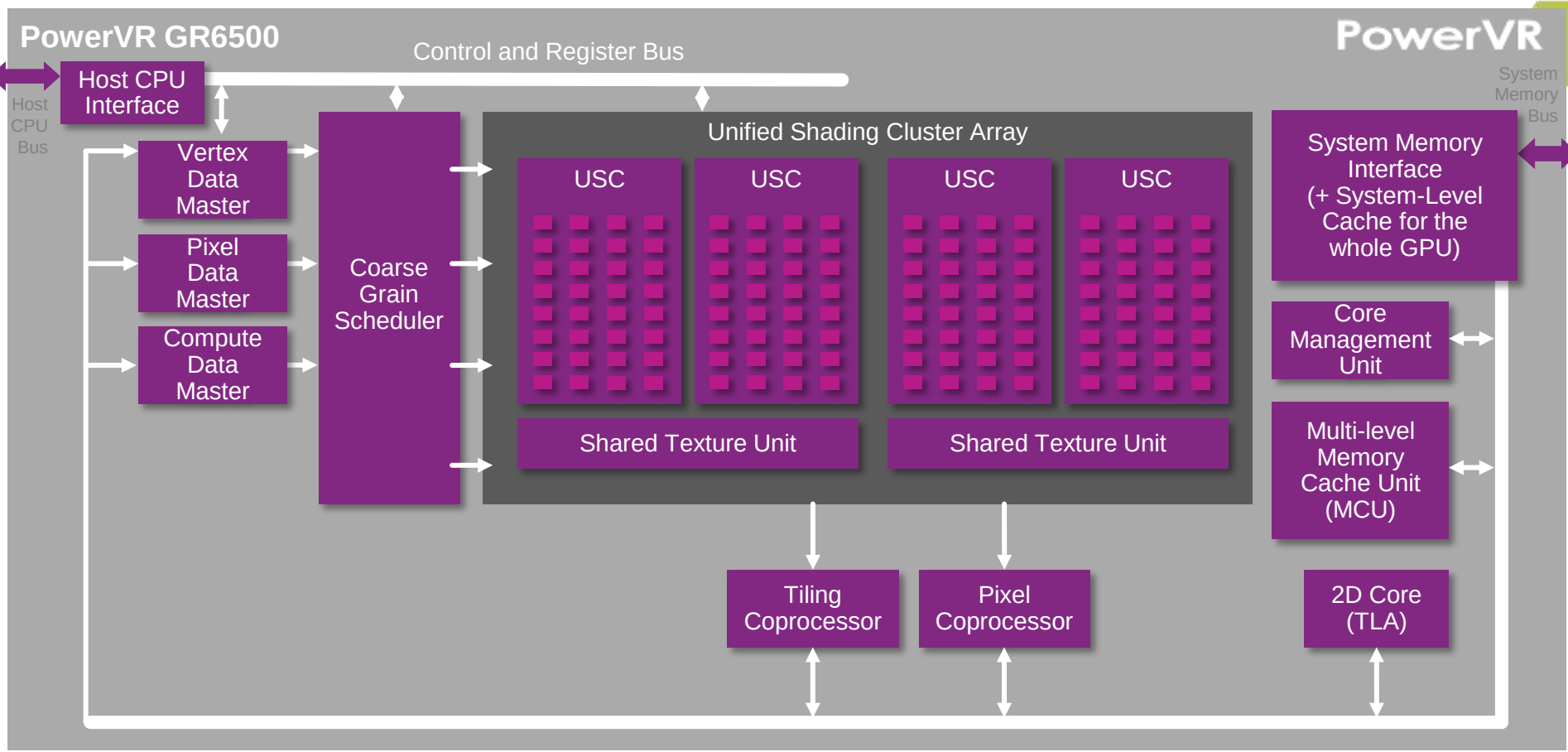
idc16

Imagination Developers Connection

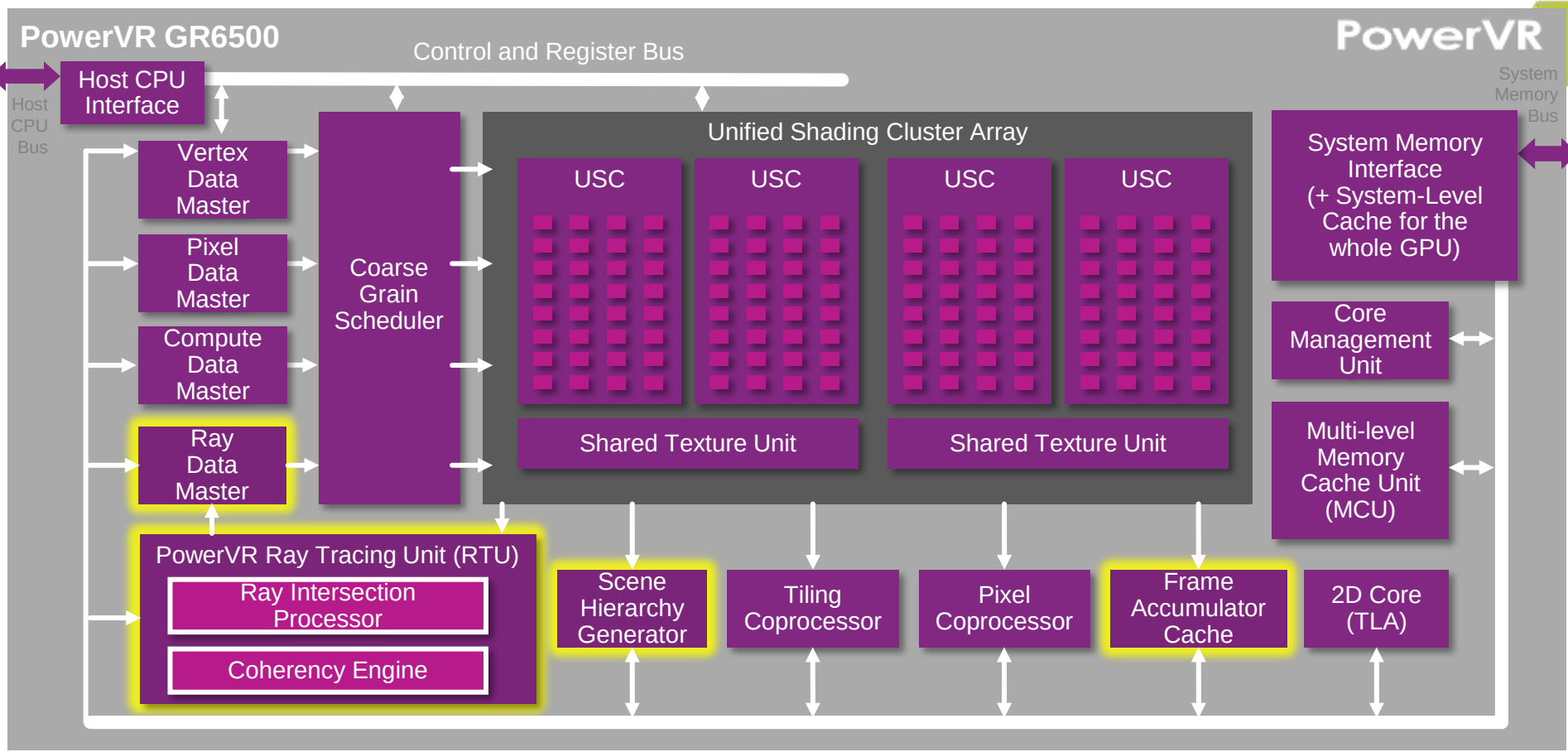
Ray Tracing on the Wizard GPU

Luke Peterson
Tobias Hector

PowerVR GPU Architecture



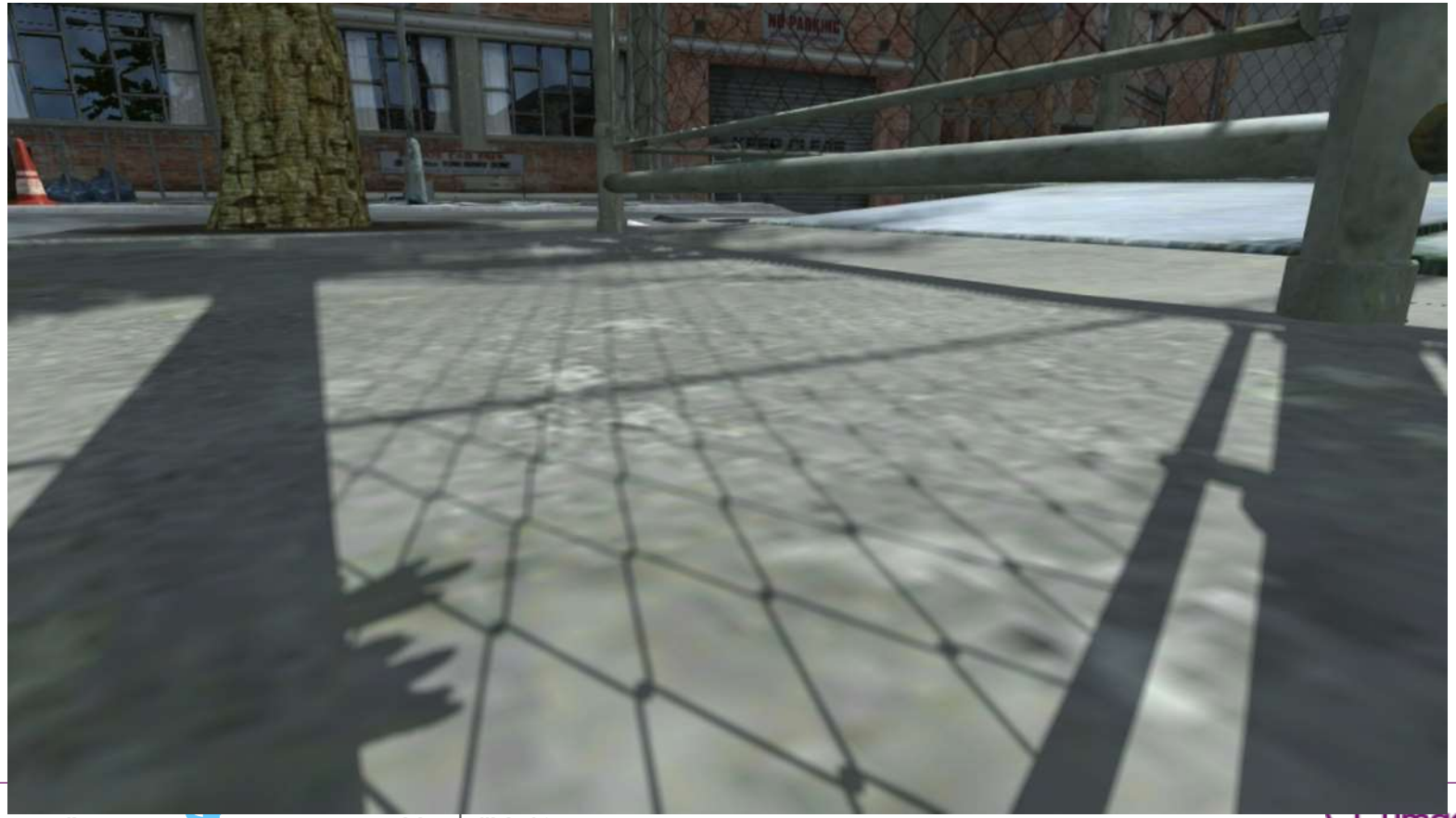
PowerVR Ray Tracing Wizard Architecture



Ray tracing is the ability for the shader program for one object to be aware of the geometry of other objects and trigger additional shaders for those objects.

