Open World Rendering Techniques in Hogwarts Legacy

GDC 2024

Rob Hall

Avalanche Software, Warner Bros Games

AVALANCHE

Background

- Game developed over 5+ years
- Migrated from in house engine to Unreal
- Shipped with Unreal 4.27 Chaos branch
- Over 22 platforms and variants
- 30, 40, 60, and 120 fps modes
- Ray tracing on PC, PS5, and Xbox Series X
- Avalanche Software's first open world RPG

PC

- Low Nvidia & AMD
- Medium Nvidia & AMD
- High Nvidia & AMD
- Ultra Nvidia & AMD
- Ultra RT Nvidia & AMD

Sony

- PS5 Fidelity
- PS5 Fidelity RT
- PS5 Performance
- PS4 Pro
- PS4

Microsoft

- XSX Fidelity
- XSX Fidelity RT
- XSX Performance
- XSS
- Xbox One X
- Xbox One

Nintendo

- Switch Handheld

High Level Overview

- Open world complexity
- Leveraging bindless resources to reduce complexity
- In depth explanations
  - Bindless shadows
  - Bindless cloth regions
- Optimizing GPU performance using automation
- Visual techniques developed for Hogwarts Legacy

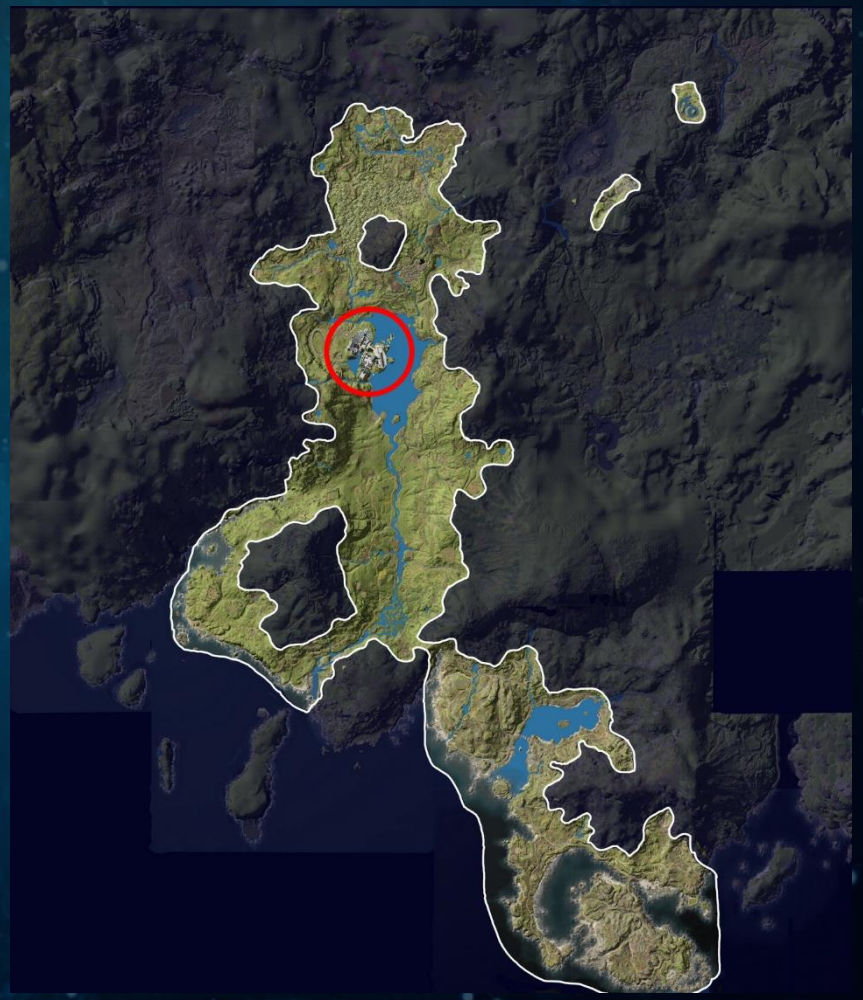
Open World Complexity

Open World Complexity

- Open world
  - ~9km²
- Hogwarts castle
- Hogsmeade
- All doors can be opened

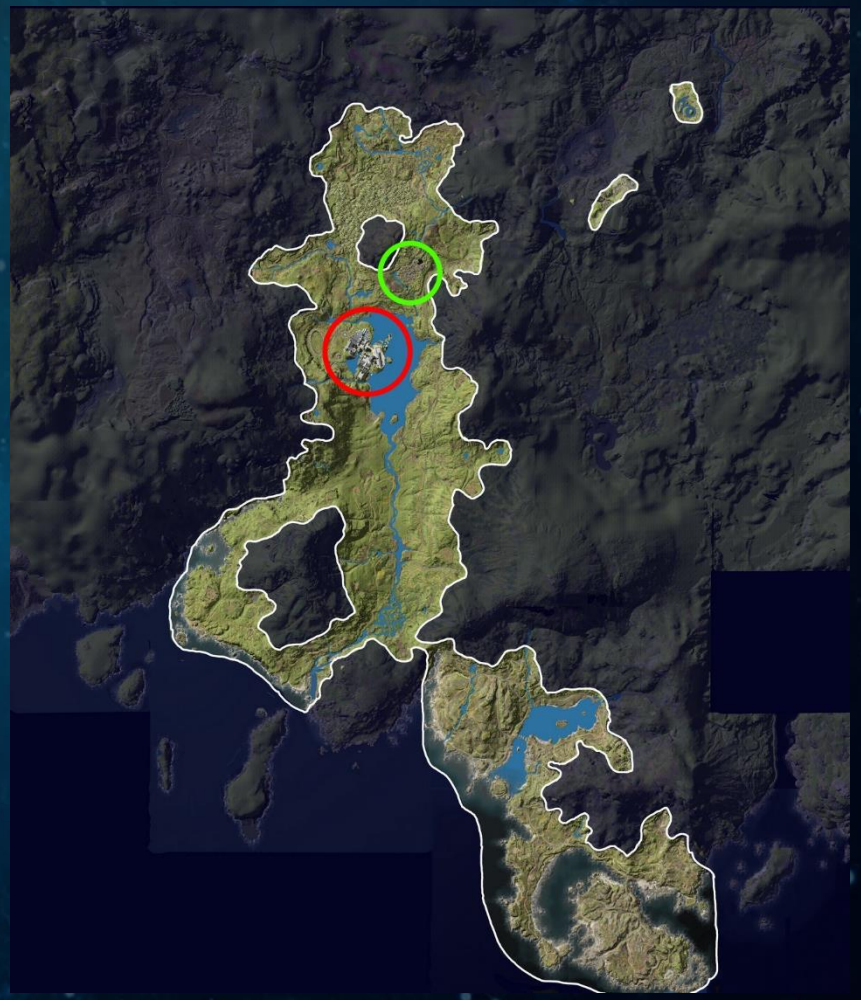
Open World Complexity

- Open world
  - ~9km²
- Hogwarts castle
- Hogsmeade
- All doors can be opened



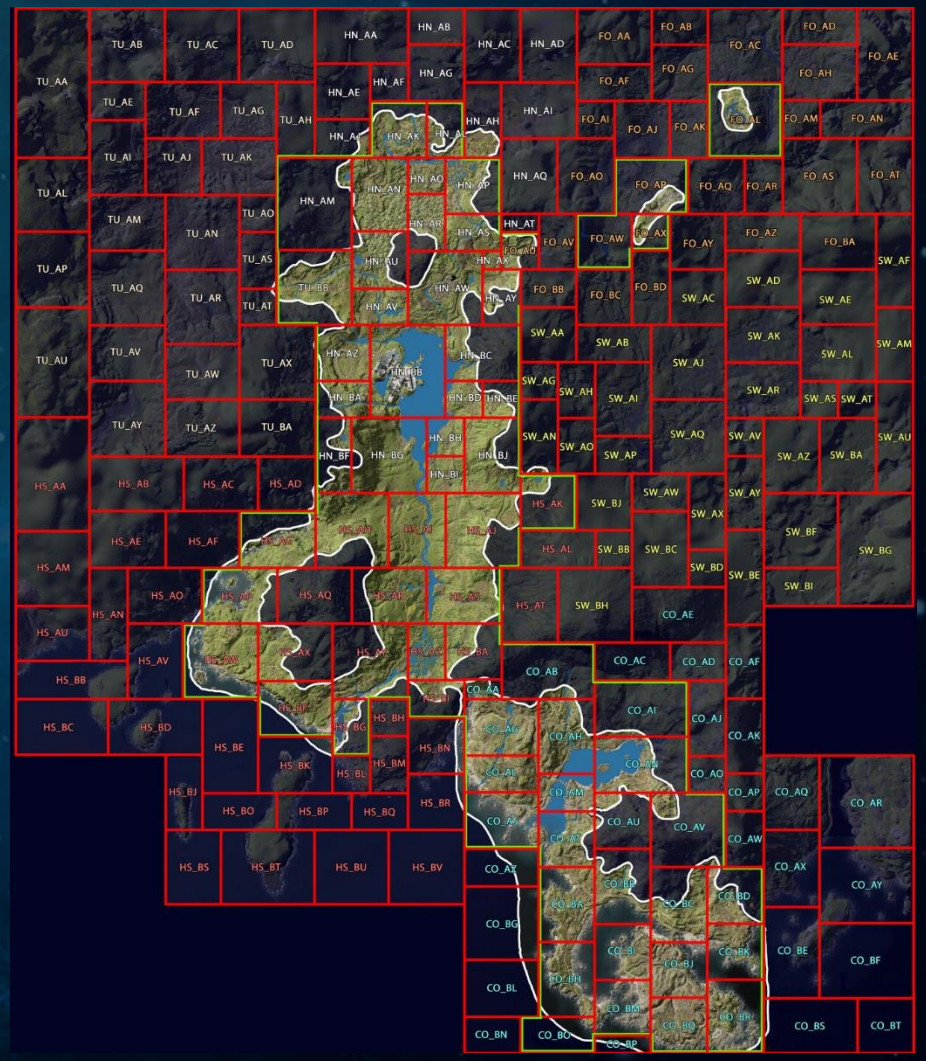
Open World Complexity

- Open world
  - ~9km²
- Hogwarts castle
- Hogsmeade
- All doors can be opened



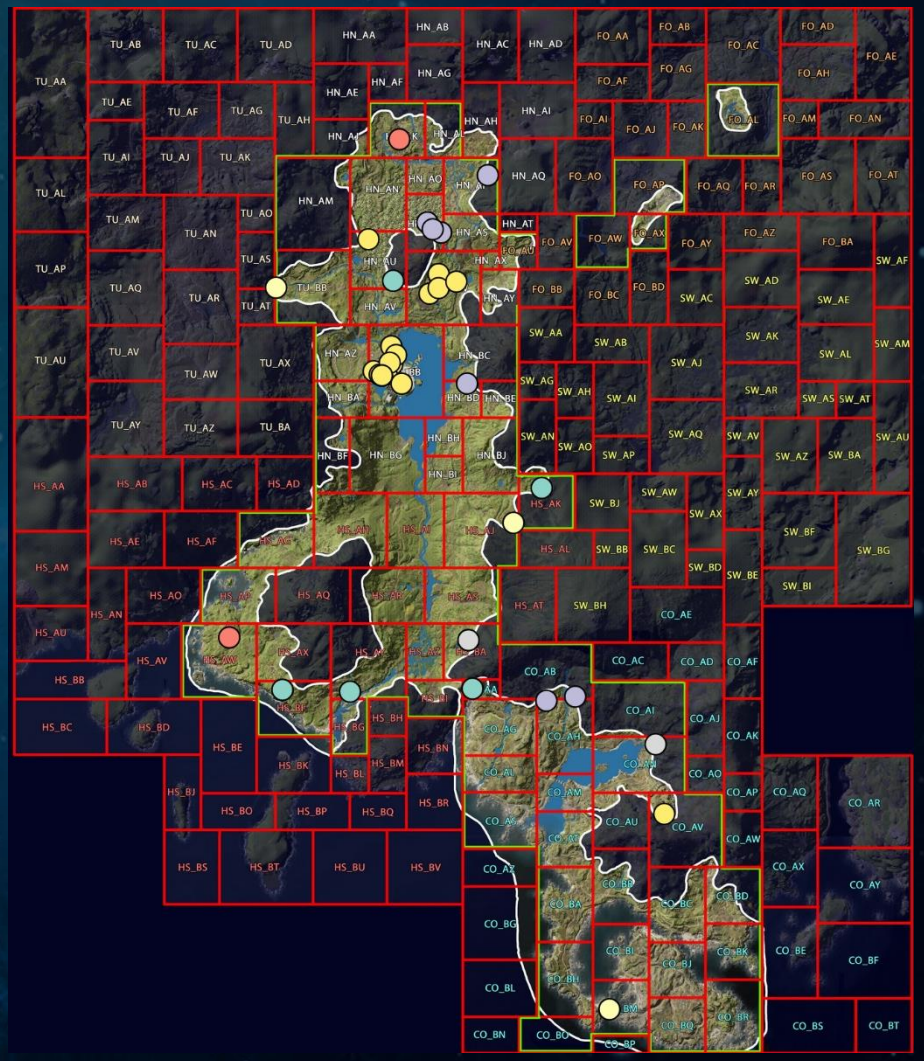
Open World Details

- 73 playable tiles
- 107 non-playable tiles
- 58 Dungeons
- 40 ruins
- 47 bandit camps
- 11 hamlets
- 113 Vaults



