

Integrating Vulkan Ray Tracing into the Godot Engine

Antonio Caggiano, LunarG



Who am I?

I'm Antonio!

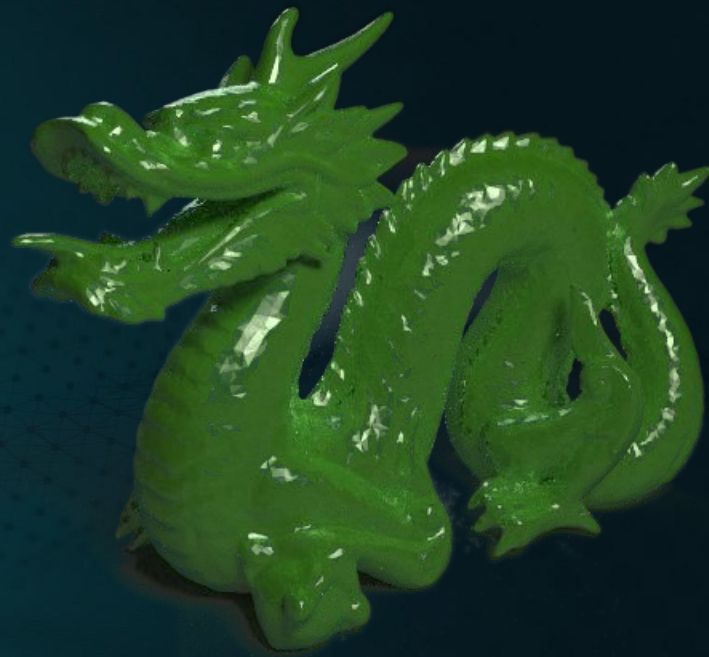
- Based in Italy
- ~8 years as a graphics engineer
- Been working at LunarG for 1 year
- I play the piano
- 42 is my favourite number



I like Computer Graphics

and learning new stuff!

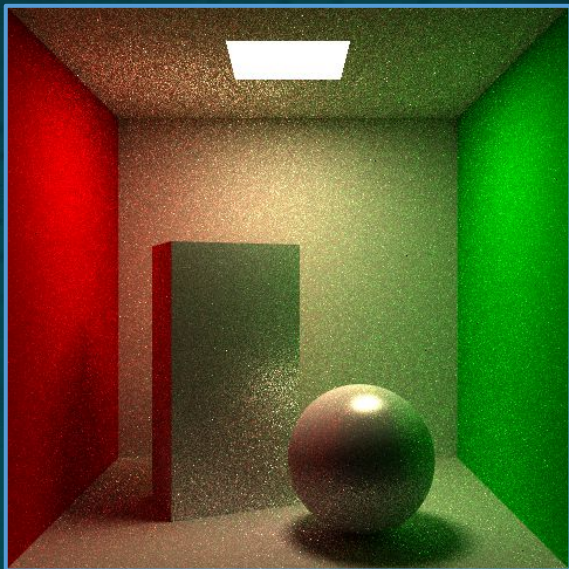
- Software renderer
- Ray tracing
- Path tracer
- Importance sampling
- Multiple importance sampling



Why Ray Tracing?

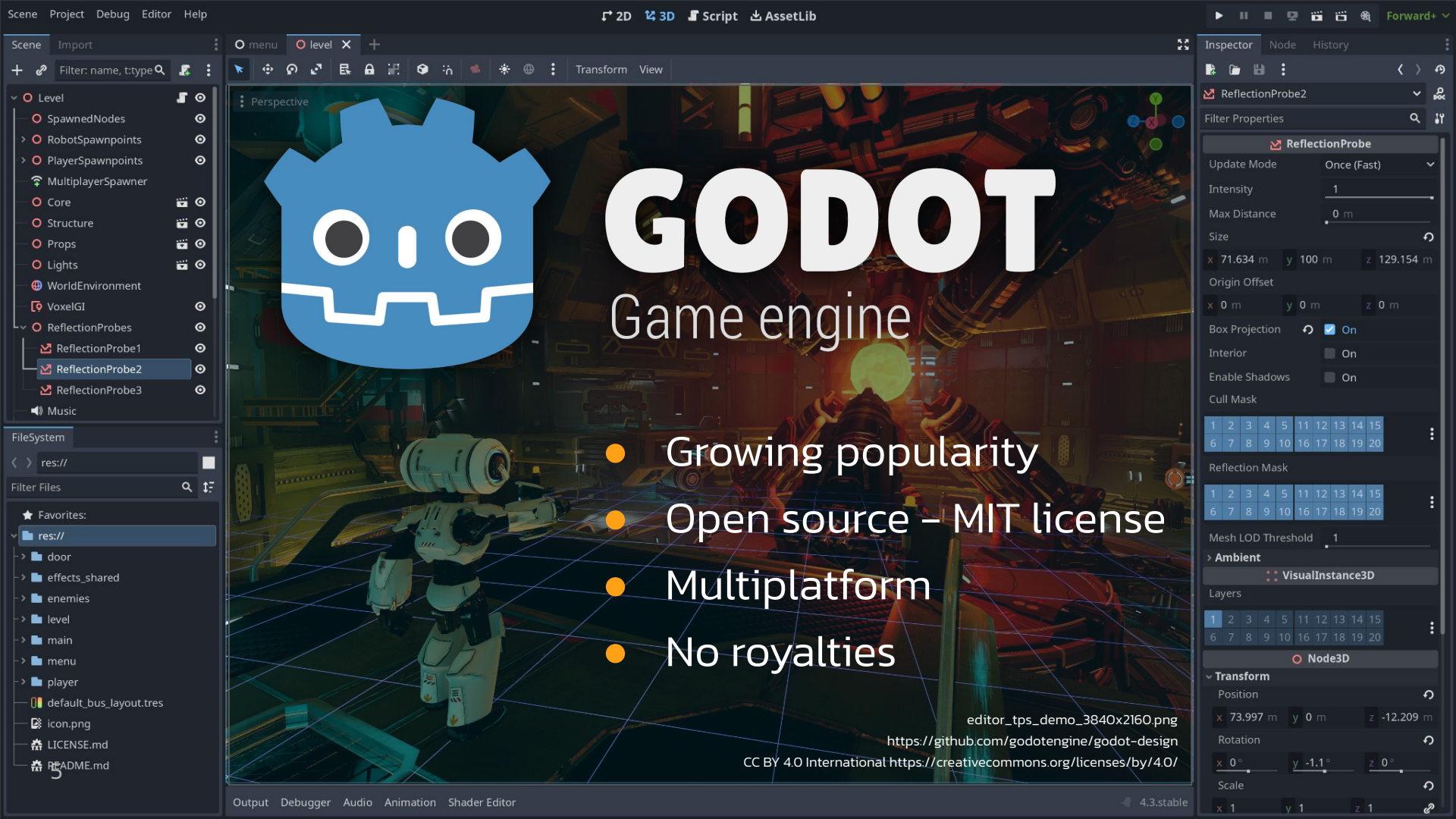
I can render awesome pictures!

