LIGHTSPEED STUDIOS | GDC

INNOVATION UNLEASHED:

High-Performance UE5 Mobile Rendering and Next-Gen Character Creation Pipeline Powered by Machine Learning

LIGHTSPEED STUDIOS

A LEADING GLOBAL GAME DEVELOPER

LightSpeed Studios is one of the world's most **innovative** and **successful** game developers, with teams across China, United States, Singapore, Canada, United Kingdom, France, Japan, South Korea, New Zealand and United Arab Emirates.

Founded in 2008, LightSpeed Studios has created over **50 games** across multiple platforms and genres for more than **4 billion registered users**. It is the co-developer of the worldwide hit **PUBG MOBILE** (co-developed with KRAFTON, Inc.).

LightSpeed Studios is made up of passionate players who advance the art & science of game development through **great stories, great gameplay, and advanced technology**. We are focused on bringing next generation experiences to gamers who want to enjoy them anywhere, anytime, across multiple genres and devices.

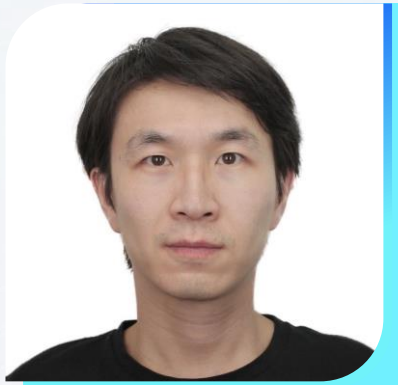
SPEAKERS

JING GONG

Principal Software Engineer
Lead
LIGHTSPEED STUDIOS

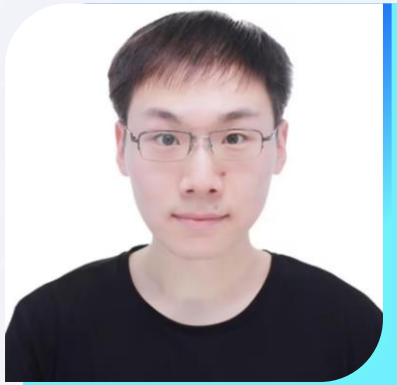
BO LI

Principal Software Engineer
LIGHTSPEED STUDIOS



XIN QIAO

Senior Software Engineer
LIGHTSPEED STUDIOS



QUAN WEN

Senior Software Engineer
LIGHTSPEED STUDIOS

LIGHTSPEED STUDIOS | GDC

Innovation Unleashed (1)

Practical Mobile Rendering in UE5

LIGHTSPEED STUDIOS | GDC

OUTLINE

PART.01 Gong Jing - Practical Mobile Deferred Shading

PART.02 Bo Li - Practical Mobile GPU-driven Pipeline

LIGHTSPEED STUDIOS | GDC

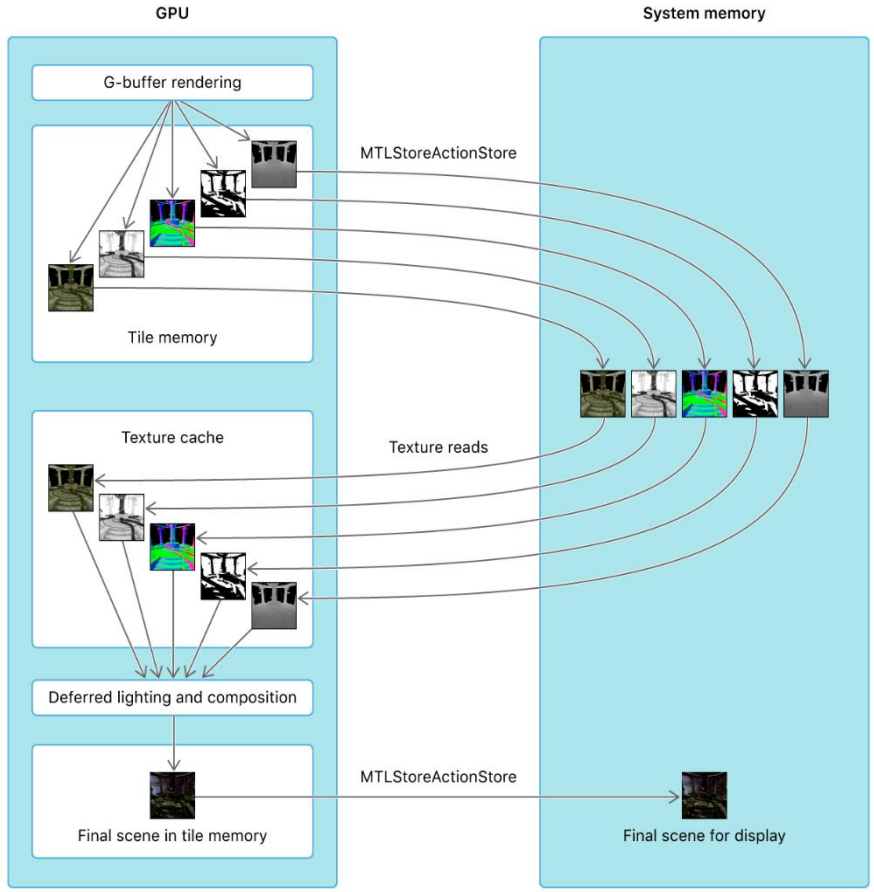
PART.01

PRACTICAL MOBILE DEFERRED SHADING

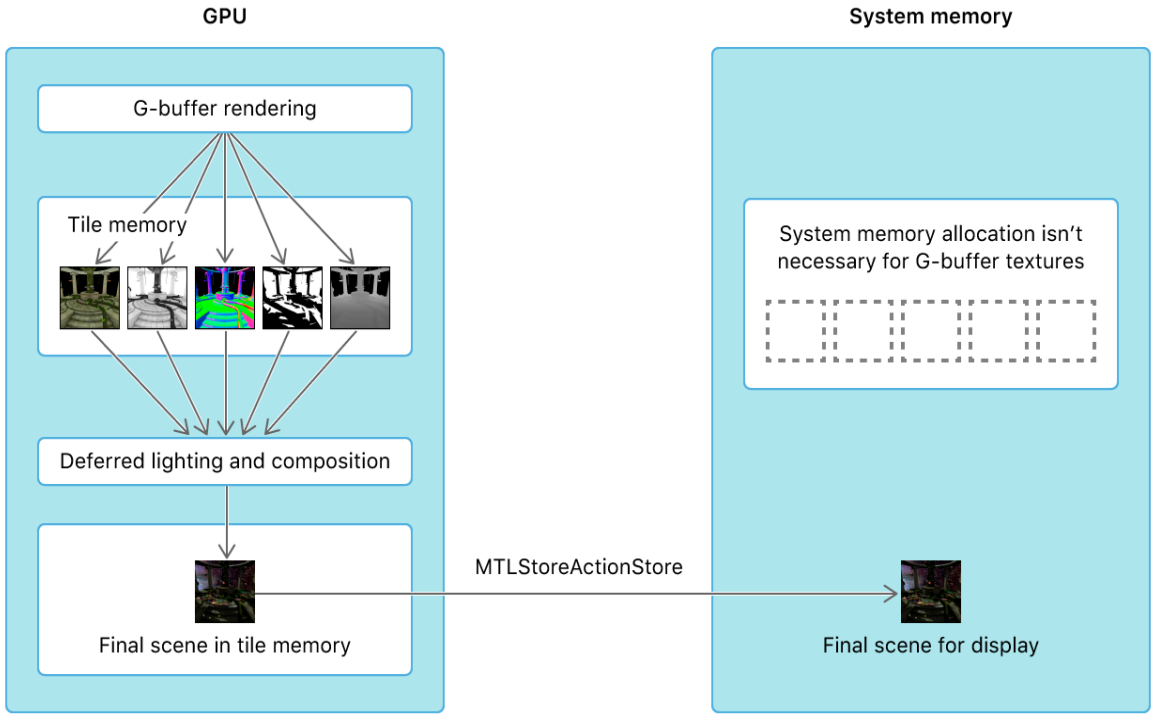
- Introduction
- Improvements to Mobile Deferred Renderer

INTRODUCTION

UE5 - Mobile Deferred Shading



Multi-Pass Deferred Shading(PC)



Single-Pass Deferred Shading(Mobile)

INTRODUCTION

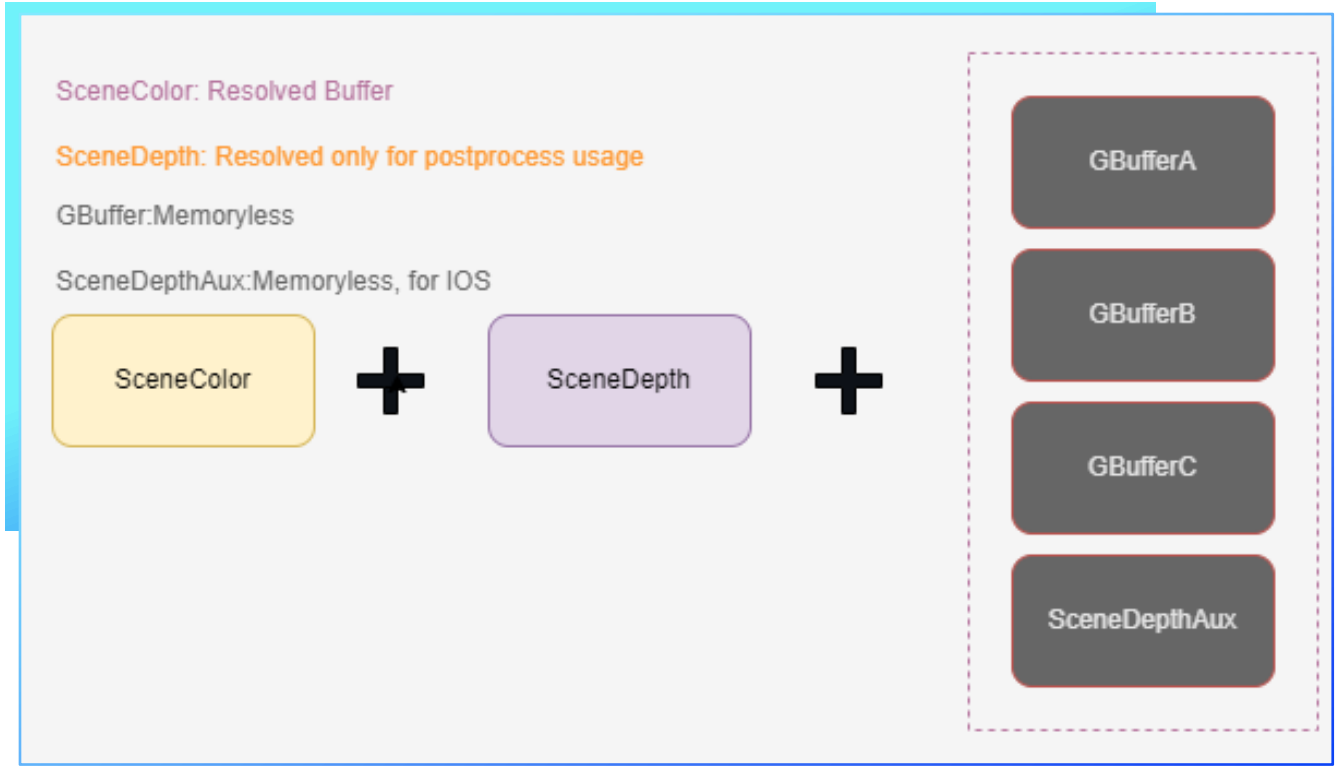
UE5 - Mobile Deferred Shading

G-Buffer on tiled memory

- Metal
  - FrameBufferFetch
- Vulkan
  - SubpassFetch
- GLES
  - Mali
    - PixelLocalStroage
  - Adreno
    - FrameBufferFetch

Device compatibility

- 128 bits pixel local storage on Mali GLES
- The limit for Vulkan 4 input attachments limit(16.3%)



INTRODUCTION

UE5 - Mobile Deferred Shading

Other Features

- Multiple shading models
  - Compress G-Buffer for different shading models
- Clustered deferred shading
- Deferred Decal
- SSAO/GTAO/FXAA/TAA
- IES Profile
- Light Function
- etc

INTRODUCTION

UE5 - Mobile Deferred Shading

Issues

- Multiple shading models are not supported when static lighting is enabled
  - Due to the limitation of G-buffer
- Multiple lighting channels are not supported
- Many-Lights shadows are not supported

INTRODUCTION

Performance issues caused by multiple shading models

- Branch divergence
- More VGPRs

```
FDirectLighting IntegrateBxDF( FGBufferData GBuffer, half3 N, half3 V, half3 L, float Falloff, half NoL, FAreaLight AreaLight, FShadowTerms Shadow )
{
    switch( GBuffer.ShadingModelID )
    {
        case SHADINGMODELID_DEFAULT_LIT:
        case SHADINGMODELID_SINGLELAYERWATER:
        case SHADINGMODELID_THIN_TRANSLUCENT:
            return DefaultLitBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_SUBSURFACE:
            return SubsurfaceBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_PREINTEGRATED_SKIN:
            return PreintegratedSkinBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_CLEAR_COAT:
            return ClearCoatBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_SUBSURFACE_PROFILE:
            return SubsurfaceProfileBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_TWOSIDED_FOLIAGE:
            return TwoSidedBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_HAIR:
            return HairBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_CLOTH:
            return ClothBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        case SHADINGMODELID_EYE:
            return EyeBxDF( GBuffer, N, V, L, Falloff, NoL, AreaLight, Shadow );
        default:
            return (FDirectLighting)0;
    }
}
```

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Improvements to Mobile Deferred Renderer

- Improved G-Buffer Layout
- Performance Optimizations
- Many-Lights Shadows

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Improved G-Buffer Layout

Support multiple shading models when static lighting is enabled

- More aggressive compression for G-Buffer

Support multiple lighting channels

- Store lighting channel mask in G-buffer

Support anisotropy shading model

- Store tangent and anisotropy in G-buffer

Optimize performance issues arising from multiple shading models

- Store ShadingModelID in stencil buffer

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Improved G-Buffer Layout

- DefaultLit
  - Encode base color as approximate sRGB to give more precision to the darks
    - FBF/PLS doesn't always work with sRGB

Name	Format	R	G	B	A
SceneColorMobile	R11G11B11_Float	Emissive/Lightmap R	Emissive/Lightmap G	Emissive/Lightmap B	
GBufferA	R10G10B10A2_Unorm	OctahedronNormal.X	OctahedronNormal.Y	IndirectIrradiance * AO	PerObjectGBufferData
GBufferB	R8G8B8A8_Unorm	Metallic	Specular	Roughness	LightingChannelMask
GBufferC	R8G8B8A8_Unorm	Sqrt(BaseColor.R)	Sqrt(BaseColor.G)	Sqrt(BaseColor.B)	PrecomputedShadowFactor

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Improved G-Buffer Layout

- Shading Model - Subsurface
  - No space for precomputed shadow factors

Name	Format	R	G	B	A
SceneColorMobile	R11G11B11_Float	Emissive/Lightmap R	Emissive/Lightmap G	Emissive/Lightmap B	
GBufferA	R10G10B10A2_Unorm	OctahedronNormal.X	OctahedronNormal.Y	Opacity_Specular(6:4)	Metallic
GBufferB	R8G8B8A8_Unorm	SubsurfaceColor.R&LightingChannelMask0	SubsurfaceColor.G&LightingChannelMask1	Roughness	SubsurfaceColor.B&LightingChannelMask2
GBufferC	R8G8B8A8_Unorm	Sqrt(BaseColor.R)	Sqrt(BaseColor.G)	Sqrt(BaseColor.B)	IndirectIrradiance*AO

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Improved G-Buffer Layout

- Shading Model – Anisotropy

Name	Format	R	G	B	A
SceneColorMobile	R11G11B11_Float	Emissive/Lightmap R	Emissive/Lightmap G	Emissive/Lightmap B	
GBufferA	R10G10B10A2_Unorm	OctahedronNormal.X	OctahedronNormal.Y	Specular_Metallic(6:4)	PrecomputedShadowFactor
GBufferB	R8G8B8A8_Unorm	OctahedronTangent.X	OctahedronTangent.Y	Roughness	Anisotropy_LightingChannelMask(5:3)
GBufferC	R8G8B8A8_Unorm	Sqrt(BaseColor.R)	Sqrt(BaseColor.G)	Sqrt(BaseColor.B)	IndirectIrradiance*AO

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Improved G-Buffer Layout

- Stencil Buffer Layout
- [1] sandbox bit, for light stencil culling
- [2-5] shading model ID
- [6] sky mask
- [7] SSAO mask
- [8] primitive receive decal bit

[1]	[2, 5]	[6]	[7]	[8]
sandbox bit	shading model ID	sky mask	SSAO mask	primitive receive decal bit

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Optimization - Multiple Shading Models

UE5

- Calculate lighting using 2 passes

```
for (light : lights)
{
    // pass stencil test if ShadingModelID is MSM_DefaultLit
    render_light(light);
    // pass stencil test if ShadingModelID is not MSM_DefaultLit
    render_light(light);
}
```

Our solution

- Calculate lighting for each shading model separately

```
for (light : lights)
    for (shading_model_id : light.shading_models)
    { // pass stencil test if ShadingModelID in stencil buffer is equal
      render_light(light);
    }
```

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Optimization - Multiple Shading Models

Our Solution

- Pros
  - Better shader performance
    - Avoid shader divergence
    - Higher GPU occupancy

Cons

- Need more draw calls
  - Solution: Use clustered deferred shading for local lights

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Optimization - Calculate direct lighting and indirect lighting separately

Indirect Lighting

- Sky lighting
- Reflections (IBL)
  - Deferred Reflection Probe Blending
- GI(Optional)

Direct Lighting

- Directional Lights
- Local Lights

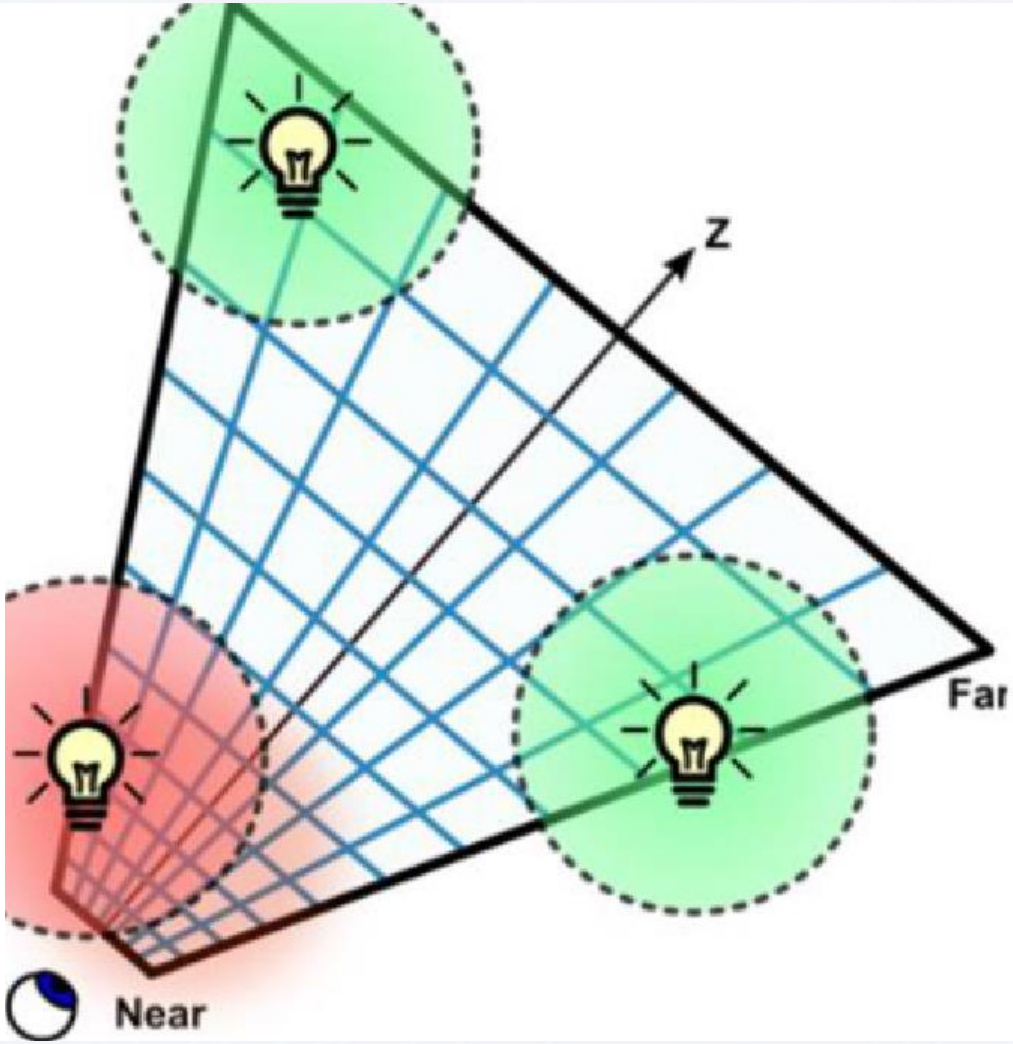
Compiler Statistics	
ALU	1386 Instructions
FP16	583 Instructions
FP32	183 Instructions
Int16	381 Instructions
Int32	182 Instructions
Control Flow	114 Instructions
Wait	113 Instructions
Device Atomic	26 Instructions
Device Load	79 Instructions
Device Store	103 Instructions
Texture Read	1 Instructions
Temporary Registers	72
Uniform Registers	151
Max Theor...Occupancy	47.917%