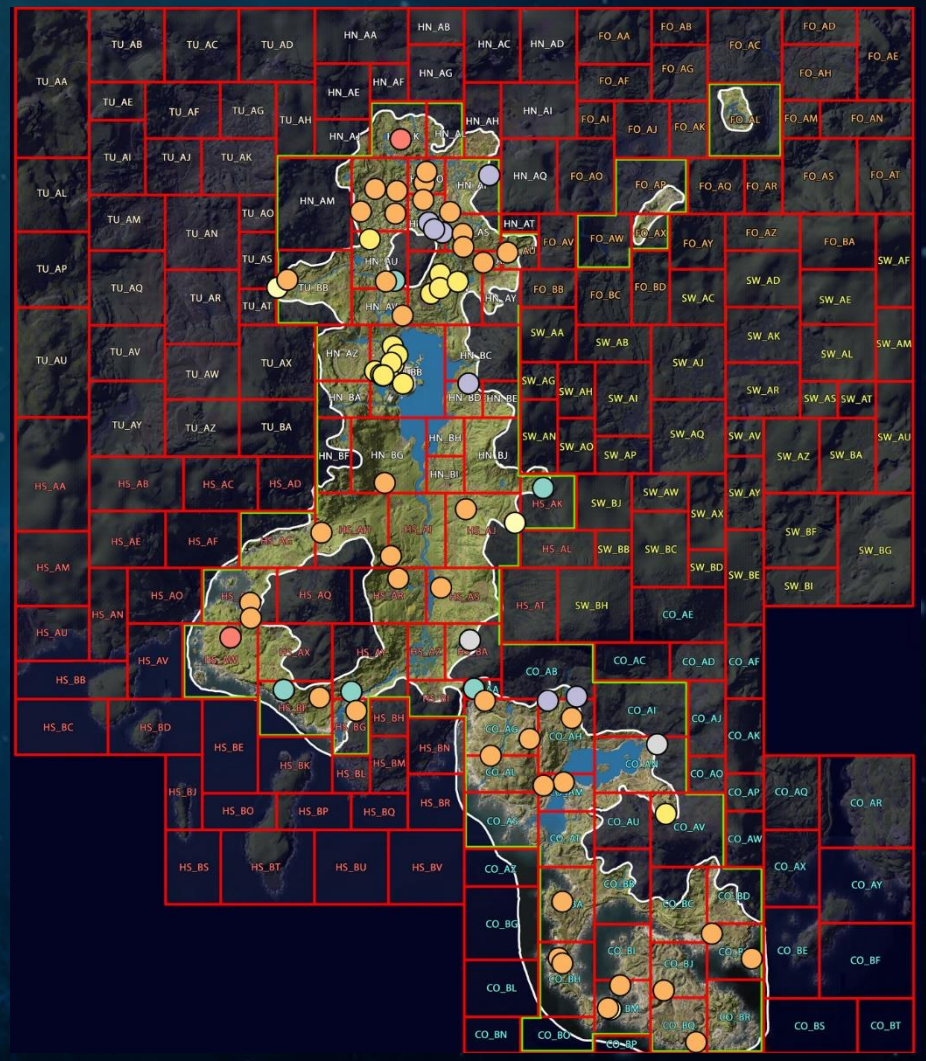
Open World Details

- 73 playable tiles
- 107 non-playable tiles
- 58 Dungeons
- 40 ruins
- 47 bandit camps
- 11 hamlets
- 113 Vaults



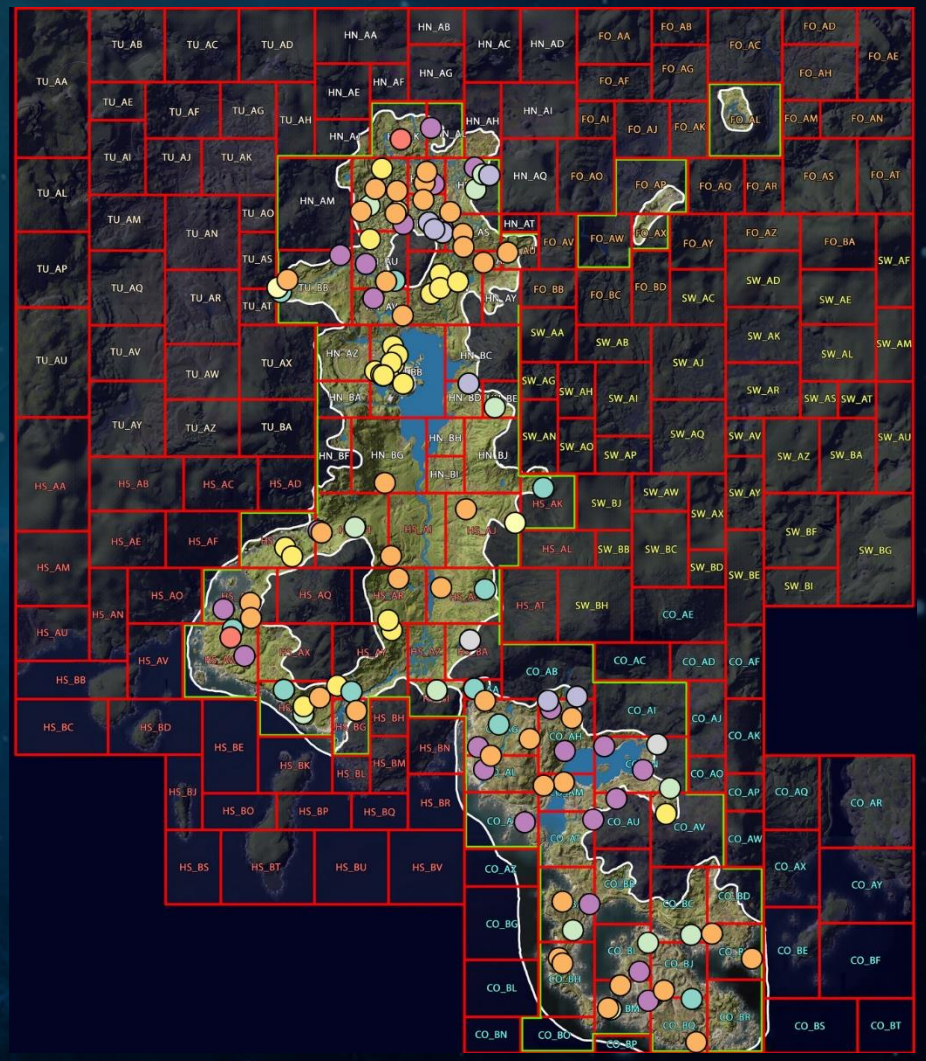
Open World Details

- 73 playable tiles
- 107 non-playable tiles
- 58 Dungeons
- 40 ruins
- 47 bandit camps
- 11 hamlets
- 113 Vaults



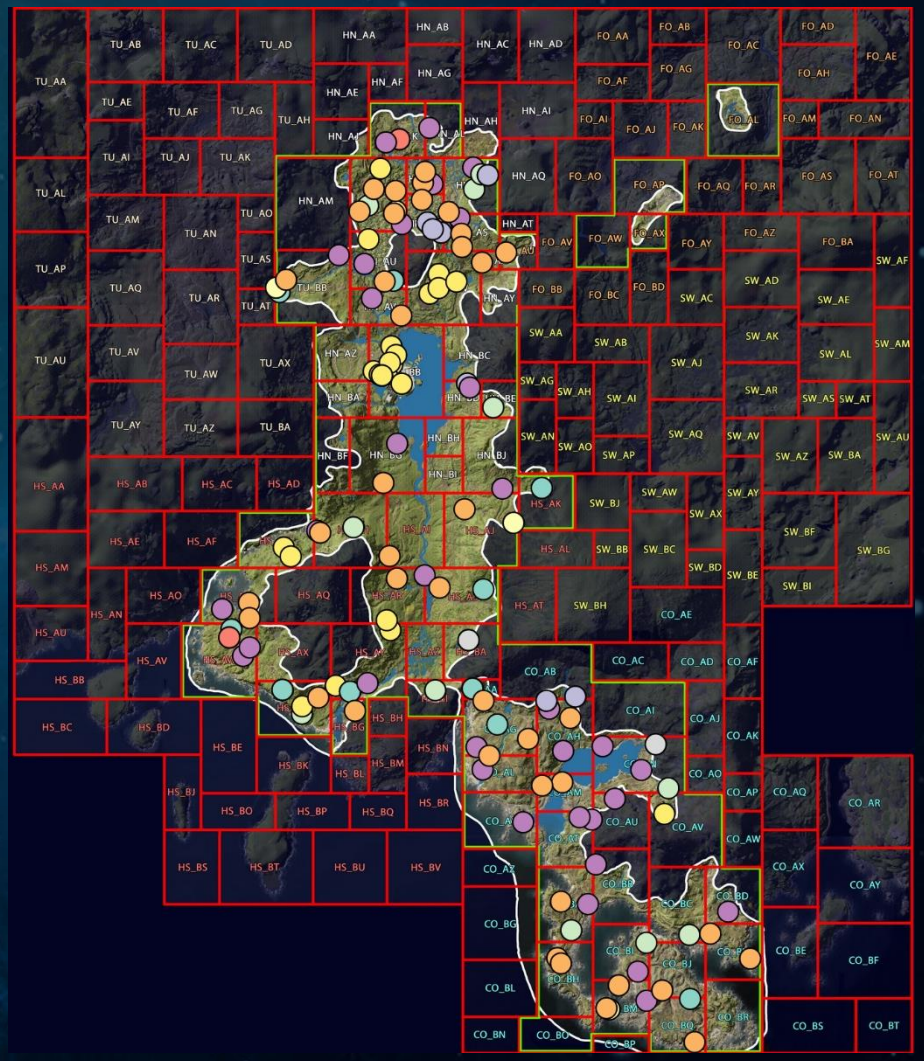
Open World Details

- 73 playable tiles
- 107 non-playable tiles
- 58 Dungeons
- 40 ruins
- 47 bandit camps
- 11 hamlets
- 113 Vaults



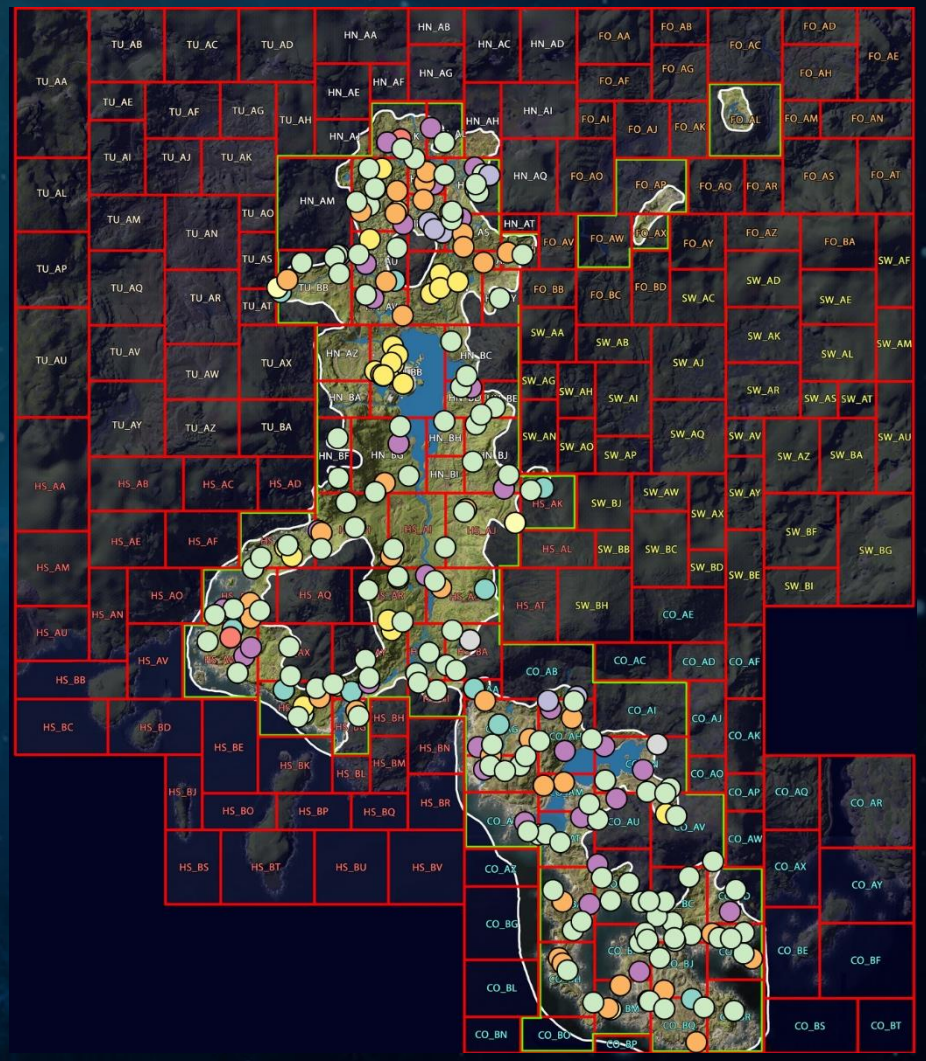
Open World Details

- 73 playable tiles
- 107 non-playable tiles
- 58 Dungeons
- 40 ruins
- 47 bandit camps
- 11 hamlets
- 113 Vaults



Open World Details

- 73 playable tiles
- 107 non-playable tiles
- 58 Dungeons
- 40 ruins
- 47 bandit camps
- 11 hamlets
- 113 Vaults

