Rendering Tech of Space Marine

Korea Game Conference 2011
Nov 6-9, 2011

Pope Kim
Senior Graphics Programmer

Daniel Barrero
PhD, Graphics Programmer

What is Space Marine?

- 3rd Person Action Game
- Shipped in September 2011
- XBOX 360, PS3 and PC
- Based on Warhammer 40,000 Franchise

Video

Goals

- Tech-driven goals
  - Supports Xbox 360, PS3 and PC
  - Runs at solid 30 frames per second
- Art-driven goals
  - Fast iteration
  - Highly detailed models
  - Lots of lights, blood, gore and explosion
- Design-driven goals
  - Lots of orks
  - Lots of blood (yes, again)

Features: Industry Standard

- Deferred Lighting
- SSAO
- Linear Space Lighting
- Full-screen AA (similar to FXAA)
- Cascaded Shadow Map
- Deferred Shadow
- Colour Correction (Colour Grading)

Features: Space Marine Specific

- World Occlusion
- Screen Space Decal
- Character Fill Light
- Ambient Saturation
- FX system that does everything
- Character Customization

Agenda

1. Deferred Lighting

2. Screen Space Decals

3. World Occlusion

WARHAMMER 40,000 SPACE MARINE

Example Scene

Rendering Passes

Pass	Budget(ms)
Depth-Pre	0.50
G-Buffer + Linear Depth	5.05
Ambient Occlusion	2.25
Lighting	8.00
Combiner Pass	5.00
Blend	0.15
Gamma Conversion	1.30
FX	2.75
Post Processing	3.70
UI	0.50
Total	29.20

Deferred Lighting Overview

- (Traditional) Forward Rendering
  - Object rendering pass: does everything
- Deferred Shading
  - Object rendering pass: saves all surface parameters
  - Lighting pass: saves lighting result
  - Combiner pass: combines lighting result + surface material in screen space

Deferred Lighting Overview

- Deferred Lighting
  - Our approach
  - Also called Light Pre-Pass [Wolf08]
  - Almost same as deferred shading except:
    - First pass doesn't saves surface material
    - Combiner pass is another object pass combining:
      - Lighting result (full-screen)
      - Surface material (from object)
  - For more details:
    `http://en.wikipedia.org/wiki/Deferred_shading`

Deferred Lighting Passes

Pass	Budget(ms)
Depth-Pre	0.50
G-Buffer + Linear Depth	5.05
Ambient Occlusion	2.25
Lighting	8.00
Combiner Pass	5.00
Blend	0.15
Gamma Conversion	1.30
FX	2.75
Post Processing	3.70
UI	0.50
Total	29.20

Our Deferred Implementation

- Rendering resolution
  - Xbox 360: 1200 x 720
  - PS3: 1138 x 640
- No multiple Render Targets
  - Saves ROP bandwidth
  - Reduces number of texture look-ups
- To achieve this:
  - G-Buffer: Encode roughness and specular together
  - Lighting Pass: spec result is saved as monochrome

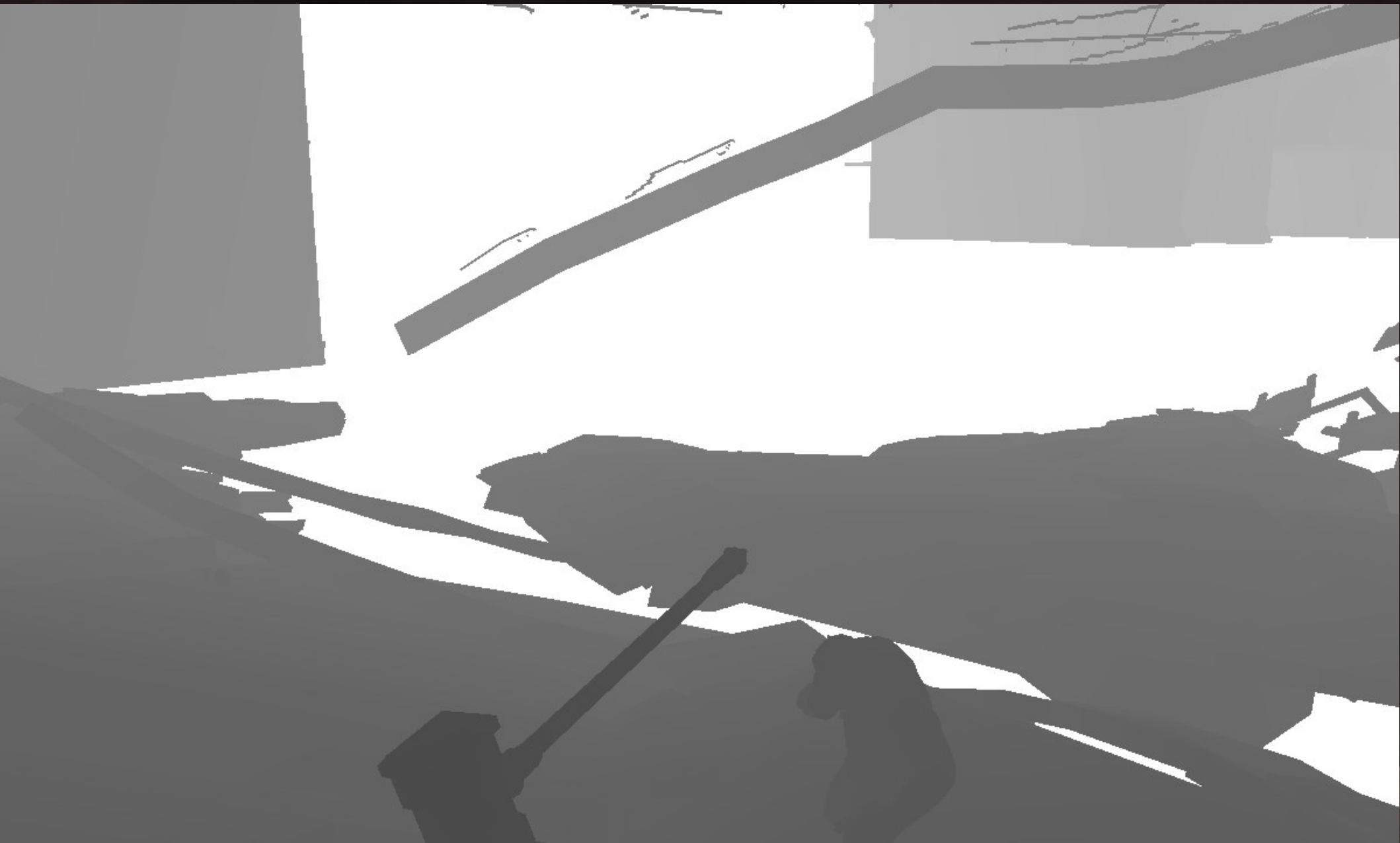
Depth-pre Pass

- To reject occluded objects early in G-Buffer
  - Double fill rate if render target is disabled
  - Hi-Z to reject before ROP(Raster Operation)
- Front-to-back
- But, drawing all objects was slow
- To fit this in 0.5 ms, we only draw
  - Maximum 75 objects
  - Big enough objects in project space
    - X360: 0.5
    - PC and PS3: 2.0
- Other objects will be drawn to Z-buffer in Gbuffer pass

WARHAMMER 40,000 SPACE MARINE

Example Scene

Depth-pre Pass



Geometry Buffer Pass

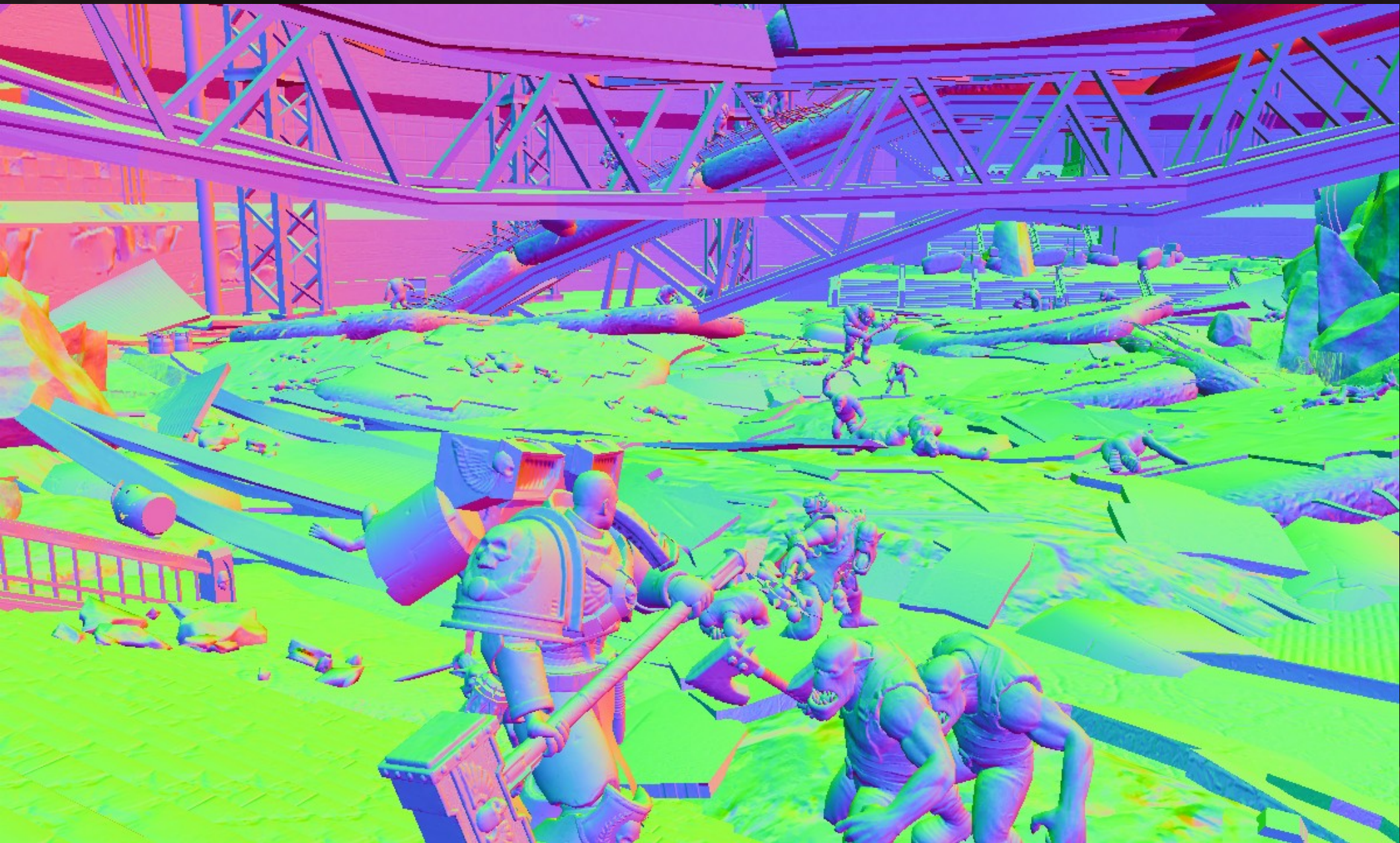
- One A8R8G8B8 RT
  - View space normal in RGB
    viewNormal.rgb * half3(0.5, 0.5, 0.66666) + half3(0.5, 0.5, 0.33333)
  - Specular power/roughness in A
- Sort by material
- Occlusion query is issued per draw call
  - If depth-pre did a good job, many objects are culled
  - The query result is used in combiner pass
  - On x360, new faster occlusion query API was used
- This is last stage where we write to depth