The indirect lighting shader statistics

Performance Optimization - Clustered Deferred Shading

Basic steps for clustered deferred shading

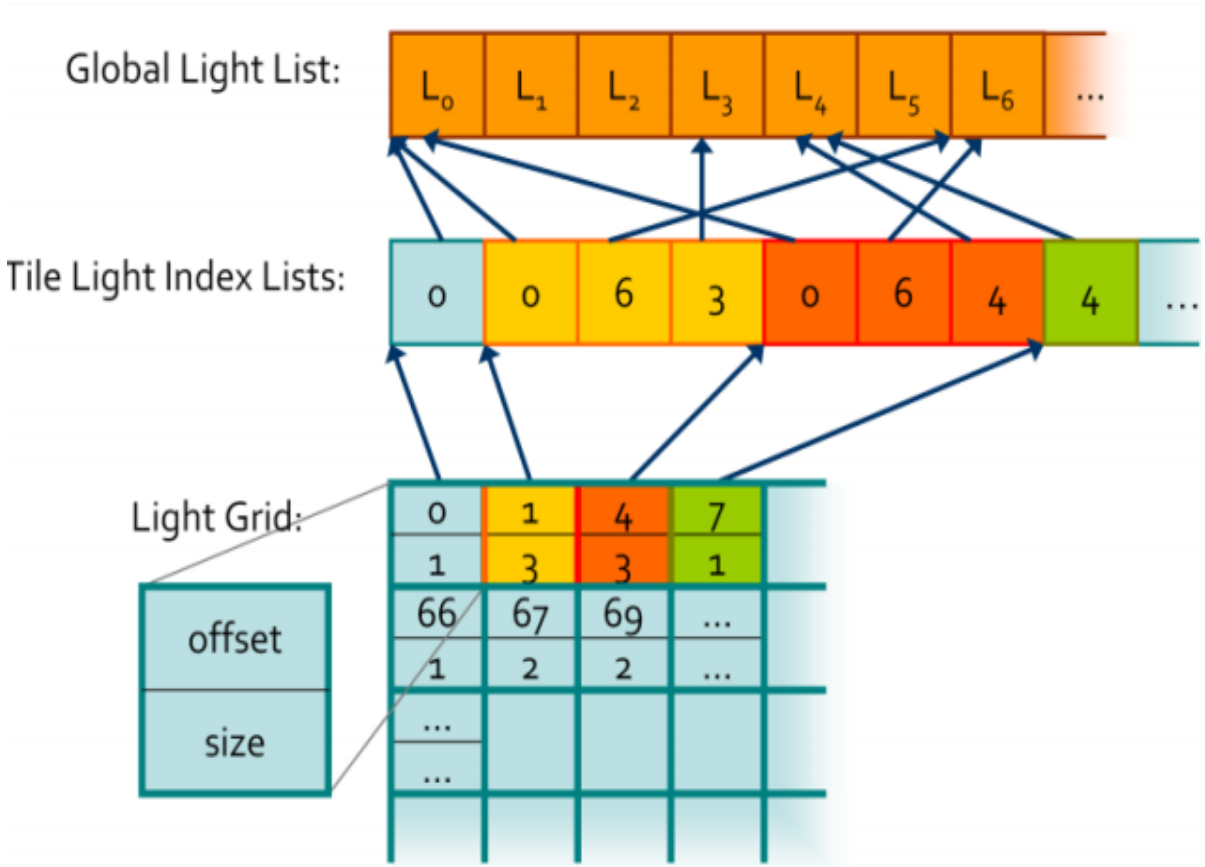
- Divide the view frustum into small clusters
- Perform light culling and assign lights to clusters
- Shading samples using light list



Performance Optimization - Clustered Deferred Shading

Data structure

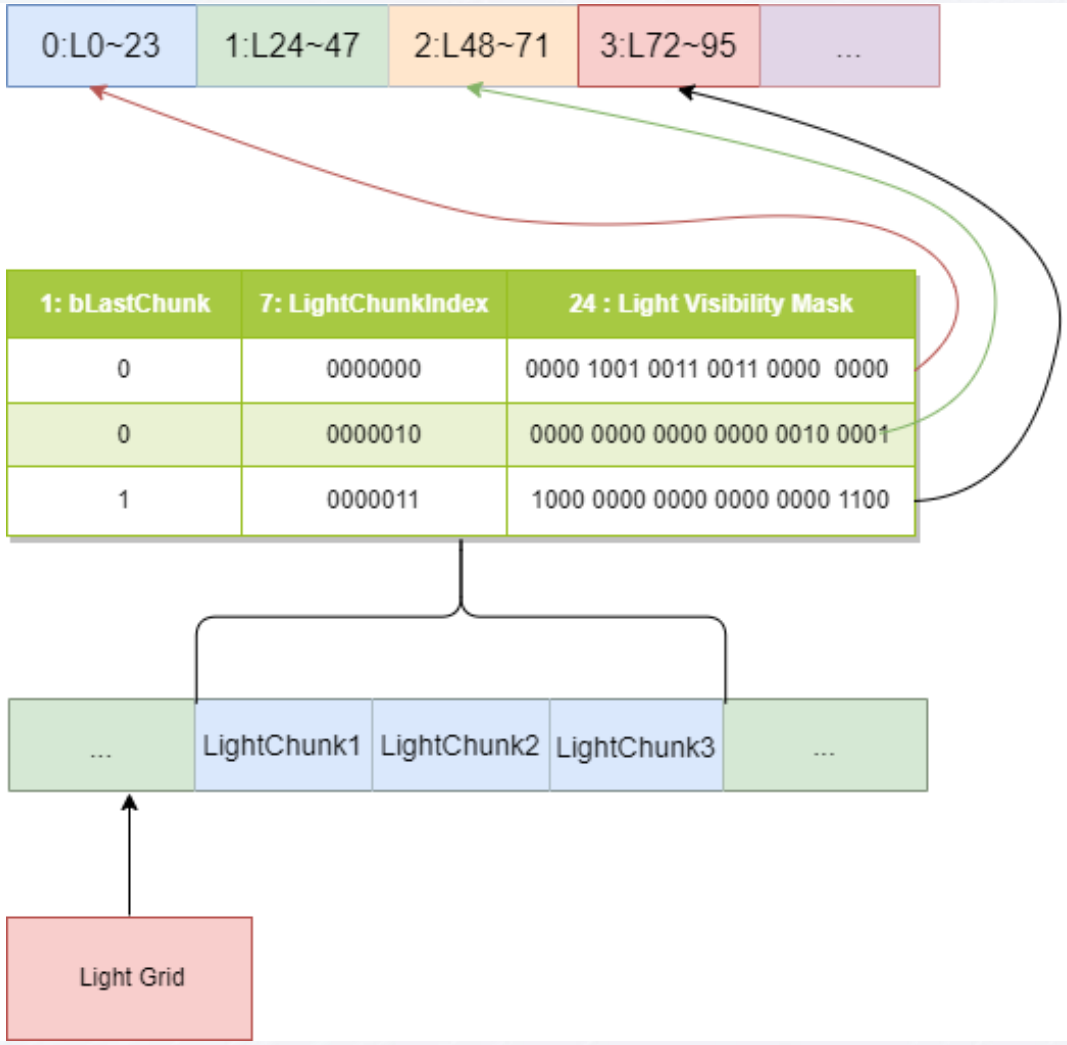
- UE5
  - Culled LightGrid data in buffer
    - R32_UINT
  - Light index list in buffer
    - R16_UINT



Performance Optimization - Clustered Deferred Shading

Data structure

- Ours
  - Divide lights into chunks
    - Sort by the depth of light
  - Culled LightGrid data in buffer
    - R32_UINT
      - 1 bit, last chunk flag
      - 7 bit, light chunk index
      - 24 bit, light visibility mask
  - Maximum number of lights
    - 127 * 24 = 3048



Performance Optimization - Clustered Deferred Shading

Our approach

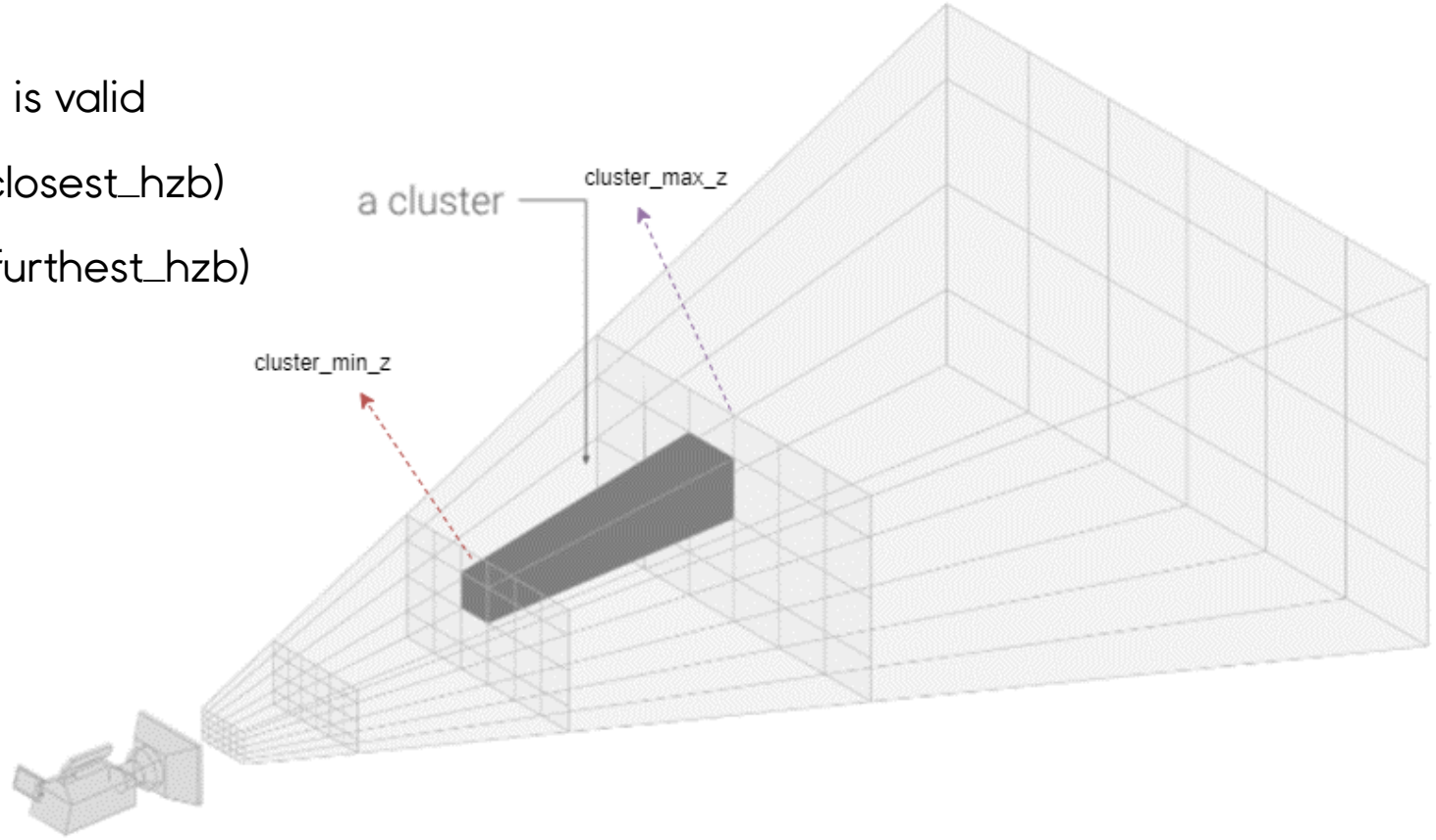
- Better performance
  - In most cases, only 1 sampling is required
- Less memory cost
  - The memory cost is only about 1/5 of UE5

	Memory
UE5	1024K
Ours	214K

Performance Optimization - Clustered Deferred Shading

Light culling

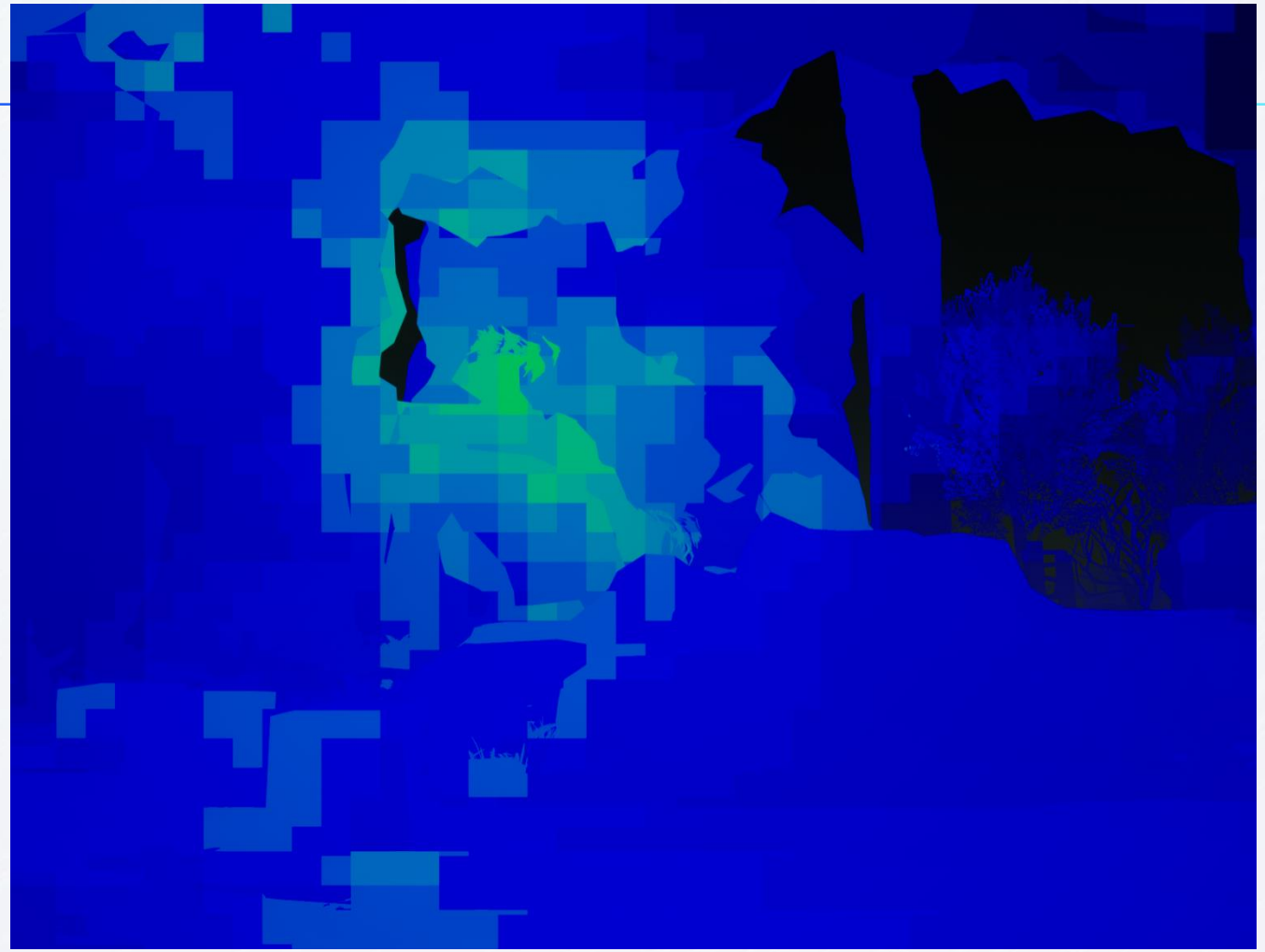
- Refine the bounds of cluster when HZB is valid
  - cluster_min_z = max(cluster_min_z, closest_hzb)
  - cluster_max_z = min(cluster_max_z, furthest_hzb)



IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

- Performance Optimization - Clustered Deferred Shading



Light Culling(Normal/With Furthest HZB)

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

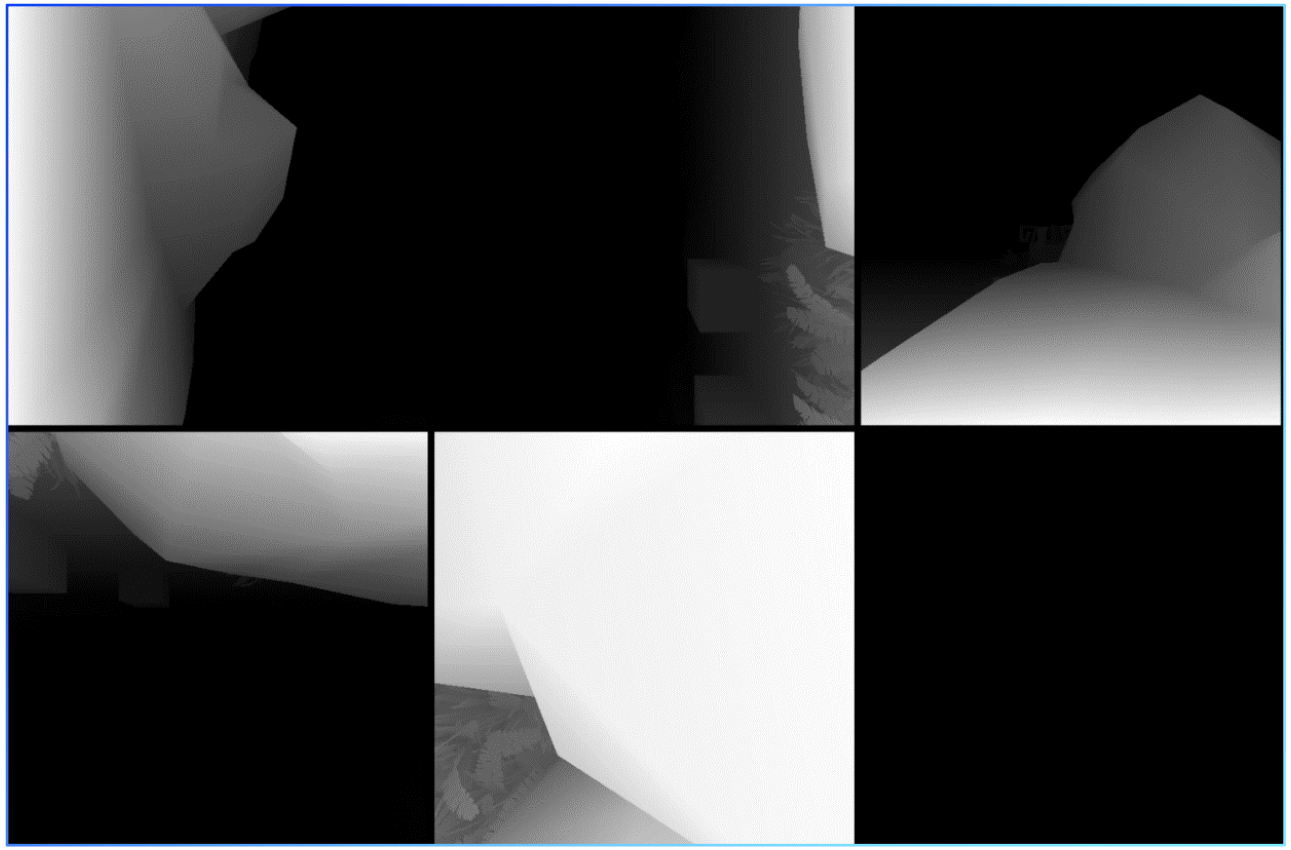
Many-Lights Shadows

UE5

- UnSupported

Ours

- Supported
  - Point Light
  - Spot Light
- Shadow Maps
  - Standard Shadow Maps
  - Shadow Map Atlas
  - Bindless Shadow Maps



Map CubeMap into 2D texture

(Pros: HardwarePCF, Bindless)

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Many-Lights Shadows

Standard Shadow Maps

- Render shadow maps for each light
- Pros
  - Good compatibility
- Cons
  - Cannot be used with clustered deferred shading