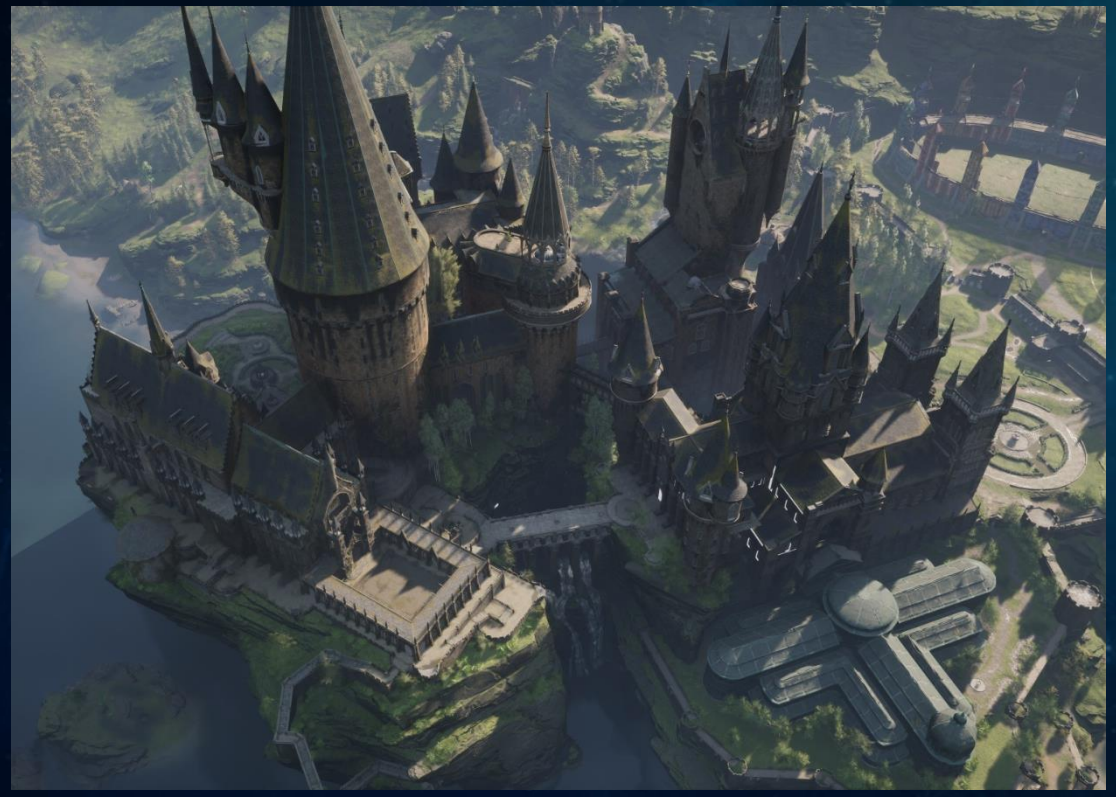


Hogsmeade

- Town containing 40 Unique buildings
- Open world is still loaded and rendering

Hogwarts Castle

- A giant castle containing over 200 rooms



Managing Complexity

- Focus on the following principles
  - Stream out everything that isn't visible
  - Cull everything
  - LOD everything
  - Everything must scale
  - Instance everything

Stream out everything that isn't visible

- First line of defense for GPU performance
- Stream in only what's relevant
- Broom flight puts an extra burden on streaming
- Minimal loading screens

View from game

Zoomed out view with streaming

Zoomed out view without streaming

Cull Everything

- All opaque objects can be rendered with a generalized dithered alpha
- Avoids an ugly pop when things are culled
- Tiny objects fade out
- NPCs fade out
- Inside/outside culling
- Cull lights
  - Distance
  - Between rooms
- Cull shadow casting features on lights
- Skip spawning particles when far from camera

LOD Everything

- Mesh LODs on every model
- Level LOD streaming
  - Bake levels into HLODs
  - Streaming swaps individual levels with baked proxy levels
  - Dithered crossfaded to prevent a pop

Actual Hogwarts Castle

Hogwarts Castle Proxy

Everything Must Scale

- Expensive graphics features are no good if they can only run in an empty room
- Expect most graphics features to be running all the time
- Everything needs an off switch that works without a visual impact

PC

PS4

Switch

Instance Everything

- Render thread is a huge bottleneck
  - Limited to around 10k objects to visibility cull per frame
  - Draw calls limited to nearly the same value
- Instance everything as much as possible
- Bindless resources is a key feature to achieve this

Enhancing Unreal

- Rendering received several enhancements not featured in Unreal Engine 4
- Bindless graphics across entire rendering pipeline
- Compute-based tiled deferred light rendering
- Baked light probe volumes
- Combined volumetric lighting and volumetric fog rendering
- Translucent, water, and volumetric materials support the same major features as opaque render pipeline including shadows
- Everything mentioned in this talk required engine changes

Bindless Resources

Bindless Overview

- Hardware GPU feature fully supported on graphics hardware for the last ten years
- Key feature is that you can instance textures
- Low level consists of an array of resources in which you index in the shader

Advantages

- Unlocks new optimization opportunities across entire rendering pipeline
- No need for texture atlases anymore
- Each texture can be a different size
- Only way to batch 3D textures
- Dynamically read textures in shader with divergent texture reads
- Instance objects that share the same shader, but have different textures
- Single pass light injection for volumetric lighting and fog

Disadvantages

- Shaders have to be built with this in mind
- GPUs that don't support bindless have no easy fallback
- Shipping version of Unreal(4.27 Chaos) doesn't support this
- Integrating into engine is a significant amount of work
  - Version upgrades always needed adjustment
  - Platform support needed
- Divergent texture reads can be expensive on certain hardware
- Limited to around 2000 resources accessible in one draw call/dispatch

Collaboration with WB Games Montreal

- WB Games Montreal shared code to support bindless in UE4
- Added our own enhancements
- What I share with respect to bindless is a combination of both studios' code

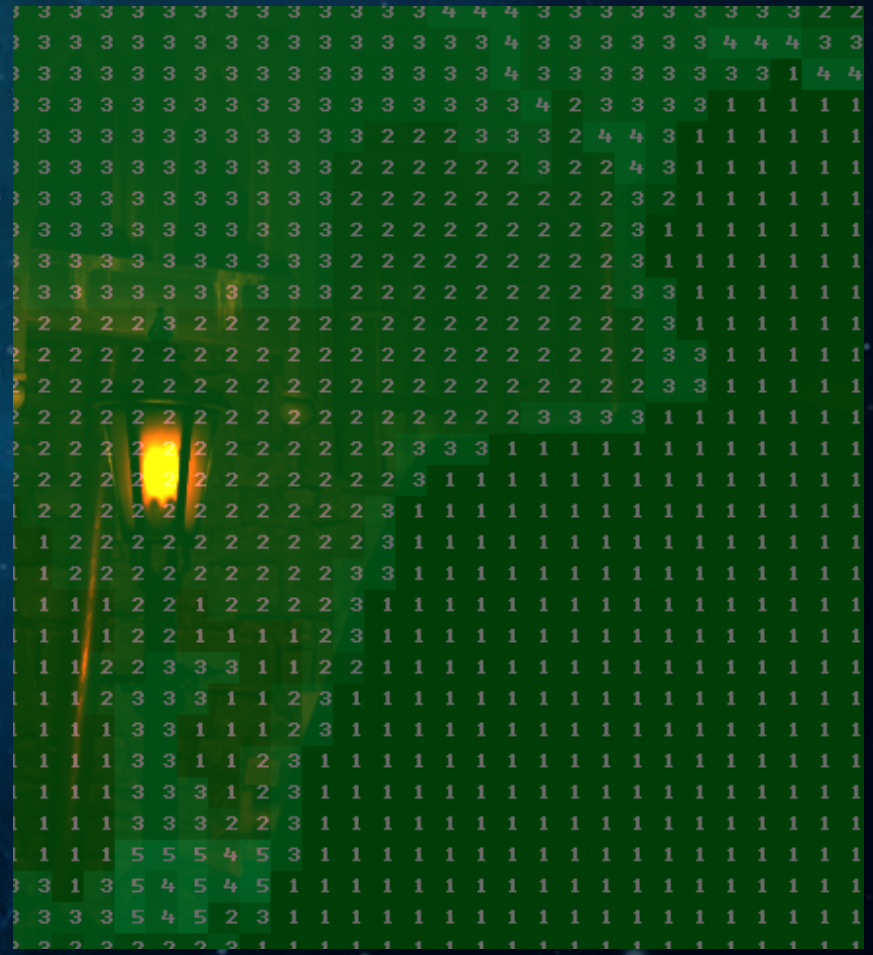
AVALANCHE

Where we use bindless

- Material Instancing
- Volumetric Lighting and Forward rendering
- Light Probes
- Reflection Captures
- Generalized arrays in materials
- Shadows
- Multi-Layered Cloth
- Distant Foliage Proxies
- Decals
- Light Textures

Tiled deferred rendering Overview

- Several bindless features use this
  - Shadows
  - Lights
  - Reflection captures
  - Light Probes
- Divide screen into 16x16 tiles
- Assign for each tile the total number of shadows, lights, reflections, and probes referenced
- 2D tiles for opaque and 3D froxels for translucent



Tiled Deferred Classification

- Categorize each tile based on which features are used
  - Lighting categorized based on
    - Hair
    - Skin
    - Subsurface Scattering
  - Directional shadows based on
    - Transmission
    - Overlapping cascades
    - Distance from camera
- Indirect compute dispatch for each classification type

Material Instancing

- Any material that shares the same shader can now be instanced

Books

Rocks

Volumetric Lighting and Forward rendering

- Support the same number of features as deferred
- Shadows, reflection captures, and probes are fully supported
- Forward rendering is still expensive, so deferred preferred
- All materials now match the environment

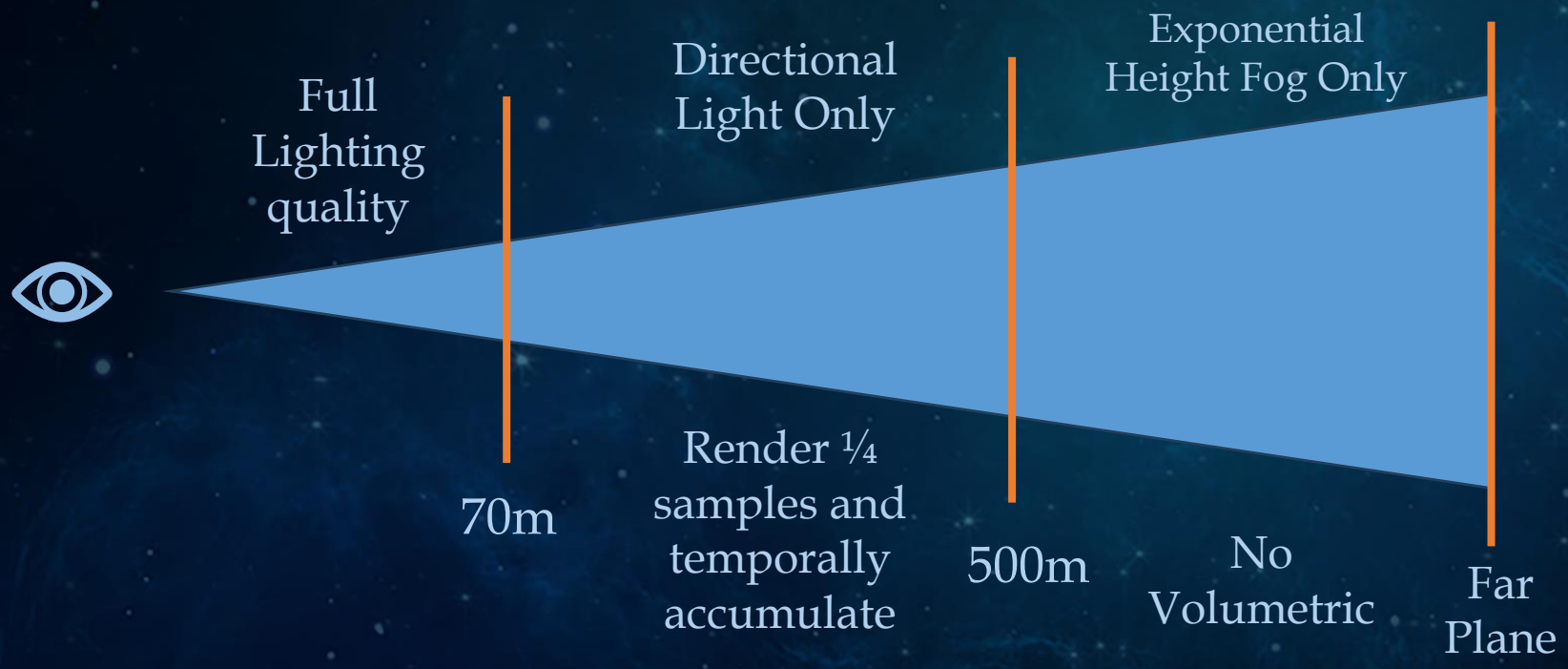
Spheres for opaque, forward and volumetrically lit materials

Volumetric Lighting Enhancements

- Render lighting data into screen space froxels
- This data can then be reused for volumetric lighting and for translucent materials
- Typically limited to a small distance due to cost
- Performance improvements extended distance to 500m

