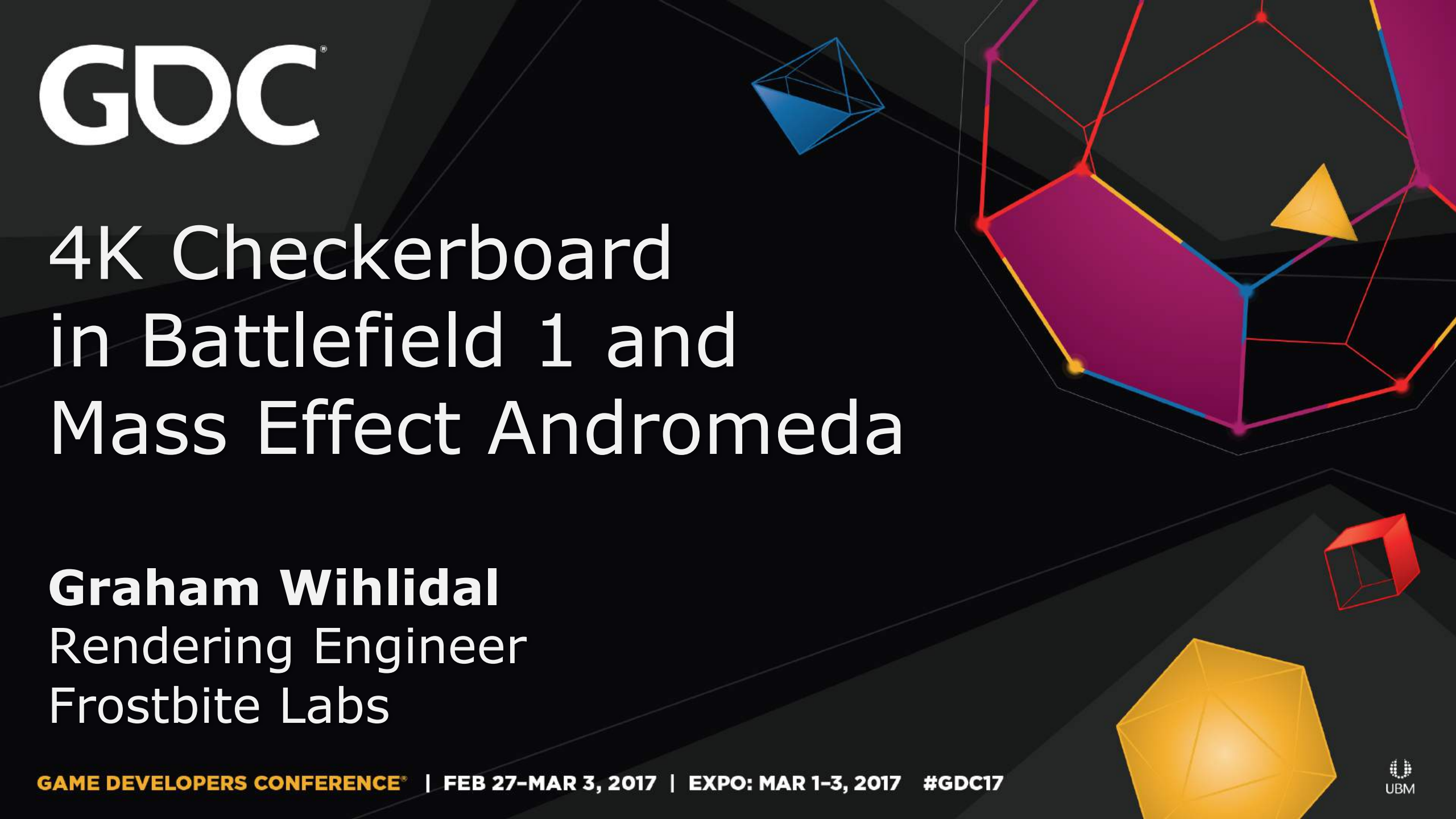


4K Checkerboard in Battlefield 1 and Mass Effect Andromeda

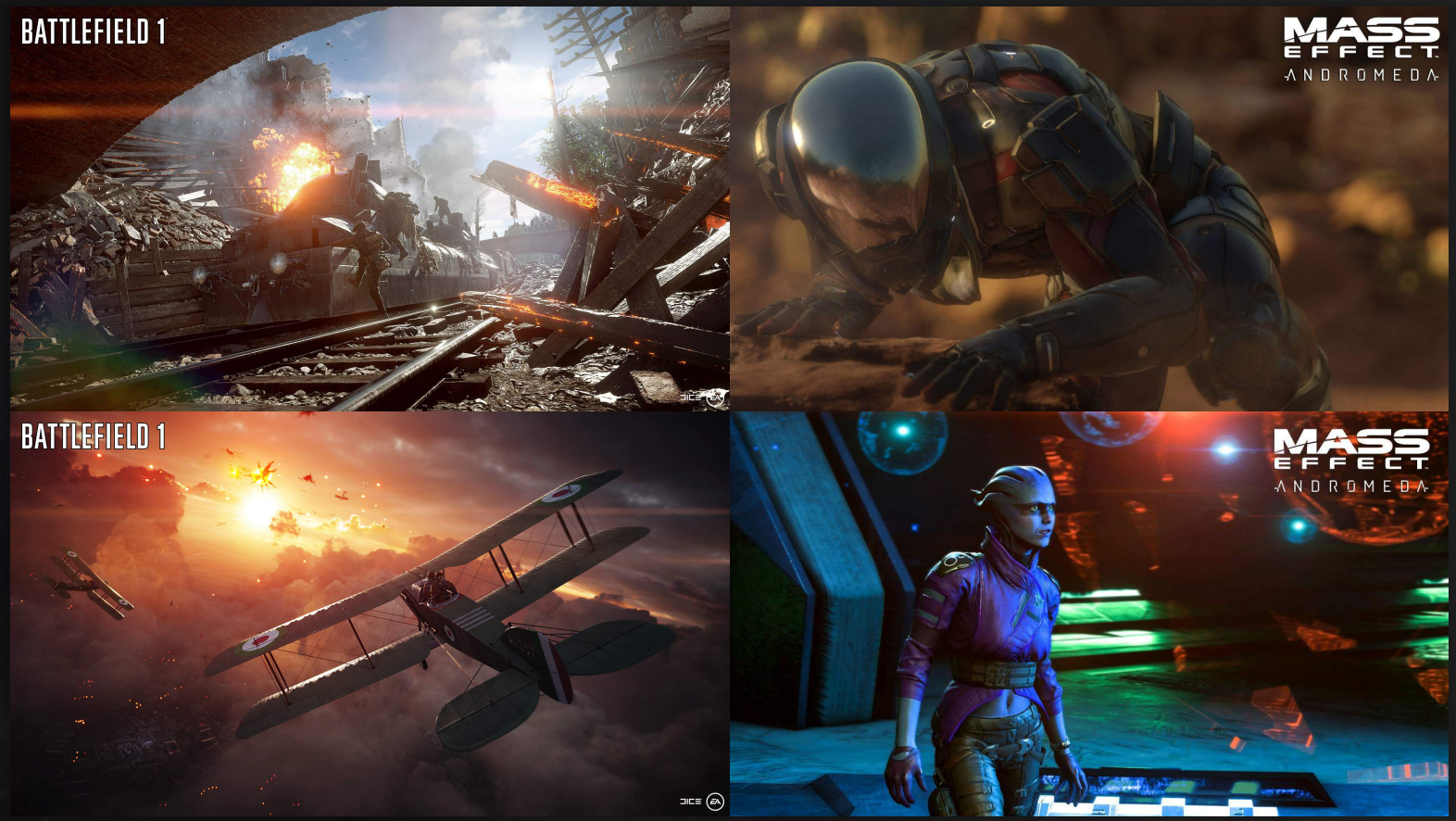
Graham Wihlidal
Rendering Engineer
Frostbite Labs

GAME DEVELOPERS CONFERENCE® | FEB 27–MAR 3, 2017 | EXPO: MAR 1–3, 2017 #GDC17

UBM

Agenda

- Motivation
- Configuration
- Features
- Optimizations
- Post Processing
- Pipeline
- Conclusion




MASS
EFFECT
ANDROMEDA

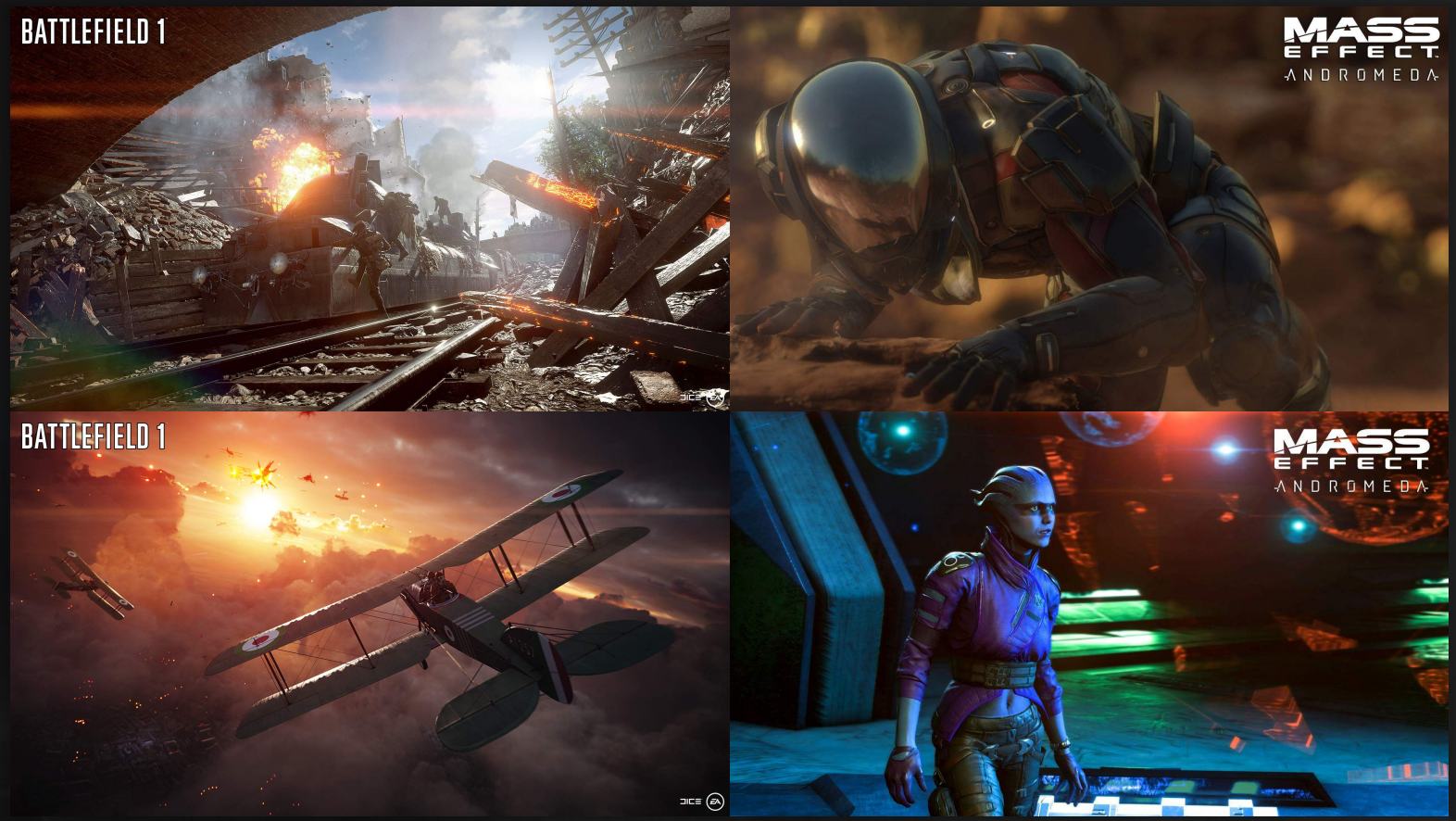

MASS
EFFECT
ANDROMEDA


BATTLEFIELD 1
DICE
EA


BATTLEFIELD 1
DICE
EA

Agenda

- **Motivation**
- Configuration
- Features
- Optimizations
- Post Processing
- Pipeline
- Conclusion



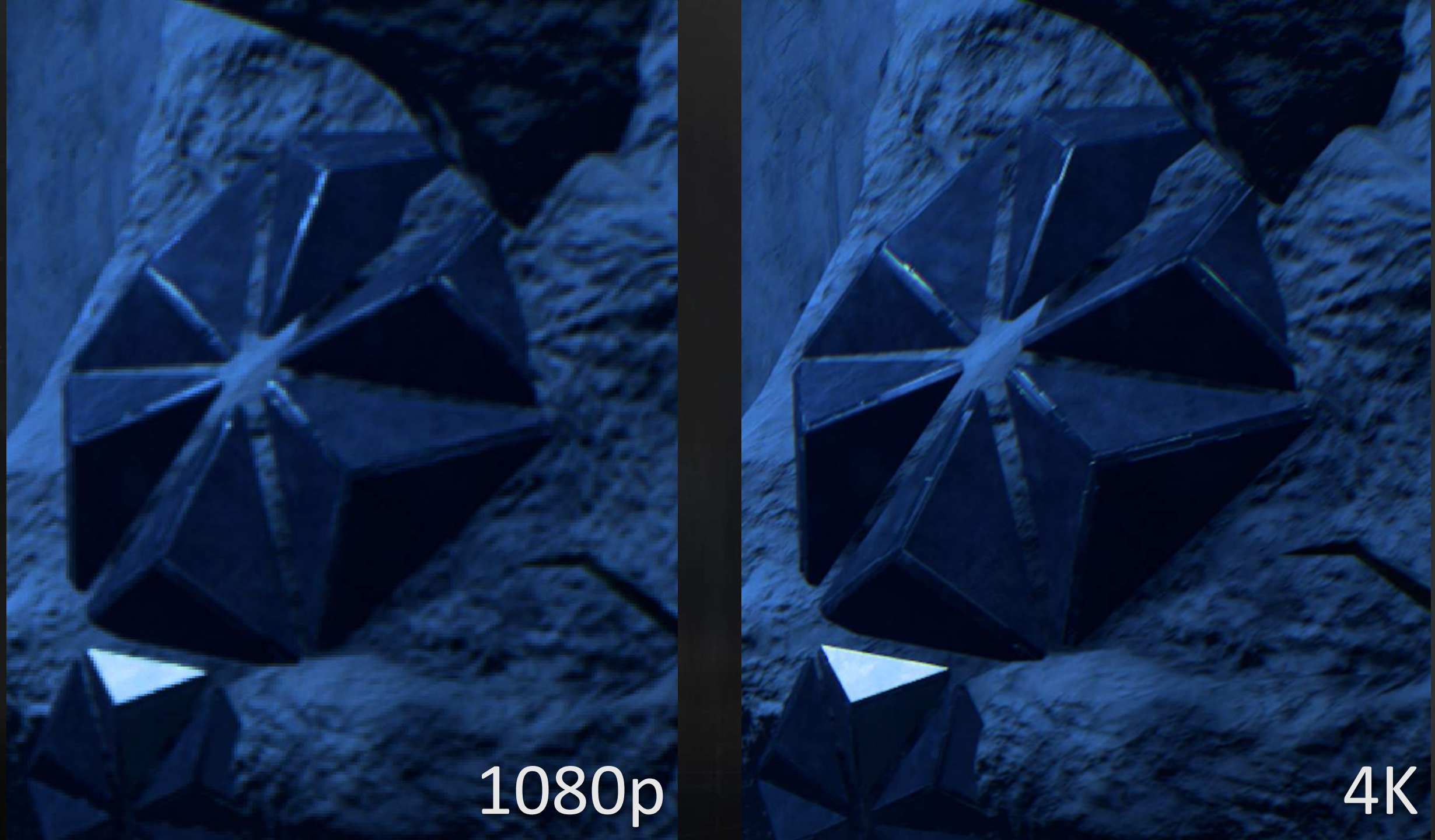
FROSTBITE LABS


1080p
W


1800p


2160p


W

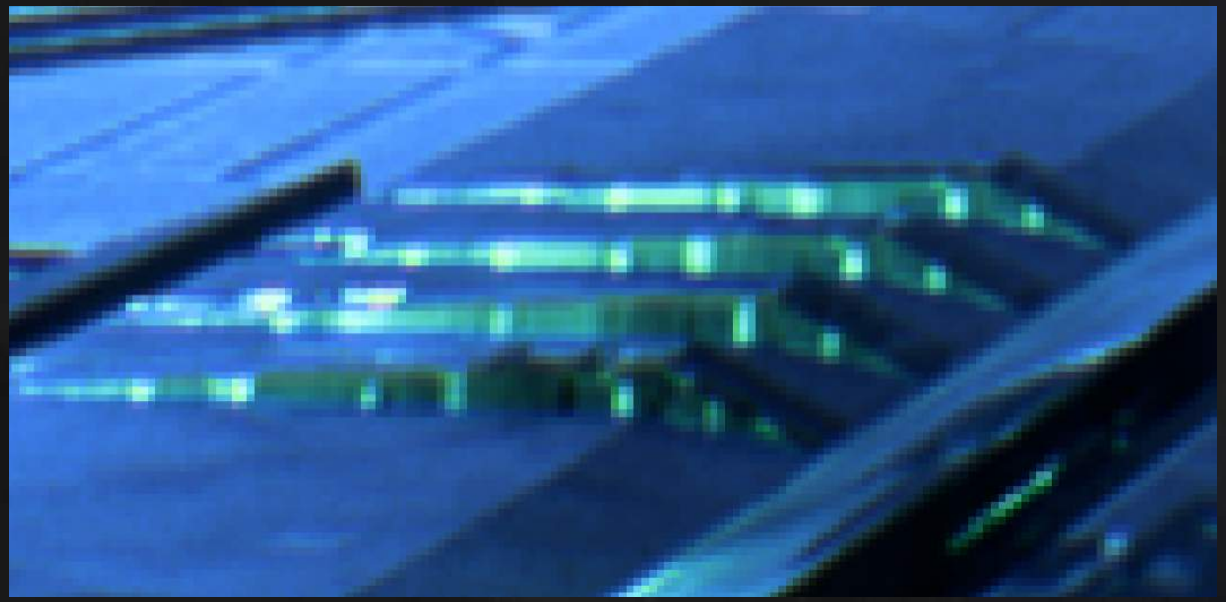

1080p
4K


W

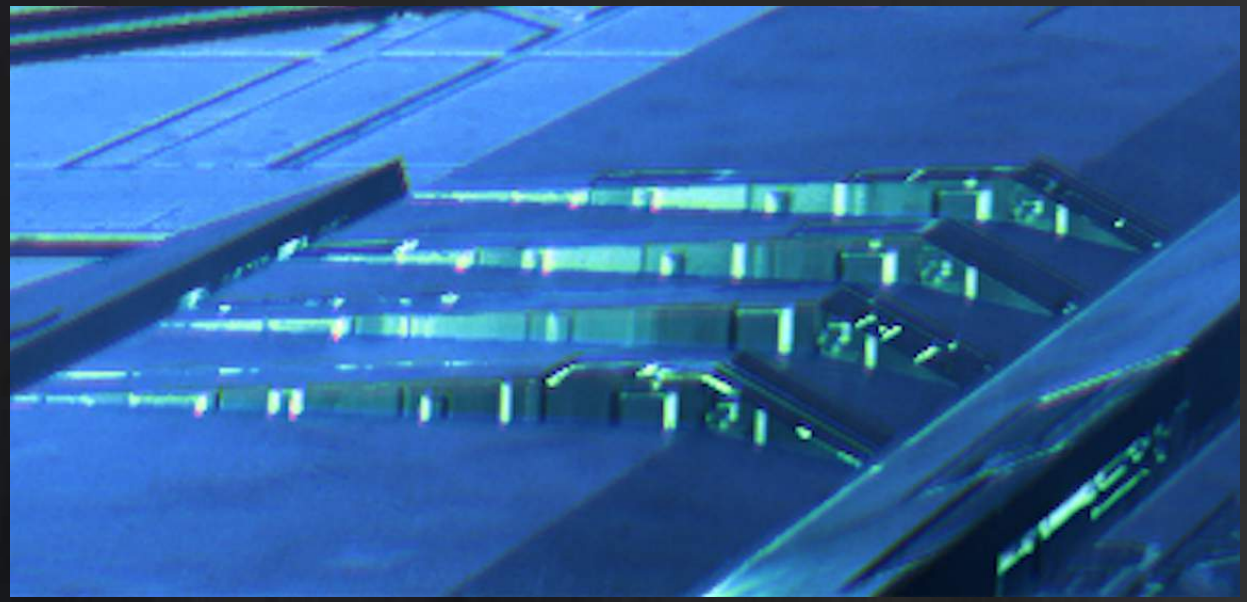

1080p


4K


W



1080p



4K


W

1080p

4K

Motivation

- *Over the past few years…*

- Visual fidelity and complexity has greatly increased

- More memory (order of magnitude)
- Primitive rate (order of magnitude)

- Computational demands (PBR, Dynamic GI, SSR, etc.)

