SRP-based High Scalability Pipeline

Speaker: 邱天行(walker) 引擎技术负责人

FunPlus

1 Background Introduction

2 LOD-based Hybrid Pipeline

3 Differentiable Production Pipeline

4 Future Plan

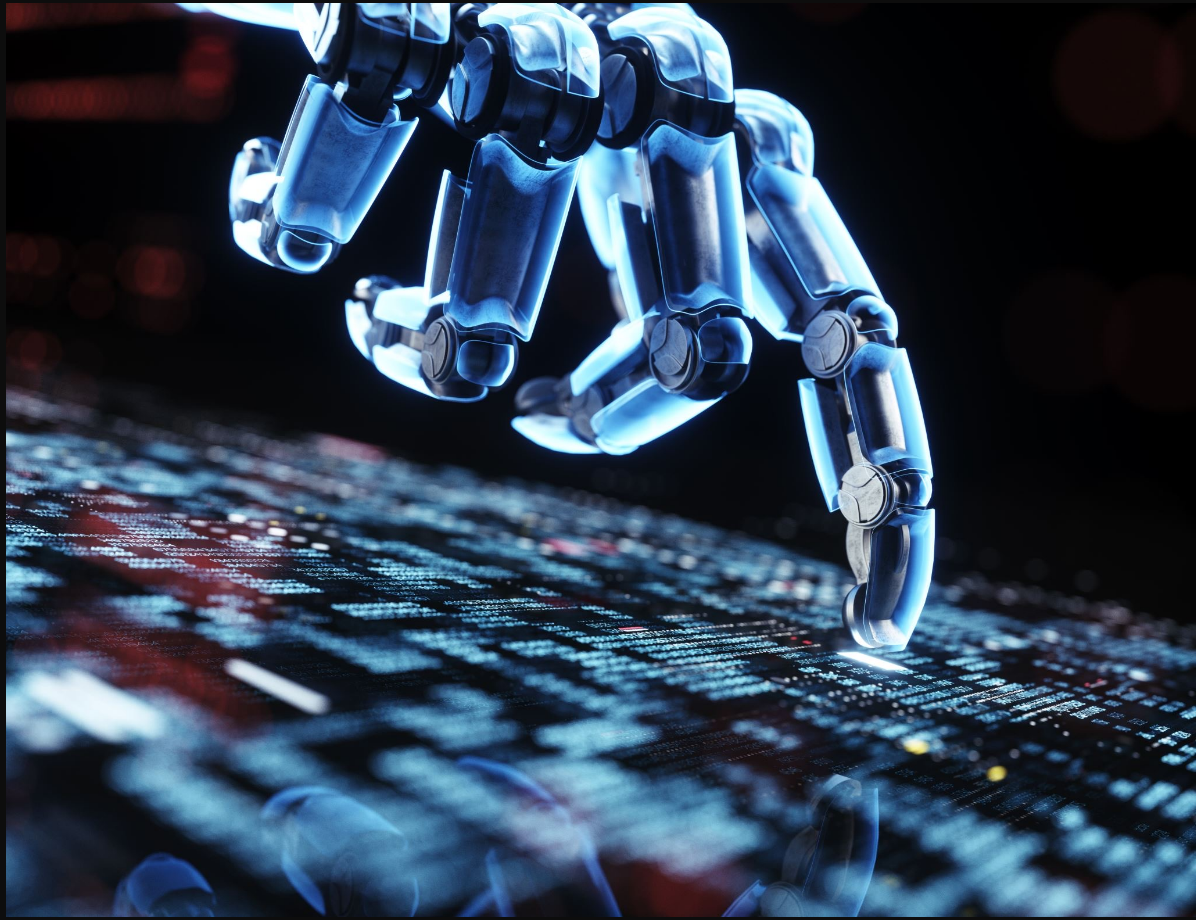
1 Background Introduction

Funplus Overview

- A game development company publishing games worldwide.
- Focusing on high-quality games and user experience.
- Distribution platforms cover PC, console, mobile devices, web games, and more.

The challenges confronting us

- The complexity and vast performance disparities among mobile devices worldwide.
- A multitude of game genres and platforms.
- URP and HDRP perform well on their respective target platforms but lack migration capabilities.
- Production pipelines may more difficult and costly than runtime pipelines.
- How to evaluate the standard each LOD level should achieve.



The solution employed

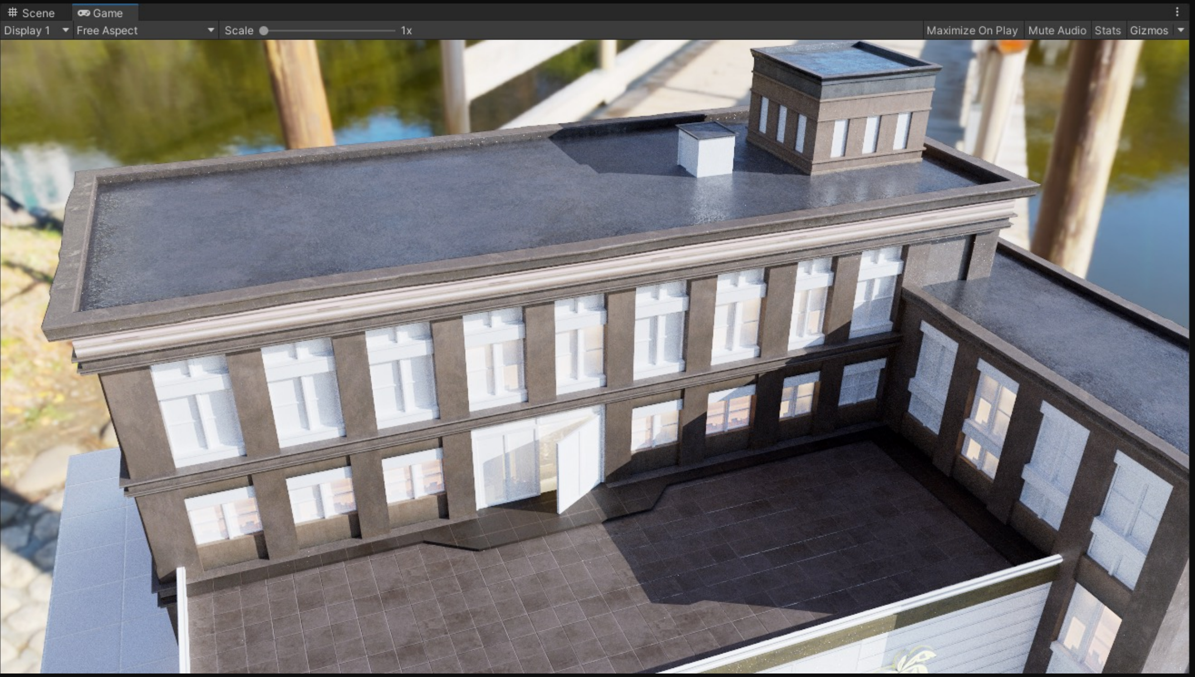
- Implementing a custom SRP pipeline to address cross-platform challenges.
- Focusing on LOD globally, rather than solely on local resources like models and materials.
- Accelerating production and iteration processes using a differentiable generation pipeline, providing information required for runtime pipeline.

② LOD-based Hybrid Pipeline

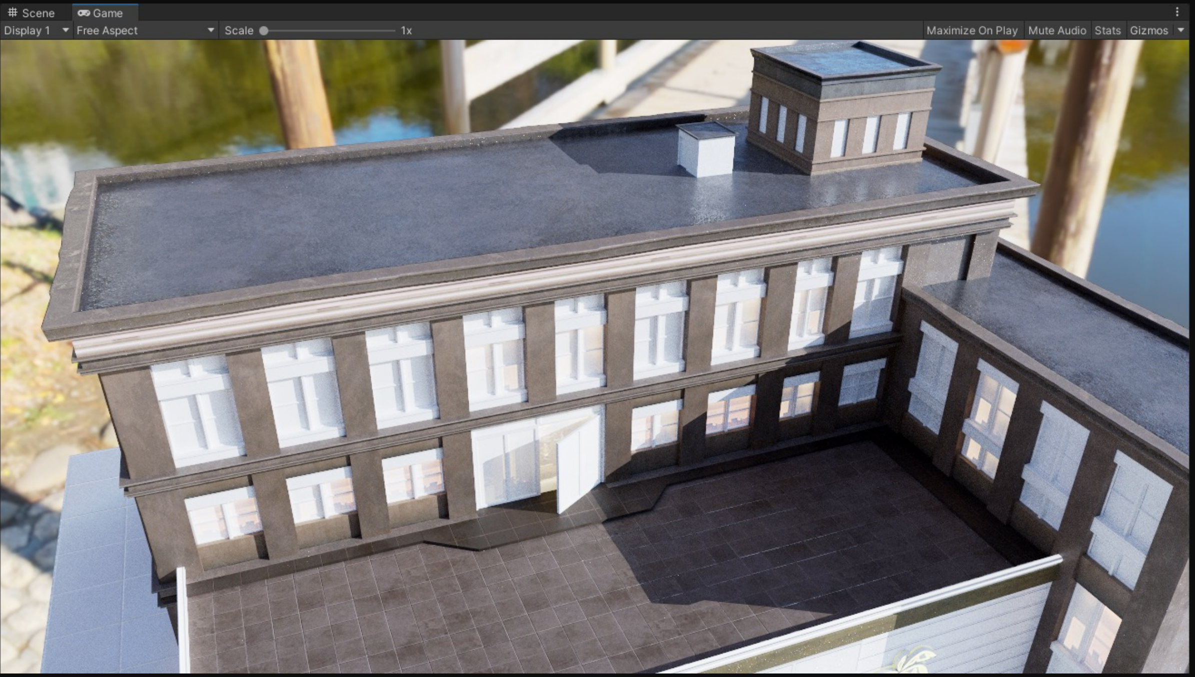
Full pipeline overview

- We employ 4 levels of LOD settings to accommodate various hardware platforms.
- L1: High-end PC and console platforms.
- L2: Low-end PC and high-end mobile devices.
- L3: Common mobile devices.
- L4: Low-end mobile devices and web platforms.

