GDC | March 21-25, 2022
San Francisco, CA

New Graphics Features for Forspoken

Keiji Matsuda
Programmer
Luminous Productions

LUMINOUS PRODUCTIONS

#GDC22

Agenda

- Overview
- Rendering Pipeline
  - Model Rendering
  - Shadows
  - Lighting & Post Effect
  - Optimization
- Automatic LOD generation

Overview

Forspoken

- An open world game for PS5 and PC.
- Follows the journey of the protagonist, Frey, who is magically transported to the vast land of Athia.

Our Game Engine

Basic Rendering Features

LUMINOUS ENGINE™

- Terrain rendering
- Curve model rendering
- Tile-based light culling
- Physically based rendering
- Cascaded shadow maps
- Punctual light shadow maps
- Screen space shadows
- Ray-traced shadows
- Procedural sky
- Specular light probes
- PRT volumes
- Screen space ambient occlusion
- Ray-traced ambient occlusion

- Physical Sky
- Volumetric light(Fog)
- Volumetric clouds
- Auto Exposure
- Color correction
- Depth of field
- Motion blur
- Glare
- Lens flare
- Temporal anti-aliasing
- Wide-gamut rendering

Basic Rendering Features

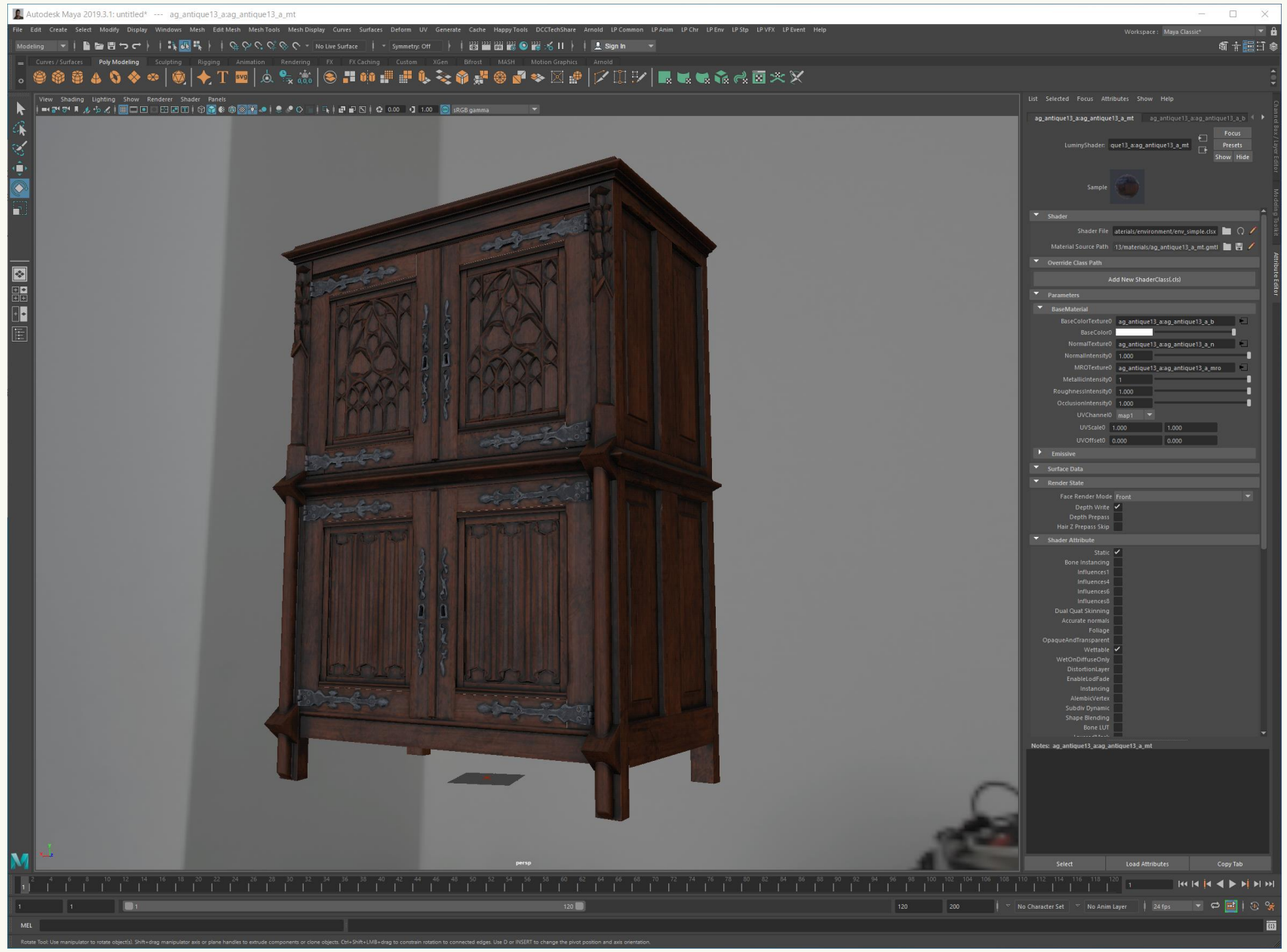
LUMINOUS ENGINE™

- Graph-based shaders
- Graph-based VFX

Basic Rendering Features

LUMINOUS ENGINE™

- Maya material plugins



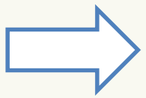
Shading

- Lambert diffuse
- Torrance-Sparrow specular
  - GGX Distribution term
  - Schlick-Smith Visibility term
  - Schlick Fresnel term
  - Multiple scattering BRDFs [STEPHEN19]
- Roughness/metallic/specular control
- Deferred rendering
  - Forward rendering for transparent or special BRDFs

AMD FidelityFX

[AMD]

- Combined Adaptive Compute Ambient Occlusion (CACAO)
- Screen Space Reflections
- Downsampler
- Contrast Adaptive Sharpening
- Super Resolution
- Denoiser
- Variable Shading

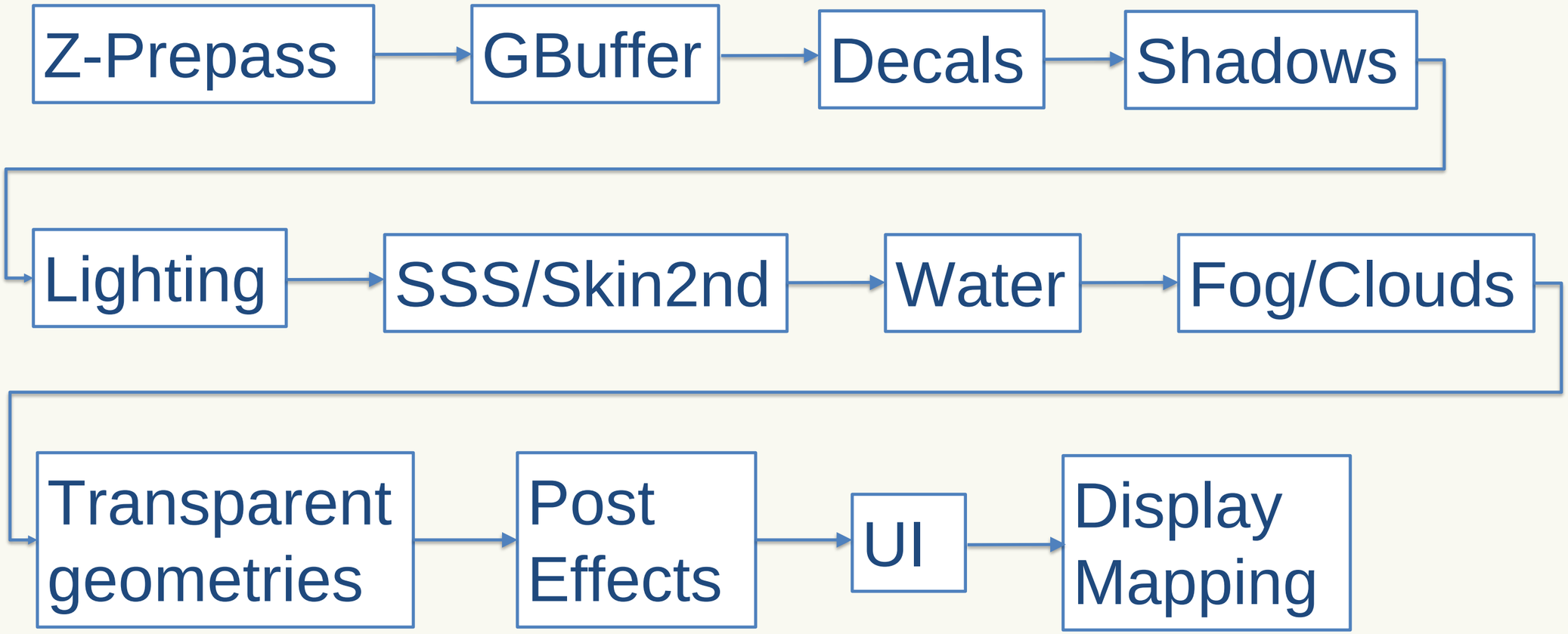


March 23 17:00-18:00
Breaking Down the World of Athia: The Technologies of Forspoken.

FORSPOKEN™

Model Rendering

Render Pipeline



Z-Prepass

- Reduces overdraw by only drawing depth before GBuffer pass.

- Only opaque objects near the camera are drawn. Excluding:
  - Distant objects
  - Cutout objects
  - Dynamic objects

GBuffer

	Format	R	G	B	A
RT0	RGBA16F or R11G11B10	Emissive/Lighting			-
RT1	R10G10B10A2	Octahedral Normal		Roughness(8) Flags(2)	Flags
RT2	R8G8B8A8	Albedo			Occlusion
RT3	R8G8B8A8	BRDF parameters (depends on Flags in RT1)			Specular(7) Indoor(1)
RT4	RG16F	Velocity		-	-

GBuffer Normal Precision

- 24-bit (12-bit octahedral normal encoding) was used previously, but 20-bit yielded sufficient results.

	Format	R	G	B	A
Old RT1	R8G8B8A8	Octahedral Normal			Roughness

	Format	R	G	B	A
RT1	R10G10B10A2	Octahedral Normal		Roughness(8) Flags(2)	Flags

GBuffer -BRDF Parameters-

- Special material BRDF parameters

Usage(RT1 flags)	RT3.r	RT3.g	RT3.b
Default	Metallic	-	-
Skin	(Metallic)	Hue shift	SSS Strength
Hair	Tangent		2nd specular
Backscatter	Thickness	Hue shift	Distortion(4) Ambient(4)
Terrain Blend	Metallic	Blend rate	-

Position Precision

- Particular attention paid to position precision in order to generate a vast world
  - Mapshift
    - Resets the coordinate system to (0,0,0) with respect to the current camera position
  - Reversed-Z [NVIDIA16]
    - Dramatic improvement in the precision of distant views
  - Improved matrix computation

Matrix Computation Improvement

$$\mathrm{Vertex}' = \mathrm{ProjectionMatrix} * \boxed{\mathrm{ViewMatrix} * \mathrm{WorldMatrix}} * \mathrm{Vertex}$$

- No View Projection matrix is used
- Calculate only the translation of the View first, and keep the calculation of large values and the rotation components separate

Removing Big Translation Values

$$\text{ViewMatrix} * \text{WorldMatrix} * \text{Vertex}$$

$$= \begin{bmatrix} v_{00} & v_{01} & v_{02} & v_{03} \\ v_{10} & v_{11} & v_{12} & v_{13} \\ v_{20} & v_{21} & v_{22} & v_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_{00} & w_{01} & w_{02} & w_{03} \\ w_{10} & w_{11} & w_{12} & w_{13} \\ w_{20} & w_{21} & w_{22} & w_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} v_{00} & v_{01} & v_{02} & 0 \\ v_{10} & v_{11} & v_{12} & 0 \\ v_{20} & v_{21} & v_{22} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_{00} & w_{01} & w_{02} & (w_{03} - V_x) \\ w_{10} & w_{11} & w_{12} & (w_{03} - V_y) \\ w_{20} & w_{21} & w_{22} & (w_{03} - V_z) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Big values