Uses of Ray Tracing

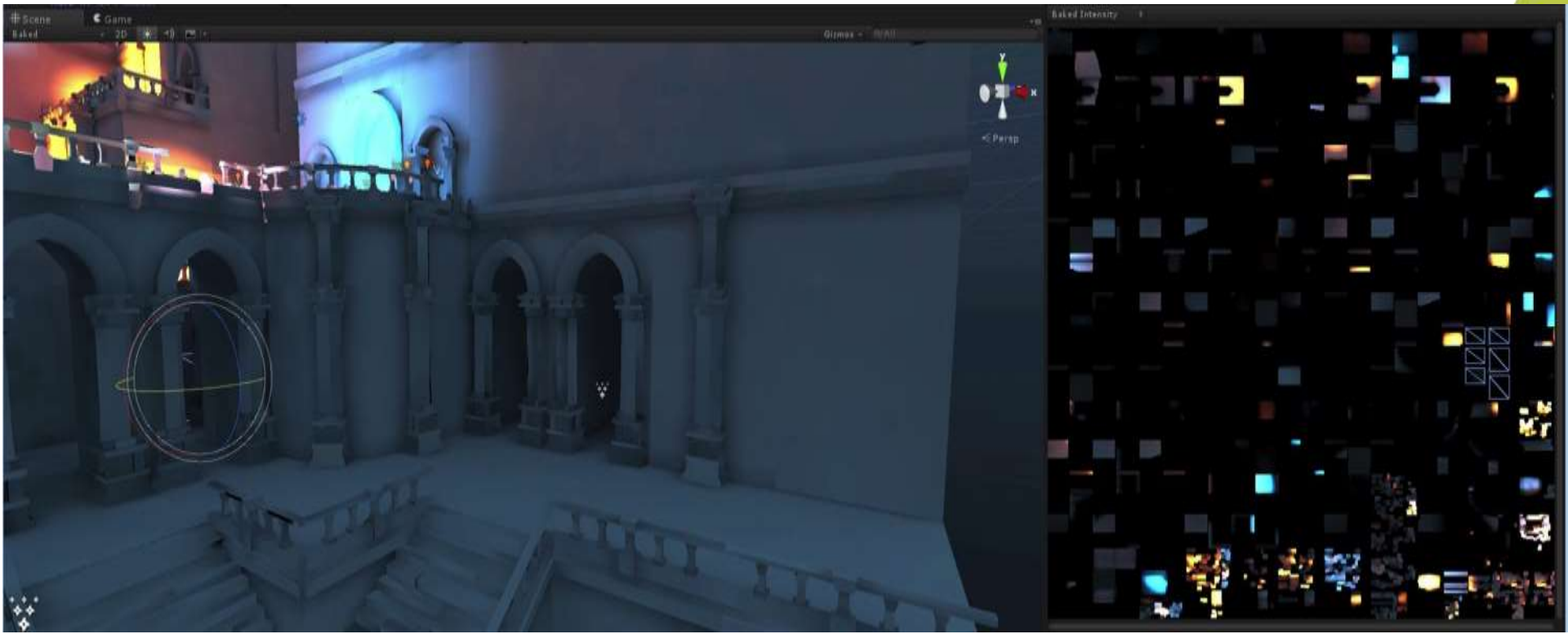
Uses of Ray Tracing



Uses of Ray Tracing

Uses of Ray Tracing

Uses of Ray Tracing



Uses of Ray Tracing

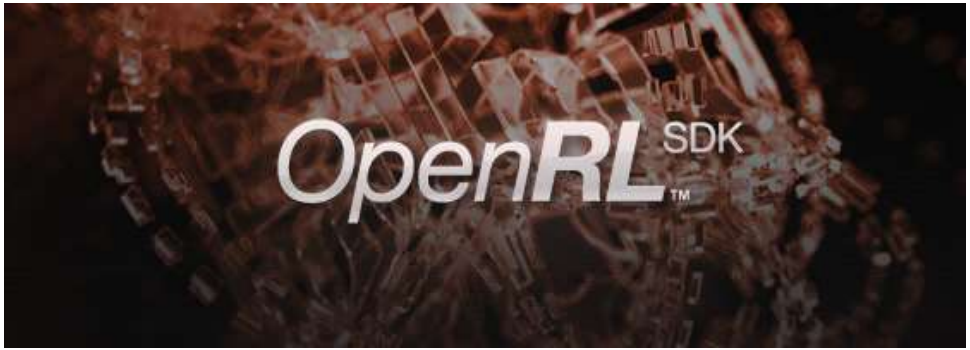
Uses of Ray Tracing

New Ray Tracing API!

New Ray Tracing API

Goodbye OpenRL

- **We had OpenRL**
  - Solid Ray Tracing API



New Ray Tracing API

Goodbye OpenRL

- **We had OpenRL**
  - Solid Ray Tracing API

- **However**
  - No integration with other GPU features
  - Required reworking to expose new features

New Ray Tracing API

Goodbye OpenRL

- **We had OpenRL**
  - Solid Ray Tracing API

- **However**
  - No integration with other GPU features
  - Required reworking to expose new features

- **So…**


OpenGL|ES™

Ray Tracing in OpenGL ES

Design Choices

- **Why OpenGL ES 3.1?**
  - Already has base features
  - Plugs in to existing SW solutions
  - Integrates with other GPU features
  - Well understood and available

Ray Tracing in OpenGL ES

Design Choices

- **Why OpenGL ES 3.1?**
  - Already has base features
  - Plugs in to existing SW solutions
  - Integrates with other GPU features
  - Well understood and available
- **Modern paradigms**
  - Direct State Access
  - App-managed resource lifetimes
  - Bindless resources

Ray Tracing in OpenGL ES

API Overview

- **Scene Construction**
  - Hardware accelerated building
  - Re-uses vertex shading
  - Output streamed into a hierarchy
    - Note: Vertex outputs are in **world space**

- **Ray Traversal**
  - Emit rays into a scene
  - Hardware accelerated traversal
  - Shade at intersections
  - Accumulate to images

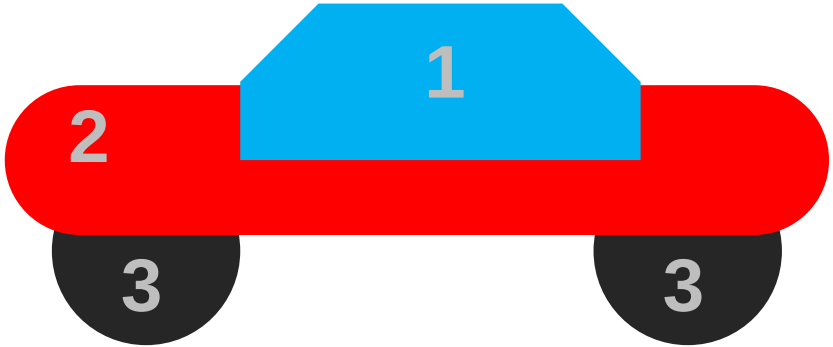
Scene Construction

Scene Construction

Scene Components and Component Groups

- **Scene components**
  - Equivalent to a “draw call”
  - May be multiple “objects”

- **Component groups**
  - Created from number of components
  - Represent a scene hierarchy



Scene Construction

Building and Merging

- **Building**
  - Components are descriptions
  - Building executes vertex shaders
  - Creates a scene hierarchy

- **Merging**
  - Merge component groups together
  - Cheaper than rebuilding
  - Used to handle different rates of change

Scene Construction

Building and Merging Illustration

- **Rebuild parts of a scene at different rates**

Scene Construction

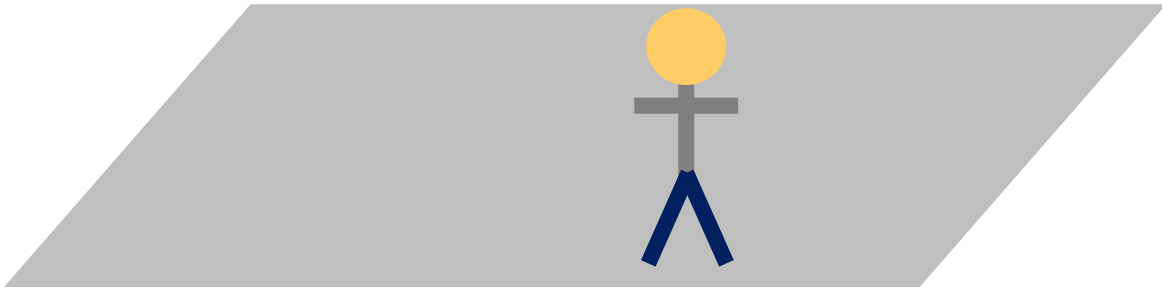
Building and Merging Illustration

- **Rebuild parts of a scene at different rates**
  - Static geometry

Scene Construction

Building and Merging Illustration

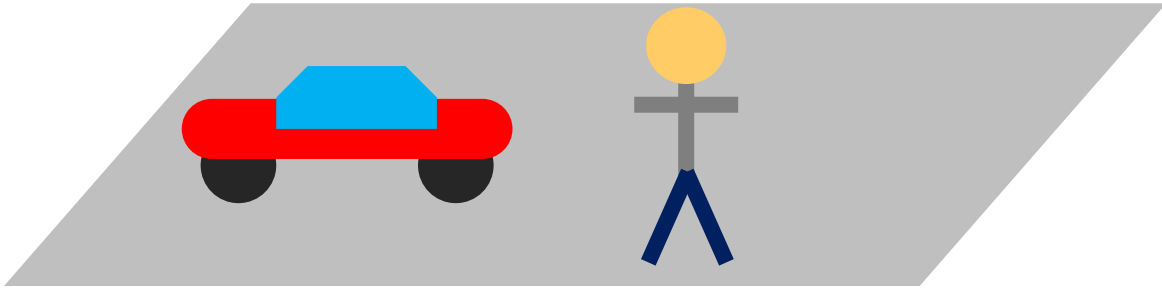
- **Rebuild parts of a scene at different rates**
  - Static geometry
  - Dynamic, per-frame