Full pipeline overview

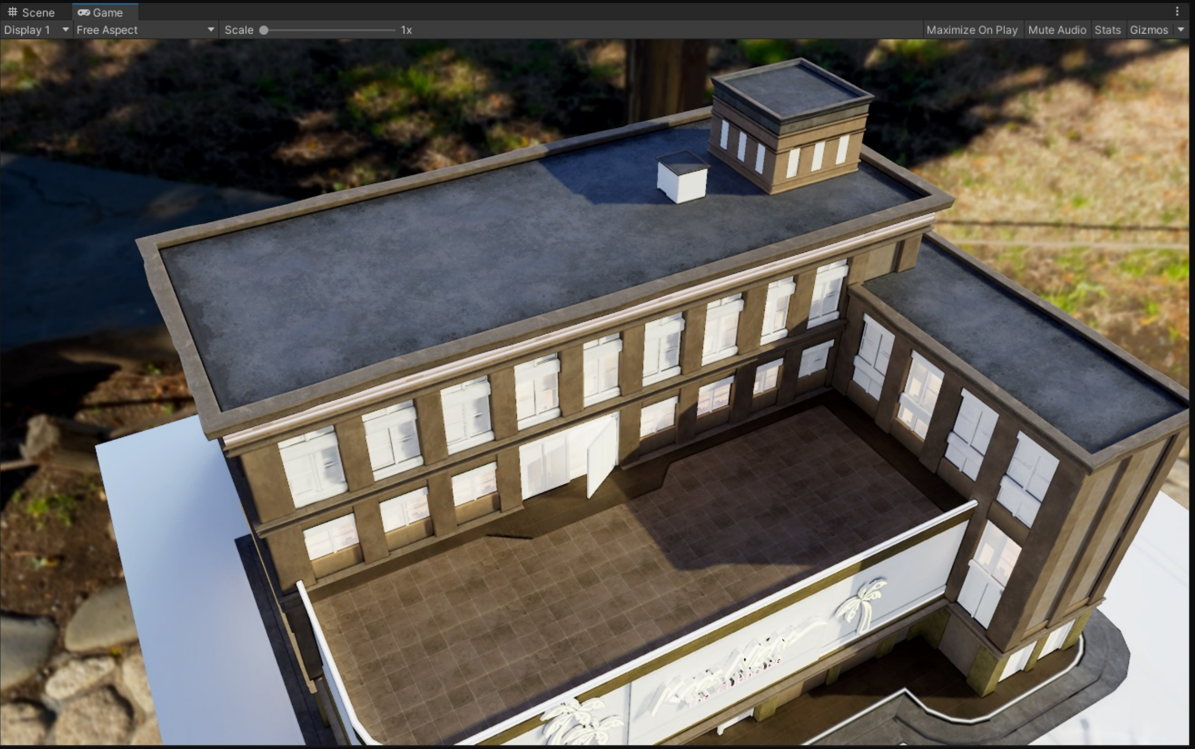


- Due to developing projects, we use a simple demo scene for easier understanding.
- This is a simple building with only one directional light and one skylight in the scene.



L1: High-end PC and console platforms

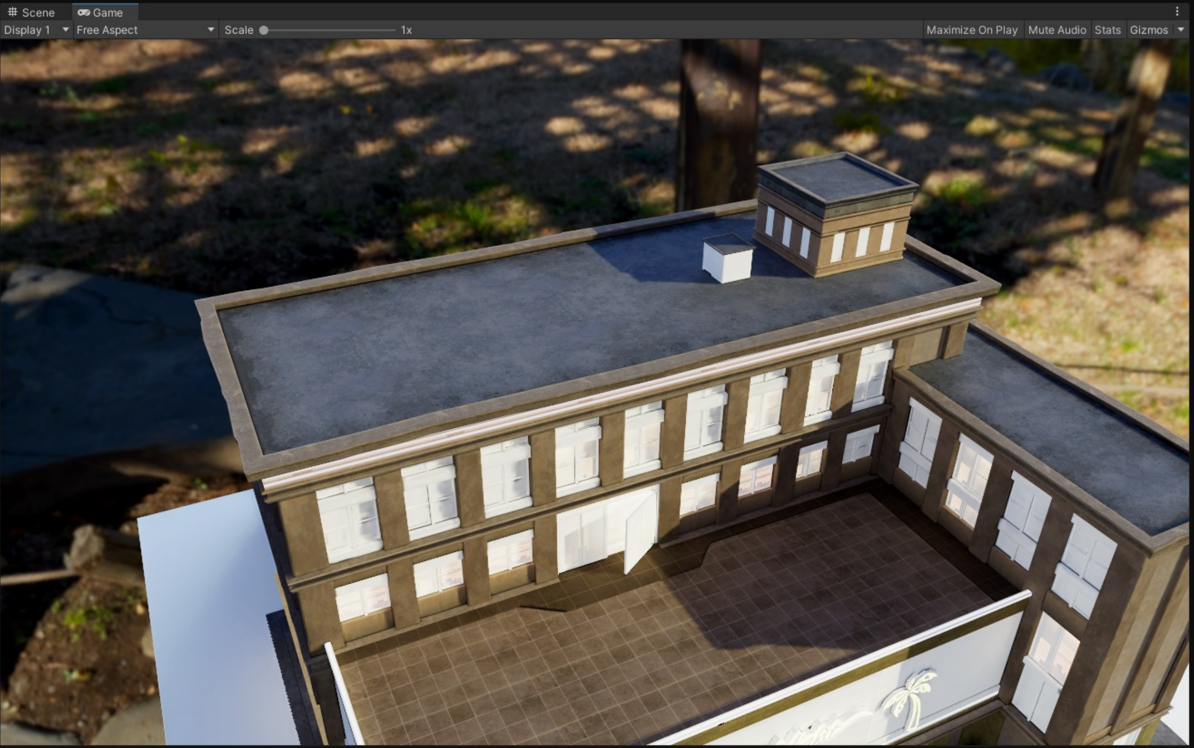
- Focus on rendering and overall quality.
- Take full advantage of runtime ray tracing.(1spp, talk later)
- Deferred shading pipeline.
- Full PBR resources, Utilize PC-oriented resource specifications directly.



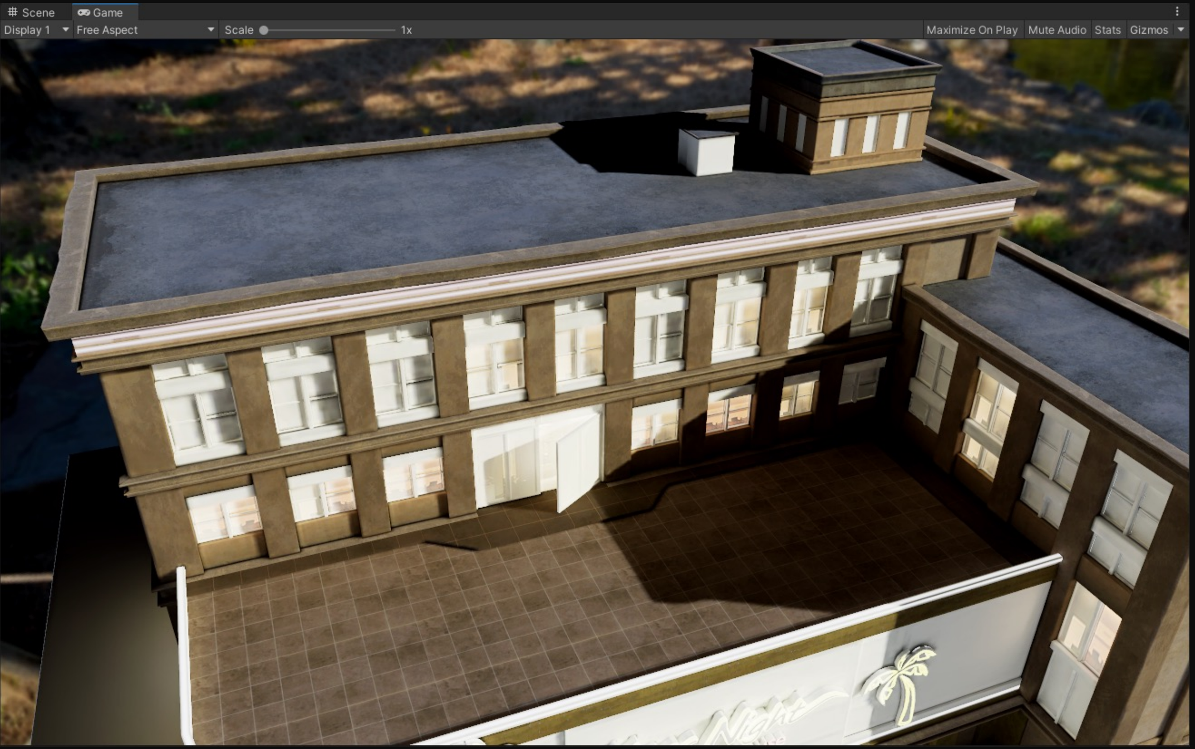
L2: Low-end PC and high-end mobile devices

- Focus on mobile device optimization and the use of modern features.
- Use precomputed data to approximate the rendering effects of the previous tier(We will talk about this later in this section.)
- Hybrid deferred shading pipeline based on different scene types and runtime loads.
- Full PBR with model reconstruction & material lod.(later on dr section)

L3: Common mobile devices



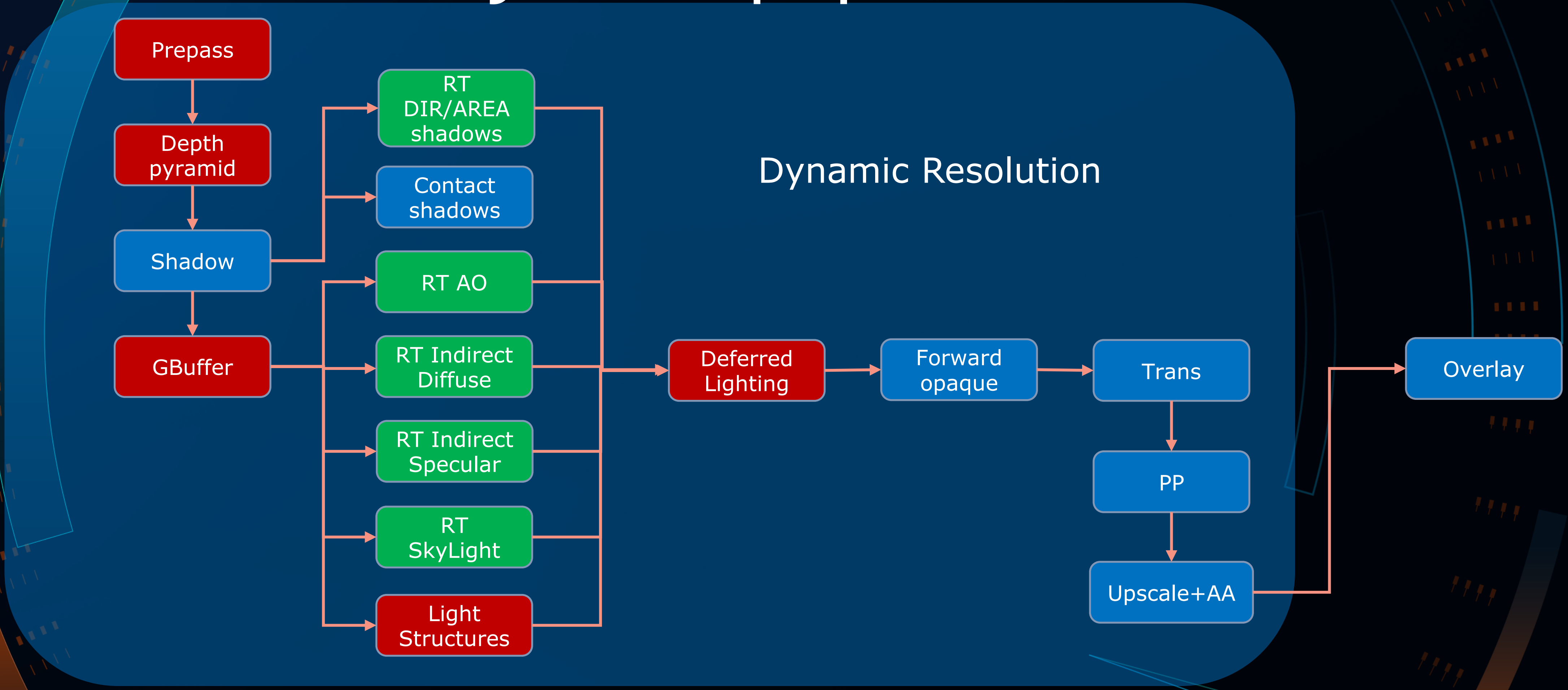
- Focus on mobile device optimization and compatibility.
- Baked lightmap + probes + forward shading.
- Limited postprocess.
- Partial PBR materials, HLOD generated by differentiable pipeline.