Shadows and GI



Reflections, refractions





GODOT

Game engine

- Growing popularity
- Open source – MIT license
- Multiplatform
- No royalties

editor_tps_demo_3840x2160.png
<https://github.com/godotengine/godot-design>
CC BY 4.0 International <https://creativecommons.org/licenses/by/4.0/>

Scene Import

Filter: name, ttype

- Level
 - SpawnedNodes
 - RobotSpawnpoints
 - PlayerSpawnpoints
 - MultiplayerSpawner
 - Core
 - Structure
 - Props
 - Lights
 - WorldEnvironment
 - VoxelGI
 - ReflectionProbes
 - ReflectionProbe1
 - ReflectionProbe2
 - ReflectionProbe3
 - Music

FileSystem

res://

Filter Files

Favorites:

- res://
 - door
 - effects_shared
 - enemies
 - level
 - main
 - menu
 - player
- default_bus_layout.tres
- icon.png
- LICENSE.md
- ADME.md

Inspector Node History

ReflectionProbe2

Filter Properties

ReflectionProbe

Update Mode: Once (Fast)

Intensity: 1

Max Distance: 0 m

Size: x 71.634 m y 100 m z 129.154 m

Origin Offset: x 0 m y 0 m z 0 m

Box Projection: On

Interior: On

Enable Shadows: On

Cull Mask: 1 2 3 4 5 11 12 13 14 15 6 7 8 9 10 16 17 18 19 20

Reflection Mask: 1 2 3 4 5 11 12 13 14 15 6 7 8 9 10 16 17 18 19 20

Mesh LOD Threshold: 1

Ambient: VisualInstance3D

Layers: 1 2 3 4 5 11 12 13 14 15 6 7 8 9 10 16 17 18 19 20

Node3D

Transform

Position: x 73.997 m y 0 m z -12.209 m

Rotation: x 0° y -1.1° z 0°

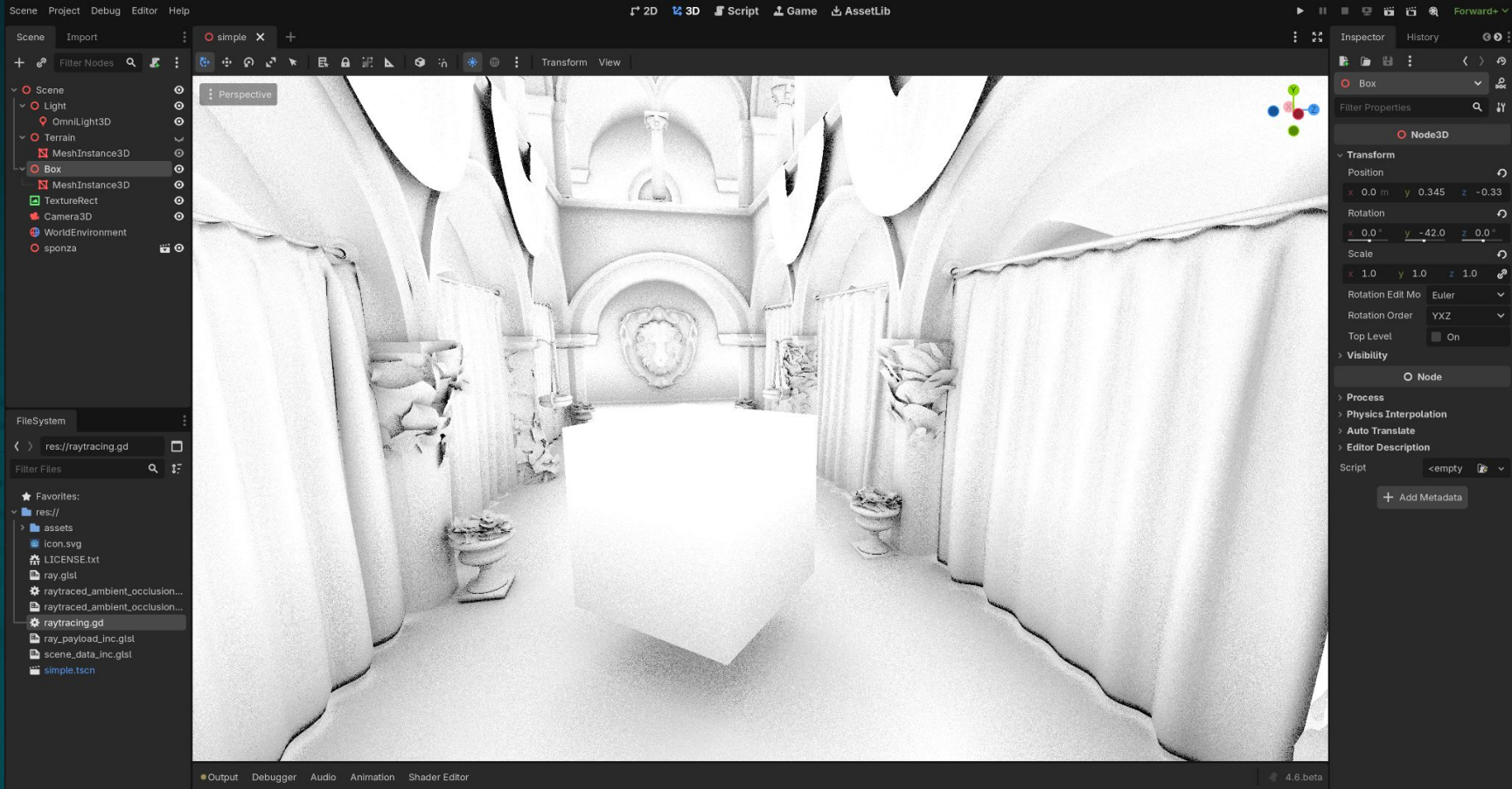
Scale: x 1 y 1 z 1

Output Debugger Audio Animation Shader Editor 4.3.stable

But no support for Hardware Ray Tracing



A-HA!



Adding Vulkan Ray Tracing to Godot

How hard could it be?



We need to write Vulkan code

No way!

1. Create Bounding Volume Hierarchy (BVH)
2. Create a Ray Tracing Pipeline
3. Create a Shader Binding Table (SBT)
4. Trace rays!

So Yeah, But

Without breaking anything!

- Fit Godot rendering abstractions
- Automatic synchronization via Render Graph
- Ray tracing API exposure via scripting
- Multi-backend friendly

What's the end goal?

Brace yourself, I am going to show
you some code!

GScript

Godot's Scripting Language

```
var rd = RenderingDevice.new()

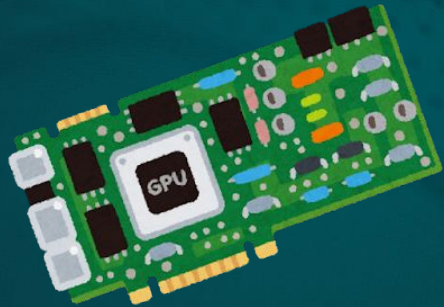
# Create BLAS and TLAS for a mesh.
blas = rd.blas_create(...)
tlas = rd.tlas_create(...)

# Bind pipeline and uniforms.
rd.raytracing_list_bind_raytracing_pipeline(raylist, raytracing_pipeline)
rd.raytracing_list_bind_uniform_set(raylist, uniform_set, 0)

# Trace rays.
rd.raytracing_list_trace_rays(raylist, width, height)
```