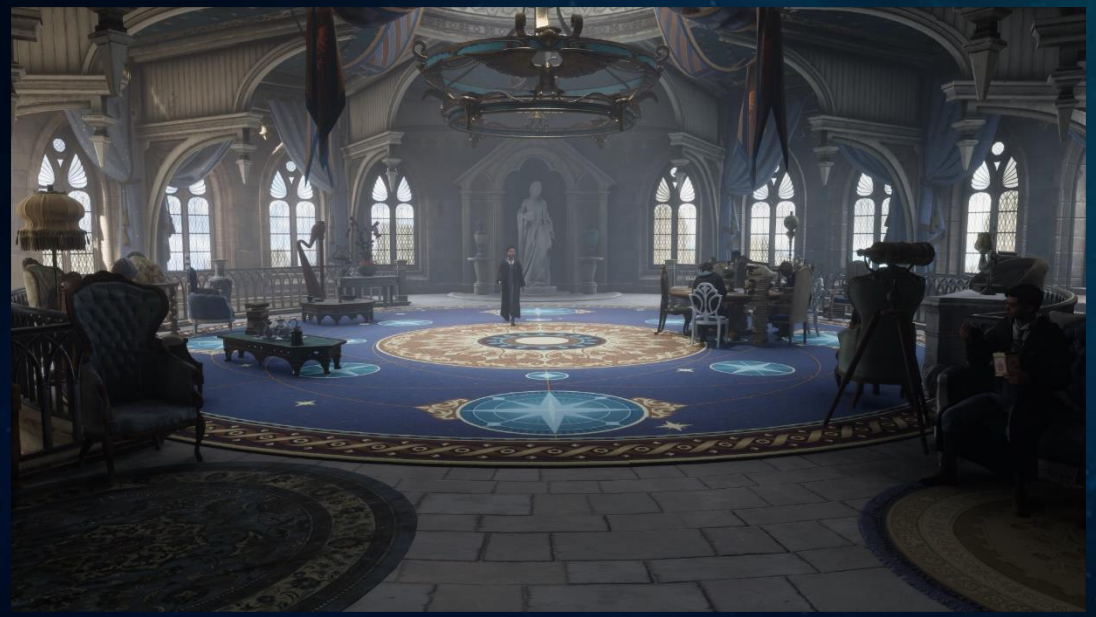
Volumetric Quality Tiers

- Split into quality levels based on Z Distance



Bindless Light Probes

- Most of Hogwarts castle isn't aligned to a grid
- Probe volumes are placed everywhere
- Bindless only way to access multiple probe volumes in a shader

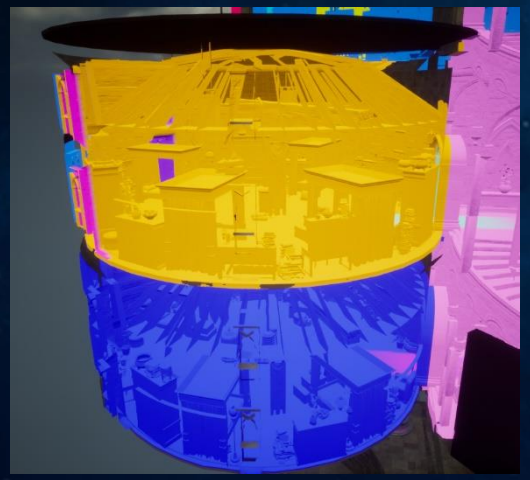


With Light Probes

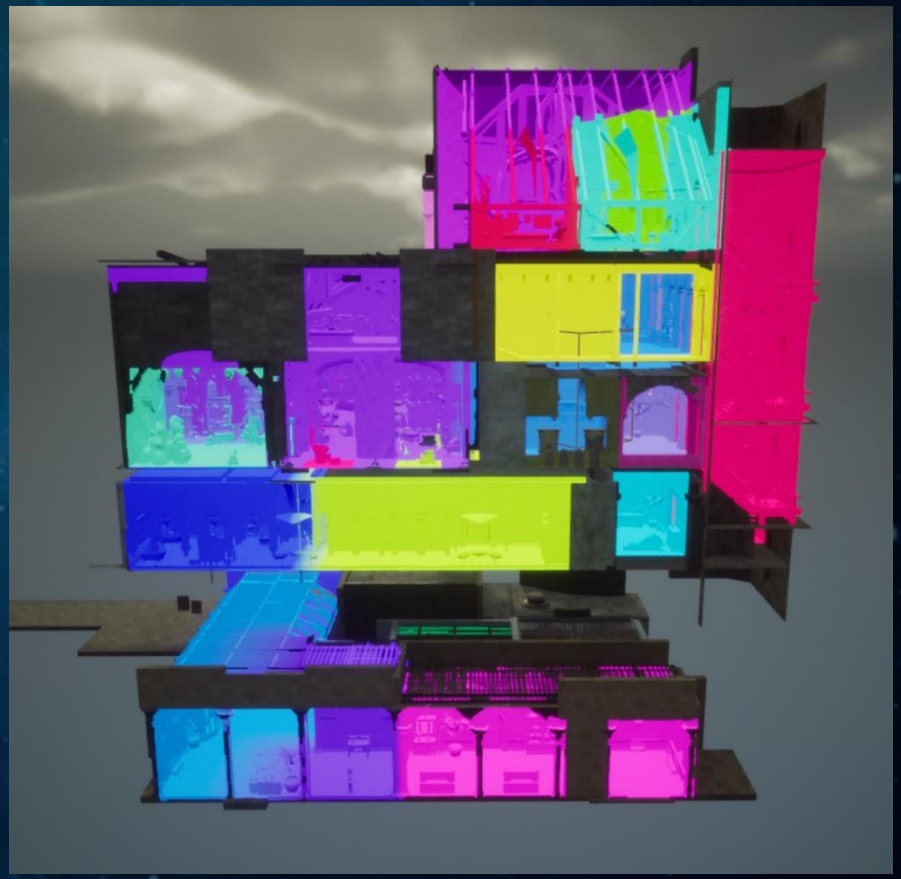
Without Light Probes

Probe Volumes

- Volumes work better in 3D space
- Cube and Cylinder volumes
- Frustum culled on the GPU
- Streaming only subset of volumes



Cylinder Volumes



Gryffindor Tower with probe volumes

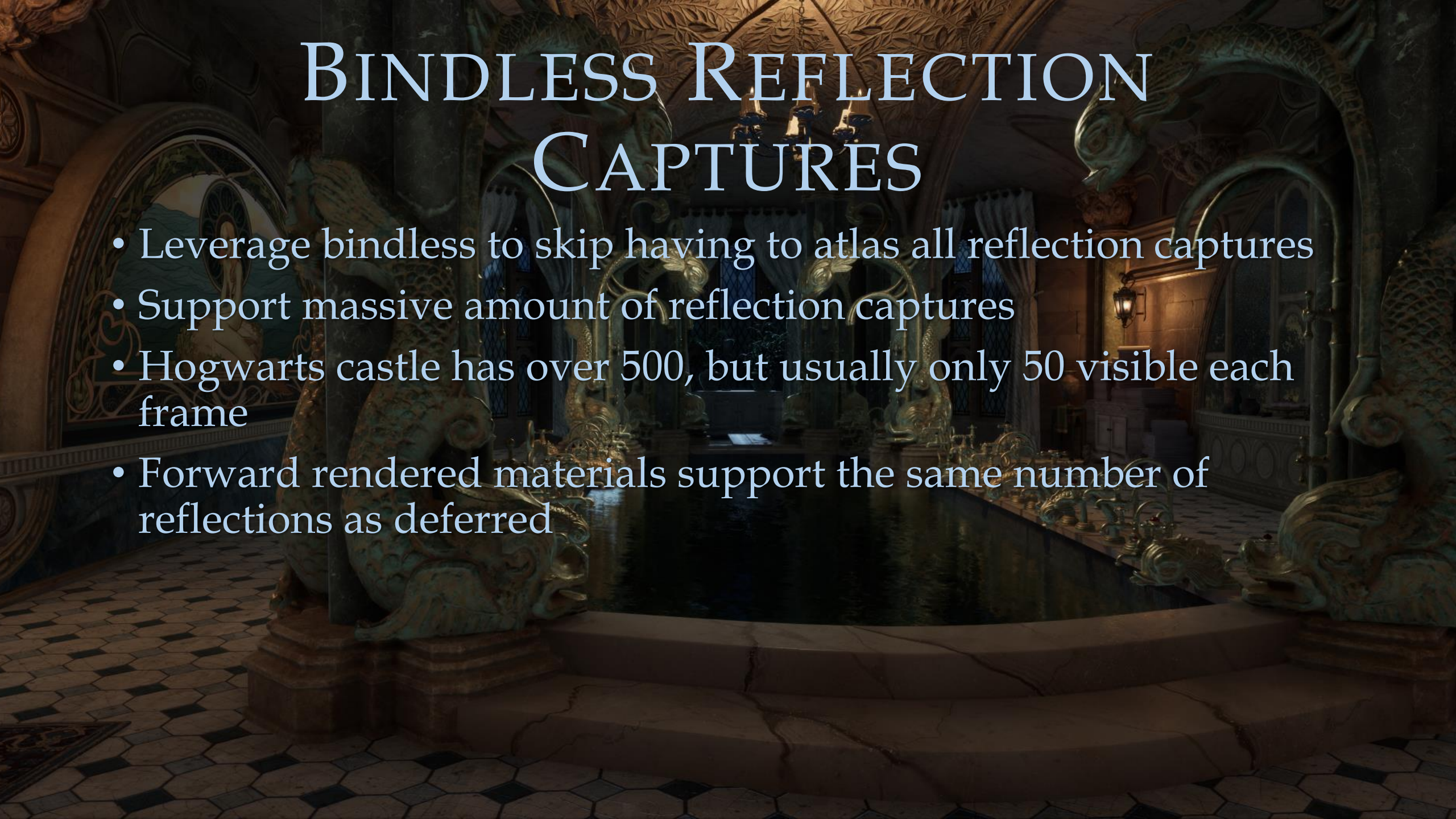
Probe Shadows

- Adds indirect shadowing everywhere
- Contact shadow trace along the dominant light direction
- Used formula for dominant lighting masking [Doghramachi]


PROBE SHADOWS


Without Probe Shadows


With Probe Shadows

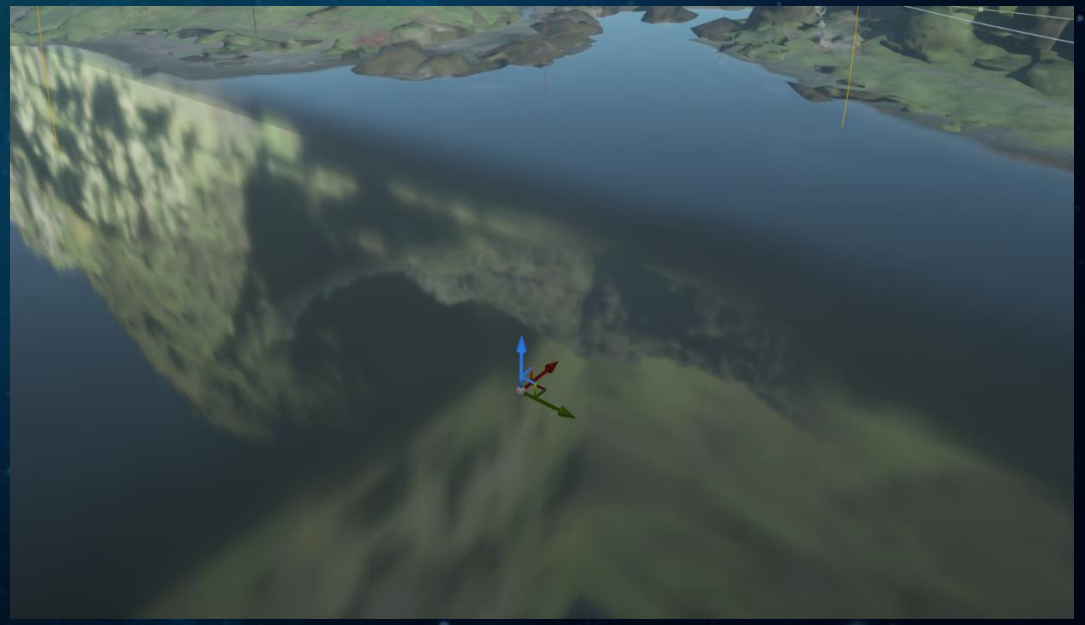


Bindless Reflection Captures

- Leverage bindless to skip having to atlas all reflection captures
- Support massive amount of reflection captures
- Hogwarts castle has over 500, but usually only 50 visible each frame
- Forward rendered materials support the same number of reflections as deferred