L4: Low-end mobile devices and web platforms

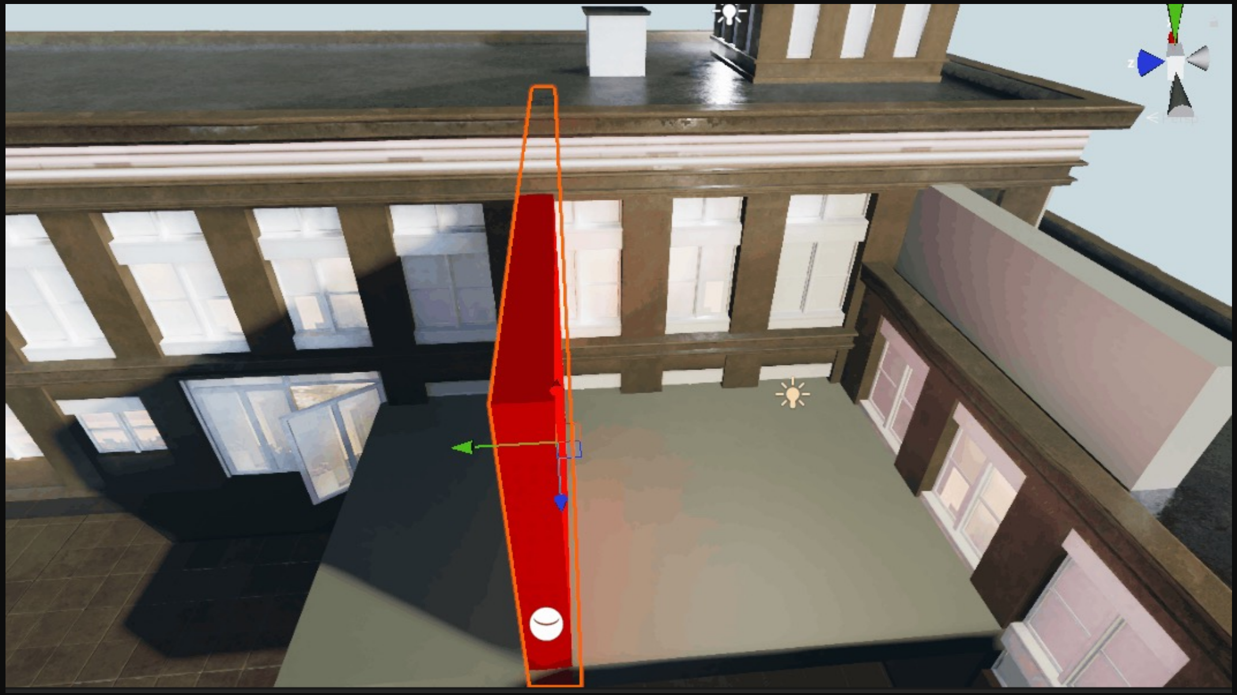
- Focus on Power consumption, frame rate, memory usage, with playability as the ultimate goal.
- Full baked scene with lightmap.
- In more aggressive scenarios, use vertex color baking for the entire scene.
- Non PBR resource.

Hybrid pipeline



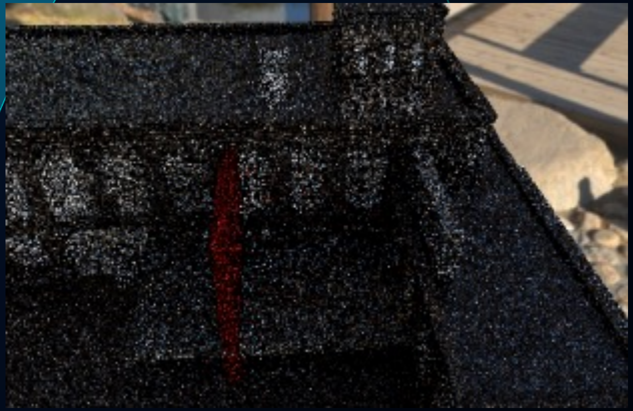
🟥 optional pass 🟩 hybrid pass 🟦 normal pass

Tips: hybrid rt ray tracing

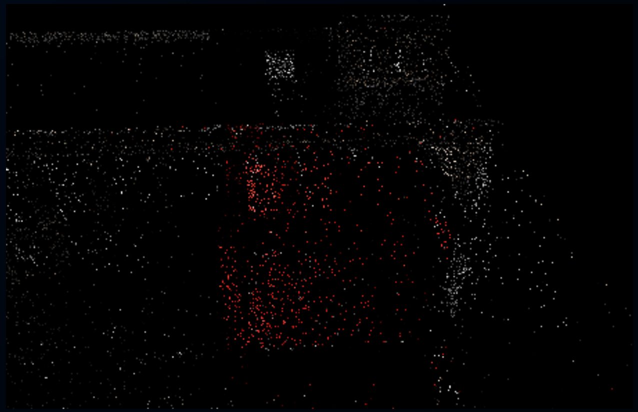


- Focuses on indirect lighting, leaving direct lighting to the forward lighting pipeline.
- 1 spp + Temporal accumulation(ReLax).
- Highly dependent on denoise.
- Apply different denoising algorithms on diffuse, specular, ao, shadow.
- More on nvidia sdk : NRD(NVIDIA Real-Time Denoisers (NRD) | NVIDIA Developer)

Tips: 1spp rt ray tracing

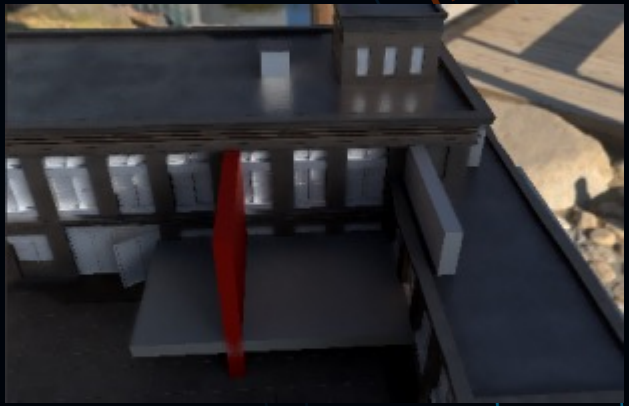


Diffuse(Before)

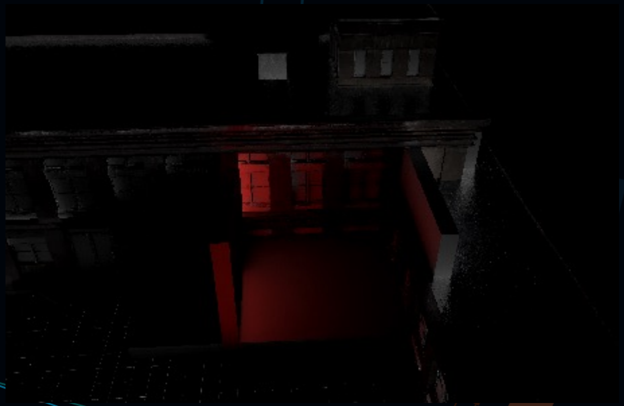


Specular(Before)

1spp ReLax Denoise

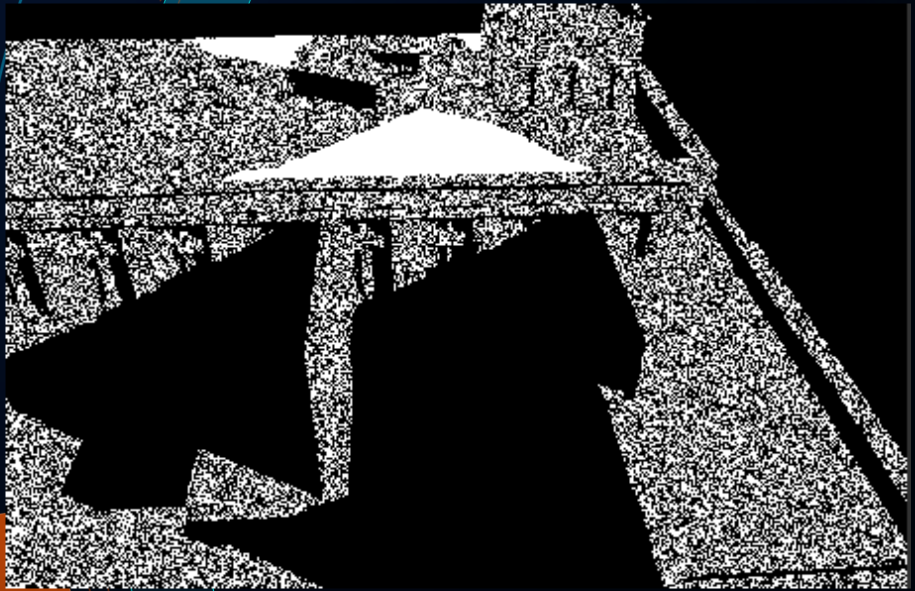


Diffuse(After)

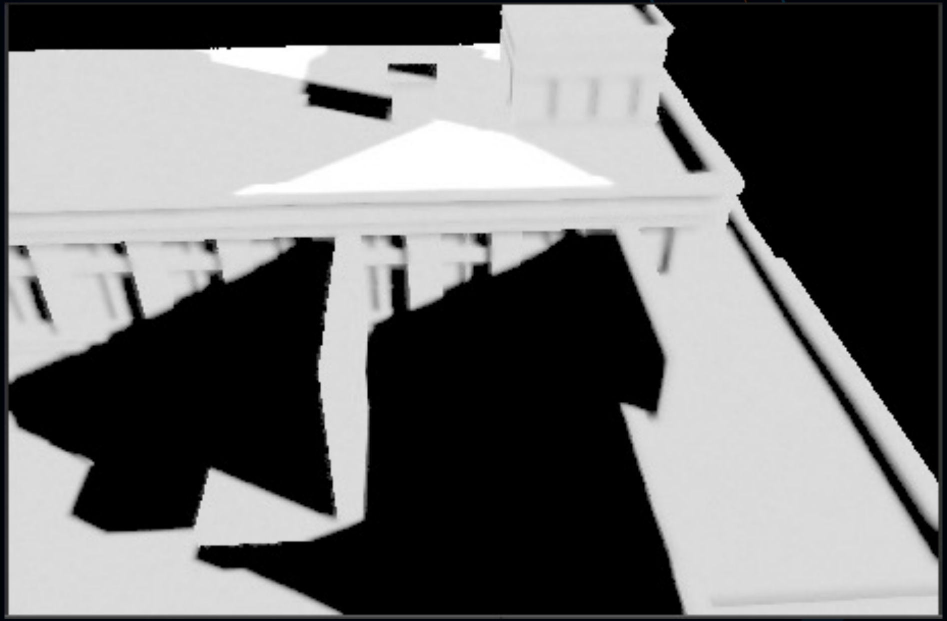


Specular(After)