WARHAMMER 40,000 SPACE MARINE

Example Scene

WARHAMMER 40,000 SPACE MARINE

Geometry Buffer



G-Buffer (Alpha, inverted)

Depth Buffer after Gbuffer Pass

Lighting Pass

- Almost all objects are lit in this pass
  - Exception: transparent objects(e.g, hair)
- Uses the resulting buffer from GBuffer pass
- Our basic diffuse lighting model is Oren-Nayar
  - Rough surface propagates lights across surface=slow fall-off
  - Usual Lambertian model very smooth surface, not ideal for human face or cloth
  - Indeed, we use roughness only for characters

Oren-Nayar Lighting Model

- We used a lookup texture for optimization for a while
  (http://content.gpwiki.org/index.php/D3DBook:(Lighting)_Oren-Nayar)

```
half computeOrenNayarLighting_TextureLookup( half3 N, half3 L, half3 V, half roughness )
{
  half LdotN = dot( L, N );
  half result = saturate(LdotN);
  if ( roughness > 0 )
  {
     // compute the other aliases
     half VdotN = dot( V, N );
     half gamma = dot( V - N * VdotN, L - N * LdotN );
     half rough_sq = roughness * roughness;
     half A = 1.0 - 0.5 * (rough_sq / (rough_sq + 0.33));
     half B = 0.45 * (rough_sq / (rough_sq + 0.09));
     // The two dot-products will be in the range of
     // 0.0 to 1.0 which is perfect for a texture lookup:
     half2 tex_coord = half2( VdotN, LdotN ) * 0.5 + 0.5;
     half C = tex2D(gOrenNayarSampler, tex_coord).x;
     result *= (A + B * max(0.0, gamma) * C);
  }
  return result;
}
```

Oren-Nayar Lighting Model

- But we made it even faster with an approximation
- Practically, we didn't notice any badness from this

```
half ComputeOrenNayarLighting_Fakey( half3 N, half3 L, half3 V, half roughness )
{
   // Through brute force iteration I found this approximation. Time to test it out.
   half LdotN = dot( L, N );
   half VdotN = dot( V, N );
   half result = saturate(LdotN);
   half soft_rim = saturate(1-VdotN/2); //soft view dependant rim
   half fakey = pow(1-result*soft_rim,2);//modulate lambertian by rim lighting
   half fakey_magic = 0.62;
   //(1-fakey)*fakey_magic to invert and scale down the lighting
   fakey = fakey_magic - fakey*fakey_magic;
   return lerp( result, fakey, roughness );
}
```

Roughness & Specular Power?

- Only characters ended up using roughness
- Used to store both spec power and roughness separately
  - Needed 2 RTs = expensive
- But aren't they necessarily same?
  - Rougher surface = less tight spec = lower spec power
- So we packed roughness and spec power into one channel
  - Bit 7: roughness flag
  - Bit 6~0: specular power
  - When bit 7 is 1, roughness is inverse curve function of specular power

Many Many Lights?

- In the example scene, we are drawing 31 lights
- Many lights + shadow sounds nice, but slow. Need optimization
- We draw lights in two passes
  - Stencil/hi-stencil masking pixels that should be lit
  - Draw lights again with stencil test
- Light mask
  - Fake shadow
  - Think it as a filter you attach in front of a concert light

WARHAMMER 40,000
SPACE MARINE

Light Mask

Light Mask

Our Lighting Buffer

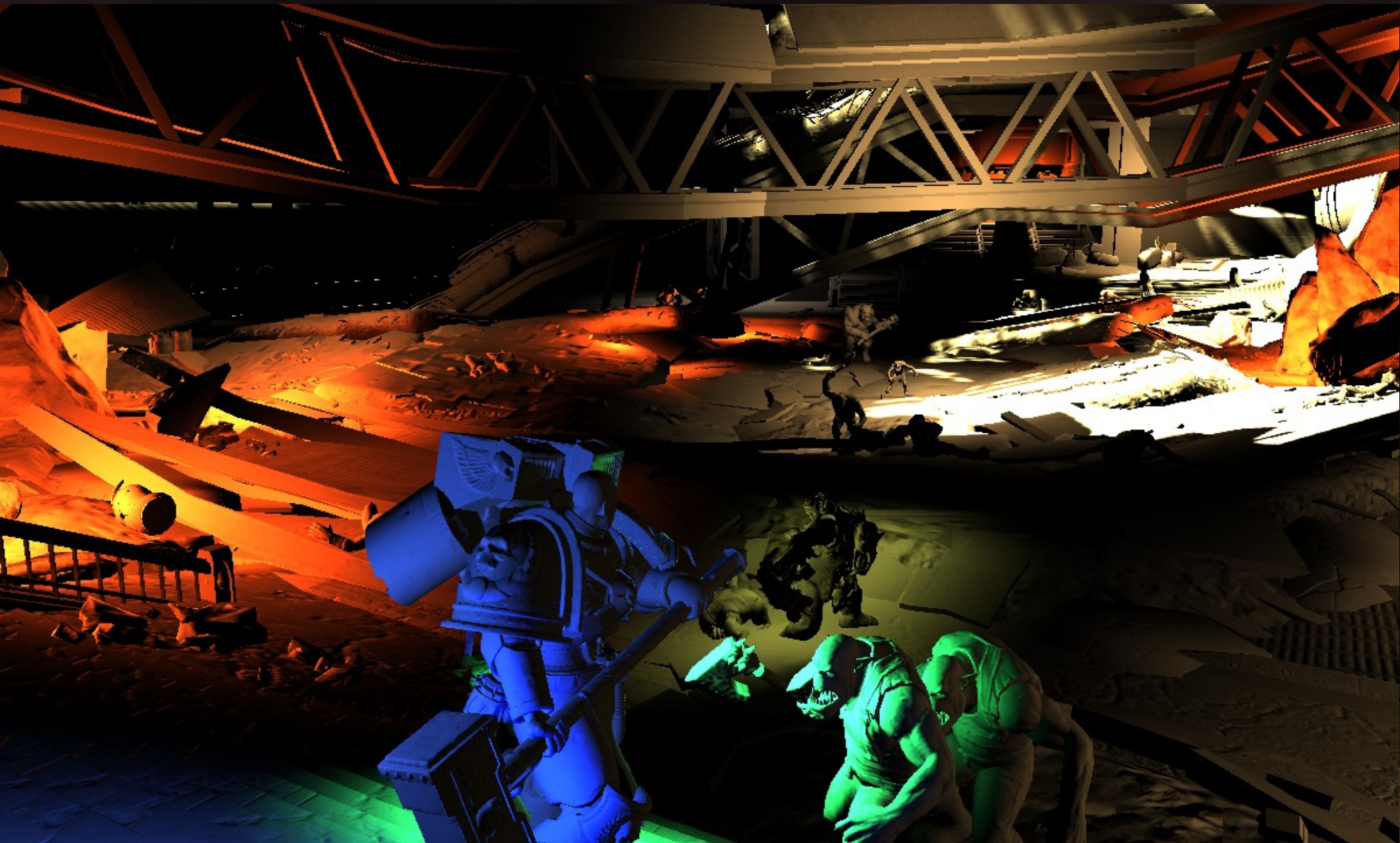
- Used to use two RTs
- Now we use only one A16R16B16G16F RT
  - RGB: diffuse lighting result
  - A: specular lighting result
    - Encoded as luminance
    - Reconstructed in combiner pass
- Reasons for using 16-bit per channel
  - Linear space lighting
  - HDR

WARHAMMER 40,000 SPACE MARINE

Example Scene

WARHAMMER 40,000 SPACE MARINE

Lighting Buffer, Diffuse



Lighting Buffer, Specular

Shadow

- Sun shadow
  - Cascade Shadow Map with 4 cascades
  - Blend between 2 neighbouring cascades to hide sudden transition
- Any lights can have shadows in theory
- 9-sample manual PCF on X360, 3-sample HW PCF on PC and PS3
  - Too expensive, especially with cascade blends
  - Solution: deferred Shadow

Deferred Shadow

- Generates shadow map for 1 cascade
- Draws current cascade on deferred buffer in screen-space
- Draws it in two passes
  - Pass 0: where no more blending needed. Stencil mark
  - Pass 1: where further blending needed with next cascade
- PCF while drawing onto deferred shadow buffer
- Result is like a light map in screen-space

