#WWDC18

Metal for Ray Tracing Acceleration

Session 606

Sean James, GPU Software Engineer
Wayne Lister, GPU Software Engineer

Metal Performance Shaders

GPU-accelerated primitives, optimized for iOS and macOS

- Image processing
- Linear algebra
- Machine learning—inference

Using Metal 2 for Compute	WWDC 2017
What’s New in Metal, Part 2	WWDC 2016

Metal Performance Shaders

NEW

GPU-accelerated primitives, optimized for iOS and macOS

- Image processing
- Linear algebra
- Machine learning—inference and training

Metal for Accelerating Machine Learning	Hall 3	Thursday 4:00PM

Metal Performance Shaders

NEW

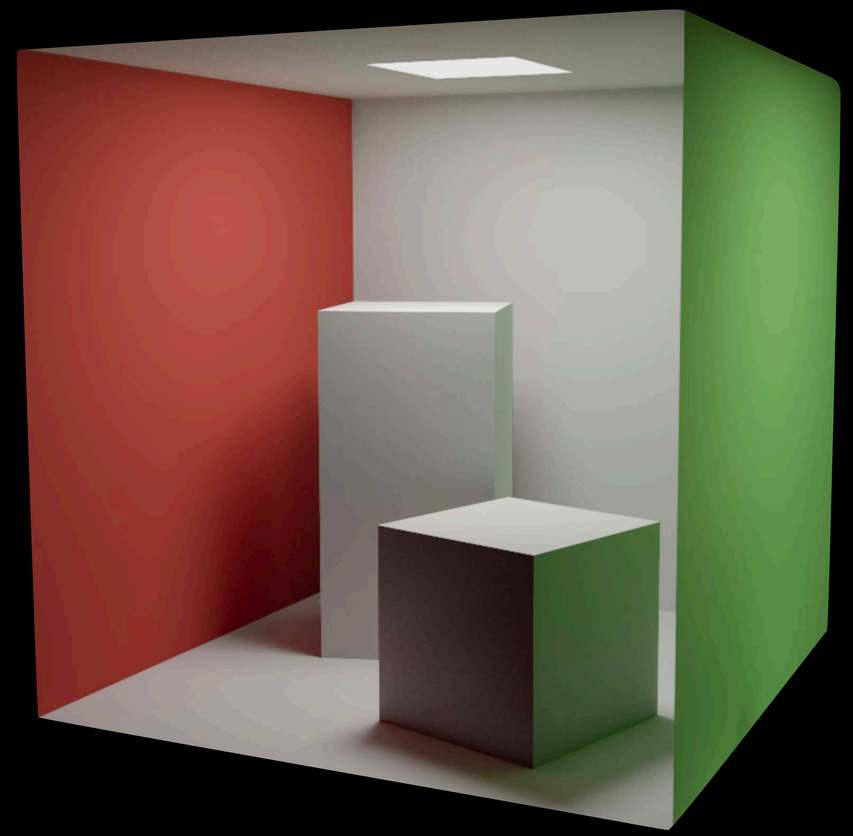
GPU-accelerated primitives, optimized for iOS and macOS

- Image processing
- Linear algebra
- Machine learning—inference and training
- Ray tracing

Metal for Accelerating Machine Learning | Hall 3 | Thursday 4:00PM

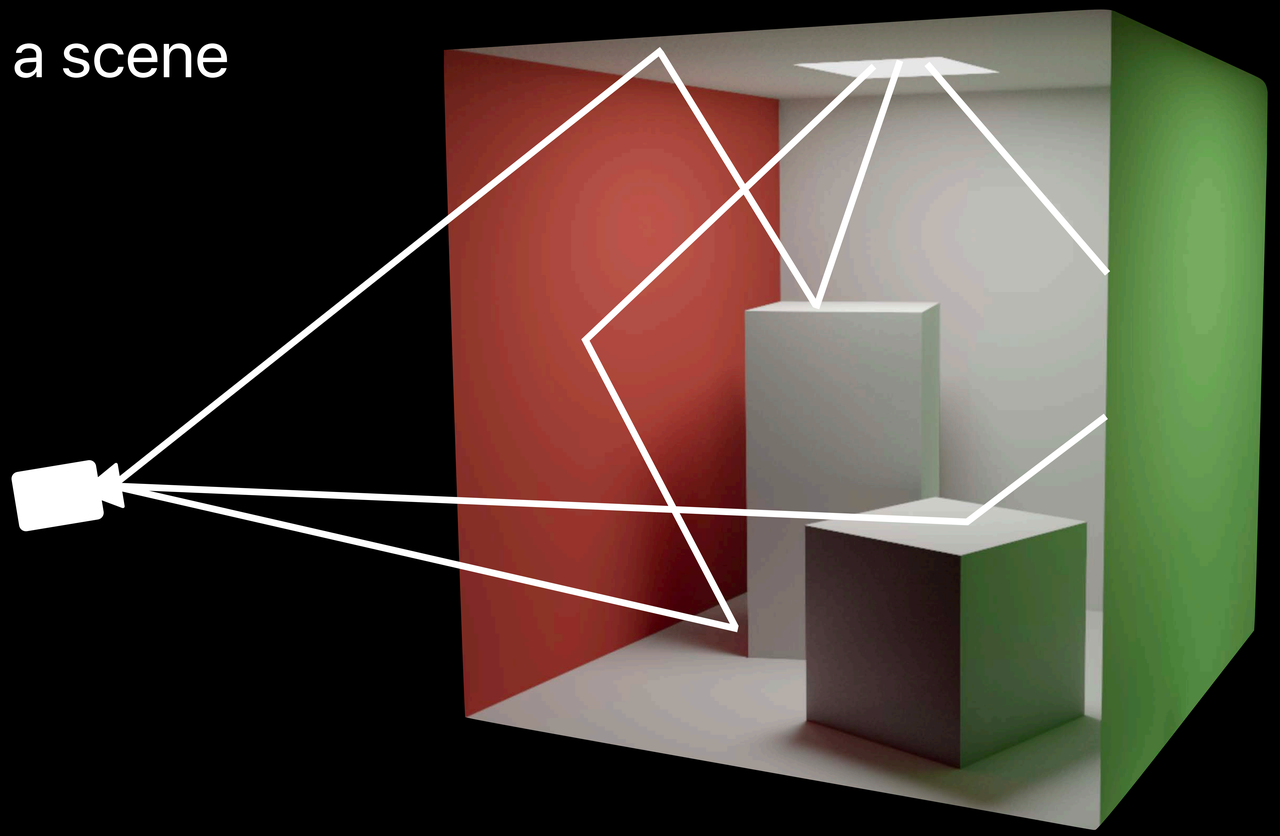
Ray Tracing

Tracing a ray’s path as it interacts with a scene



Ray Tracing

Tracing a ray’s path as it interacts with a scene

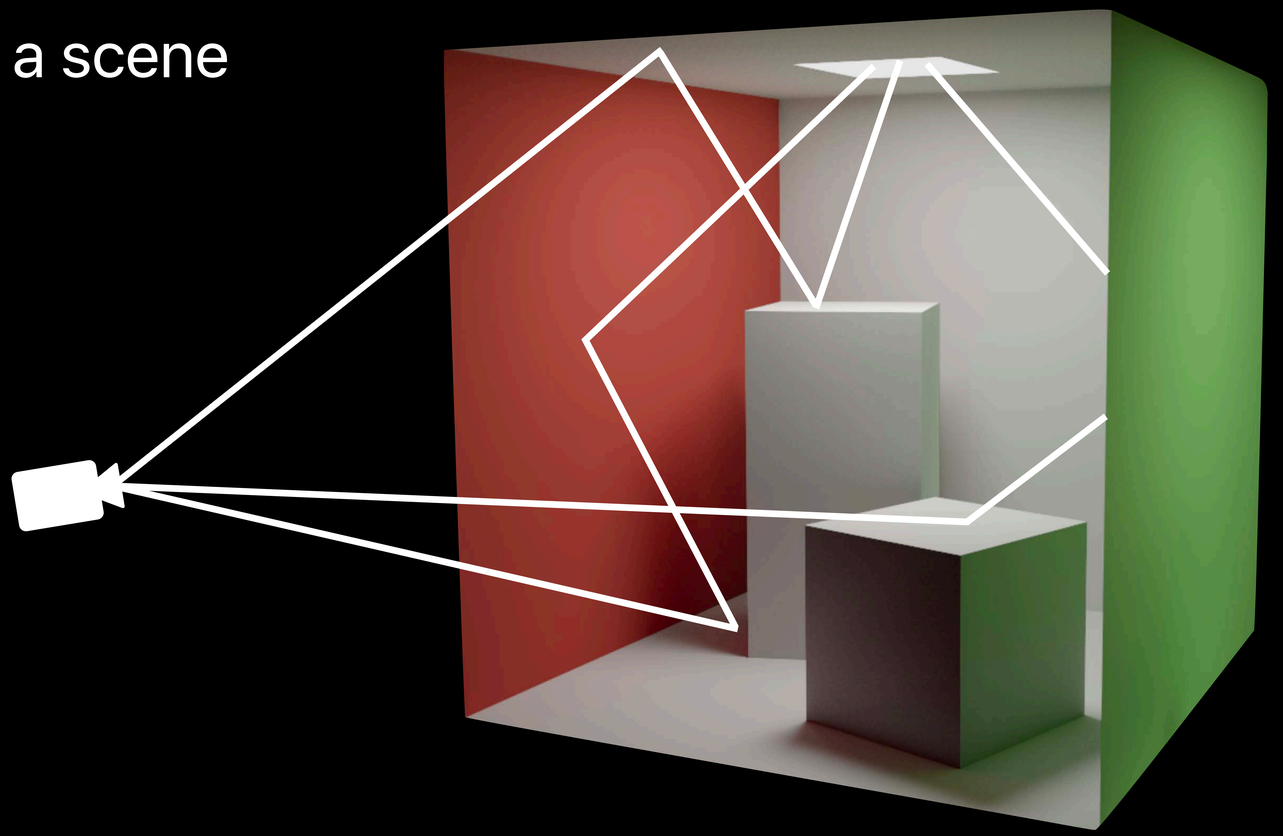


Ray Tracing

Tracing a ray’s path as it interacts with a scene

Applications

- Rendering
- Audio and physics simulation
- Collision detection
- AI and pathfinding



Rasterization

Projects triangles onto the screen one at a time

Rasterization

Projects triangles onto the screen one at a time

Fast—method of choice for games and real-time applications

Rasterization

Projects triangles onto the screen one at a time

Fast—method of choice for games and real-time applications

Difficult to model behavior of light

Ray Tracing

Reflections

Can be computed accurately with ray tracing

Ray Tracing

Soft shadows

Can be computed directly with ray tracing

Realistic transition from hard to soft shadows

Ray Tracing

Global illumination

Naturally modeled with ray tracing

Ray Tracing

Many other effects—ambient occlusion, refraction, area lights, depth of field, motion blur

Ray Tracing

Many other effects—ambient occlusion, refraction, area lights, depth of field, motion blur

Method of choice for photorealistic, offline rendering

Ray Tracing

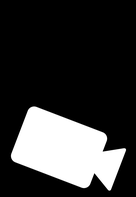
Many other effects—ambient occlusion, refraction, area lights, depth of field, motion blur

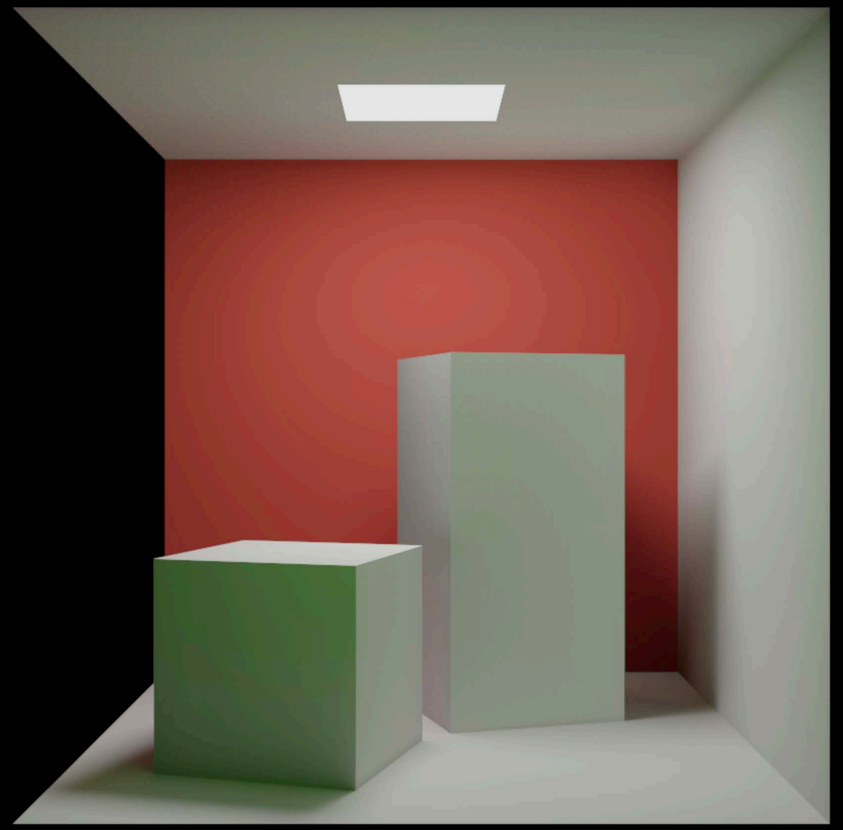
Method of choice for photorealistic, offline rendering

Significantly more computationally expensive—doing more work to simulate physical effects

Rendering with Ray Tracing

Work backwards from camera to light source

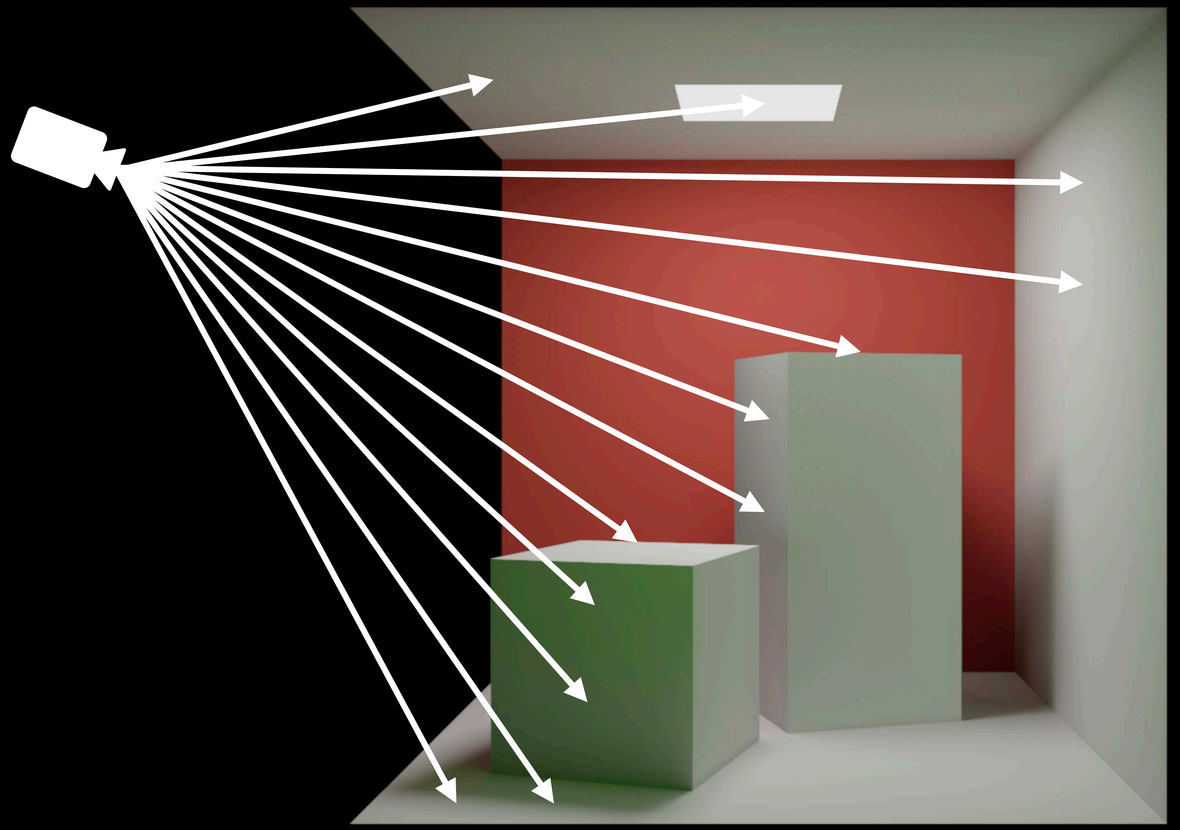




Rendering with Ray Tracing

Work backwards from camera to light source

Cast primary rays from the camera into the scene

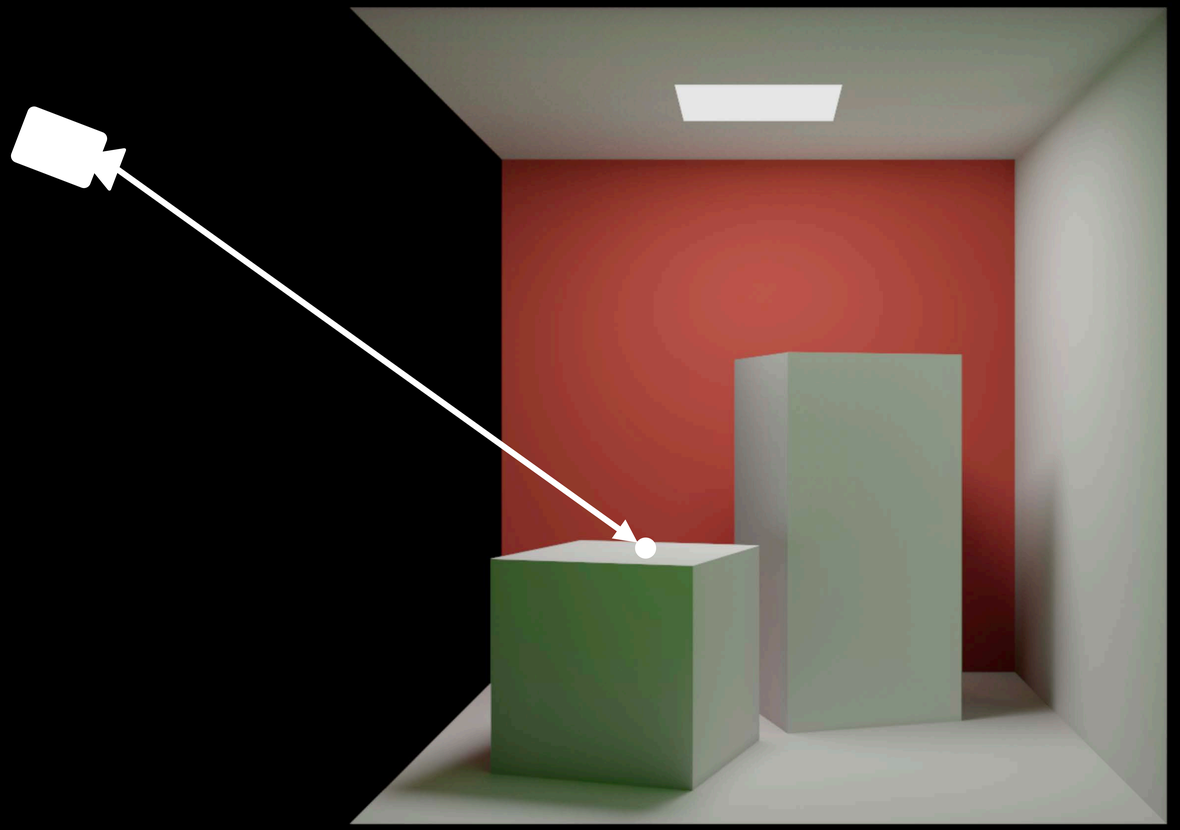


Rendering with Ray Tracing

Work backwards from camera to light source

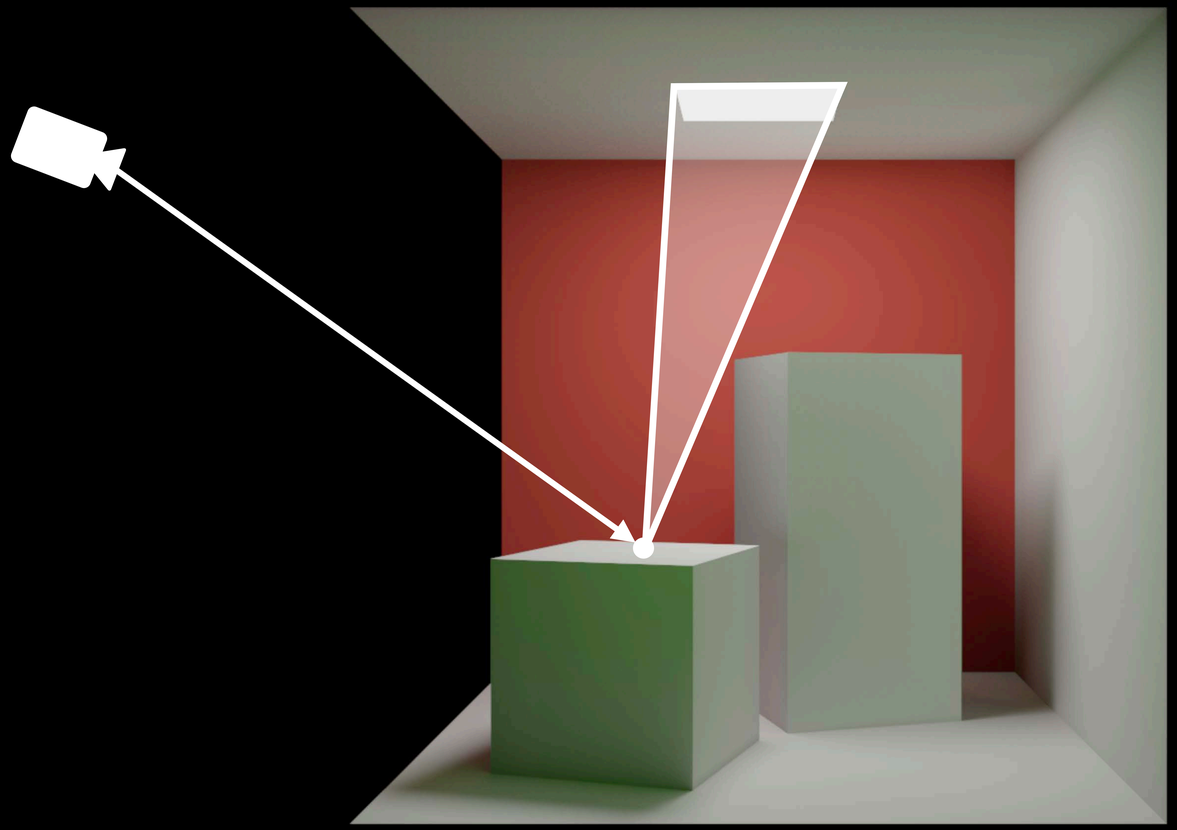
Cast primary rays from the camera into the scene

Compute shading at intersection points



Rendering with Ray Tracing

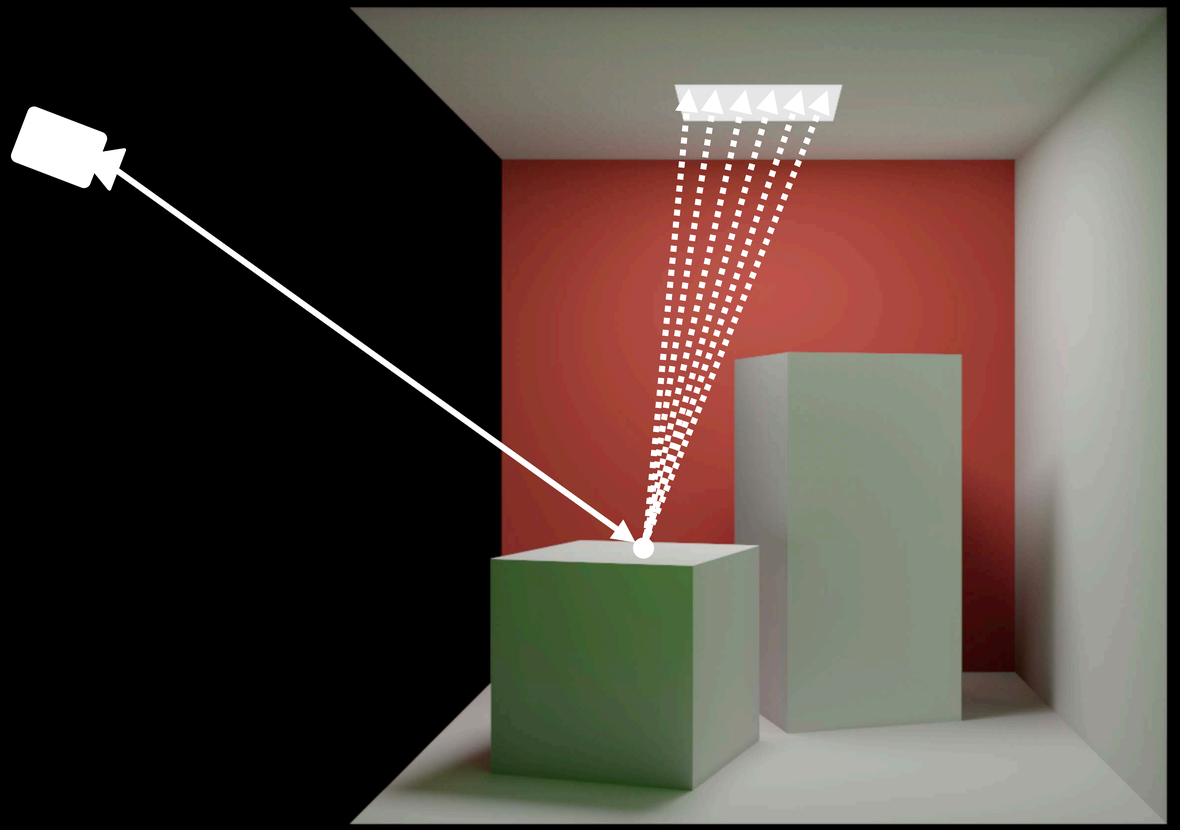
Compute direct lighting contribution



Rendering with Ray Tracing

Compute direct lighting contribution

Cast shadow rays from intersection point to light source

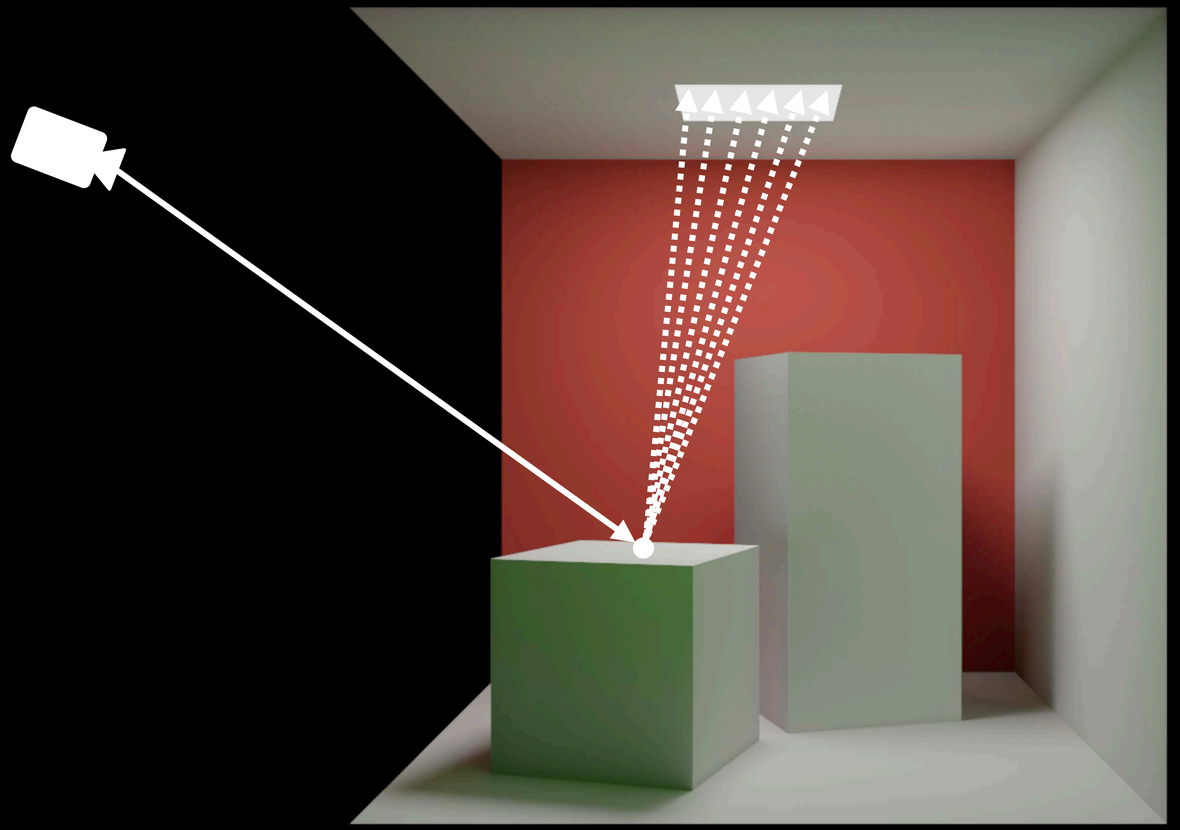


Rendering with Ray Tracing

Compute direct lighting contribution

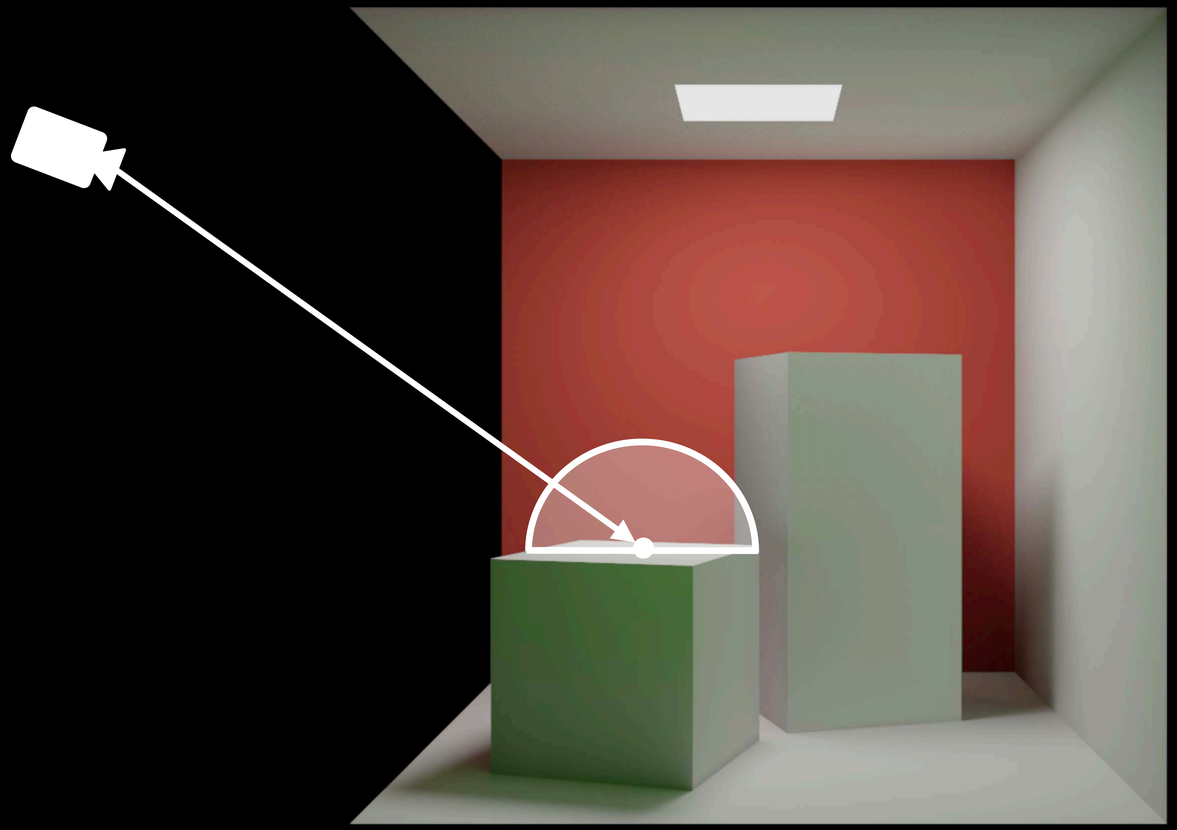
Cast shadow rays from intersection point to light source

In shadow if shadow ray does not reach light



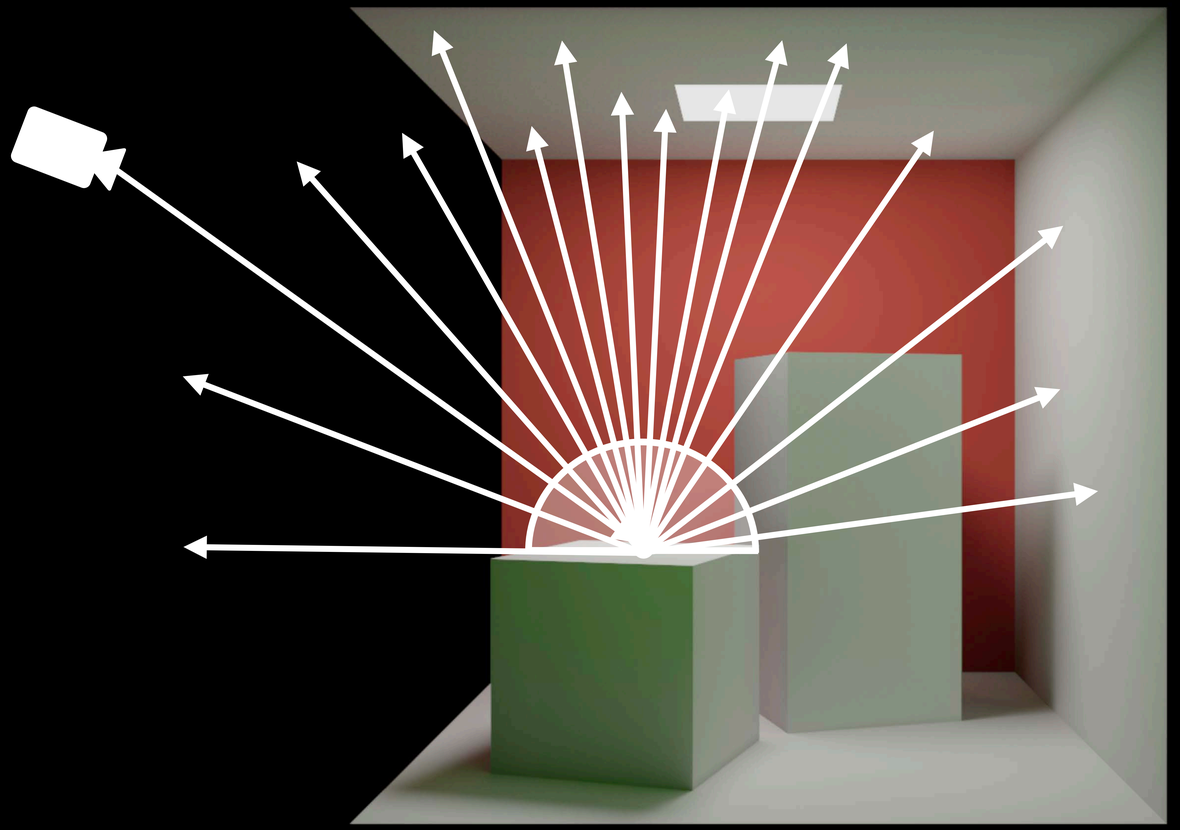
Rendering with Ray Tracing

Cast secondary rays from intersection point in random directions



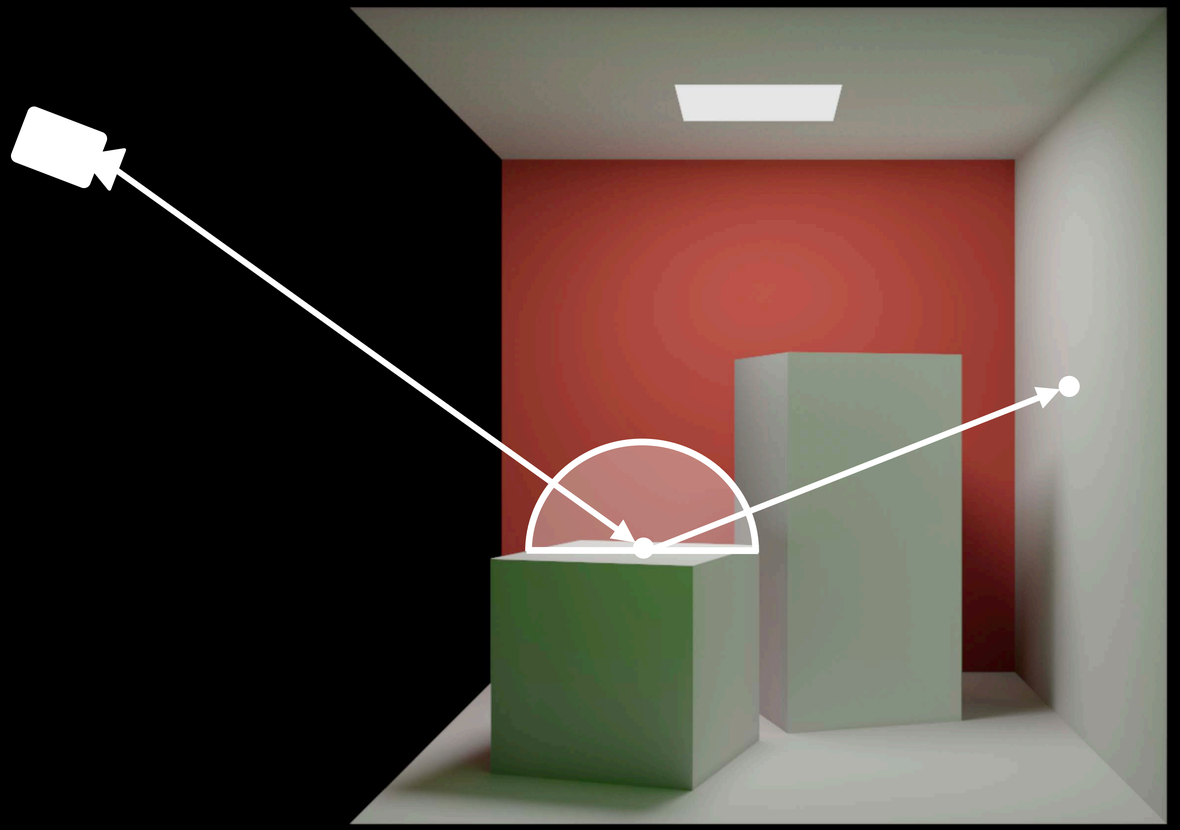
Rendering with Ray Tracing

Cast secondary rays from intersection point in random directions



Rendering with Ray Tracing

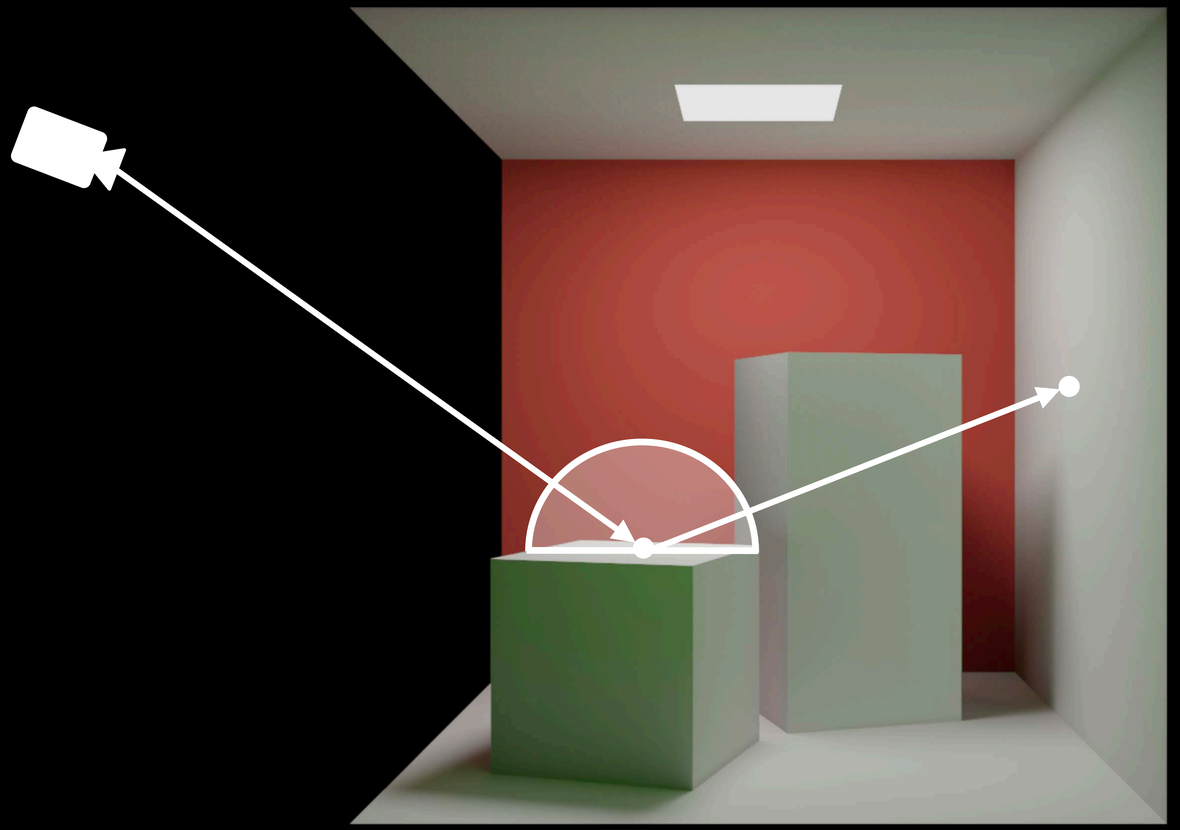
Cast secondary rays from intersection point in random directions



Rendering with Ray Tracing

Cast secondary rays from intersection point in random directions

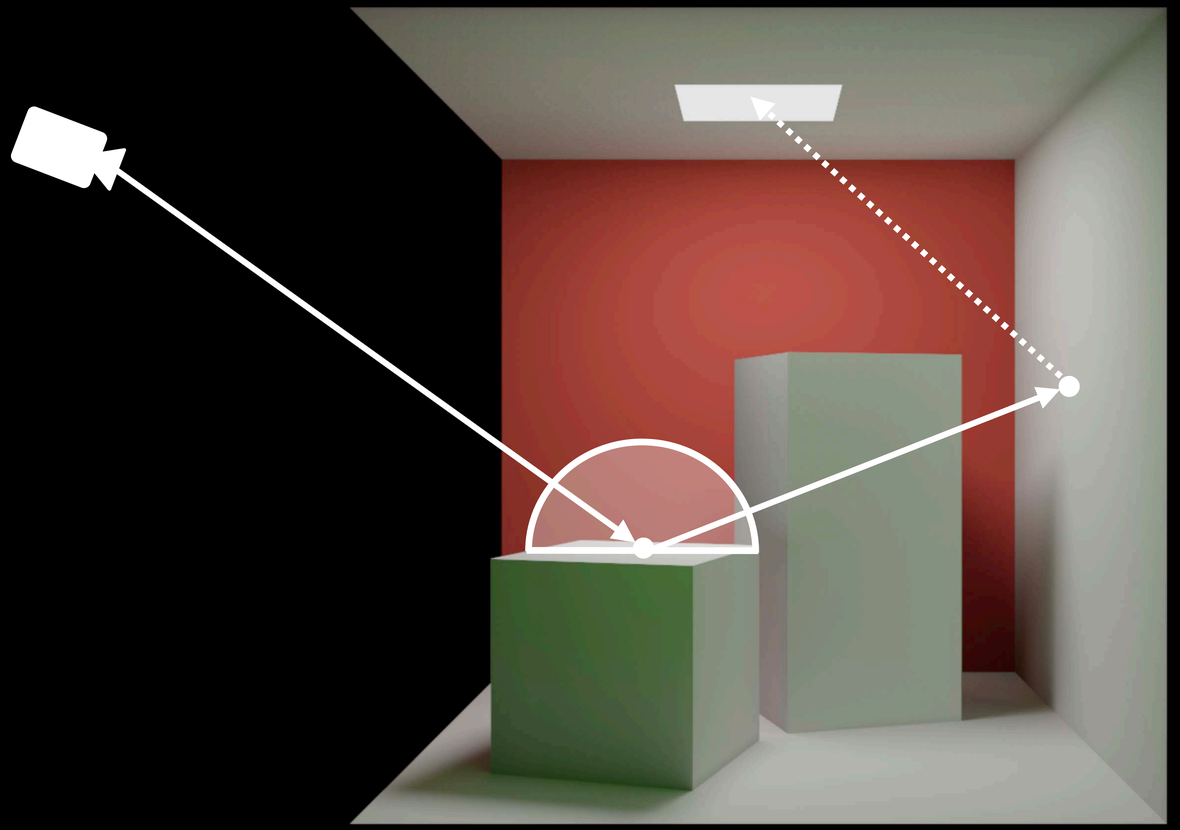
Add direct lighting reflected from secondary intersection point



Rendering with Ray Tracing

Cast secondary rays from intersection point in random directions

Add direct lighting reflected from secondary intersection point

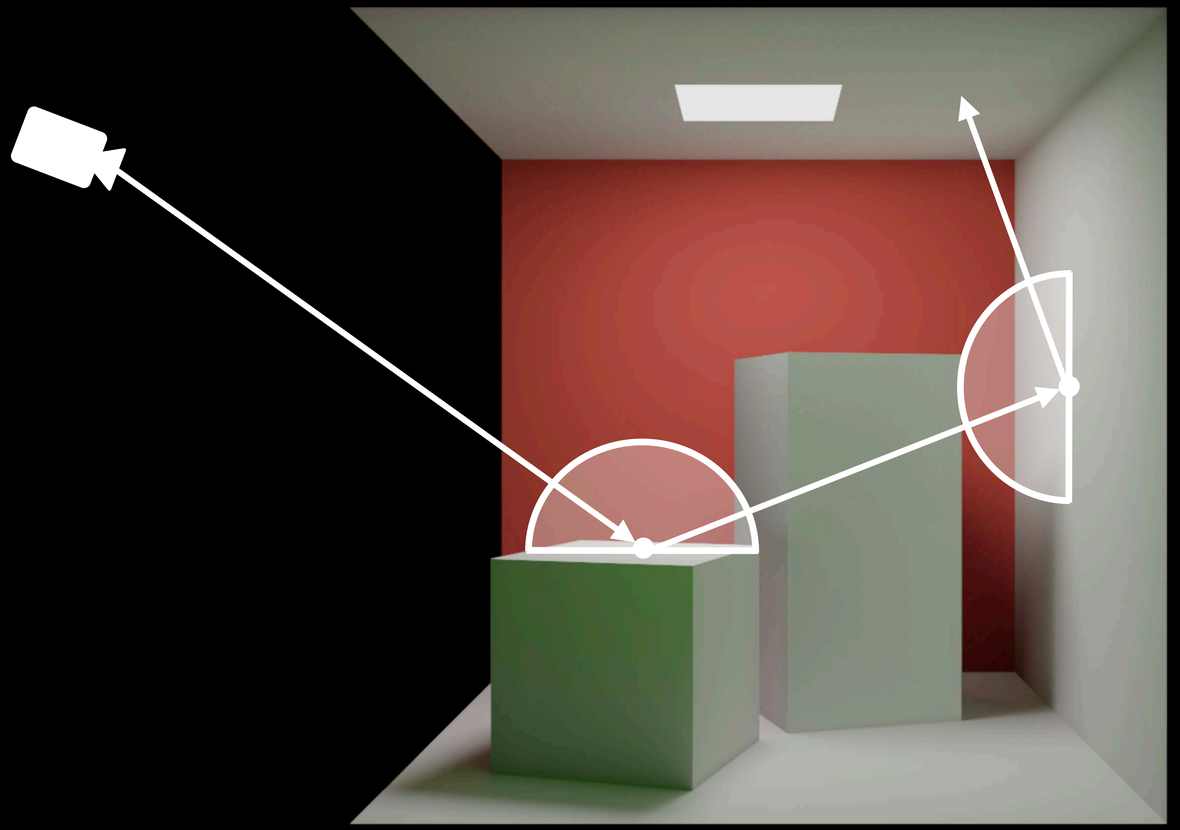


Rendering with Ray Tracing

Cast secondary rays from intersection point in random directions

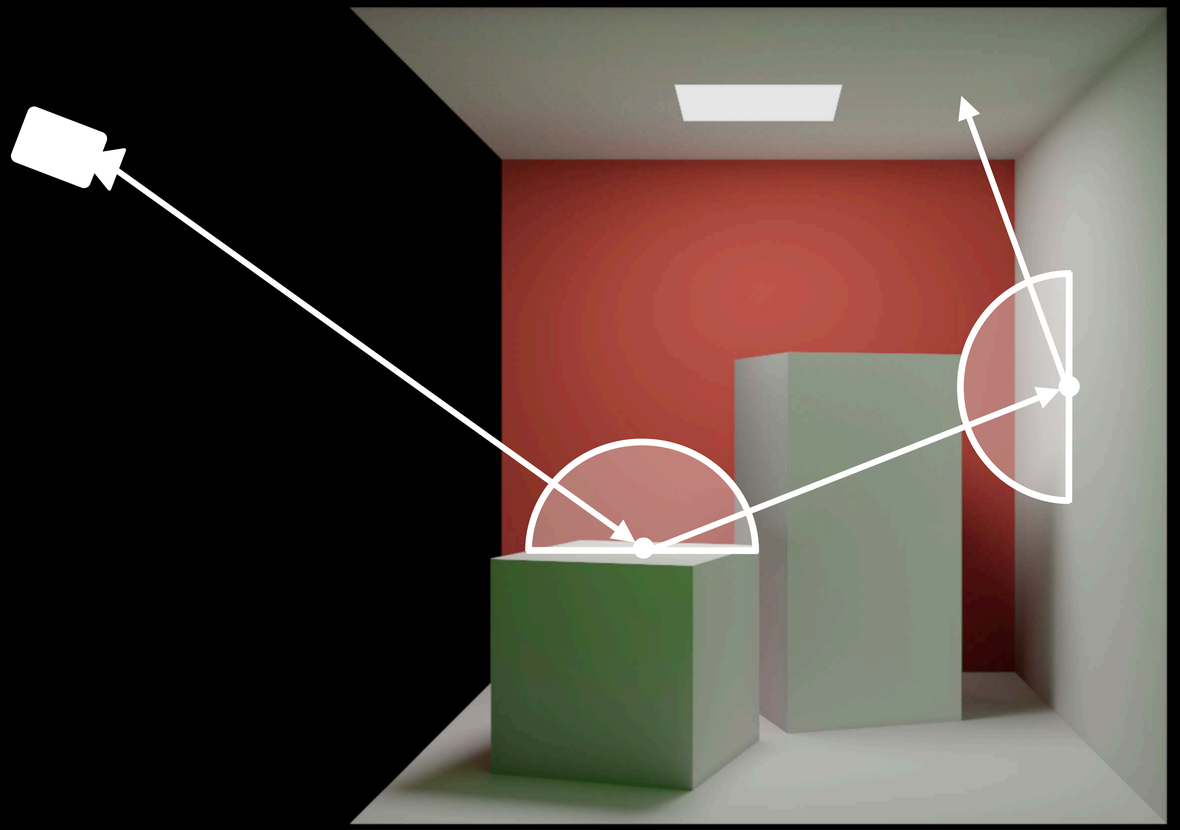
Add direct lighting reflected from secondary intersection point

