Ivans Minajevs

Salamander

Graphics Programming II

Overview

High-Level Overview

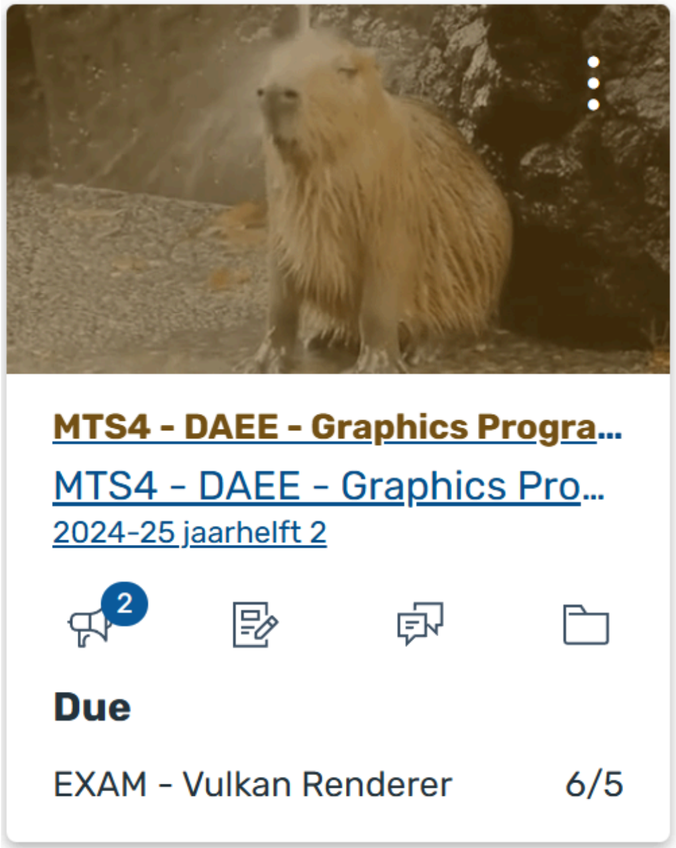
Code Architecture

Feature Showcase

Memory Usage

Graphics Debug Capture

Performance Statistics



High Level Overview

High-Level Overview

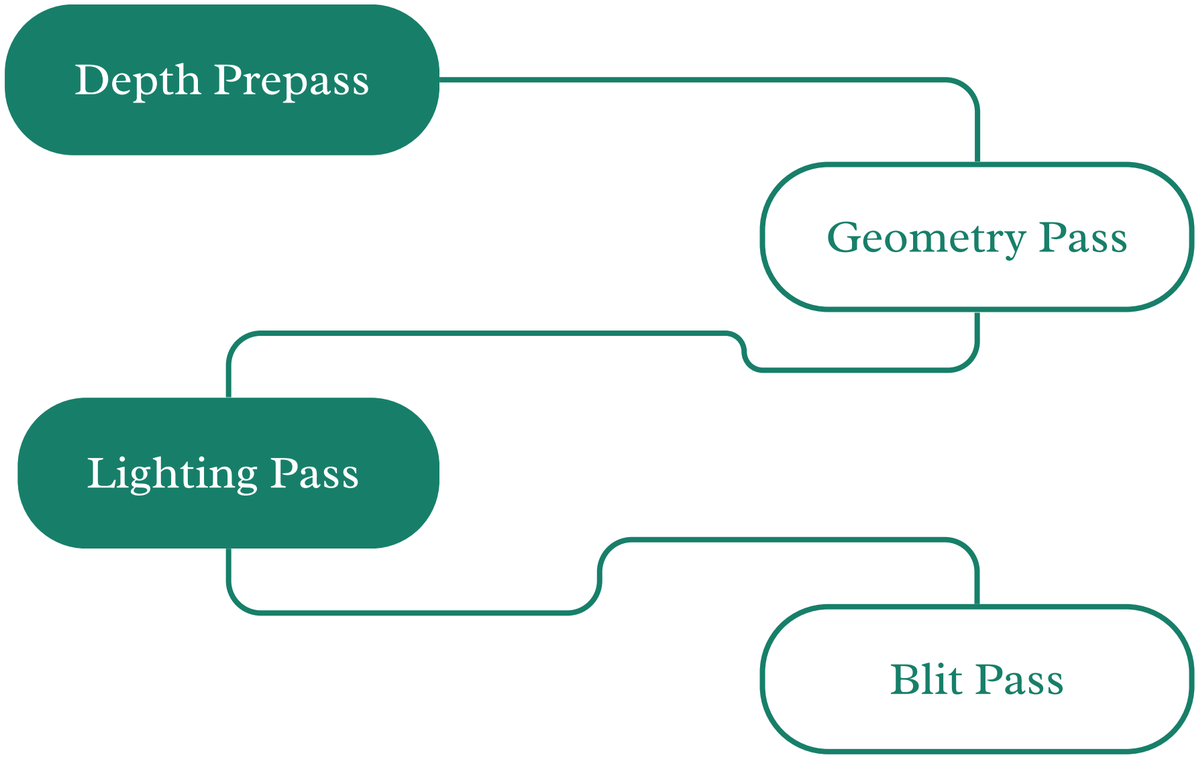
Code Architecture

Feature Showcase

Memory Usage

Graphics Debug Capture

Performance Statistics



Depth prepass

Depth–Stencil-Only Pass

Depth attachment cleared and configured (no color outputs)

Push Constants

vertexBufferAddress → GPU address to SSBO buffer with vertex data

Draw Sequence

1. Bind depth-only pipeline
2. Bind index buffer and descriptor set
3. Loop over primitives → *vkCmdPushConstants* → *vkCmdDrawIndexed*

Layout Transitions & Copy

1. Per-frame depth image: UNDEFINED → DEPTH_STENCIL_ATTACHMENT_OPTIMAL
2. After rendering: DEPTH_STENCIL_ATTACHMENT_OPTIMAL → GENERAL
3. Copy depth into a sampling texture via *vkCmdCopyImage*
4. Transition both images to DEPTH_STENCIL_READ_ONLY_OPTIMAL

Result

1. Per-frame depth buffer
2. Separate *depth texture* ready for sampling in later passes (world position from depth, skybox check)

Depth prepass

Geometry Pass

Attachment Setup

- Albedo: VK_FORMAT_R8G8B8A8_SRGB
- Normal: VK_FORMAT_R8G8B8A8_UNORM
- Params: VK_FORMAT_R8G8_UNORM
- Depth: VK_FORMAT_D32_SFLOAT