Motivation

- *Over the past few years…*
- Increases in rendering resolution has not
- Majority of current titles are 900p or 1080p
- It's time that 'perceived' resolution got a bump!

Motivation

- Detailed geometry results in aliasing artifacts
  - Lack of high resolution geometry information
  - Image quality is reduced in favor of temporal stability
- Frostbite had deferred MSAA support previously
  - Maintenance nightmare (i.e. shader permutations)
  - Removed to reduce complexity
- Sony added hardware features to PS4™Pro
  - Easier and faster decoupling of geometry and shading rate
  - Less risk for initial adoption

Motivation

- Reduce shading cost in majority of graphics pipeline
  - Shade only a subset of pixels
  - Compute geometry information for all pixels
  - Some information is lost
- 4k sampling rate → adjacent pixels are strongly correlated
  - Assuming they belong to the same surface
- High quality geometry-aware resolve to reconstruct

History

- SIE (WWS ATG): PS3™ Edge MLAA ('09)
  - Used Object IDs to drive MLAA edge detection in SOCOM4 [5]
- SIE (WWS ATG): PS4™ AA Prototype ('13, unreleased)
  - Use EQAA for higher resolution depth
  - Reconstruct full-resolution image, then resolve down
  - Layer further AA techniques on resulting image
- Guerrilla Games: 'Killzone: Shadowfall' ('13)
  - Temporal super-resolution with alternating pixel-column
  - Difference blend operator [3]

(*) PS4™Pro custom hardware features

History

- SIE: PS4™Pro Architecture (’13-15)
  - Alternating Packed Checkerboard Sampling [15]
  - Texture Gradient Adjustment(*)
  - Pixel Shader Invocation Control(*)
  - High-Resolution Object and Primitive ID Buffers(*)
- SIE (WWS ICE/ATG): 4KCB / 4KG Demos (’15)
  - First implementations of these techniques in game titles
  - inFamous: First Light, Knack, Uncharted 4 - on PS4™Pro hardware [16,17]
- Ubisoft: Rainbow Six | Siege (‘15)
  - MSAA checkerboard implementation [1]
- EA Frostbite | Labs : PS4™Pro Support for Frostbite ('16)
  - ‘Battlefield 1’ and ‘Mass Effect Andromeda’

Exploration

- Tried a number of high resolution techniques
  - Variety of resolutions
  - 50” TV at living room distance
- Super-sampling \ native 4K
  - Looks great!
  - Perf timers don’t…
- Reduced resolution to 1800p
  - Still too expensive @ 60Hz

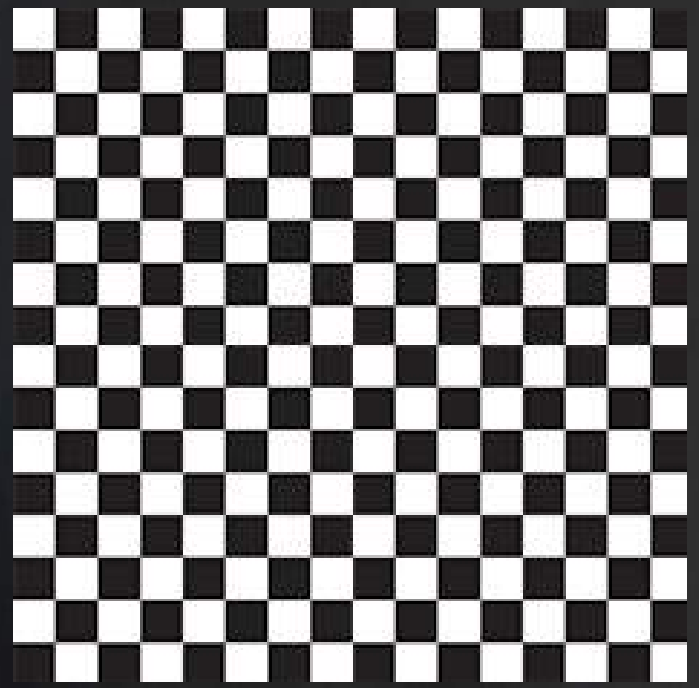
1440		720
1800	/ 2 =	900
2160		1080


0 35:01 0
A B C D E F
AVIS IMPORTANT!
COURRIER DU SOIR
TAIZE
32 / 96
11
2160p
29.07ms
PS4™Pro


0 35:01 0
A B C D E F
AVIS IMPORTANT!
COURRIER DU SOIR
TAIZE
32 / 96
1
1800p
21.07ms
PS4™Pro

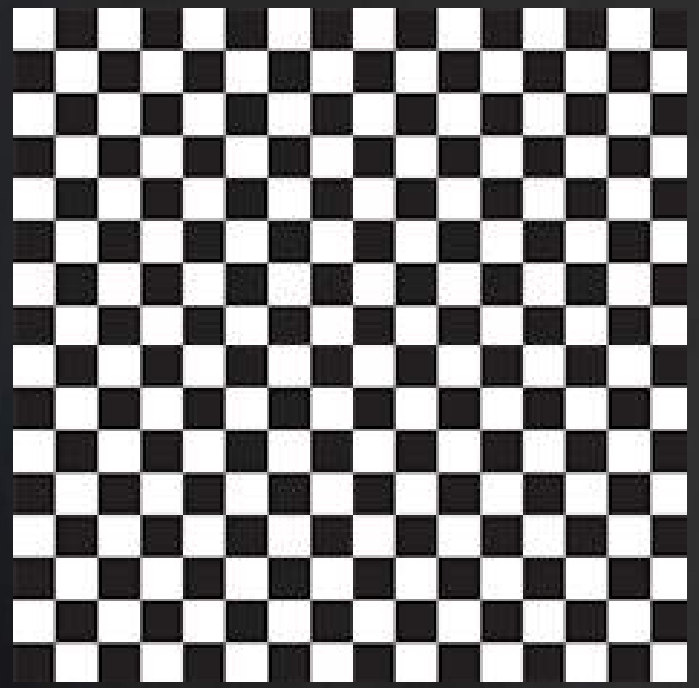
Exploration

- Variable shading rate is popular
- Checkerboard is a practical idea
- Greatly increase resolution
- Without much performance cost



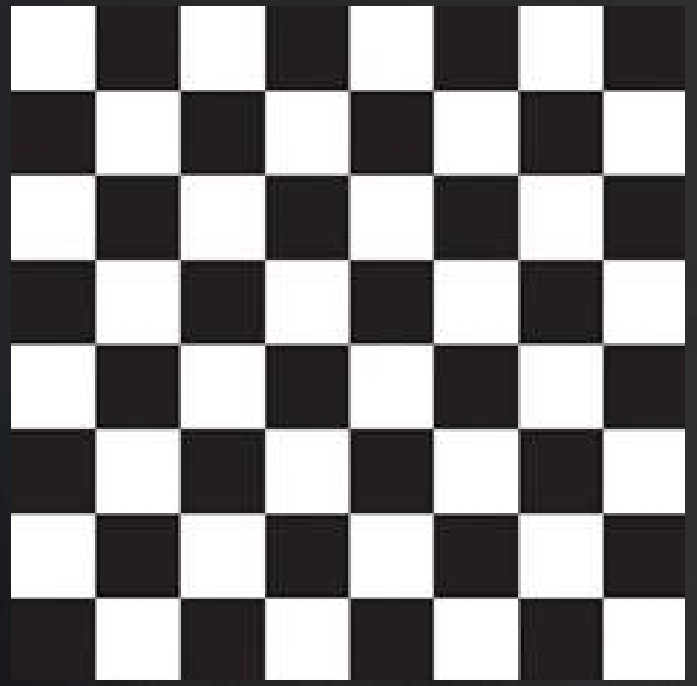
Exploration

- Stencil out 1x1 blocks
- Using **stencil is terrible,** no performance gain
- GPU shades in 2x2 quads
- Same cost as rendering native 4K
  - Throw away half of the work
  - 50% inactive lanes



Exploration

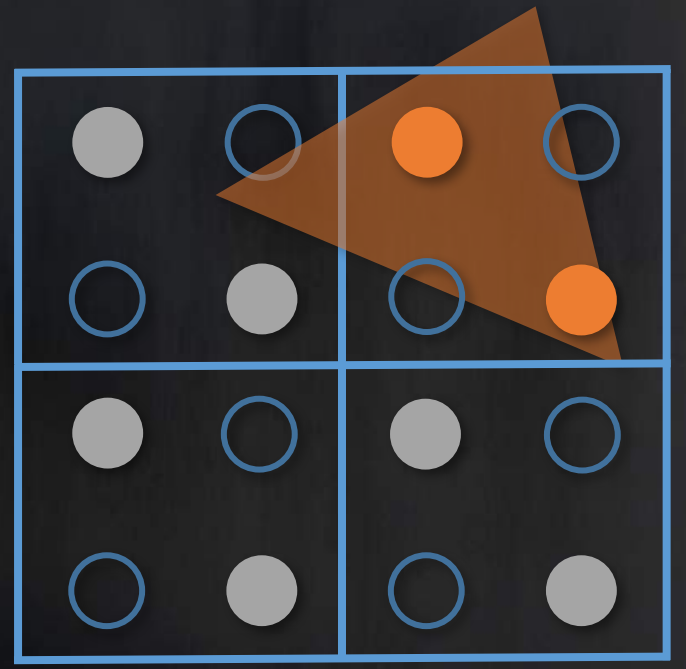
- Stencil out 2x2 blocks
- Sampling distribution is bad; less correlation
  - Great coverage in a 2x2 block, then a huge hole
  - Blurry dilated color bits every 2nd quad
- End up blurring even more to solve
  - What's the point?




YEAAAAAAAAAAH
HOW ABOUT NO

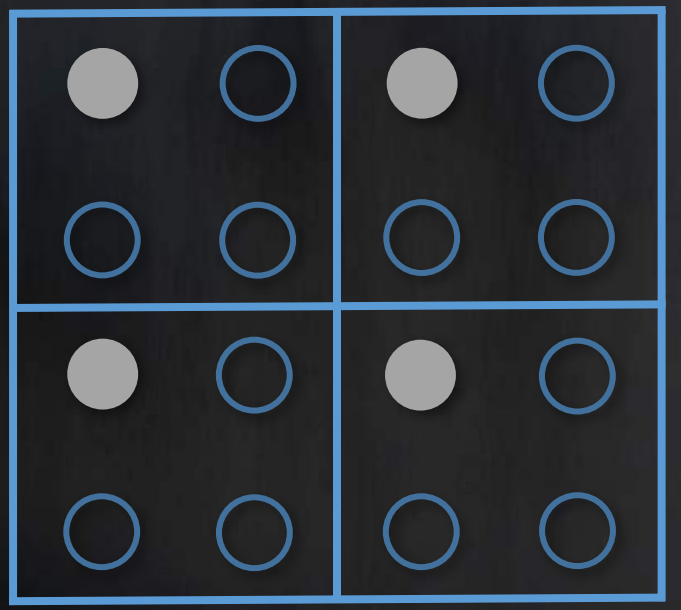
Exploration

- 2x color and 4x depth checkerboard
  - Available on any MSAA platform
  - Requires all shaders to load from MS textures (lots of changes)
  - Sub-optimal – more on this later
- Alternatively, 2x color and 2x depth [1]



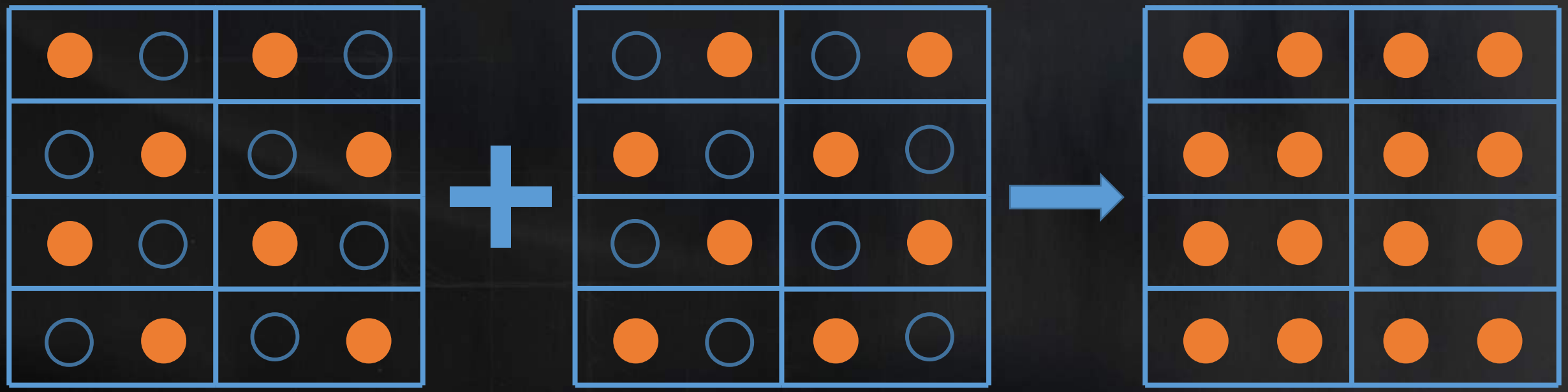
Exploration

- 1x color and 4x depth geometry resolve (4K Geometry)
  - 1080p to 4K
  - Was a good idea, and is similar to SRAA [14]
  - Single Pass + IDs + custom reconstruction
  - Quality wasn't high enough
- Abandoned early in favor of 4K CB
  - Comparable implementation cost
  - More research would improve concept



Exploration

- Settled on "packed checkerboard" technique
  - Started with PS4™Pro reference implementation
  - Customized + optimized further, and incorporated our own TAA




PETTICOAT LANE
2016 PlayStation.Blog
BEST USE OF PS4 PRO


Spatial + Temporal Resolve


0 35:01 0
A B C D E F
1800p CB
15.99ms
TAIZE
AVIS IMPORTANT!
COURRIER DU SOIR
32 / 96
PS4™Pro

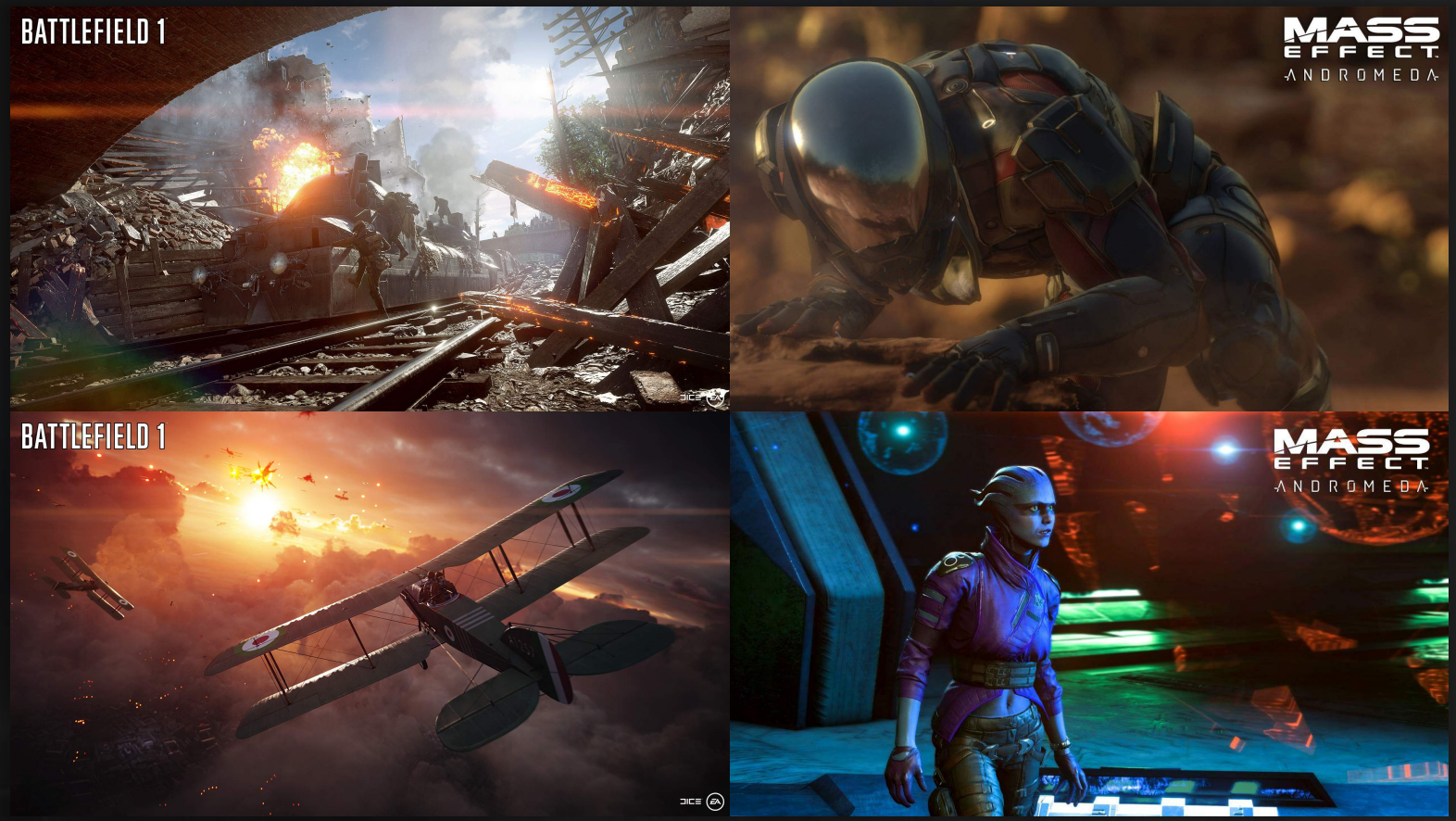
PS4™Pro 1800p 21.07ms

PS4™Pro

1800p CB 15.99ms

Agenda

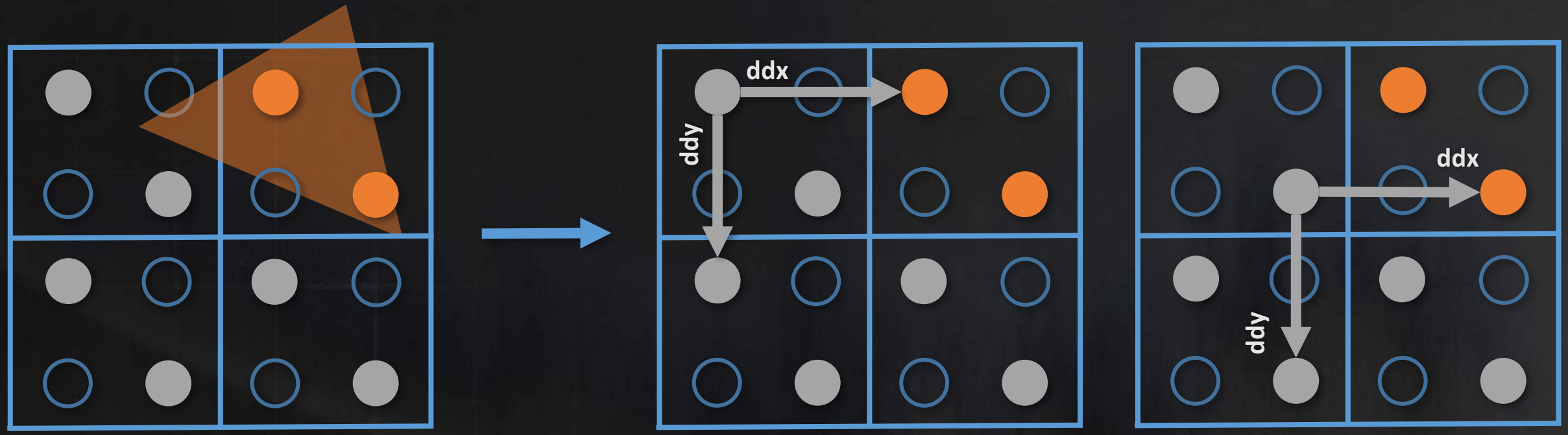
- Motivation
- **Configuration**
- Features
- Optimizations
- Post Processing
- Pipeline
- Conclusion



EQAA – What is it?