$(V_x, V_y, V_z) = \text{ViewPosition}$

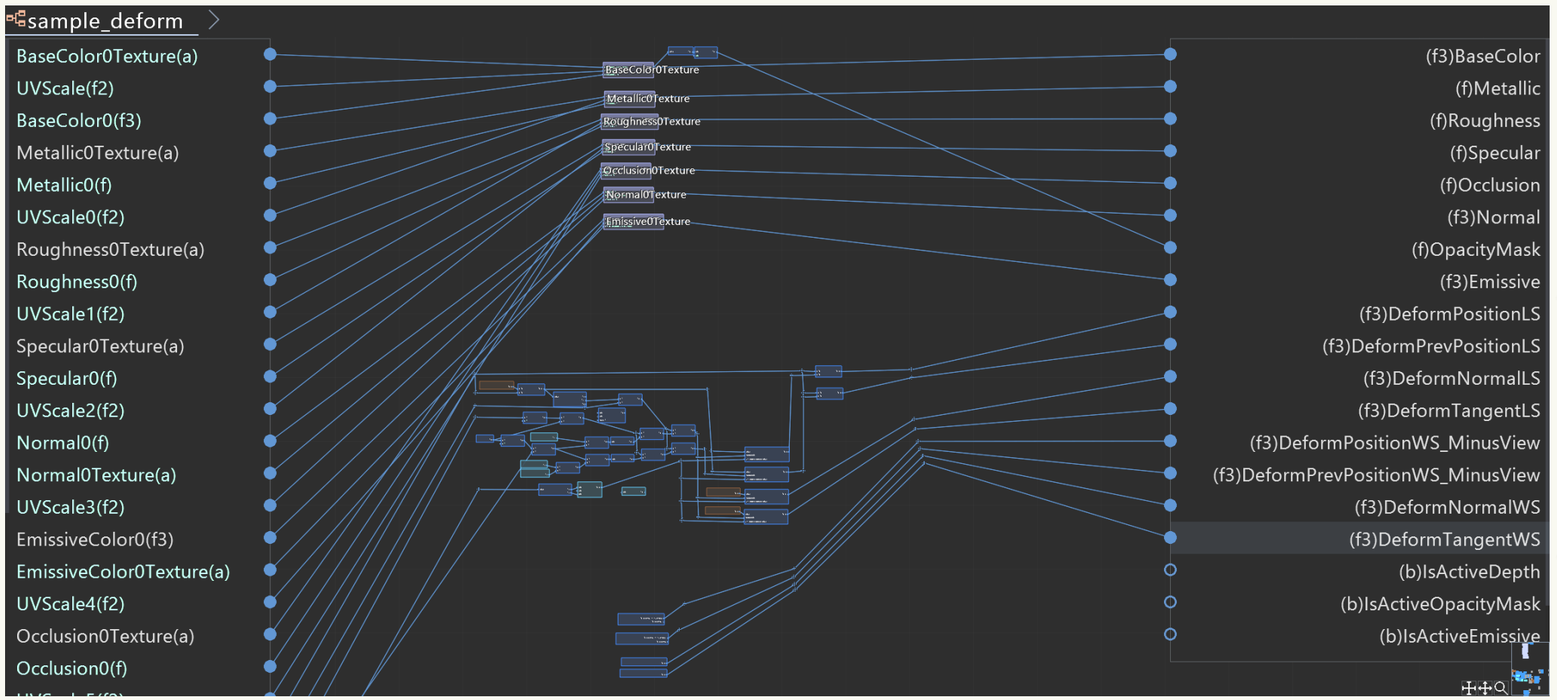
Matrix Computation Improvement

- The View Projection matrix in the shadow pass caused shadow flickering.

- Velocity precision (TAA and motion blur) was also affected.

Graph-Based Vertex Shaders

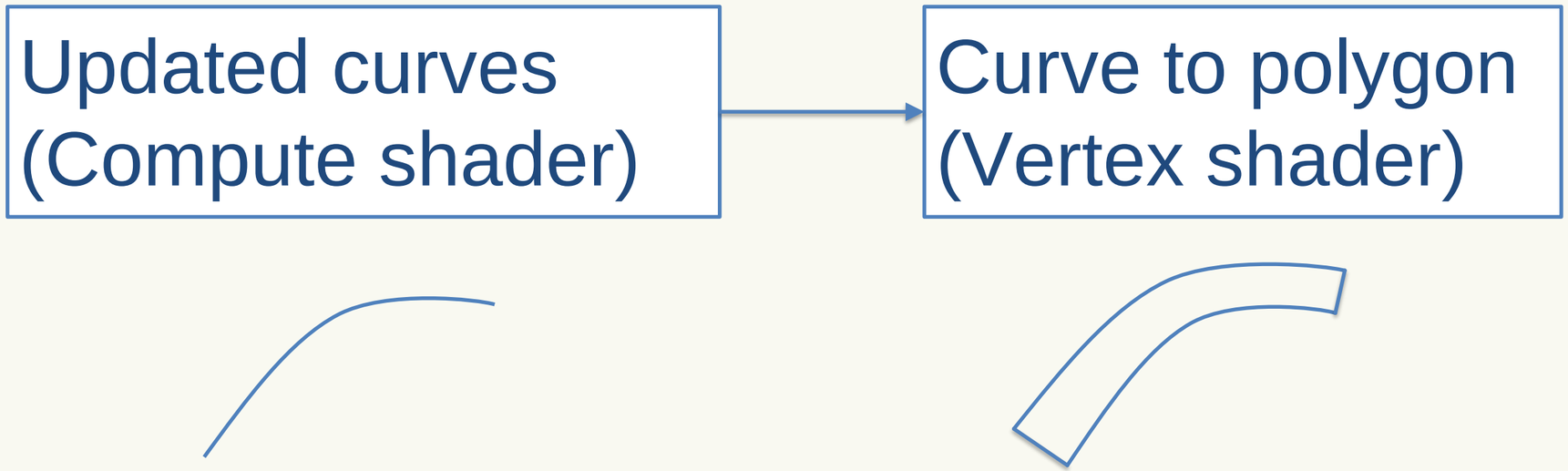
- In the shader graph, there are position outputs in local space and view centered world space.



Local space
and
View centered world space

Luminous Hair

- A hair rendering system that utilizes billboards that are guided by the curves



Luminous Hair

- Curly/wavy hair
- Hair thickness
- Attributes
  from the scalp
- Ornatrix Hair

Luminous Hair

- Simple polygon hair for NPCs
  - Using compute shaders on hair for a large number of character is too costly.
  - Luminous Hair delivers better-quality results, but at a higher polygon count.

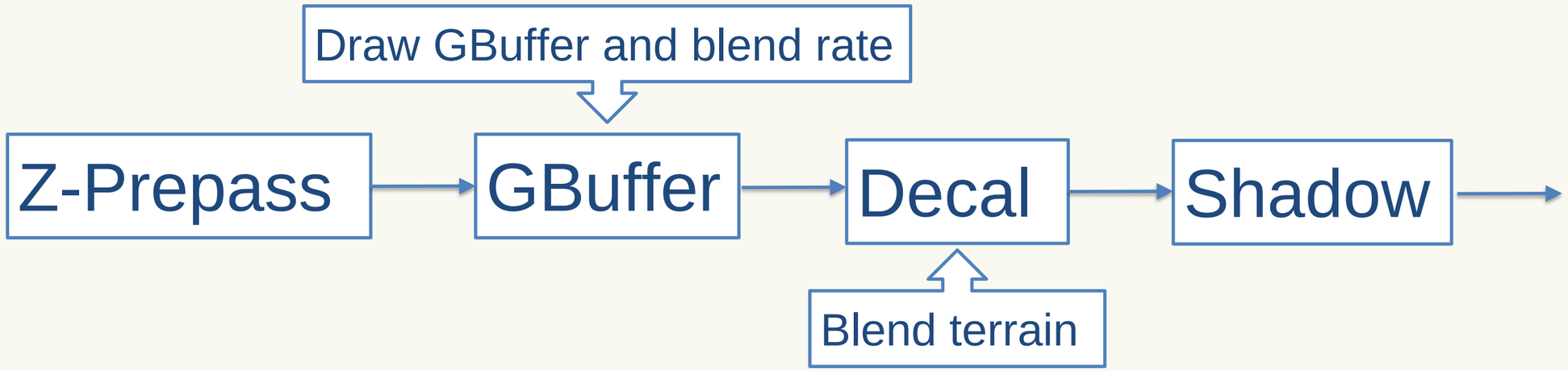
Terrain Blending

- Blending static models and terrain.

Terrain Blending -GBuffer-

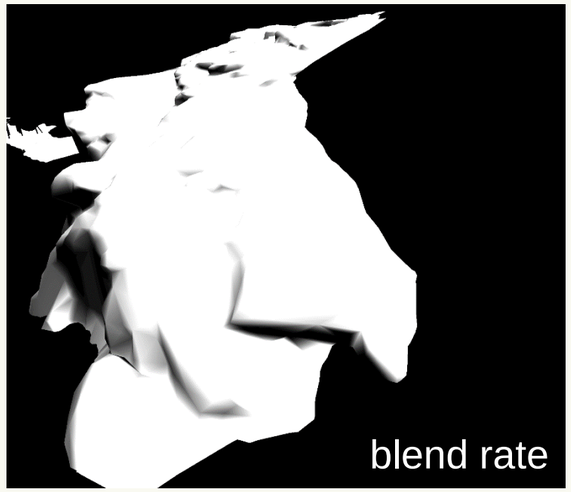
- Store terrain blend rate in GBuffer pass

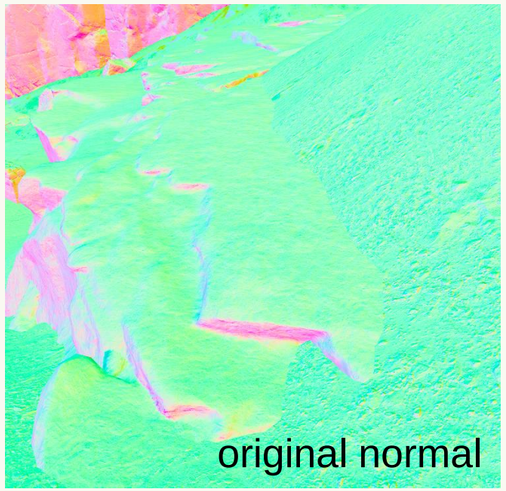
Usage	RT3.r	RT3.g	RT3.b
Terrain Blend	Metallic	Blend rate	-



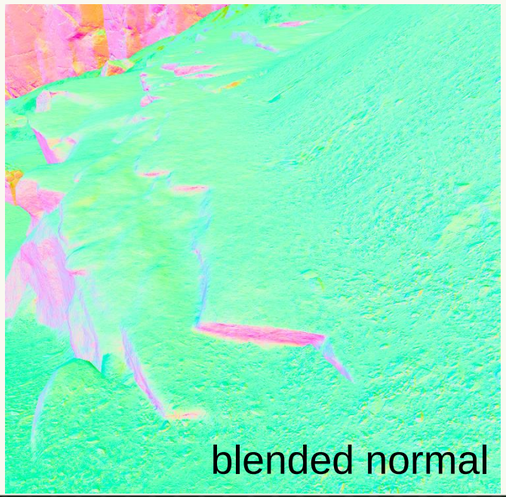
Terrain Blending -Blend Rate-

GBuffer





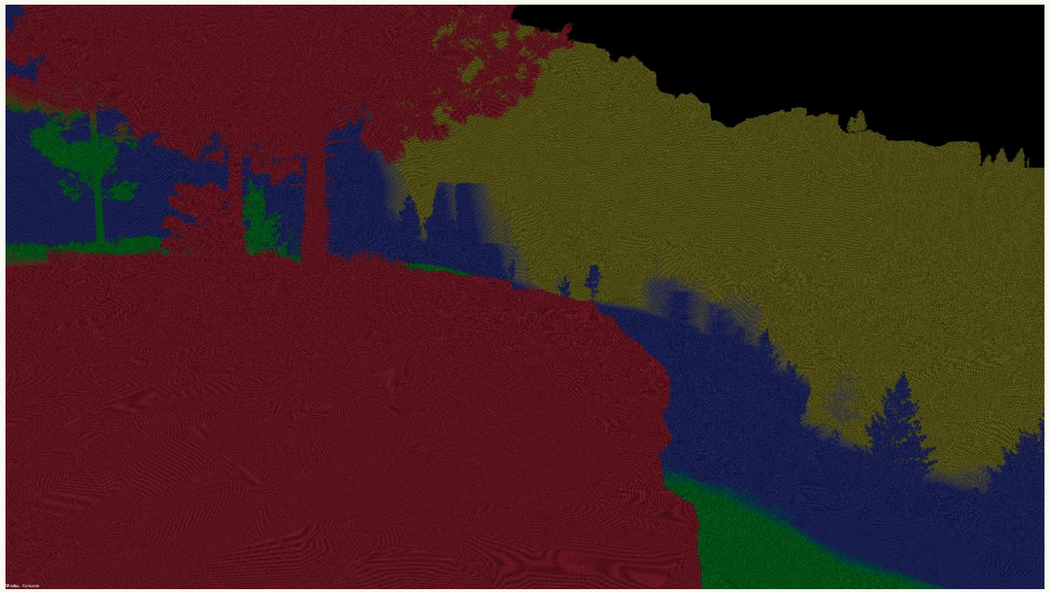
Terrain-blended GBuffer



Shadows

Directional Shadows

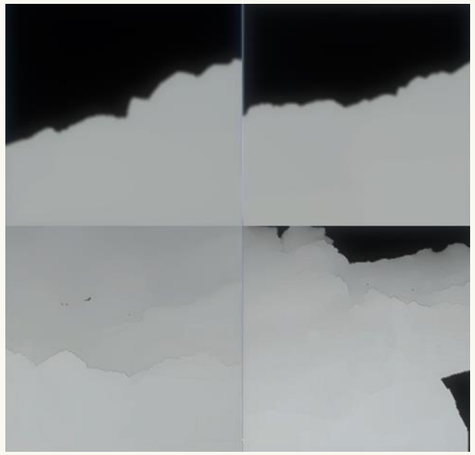
- Four 2048x2048 cascaded shadow maps are used to generate shadows across a broad area.