Layout Transitions

1. Albedo/Normal/Params: UNDEFINED → COLOR_ATTACHMENT_OPTIMAL
2. Depth: already in DEPTH_STENCIL_READ_ONLY_OPTIMAL
3. Lastly Albedo/Normal/Params: COLOR_ATTACHMENT_OPTIMAL → SHADER_READ_ONLY_OPTIMAL

Fragment Outputs & Encoding

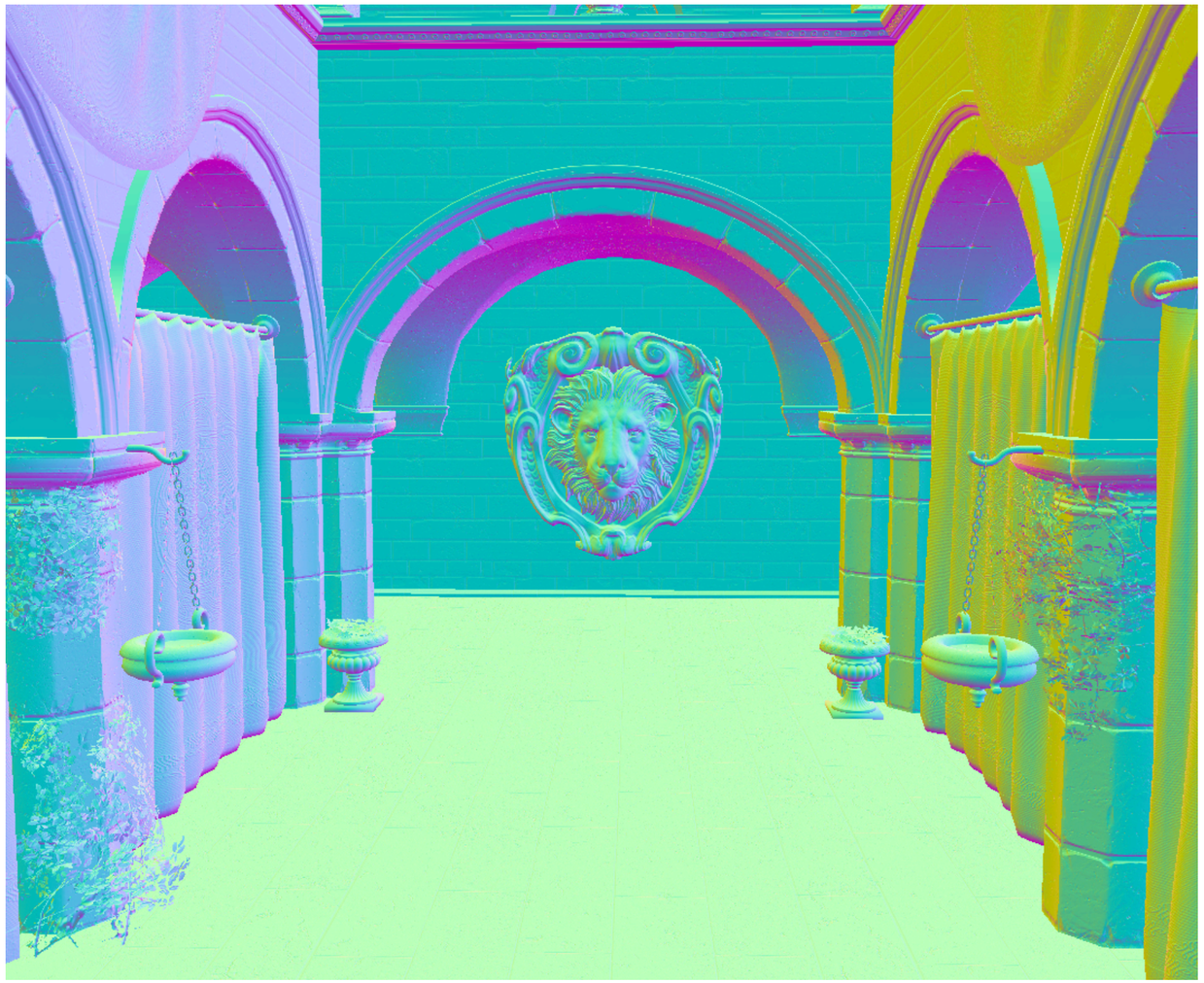
1. *Albedo*: Sample *base color*, *alpha-test*
2. *Normal*: Sample *normal map* (if present) → apply *TBN* → compute *world-space normal* → encode final normal as *(normal * 0.5) + 0.5* → store in RG8B8A8 UNORM
3. *Params*: Sample metallic-roughness →
   Roughness → store in R,
   Metallic → store in G

Result

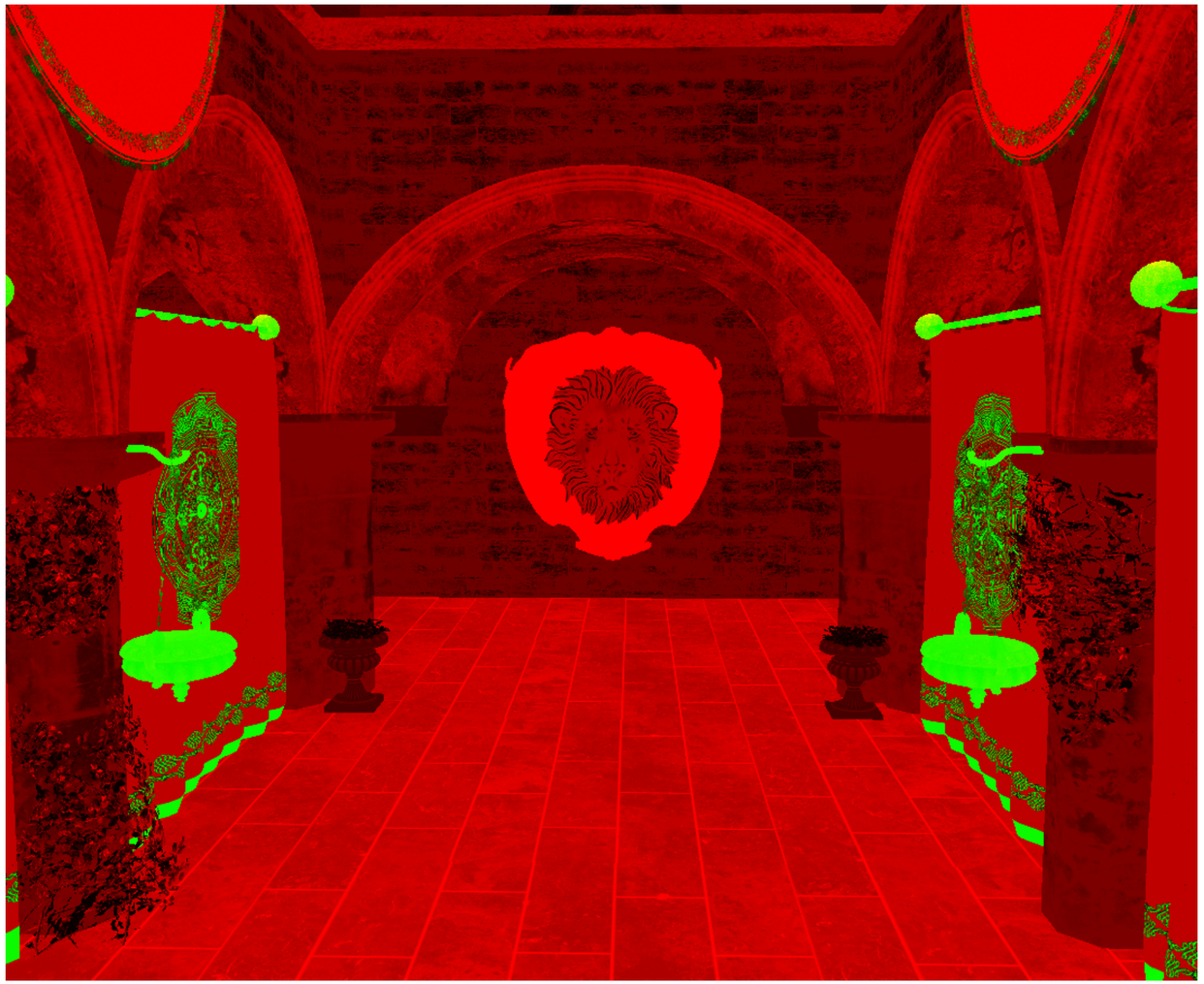
1. Three shader-readable G-buffer textures (*albedo, normal, roughness/metallic*)
2. Ready for deferred *Lighting Pass*

