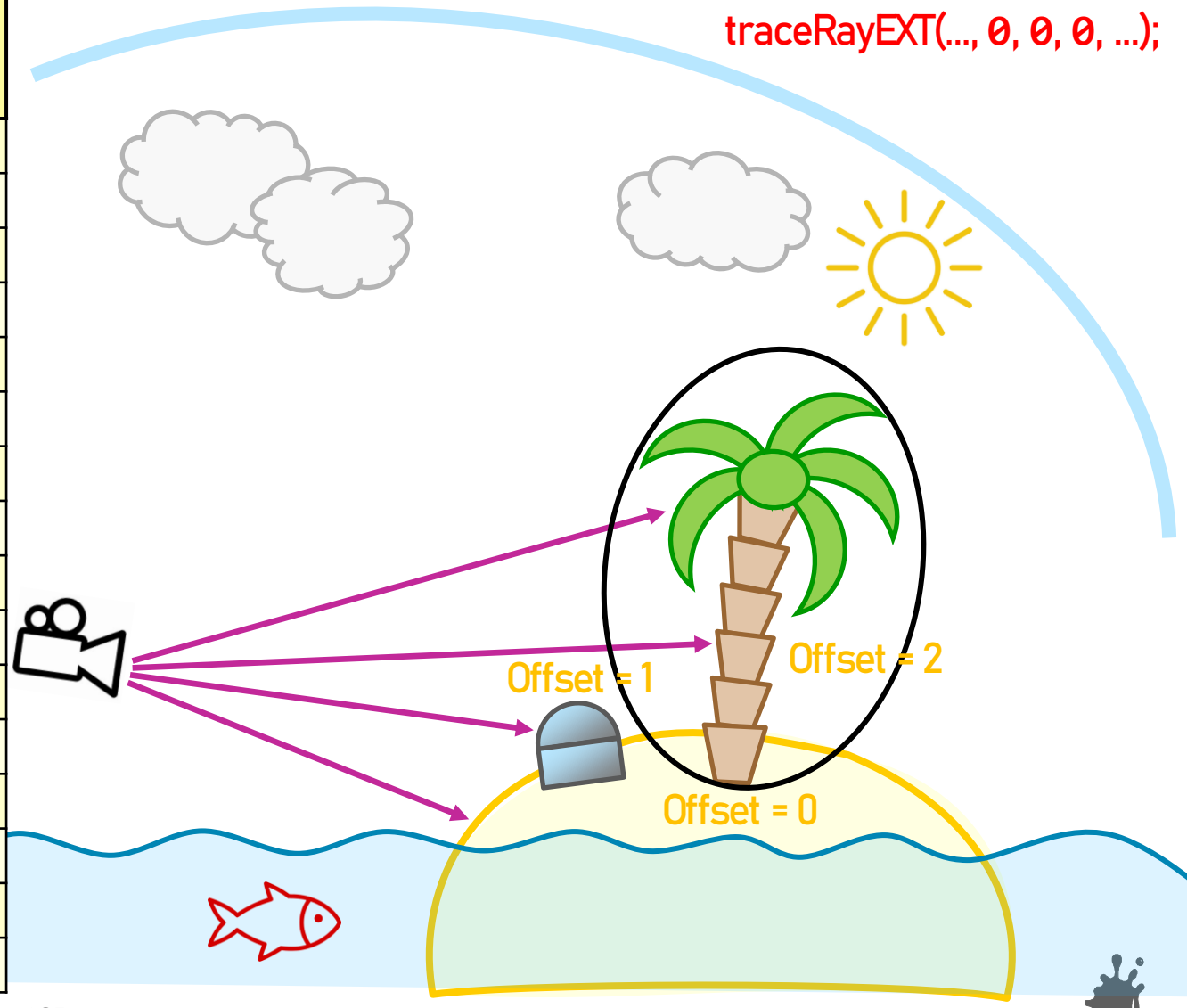
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			



Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

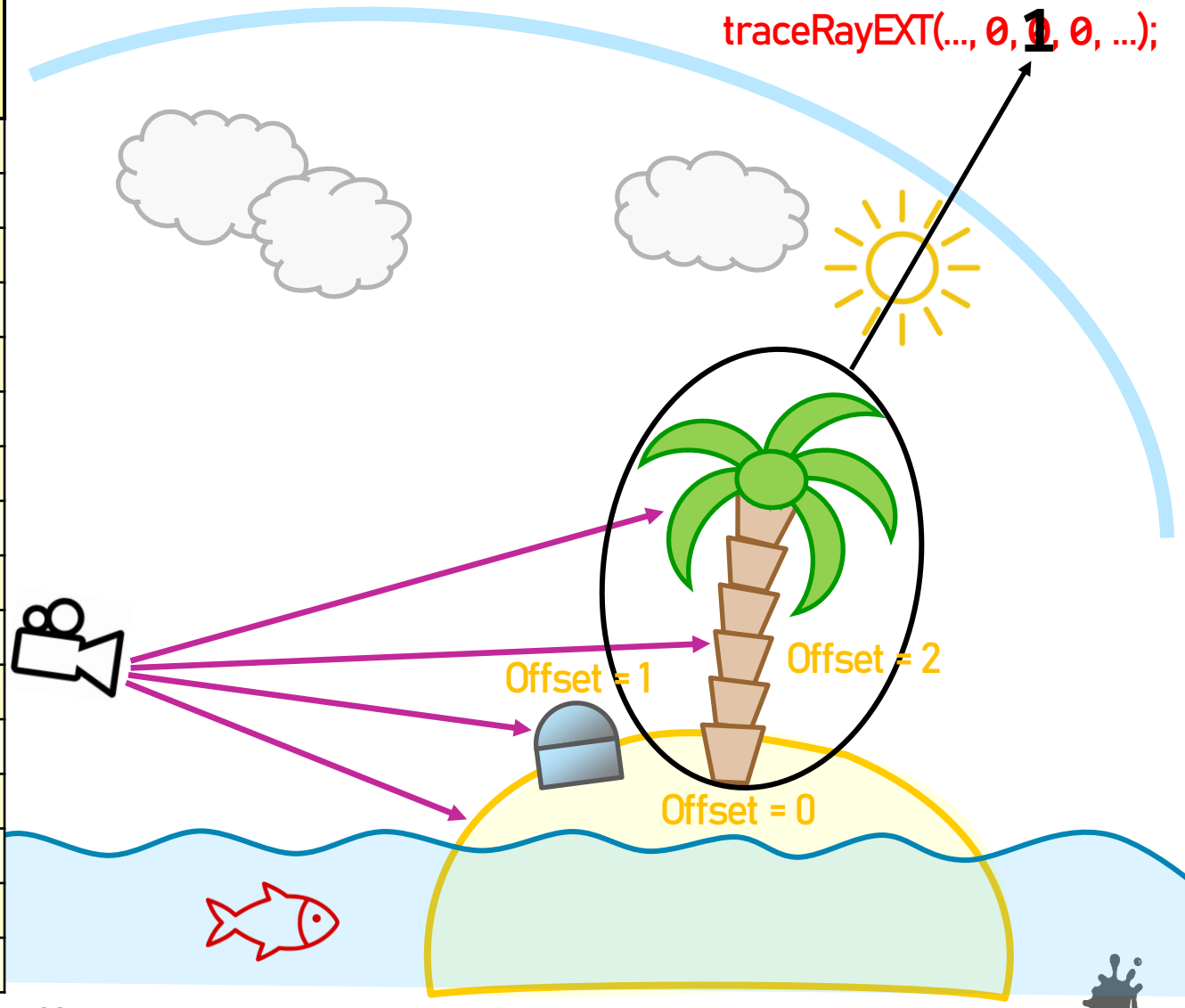
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			



Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			



Indexing for Hit Groups

- Computing a Hit Shader Binding Table Record Address:

traceRayEXT(..., 0, 1, 0, ...);

```
hitGroupRecordAddress =
    start + stride * ( offset
                     + sbtRecordOffset
                     + (geometryIndex * sbtRecordStride)
                     )
```

where:

- `start` = `VkStridedDeviceAddressRegionKHR::deviceAddress` parameter of vkCmdTraceRaysKHR
- `stride` = `VkStridedDeviceAddressRegionKHR::stride` parameter of vkCmdTraceRaysKHR
- `offset` = `VkAccelerationStructureInstanceKHR::instanceShaderBindingTableRecordOffset` member, passed through acceleration structure instance data for top-level acceleration structure build
- `geometryIndex` = index of geometry in Bottom-Level Acceleration Structure (0,1,2,3,..)
- `sbtRecordOffset` = offset parameter of traceRayEXT GLSL function (index-offset, not byte-offset)
- `sbtRecordStride` = stride parameter of traceRayEXT GLSL function (index-stride, not byte-stride)

Indexing for Hit Groups

- Computing a H

```
hitGroupReco
    start +
```

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;

// ...
```

where:

- start = VkStri
- stride = VkStr
- offset = VkAccelerationStructureInstanceKHR::instanceShaderBindingTableRecordOffset member, passed through acceleration structure instance data for top-level acceleration structure build
- geometryIndex = index of geometry in Bottom-Level Acceleration Structure (0,1,2,3,..)
- sbtRecordOffset = offset parameter of traceRayEXT GLSL function (index-offset, not byte-offset)
- sbtRecordStride = stride parameter of traceRayEXT GLSL function (index-stride, not byte-stride)

Indexing for Hit Groups

- Computing a H

```
hitGroupReco
    start +
```

where:

- start = VkStri
- stride = VkStr
- offset = VkAcc
  passe
- geometryInde
- sbtRecordOff
- sbtRecordStr

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;

// ...

VkAccelerationStructureBuildGeometryInfoKHR geometryInfos = {};
geometryInfos.sType =
              VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_BUILD_GEOMETRY_INFO_KHR;
geometryInfos.type = VK_ACCELERATION_STRUCTURE_TYPE_BOTTOM_LEVEL_KHR;
geometryInfos.flags = VK_BUILD_ACCELERATION_STRUCTURE_PREFER_FAST_TRACE_BIT_KHR;
geometryInfos.geometryCount = 1;
geometryInfos.pGeometries = &triangleGeometry;
```

Indexing for Hit Groups

- Computing a H

```
hitGroupReco
    start +
```

where:

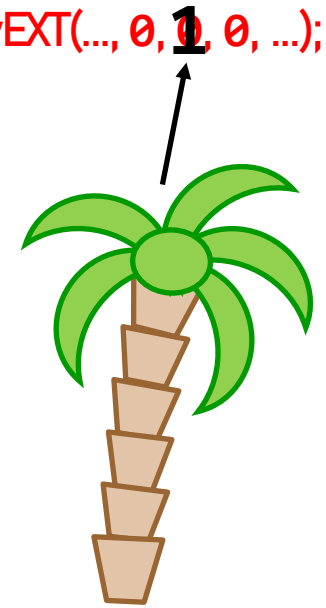
- start = VkStri
- stride = VkStr
- offset = VkAcc
  passe
- geometryInde
- sbtRecordOff
- sbtRecordStr

```
VkAccelerationStructureGeometryKHR triangleGeometry = {};
triangleGeometry.sType = VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_KHR;
triangleGeometry.flags = VK_GEOMETRY_OPAQUE_BIT_KHR;
triangleGeometry.geometryType = VK_GEOMETRY_TYPE_TRIANGLES_KHR;
triangleGeometry.geometry.triangles.sType =
   VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_GEOMETRY_TRIANGLES_DATA_KHR;
triangleGeometry.geometry.triangles.vertexData = vertexBufferDeviceAddress;
triangleGeometry.geometry.triangles.indexData = indexBufferDeviceAddress;
triangleGeometry.geometry.triangles.vertexFormat = VK_FORMAT_R32G32B32_SFLOAT;
triangleGeometry.geometry.triangles.maxVertex = 100;
triangleGeometry.geometry.triangles.vertexStride = sizeof(MyVertexData);
triangleGeometry.geometry.triangles.indexType = VK_INDEX_TYPE_UINT32;

// ...

VkAccelerationStructureBuildGeometryInfoKHR geometryInfos = {};
geometryInfos.sType =
              VK_STRUCTURE_TYPE_ACCELERATION_STRUCTURE_BUILD_GEOMETRY_INFO_KHR;
geometryInfos.type = VK_ACCELERATION_STRUCTURE_TYPE_BOTTOM_LEVEL_KHR;
geometryInfos.flags = VK_BUILD_ACCELERATION_STRUCTURE_PREFER_FAST_TRACE_BIT_KHR;
geometryInfos.geometryCount = 100;
geometryInfos.pGeometries = ...;
```

Indexing for Hit Groups

- Computing a Hit Shader Binding Table Record Address:

```
hitGroupRecordAddress =
    start + stride * ( offset
                     + sbtRecordOffset
                     + (geometryIndex * sbtRecordStride)
                     )
```