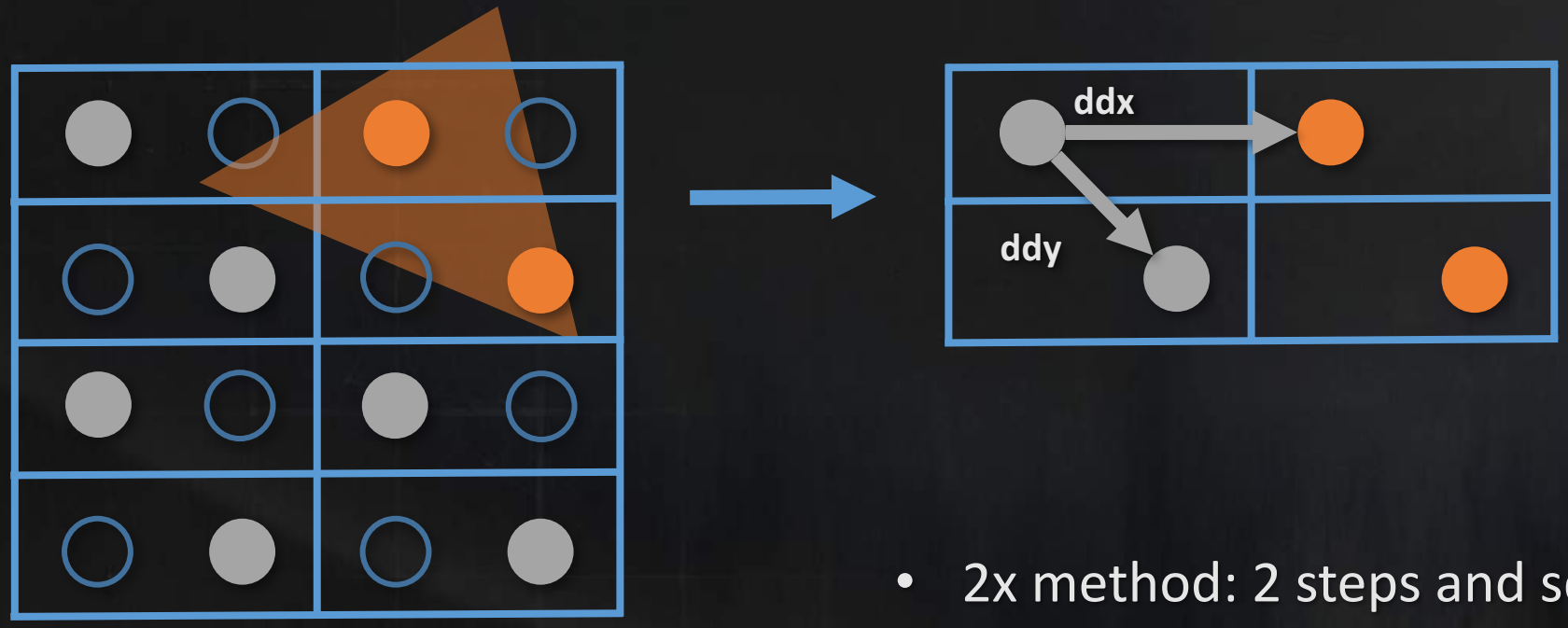
- EQAA [7] (AMD) is a superset of MSAA
- Possible to store fewer color fragments than depth fragments
  - Color <= ID <= Depth
- 4K checkerboard exploits this configuration
  - 1x color fragment
  - 2x ID fragments
  - 2x depth fragments

Shading Quads – 2x Color : 4x Depth



8 Shaded, 2 Stored

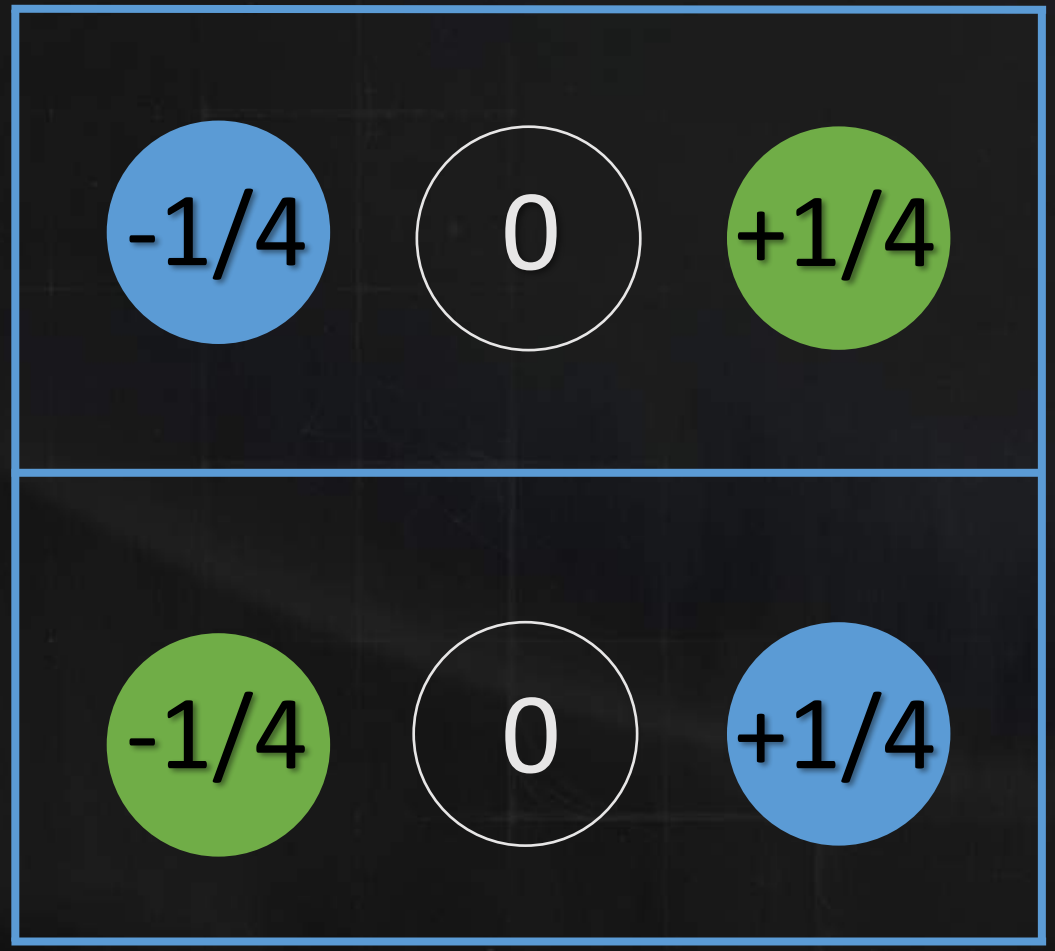
Shading Quads – 1x Color : 2x Depth

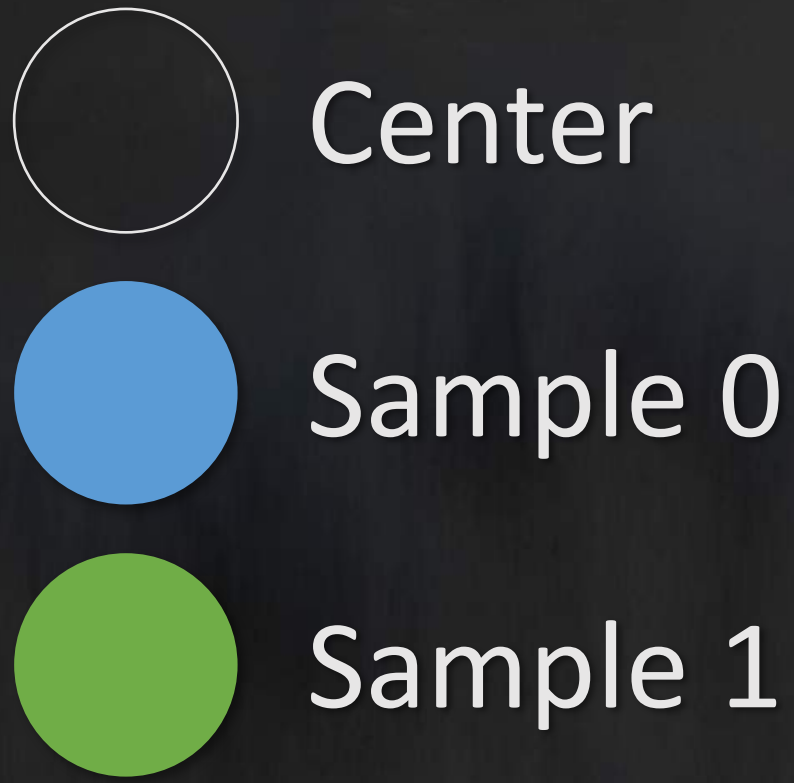


- 2x method: 2 steps and sqrt(2) steps.
- 4x method: 2 steps and sqrt(8) steps.

4 Shaded, 2 Stored

EQAA Checkerboard Layout

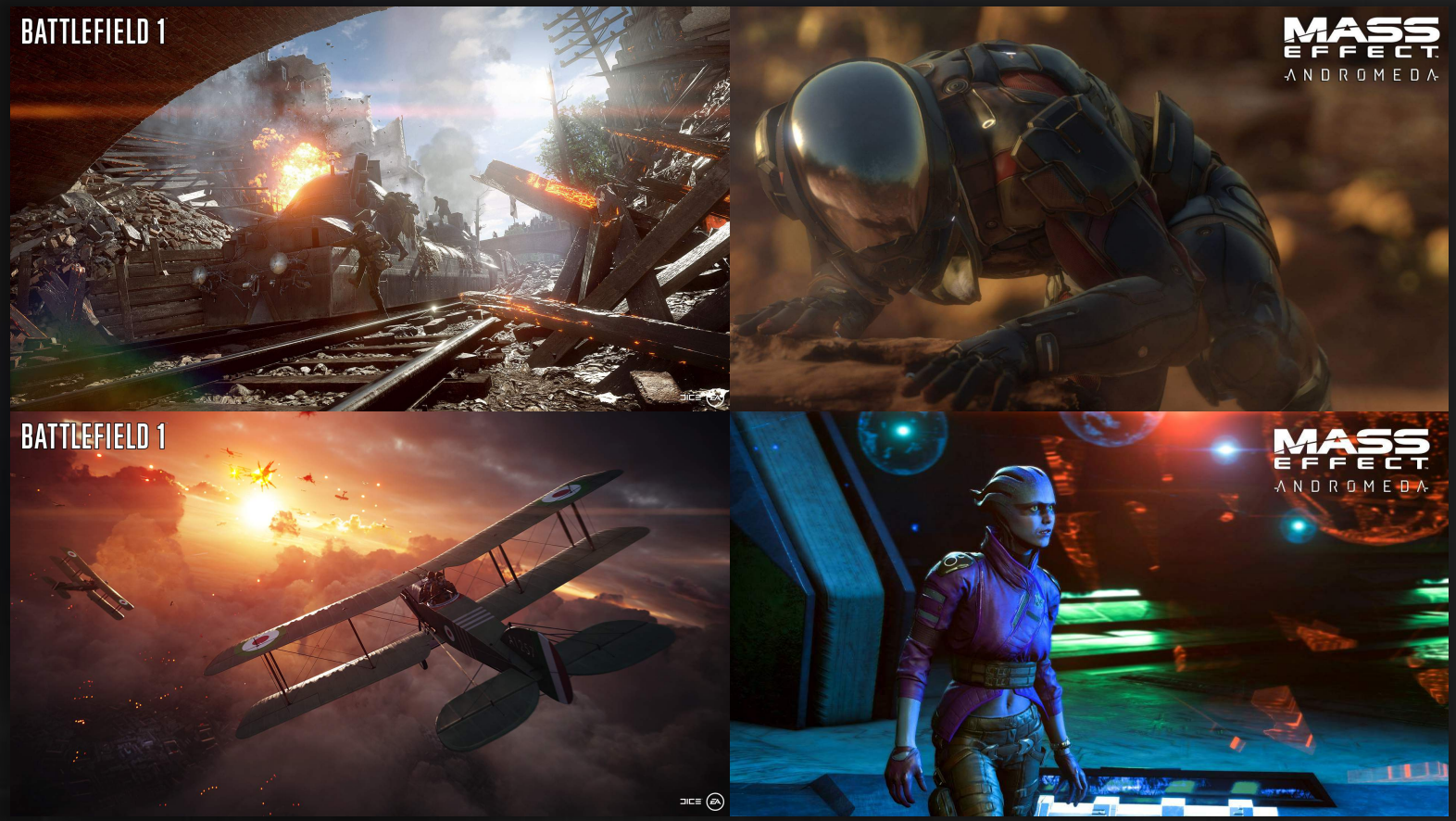




Note: Positions are specified per-quad

Agenda

- Motivation
- Configuration
- **Features**
- Optimizations
- Post Processing
- Pipeline
- Conclusion



Object and Primitive ID Buffer

- Generated at a resolution higher than shading resolution
- Each visible geometry sample
  - Identifier stored in image buffer
  - Uniquely identifies object and primitive
- Instanced draws provide a separate object ID per instance
  - Draws take pointer to an array of object IDs, one for each instance
- Tessellated draws are given a primitive ID per input patch

Object and Primitive ID Buffer

- Written during depth-only passes, or main scene render
- Depth/ID-only
  - Consumes less memory bandwidth than shading passes
  - Performed without shader involvement (on PS4™Pro)
  - Allows asynchronous compute jobs running in parallel
- CPU and GPU overhead
  - We don't use full pre-pass
  - Alpha tested objects + dominant occluders
- Main shading
  - Easier to integrate if no existing depth-only pass
  - Competes with memory bandwidth
  - ID-only samples don't pay for shading

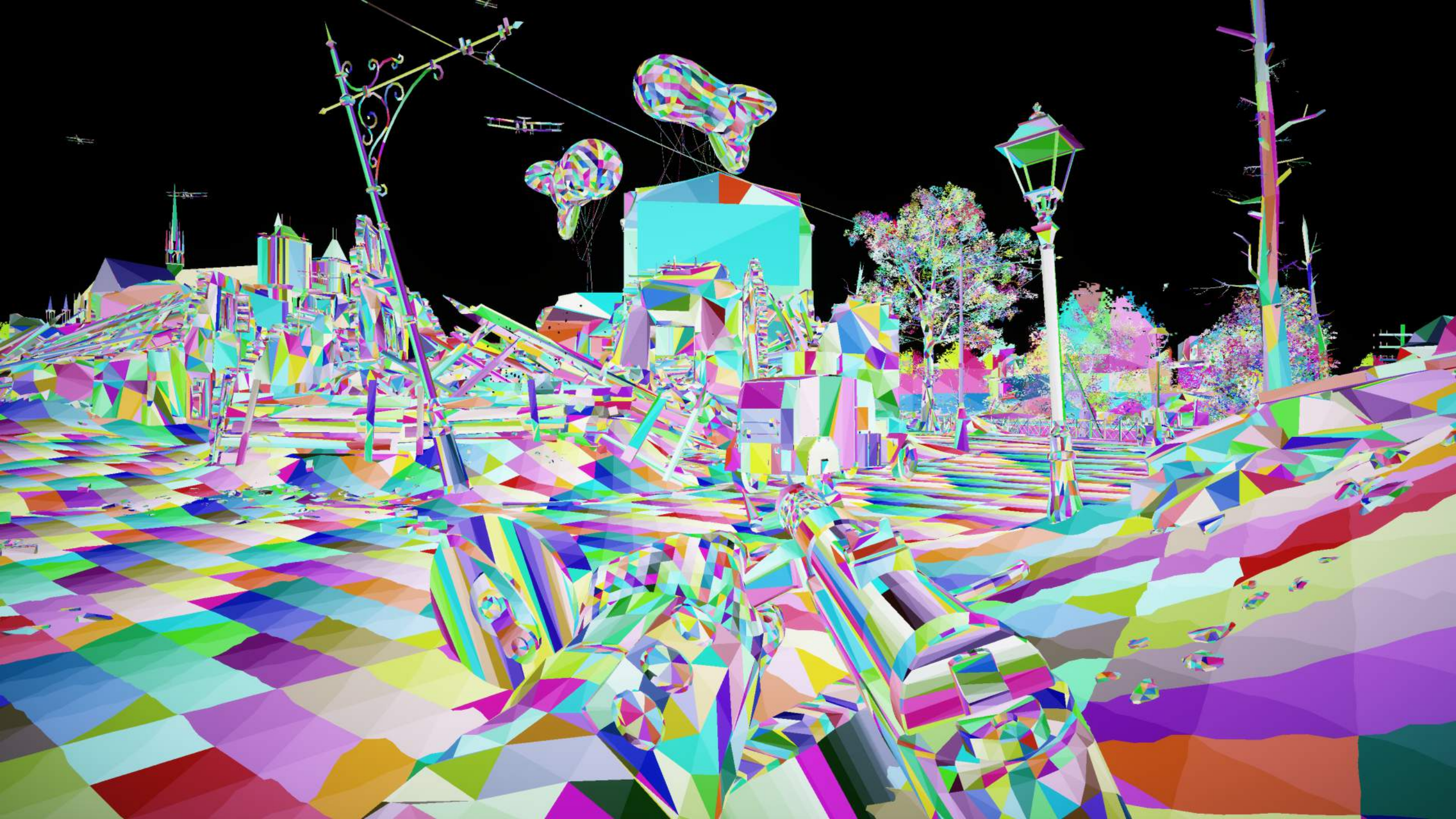
Object and Primitive ID Buffer

- 32-bit, 16-bit, or 8-bit
- We use 31-bits wide
  - MSB ignored by hardware
  - 14-bit object ID + 17-bit primitive ID
- Primitive ID reset to 0 at the end of each instance or draw