Repeat to simulate light bouncing



Rendering with Ray Tracing

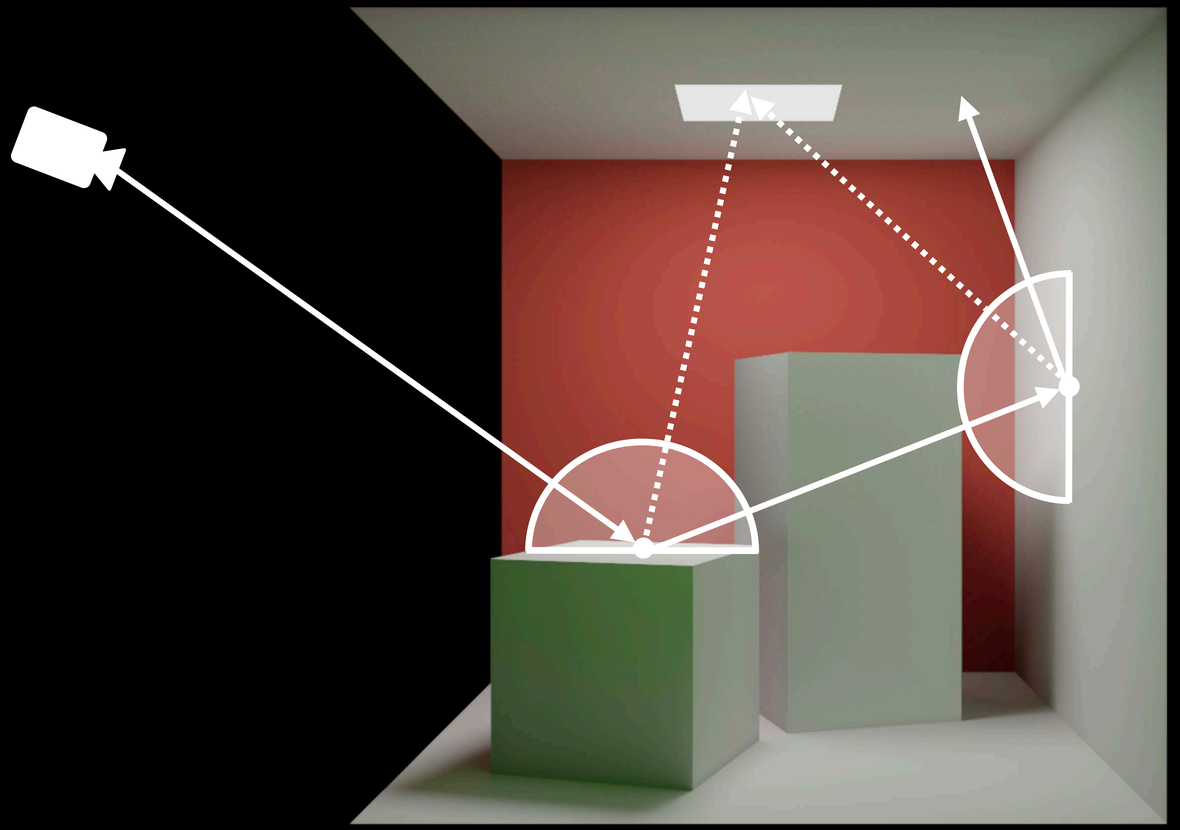
Ray count grows exponentially with number of bounces



Rendering with Ray Tracing

Ray count grows exponentially with number of bounces

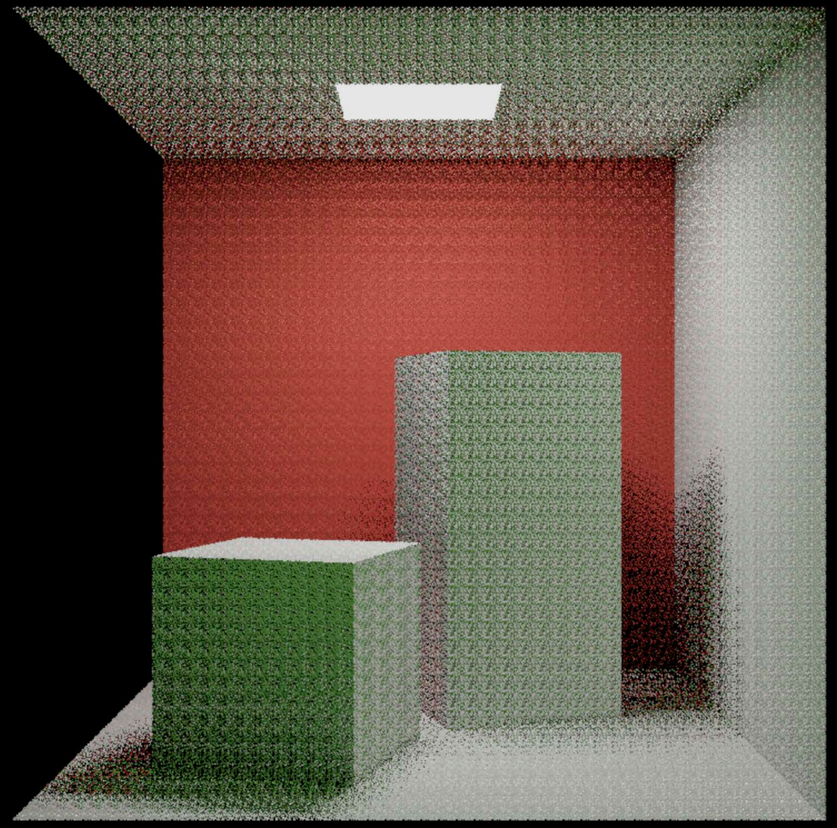
Avoid growth—one shadow ray and secondary ray per bounce



Rendering with Ray Tracing

Ray count grows exponentially with number of bounces

Avoid growth—one shadow ray and secondary ray per bounce

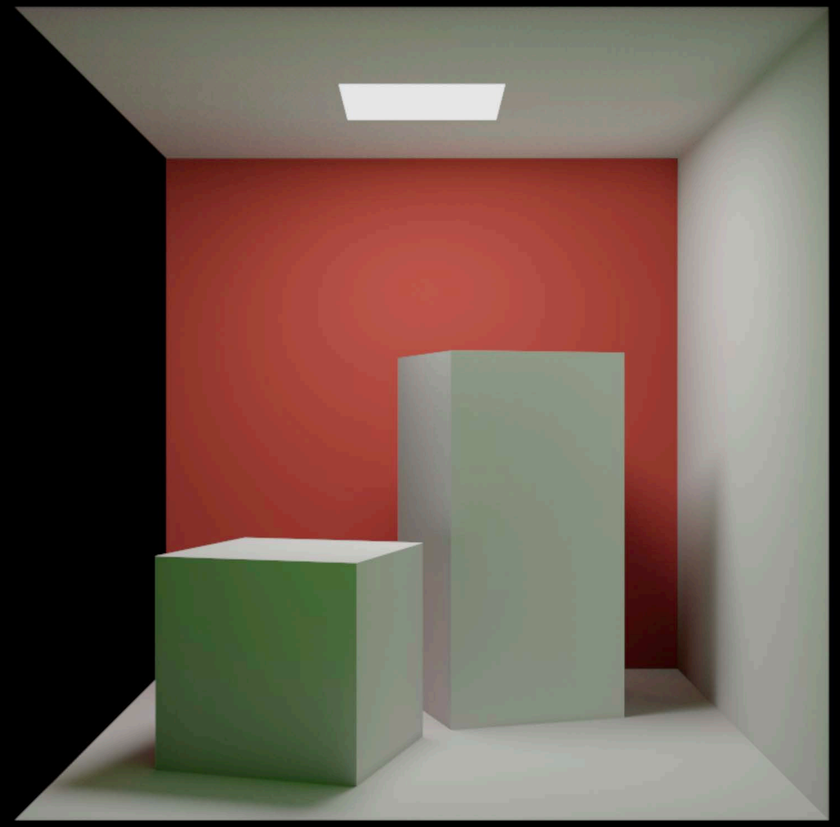


Rendering with Ray Tracing

Ray count grows exponentially with number of bounces

Avoid growth—one shadow ray and secondary ray per bounce

Average over multiple frames

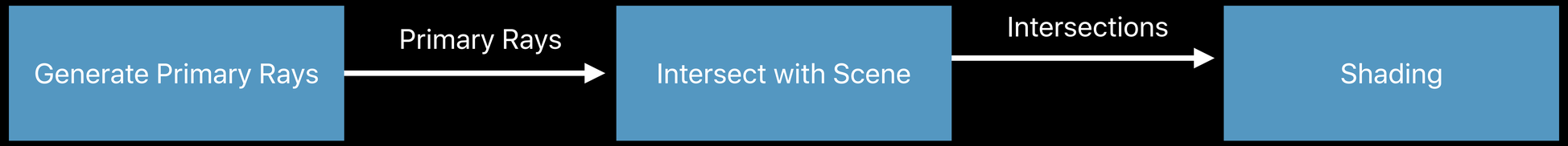


Rendering with Ray Tracing

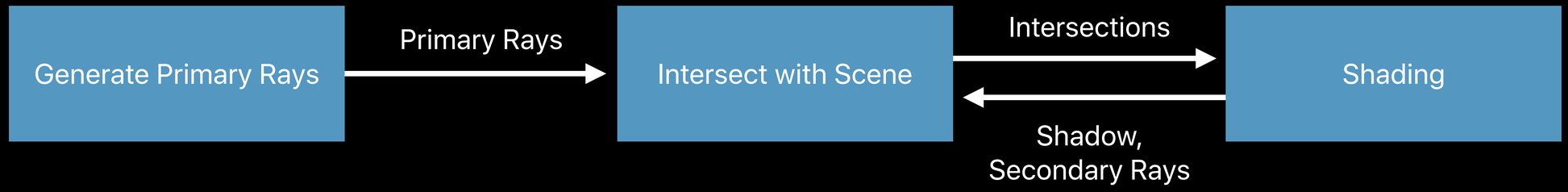
Rendering with Ray Tracing

Rendering with Ray Tracing

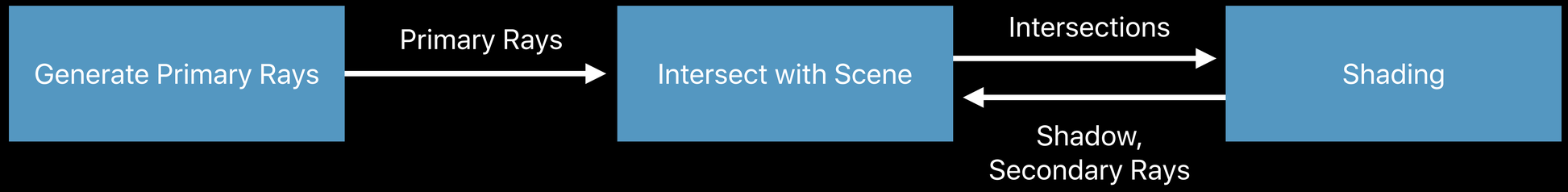
Rendering with Ray Tracing



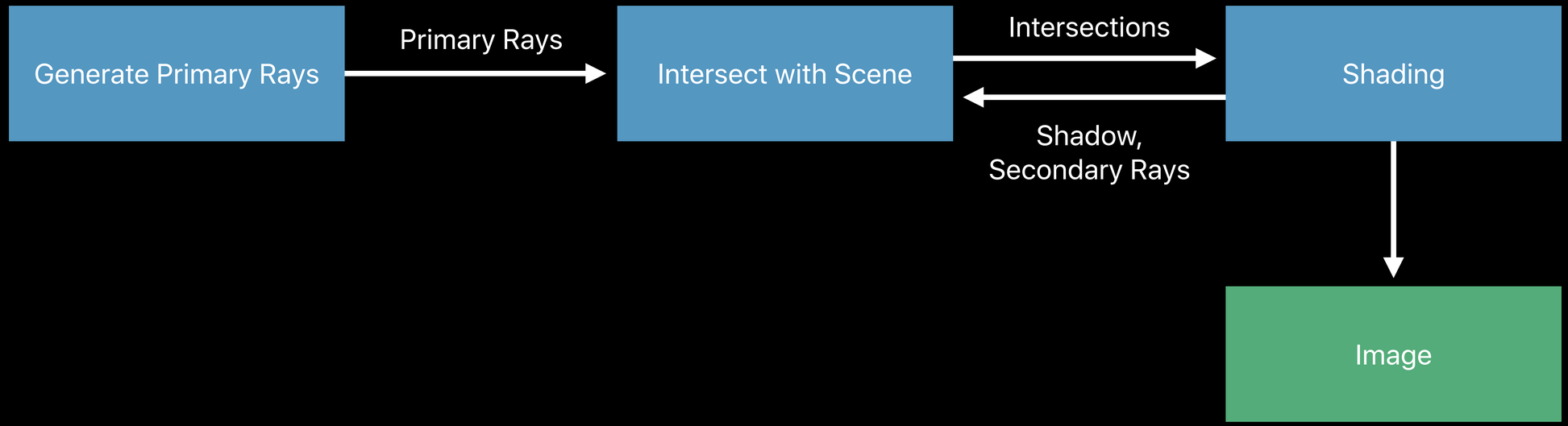
Rendering with Ray Tracing



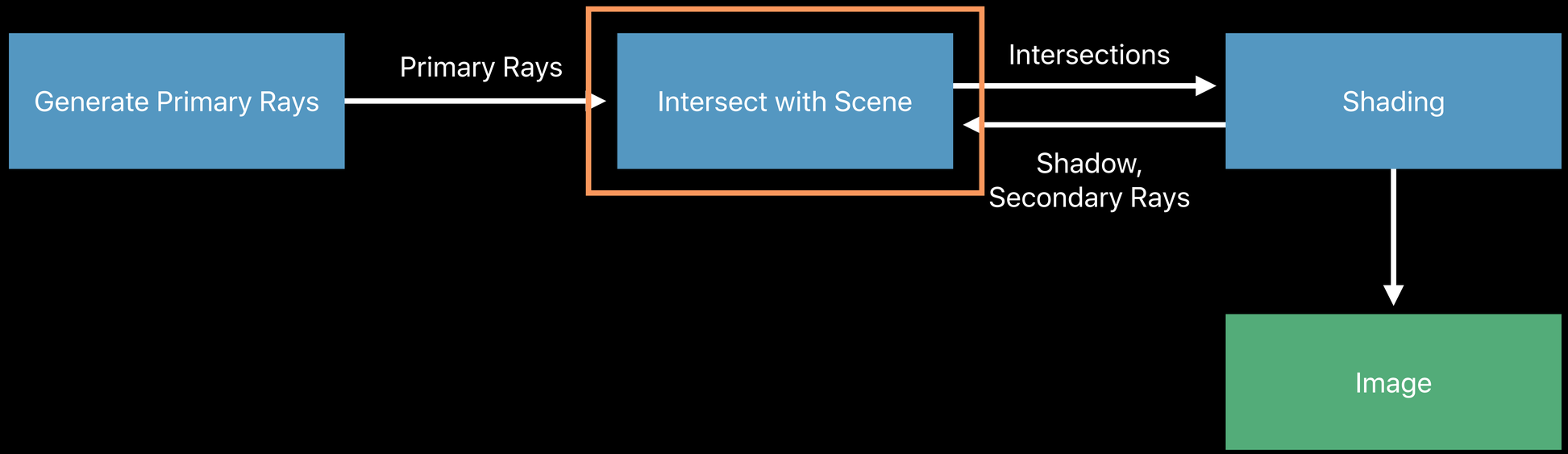
Rendering with Ray Tracing



Rendering with Ray Tracing



Rendering with Ray Tracing



MPSRayIntersector

NEW

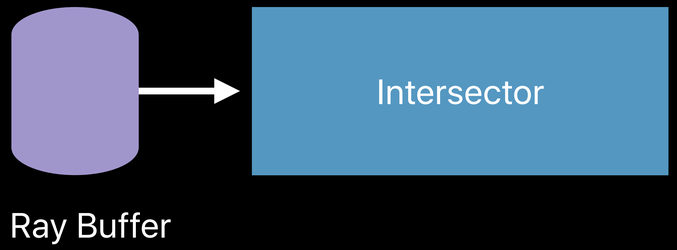
Ray intersector accelerates ray/triangle intersection tests on the GPU

MPSRayIntersector

NEW

Ray intersector accelerates ray/triangle intersection tests on the GPU

Accepts batches of rays in a Metal buffer



MPSRayIntersector

NEW

Ray intersector accelerates ray/triangle intersection tests on the GPU

Accepts batches of rays in a Metal buffer

Returns one intersection per ray

MPSRayIntersector

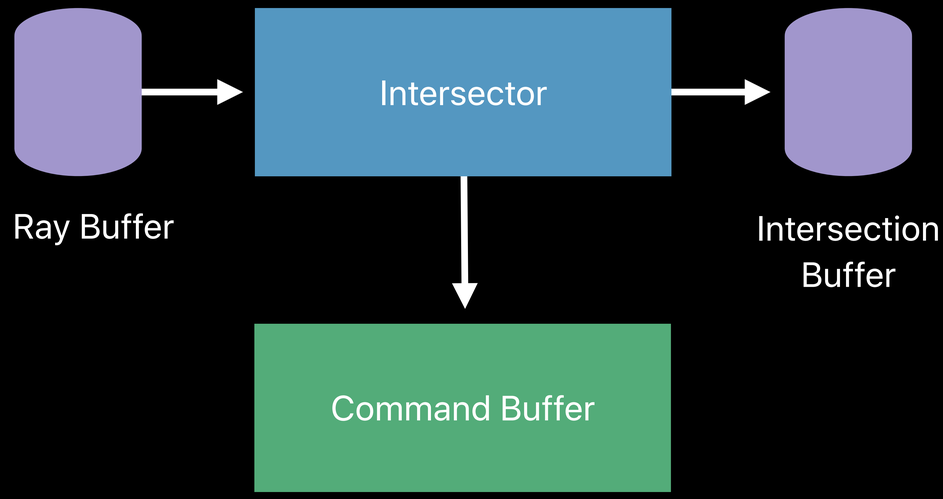
NEW

Ray intersector accelerates ray/triangle intersection tests on the GPU

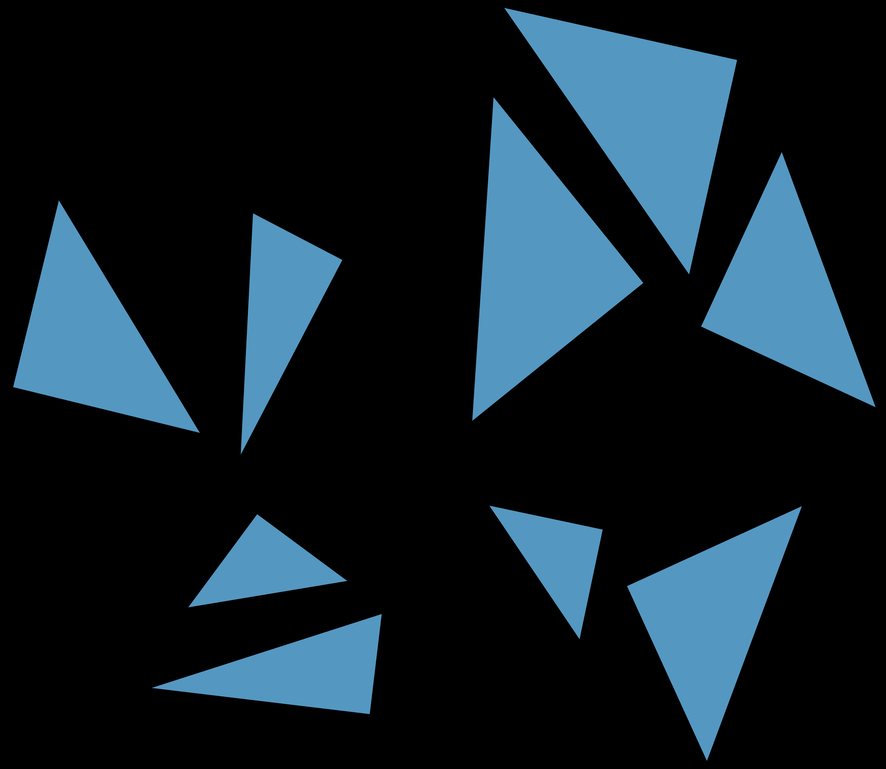
Accepts batches of rays in a Metal buffer