Reflections on Water

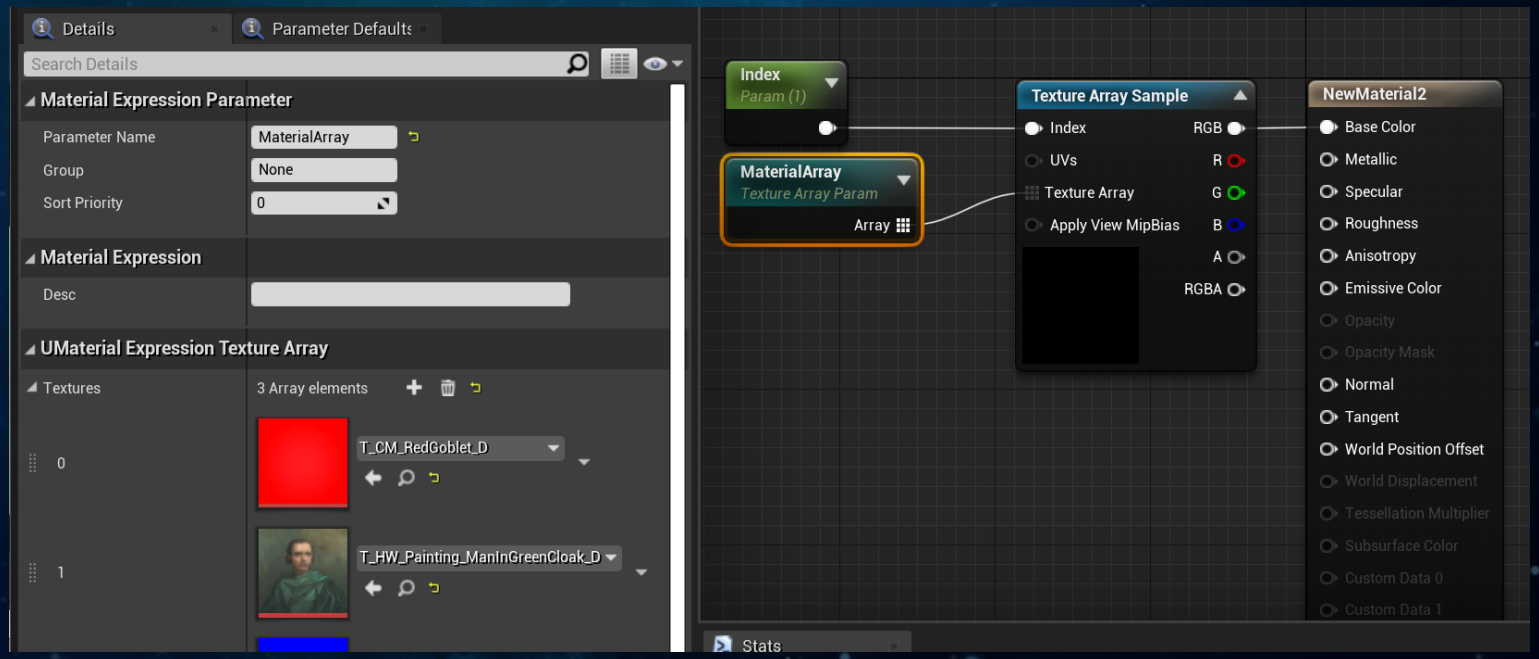
- Parallax corrected maps don't work great with water due to seams between volumes
- Use non parallax corrected cubemap for water
- Assign nearest reflection capture to camera
- Crossfade between reflection captures based on camera position



Seams on water with parallax corrected maps

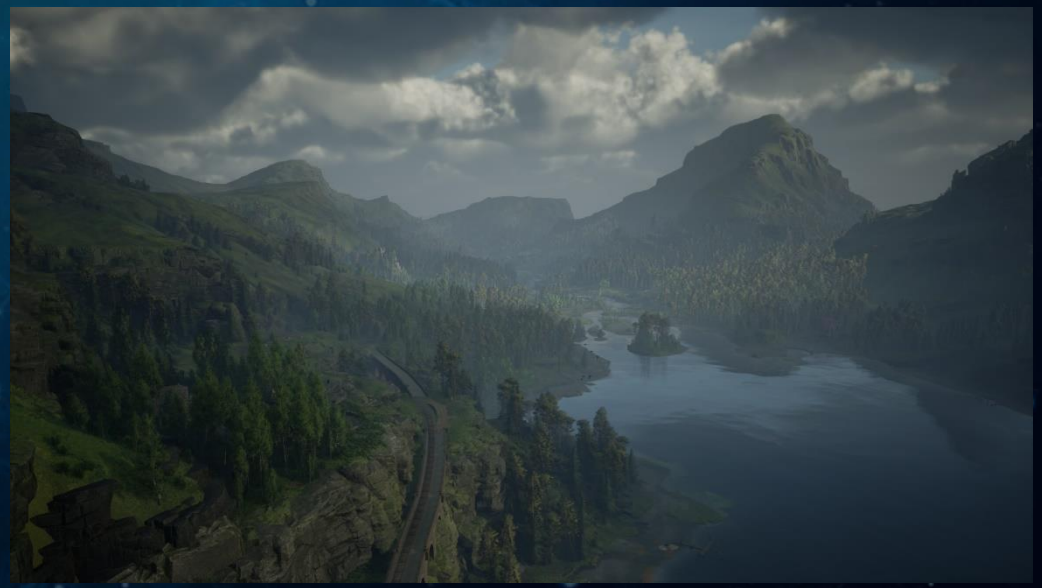
Generalized arrays in materials

- Allow for artists to use parameterized arrays
- Viable alternative to the following
  - Flow control to sample textures
  - Atlased textures
  - Texture arrays



Bindless Distant Foliage Proxies

- Used to render at trees at very far away distances
- Additional step after baked foliage proxies
- Further reduce draw calls by running a batching step to instance all distant foliage together



Bindless Shadows

Light Complexity

- Tiled Deferred Lighting handles light overlap a lot more efficiently
- How can we render with hundreds of shadow casting lights?

Bindless Shadows

- Access a significant number of shadow maps in the same rendering pass
- For opaque materials, compute shadows via separate compute pass
  - One dispatch per light type
  - Data packed into an SRV, which is then looked up in compute light pass
- All shadow maps available in volumetric and forward render passes
- Observed over 250 shadow maps used in one frame in several parts of the game

Point and Spot Shadows

- Fully dynamic shadows for every light still not feasible
- Leverage a set of features that scale based on distance from camera
- Over 200 shadow maps
- Biggest focus is ensuring most shadows are cached

Feature matrix per light

- Cached Shadow Map
- Movable Shadow Map
- Contact Shadow
- Capsule Shadow
- Each of these features can be turned on and off independently

Cached Shadow Maps

- Majority of all shadow maps are cached
- Resolution computed based on size on screen
- Allocated by resolution buckets

Spot Lights

Number	Resolution
4	2048
16	1024
32	512
32	256
32	128
32	64
32	32

Point Lights

Number	Resolution
2	1024x6
8	512x6
32	256x6
32	128x6
32	64x6
32	32x6

Movable Shadow Maps

- Render only nearest movables per frame
- Turn off on non-important lights
- Boost mode during cinematics for 4k shadow maps
- Write out 1/32 resolution UAV used to skip reading movable shadow map

Contact Shadow

- Adds additional details on cinematics
- Alternatively get some shadowing without rendering a full shadow map

Capsule Shadow

- Used as a proxy for skinned geometry when no movable shadows are enabled
- Capsules computed mathematically
- Reduce per-pixel complexity with capsule volume culling
- Enhanced UE4's existing capsule shadows

Lumos Shadows

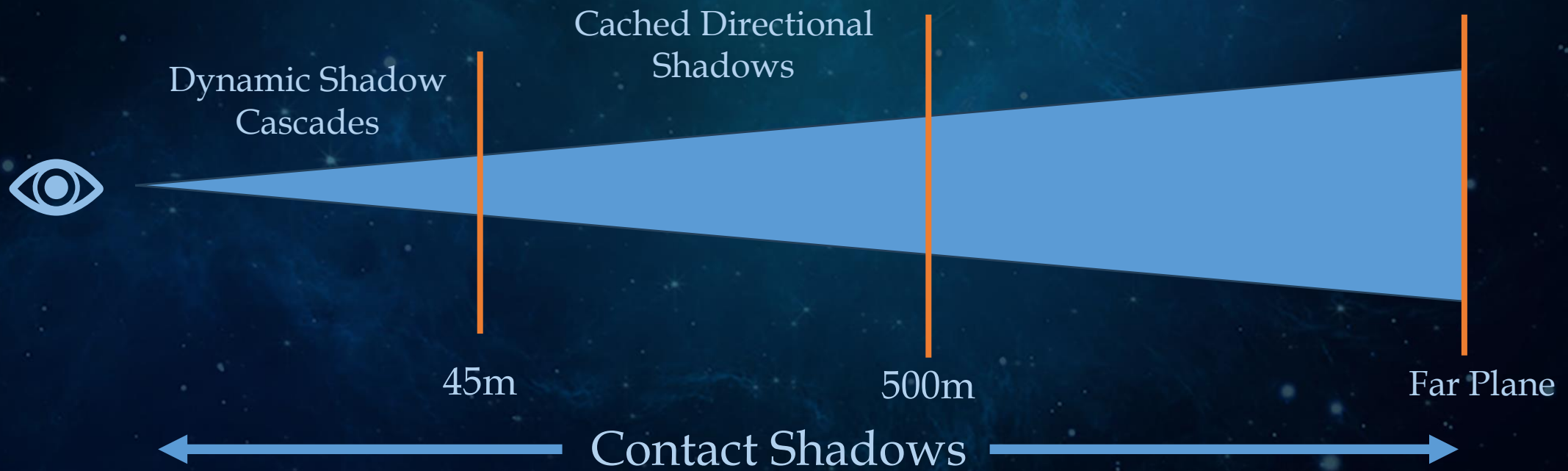
- Lumos is the spell used to spawn a light source on your wand
- Maintain authenticity of the movies using a point light
- Requires a dynamic cube map shadow, which is expensive
- High end platforms can handle this well
- What about low end platforms?

Lumos Shadow Scalability

- Shadows active only when avatar casts Lumos and on not on any other NPC
- Use SV_RenderTargetArrayIndex in vertex shader so one pass point light no longer needs to render with a geometry shader
- Skip trying to cache shadow map since light is always moving
- Most important part of shadow is avatar and other movable characters
  - Low end platforms render only movables to maintain illusion of movable shadows

Directional Light Shadows

- One of the largest expenses in the game
- Limit to one directional light



Dynamic Cascaded Shadows

- Up to three cascades
- Fully dynamic
- Moves with time of day

Contact Shadows

- Saves a significant amount of performance by not rendering small objects like grass and rocks
- Adds shadows in the distance where there's no shadow maps


With Contact Shadows


Without


With Contact Shadows

Cached Directional Shadows

- Shadows get significantly more expensive to render the farther from the camera you get
- Not easy to cache directional shadows, especially with a moving directional light
- Leveraged a tile-based caching system
- Based on the work of [Turchyn]
- This is very fast to compute with bindless shadow maps
- Scaled very well from the fastest platforms down to the Nintendo Switch

Overview

- Project camera frustum points to light space
- Create light space convex hull around points
- Split into 128x128m tiles aligned to a light space grid
- Shadow map tiles can be reused
- Limit to rendering only one tile per frame