- Optimized using Static Shadows and Screen Space Shadows

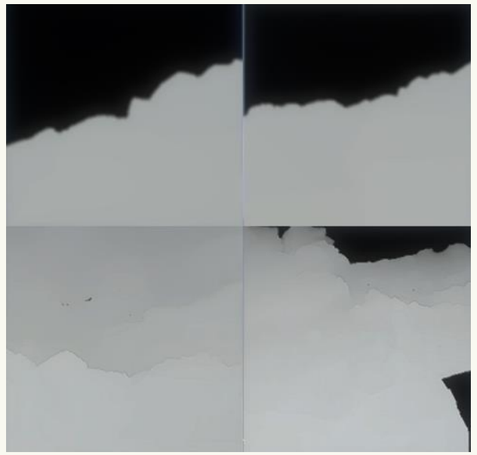
Static Shadows

When a static shadow is updated:

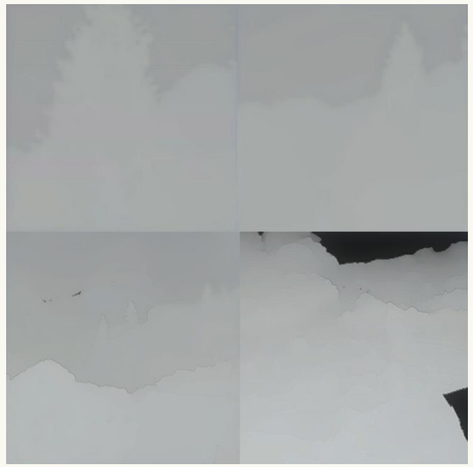


A static mesh is drawn onto the static shadow map.

Every frame:



A static shadow map is used to clear the shadow map.



A dynamic mesh is drawn onto the shadow map.

Screen Space Shadows

- Post effect shadows generated using ray marching
- Used for grass models
  - Effective for fields with a lot of grass
  - Pixel shader is used for cutout models
  - Static shadows are excluded since grass is a dynamic object
  - GPU performance isn’t dependent on the number of models

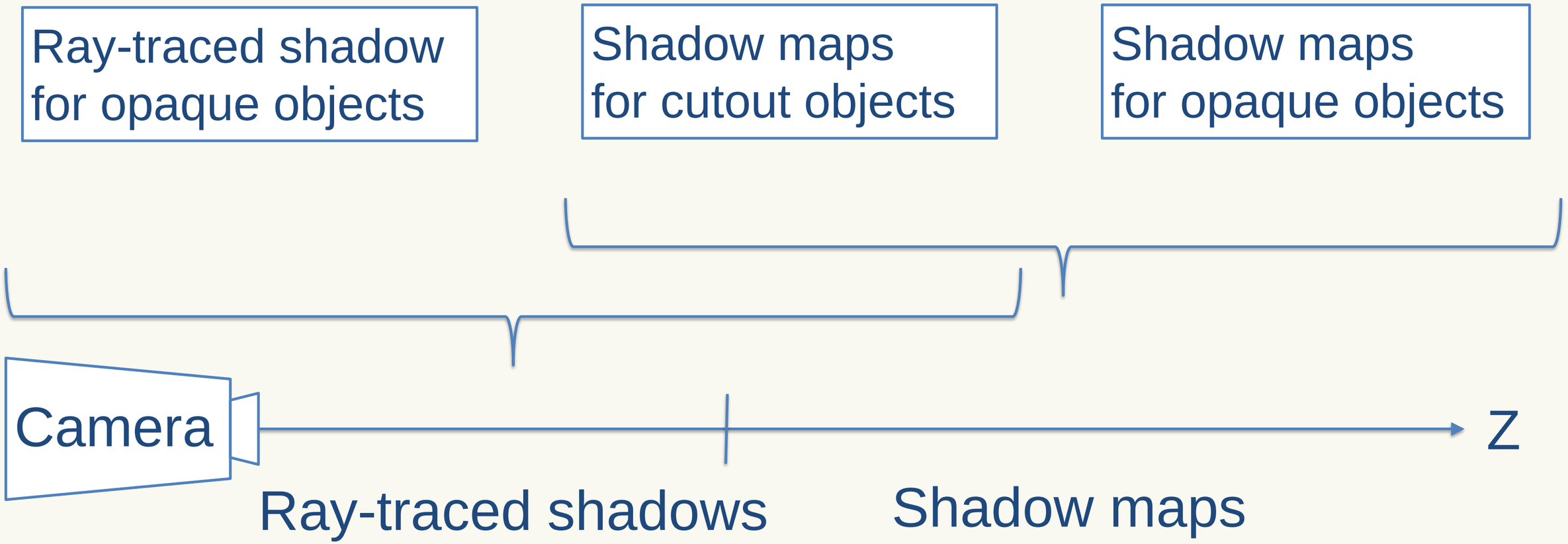
Screen Space Shadows - on -

Screen Space Shadows -off-

Hybrid Ray-Traced Shadows

- Hybrid ray-traced shadows with shadow maps
- BVHs are created for models in the near view
- Inline ray-tracing
- Exclude cutout models
  - Skip closest hit shaders

Hybrid Ray-Traced Shadows



Hybrid Ray-Traced Shadows

- Pre-create a mask in areas where ray-traced shadows are used
- Cast rays in Mask< one pixel