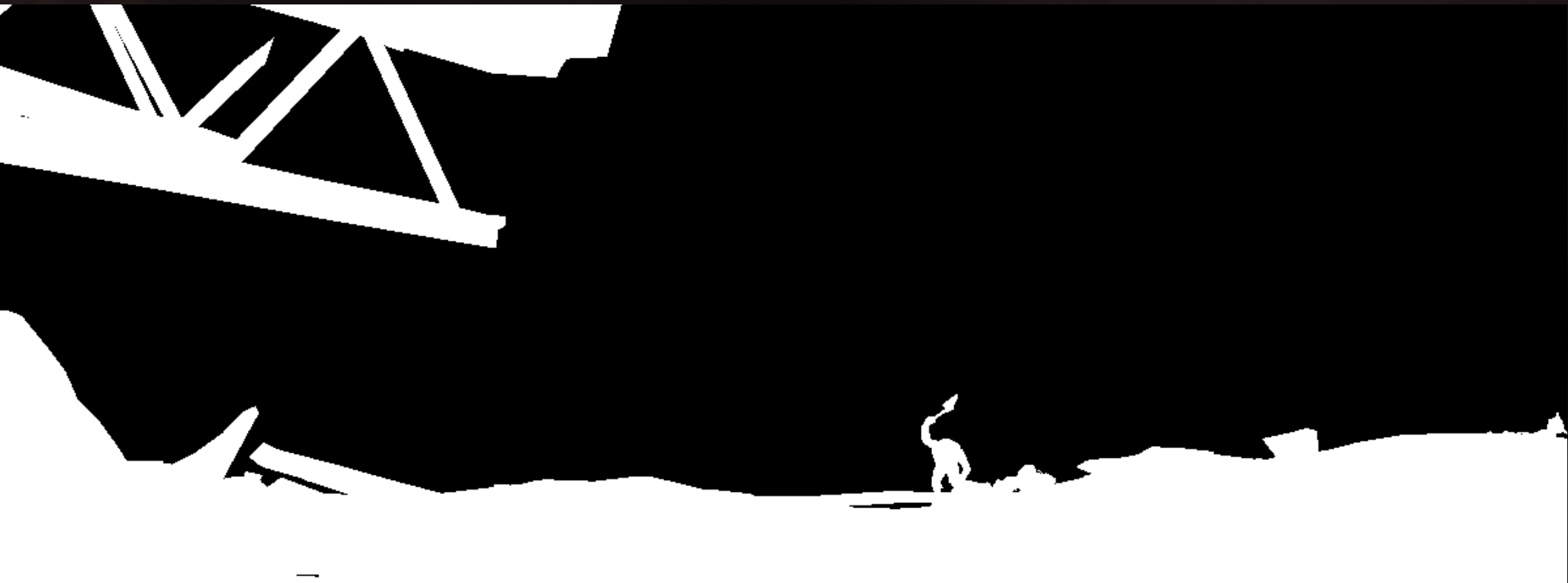
WARHAMMER 40,000 SPACE MARINE

Example Scene

WARHAMMER 40,000 SPACE MARINE

Deferred Shadow Buffer–C2, P0

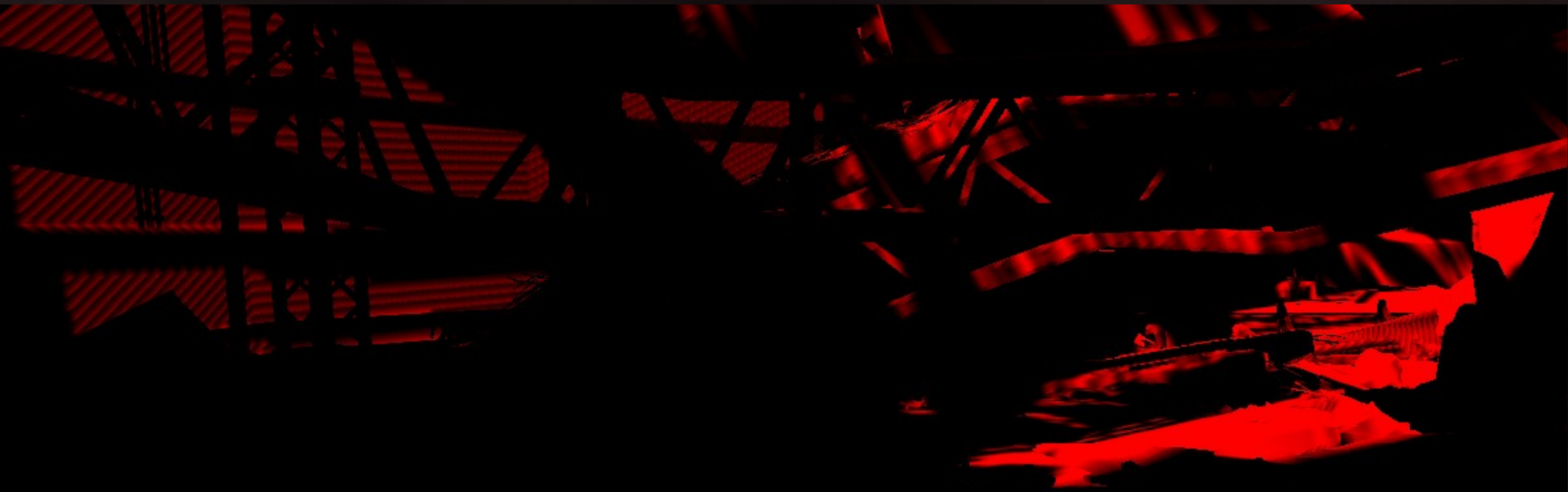
Deferred Shadow Buffer–C2, P0



WARHAMMER 40,000 SPACE MARINE

Deferred Shadow Buffer–C2, P0

Deferred Shadow Buffer–C2, P1



WARHAMMER 40,000 SPACE MARINE

Deferred Shadow Buffer–C1, P0

WARHAMMER 40,000 SPACE MARINE

Deferred Shadow Buffer–C1, P0

Better Example

Better Example

Better Example

WARHAMMER 40,000 SPACE MARINE

Better Example

Better Example

Better Example

Better Example

WARHAMMER 40,000 SPACE MARINE

Better Example

Combiner Pass

- Another geometry pass that combines
  - Lighting result
  - Two ambient colours based on Ambient Occlusion factor
  - Ambient saturation
  - Character fill light
- Everything is written in linear space in 16-bit per channel

WARHAMMER 40,000
SPACE MARINE

Combiner Pass

Character Fill Light

- Used rim light once
- What we wanted: making characters pop out in the shadow
- So borrowed the concept of fill lights from movie industry
- First implementation: 3 directional lights on characters
  - Based on camera angle (top, left and right)
  - Turns out to be too expensive
- Shipped (optimized) version: 2 directional lights
  - One for diffuse only
  - The other for spec only
  - Blended in based on diffuse lighting

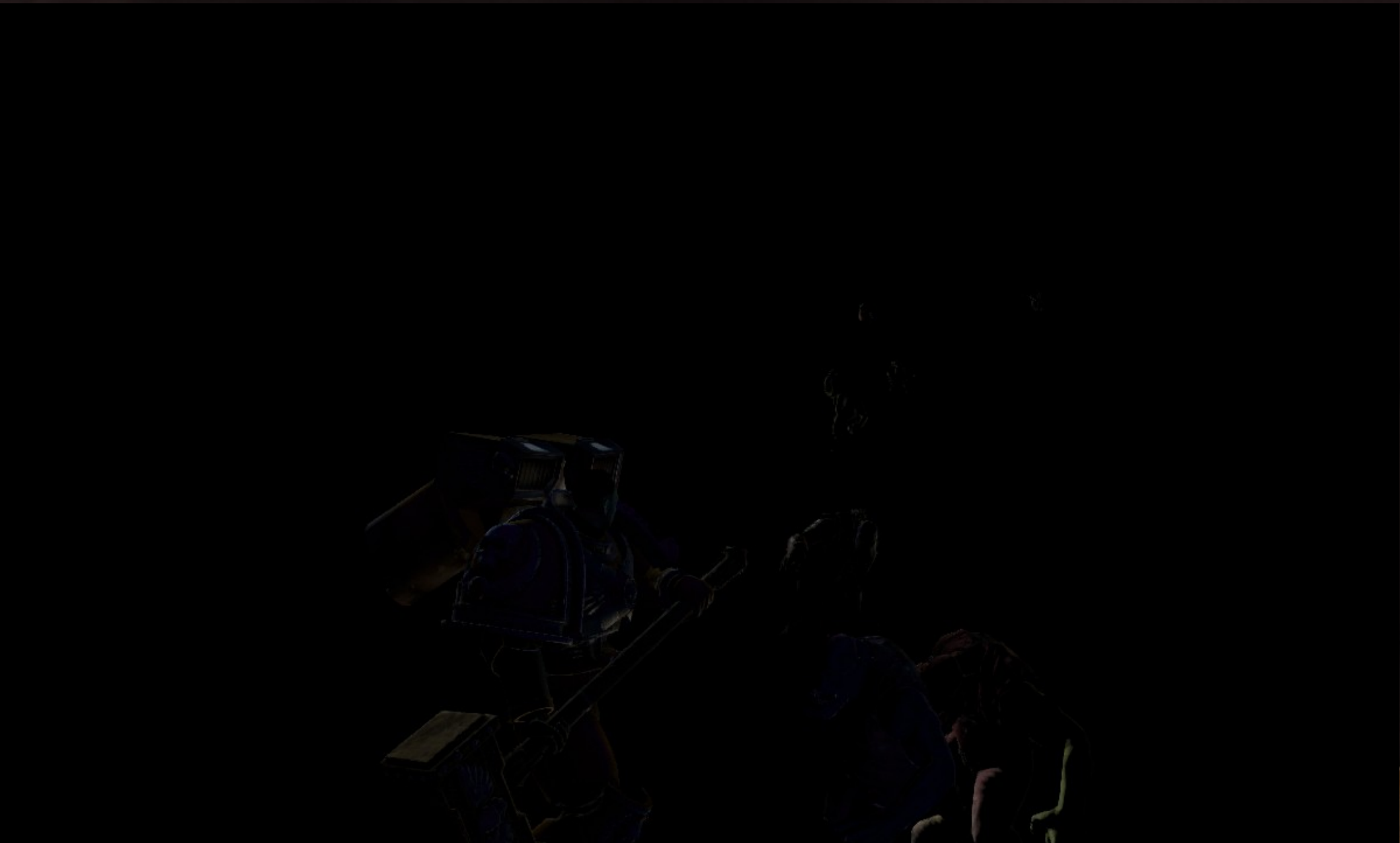
WARHAMMER 40,000 SPACE MARINE

Example Scene

WARHAMMER 40,000 SPACE MARINE

Example Scene(no Fill Light)

Example Scene (Difference)