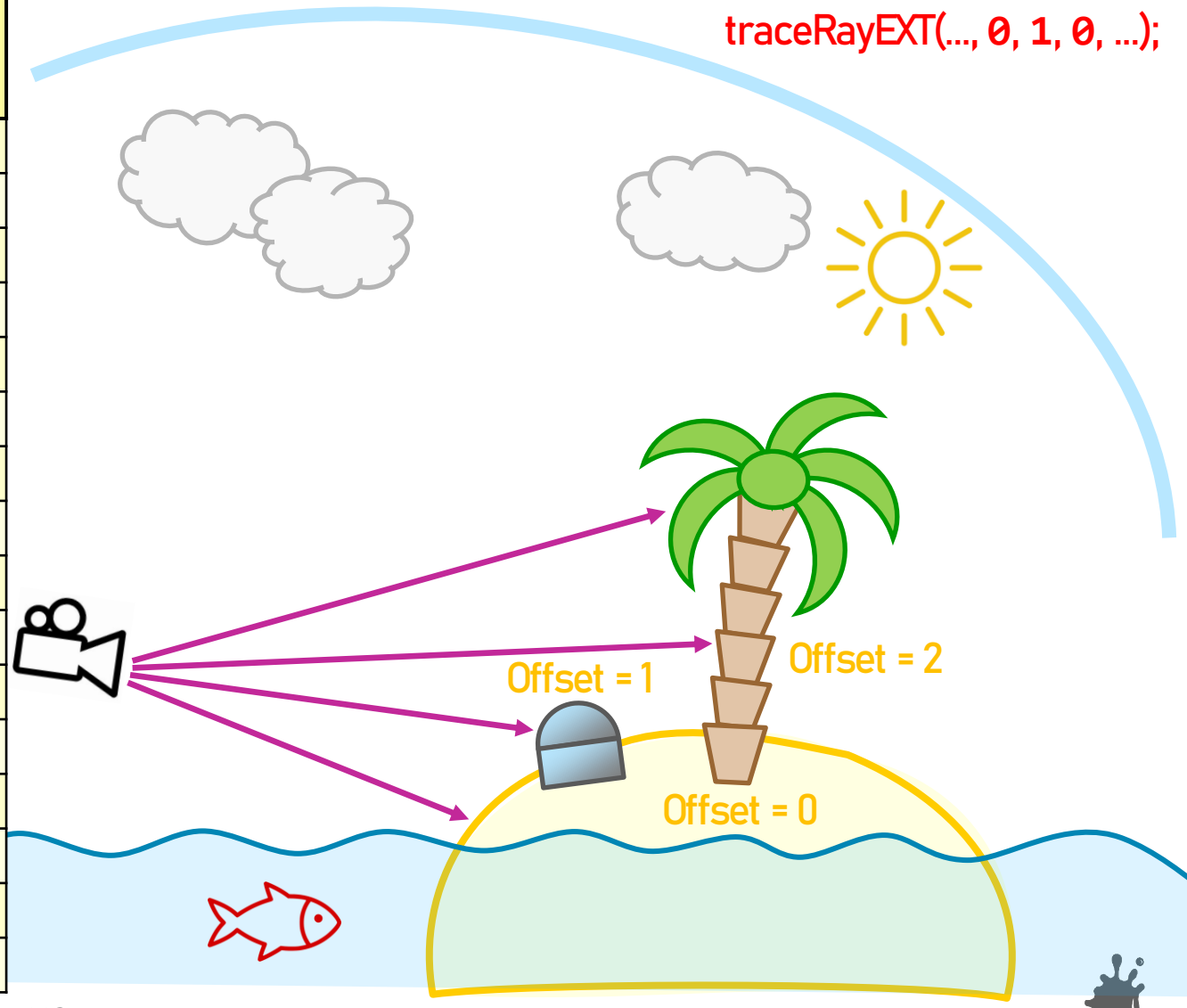
where:

- `start` = `VkStridedDeviceAddressRegionKHR::deviceAddress` parameter of vkCmdTraceRaysKHR
- `stride` = `VkStridedDeviceAddressRegionKHR::stride` parameter of vkCmdTraceRaysKHR
- `offset` = `VkAccelerationStructureInstanceKHR::instanceShaderBindingTableRecordOffset` member, passed through acceleration structure instance data for top-level acceleration structure build
- `geometryIndex` = index of geometry in Bottom-Level Acceleration Structure (0,1,2,3,..)
- `sbtRecordOffset` = offset parameter of traceRayEXT GLSL function (index-offset, not byte-offset)
- `sbtRecordStride` = stride parameter of traceRayEXT GLSL function (index-stride, not byte-stride)

Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

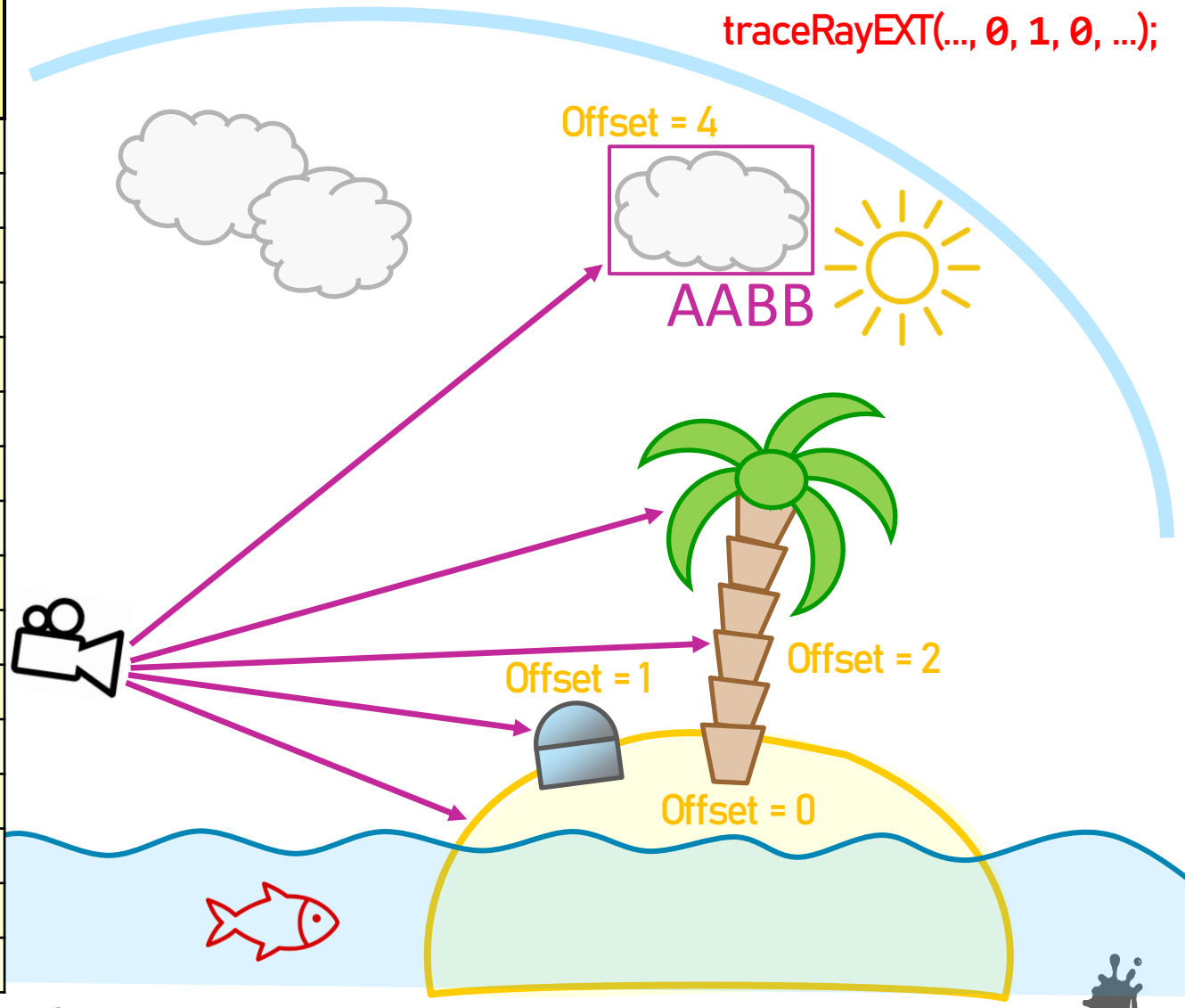
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			



Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

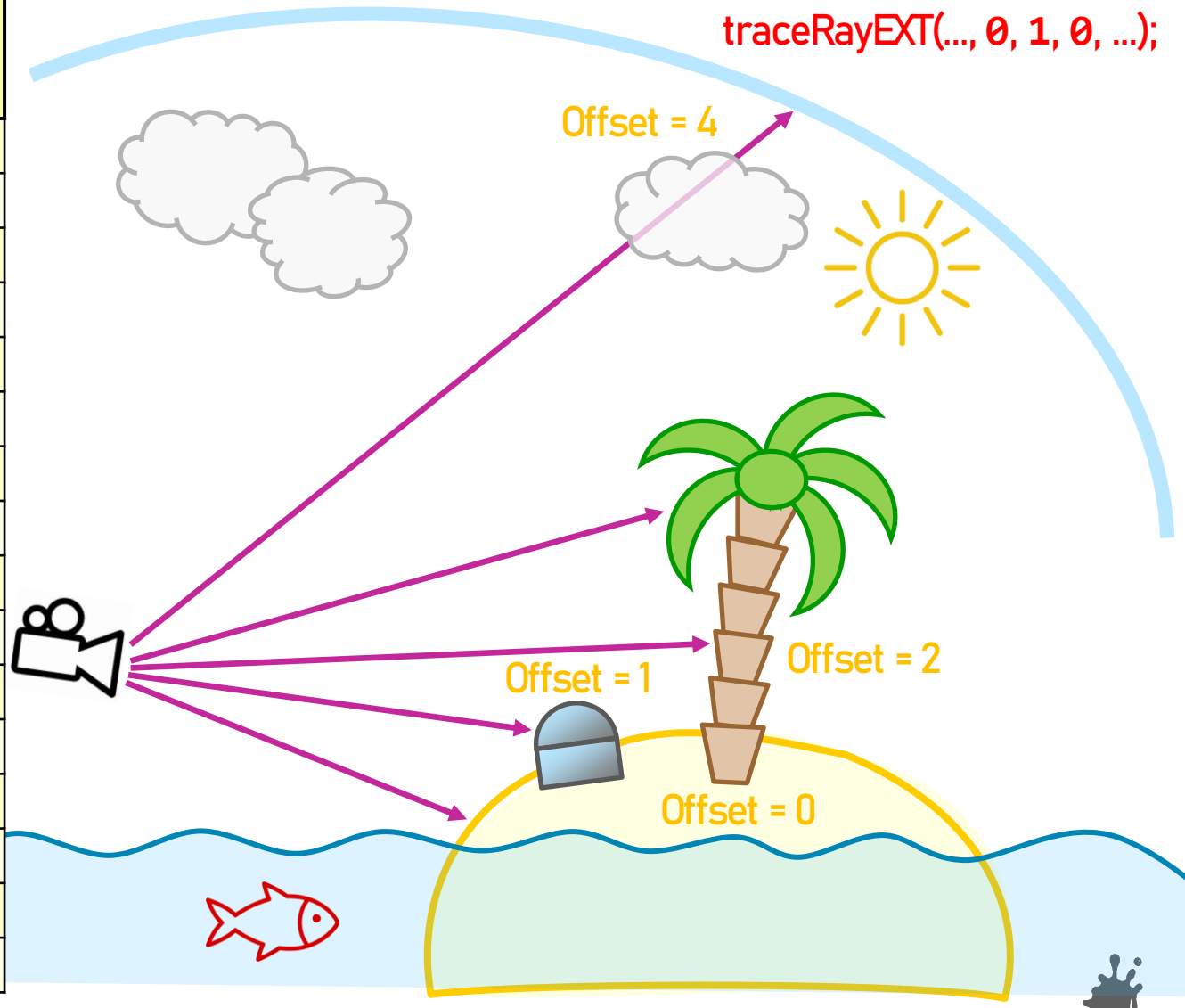
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			



Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

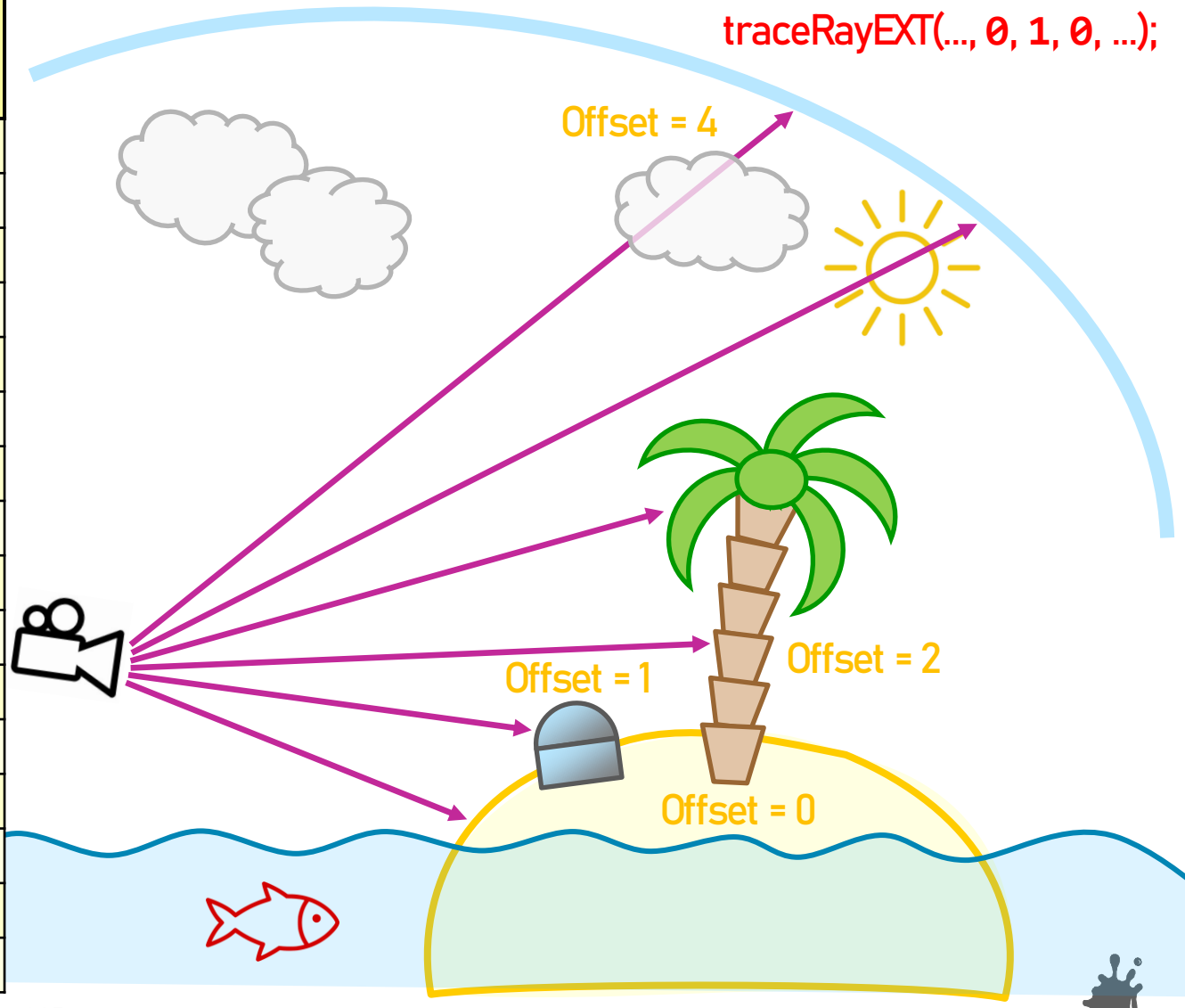
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			



Shader Binding Table Example

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6			
7			
8			
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			

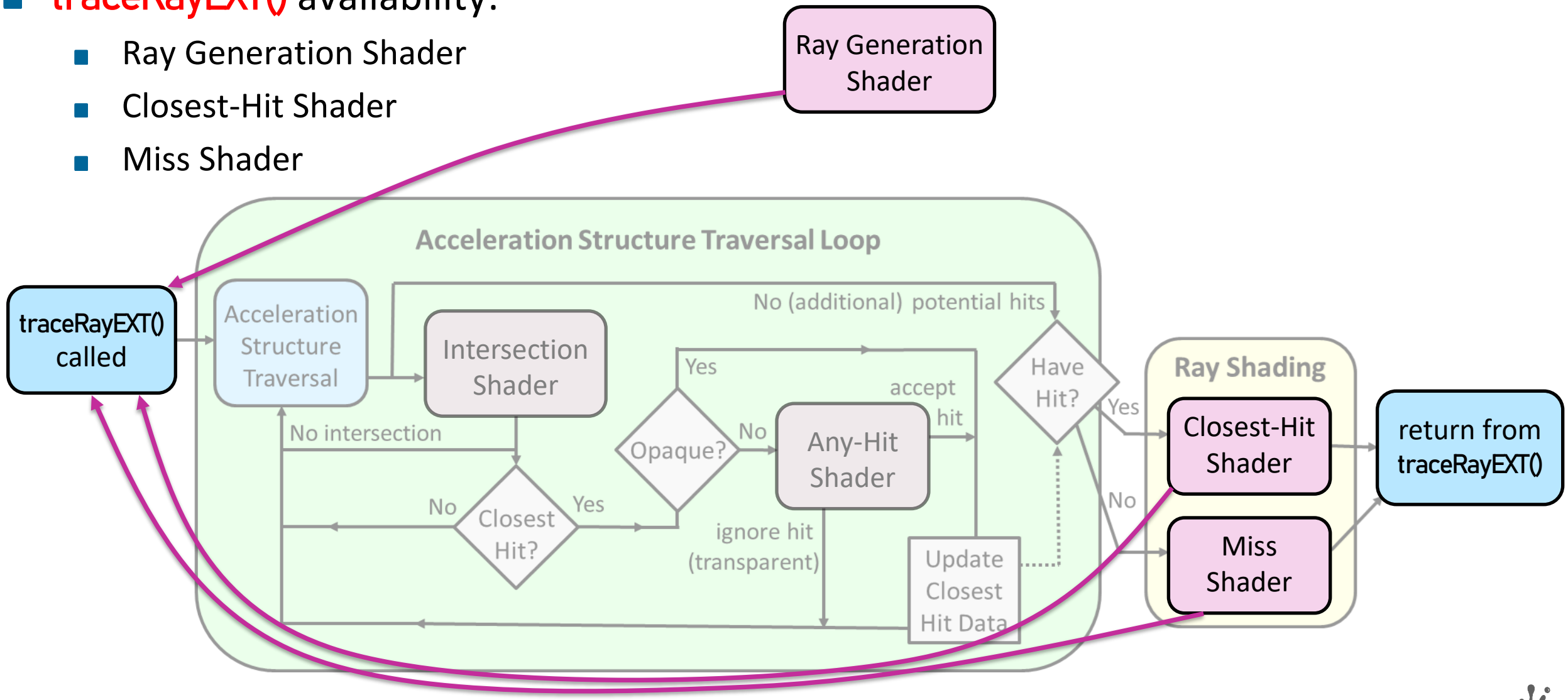


Recursive Rays

- So far:
  - Only evaluating first hit
  - Actually, this is "Ray Casting"
- Now:
  - Recursive rays
  - I.e., spawn more rays via traceRayEXT() within ray tracing pipeline traversal

Recursive Rays

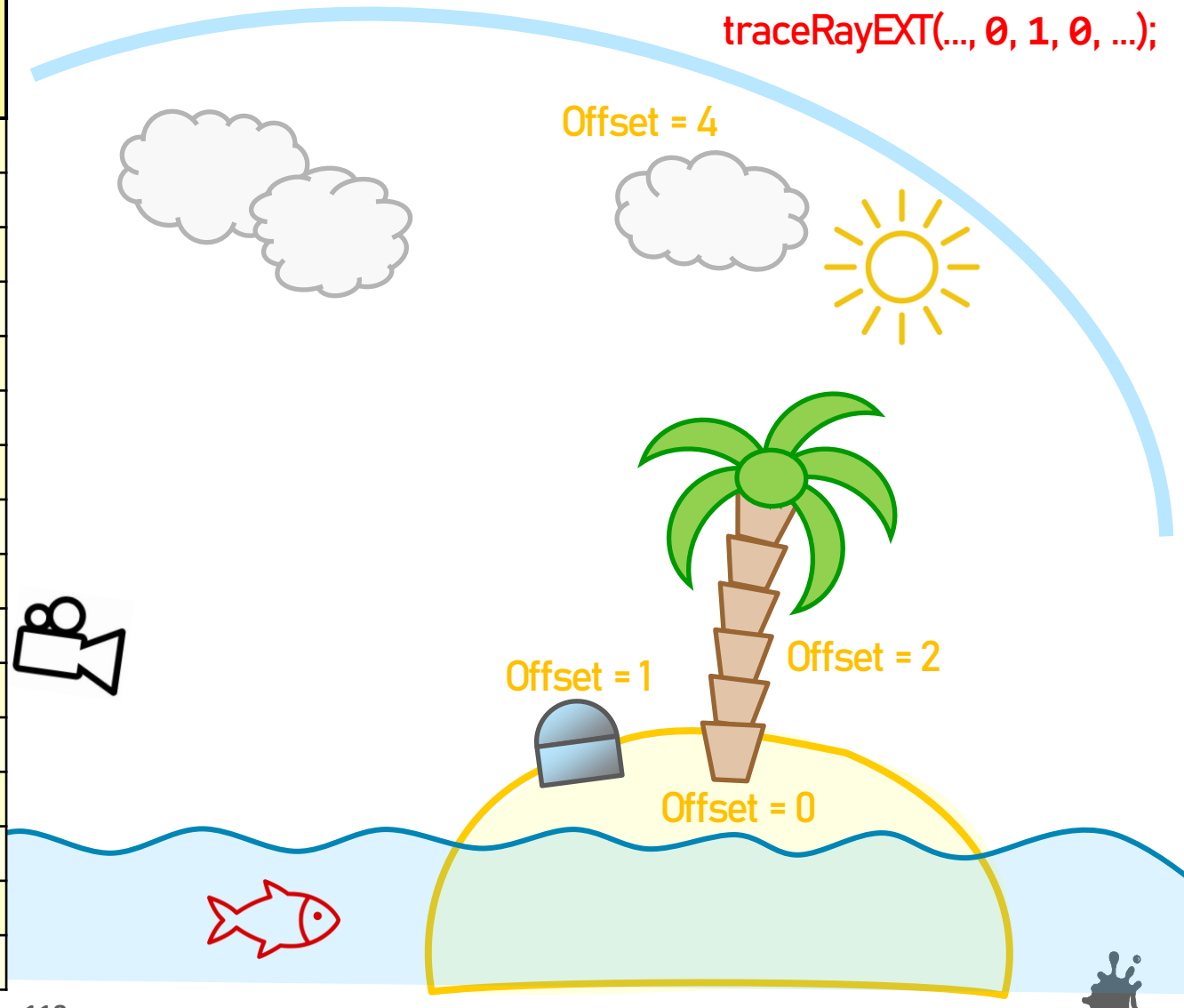
- traceRayEXT() availability:
  - Ray Generation Shader
  - Closest-Hit Shader
  - Miss Shader



Shader Binding Table Example + Reflection/Refraction

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

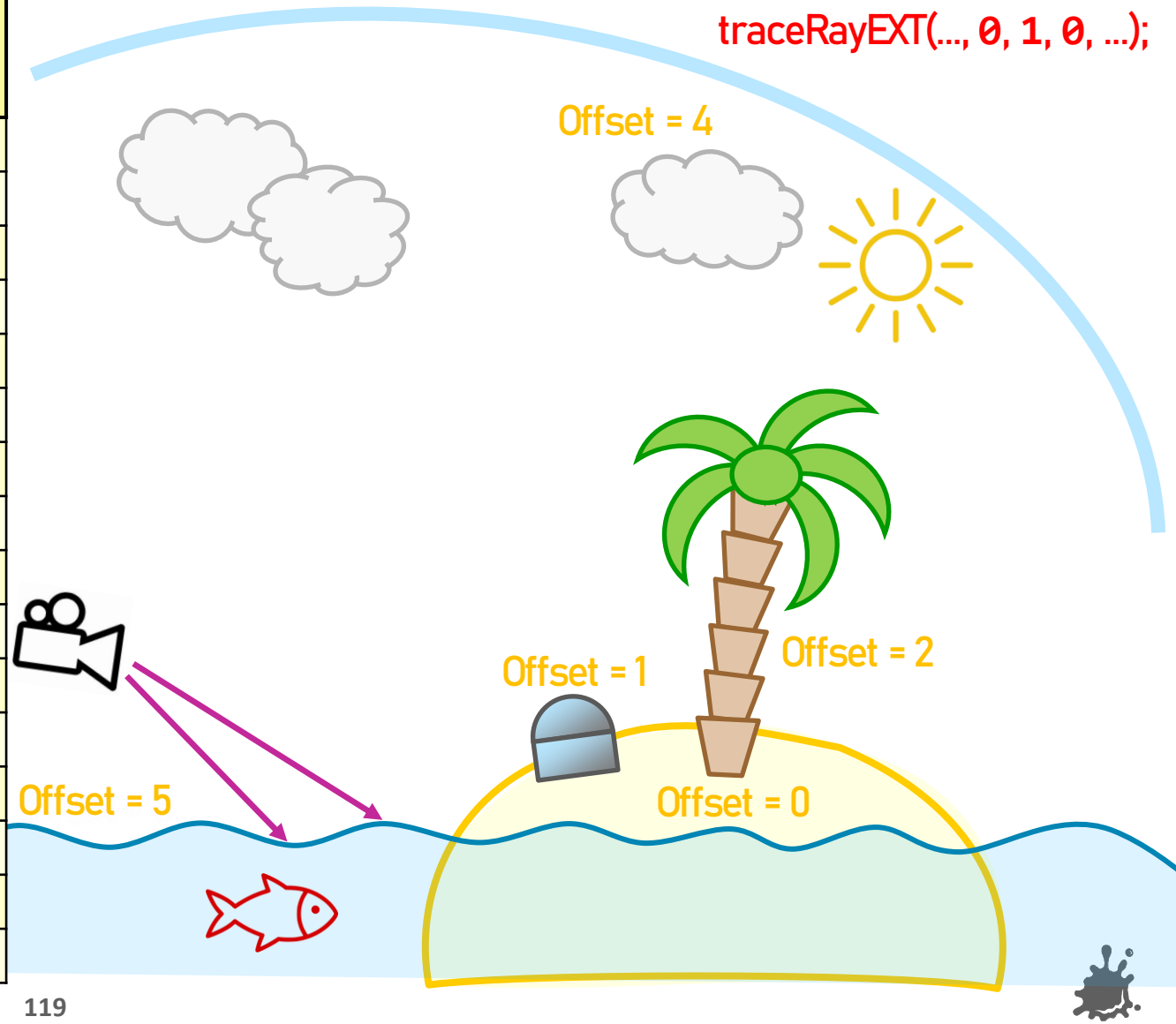
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6			
7			
8			
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			