Hybrid Ray-Traced Shadows - on -

Hybrid Ray-Traced Shadows - off -

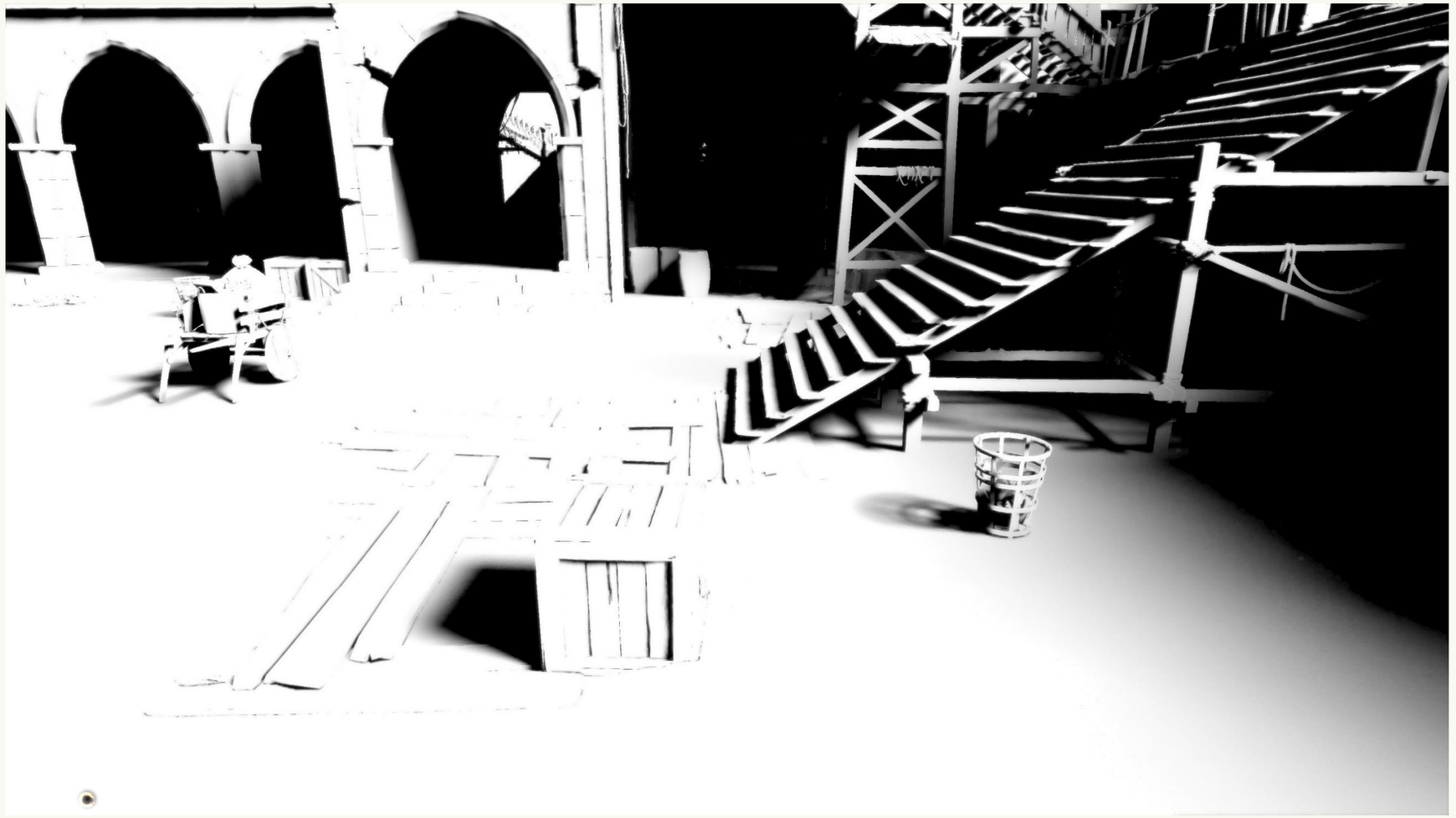
Hybrid Ray-Traced Shadows

- One ray per frame
- Denoiser is used to smooth out the image

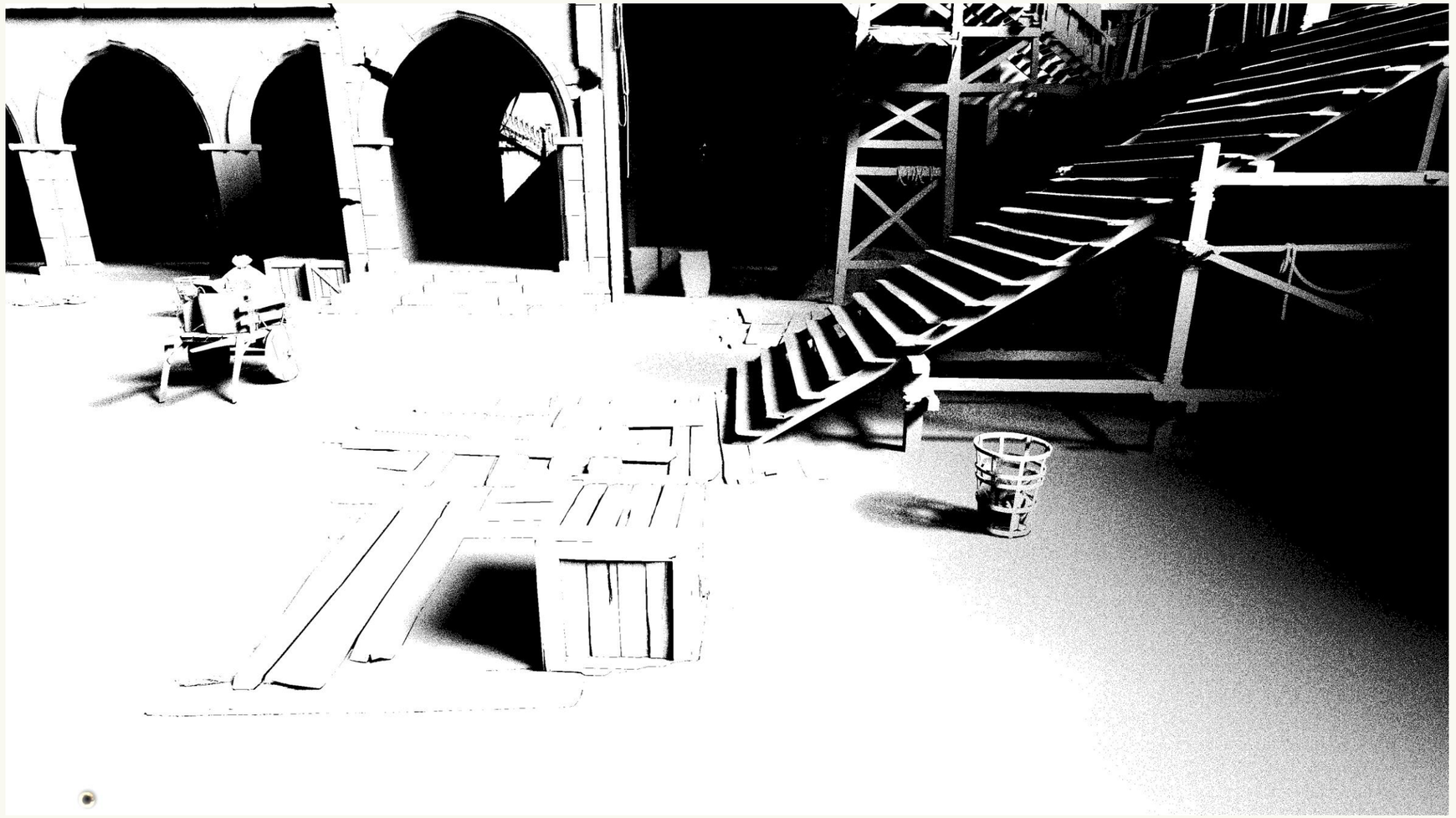
Hybrid Ray-Traced Shadows - Denoiser on -

Hybrid Ray-Traced Shadows - Denoiser off -

Hybrid Ray-Traced Shadows - Denoiser on -



Hybrid Ray-Traced Shadows - Denoiser off -



Ray-traced Ambient Occlusion

- Similar processing flow to ray-traced shadows

- Applied to the near view only
- Pre-create a mask
- One ray per frame
- Denoiser

Ray-traced AO - on -

Ray-traced AO - off -

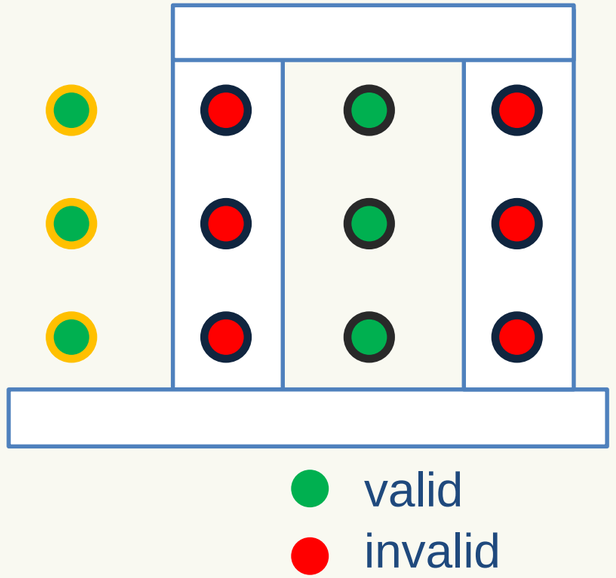
Lighting & Post Effects

Precomputed Radiance Transfer (PRT)

- Diffuse ambient light using uniform grid probes of Spherical Harmonics(9 coef)

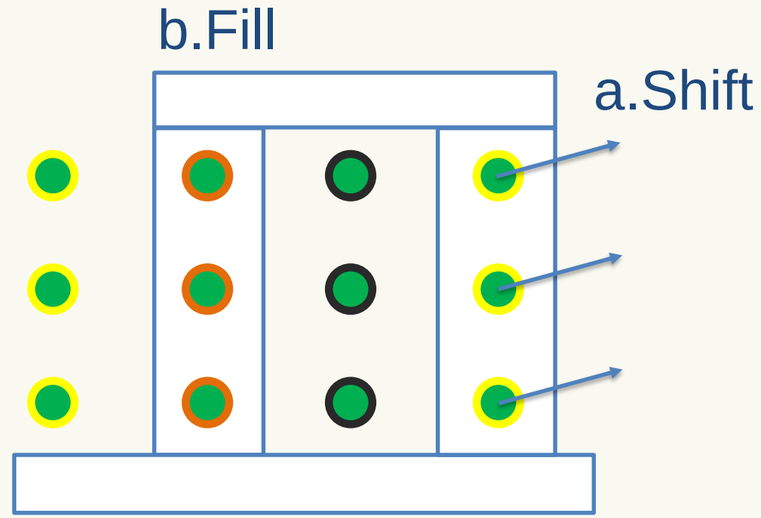
PRT - Baking -

- Use ray-tracing for baking
- Create cubemaps with light bounces
  - Convert cubemaps to SHs
    - Sky visibility SH
    - Punctual light SH
- Probes located inside polygons are deemed invalid
  - Check for differences in depth images rendered with/without backface culling



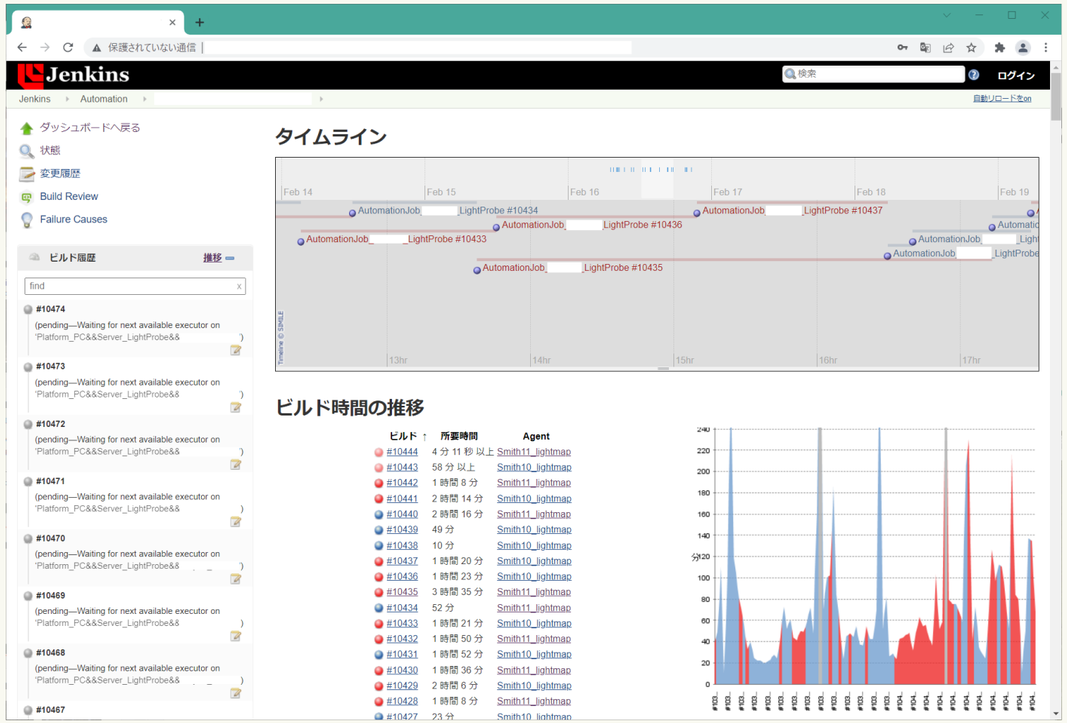
PRT - Baking -

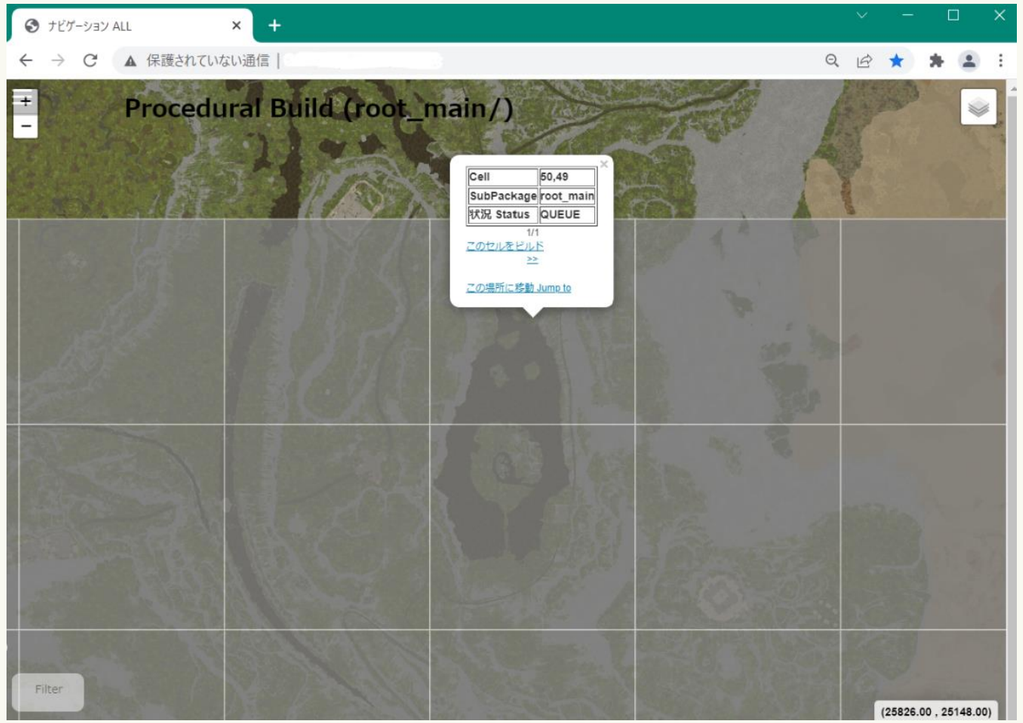
- Available bake options:
  a. Shift bake positions
  b. Fill in invalid probes
     with neighboring valid probes
  c. Blur adjacent valid probes