Scene Construction

Building and Merging Illustration

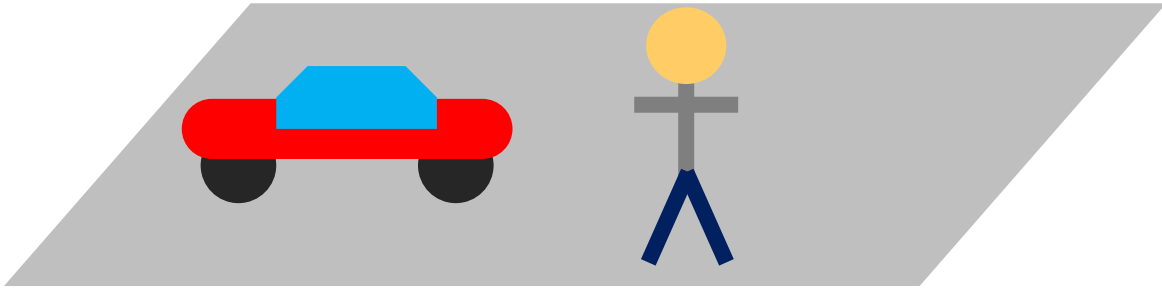
- **Rebuild parts of a scene at different rates**
  - Static geometry
  - Dynamic, per-frame
  - Other levels between the two



Scene Construction

Building and Merging Illustration

- **Rebuild parts of a scene at different rates**
  - Static geometry
  - Dynamic, per-frame
  - Other levels between the two

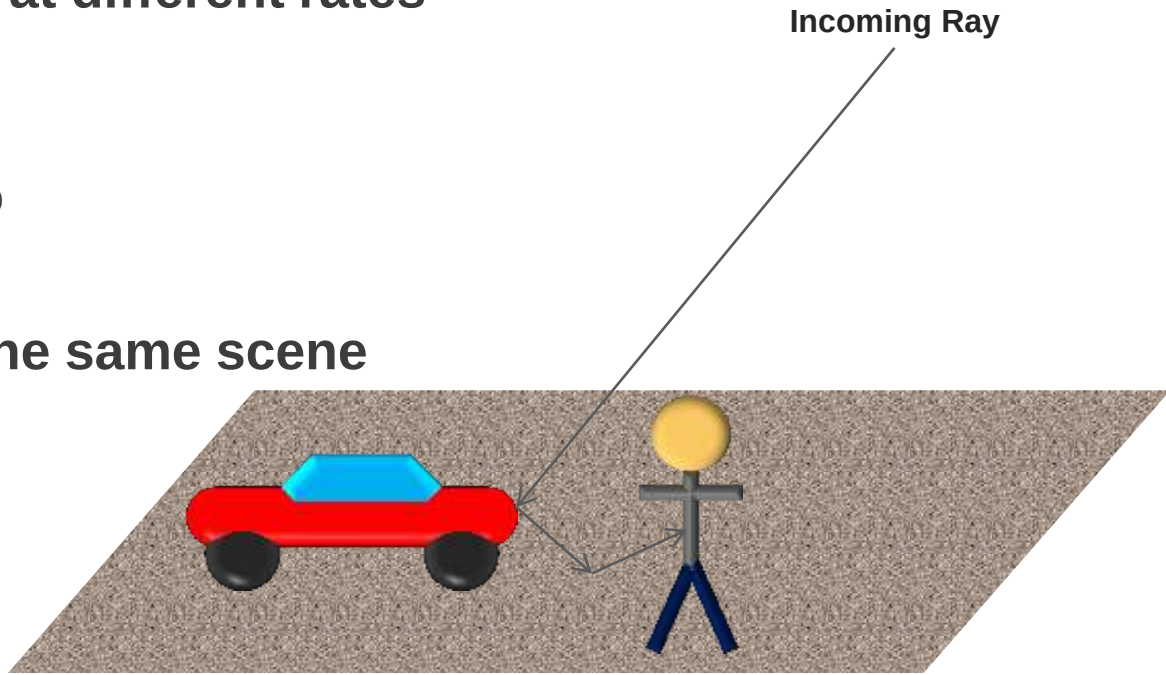


Scene Construction

Building and Merging Illustration

- **Rebuild parts of a scene at different rates**
  - Static geometry
  - Dynamic, per-frame
  - Other levels between the two

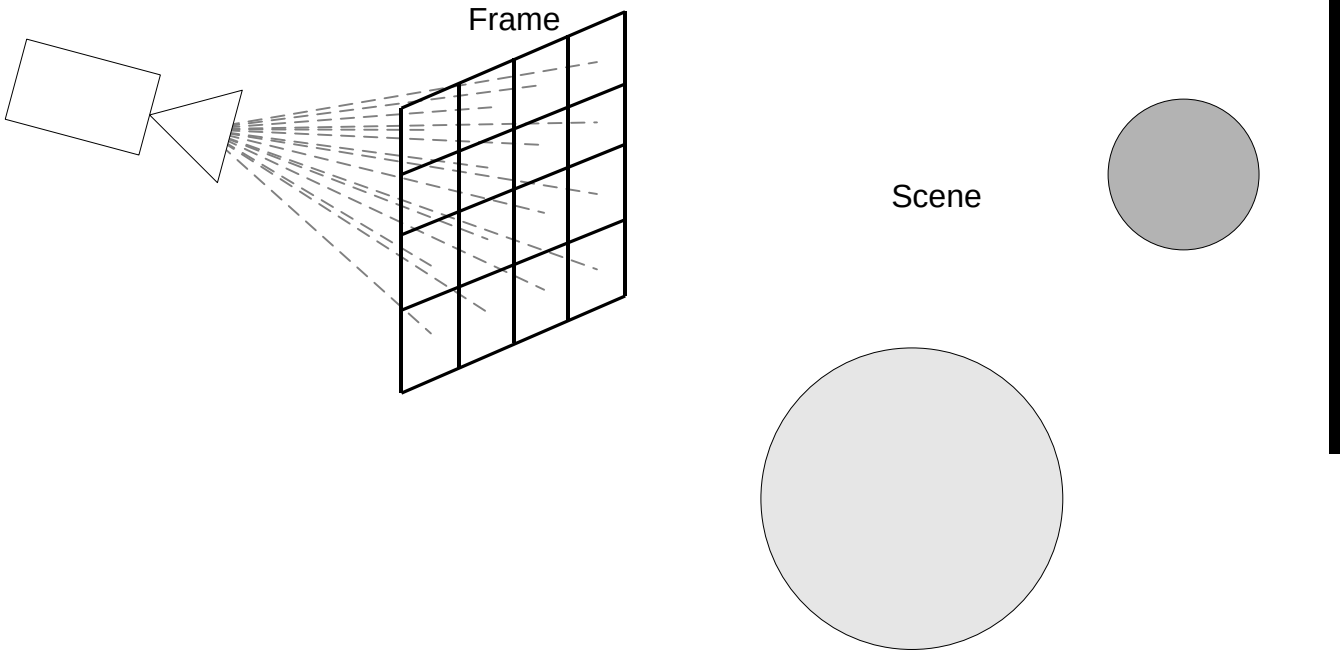
- **All traversed as part of the same scene**