Geometry Pass - Albedo

Geometry Pass - Normals



Geometry Pass - R/M



Lighting Pass

Fragment Outputs & Encoding

- *Sample G-Buffer*:
  Albedo (RGBA8 SRGB)
  Normal (decoded from UNORM)
  Roughness/Metallic (RG8)
  Depth (D32_FLOAT) → reconstruct world position
- *Direct Lighting (PBR)*:
  Directional: shadow + Cook-Torrance BRDF
  Point: attenuation + Cook-Torrance BRDF
- *Indirect (IBL)*:
  Sample irradiance cubemap (diffuse) → ambient term

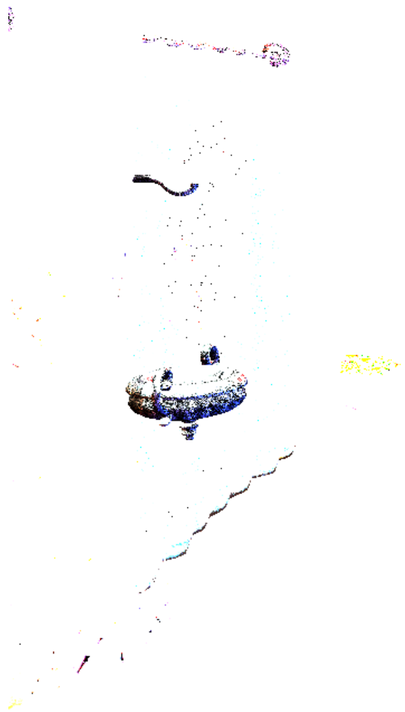
Begin Dynamic Rendering

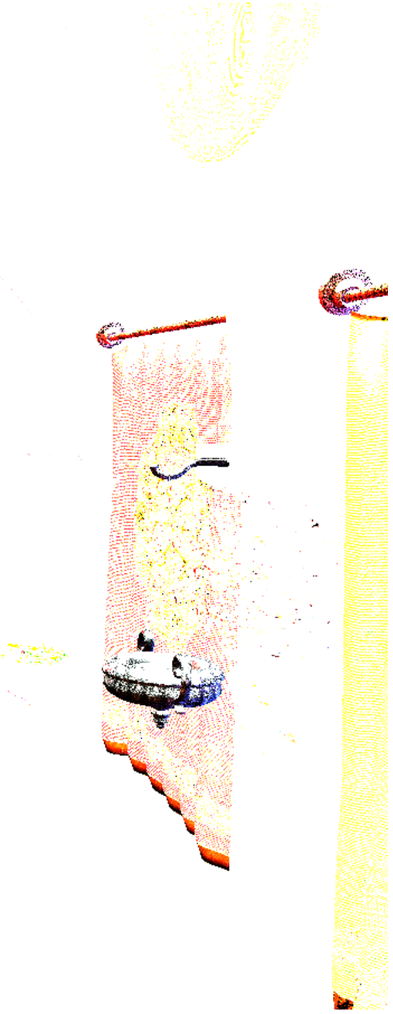
- Bind lighting pipeline & descriptor set (G-Buffer + depth + IBL + shadow map + light UBOs)
- *Full-screen triangle* draw (no vertex/index buffers)

Result

1. Combined PBR color → HDR target
2. Transition HDR: COLOR_ATTACHMENT_OPTIMAL → SHADER_READ_ONLY_OPTIMAL (for post)

Lighting Pass





Tone Mapping Pass

Code Architecture

High-Level Overview

Code Architecture

Feature Showcase

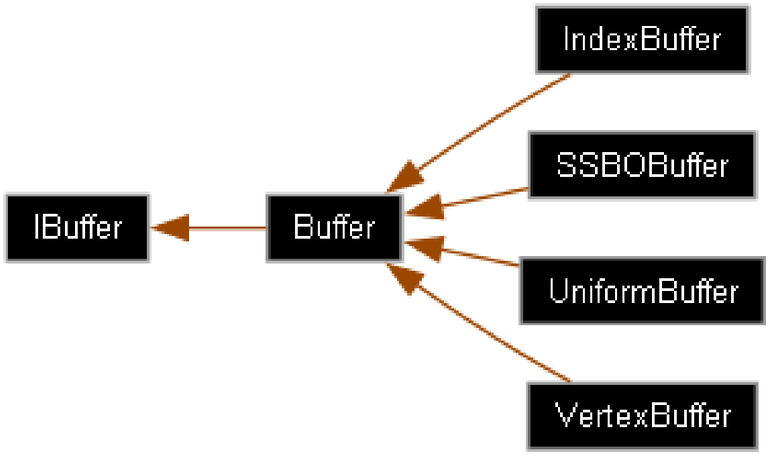
Memory Usage

Graphics Debug Capture

Performance Statistics

Code Architecture

Buffers setup



BufferManager Class Reference

```
#include <buffer_manager.h>
```

Public Member Functions

	BufferManager (VkDevice device, VmaAllocator allocator, CommandManager *commandManager)
	BufferManager (const BufferManager &)=delete
BufferManager &	operator= (const BufferManager &)=delete
	BufferManager (BufferManager &&)=delete
BufferManager &	operator= (BufferManager &&)=delete
ManagedBuffer	createBuffer (VkDeviceSize size, VkBufferUsageFlags usage, VmaMemoryUsage memoryUsage)
void	copyBuffer (VkBuffer srcBuffer, VkBuffer dstBuffer, VkDeviceSize size) const
VmaAllocator	allocator () const