Tips: 1spp rt ray tracing



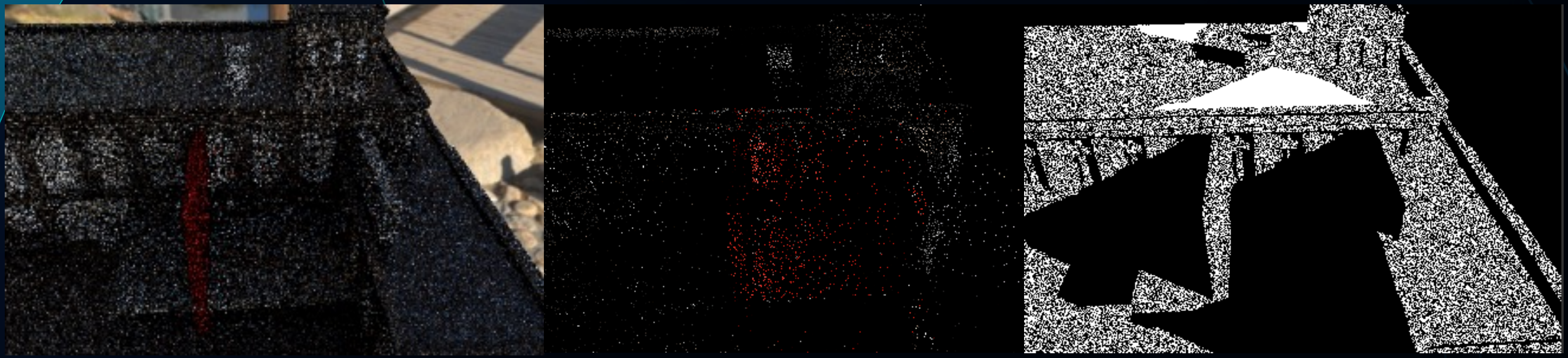
1spp ReLax Denoise



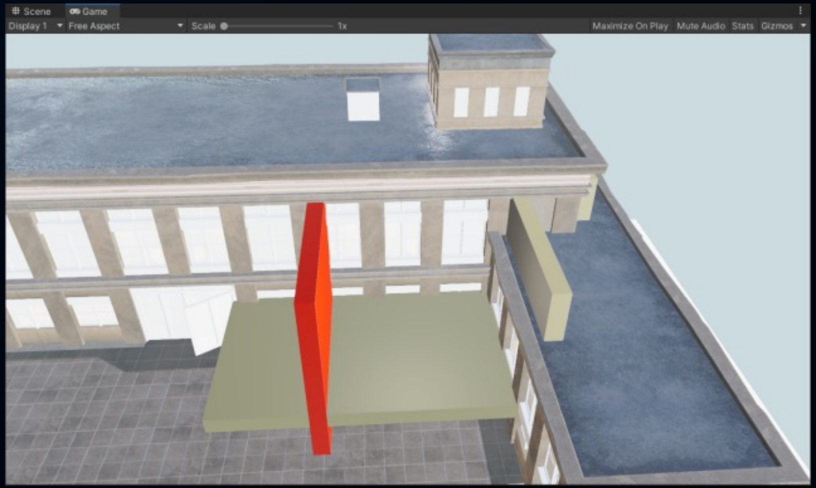
Shadow(Before)

Shadow(After)

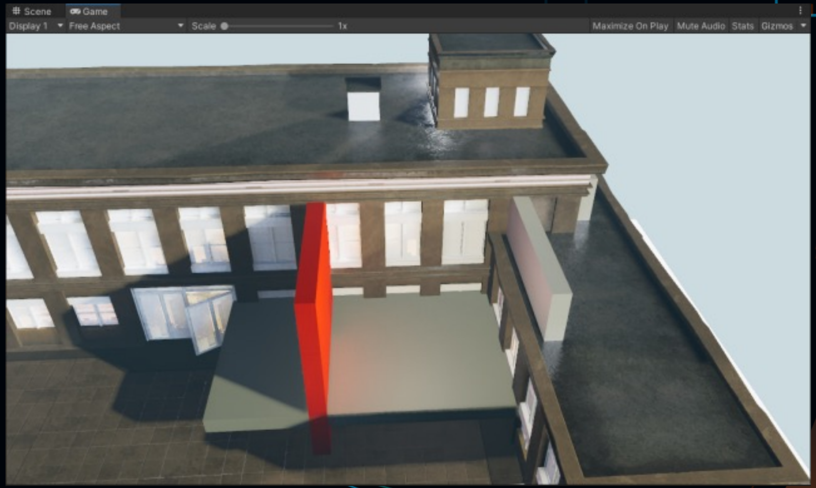
Tips: 1spp rt ray tracing

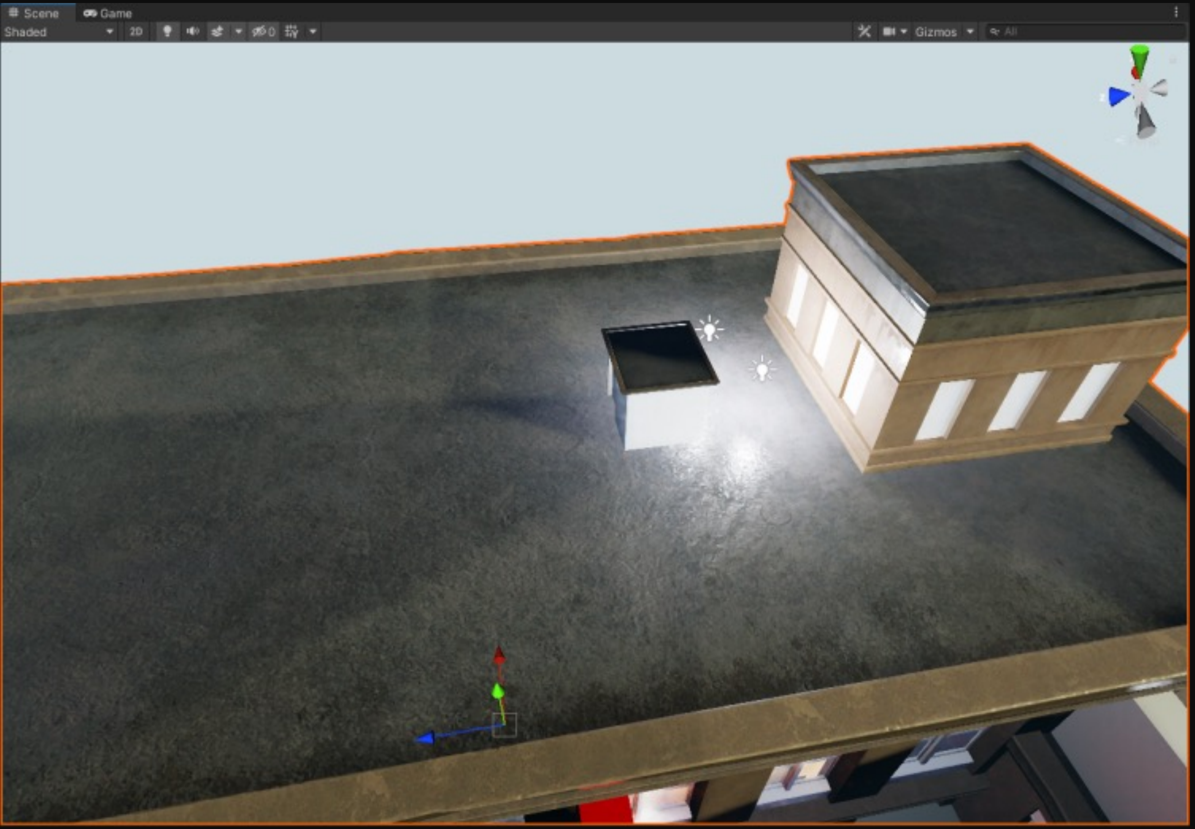


+



=

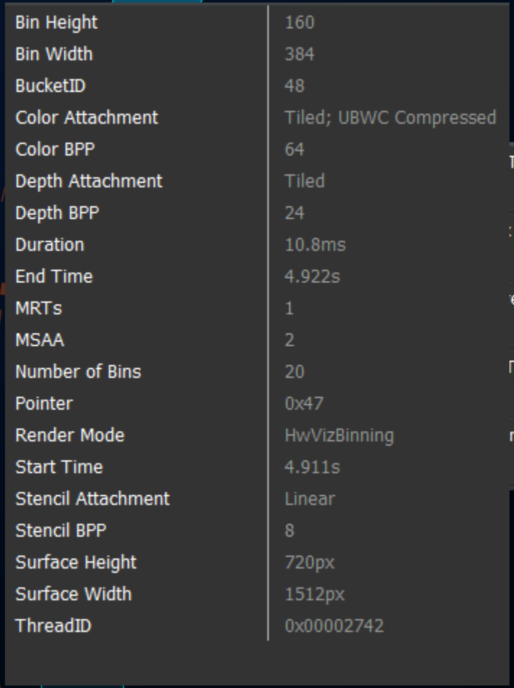




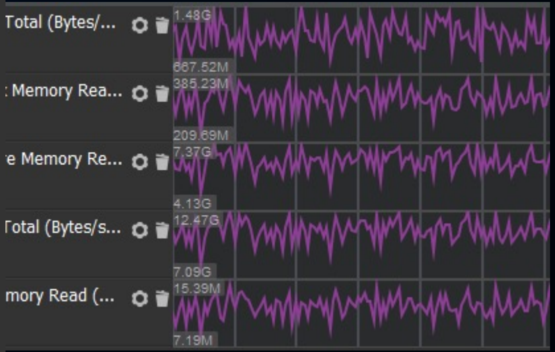
Tips: direct lighting

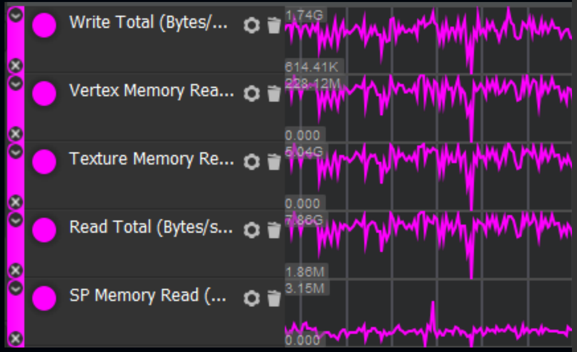
- Deferred and forward are not conflicting.
- Utilize dynamic pipeline to switch between deferred and forward rendering.
- The core optimization lies in the utilization of on-chip memory.
- Dependent on MRT framebuffer fetch(snapdragon) or PLS(mali).
- 192bit Bandwidths limit.

