SIGGRAPH2010

The People Behind the Pixels

CryENGINE 3:
reaching the speed of light

Anton Kaplanyan
Lead researcher at Crytek

Agenda

- Texture compression improvements
- Several minor improvements
- Deferred shading improvements

TEXTURES

Agenda: Texture compression improvements

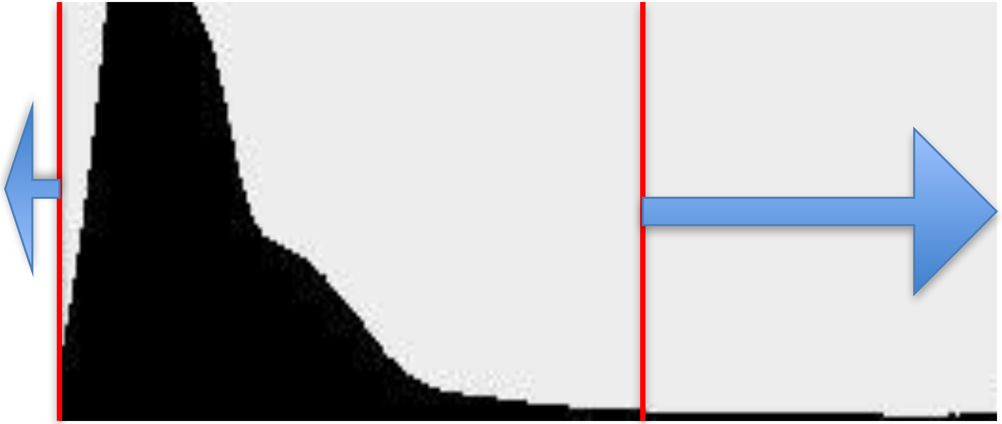
1. Color textures
   - Authoring precision
   - Best color space
   - Improvements to the DXT block compression
2. Normal map textures
   - Normals precision
   - Improvements to the 3Dc normal maps compression

Color textures

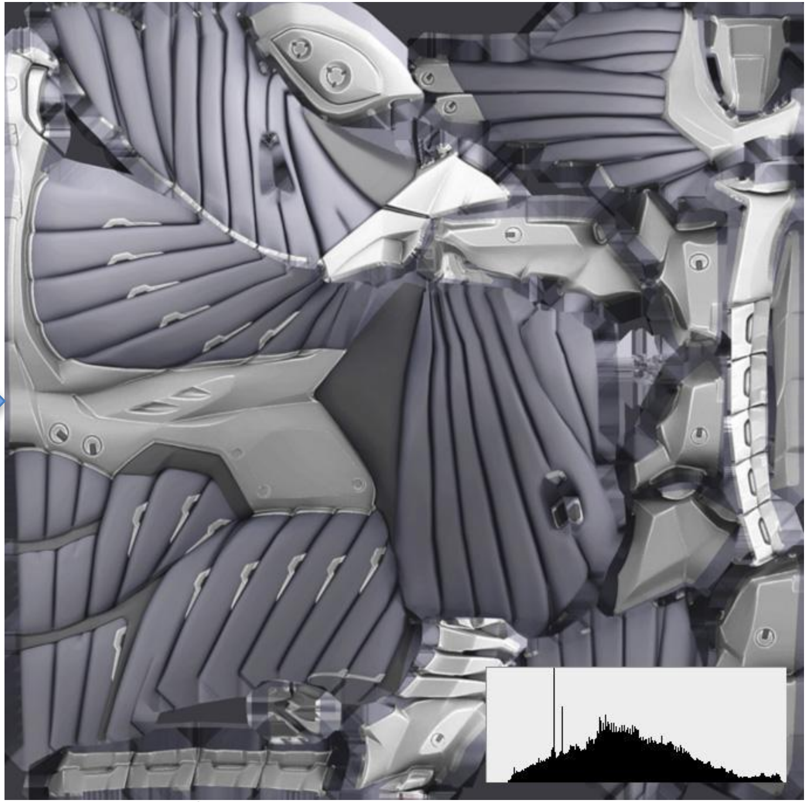
- What is color texture? Image? Albedo!
  - What color depth is enough for texture? 8 bits/channel?
  - Depends on lighting conditions, tone-mapping and display etc.
- **16-bits/channel** authoring is a **MANDATORY**
  - Major authoring tools are available in Photoshop in 16 bits / channel mode
- All manipulations mentioned below **don’t make sense** with 8 b/channel source textures!

Histogram renormalization

- Normalize color range before compression
  - Rescale in shader: two more constants per texture
  - Or premultiply with material color on CPU



Histogram renormalization



Histogram renormalization example

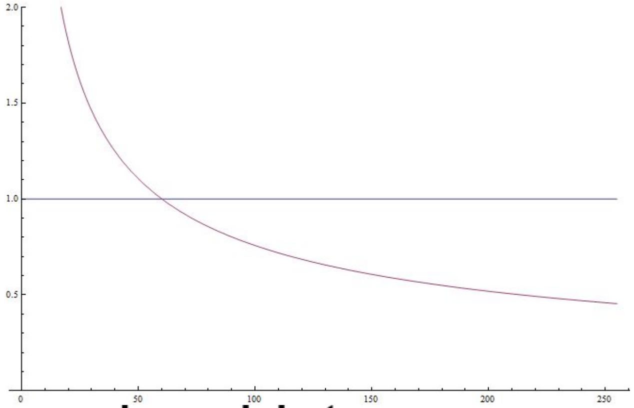
DXT w/o renormalization

DXT with renormalization

Gamma vs linear space for color textures

- Two h/w color spaces for free: linear/gamma
  - Solution to the equation

$$x_\gamma{}' = x'$$



  - Median (**linear space**):

$$x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{\mathbf{255}}$$

  - Choose the right color space based on histogram:
  - <u>Rule of thumb:</u> use linear if >75% of pixels are above the median

Gamma vs linear space on Xbox 360

- Two h/w color spaces for free: linear/gamma
  - Solution to the equation

$$x_\gamma{}' = x'$$

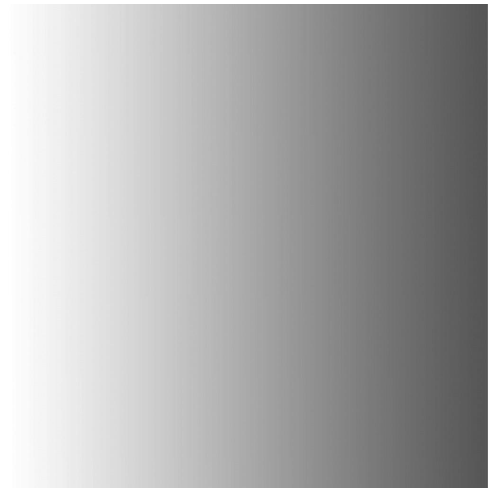
  - Median (**linear space**):

$$x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{\mathbf{255}}$$

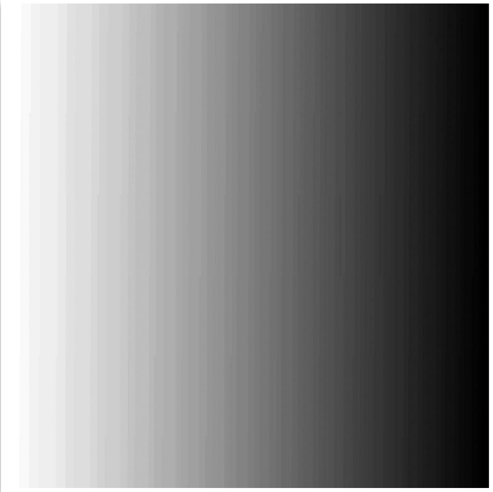
  - Choose the right color space based on histogram:
  - Rule of thumb: use linear if >75% of pixels are above the median

Gamma / linear space example

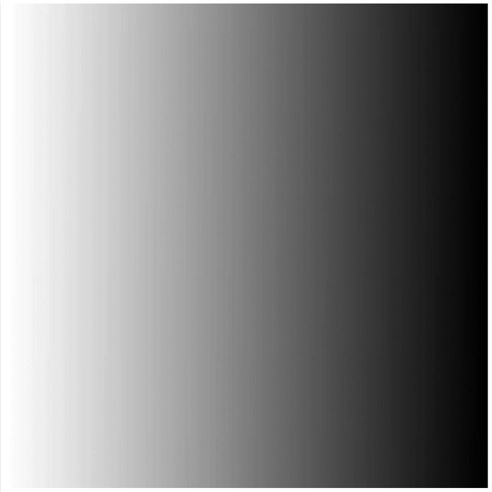
Source image (16 b/ch)



Gamma (contrasted)



Linear (contrasted)



Normal maps precision

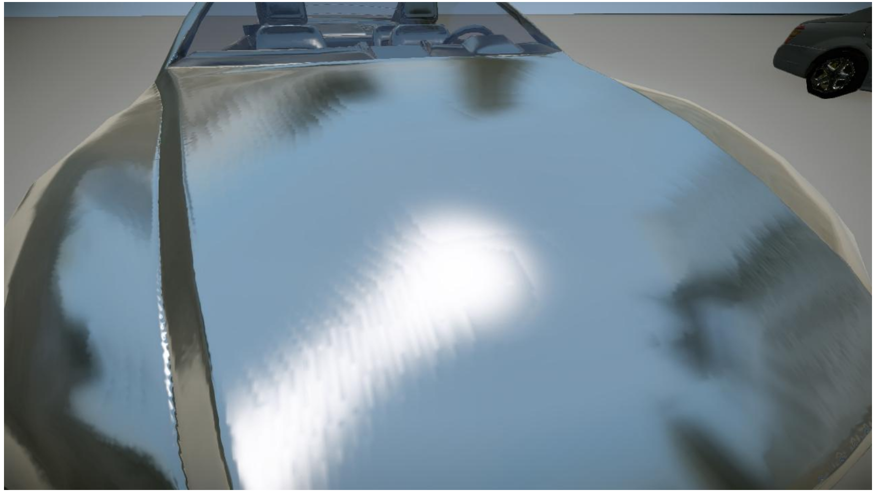
- Artists used to store normal maps into 8b/ch texture
  - Normals are quantized from the very beginning!
- Changed the pipeline to ALWAYS export **16b/channel normal maps**!