Returns one intersection per ray

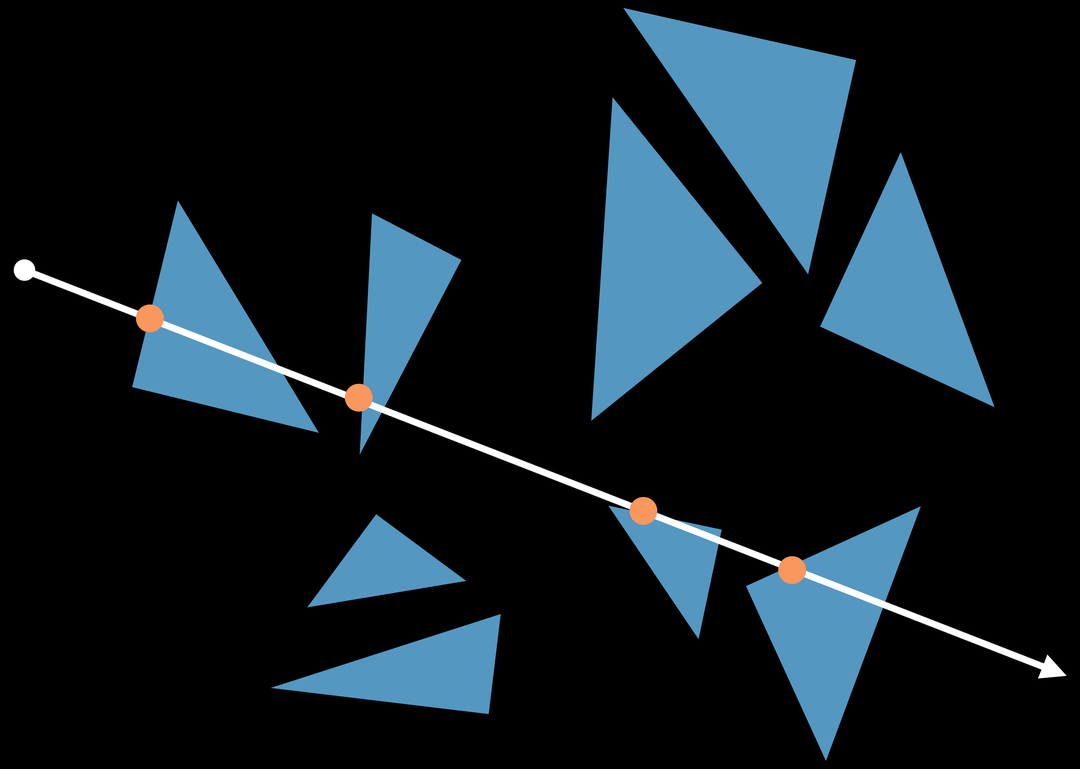
Encodes into a Metal command buffer



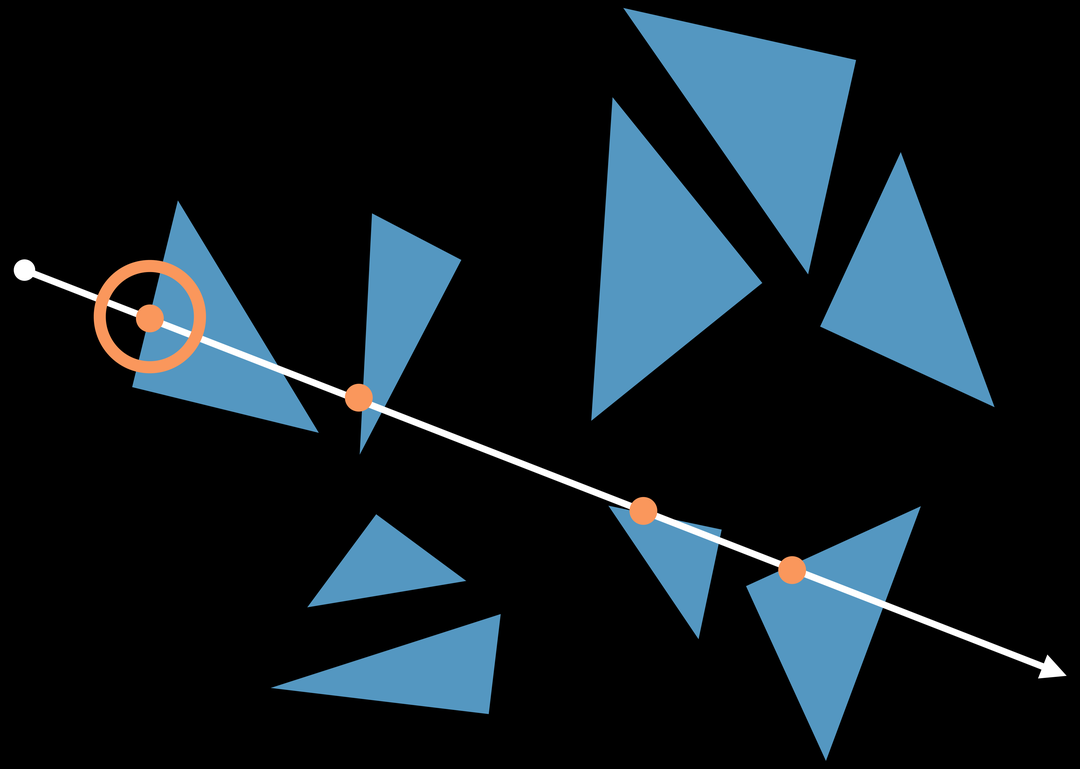
Accelerating Ray/Scene Intersection



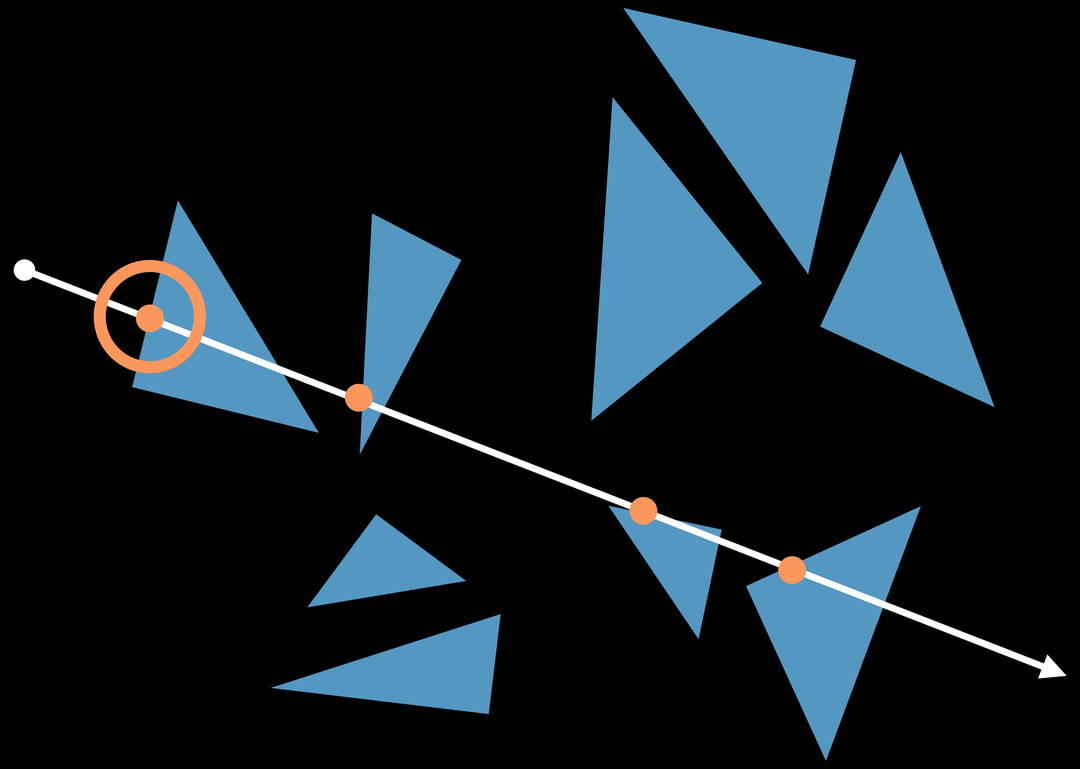
Accelerating Ray/Scene Intersection



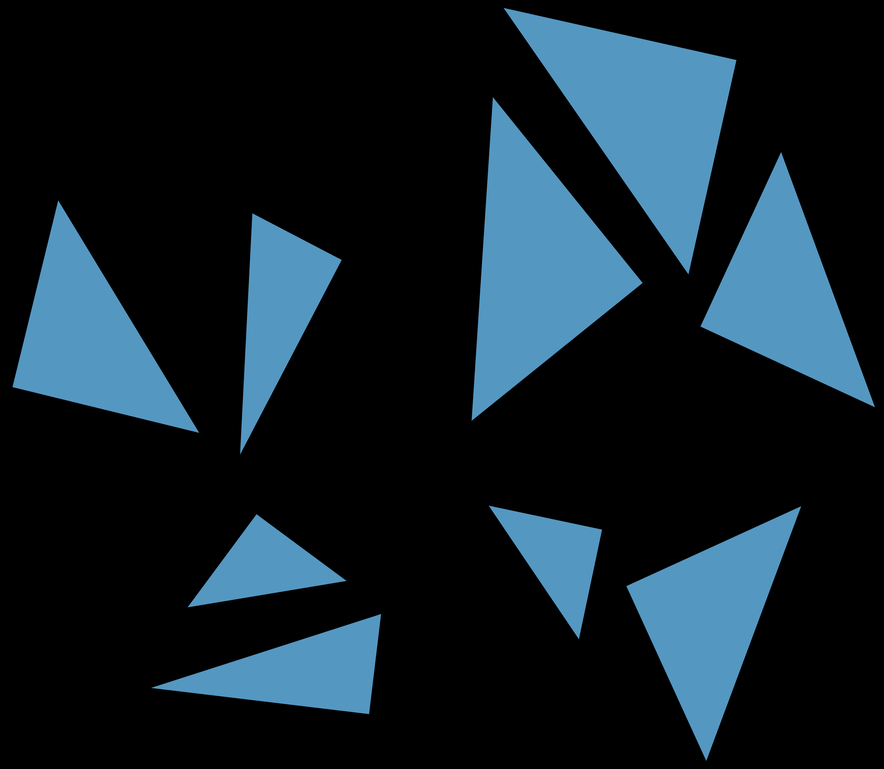
Accelerating Ray/Scene Intersection



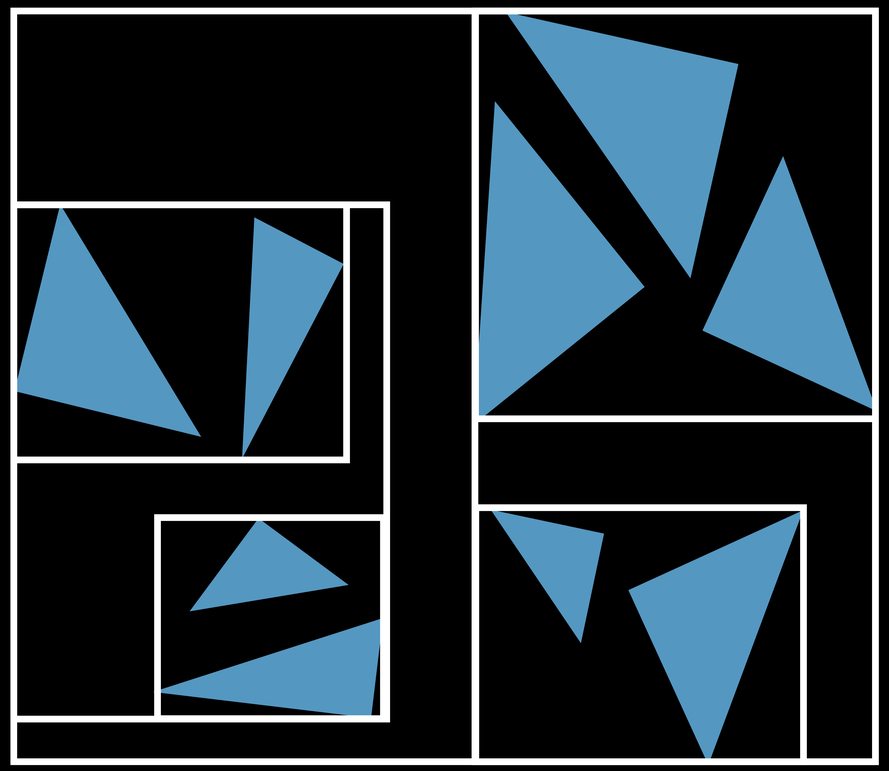
Accelerating Ray/Scene Intersection

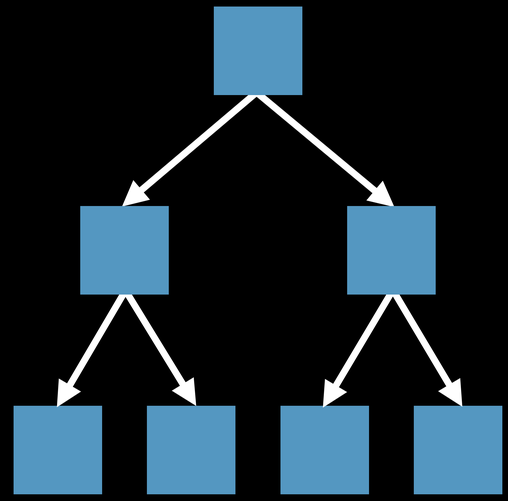


Accelerating Ray/Scene Intersection

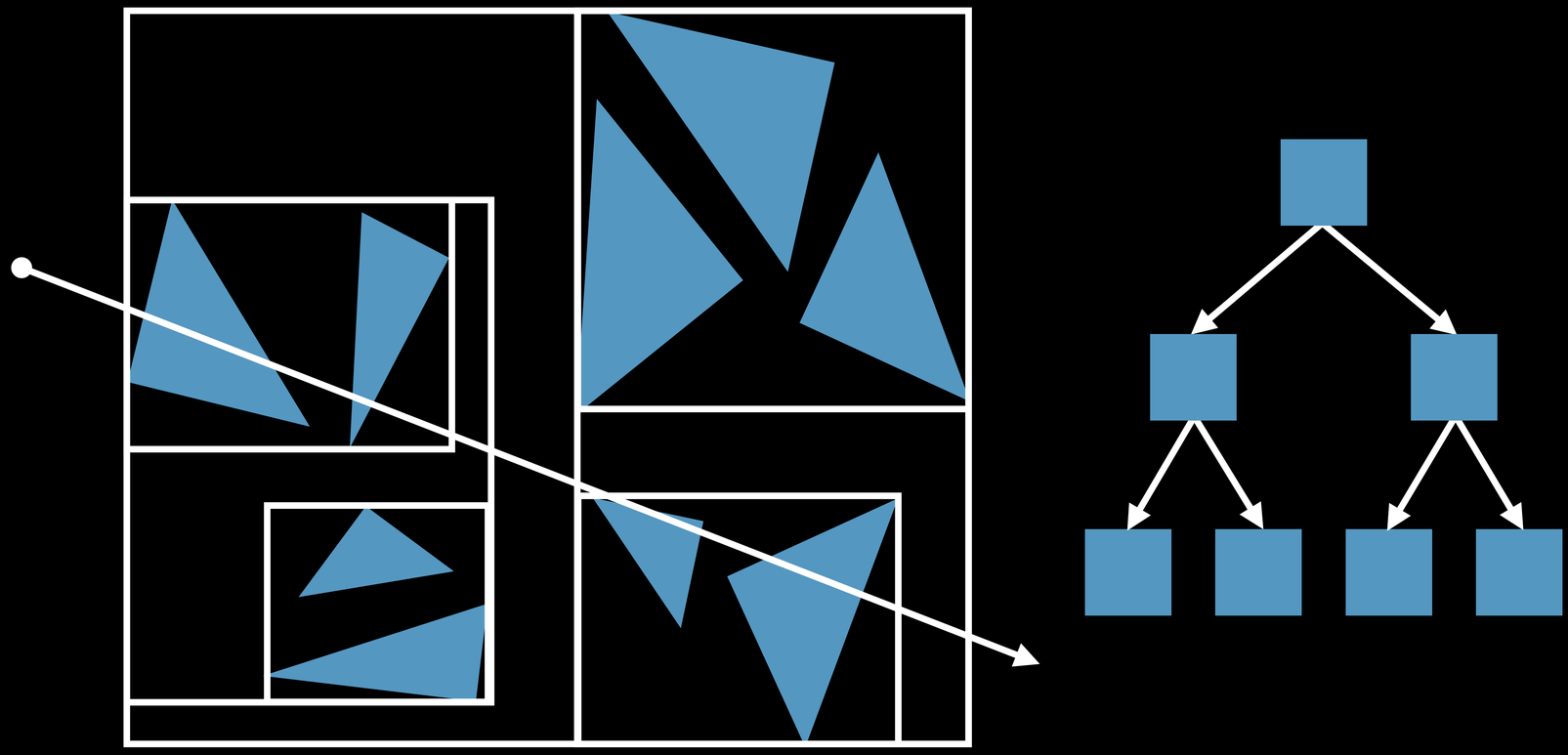


Accelerating Ray/Scene Intersection

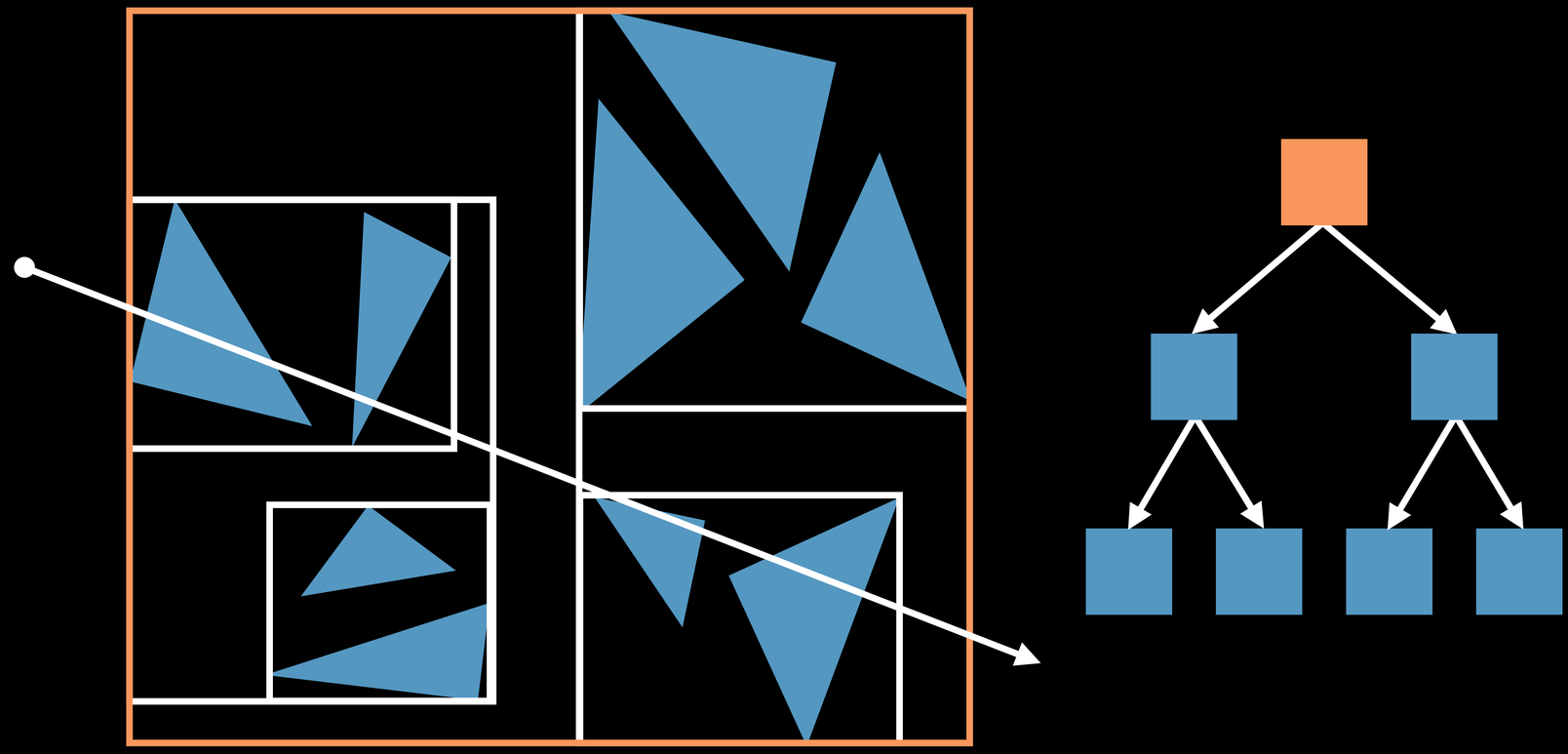




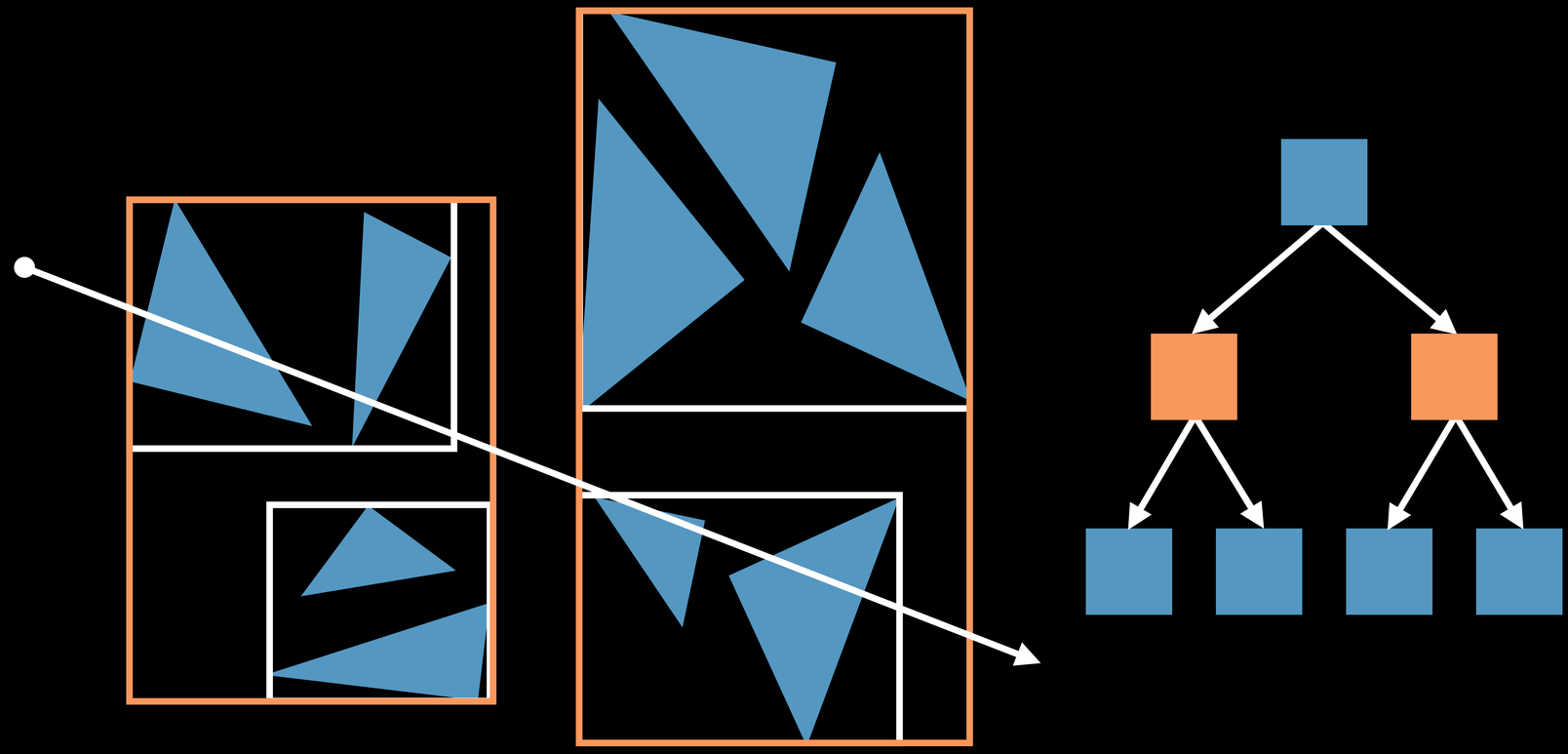
Accelerating Ray/Scene Intersection



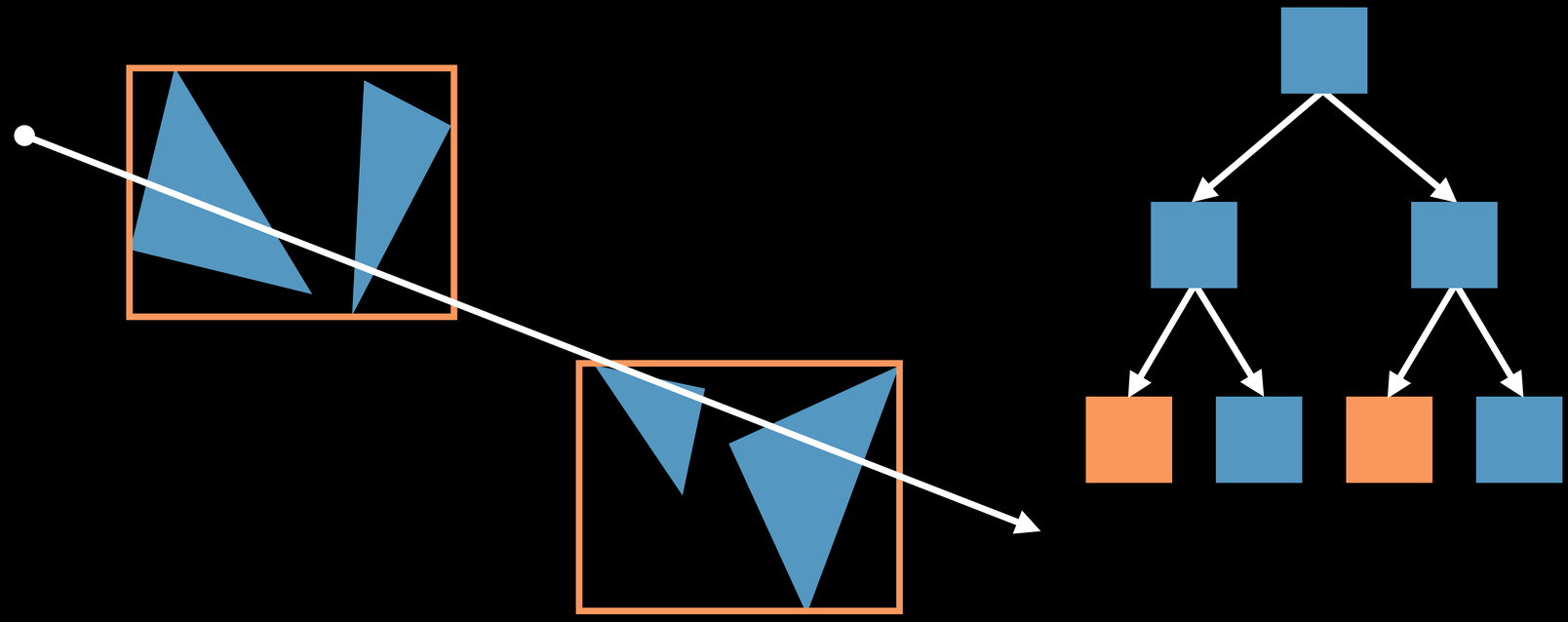
Accelerating Ray/Scene Intersection



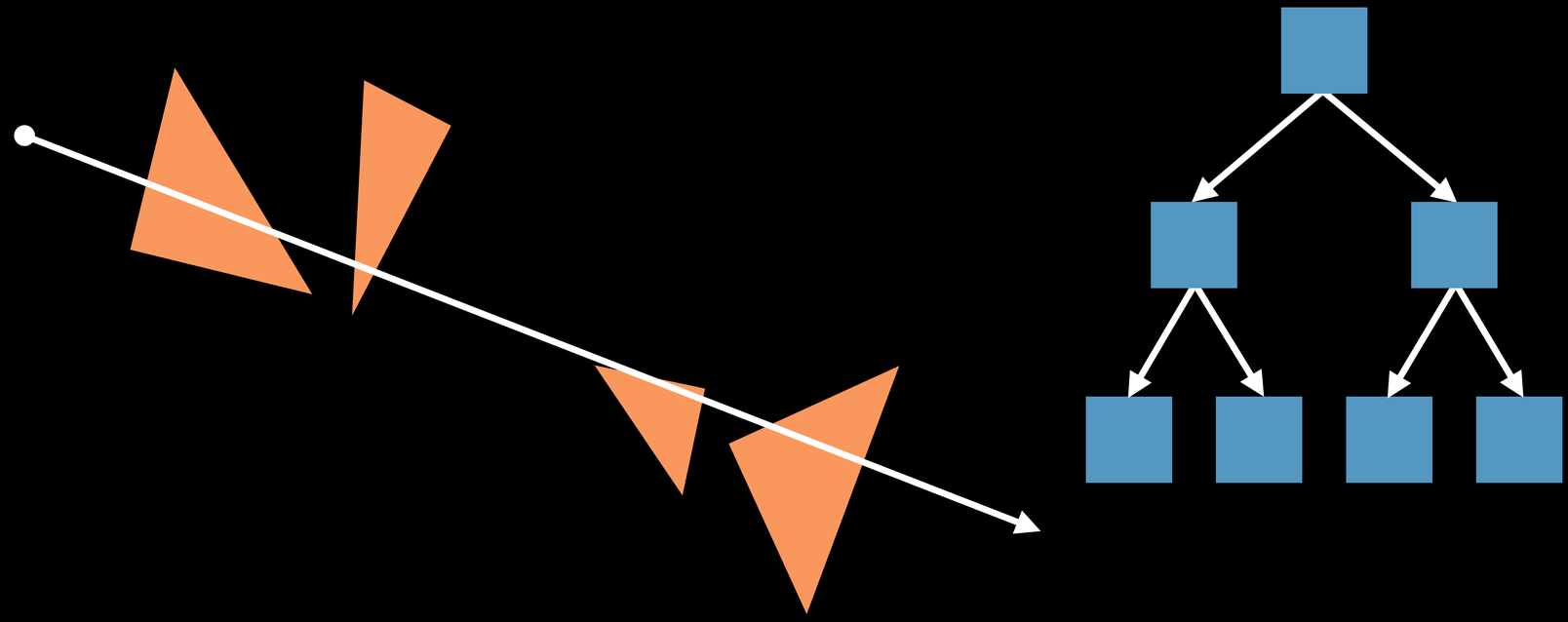
Accelerating Ray/Scene Intersection



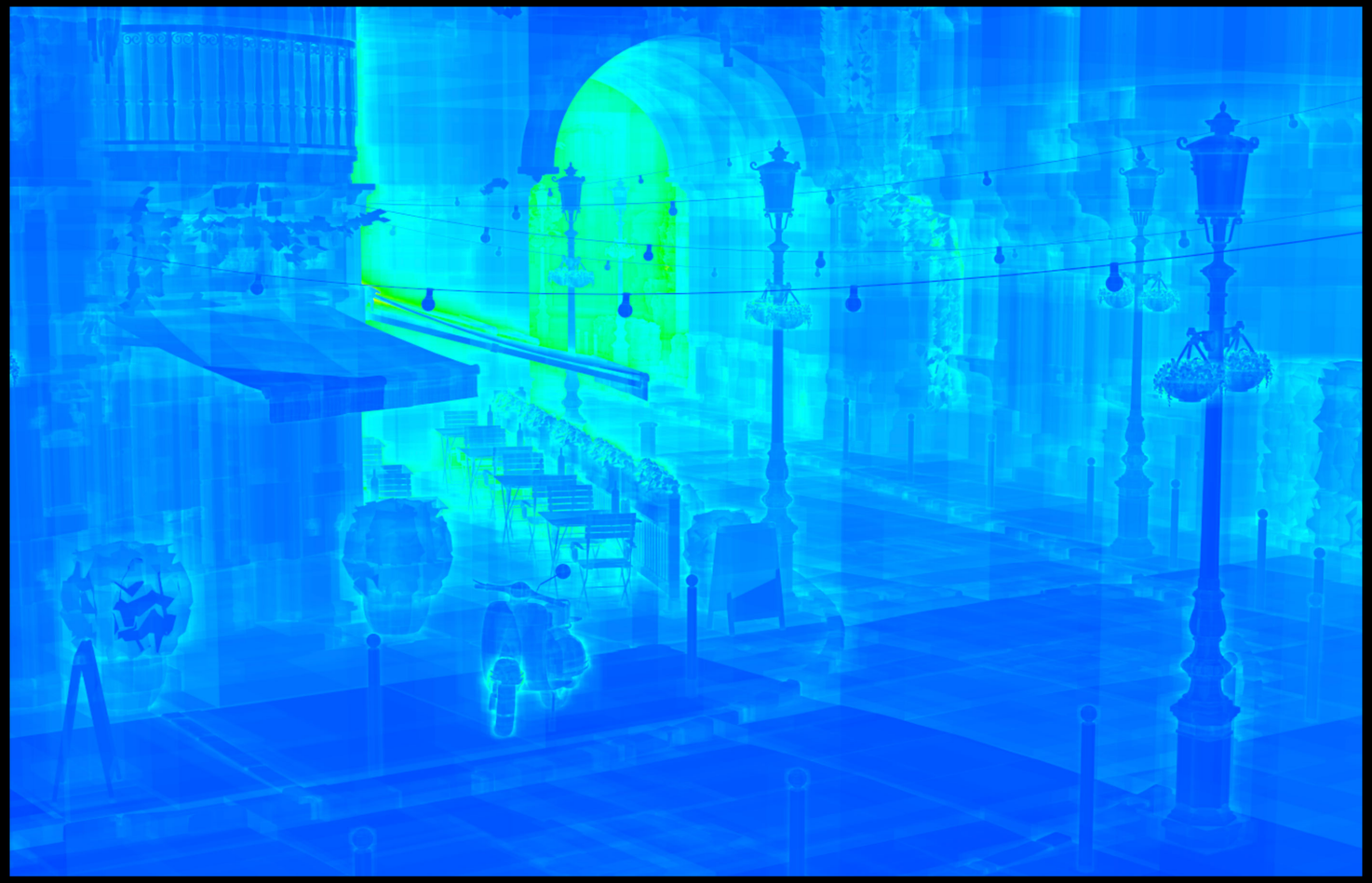
Accelerating Ray/Scene Intersection



Accelerating Ray/Scene Intersection

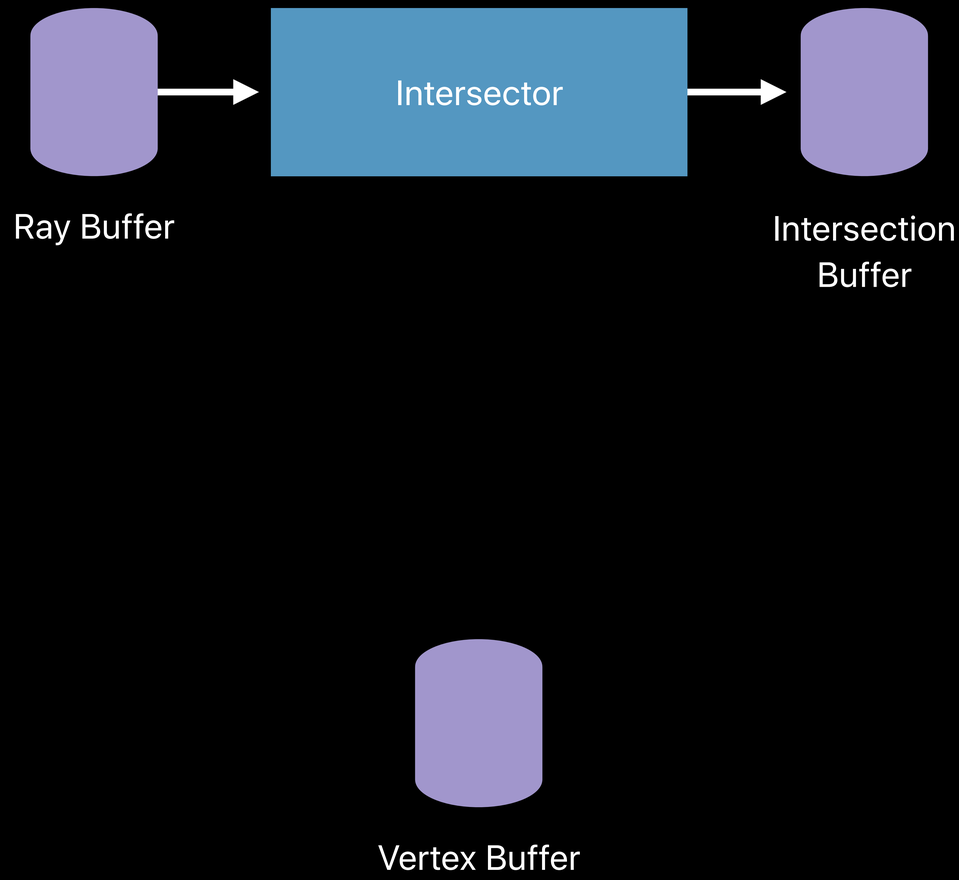


Accelerating Ray/Scene Intersection



MPSRayIntersector

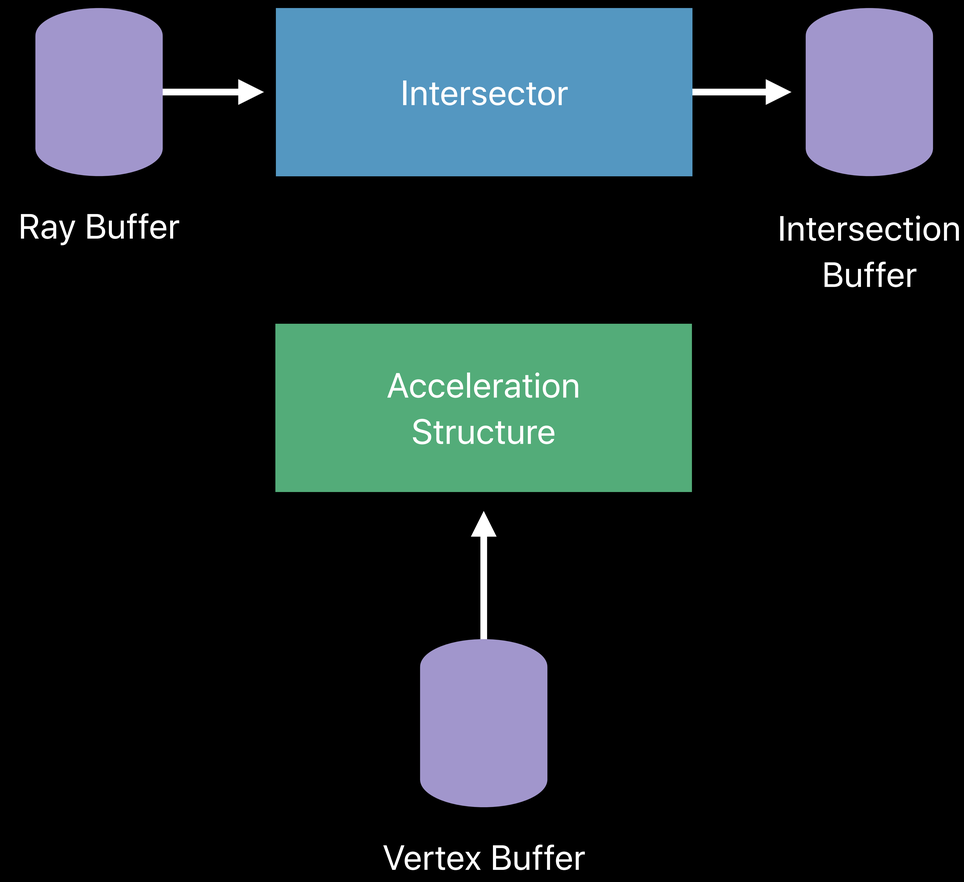
Scene represented by triangle vertices
in a vertex buffer



MPSRayIntersector

Scene represented by triangle vertices in a vertex buffer

Build an acceleration structure over the vertex buffer

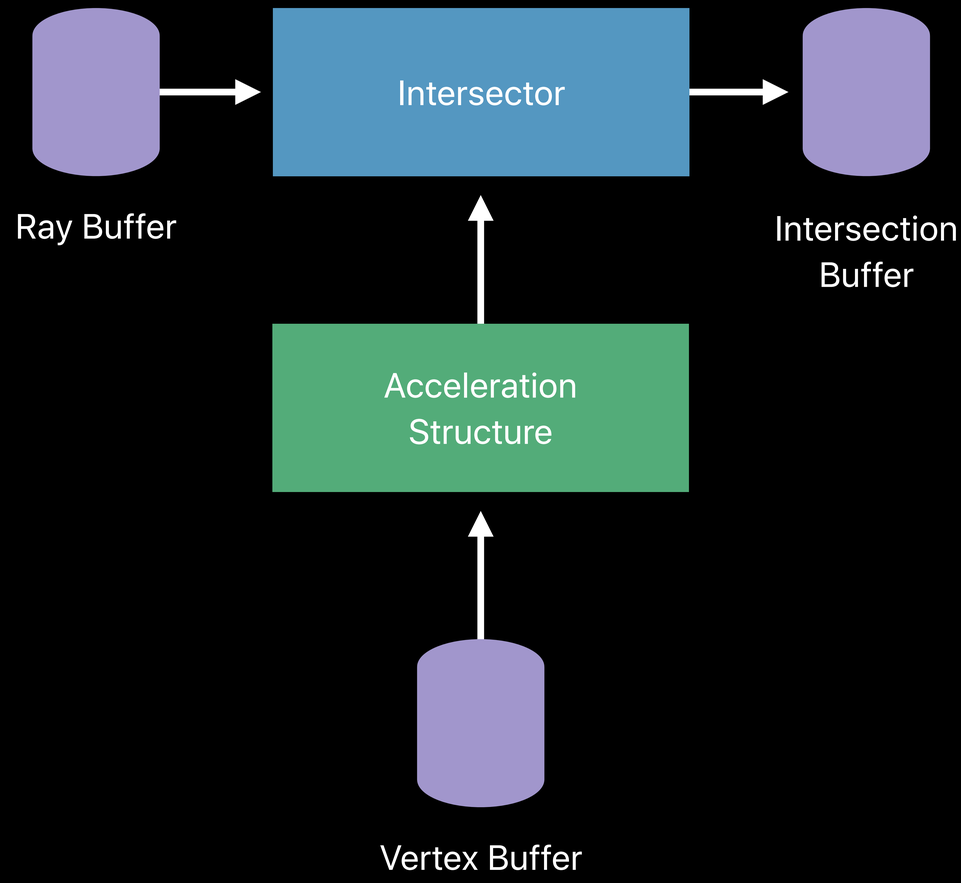


MPSRayIntersector

Scene represented by triangle vertices in a vertex buffer

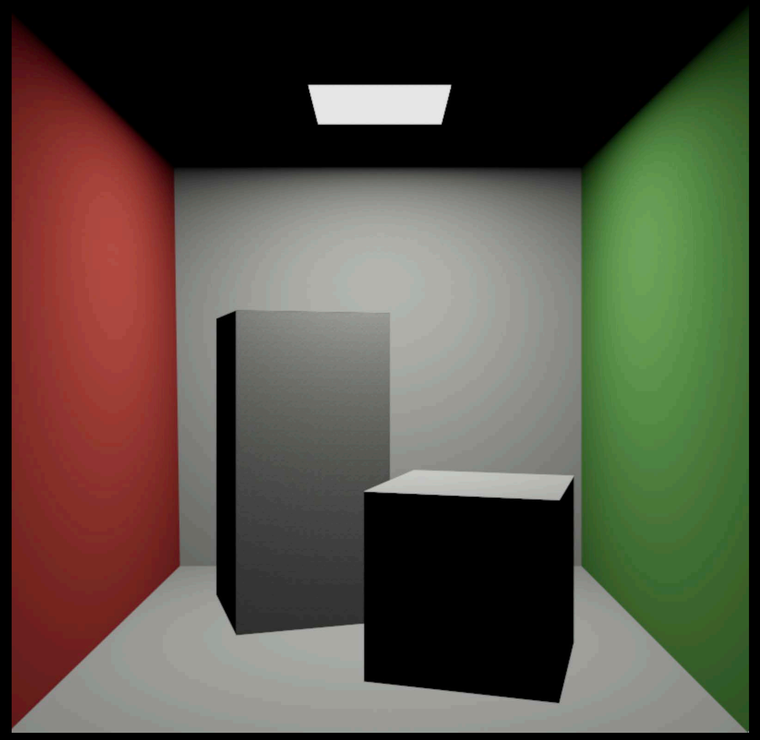
Build an acceleration structure over the vertex buffer

Pass acceleration structure to intersector



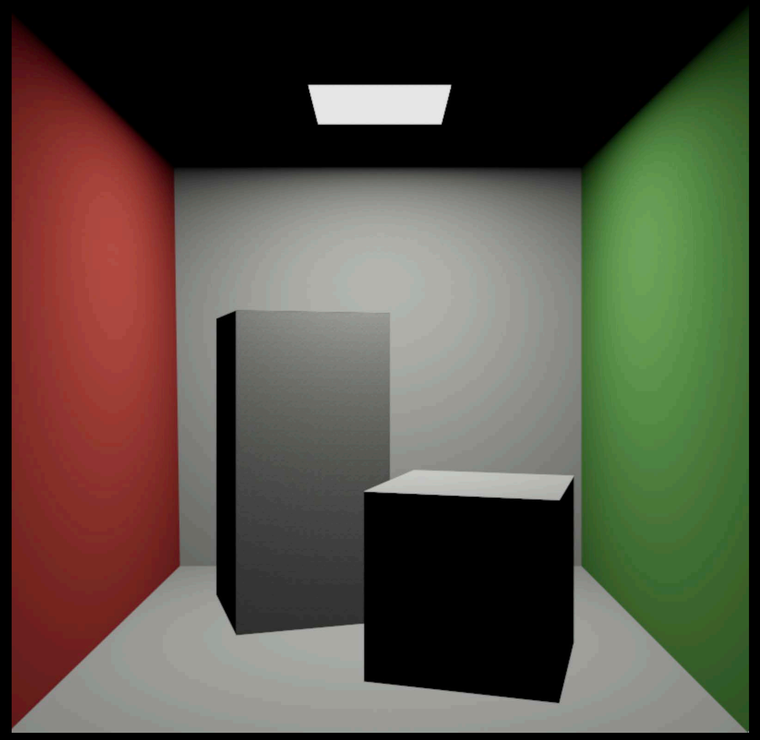
Ray Tracing with Metal Performance Shaders

Ray Tracing with Metal Performance Shaders

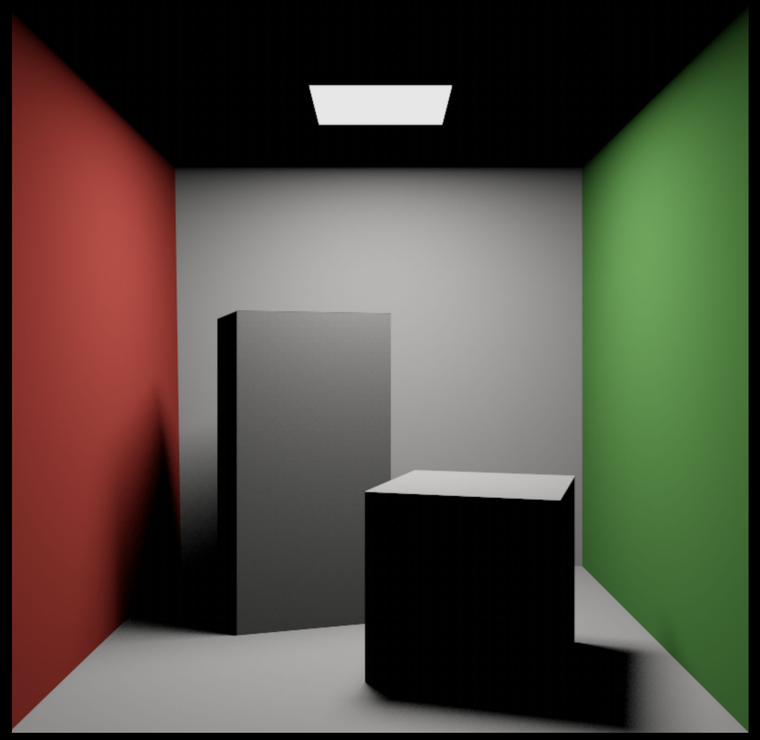


Primary Rays and Shading

Ray Tracing with Metal Performance Shaders

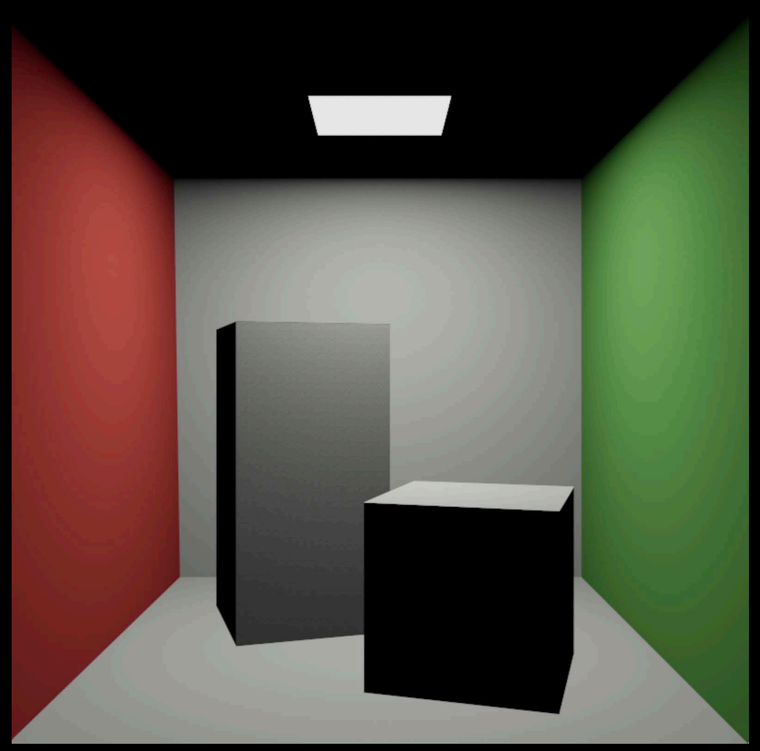


Primary Rays and Shading

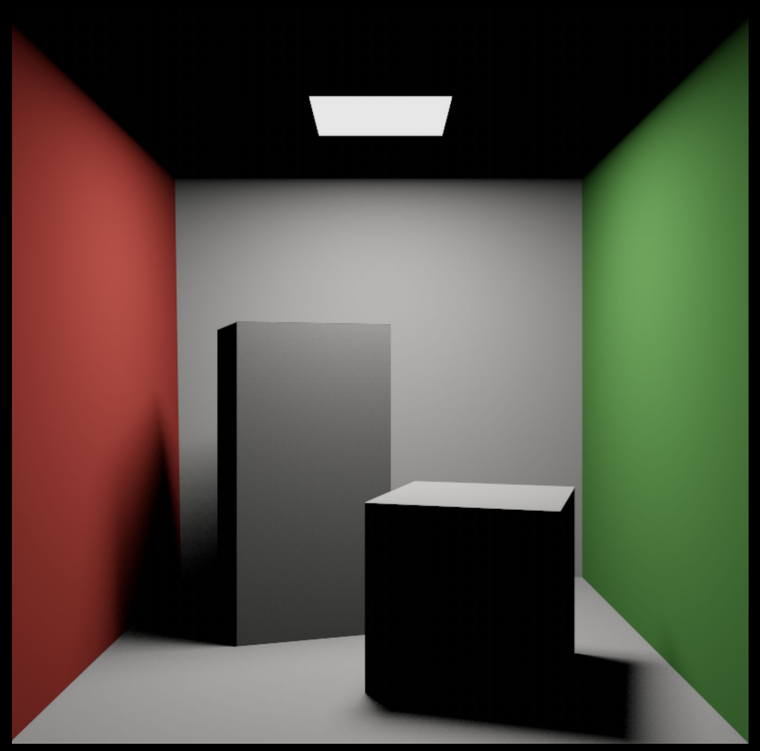


Shadow Rays

Ray Tracing with Metal Performance Shaders



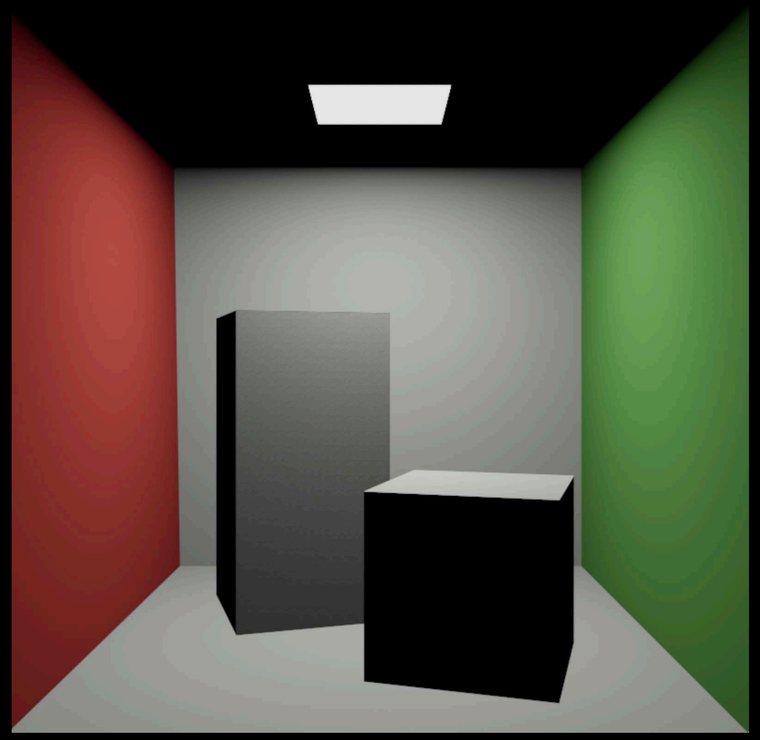
Primary Rays and Shading



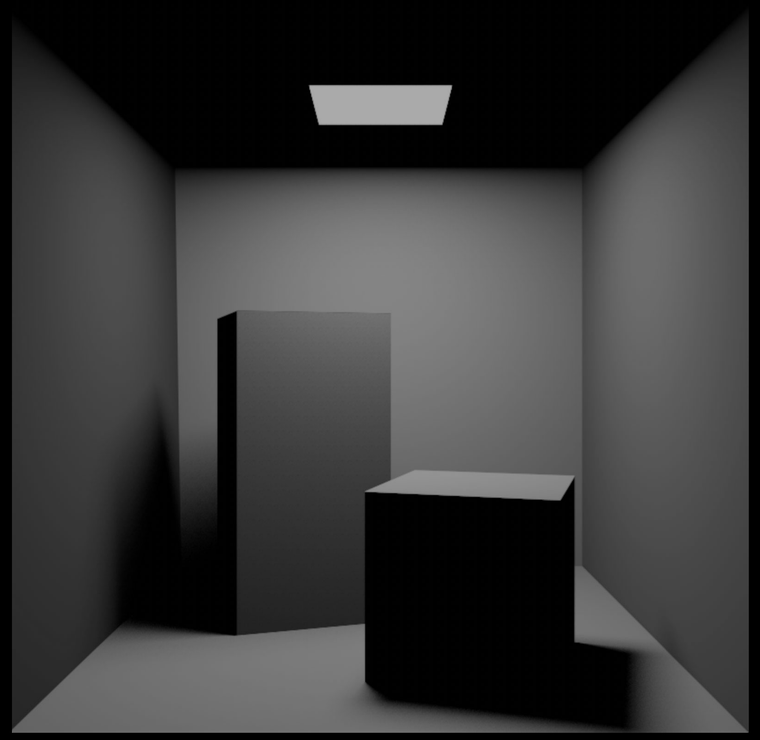
Shadow Rays

Secondary Rays

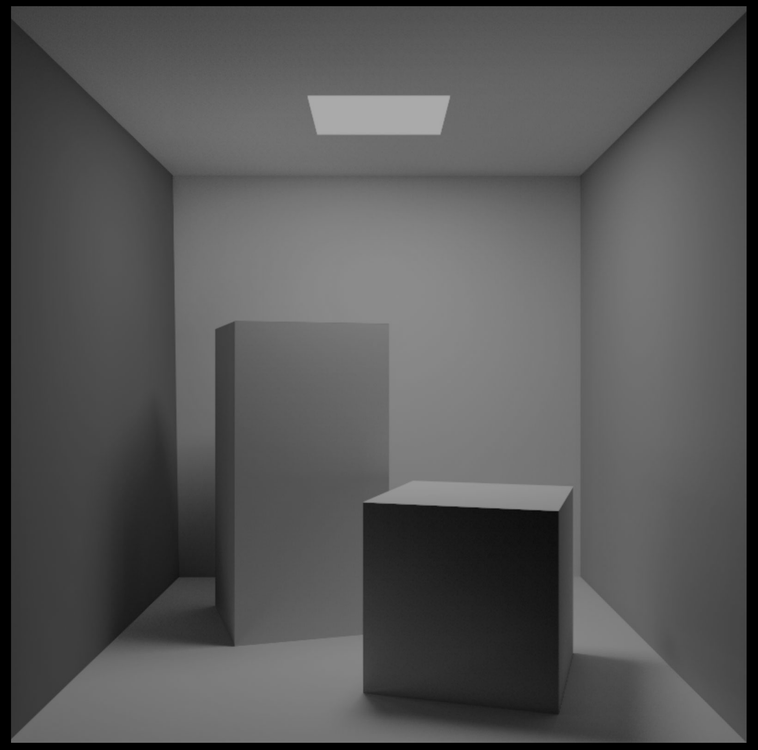
Ray Tracing with Metal Performance Shaders



Primary Rays and Shading



Shadow Rays

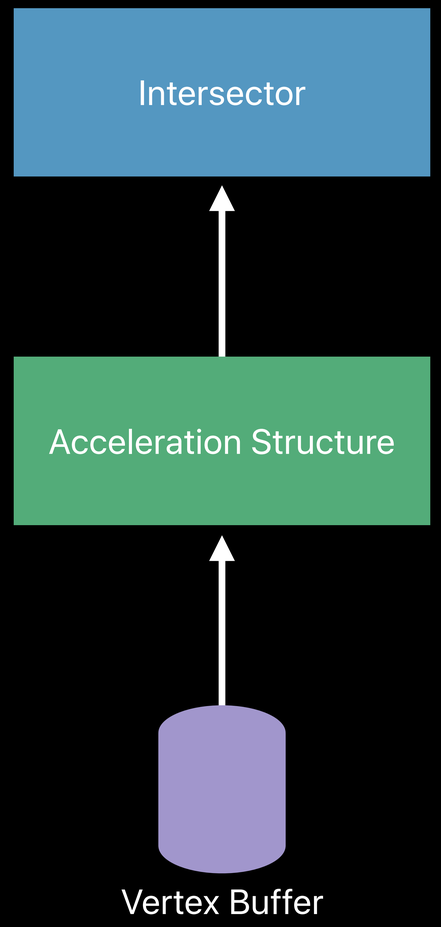


Secondary Rays

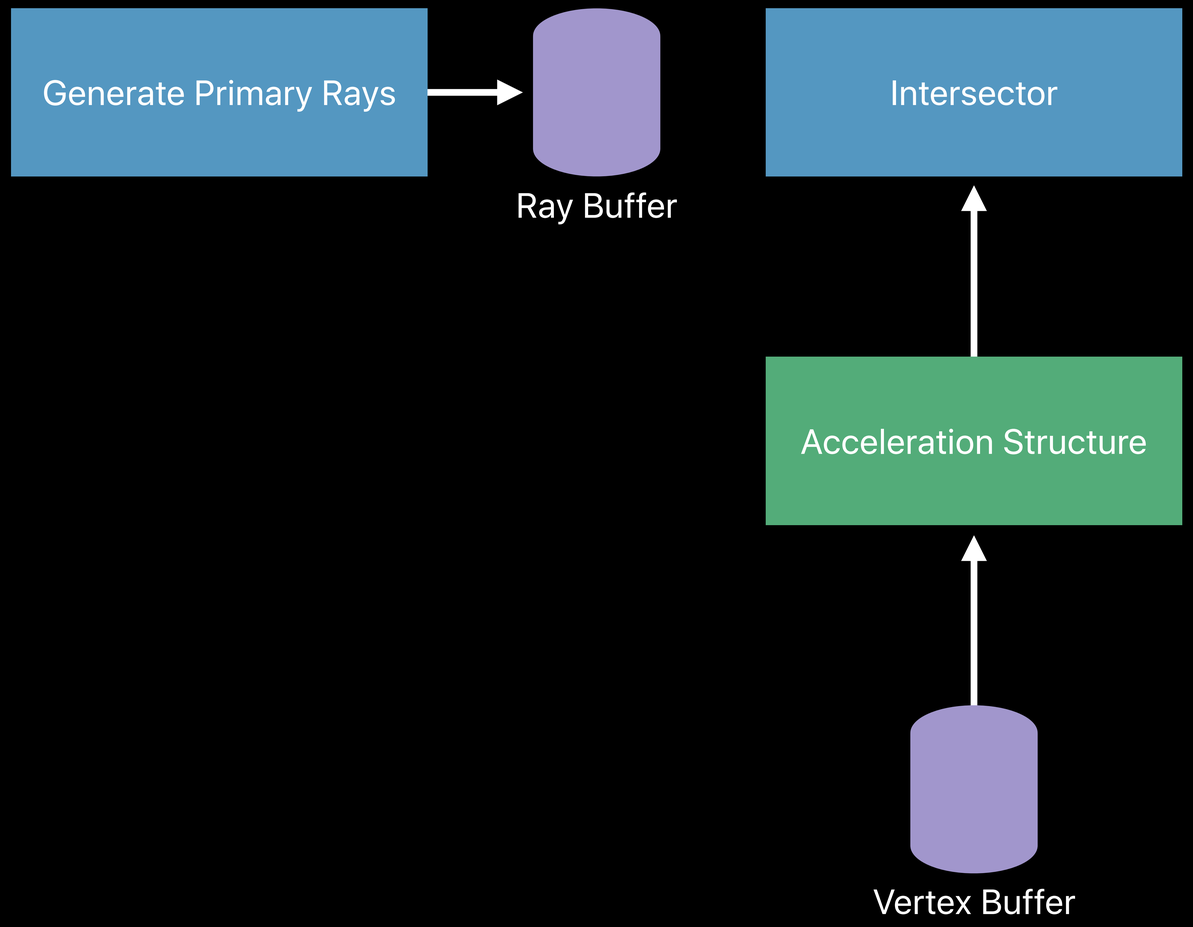
Primary Rays and Shading

Primary Rays and Shading

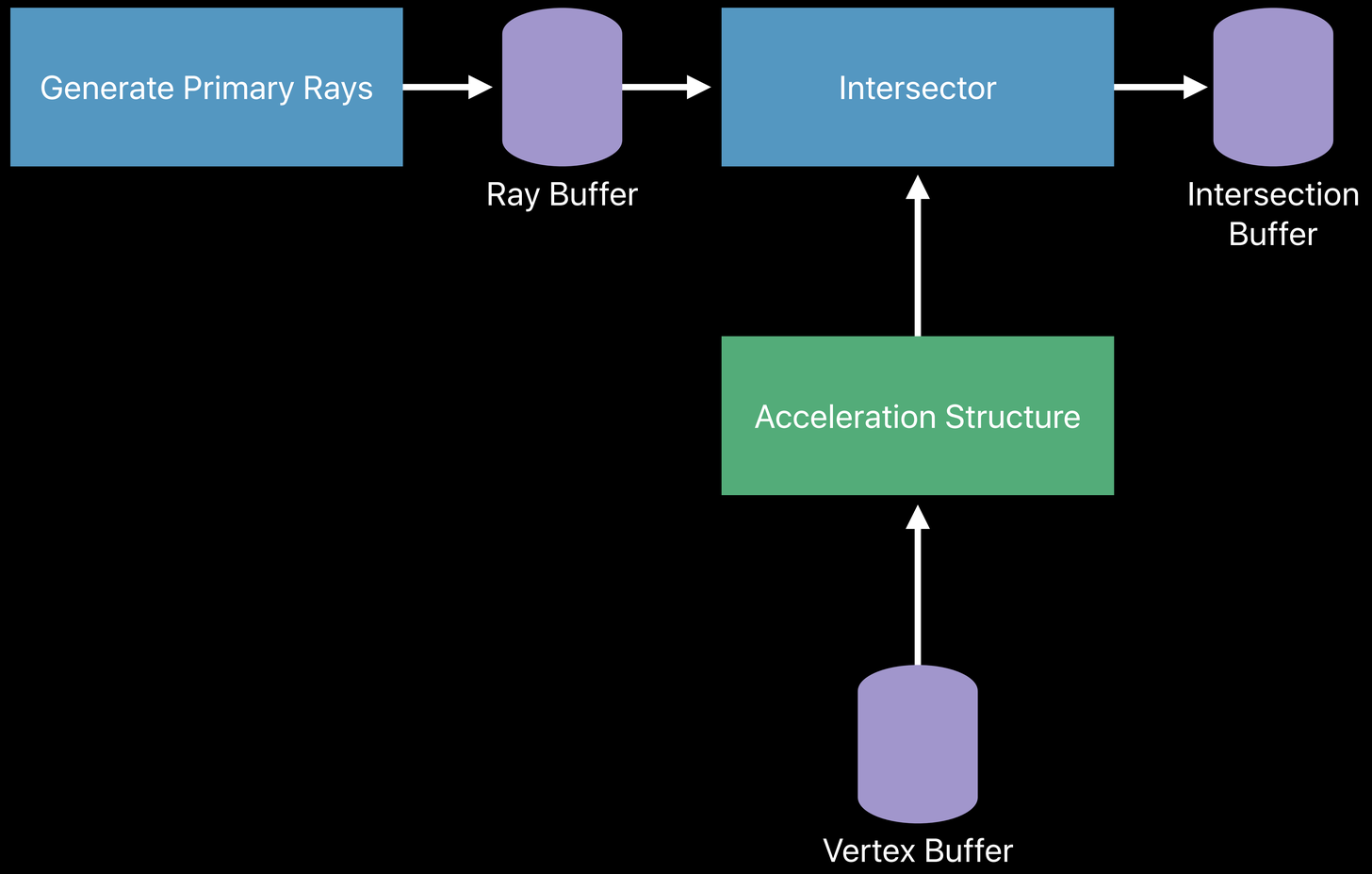
Primary Rays and Shading



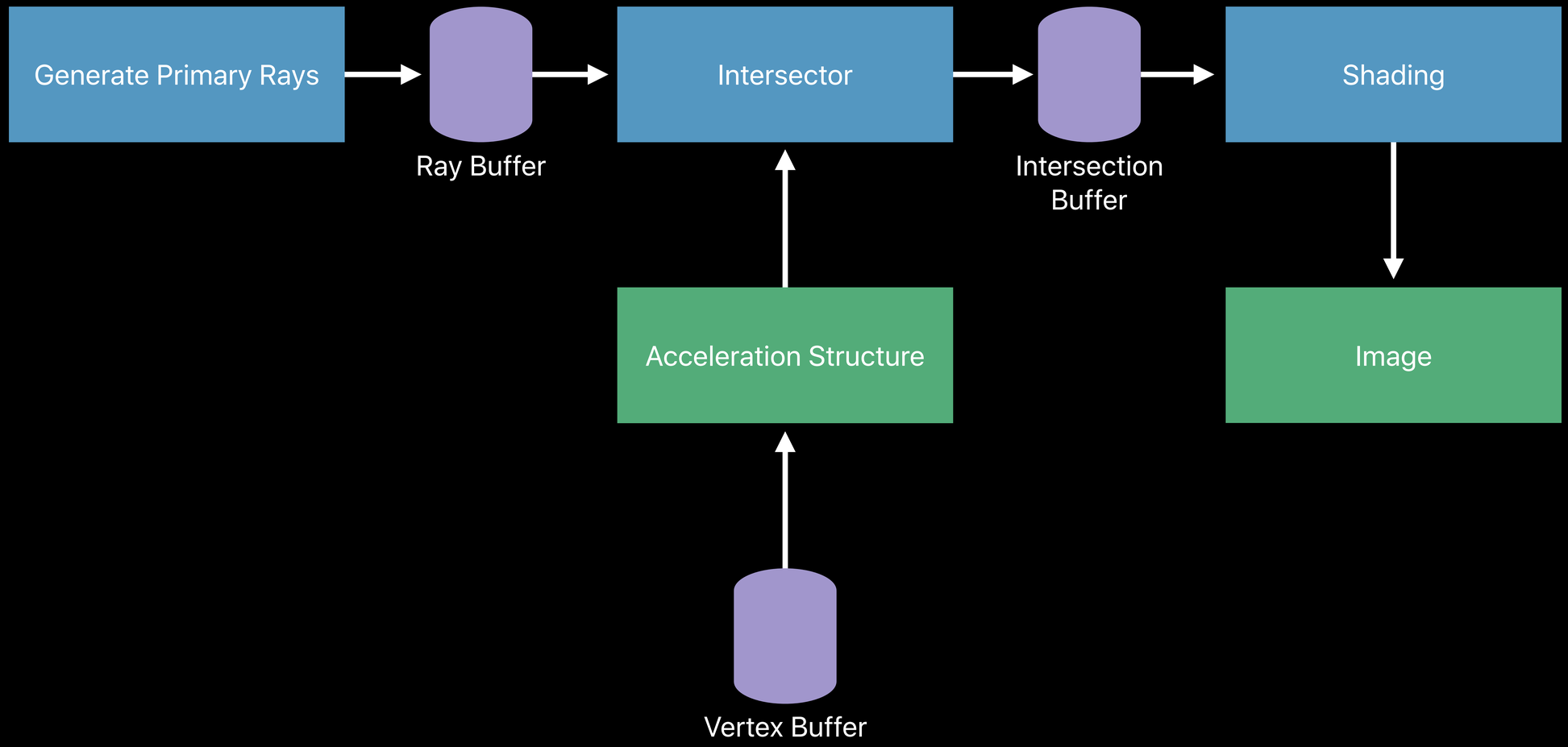
Primary Rays and Shading



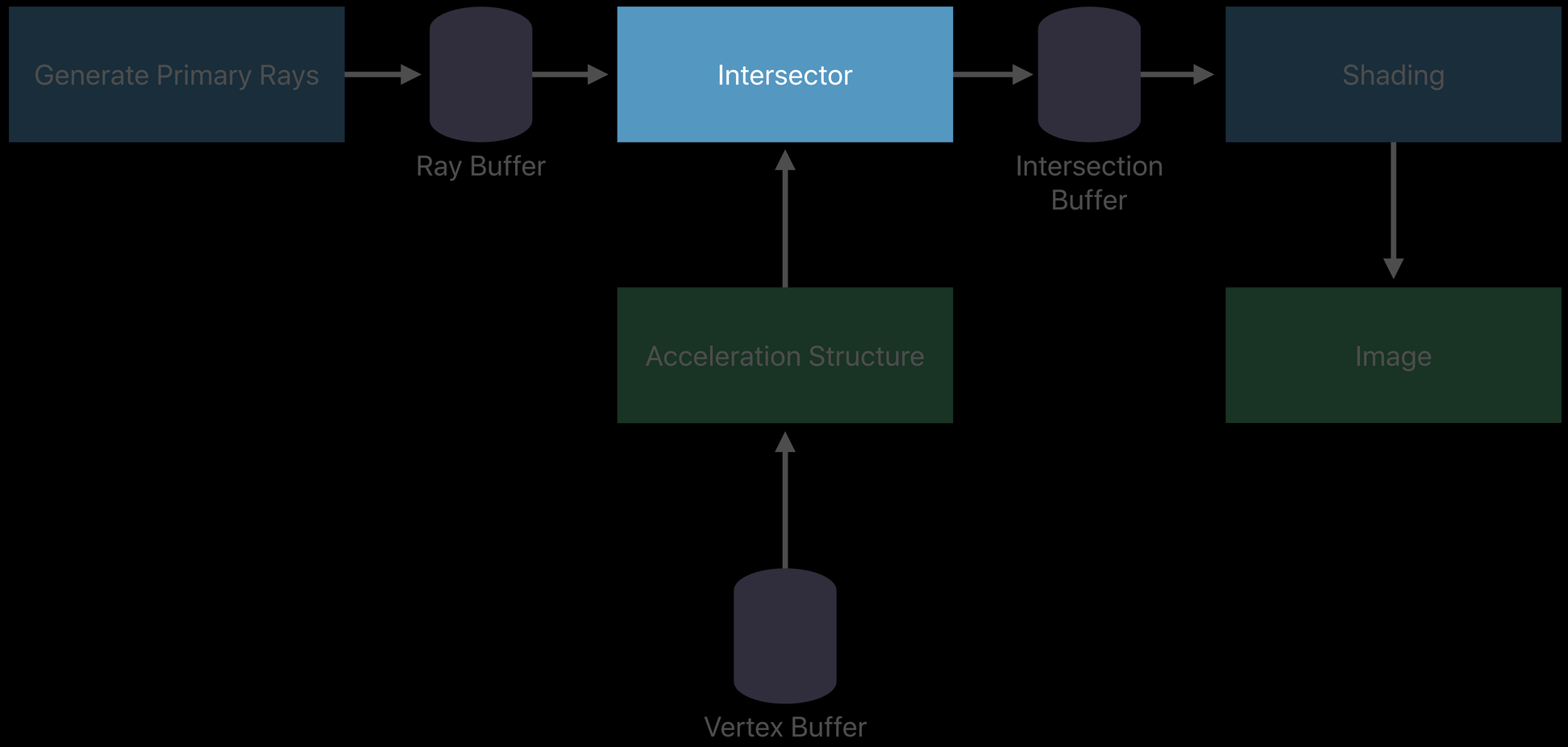
Primary Rays and Shading



Primary Rays and Shading



Primary Rays and Shading

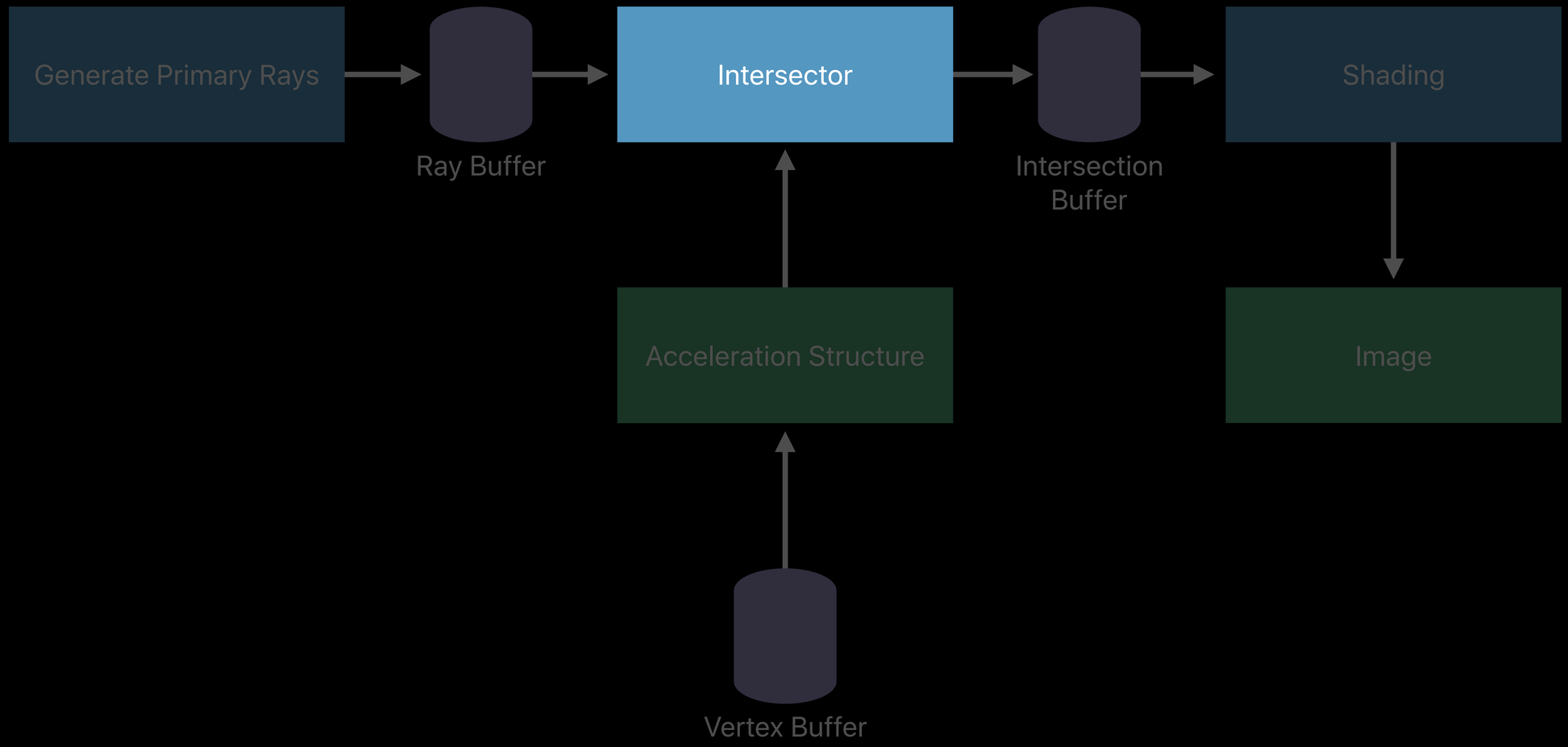


Creating an Intersector

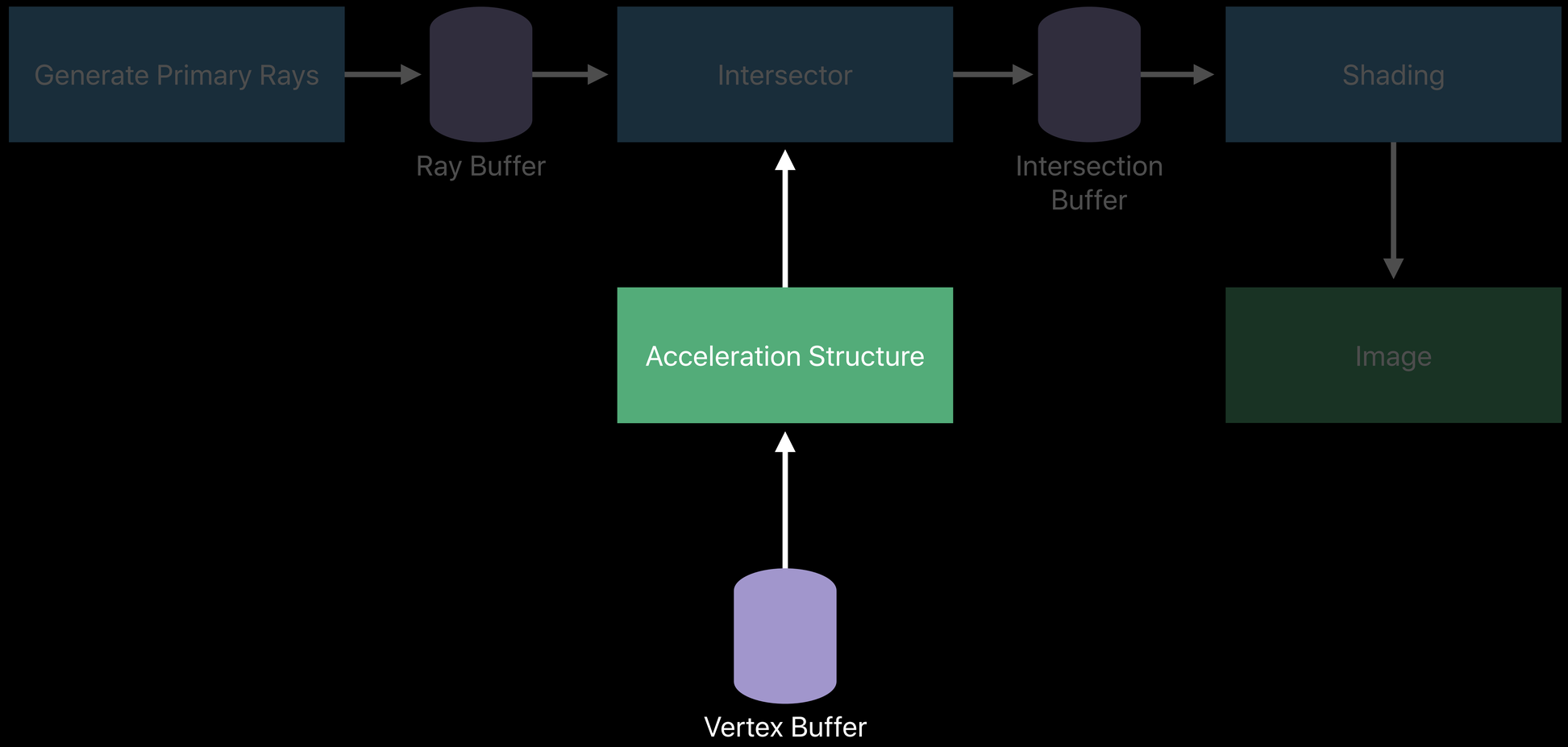
Create an MPSRayIntersector with a Metal device:

```
let intersector = MPSRayIntersector(device: device)
```

Primary Rays and Shading



Primary Rays and Shading



Creating an Acceleration Structure

Create an MPSTriangleAccelerationStructure with a Metal device:

```
let accelerationStructure = MPSTriangleAccelerationStructure(device: device)
```

Creating an Acceleration Structure

Create an MPSTriangleAccelerationStructure with a Metal device:

```
let accelerationStructure = MPSTriangleAccelerationStructure(device: device)
```

Assign vertex buffer:

```
accelerationStructure.vertexBuffer = vertexBuffer
accelerationStructure.triangleCount = triangleCount
```

Creating an Acceleration Structure

Create an MPSTriangleAccelerationStructure with a Metal device:

```
let accelerationStructure = MPSTriangleAccelerationStructure(device: device)
```