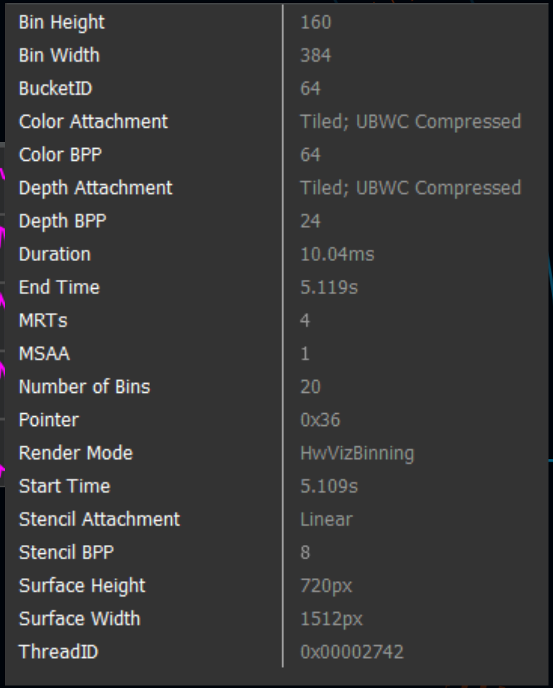
Tips: direct lighting



Bin Height	160
Bin Width	384
BucketID	48
Color Attachment	Tiled; UBWC Compressed
Color BPP	64
Depth Attachment	Tiled
Depth BPP	24
Duration	10.8ms
End Time	4.922s
MRTs	1
MSAA	2
Number of Bins	20
Pointer	0x47
Render Mode	HwVizBinning
Start Time	4.911s
Stencil Attachment	Linear
Stencil BPP	8
Surface Height	720px
Surface Width	1512px
ThreadID	0x00002742



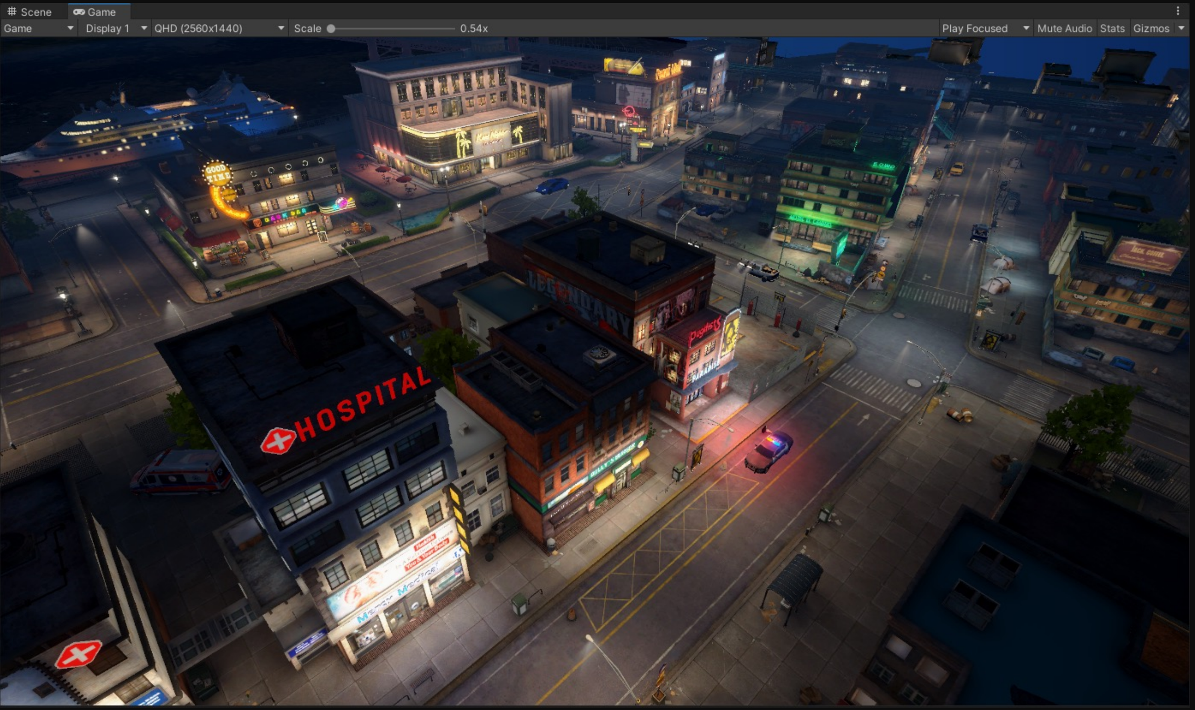




Bin Height	160
Bin Width	384
BucketID	64
Color Attachment	Tiled; UBWC Compressed
Color BPP	64
Depth Attachment	Tiled; UBWC Compressed
Depth BPP	24
Duration	10.04ms
End Time	5.119s
MRTs	4
MSAA	1
Number of Bins	20
Pointer	0x36
Render Mode	HwVizBinning
Start Time	5.109s
Stencil Attachment	Linear
Stencil BPP	8
Surface Height	720px
Surface Width	1512px
ThreadID	0x00002742

Forward

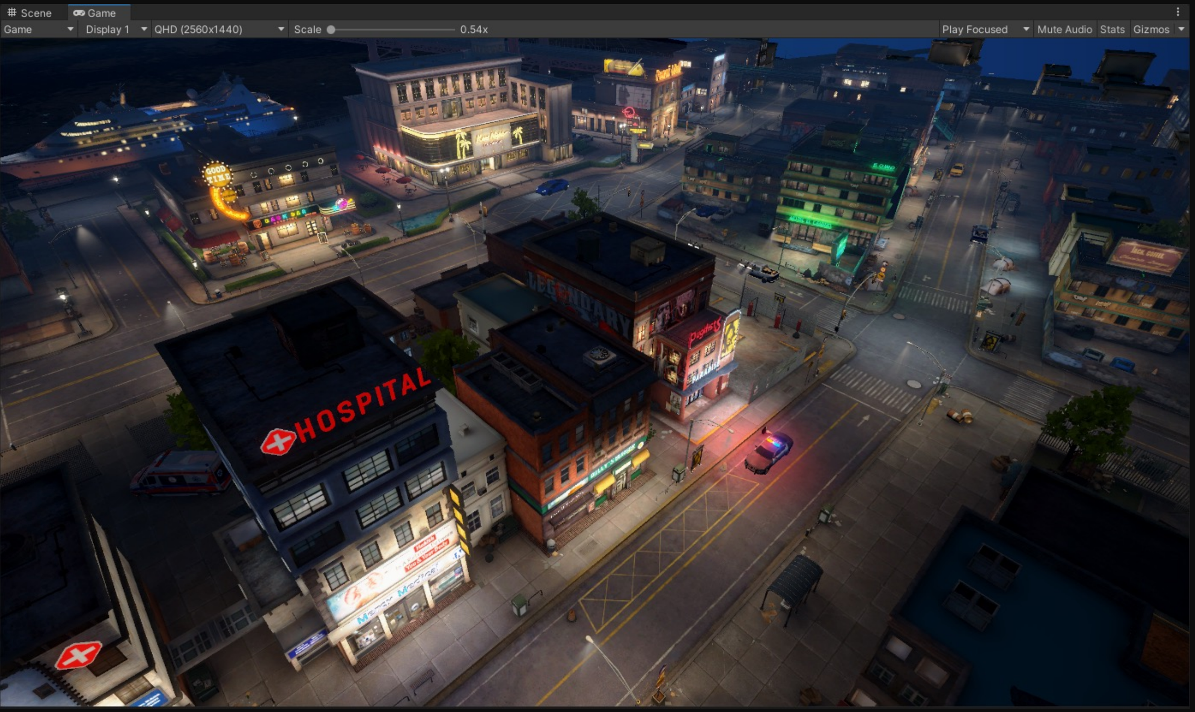
Deferred



Tips: lightmap + PRT probes/SH probes

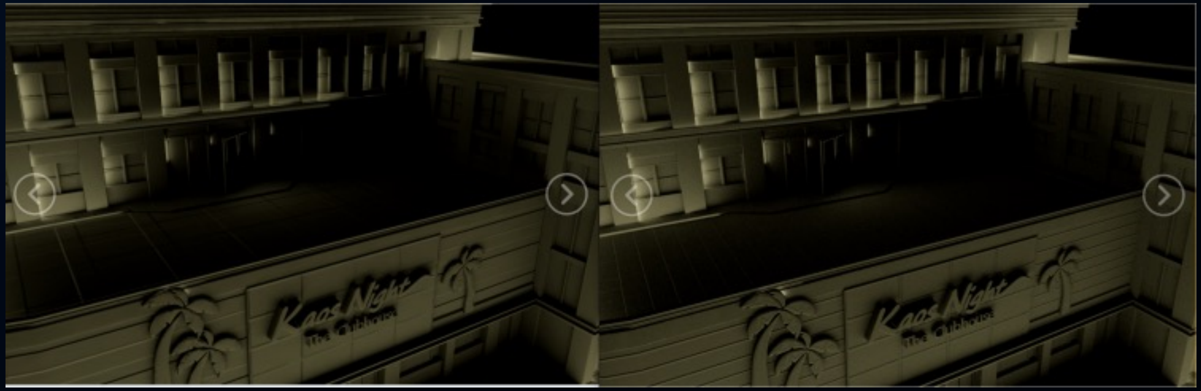
- Supports various lightmap modes, such as full-baked, indirect, AHD, etc.
- PRT in cases.
- Probe supports per-pixel interpolation mode and per-object interpolation mode.
- Baking by custom baking tool.

Tips: Custom baking tool

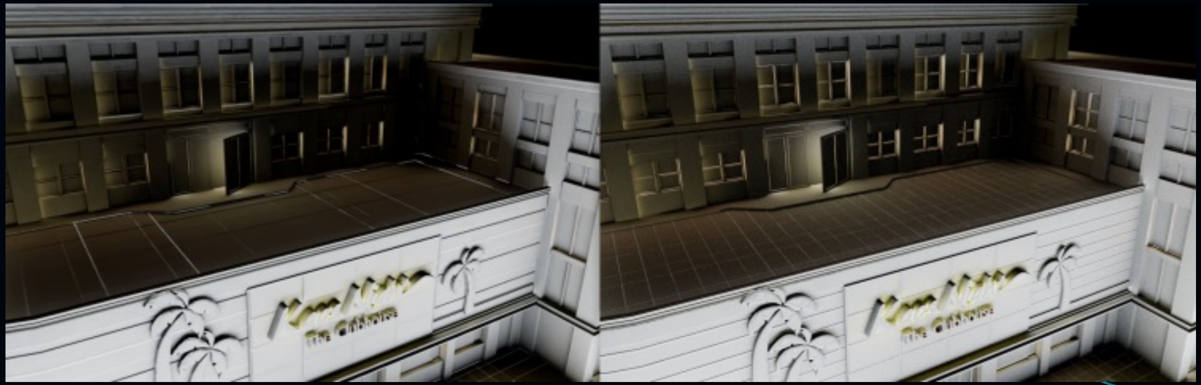


- Quality improvement on lightmap mode.
- Producible data types include: lightmap, sh, prt matrix, ao, etc.
- Preview mode supported by runtime raytracing Approximation.
- Utilizes OptiX framework and CUDA, integrated as an external tool, to unify usage scenarios across multiple engines.