Object and Primitive ID Buffer

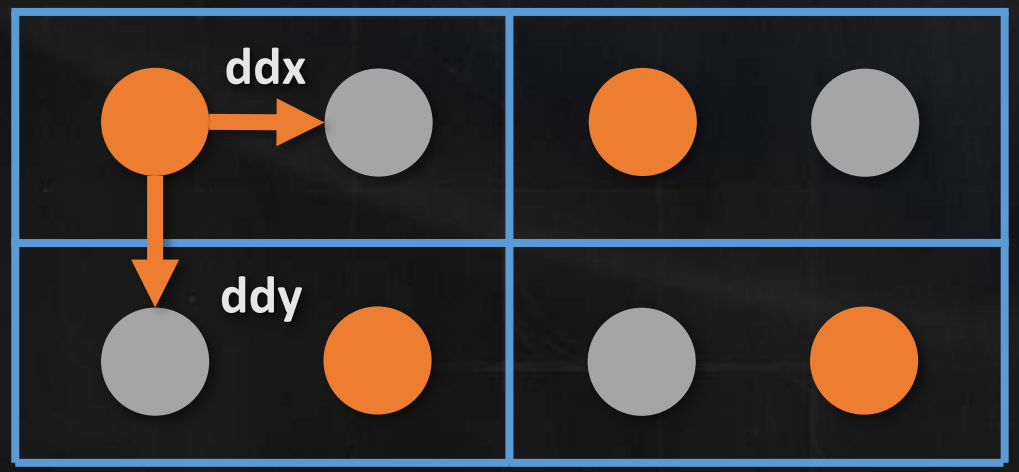
- Color target bound as 8th MRT
  - Uses CB's data paths and color target tile modes
  - Safe to use null pixel shader
  - Not treated as a pixel shader target during ID propagation
- IDs can also be exported from vertex shader
  - Could reduce number of VS wavefronts active on chip
- Many other interesting use cases
  - Not covered in this presentation ☺
- Can be used in resolve to improve quality of final image [5]


CHARCUTERIE

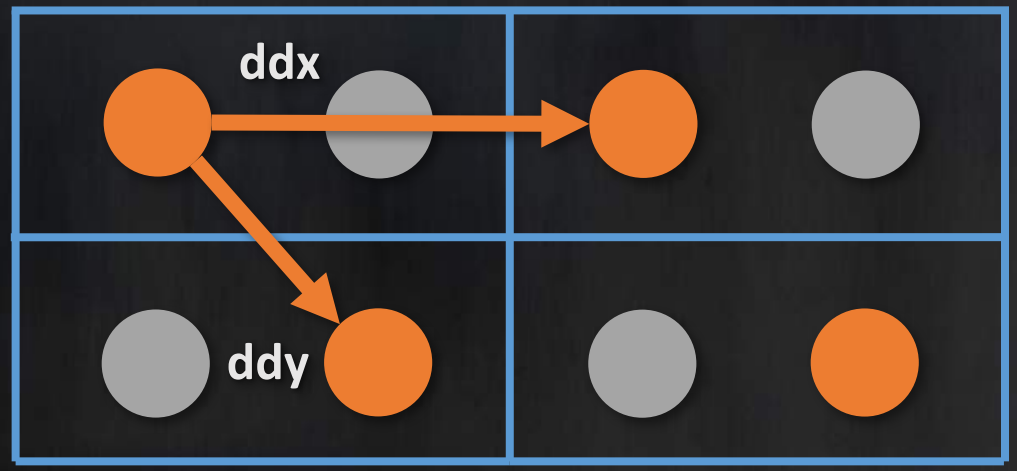


Gradient Adjust

- Horizontal gradient is twice the expected value
- Vertical gradient is stretched and rotated 45˚
- Resembles bad anisotropic filtering



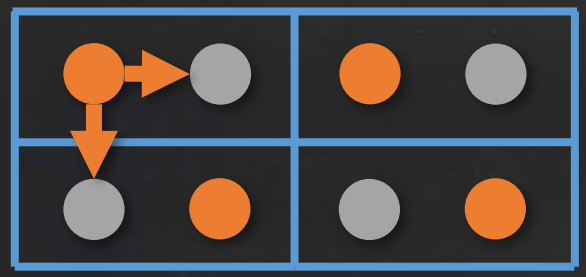
Expected Gradient



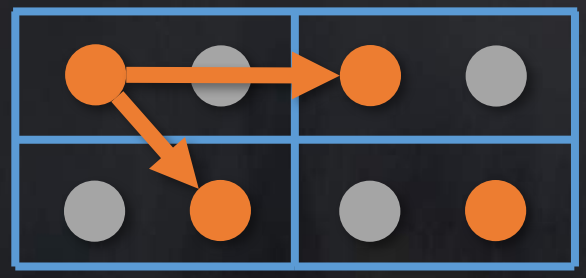
Real Gradient

Gradient Adjust

- Need to apply non-uniform rescale
  - Rectangular pixels
- LOD Bias
  - Doesn't work – scales uniformly
- Use SampleGrad fetch:



Expected Gradient



Real Gradient

```
tex.SampleGrad(samp, input.uv0, duvdx_adj, duvdy_adj);
```

Gradient Adjust

$$\begin{vmatrix} d[uvw]'_{dx} \\ d[uvw]'_{dy} \end{vmatrix} = \begin{vmatrix} factor_{00} & factor_{01} \\ factor_{10} & factor_{11} \end{vmatrix} * \begin{vmatrix} d[uvw]_{dx} \\ d[uvw]_{dy} \end{vmatrix}$$

Identity (No Adjustment):

$$\begin{vmatrix} d[uvw]'_{dx} \\ d[uvw]'_{dy} \end{vmatrix} = \begin{vmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{vmatrix} * \begin{vmatrix} d[uvw]_{dx} \\ d[uvw]_{dy} \end{vmatrix}$$

Gradient Adjust

Frame N + 0:

$$\begin{vmatrix} d[uvw]'_{dx} \\ d[uvw]'_{dy} \end{vmatrix} = \begin{vmatrix} 0.5 & 0.0 \\ -0.5 & 1.0 \end{vmatrix} * \begin{vmatrix} d[uvw]_{dx} \\ d[uvw]_{dy} \end{vmatrix}$$

Frame N + 1:

$$\begin{vmatrix} d[uvw]'_{dx} \\ d[uvw]'_{dy} \end{vmatrix} = \begin{vmatrix} 0.5 & 0.0 \\ 0.5 & 1.0 \end{vmatrix} * \begin{vmatrix} d[uvw]_{dx} \\ d[uvw]_{dy} \end{vmatrix}$$

Gradient Adjust

- Manual gradient correction with shader ALU
  - ~10% extra cost in main shading
- SampleGrad issue is more expensive than normal fetch (in cy)
  - No change in bandwidth / latency
  - Fetch and filter are unaffected
- Increased register pressure
  - Keep derivatives around
  - Need to be careful!

Gradient Adjust

- However…
- Only shade half the pixels
- Not all instructions in the shader are a fetch
- Overall win

Gradient Adjust

- PS4™Pro has special hardware and compiler support!
- Hardware can perform this in the texture unit
- Affine transform stored in texture unit
- Just turn it on ☺ ALU portion is free

Gradient Adjust

SUCKER PUNCH

Gradient Adjust

- Most cases can be automatically adjusted
- Explicit gradients (ddx/ddy) need to be manually corrected
  - Custom filtering
  - Virtual texturing
  - etc.


Broken Gradients
Programmer Art ☺


Adjusted Gradients
Programmer Art ☺

Images courtesy of SIE [15]

Barycentric Evaluation

✘ Pixel Positions

✓ Sample Positions

Images courtesy of SIE [15]

Barycentric Evaluation

- Pixel Positions

Images courtesy of SIE [15]

Barycentric Evaluation

✓ Sample Positions

Alpha Unrolling

- Alpha test computes depth/coverage inside pixel shader
  - Instead of relying on scan-converter, like opaque
- By default, pixel shader runs at pixel rate
- All samples of a pixel share output of single shader invocation
- IDs at shading rate instead of full rate
- Serious problem with hole reconstruction

Alpha Unrolling

- Solution: Run samples at coverage rate!
  - Generate full resolution depth and IDs
- Each pixel quad is unrolled
  - Shading quad created per sample
- Large increase in pixel shader work
  - Important to switch off when not needed

Alpha Unrolling

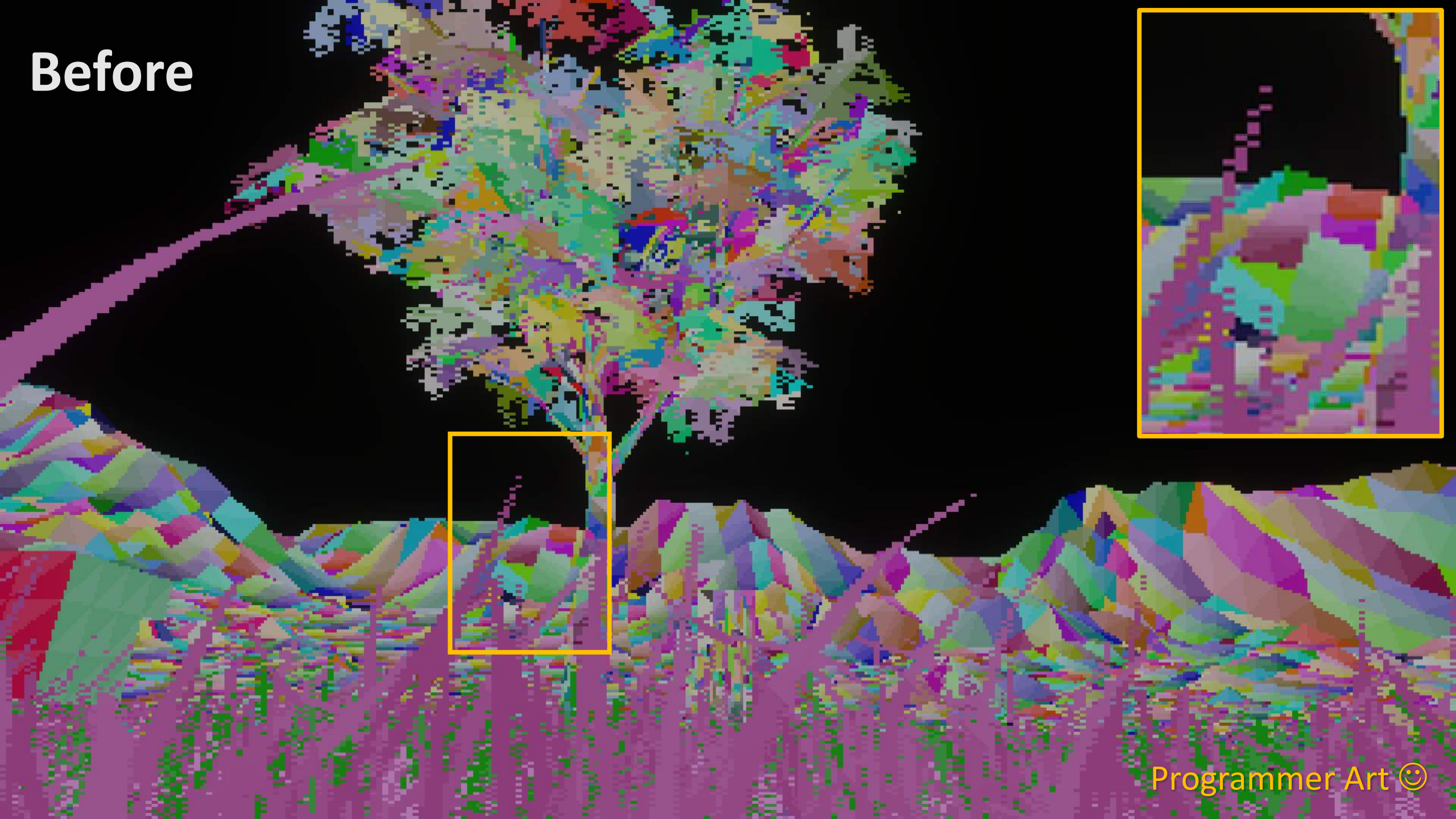
- Run minimal pass to calculate coverage
  - Computes coverage/depth (Clip / Depth Write)
  - Coverage rate (2x)
- Run expensive shading pass
  - Computes color values (Depth Equals)
  - Benefits from maximal hidden surface removal
  - Pixel rate (1x)

Alpha Unrolling

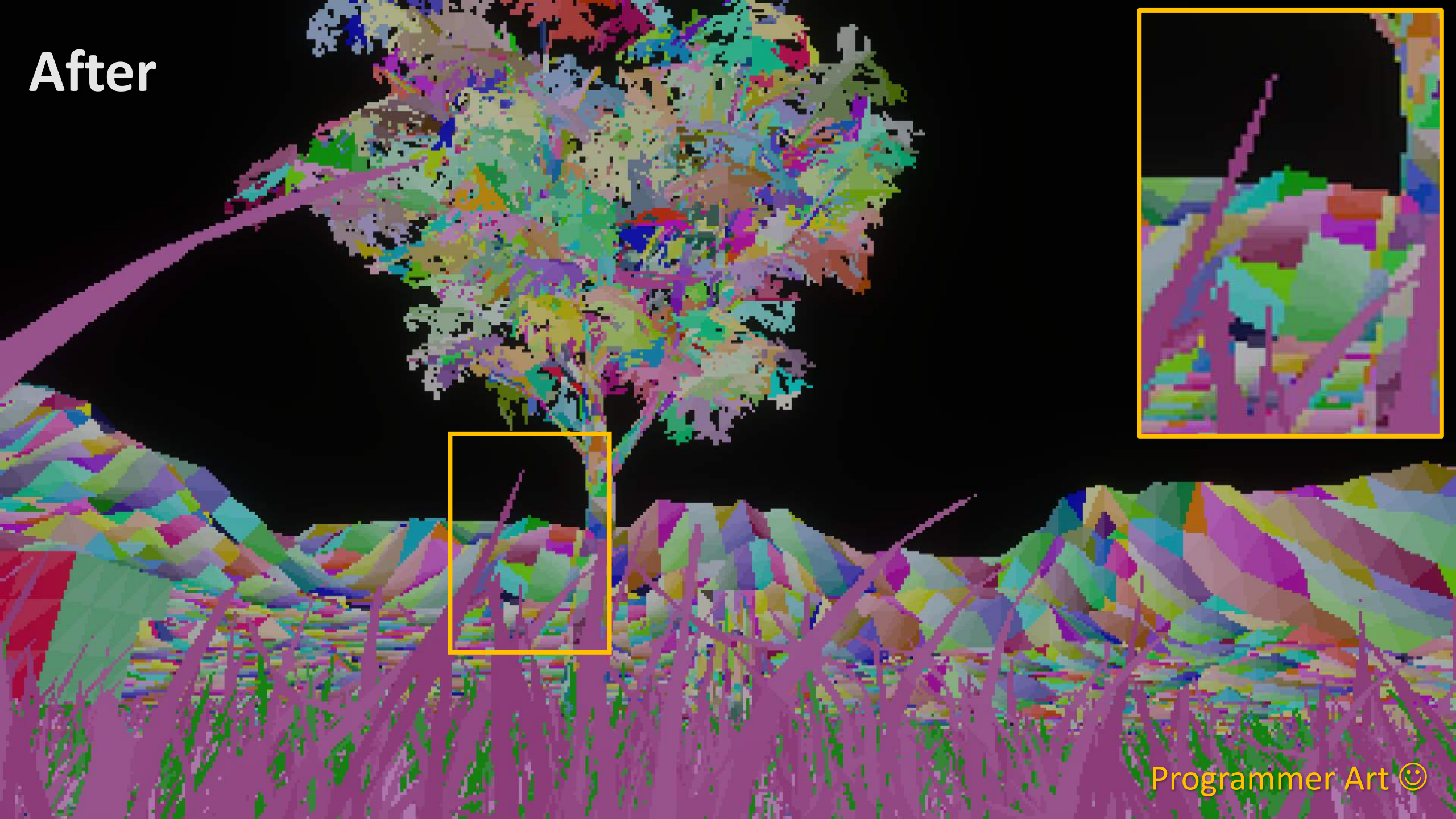
- Positions need to be invariant!
- Positions written by coverage and shading need to match
- Subtle differences in computation can lead to z-fighting
- Disable “fast math” for everything that goes into the position


Before
Programmer Art ☺


After
Programmer Art ☺


Before
Programmer Art ☺

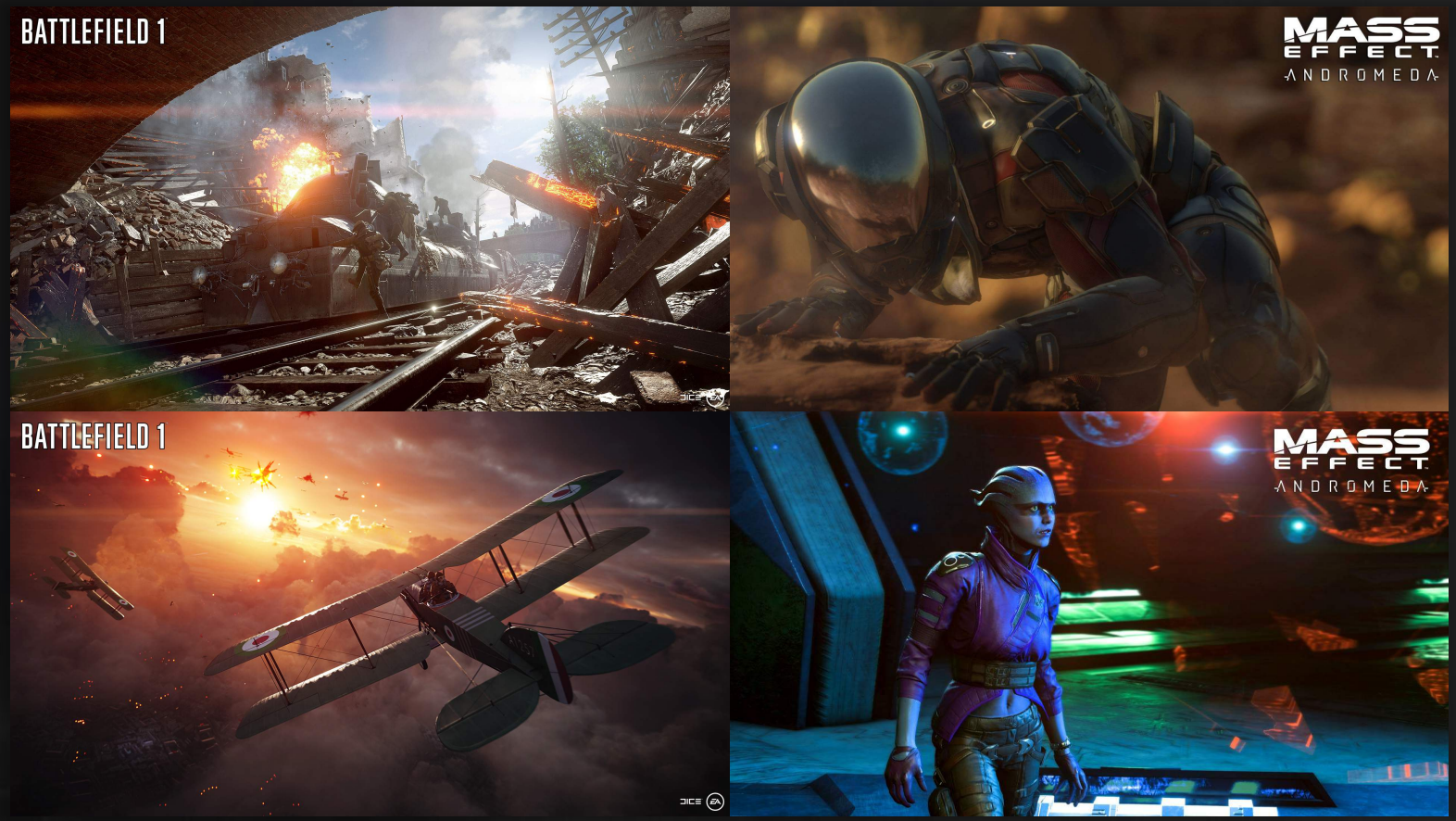
After



Programmer Art ☺

Agenda

- Motivation
- Configuration
- Features
- **Optimizations**
- Post Processing
- Pipeline
- Conclusion



PS Invoke

- Every sample position…
  - If covered by a triangle, can trigger pixel shading
  - Checkerboard uses 1x color and 2x depth/IDs
  - Half of the shading does not contribute to final image
- How can we prevent over-shading?
- Pixel Shader Invocation Control (PS Invoke)
  - Can make samples "non-shading"
  - Get the shading work back to what it would be with a single sample

FP16

- PS4™Pro has support for FP16 GCN instructions
- Used throughout checkerboard resolve shader
- 30% performance improvement

EQAA Depth Resolve

- Many passes do not require higher-resolution depth and IDs
  - Don't waste depth block (DB) bandwidth! Can be ½ instead
- Transparent objects + post-processing are candidates
  - Do not write depth
  - Read from single-sample!
- Resolve to single-sample depth and stencil
  - Occurs after partial pre-pass + gbuffer laydown

EQAA Depth Resolve

- Simple approach is to copy depth with compute shader
  - Requires depth\HTILE decompression ☹
  - Slow!
- Use color block (CB) trick to resolve without decompression!
- AMD Evergreen Acceleration document [8] describes:
  - CB copy of depth to color target
  - No decompression needed
  - Great! But we want a usable depth surface in the end, not color
    - Buckle up.... ☺

EQAA Depth Resolve

- Dummy shader that writes R32F 0.0f
  - exp mrt0, 0.0, off, off, off vm done
- Alias the destination depth target as a color target
  - 2d non-displayable thin and 1xAA depth micro tiling are the same
- Set DEPTH_COPY bit on DB_RENDER_CONTROL
  - PS puts a dummy value on mrt0.x
  - DB replaces it later
- The CB writes the depth to the destination Z surface
  - Without HTILE!