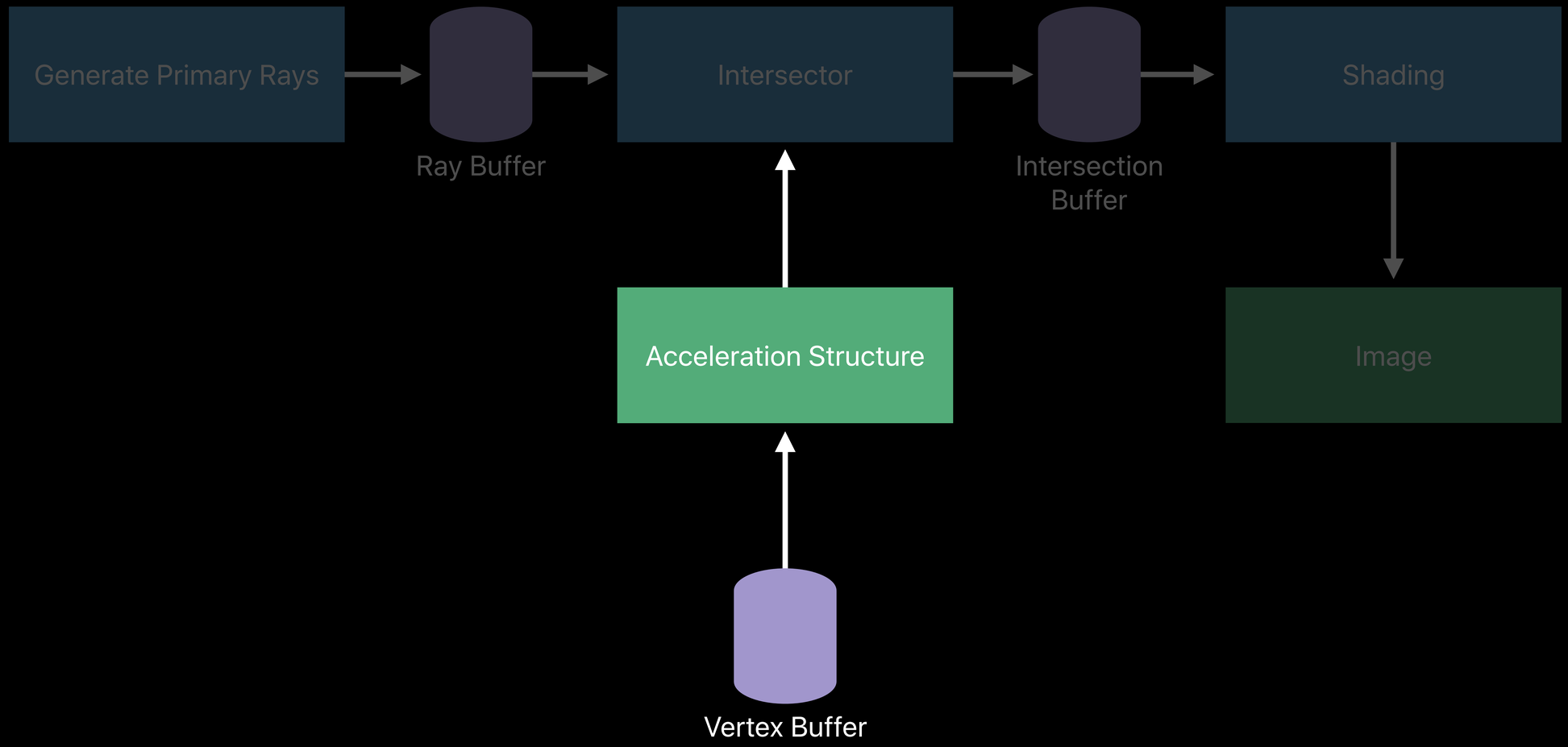
Assign vertex buffer:

```
accelerationStructure.vertexBuffer = vertexBuffer
accelerationStructure.triangleCount = triangleCount
```

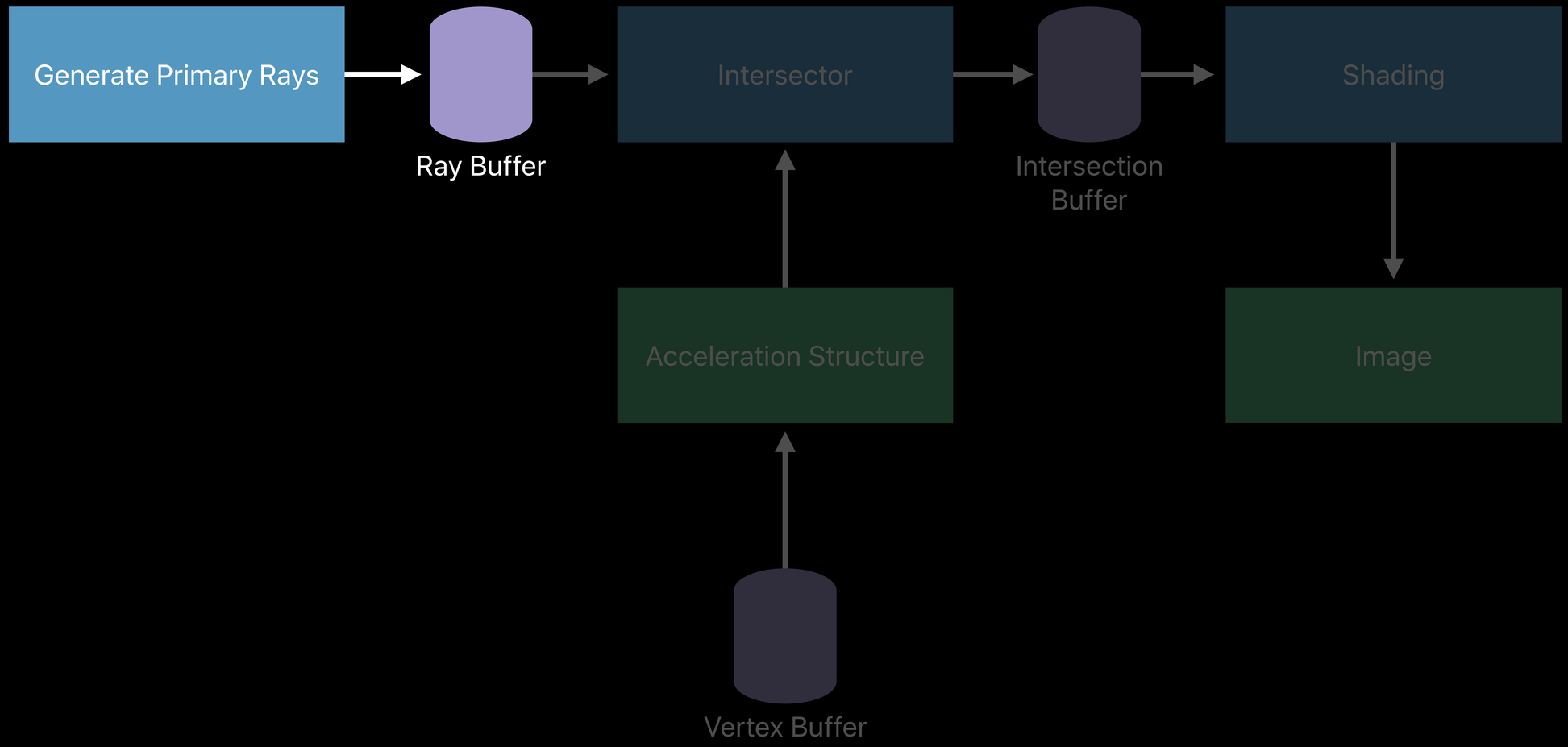
Build acceleration structure:

```
accelerationStructure.rebuild()
```

Primary Rays and Shading

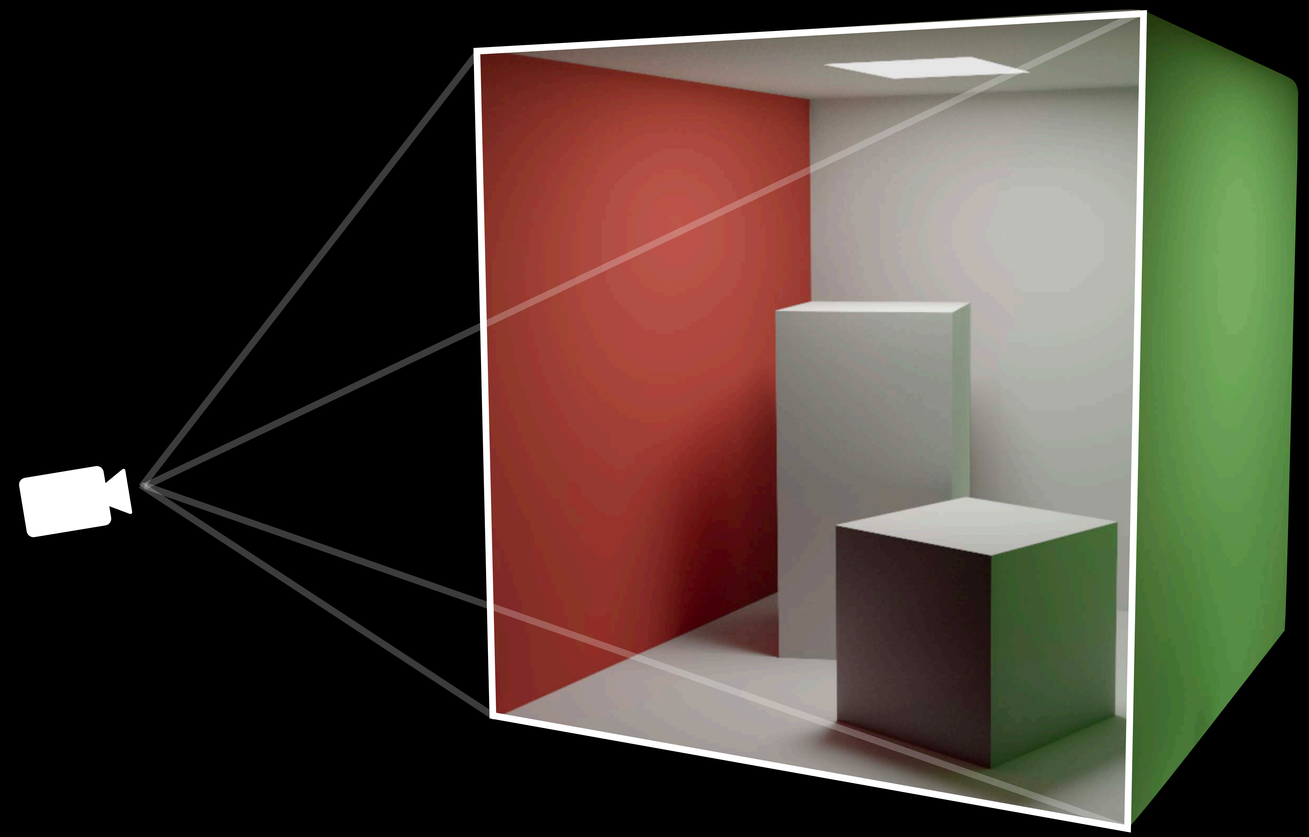


Primary Rays and Shading



Generating Primary Rays

Launch one thread per pixel

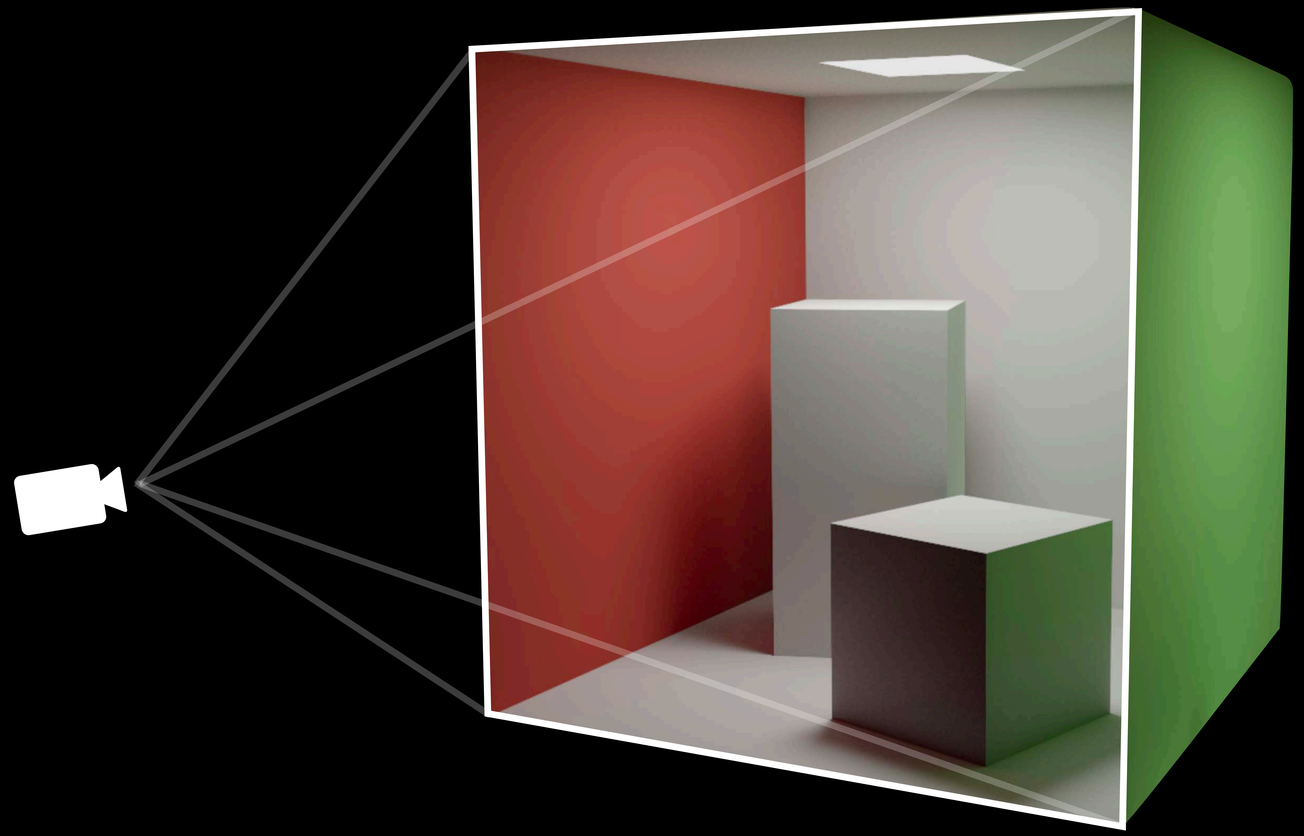


Generating Primary Rays

Launch one thread per pixel

Write Ray struct to ray buffer:

```
struct Ray {
    float3 origin;
    float3 direction;
};
```

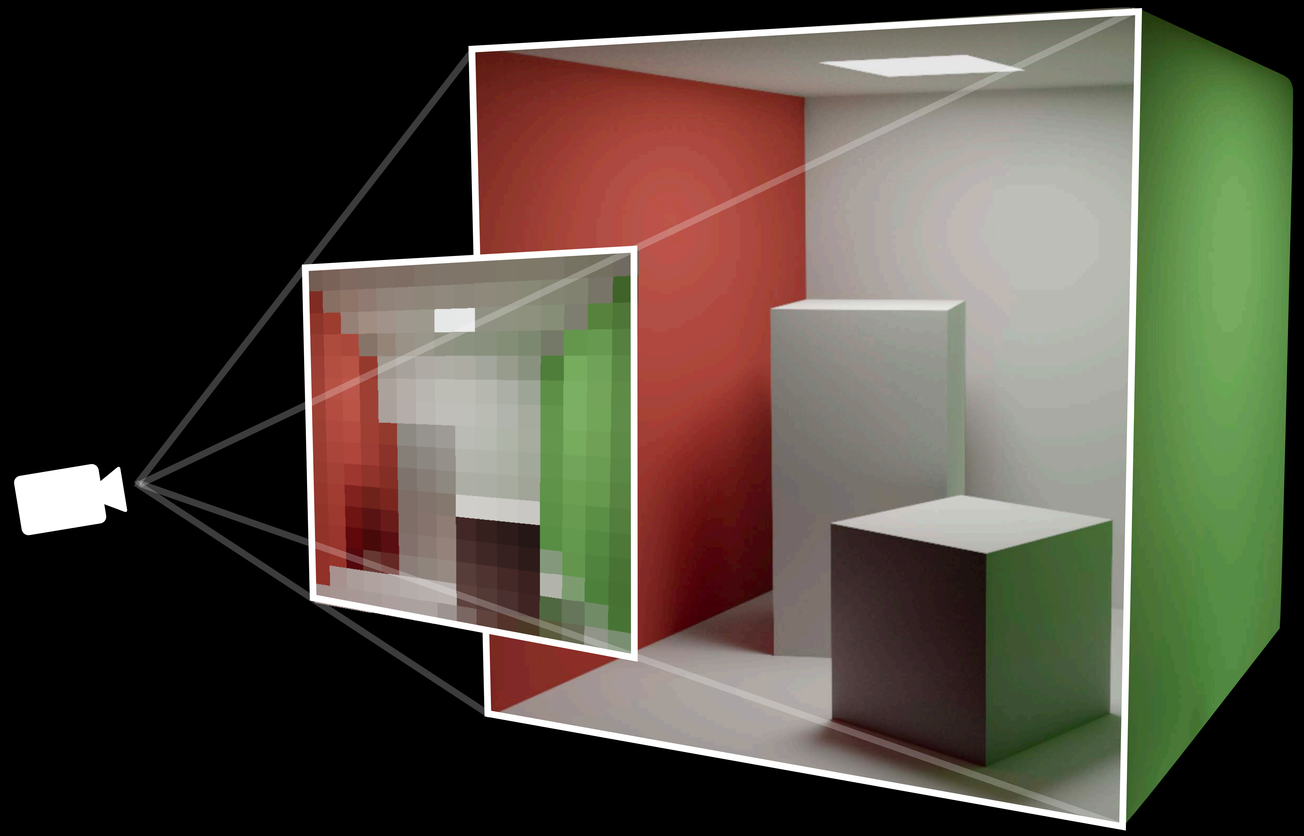


Generating Primary Rays

Launch one thread per pixel

Write Ray struct to ray buffer:

```
struct Ray {
    float3 origin;
    float3 direction;
};
```

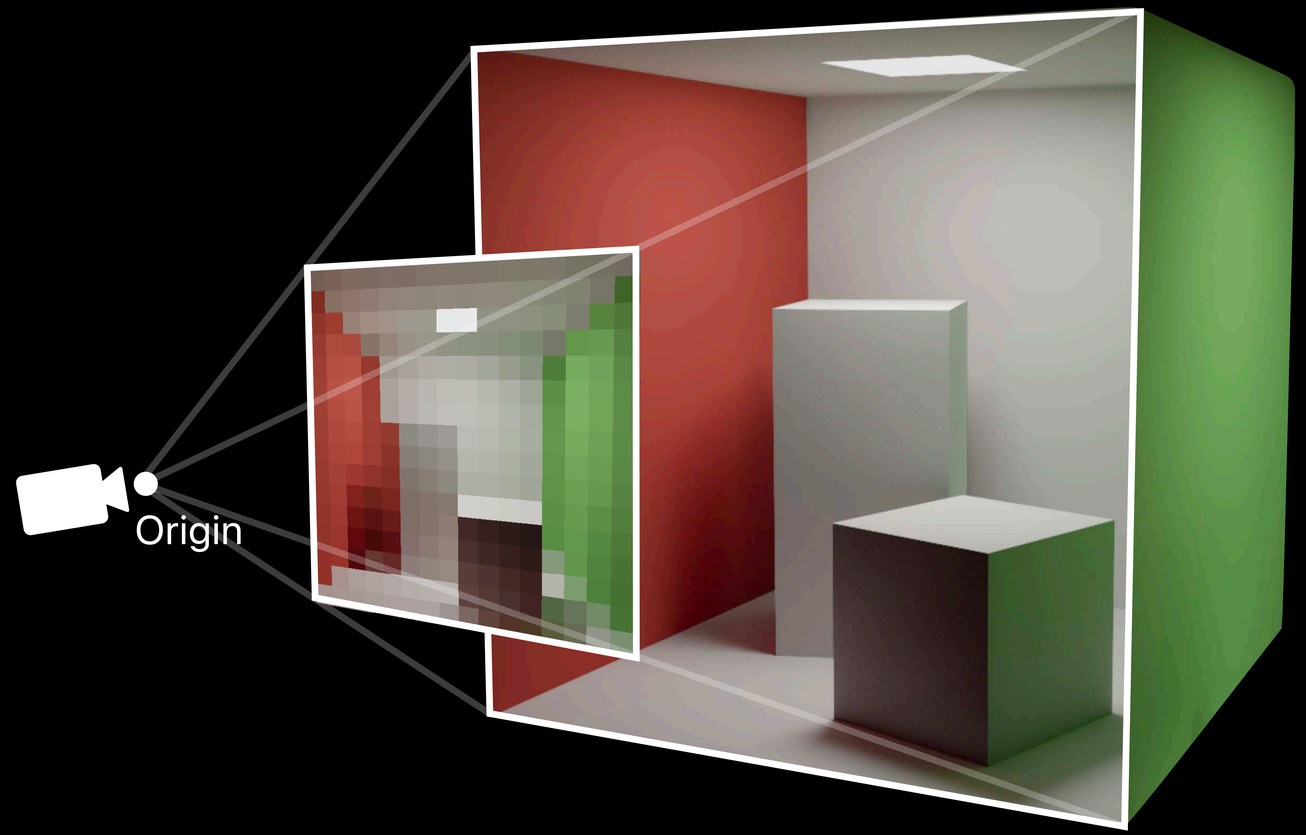


Generating Primary Rays

Launch one thread per pixel

Write Ray struct to ray buffer:

```
struct Ray {
    float3 origin;
    float3 direction;
};
```

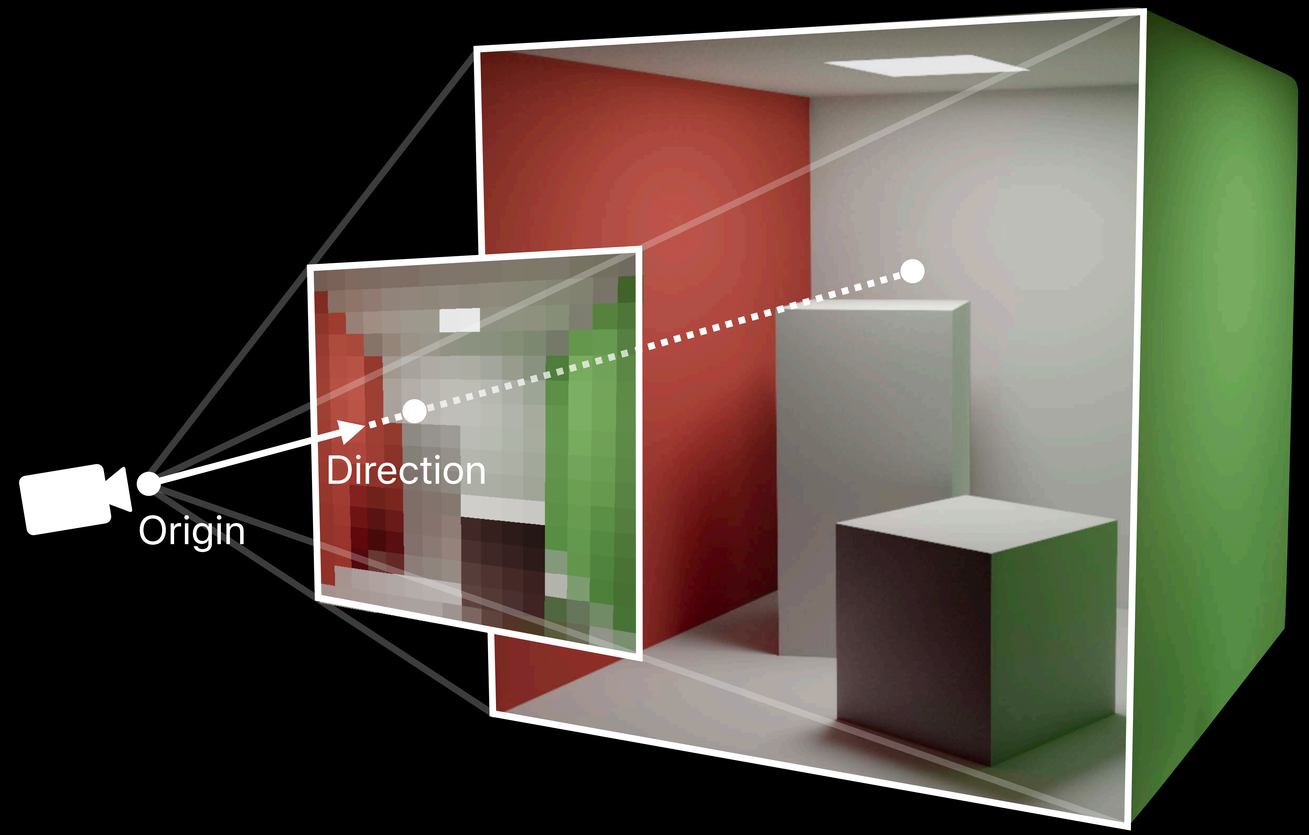


Generating Primary Rays

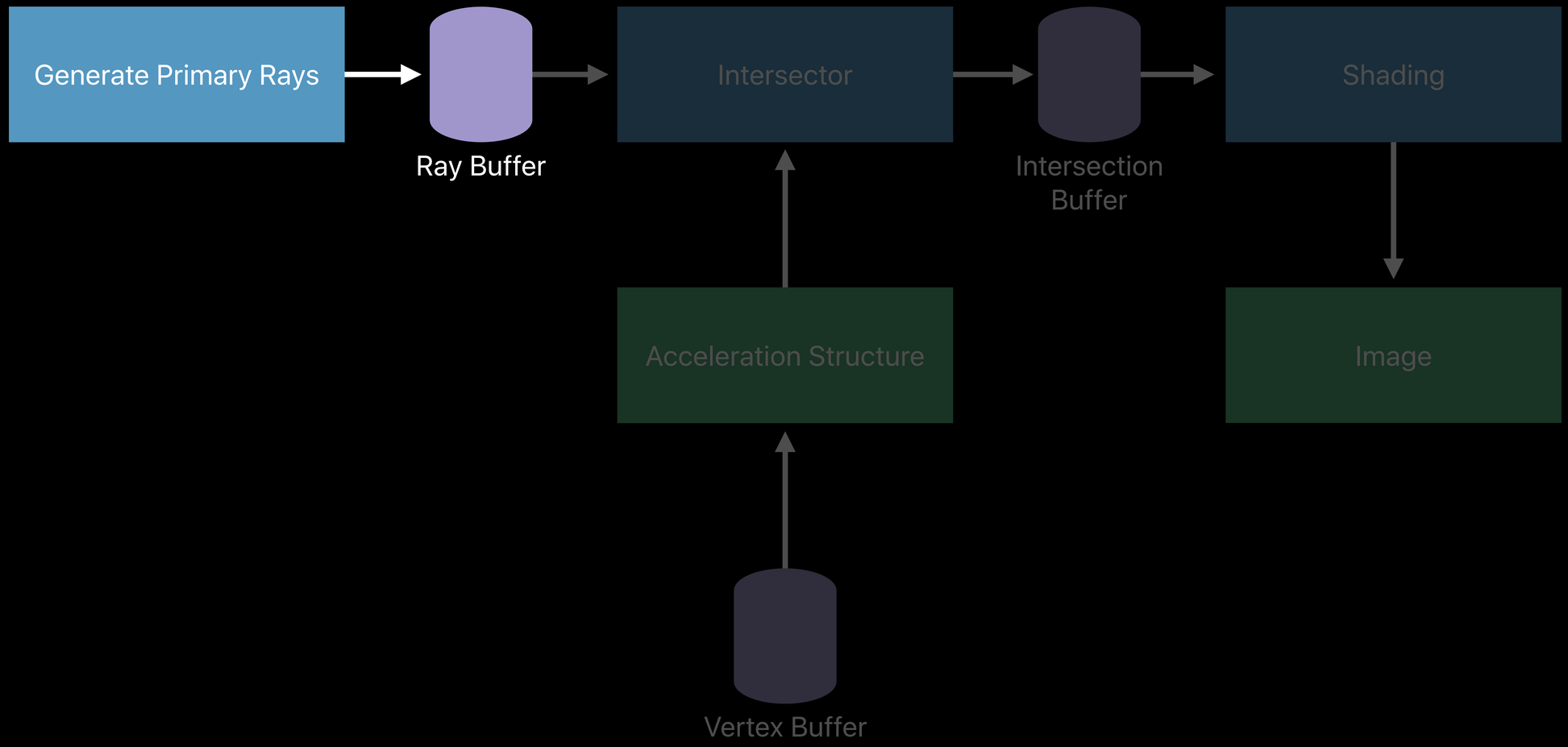
Launch one thread per pixel

Write Ray struct to ray buffer:

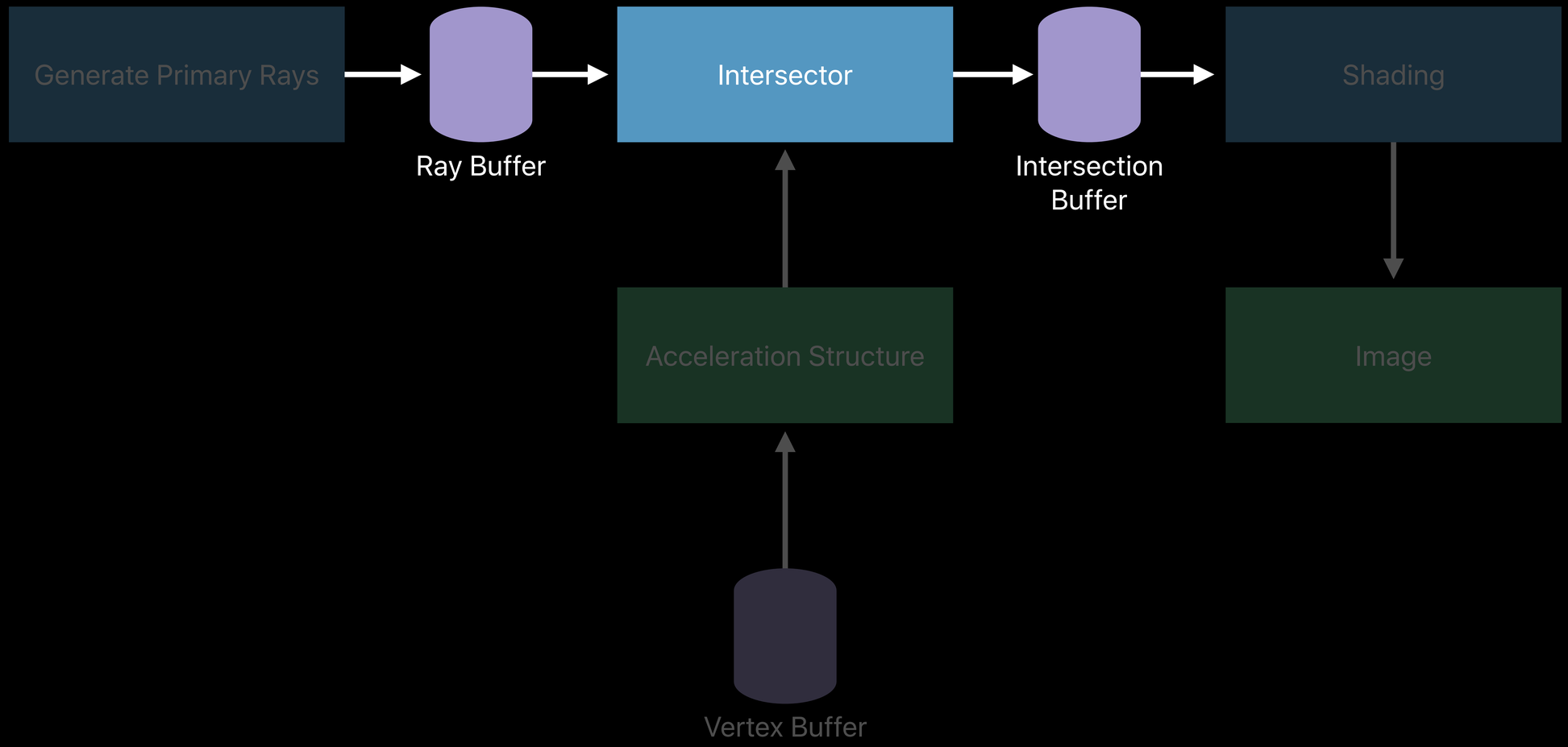
```
struct Ray {
    float3 origin;
    float3 direction;
};
```



Primary Rays and Shading



Primary Rays and Shading



Finding Intersections with the Scene

Encode intersection test into a command buffer:

```
intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Intersections with the Scene

Encode intersection test into a command buffer:

```
intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Intersections with the Scene

Encode intersection test into a command buffer:

```
intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Intersections with the Scene

Encode intersection test into a command buffer:

```
intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Intersections with the Scene

Encode intersection test into a command buffer:

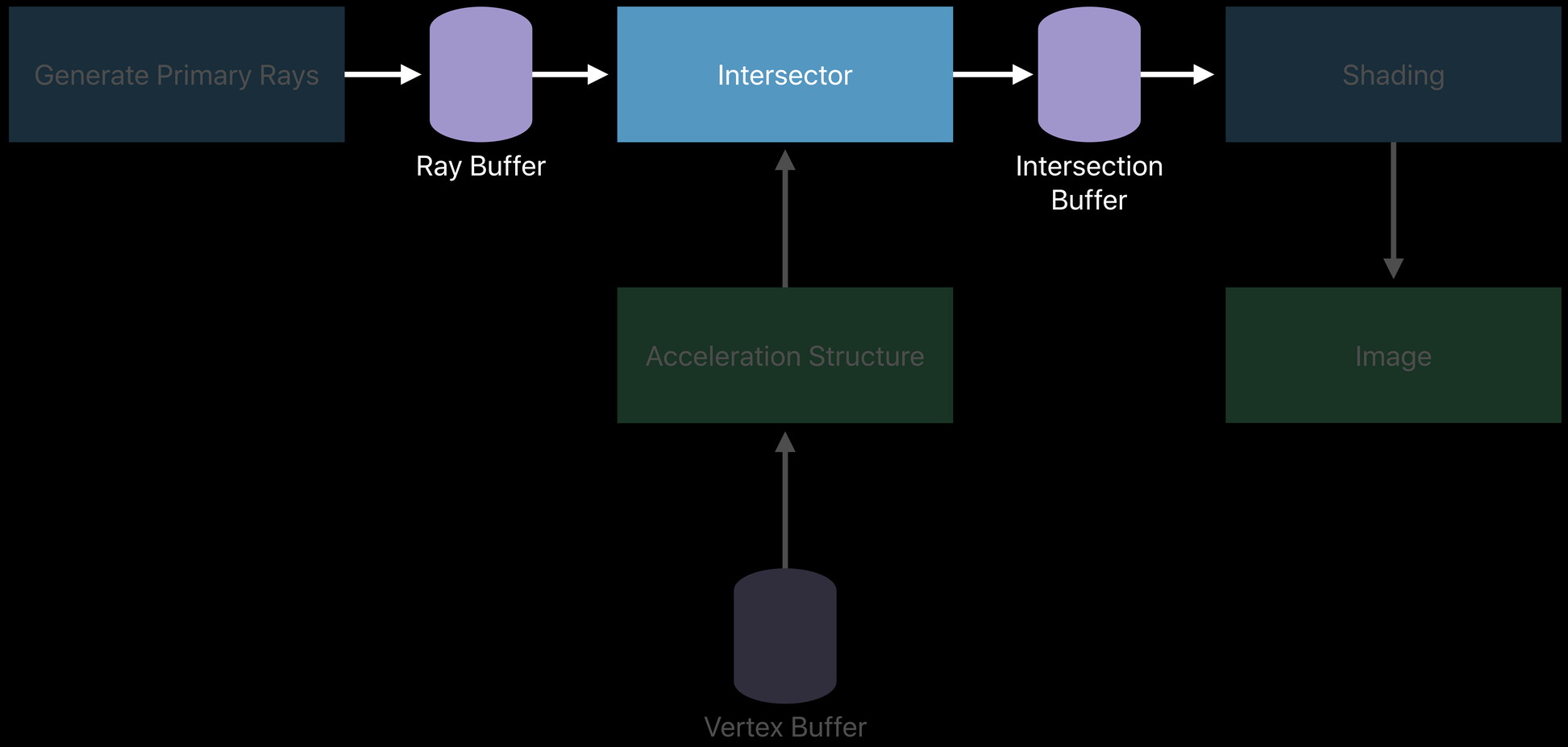
```
intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Intersections with the Scene

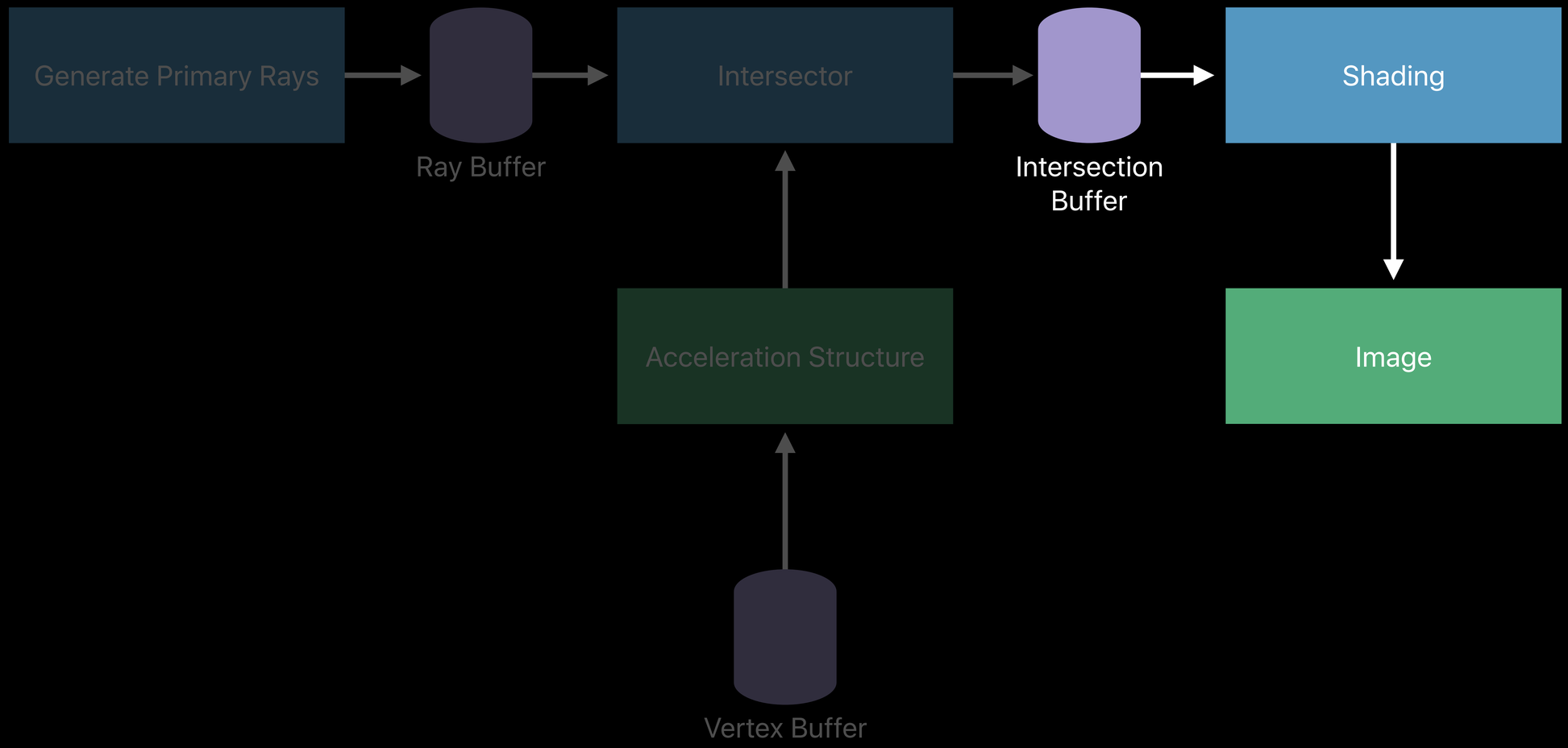
Encode intersection test into a command buffer:

```
intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Primary Rays and Shading



Primary Rays and Shading



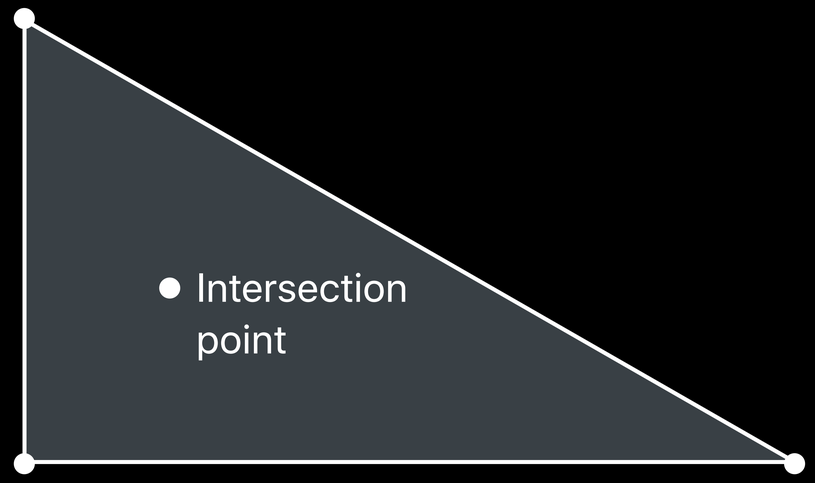
Shading

Apply lighting and textures similar to fragment shader

Depends on intersection point and vertex attributes

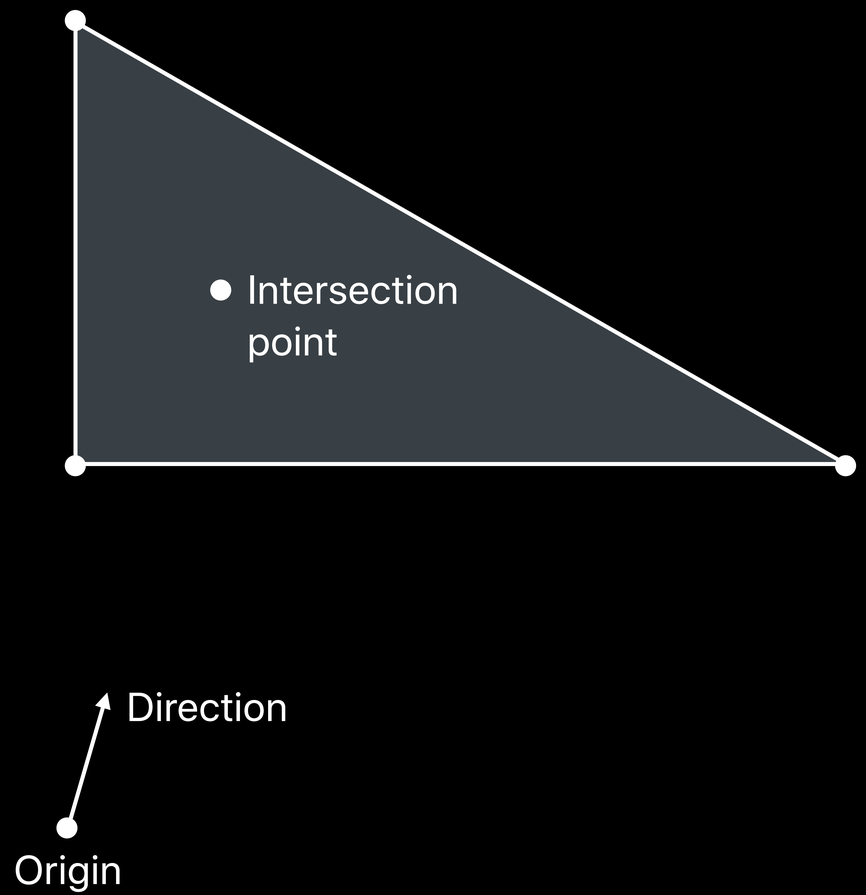
Shading

```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```



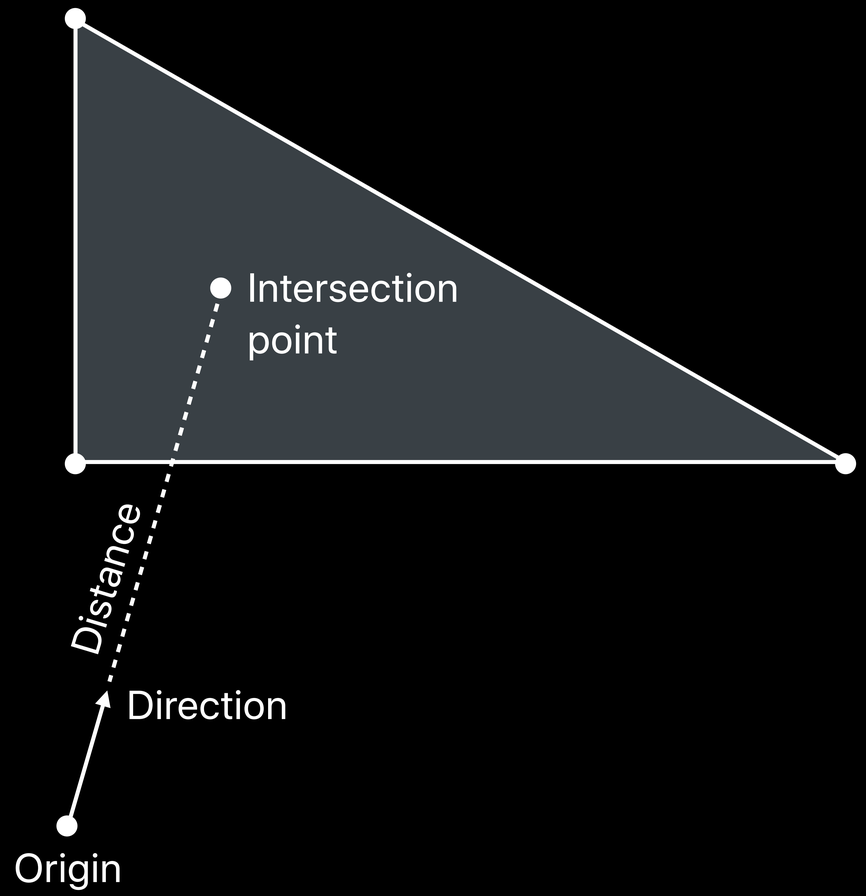
Shading

```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```



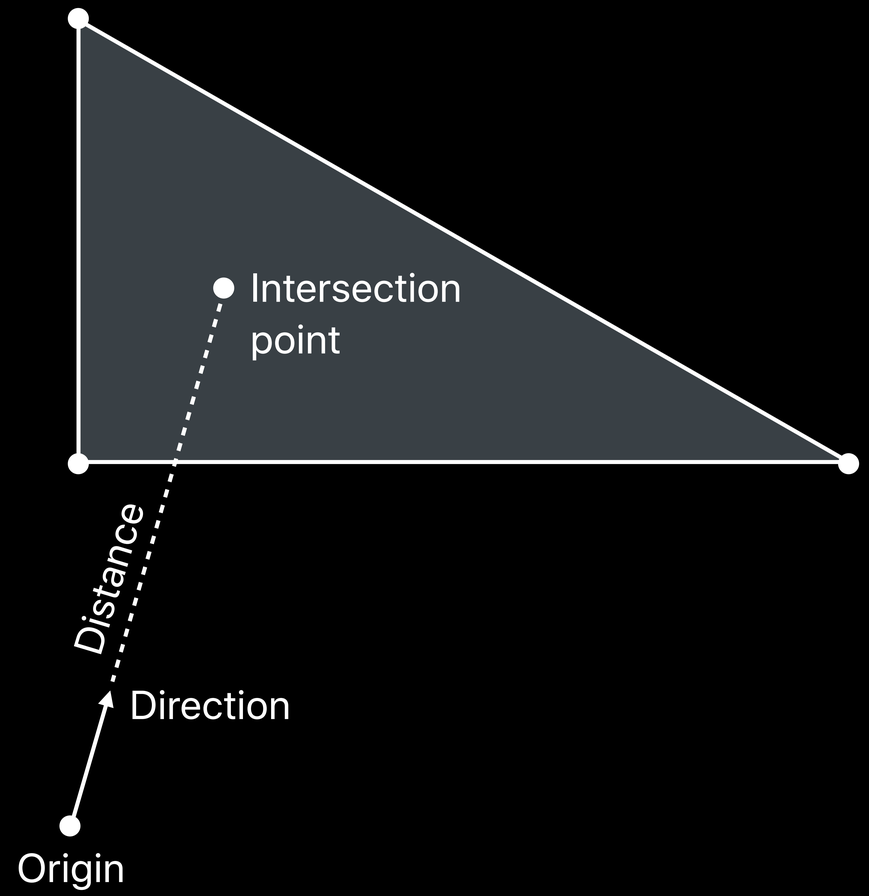
Shading

```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```



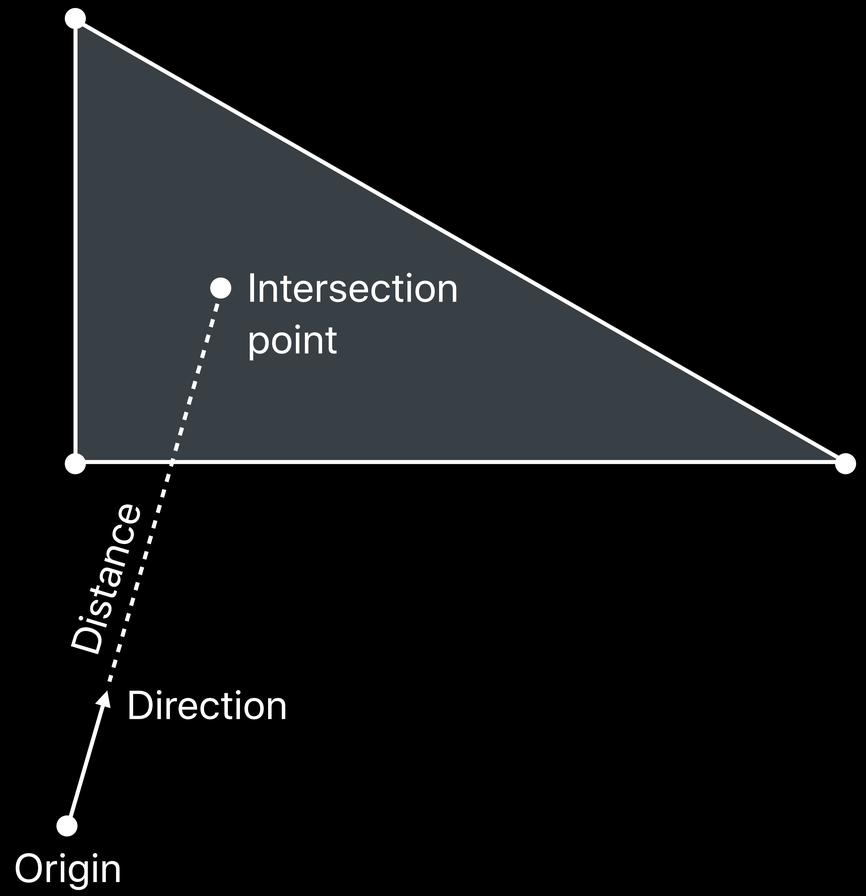
Shading

```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```



Shading

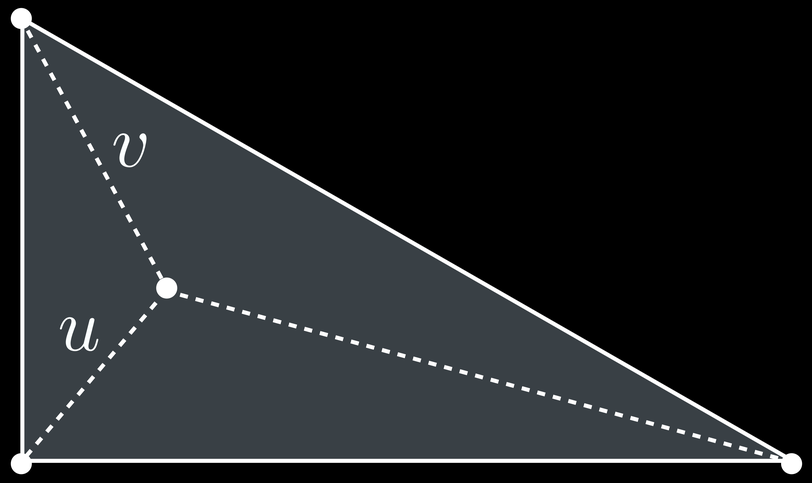
```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```



Shading

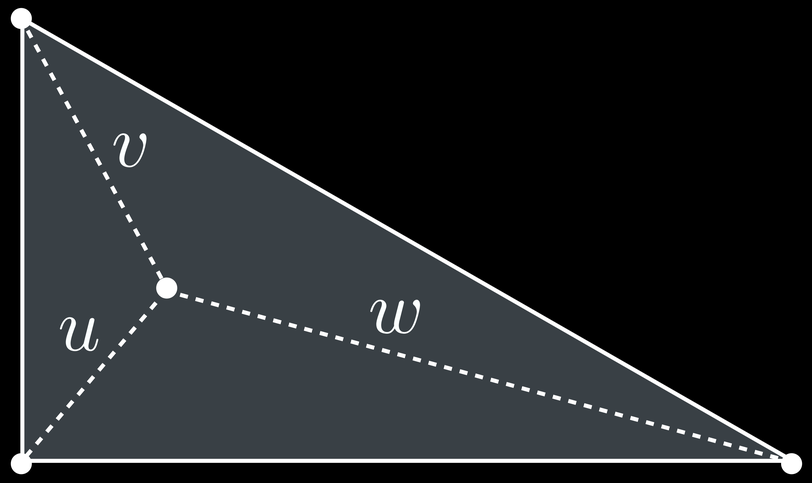
```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```