Code Architecture

Texture manager

TextureManager Class Reference

```
#include <texture_manager.h>
```

Public Member Functions

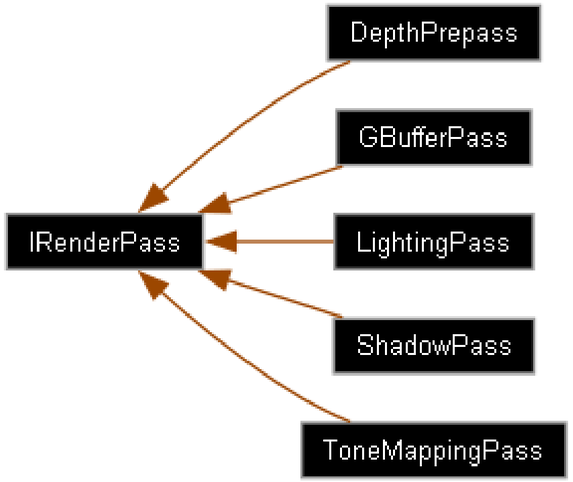
```
                                    TextureManager (VkDevice device, VmaAllocator allocator, CommandManager *commandManager, BufferManager *bufferManager, DebugMessenger
                                    *debugMessenger)
                                    TextureManager (const TextureManager &)=delete
                  TextureManager & operator= (const TextureManager &)=delete
                                    TextureManager (TextureManager &&)=delete
                  TextureManager & operator= (TextureManager &&)=delete
                  ManagedTexture & loadTexture (const std::string &filepath, VkFormat format=VK_FORMAT_R8G8B8A8_SRGB)
                  ManagedTexture & loadHDRTexture (const std::string &path)
                  ManagedTexture & createTexture (uint32_t width, uint32_t height, VkFormat format, VkImageUsageFlags usage, VmaMemoryUsage memoryUsage, VkImageAspectFlags
                                    aspect, bool createSampler=false)
                  ManagedTexture & createTexture (const unsigned char *data, uint32_t width, uint32_t height, uint32_t channels)
                  ManagedTexture & createCubeTexture (uint32_t size, VkFormat format, VkImageUsageFlags usage, VmaMemoryUsage memoryUsage)
                             void createImage (uint32_t width, uint32_t height, VkFormat format, VkImageTiling tiling, VkImageUsageFlags usage, VmaMemoryUsage memoryUsage,
                                    VkImage &image, VmaAllocation &allocation, uint32_t layers, VkImageCreateFlags flags) const
const std::vector< ManagedTexture > & getTextures () const
```

Static Public Member Functions

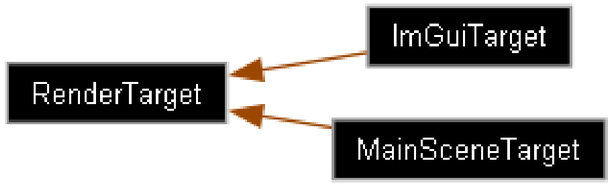
```
static void transitionSwapChainLayout (VkCommandBuffer cmd, VkImage image, VkImageLayout oldLayout, VkImageLayout newLayout, VkPipelineStageFlags2 srcStageMask,
            VkPipelineStageFlags2 dstStageMask, VkAccessFlags2 srcAccessMask, VkAccessFlags2 dstAccessMask)
 static int getSamplerIndex ()
static void incrementSamplerIndex ()
```