Ambient Colour / Saturation

- We have two ambient colours
- Based on occlusion depth, we blend between these two colours (think them as deep and shallow shadow colours)
- However, sometimes we want diffuse texture to stand out
  - Especially on Space Marines
  - We call it ambient saturation
- We have two global ambient saturation each for:
  - Environments
  - Character

```
unlitColour *= lerp(lum(diffuse),diffuse,saturation);
```

Overcoming Cons of Deferred Lighting

- Transparent Objects
  - We used hard-alpha (alpha testing) as much as possible
  - Some special treatment for hair
  - If it's a co-planar object, we use a SSD
- Anti-aliasing
  - Hard with DX9 level GPUs
  - made something similar to FXAA

Ambient Occlusion

Pass	Budget(ms)
Depth-Pre	0.50
G-Buffer + Linear Depth	5.05
Ambient Occlusion	2.25
Lighting	8.00
Combiner Pass	5.00
Blend	0.15
Gamma Conversion	1.30
FX	2.75
Post Processing	3.70
UI	0.50
Total	29.20

Ambient Occlusion

- Will explain later in detail

Gamma Conversion

Pass	Budget(ms)
Depth-Pre	0.50
G-Buffer + Linear Depth	5.05
Ambient Occlusion	2.25
Lighting	8.00
Combiner Pass	5.00
Blend	0.15
Gamma Conversion	1.30
FX	2.75
Post Processing	3.70
UI	0.50
Total	29.20

Gamma Conversion

- We do lighting and combiner in linear space (HDR)
- In this step, we separate HDR image into LDR and bloom intensity
- LDR image is stored in RGB (in gamma space)
- Bloom intensity is stored in Alpha channel
  - Based on pixel luminance and threshold set by artists
  - Later used in bloom pass
- For more about linear lighting, please take a look at this presentation by John Hable:
  - http://filmicgames.com/Downloads/GDC_2010/Uncharted2-Hdr-Lighting.pptx

WARHAMMER 40,000 SPACE MARINE

Gamma Conversion (LDR)

Gamma Conversion(HDR Intensity)

FX(Particle) Pass

Pass	Budget(ms)
Depth-Pre	0.50
G-Buffer + Linear Depth	5.05
Ambient Occlusion	2.25
Lighting	8.00
Combiner Pass	5.00
Blend	0.15
Gamma Conversion	1.30
FX	2.75
Post Processing	3.70
UI	0.50
Total	29.20

FX(Particle) Pass

- Relic is known for the crazy amount of particles
- Additive, blend, distortion, soft particles, lit particles...
- Mesh or Decal spawn
- Particle updates and submission, heavily multi-threaded
- God rays
- Dynamic FX lights
  - Can have shadow -> expensive
  - A lot of time, we use light mask
- Optimization:
  - Half-res particles when FPS is slower
  - Batch through pre-multiplied alpha and texture page
- Too big of a system to fit in this presentation, but is a collection of standard industry practices and techniques

WARHAMMER 40,000 SPACE MARINE

Without FX

WARHAMMER 40,000 SPACE MARINE

With FX

Post Processing

Pass	Budget(ms)
Depth-Pre	0.50
G-Buffer + Linear Depth	5.05
Ambient Occlusion	2.25
Lighting	8.00
Combiner Pass	5.00
Blend	0.15
Gamma Conversion	1.30
FX	2.75
Post Processing	3.70
UI	0.50
Total	29.20

(Uber) Post-Processing

- Uber pass processing almost all the post-processing at once
- Mostly to save Resolve cost on Xbox 360
  - Generate Distortion Vector Texture
  - Generate Depth-of-Field + Camera Motion Blur
  - Generate HDR Bloom Texture
    - Colour Correction
    - AA
    - Vignette

Generate Distortion Vector Texture

- Nothing special
- All from FX
- 2D UV offset is stored in screen-space
- Later this offset is used when combining all post-processing effect with the frame buffer

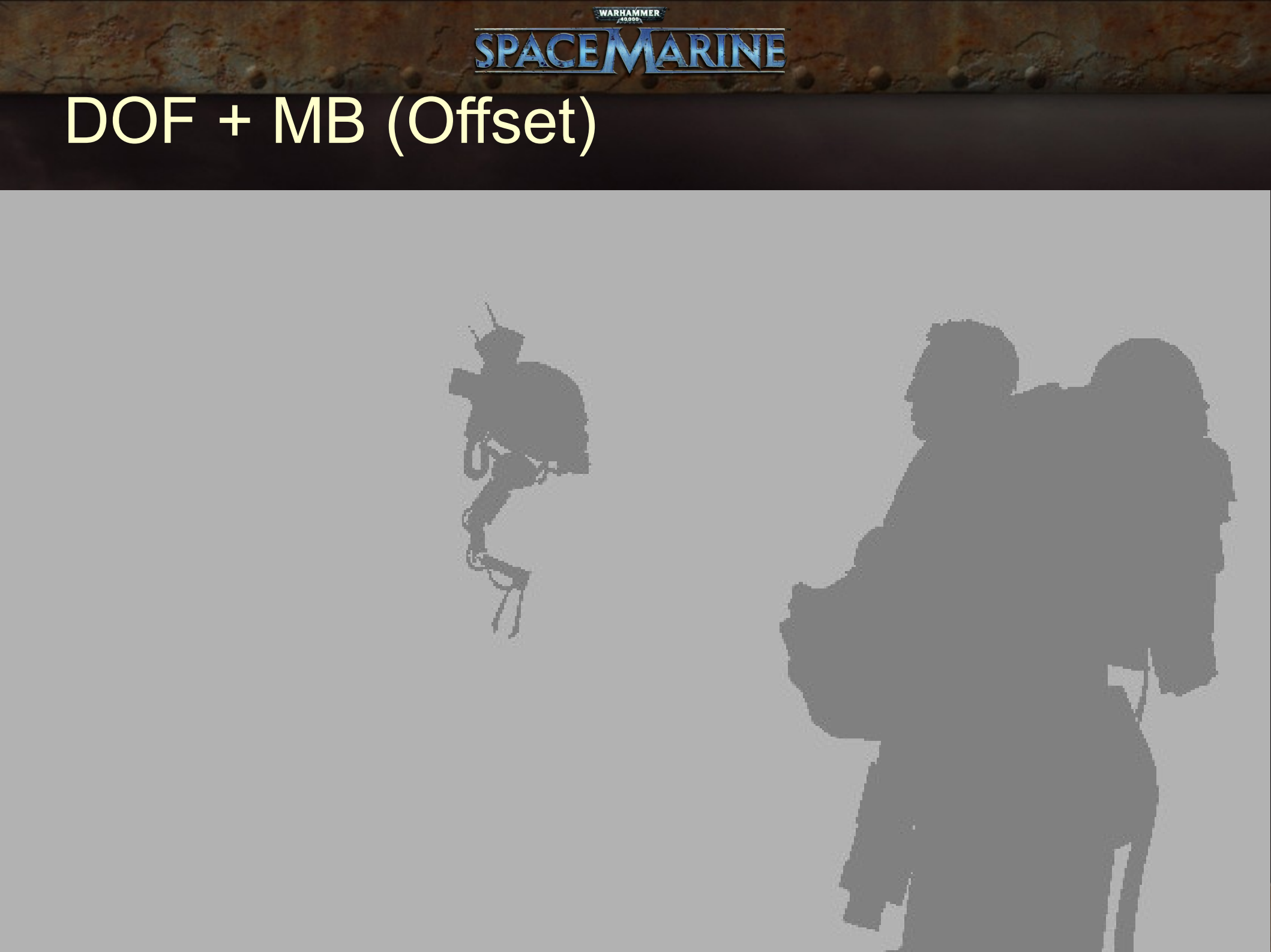
Generate DOF + MB

- Both Depth-of-Field and Camera Motion Blur are blurry
- So let's process both at the same time
  - Pass 0: generates a blur offset based on distance to camera and camera motion blur
  - Pass 1: separable masked gaussian blur
  - Pass 2: separable masked gaussian blur + store mask into Alpha channel
- All done in half-res
- Later mask value (in Alpha channel) is used to blend between this texture and the full-res frame buffer

WARHAMMER 40,000 SPACE MARINE

DOF + MB (Before)

DOF + MB (Offset)



DOF + MB (Pass 1)

WARHAMMER 40,000 SPACE MARINE

DOF + MB (Pass 2)

Generate HDR Bloom Texture

- Nothing special
- Close to Halo 3 way
- We don't do tone mapping
- Done in 3 passes with half-res frame buffer
  - ½ downsample and blur
  - ½ downsample again and blur
  - Merge back to ¼ res buffer and blur