Ray Traversal

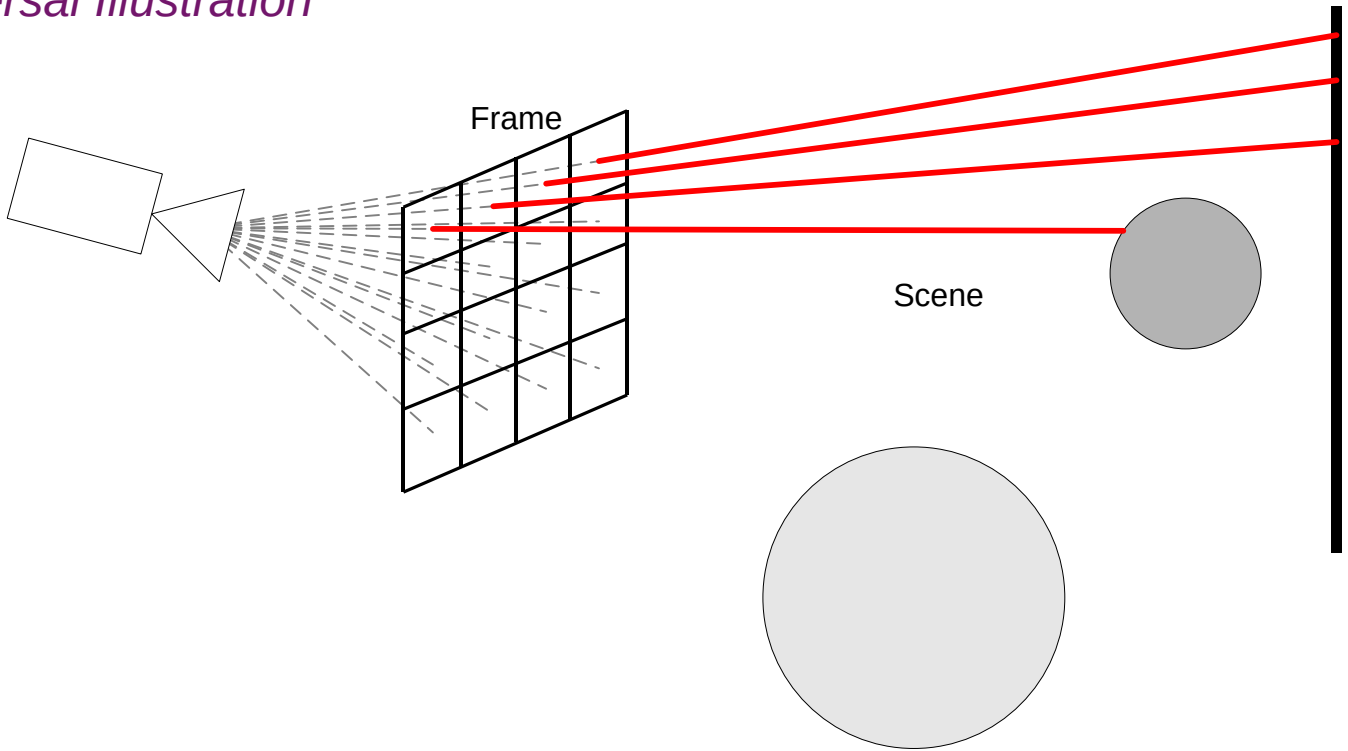
Ray Traversal

Ray Traversal Illustration



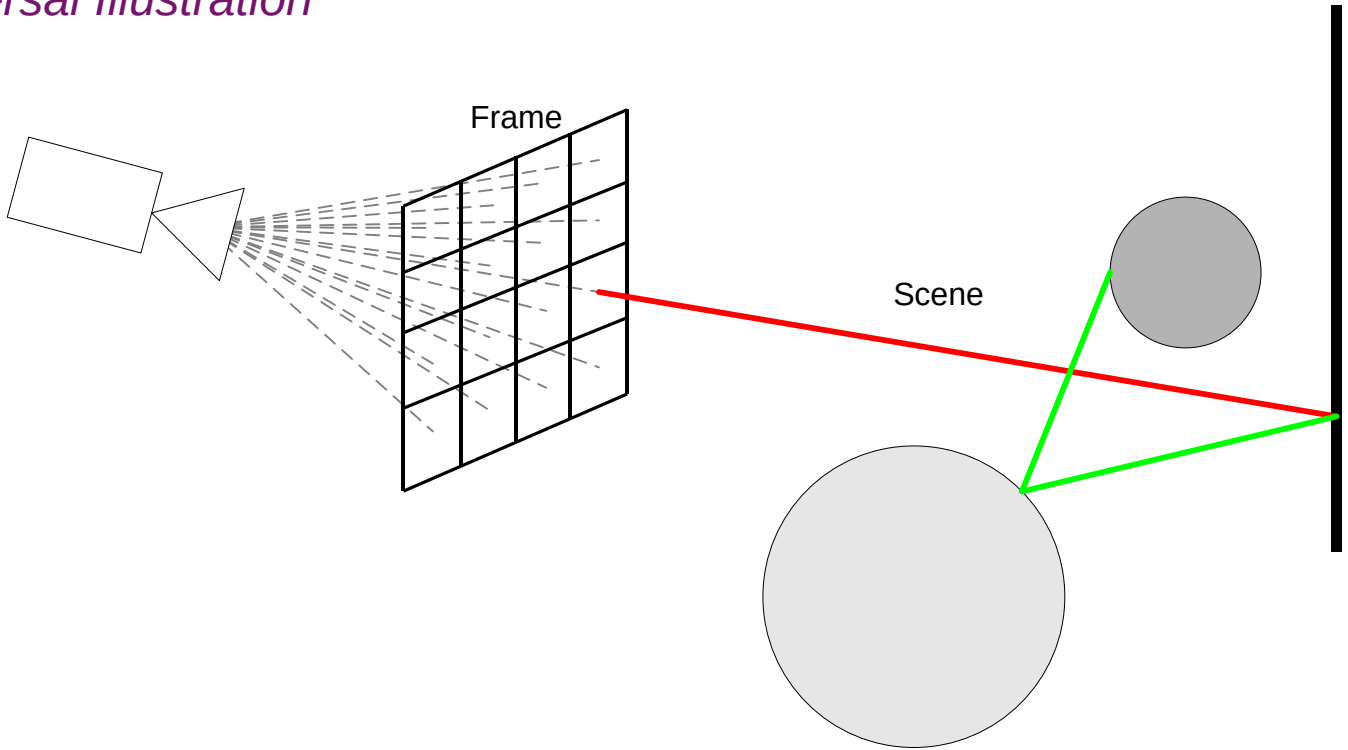
Ray Traversal

Ray Traversal Illustration



Ray Traversal

Ray Traversal Illustration



Ray Traversal

Ray Emission

- **What is a ray?**
  - A typed data packet
  - Built-in values describe traversal
  - May contain user-defined values

```
raytypeIMG {
    highp vec3 gl_OriginIMG,
    highp vec3 gl_DirectionIMG,
    highp float gl_MaxDistanceIMG,
    lowp uint gl_SceneIMG,
    bool gl_IsOutgoingIMG,
    bool gl_RunPrefixProgramIMG,
    highp rayprogramIMG gl_PrefixRayProgramIMG,
    bool gl_OcclusionTestIMG,
    mediump ivec2 gl_PixelIMG,
    mediump uint gl_BounceCountIMG,
    bool gl_FlipFacingIMG
};
```

Ray Traversal

Ray Emission

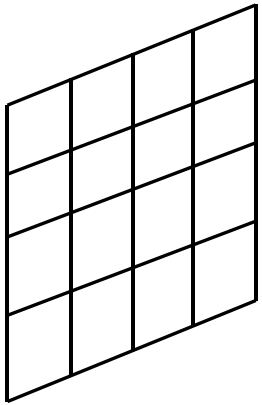
- **What is a ray?**
  - A typed data packet
  - Built-in values describe traversal
  - May contain user-defined values

```
raytypeIMG {
    highp vec3 gl_OriginIMG,
    highp vec3 gl_DirectionIMG,
    highp float gl_MaxDistanceIMG,
    lowp uint gl_SceneIMG,
    bool gl_IsOutgoingIMG,
    bool gl_RunPrefixProgramIMG,
    highp rayprogramIMG gl_PrefixRayProgramIMG,
    bool gl_OcclusionTestIMG,
    mediump ivec2 gl_PixelIMG,
    mediump uint gl_BounceCountIMG,
    bool gl_FlipFacingIMG
};
```

Ray Traversal

Frame Shader

- **What is it?**
  - Shader that runs at beginning of traversal
  - Runs a user-defined number of invocations
- **Purpose**
  - Setup and emission of primary rays for each invocation



Ray Traversal

Frame Shader Example

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;


void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    emitRayIMG(primary);
}
```

Ray Traversal

Frame Shader Example

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;



void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    emitRayIMG(primary);
}
```

Ray Traversal

Frame Shader Example

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;



void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    emitRayIMG(primary);
}
```

Ray Traversal

Frame Shader Example

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;



void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    emitRayIMG(primary);
}
```

Ray Traversal

Frame Shader Example

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;


void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    emitRayIMG(primary);
}
```

Ray Traversal

Frame Shader Example

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;



void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    emitRayIMG(primary);
}
```

Ray Traversal

Ray Shader

- **What is it for?**
  - Shading at ray intersections
  - Executes when a ray hits geometry
  - Similar to a fragment shader

- **Capabilities**
  - Access incoming rays
  - Access intersection result, including vertices
  - Initialise and emit rays
  - Accumulate to images

Ray Traversal

Ray Shader Example

```
#version 310 es

layout(binding = 0) raytypeIMG PrimaryRay {};

layout(location = 0, binding = 0, rgba8) uniform lowp accumulateonly image2D accumulationBuffer;

rayInputHandlerIMG(in PrimaryRay primary)
{
    void main() {
        imageAddIMG(accumulationBuffer, ivec2(primary.gl_PixelIMG), vec4(0.0, 0.5, 1.0, 1.0));
    }
}
```

Ray Traversal

Ray Shader Example

```
#version 310 es

layout(binding = 0) raytypeIMG PrimaryRay {};

layout(location = 0, binding = 0, rgba8) uniform lowp accumulateonly image2D accumulationBuffer;

rayInputHandlerIMG(in PrimaryRay primary)
{
    void main() {
        imageAddIMG(accumulationBuffer, ivec2(primary.gl_PixelIMG), vec4(0.0, 0.5, 1.0, 1.0));
    }
}
```

Ray Traversal

Ray Shader Example

```
#version 310 es

layout(binding = 0) raytypeIMG PrimaryRay {};

layout(location = 0, binding = 0, rgba8) uniform lowp accumulateonly image2D accumulationBuffer;

rayInputHandlerIMG(in PrimaryRay primary)
{
    void main() {
        imageAddIMG(accumulationBuffer, ivec2(primary.gl_PixelIMG), vec4(0.0, 0.5, 1.0, 1.0));
    }
}
```

Ray Traversal

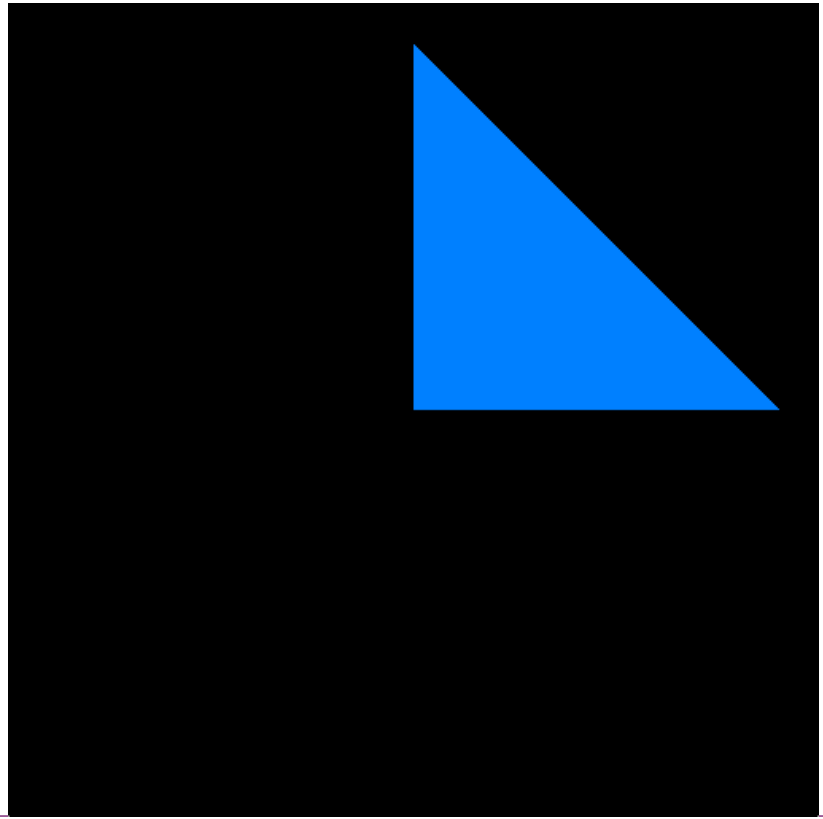
Ray Shader Example

```
#version 310 es

layout(binding = 0) raytypeIMG PrimaryRay {};

layout(location = 0, binding = 0, rgba8) uniform lowp accumulateonly image2D accumulationBuffer;

rayInputHandlerIMG(in PrimaryRay primary)
{
    void main() {
        imageAddIMG(accumulationBuffer, ivec2(primary.gl_PixelIMG), vec4(0.0, 0.5, 1.0, 1.0));
    }
}
```

Ray Traversal

Draw One Triangle

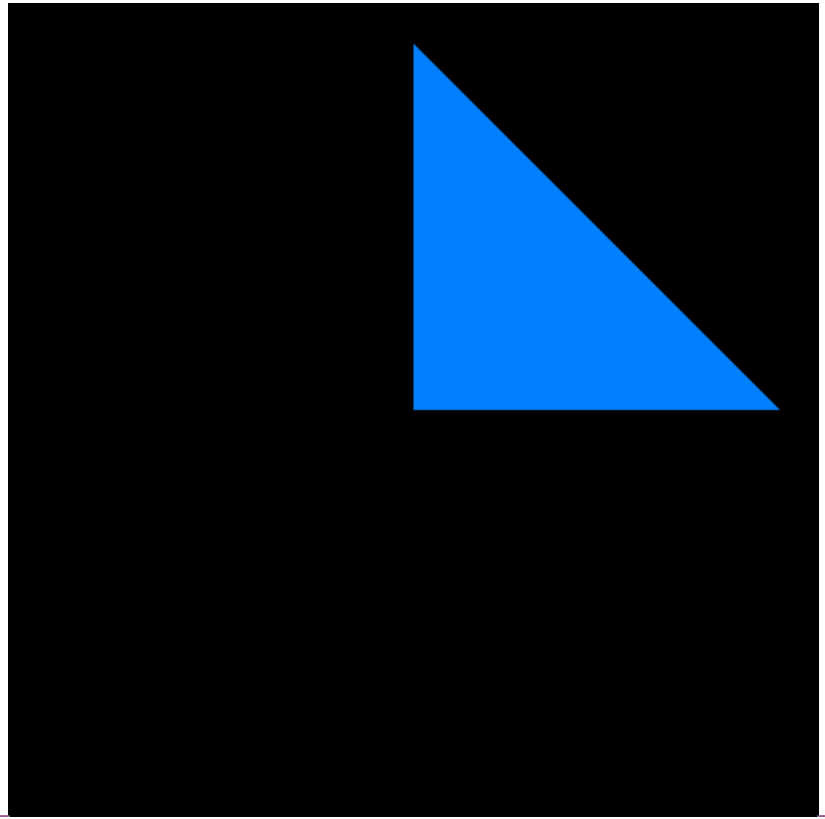
- 1st ray traced image today!