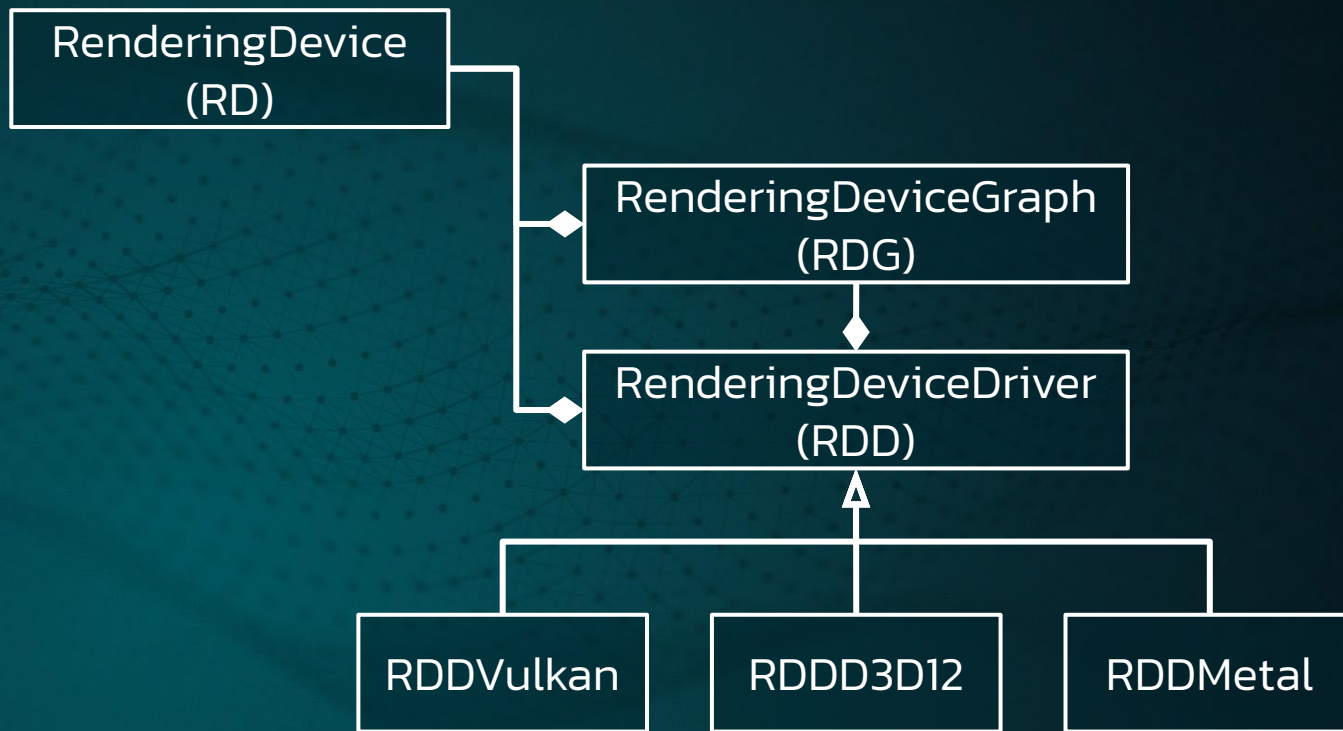
Where does GPU rendering live in the engine?

Godot HAL



Godot HAL

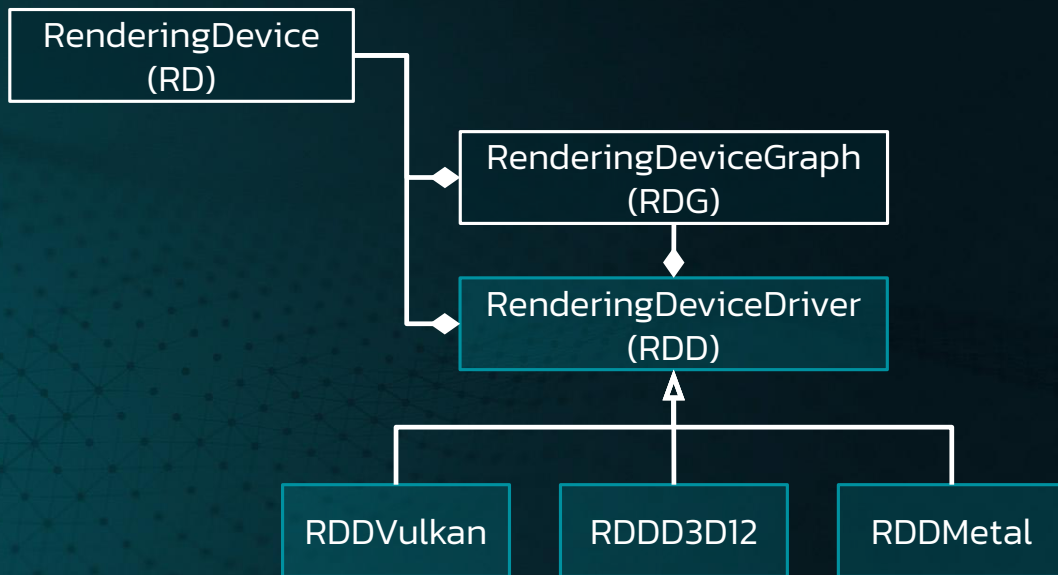
RD, RDG, RDD



Rendering Device Driver

Per-backend drivers

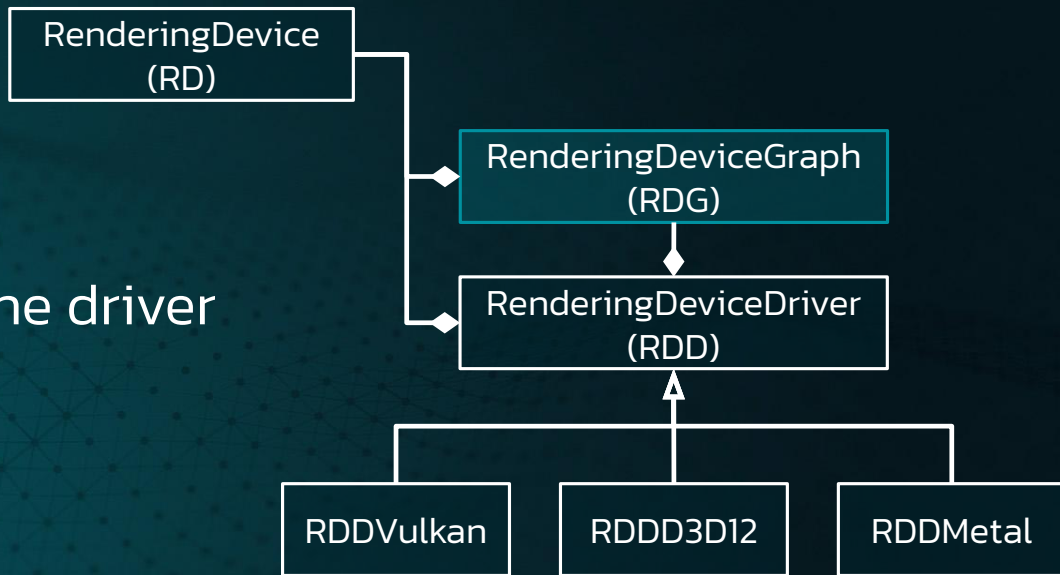
- Vulkan
- D3D12
- Metal



Rendering Device Graph

Authority on synchronization

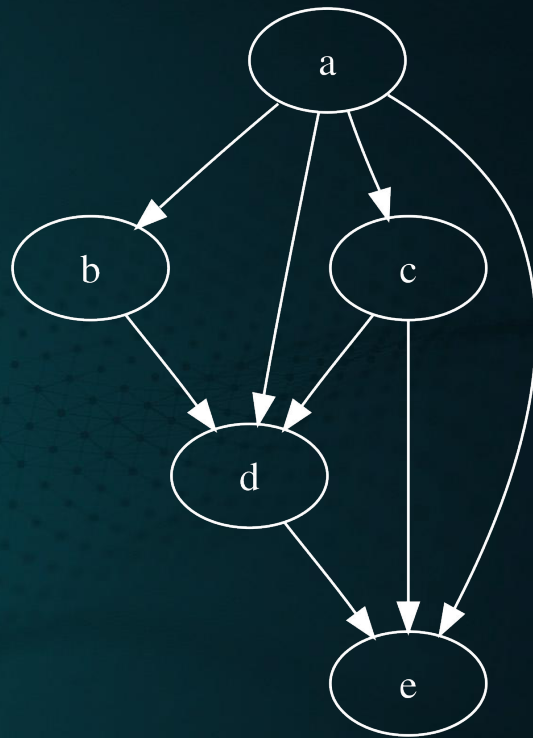
- Records commands
- Tracks resource usages
- Generate barriers
- Submit commands to the driver