Barycentric coordinates u and v



Shading

```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```

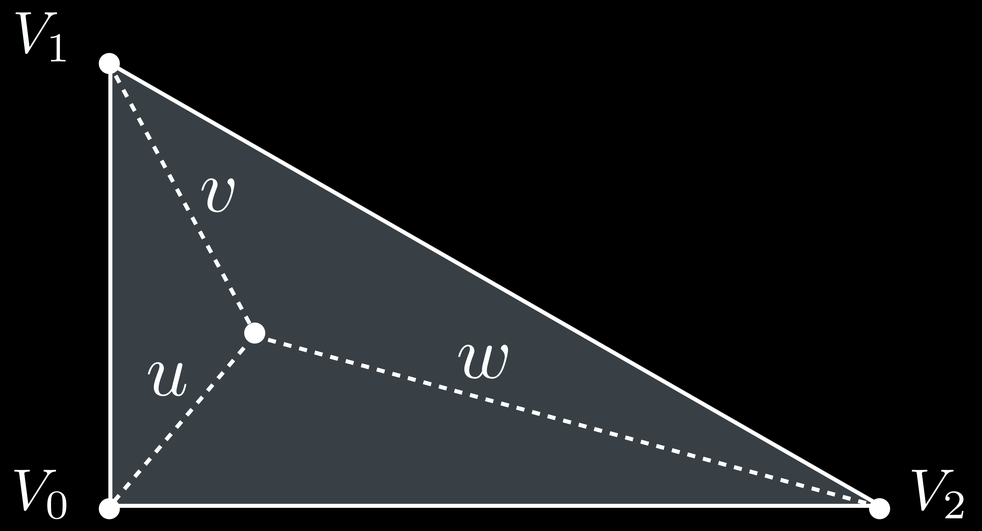


Barycentric coordinates u and v

Third coordinate: $w = 1 - u - v$

Shading

```
struct Intersection {
    float distance;
    int primitiveIndex;
    float2 coordinates;
};
```

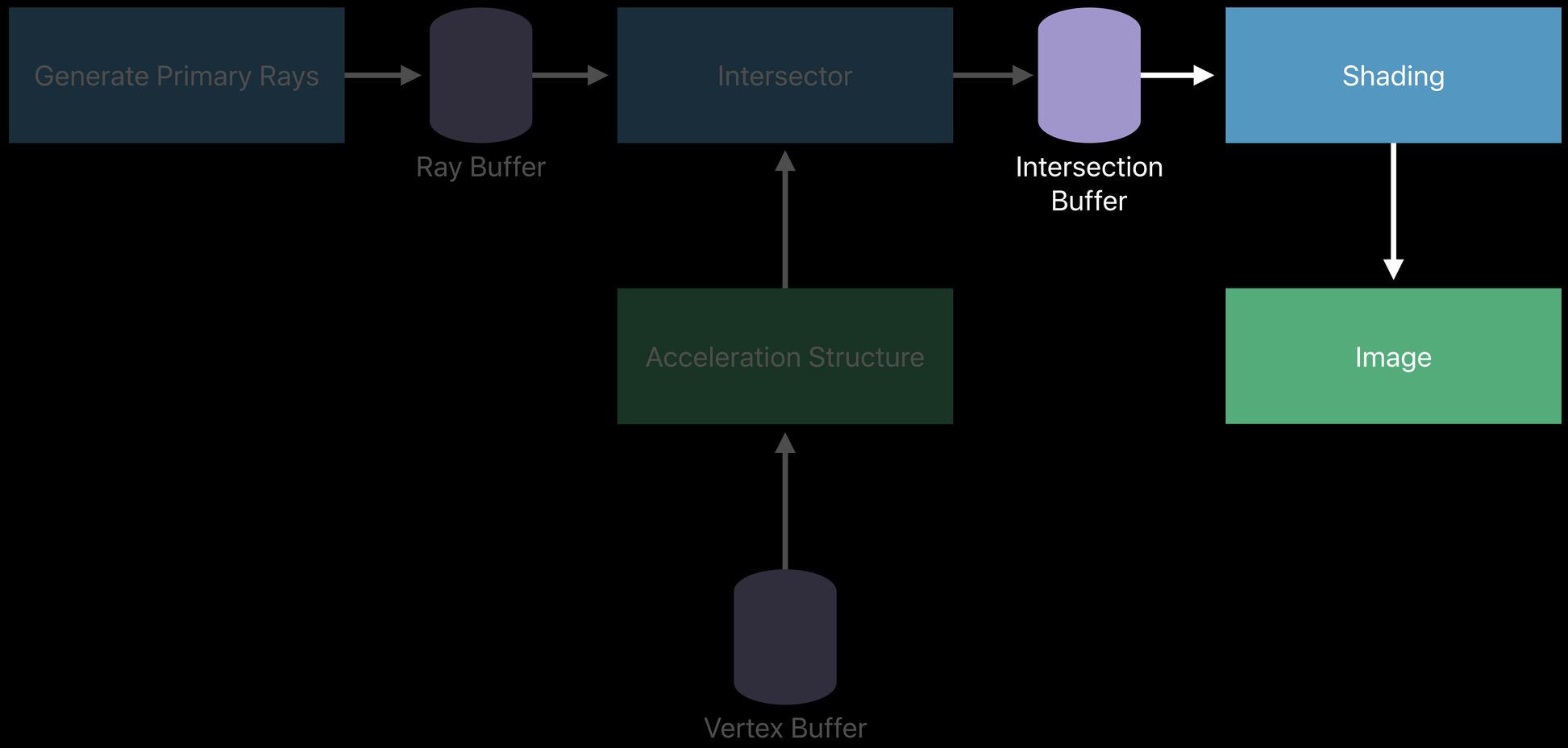


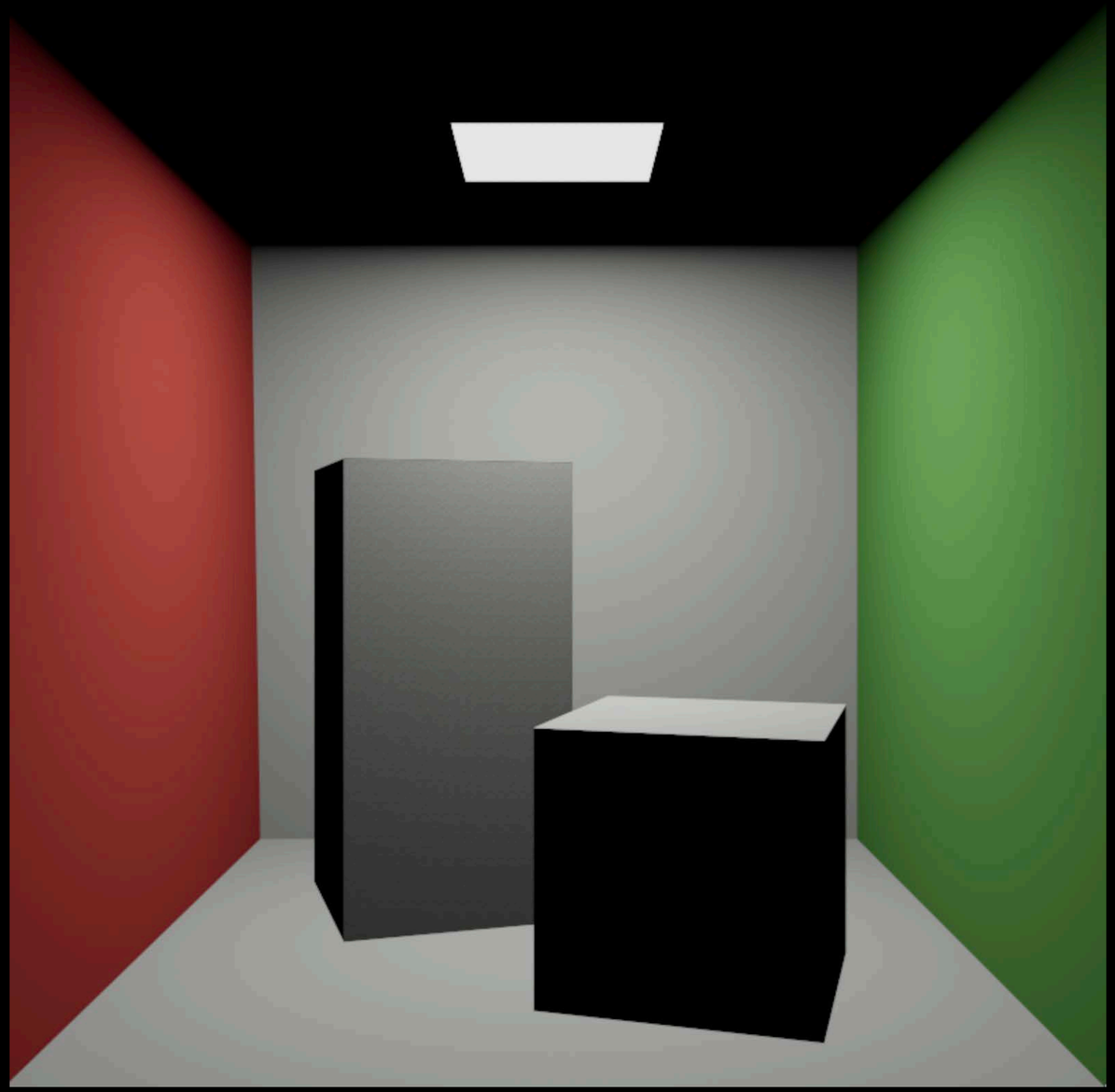
Barycentric coordinates u and v

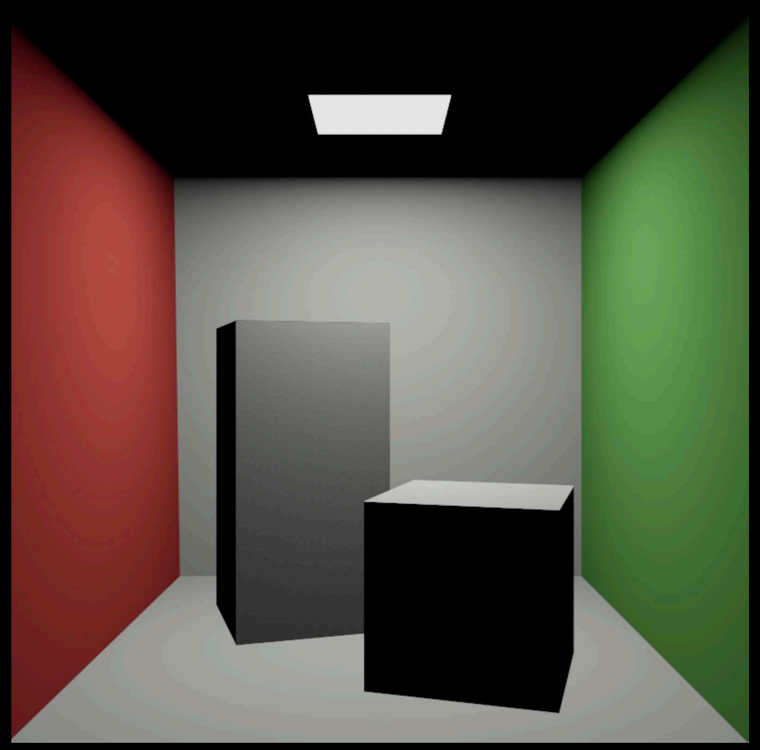
Third coordinate: $w = 1 - u - v$

Interpolated: $uV_0 + vV_1 + wV_2$

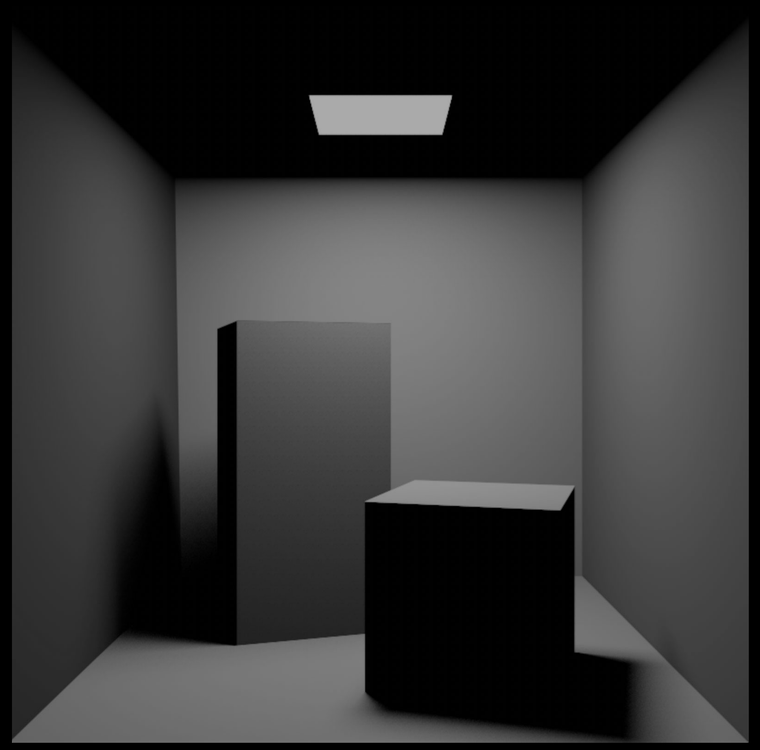
Primary Rays and Shading



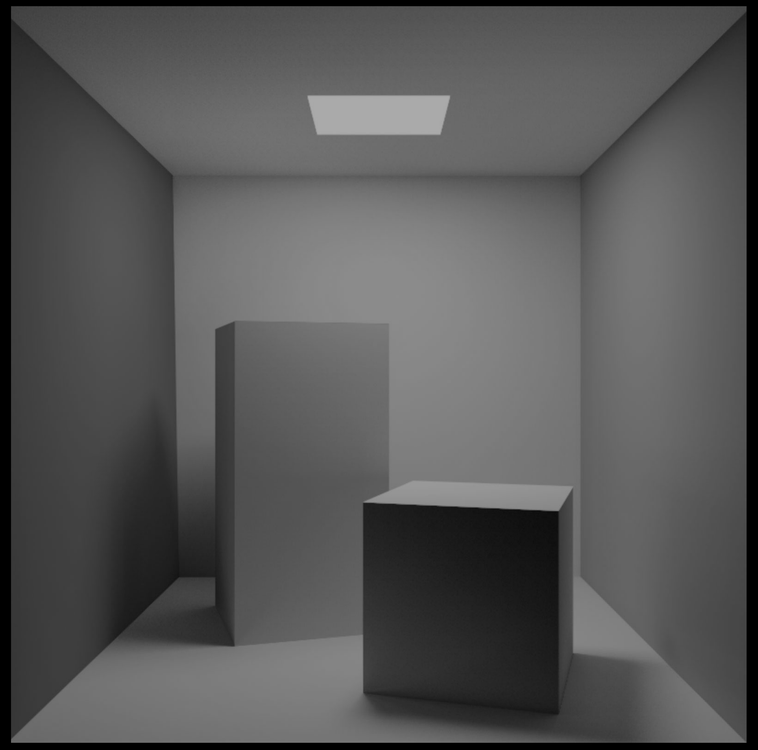




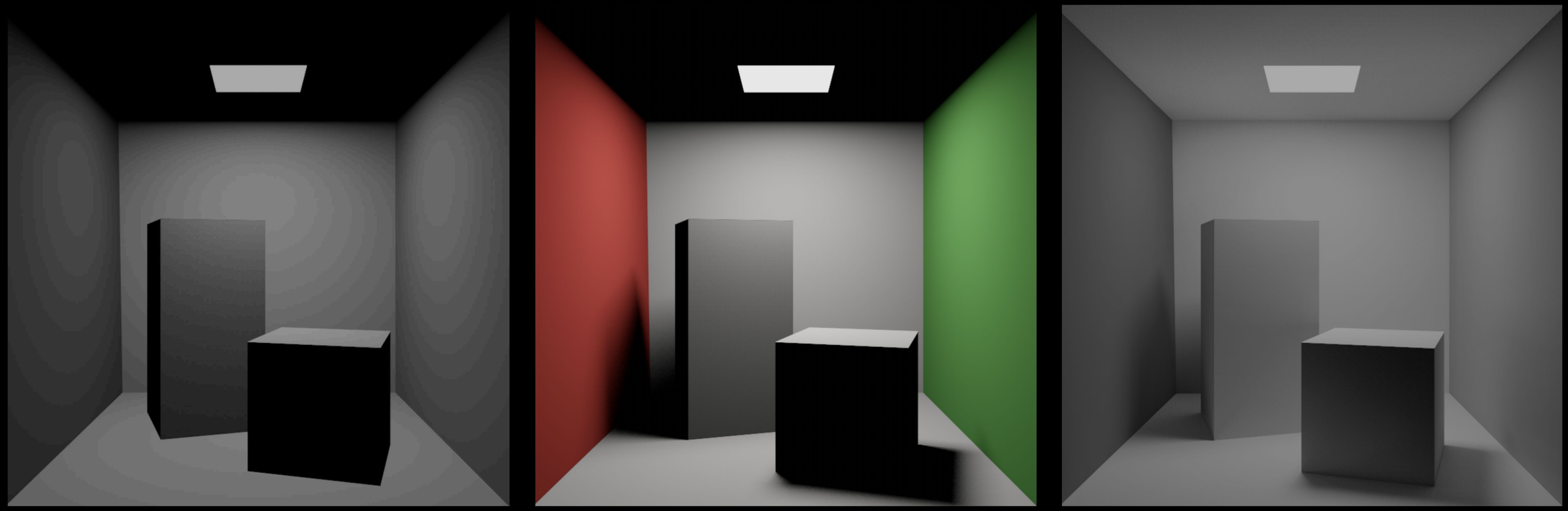
Primary Rays and Shading



Shadow Rays



Secondary Rays



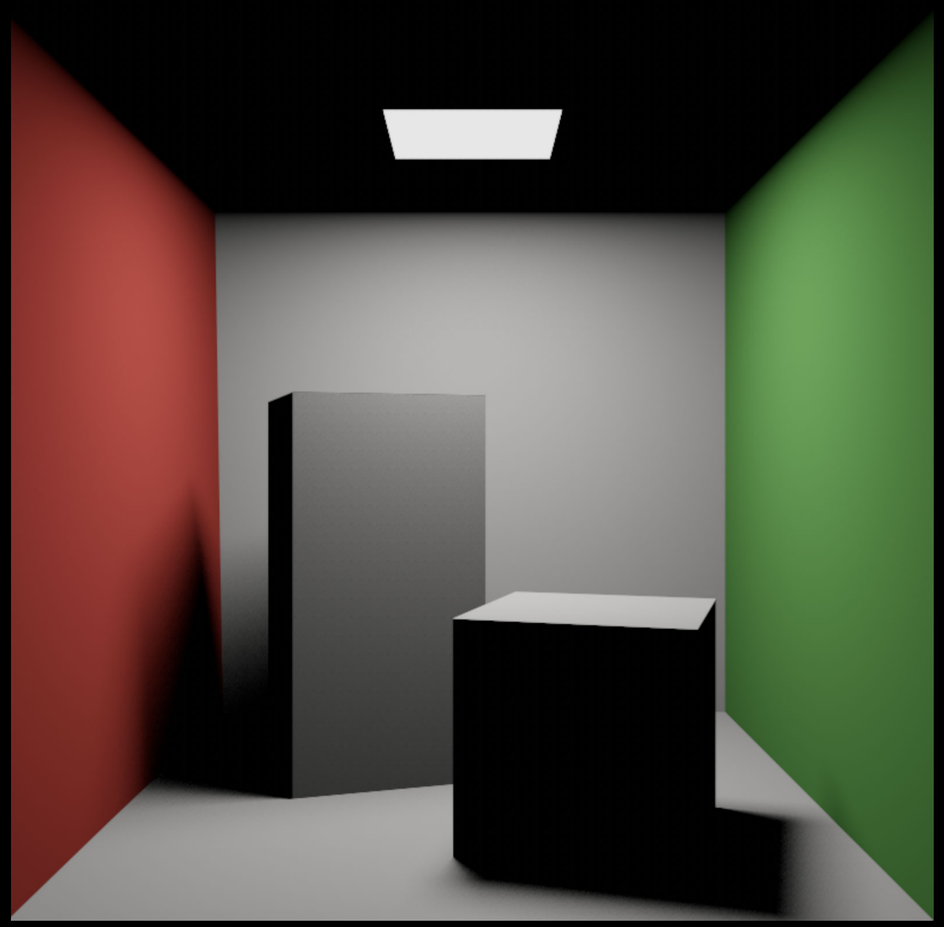
Primary Rays and Shading

Shadow Rays

Secondary Rays

Shadow Rays

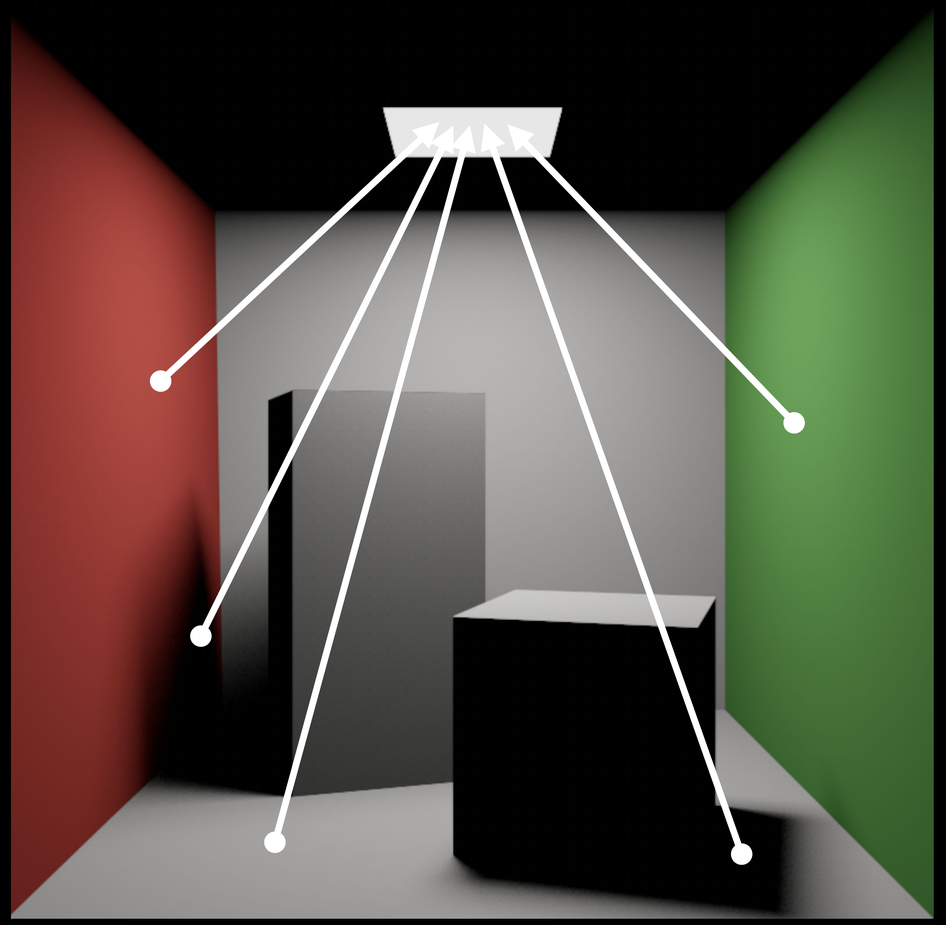
Check if shading point is in shadow before adding color to image



Shadow Rays

Check if shading point is in shadow before adding color to image

Cast a ray from shading point to light source

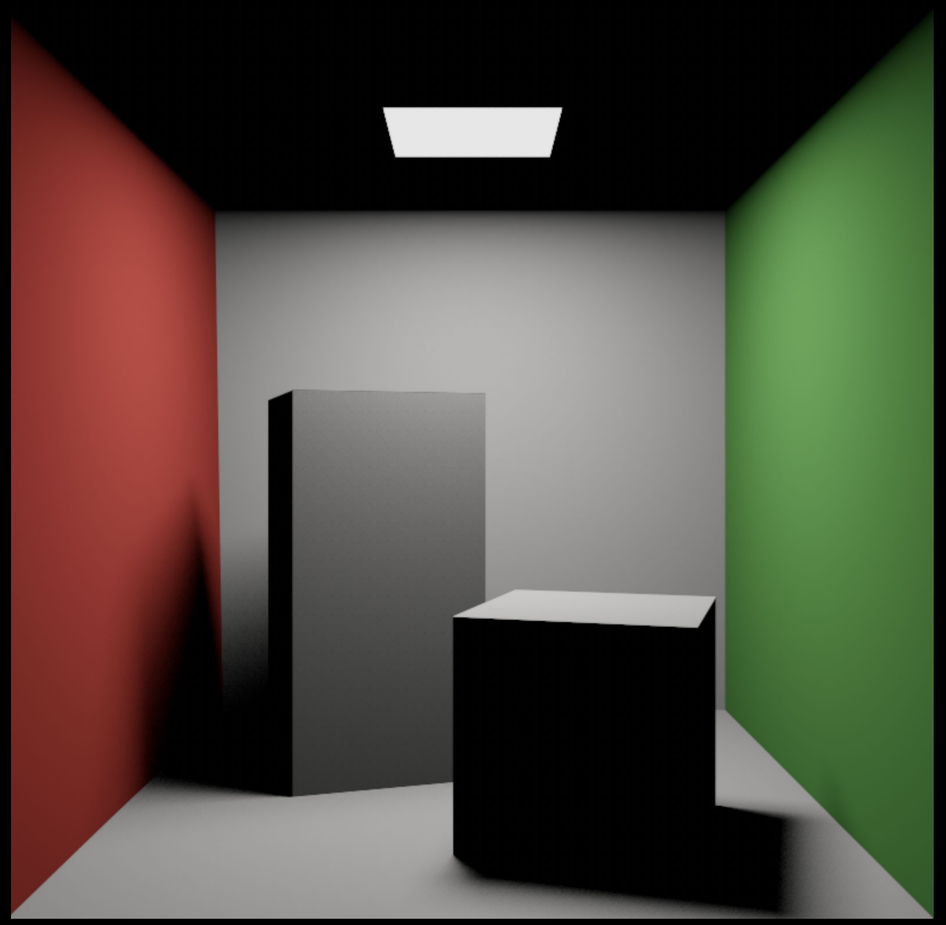


Shadow Rays

Check if shading point is in shadow before adding color to image

Cast a ray from shading point to light source

If shadow ray does not reach light, shading point is in shadow

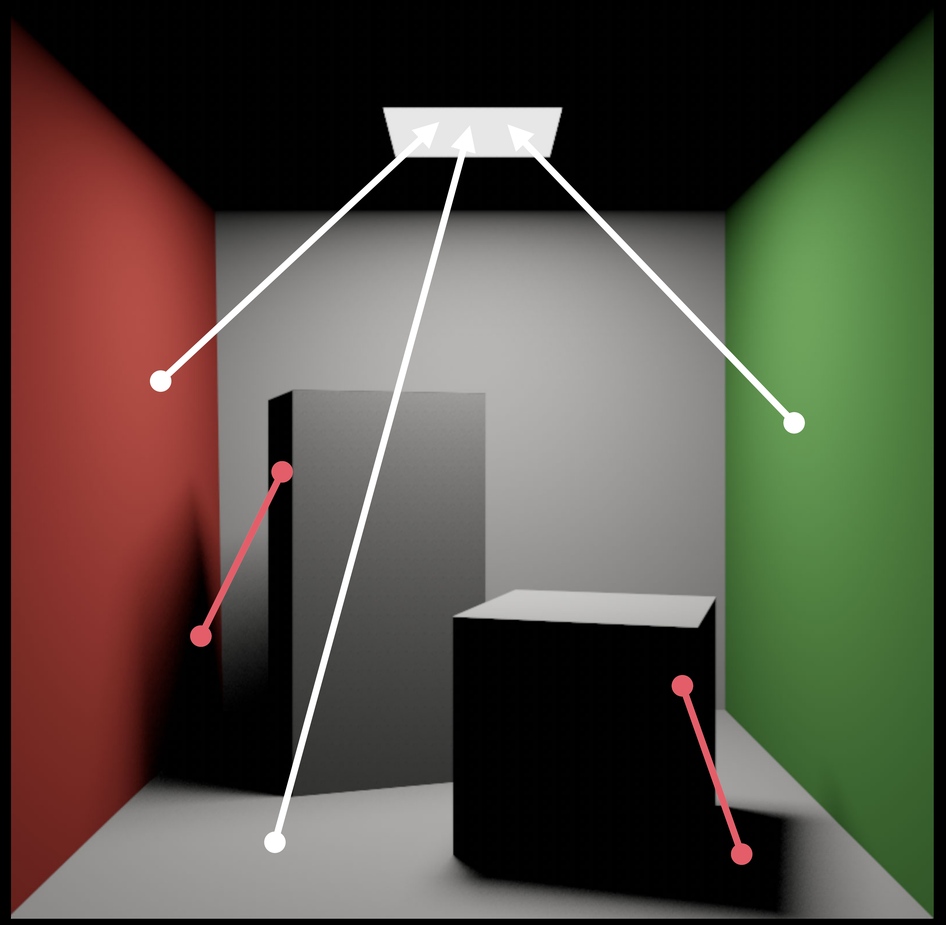


Shadow Rays

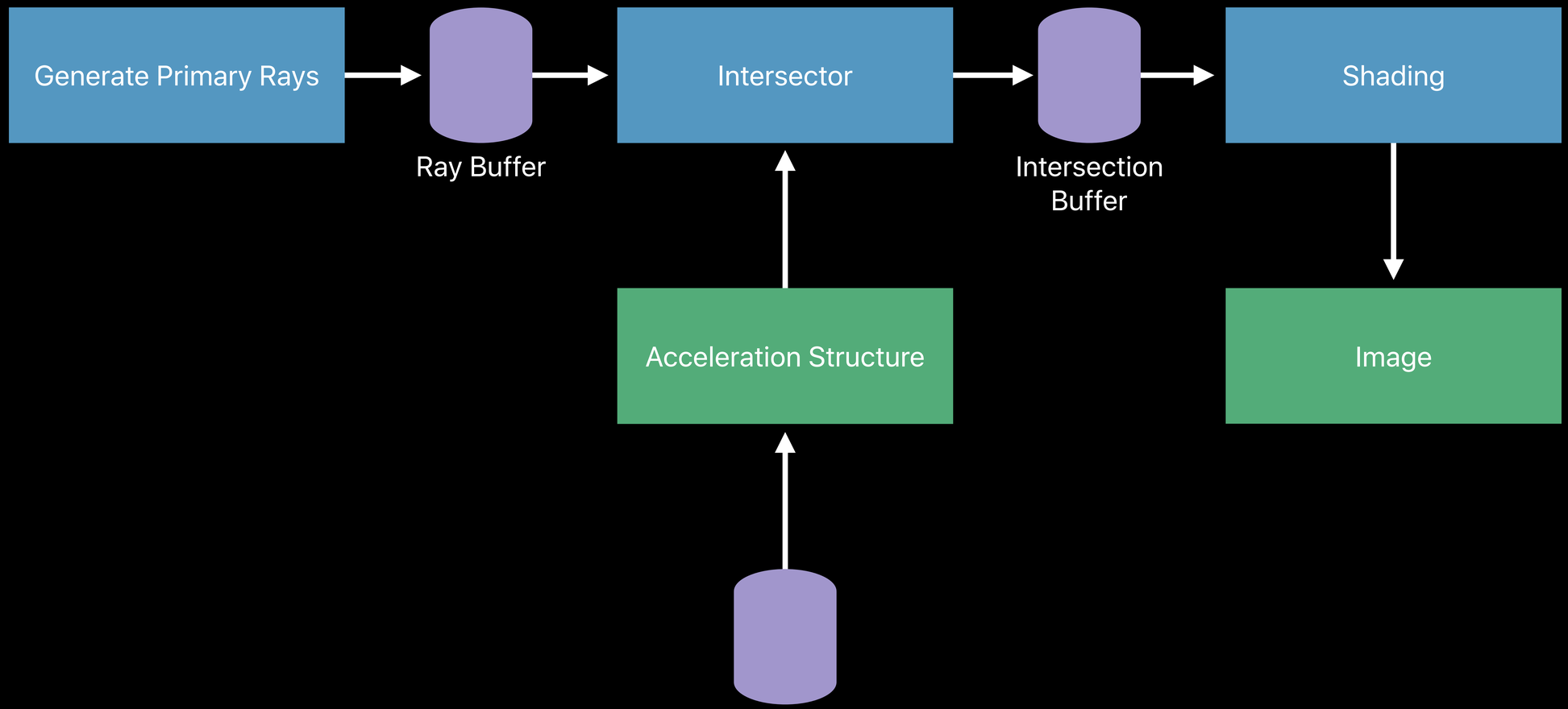
Check if shading point is in shadow before adding color to image

Cast a ray from shading point to light source

If shadow ray does not reach light, shading point is in shadow



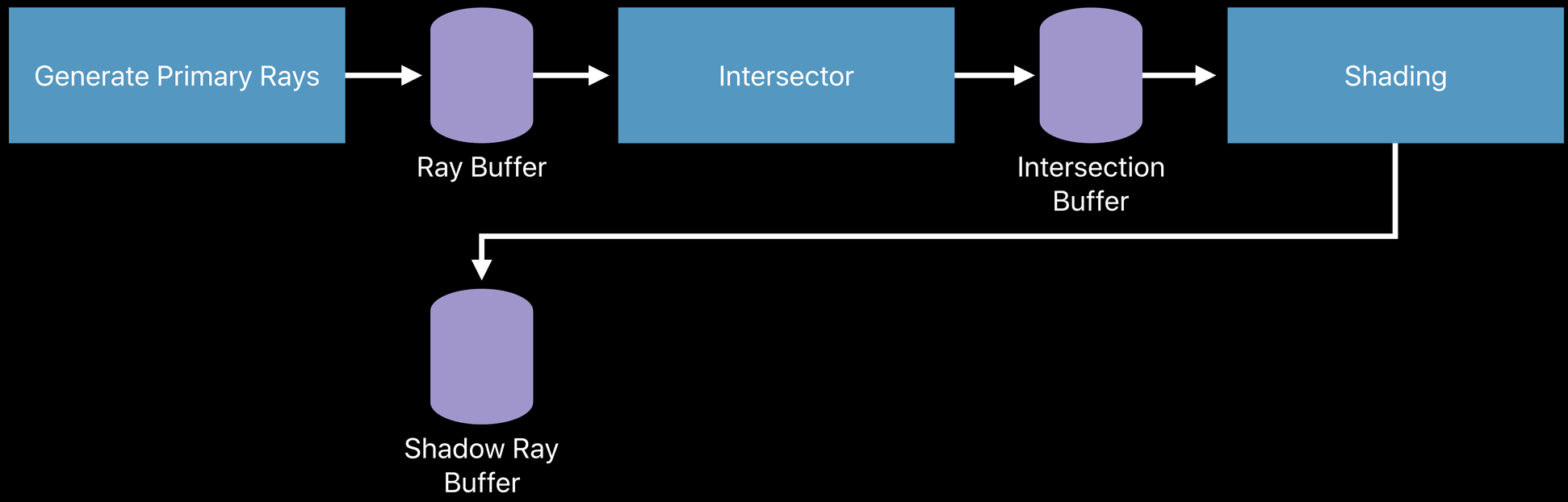
Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Acceleration Structure
Image

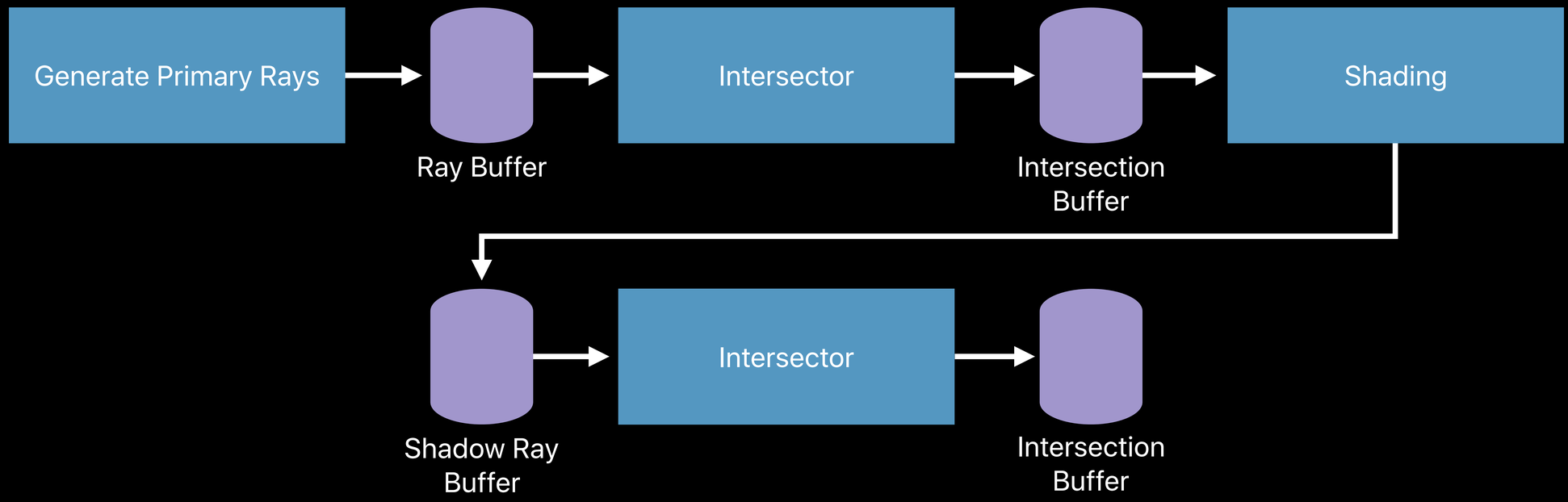
Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading

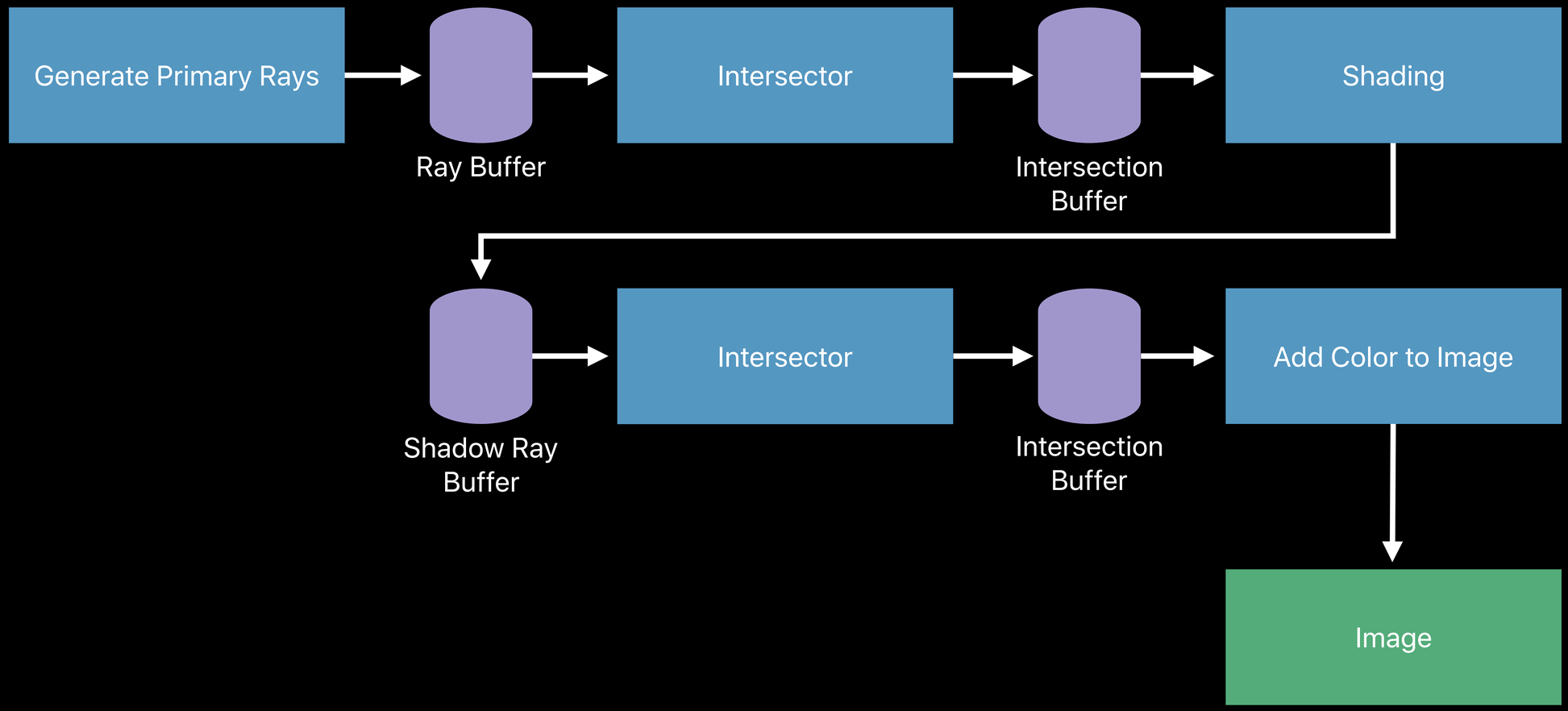
Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer

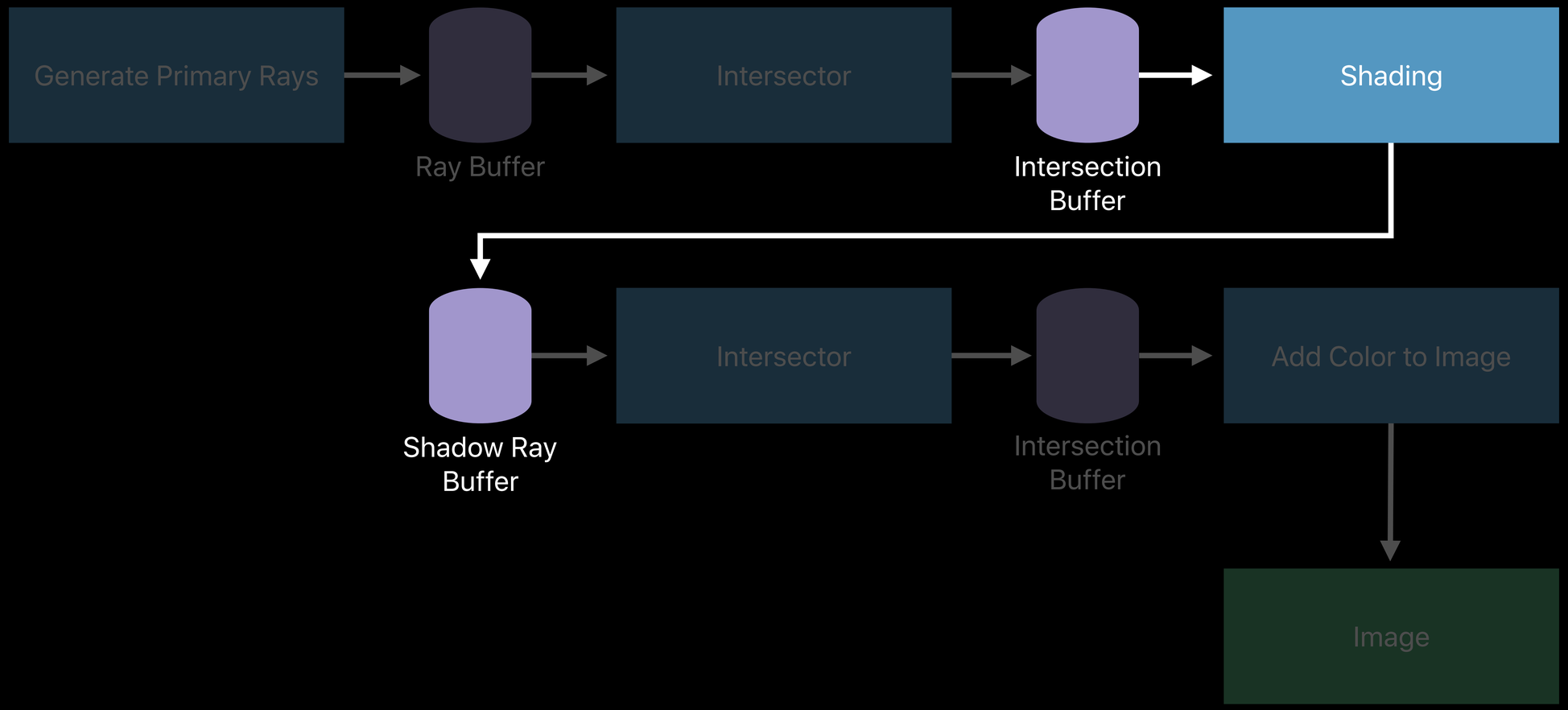
Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer
Intersector
Intersection Buffer

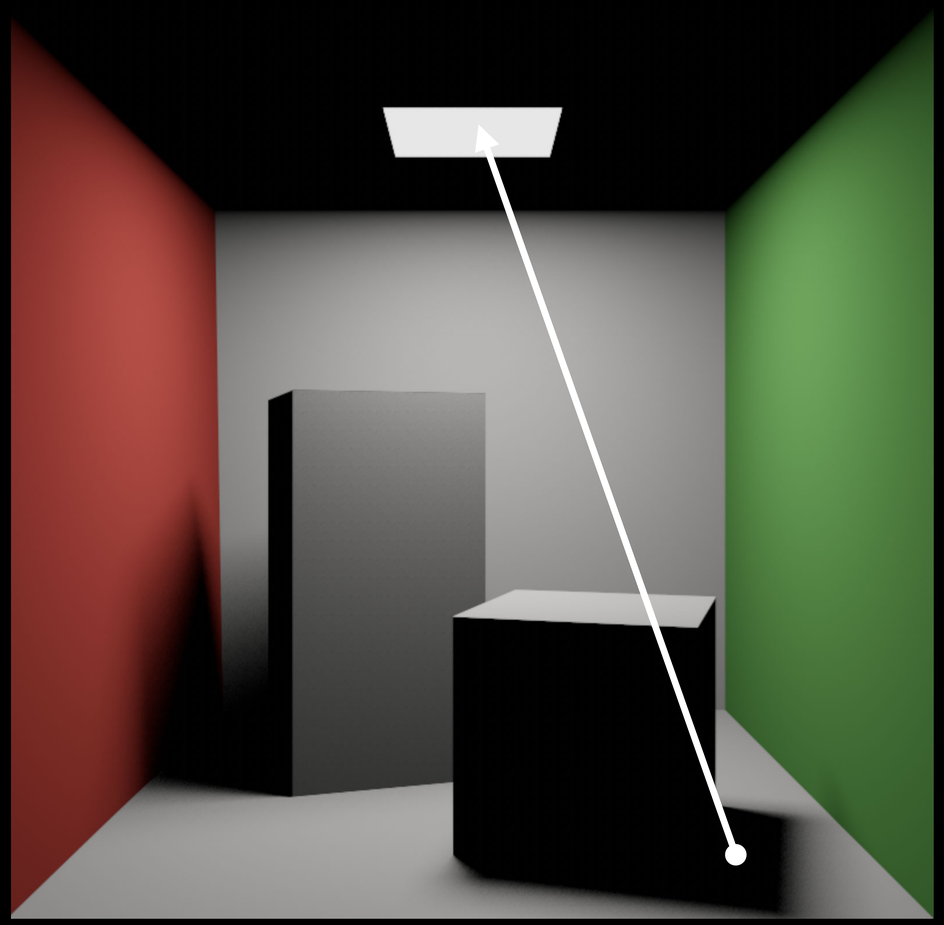
Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer
Intersector
Intersection Buffer
Add Color to Image
Image

Shadow Rays

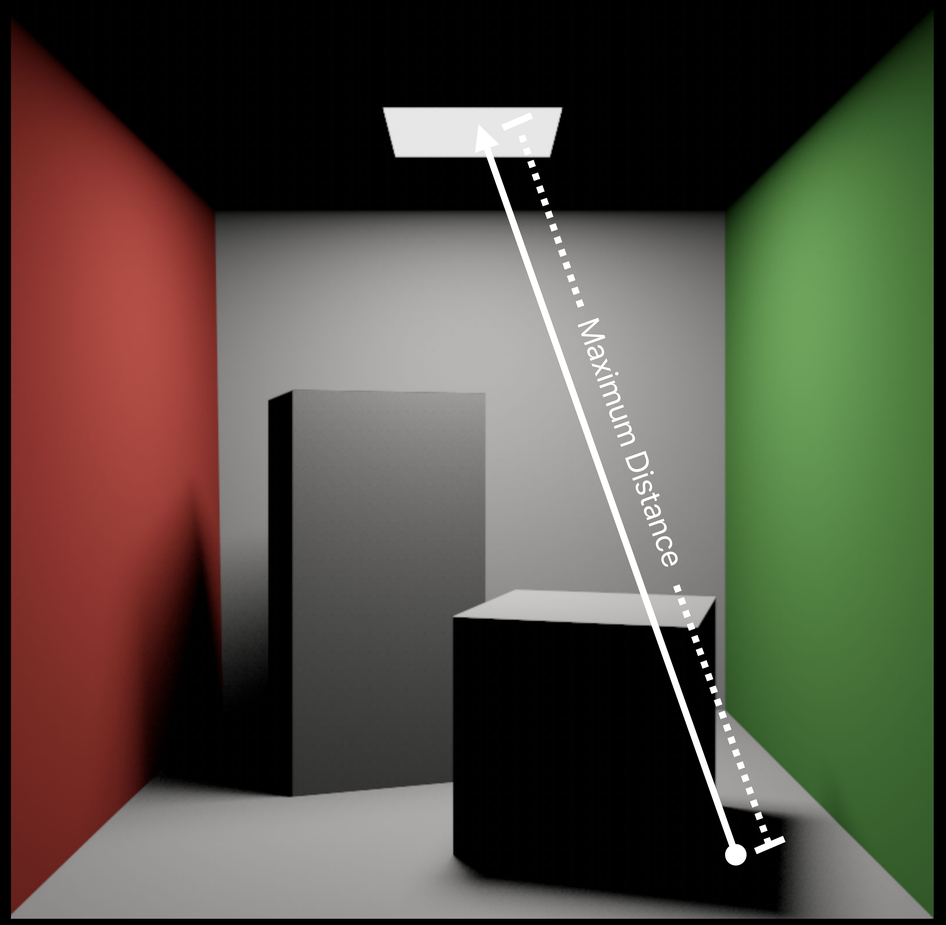

Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer
Intersector
Intersection Buffer
Add Color to Image
Image

Shadow Rays versus Primary Rays



Shadow Rays versus Primary Rays

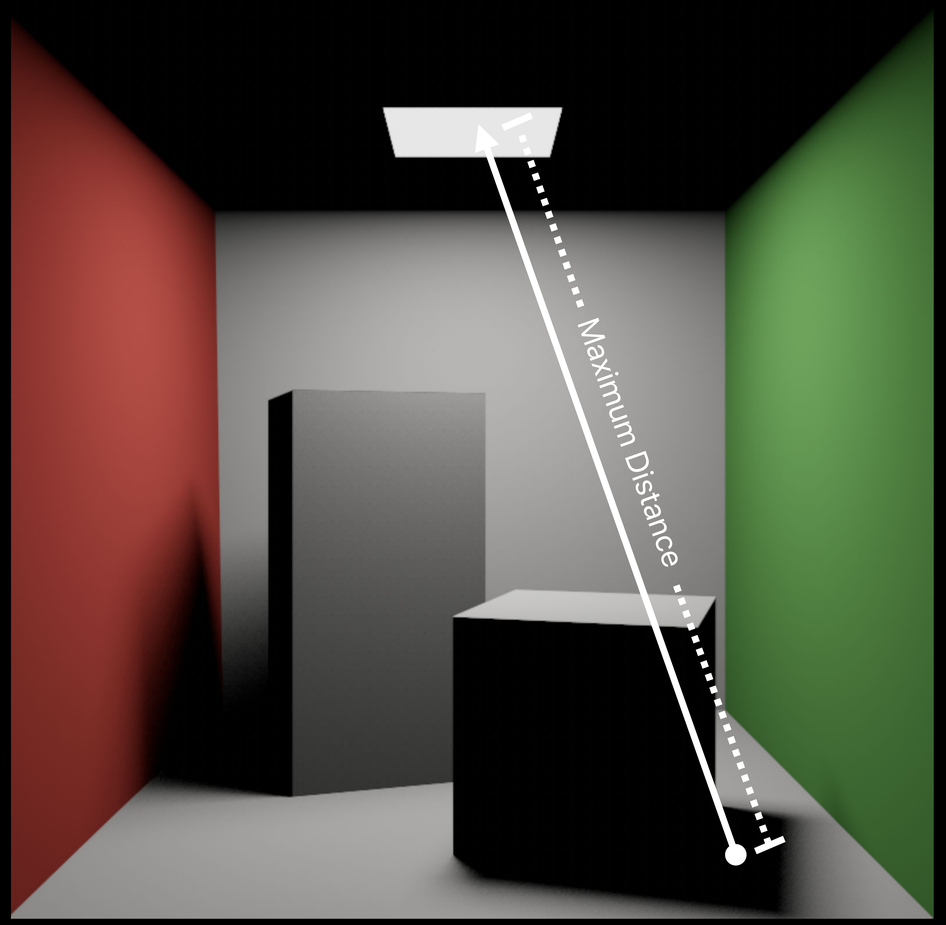
Maximum intersection distance



Shadow Rays versus Primary Rays

Maximum intersection distance

Don’t need triangle index or barycentric coordinates

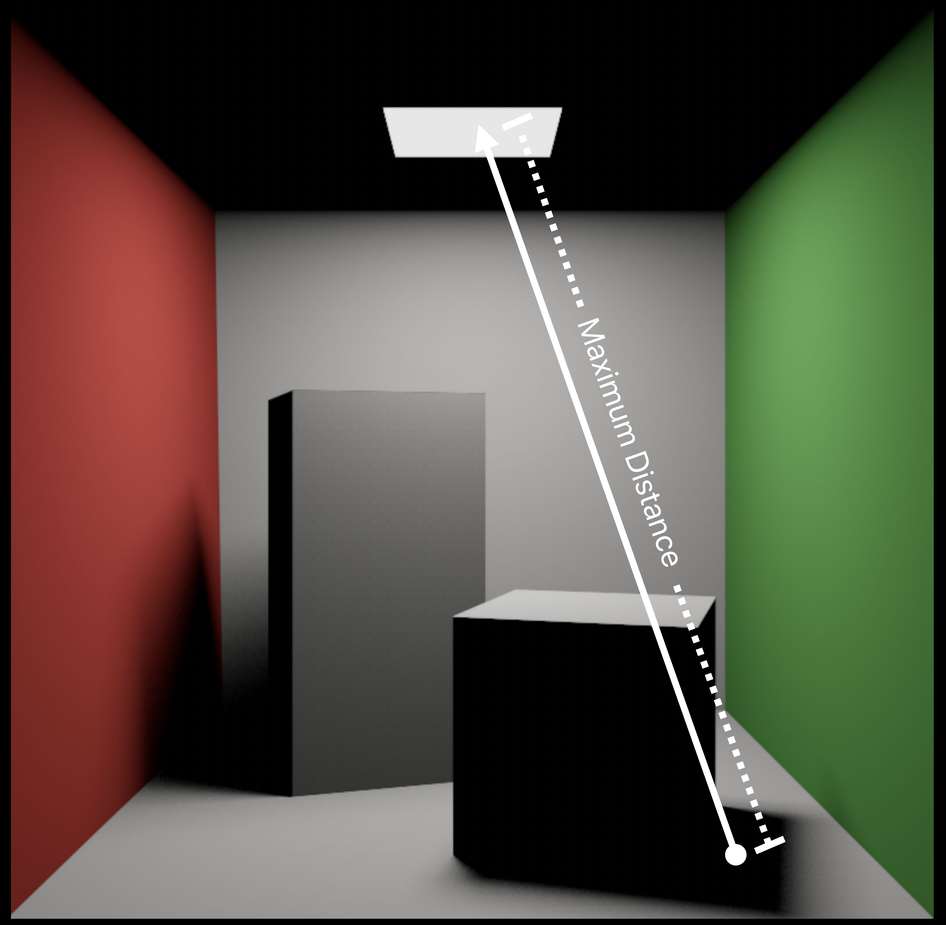


Shadow Rays versus Primary Rays

Maximum intersection distance

Don’t need triangle index or barycentric coordinates

Propagate color from shading kernel to final kernel



Customizing the Ray Struct

Ray type is configurable