EQAA Depth Resolve

- Stencil is done in a similar manner
  - Set STENCIL_COPY bit in DB_RENDER_CONTROL
- CB writes stencil to G channel
- Destination is a color target aliasing stencil
  - Only R channel present
- Adjust CB color info register to swizzle
  - G to R on write
  - As per COMP_SWAP [4] to configure as SWAP_ALT

EQAA Depth Resolve

```
// Make sure the compiler does not export and clamp to 16 bit depth


float2 psMain() : SV_Target
{
    // Shader will export dummy PS value to MRT
    // DB will switch R to depth value (relies on DB_RENDER_CONTROL.DEPTH_COPY=1)
    // DB will switch G to depth value (relies on DB_RENDER_CONTROL.STENCIL_COPY=1) - then swizzle YX00
    // R=Depth, G=Stencil
    return float2(0.0f, 0.0f);
}
```


0 44:26 0
1,000 SCORE TO WIN
A B C D E F
32 / 96
11
cam transition


0 44:26 0
1,000 SCORE TO WIN
A B C D E F
32 / 96
11
cam transition

EQAA Depth Resolve

- Original HTILE has correct ranges
  - but has compressed ZMask codes
- Patch up the HTILE by copying it
  - Fast compute shader! (6µs)
  - Force ZMask bits to 0xF (expanded)
  - Store patched meta data
- HTILE acceleration works on the 1xAA destination
  - Any further writes will compress as expected

EQAA Depth Resolve

- HTILE copy:

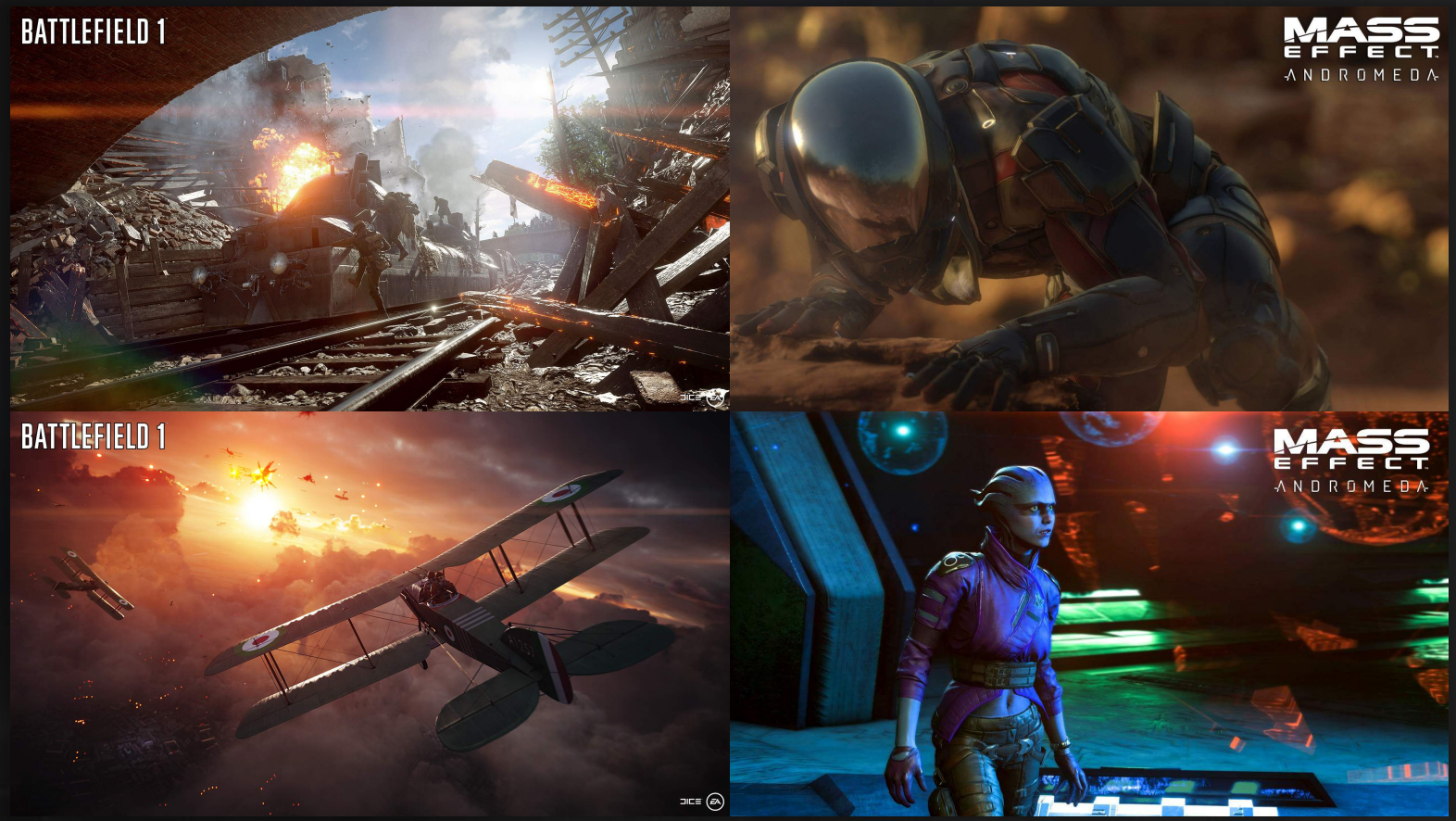
```
Buffer<uint4> g_htileSource      : register(t0);
RWBuffer<uint4> g_htileTarget    : register(u0);

// 16200 threads for 1920x2160

[numthreads(64, 1, 1)]
void main(uint3 threadId : SV_DispatchThreadID)
{
    uint4 htileValues = g_htileSource[threadId.x];

    // Mark 4 tiles as "expanded"
    htileValues |= 0xF; // 3:0 ZMask

    g_htileTarget[threadId.x] = htileValues;
}
```

EQAA Depth Resolve

- This saved us 1ms over basic copy!
- Leaves the source depth and stencil fully compressed
  - DBs and CBs do the work
    - Which understand compression
  - Instead of shader cores doing it
    - Would require decompression
- Technique is completely bandwidth-bound ☺
  - ~0.1ms for depth, ~0.1ms for stencil

Agenda

- Motivation
- Configuration
- Features
- Optimizations
- **Post Processing**
- Pipeline
- Conclusion



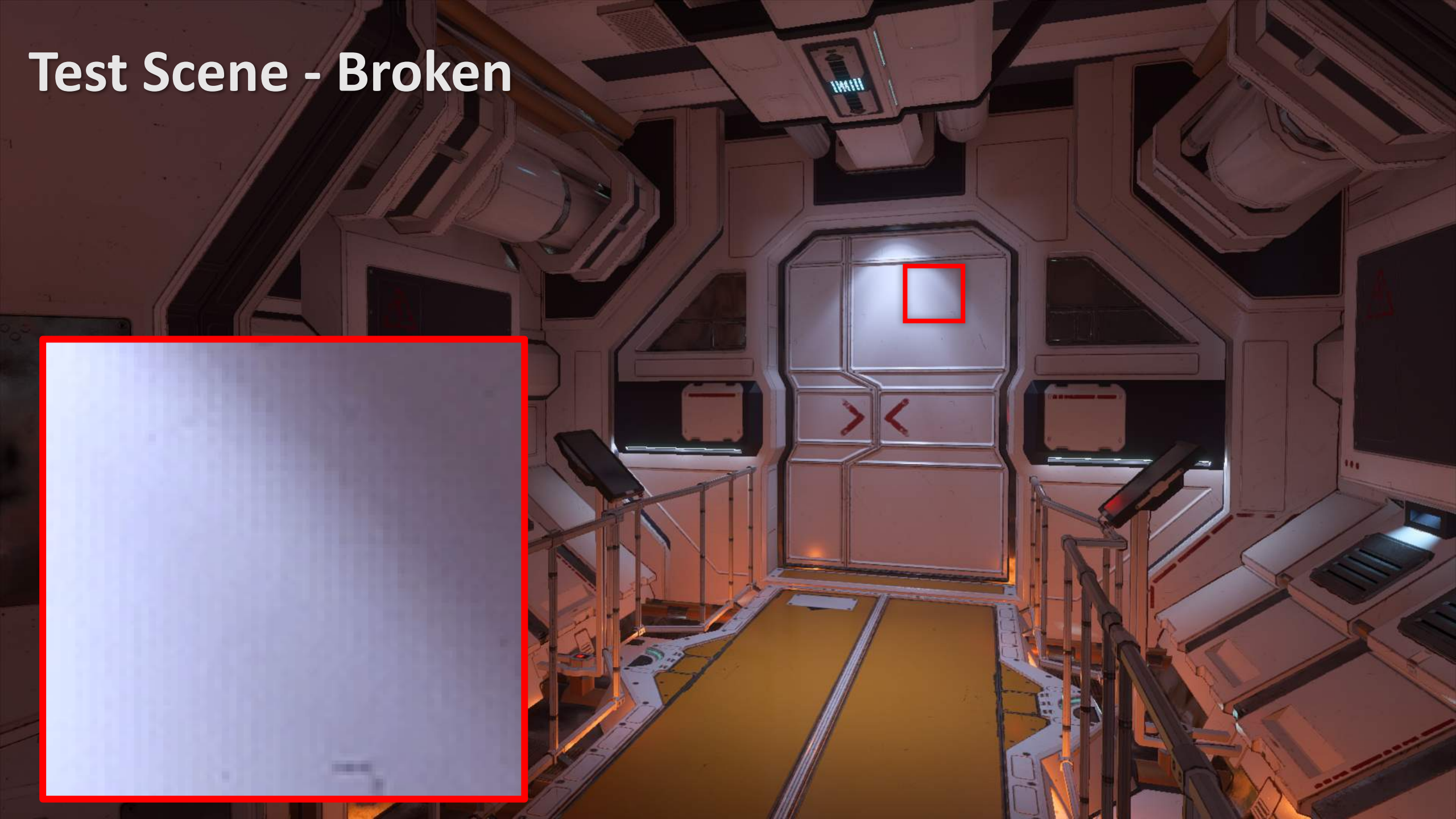
Post Processing

- Trade-off between performance and quality
  - Move as much post-processing before CB resolve as possible
- Very time consuming and painful work
  - Evolving codebase breaks checkerboard often
  - New concept to many
- Auto-test checkerboard pipeline if possible

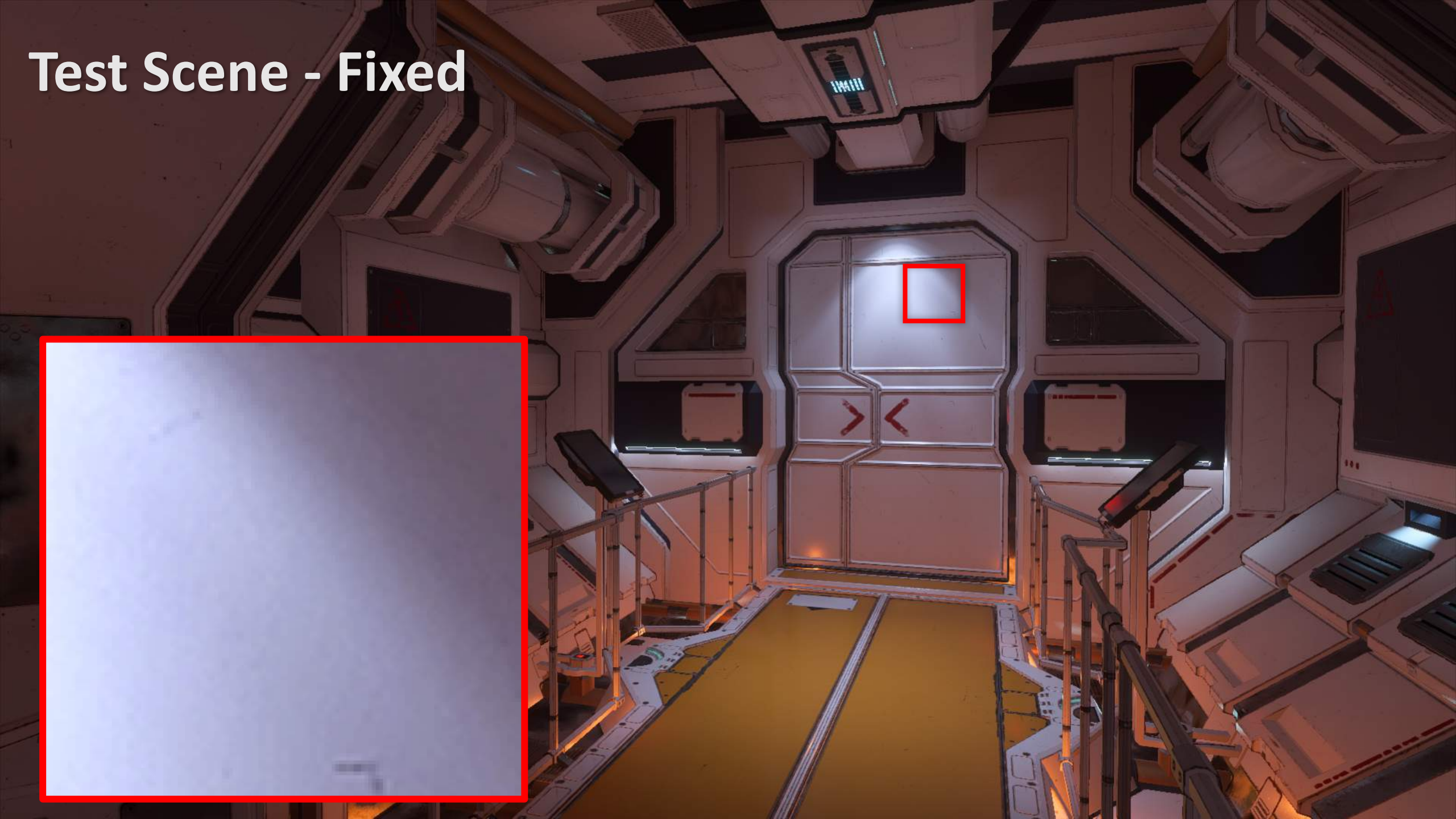
Post Processing

- Operations that ignore geometry provide little value at 4K
  - i.e. SSAO, luminance estimation, etc..
- Limiting color propagation over an edge - run at lower res
- Need to account for checkerboard "pixel grid"
  - Linear sampling of checkerboard surface is problematic
  - Aspect ratio of the buffer is different from normal 16:9
- Most cases you can ½ the horizontal filter width

Test Scene - Broken



Test Scene - Fixed



Post Processing

Corrected with:

```
float2 getCheckerboardUvOffset(uint2 pixCoord, uint cbStatePacked)
{
    float halfWidth = asfloat(cbStatePacked);
    uint parityBit = cbStatePacked;
    return float2(halfWidth * (((pixCoord.y + parityBit) & 1) - 0.5), 0);
}

screenUv += getCheckerboardUvOffset(pixelCoord, g_checkerboardUvOffsetPacked);
```

Offsets UV used for clip-space position reconstruction

```
static u32 getCheckerboardUvOffsetPacked(u32 viewWidth, bool enabled, u32 frameIndex)
{
    union
    {
        float f;
        u32 u;
    } packed;

    if (enabled)
    {
        packed.f = 0.5f / float(viewWidth);
        packed.u &= ~u32(1);
        packed.u |= (frameIndex & 1u);
    }
    else
    {
        packed.f = 0.0f;
    }

    return packed.u;
}
```

BF1 : PS4™Pro

1600x1800

- Clear (IDs and Depth)
- Partial Z-Pass
- G-Buffer Laydown
- Resolve AA Depth
- G-Buffer Decals
- HBAO + Shadows
- Tiled Lighting + SSS
- Emissive
- Sky
- Transparency
- Velocity Vectors

3200x1800

- CB Resolve + Temporal AA
- Motion Blur
- Foreground Transparency
- Gaussian Pyramid
- Final Post-Processing
- Silhouette Outlines

3840x2160

- Display Mapping + Resample

MEA : PS4™Pro

1600x1800

- Clear (IDs and Depth)
- Partial Z-Pass
- G-Buffer Laydown
- Resolve AA Depth
- G-Buffer Decals
- HBAO + Shadows
- Tiled Lighting + SSS
- Emissive
- Sky
- Transparency
- Velocity Vectors

3200x1800

- CB Resolve + Temporal AA
- Sprite Depth-of-Field
- Motion Blur
- Foreground Transparency
- Gaussian Pyramid
- Final Post-Processing
- Silhouette Outlines