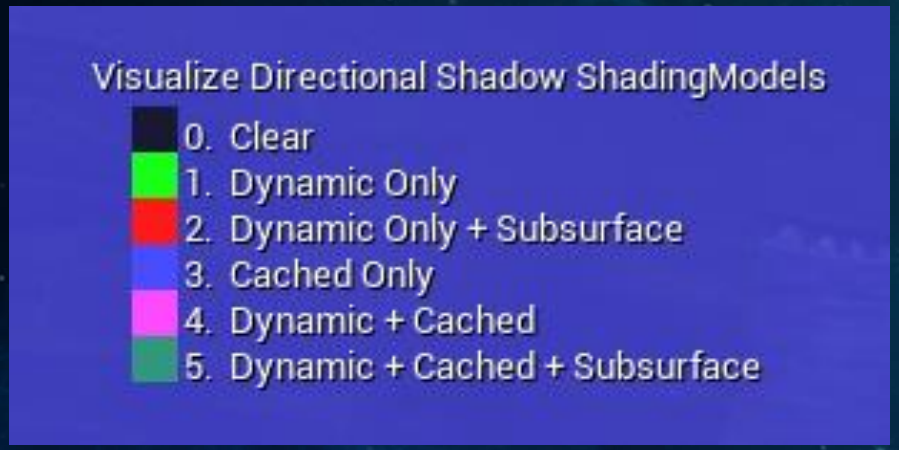
Light Space Tiles

Shadow projection

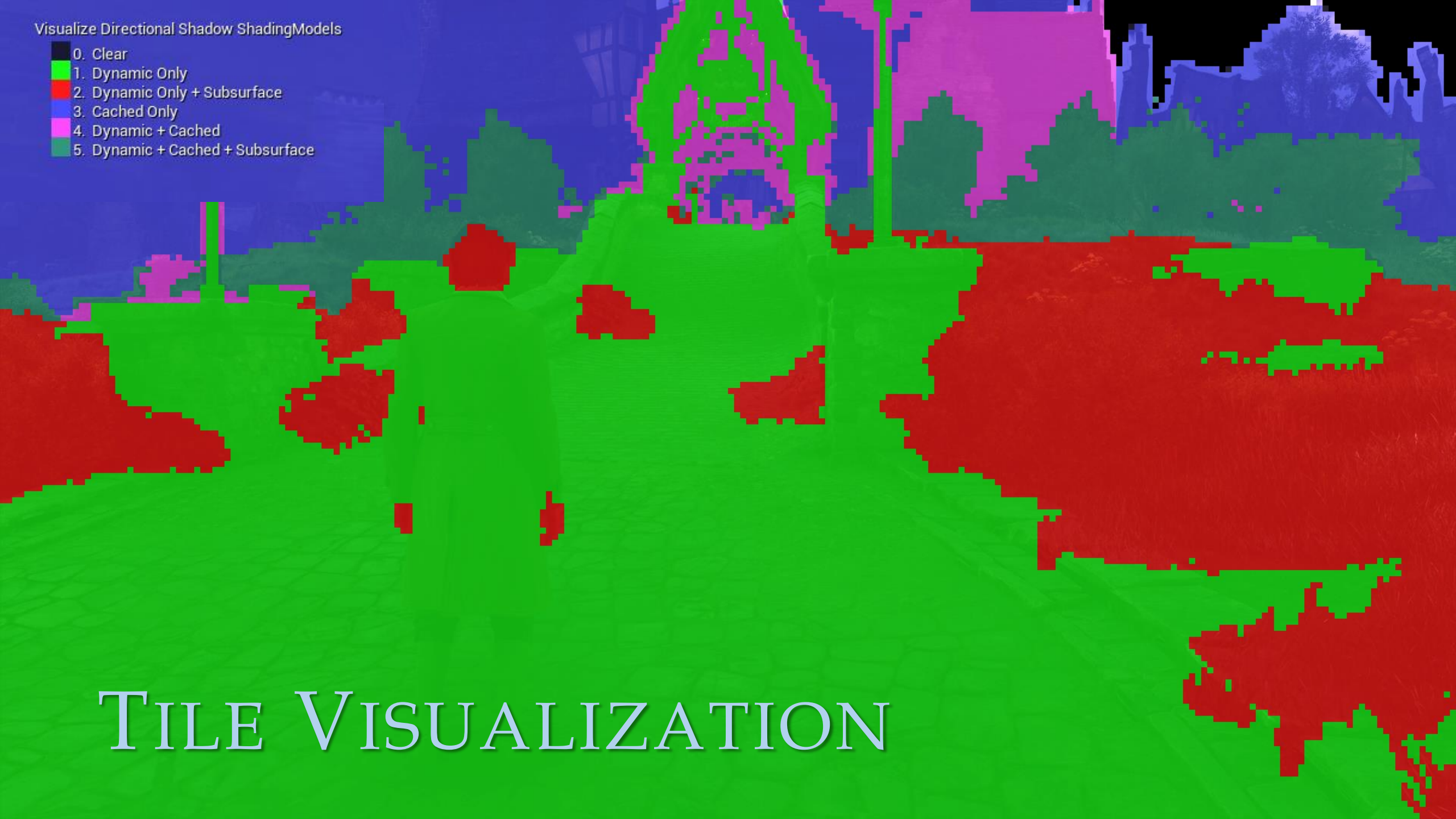
- Pre-sort tiles based on distance from camera
- Run a compute job that assigns each screen space tile any shadow map tiles that collide
- Run another compute job that calculates the shadows per pixel
  - Pick only the first shadow map that is in bounds and skip the rest

Tiled Deferred Classification

- Mentioned earlier how this is performed
- Shadows classified by the following permutation combos



- Reduces the number of dynamic branches and shader GPRs



Tile Visualization

Shadow Projection Performance

- Cached directional shadows and tile classification made this very inexpensive to calculate
  - Scalability settings to make up the difference between platforms
- GPU time to render entire directional light shadow mask
  - Dynamic cascades
  - Cached shadows
  - Contact shadows

Platform	Time
PS5	0.47ms
PS4	0.65ms
Switch	0.95ms

Caching only One Tile per frame

- Usually not enough without any extra effort
- Render closest uncached tiles first
- Camera cuts cause surge in requested tiles
  - Render two tiles per frame after a camera cut until cache has been filled
- Avoid flushing cache at all costs

Tile Cache Management

- Conditions where the cache is invalidated
  - Moving objects
  - Time of day
  - Streaming in/out objects
  - LOD transitions

Moving Objects

- Skip rendering all moving objects in cache
- Foliage that waves in wind still rendered in cache, but ignore invalidation
- Not that visible due to 45m dynamic cascade distance

TIME OF DAY

- Game runs full time of day in 48 minutes
- Constantly changing light direction prevents caching
- Cache light direction
  - 12 second window where tiles are cached
- After window has elapsed, start caching tiles for new light direction
  - Takes around 50 frames for cache to be filled
  - Takes longer when camera is rotating since existing cache needs to be updated first
- Crossfade in shadows from new light direction

Shadow Crossfading

- Sped up so it is easier to see
- Crossfades normally occur every 12 seconds

Streaming in/out

- For each object streaming in/out
  - Check if bounds intersect with existing tiles in cache
    - Increment dirty value for tile
- Re-render tiles only if existing cache is full and there's no crossfading active
  - Time threshold
  - Dirty value threshold
  - No individual tile crossfading as you tend to not notice it in the distance

LOD Changes

- Self occlusion artifacts when shadow map LOD is mismatched from main camera

Matching LODs

Mismatched LODs

- LODs are always swapping so it's not feasible to invalidate all the time
- Only large LOD discrepancies are noticeable
- Solution: Store position of camera when tile is cached
  - Only invalidate tile when distance from camera position and cached position is very large

Shadows at sunrise and sunset

- Grazing angles cost significantly more to render than overhead
- Always playing at the base of a valley where mountains always shadow at sunrise/sunset
- Turn off directional light at sunrise/sunset
  - 5:20am-7:12am
  - 5:05pm-6:47pm
- Directional light still rendered in atmosphere and skylight
- Despite having no direct shadows, light probes gives good ambient shadows

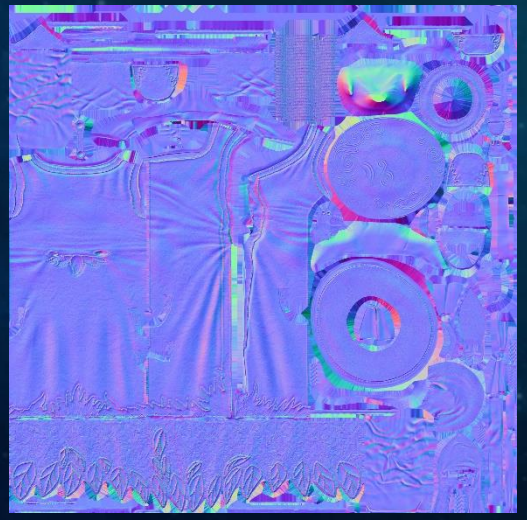
Bindless cloth regions

Multilayer Materials

- Originally used a material with up to 16 cloth layers blended via weight maps
- How can we use bindless to make this faster?

Cloth Material

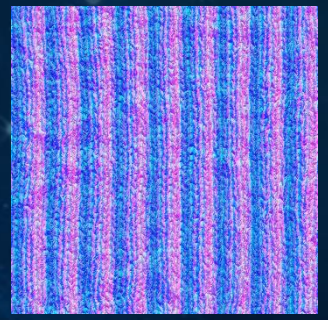
- Cloth material blends both macro and micro detail
- Macro detail like cloth wrinkles, and other specific detail
  - No multilayered regions here, just a standard unwrapped UV

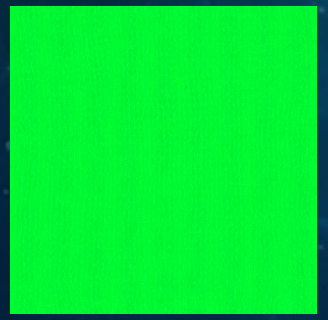
Micro Detail

- Multi-layered regions only used on micro detail
- Micro detail for tileable cloth seams and textile tile patterns
  - Sharpness in extreme close ups
- Each layer only three textures using PBR principles

Base Color



Normal



Packed Metalness, Roughness, and Fuzz

- Represents cloth, leather, silk, fur, metal, and other materials

Bindless Cloth Regions

- Read a texture that determines which layers to blend
- Shader assumes only 1-2 layers to blend
  - All layers use the same code
- Majority of cloth layers don't overlap too much, so majority of pixels only blend one layer

Blend mask Textures

- Start with up to 16 blend mask textures

- Baked into two textures
  - Region ID
  - Blend

Packing Method

- For each pixel
  - For all mask textures, get the two largest values
    - Each value is a max of all adjacent pixels
  - If only one value
    - Write out mask index into region ID texture
    - Write out 1 in blend texture
  - If two
    - Pack both mask indices into one 8-bit value into region ID texture
      - Two 4-bit IDs are bitmask OR together
    - Write out weight used to lerp both layers in blend texture

Result: Packed Region ID and Blend Texture

Packed Region ID
Not compressed

Blend Texture

Bilinear Filtering

- Region ID textures are packed and cannot be filtered
- Blend texture can be filtered
  - Remember that region IDs are derived from individual blend mask textures
  - Region IDs just need to be dilated across adjacent pixels to cover the sample extents for bilinear filtering

Mip Map Generation

- Standard mip generation using averages will not work for region ID texture
- For each mip level
  - Downsample all original blend mask textures
  - Generate region ID map from these new blend mask textures
  - Dilate adjacent pixels again

Sampling Cloth Regions

- Sample HLSL Code

```
int IDPacked = IDTexture.Sample(IDTextureSampler, UV) * 255;

int ID0 = IDPacked & 15;

float2 MicroUV = RotateScaleOffsetUVs(UV, RotationScale, Offset);
float MicroTexture0 = BindlessTable[NonUniformResourceIndex(ID0)].Sample(MeshTextureSampler, MicroUV);

int ID1 = IDPacked >> 4;
if(ID1)
{
    float MicroTexture1 = BindlessTable[NonUniformResourceIndex(ID0)].Sample(MeshTextureSampler, MicroUV);
    float Weight = WeightTexture.Sample(MeshTextureSampler, UV);
    MicroTexture0 = lerp(MicroTexture0, MicroTexture1, Weight);
}
```

- NonUniformResourceIndex in HLSL enables divergent texture reads
- In Unreal, this is handled in a material graph using the generalized arrays in materials feature I mentioned earlier

Generalized Micro detail

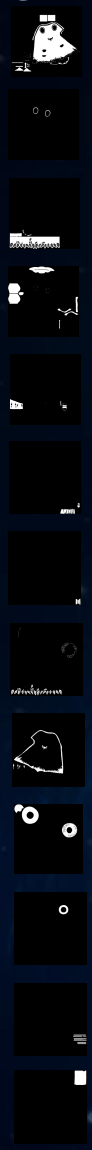
- Every layer must have the exact same code
- UV code generalized via a 2x2 RotationScale matrix and float2 bias

```
float2 RotateScaleOffsetUVs(float2 UV, float4 RotationScale, float2 Offset)
{
    return float2(dot(UV, RotationScale.xy), dot(UV, RotationScale.zw)) + Offset;
}
```

- Rely on PBR concepts so that the material can represent all cloth varieties
  - Textures for Base Color, Normals, Metalness, Roughness, and Fuzz
  - Parameters for Specular, Base Color Tint, etc.

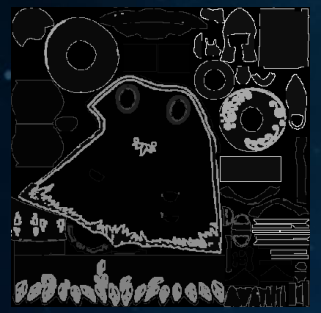
Final Result

Painted
Region masks



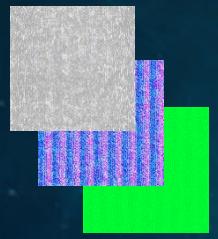
Packed
to

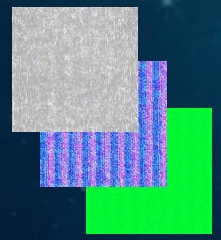
Region ID



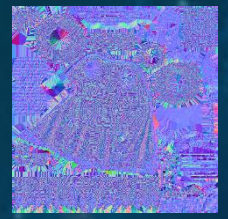
Region Weight

Micro
Detail





Macro
Detail




Cloth up close

Disadvantages

- Divergent texture reads are required and slow on some graphics hardware
- Low mip maps can have some bleeding problems on region edges
- Tri or quad points between regions don’t blend well with only 2 layers per pixel

Advantages

- Not sampling 16 layers in one material
  - Non-packed
    - (3 PBR textures + Mask Texture)* 16 layers = 64 texture samples
  - Packed
    - 3 PBR textures * 2 layers + ID + Weight = 8 texture samples
- High resolution maps on micro details
- Micro detail maps shared across all cloth materials in game
- Small details can be added without adding cost
- No need to author unique materials per house

Optimizing GPU performance using automation

Optimizing the GPU

- Despite all these mentioned optimizations, more optimizations were needed from both content and code
- Automation tests helped get us to reach our performance budgets

Avalanche's Automation System

- Runs several times a day
- Tracks CPU and GPU counters
- Over 300 tests all over the world
- Each test rotates the camera three times with three different times of day
- Recorded video
- Details in another talk [Villeta]

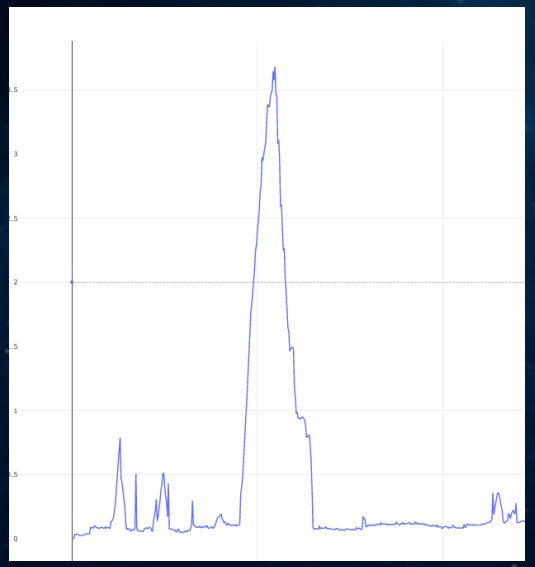
Markers of all tests

Results

- Ensures optimizations covers the entire game
- Any performance regressions were quickly fixed
- All teams were on the hunt to drop their numbers
- Results for just PS4 with 2 years of optimization data
  - Started optimizations with only 25% of tests in budget
  - Shipped with 95% of tests in budget
  - One test dropped from 68ms per frame to 26ms
- All platforms had similar results