```
struct Ray {
    packed_float3 origin;
    float minDistance;
    packed_float3 direction;
    float maxDistance;
    float3 color;
};
```

Customizing the Ray Struct

Ray type is configurable

Choose what data is provided to the intersector

```
struct Ray {
    packed_float3 origin;
    float minDistance;
    packed_float3 direction;
    float maxDistance;
    float3 color;
};
```

Customizing the Ray Struct

Ray type is configurable

Choose what data is provided to the intersector

Append app-specific data

```
struct Ray {
    packed_float3 origin;
    float minDistance;
    packed_float3 direction;
    float maxDistance;
    float3 color;
};
```

Customizing the Ray Struct

Ray type is configurable

Choose what data is provided to the intersector

Append app-specific data

```
struct Ray {
    packed_float3 origin;
    float minDistance;
    packed_float3 direction;
    float maxDistance;
    float3 color;
};
```

Configure MPSRayIntersector for new Ray struct type:

```
intersector.rayDataType = .originMinDistanceDirectionMaxDistance
intersector.rayStride = MemoryLayout<Ray>.stride
```

Customizing the Ray Struct

Ray type is configurable

Choose what data is provided to the intersector

Append app-specific data

```
struct Ray {
    packed_float3 origin;
    float minDistance;
    packed_float3 direction;
    float maxDistance;
    float3 color;
};
```

Configure MPSRayIntersector for new Ray struct type:

```
intersector.rayDataType = .originMinDistanceDirectionMaxDistance
intersector.rayStride = MemoryLayout<Ray>.stride
```

Customizing the Ray Struct

Ray type is configurable

Choose what data is provided to the intersector

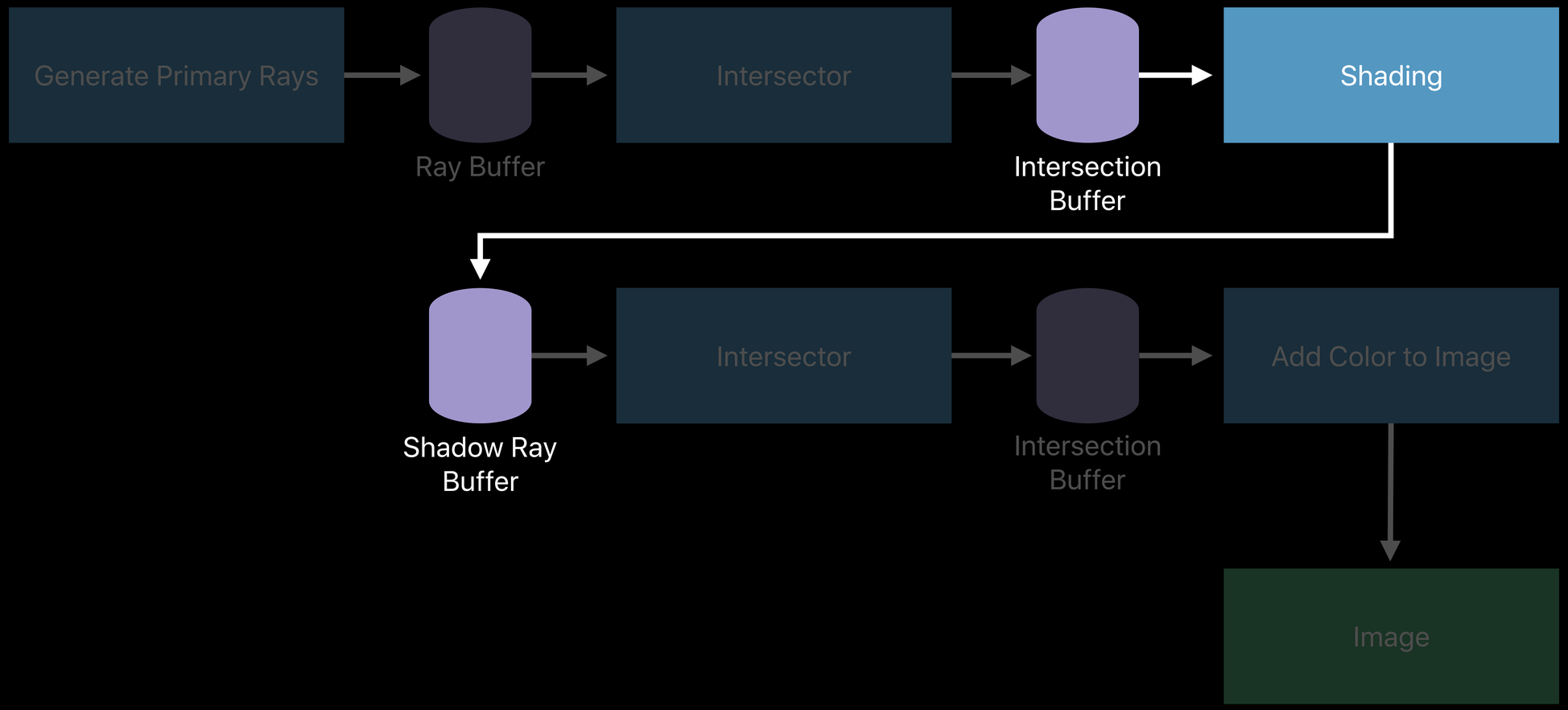
Append app-specific data

```
struct Ray {
    packed_float3 origin;
    float minDistance;
    packed_float3 direction;
    float maxDistance;
    float3 color;
};
```

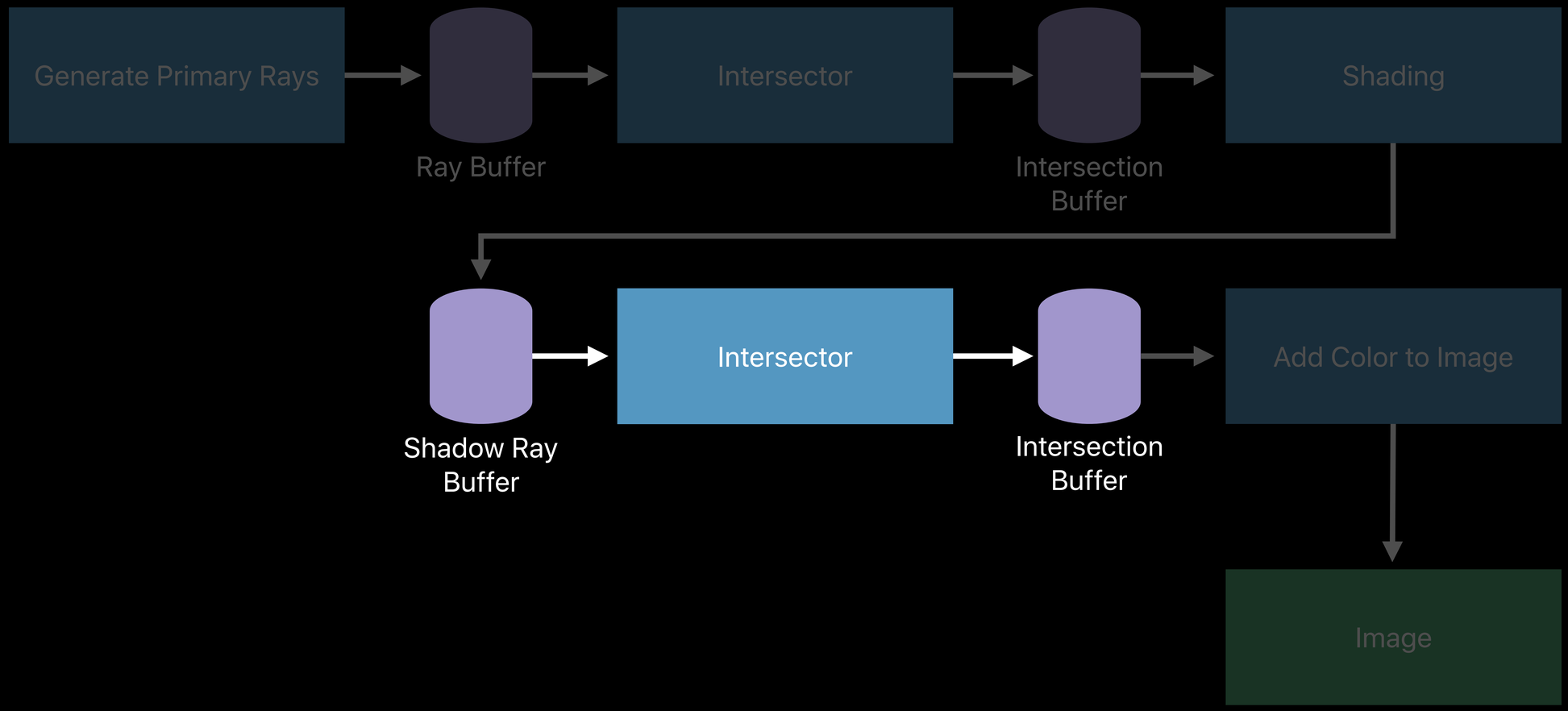
Configure MPSRayIntersector for new Ray struct type:

```
intersector.rayDataType = .originMinDistanceDirectionMaxDistance
intersector.rayStride = MemoryLayout<Ray>.stride
```

Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer
Intersector
Intersection Buffer
Add Color to Image
Image

Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer
Intersector
Intersection Buffer
Add Color to Image
Image

Finding Shadow Ray Intersections

Modify call to MPSRayIntersector:

```
intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Shadow Ray Intersections

Modify call to MPSRayIntersector:

```
intersector.intersectionDataType = .distance

intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Shadow Ray Intersections

Modify call to MPSRayIntersector:

```
intersector.intersectionDataType = .distance

intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Shadow Ray Intersections

Modify call to MPSRayIntersector:

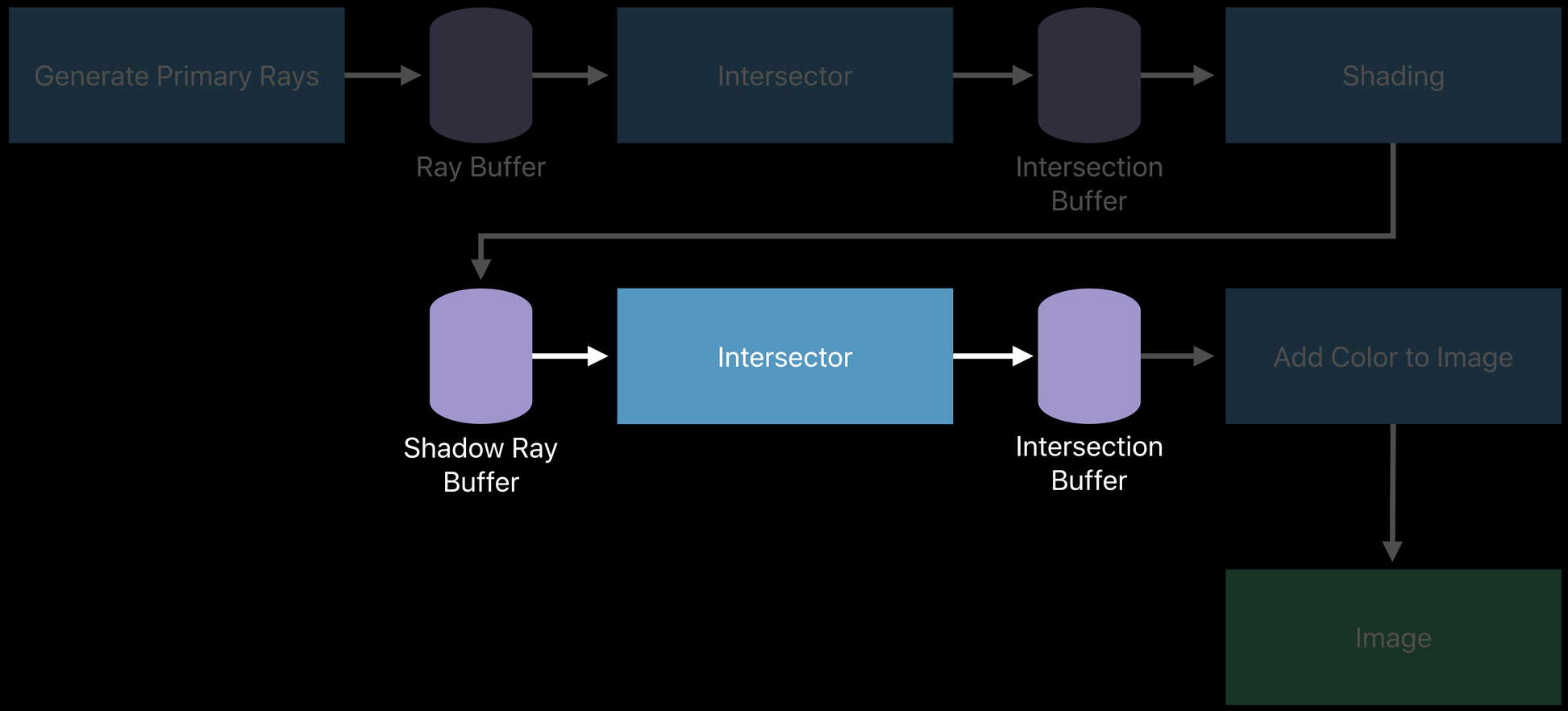
```
intersector.intersectionDataType = .distance

intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .nearest,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Finding Shadow Ray Intersections

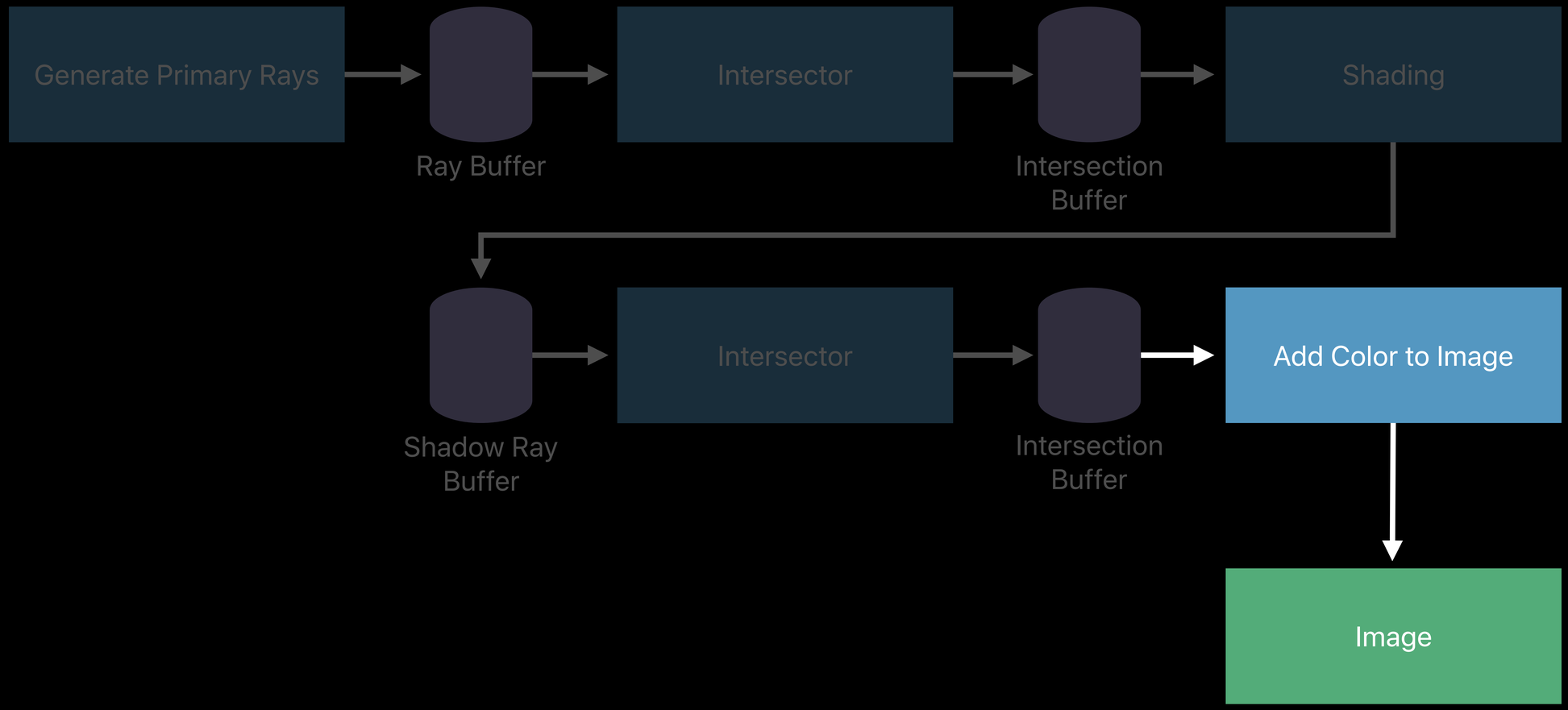
Modify call to MPSRayIntersector:

```
intersector.intersectionDataType = .distance

intersector.encodeIntersection(commandBuffer: commandBuffer,
                               intersectionType: .any,
                               rayBuffer: rayBuffer,
                               rayBufferOffset: 0,
                               intersectionBuffer: intersectionBuffer,
                               intersectionBufferOffset: 0,
                               rayCount: rayCount,
                               accelerationStructure: accelerationStructure)
```

Shadow Rays


Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer
Intersector
Intersection Buffer
Add Color to Image
Image

Shadow Rays

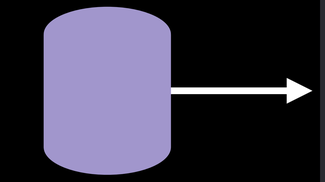

Generate Primary Rays
Ray Buffer
Intersector
Intersection Buffer
Shading
Shadow Ray Buffer
Intersector
Intersection Buffer
Add Color to Image
Image

Adding Ray Color to the Image

Shadow Ray Buffer

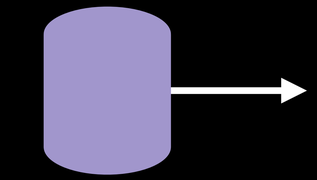
Intersection Buffer

Adding Ray Color to the Image



Shadow Ray Buffer

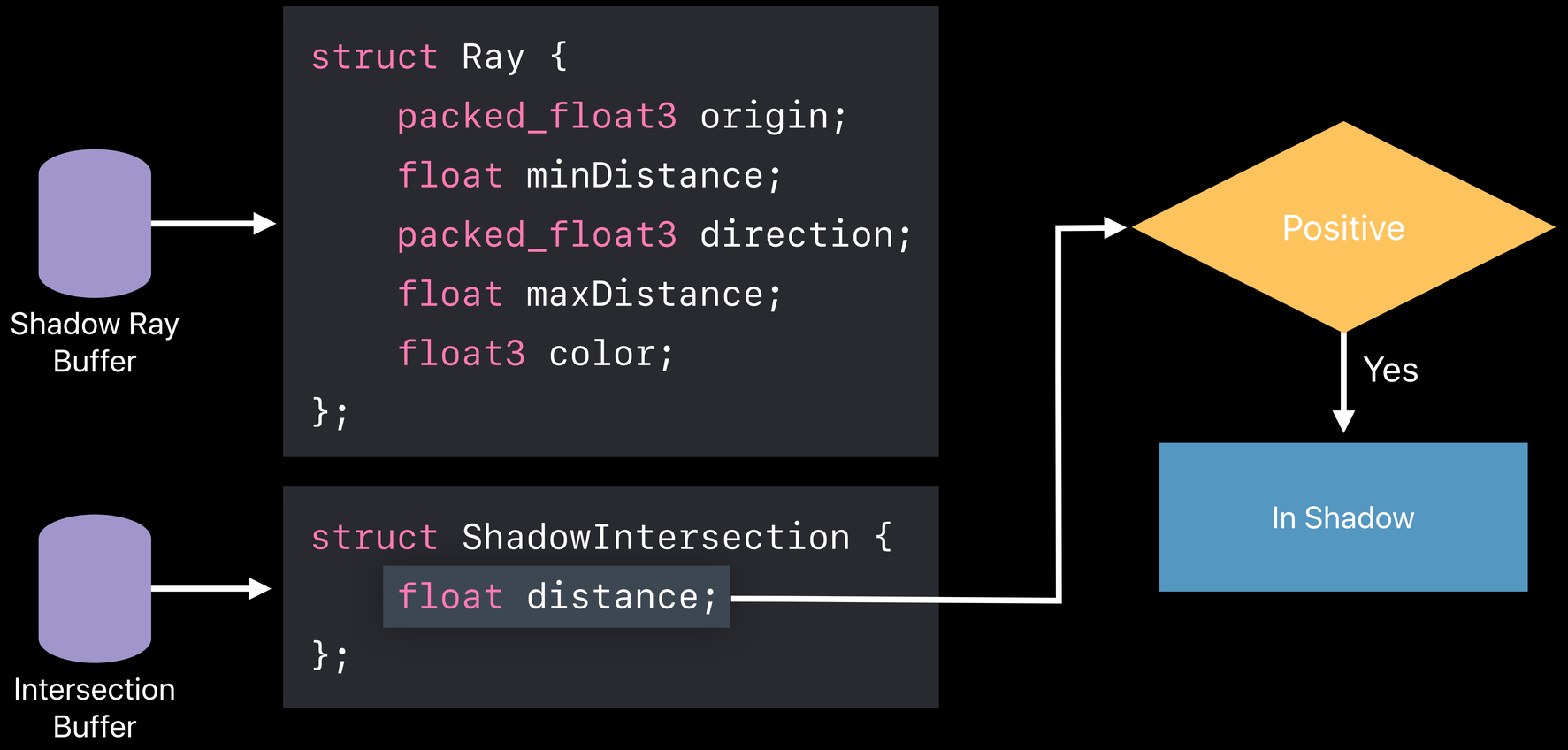
```
struct Ray {
    packed_float3 origin;
    float minDistance;
    packed_float3 direction;
    float maxDistance;
    float3 color;
};
```



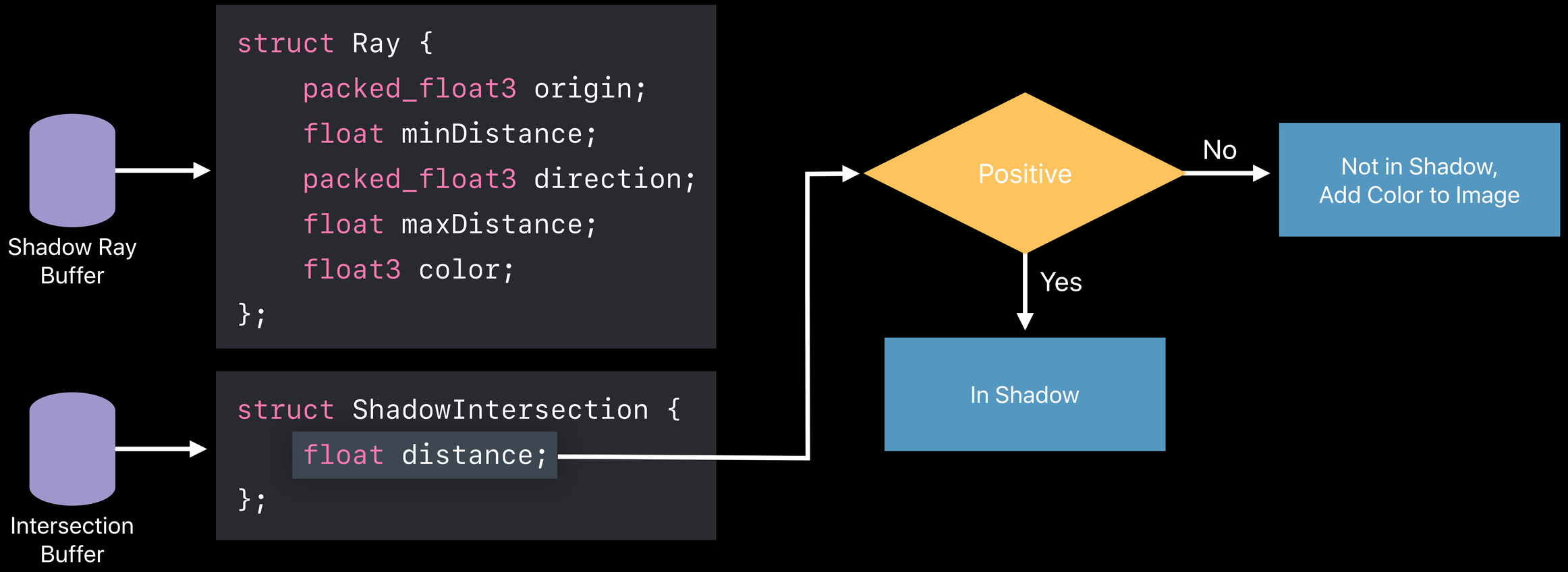
Intersection Buffer