Rendering Device Graph

No ad-hoc pipeline barriers in high-level code: the graph handles it

- Direct Acyclic Graph (DAG)
- Rendering operations are nodes
- Hazards become edges



GPU synchronization in Godot 4.3 is getting a major upgrade

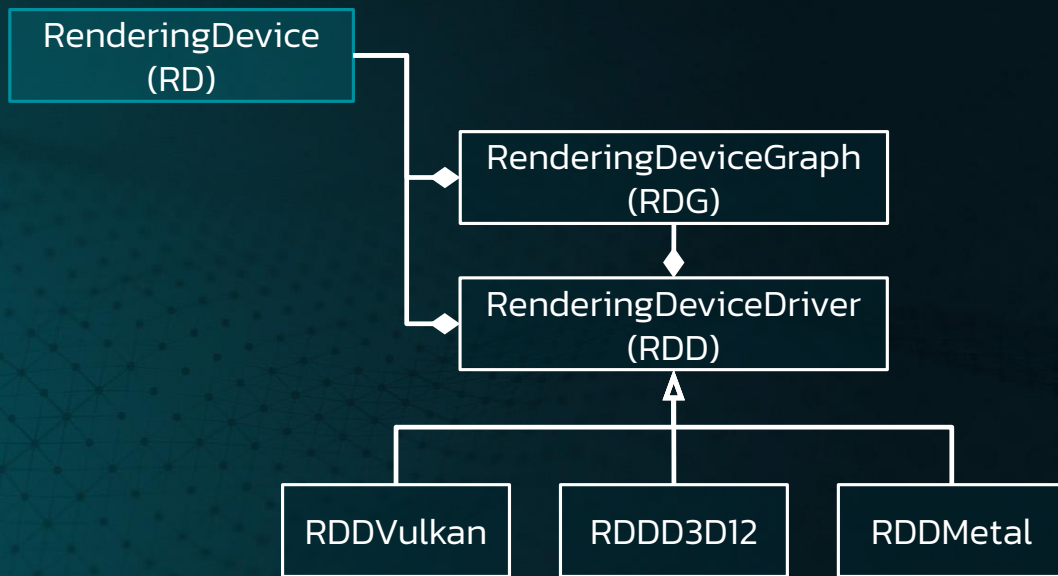
– Darío Banini

<https://godotengine.org/article/rendering-acyclic-graph/>

Rendering Device

The GPU API

- Buffers
- Textures
- Pipelines
- Command recording



Object Model

Engine words

- Resources

- VkBuffer
- VkImage
- VkPipeline
- VkAccelerationStructureKHR
- ...

- Commands

- vkCmd*

- Usages

- READ
- WRITE
- BUILD_INPUT
- SAMPLED
- STORAGE
- ...

Ray tracing hello world: one triangle, one ray per pixel

What does it take?



Roadmap

1. Extensions and Properties
2. New Types
3. New API
4. Acceleration Structures
5. Shaders authoring
6. Recording commands

Vulkan Extensions

RenderingDeviceDriverVulkan::_initialize_device_extensions()

- VK_KHR_acceleration_structure
- VK_KHR_ray_tracing_pipeline
- VK_KHR_buffer_device_address
- VK_KHR_deferred_host_operations

Size and alignment requirements

Queried once and stored

- Acceleration structure:
 - Scratch address alignment
- Shader group:
 - Handle size
 - Handle alignment
 - Base alignment



Pipeline Type

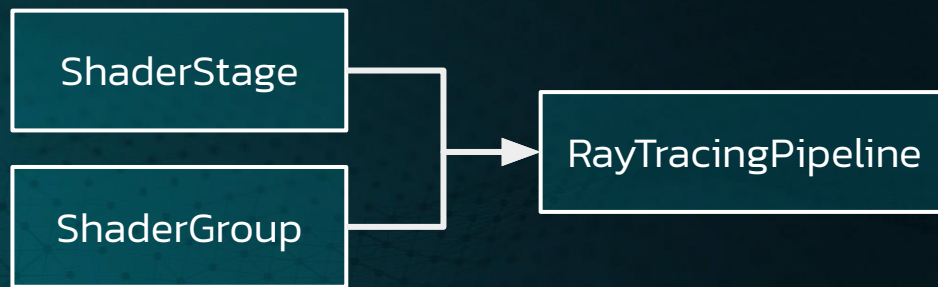
New enumeration value

- Rasterization
- Compute
- Raytracing

NEW

Shader Stages

- Raygen
- Miss
- Closest-hit
- Any-hit
- Intersection



Engine Resources

Stored into "owners", referenced via lightweight resource IDs (RID)

- Bottom Level Acceleration Structure (BLAS)
- Top Level Acceleration Structure (TLAS)
- RayTracing Pipeline



Engine APIs

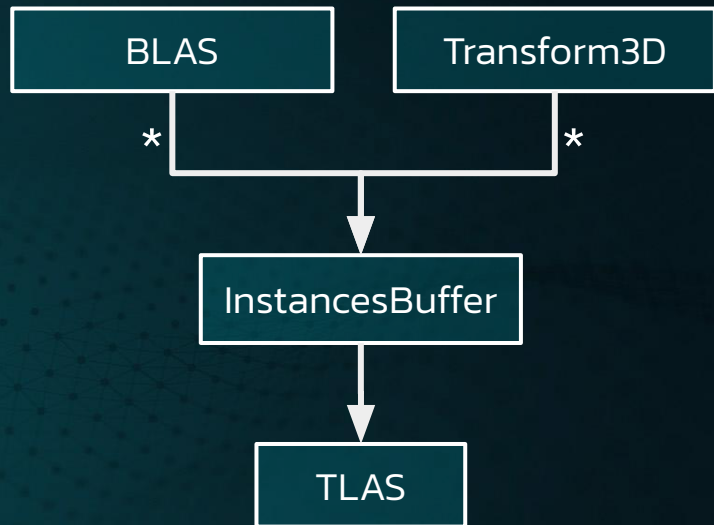
RenderingDevice functions

- blas_create()
- tlas_instances_buffer_*
- tlas_create()
- acceleration_structure_build()
- raytracing_pipeline_create()

Instances buffer

Put BLASs together into a TLAS

- Created with an instance count to reserve space
- Filled with BLAS references and transforms
- Used as input to create TLAS



Scratch buffer

An AS build-time operation

1. Driver reports scratch size of an AS
2. Engine allocates or reuse a buffer to be used as scratch
3. Gets the devices address of the buffer
4. Aligns the address to the required scratch alignment
5. Feeds it into the build command



Acceleration Structure Usage

From which barriers are implied

- Build Input



- Read-Write



- Read



Shader Authoring

New stage markers in source files

- `#[raygen]`
- `#[miss]`
- `#[closest_hit]`
- `#[any_hit]`
- `#[intersection]`

```

#[raygen]

#version 460
#extension GL_EXT_ray_tracing : enable

layout(location = 0) rayPayloadEXT vec3 payload;

layout(set = 0, binding = 0, rgba32f) uniform image2D image;

layout(set = 0, binding = 1) uniform accelerationStructureEXT tlas;

void main() {
    vec2 pixel_center = vec2(gl_LaunchIDEXT.xy) + vec2(0.5);
    vec2 in_uv = pixel_center / vec2(gl_LaunchSizeEXT.xy);
    vec2 d = in_uv * 2.0 - 1.0;

    vec4 target = vec4(d.x, d.y, 1.0, 1.0);
    vec4 origin = vec4(0.0, 0.0, 0.0, 1.0);
    vec4 direction = vec4(normalize(target.xyz), 0);

    float t_min = 0.001;
    float t_max = 10000.0;

    traceRayEXT(tlas, gl_RayFlagsOpaqueEXT, 0xFF, 0, 0, 0,
                origin.xyz, t_min, direction.xyz, t_max, 0);

    imageStore(image, ivec2(gl_LaunchIDEXT.xy),
               vec4(payload, 1.0));
}