Ray Traversal

Draw One Triangle

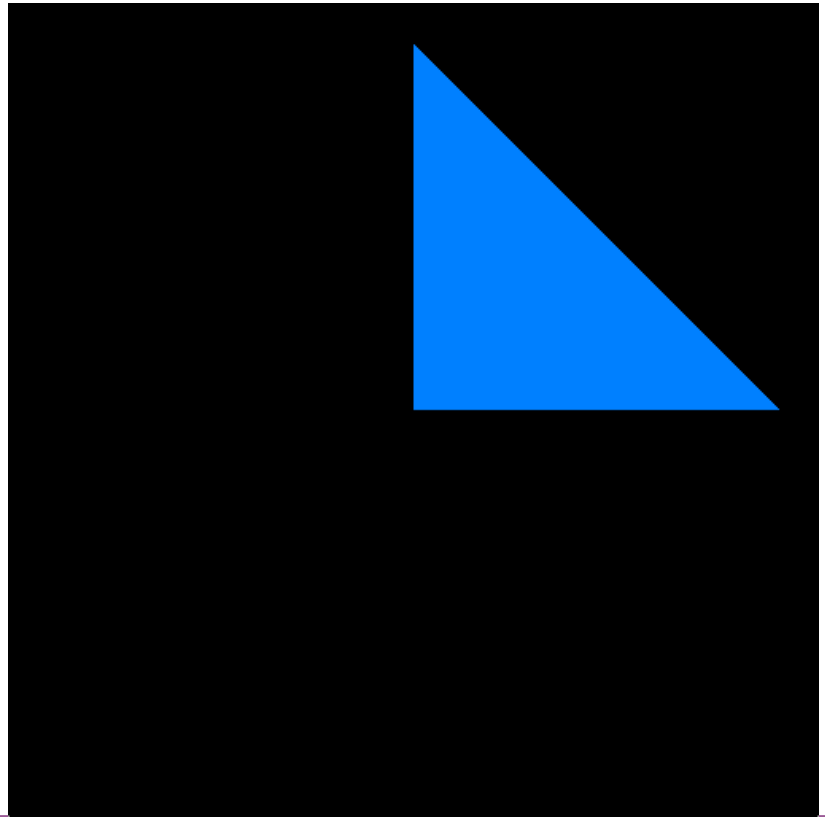
- **1st ray traced image today!**
  - It's a single triangle.



Ray Traversal

Draw One Triangle

- **1st ray traced image today!**
  - It's a single triangle.
  - Woo.

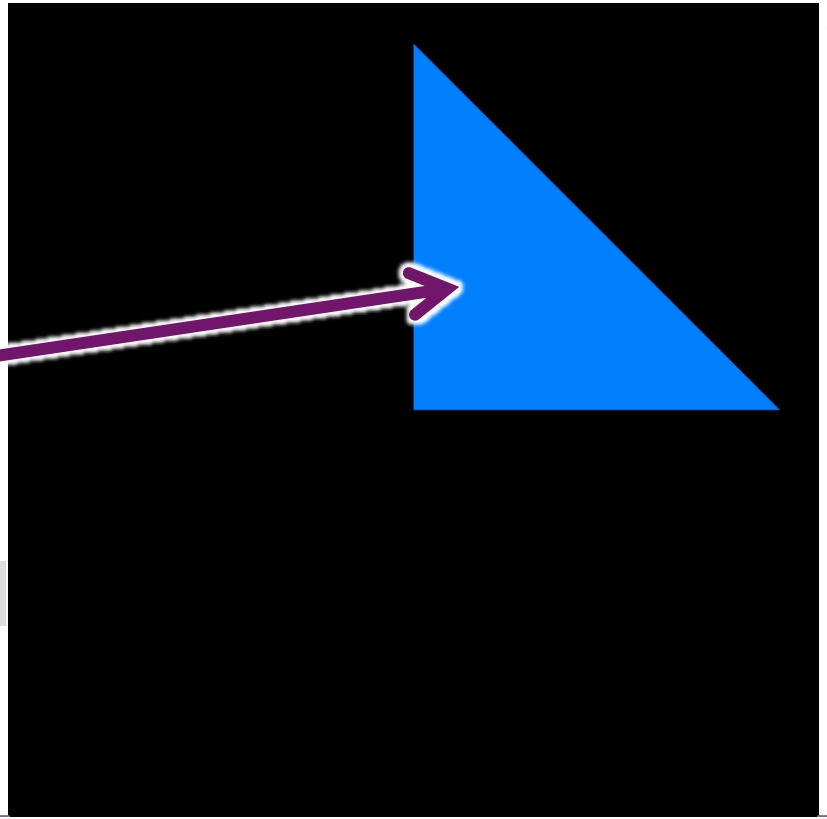


Ray Traversal

Draw One Triangle

- **Draw a simple triangle**
  - No lighting
  - No complex geometry
- **Blue accumulated to image**
  - Only added once
  - This line of code:

```
imageAddIMG(accumulationBuffer, ivec2(primary.gl_PixelIMG),
vec4(0.0, 0.5, 1.0, 1.0));
```



Ray Traversal

Bindless Ray Shaders

- **Bindless Ray Programs**
  - Semi-arbitrary shader execution
  - Separate from ray intersection shading

- **Separate to component ray shaders**
  - Not associated with geometry
  - Specialised uses
  - Has directly associated resources

Ray Traversal

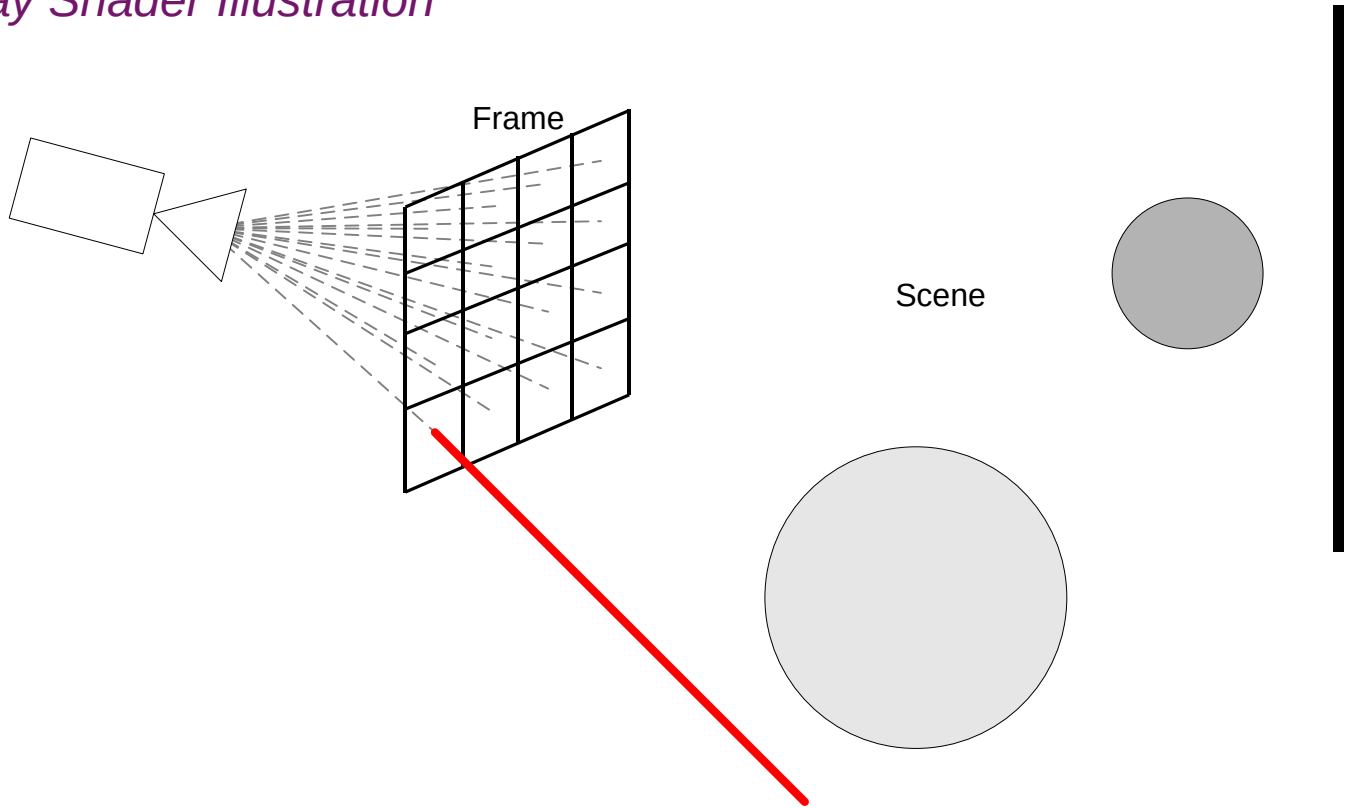
Bindless Ray Shader Creation

```
// Attach a ray program to the program
glAttachShader(myRayProgram, myRayShader);

// Set the program to separable
glProgramParameter(myRayProgram, GL_PROGRAM_SEPARABLE, GL_TRUE);

// Link the program
glLinkProgram(myRayProgram);

// Bind some buffers to it
glRayProgramBufferRangeIMG(myRayProgram, GL_UNIFORM_BUFFER, 0, myUniformBuffer, 0, myUniformBufferSize);

// Create a bindless handle
GLuint64 myRayProgramHandle = glGetRayProgramHandleIMG(myRayProgram);
```