– Modify your tools to export that by default

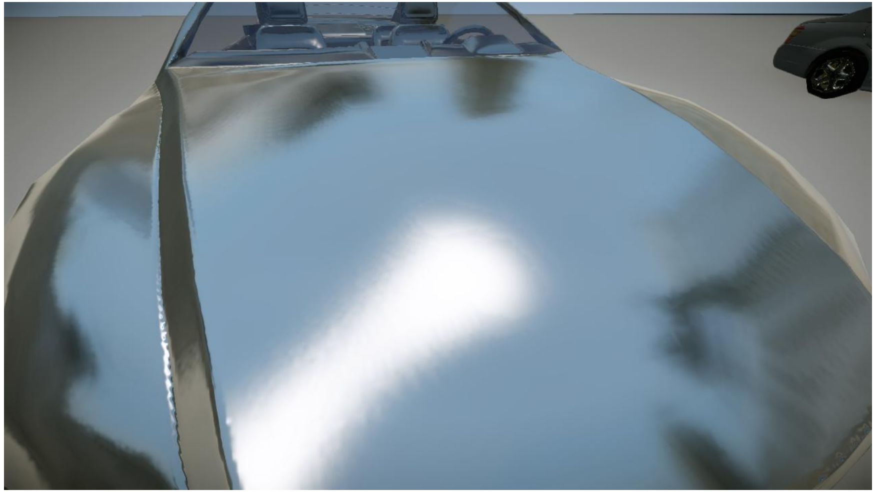
– Transparent for artists

16-bits normal maps example

3Dc from 8-bits/channel source



3Dc from 16-bits/channel source



3Dc encoder improvements

- Two h/w color spaces for free: linear/gamma
  - Solution to the equation

$$x_\gamma{}' = x'$$

  - Median (**linear space**):

$$x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{\mathbf{255}}$$

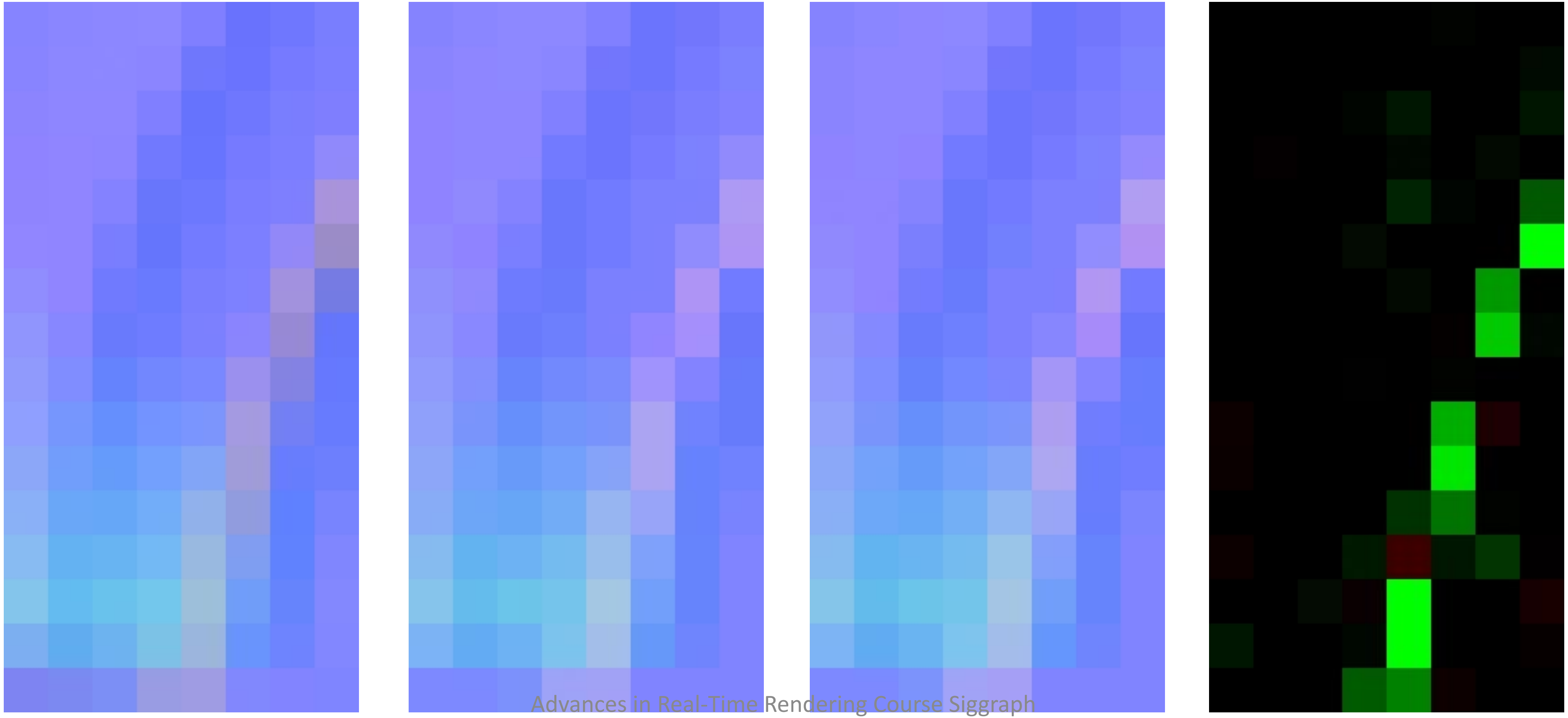
  - Choose the right color space based on histogram:
  - <u>Rule of thumb:</u> use linear if >75% of pixels are above the median

3Dc encoder improvements, cont'd

- One 1024x1024 texture is compressed in **~3 hours** with CUDA on Fermi!
  - Brute-force exhaustive search
  - Too slow for production
- Notice: solution is close to common 3Dc encoder results
- Adaptive approach: compress as 2 alpha blocks, measure error for normals. If the error is higher than threshold, run high-quality encoder