```

```

#[miss]

#version 460
#extension GL_EXT_ray_tracing : enable

layout(location = 0) rayPayloadInEXT vec3 payload;

void main() {
    payload = vec3(1.0, 0.0, 0.0);
}

#[closest_hit]

#version 460
#extension GL_EXT_ray_tracing : enable

layout(location = 0) rayPayloadInEXT vec3 payload;

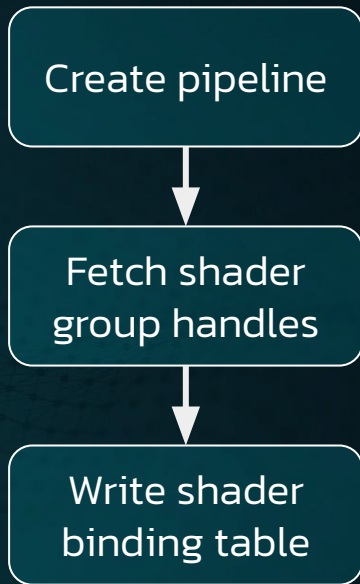
void main() {
    payload = vec3(0.0, 1.0, 0.0);
}

```

Shader Binding Table

This is a bit of a headache

- It contains SHADER GROUP HANDLES
- Key constraints:
 - Shader group handle size
 - Shader group handle alignment
 - Shader group base alignment
 - Region strides



Command lists

Where the recording happens

- Draw list
- Compute list
- Raytracing list



NEW

Raytracing list

RenderingDevice functions

- raytracing_list_begin()
- raytracing_list_end()
- raytracing_list_bind()
- raytracing_list_push()
- raytracing_list_trace_rays()

```
var rd = RenderingDevice.new()
assert(rd.has_feature(RenderingDevice.SUPPORTS_RAYTRACING_PIPELINE))

# Create BLAS and TLAS for a mesh.
blas = rd.blas_create(vertex_array, index_array, RenderingDevice.ACCELERATION_STRUCTURE_GEOMETRY_OPAQUE)
instances_buffer = rd.tlas_instances_buffer_create(1)
rd.tlas_instances_buffer_fill(instances_buffer, [blas], [Transform3D()])
tlas = rd.tlas_create(instances_buffer)

# Build acceleration structures.
rd.acceleration_structure_build(blas)
rd.acceleration_structure_build(tlas)

var raylist = rd.raytracing_list_begin()

# Bind pipeline and uniforms.
rd.raytracing_list_bind_raytracing_pipeline(raylist, raytracing_pipeline)
rd.raytracing_list_bind_uniform_set(raylist, uniform_set, 0)

# Trace rays.
var width = get_viewport().size.x
var height = get_viewport().size.y
rd.raytracing_list_trace_rays(raylist, width, height)

rd.raytracing_list_end()
```

```
func _initialize_raytracing_pipeline():

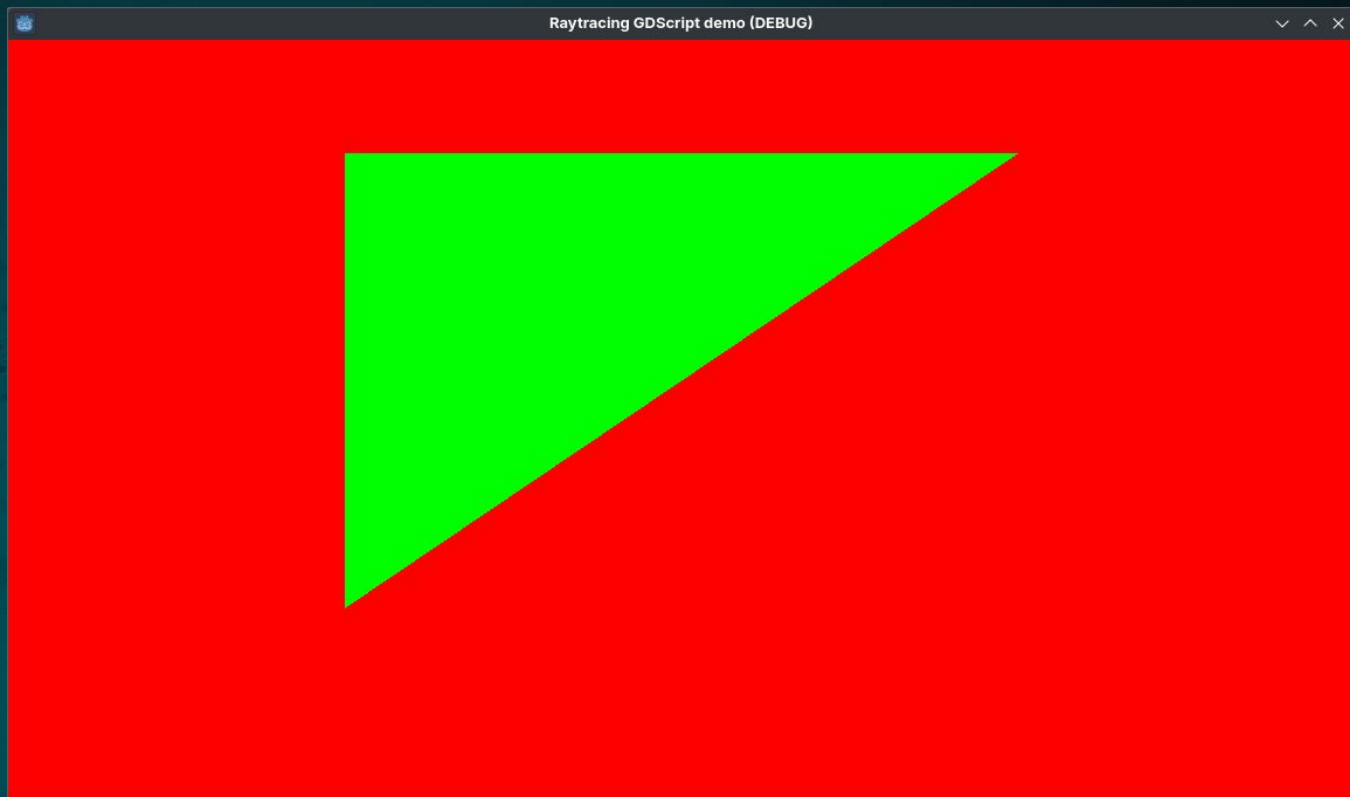
    # Load shaders and create raytracing pipeline
    var shader_file = load("res://ray.glsl")
    var shader_spirv = shader_file.get_spirv()
    shader = rd.shader_create_from_spirv(shader_spirv)
    raytracing_pipeline = rd.raytracing_pipeline_create(shader)

    # Storage image uniform
    var image_uniform = RDUniform.new()
    image_uniform.uniform_type = RenderingDevice.UNIFORM_TYPE_IMAGE
    image_uniform.binding = 0
    image_uniform.add_id(output_image)


    # Acceleration structure uniform
    var as_uniform = RDUniform.new()
    as_uniform.uniform_type = RenderingDevice.UNIFORM_TYPE_ACCELERATION_STRUCTURE
    as_uniform.binding = 1
    as_uniform.add_id(tlas)

    uniform_set = rd.uniform_set_create([image_uniform, as_uniform], shader, 0)
```

Demo: Triangle



Vulkan raytracing plumbing #99119

[Edit](#)[Code](#)[Merged](#)Repitelo merged 1 commit into [godotengine:master](#) from [Fahien:fahien/raytracing-base](#)  last week[Conversation](#) 172[Commits](#) 1[Checks](#) 20[Files changed](#) 28+2,998 -114 

Fahien commented on Nov 12, 2024 · edited

[Contributor](#) ...