Colour Correction

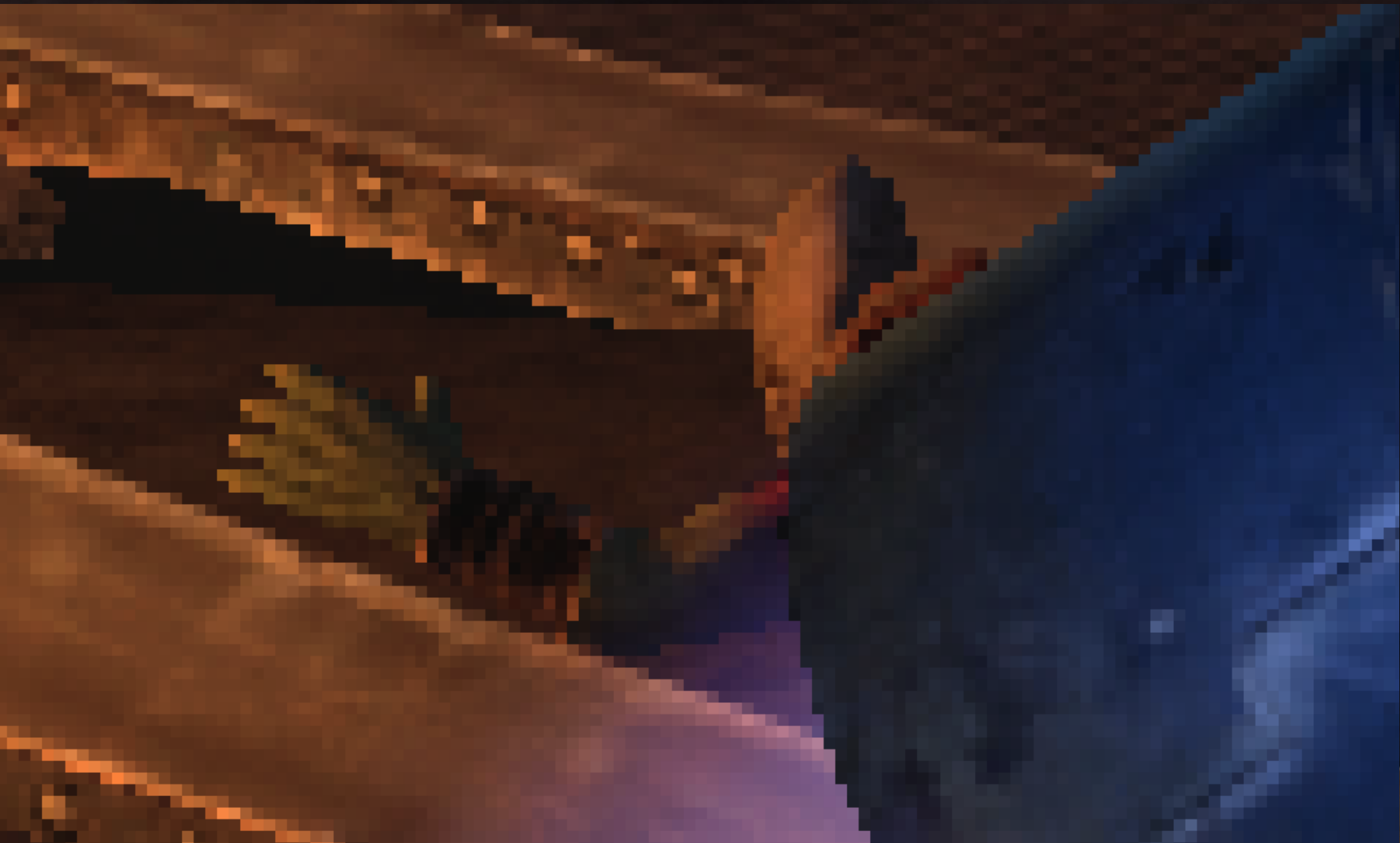
- Originally implemented for a previous unreleased project
- We use a 32x32x32 3D texture
  - Each dimension represent a channel's input value
  - Value stored in the texture replaces the input value
- We created a Photoshop plug-in to help authoring the colour grade texture
  - Take a screenshot from game
  - Load the image in to Photoshop
  - Use whatever layer modifiers to male it look good
  - Press the Magic button
  - Done!

Anti-Aliasing

- Simplified version of FXAA using just 4 samples to find the slope of the luminance edge
- Use the slope to do a 3 pixel blending between blurred and unblurred versions of the textures
- Total 7 texture look-ups

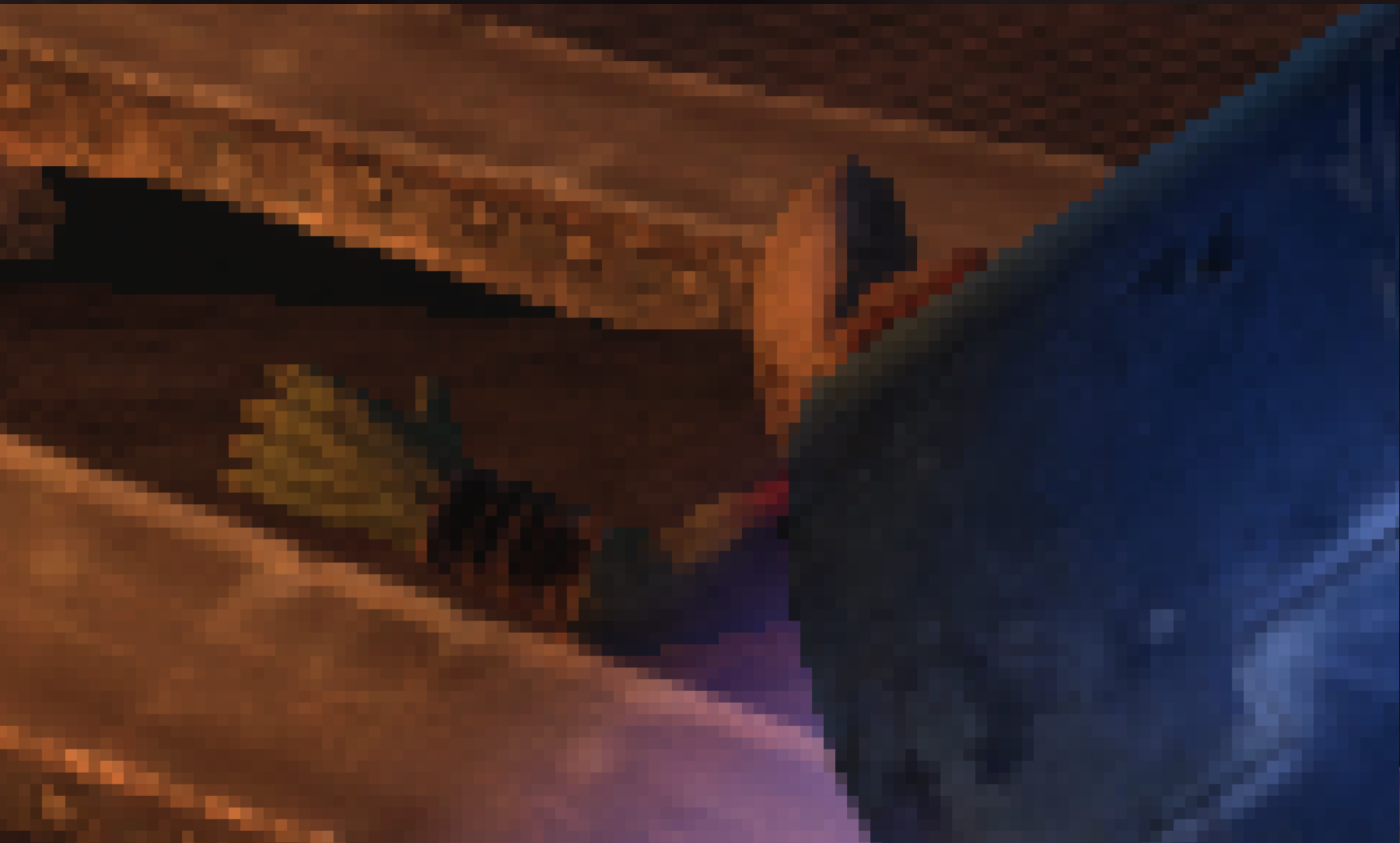
WARHAMMER 40,000 SPACE MARINE

Antialiasing (No AA)



WARHAMMER 40,000 SPACE MARINE

Antialiasing



WARHAMMER 40,000
SPACE MARINE

Before Post-Processing

WARHAMMER 40,000 SPACE MARINE

After Post-Processing

UI

Pass	Budget(ms)
Depth-Pre	0.50
G-Buffer + Linear Depth	5.05
Ambient Occlusion	2.25
Lighting	8.00
Combiner Pass	5.00
Blend	0.15
Gamma Conversion	1.30
FX	2.75
Post Processing	3.70
UI	0.50
Total	29.20

UI

- Nothing much to talk about
- We use Scaleform

First Half is Over!!

Questions so far?

Agenda

1. Deferred Lighting
2. **Screen Space Decals**
3. World Occlusion

Traditional Decals

- Example: Bullet Hole
  - Get the underlying collision geometry at collision point
  - Walk through the geometry to find out the smallest patch possible to represent the bullet hole
  - Make a mesh patch with the vertices found in step 2
  - Draw the underlying geometry
  - Draw the patch with the bullet hole textures
- Problems
  - Decal stretching
  - Collision detection is expensive
  - Waste of vertex space: a plane cannot be simply 4 verts anymore

Screen Space Decals (SSDs)

- We already have all surface information in screen-space
  - Normals
  - Depth
  - Position (driven from depth info)
- So why don't we use these info instead of making a mesh patch?
- Benefits:
  - Our collision meshes are super simple
  - Our Havok broadphase is super fast

SSD Inspiration

SSD Inspiration

Screen Space Decals

- Same Example: Bullet Hole
  1. Draw underlying visual geometries onto scene
  2. Rasterize a SSD box at the collision point
  3. Read the scene depth for each pixel
  4. Calculate 3D position from the depth
  5. If this position is outside of the SSD box, reject
  6. Otherwise, draw the pixel with the bullet hole texture

Screen Space Decals

- Not just for FX
- It's literally used everywhere:
  - Blood splat on the wall or ground
  - Stone wall decoration
  - Burnt mark on concrete slate
  - Ork painting
  - Rubble piles on the ground
  - Bullet holes
  - Explosion damage
  - ……and list goes on
- Yup, environment artists love it!