VFX Performance Spikes

- Resulted in optimizing most VFX related spikes from 50ms to under 3ms
- Link graph's frame number with video to see which VFX needs to be optimized



Platforms Optimization

- Automation can expose unique performance quirks on each platform
- Use results to adjust scalability settings between platforms
- Easy to find problems on low end, and some can be reapplied to all platforms

PC

- Low Nvidia & AMD
- Medium Nvidia & AMD
- High Nvidia & AMD
- Ultra Nvidia & AMD
- Ultra RT Nvidia & AMD

Sony

- PS5 Fidelity
- PS5 Fidelity RT
- PS5 Performance
- PS4 Pro
- PS4

Microsoft

- XSX Fidelity
- XSX Fidelity RT
- XSX Performance
- XSS
- Xbox One X
- Xbox One

Nintendo

- Switch Handheld

Visual techniques developed for Hogwarts Legacy

Summary

- Time of Day
- Seasons
- Material Permuter
- Weather
- Live Paintings
- Clouds

Time of Day

- Full cycle in 48 minutes
- Sun arc fixed to one specific day of the year
  - Needed for baked light probes
- Everything must be carefully arranged to get it right
  - Lights
  - Fog
  - Clouds
  - Sky
  - Emissive materials

Physically based light values

- Use physically based lights to get the right balance between lights
- Pre-exposed color is required with a sun light this bright
- HDR works extremely well

Light	Value
Sun	100,000 lux
Moon	1 lux
Sky	1-31,000 lux
Point and Spot lights	1-30 Candelas

HDR Enhancements

- Added ACES SSTS parametric tone curve
- This is a single-stage tone scale that adapts ACES luminance values to display luminance values
- Used HGIG display querying to get the display luminance values
- Added HDR visualizer to help author dynamic range

Light leaking

- Extremely bright sun light makes it obvious if we have any light leaking indoors
- Volumetric lighting and fog are very easy to leak as they render out to low resolution screen space froxels
- Used inside volumes to block light in addition to just geometric blockers

With light leaking

Without light leaking

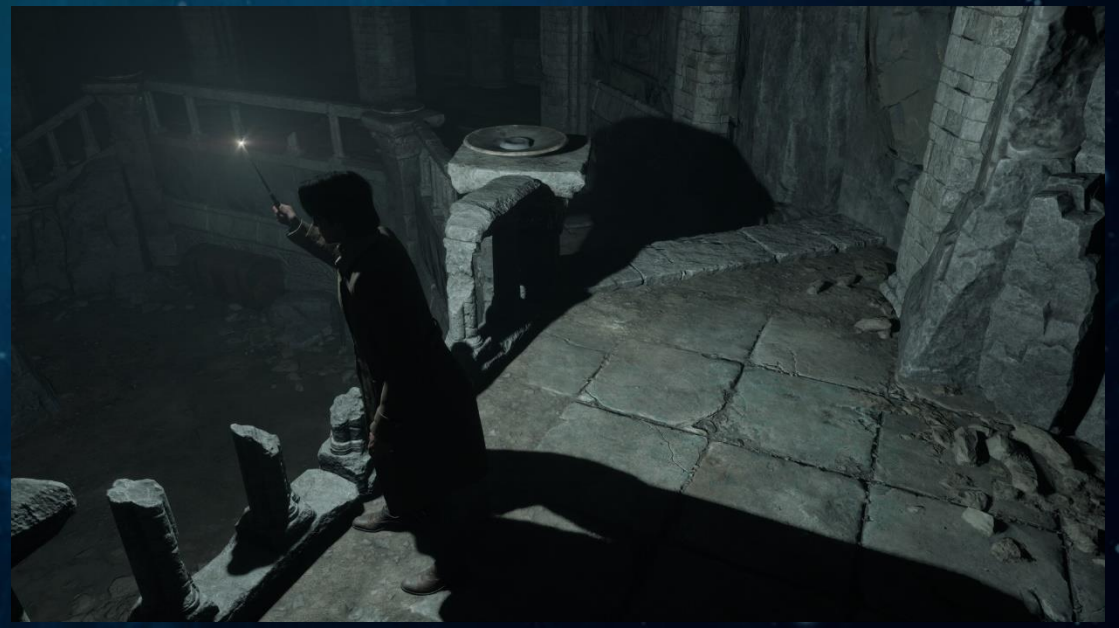
Exposure management

- Drove 8 post processing parameters with curves for time of day
  - 5 curves for lighting tied to time of day
  - 3 curves tied to sun's altitude
- Use an exposure compensation curve, no exposure metering mask
- Outdoors exposure min/max set at [-14..16]

Exposure Compensation Curve

Adaptive Lights and Emissive color

- Used to make sure lights and materials are not too bright or dark in variable situations
- Four methods
  - Autoexposure based
  - Average scene luminance based
  - Probe based
  - Time of day adjusted



Autoexposure

- Current exposure of the camera
- Lumos must look bright regardless of where you are
  - Breaks down when moving from outdoors to indoors
  - Add a special flicker effect on lumos to reset auto exposure

VFX

- All magic effects need to look bright regardless of time of day
- Typical VFX uses fixed brightness, which would look dim during the day and too bright at night
- Used auto exposure to ensure everything would be the right brightness and bloom correctly

Problems with Average Scene Luminance

- Value based on previous frame's average luminance
- Falls apart on first frame after a camera cut
- We can either override with an authored value or stick with last frame's value
- Not guaranteed that this will work all the time
- Lumos would flicker to black whenever we had a camera cut, large exposure changes, and other conditions

Probe Based Adaptation

- Use light probe data for adaptation
- Probe data is available on the same frame rendered
- Cinematics rely on this to get flicker free adaptive lights on camera cuts
- Ghosts also rely on probe adaptation to adjust their brightness to match the environment

Generating Light Probe Adaptation

- Sample probes about a meter from the camera
- Probes are sampled exactly like a forward rendered material
- Save results to a buffer
- Material and Lighting passes reads this buffer to adapt

Time of Day adjusted Lights

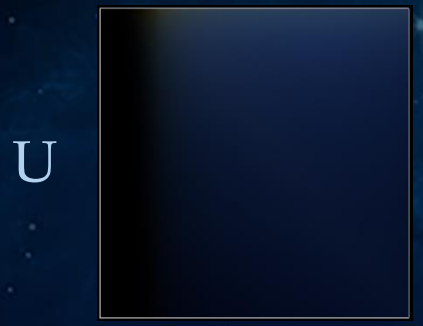
- Sampled SkyAtmosphere shader on CPU
- Adjust light brightness and/or color for time of day

Time of Day Emissive Materials

- Update parameters on the CPU
  - Windows
  - Lanterns

Sky Adjustments

- Sky goes completely black during dawn/dusk
- Added HDR horizon to zenith texture in sky atmosphere shaders
  - Simulates nighttime scattering
  - U = negated height of the sun in degrees from -12 to 90
  - V = Z of the sky dome



5x brighter to visualize

Night Sky

- Moon based on real NASA data with accurate lunar phase and lighting
- Four visible planets
- Accurate stars with 41,475 brightest stars in the sky
  - Positioned accurately to the 1890's time and location
- Hand painted milky way galaxy

Night Sky

- Star count scales based on platform
- Rendered with Unreal's particle system via a custom Niagara data interface
- Astronomy minigame uses the exact night sky to highlight constellations and individual stars

Seasons


Summer


Autumn


WINTER


Spring

Seasons

- Adding season specific calculations to each material would be expensive, especially if you have unique resources per season
- Not all things have to change based on season
- Usually only foliage, grass, and the landscape textures

Season based asset swap

- Only stream assets based on the current season
- Materials
- Meshes
- Levels
- Foliage
- Landscape Textures
- Particles

When to change seasons

- Visually striking when the season changes
- Season not designed to seamlessly change
- Story advances the season
- Used a prerendered video to transition seasons

Region-based Seasons

- Coast part of the world has no snowfall unlike the rest of the game
- Only draw snow in the mountains

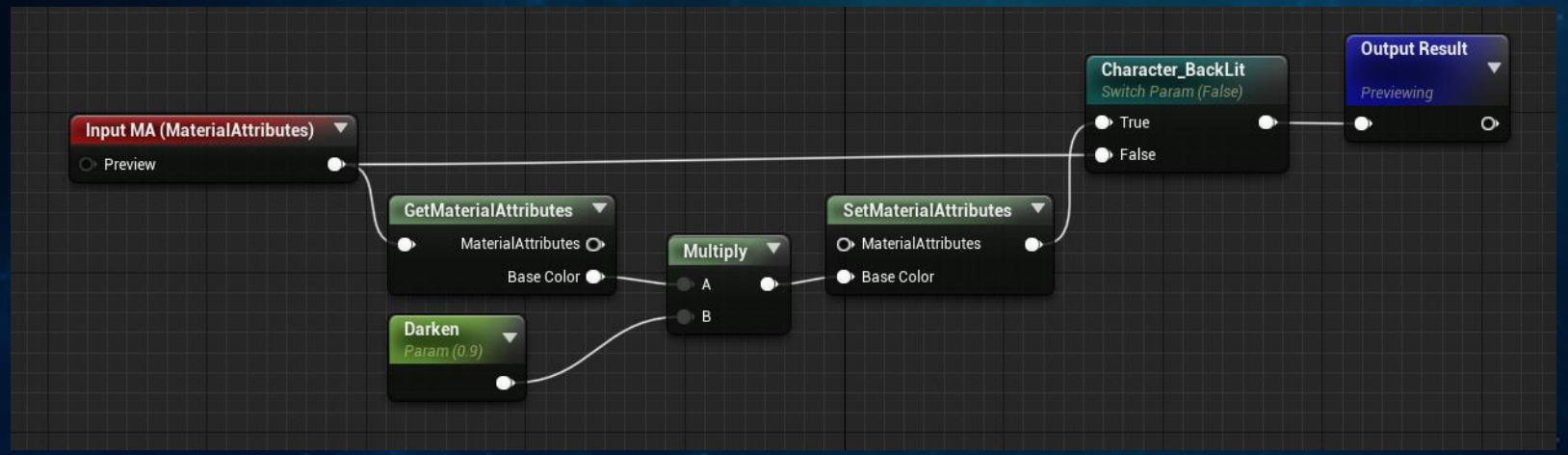
Region-Based Seasons

- Certain missions have fixed winter high up in the mountains
- No extra artist involvement here as everything was already set up for seasons

Material Permuter

Material Permuter

- Performs live material swaps
- Builds permuted materials via toggled specific static switch parameters in material graph

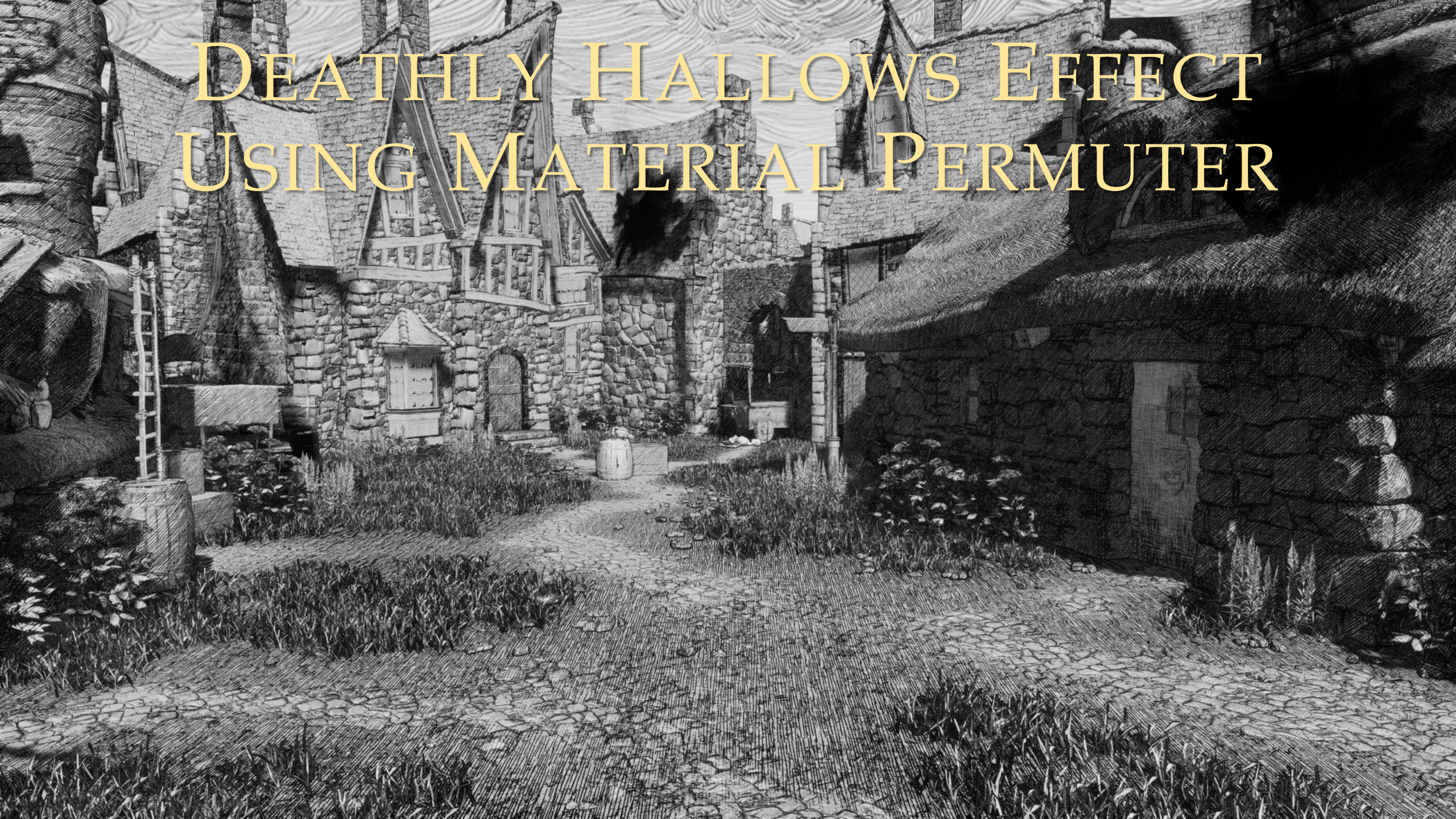


- Used for weather swaps, magic, and live paintings
  - 52 different static switches in game
  - Total of 1478 permuted materials