Here's a bunch of code adding some Vulkan raytracing stuff to the rendering device:

- Vulkan implementations in `RenderingDeviceDriverVulkan`
- Raytracing instruction list in `RenderingDeviceGraph`
- Functions to create acceleration structures and raytracing pipelines in `RenderingDevice`

There's more in the [fahien/raytracing-test](#) branch, with code handling raygen, miss, and closest-hit shaders, but it's a hack on top of the forward clustered renderer, and it needs some "guided" refactoring.

Here's a sample which uses GDScript to drive the renderer: [raytracing-gdscript-demo](#)

I hope this changes would look useful to jump-start raytracing support.

Relevant for [godotengine/godot-proposals#5162](#)



62



32







31



20

Reviewers

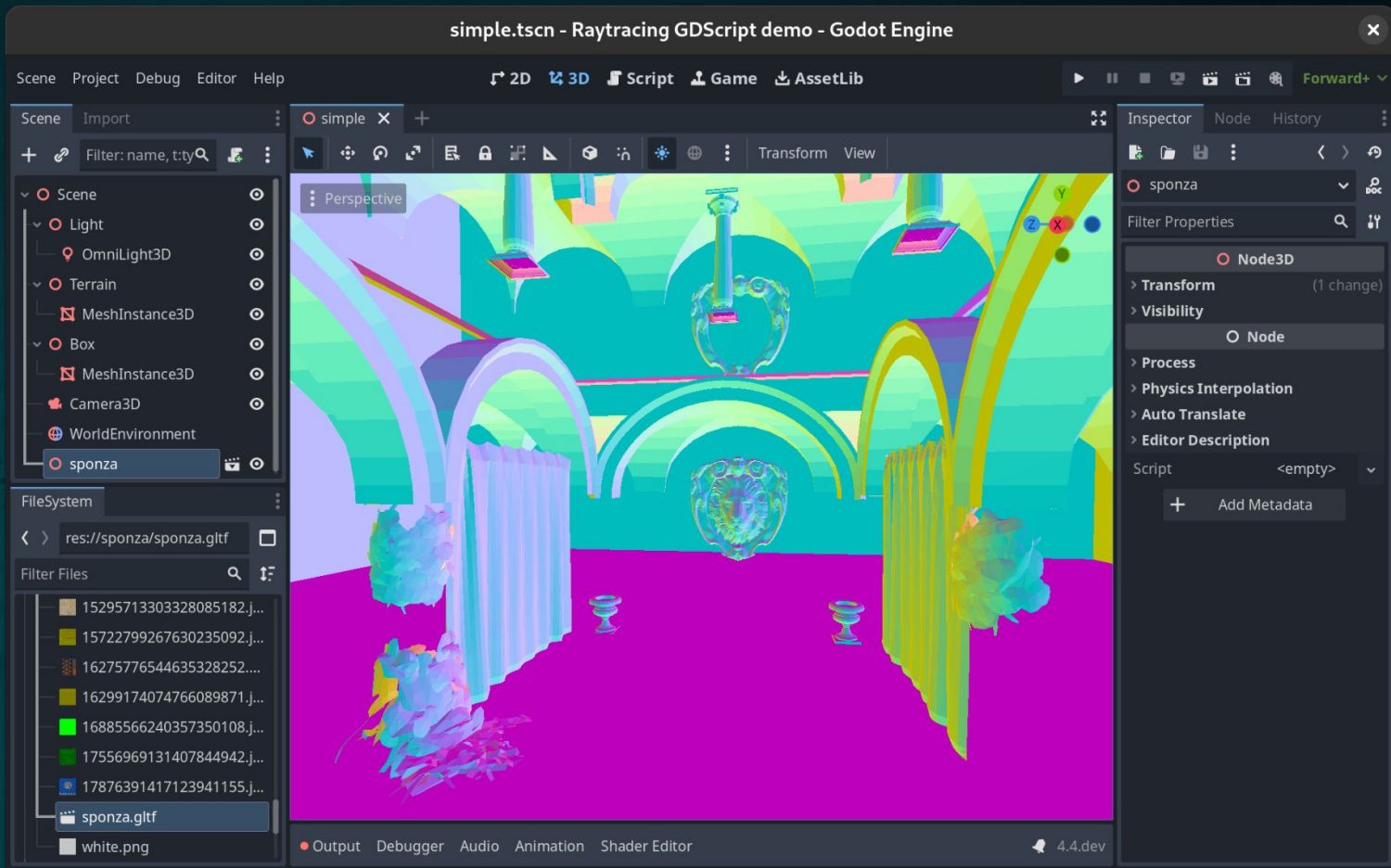
 Calinou  Mickeyon  AThousandShips  DarioSamo  clayjohn [+1 more reviewer](#)  zeux 

Assignees

No one assigned

Labels

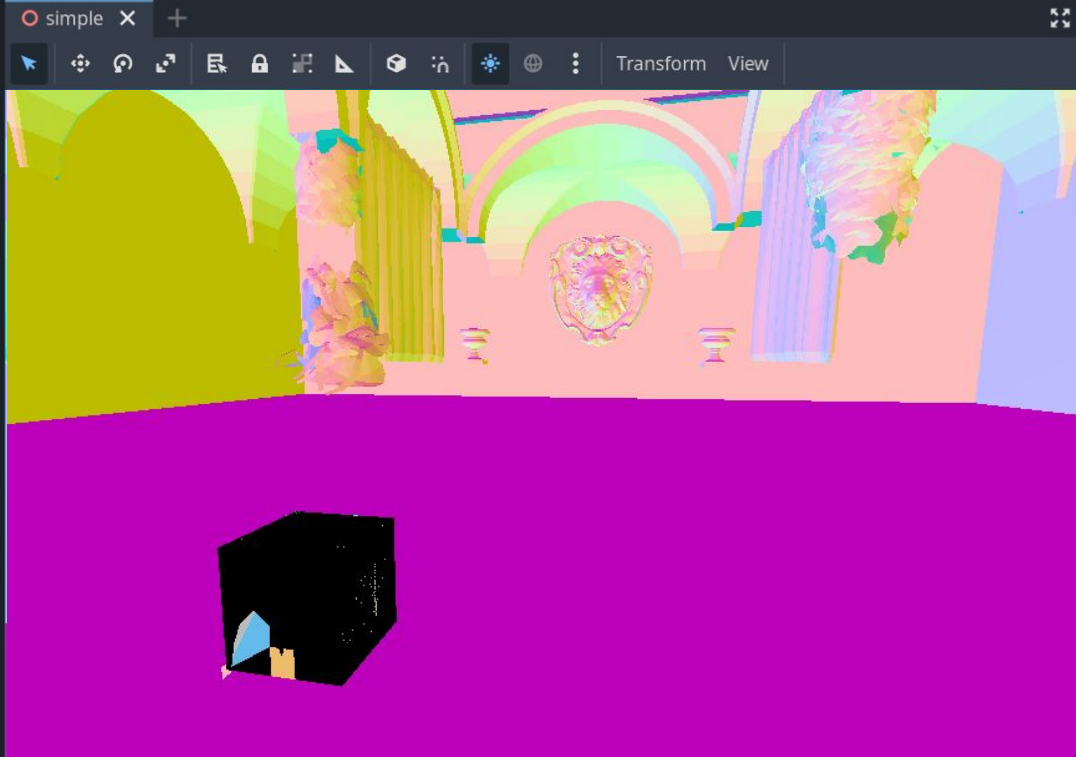
[enhancement](#)[topic:rendering](#)



Scene Import

Filter: name, t:ty

- Scene
 - Light
 - OmniLight3D
 - Terrain
 - MeshInstance3D
 - Box
 - MeshInstance3D
 - Camera3D
 - WorldEnvironment
 - sponza**



FileSystem

res://sponza/sponza.gltf

Filter Files

- 15295713303328085182.j...
- 15722799267630235092.j...
- 16275776544635328252....