Shader Binding Table Example + Reflection/Refraction

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7			
8			
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			

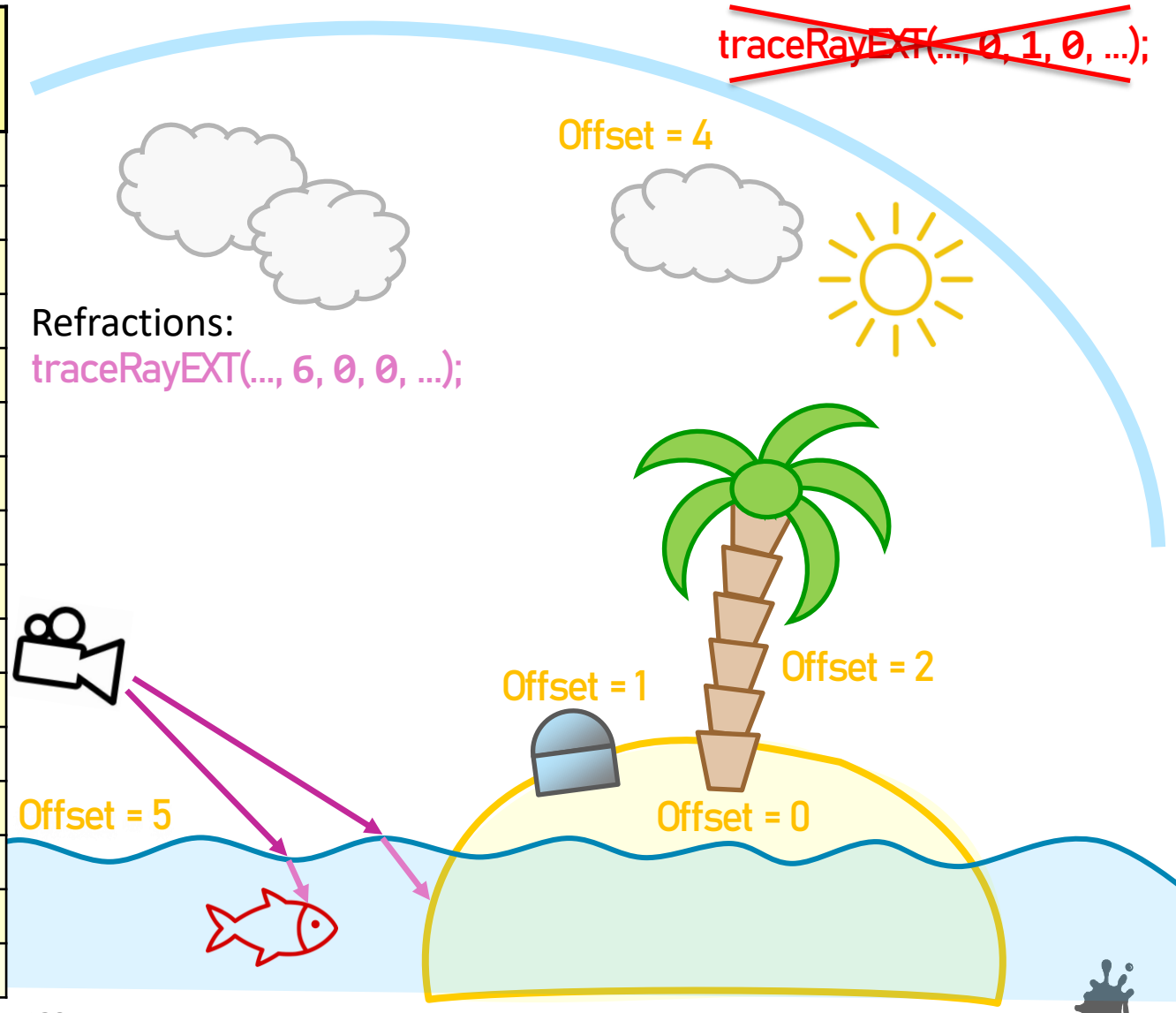


Shader Binding Table Example + Reflection/Refraction

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			

~~traceRayEXT(..., 0, 1, 0, ...);~~



Indexing for Hit Groups

- Computing a Hit Shader Binding Table Record Address:

~~traceRayEXT(..., 0, 1, 0, ...);~~

traceRayEXT(..., 6, 0, 0, ...);

```
hitGroupRecordAddress =
    start + stride * ( offset
                     + sbtRecordOffset
                     + (geometryIndex * sbtRecordStride)
                     )
```

where:

- `start` = `VkStridedDeviceAddressRegionKHR::deviceAddress` parameter of vkCmdTraceRaysKHR
- `stride` = `VkStridedDeviceAddressRegionKHR::stride` parameter of vkCmdTraceRaysKHR
- `offset` = `VkAccelerationStructureInstanceKHR::instanceShaderBindingTableRecordOffset` member, passed through acceleration structure instance data for top-level acceleration structure build
- `geometryIndex` = index of geometry in Bottom-Level Acceleration Structure (0,1,2,3,..)
- `sbtRecordOffset` = offset parameter of traceRayEXT GLSL function (index-offset, not byte-offset)
- `sbtRecordStride` = stride parameter of traceRayEXT GLSL function (index-stride, not byte-stride)

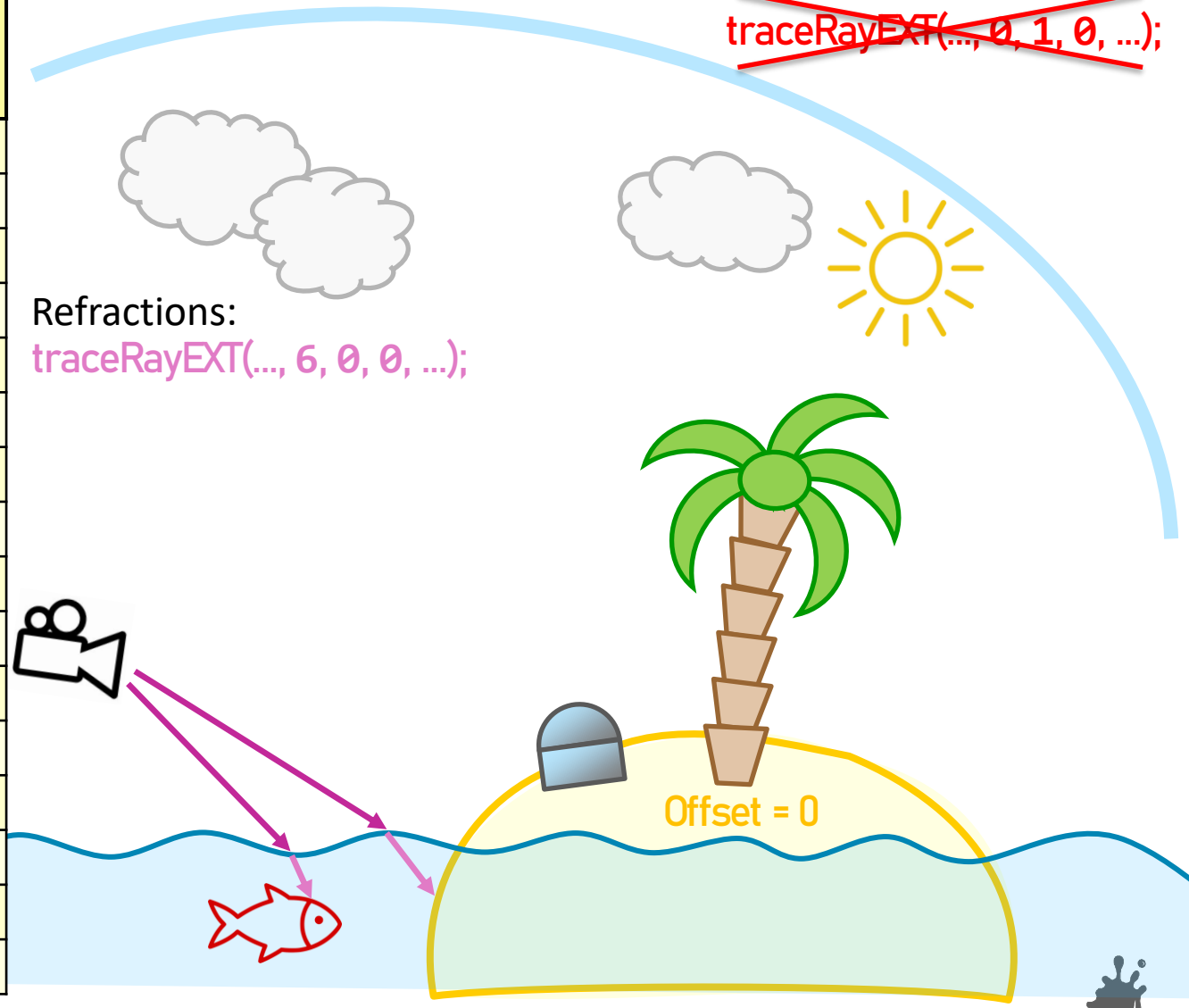
Shader Binding Table Example + Reflection/Refraction

`vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;`

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			

~~traceRayEXT(..., 0, 1, 0, ...);~~

Refractions:
`traceRayEXT(..., 6, 0, 0, ...);`

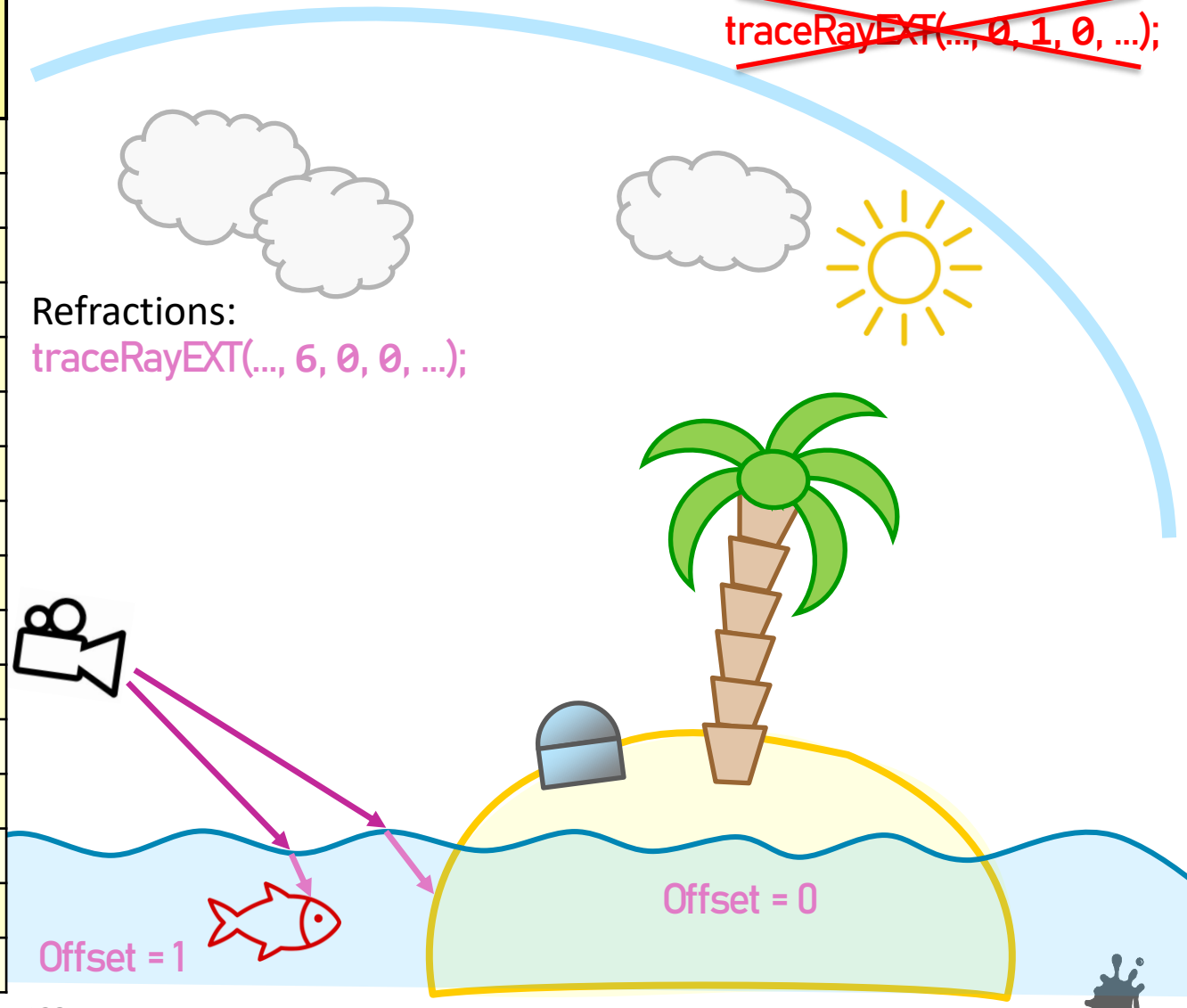


Shader Binding Table Example + Reflection/Refraction

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

~~traceRayEXT(..., 0, 1, 0, ...);~~

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			

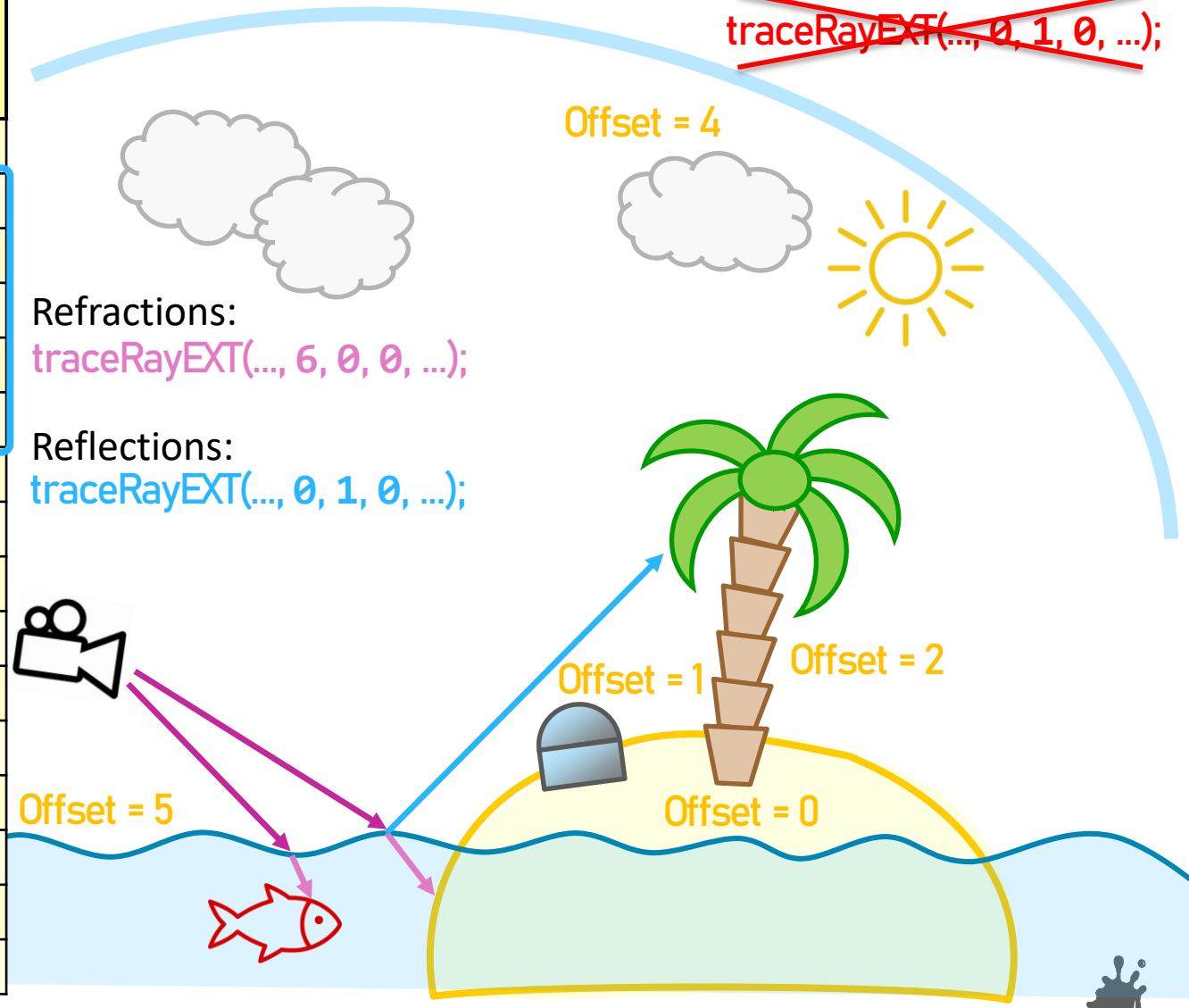


Shader Binding Table Example + Reflection/Refraction

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			

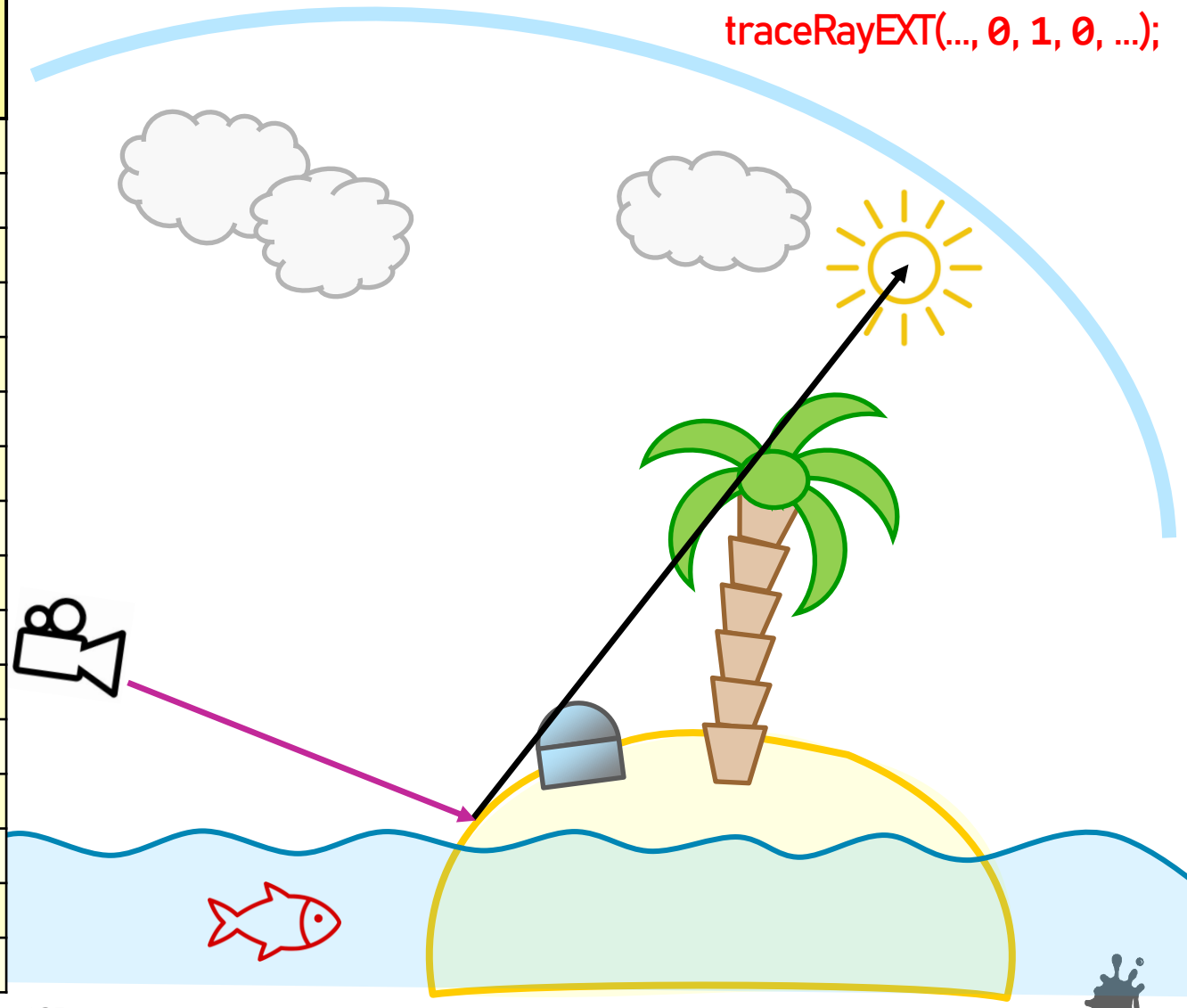
~~traceRayEXT(..., 0, 1, 0, ...);~~



Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

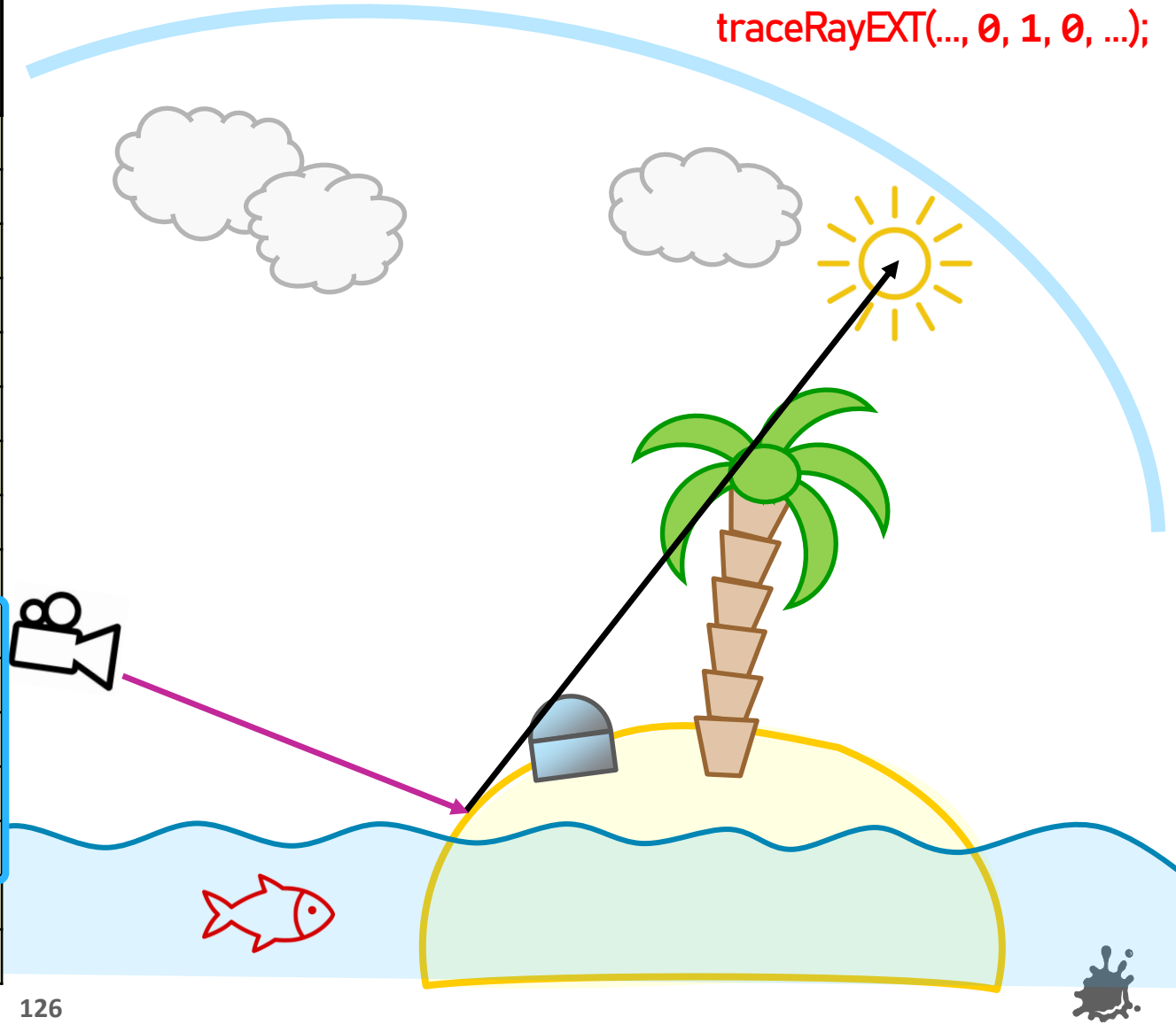
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9			
10			
11			
12			
13			
14	*Miss Shader:* SetSkyboxColor		
15			



Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

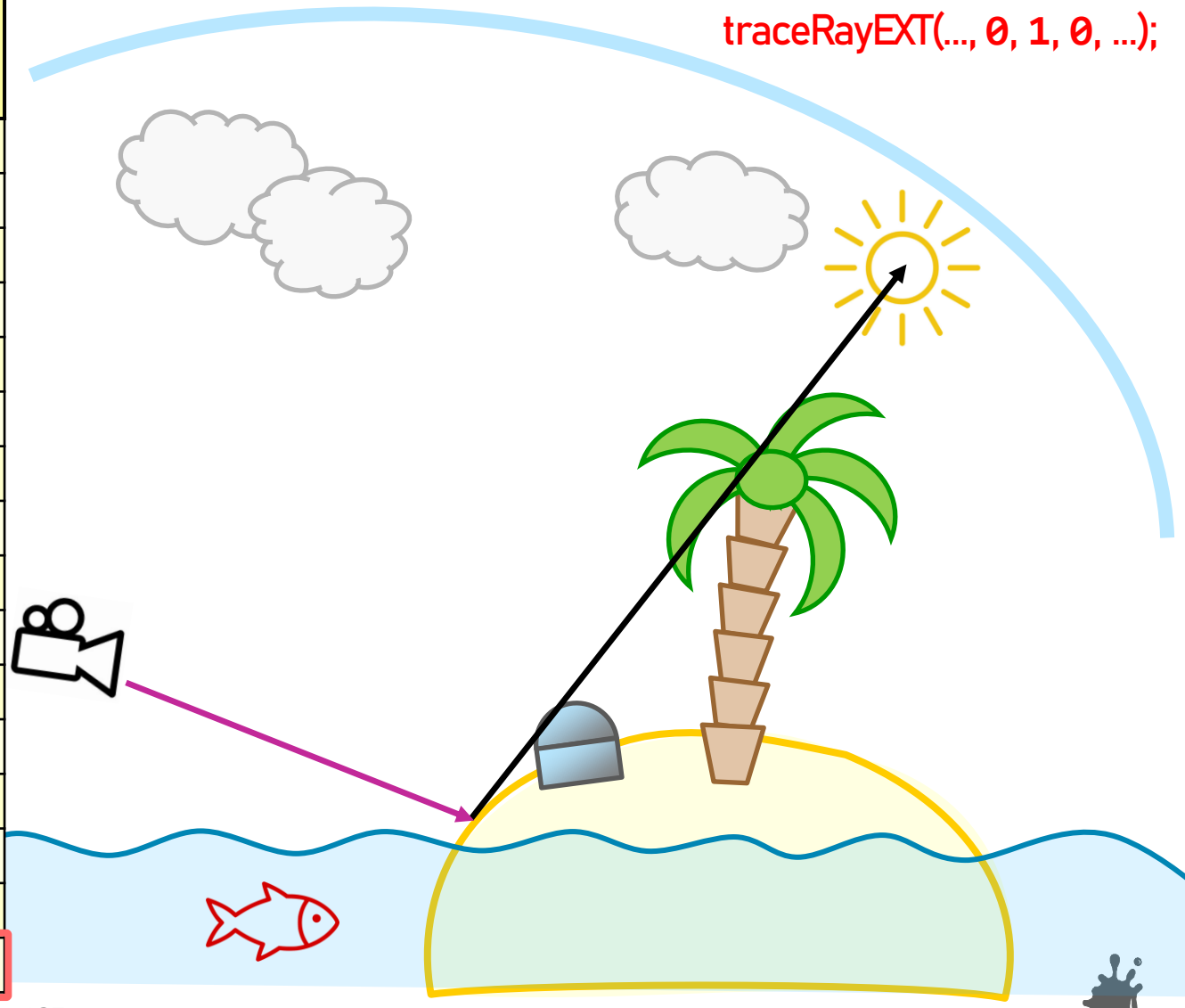
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9	AddShadow	-	Triangles (built-in)
10	AddShadow	-	Triangles (built-in)
11	AddShadow	-	Triangles (built-in)
12	AddShadow	CheckAlphaTexture	Triangles (built-in)
13	-	AddSomeShadow	ProceduralClouds
14	*Miss Shader:* SetSkyboxColor		
15	*Miss Shader:* NoShadowAdded		



Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

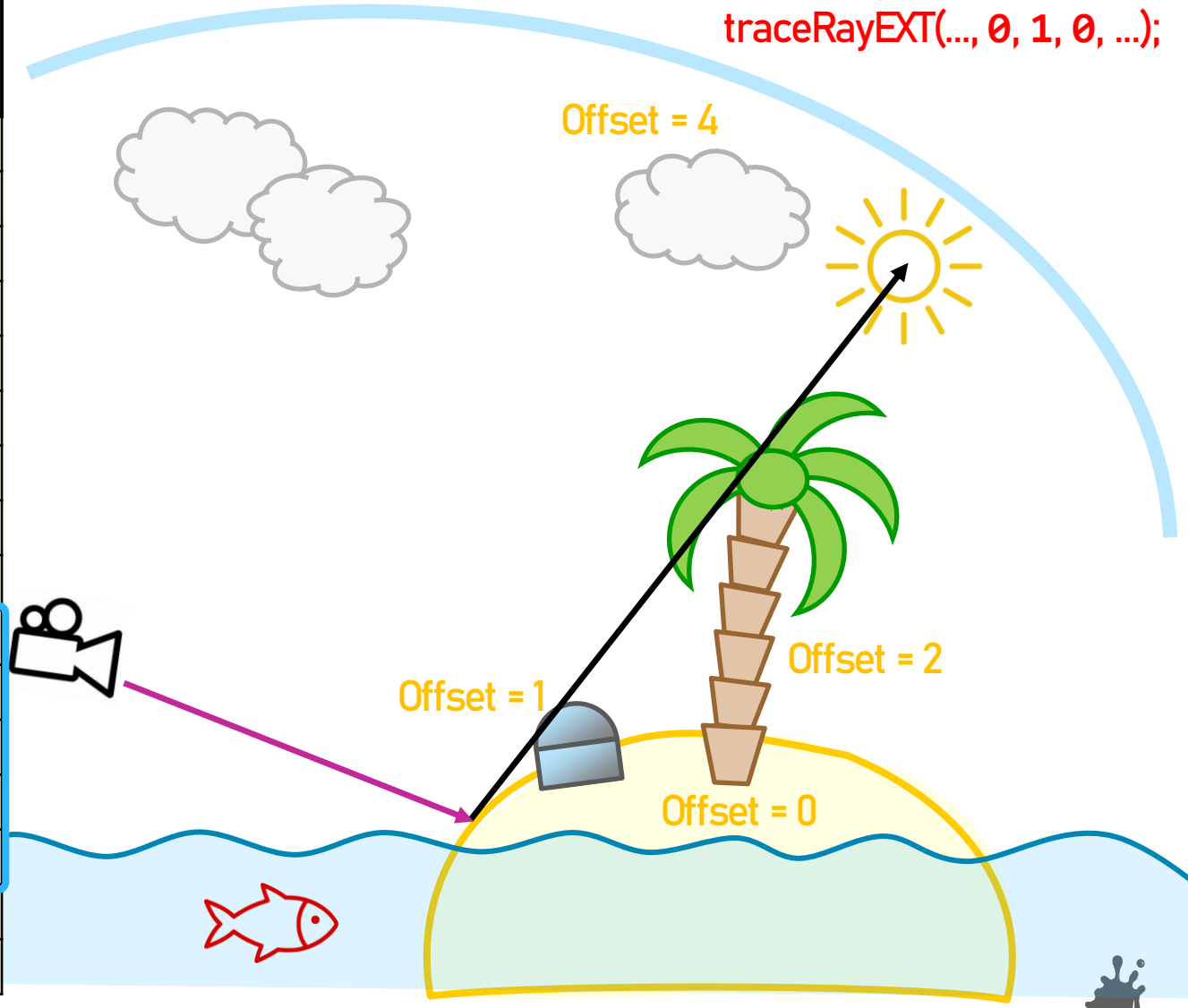
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9	AddShadow	-	Triangles (built-in)
10	AddShadow	-	Triangles (built-in)
11	AddShadow	-	Triangles (built-in)
12	AddShadow	CheckAlphaTexture	Triangles (built-in)
13	-	AddSomeShadow	ProceduralClouds
14	*Miss Shader:* SetSkyboxColor		
15	*Miss Shader:* NoShadowAdded		



Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

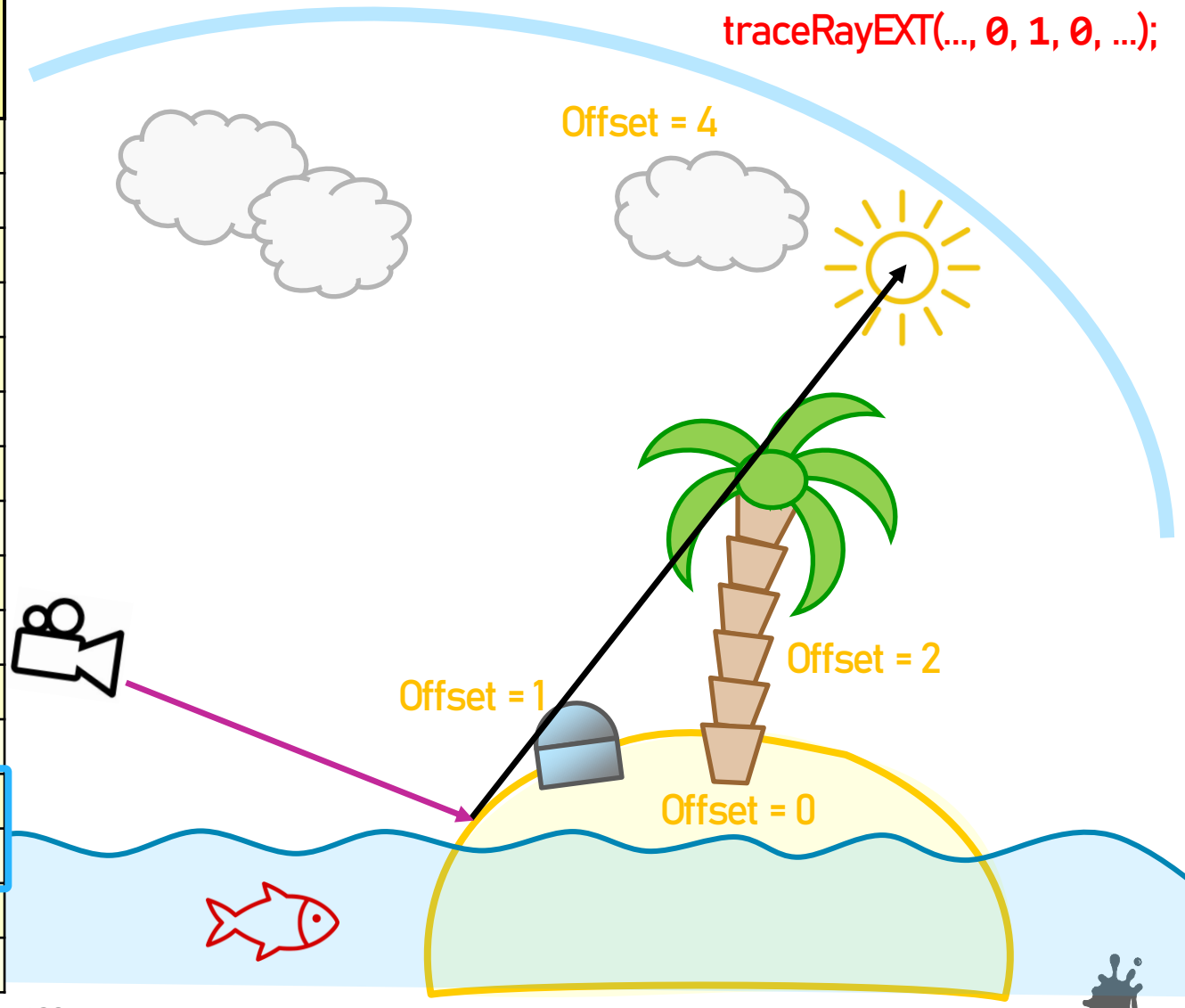
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9	AddShadow	-	Triangles (built-in)
10	AddShadow	-	Triangles (built-in)
11	AddShadow	-	Triangles (built-in)
12	AddShadow	CheckAlphaTexture	Triangles (built-in)
13	-	AddSomeShadow	ProceduralClouds
14	*Miss Shader:* SetSkyboxColor		
15	*Miss Shader:* NoShadowAdded		



Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

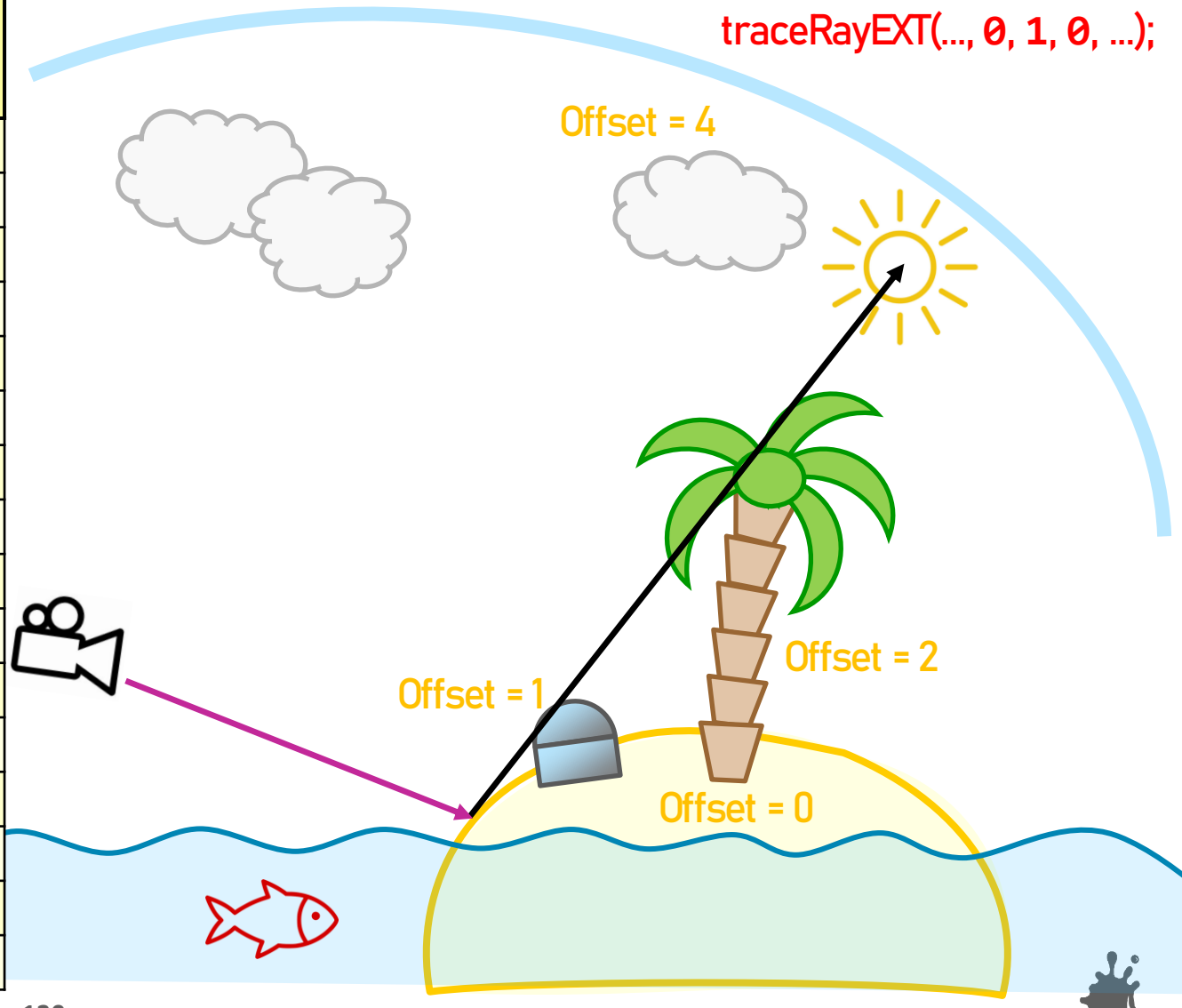
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9	AddShadow	-	Triangles (built-in)
10	AddShadow	-	Triangles (built-in)
11	AddShadow	-	Triangles (built-in)
12	AddShadow	CheckAlphaTexture	Triangles (built-in)
13	-	AddSomeShadow	ProceduralClouds
14	*Miss Shader:* SetSkyboxColor		
15	*Miss Shader:* NoShadowAdded		



Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

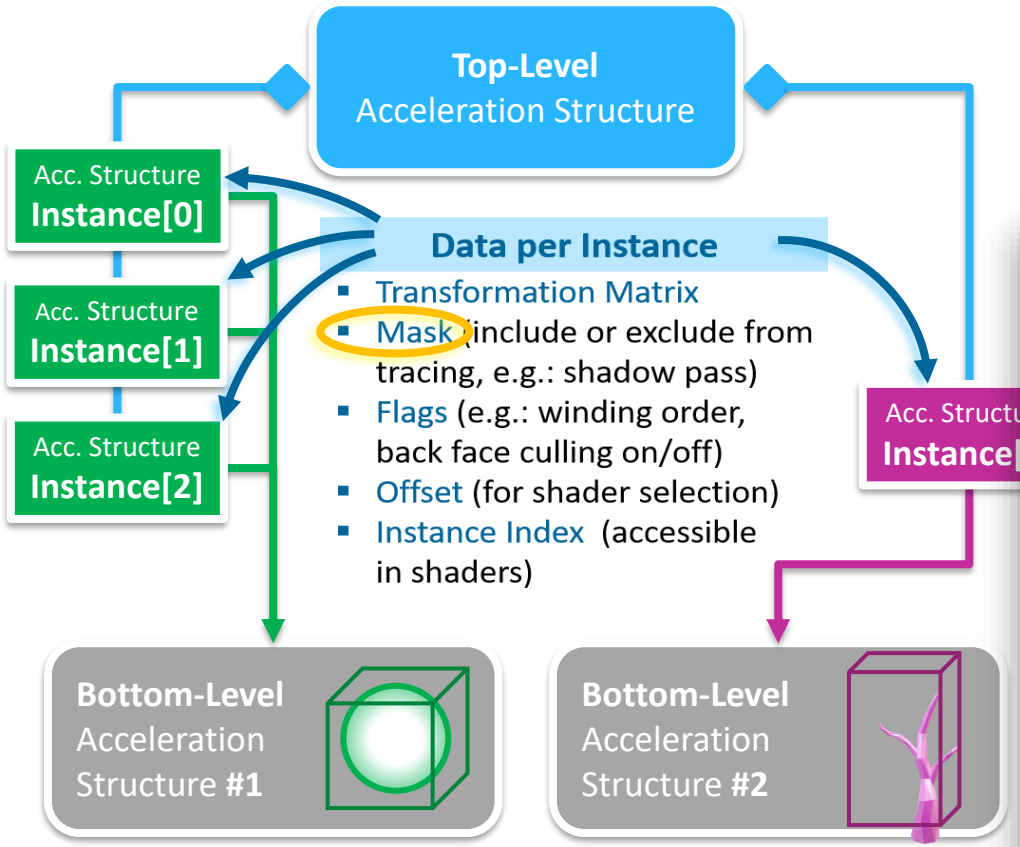
Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9	AddShadow	-	Triangles (built-in)
10	AddShadow	-	Triangles (built-in)
11	AddShadow	-	Triangles (built-in)
12	AddShadow	CheckAlphaTexture	Triangles (built-in)
13	-	AddSomeShadow	ProceduralClouds
14	*Miss Shader:* SetSkyboxColor		
15	*Miss Shader:* NoShadowAdded		



Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

traceRayEXT(..., 0, 1, 0, ...);



Offset = 4

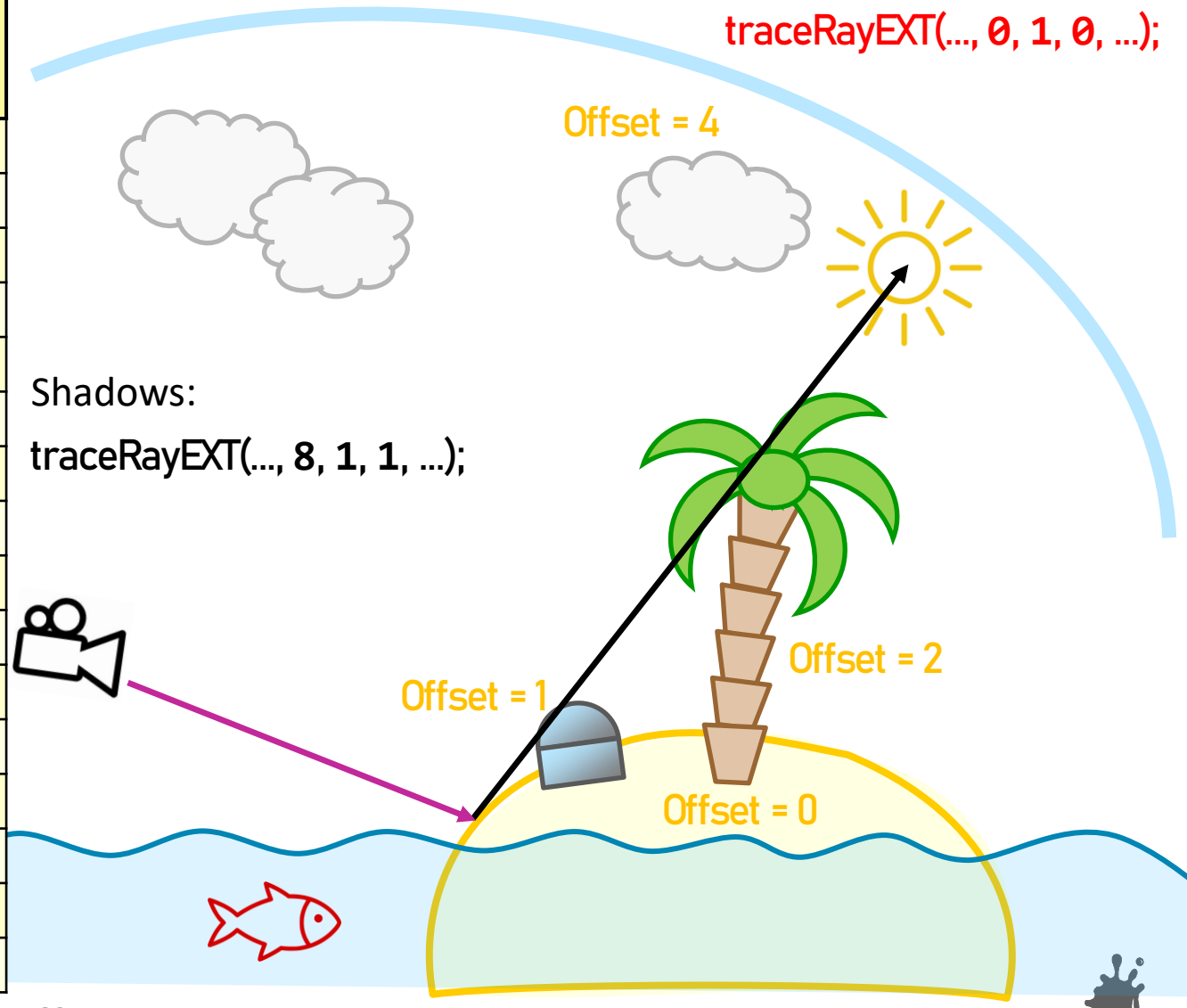
GLSL

```
#version 460
#extension GL_EXT_ray_tracing : require

layout(set = 0, binding = 0) uniform accelerationStructureEXT as;
// ...
void main() {
  // ...
  traceRayEXT(as,
    gl_RayFlagsOpaqueEXT, 0xF0,
    0 /*sbtRecordOffset*/, 0 /*sbtRecordStride*/, 0 /*missIndex*/,
    origin, 0.1, direction, 100.0,
    0 /*location of payload*/);
  // ...
}
```

Shader Binding Table Example + Shadow Rays

vkCmdTraceRaysKHR with rayGenAddr = start + 0 * stride, missAddr = start + 14 * stride, hitGroupsAddr = start + 1 * stride;