WARHAMMER 40,000 SPACE MARINE

Scene with SSD

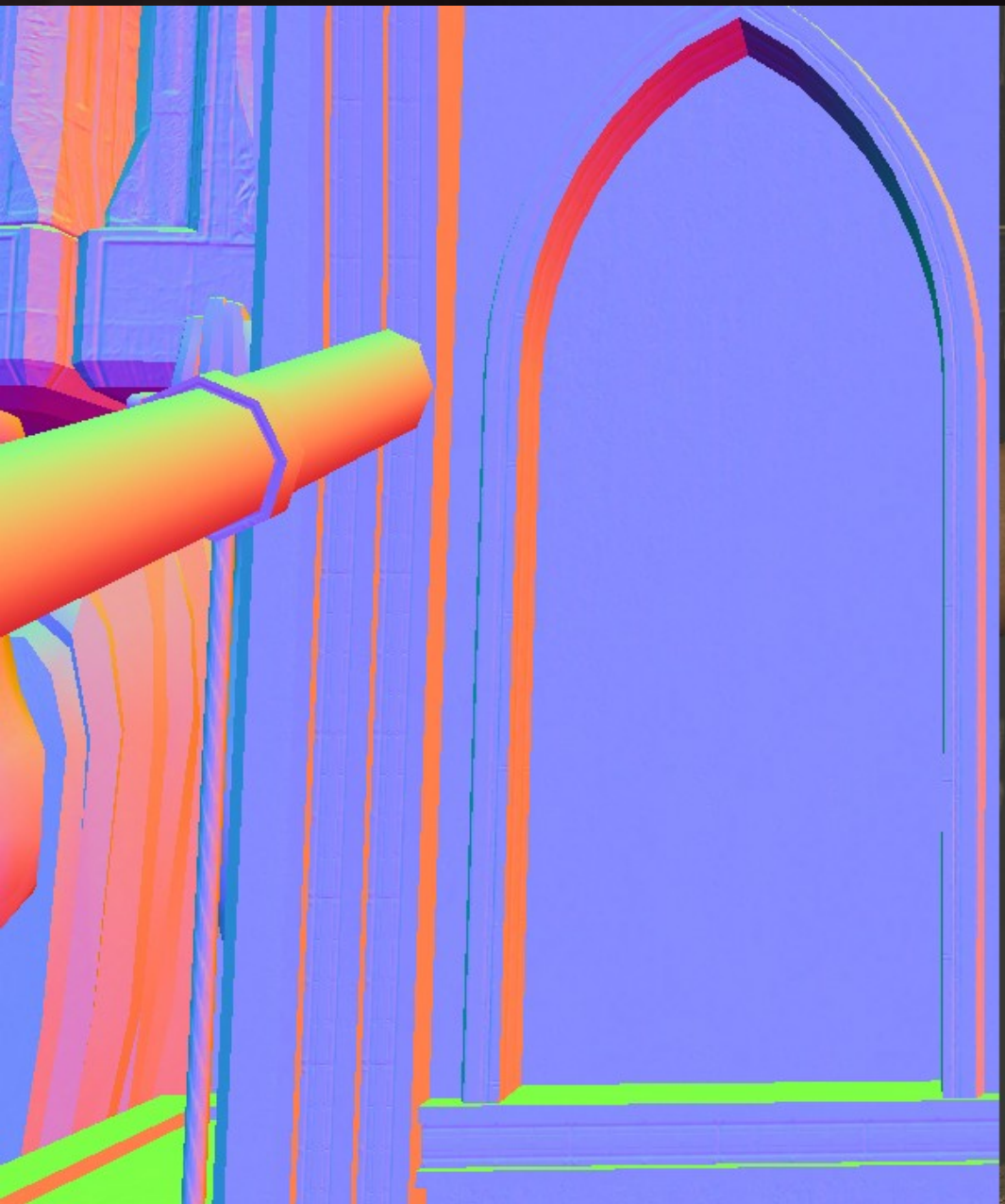
WARHAMMER 40,000 SPACE MARINE

Scene without SSD

Screen Space Decals

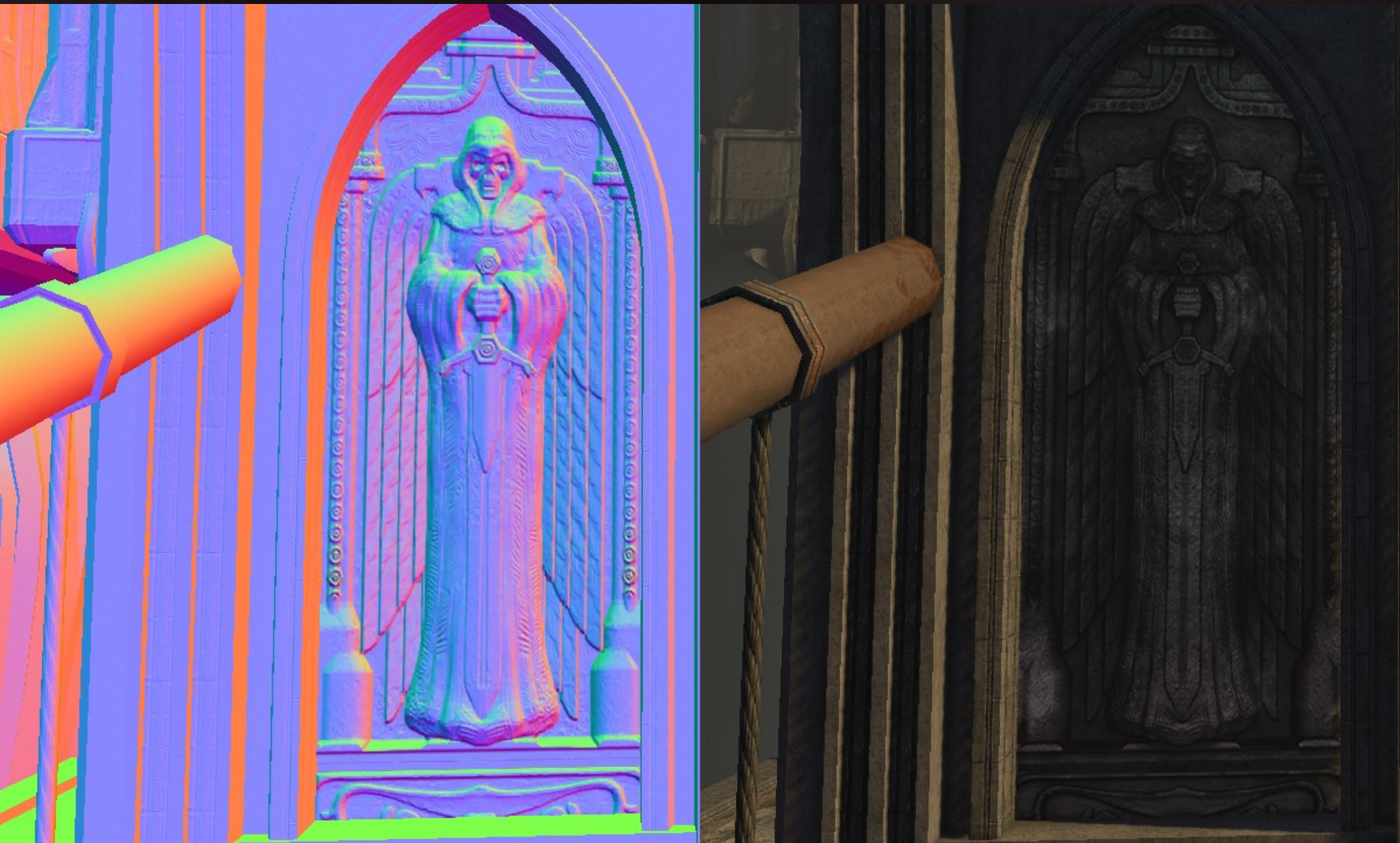
- SSD drawing is split into two passes
  - in G-Buffer: update scene normals
  - in Combiner: combine lighting result and surface materials
- Gives artists a lot of freedom
  - Normal-only decals (e.g, stone wall decoration)
  - Colour-only decals (e.g, blood splat)
  - Normal + colour decals (e.g, thick ork painting)
- All the lighting & ambient occlusion are FREE