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

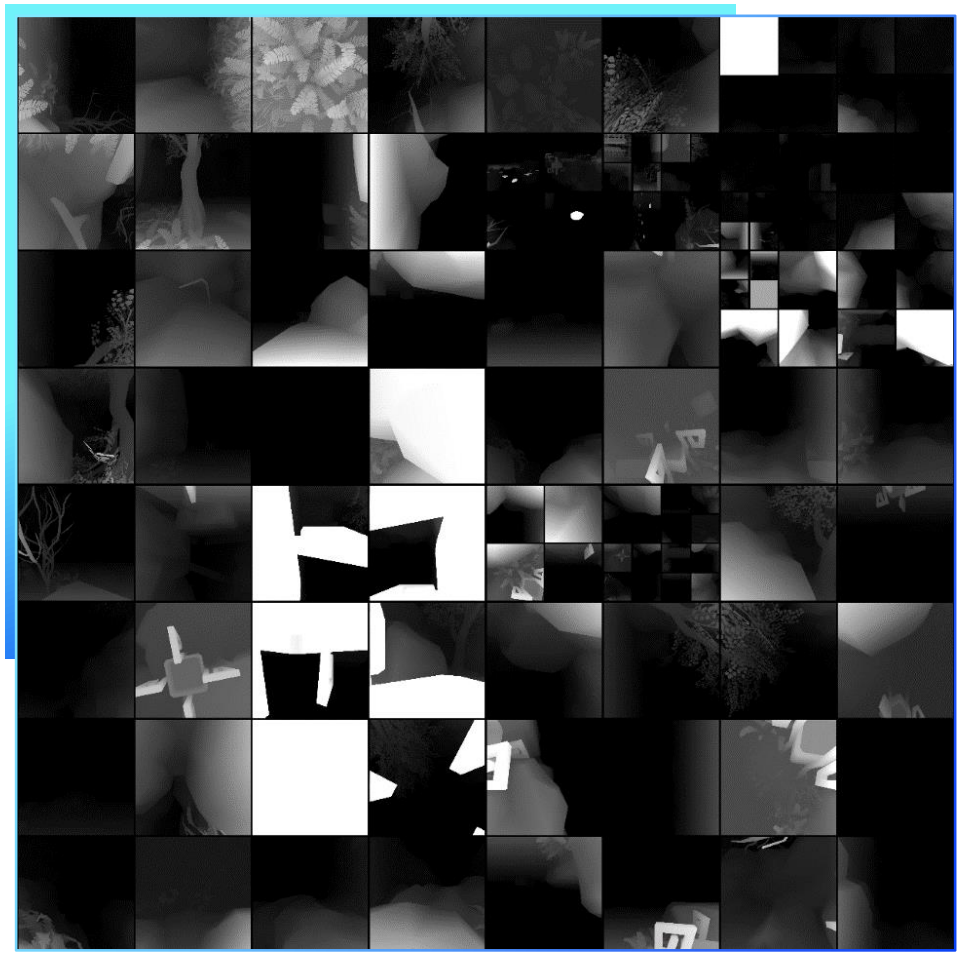
LIGHTSPEED STUDIOS | GDC

Many-Lights Shadows

Shadow map atlas

- Render all shadow maps to an atlas texture

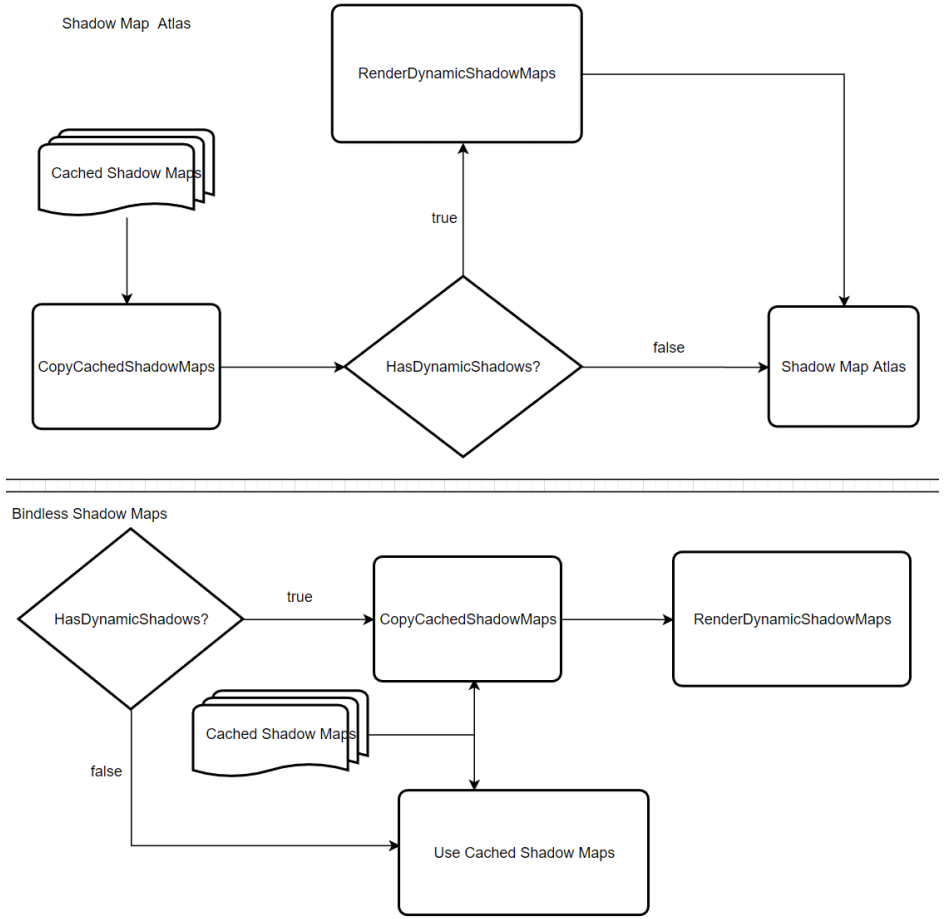
  Pros
  - Can be used with Clustered Deferred Shading
  - Good compatibility
- Cons
  - Lack of flexibility
    - Managing atlas texture is very difficult
  - Performance Issues
    - Additional shadow map copy costs



Many-Lights Shadows

Bindless shadow maps

- Render shadow maps for each light and sample from a shadow map array during shading
- Pros
  - Can be used with Clustered Deferred Shading
  - Better Performance
- Cons
  - Device compatibility is not very good
    - Requires GPU to support bindless



Additional shadow map copy overhead

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Test

- Scene - Epic ActionRPG
  - 1 Directional Light, 30 Point Lights, 10 SpotLights
  - 5 Shading Models(DefaultLit, TwoSideFoliage, Subsurface, ClearCoat, Preintergrated Skin)

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Test

- Google Pixel5, Snapdragon 765G, Adreno 620, 1512x720
- Frequency: 500 MHz

	UE5（Standard）	UE5（Clustered Deferred Shading）	Ours(Standard)	Ours（Optimized Clustered Deferred Shading）
FPS	13.8	13	14.5	17.8
GPU Times(ms)	72.9	77.0	68.9	56.7
DrawCall	403	283	455	283

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Test

- Snapdragon 8 Gen 1 QRD, Adreno 730, 1552x720,
- Frequency: 500 MHz

	UE5（Standard）	UE5（Clustered Deferred Shading）	Ours(Standard)	Ours（Optimized Clustered Deferred Shading）
FPS	60	60	60	60
GPU Times(ms)	7.9	8.2	8.1	7.0
DrawCall	403	283	455	283

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Test

- IPhone12 Mini, 1624x750

	UE5 (Standard)	UE5 (Clustered Deferred Shading)	Ours(Standard)	Ours (Optimized Clustered Deferred Shading)
FPS	30	30	30	30
GPU Times(ms)	11.3	13.1	10.9	9.8
Power(mW)	2085	2465	2026	1838
DrawCall	403	283	455	283

IMPROVEMENTS TO MOBILE DEFERRED RENDERER

LIGHTSPEED STUDIOS | GDC

Performance Test

UE5

Compiler Statistics	
ALU	2437 Instructions
FP16	986 Instructions
FP32	323 Instructions
Int16	710 Instructions
Int32	325 Instructions
Control Flow	205 Instructions
Wait	195 Instructions
Device Atomic	43 Instructions
Device Load	143 Instructions
Device Store	170 Instructions
Texture Read	2 Instructions
Temporary Registers	92
Uniform Registers	239
Max Theor...Occupancy	35.417%

DirectLighting + IndirectLighting
GPU Occupancy: ~35%

OUR SOLUTION

Compiler Statistics	
ALU	1233 Instructions
FP16	450 Instructions
FP32	167 Instructions
Int16	396 Instructions
Int32	169 Instructions
Control Flow	108 Instructions
Wait	111 Instructions
Device Atomic	20 Instructions
Device Load	83 Instructions
Device Store	79 Instructions
Texture Read	1 Instructions
Temporary Registers	52
Uniform Registers	152
Max Theor...Occupancy	60.417%

DirectLighting
GPU Occupancy: ~60%

Compiler Statistics	
ALU	1386 Instructions
FP16	583 Instructions
FP32	183 Instructions
Int16	381 Instructions
Int32	182 Instructions
Control Flow	114 Instructions
Wait	113 Instructions
Device Atomic	26 Instructions
Device Load	79 Instructions
Device Store	103 Instructions
Texture Read	1 Instructions
Temporary Registers	72
Uniform Registers	151
Max Theor...Occupancy	47.917%

IndirectLighting
GPU Occupancy: ~48%

LIGHTSPEED STUDIOS | GDC

OUTLINE

PART.01 Gong Jing - Practical Mobile Deferred Shading

PART.02 Bo Li - Practical Mobile GPU-driven Pipeline

LIGHTSPEED STUDIOS | GDC

PART.02

PRACTICAL MOBILE GPU-DRIVEN PIPELINE

- Introduction
- Practical Mobile GPU-Driven Pipeline
- Performance Discussion

INTRODUCTION

LIGHTSPEED STUDIOS | GDC

Motivation : Scale efficiently

Increase geometry efficiency

- ~10000s of unique static meshes on screen
- Not just instances of a handful of mesh types

Increase material efficiency

- ~1000s of materials with different texture settings
- Not instanced ones with slight variations (vertex color, etc)

Efficiency matters

- Not just push hardware to the limit on mobile

INTRODUCTION

LIGHTSPEED STUDIOS | GDC

Why choose a GPU-Driven pipeline?

Take advantage of exiting game engines without too many modifications on low level APIs

- Commercial engine can be less multi-threaded than specialized engines

Improve computational efficiency for parallel workloads

Scalability: Larger work chunks -> Better GPU utilization

- Less context switches

Finer-grain culling

- Less wasted vertex/pixel work
- Efficient culling for shadow views

INTRODUCTION

LIGHTSPEED STUDIOS | GDC

Existing GPU-Driven pipeline solutions

Nanite for mobile?

- Optimized for desktop and console
  - Compatibly with mobile specific: Binning/TBDR/HW Prefetches
  - Sub-pass concept restricts switching between computation and render operations
    - Forces GMEM reload
  - Unusual depth buffer usage required for material sorting
  - High shader complexity

- Will talk about performance later!

UNREAL ENGINE

PRACTICAL MOBILE GPU-DRIVEN PIPELINE

Primitive Choices : Non-Native Types

Software Indexed Primitive : Common properties

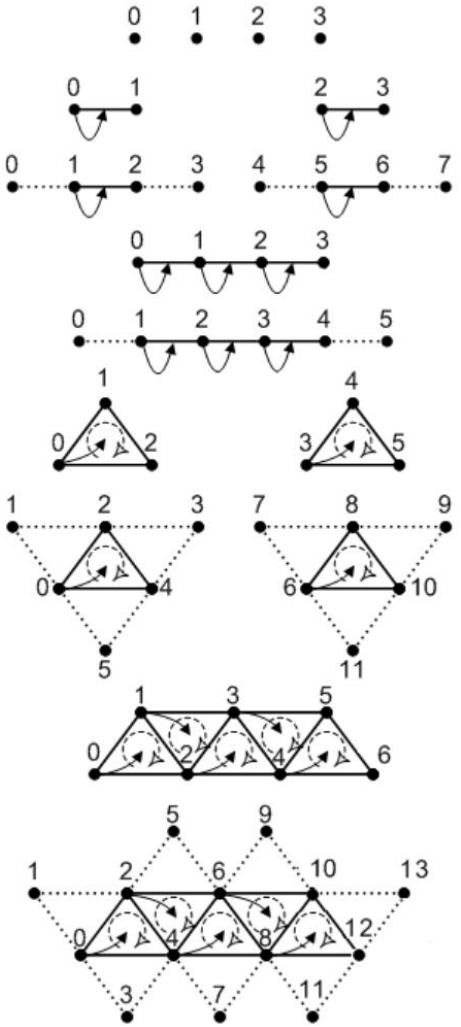
- Use VertexID to fetch index buffer
- Mixing multiple meshes without MDI
- Compact index encoding (6~8bit possible)
- Allow reuse vertices

1# Software Indexed Triangle Lists

- Less-compact index memory usage(3 indices for 1 Triangle)
- No vertex transform reuse

2# Software Indexed Triangle Strips

- Better index memory usage(1 indices for 1 Triangle)
- Moderate vertex transform reuse



Mesh Cluster VB/IB Construction

Software Indexed Triangle Strips (64 indices clusters)

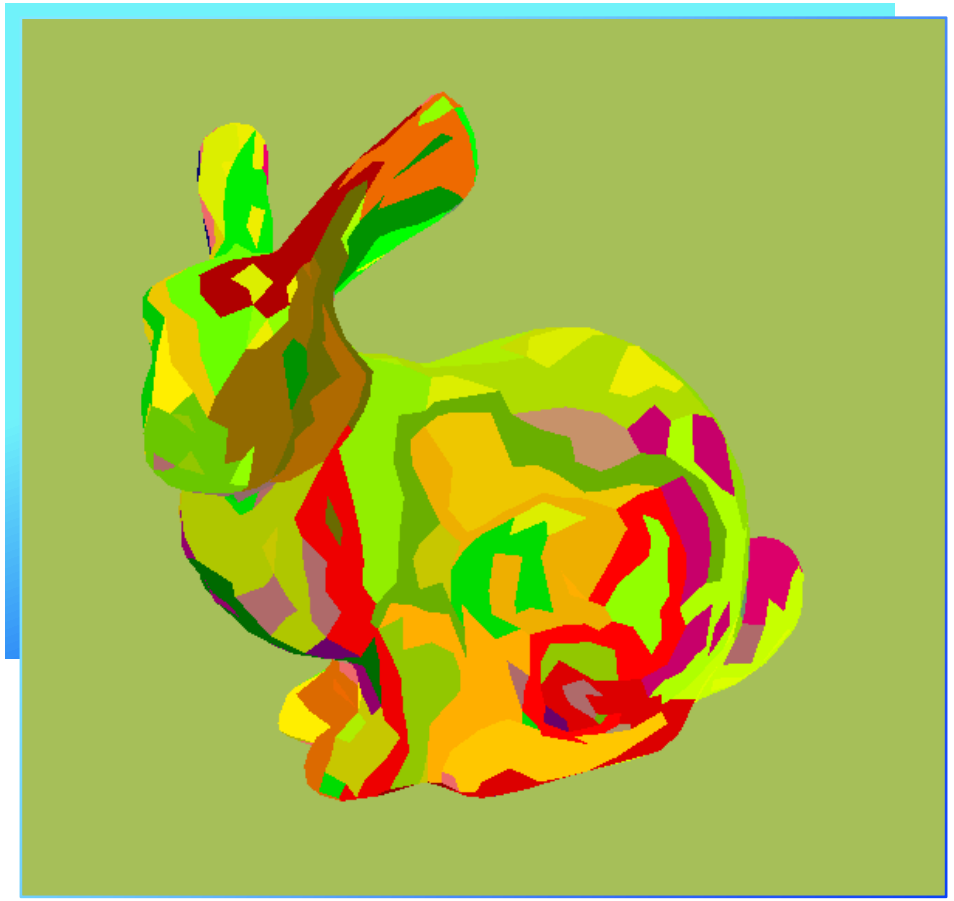
- 8-bit indices with a 256-entry vertex pool
  - Allowing share vertices across clusters
  - Simple encoding/decoding
- 40% vertex buffer size reduction compared to non-indexed strips

Primitive restart

- Use NAN as position
- No special indices like 0xffff or 0xffffffff
  - Avoids high penalty on some popular mobile GPUs

Compressed VB

- Integer position avoiding seams
- Quaternion tangent frame
- Unified buffers with flexible vertex stride



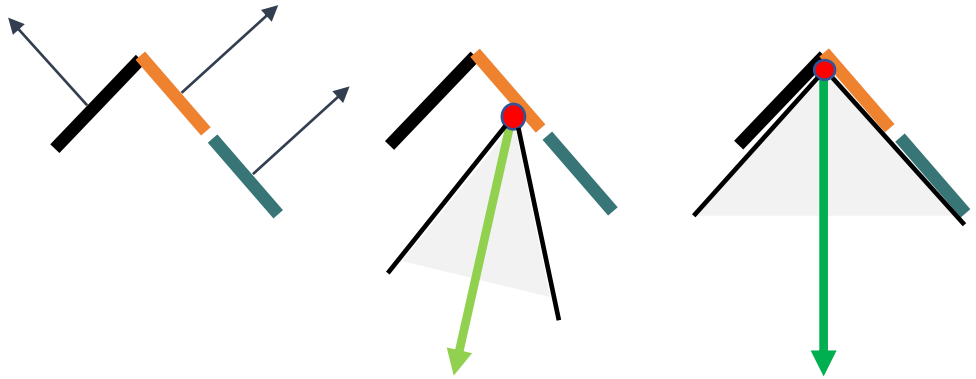
Culling Cone construction

Started with GeometryFX

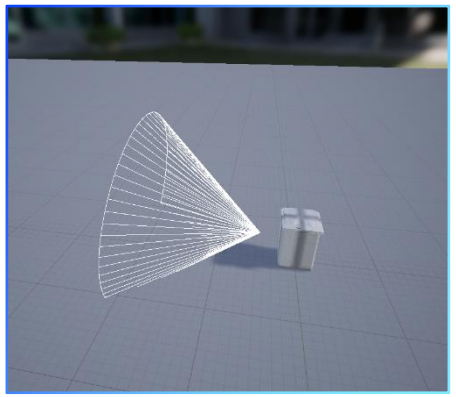
- Can generate sub-optimal axis

Anchored Cone Construction

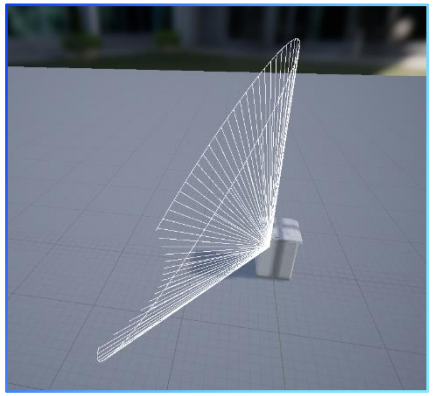
- Use a linear equation solver to find the best cone center within the triangle normal hull
- Hugues Hoppe Silhouette clipping
  - https://hhoppe.com/proj/silclip/



Input | Not Optimized | Optimized



Before best center search



After best center search

Mesh Cluster Culling: Main view occlusion

Reprojected HZB from previous depth

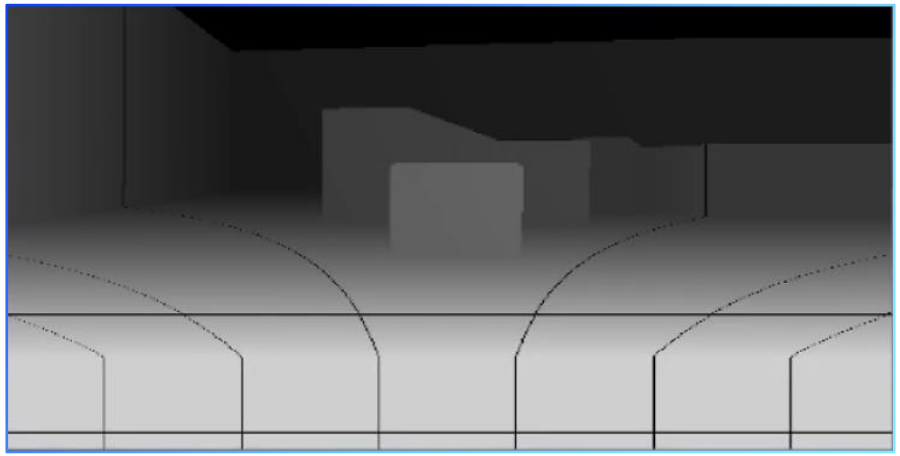
- 3x3 dilation pass
- Single pass 5 Mip chain