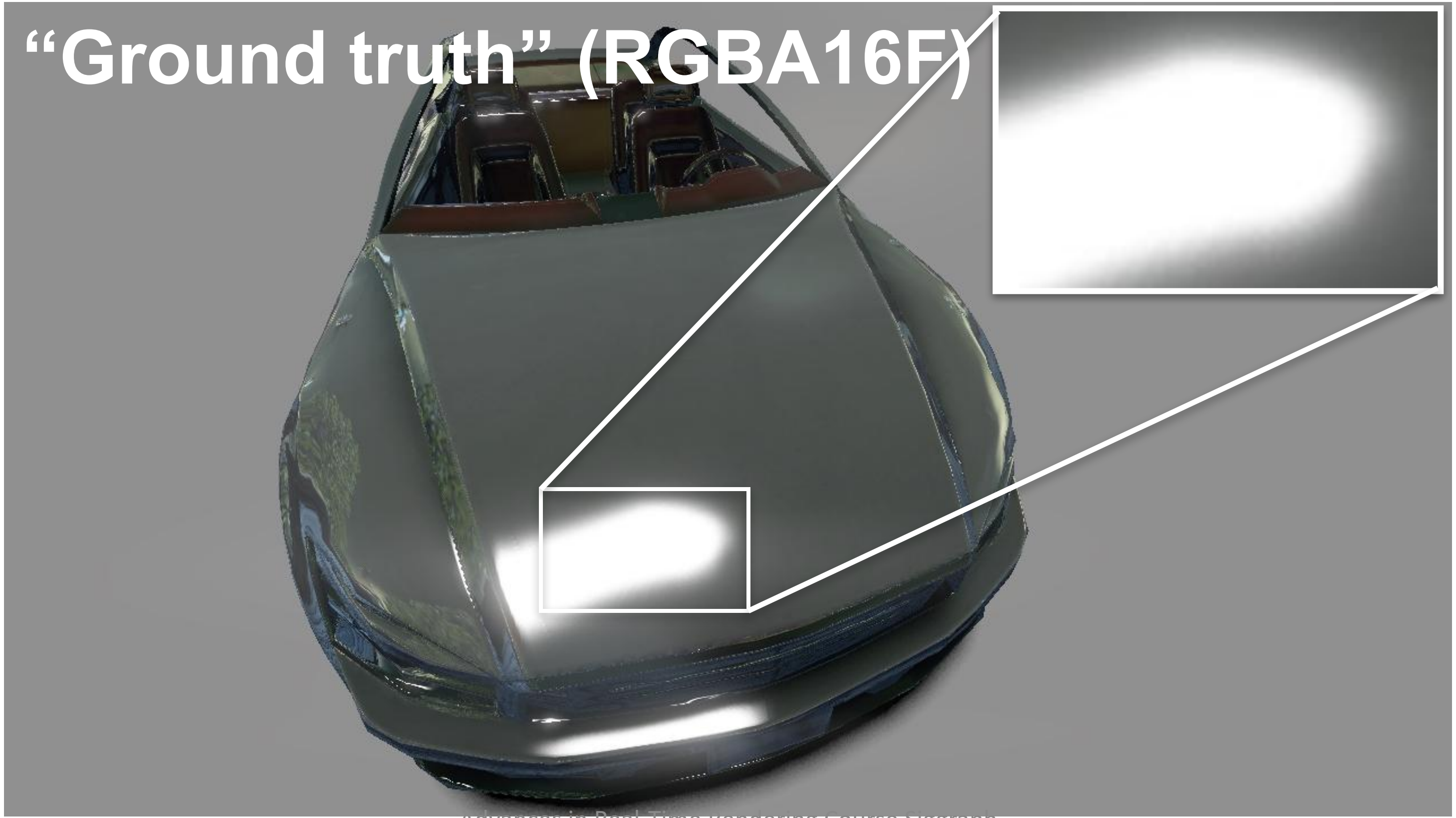
3Dc improvement example

Original nm, 16b/c | Common encoder | Proposed encoder | Difference map

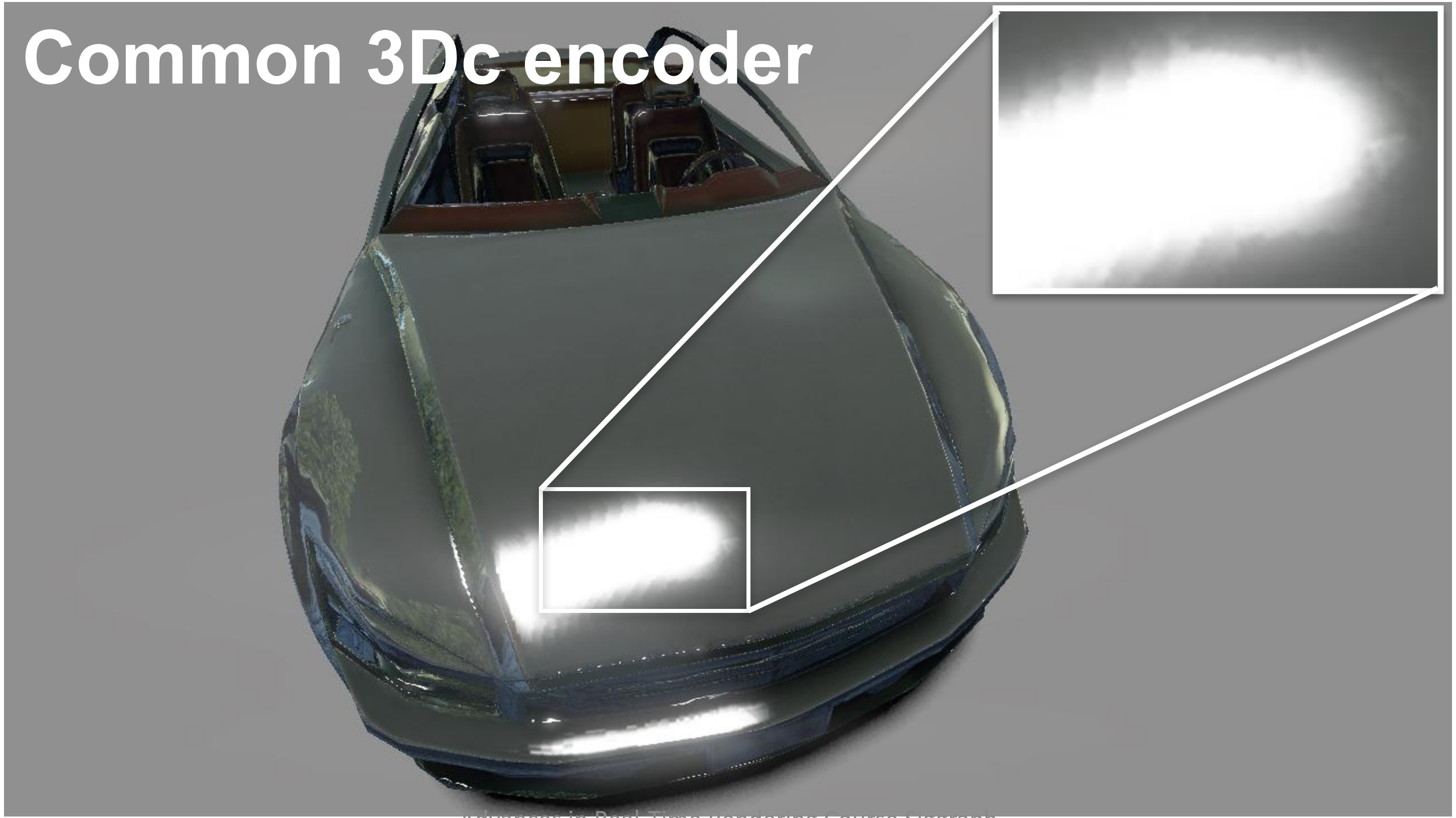


a | b | c | d

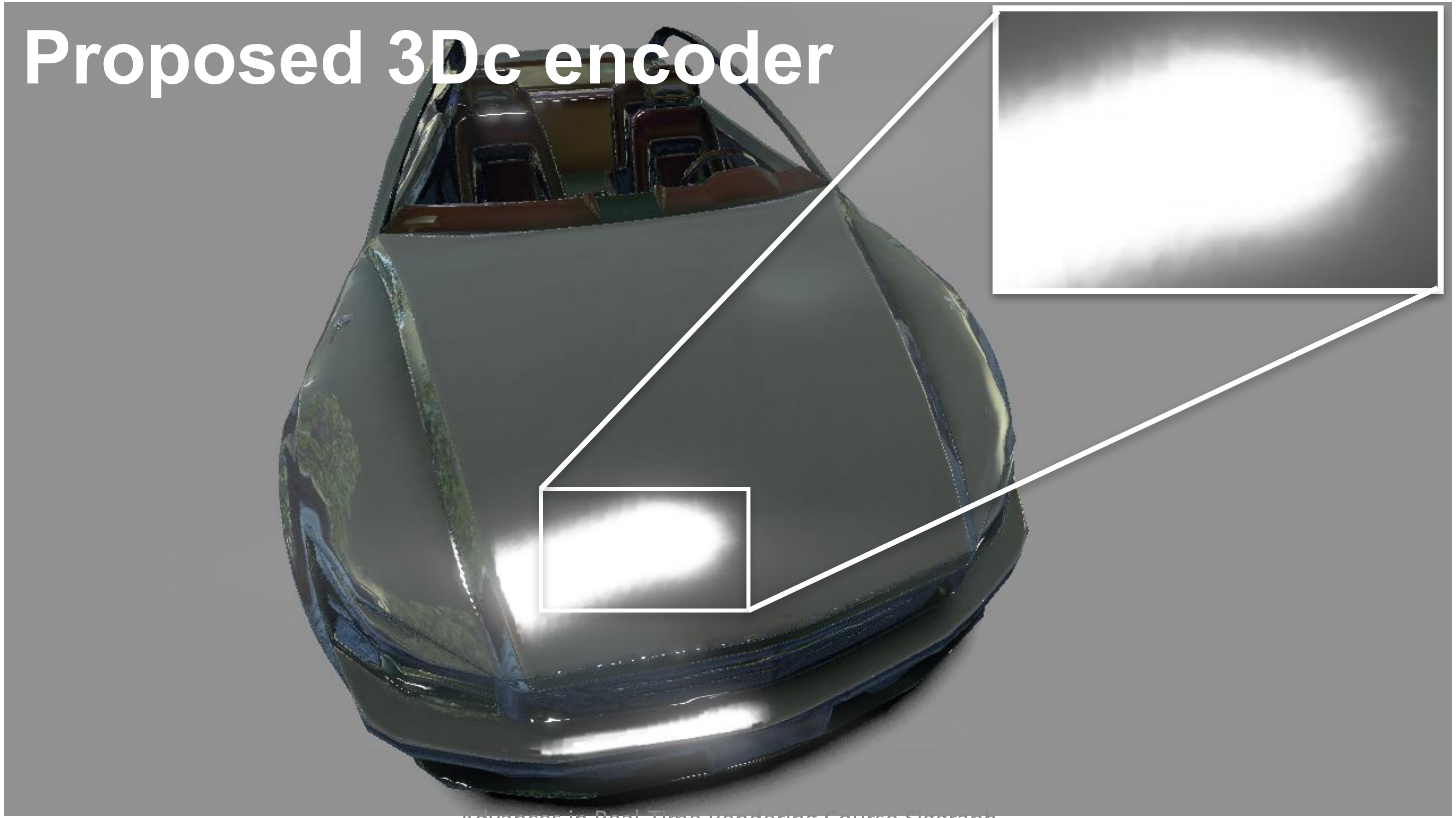
3Dc improvement example



3Dc improvement example



3Dc improvement example



DIFFERENT IMPROVEMENTS

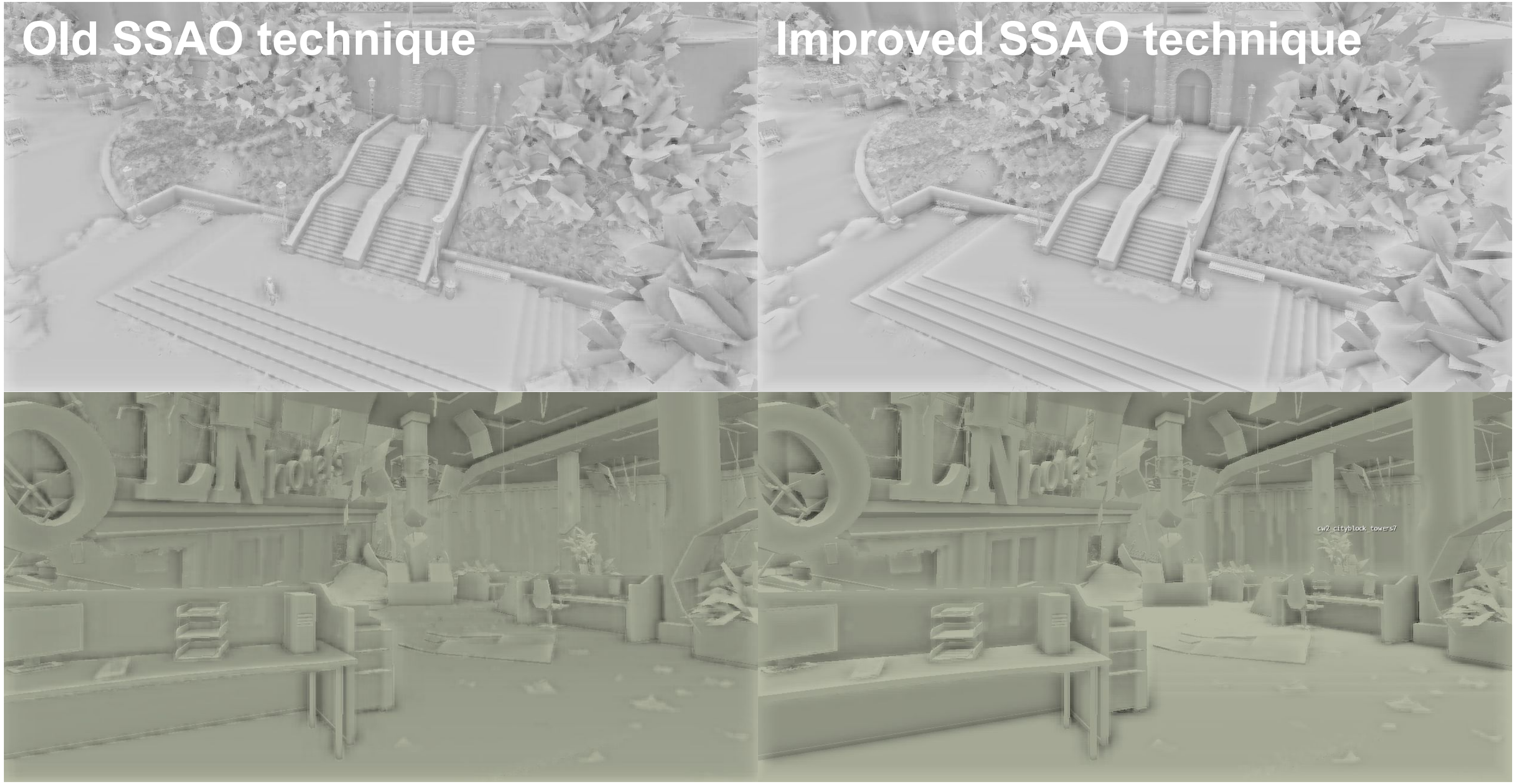
Occlusion culling

- Use software z-buffer (aka coverage buffer)
  - Downscale previous frame's z buffer on consoles
    - Use conservative occlusion to avoid false culling
  - Create mips and use hierarchical occlusion culling
    - Similar to Zcull and Hi-Z techniques
    - Use AABBs and OOBBs to test for occlusion
  - On PC: place occluders manually and rasterize on CPU
    - CPU↔GPU latency makes z buffer useless for culling

SSAO improvements

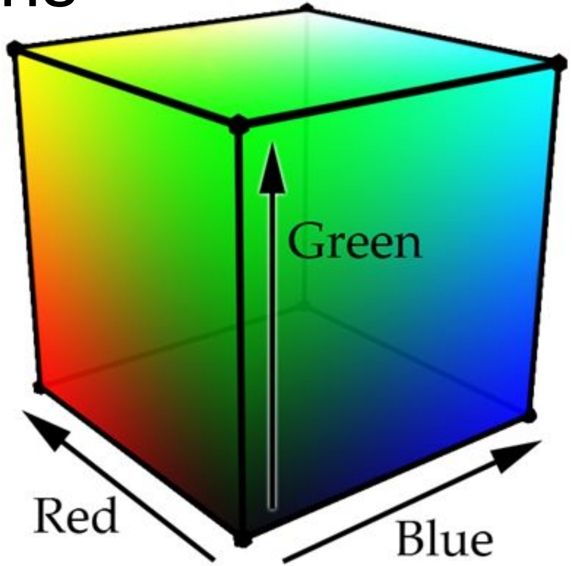
- Encode depth as 2 channel 16-bits value [0;1]
  - Linear detph as a rational: depth=x+y/255
- Compute SSAO in half screen resolution
  - Render SSAO into the same RT (another channel)
  - Bilateral blur fetches SSAO and depth at once
- Volumetric Obscurrance [LS10] with **4**(!) samples
- Temporal accumulation with simple reprojection
- Total performance: **1**ms on X360, **1.2**ms on PS3

Improvements examples on consoles



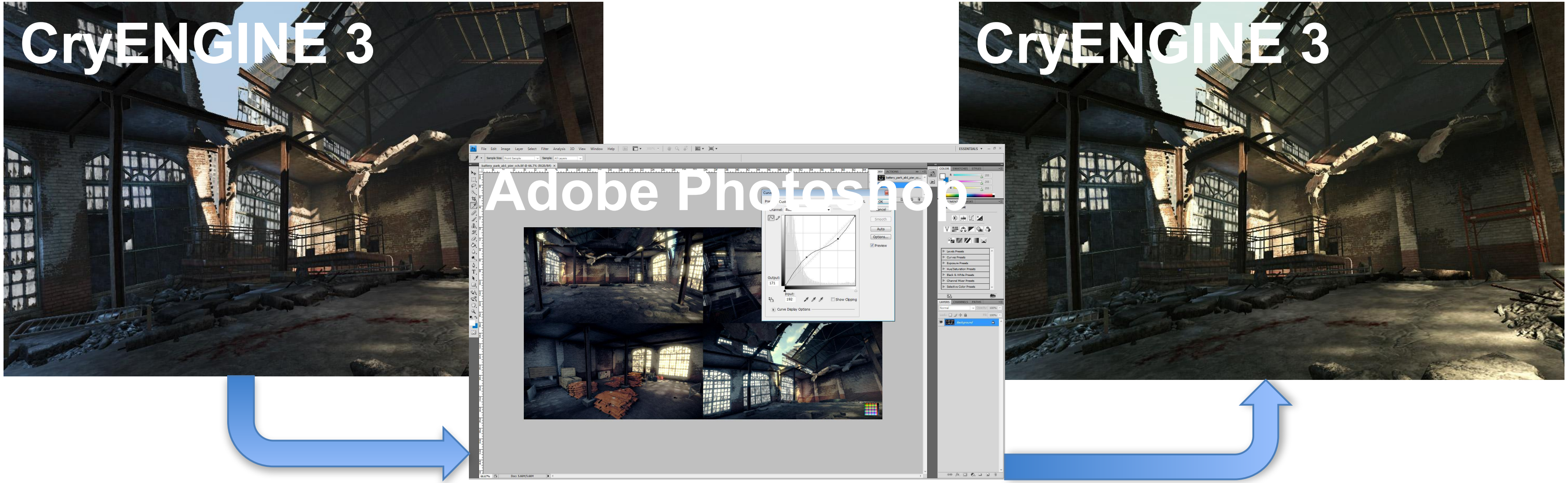
Color grading

- Bake all global color transformations into 3D LUT [SELAN07]
  - 16x16x16 LUT proved to be enough
- Consoles: use h/w 3D texture
  - Color correction pass is one lookup
    - newColor = tex3D(LUT, oldColor)



Color grading

- Use Adobe Photoshop as a color correction tool
- Read transformed color LUT from Photoshop



Color chart example for Photoshop

DEFERRED PIPELINE

Why deferred lighting?