Code Architecture

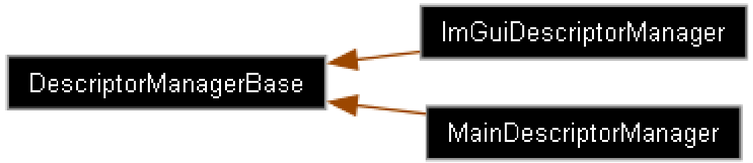
Rendering



Render passes



Render targets



Descriptor manager

Feature showcase

High-Level Overview

Code Architecture

Feature Showcase

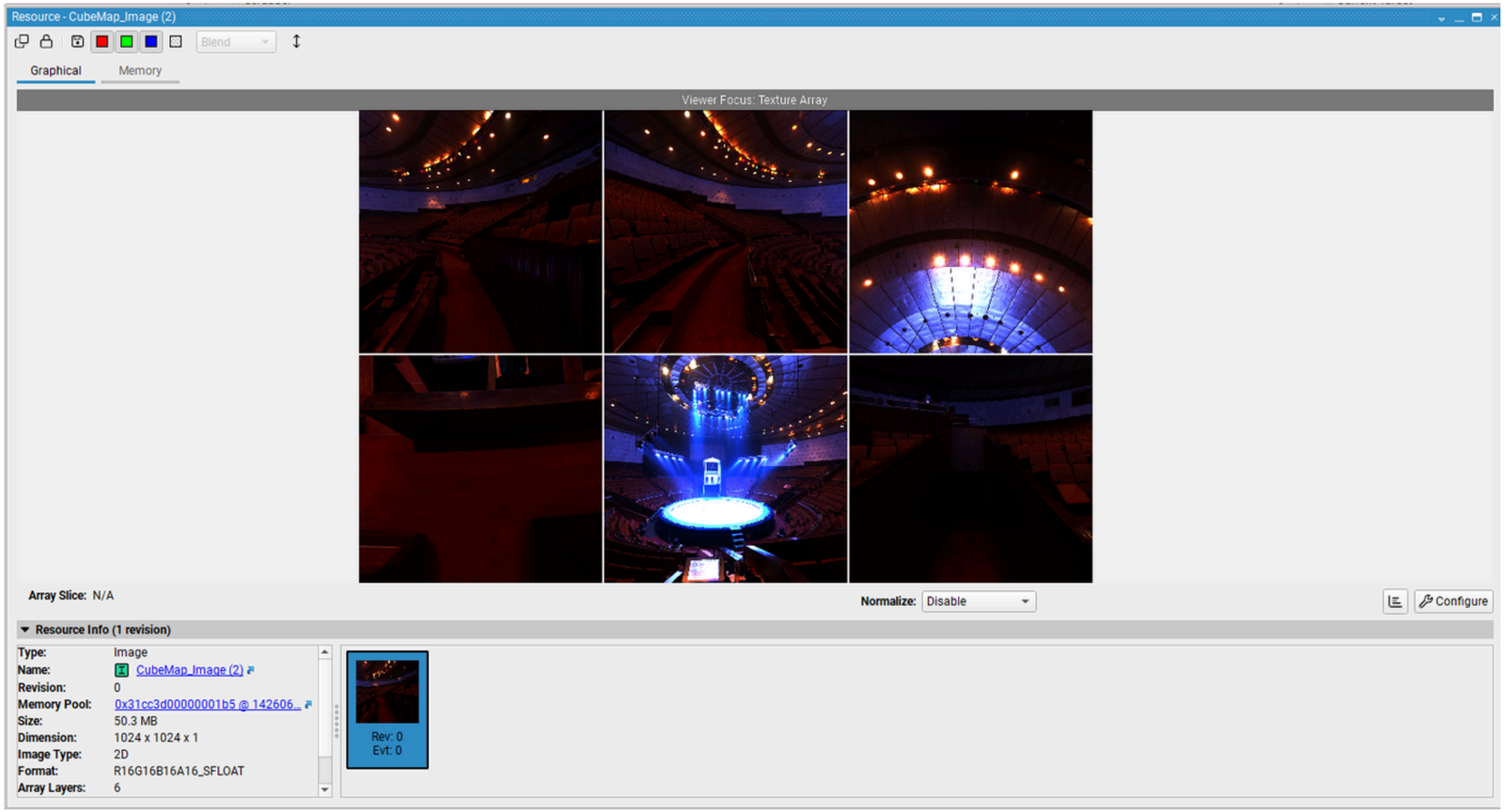
Memory Usage

Graphics Debug Capture

Performance Statistics

Feature showcase

Environment cube map

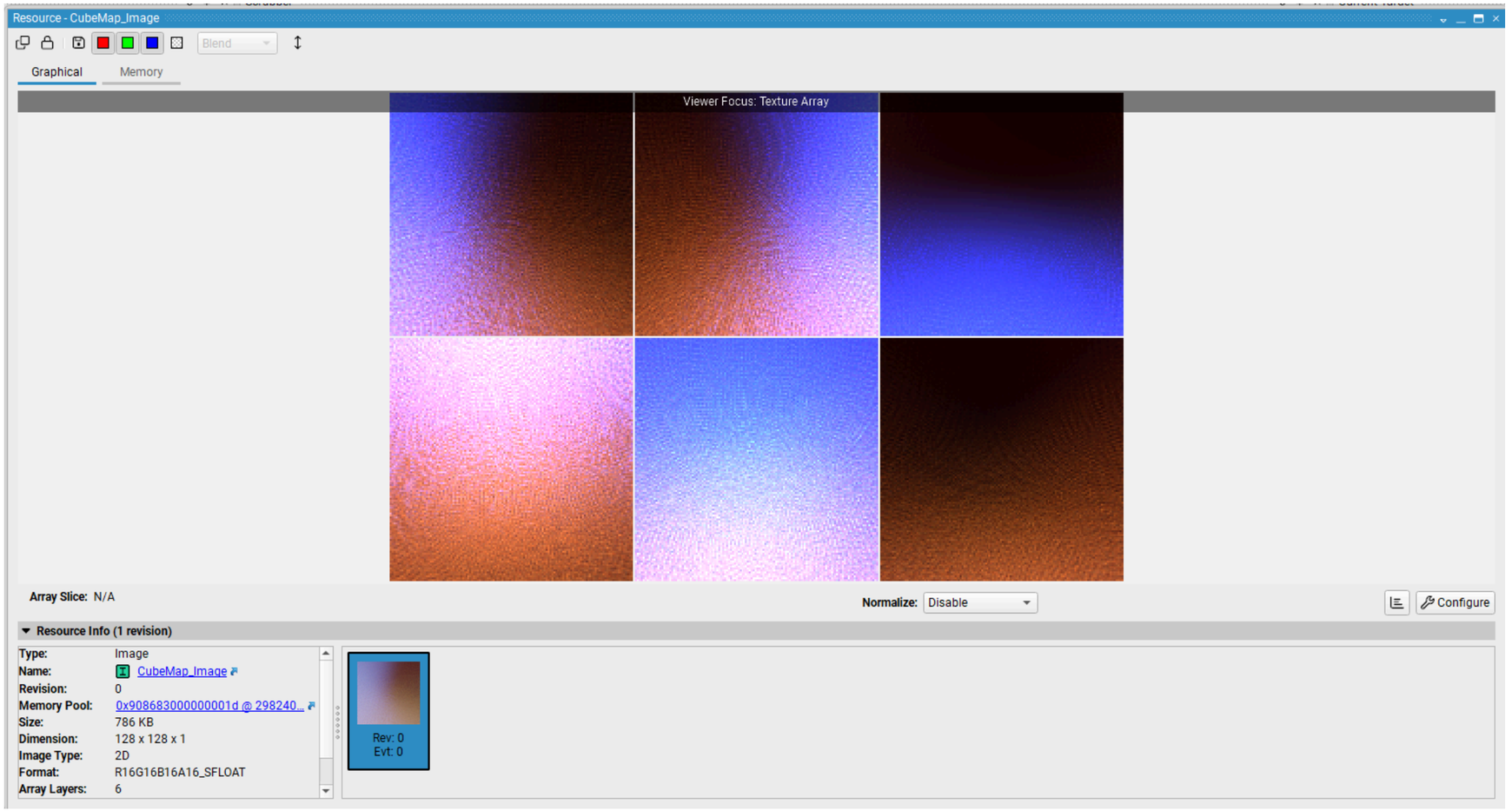


Feature showcase

Skybox

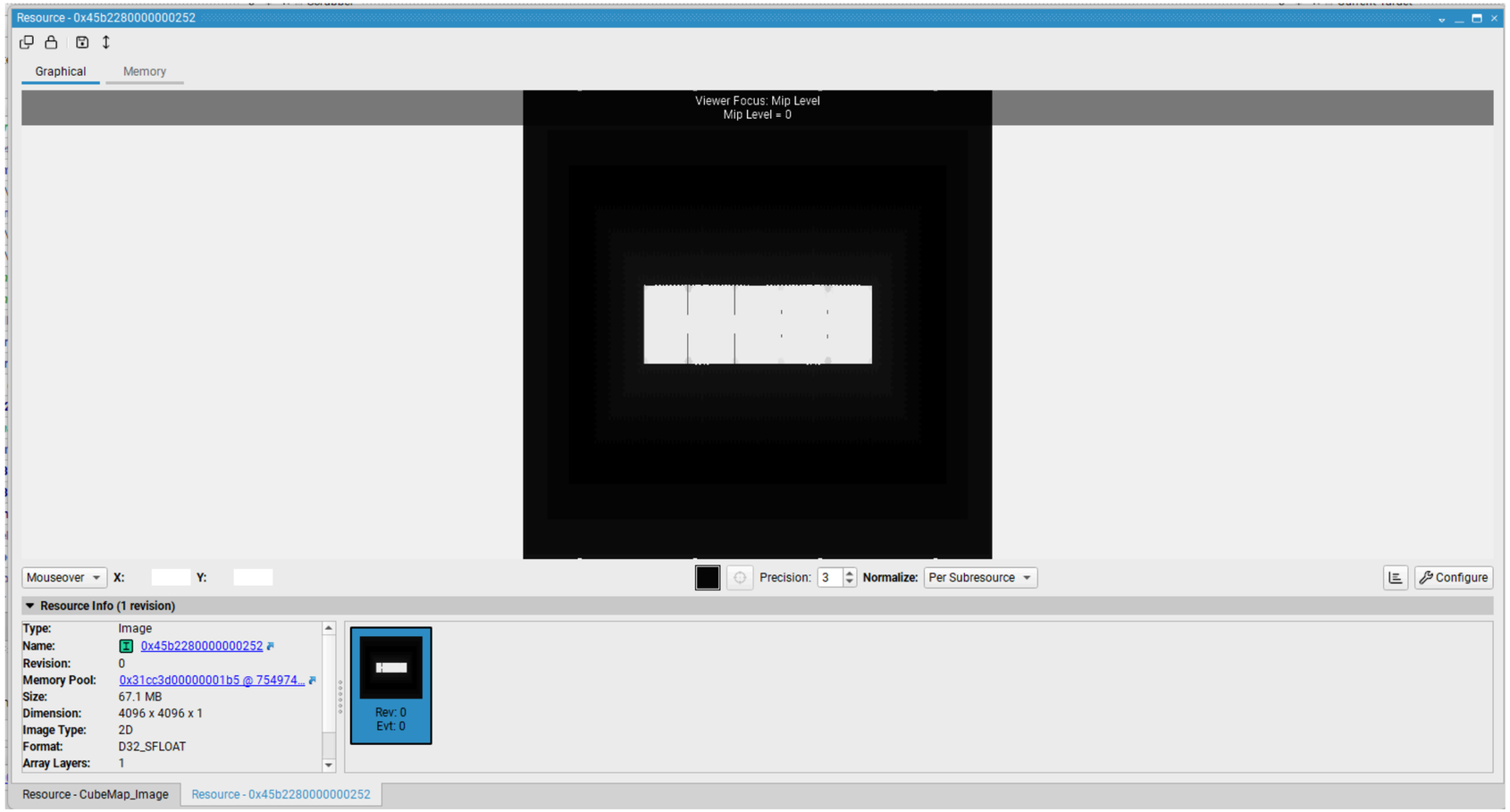
Feature showcase

Diffuse irradiance



Feature showcase

Shadows



Memory Usage

High-Level Overview

Code Architecture