PRT - Baking -

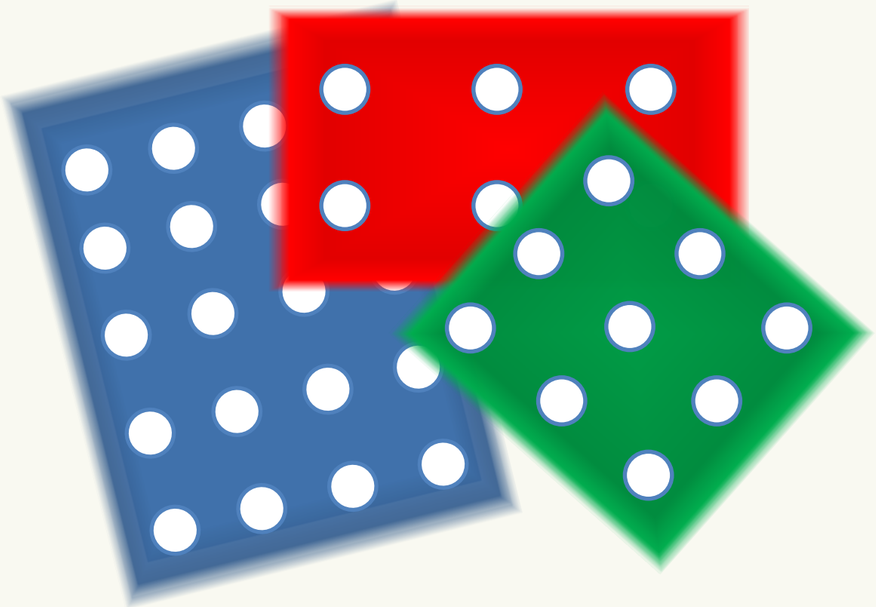
- Baked on the server on a daily basis





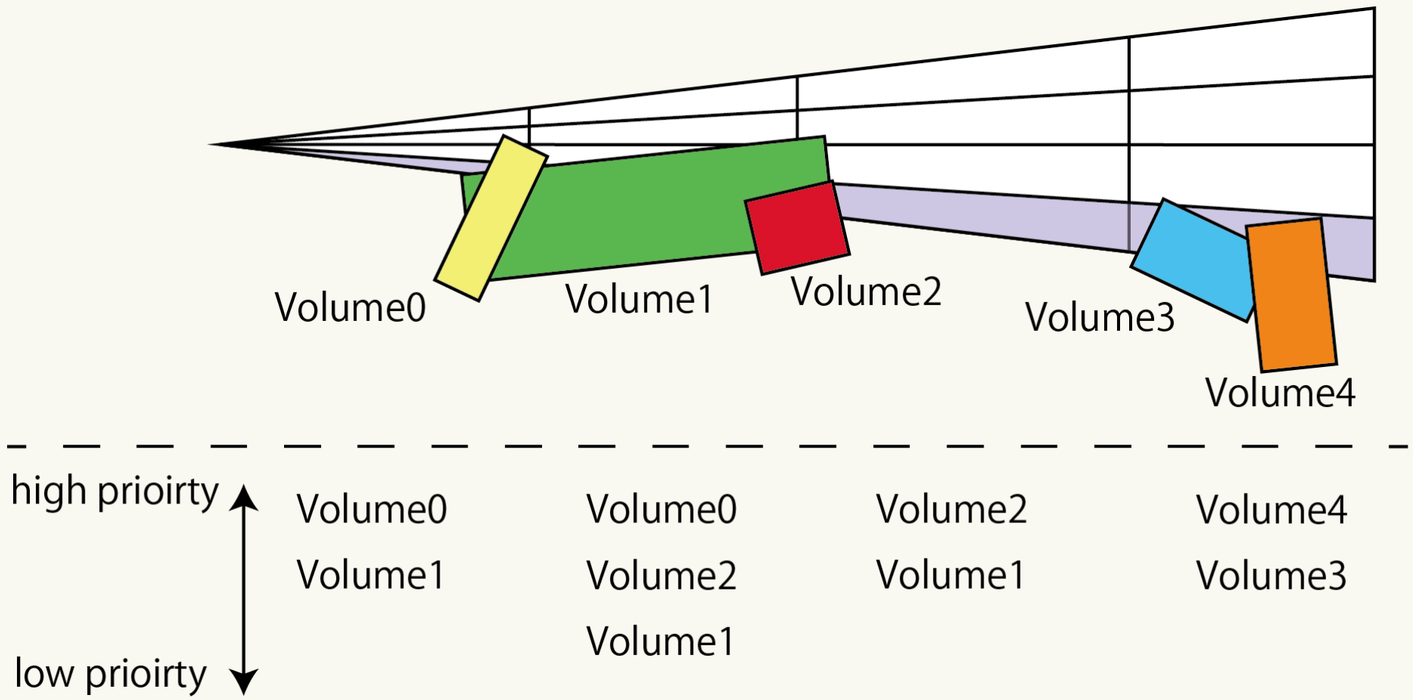
PRT - Placement -

- Place volumes as entities.
- Sort by a priority value.



PRT - Culling -

- Frustum culling
- Cluster culling



- Sort in each cluster

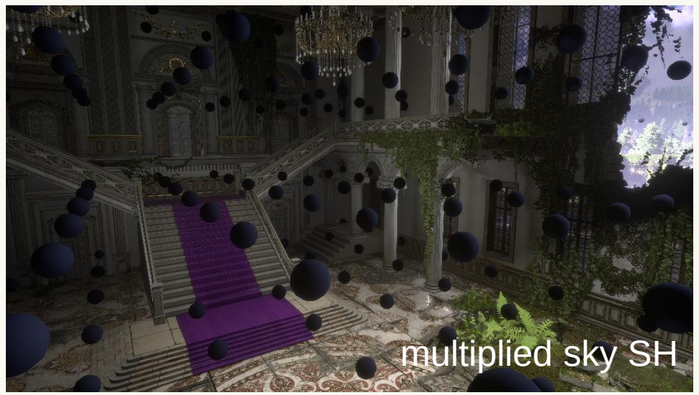
PRT - Updating SH -

+

*

=

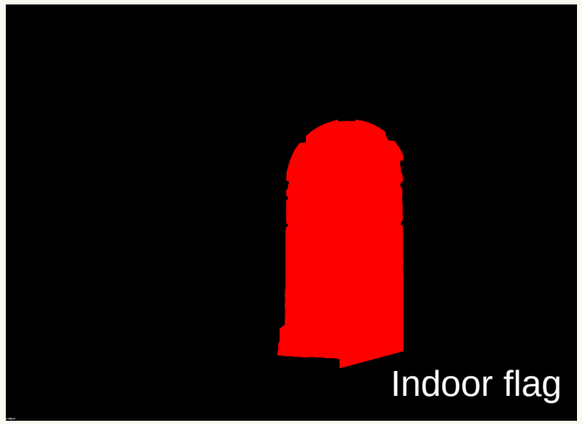
+

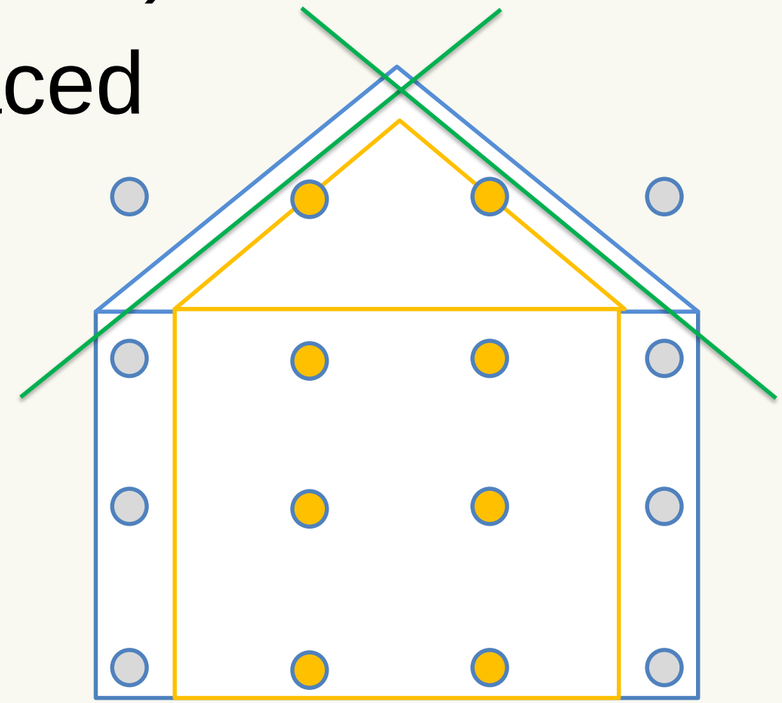


=

PRT - Sampling -

- Calculate weights based on valid probes
- Masking with indoor flags (GBuffer RT3.a)
- Four extra obstacle planes can be placed





Specular Correction

- Scales IBL specular based on irradiance volume values
  - Specular IBL probes are not as dense as Irradiance volume probes

Similar approach: [JT16]

Specular Correction

- Store SH0 in the last mipmap (1x1) of each IBL in advance

$$Corrected\ IBL\ pixels\ = IBL\ pixels * \frac{SH0\ in\ PRT}{SH0\ in\ IBL}$$

Specular Correction - on -

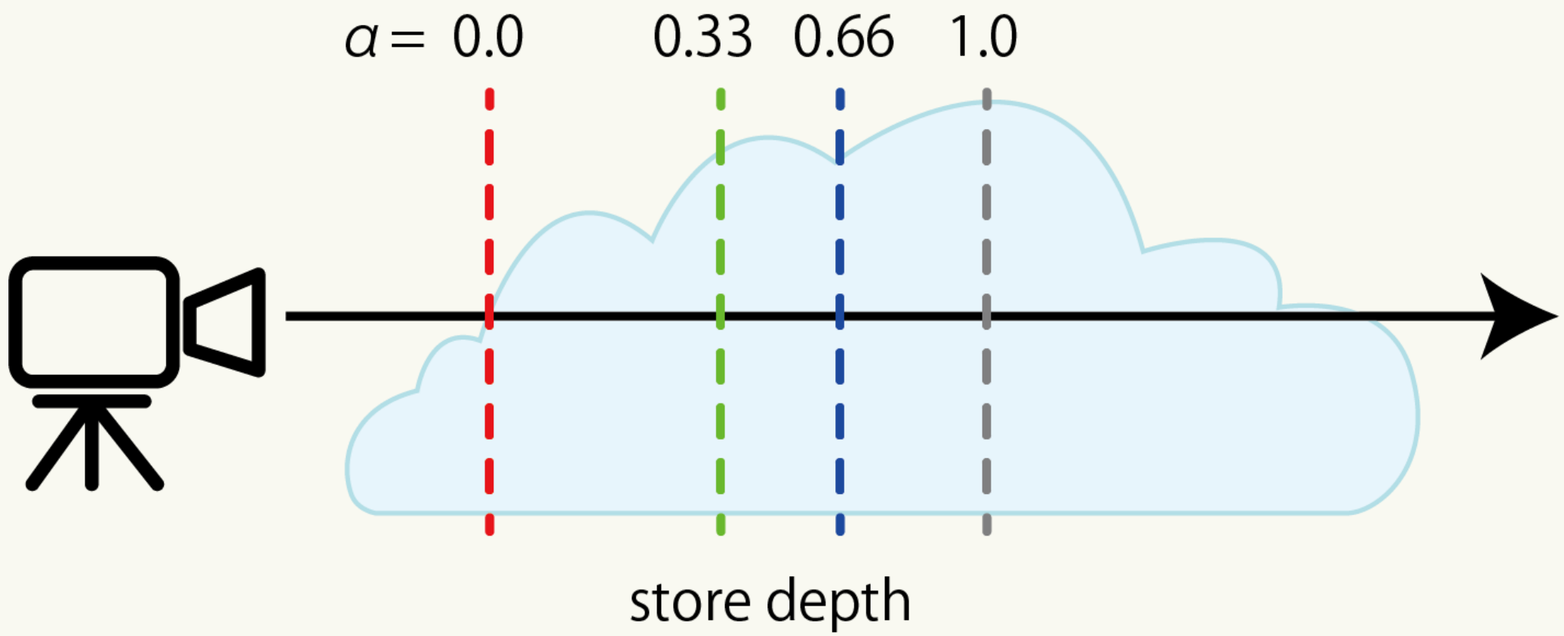
Specular Correction - off -

Volumetric Cloud

- Ray marching-based cloud rendering [FABIAN19]
  - Ray march weather maps and noise textures for distant clouds
  - Create a frustum voxel grid for clouds in the near view
  - Temporal sampling

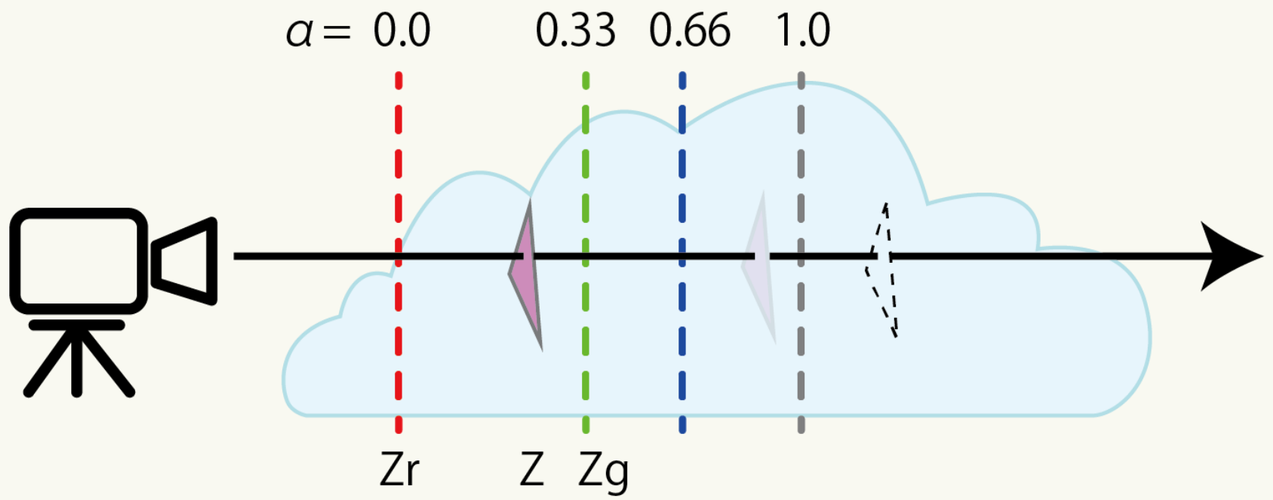
Clouds and Transparent Objects

- Export depth values when the density reaches a certain level.