Tips:baking tool preview mode

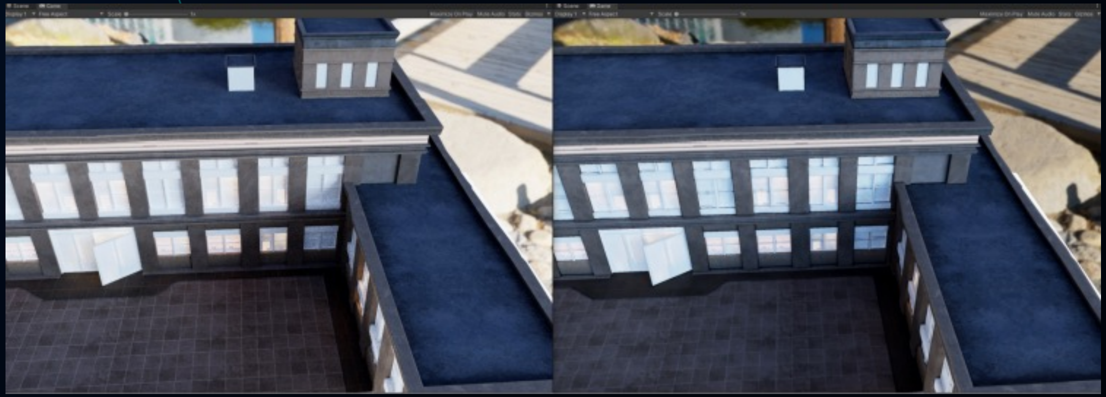


Direct lighting preview vs baked result

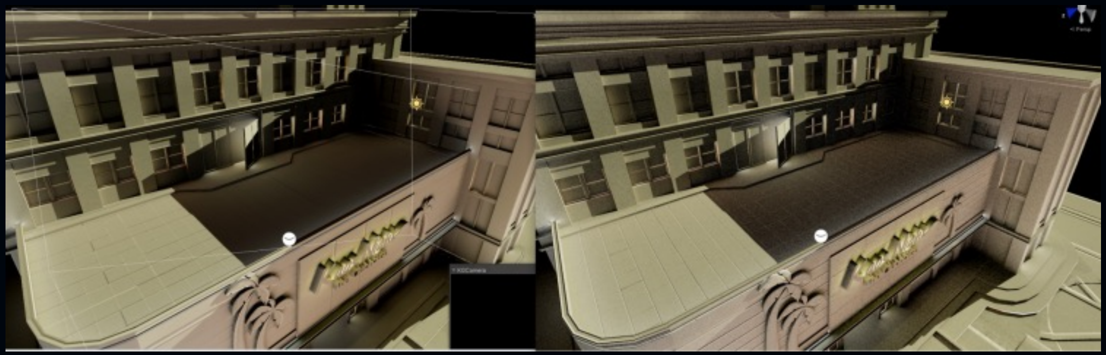


indirect lighting preview vs baked result

Tips:baking tool preview mode

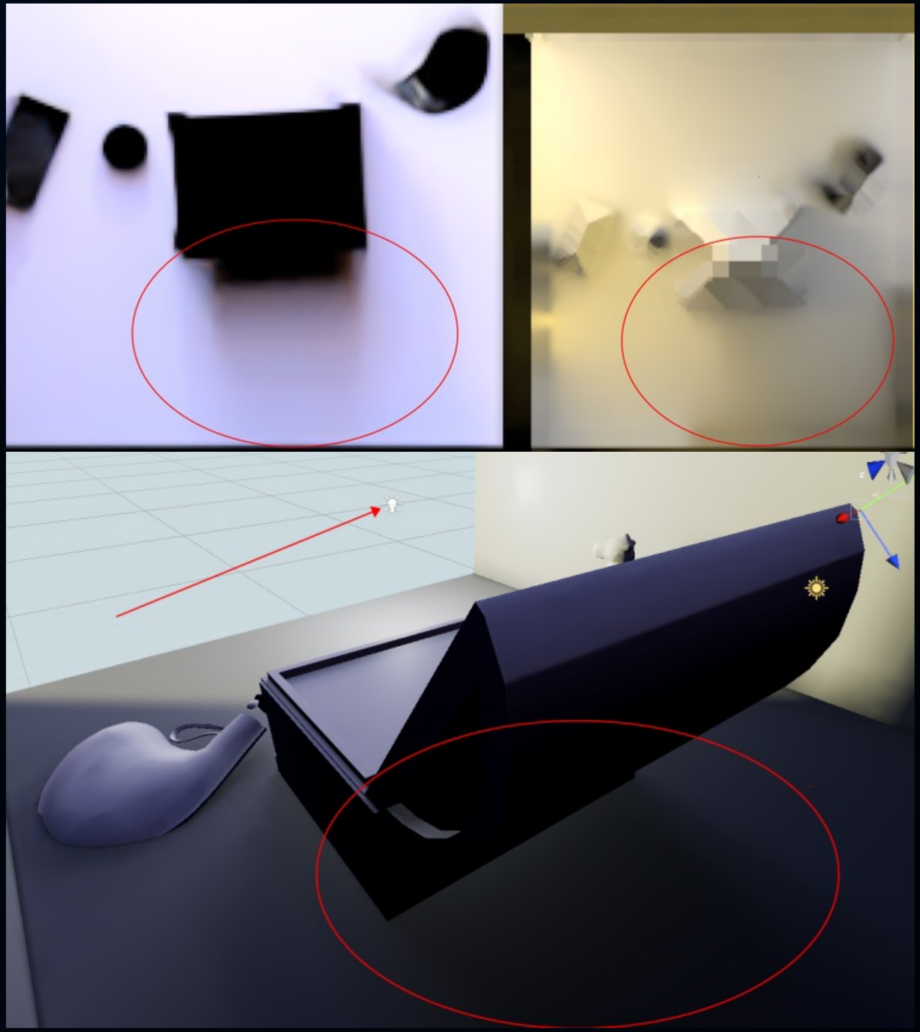


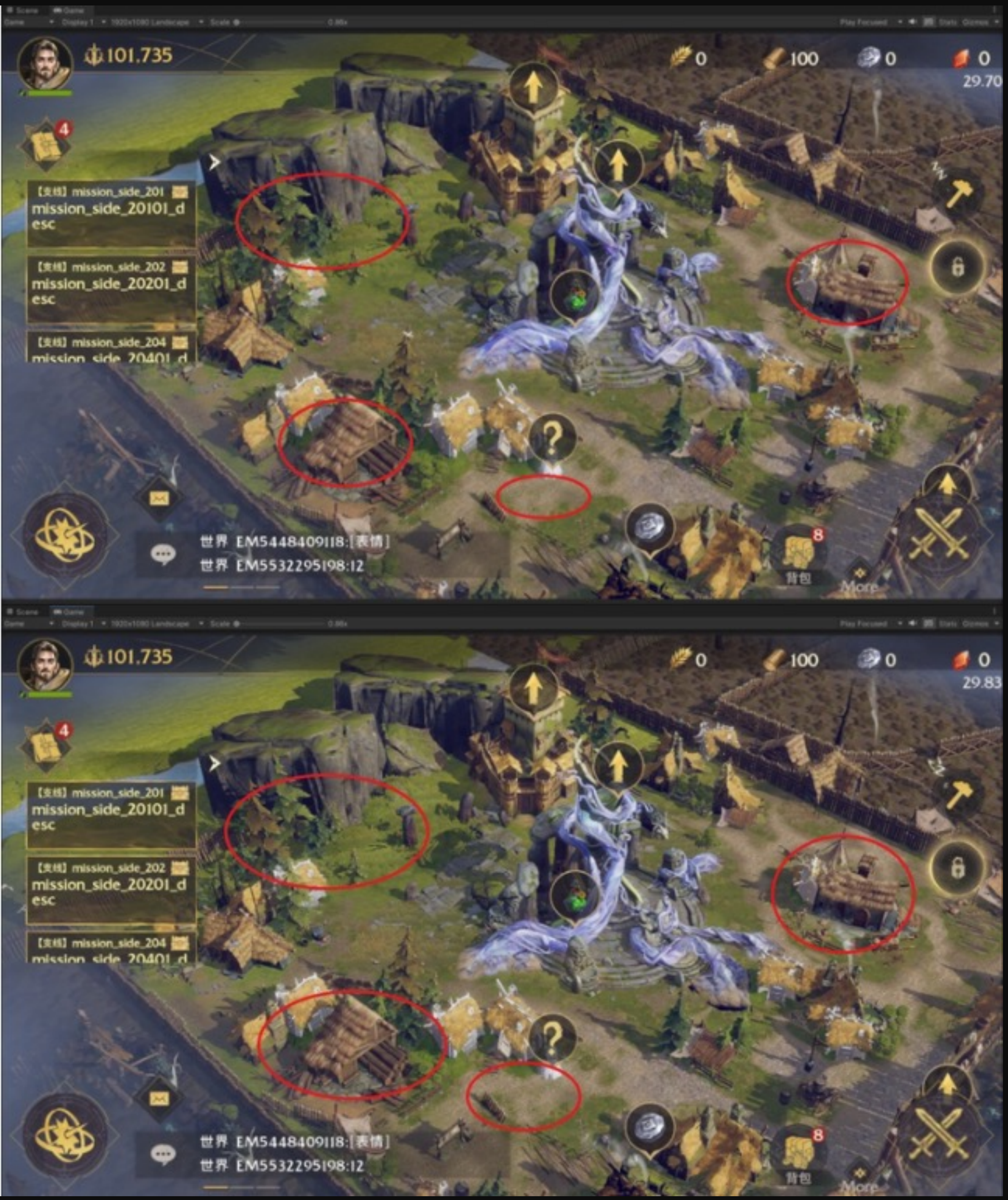
sky lighting preview vs baked result



Final result comparation

Tips:baking tool quality improvement

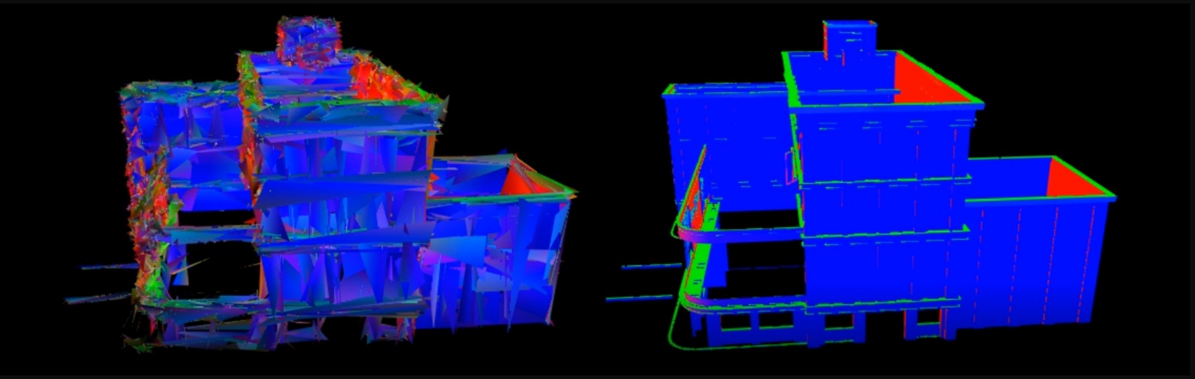




Tips: Dynamic RS + Upscale

- Adjusts dynamic resolution based on current frame rate and game logic feedback.
- Minimizing frame rate fluctuations and rendering jitter are the biggest challenges of dynamic resolution technology.
- We use nvidia framework NIS(NVIDIA Image Scaling SDK) for upscaling on mobile platform.
- DLSS3 is generally a good choice on the PC platform.
- TAA can be done with upscale in same pass.