Material Permuter Details

- Override type of material, like translucency, double sided, dithered opacity, etc.
- Can also be used to override material parameters and swap rather than permuting shaders
- Any material setting can be overridden and not just settings shown in the UI
- Allow exceptions for unique materials that don't permute very well to allow for hand crafted materials

Deathly Hallows Effect Using Material Permuter



Red/Blue Portal Dungeons

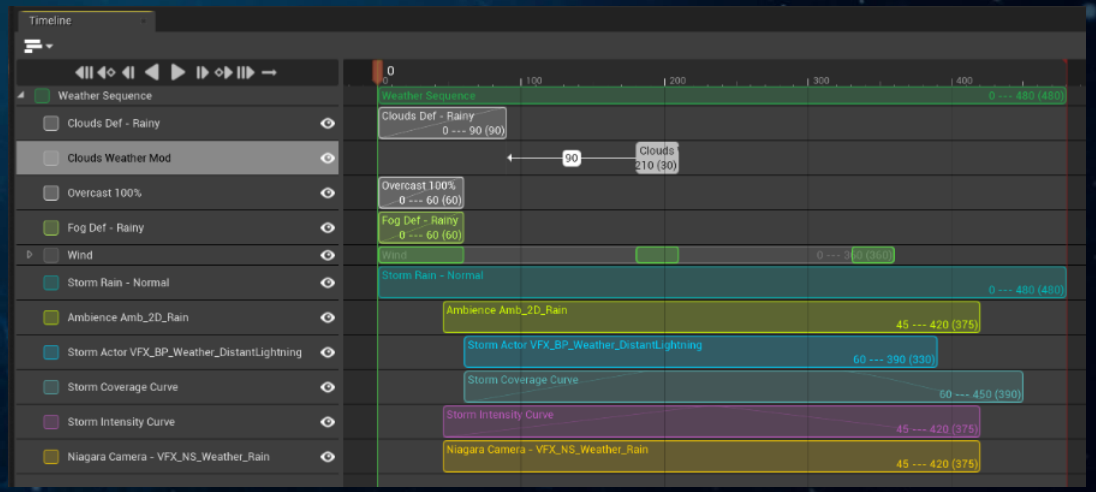
SkinFX

- SkinFX is a blueprint-based material parameter animation system built for FX Artists
- Used for magic and other effects

Weather

Weather Authoring

- Control weather based on time of day
- Highly authorable
- Generalized to support multiple types of weather without custom code for each
- Dynamically rendered tiled weather map used for rain drops
- Alpha value used for weather accumulation



Weather Decal

- Render full screen decal that covers everything
  - Avoids baking weather into all materials
- Anything that doesn't look good use a material swap via the material permuter
  - Moving objects
  - Translucent materials
- Use stencil masking on swapped materials to avoid rendering in the weather decal
- Stencil mask also used to exclude weather on anything

Weather on moving objects

- Use material permuter to swap to weather variants
- Each moving object stores a weather state and sent to material
- Accumulate weather over time
- Decay weather if indoors
- Can be expensive, so scale based on distance from the camera
  - Only activate on nearby objects
  - Update indoor/outdoor state less frequently
  - Use AABBs rather than Convex Hulls

Weather Masking

- Weather should not be received indoors or on under hangs
- Dynamic top-down render pass would be easy, but too expensive
- Mask weather in the following ways:
  - Captured top-down virtual texture to mask trees and buildings
  - Normals to mask under hangs
  - Inside Volumes
- Translucent particles like raindrops and snow flurries masked with distance fields baked from inside volumes

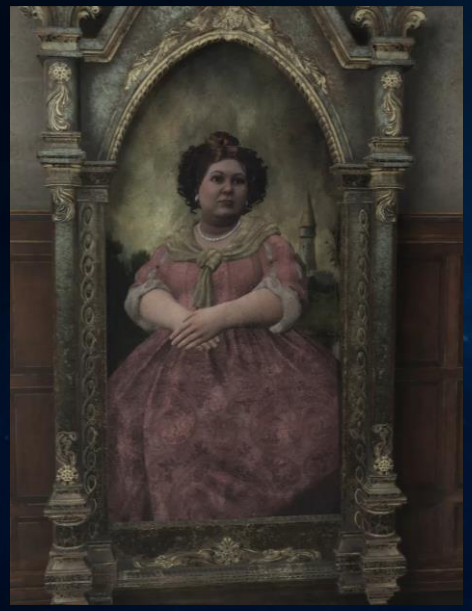
Snow

- Decal doesn't look good on everything
- Foliage billboards doesn't look great
- Winter already has baked in snow, so rely on this for snow foliage

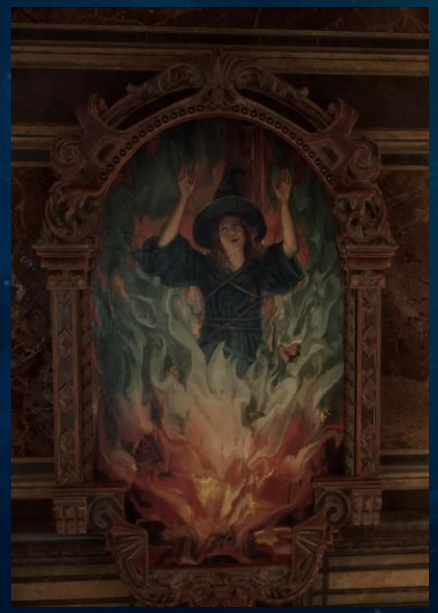
Live Paintings

Four Quality Tiers

3D Dynamic Characters



2D Dynamic Characters



Baked Movie Atlas

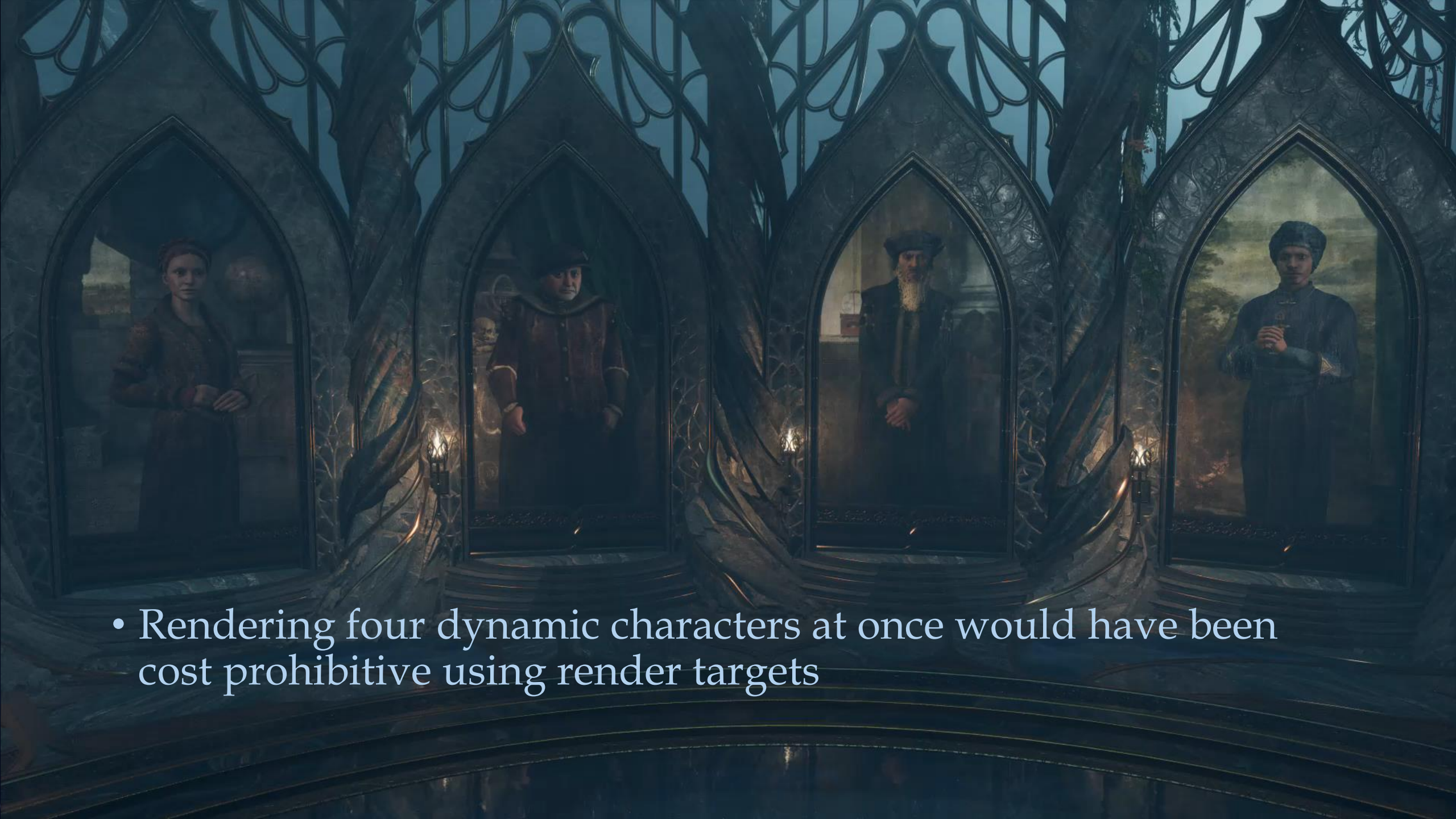
Fixed Paintings

3D Dynamic Characters

- Easy version is rendering to a render target
- Expensive to render and resolution will never match on screen
- Instead flatten rendering to a plane

Flattening

- Depth buffer is still needed, so flatten as much as possible while preserving some depth range
- Relatively cheap to render
- Use material permuter to swap to flattened shaders
- Shader performs flattening and lighting



- Rendering four dynamic characters at once would have been cost prohibitive using render targets

Baked Movie Atlas

- Bake 3D characters to recorded movies
- Save overhead of rendering dozens of movies each frame
- Varies between 2-25 movies
- Limit only one atlas active at once
- Any other movie atlases fall back to static picture

Volumetric Clouds

- Developed in house
- Based on the work of Nubis[Schneider]

Final Thoughts

- Hogwarts Legacy was a very complex game which we managed with a variety of techniques
- Bindless resources helped us achieve our performance targets across all platforms
- Automated tests was a critical component in this process
- All our visual techniques helped us stand out amongst other games using Unreal Engine
- Really proud of Hogwarts Legacy's success

Thanks to

- Avalanche Render Tech Team
  - Chris Hall
  - Jeff Gosztyla
  - David Foldes
  - John Pham
  - Pete Anderson
  - Jimmie Nelson
- External Engineers
  - Ignacio Decia
  - Marzio Cuello
  - Santiago Pacheco
  - Guillermo Baez
  - Simon Coenen
  - Govind Venkatesh
  - Dan Moody
- All the amazing artists that worked on the game

Thank You!

- rob.hall@wbgames.com

References

- Bolz, J. 2009. OpenGL Bindless Extensions. Retrieved from https://developer.download.nvidia.com/opengl/tutorials/bindless_graphics.pdf
- Doghramachi, H. 2020. Lighting Technology of The Last of Us Part II. In SIGGRAPH 2020.
- Hall, R., Hall, C., and Edwards, D. Rendering in Cars 2. In SIGGRAPH Courses, Advances In Real-Time Rendering in 3D Graphics and Games 2011
- Lauritzen, A. 2010. Deferred Rendering for Current and Future Rendering Pipelines, Beyond Programmable Shading. In SIGGRAPH 2010.
- Li, B. 2019. A Scalable Real-Time Many-Shadowed-Light Rendering System. In SIGGRAPH 2019.
- Schneider, A. Nubis: Authoring Real-Time Volumetric Cloudscapes with the Decima Engine. In SIGGRAPH Courses, Advances in Real-Time Rendering in 3D Graphics and Games 2017
- Tatarchuk, N. Artist-Directable Real-Time Rain Rendering in City Environments. In SIGGRAPH Courses, Advanced Real-Time Rendering in 3D Graphics and Games 2006
- Turchyn, P. 2011. GPU Pro 2, ch. Fast Soft Shadows via Adaptive Shadow Maps
- Villeta, J. and Nelson, R. 2023. 'Hogwarts Legacy': An Inside Look at Developing a Cross-Platform Open World Game in Unreal Engine, In Unreal Fest 2023.