```
struct ShadowIntersection {
    float distance;
};
```

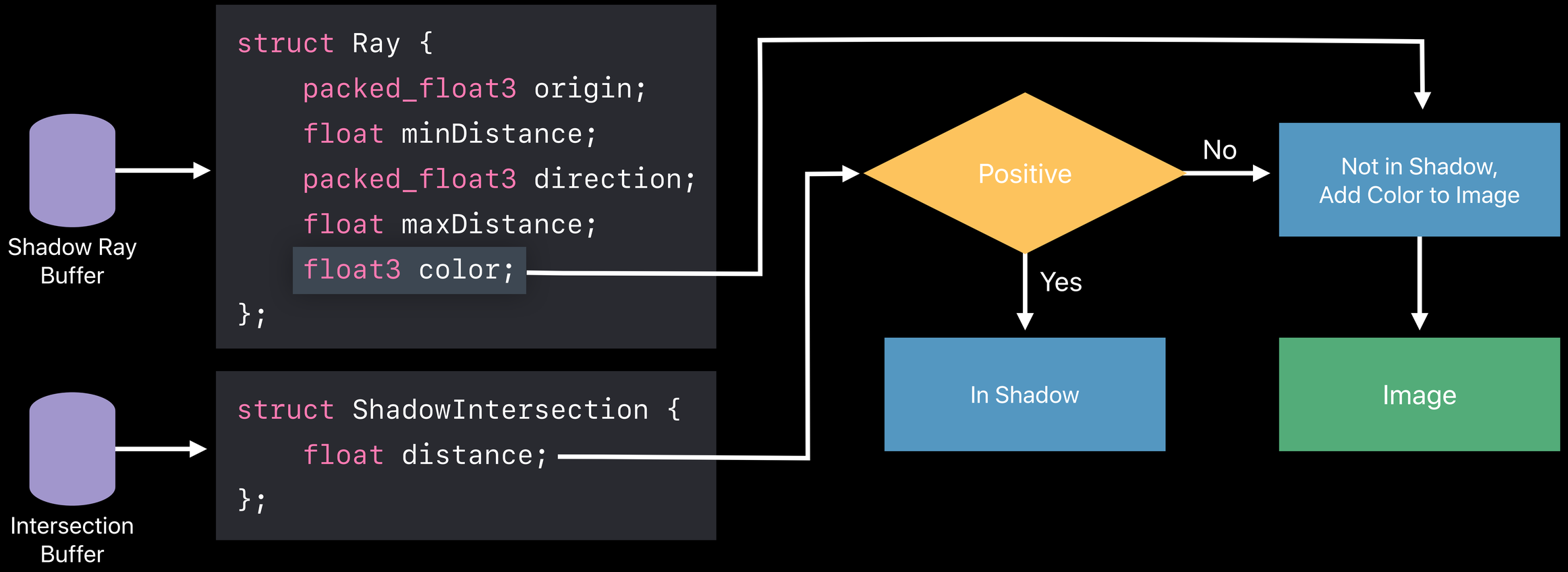
Adding Ray Color to the Image

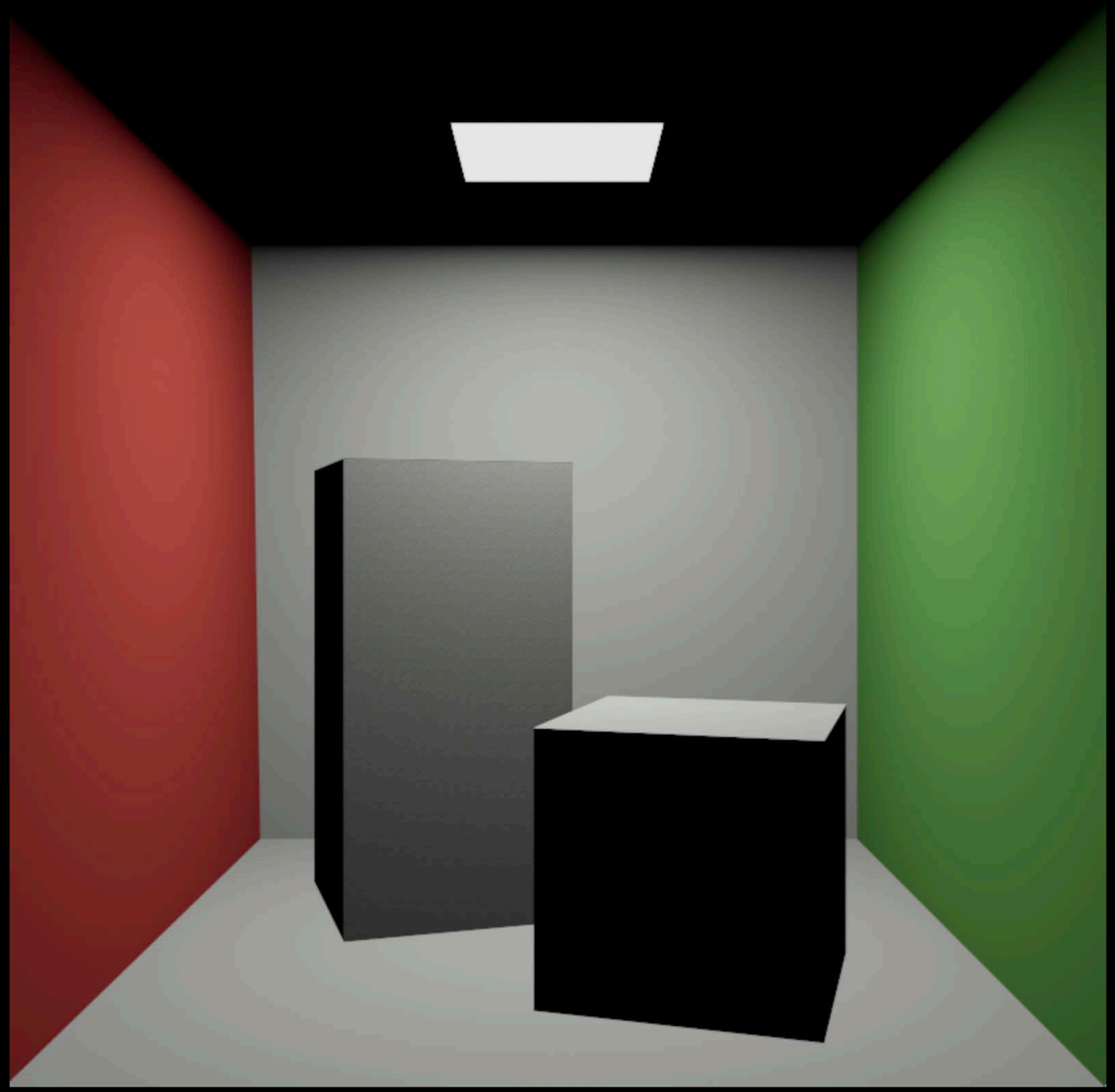


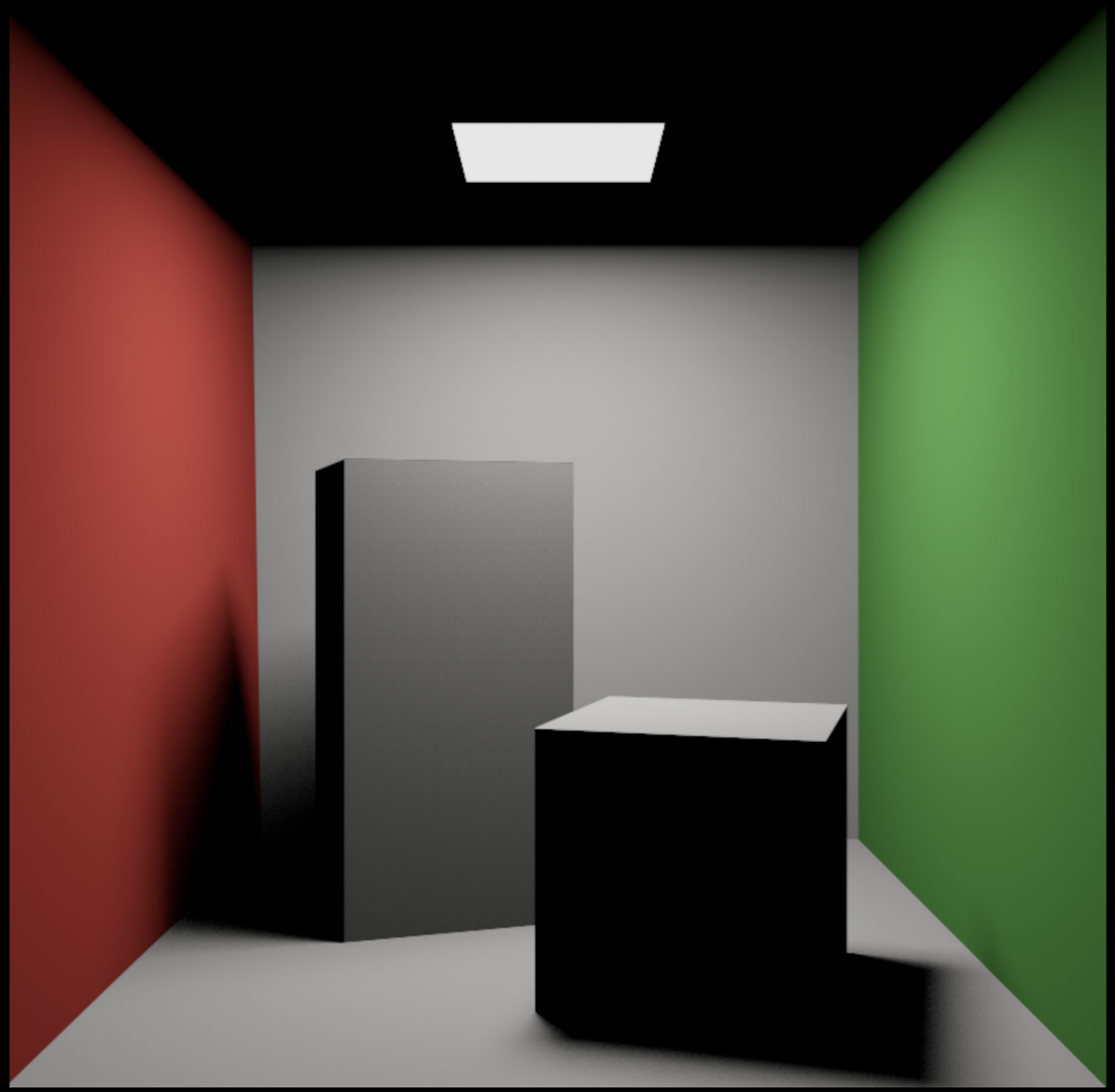
Adding Ray Color to the Image

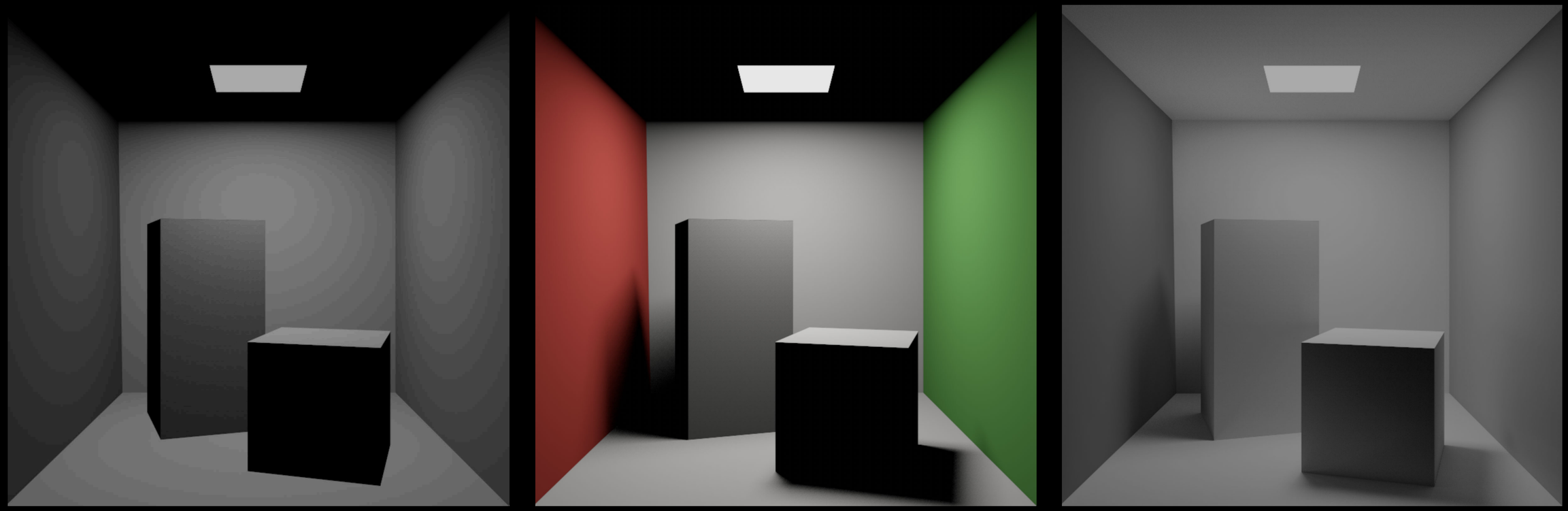


Adding Ray Color to the Image





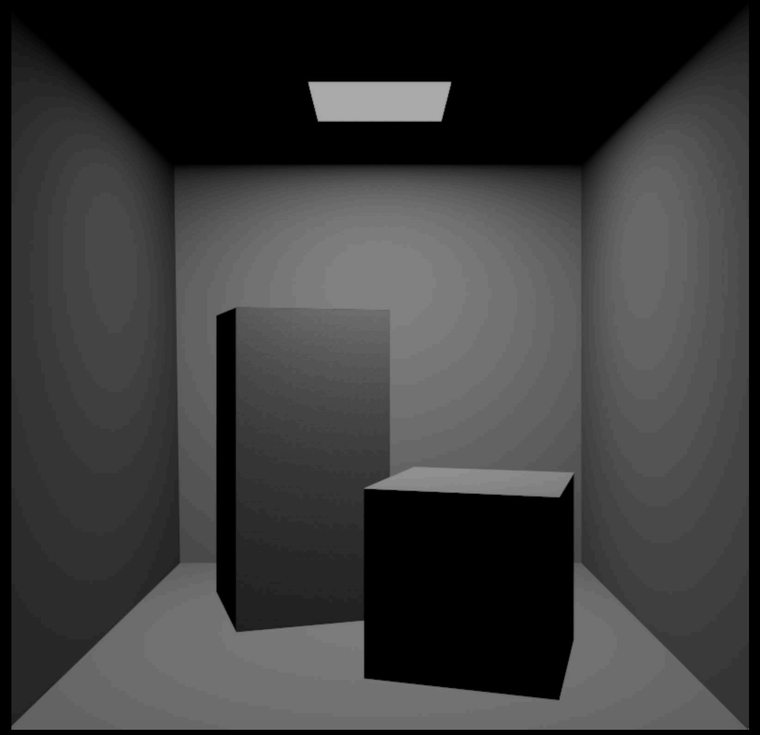




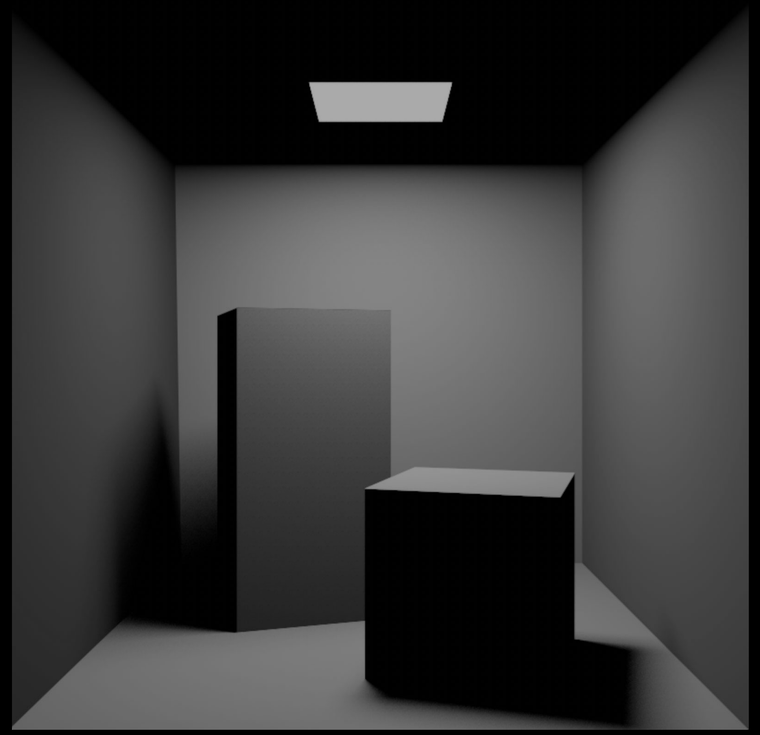
Primary Rays and Shading

Shadow Rays

Secondary Rays



Primary Rays and Shading



Shadow Rays

Secondary Rays

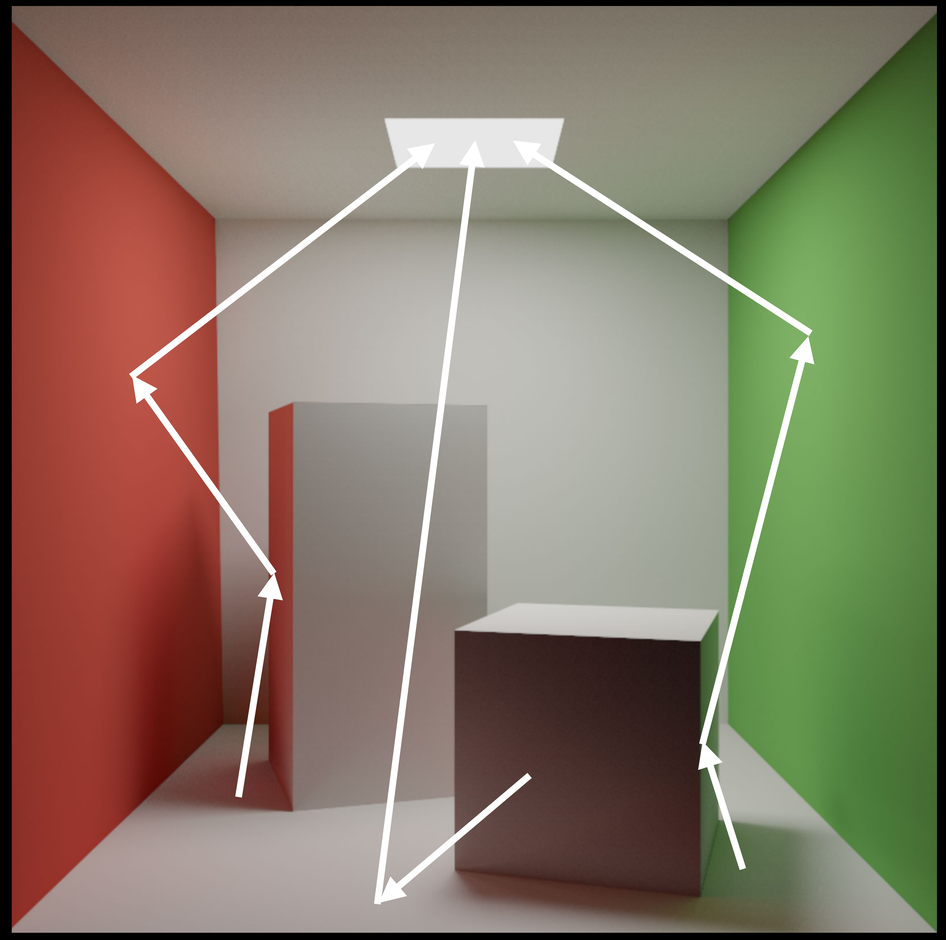
Secondary Rays

Simulate light bouncing around the scene

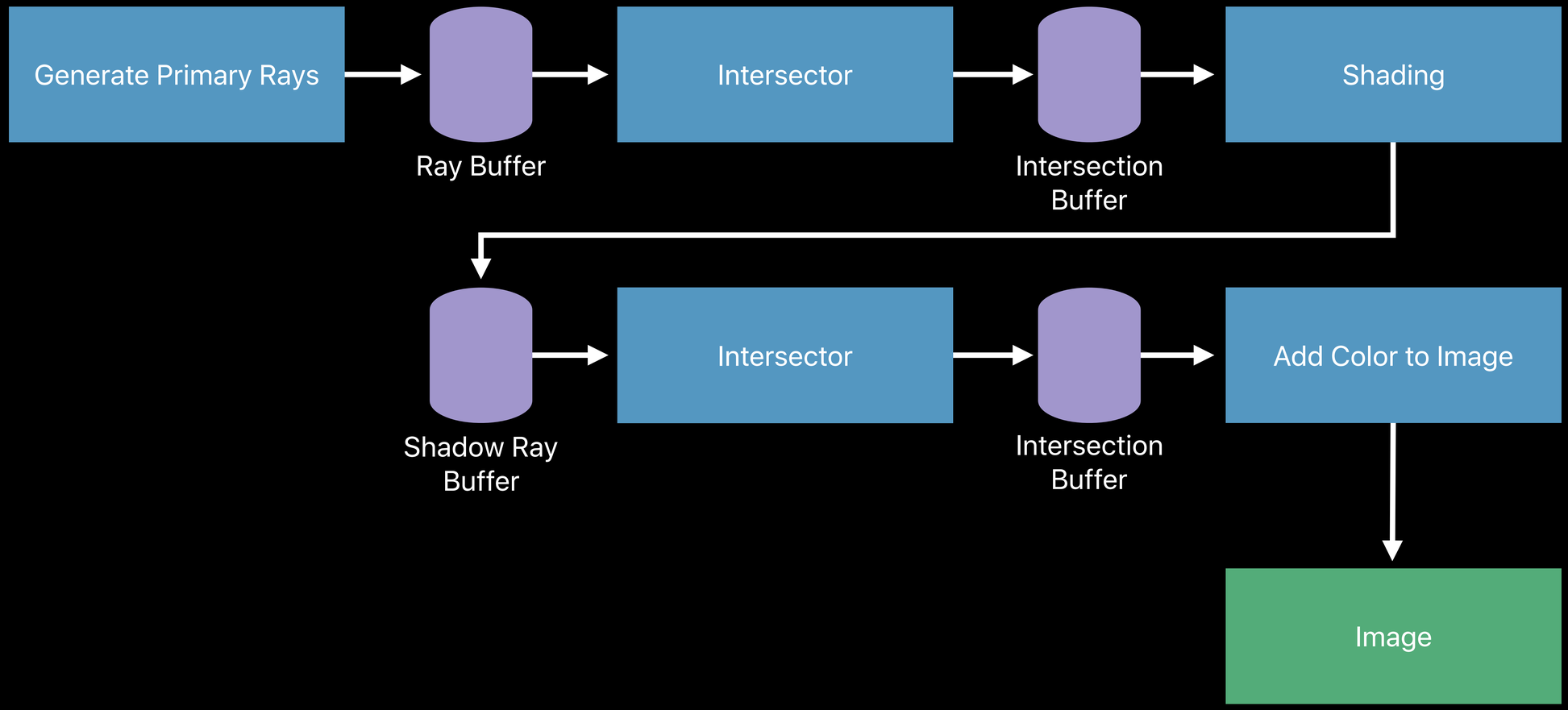
Secondary Rays

Simulate light bouncing around the scene

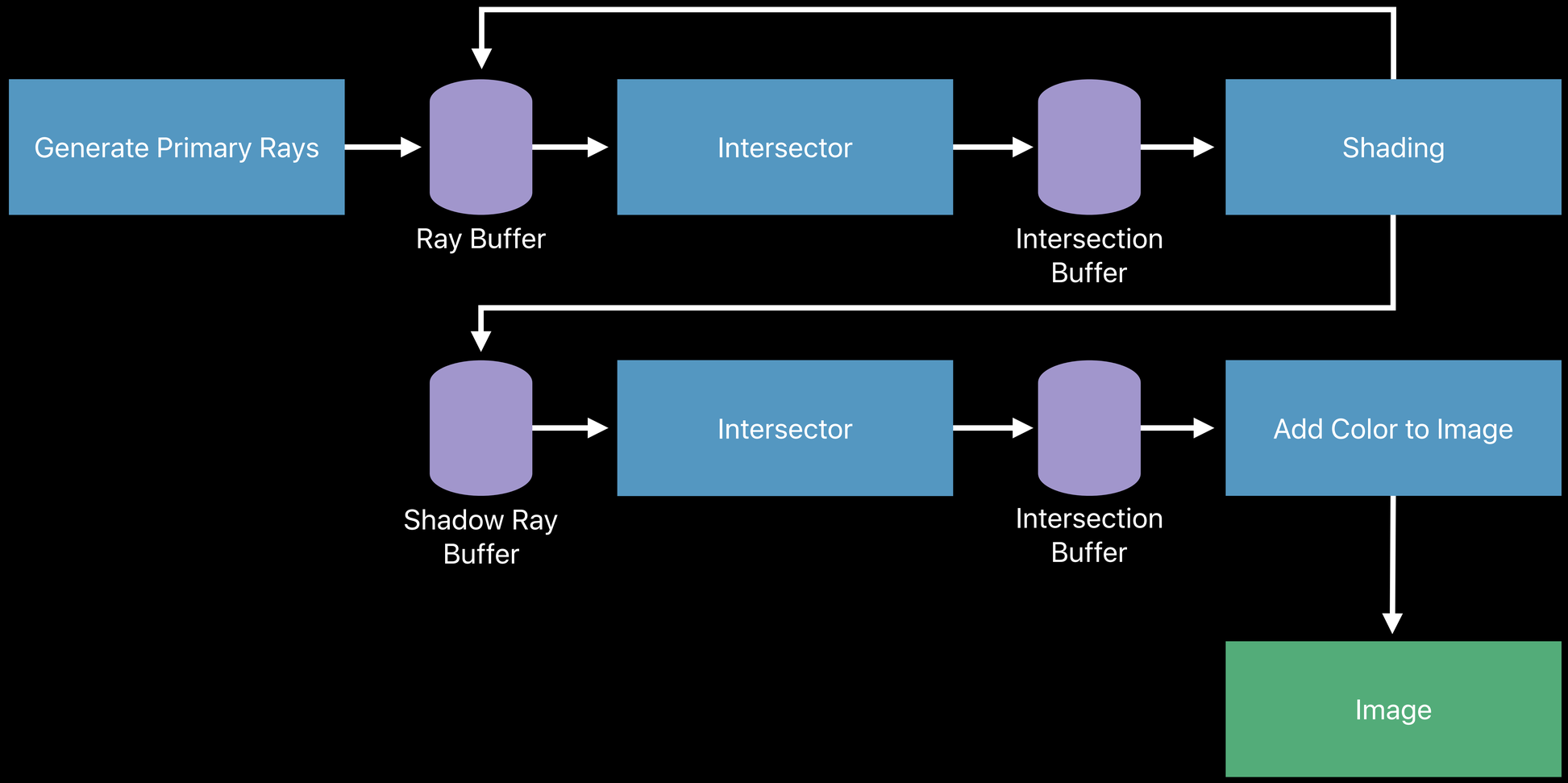
Iterate, continuing in random direction
at each step



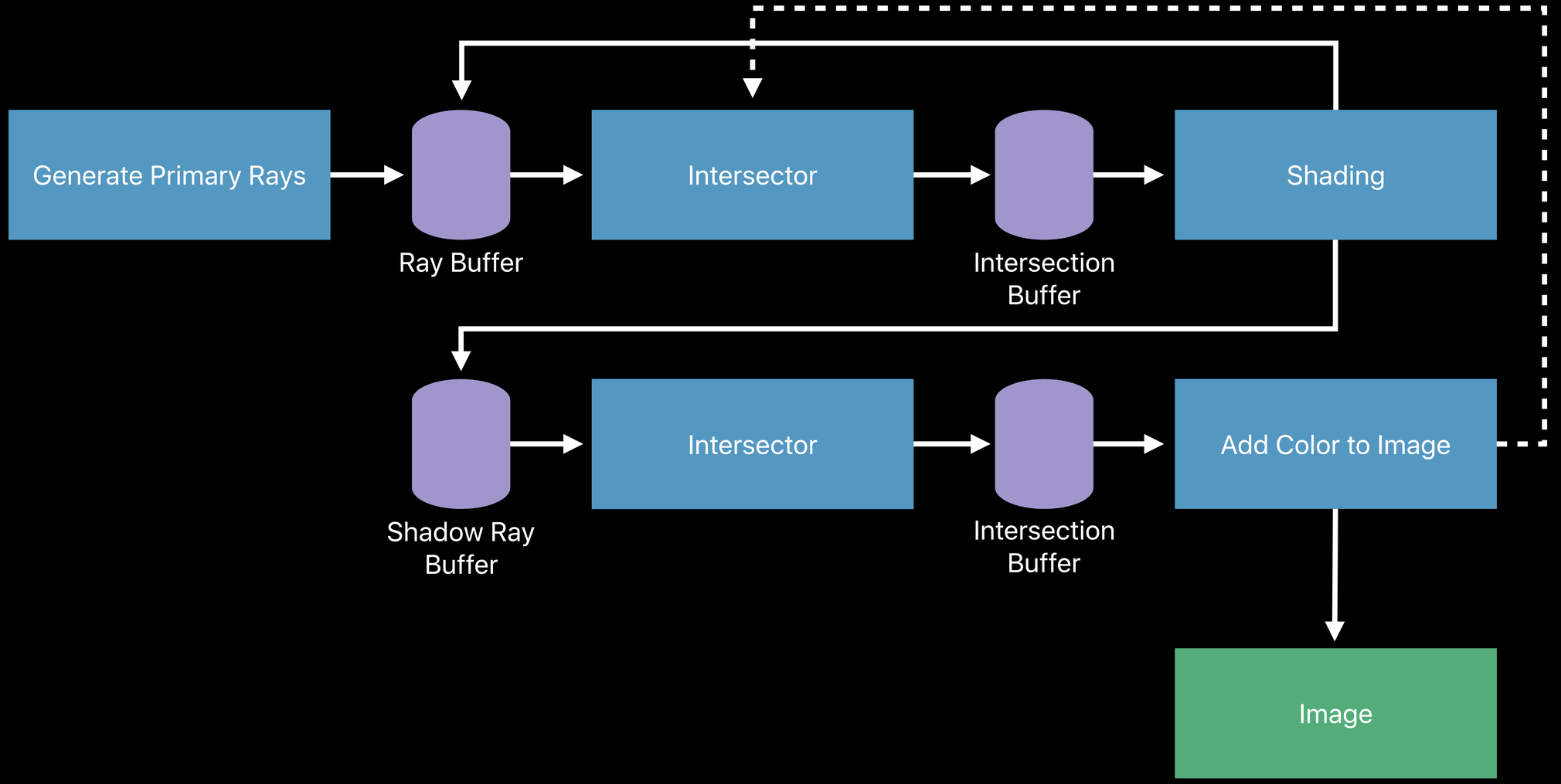
Secondary Rays



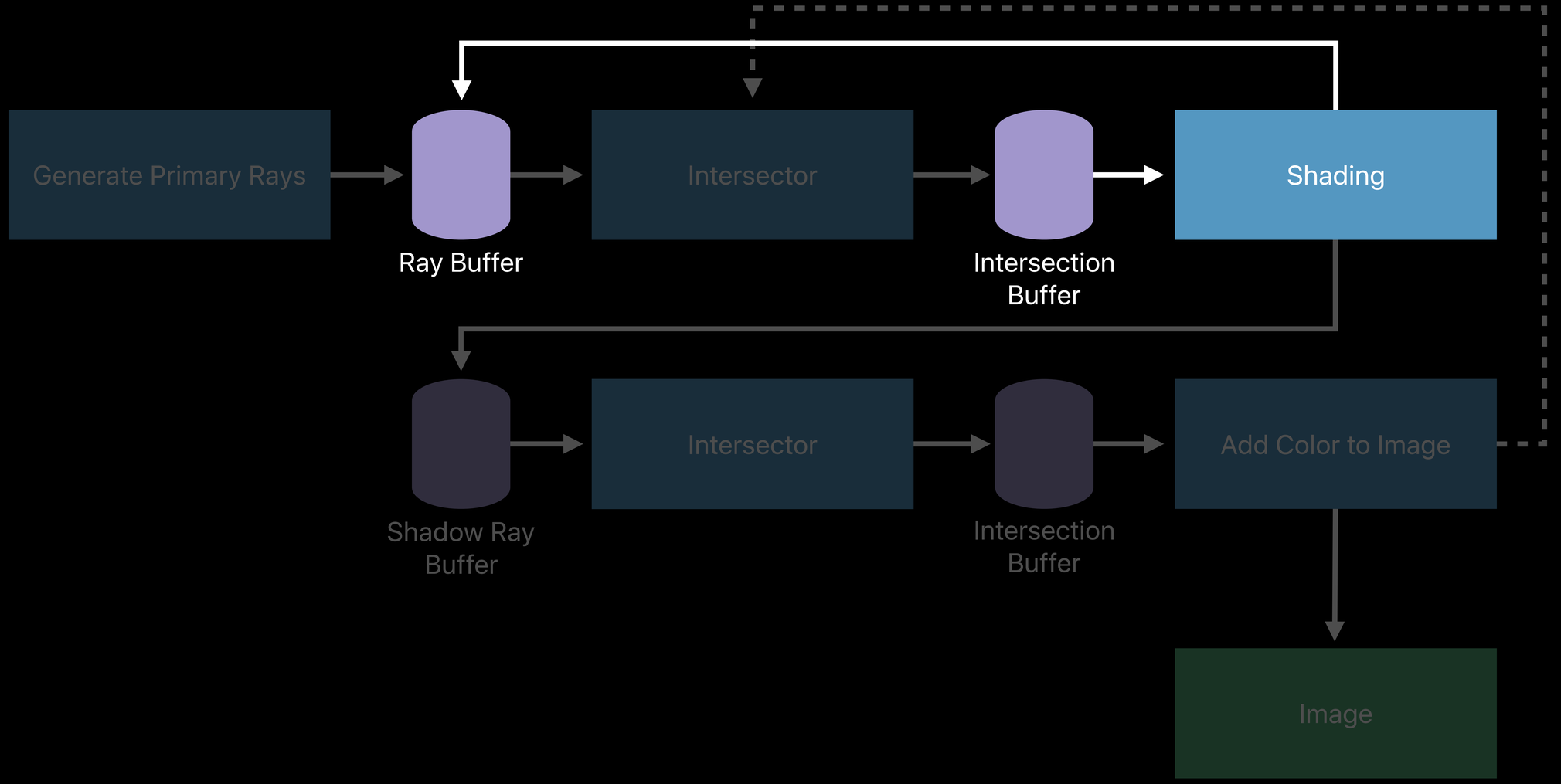
Secondary Rays



Secondary Rays



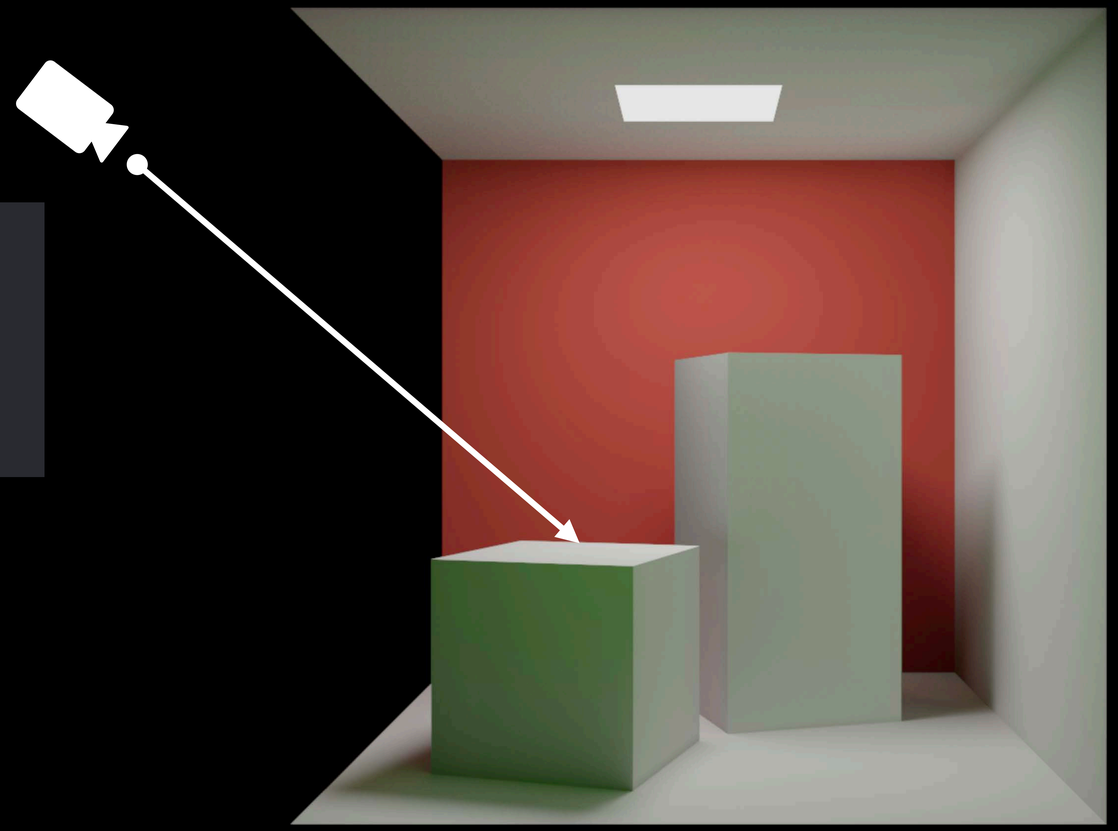
Secondary Rays



Updating Secondary Rays

Update secondary rays during each iteration:

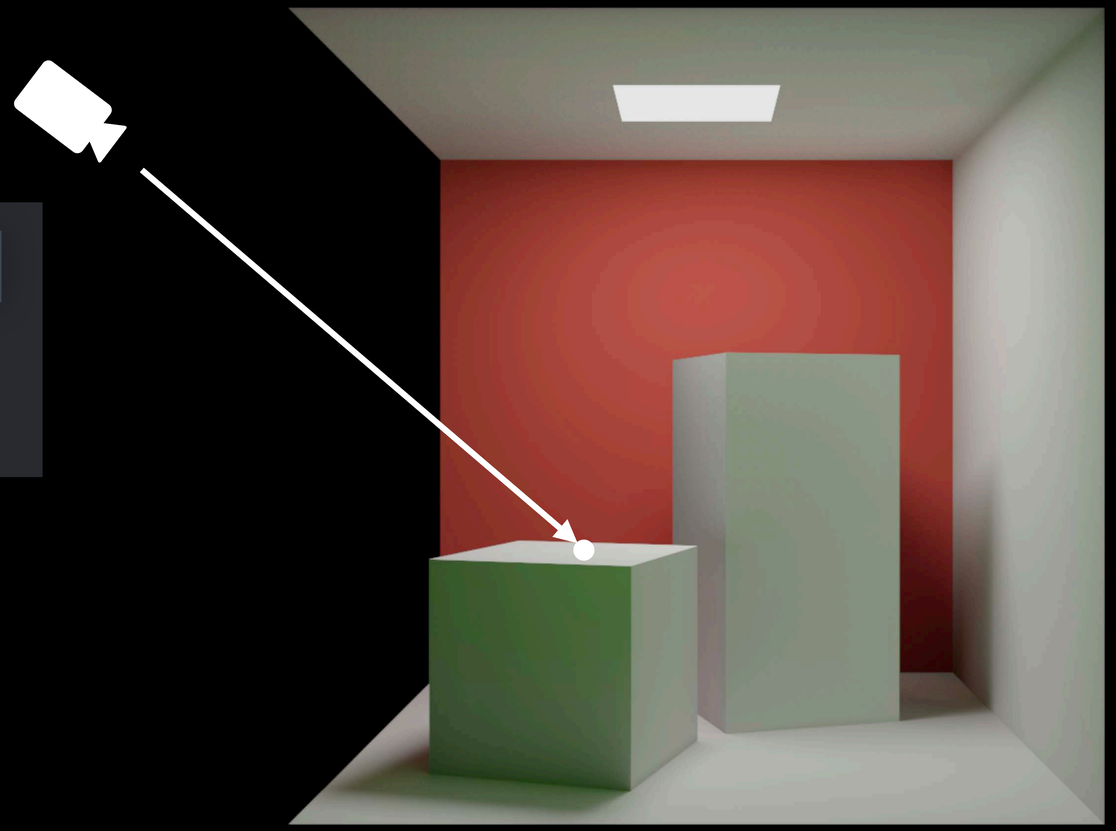
```
ray.origin = intersectionPoint;
ray.direction = getRandomDirection(surfaceNormal);
ray.color *= surfaceColor;
```



Updating Secondary Rays

Update secondary rays during each iteration:

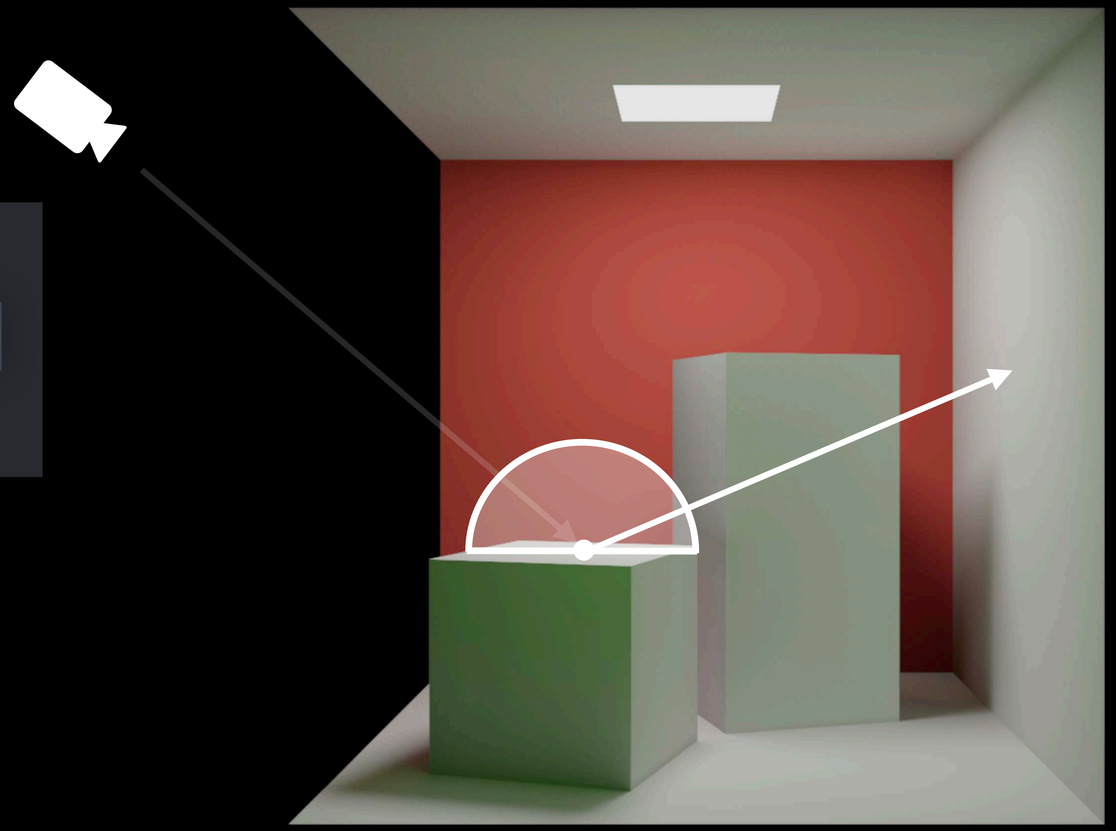
```
ray.origin = intersectionPoint;
ray.direction = getRandomDirection(surfaceNormal);
ray.color *= surfaceColor;
```



Updating Secondary Rays

Update secondary rays during each iteration:

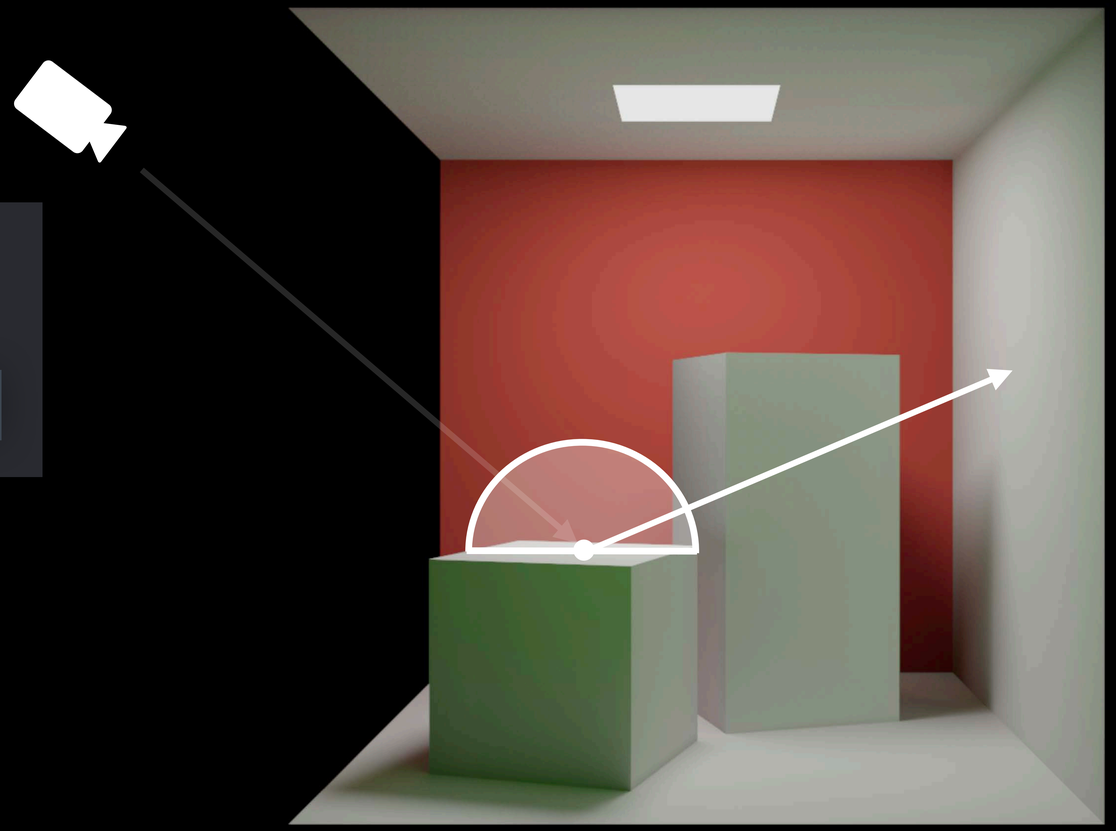
```
ray.origin = intersectionPoint;
ray.direction = getRandomDirection(surfaceNormal);
ray.color *= surfaceColor;
```

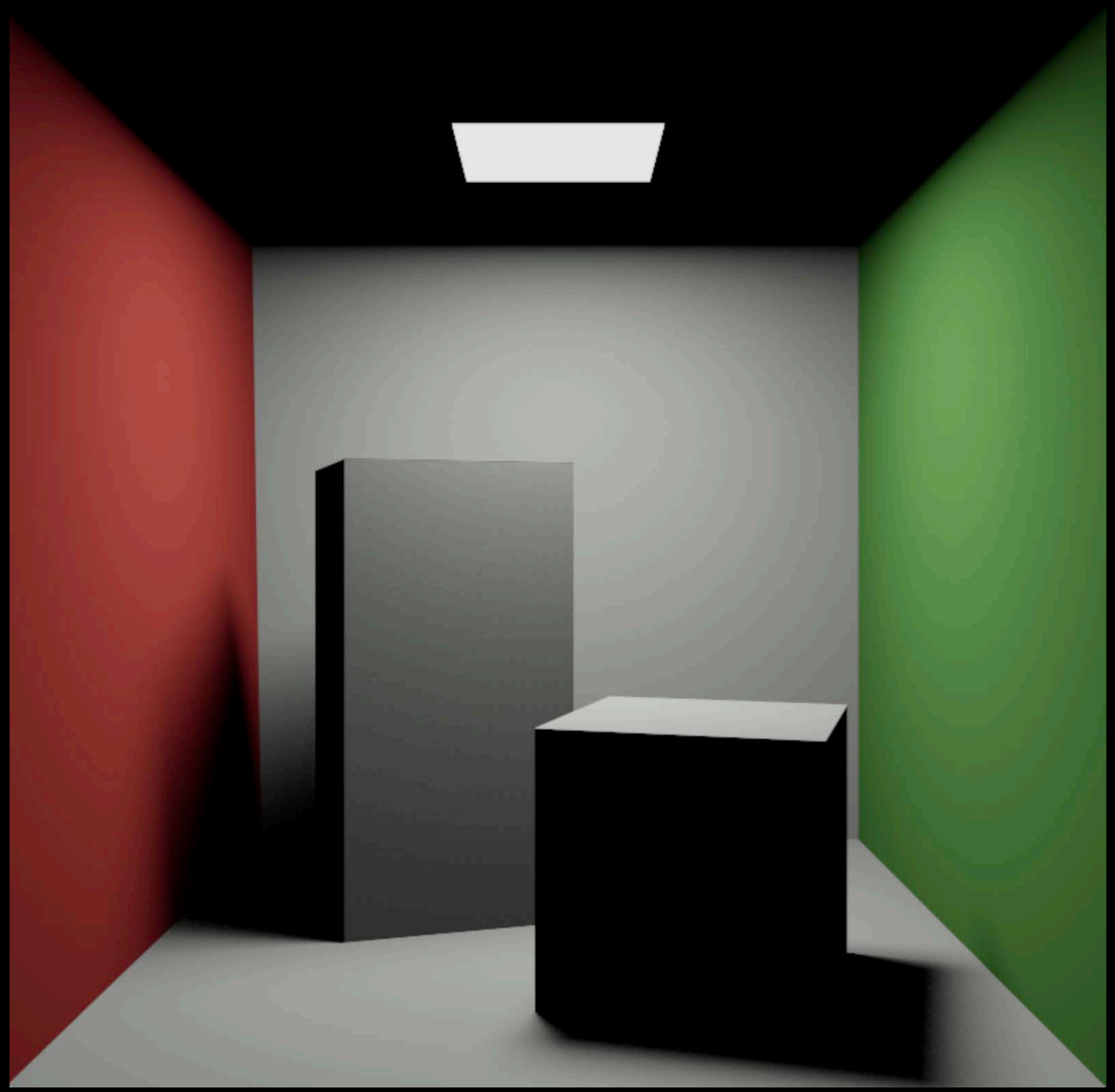


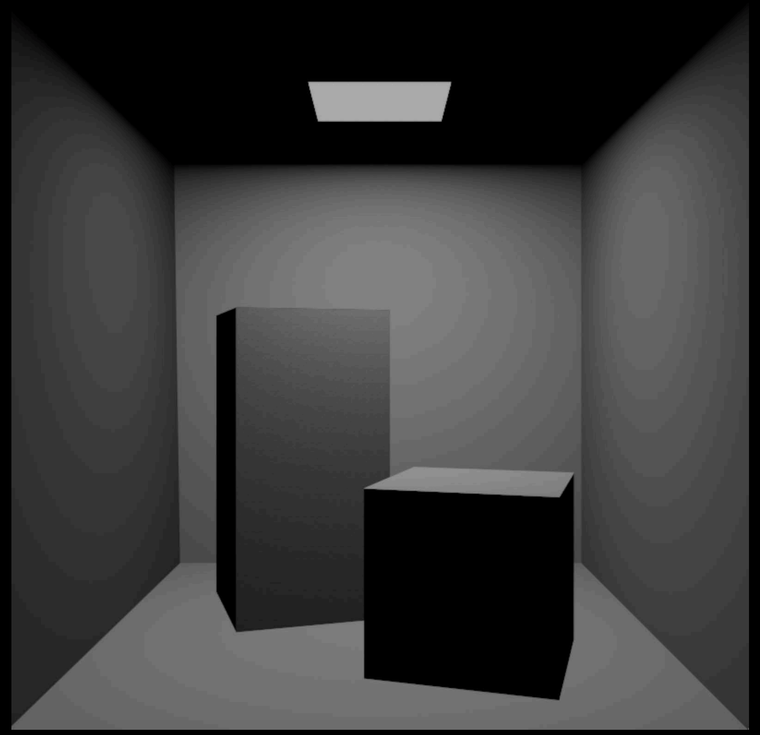
Updating Secondary Rays

Update secondary rays during each iteration:

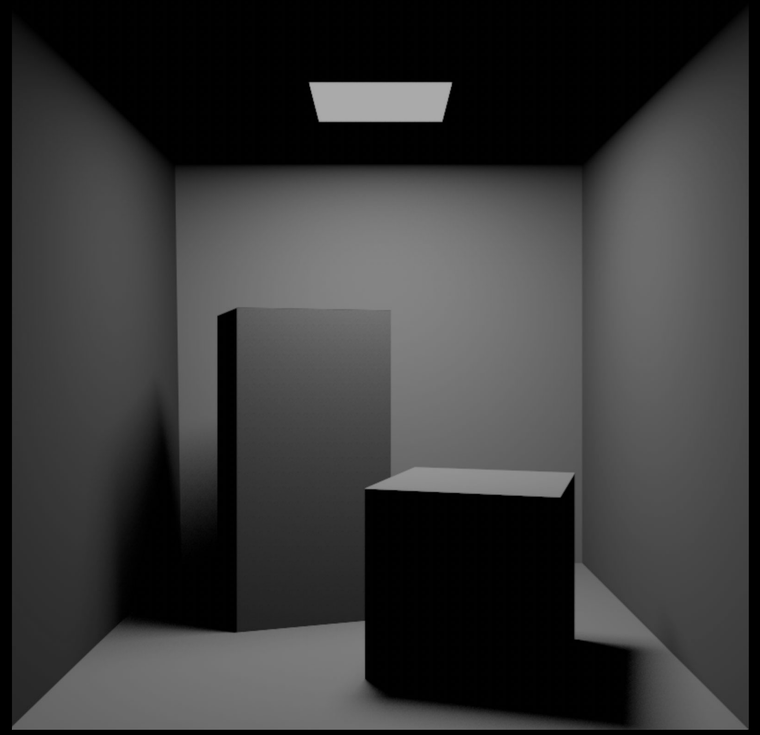
```
ray.origin = intersectionPoint;
ray.direction = getRandomDirection(surfaceNormal);
ray.color *= surfaceColor;
```





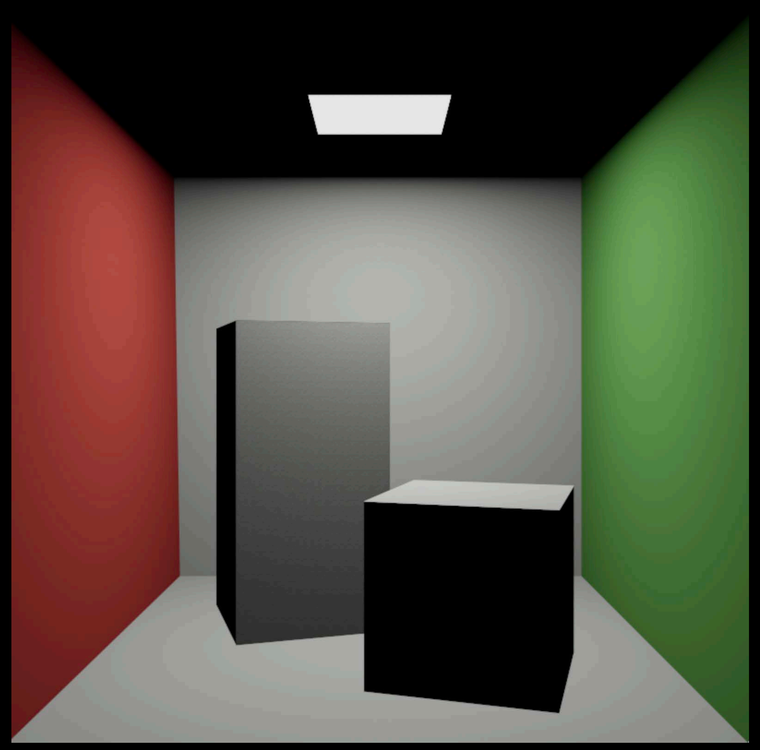


Primary Rays and Shading

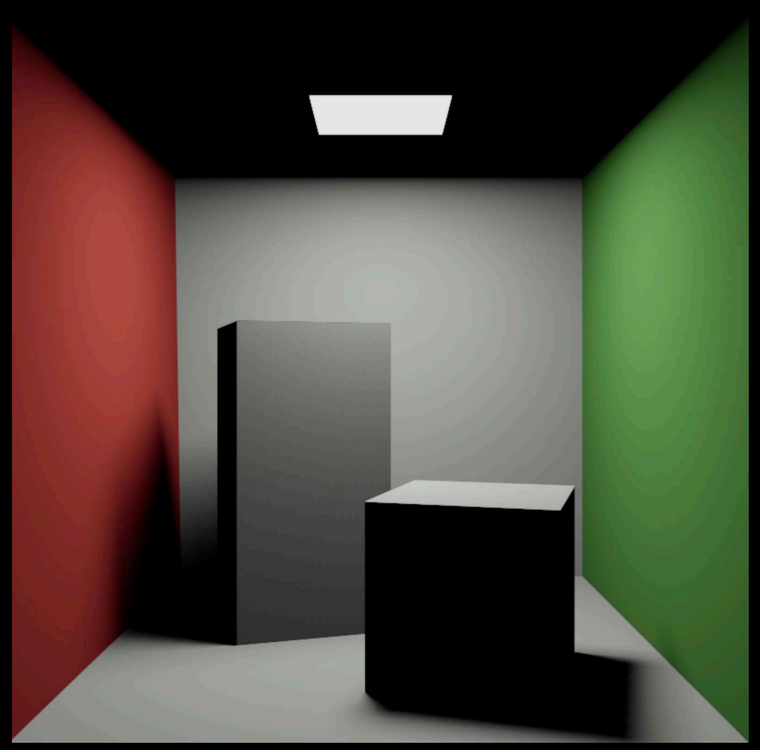


Shadow Rays

Secondary Rays



Primary Rays and Shading



Shadow Rays

Secondary Rays

Demo

Sample Code

This app is available as a sample

Add your own geometry, light sources, camera model, shading model

Refer to documentation for more information

Extending to Multiple GPUs

Wayne Lister, GPU Software Engineer

Multi-GPU Ray Tracing

Split work across GPUs

Multi-GPU Ray Tracing

Split work across GPUs

Copy data between GPUs

Multi-GPU Ray Tracing

Split work across GPUs

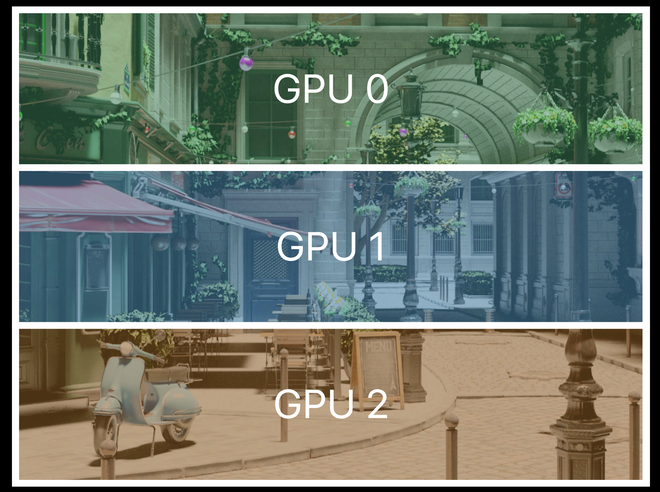
Copy data between GPUs

Synchronize execution

Splitting Work Across GPUs

Rendering

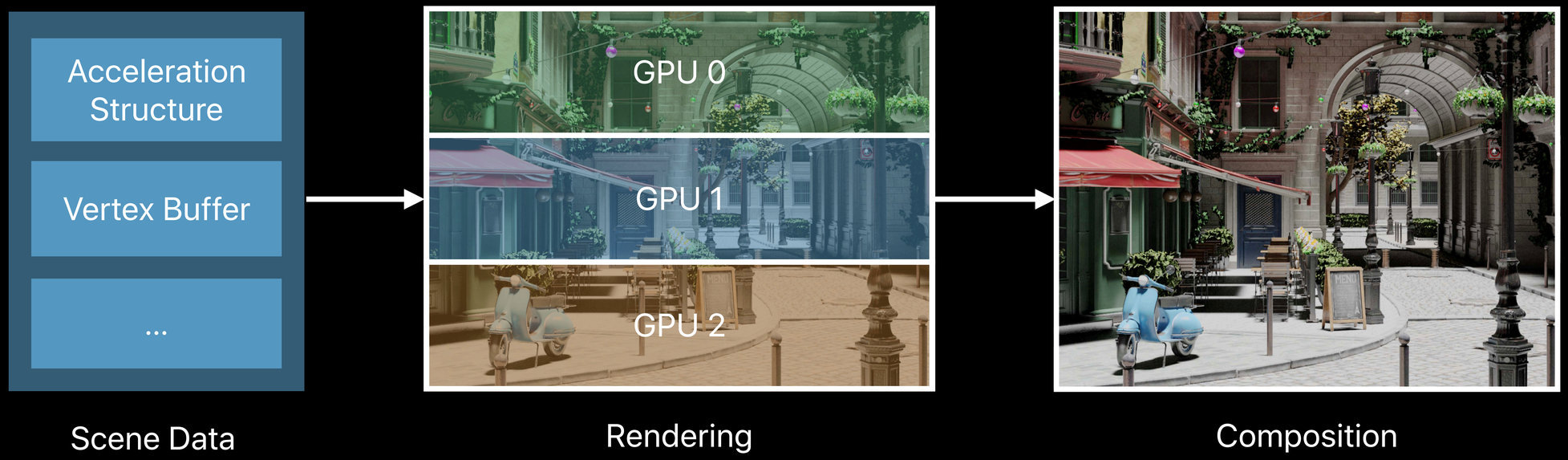
Splitting Work Across GPUs



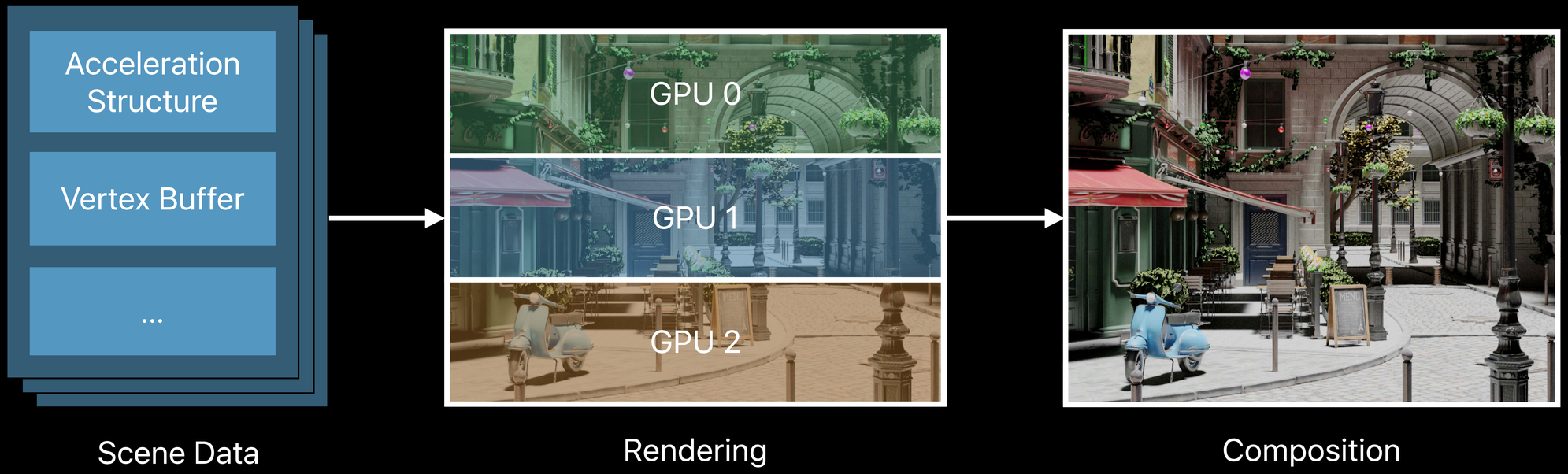
Rendering

Splitting Work Across GPUs

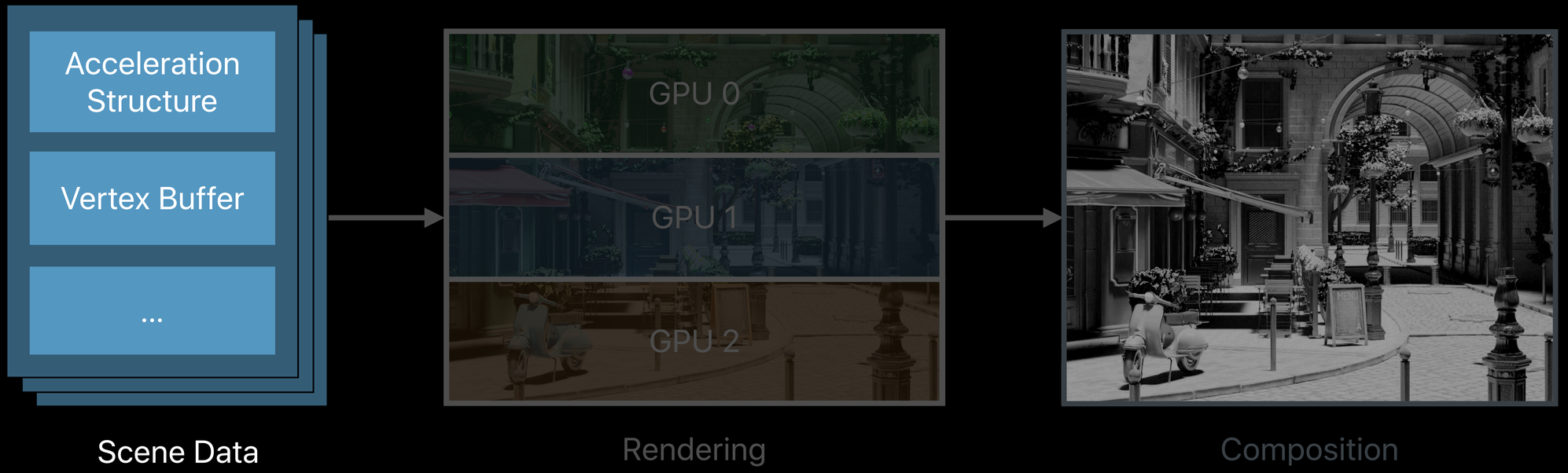
Splitting Work Across GPUs



Splitting Work Across GPUs

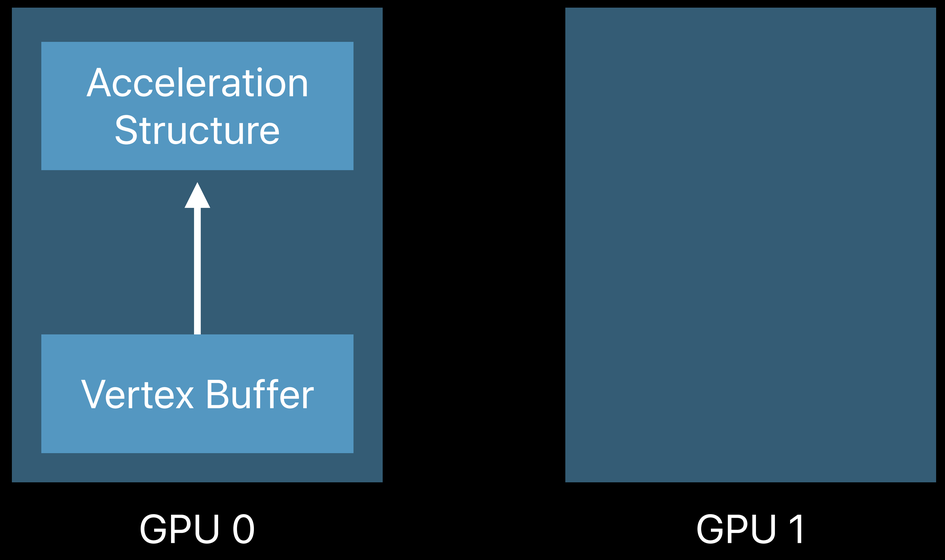


Splitting Work Across GPUs



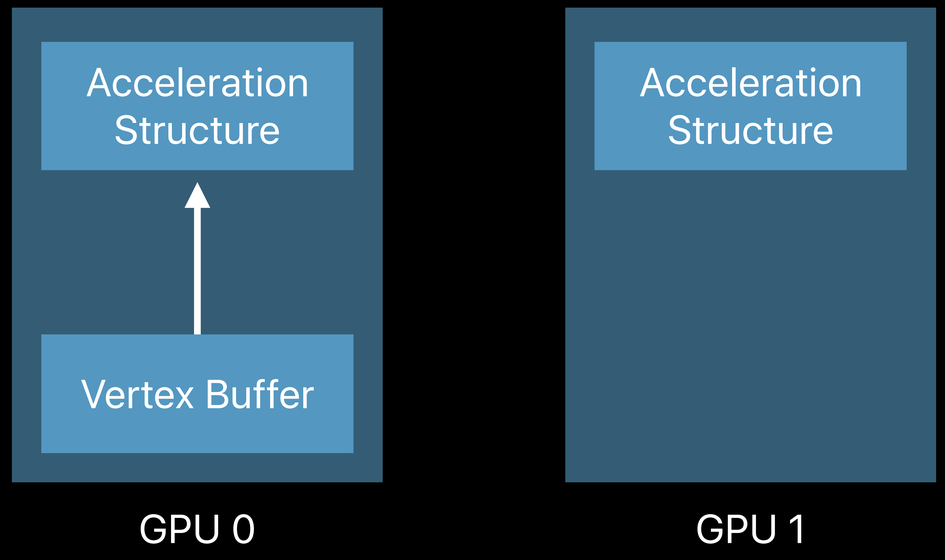
Copying Scene Data

```
let copy = accelerationStructure.copy(with: nil, device: newDevice)
copy.vertexBuffer = copyBuffer(accelerationStructure.vertexBuffer, device: newDevice)
```



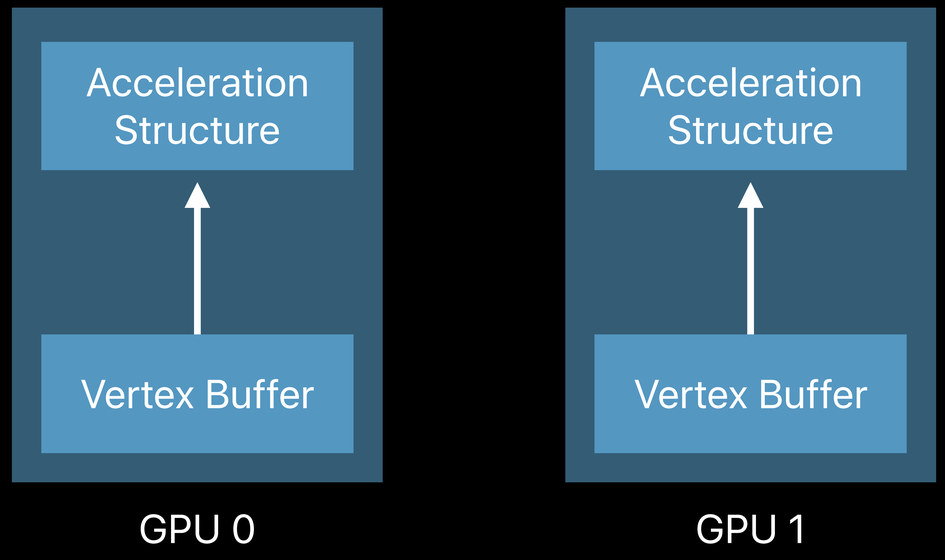
Copying Scene Data