③ Differentiable Production Pipeline

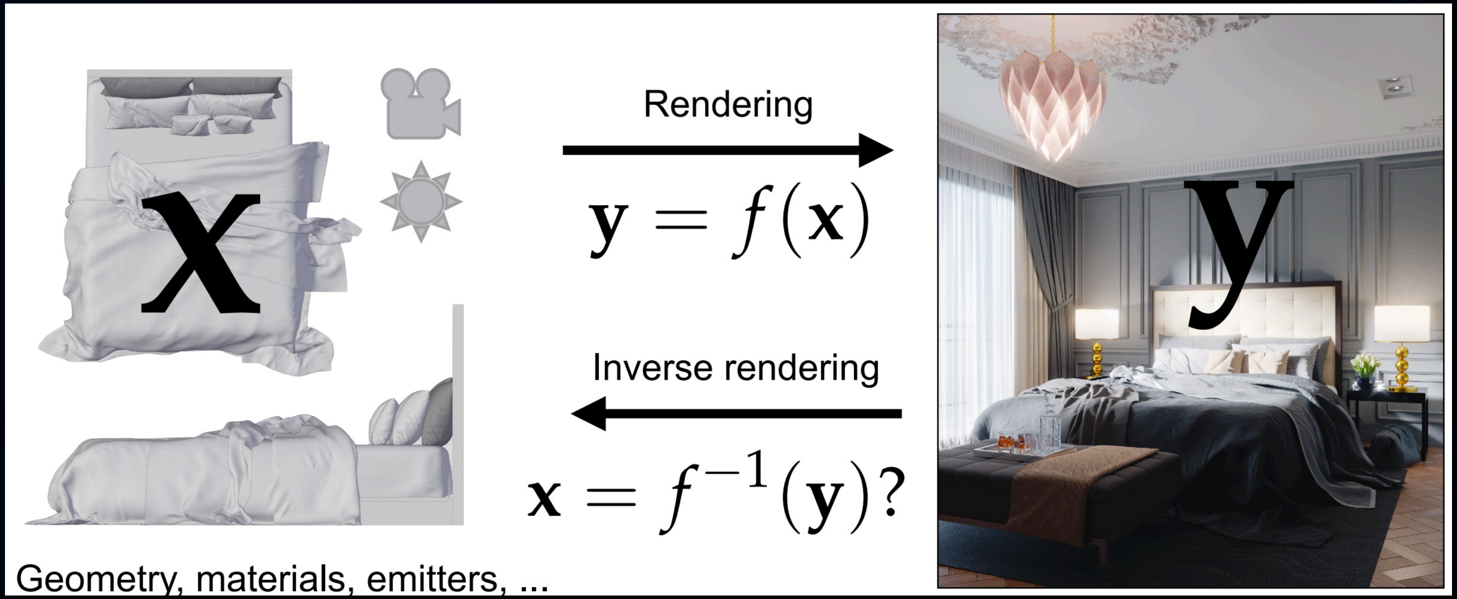


Why differentiable pipeline?

- LOD is the most fundamental requirement for scalable pipelines.
- “Evaluative capabilities” are currently the most lacking aspect in LOD production pipelines.
- LOD is not just about model reduction, current pipelines have limited capabilities in this regard.
- In addition to LOD, we can integrate modern AIGC capabilities into the pipeline for continuous iteration in style transfer and scene creation.

Fundamentals

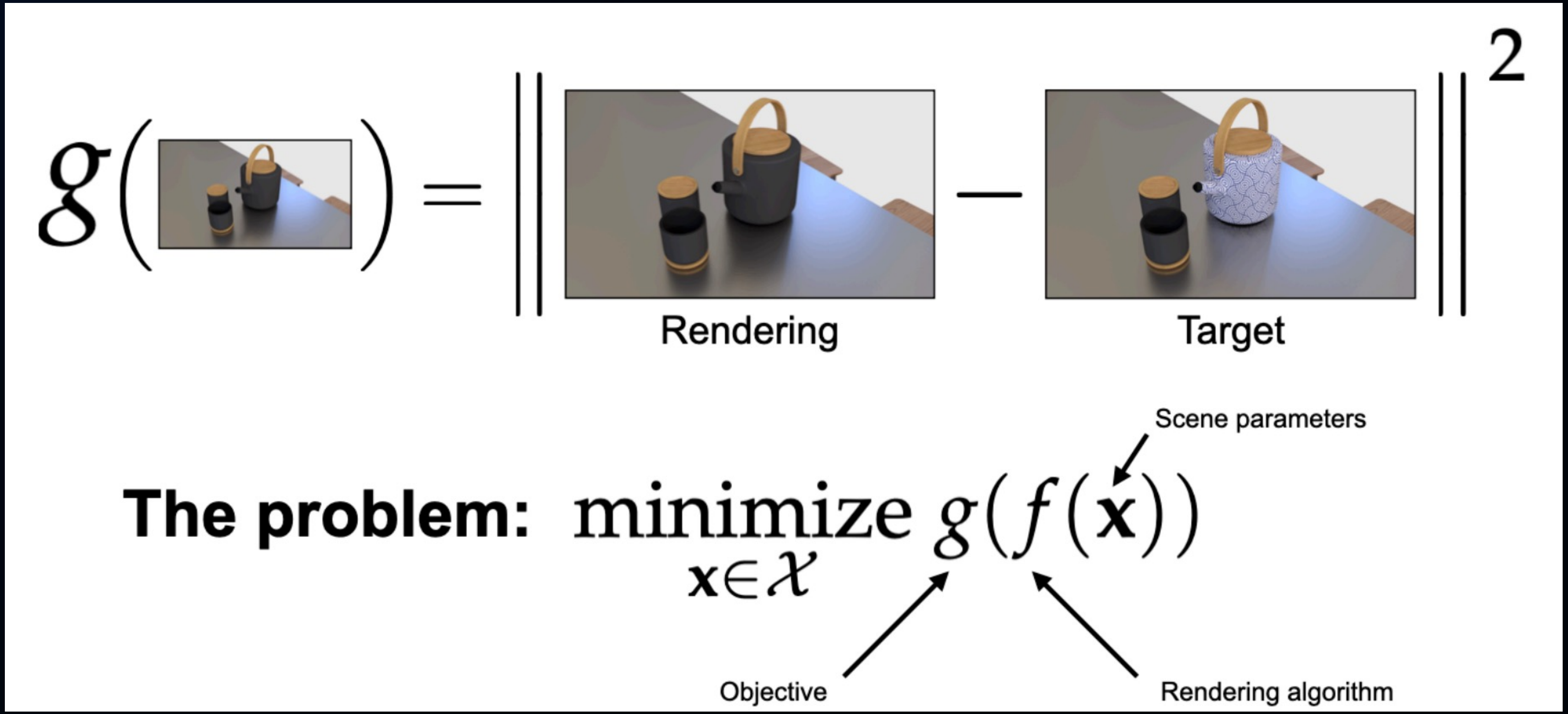
- A function is differentiable at a point if its change at that point can be approximated by its derivative (slope) at that point.



$\mathbf{x}$
Rendering
$\mathbf{y} = f(\mathbf{x})$
Inverse rendering
$\mathbf{x} = f^{-1}(\mathbf{y})?$
$\mathbf{y}$
Geometry, materials, emitters, ...