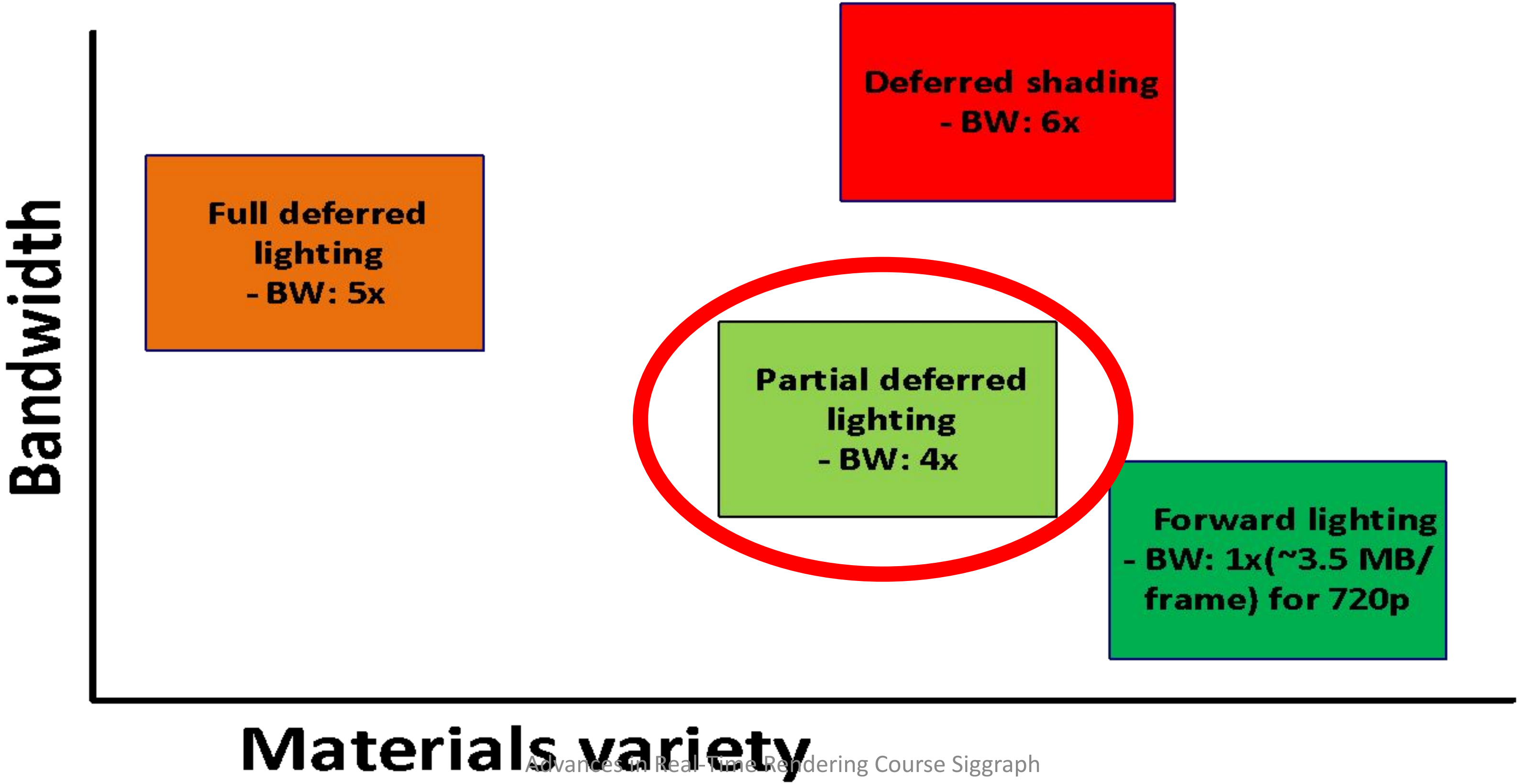
Why deferred lighting?

Why deferred lighting?

Introduction

- Good decomposition of lighting
  - No lighting-geometry interdependency
- Cons:
  - **Higher memory and bandwidth requirements**

Deferred pipelines bandwidth



Major issues of deferred pipeline

- No anti-aliasing
  - Existing multi-sampling techniques are too heavy for deferred pipeline
  - Post-process antialiasing doesn't remove aliasing completely
    - Need to super-sample in most cases
- Limited materials variations
  - No anisotropic materials
- Transparent objects are not supported

Lighting layers of CryENGINE 3

- Indirect lighting
  - Ambient term
  - Tagged ambient areas
  - Local cubemaps
  - Local deferred lights
  - Diffuse Indirect Lighting from LPVs
  - SSAO
- Direct lighting
  - All direct light sources, with and without shadows

G-Buffer. The smaller the better!

- Minimal G-Buffer layout: 64 bits / pixel
  - **RT0**: Depth 24bpp + Stencil 8bpp
  - **RT1**: Normals 24 bpp + Glossiness 8bpp
- Stencil to mark objects in lighting groups
  - Portals / indoors
  - Custom environment reflections
  - Different ambient and indirect lighting

G-Buffer. The smaller the better, Cont'd

- Glossiness is non-deferrable
  - Required at lighting accumulation pass
  - Specular is non-accumulative otherwise
- Problems of this G-Buffer layout:
  - Only Phong BRDF (normal + glossiness)
    - **No aniso materials**
  - Normals at 24bpp are too quantized
    - Lighting is banded / of low quality

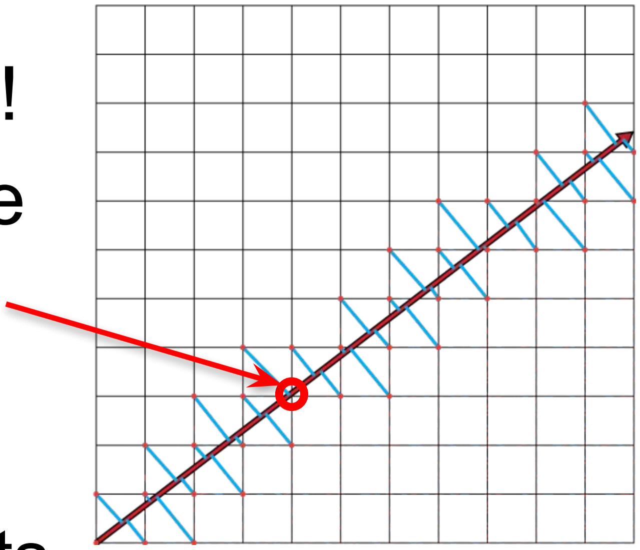
STORING NORMALS IN G-BUFFER

Normals precision for shading

- Normals at 24bpp are too quantized, lighting is of a low quality
- 24 bpp should be enough. What do we do wrong? We store normalized normals!
- Cube is 256x256x256 cells = 16777216 values
- We use only cells on unit sphere in this cube:
  - **~289880** cells out of **16777216**, which is ~ **1.73 %** ! !

Normals precision for shading, part III

- We have a cube of 256^3 values!
- Best fit: find the quantized value with the minimal error for a ray
  - Not a real-time task!
    - Constrained optimization in 3DDDA
- Bake it into a cubemap of results
  - Cubemap should be huge enough (obviously > 256x256)



Normals precision for shading, part III

- Extract the most meaningful and unique part of this symmetric cubemap
- Save into 2D texture
- Look it up during G-Buffer generation
- Scale the normal
- Output the adjusted normal into G-Buffer
- See appendix A for more implementation details

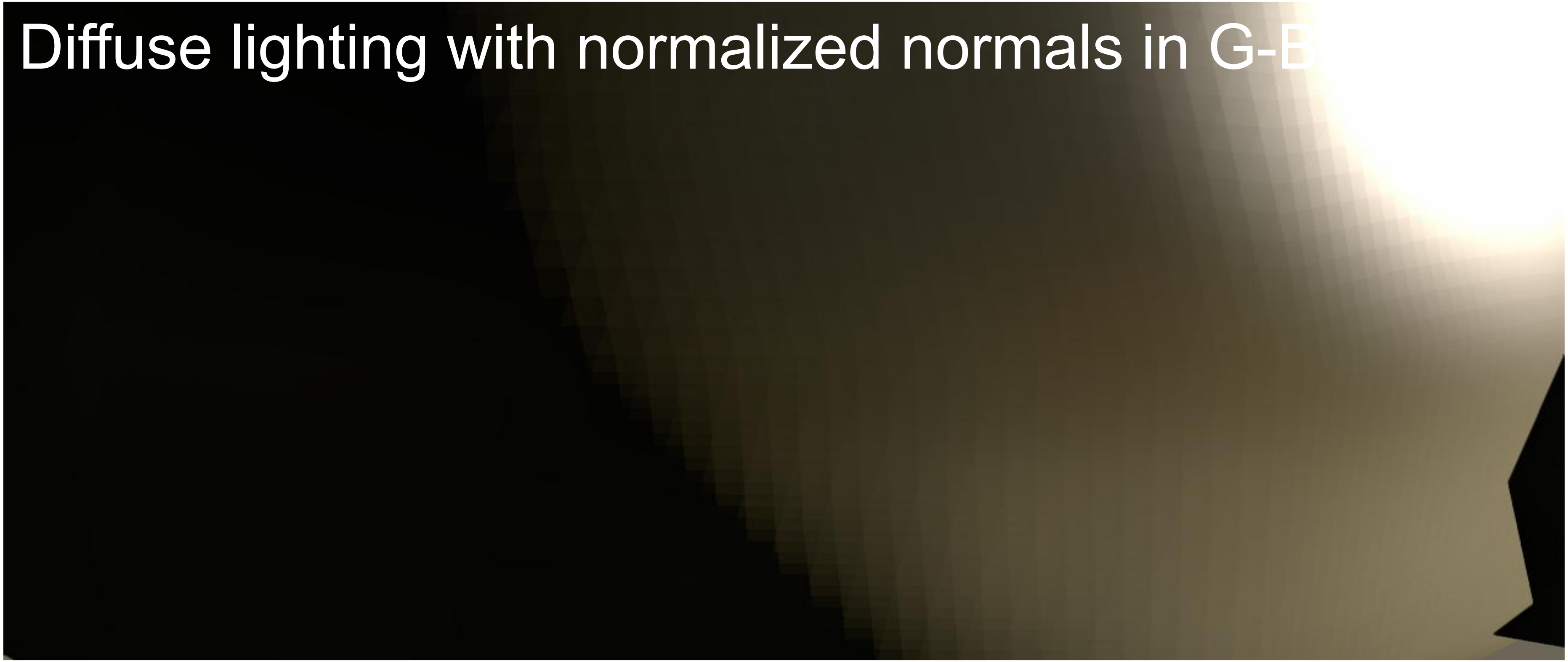
Best fit for normals

- Supports alpha blending
  - Best fit gets broken though. Usually not an issue
- Reconstruction is just a normalization!
  - Which is usually done anyway
- Can be applied to some selective smooth objects
  - E.g. disable for objects with detail bump
- Don’t forget to create mip-maps for results texture!