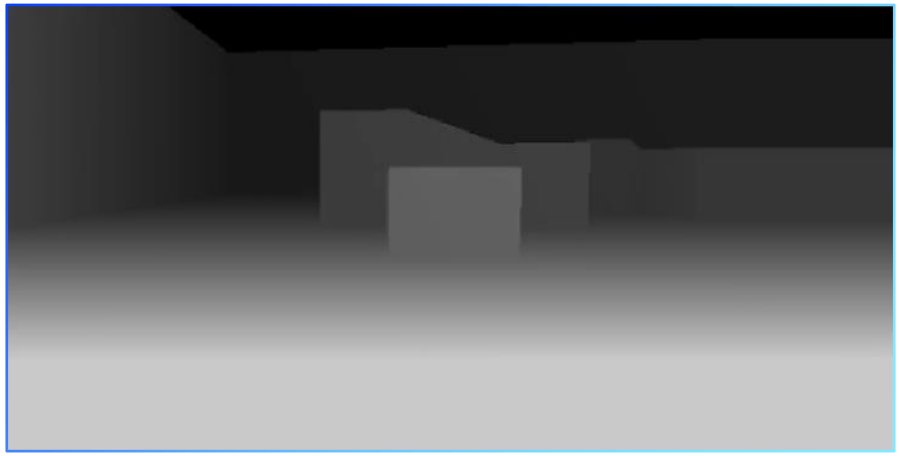
Flat hierarchical culling

- Instance AABB -> Cluster AABB

Optional: Lossy occlusion culling for less important objects

- Foliage
- Small props

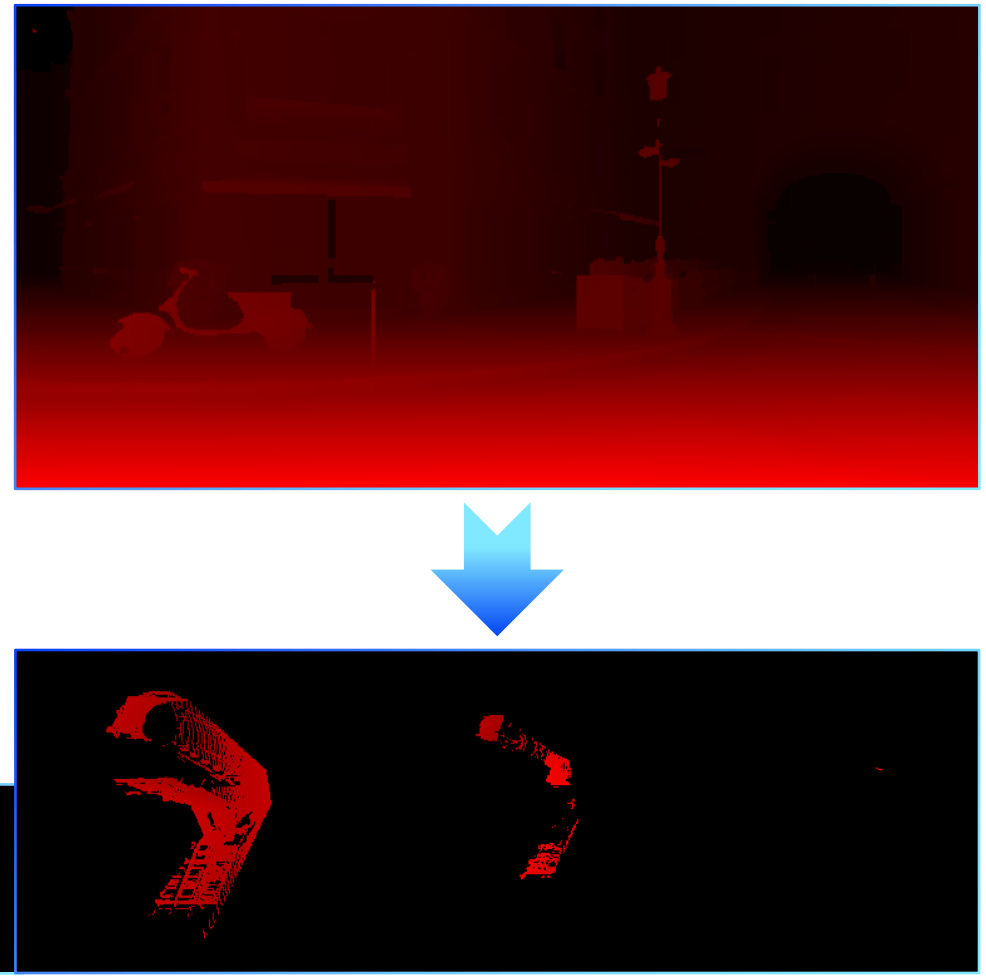




Mesh Cluster Culling: Shadow occlusion

Hierarchical point cloud shadow culling

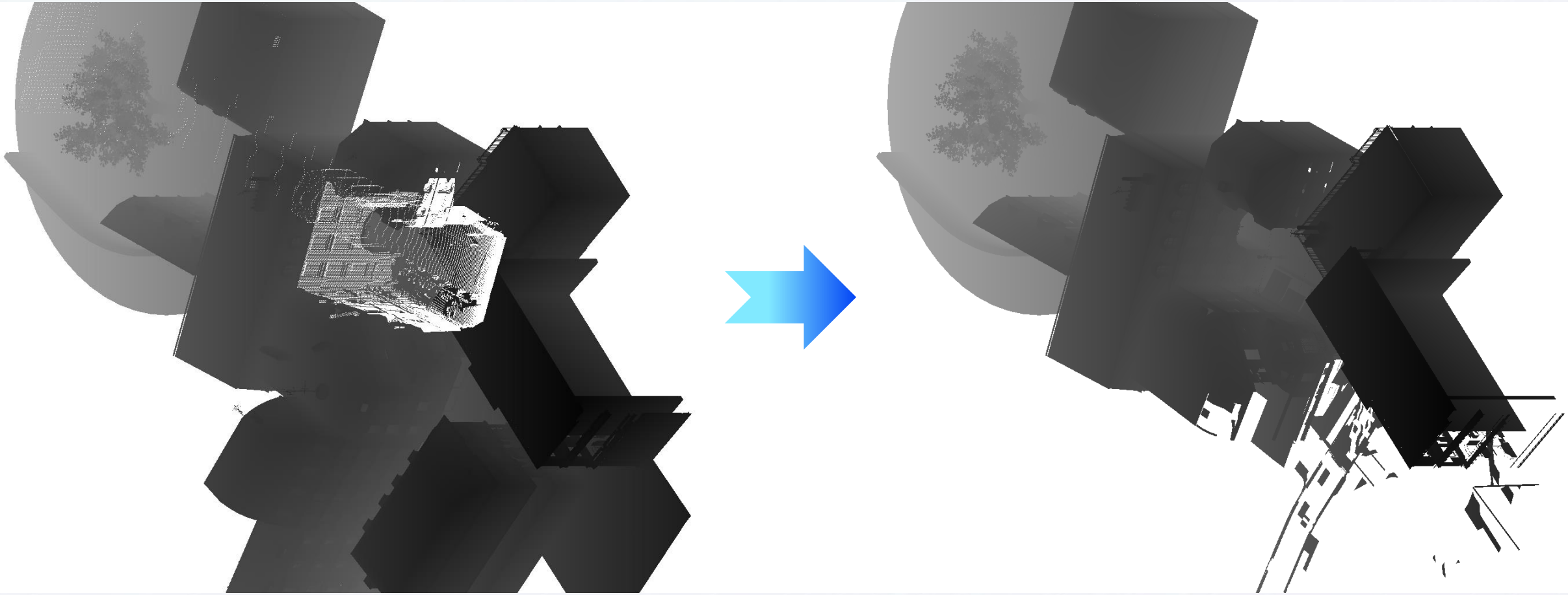
- Scattered from view space to shadow space
- Single-pass multi-mip projection
  - Efficient large object culling
- Single-pass multi-cascade generation
- Software Fast Clear
  - Inspired by hardware features
  - Only dirty regions(and its Mips) needs to be cleared



PRACTICAL MOBILE GPU-DRIVEN PIPELINE

LIGHTSPEED STUDIOS | GDC

- Mesh Cluster Culling: Shadow occlusion



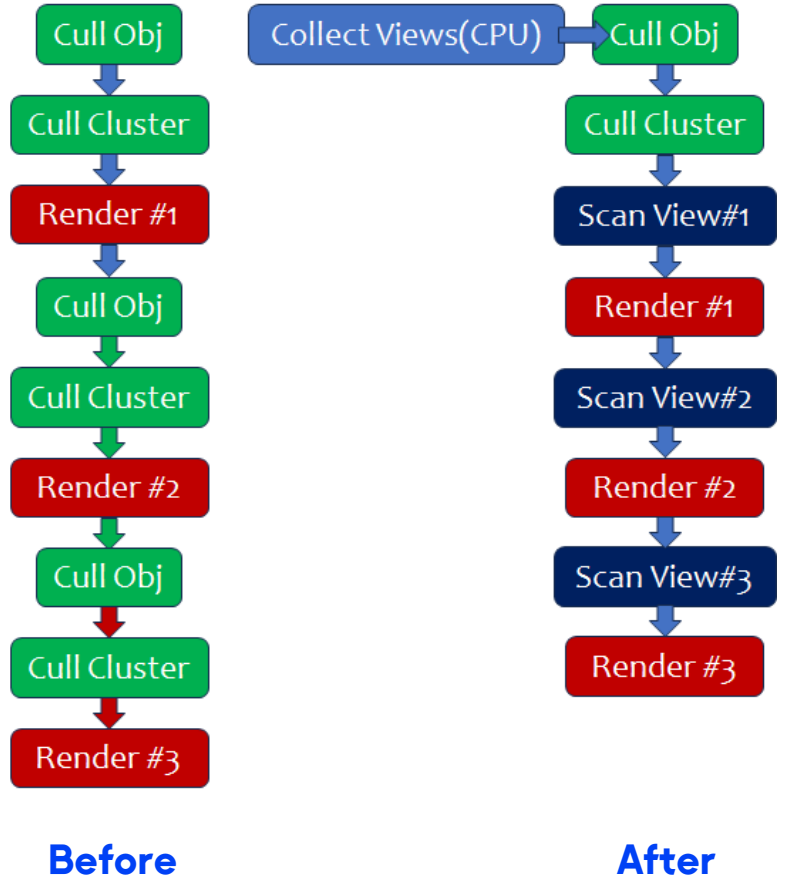
Mesh Cluster Culling: Multi-View culling

Many views needs to be rendered in a typical game

- Main viewport
- CSM * (3 ~ 4)
- Local light shadows

Single-pass culling for up to 8 views

- Output non-zero visibility mask for each view
- A Byte for each instance/cluster(8 views)
- Each view only needs a fast scan and compact(0.05ms)
- Reduced ~60% compute cost



GPU Driven Buffer Defragments

Compute shader based

- Allowing streaming in/out
- Necessary to support scene management

Avoid CPU-GPU sync point

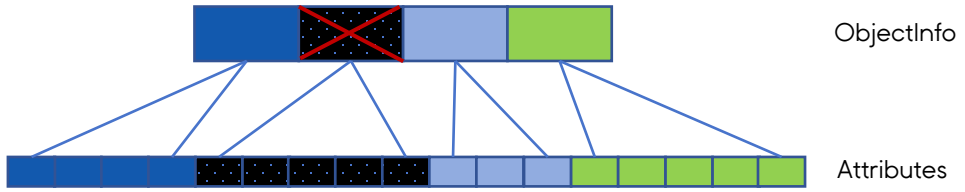
- No need to delay frame or block GPU operations

Triggered by multiple events

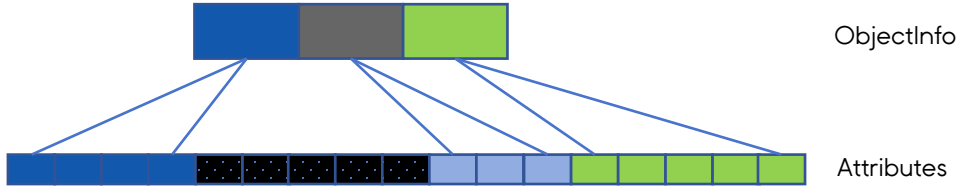
- Deleted instances above threshold
- Add/Remove meshes / Free spaces running low

Multi-frame state machine

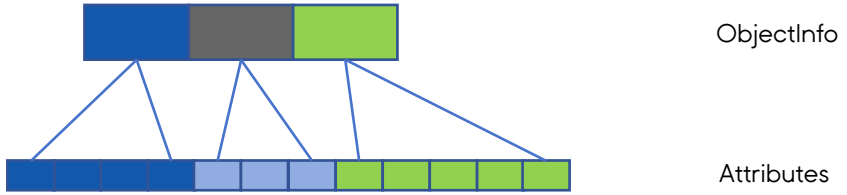
- Amortize GPU cost per-frame
- Can be flushed when update GPU objects are requested
- 5 States: Cluster/MaterialID/Position/Attributes etc



Event: Object Deleted



Phase 1: Defrag ObjectInfo



Phase 2: Defrag Vertex Attributes

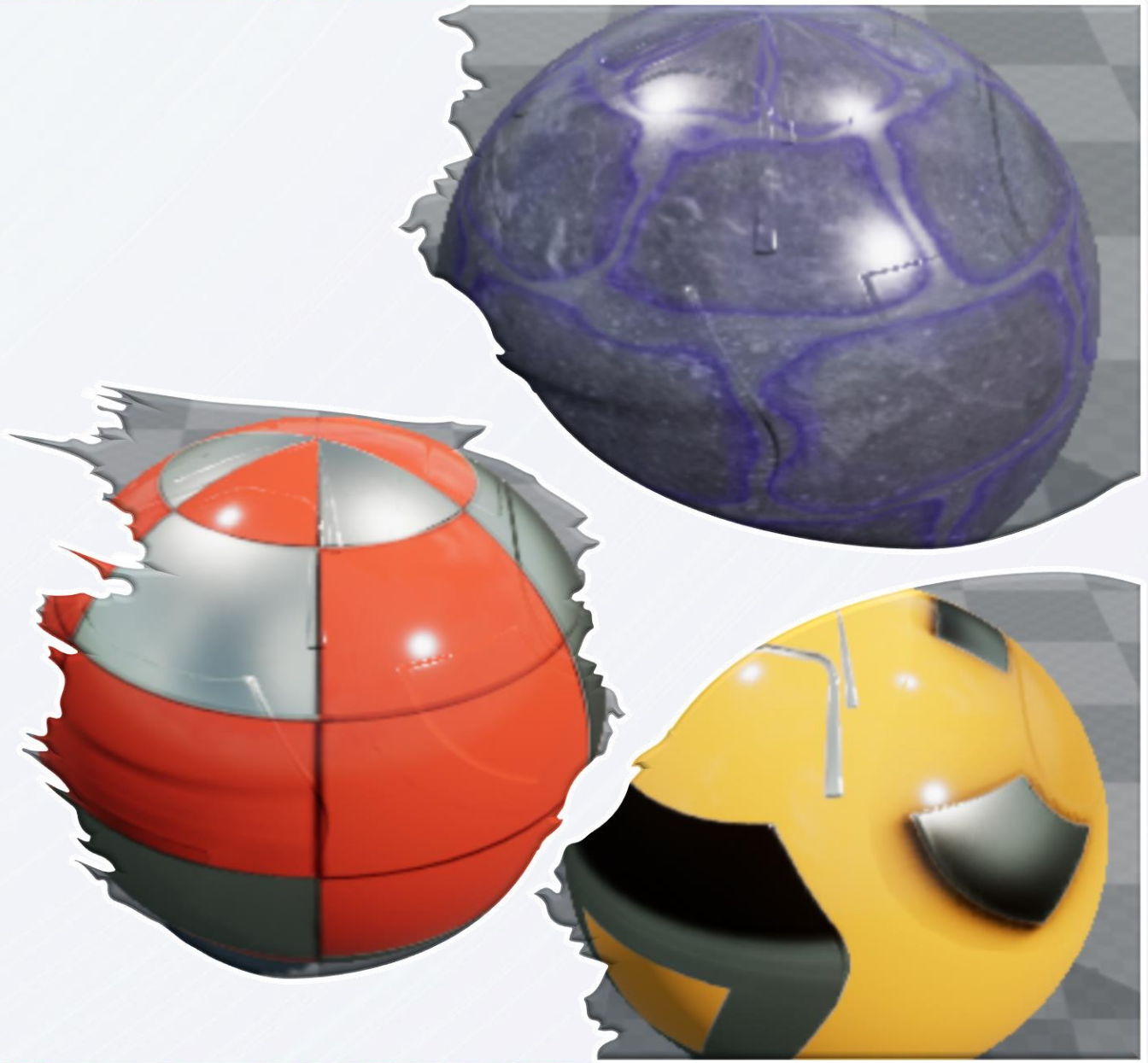
PRACTICAL MOBILE GPU-DRIVEN PIPELINE

LIGHTSPEED STUDIOS | GDC

Material batching

Based on UE's MaterialInstance Override

- Single draw call for multiple compatible MaterialInstances
- Supports overriding most Texture/PreShader Parameters
- Automatically replace Texture/PreShader access with indexed version
- Detecting compatible material instances



PRACTICAL MOBILE GPU-DRIVEN PIPELINE

LIGHTSPEED STUDIOS | GDC

Material batching

Virtual Textures(SVT)

- Pros
  - Allowing fine-grained streaming
  - Already developed by UnrealEngine
- Cons
  - Extra page texture access
  - More ALUs for manual mips/ddx/ddy
  - Fixed texture format
  - Border memory waste

Bindless Textures

- Pros
  - Smaller impact to shaders
  - Flexible texture formats
  - Hardware Anisotropic filtering and mip-maps
  - Basic support on most Vulkan Devices
- Cons
  - Extra texture descriptor access(T#)
  - Hardware support can vary
    - T# cache?
    - Scalar Registers only?(AMD)
  - Performance beats SVT by about 15%

Vulkan Bindless RHI Implementation

Supported by MOST Vulkan Android devices (90%+)

- Only requires shaderSampledImageArrayDynamicIndexing

Vulkan Bindless RHI Implementation

Supported by MOST Vulkan Android devices (90%+)

- Only requires shaderSampledImageArrayDynamicIndexing
- Not to be confused with shaderSampledImageArrayNonUniformIndexing
  - Needed for NonUniformResourceIndex on some vendors (AMD)
  - Works fine without it in practice on Mobile
  - Doesn't require VK_EXT_descriptor_indexing

Vulkan Bindless RHI Implementation

Supported by MOST Vulkan Android devices (90%+)

- Only requires shaderSampledImageArrayDynamicIndexing
- Not to be confused with shaderSampledImageArrayNonUniformIndexing
  - Needed for NonUniformResourceIndex on some vendors (AMD)
  - Works fine without it in practice on Mobile
  - Doesn't require VK_EXT_descriptor_indexing

Doesn't require VkDescriptorIndexingFeatures.runtimeDescriptorArray

- Fixed size array rather than unbounded array

Array size limited by maxDescriptorSetSampledImages

- Mali G72: 256, Adreno 650: 512k
- For max compatibility, use 224 Bindless texture slots

Metal Bindless RHI Implementation

Implemented with Argument Buffer Tier 2

- Tier 1 doesn't support dynamic indexing
- Natural concept mapping with UE uniform buffers
  - Mixing constants and resources
- Also comes with performance benefits of Argument buffers
  - Less Driver managements
  - Less API binding costs

Shader front-end and API modifications

- Resource index remapping with Argument Buffer pointers
- Add Metal argument buffer support in RHI state cache

PERFORMANCE DISCUSSIONS

Nanite as mobile GPU-driven front-end?

Nanite is also a very capable GPU-Driven Mesh pipeline

- Why not used it on Mobile platforms?

We experimented just that!

- Nanite direct G-buffer output on Mobile
  - Remove Visibility buffer etc
  - Plug Nanite HWRasterize into BasePass shaders
  - Tangent stream was added to Nanite vertex buffer for G-buffer rendering
- Many mobile platform adjustments to make it work
  - Enable Vulkan shader model 6.0 with DXC for subgroup
  - Reduce Nanite pre-allocated buffer size
  - Disable vertex compression (incorrect result)
  - Disable persistent cull (device lost)

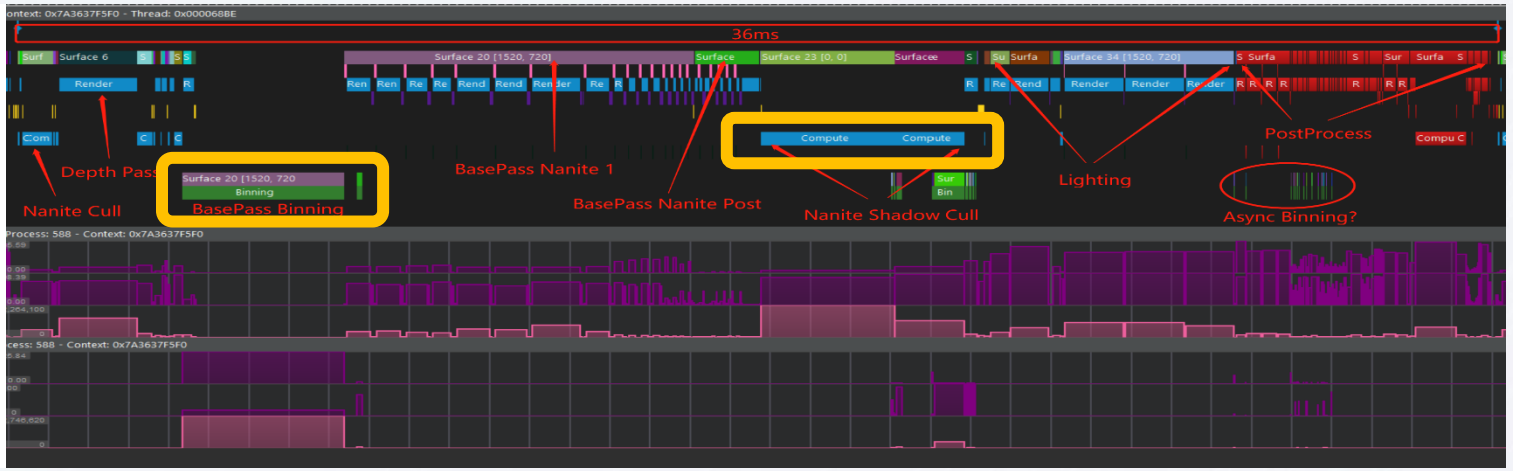
PERFORMANCE DISCUSSIONS

LIGHTSPEED STUDIOS | GDC

- Nanite as mobile GPU-driven front-end : Snapdragon 8Gen2 SDP capture

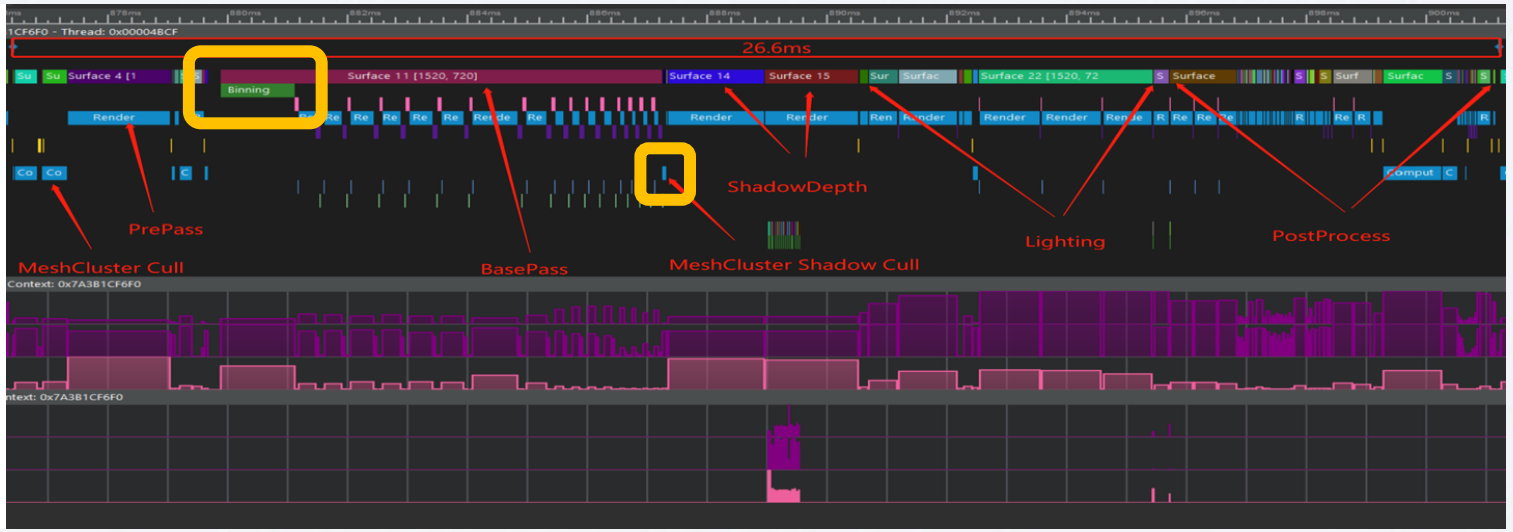
NANITE PORT:

36ms per frame



OURS:

26.6ms per frame



PERFORMANCE DISCUSSIONS

LIGHTSPEED STUDIOS | GDC

Nanite as mobile GPU-driven front-end : Snapdragon 8Gen2 Results

Performance lower compared to custom GPU-Driven Pipeline

- 47fps -> 30fps
- Detailed performance analysis follows

	Our MeshCluster	Our Nanite Mobile port
DepthPass+Culling(ms)	3.22	4.11
BasePass + Binning(ms)	7.37	14.4
Shadow Culling(ms)	0.05	5.06
Shadow Rendering(ms)	3.18	N/A
Lighting(ms)	2.9	2.9
Postprocess(ms)	5.67	6.52

Conclusion:

- A custom GPU-driven pipeline is more efficient on Mobile

PERFORMANCE DISCUSSIONS

LIGHTSPEED STUDIOS | GDC

Internal case study

Internal R&D project

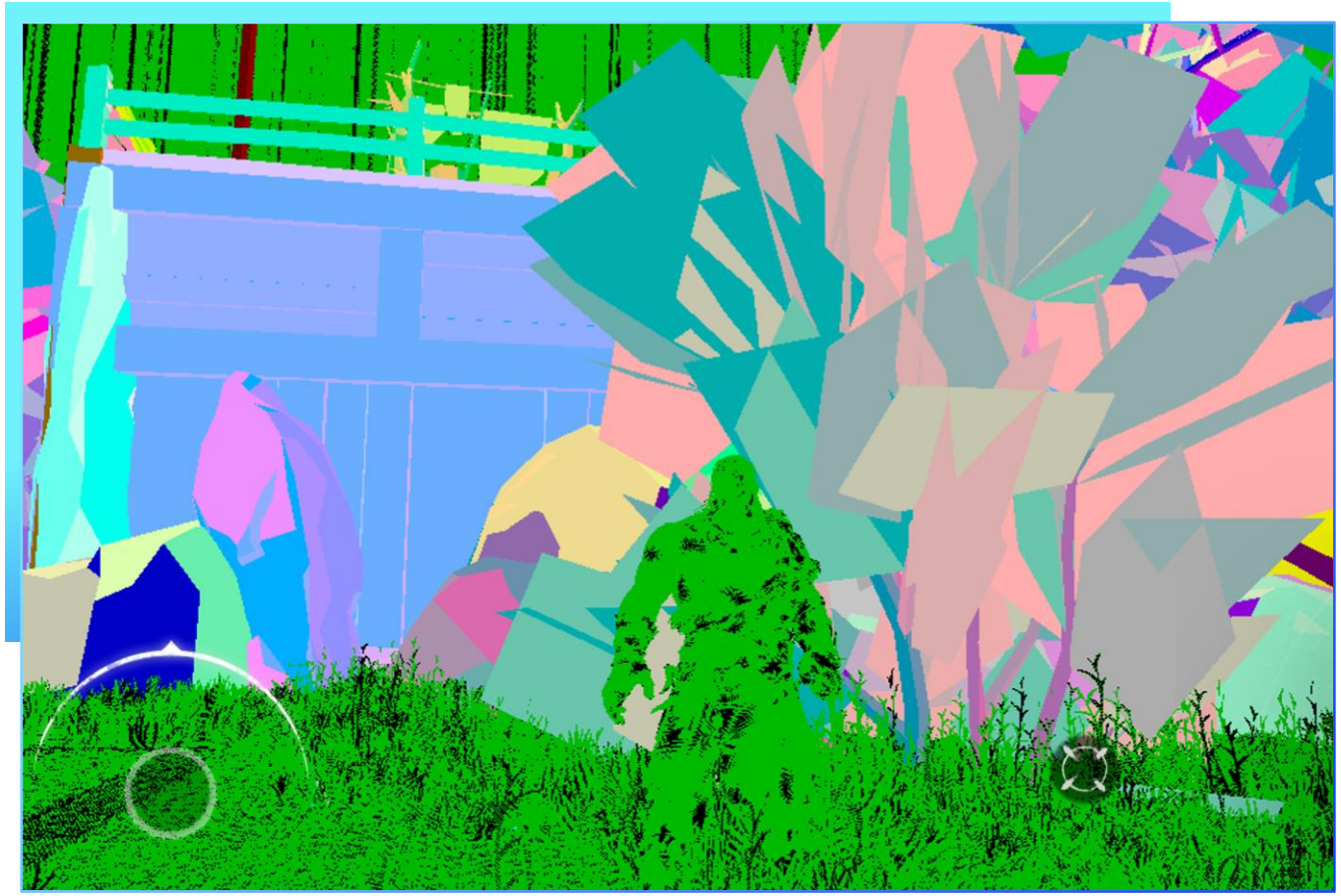
- Samsung S20 (SD865)

Represents actual game development

- Heavier CPU load
- Gameplay/Physics/Online

GPU Driven Static Mesh

- Building
- Props
- Iconic Plants



PERFORMANCE DISCUSSIONS

LIGHTSPEED STUDIOS | GDC

Internal case study

GPU-only cost reduced by 7.1ms (17% improvement)

- BasePass-only improvement: 33.7%

	Total Frame time	Compute+Culling	Shadow	Binning	Basepass	PostProcessing
MC off	49.84	5.87	1.59	2.352	23.13	15.85
MC on	42.69	6.91	1.32	2.325	15.34	15.4
Diff	14.35%	-17.72%	16.98%	1.15%	33.68%	2.84%

System performance

- Framerate 12fps -> 25fps
- CPU cost can play a big role here

PERFORMANCE DISCUSSIONS

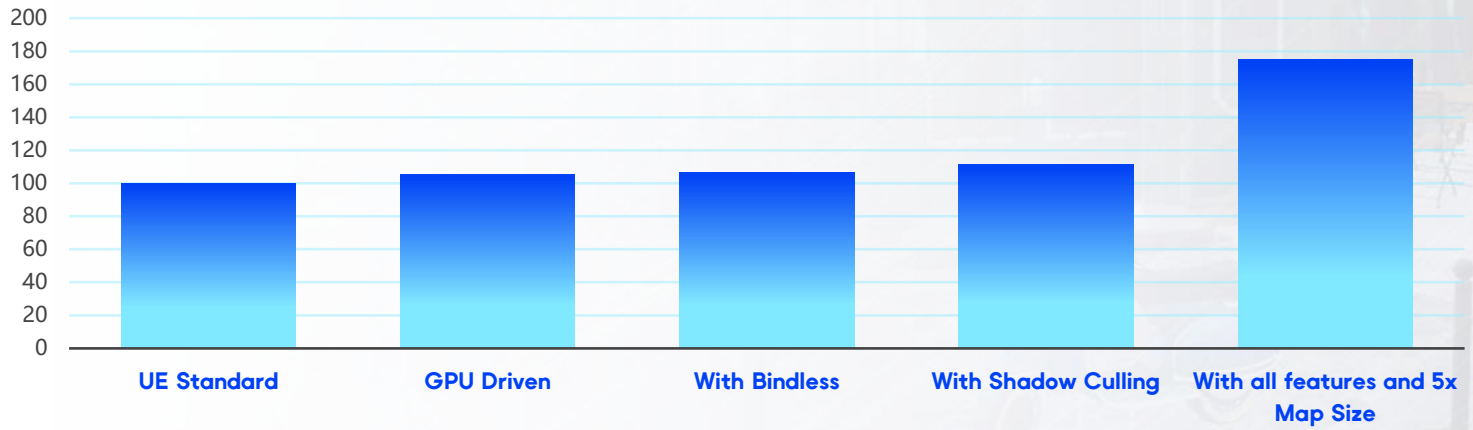
Performance scalability

Performance scale with map size

- Performance increase 12% with single Amazon Bistro
- Performance increase 75% with 5x Amazon Bistro

Expected scales better with future mobile platforms

- Wider GPUs, More efficient parallel computing



Amazon Bistro Demo %Performance

LIGHTSPEED STUDIOS | GDC

Innovation Unleashed (2)

CyberHuman: Next-Gen Character Creation Pipeline Powered by Machine Learning

LIGHTSPEED STUDIOS | GDC

OUTLINE

PART.01 Introduction

PART.02 Character modeling, rigging, and animation pipeline

PART.03 Summary and future works

LIGHTSPEED STUDIOS | GDC

PART.01

INTRODUCTION

BACKGROUND

Character customization technology

MetaHuman: a game-changer in character creation technology

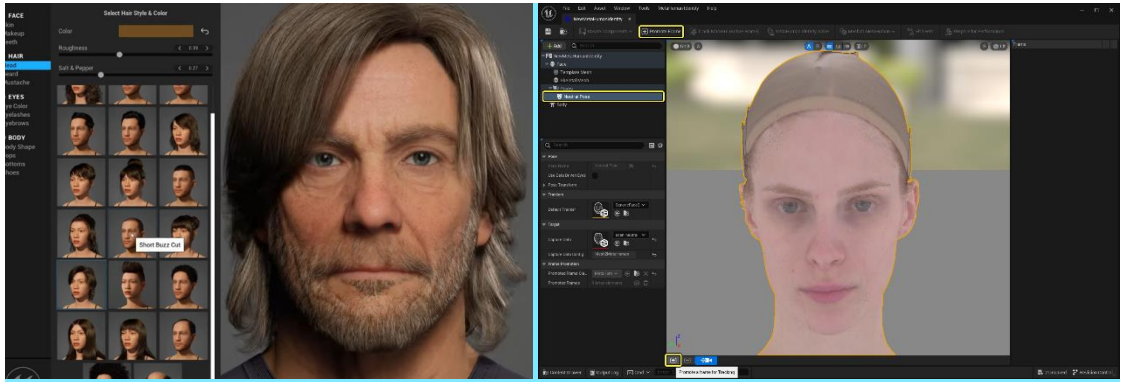
- Incredibly realistic characters creation
- User-friendly interfaces and streamlined workflows
- Seamlessly integrated with Unreal Engine

MetaHuman powers come with limitations

- Limited gameplay creation
- Intuitive modals (image or text) not supported
- Inconsistent mesh shapes
- No simple edit of rig poses accessible
- No official Lipsync solution

CyberHuman

- Character creation solution powered by machine
- learning
- Rich toolset for both DCC and gameplay
- MetaHuman assets compatibility



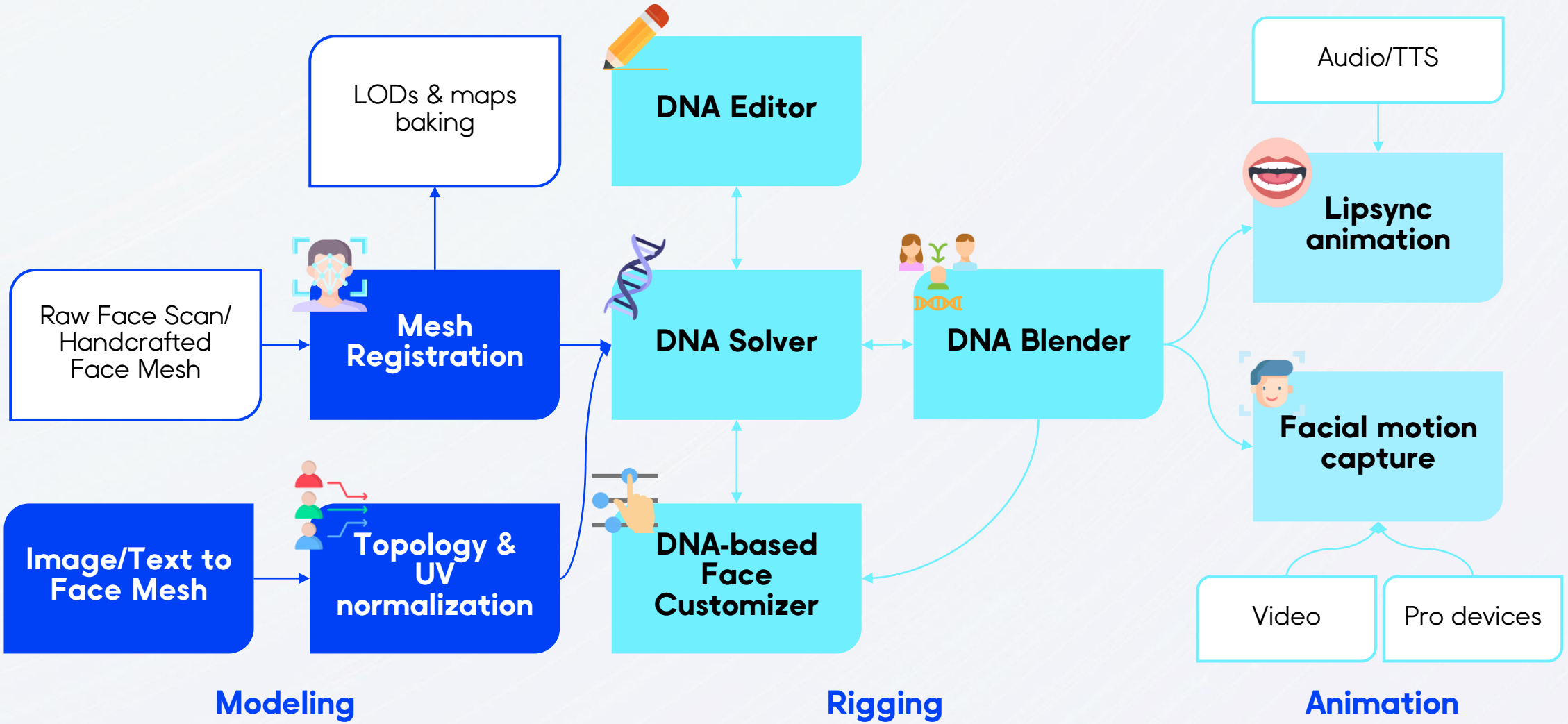
MetaHuman creator & mesh to MetaHuman

Modeling Rigging Animation

CyberHuman


Gen
Hair
Mix
Shapes
Lipsync
Image
Text
Type here...
Reset
Create
Menu
Anim
Save Param
Load Param

CYBERHUMAN PIPELINE

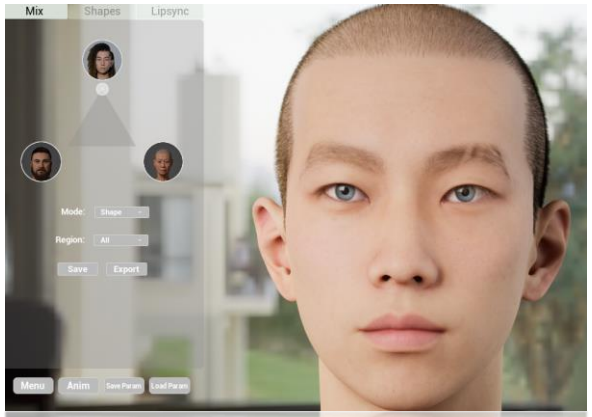


LIGHTSPEED STUDIOS | GDC

PART.02

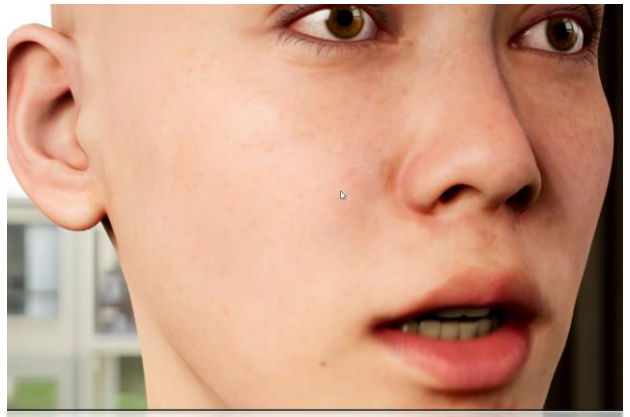
CHARACTER MODELING, RIGGING, AND ANIMATION PIPELINE

CYBERHUMAN PIPELINE



MODELING

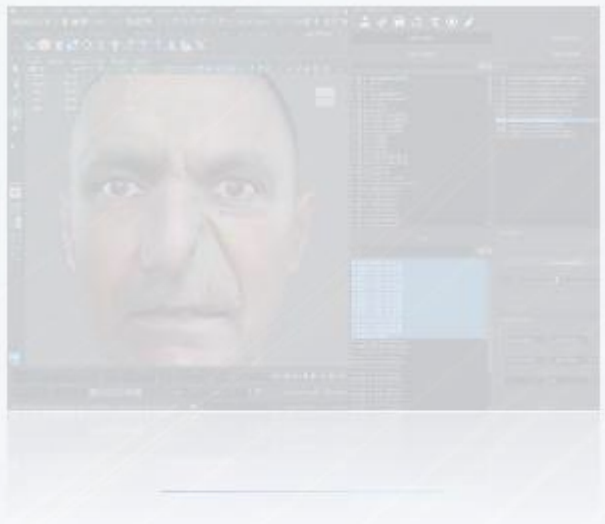
RIGGING



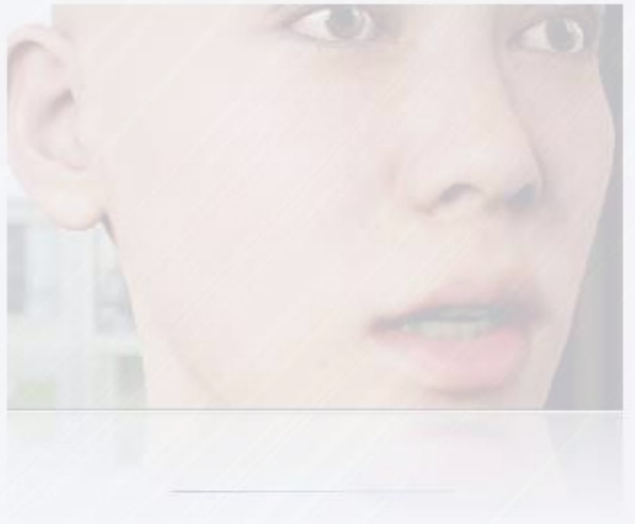
ANIMATION

CYBERHUMAN PIPELINE

MODELING



RIGGING



ANIMATION

MODELING FROM A RAW MESH

Fully automatic registration

One of the basic techniques in face related tasks

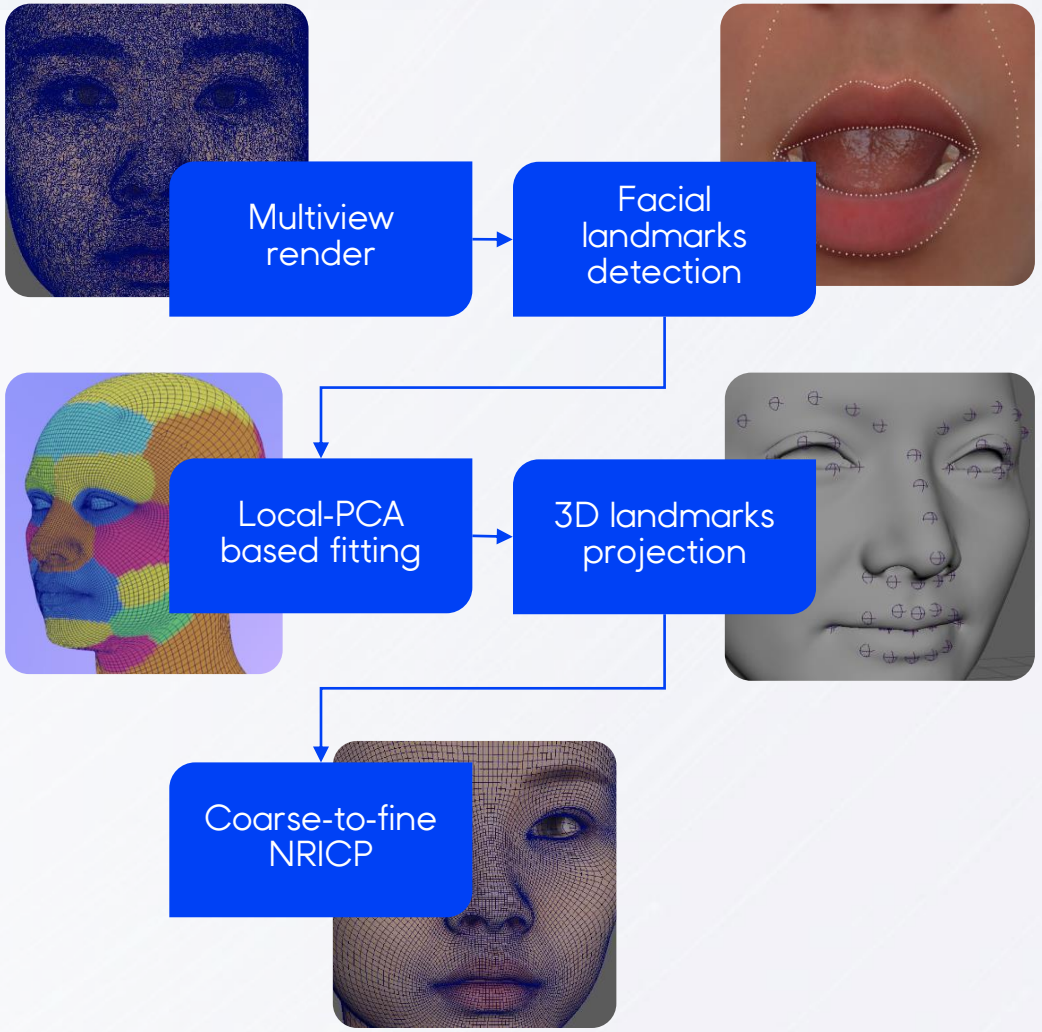
- Template topology to represent target geometry
- Applied to topology conversion of scan, auto rigging, lip sync, etc.