Ref: PBDR in SIGGRAPH 2020

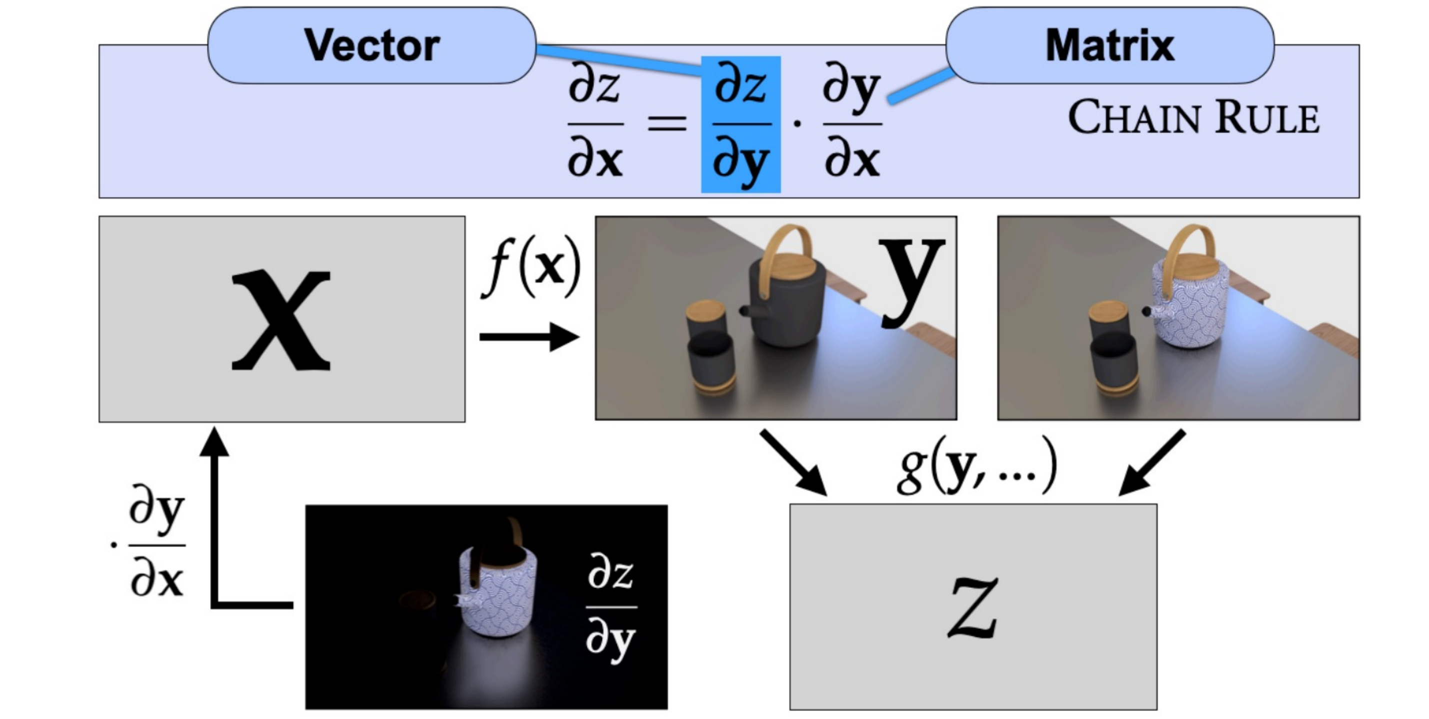
Fundamentals



The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$

Ref: PBDR in SIGGRAPH 2020

Fundamentals

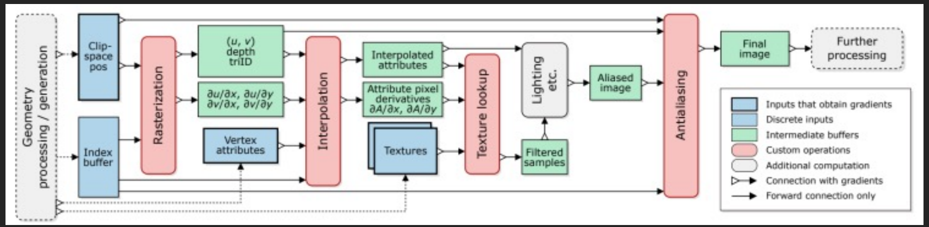


Vector
Matrix
$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
CHAIN RULE
$\mathbf{x}$
$f(\mathbf{x})$
$\mathbf{y}$
$g(\mathbf{y}, \ldots)$
$\cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\frac{\partial z}{\partial \mathbf{y}}$
z

Ref: PBDR in SIGGRAPH 2020

Framework Adopted

- There are typically two mainstream approaches to choose from: rasterization-based and path tracing-based.
- We employ a hybrid pipeline to capitalize on the strengths of each approach.
- Nvdiff as rasterization-based backend.
- Suba3 as path-tracing backend.
- As an external program for Unity, connected through the import and export processes.



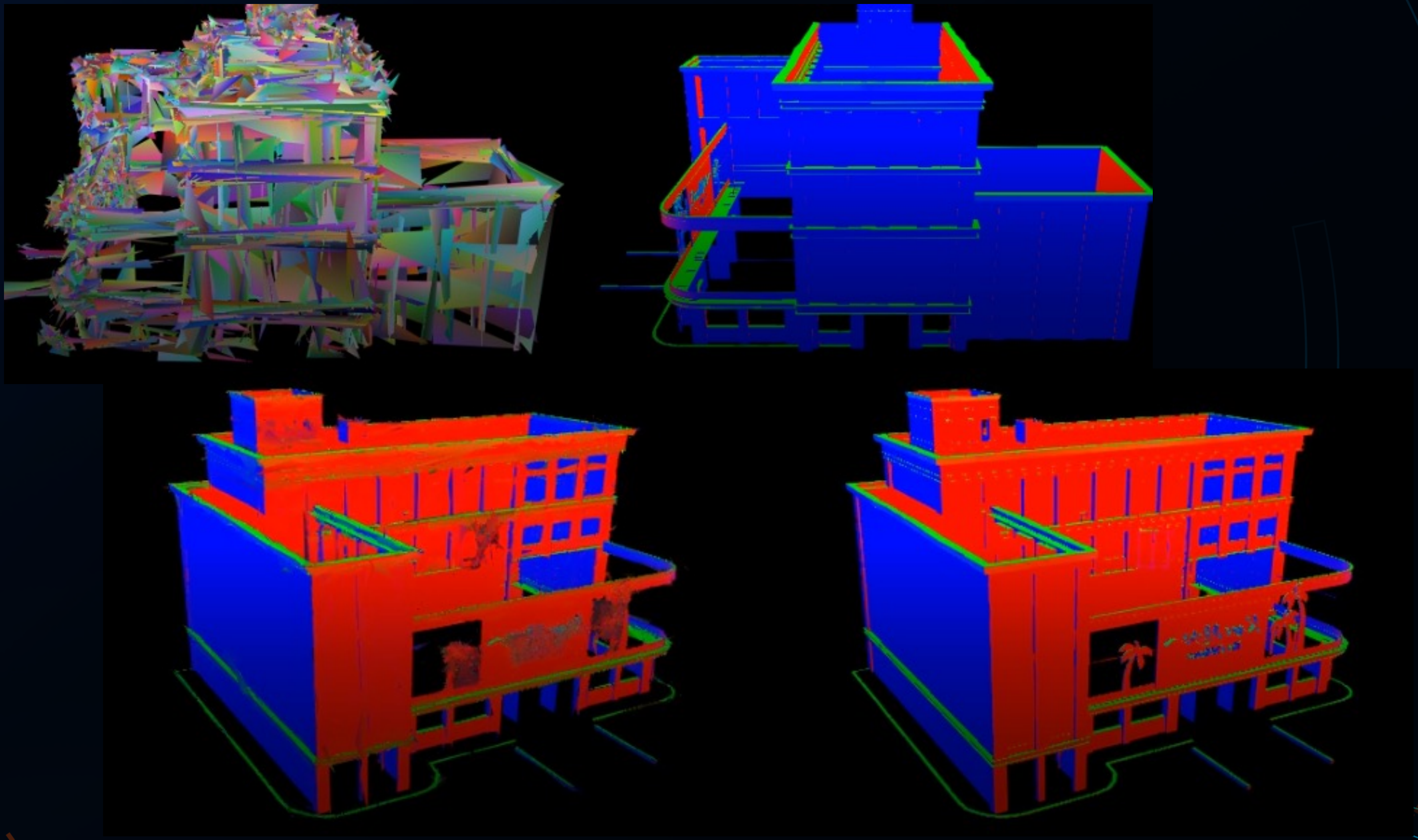
Some implementation details

- Using the architecture diagram from NVDIFF, we build upon this process for our custom extensions.

Implementation details

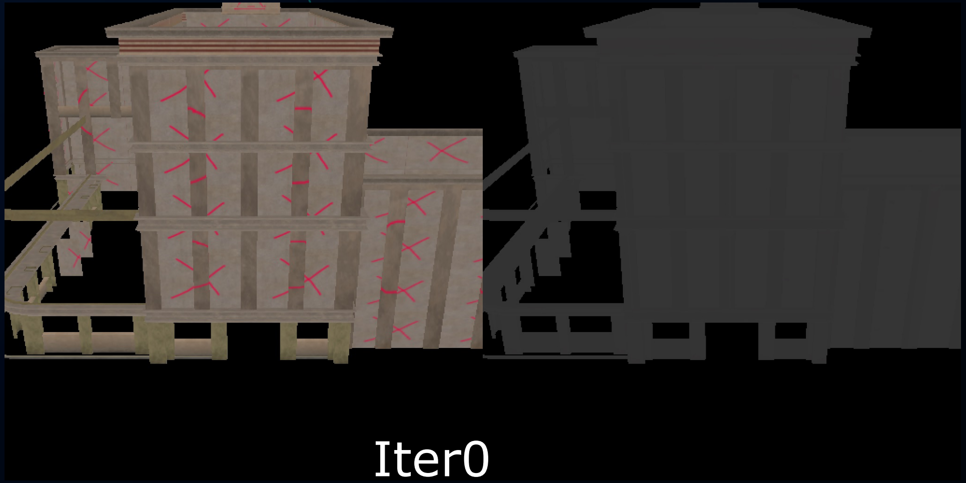
- Utilizing USD as the scene interchange format.
- Use PyTorch + cuda as AD backend.
- Current modeling parameters include: vertex information, texture information, standard PBR material parameters, lighting information, camera information, and some post-processing information.
- In a single iteration, the rasterization process can be completed within milliseconds, while the ray tracing process takes close to 1 second; continuous optimization is ongoing.

Some simple demonstrations aid in intuitive understanding.

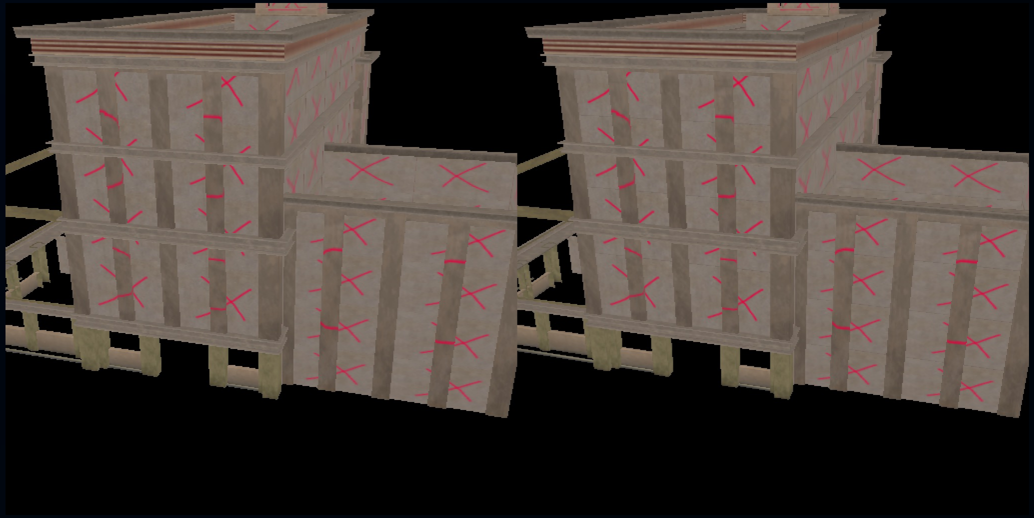


Mesh opt

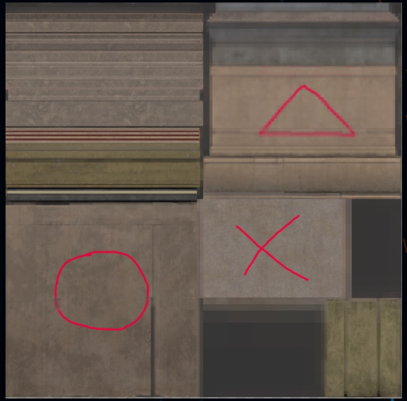
Some simple demonstrations aid in intuitive understanding.



Iter0



Iter100



Texture opt

Texture origin

More on materials

Time limit…

④ Future Planning

Integrate with other AIGC-related workflows

- SD + DR?
- Rapid scene style transfer.
- Scene prototype development.
- Integrate with existing PCG pipeline for semi-automatic generation.

Thanks