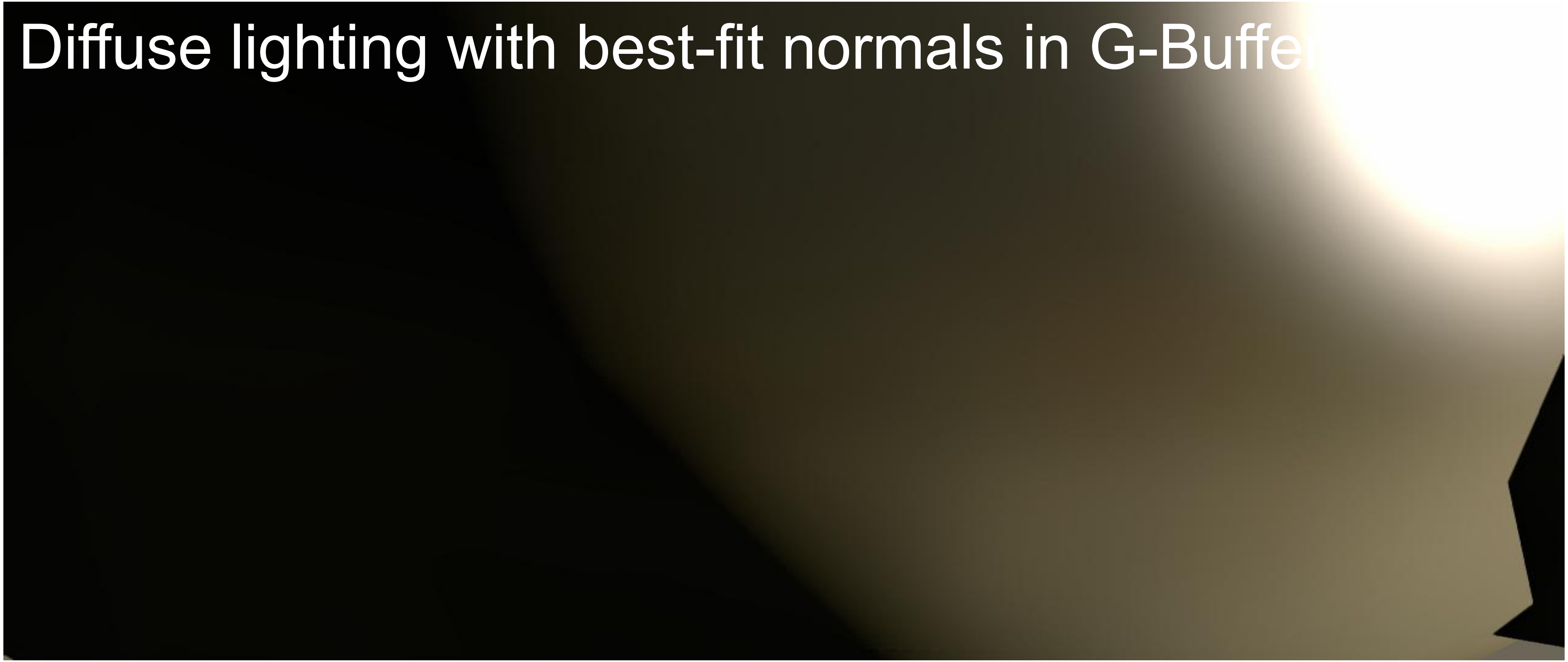
Storage techniques breakdown

1. **Normalized normals:**
   - **~289880** cells out of **16777216**, which is ~ **1.73 %**
2. **Divided by maximum component:**
   - **~390152** cells out of **16777216**, which is ~ **2.33 %**
3. **Proposed method (best fit):**
   - **~16482364** cells out of **16777216**, which is ~ **98.2 %**
     - Two orders of magnitude more