Default Ray Shaders

Default Ray Shaders

Default Ray Shader Illustration



Default Ray Shaders

Frame Shader Example: Setting a default shader

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;
layout(location = 1) uniform highp rayprogramIMG defaultRayShader;

void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    emitRayIMG(primary, defaultRayShader);
}
```

Default Ray Shaders

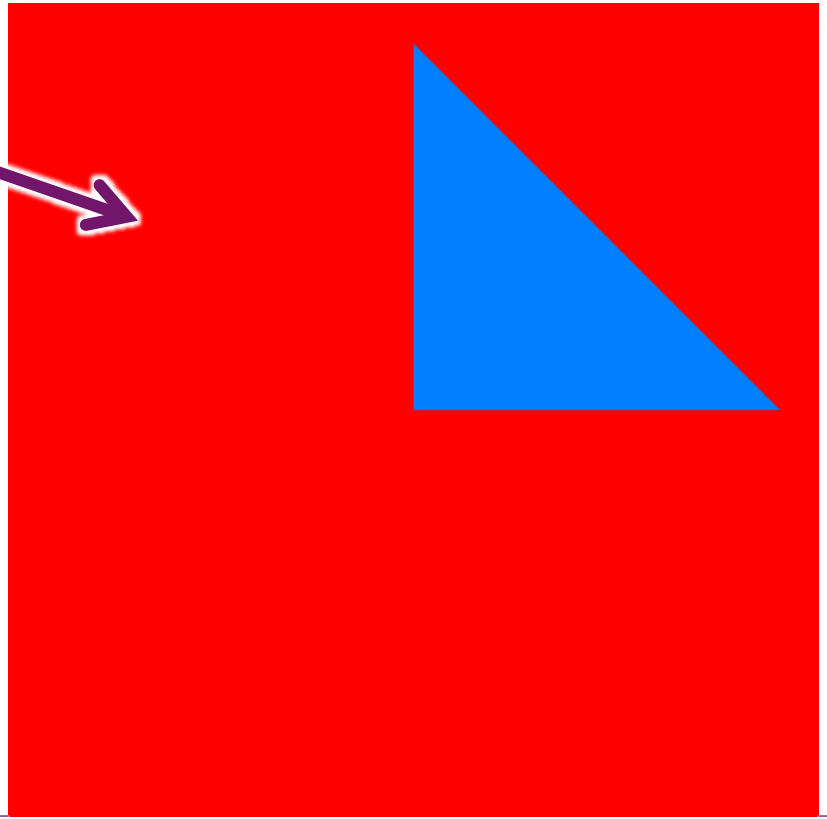
Ray Shader Example: Default

```
#version 310 es

layout(binding = 0) raytypeIMG PrimaryRay {};

layout(location = 0, binding = 0, rgba8) uniform lowp accumulateonly image2D accumulationBuffer;