Problems in Wrap4D

- Artifacts caused by large face count differences
- Manual work needed and not applicable to automatic tools

Our advantages

- Facial landmark based fully-automatic registration
- Patch-PCA based fitting and 2D-3D landmarks detection
- Point-to-surface ICP and coarse-to-fine iterative optimization



MODELING FROM RAW MESH

Two-pass registration

Landmarks
- Fully automatic 2D-3D landmarks detection and projection

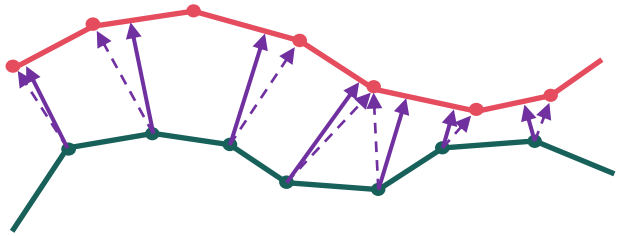
Patch-PCA fitting
- Fast PCA parameters optimization
- Very close to but not exactly the same as input shape

NRICP
- Time-consuming iterations and slow convergence
- Combined with PCA fitting
- Coarse-to-fine strategy

Constraints
- Point-to-surface ICP, 3D landmarks, patch smoothness, local smoothness, regularizations

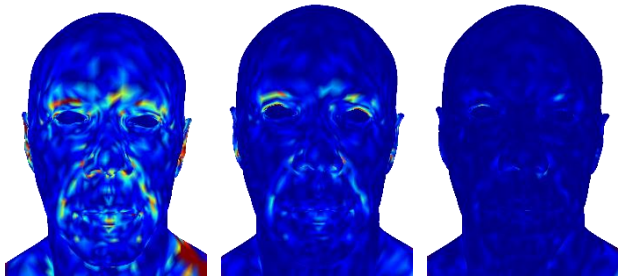
$$\min_{\tilde{\mathbf{v}}_1 \ldots \tilde{\mathbf{v}}_n} \quad E(\mathbf{v}_1 \ldots \mathbf{v}_n, \mathbf{c}_1 \ldots \mathbf{c}_n) = w_S E_S + w_I E_I + w_C E_C$$
$$\text{subject to} \quad \tilde{\mathbf{v}}_{s_k} = \mathbf{m}_k, \quad k \in 1 \ldots m$$

Point-to-surface ICP



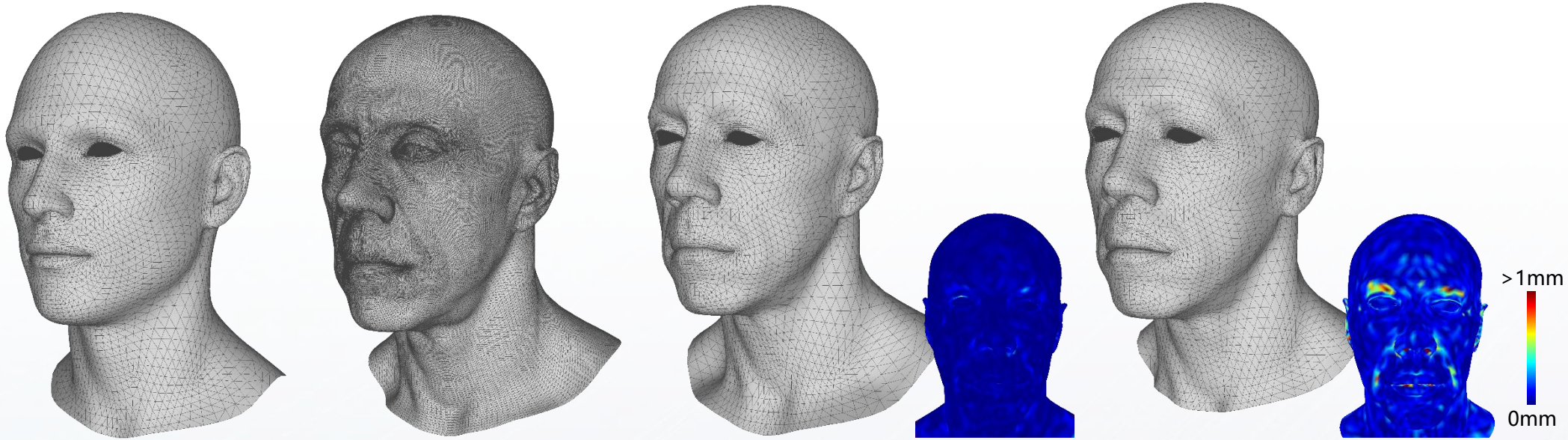
$$E_C = \sum_{i=1}^{n} \|\mathbf{v}_i - P(\bar{\mathbf{v}}_i)\|^2$$
$$\approx \sum_{i=1}^{n} \|(\mathbf{v}_i - \mathbf{c}_i) \cdot \mathbf{n}_i\|^2$$

Coarse-to-fine



MODELING FROM RAW MESH

Ours vs. Wrap4D

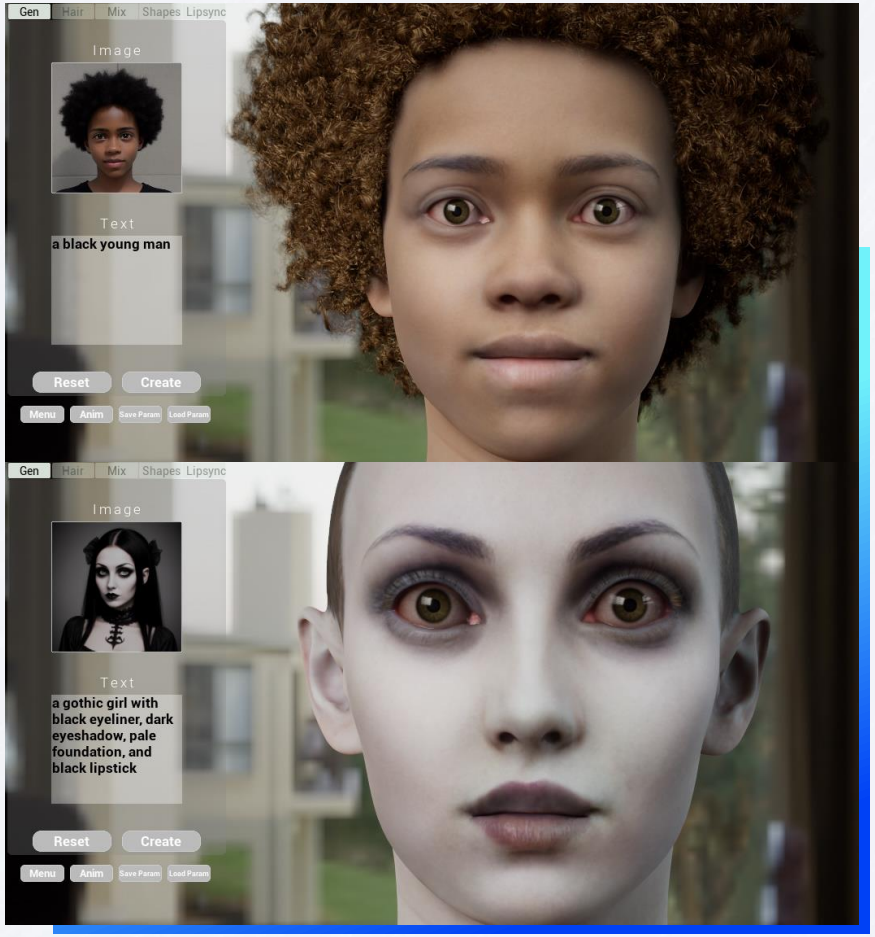


Template | Target | Ours | Wrap4D

	Mean error	Extension	DCC & UE integration	Automatic batch processing
Ours	≈0.07mm			
Wrap4D	≈0.09mm			


Gen
Hair
Mix
Shapes
Lipsync
Image
Text
Type here...
Reset
Create
Menu
Anim
Save Param
Load Param

MODELING FROM IMAGES OR TEXTS



MULTI-MODAL FACE CREATION MODULES

Image Diffusion & Normalization Model
- Prepares input images for subsequent processing by ensuring consistent formats and normalization.
- Improves the overall stability and robustness of the reconstruction process.

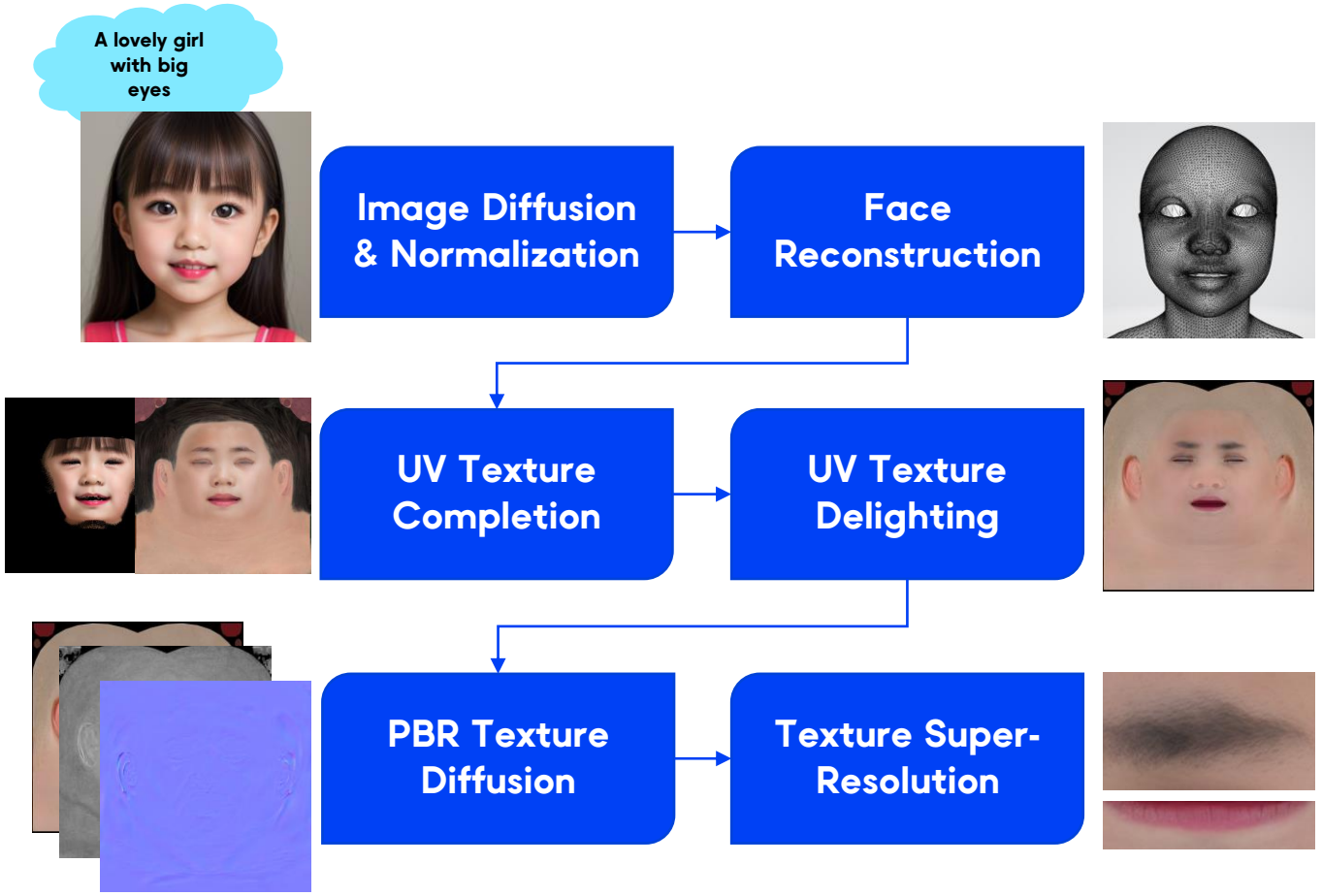
Face Reconstruction Model (BFM, FFHQ-UV)
- Leverages existing 3D face models (BFM, FFHQ-UV) to provide a foundation for reconstruction.
- Ensures realistic and anatomically accurate facial structures.

UV Texture Completion Model
- Fills in missing or incomplete regions of the UV map, ensuring texture coverage.
- Improves the overall completeness and consistency of the reconstructed face.

PBR Texture Diffusion Model
- Generates physically-based rendered (PBR) textures for the reconstructed face.
- Enhances realism by incorporating material properties and lighting effects.

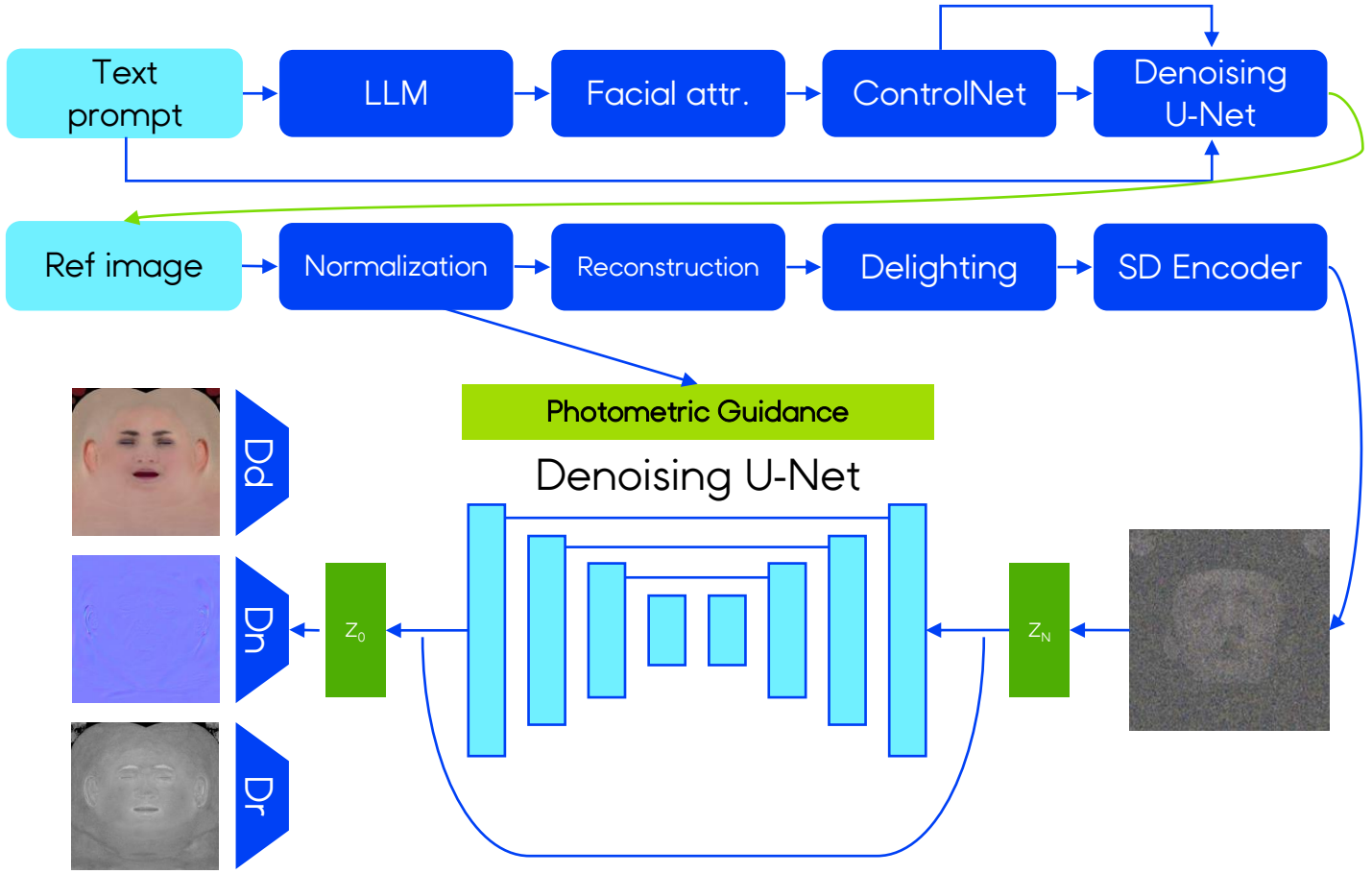
UV Texture Delighting Model
- Refines the generated textures by adjusting lighting and color balance.
- Optimizes the visual appeal and consistency of the final facial texture.

UV Texture Super-Resolution Model
- Enhances the resolution and detail of the UV textures.