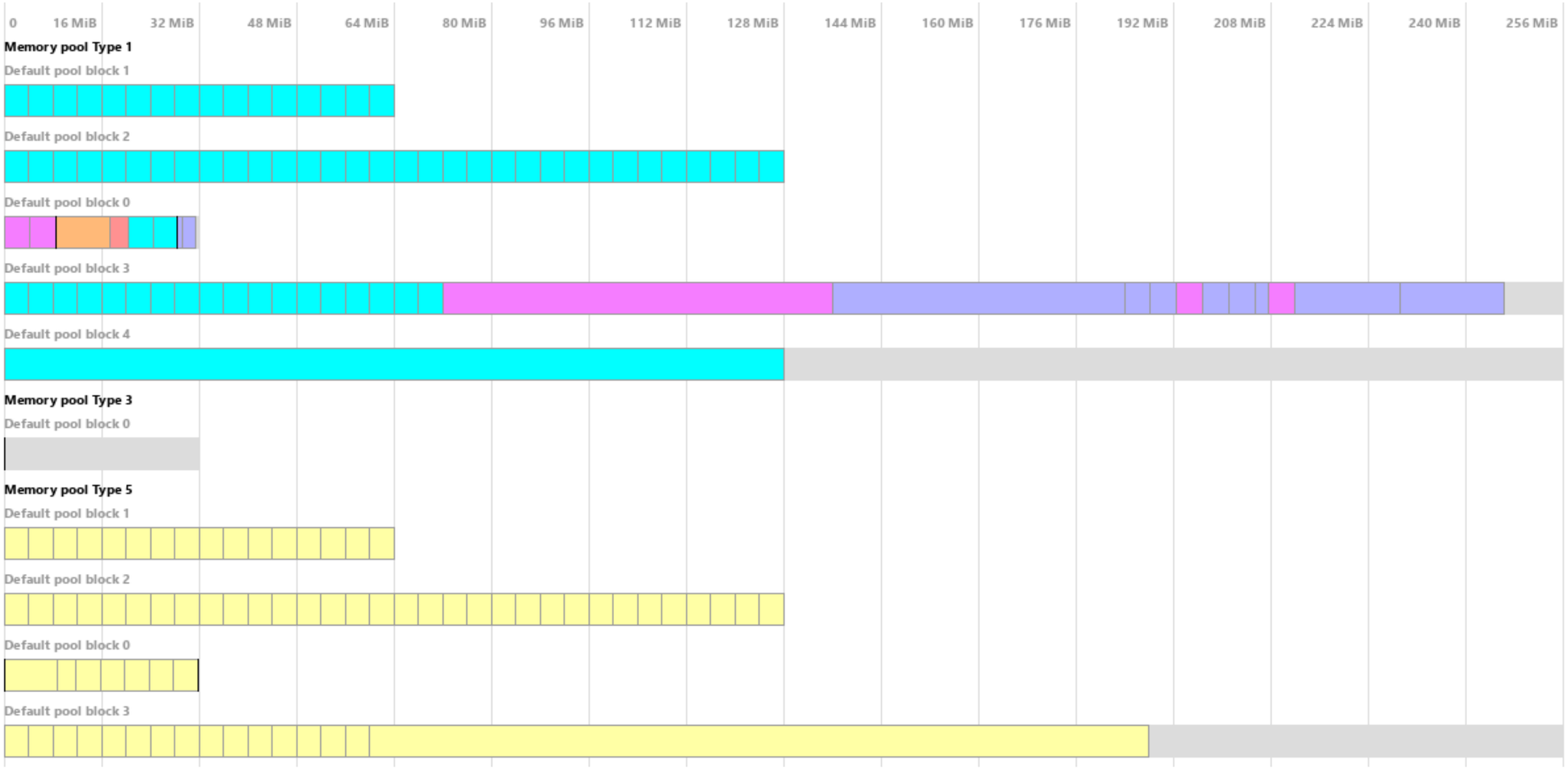
Feature Showcase

Memory Usage

Graphics Debug Capture

Performance Statistics

Memory Usage



Graphics Debug Capture

High-Level Overview

Code Architecture

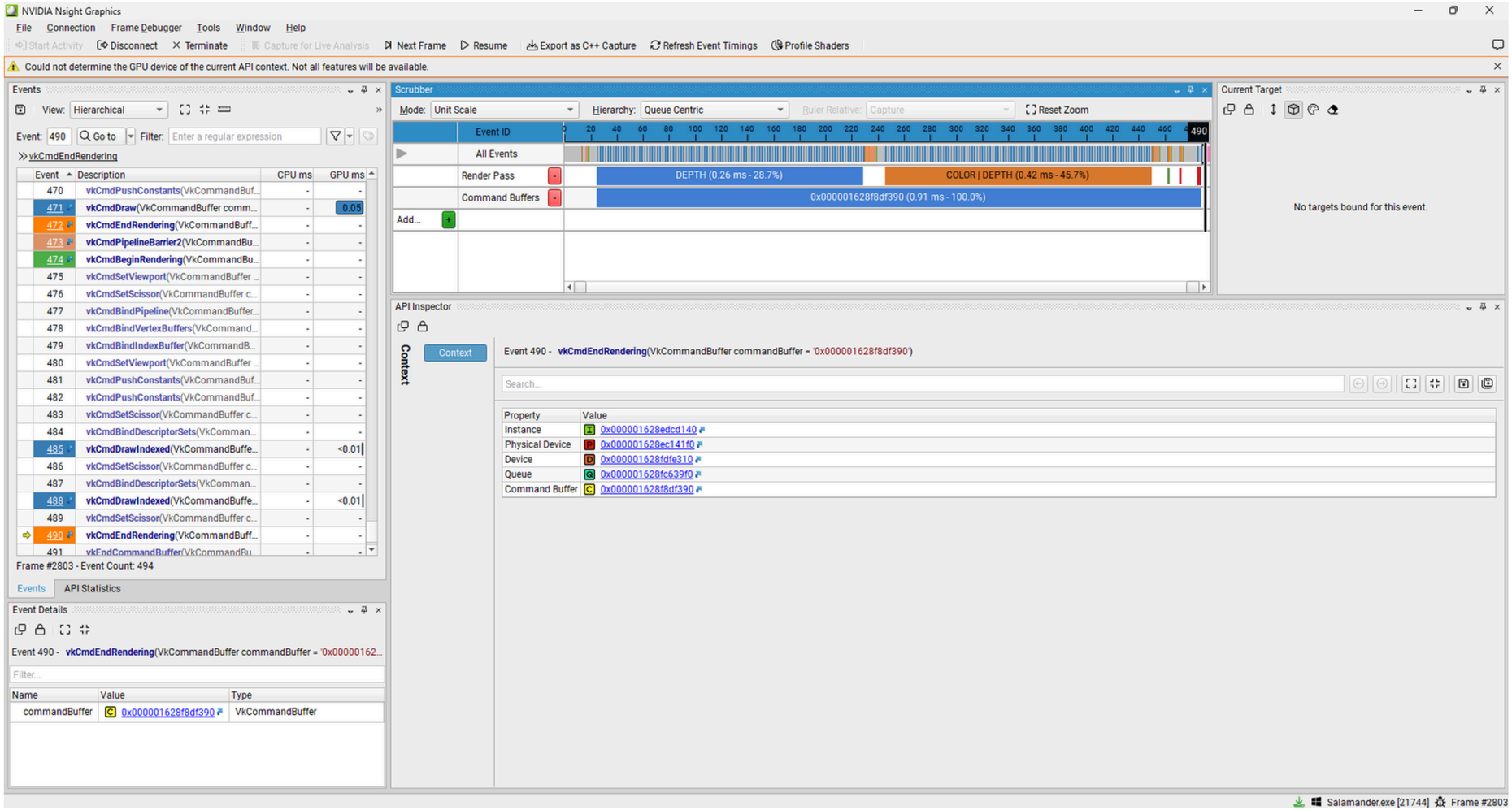
Feature Showcase

Memory Usage

Graphics Debug Capture

Performance Statistics

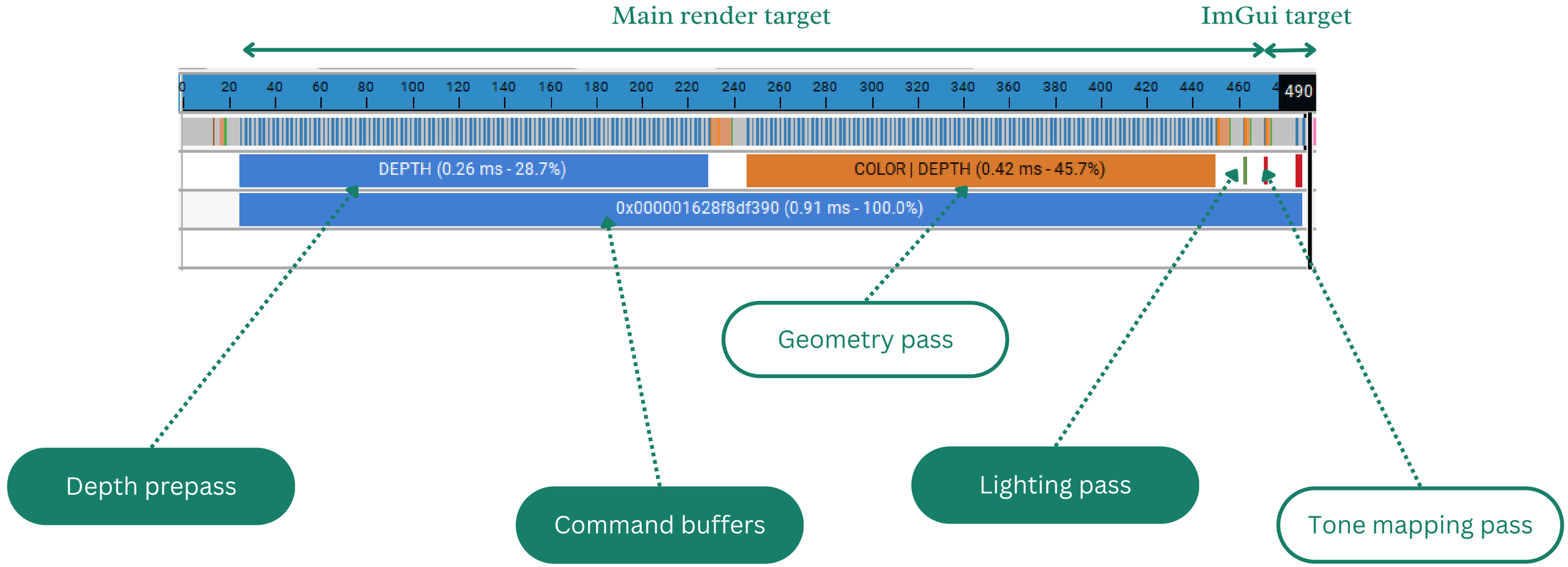
Graphics Debug Capture



DIGITAL ARTS & ENTERTAINMENT howest / we develop people

Graphics Debug Capture

Closer view



Performance Statistics

High-Level Overview

Code Architecture

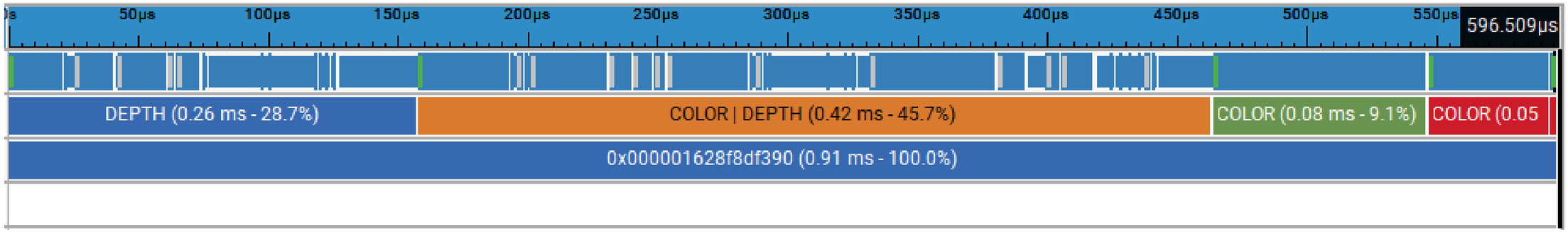