rayInputHandlerIMG(in PrimaryRay primary)
{
    void main() {
        imageAddIMG(accumulationBuffer, ivec2(primary.gl_PixelIMG), vec4(1.0, 0.0, 0.0, 1.0));
    }
}
```

Default Ray Shaders

Draw One Triangle – Missed!

- **Accumulates red on miss**
  - Default shader runs on miss
  - Triangle not hit
- **Better uses:**
  - Skyboxes
  - World partitions
  - Simple light sources



Default Ray Shaders

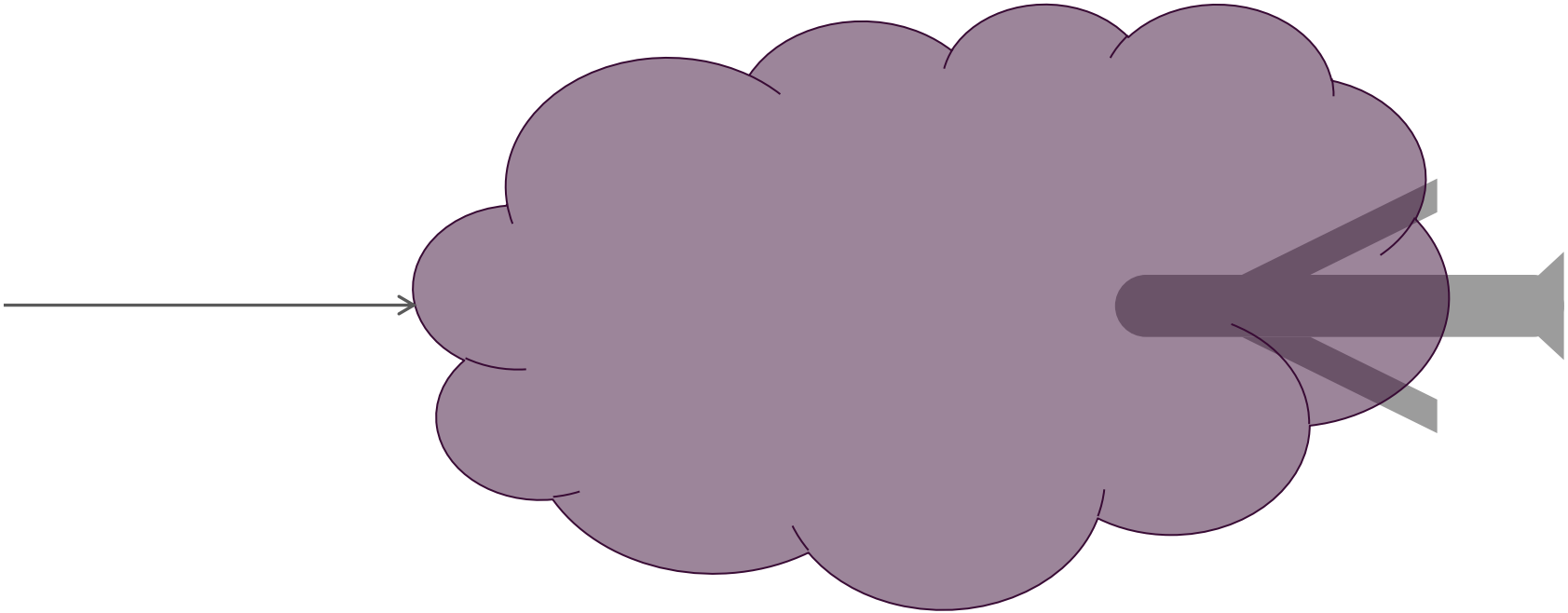
More Interesting things

- **Can obviously do lots more!**
  - Things like…

Prefix Ray Shaders

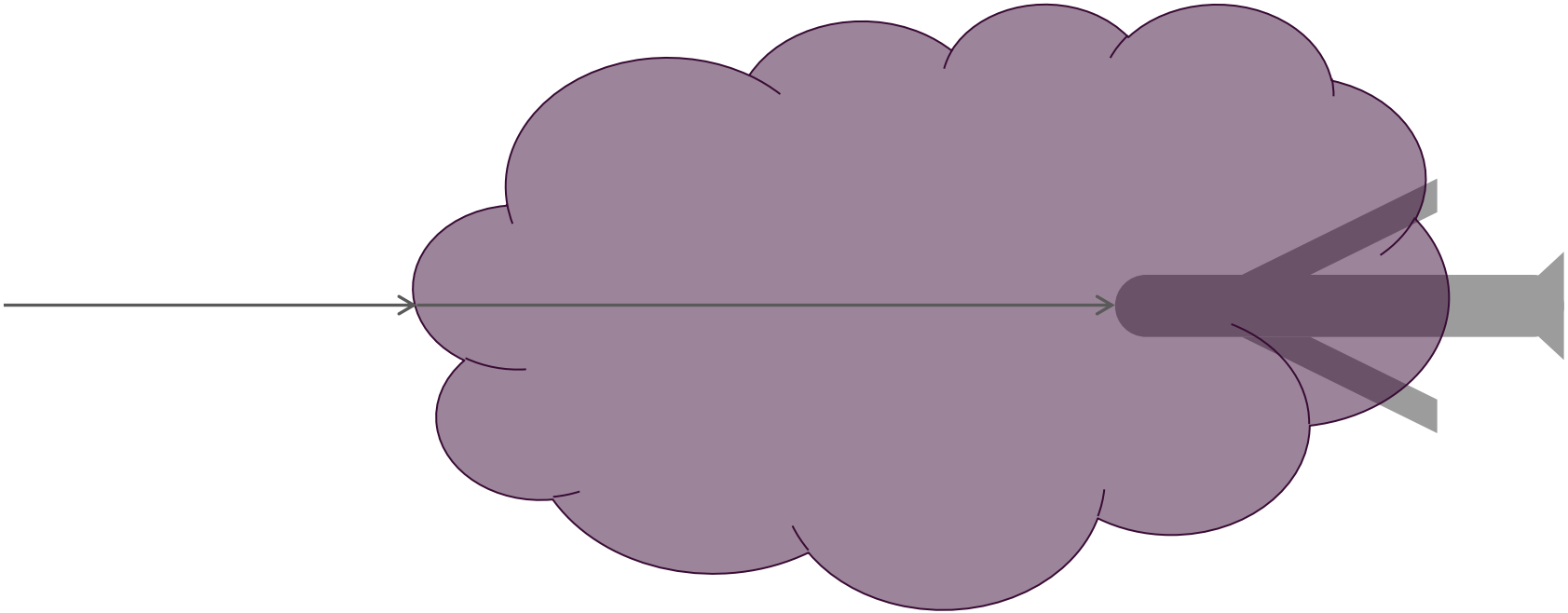
Prefix Ray Shaders

An illustration



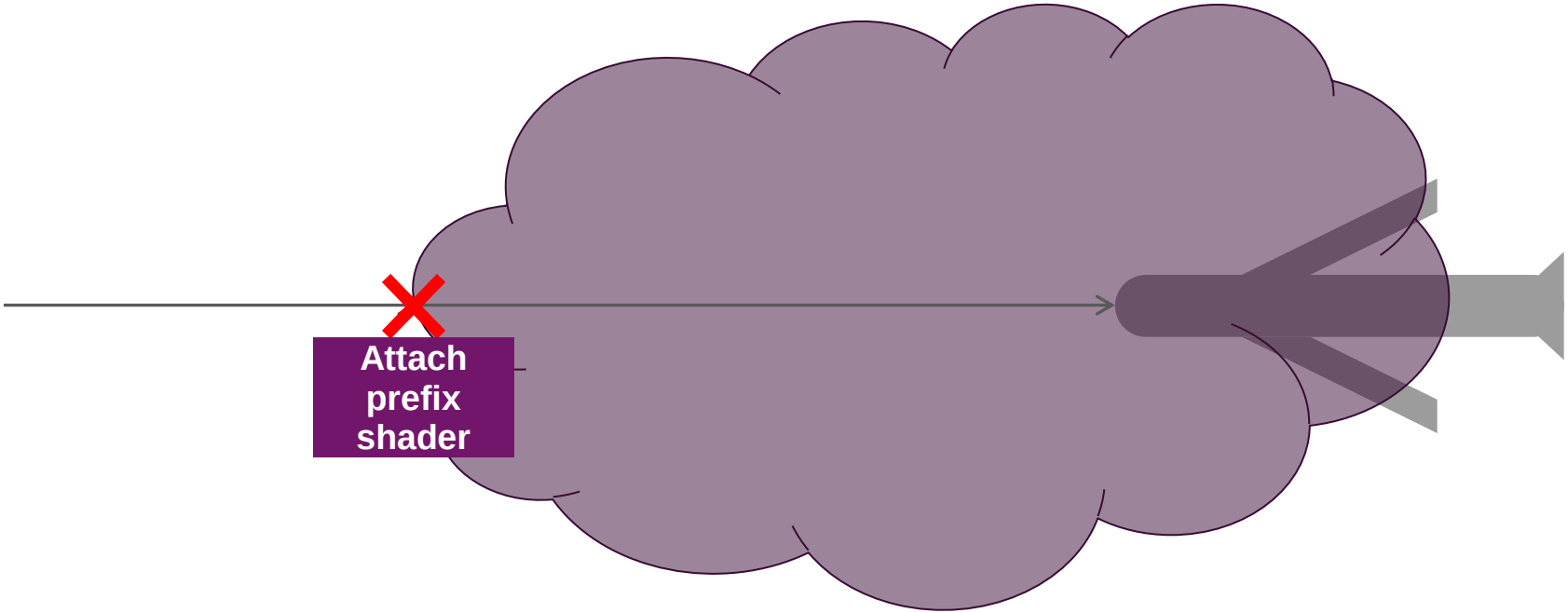
Prefix Ray Shaders

An illustration



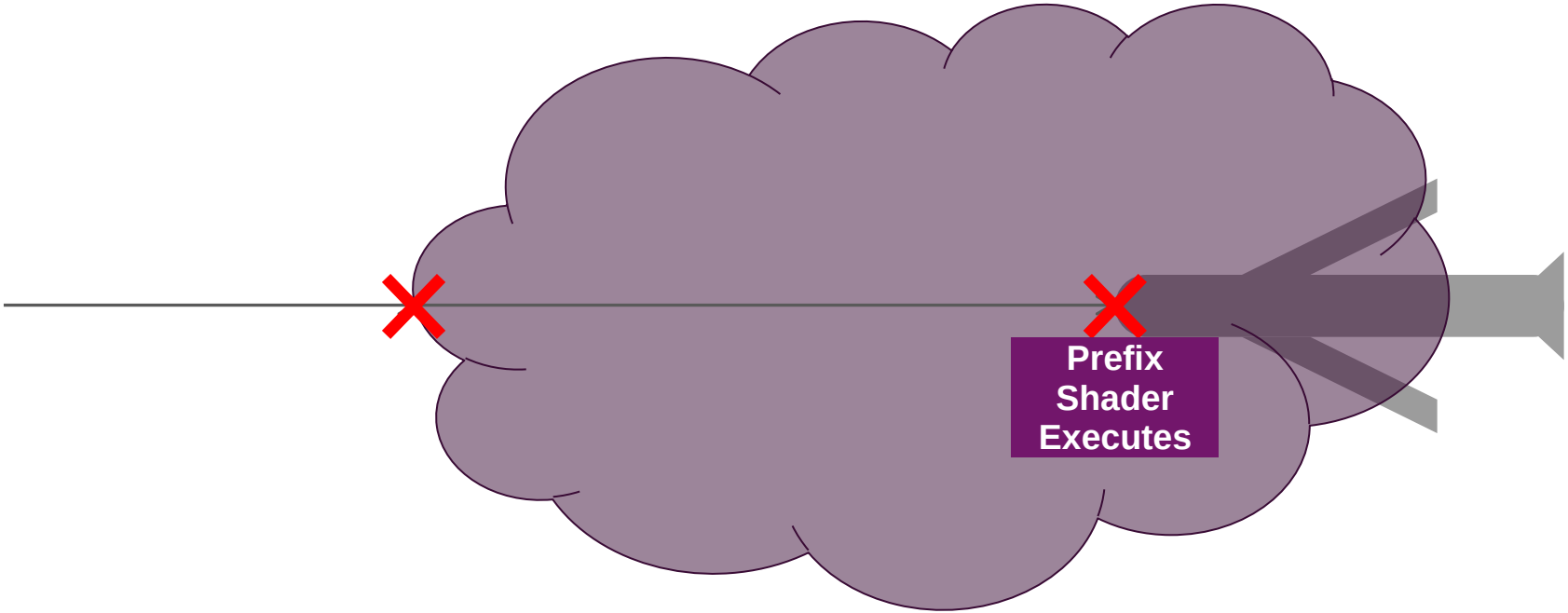
Prefix Ray Shaders

An illustration



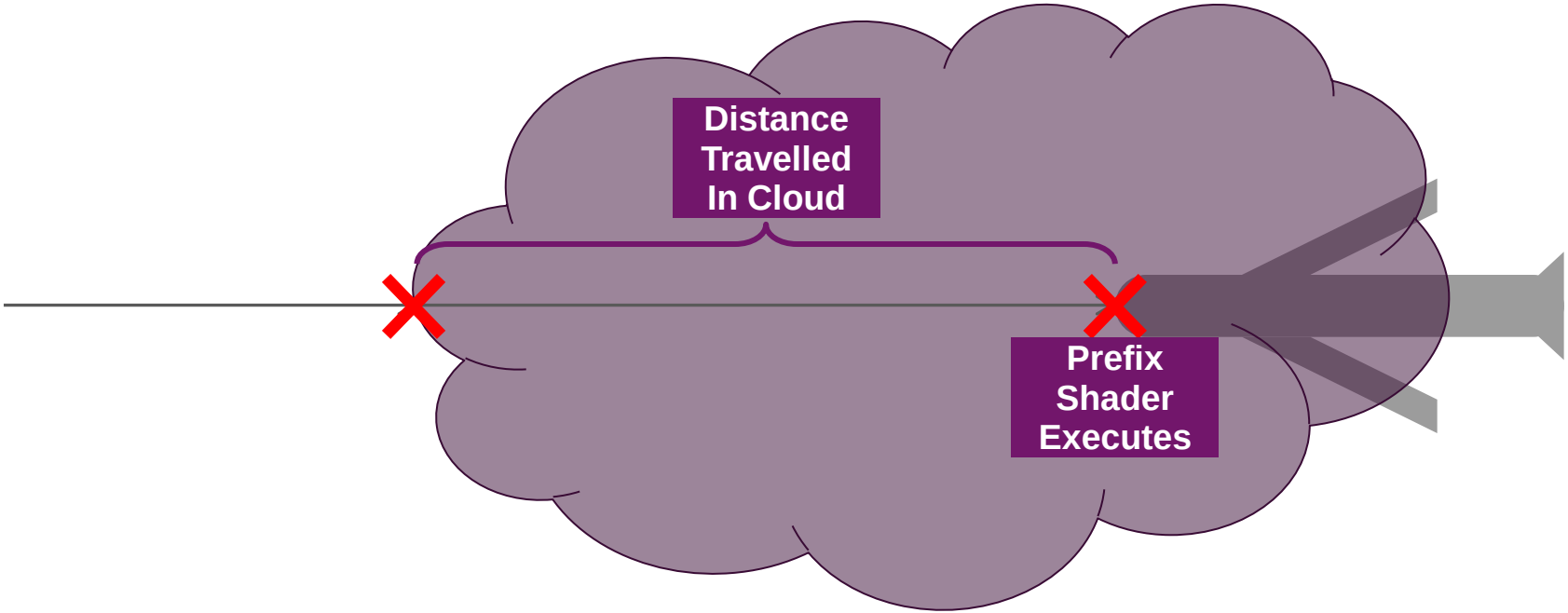
Prefix Ray Shaders

An illustration



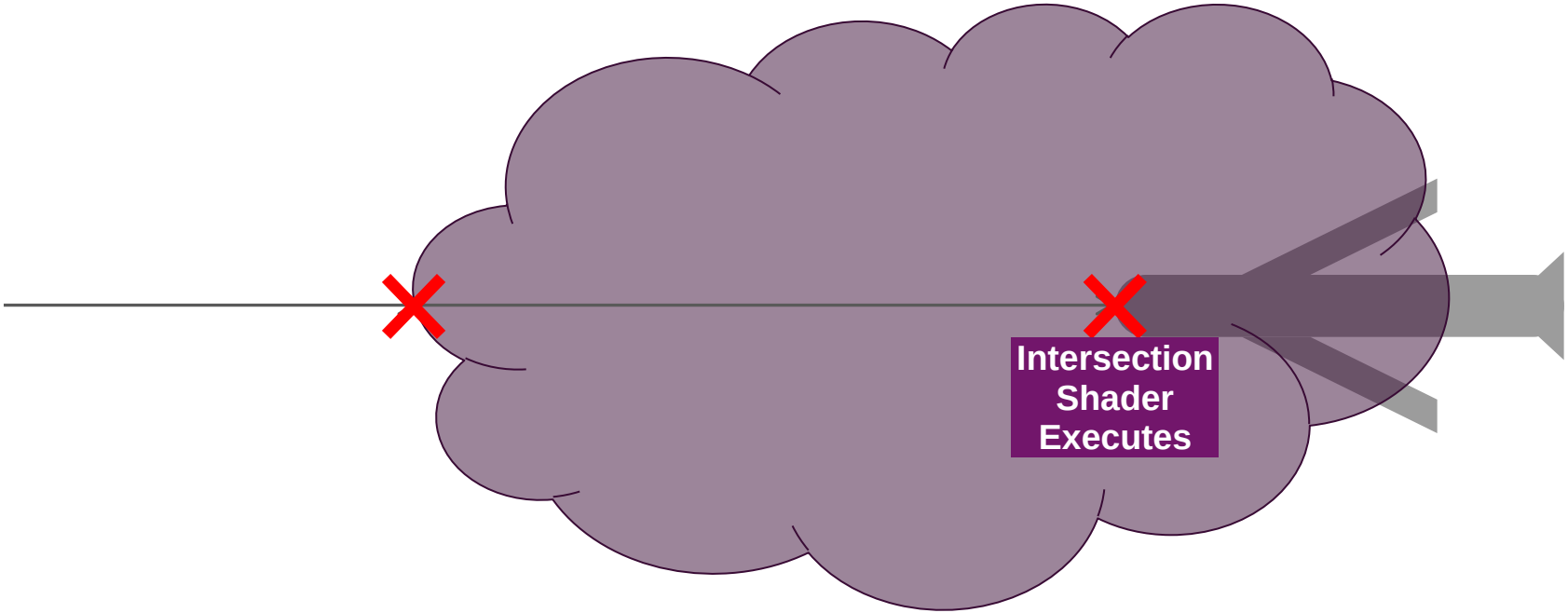
Prefix Ray Shaders

An illustration



Prefix Ray Shaders

An illustration



Prefix Ray Shaders

Frame Shader Example: Setting a prefix shader