FACE SHAPE AND TEXTURE RECONSTRUCTION

- Large language models are used to understand the textual intentions and extract facial attributes like face shape, hair style, etc. A control network then guides the generation of a reference portrait image
- Image attributes in w+ latent space of StyleGAN2 are edited to normalize lighting, eyeglasses, facial expression, etc
- Sampling is guided by photometric gradients to improve texture-mesh alignment and retain fine facial details
- High-quality physically based rendering (PBR) textures like diffuse, normal, roughness maps are generated by decoders
- 4x upscaling are applied to textures at the end of the pipeline



FACE TEXTURE DELIGHTING

UV lighting ground truth data generation

- Utilizes high-definition light stage scanned mesh geometry and associated texture maps for shading
- Employs random environment maps and camera positions for efficient rendering

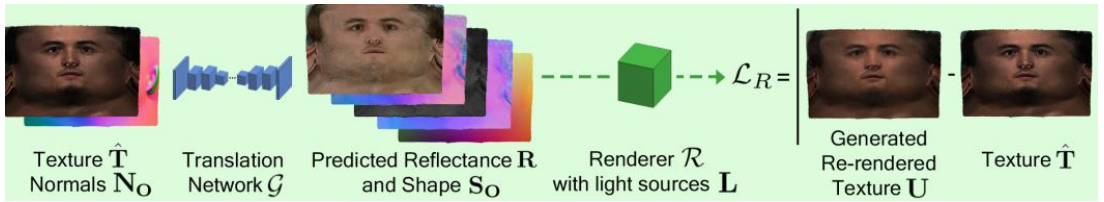
UV textures translation networks

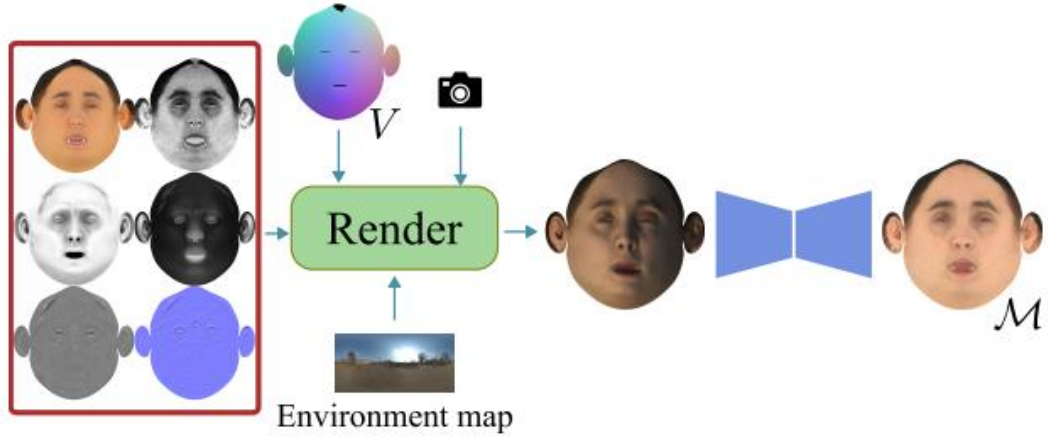
- Modified version of pix2pixHD with shape geometry as additional features
- Predictions are further enhanced by photorealistic differentiable rendering

Scalp area inpainting by Poisson blending

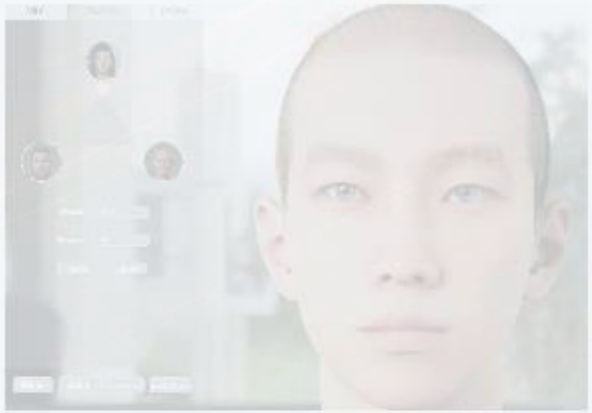
Synthetic lighting in UVs

Albedo GT



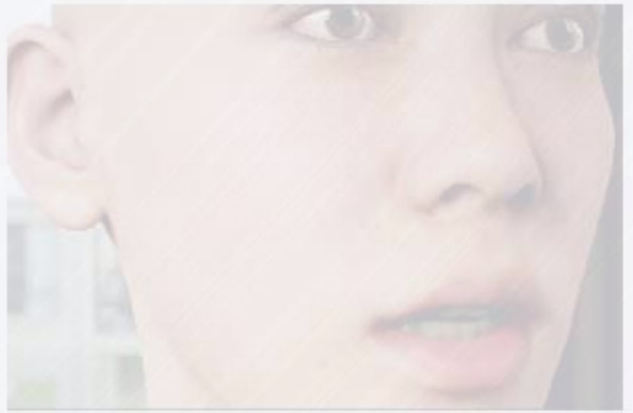


PIPELINE OVERVIEW



MODELING

RIGGING



ANIMATION

METAHUMAN DNA

Rig logic and DNA

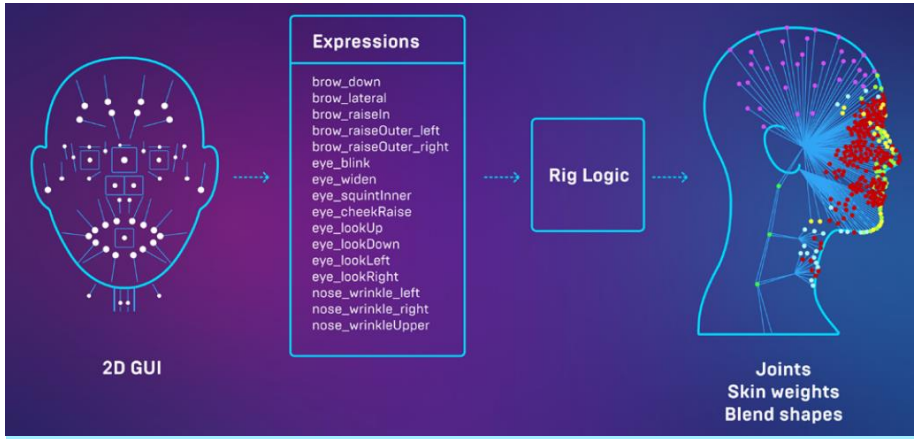
Rig logic

- Runtime facial rig evaluation solver system developed by 3Lateral
- Relying on a universal set of rules for defining the muscular system of a human face
- Runtime evaluation, reduction of parameters, lossless animation compression, reusability, non-linear animation mixing, etc.

DNA

- Storage format of the complete description of a 3D object's rig and geometry
- CRS matrix, joint groups, corrective expressions

Rig logic UI



Data flow from rig logic to final expression

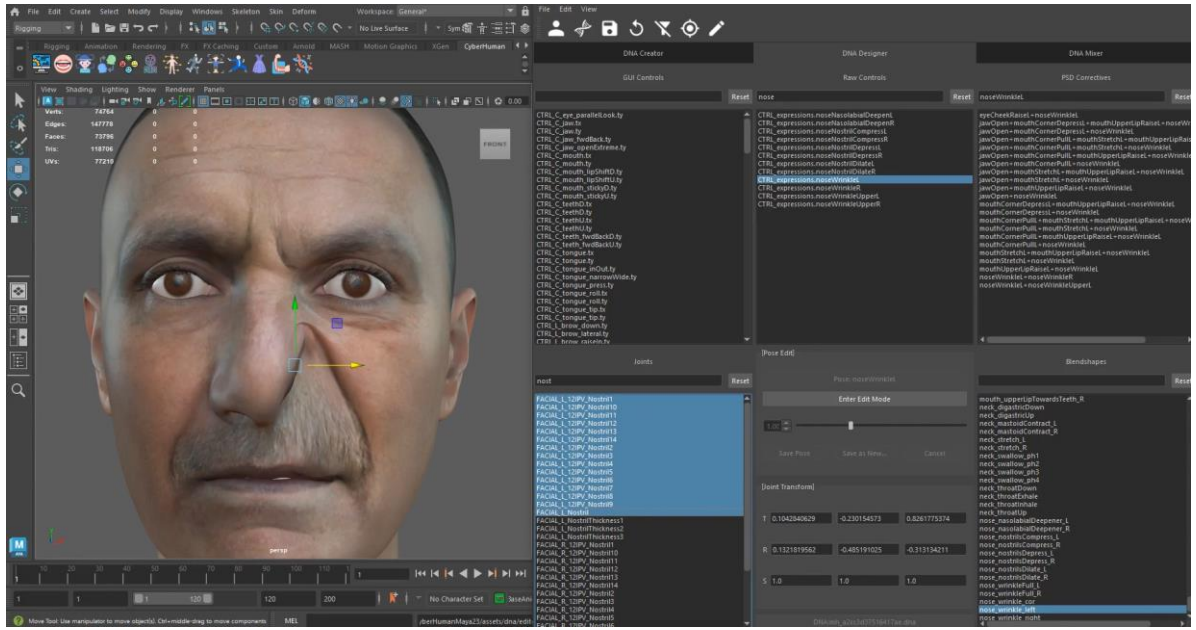
DNA STUDIO

DNA studio vs MetaHuman tools

DNA studio

- Efficient and flexible construction and manipulation of MetaHuman DNA
- Toolset: DNA creation, DNA blend, face customization, pose edit

	DNA studio	MetaHuman creator & Mesh to MetaHuman
Platform	UE and Maya plugin	UE and Web service
Rig quality	High	High, but sometimes with artifacts
DNA creation efficiency	About 10~15s	About 30~60s
DNA blend	Regional blend, texture supported	Regional blend, texture unsupported
Face customization	Supported & user-defined	Supported & predefined
Pose edit	Supported	Unsupported



DNA STUDIO

DNA solver

Efficient and high-fidelity DNA generation

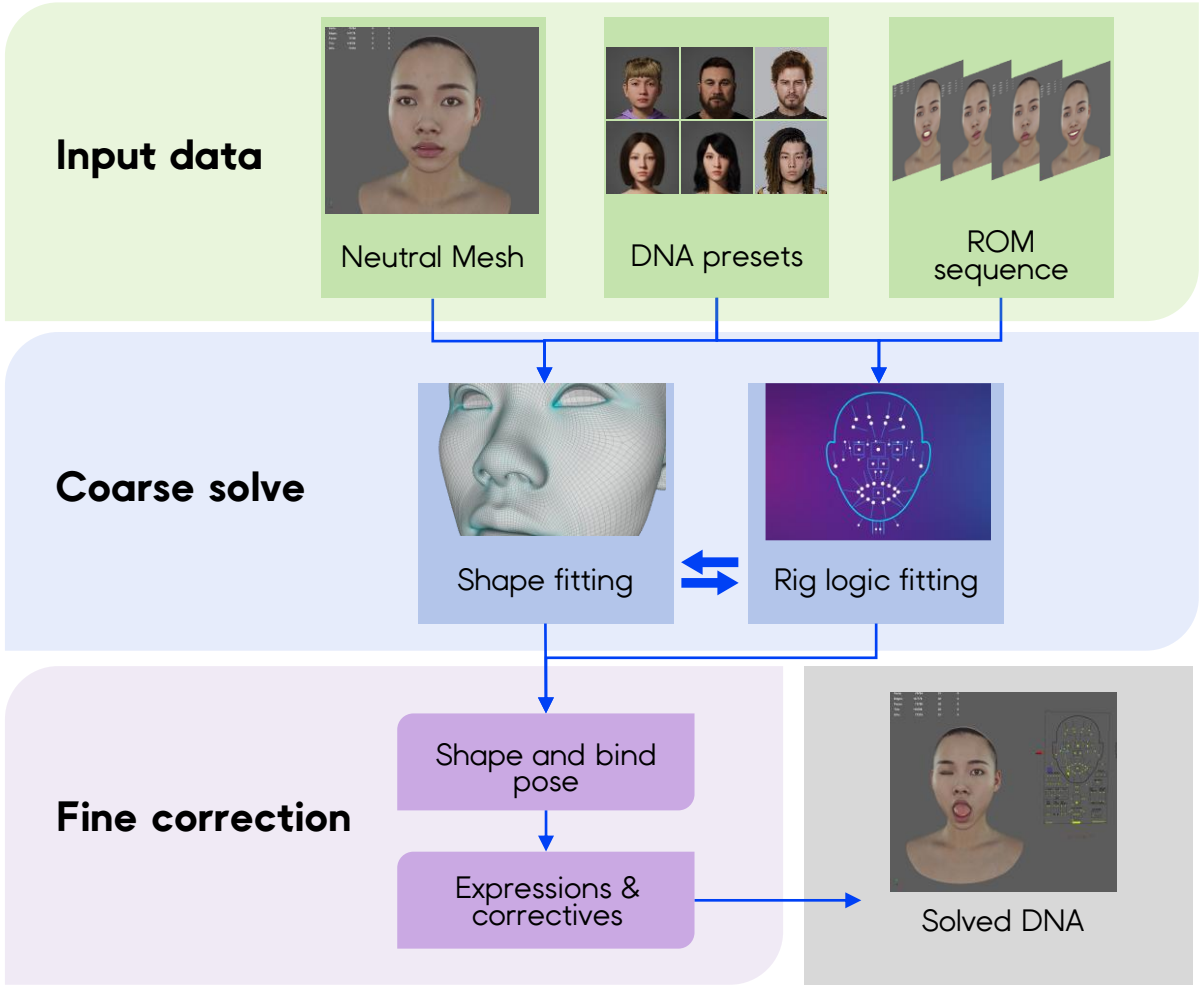
- Neutral face mesh or ROM sequence (MetaHuman topology) as input

Iterative coarse DNA solve by 100+ DNA presets

- Shape: landmark and ICP based regional blend parameters fitting according to neutral mesh
- Rig logic: DNA controls and joint-group blend parameters optimization according to ROM meshes

Fine DNA correction by auto-rigging

- Shape, face parts and bind poses refinement by RBF-based deform
- Expressions and correctives refinement based on SSDR



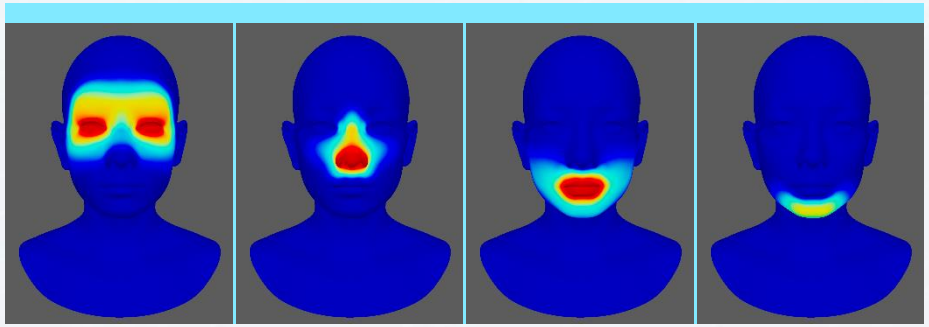
DNA blender

Fast generation of various high-quality characters

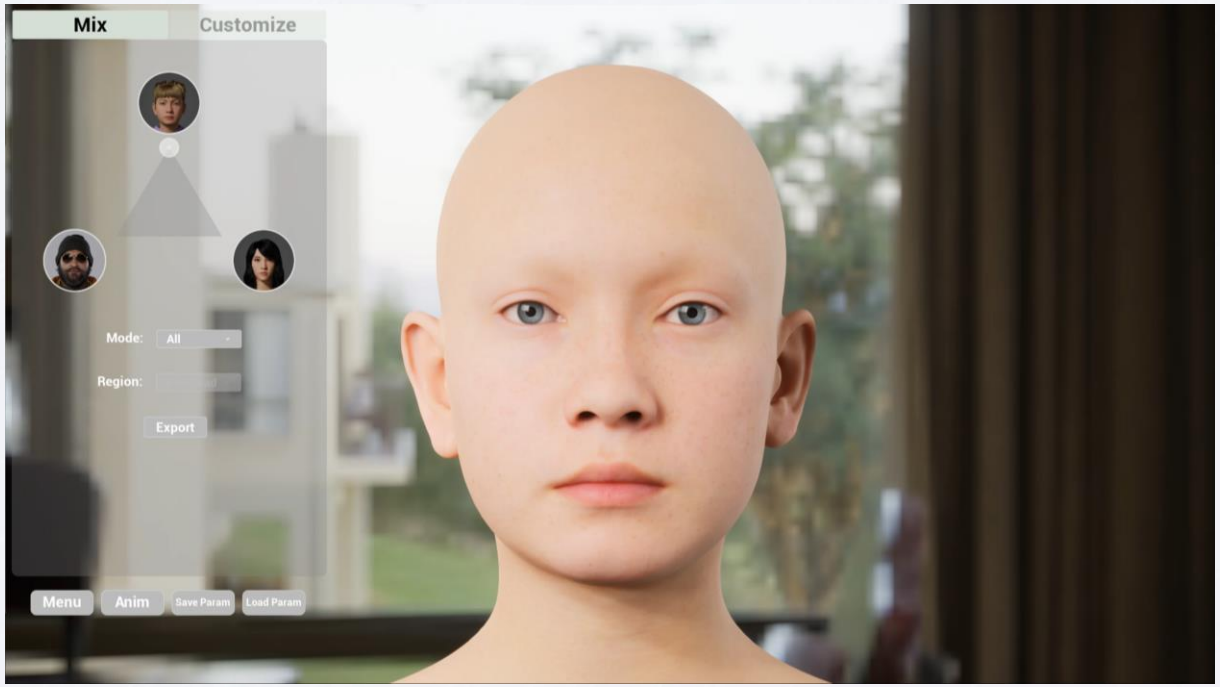
- Flexible and intuitive regional blend for both artist and game players
- Maya & UE supported

Blend weights

- Vertices: skinning weights of joint group sets
- Joints: weighted sum of influencing vertices



Vertex blend weights



DNA blend in UE

DNA STUDIO

LIGHTSPEED STUDIOS | GDC

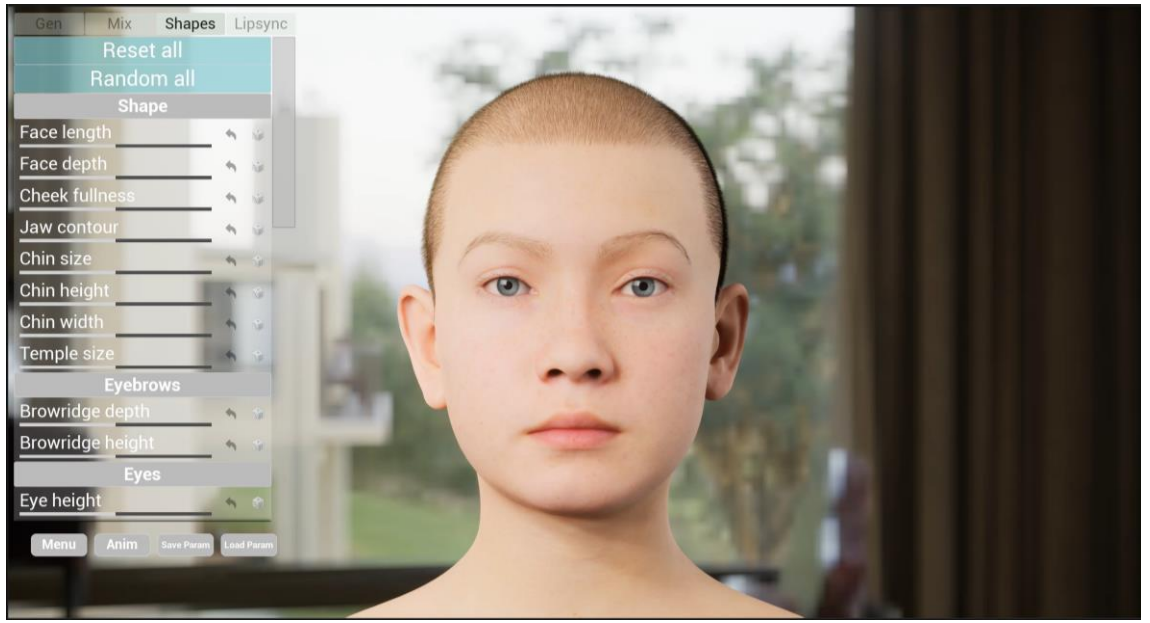
DNA-based face customization

Traditional joint-based techniques

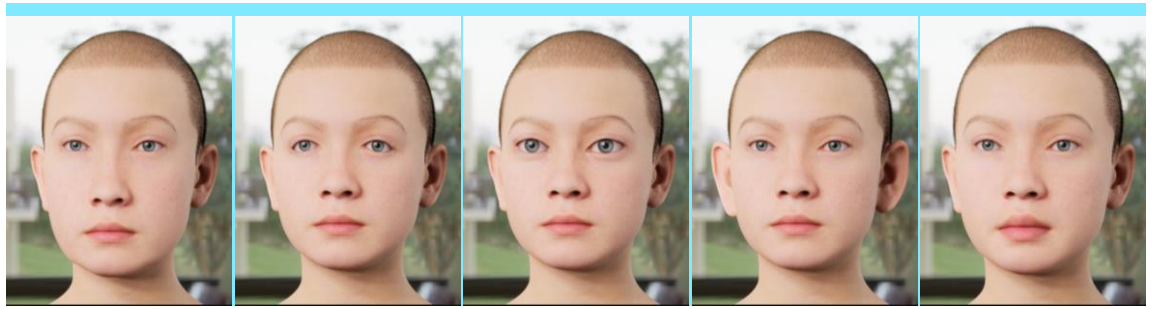
- Artifacts caused by multiple activated controllers
- Hard-coded pose combination logic and not artist-friendly

Our advantages

- Generalized DNA structure to control both face shape and expression
- Face customization DNA poses generated by blendshapes and joint poses
- Compact DNA data storage and runtime SIMD optimization
- Maya & UE supported