Feature Showcase

Memory Usage

Graphics Debug Capture

Performance Statistics

Performance Statistics



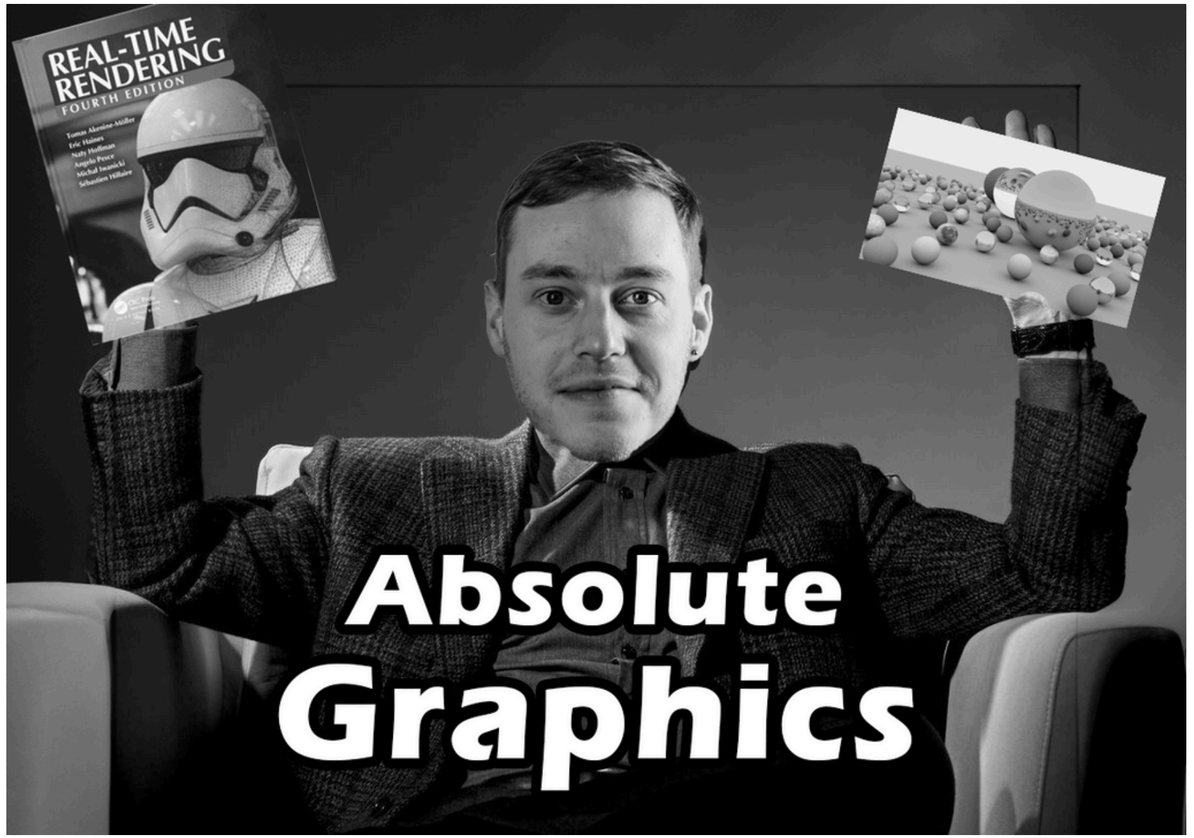
Small "proud of" moment

```
struct DescriptorUpdateInfo {
    uint32_t binding;
    VkDescriptorType type;
    union {
        VkDescriptorBufferInfo* bufferInfo;
        VkDescriptorImageInfo* imageInfo;
    };
    uint32_t descriptorCount;
    bool isImage;
};
```



```
std::vector<MainDescriptorManager::DescriptorUpdateInfo> updates = {
    {
        .binding = 0,
        .type = VK_DESCRIPTOR_TYPE_UNIFORM_BUFFER,
        .bufferInfo = &m_globalData->frameData[i].bufferInfo,
        .descriptorCount = 1,
        .isImage = false
    },
    {
        .binding = 1,
        .type = VK_DESCRIPTOR_TYPE_COMBINED_IMAGE_SAMPLER,
        .imageInfo = &albedoInfo,
        .descriptorCount = 1,
        .isImage = true
    },
    {
        .binding = 2,
        .type = VK_DESCRIPTOR_TYPE_COMBINED_IMAGE_SAMPLER,
        .imageInfo = &normalInfo,
        .descriptorCount = 1,
        .isImage = true
    },
```

Thank you for an amazing course!



DIGITAL ARTS & ENTERTAINMENT howest / we develop people