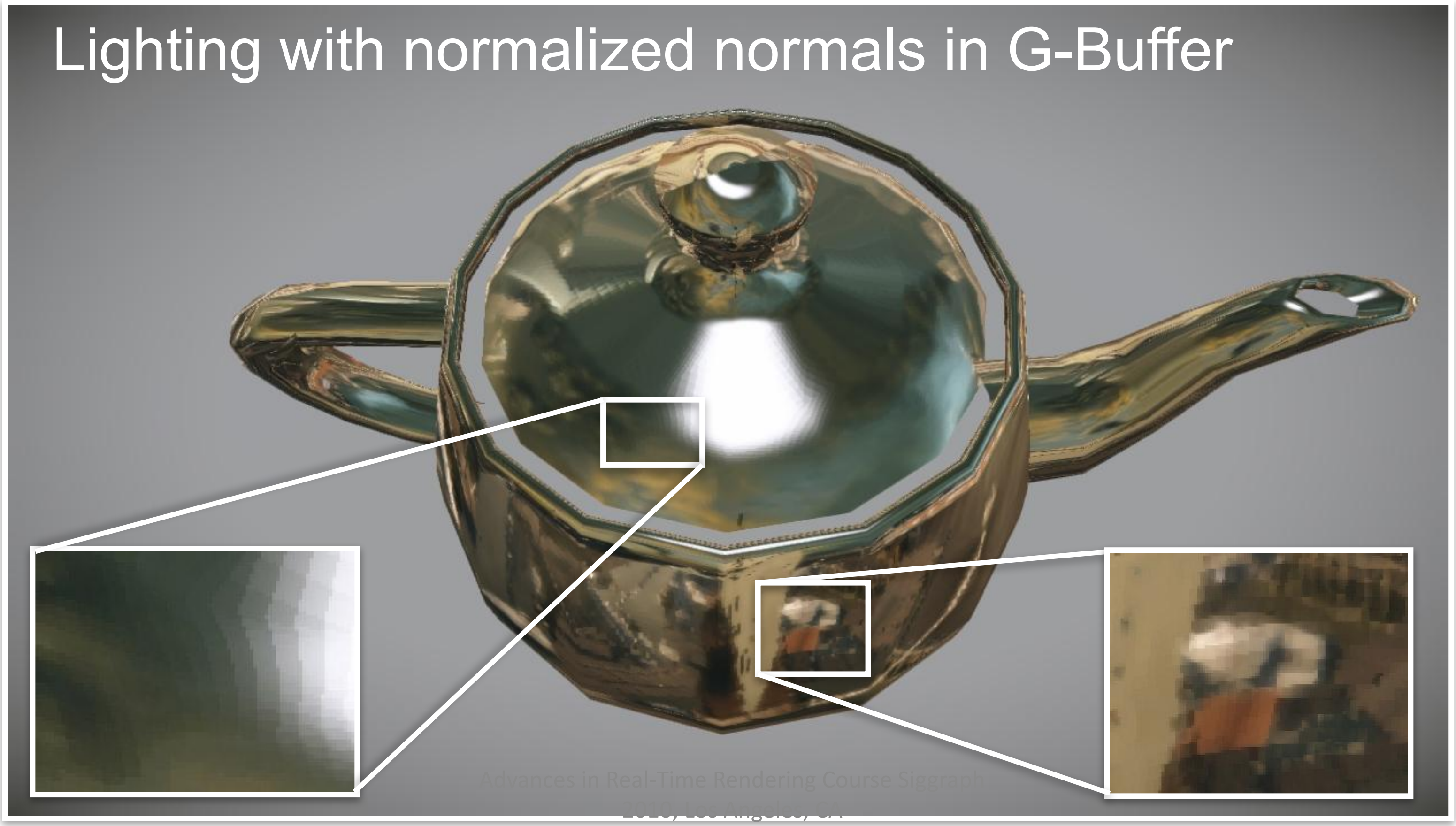
Normals precision in G-Buffer, example



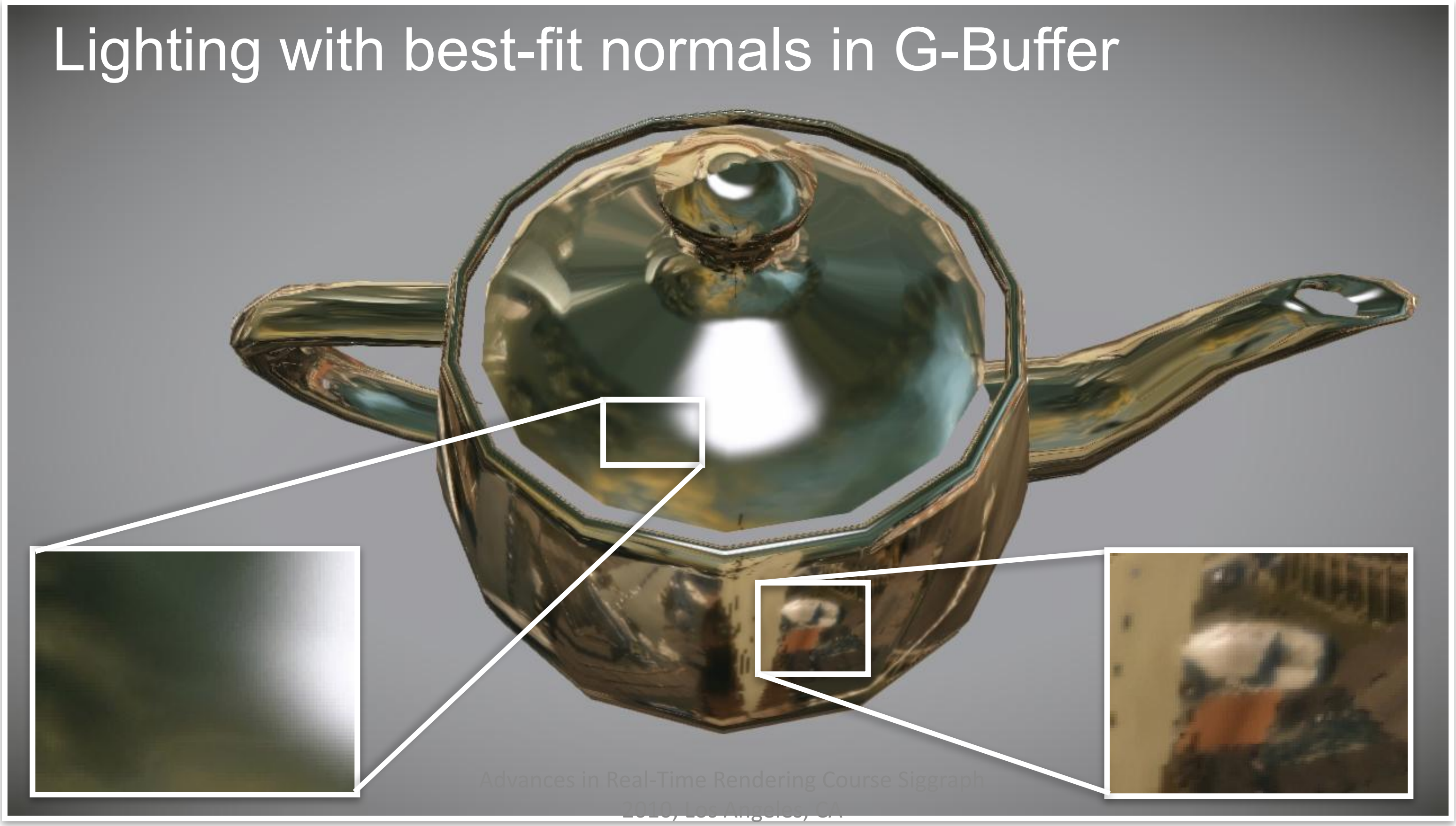
Normals precision in G-Buffer, example



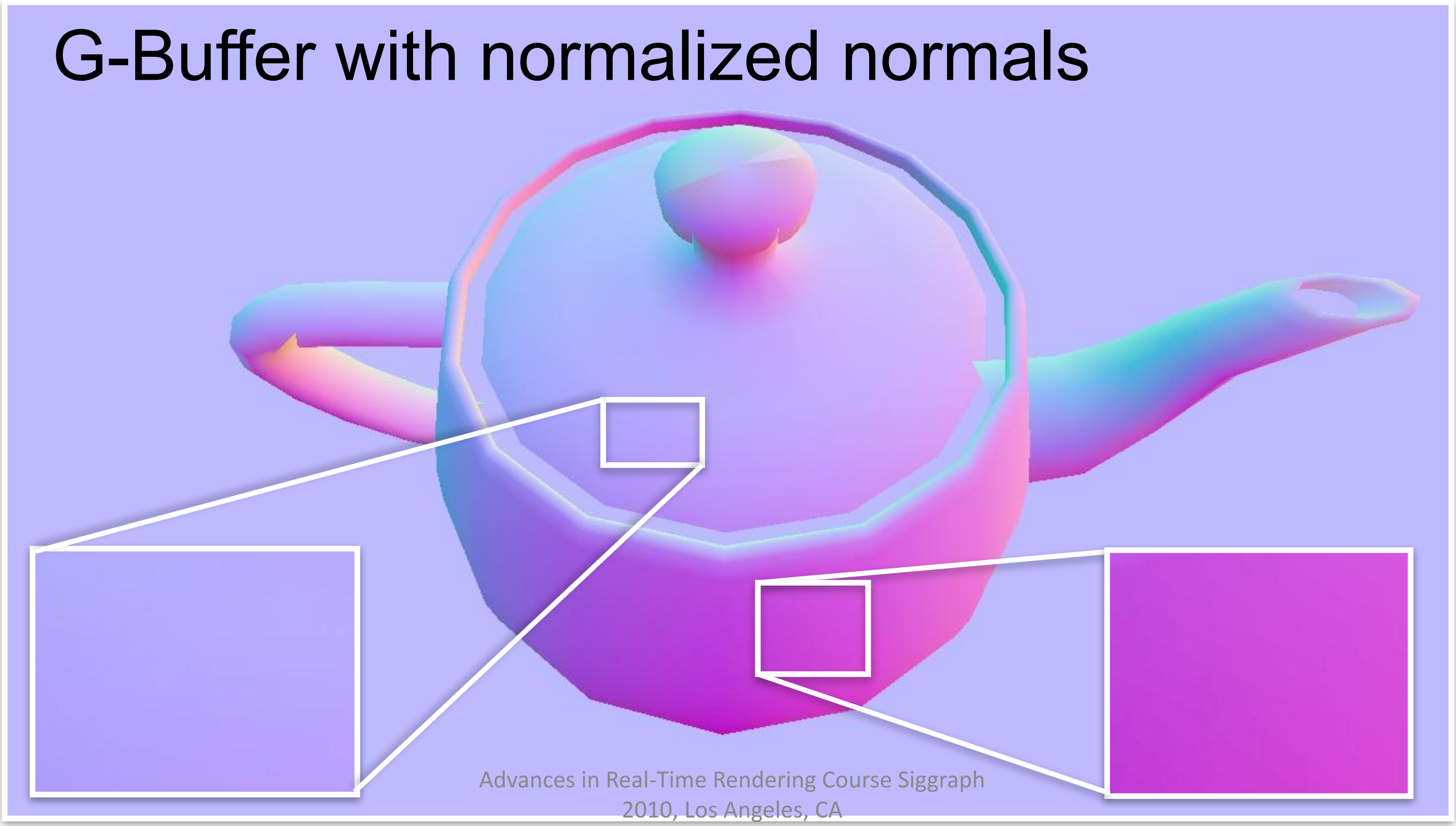
Normals precision in G-Buffer, example



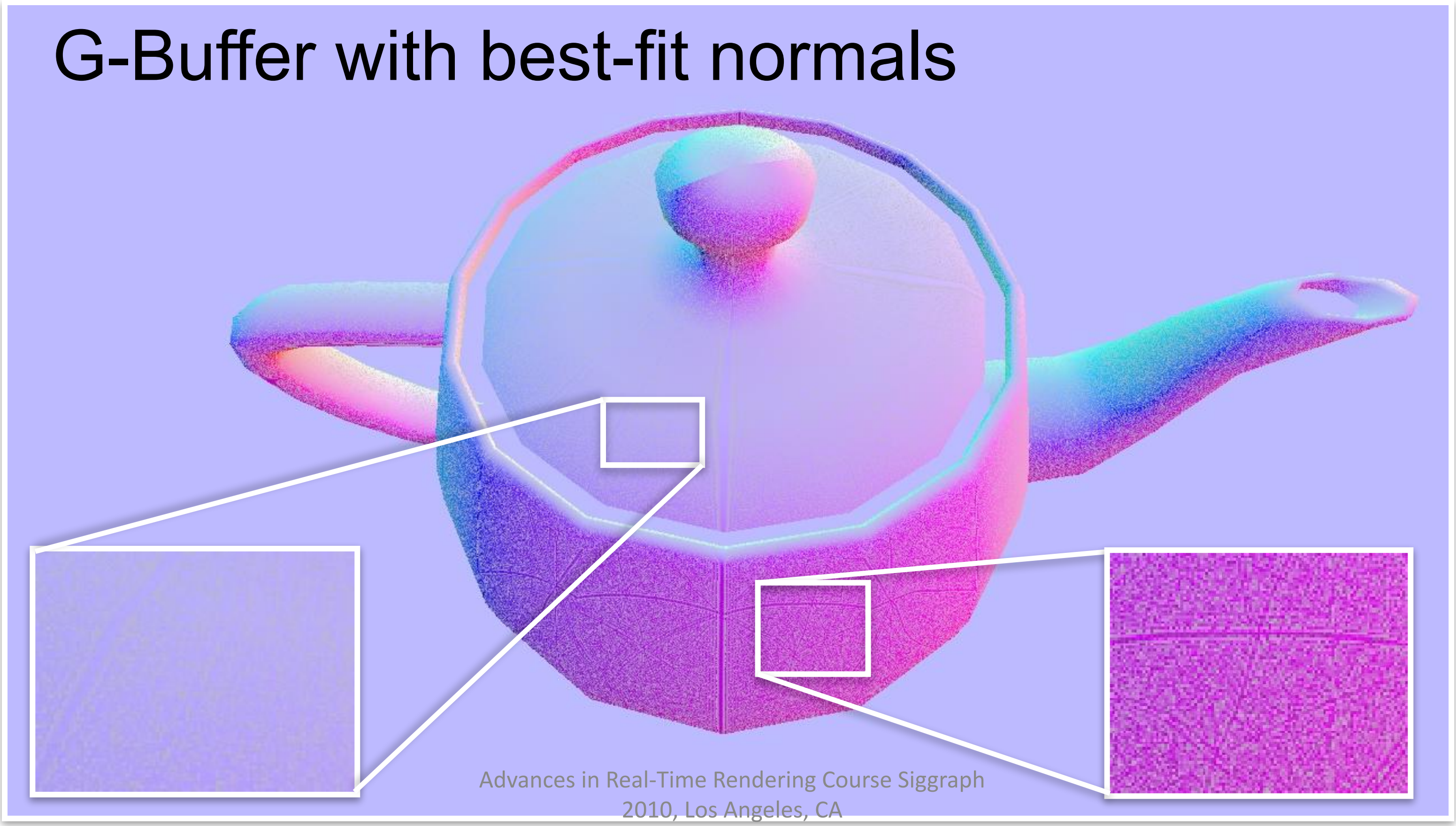
Normals precision in G-Buffer, example



Normals precision in G-Buffer, example



Normals precision in G-Buffer, example



PHYSICALLY-BASED BRDFS

Lighting consistency: Phong BRDF

- Two h/w color spaces for free: linear/gamma
  - Solution to the equation

$$x_\gamma{}' = x'$$

  - Median (**linear space**):

$$x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{\mathbf{255}}$$

  - Choose the right color space based on histogram:
  - <u>Rule of thumb:</u> use linear if >75% of pixels are above the median

Consistent lighting example

Consistent lighting example

Consistent lighting example

HDR…
VS BANDWIDTH VS PRECISION

HDR on consoles

- Can we achieve bandwidth the same as for LDR?
- **PS3**: RGBK (aka RGBM) compression
  - RGBA8 texture – the same bandwidth
  - RT read-backs solves blending problem
- **Xbox360**: Use R11G11B10 texture for HDR
  - Same bandwidth as for LDR
    - Remove _AS16 suffix for this format for better cache utilization
  - Not enough precision for linear HDR lighting!

HDR on consoles: dynamic range

- Use dynamic range scaling to improve precision
- Use average luminance to detect the efficient range
  - Already computed from previous frame
- Detect lower bound for HDR image intensity
  - The final picture is LDR after tone mapping
  - The LDR threshold is 0.5/255=1/510
  - Use inverse tone mapping as estimator

HDR on consoles: lower bound estimator

- Two h/w color spaces for free: linear/gamma
  - Solution to the equation

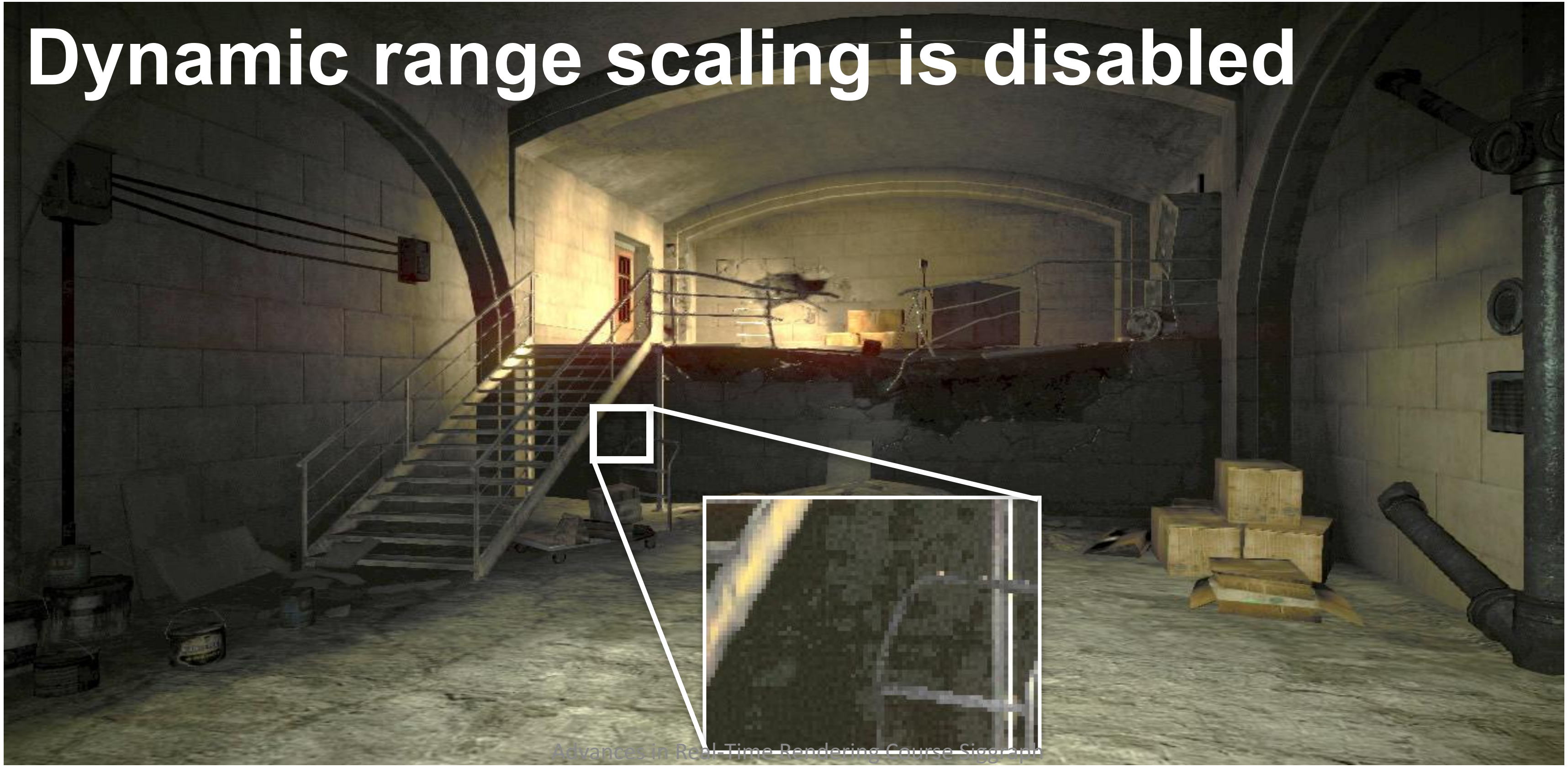
$$x_{\gamma}' = x'$$

  - Median (**linear space**):