Shader Binding Table			
Idx	*[Closest Hit Shader]*	*[Any Hit Shader]*	*[Intersection Shader]*
0	*Ray Generation Shader:* GenerateRaysAndStoreResults		
1	ShadeTerrain	-	Triangles (built-in)
2	ShadeTreasure	-	Triangles (built-in)
3	ShadePalmTreeTrunk	-	Triangles (built-in)
4	ShadePalmTreeLeaves	CheckAlphaTexture	Triangles (built-in)
5	-	HandleClouds	ProceduralClouds
6	ShadeWaterSurface	-	ProceduralWater
7	ShadeTerrainUnderwater	-	Triangles (built-in)
8	ShadeFishUnderwater	-	Triangles (built-in)
9	AddShadow	-	Triangles (built-in)
10	AddShadow	-	Triangles (built-in)
11	AddShadow	-	Triangles (built-in)
12	AddShadow	CheckAlphaTexture	Triangles (built-in)
13	-	AddSomeShadow	ProceduralClouds
14	*Miss Shader:* SetSkyboxColor		
15	*Miss Shader:* NoShadowAdded		



Indexing for Miss Shaders

- Computing a Miss Shader Binding Table Record Address:

```
missRecordAddress = start + stride * missIndex
```

where:

- `start` = `VkStridedDeviceAddressRegionKHR::deviceAddress` parameter of vkCmdTraceRaysKHR
- `stride` = `VkStridedDeviceAddressRegionKHR::stride` parameter of vkCmdTraceRaysKHR
- `missIndex` = parameter of traceRayEXT GLSL function (index-offset, not byte-offset)

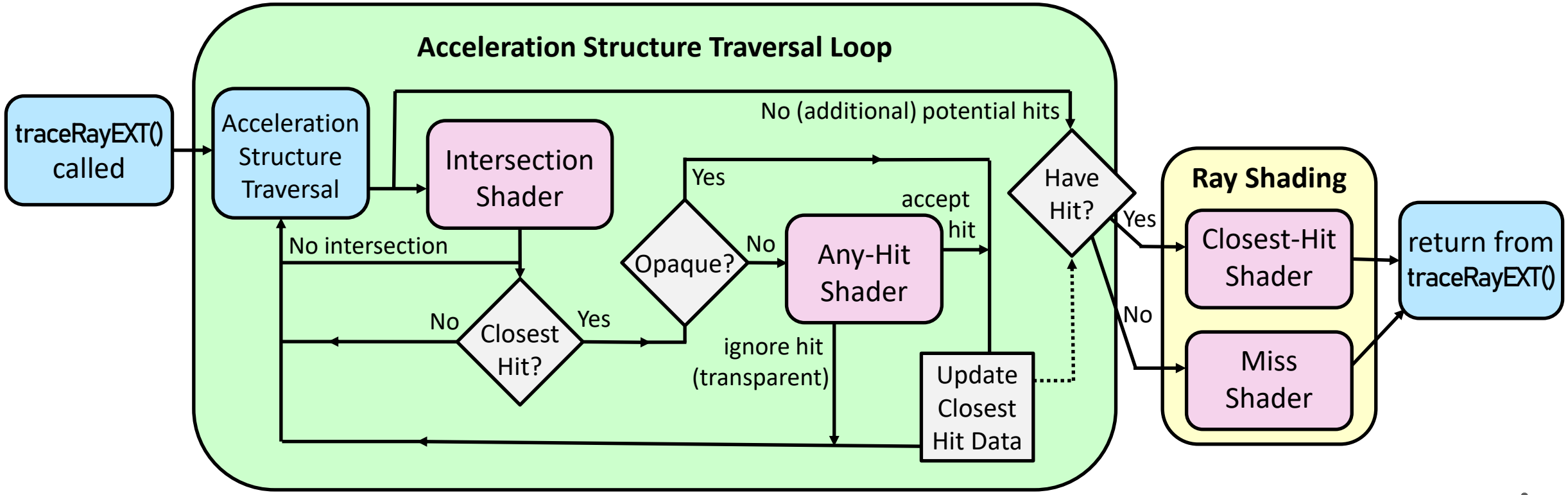
Ray Query

Ray Query

- Ray Query requires
  - Vulkan 1.1
  - VK_KHR_spirv_1_4
  - VK_KHR_acceleration_structure
  - VK_KHR_ray_query
- Access an acceleration structure from **any(!!)** shader
  (i.e., from compute shaders, fragment shaders, intersection shaders, …)
- Not the full ray tracing pipeline runs
- Only the fixed function ray -> bounding volume hierarchy intersection is performed

Ray Query

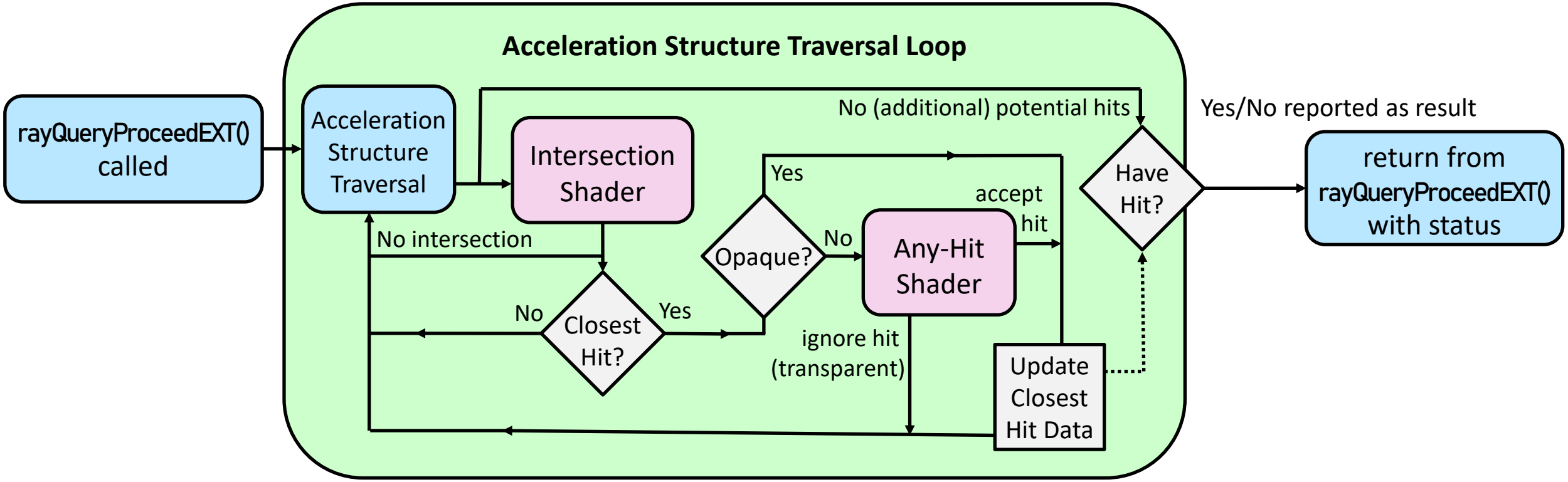
- Compared to **traceRayEXT()**, most notably, **no custom shaders** are executed during a Ray Query



Adapted from: Chris Wyman and Shawn Hargreaves and Peter Shirley and Colin Barré-Brisebois, Introduction to DirectX Raytracing, ACM SIGGRAPH 2018 Courses, https://intro-to-dxr.cwyman.org

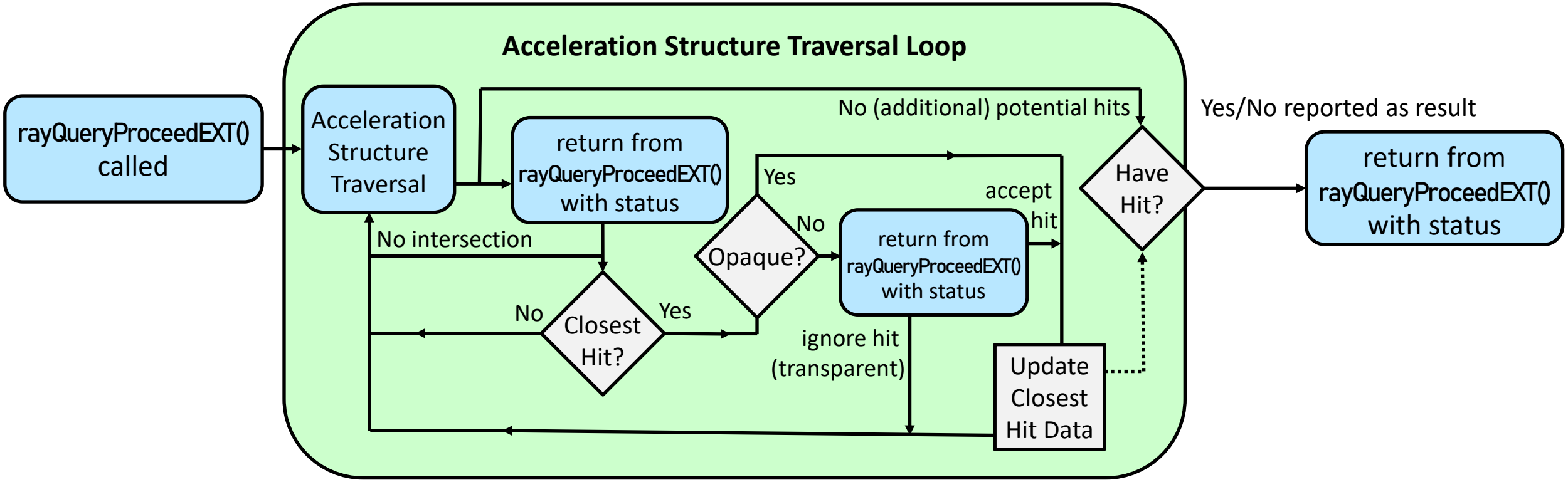
Ray Query

- Compared to **traceRayEXT()**, most notably, **no custom shaders** are executed during a Ray Query
- Call **rayQueryProceedEXT()** in a loop



Ray Query

- Compared to **traceRayEXT()**, most notably, **no custom shaders** are executed during a Ray Query
- Call **rayQueryProceedEXT()** in a loop



Ray Query

```
rayQueryEXT rayQuery;
rayQueryInitializeEXT(rayQuery, topLevelAS,
                      gl_RayFlagsTerminateOnFirstHitEXT,
                      0xFF, origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/);

while(rayQueryProceedEXT(rayQuery)) {
    if (rayQueryGetIntersectionTypeEXT(rayQuery, false) == gl_RayQueryCandidateIntersectionTriangleEXT) {
        ... // Determine if an opaque triangle hit occurred
        if (opaqueHit) {
            rayQueryConfirmIntersectionEXT(rayQuery);
        }
    }
    else if (rayQueryGetIntersectionTypeEXT(rayQuery, false) == gl_RayQueryCandidateIntersectionAABBEXT) {
        ... // Determine if an opaque hit occurred in an AABB
        if (opaqueHit) {
            rayQueryGenerateIntersectionEXT(rayQuery, ...);
        }
    }
}

if (rayQueryGetIntersectionTypeEXT(rayQuery, true) == gl_RayQueryCommittedIntersectionNoneEXT) {
    // Not shadow!
} else {
    // Shadow!
}
```


Adapted from: The Khronos® Group Inc., Ray Tracing In Vulkan, 2020, https://www.khronos.org/blog/ray-tracing-in-vulkan

Ray Query

```
rayQueryEXT rayQuery;
rayQueryInitializeEXT(rayQuery, topLevelAS,
                      gl_RayFlagsTerminateOnFirstHitEXT,
                      0xFF, origin, 0.1 /*min hit distance*/, direction, 100.0 /*max ray length*/);

while(rayQueryProceedEXT(rayQuery)) {
    if (rayQueryGetIntersectionTypeEXT(rayQuery, false) == gl_RayQueryCandidateIntersectionTriangleEXT) {
        ... // Determine if an opaque triangle hit occurred
        if (opaqueHit) {
            rayQueryConfirmIntersectionEXT(rayQuery);
        }
    }
    else if (rayQueryGetIntersectionTypeEXT(rayQuery, false) == gl_RayQueryCandidateIntersectionAABBEXT) {
        ... // Determine if an opaque hit occurred in an AABB
        if (opaqueHit) {
            rayQueryGenerateIntersectionEXT(rayQuery, ...);
        }
    }
}

if (rayQueryGetIntersectionTypeEXT(rayQuery, true) == gl_RayQueryCommittedIntersectionNoneEXT) {
    // Not shadow!
} else {
    // Shadow!
}
```

Call it in a loop

Provide intersection information manually

Evaluate if hit or miss

Adapted from: The Khronos® Group Inc., Ray Tracing In Vulkan, 2020, https://www.khronos.org/blog/ray-tracing-in-vulkan

Real-Time Rendering

186.140, 2021W, VU, 4.5ECTS

Thank you for your attention!

Johannes Unterguggenberger

Institute of Visual Computing & Human-Centered Technology

TU Wien, Austria