Clouds and Transparent Objects

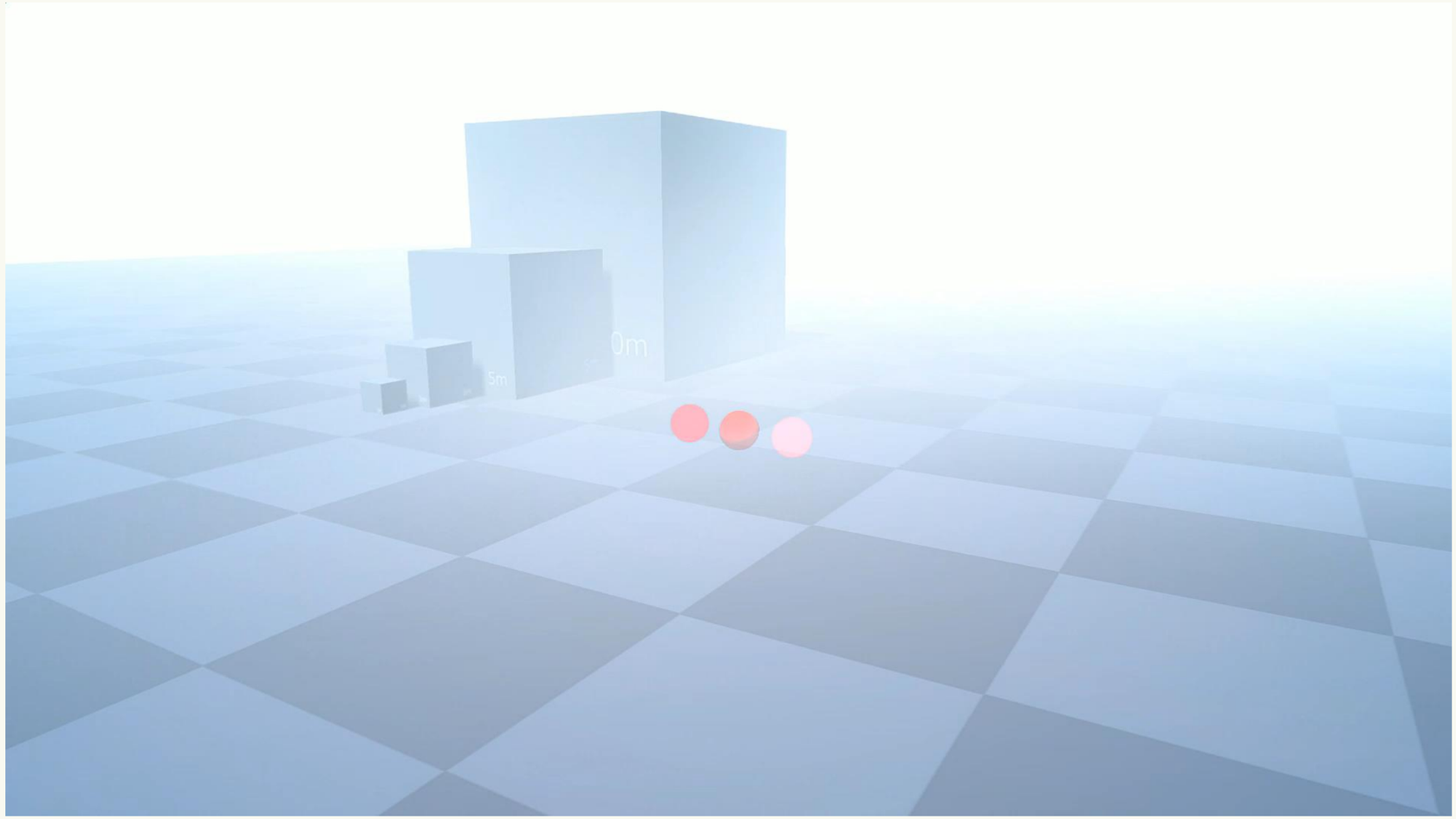
- Locate two nearest points on the depth and use their corresponding alphas for interpolation



$$\text{Polygon } \alpha = 1 - \left(\frac{Z - Z_r}{Z_g - Z_r} * 0.33 + \alpha_r\right)$$

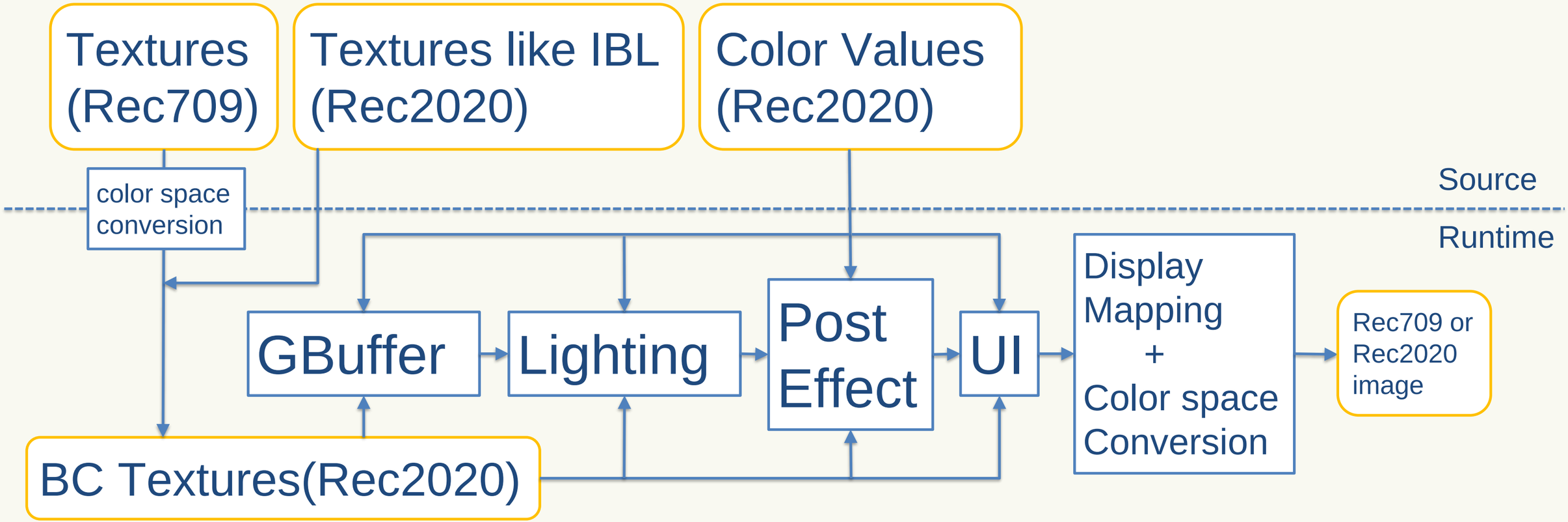
$$\alpha_r = 0.0$$

Clouds and Transparent Objects



Wide Gamut

- Changed the rendering color space to Rec2020(D65)



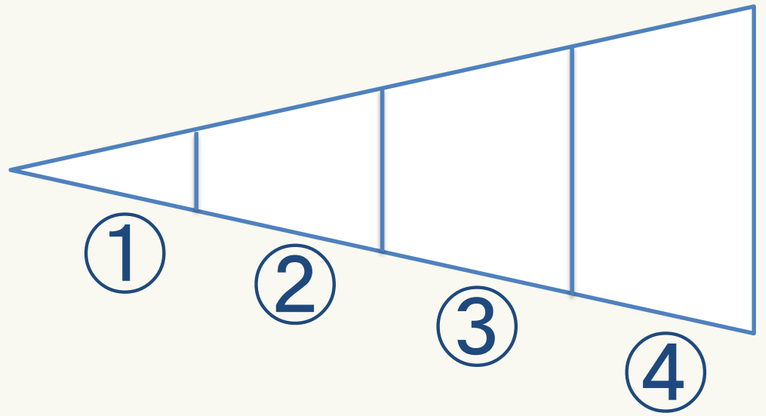
Optimization

GBuffer Sort Keys

- What we want to achieve:
  - Reduced overdraw
  - Ability to draw the same-state objects at the same time

GBuffer Sort Keys

- Categorize objects into four z-blocks
- Draw objects in z-blocks from near to far



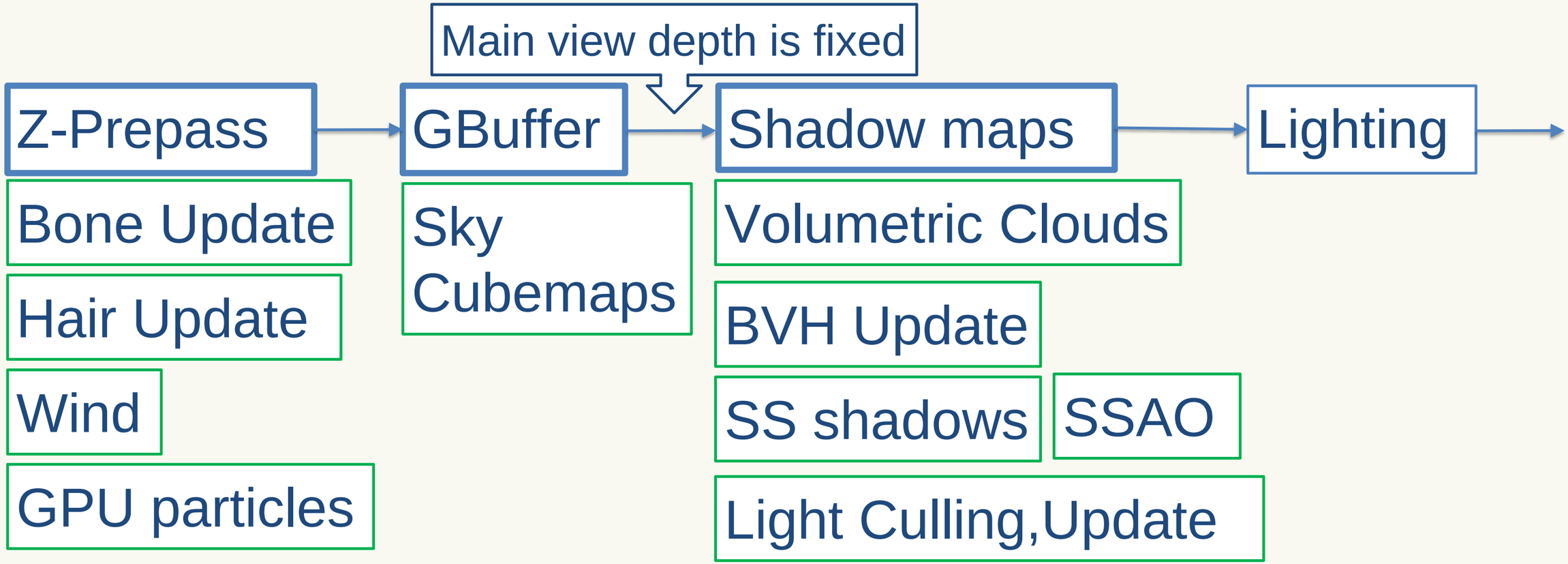
- Instanced drawing for objects with the same material and mesh
- Draw objects rendered in Z-prepass last

GBuffer Sort Keys

```
// 64 bits: |-- priority 4 bits --|-- z block 2 bits --|-- stencils 8 bits --|
// -- material id 16 bits --|-- material flags 4 bits --|-- mesh id 16 bits --|
// -- depth 14 bits --|
key  = (priority      & 0xf)    << 60;
key |= (zRangeBlock   & 0x3)    << 58;
key |= (stencilBits   & 0xff)   << 50;
key |= (materialID    & 0xffff) << 34;
key |= (materialFlags & 0xf)    << 30;
key |= (meshID        & 0xffff) << 14;
key |= (depth14bit    & 0x3fff);
```