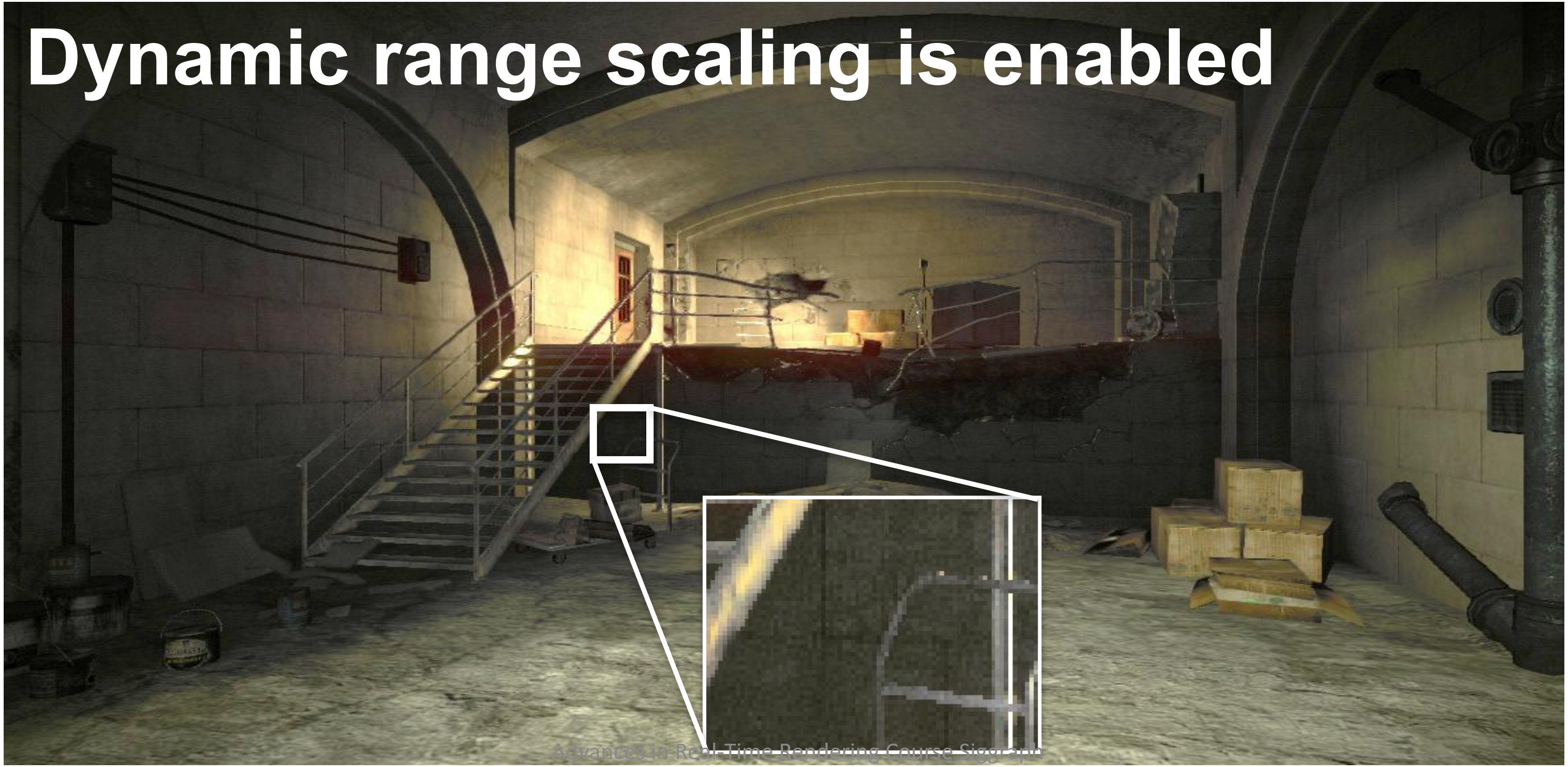
$$x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{255}$$

  - Choose the right color space based on histogram:
  - Rule of thumb: use linear if >75% of pixels are above the median