3840x2160

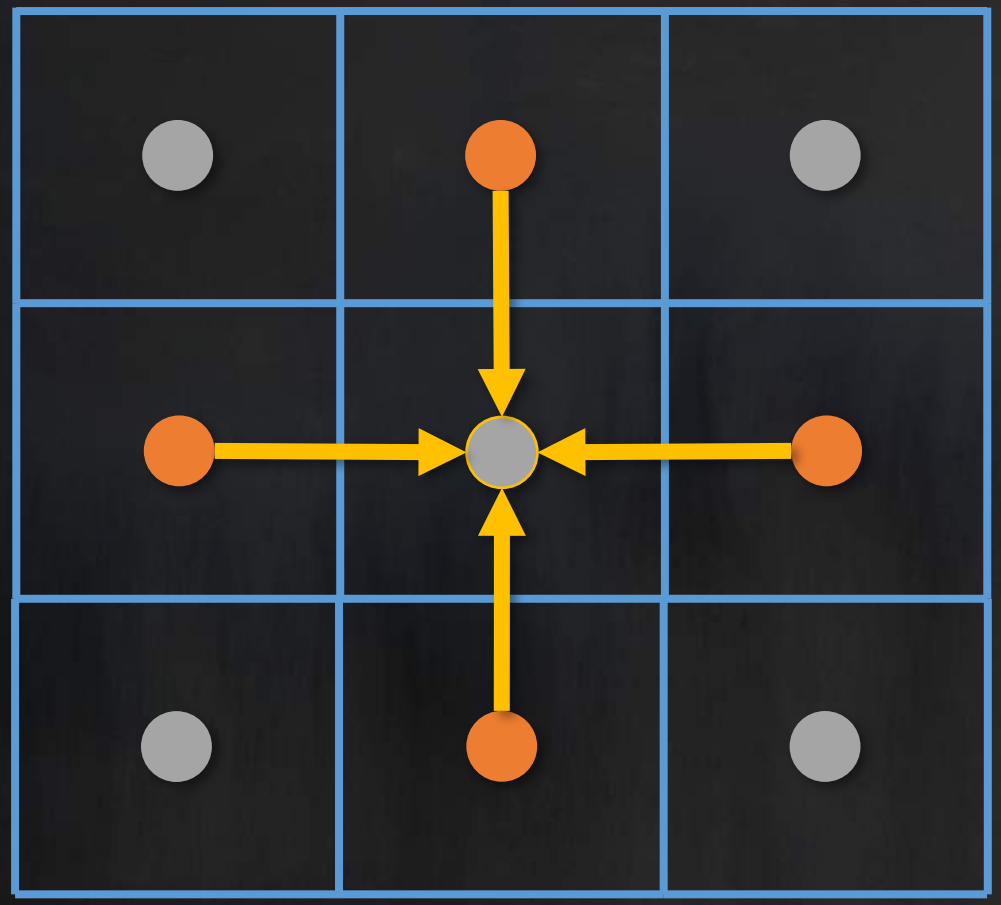
- Display Mapping + Resample

Checkerboard Resolve

- Spatial Component
  - Color Bounding Box
  - Differential Blend Operator
  - Object and Primitive IDs
- Temporal Component
  - Sub-Pixel Jittering
  - Velocity-Based Reprojection
  - Neighborhood Clamping

Spatial Component

- Color Bounding Box
- Differential Blend Operator
- Object and Primitive IDs

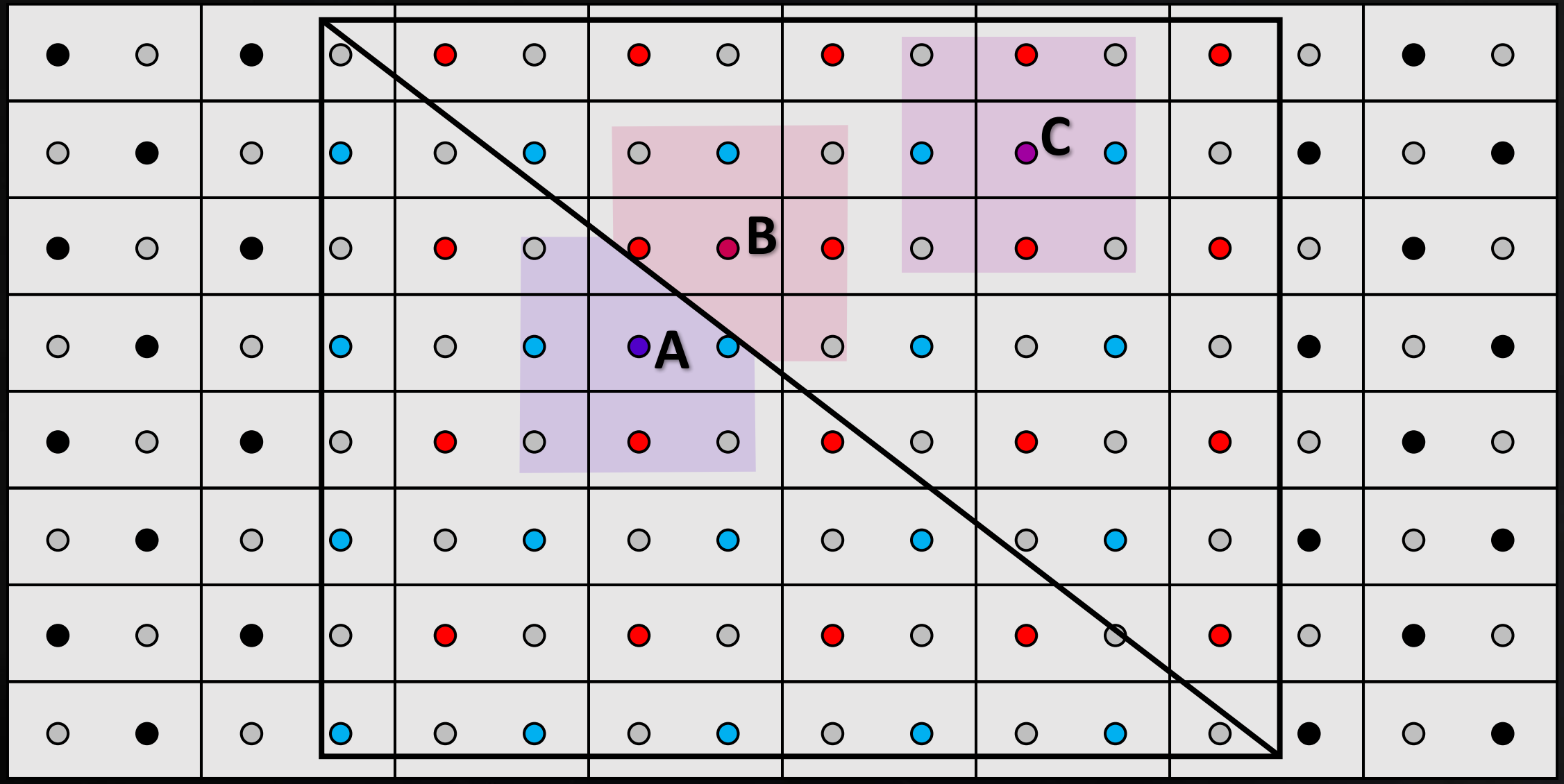


Color Bounding Box

- Checkerboard resolve does not use depth
- Use color to determine if edges are soft
- Avoid contribution to a pixel from different objects
- If all neighbors are different objects, average is used

Image courtesy of SIE [15]

Color Bounding Box



Color Bounding Box

✘ Copy Single Sample

✓ Comparison Heuristic

SUCKER PUNCH

Color Bounding Box

- ✘ Copy Single Sample

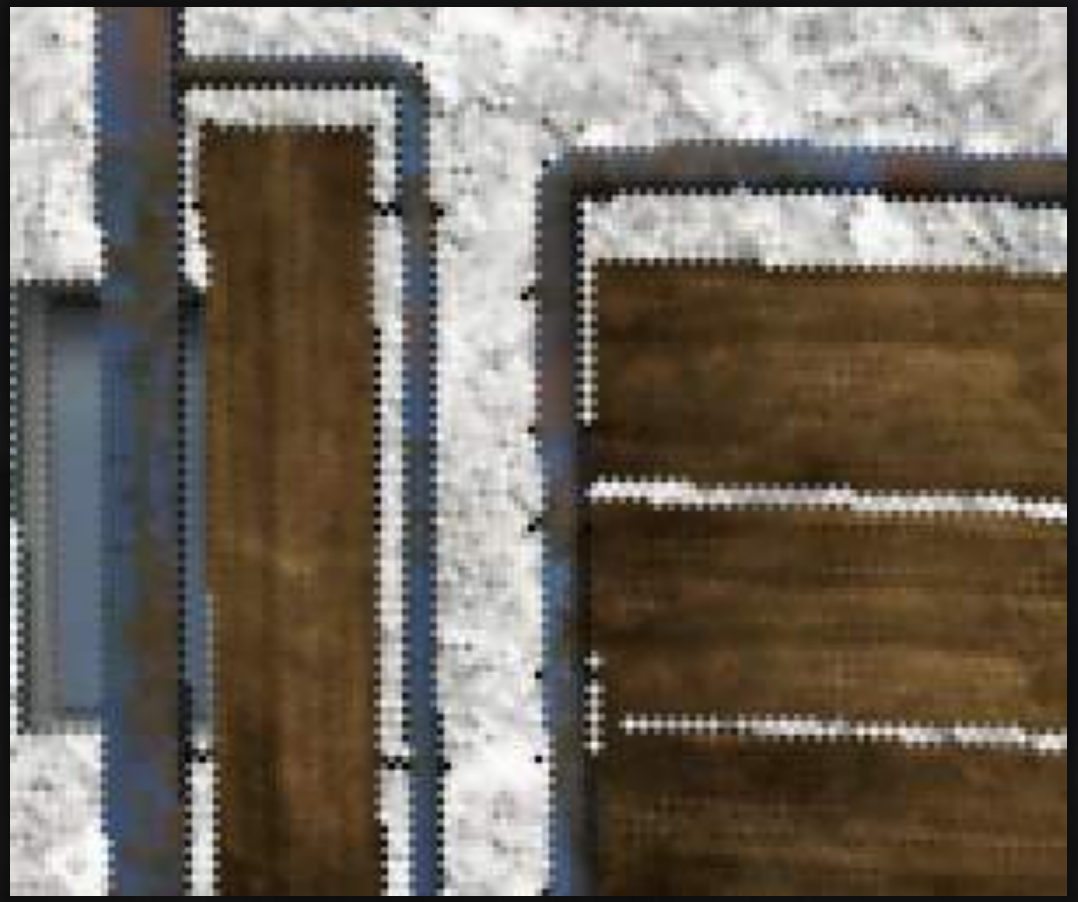
Color Bounding Box

- ✓ Comparison Heuristic

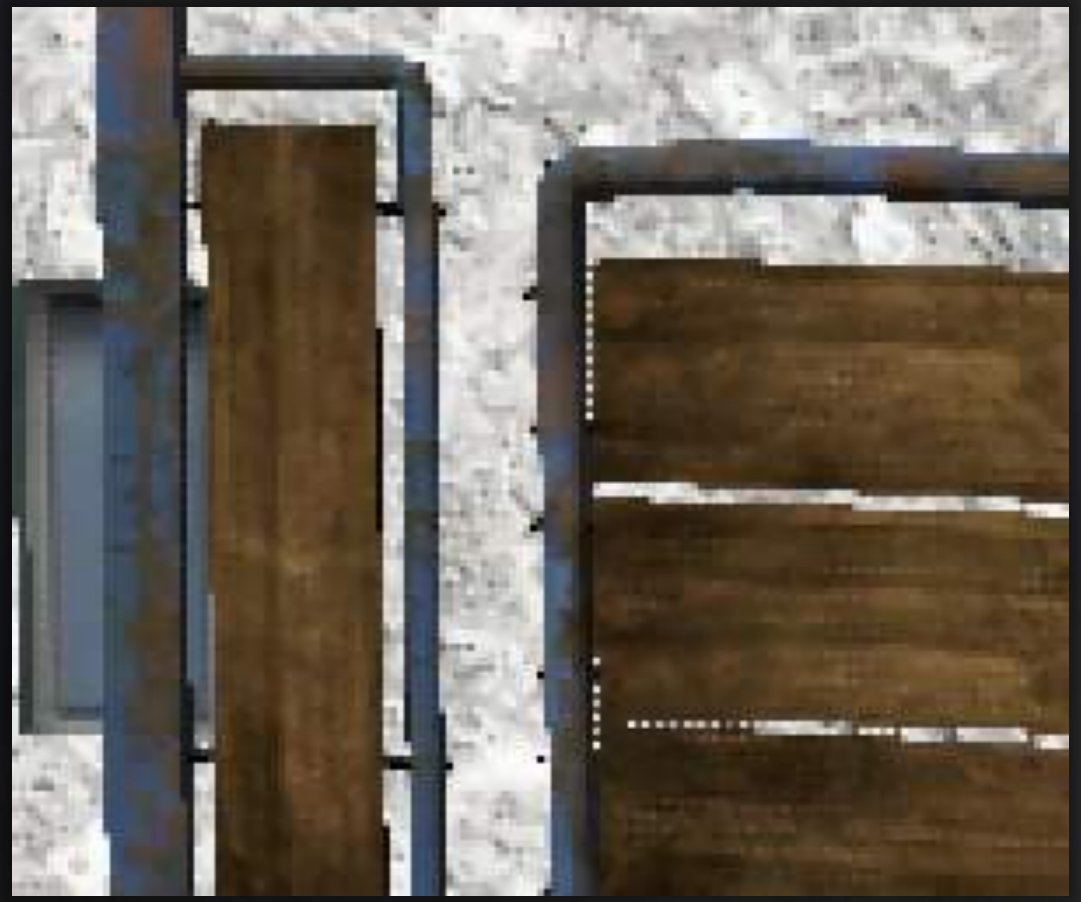
SUCKER PUNCH

Images courtesy of SIE [15]

Differential Blend Operator



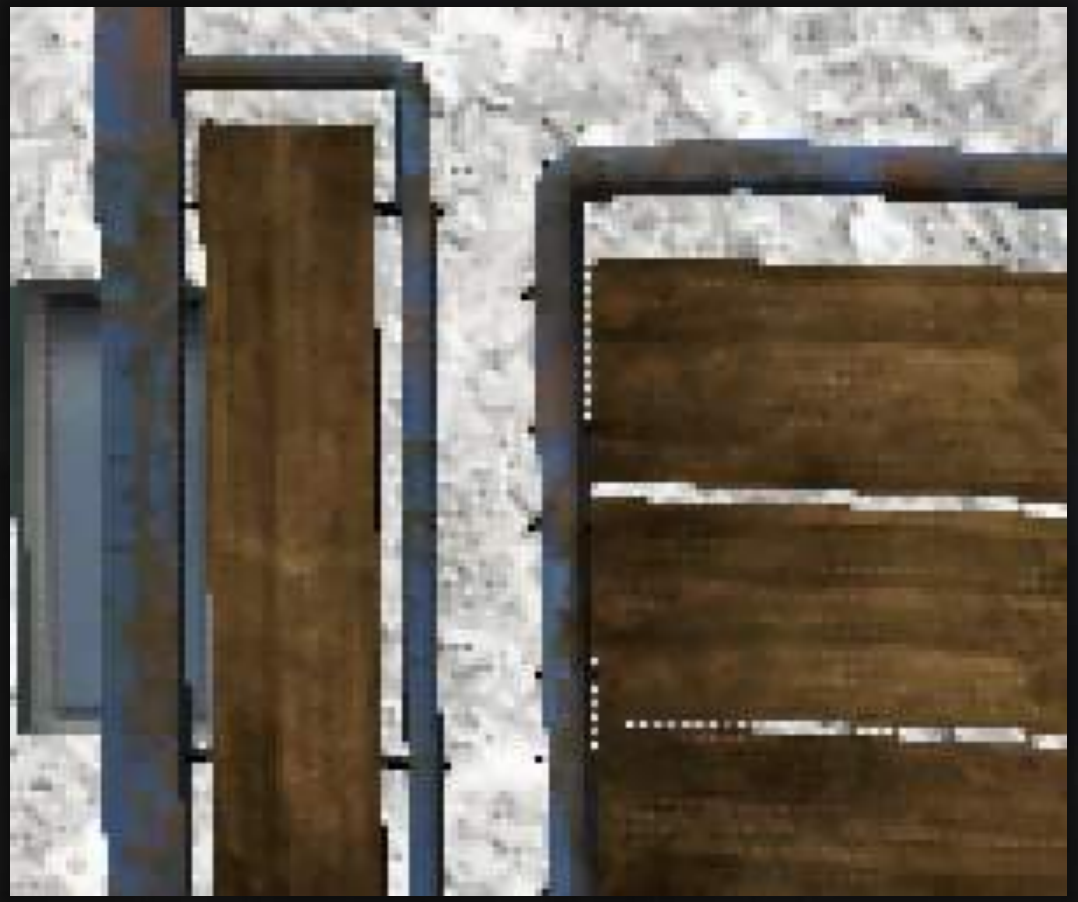
✘ Neighborhood Average



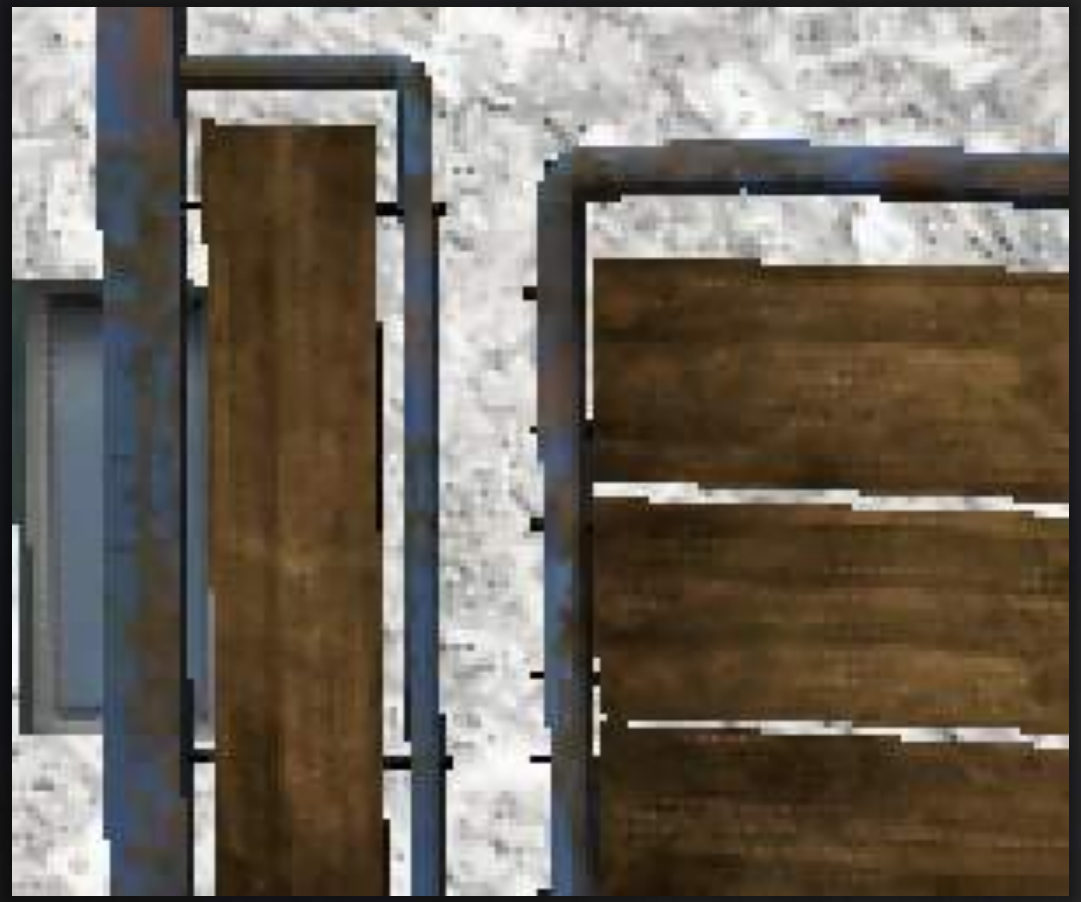
✓ Differential Blending

Images courtesy of SIE [15]