Normal Only SSD

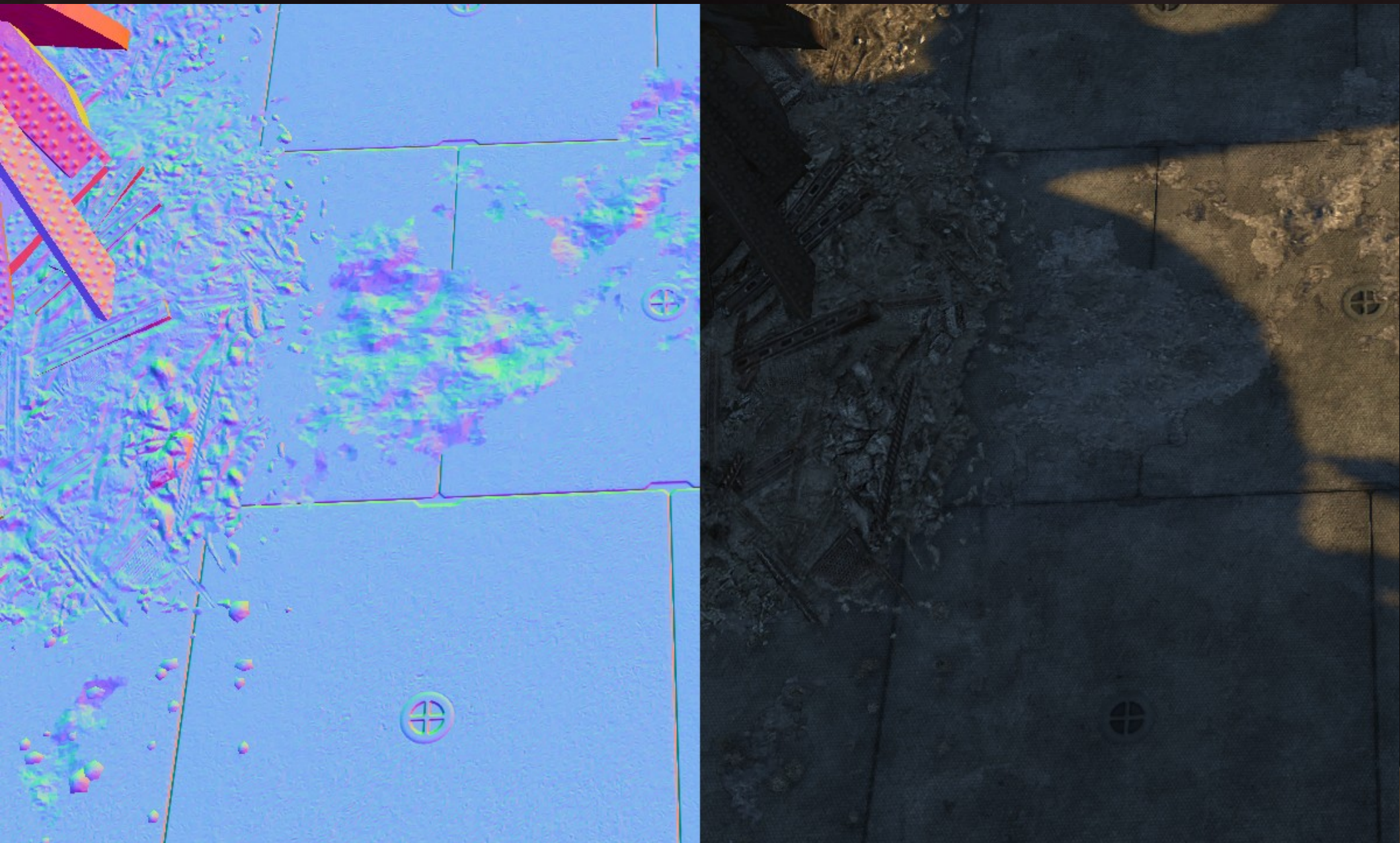


WARHAMMER 40,000 SPACE MARINE

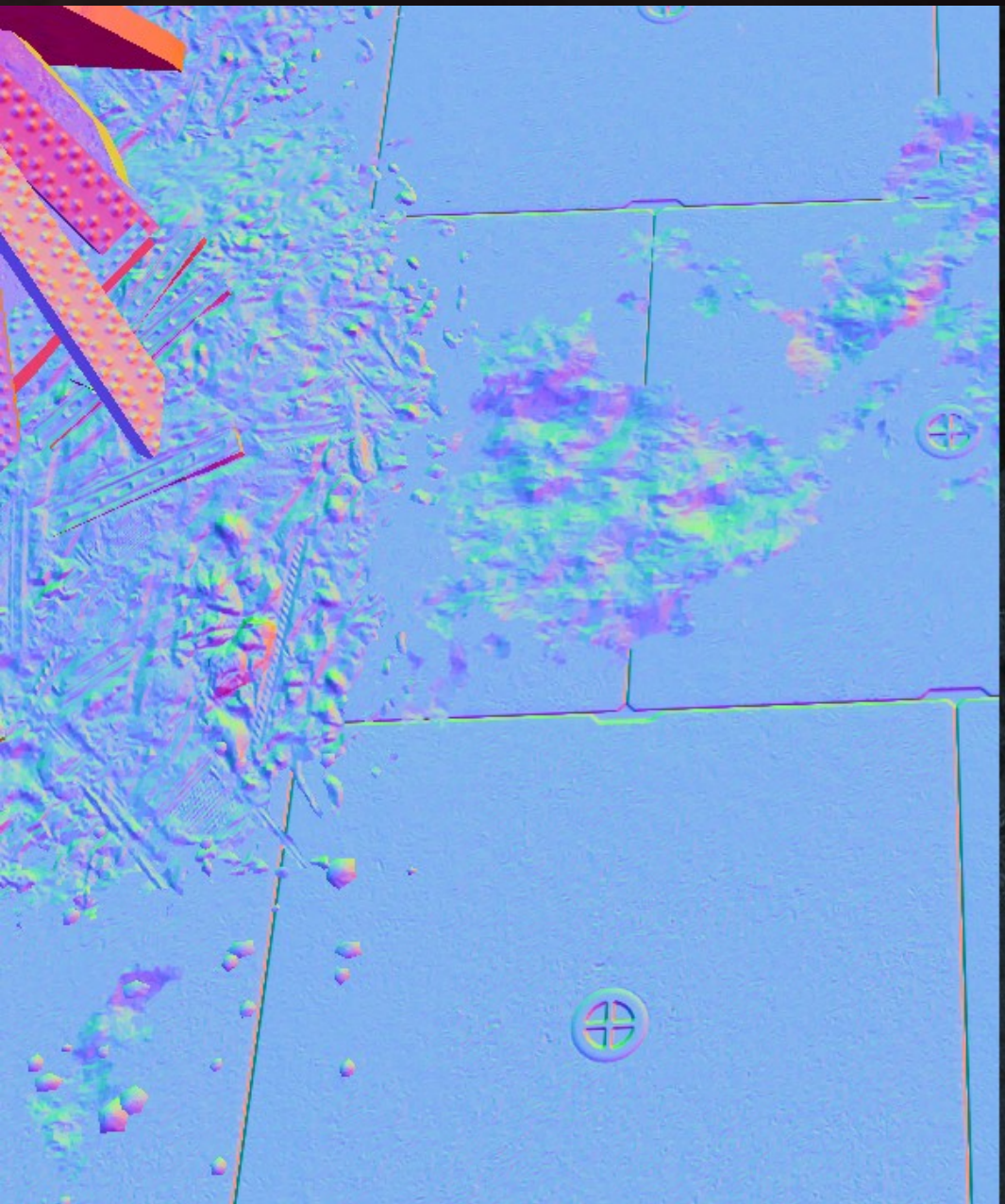
Normal-only SSD



Colour-only SSD

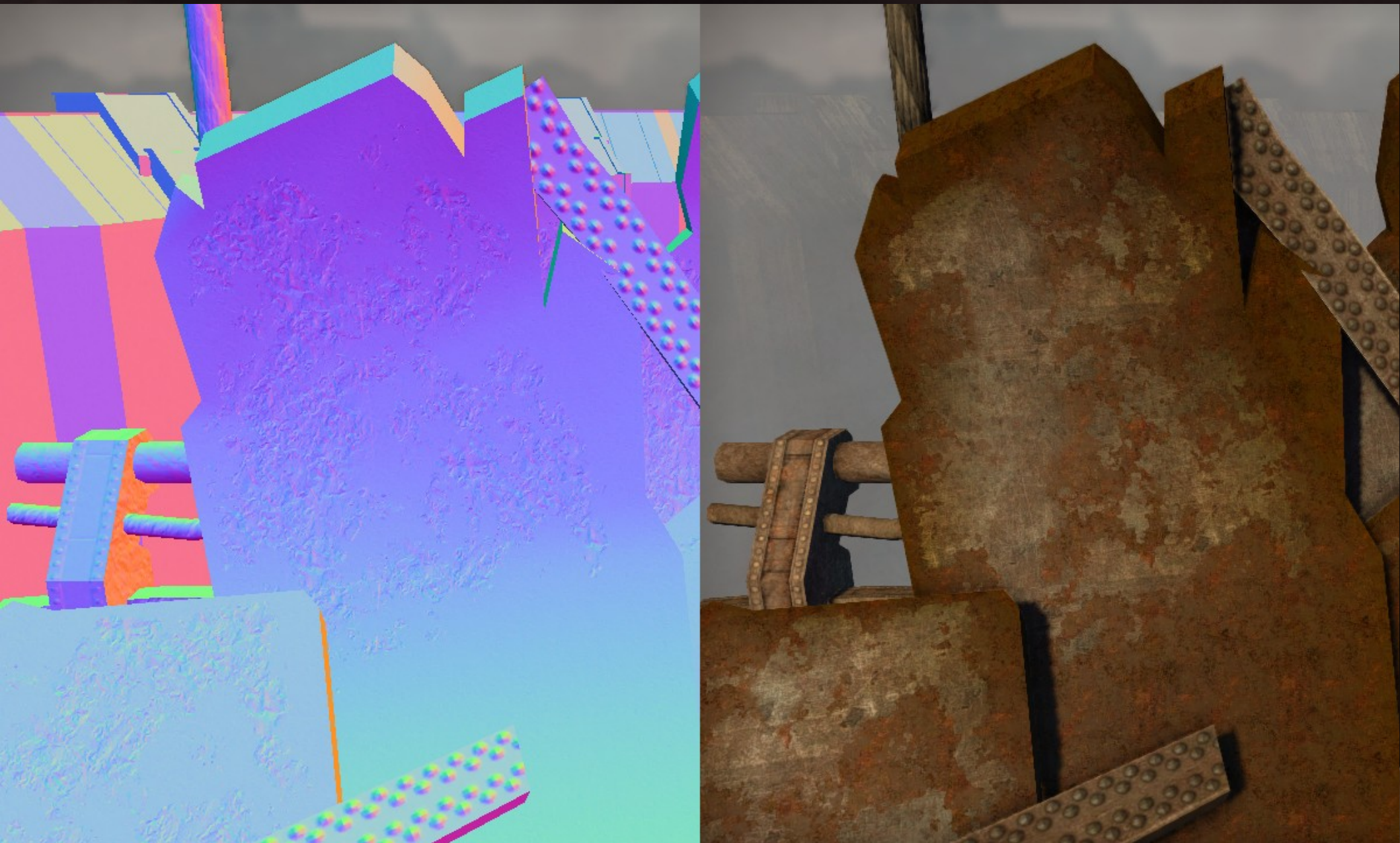


Colour-only SSD

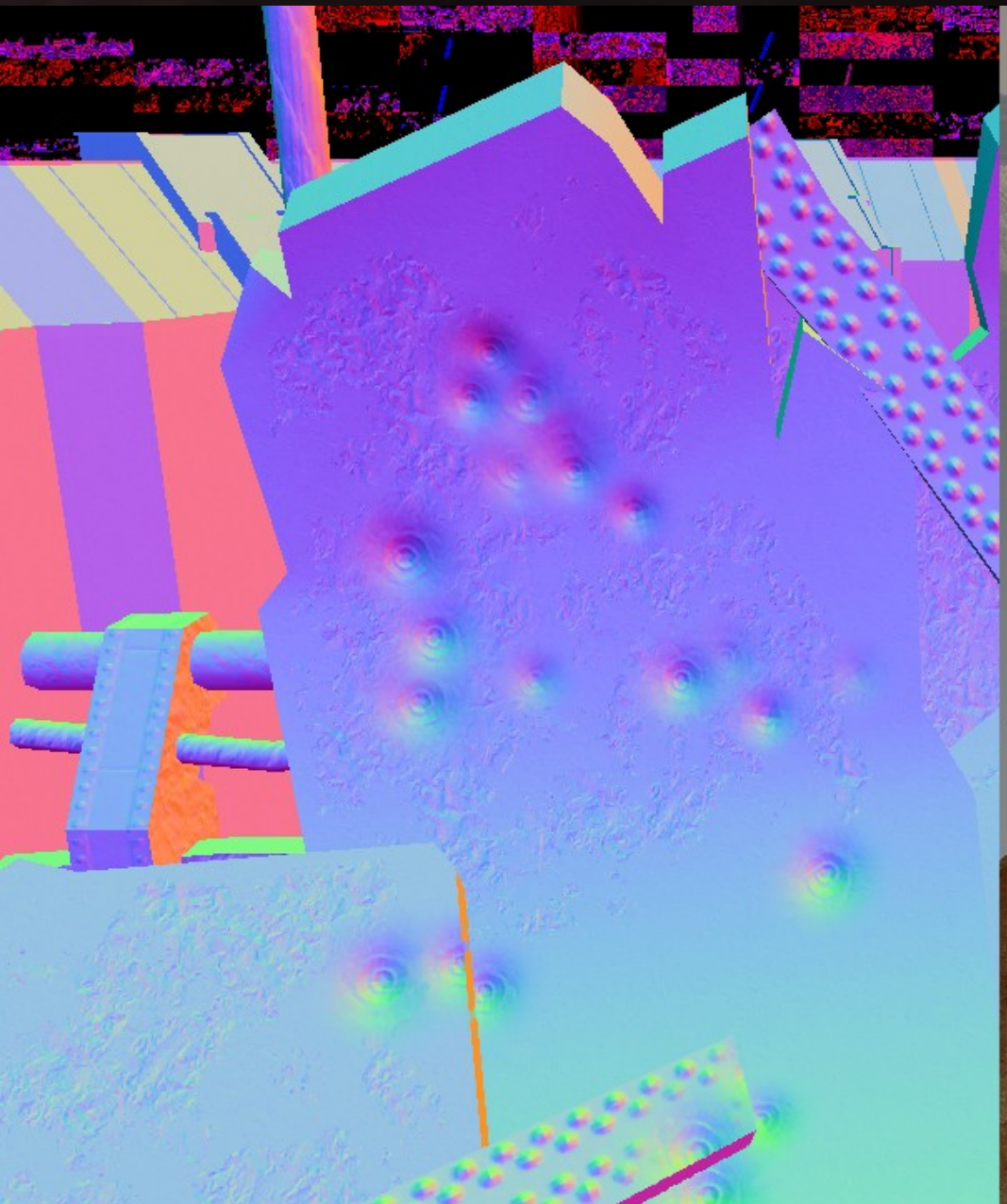


WARHAMMER 40,000 SPACE MARINE

Normal + Colour SSD



Normal + Colour SSD



SSD - Problems

- Stretching
- Moving objects
- Performance

SSD - Stretching

SSD - Stretching

- Problem
  - Caused by projection boxes not fully submerged into underlying geometry
  - Round objects have similar problem
- Solution
  - Using 3D textures was an option, but expensive
  - Instead, we reject pixels with normal threshold
    - Get normal from underlying surface
    - Dot-product this with SSD box's orientation
    - If it's over an artist-set threshold, reject

SSD – Moving Objects

- Problem
  - If an object is moving through a SSD projection volume, you will see SSDs sliding across the object's surface
- Solution
  - Stencil masking moving objects
  - All skinned or dynamic objects don't receive decals by default
  - Artists can override it (some skinned objects are not moving. e.g, props in healthy state)

WARHAMMER 40,000 SPACE MARINE

SSD – Moving Objects

SSD – Moving Objects

SSD - Performance

- Problem
  - Box projection in screen-space
  - Can only reject in pixel shader
  - If a SSD box is covering full-screen, it's same as a full-screen draw pass
- Solution
  - Make decals thin, but not too thin
  - Utilize Hi-Z as much as possible for early rejection
  - Doesn't solve FX decals' problem, but that's fine

Thick SSD = BAD

Thin SSD = GOOD

Agenda

1. Deferred Lighting
2. Screen Space Decals
3. **World Occlusion**

Ambient Occlusion

- Simulates occluded and bounced lights
- We use two techniques
  - World Occlusion
    - Our own approach
    - Based on ambient occlusion technique used in COH
    - Vertical occlusion in world space
- SSAO
  - Same as other games
  - Depth-only comparison in screen space

WARHAMMER 40,000 SPACE MARINE

World Occlusion

World Occlusion Only

World Occlusion (Old Version)

- Uses vertical distance between objects as attenuation factor
- Proxy simplified objects are rendered on a depth only target
- This normalized value is then blurred
- This blurred value is used to blend between two ambient light values
  - (^_^) Works great for outdoor scenes
  - (T_T) For interior scenes, not so much

World Occlusion (New Version)

- Solves the problem of interior scenes
- No more, the highest distance to the ground