```
let copy = accelerationStructure.copy(with: nil, device: newDevice)
copy.vertexBuffer = copyBuffer(accelerationStructure.vertexBuffer, device: newDevice)
```

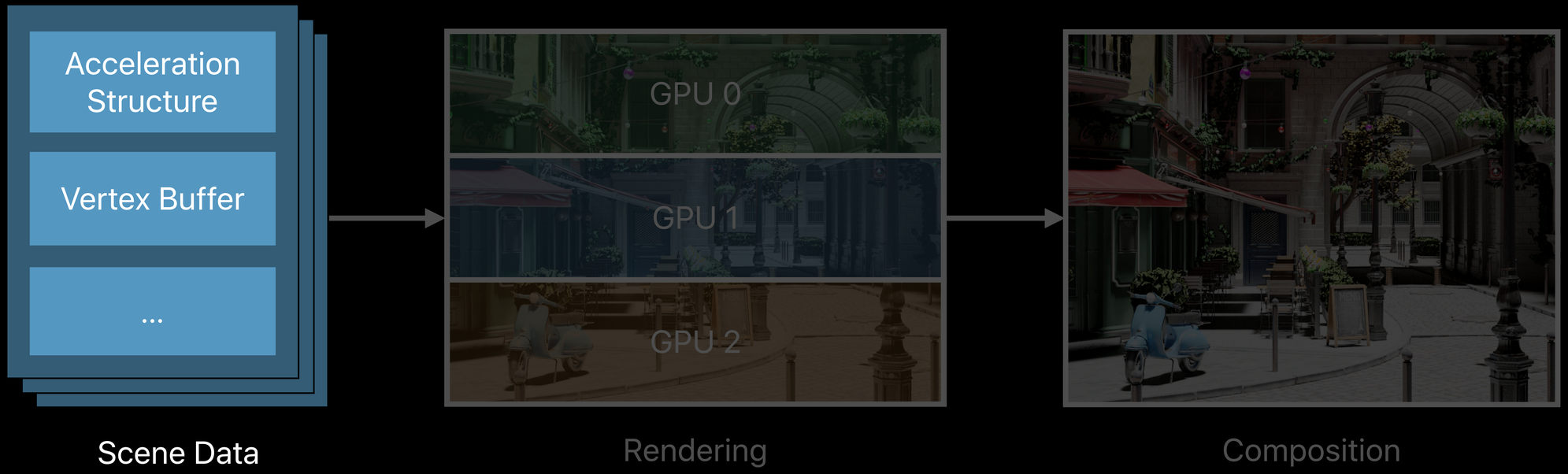


Copying Scene Data

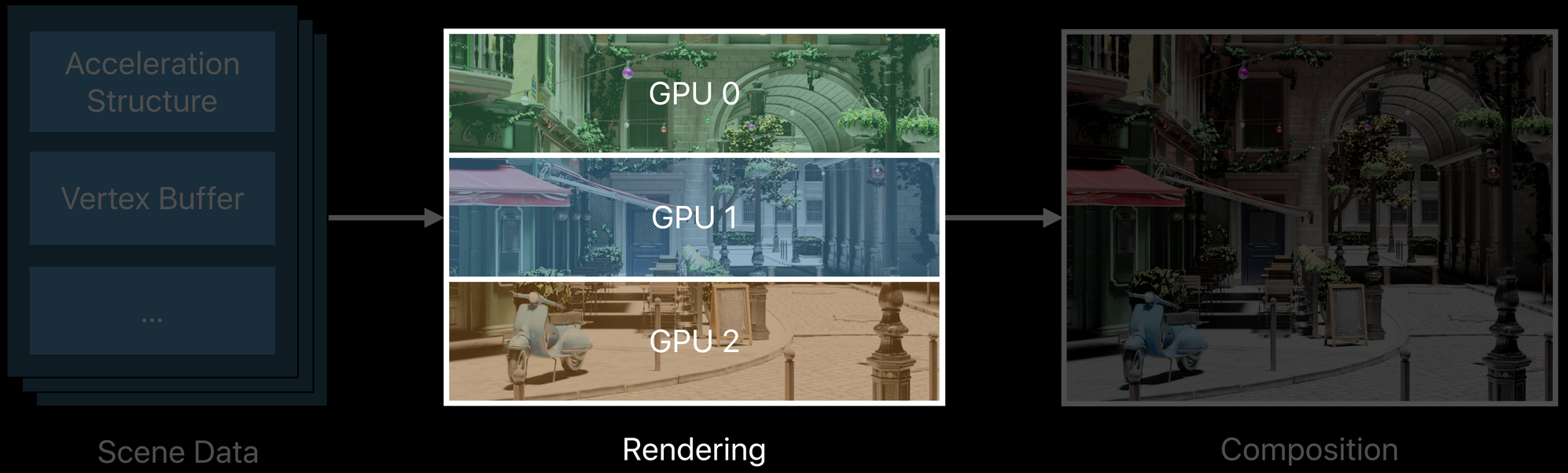
```
let copy = accelerationStructure.copy(with: nil, device: newDevice)
copy.vertexBuffer = copyBuffer(accelerationStructure.vertexBuffer, device: newDevice)
```



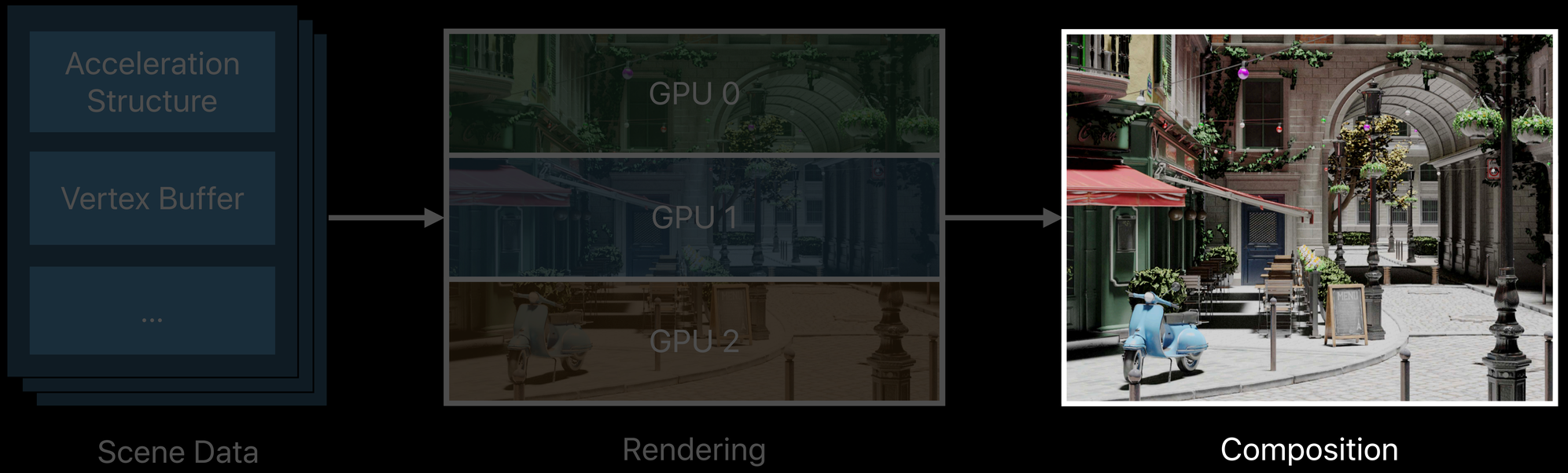
Splitting Work Across GPUs



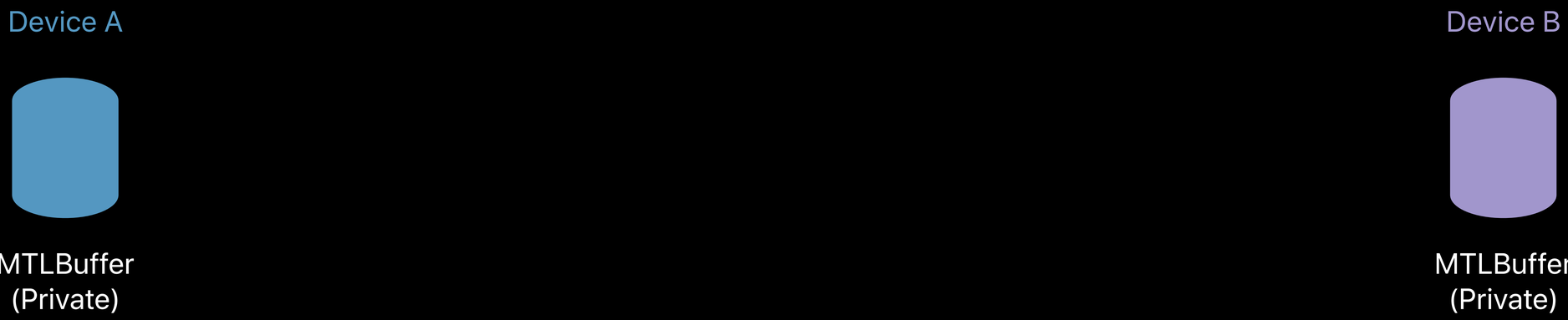
Splitting Work Across GPUs



Splitting Work Across GPUs

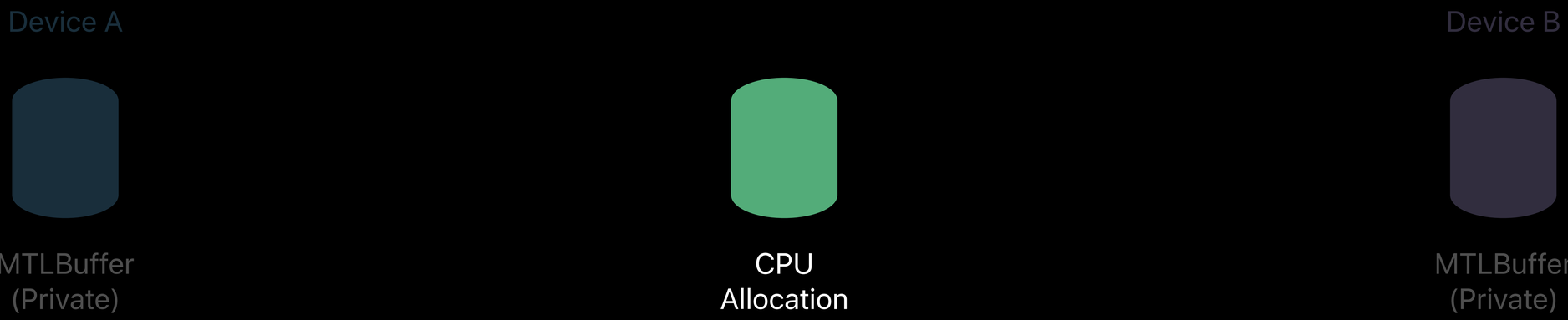


Creating a Shared CPU Allocation

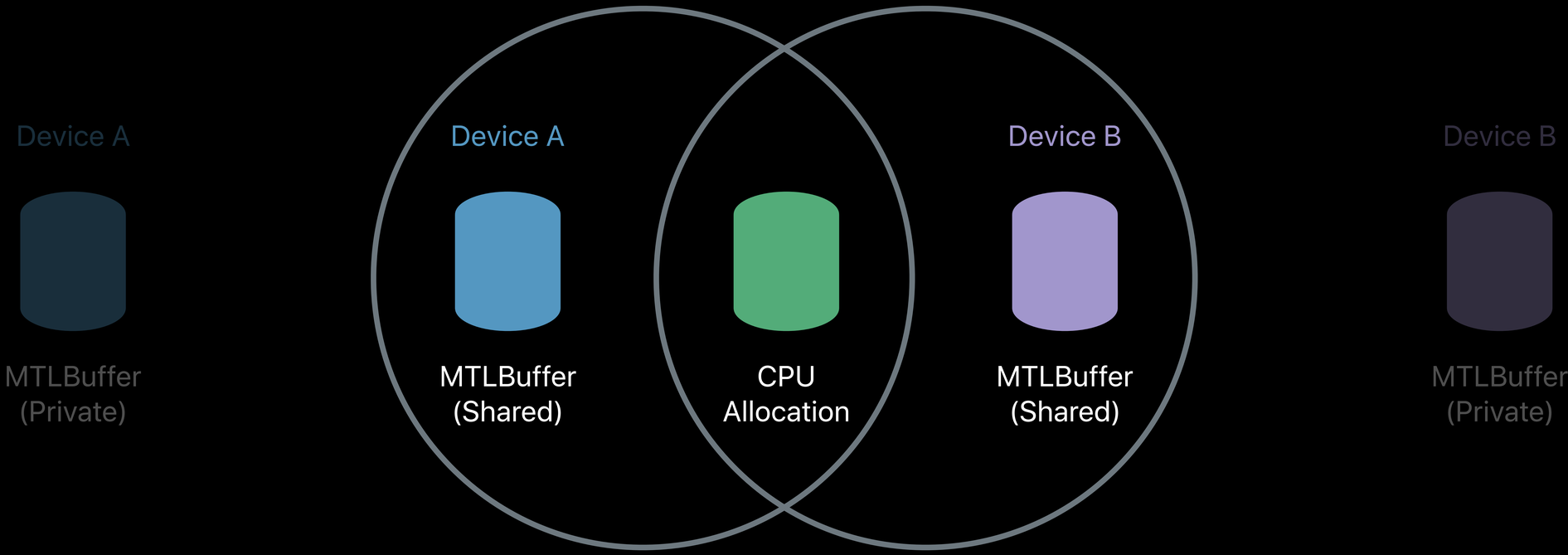


Creating a Shared CPU Allocation

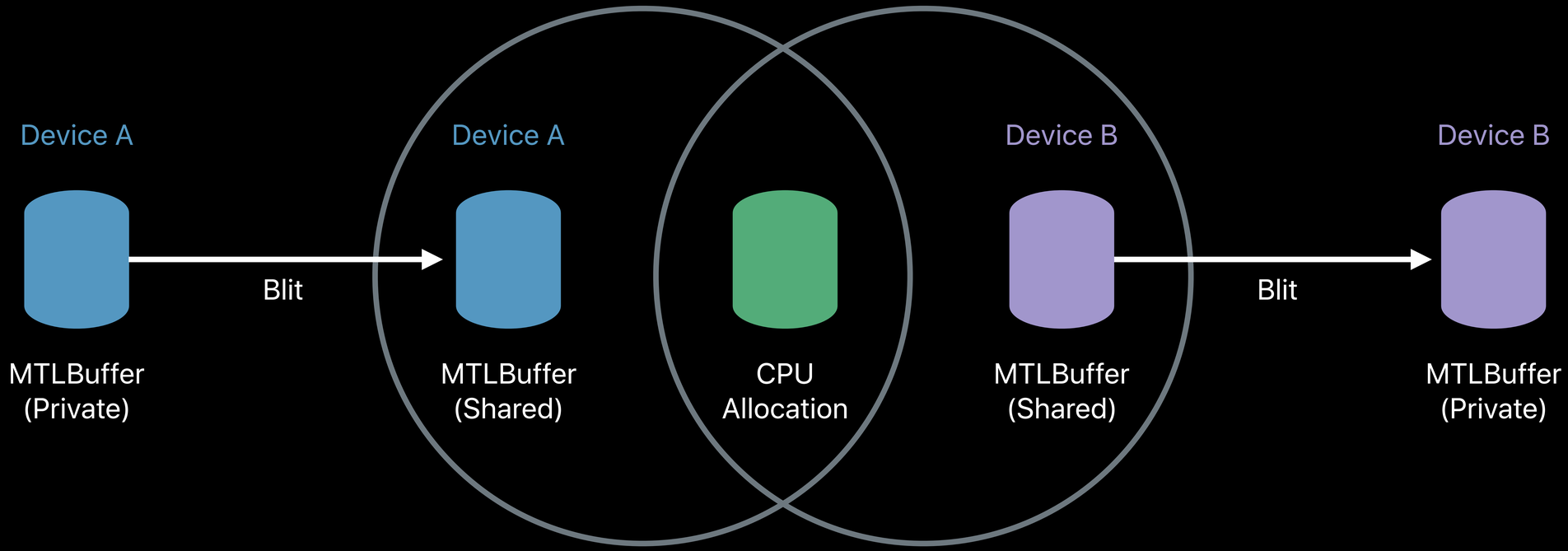
Creating a Shared CPU Allocation



Creating a Shared CPU Allocation



Creating a Shared CPU Allocation



Creating a Shared CPU Allocation

```
let bufferA = deviceA.makeBuffer(length: bufferLength,
                                 options: .shared)!

let bufferB = deviceB.makeBuffer(bytesNoCopy: bufferA.contents(),
                                       length: bufferA.length,
                                      options: .shared
                                  deallocator: nil)!
```

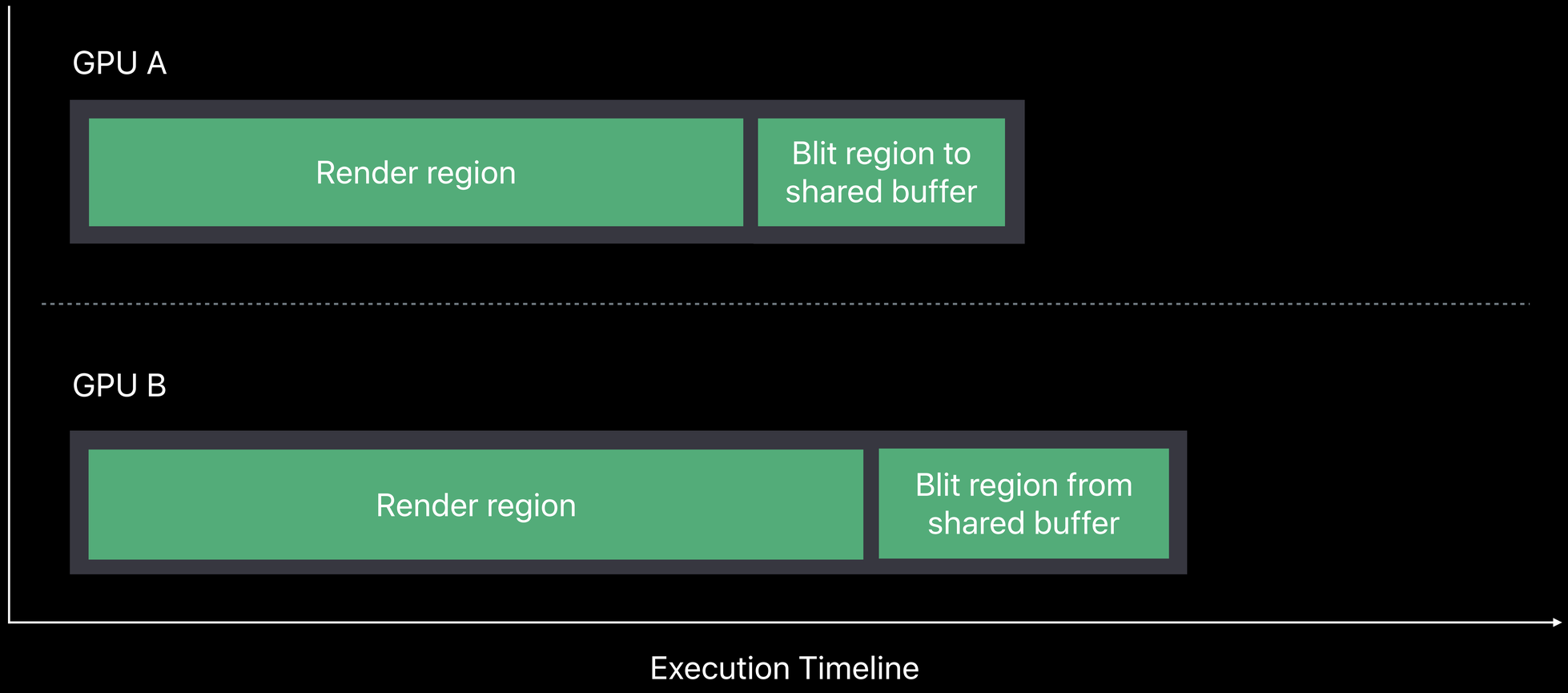
Creating a Shared CPU Allocation

```
let bufferA = deviceA.makeBuffer(length: bufferLength,
                                 options: .shared)!

let bufferB = deviceB.makeBuffer(bytesNoCopy: bufferA.contents(),
                                       length: bufferA.length,
                                      options: .shared
                                  deallocator: nil)!
```

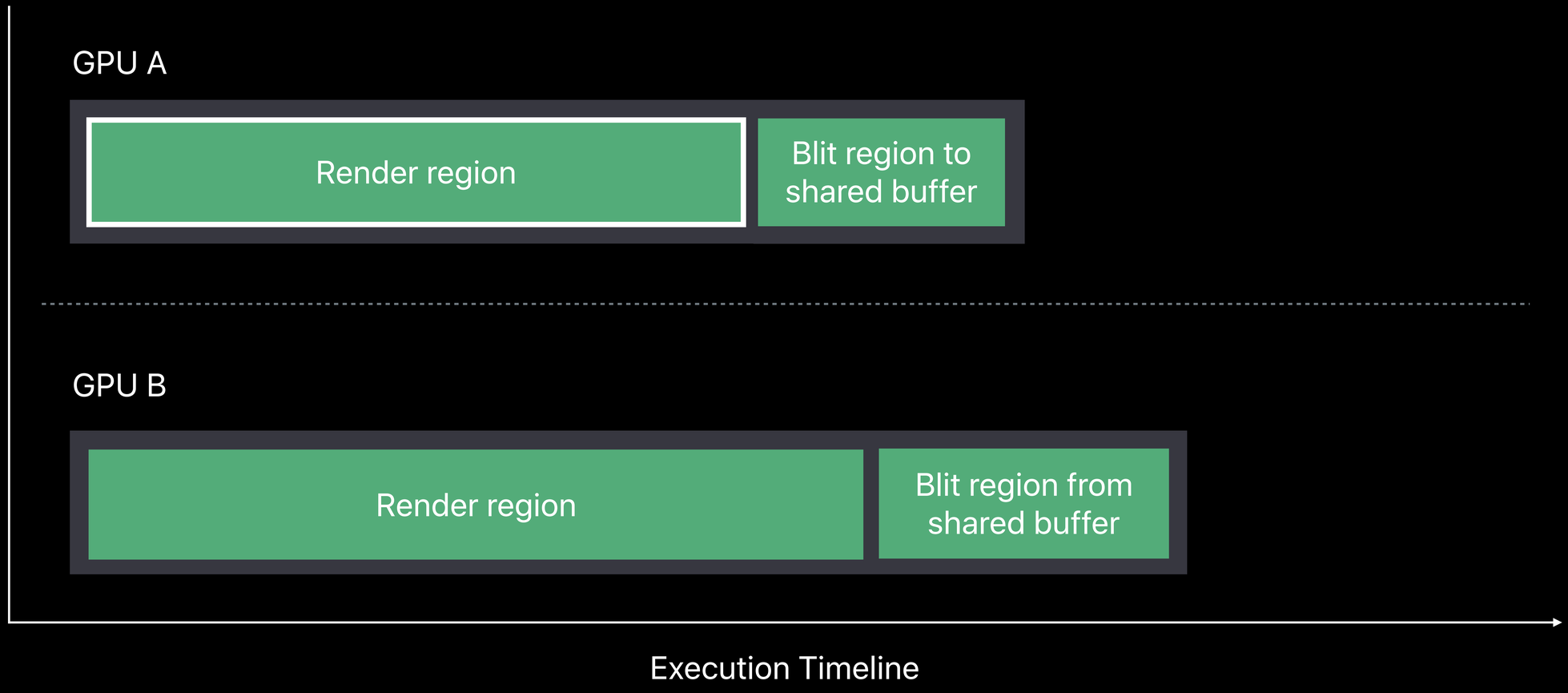
Creating a Shared CPU Allocation

```
let bufferA = deviceA.makeBuffer(length: bufferLength,
                                 options: .shared)!

let bufferB = deviceB.makeBuffer(bytesNoCopy: bufferA.contents(),
                                       length: bufferA.length,
                                      options: .shared
                                  deallocator: nil)!
```

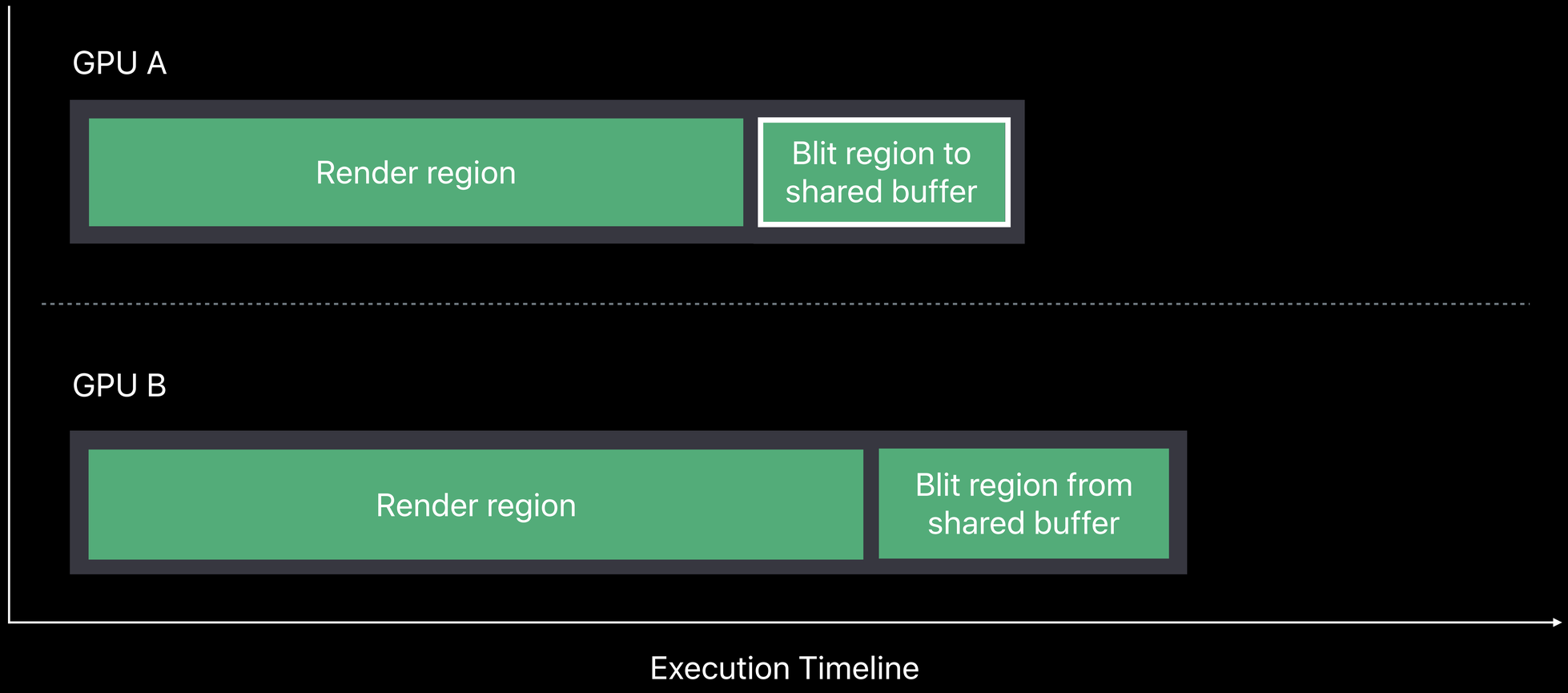
Synchronizing with Metal Events



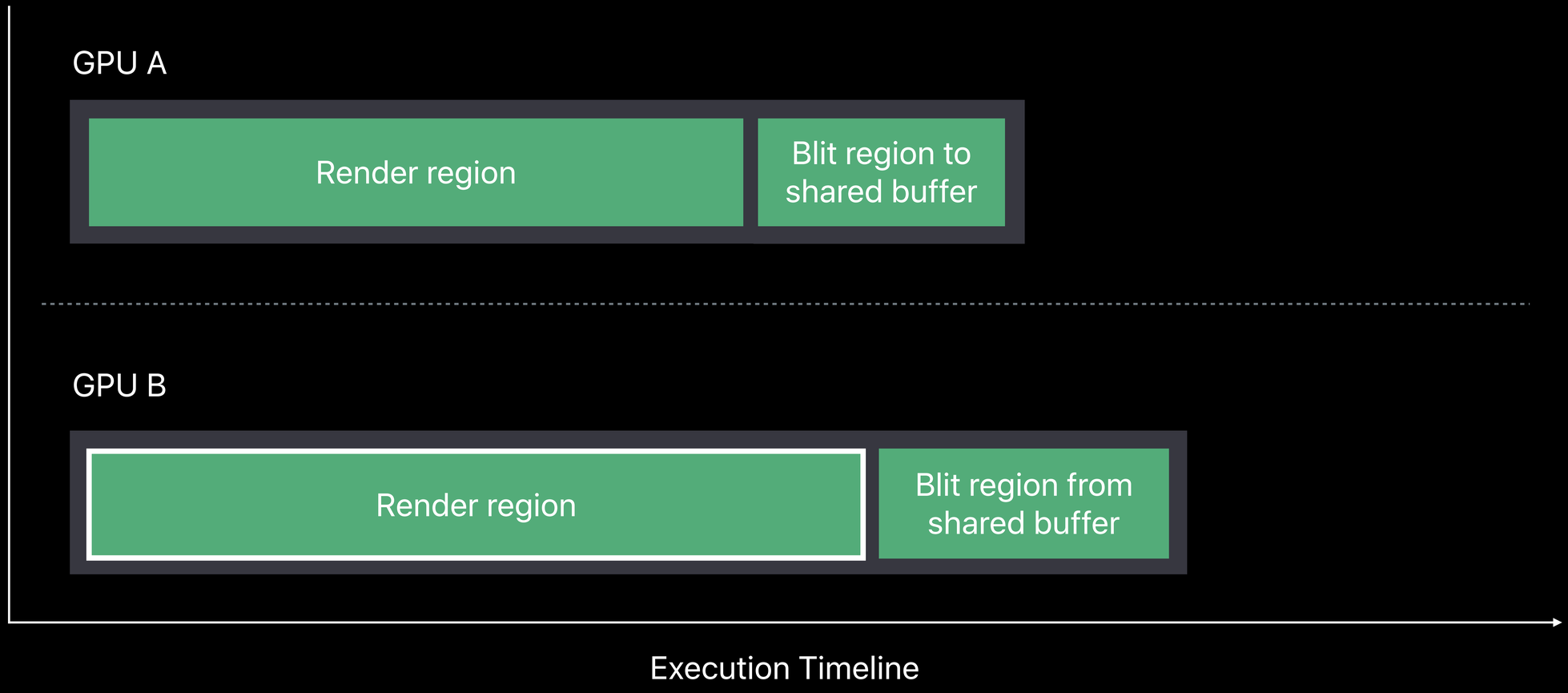
Synchronizing with Metal Events



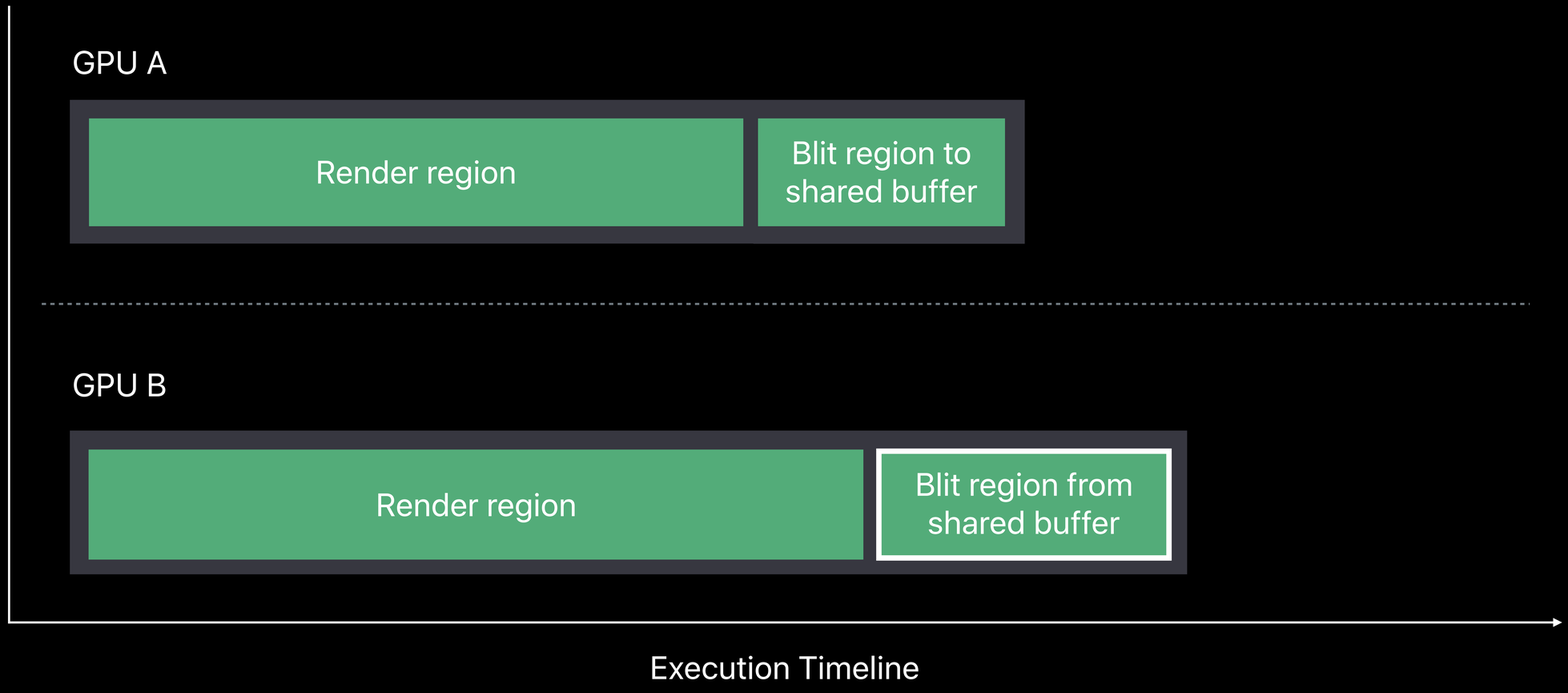
Synchronizing with Metal Events



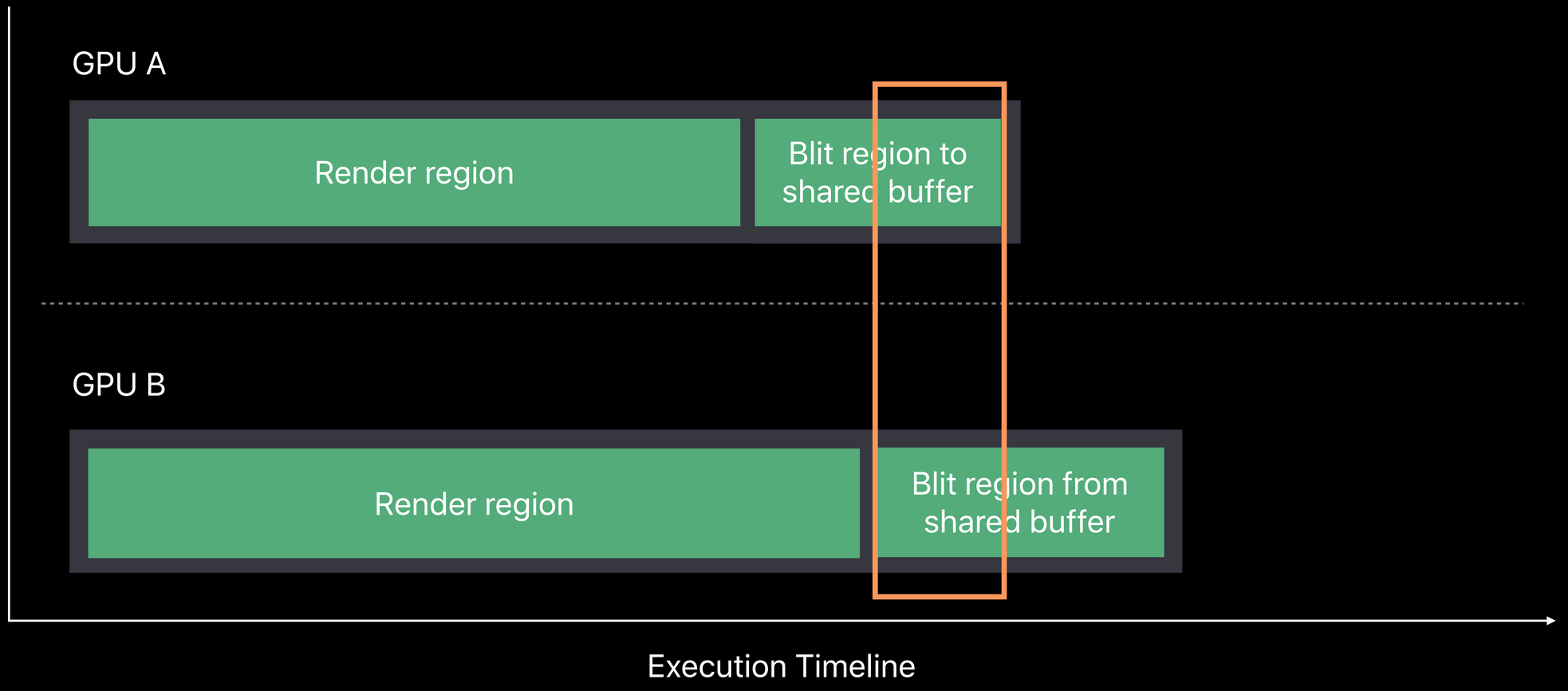
Synchronizing with Metal Events



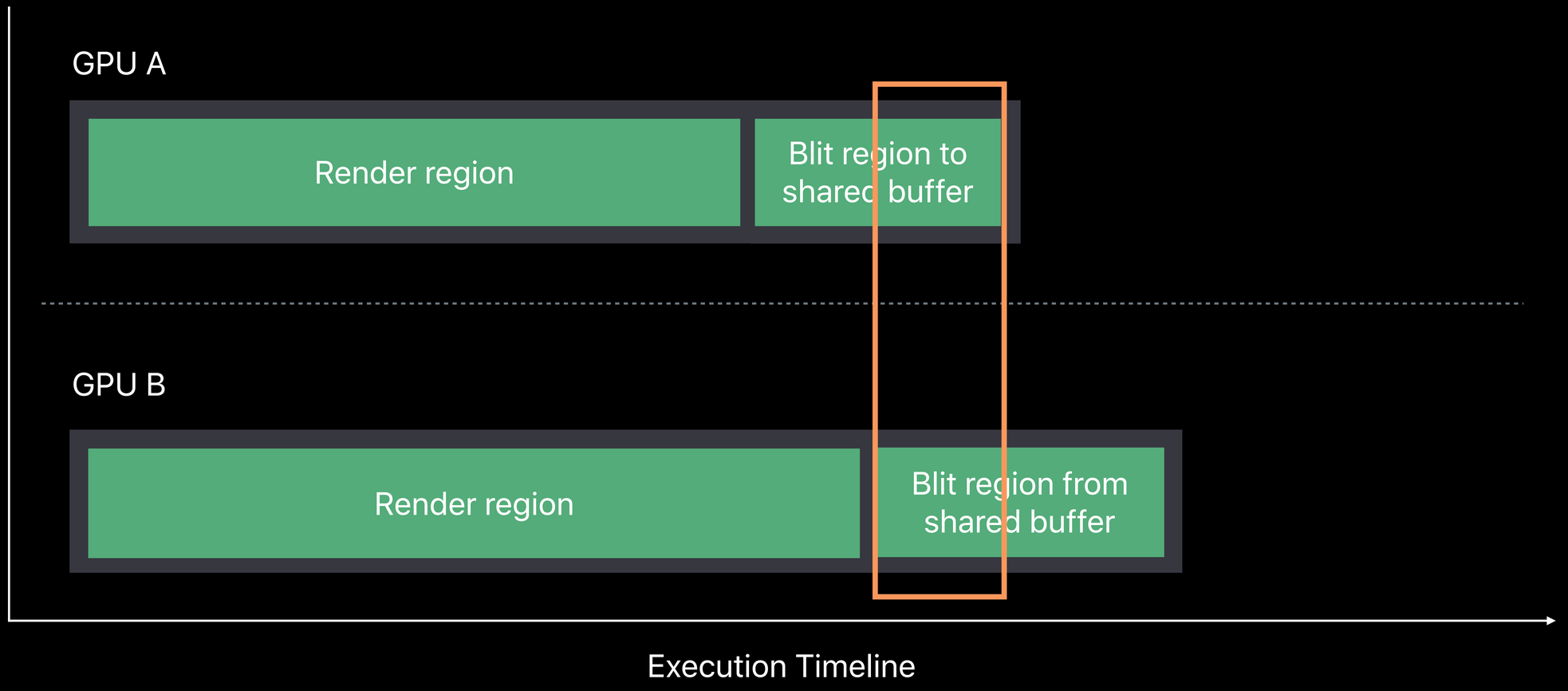
Synchronizing with Metal Events



Synchronizing with Metal Events

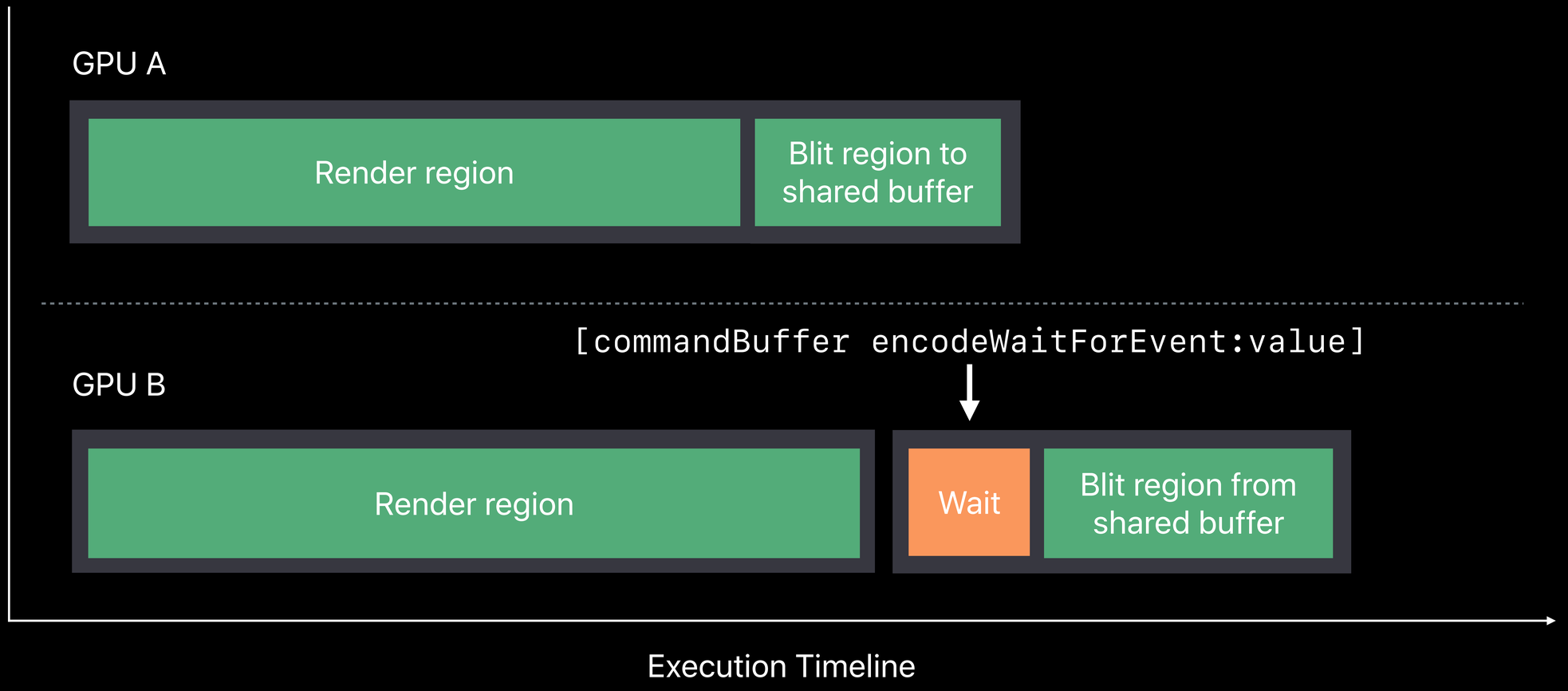


Synchronizing with Metal Events



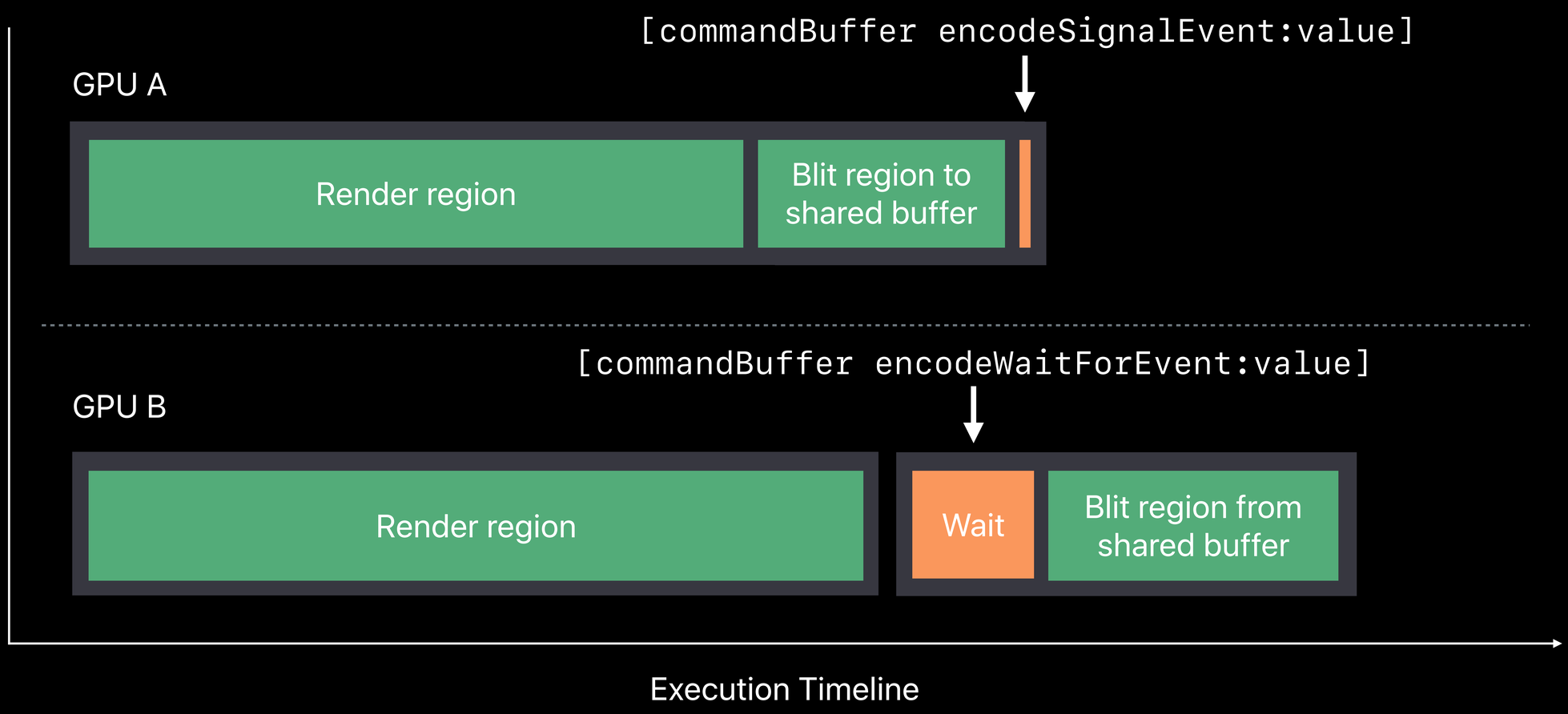
Synchronizing with Metal Events

NEW



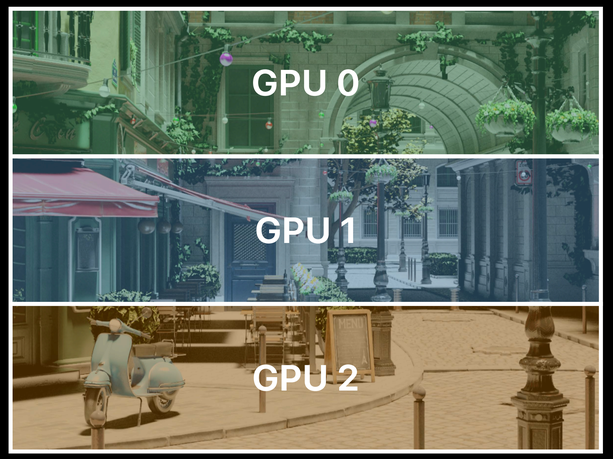
Synchronizing with Metal Events

NEW



Load Balancing

GPUs can have different performance

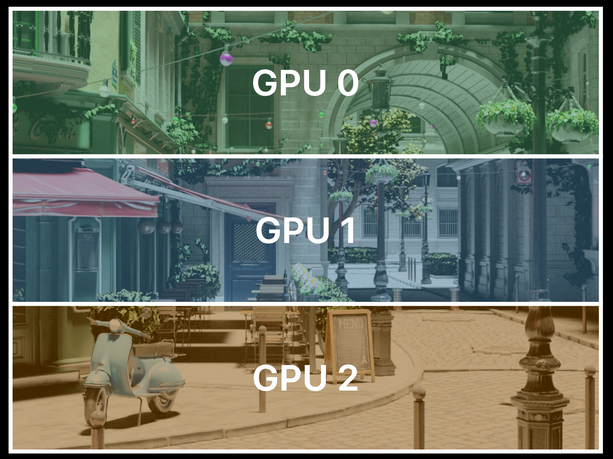


Fixed Partitions

Load Balancing

GPUs can have different performance

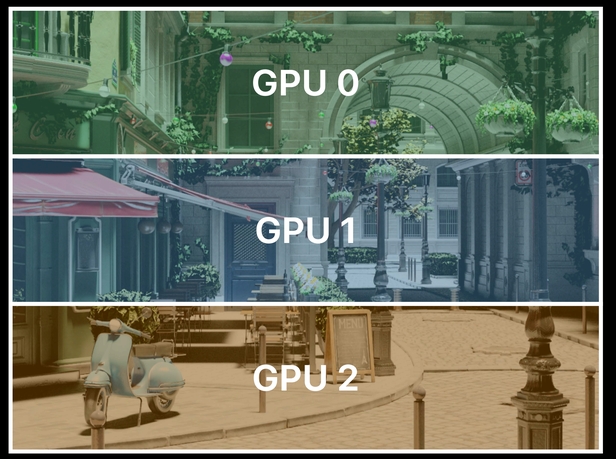
Some regions are more complex to render



Fixed Partitions

Load Balancing

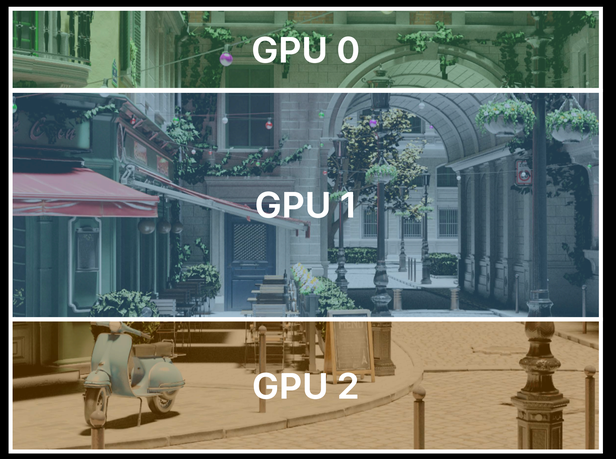
Adjust region sizes to keep GPUs fully utilized



Fixed Partitions

Load Balancing

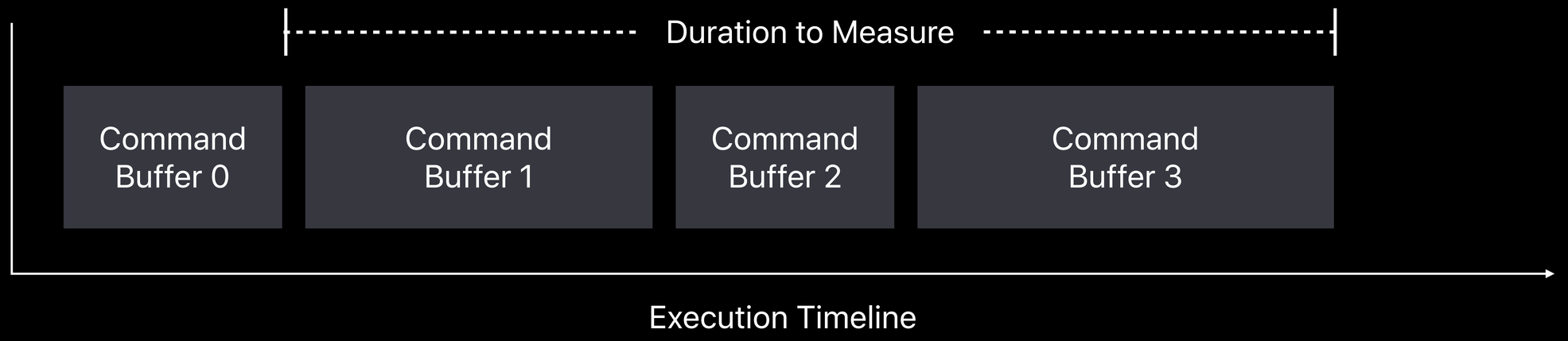
Adjust region sizes to keep GPUs fully utilized



Adaptive Partitions

Timing GPU Work

```
commandBuffer.addCompletedHandler { commandBuffer in
    completionTime[commandBuffer] = mach_absolute_time()
}
```



Demo

Summary

Accelerates ray/triangle intersection on the GPU

Summary

Accelerates ray/triangle intersection on the GPU

Optimized for macOS and iOS

Summary

Accelerates ray/triangle intersection on the GPU

Optimized for macOS and iOS

Scales across multiple GPUs

More Information

https://developer.apple.com/wwdc18/606

Metal for Ray Tracing Acceleration Lab	Technology Lab 6	Thursday 12:00 PM


WWDC18