Inspector Node History

sponza

Filter Properties

Node3D

- Transform (1 change)
- Visibility
- Node
- Process
- Physics Interpolation
- Auto Translate
- Editor Description

Script <empty>

+ Add Metadata

Mode: Fly Camera

Position (X): _____

Position (Y): _____

Position (Z): _____

Vertex Data

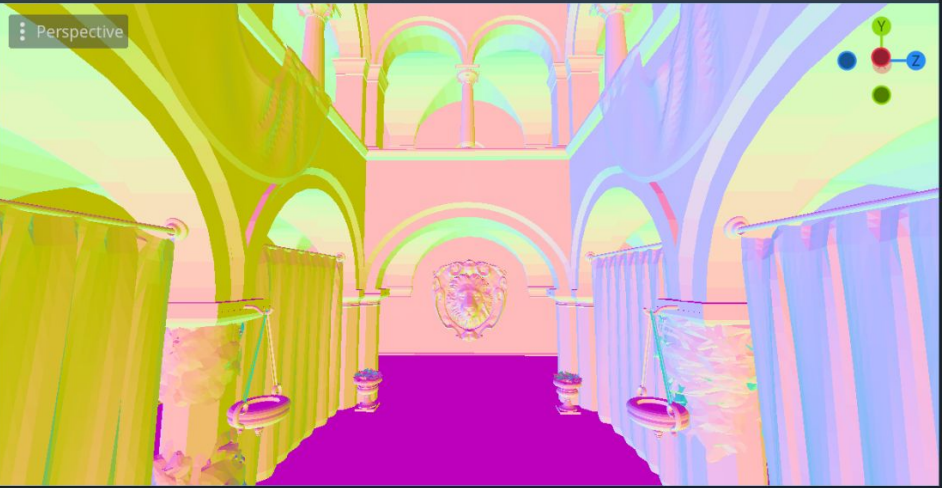
Axis: Address View: 0x0 - 0x1a40

	x	y	z	w
0x00000000	0.4996	0.0000	0.8714	0.2500
0x00000008	0.3004	0.0000	1.0000	0.2500
0x00000010	0.4996	0.0000	1.0000	0.2500
0x00000018	0.4996	0.0000	0.8714	0.2500
0x00000020	0.3004	0.0000	0.8714	0.2500
0x00000028	0.3004	0.0000	1.0000	0.2500
0x00000030	0.4996	0.0000	1.0000	0.5000
0x00000038	0.3004	0.8322	1.0000	0.5000
0x00000040	0.4996	0.8322	1.0000	0.5000

Cell: 0 0

Cull Disabled

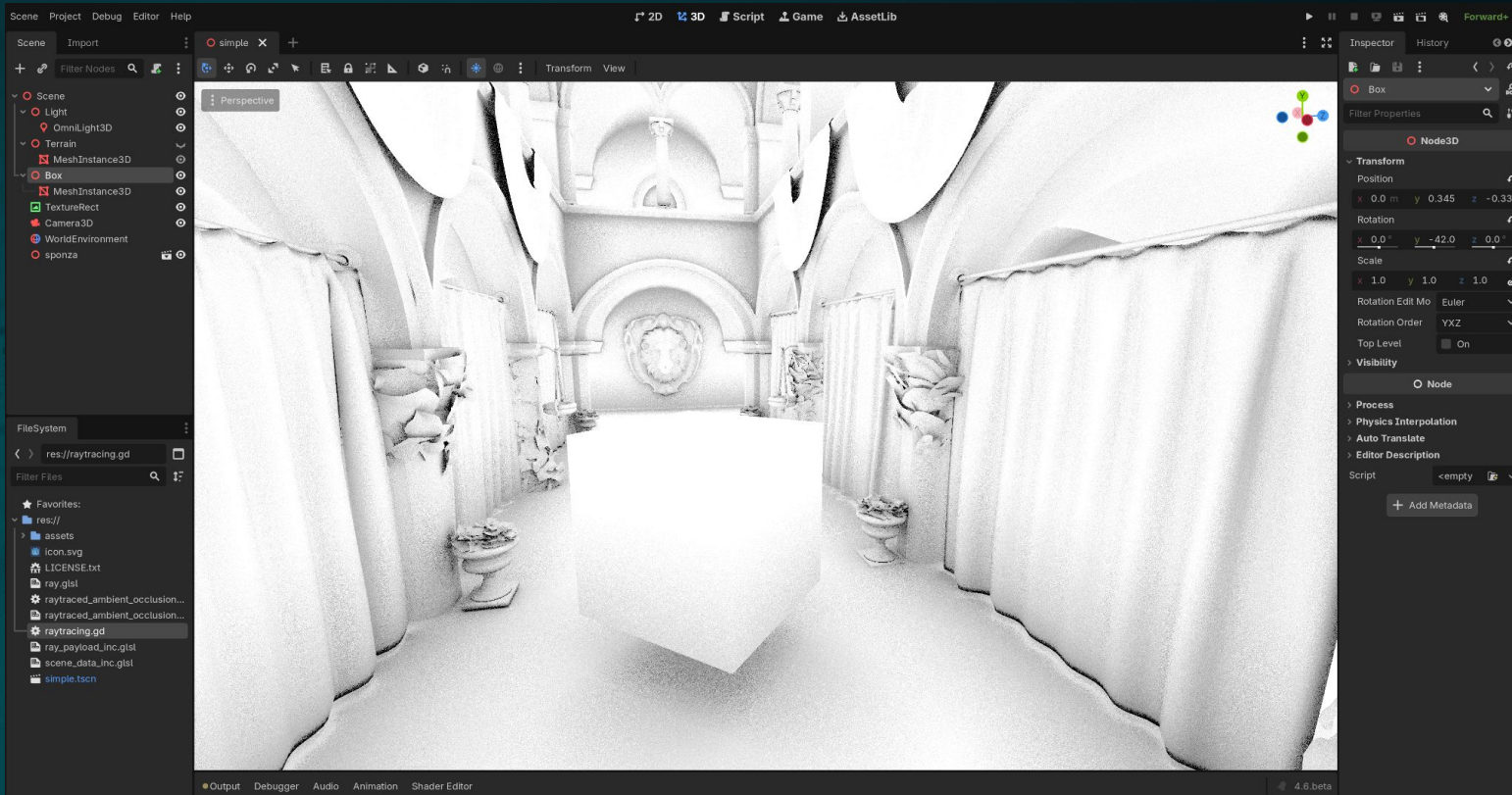
- Scene
 - Camera3D
 - WorldEnvironment
 - sponza