Async Compute

- Executed in parallel during the geometry stages



Variable Rate Shading

- Used in the VFX and lighting stages

Lighting with VRS

- Lighting is done with vertex and pixel shaders due to the use of stencil masks
- VRS Tier2 (VRS tile)

FidelityFX
Variable Shading

Lighting

VFX with VRS

- VRS Tier1 (fixed sample rate) is used for VFX
- Works well with VFX where the pixel shader tends to be a bottleneck

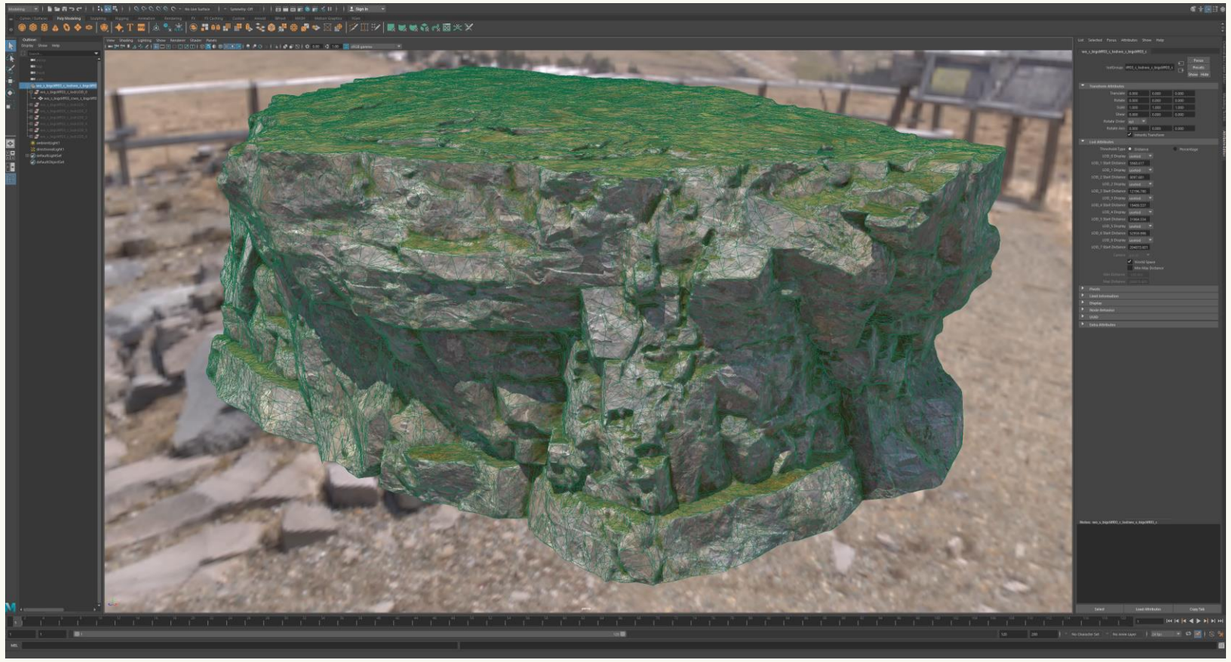
VFX with VRS

- Soft particles cause jagged edges
- Used 2x2 VRS instead of 4x4 VRS.

Automatic LOD Generation

LOD Workflow

- What artists want:
  - Auto-generated LOD models.

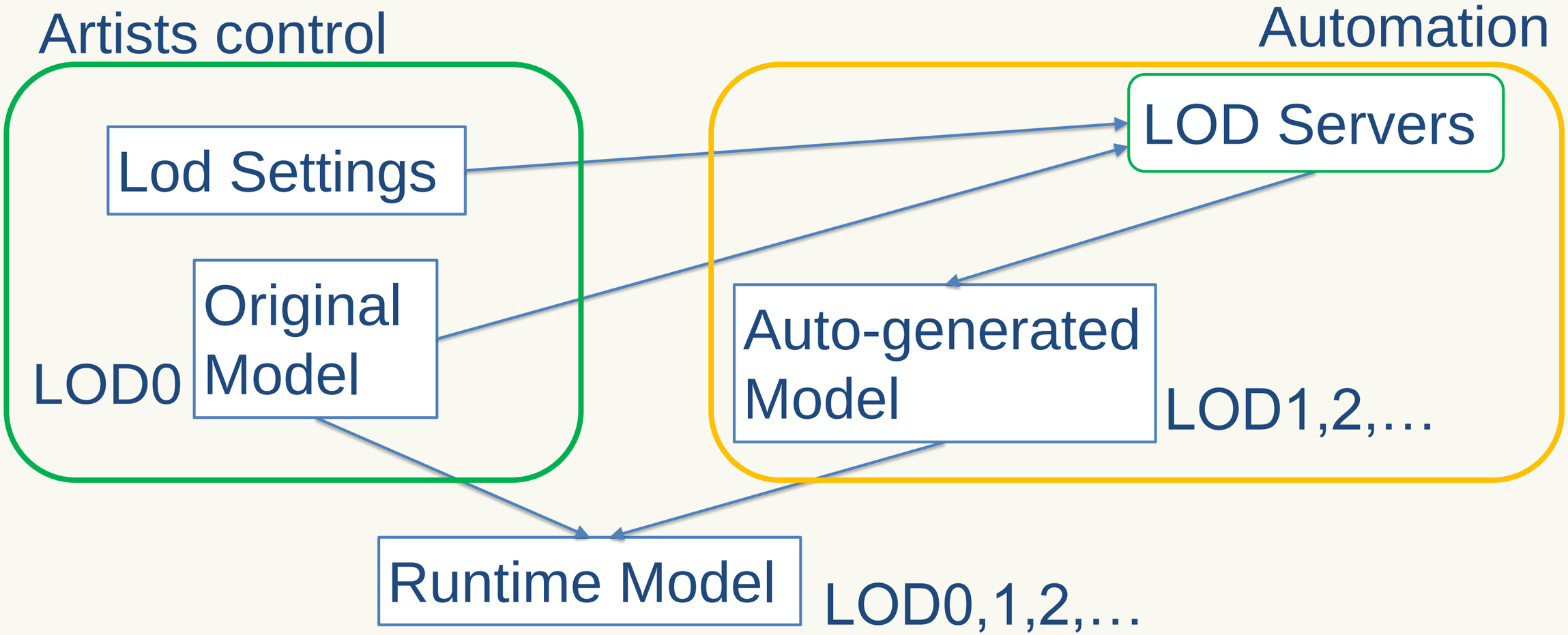


Automatic LOD Generation

- Focus only on mesh reduction
- No material combining
- Run Simplygon reduction on the server

[SYMPLYGON]

LOD Workflow



LOD Setting

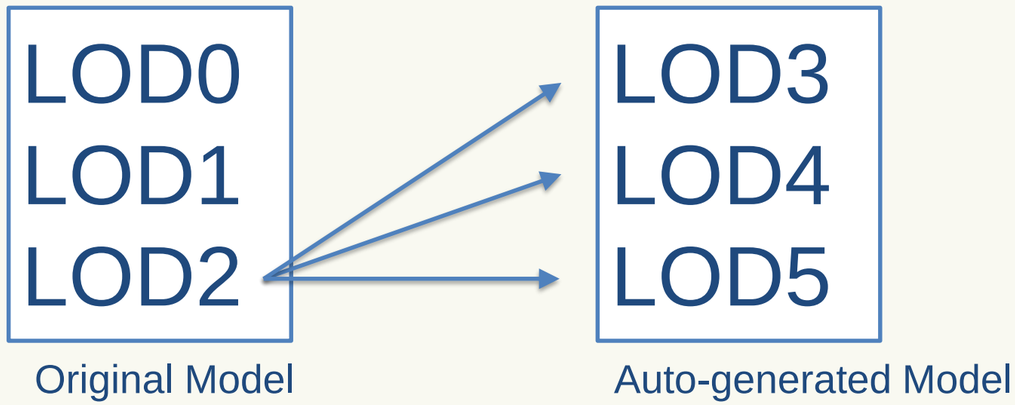
- Reduction settings can be specified in detail for each file
- Specify screen height threshold for individual LOD

```
[LodConfig]
DesiredLevels=3
PreserveSymmetry=0
RepairInvalidNormals=1
[Lod1]
Threshold=0.6
[Lod2]
Threshold=0.36
[Cull]
Threshold=0.02
```

Sample LOD settings

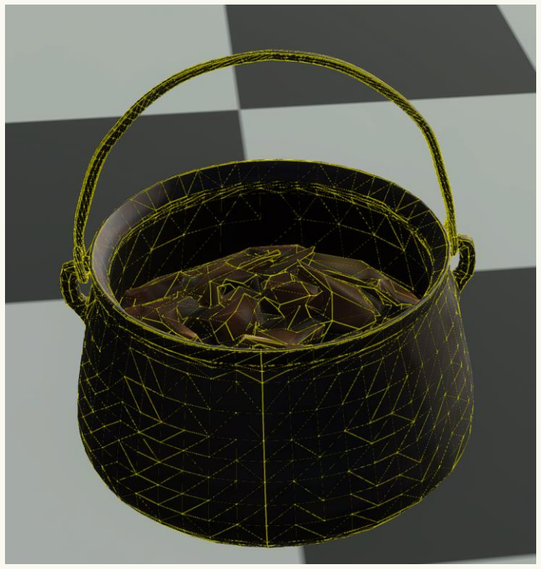
Handcrafted vs. Auto-Generated LODs

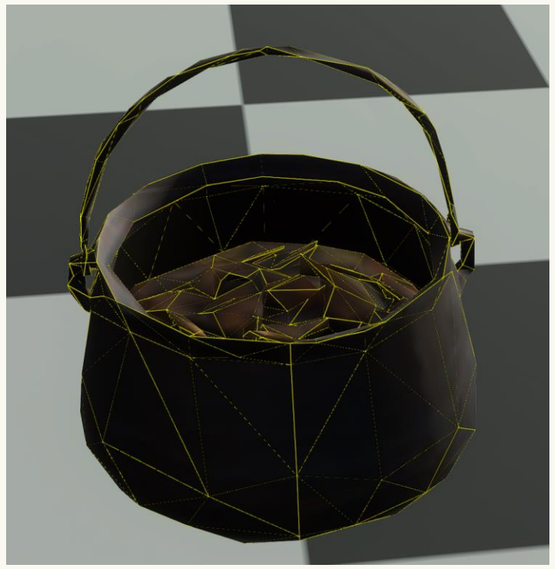
- Allow artists to handcraft LODs halfway through and auto-generate the rest
- Adjust the distance setting for auto-generated LODs while leaving it intact for handcrafted LODs

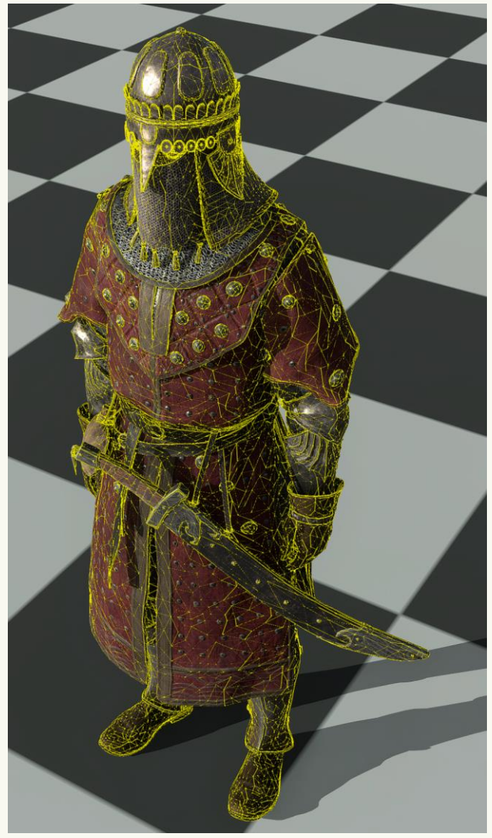


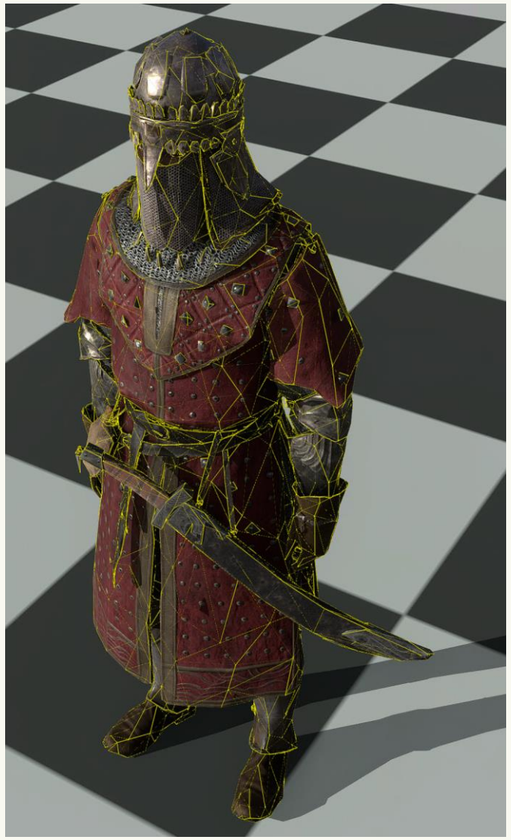
What worked well

- Props e.g. pots, rocks
- Characters (except for hair)