Customization in UE

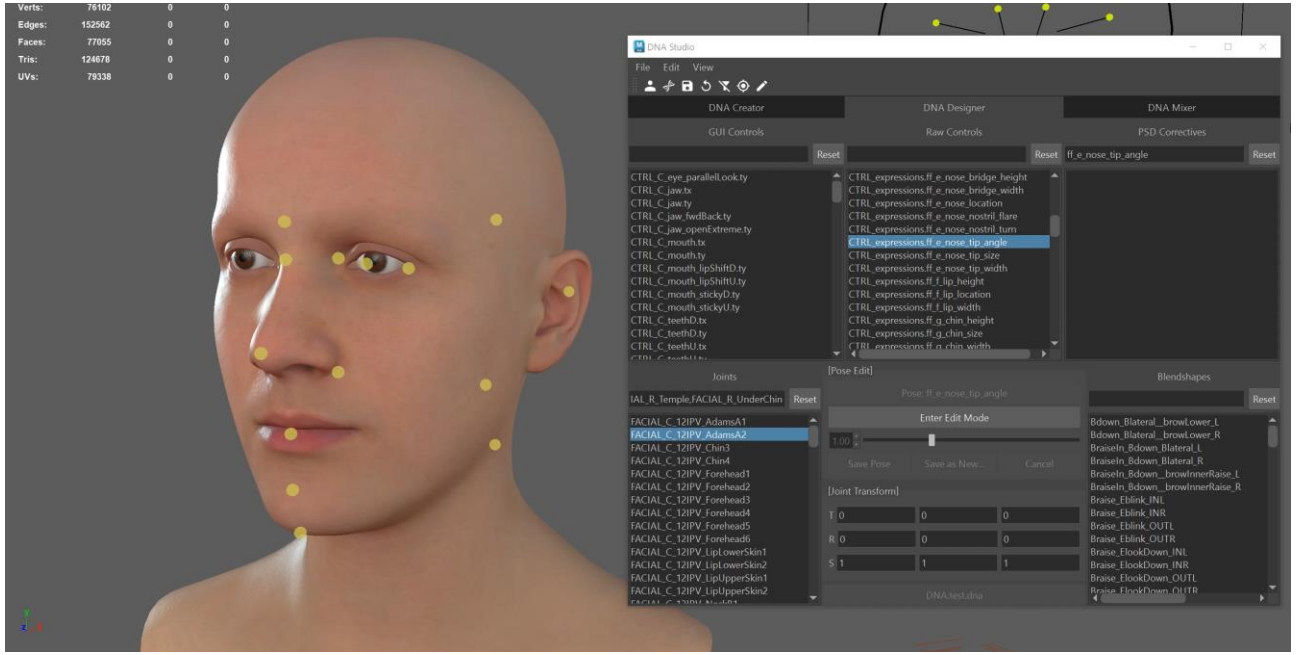


Face length | Eye tilt | Eye scale | Ear size & turn | Lip thickness

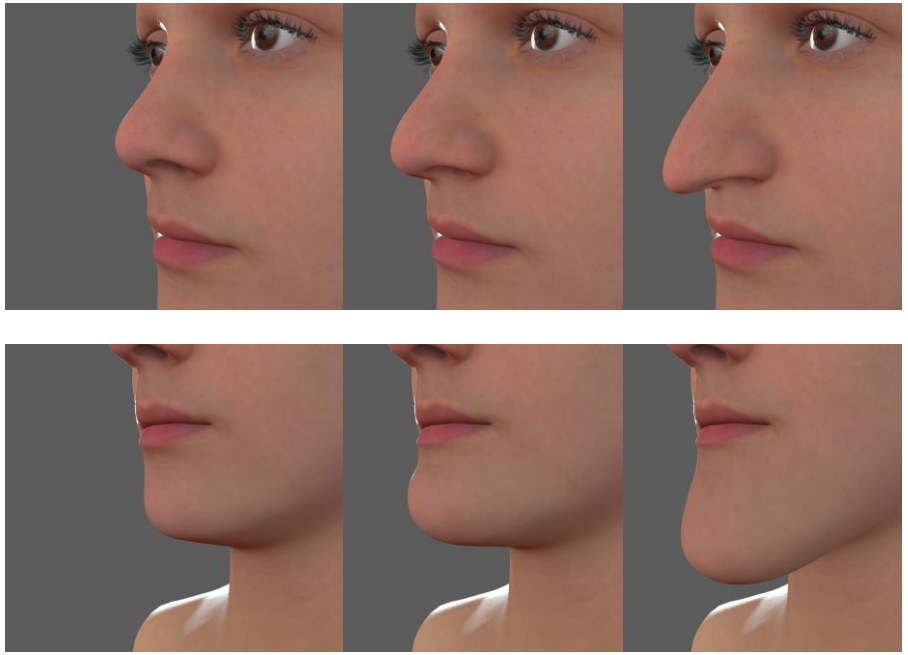
DNA edit

Efficient and direct edit of DNA poses

- Unified edit for both shape (face customization) and expression
- User-defined exaggerated and characterized poses

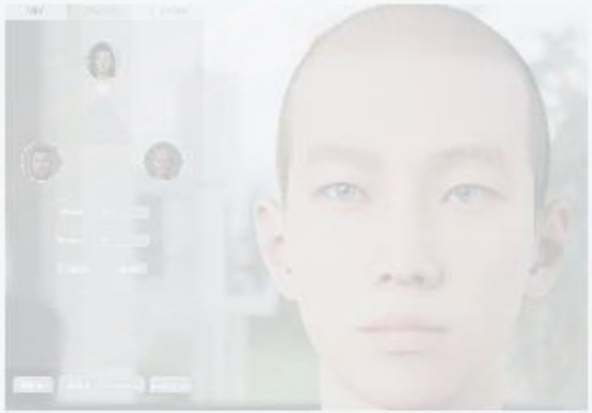


Pose modified for face customization

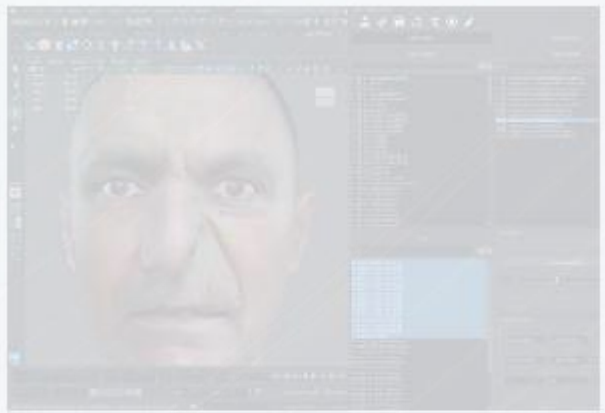


Neutral | Predefined pose | Edited pose

PIPELINE OVERVIEW



MODELING



RIGGING

ANIMATION

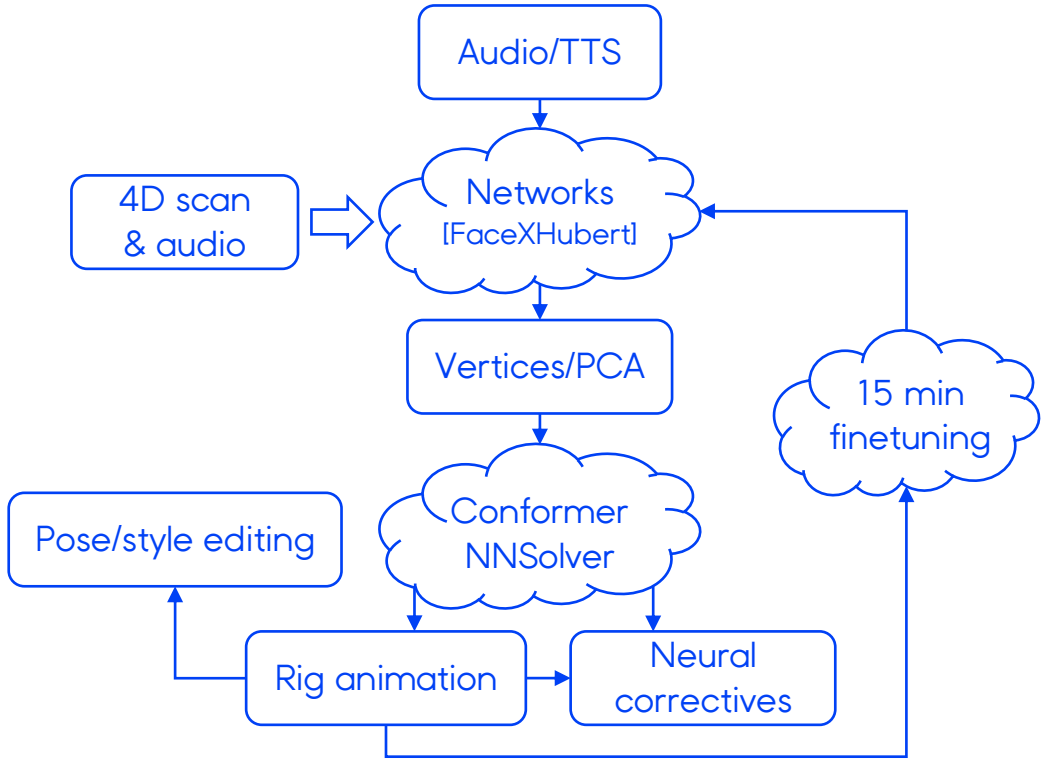

Gen
Mix
Shapes
Lipsync
-Emotion-
Main
Neutral
Gen Emotion
View1
None
View2
None
View3
None
View4
None
-Mode-
Input
-Strength-
Neutr
-TTS-
en-US-ElizabethNeural
transitions between states. This, in turn, substantially reduces the amount of time needed to animate a sequence.
ENTER
Export Anim
Select Audio...
A:"Motion Matching is a simple yet powerful way of animating characters in games. It is an algorithm that plots the future trajectory of the animated character. The software can then match the correct animation with the movement input from the user. Compared to other methods, it
Menu
Anim
Save Param
Load Param

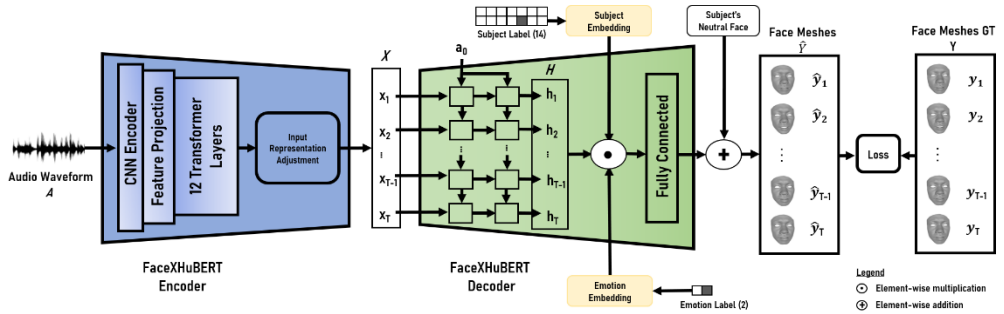
LIP SYNC ANIMATION

Speech animation from text or audio

Single mesh to audio-driven talking head

- High performance for interactive application, e.g. GPT-driven dialogs
- Lip vertices prediction with rich facial details
- Emotional speech animation supported
- High-quality neural solver of MetaHuman control rig parameters
- Neural correctives as supplementary facial expressiveness
- Maya and UE plugins for asset creation and runtime animation





LIP SYNC ANIMATION

Full-fledged runtime and editor

Main window emotion & Gen emotion
- Emotion applied, AI-generated or animation preset

Sub window emotion
- Emotion applied in sub-viewport

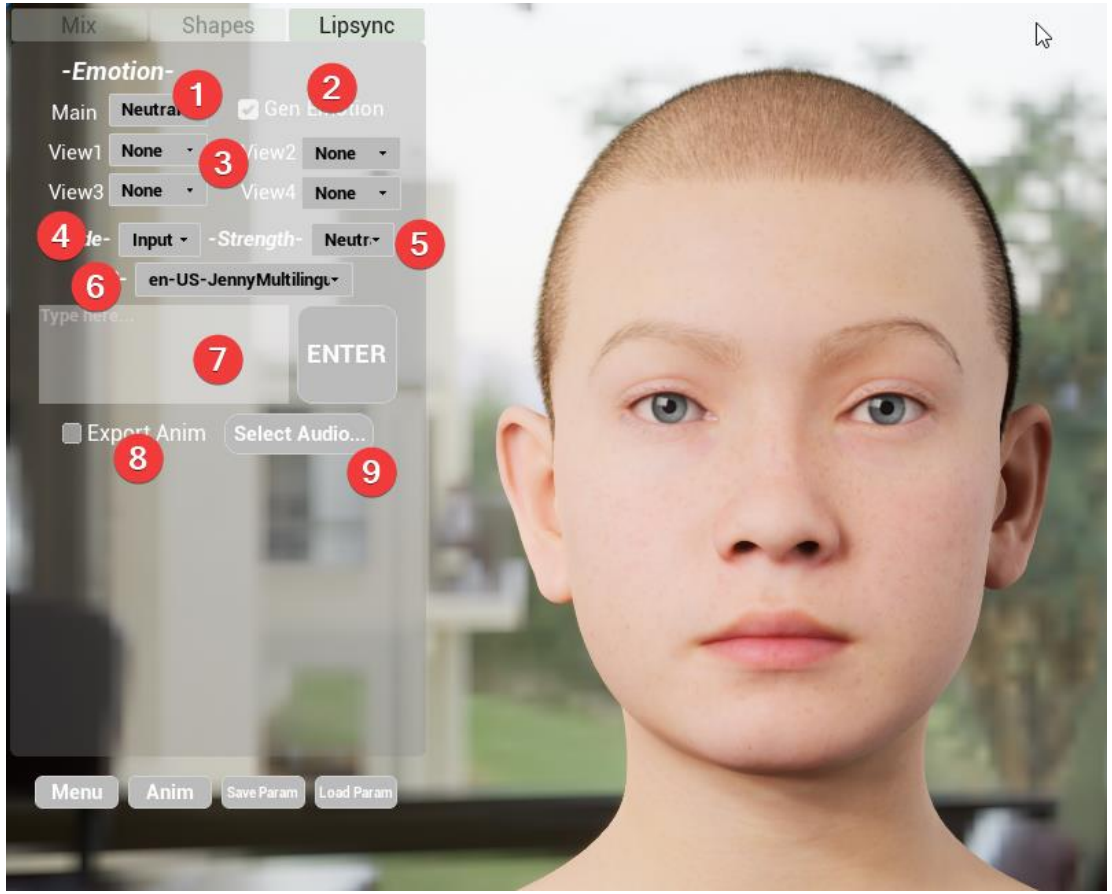
Lip sync mode
- Text inputs or chats (Google Gemini by default)

Lip sync intensity
- Neutral, strong or mild

TTS timbres
- Azure TTS timbres

Audio selection
- Selected audio to lip animation

Animation export
- Generated animation and audio exported to animation sequence



Lip sync UI in UE

FACIAL MOTION CAPTURE

LIGHTSPEED STUDIOS | GDC

RGB video stream to facial expression parameters

DCC

- Convenient asset capture and curve export for artist
- MetaHuman control rig and ARKit blendshapes supported

UE runtime

- Realtime capture for PC, IOS, Android
- Lightweight network performance, e.g. Huawei Mate30 5-8ms CPU
- Dynamic model loading based on mobile configuration

Facial capture in Maya

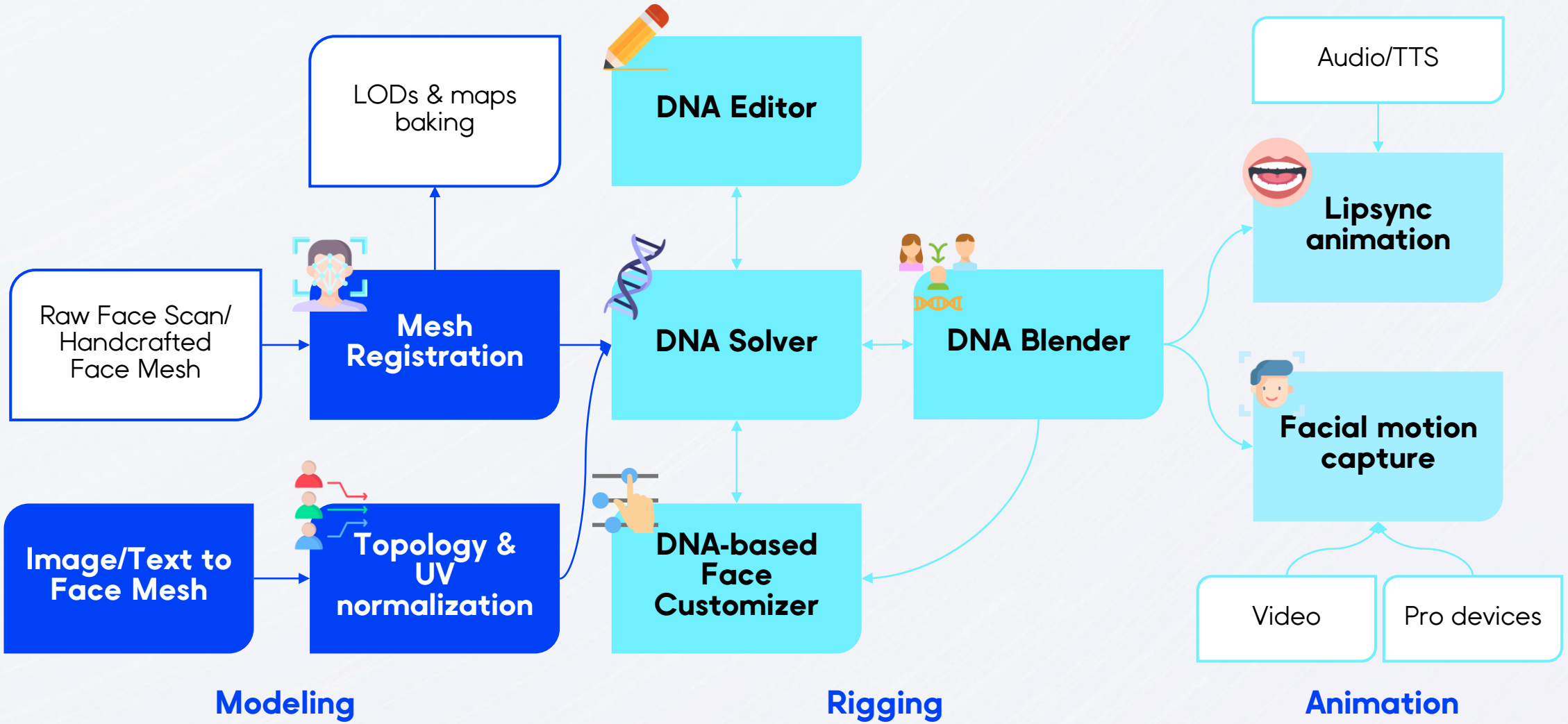
Facial capture in Undawn
(developed by Lightspeed studio)

LIGHTSPEED STUDIOS | GDC

PART.03

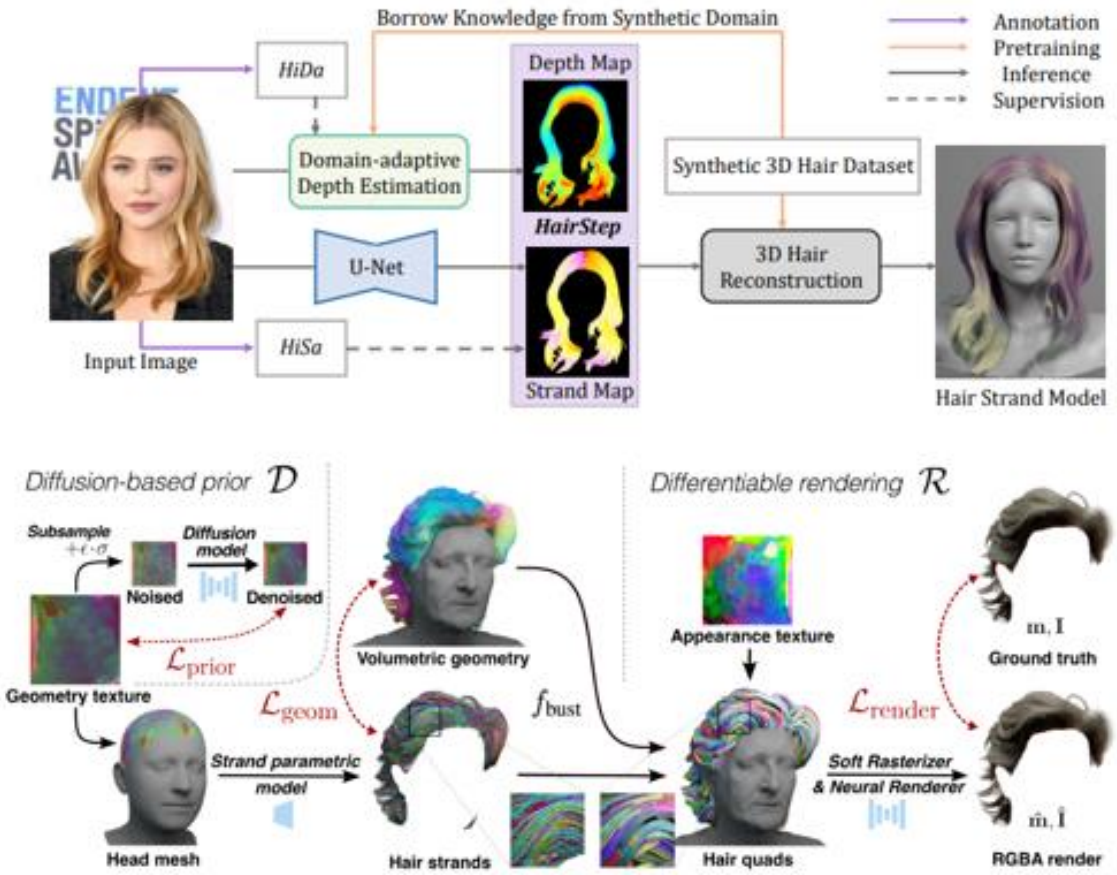
SUMMARY AND FUTURE WORKS

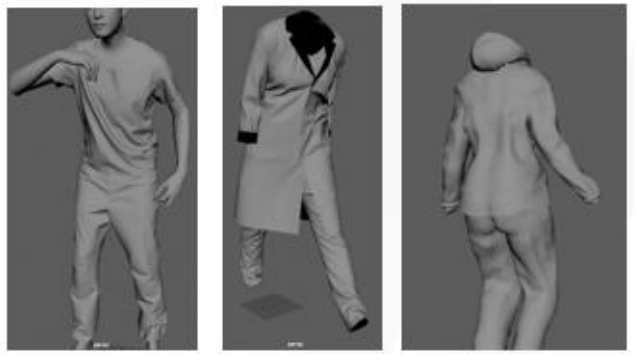
CYBERHUMAN PIPELINE



FUTURE WORKS

- Hair generation
- Body shape customization
- Cloth generation





THANK YOU

GDC 2024 | SAN FRANCISCO