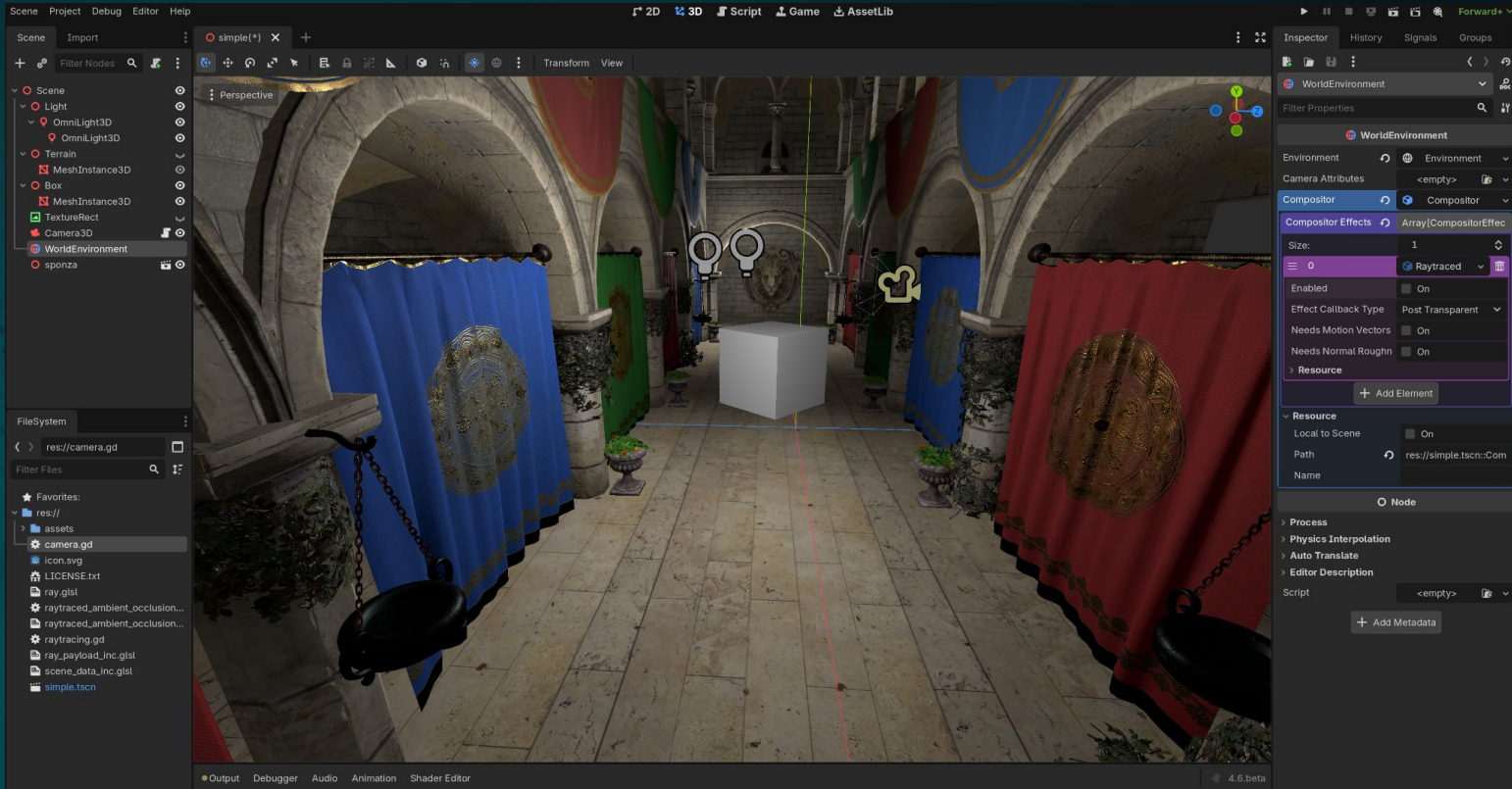
- 1629917407476608987...
- 1688556624035735010...
- 1755696913140784494...
- 1787639141712394115...
- sponza.gltf
- white.png
- suzanne
- triangle
- icon.svg

Inspector panel showing node properties and a hierarchy view.

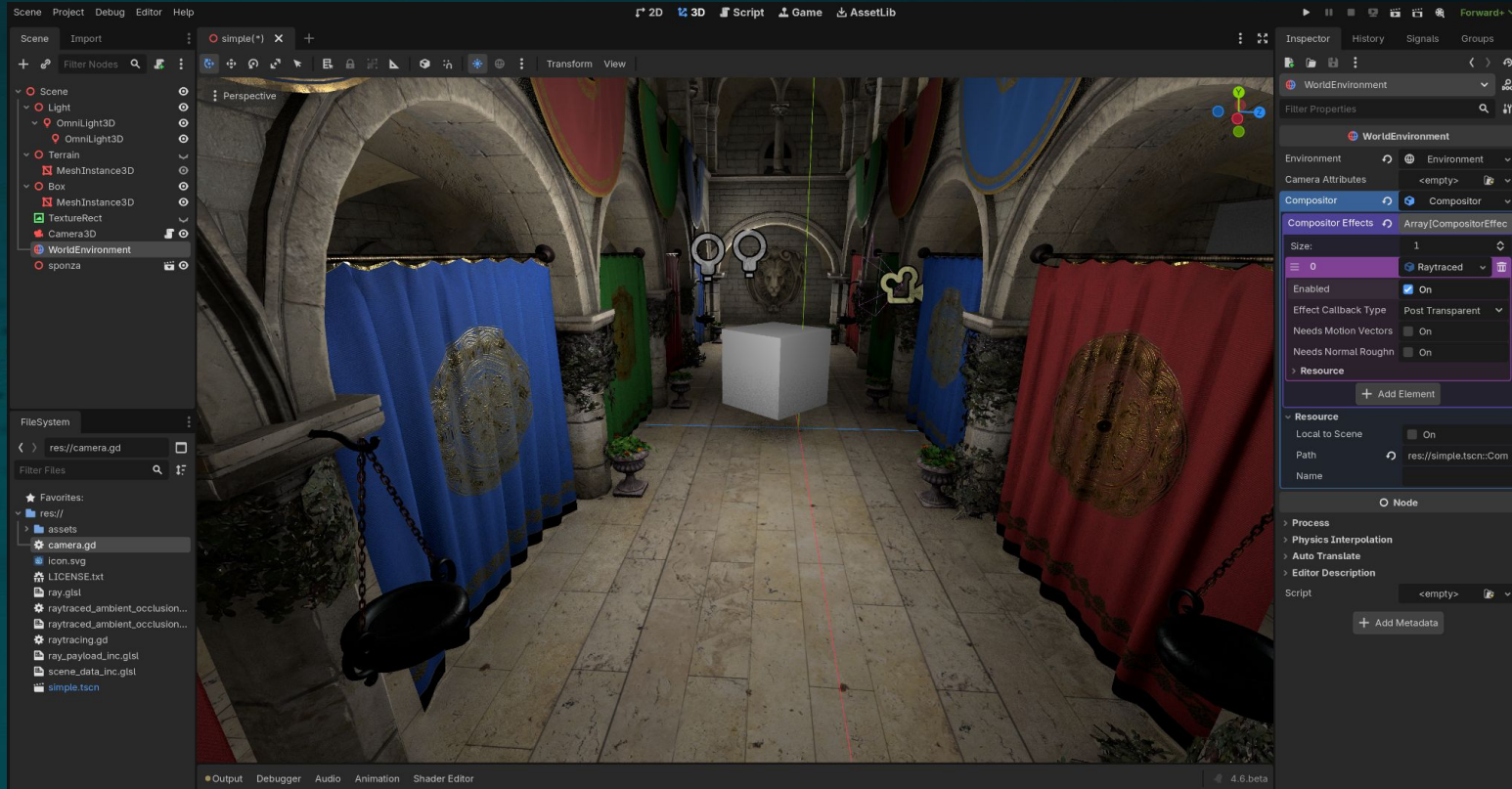
Demo: RTAO



Demo: Sponza Forward+

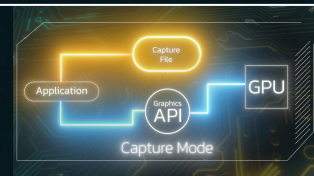


Demo: Sponza Forward+ RTAO





Come to the LunarG Table!
See KosmicKrisp & GFXReconstruct



Take the 2026 Vulkan
Ecosystem Survey!



LunarG Presentations
Vulkanised 2026



LunarG Presentations
**Shading Languages
Symposium 2026**