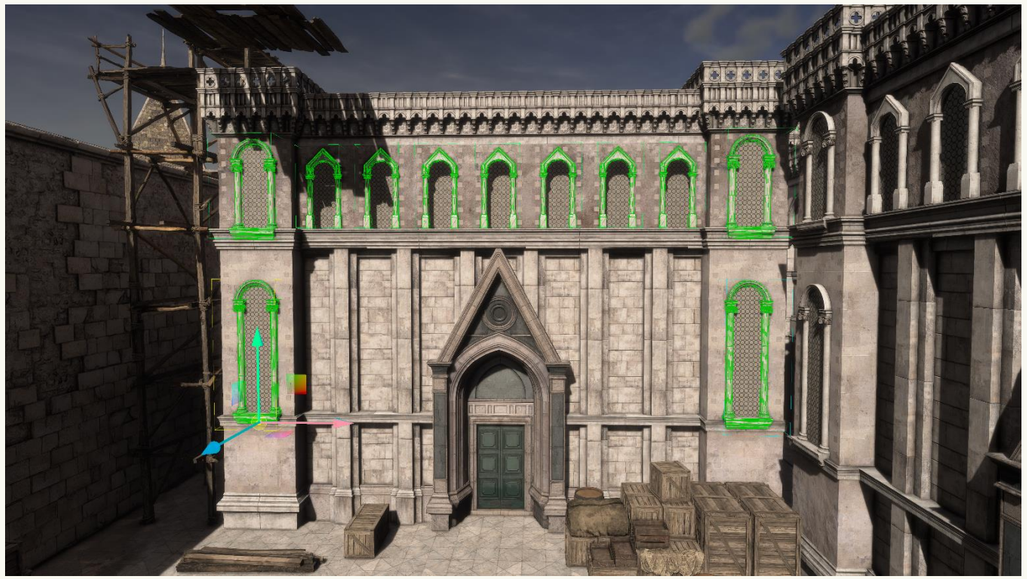


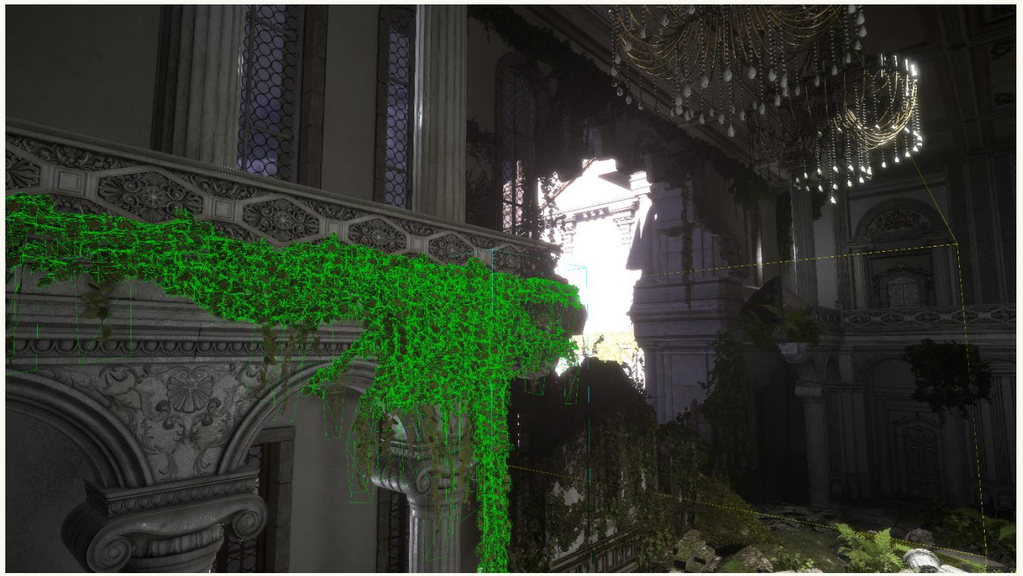




What didn’t work well

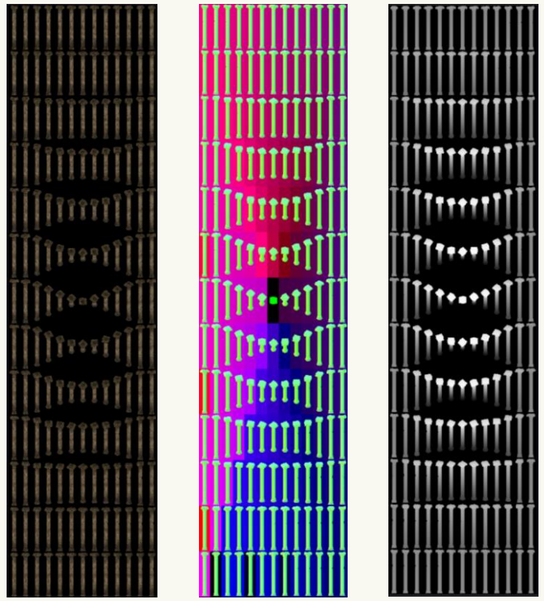
- Buildings consisting of many elements e.g. windowsill
- Polygon cracks caused by unadjusted LODs
- High-poly models with multiple meshes

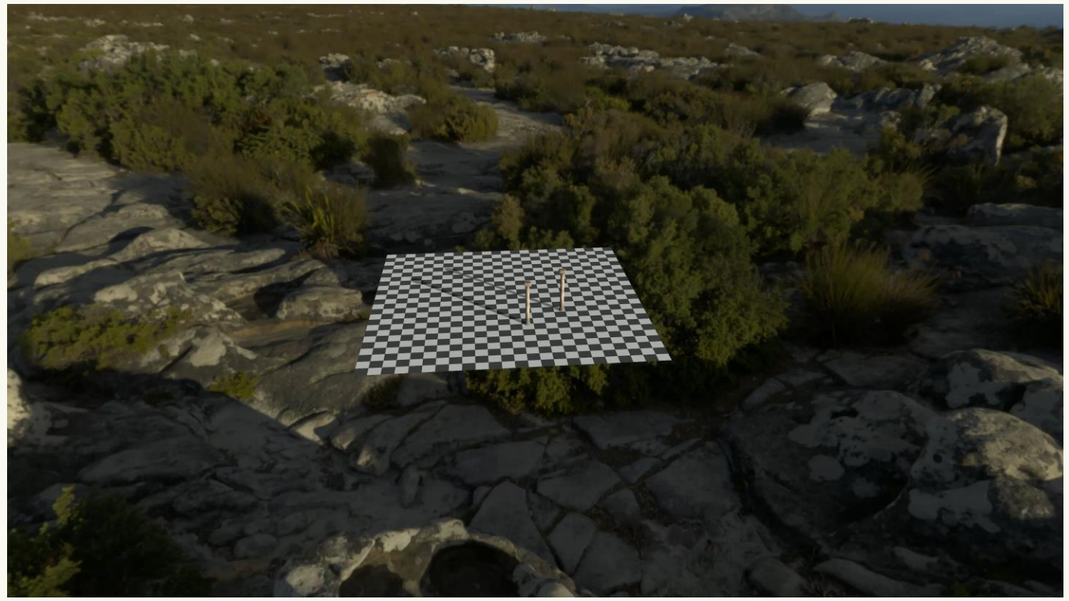




Auto-Generated Imposter LODs

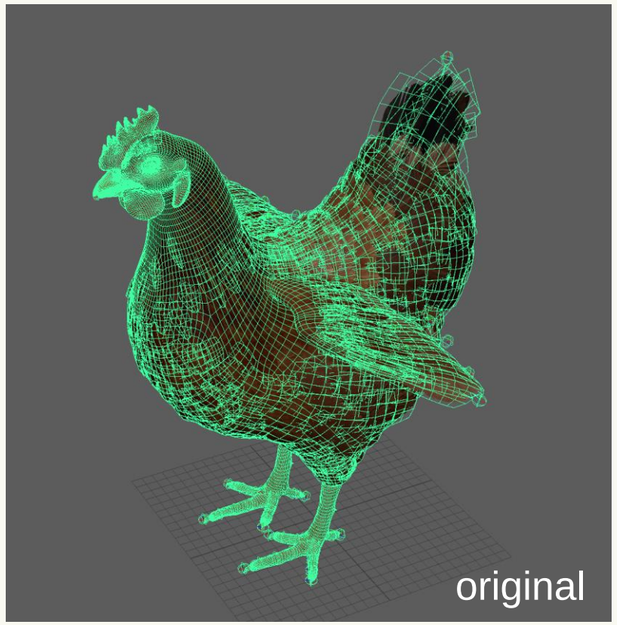
- Imposter textures are baked using the game
- No longer used for tree models due to the imposter shader running slow

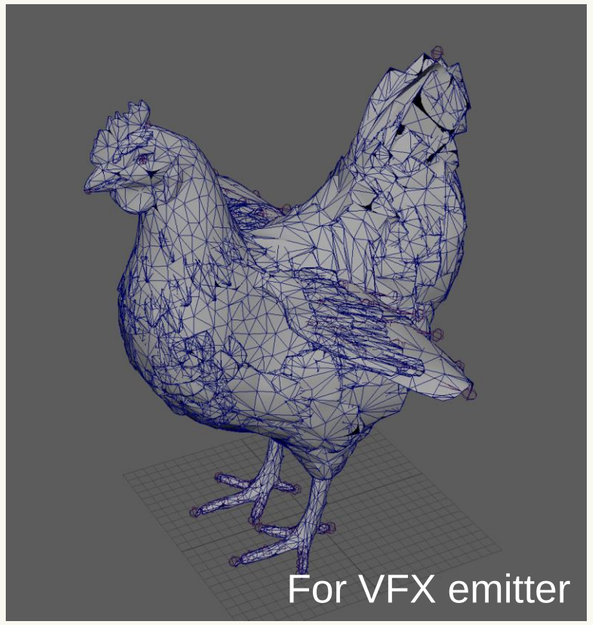




Auto-Generated VFX Emitter Mesh

- Auto-generated the VFX emitter mesh using “Remesh” to ensure consistent polygon density





Summary

- Render Pipeline
  - Model Rendering
  - Shadows
  - Lighting & Post Effects
  - Optimization
- Automatic LOD Generation

Special Thanks

- Akihiro Nakamoto
- Hiroshi Iwasaki
- Kento Masuno
- Ryota Sakamoto
- Shaoti Lee
- Takeshi Aramaki
- Toni Georgiev
- Tao Yang
- Yoshiro Aoki
- Yuki Kawana
- Yusuke Hasuo

References

[AMD] AMD, FidelityFX
https://gpuopen.com/effects/

[NVIDIA15] Nvidia, Depth Precision Visualized
https://developer.nvidia.com/content/depth-precision-visualized

[STEPHEN19] Stephen McAuley, A Journey Through Implementing Multiscattering BRDFs and Area Lights, Siggraph2019
https://advances.realtimerendering.com/s2019/index.htm

[FABIAN19] Fabian Bauer, Creating the Atmospheric World of *Red Dead Redemption 2*: A Complete and Integrated Solution, Siggraph2019
https://advances.realtimerendering.com/s2019/index.htm

[JT16] JT Hooker, Volumetric Global Illumination At Treyarch, Siggraph2016
https://www.activision.com/cdn/research/Volumetric_Global_Illumination_at_Treyarch.pdf

[SIMPLYGON] Simplygon
https://www.simplygon.com/

Thank you for listening!