```
#version 310 es
layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) out;
out PrimaryRay primary;
layout(location = 0) uniform highp vec2 inverseFrameSize;
layout(location = 1) uniform highp rayprogramIMG prefixRayShader;

void main() {
    initRayIMG(primary);
    primary.gl_OriginIMG = vec3(0.0);
    primary.gl_DirectionIMG = vec3((vec2(gl_FrameCoordIMG) * inverseFrameSize - 0.5).xy, -1.0);
    primary.gl_PixelIMG = gl_FrameCoordIMG;
    primary.gl_PrefixRayProgramIMG = prefixRayShader;
    primary.gl_RunPrefixRayProgramIMG = true;
    emitRayIMG(primary);
}
```

Prefix Ray Shaders

Prefix Ray Shader Example

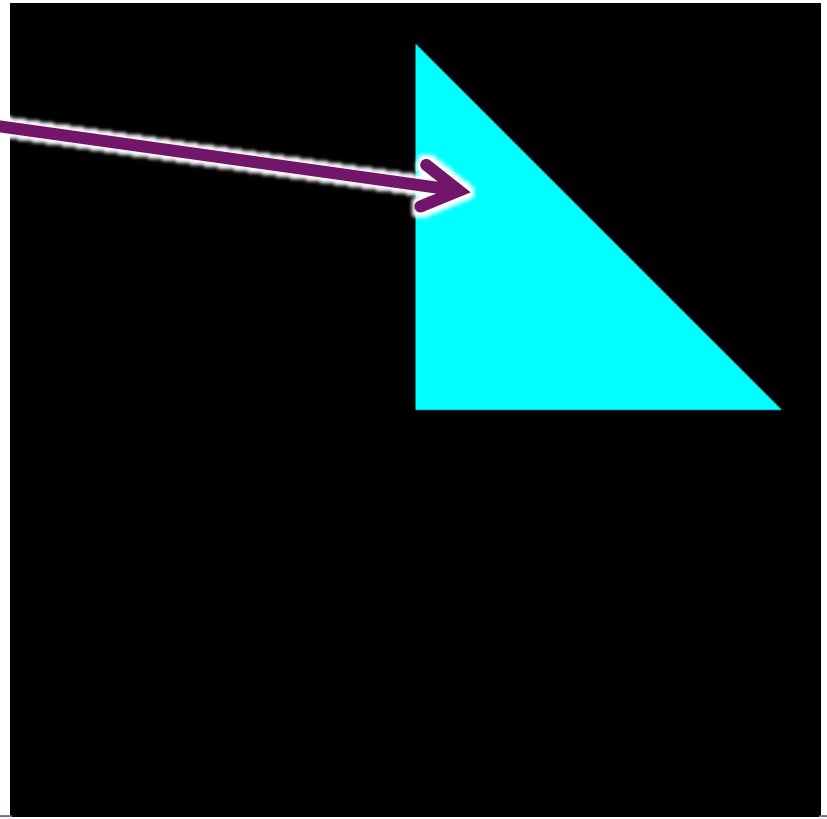
```
#version 310 es

layout(binding = 0) raytypeIMG PrimaryRay {};
layout(max_rays = 1) PrimaryRay;
layout(location = 0, binding = 0, rgba8) uniform lowp accumulateonly image2D accumulationBuffer;

rayInputHandlerIMG(inout PrimaryRay primary)
{
    void main() {
        imageAddIMG(accumulationBuffer, ivec2(primary.gl_PixelIMG), vec4(0.0, 0.5, 0.0, 0.0));
        primary.gl_RunPrefixProgramIMG = false;
        shadeRayIMG(primary);
    }
}
```

Prefix Ray Shaders

Draw One Triangle – Prefixed!

- **Accumulated cyan**
  - Still running hit shader
  - 0.5 Green added in prefix
  - 0.5 Green and 1 blue added in hit
  - Results in Cyan!

- **Better uses:**
  - Clouds, smoke, water
  - Other non-trivial mediums

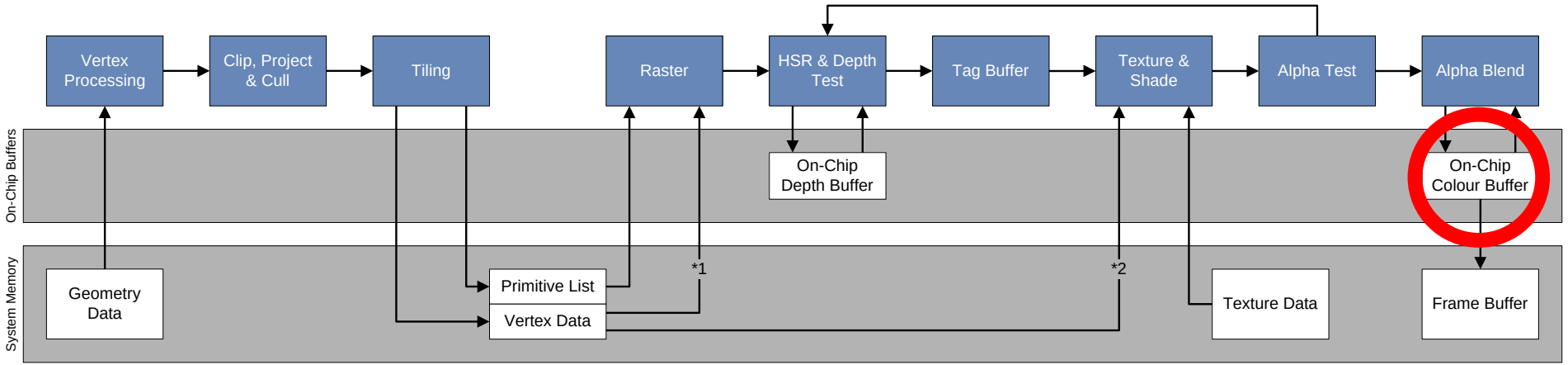


Hybrid Ray Shading

Hybrid Ray Shading

Our GPU

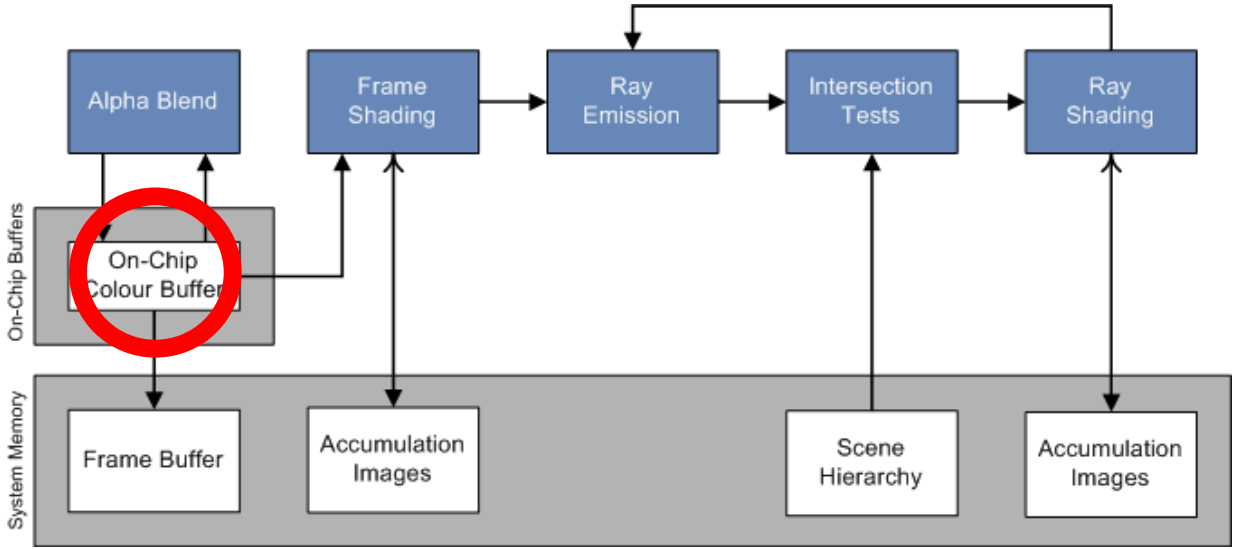
- **TBDR Pipeline Recap**
  - Ignore most of it for now
  - On-chip colour buffer!



Hybrid Ray Shading

Hybrid Pipeline

- **Frame shader can access on-chip colour buffer**
  - If run at the end of a render
  - Access PLS/Framebuffer Fetch as inputs only



Hybrid Ray Shading

How do you use hybrid?

- **Rasterizer generates initial data**
  - Writes out G-Buffer into Pixel Local Storage
- **Ray tracing for additional effects**
  - Shadows
  - Reflections
  - Etc.
- **Post-process as normal**
  - Use accumulation buffers as input

That’s not everything

We just scratched the surface

- **This was just an overview!**
  - Much more to talk about

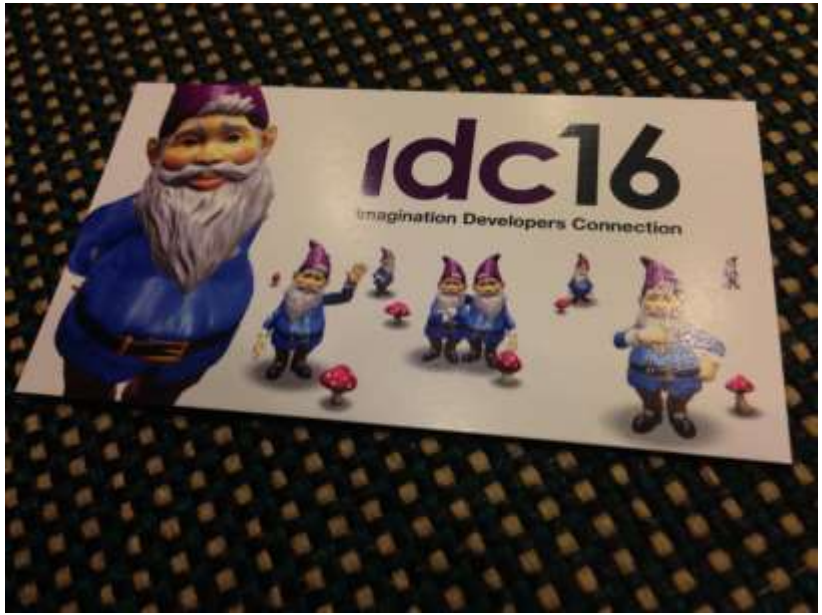
- **More talks!**
  - 11:50-12:20
    - Real Time Ray Tracing Techniques
  - 12:20-12:50
    - Global Illumination, Progressive Light Map Baking, Fully Interactive GI
  - 12:50-1:20
    - Advanced Ray Tracing Techniques

Questions?

Get in touch:

- *Luke.Peterson@imgtec.com*, *Tobias.Hector@imgtec.com*
- *@tobskihectov*

Please come and visit us at booth #1902 in the South Hall to see our demos and to collect your very own Vulkan™ Gnome t-shirt!




idc16
Imagination Developers Connection