- Depth peeling to allow up to 4 different depth layers
- Rendered using a top-down view camera

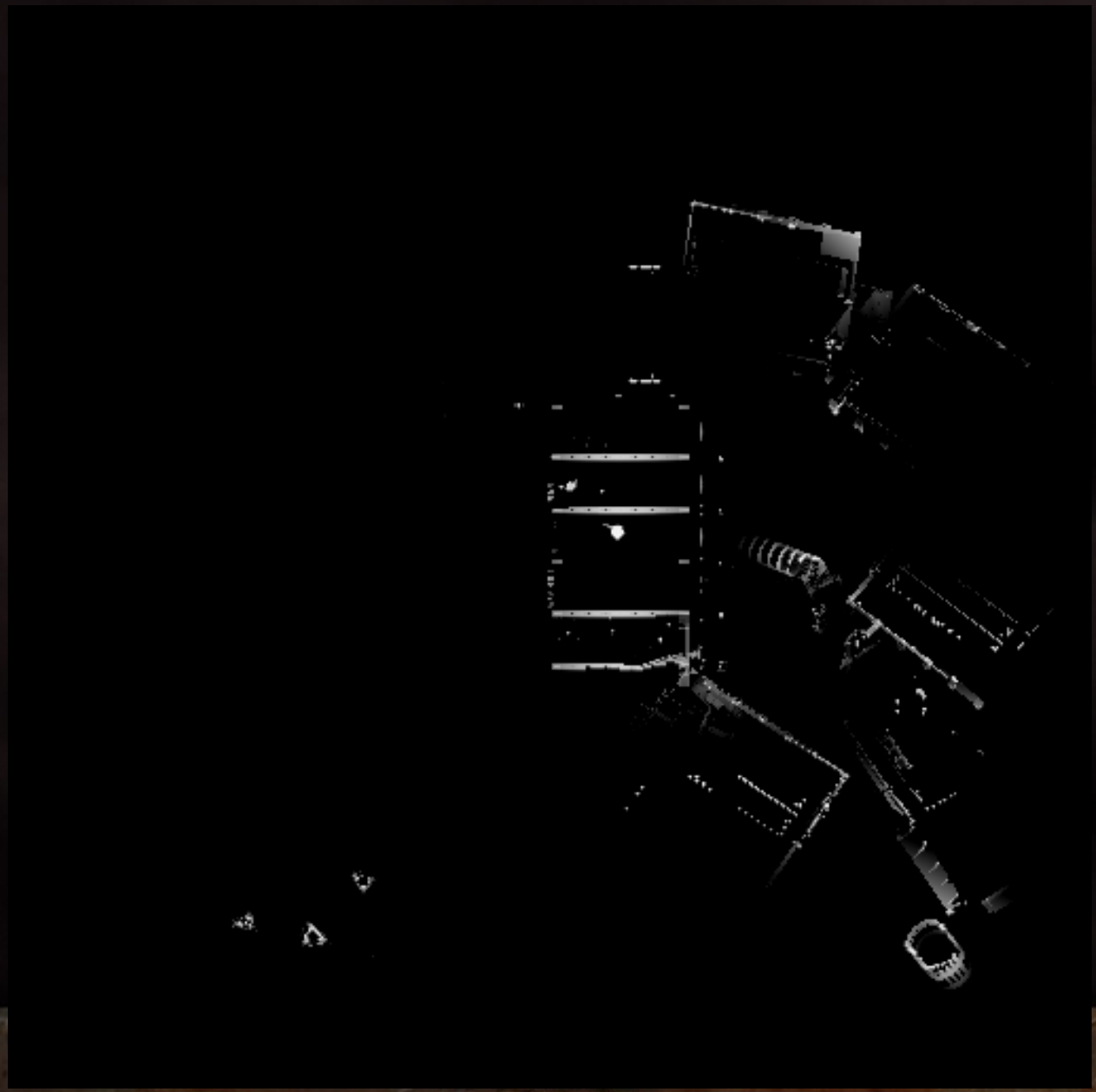
World Occlusion

- Then world occlusion texture is blurred
  - At different amounts for each level
  - So that vertical walls can cast/darken ambient light at the crevices with the ground
  - This fakes horizontal bleeding of world occlusion

World Occlusion (Top Layer)

World Occlusion

World Occlusion



World Occlusion (Bottom Layer)

Applying World Occlusion

For each pixel on the screen:

1. Compute the world position of the pixel
2. Find out in what depth layer of world occlusion this pixel is
3. Compute the distance to the corresponding value stored in the world occlusion texture
4. Use normalized result to blend between the low and high ambient colour values (same time as SSAO)

World Occlusion

World Occlusion (Optimization)

- Problems
  - Drawing all objects 4 times is slow
  - Applying it in full-screen is slow
- Solutions
  - Don't generate world occlusion map every frame
  - Artists tag certain objects as World Occlusion contributors
  - Layer heights are controlled by artists per region
  - Use a half-res render texture when applying

SSAO

- A lot of trials and errors
  - Toy Story 3(game) way: way too noisy
  - Improved version of Crytek way: noisy co-planar surface
  - Simplified volumetric SSAO: too many samples
- Our final approach was rather simple
  - Depth-only comparison
  - Circular neighbour samples: 16 for X360, 9 for PS3
  - No filtering of any kind

WARHAMMER 40,000 SPACE MARINE

SSAO

References

- [Timothy11] FXAA: http://timothylottes.blogspot.com/2011/03/nvidia-fxaa.html
- [Aras09] Deferred Cascade Shadow Maps http://aras-p.info/blog/2009/11/04/deferred-cascaded-shadow-maps/
- [Wolf08] Light Pre-Pass renderer: http://diaryofagraphicsprogrammer.blogspot.com/2008/03/light-pre-pass-renderer.html
- [Val07] Deferred Rendering in Killzone 2: http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf
- [Jeremy05] Using Lookup Tables to Accerlate Color Transformations: http://http.developer.nvidia.com/GPUGems2/gpugems2_chapter24.html

Acknowledgements

- Space Marine rendering programmers:
  - Ian Thomson
  - Karl Schmidt
  - Rod Reddekopp
  - Angelo Pesce
- Space Marine artists for producing amazing content and pushing us to do more work!

Got Questions?

- Pope Kim
  - blindrenderer@gmail.com
  - Twitter
    - Korean: BlindRendererKR
    - English: BlindRenderer
- Daniel Barerro
  - daniel.barrero@gmail.com