Differential Blend Operator



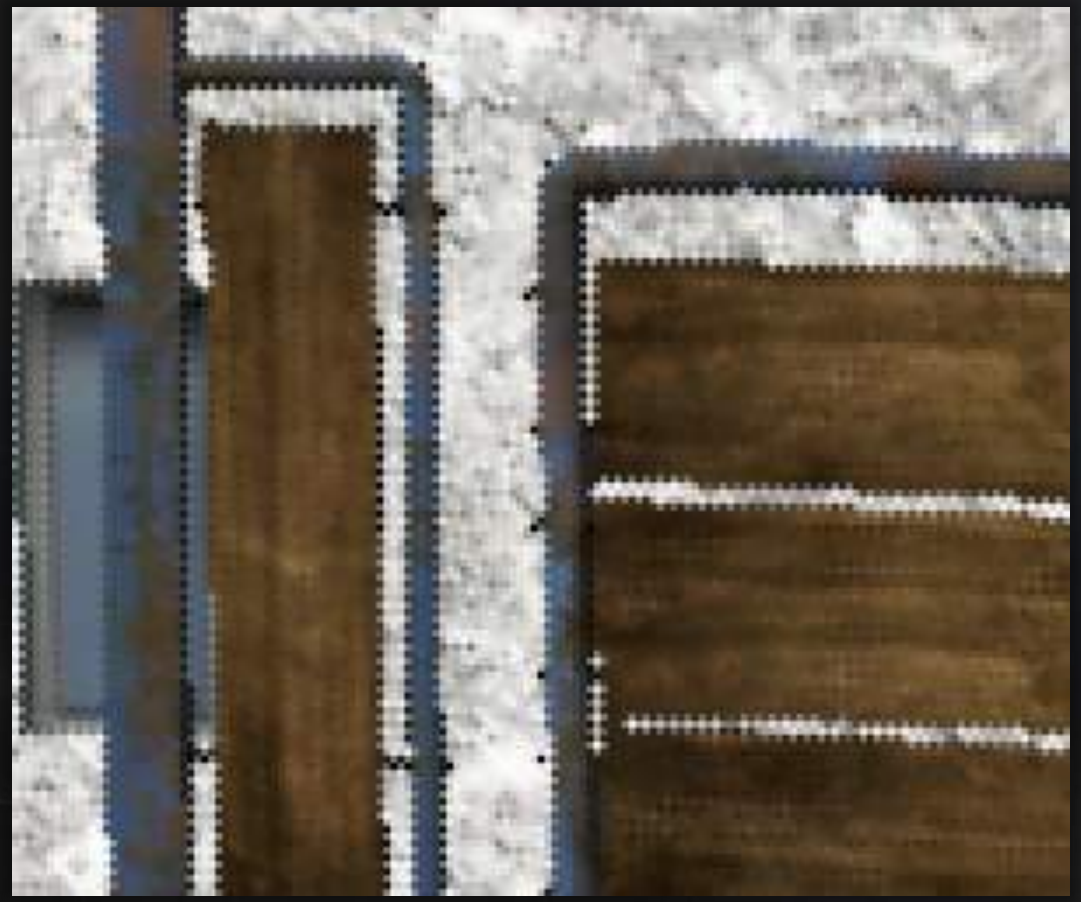
✓ Differential Blending



✓✓ Differential Blending + IDs

Images courtesy of SIE [15]

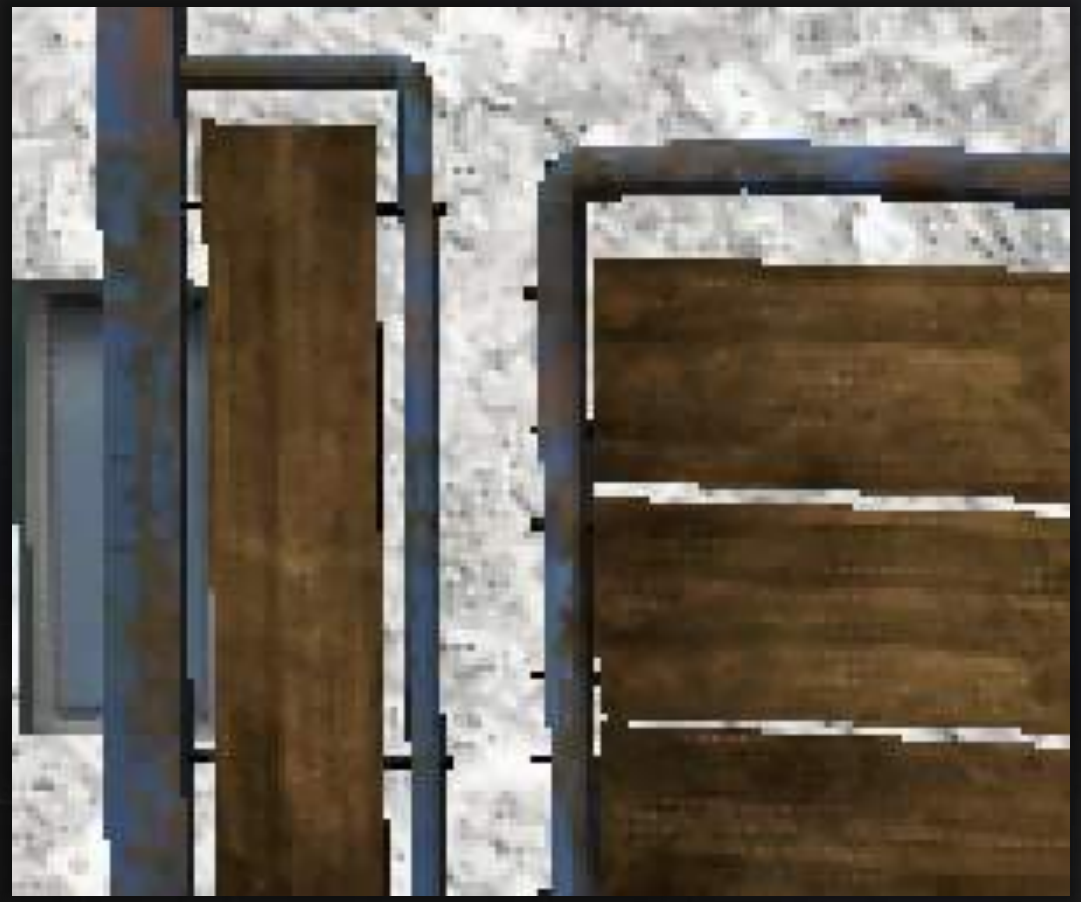
Differential Blend Operator



- Neighborhood Average

Images courtesy of SIE [15]

Differential Blend Operator



✓✓ Differential Blending + IDs

Differential Blend Operator

```
float colorDiffBlend(float3 a, float3 b)
{
    float differential = a - b;
    float len = sqrt(dot(differential, differential));
    return 1.0f / (len + 0.001f);
}

float differenceWeights[2] =
{
    (objectId[UP]   != objectId[DOWN])  ? 1.7f : colorDiffBlend(colors[UP],   colors[DOWN]),
    (objectId[LEFT] != objectId[RIGHT]) ? 1.7f : colorDiffBlend(colors[LEFT], colors[RIGHT])
};
```

Images courtesy of SIE [15]

Object and Primitive IDs

✘ Object ID
✘ Primitive ID

✘ Object ID
✓ Primitive ID

✓ Object ID
✘ Primitive ID

✓ Object ID
✓ Primitive ID

Temporal Component

- Similar to existing research - [2][11][12]
- Not covered here
  - Sub-Pixel Jittering
  - Neighborhood Clamping
- Velocity-Based Reprojection
  - Updated for checkerboard

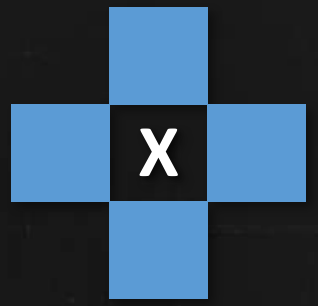
Velocity-Based Reprojection

- Objects and surfaces are moving
- Pixel grid is stationary

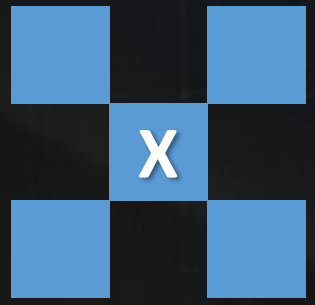
- Need to correlate 3D surfaces in motion
- Surfaces write per-pixel velocity vectors

- Dilate velocities to keep anti-aliased edges
  - Use front-most velocity in 3x3 window
  - Depth is expensive to fetch!

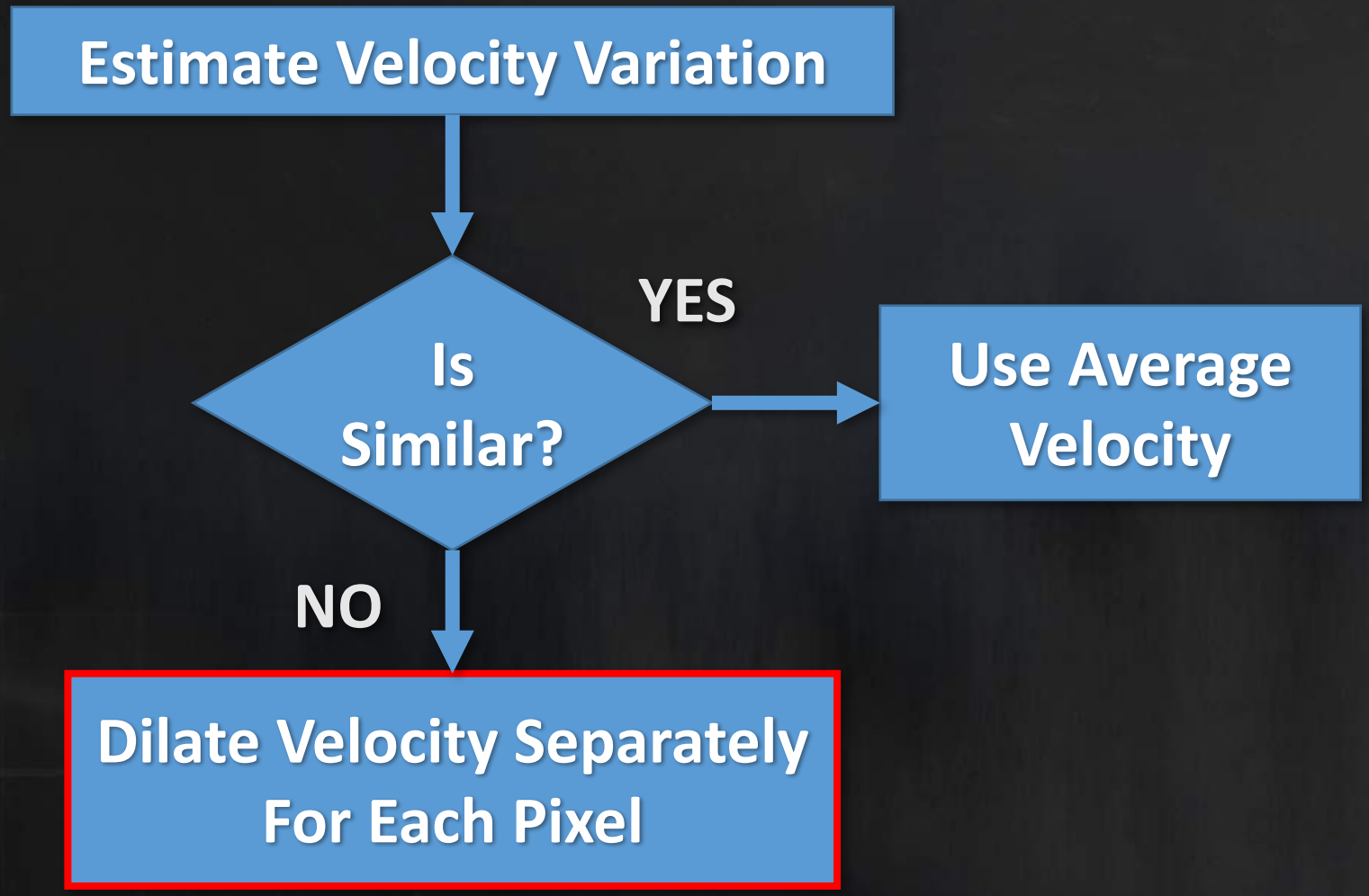
2 Pixels Per Thread
Pass-through and Filtered



Filtered Pixel

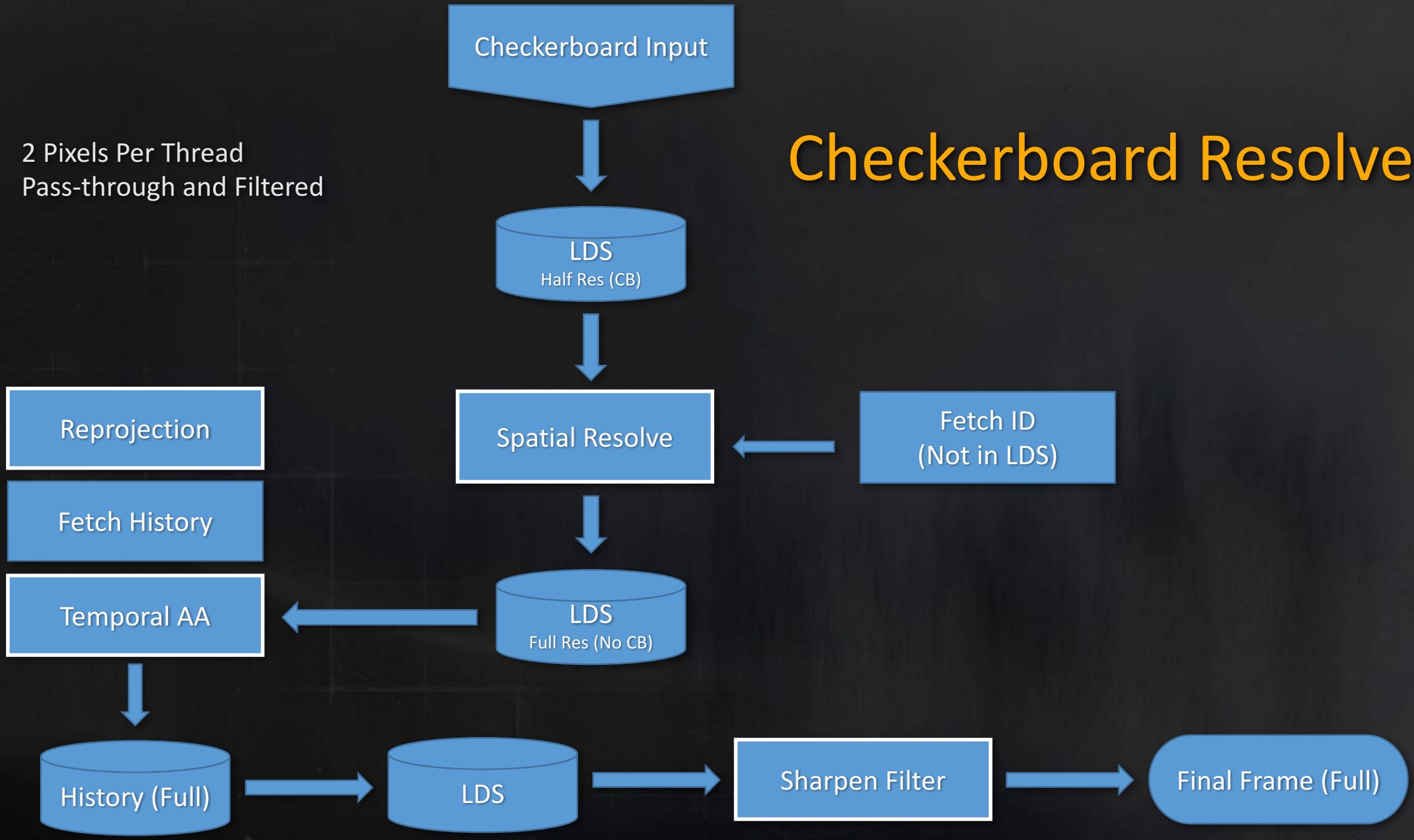


Pass-through Pixel



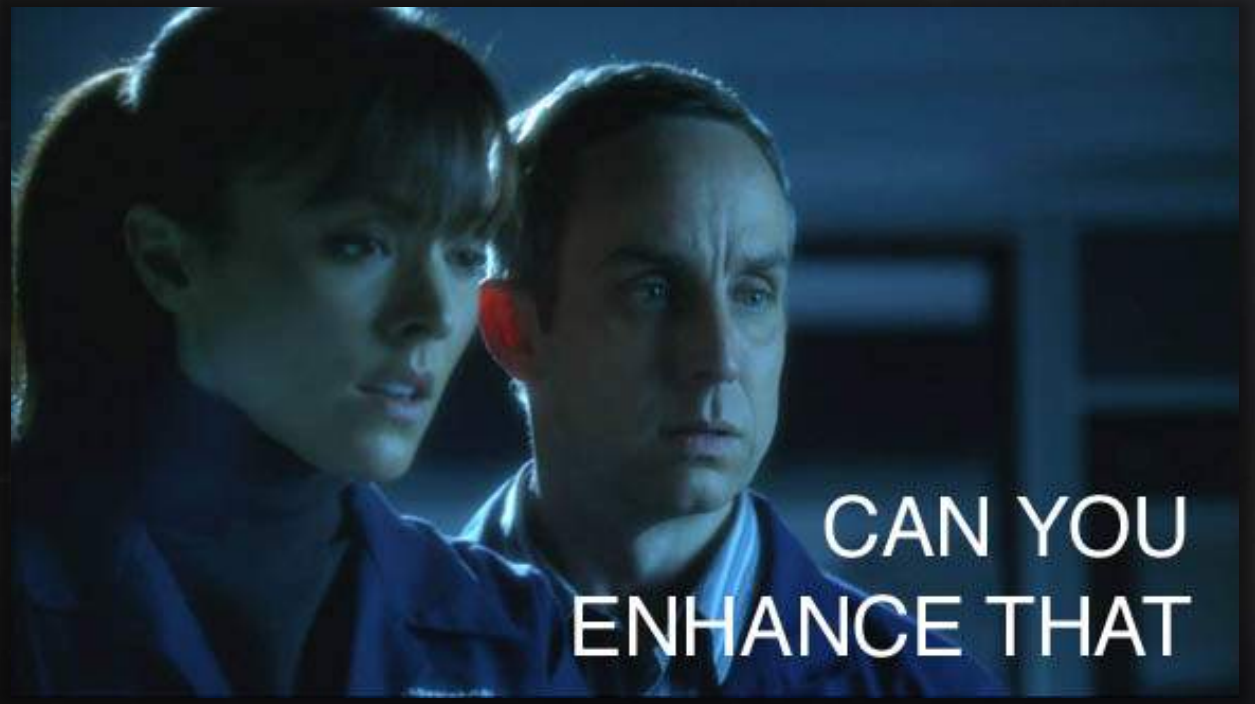
Checkerboard Resolve

2 Pixels Per Thread
Pass-through and Filtered



Sharpen Filter

- Blurred image lacks high-frequency components
  - Hard to discern objects, visual cortex lacks information
  - Ganglion cells respond acutely to high-frequency components [6]



Sharpen Filter

- Amplify (recover) high-frequency components
  - Greatly enhance visual quality
  - Performed after image reconstruction in temporal AA
- Be careful to not reintroduce aliasing
  - (false high-frequency)
- Be careful to not introduce ringing
  - (over-sharpening)


Before

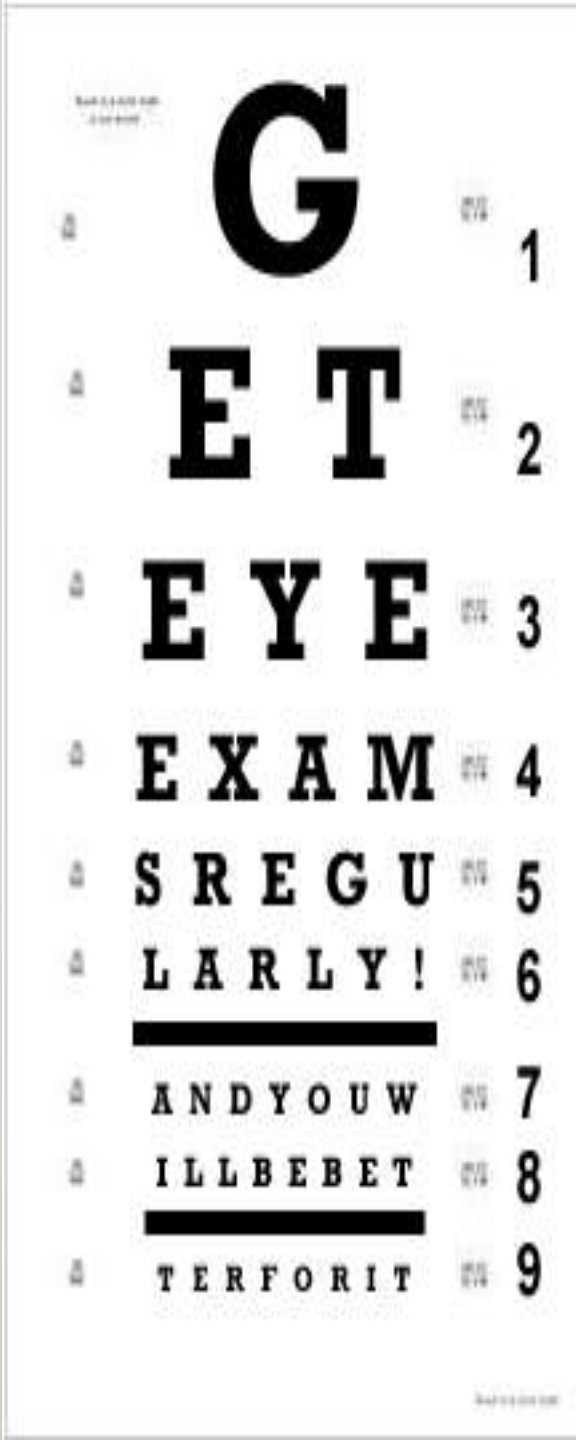

After


Before


After


Before


After


G
1
ET
2
EYE
3
EXAM
4
SREGU
5
LARLY!
6
ANDYOUW
7
ILLBEBET
8
TERFORIT
9

Post Processing

- Once checkerboard is in enabled, brace yourself
- Every single artifact now looks like a checkerboard artifact
- Add lots of debug overlays to help debug
- Teach others how to diagnose!

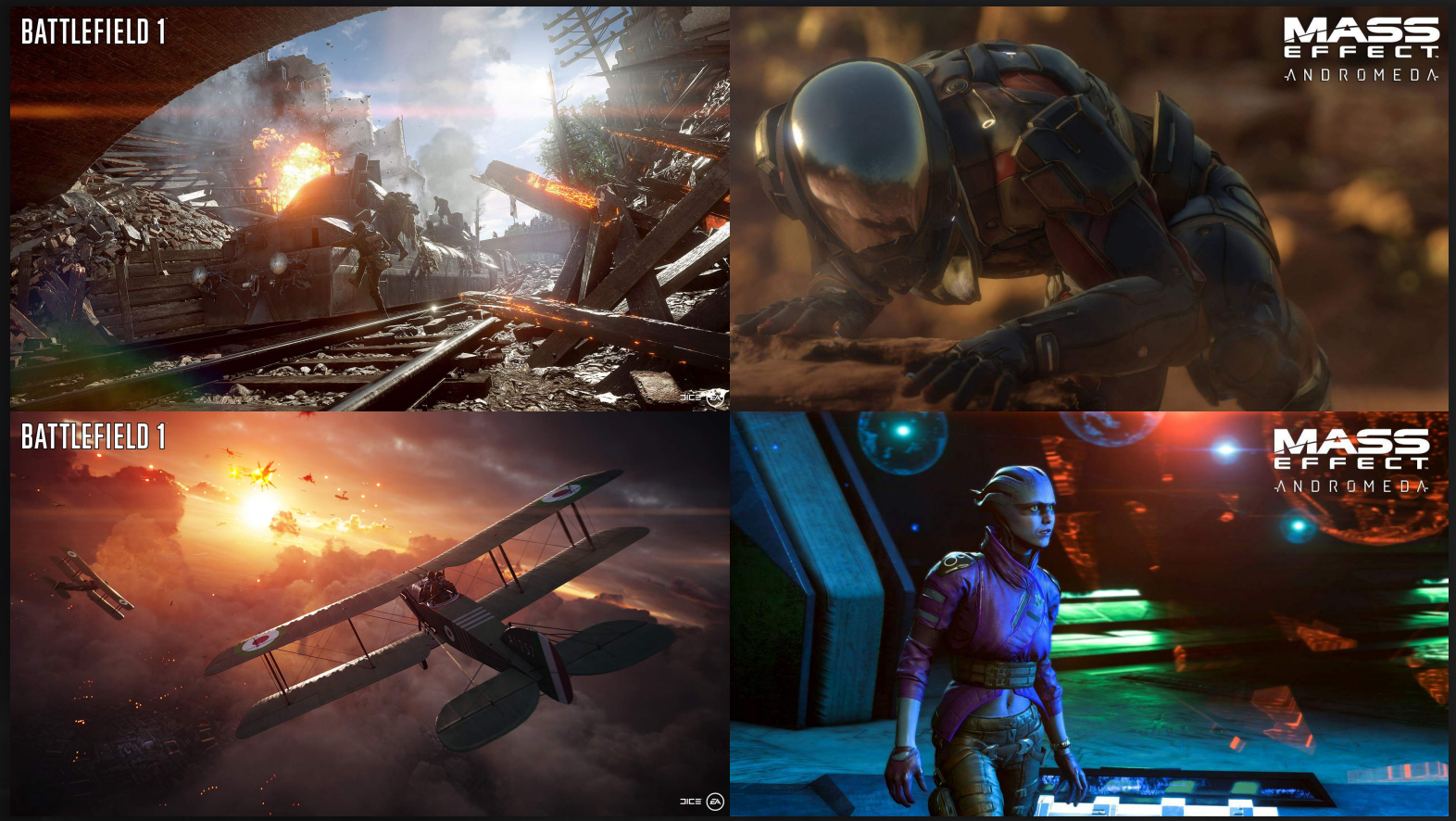

E
S
W


E
N
S
W

"Call me old fashioned, but I love a good witch-hunt!"

Agenda

- Motivation
- Configuration
- Features
- Optimizations
- Post Processing
- **Pipeline**
- Future Work



Render Target Aliasing

- Biggest pain of 4K transition was memory!
  - Majority of our render targets were to blame
  - At the time, no fancy memory management of them
- Extra effort spent here impacted other 4K improvements
- "11th hour" explicit aliasing to ship – saved ~230mb
- Future Titles
  - See: FrameGraph: Extensible Rendering Architecture in Frostbite


30/120
10

Dynamic Resolution Scaling

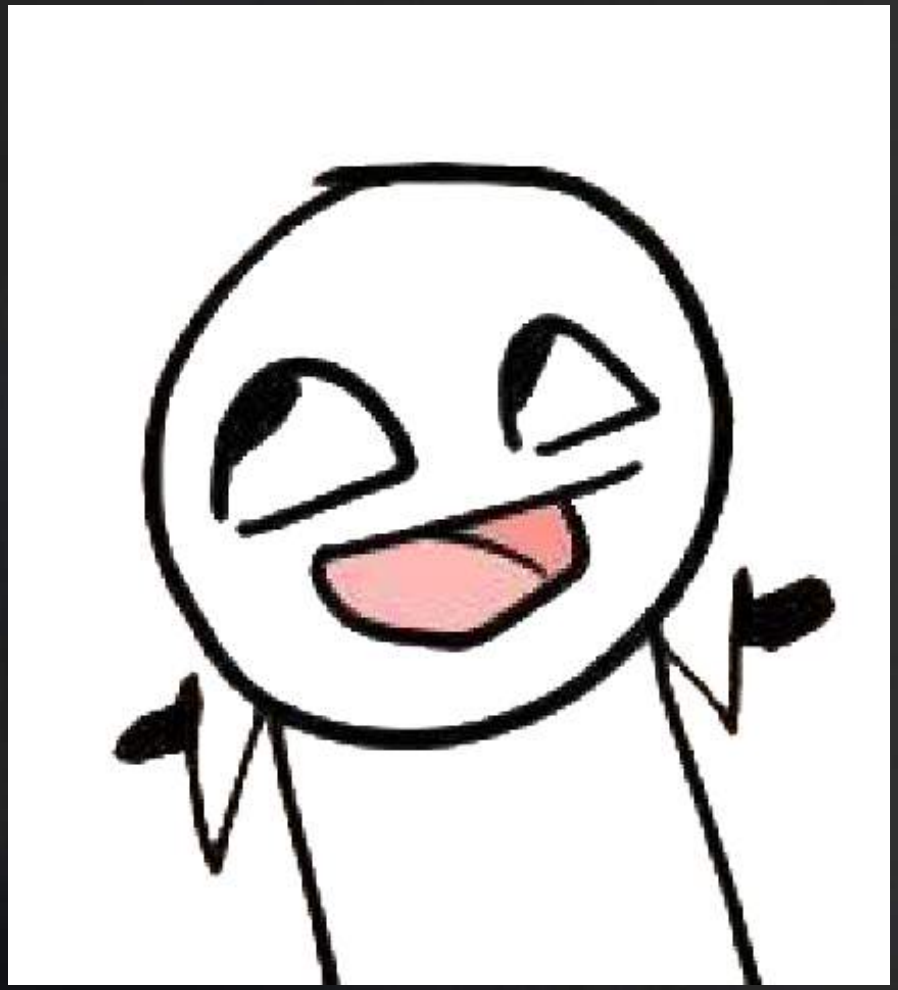
- Developed by DICE and Microsoft
- Plays nicely with 4K checkerboard
- Checkerboard always active
- Dynamic scaled initial frame resolution
  - Determined with running performance heuristic

Dynamic Resolution Scaling

- BF1 contains a number of infrequently running GPU tasks
- Caused resolution to slightly adjust almost every frame
  - Not an issue
- Worked with jitter to provide variation in subpixel detail
- Tried preventing upscale if camera wasn't moving
  - Resulted in noticeably lower quality image

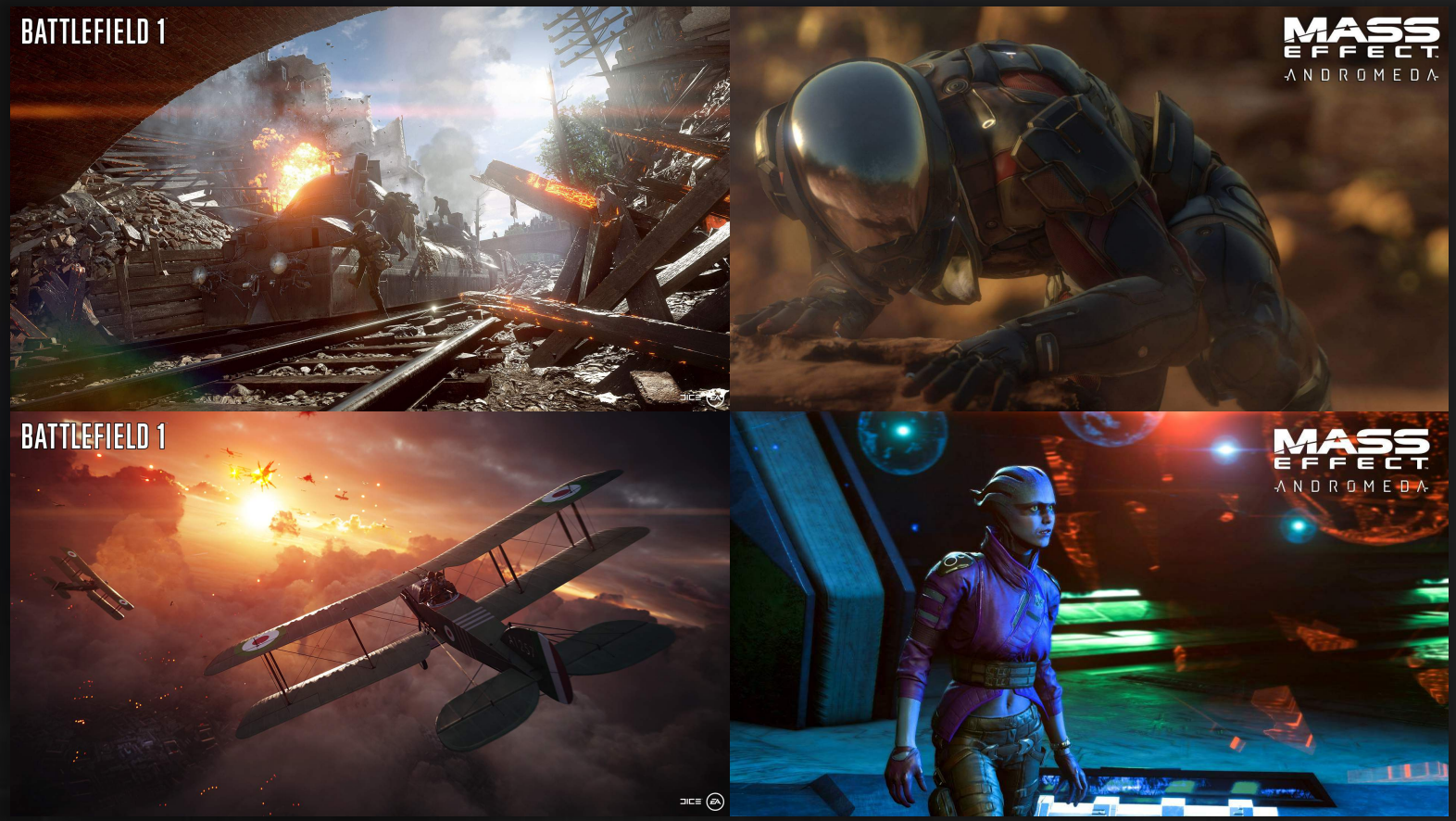
Dynamic Resolution Scaling

- No one complained about it!



Agenda

- Motivation
- Configuration
- Features
- Optimizations
- Post Processing
- Pipeline
- **Conclusion**




1800p CB
15.99ms
1800p
21.07ms
PS4™Pro

Performance

- **Total:** 15.99 ms

- **Clear IDs:** 0.57 ms
- **Copy Expanded HTile [3x]:** 0.02 ms
- **Resolve EQAA Depth [3x]:** 0.69 ms
- **CB Resolve + Temporal AA:** 1.15 ms

- **Everything Else:** 13.56 ms

- 1800p CB vs 1800p (saves **5.08 ms**)

PS4™Pro

Performance

Timer	1800p	1800p CB
G-Buffer	4.25 ms	3.11 ms
Shadows	4.13 ms	2.63 ms
Lighting	2.31 ms	1.61 ms
Sky	0.63 ms	0.32 ms
Half Res	1.54 ms	0.77 ms
Velocity	0.57 ms	0.25 ms
HBAO	2.47 ms	1.08 ms

PS4™Pro


1800p CB
23.42ms
1800p
36.82ms
PS4™Pro

Performance

PS4™Pro

- **Total:** 23.42 ms

- **Clear IDs:** 0.41 ms
- **Copy Expanded HTile [3x]:** 0.02 ms
- **Resolve EQAA Depth [3x]:** 0.80 ms
- **CB Resolve + Temporal AA:** 2.00 ms

- **Everything Else:** 20.19 ms

- 1800p CB vs 1800p (saves **13.40 ms**)

Performance

Timer	1800p	1800p CB
G-Buffer	5.11 ms	4.41 ms
Shadows	3.70 ms	3.32 ms
Lighting	13.5 ms	7.32 ms
Sky	0.50 ms	0.29 ms
Half Res	2.53 ms	1.48 ms
Velocity	0.32 ms	0.27 ms
HBAO	3.11 ms	1.27 ms

PS4™Pro

Future Work

- Further improvements to checkerboard resolve
- Alternative approaches to alpha testing
  - Alpha mask export
  - Pixel rate coverage (not sample)
- More uses for IDs
  - Temporally stable object and primitive IDs?
  - Replace stencil-based tagging [9]
    - Help eliminate ghosting with temporal re-projection
  - Remove heuristics in favor of comparisons

Future Work

- Packed checkerboard on other platforms
  - Hardware features like ID buffer only exist on PS4™Pro
  - Reasonable workarounds exist for Xbox™ and base PS4™.
- Need EQAA + programmable sample locations
  - Vulkan + DirectX12
- Driver support for efficient EQAA depth resolve
  - Leave source compressed
  - Bandwidth-bound

Future Work

- Filtered visibility buffer [10] + decoupled shading
- G-buffers are challenging at high resolutions
- Classic deferred shading isn't the answer
- Decouple g-buffer from screen resolution

Special Thanks

- Tobias Berghoff
- Nicolas Serres
- Tim Dann
- Tomasz Stachowiak
- Yasin Uludag
- Keven Cantin
- Mark Cerny
- Johan Andersson
- Christina Coffin
- Chris Brennan
- Timothy Lottes
- Rob Srinivasiah
- Jason Scanlin
- Dave Simpson
- Bart Wronski
- Martin Fuller
- James Stanard
- Ivan Nevraev
- Colin Barré-Brisebois
- Matthäus Chajdas
- Frostbite Rendering
- Sucker Punch

References

- [1] Rendering 'Rainbox Six | Siege' – Jalal El Mansouri
- [2] High Quality Temporal Supersampling – Brian Karis
- [3] Taking Killzone Shadow Fall Image Quality Into The Next Generation – Michal Valient
- [4] Radeon Southern Islands 3D/Compute Register Reference Guide – AMD
- [5] MLAA on PS3 – Tobias Berghoff, Cedric Perthuis
- [6] Hubel, D. H., and T. N. Wiesel. 1979. "Brain Mechanisms of Vision." *Scientific American* 241(3), pp. 150–162.
- [7] EQAA Modes for AMD 6900 Series Graphics Cards – AMD
- [8] Radeon Evergreen/Northern Islands Acceleration - AMD
- [9] Temporal Antialiasing in Uncharted 4 – Naughty Dog
- [10] The Filtered and Culled Visibility Buffer – Wolfgang Engel
- [11] Real-Time Global Illumination and Reflections in Dust 514 – Hugh Malan
- [12] Anti-Aliasing Methods in CryENGINE 3 – Tiago Sousa
- [13] Dynamic Resolution Rendering – Doug Binks
- [14] Subpixel Reconstruction Antialiasing – NVIDIA
- [15] "PlayStation®4 SDK / PS4™Pro High Resolution Technologies " - SIE (Tobias Berghoff, Tim Dann)
- [16] "Implementing 4K Checkerboard - Lessons from 4K Knack and U4 Demos", Jason Scanlin - SIE WWS ICE, May 2016
- [17] "Making the inFAMOUS 4K Demo", Tobias Berghoff - SIE WWS ATG, May 2016

Questions?

Graham Wihlidal
graham@frostbite.com
Twitter - @gwihlidal

Thank You!