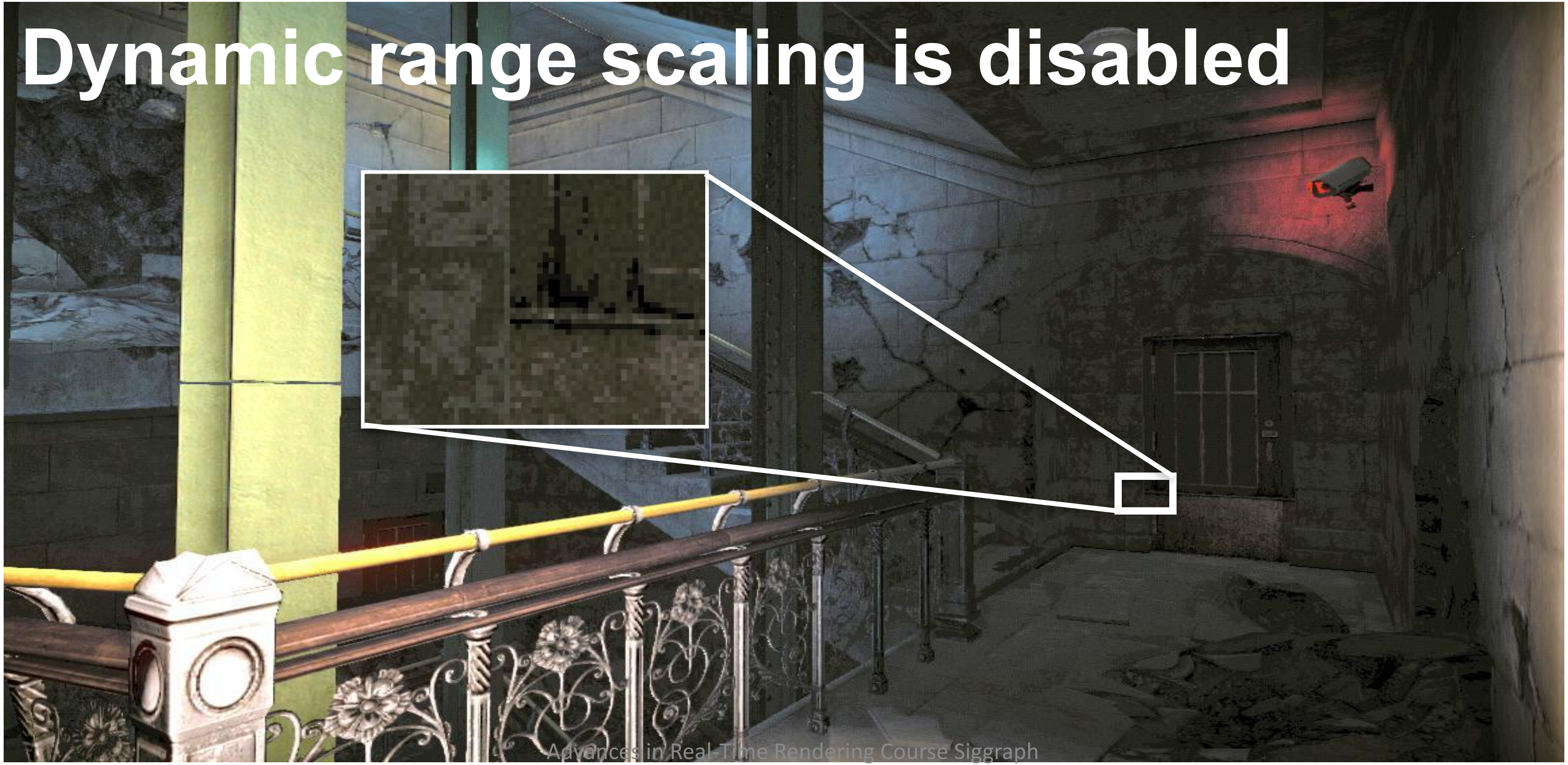
HDR dynamic range example



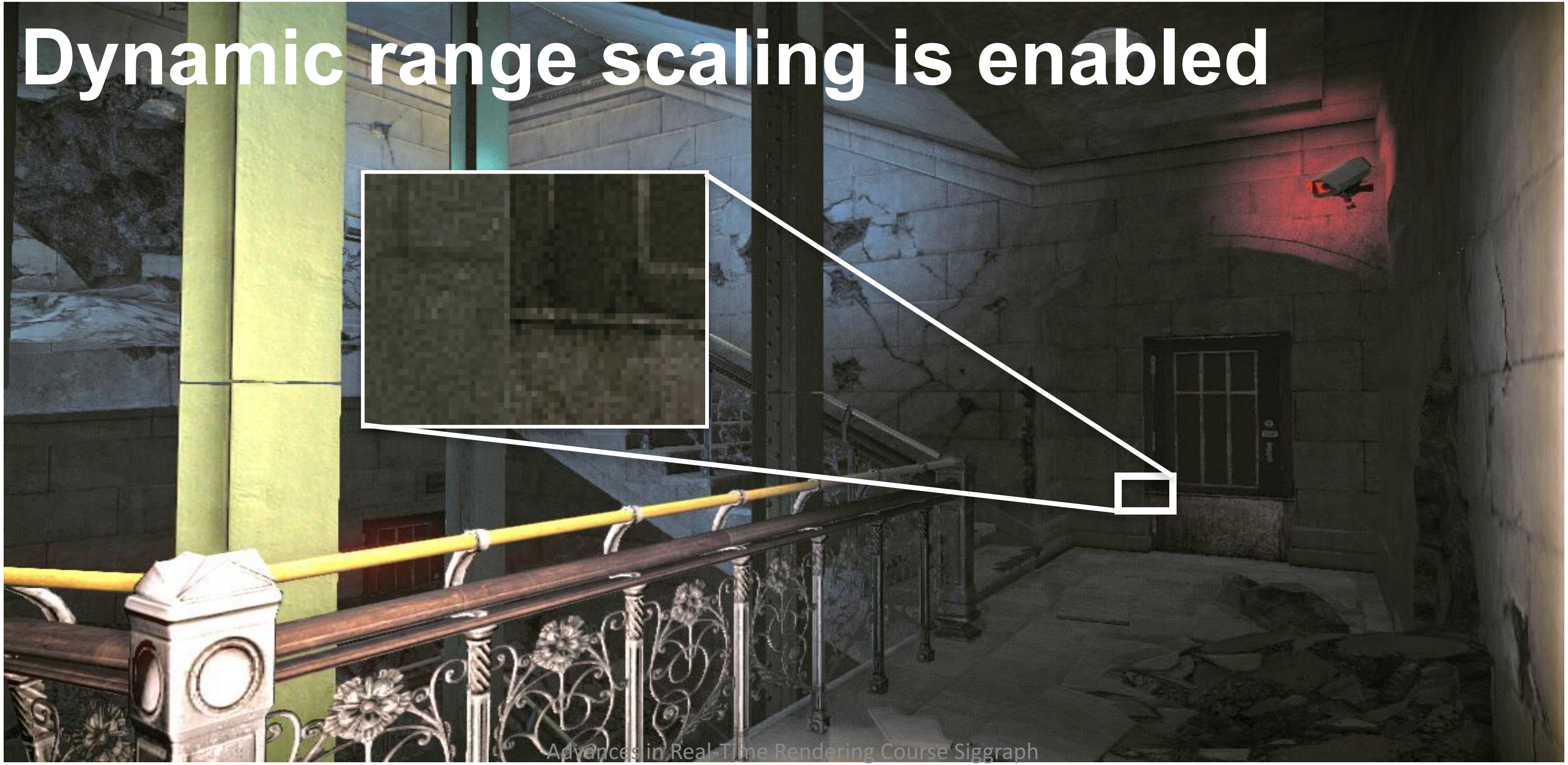
HDR dynamic range example



HDR dynamic range example

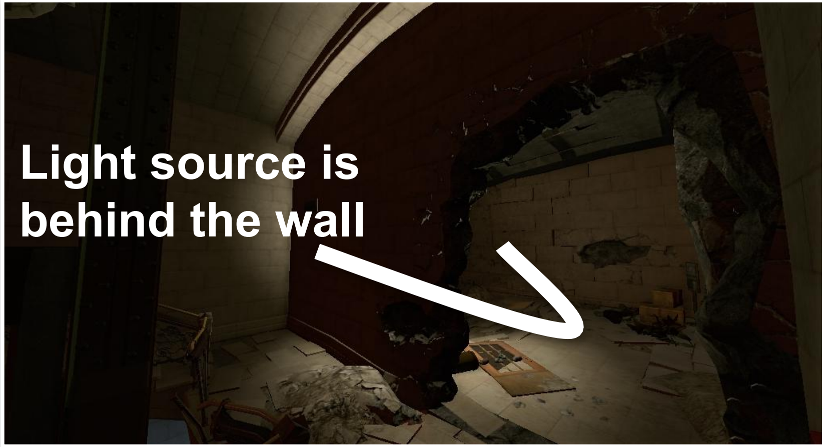


HDR dynamic range example



LIGHTING TOOLS: CLIP VOLUMES

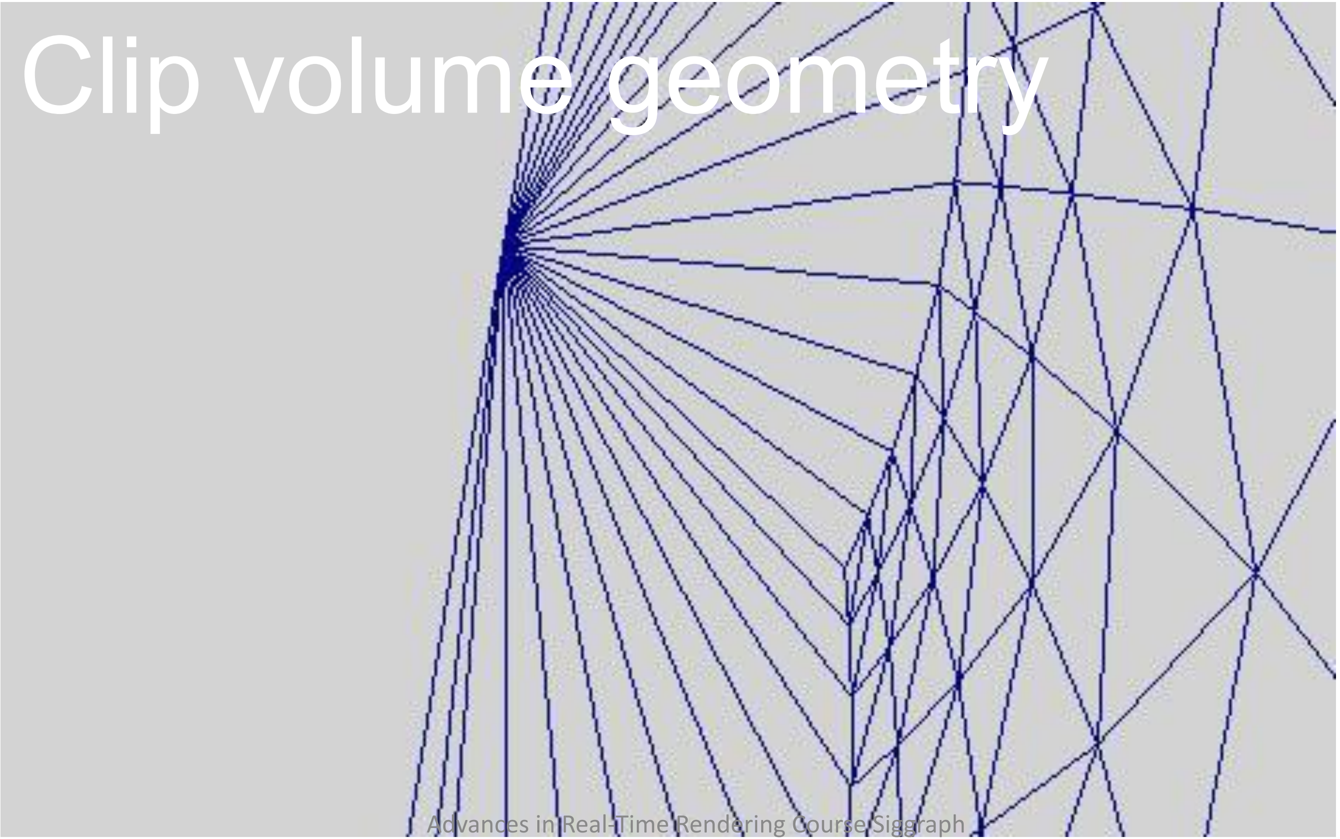
Clip Volumes for Deferred Lighting



- Deferred light source w/o shadows tend to bleed:
  - Shadows are expensive
- Solution: use artist-defined clipping geometry: **clip volumes**
  - Mask the stencil in addition to light volume masking
  - Very cheap providing fourfold stencil tagging speed

Clip Volumes example

Clip Volumes example



Clip Volumes example

Clip Volumes example

Clip Volumes example

DEFERRED LIGHTING AND ANISOTROPIC MATERIALS

Anisotropic deferred materials

- G-Buffer stores only **normal** and **glossiness**
  - That defines a BRDF with a **single Phong lobe**
- We need **more lobes** to represent anisotropic BRDF
  - Could be extended with **fat G-Buffer** (**too heavy** for production)
- Consider one **screen pixel**
  - We have **normal** and **view vector**, thus BRDF is defined on sphere
  - Do we need all these lobes to illuminate this pixel?
  - Lighting distribution is **unknown** though

Anisotropic deferred materials, part I

- Idea: Extract the major Phong lobe from NDF
  - Use microfacet BRDF model [CT82]:

    - Two h/w color spaces for free: linear/gamma
      - Solution to the equation $x_{\gamma}' = x'$
      - Median (**linear space**): $x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{\mathbf{255}}$
      - Choose the right color space based on histogram:
      - Rule of thumb: use linear if >75% of pixels are above the median

  - Fresnel and geometry terms can be deferred
  - Lighting-implied BRDF is proportional to the NDF:

    - Two h/w color spaces for free: linear/gamma
      - Solution to the equation $x_{\gamma}' = x'$
      - Median (**linear space**): $x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{\mathbf{255}}$
      - Choose the right color space based on histogram:
      - Rule of thumb: use linear if >75% of pixels are above

- Approximate NDF with **Spherical Gaussians** [WRGSG09]
  - Need only ~7 lobes for Anisotropic Ward NDF

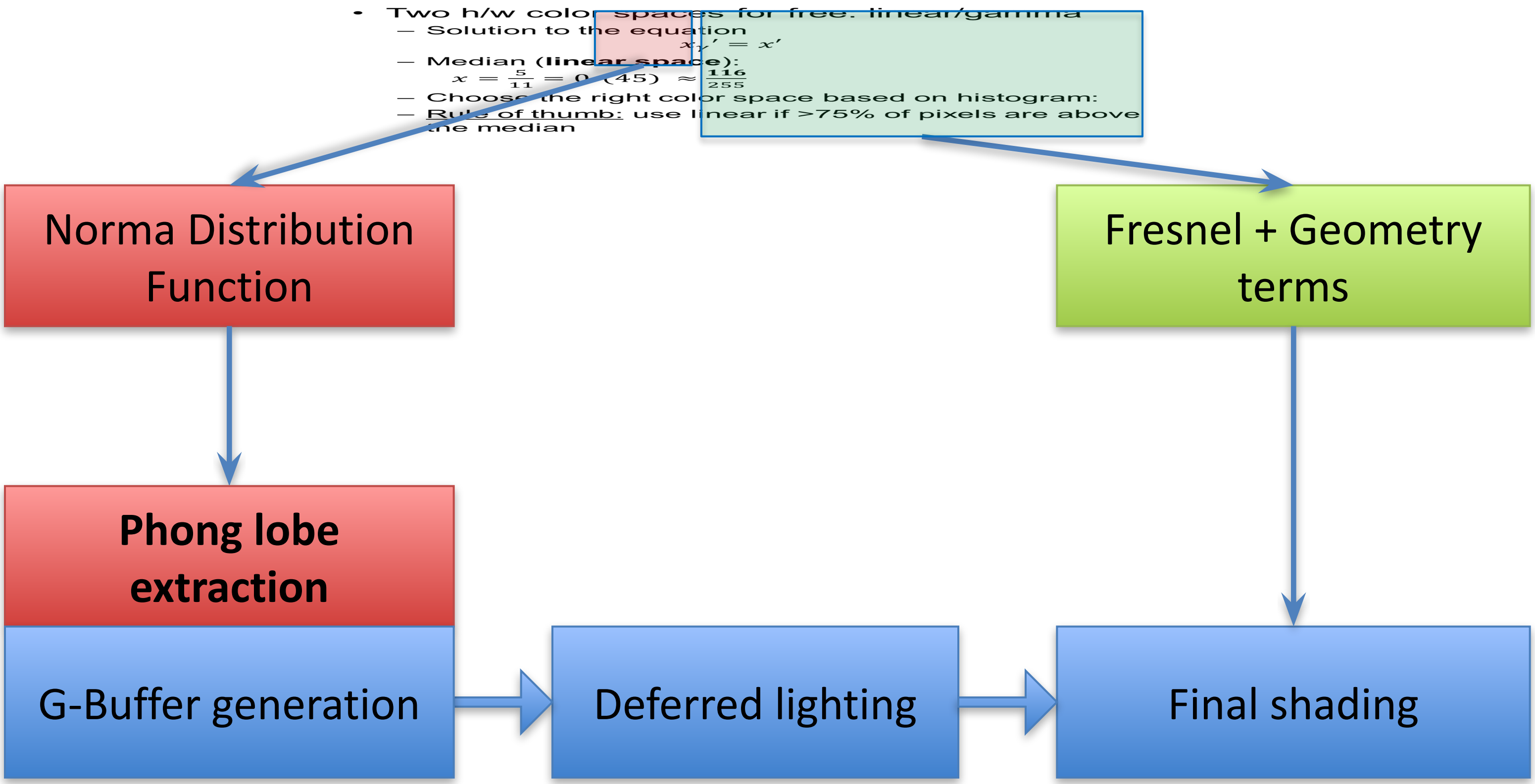
Anisotropic deferred materials, part II

- Approximate lighting distribution with **SG** <u>per object</u>
  - Merge SG functions if appropriate
  - Prepare several approximations for huge objects
- Extract the principal Phong lobe into G-Buffer
  - Convolve lobes and extract the mean normal (next slide)
- Do a usual deferred Phong lighting
- Do shading, apply Fresnel and geometry term

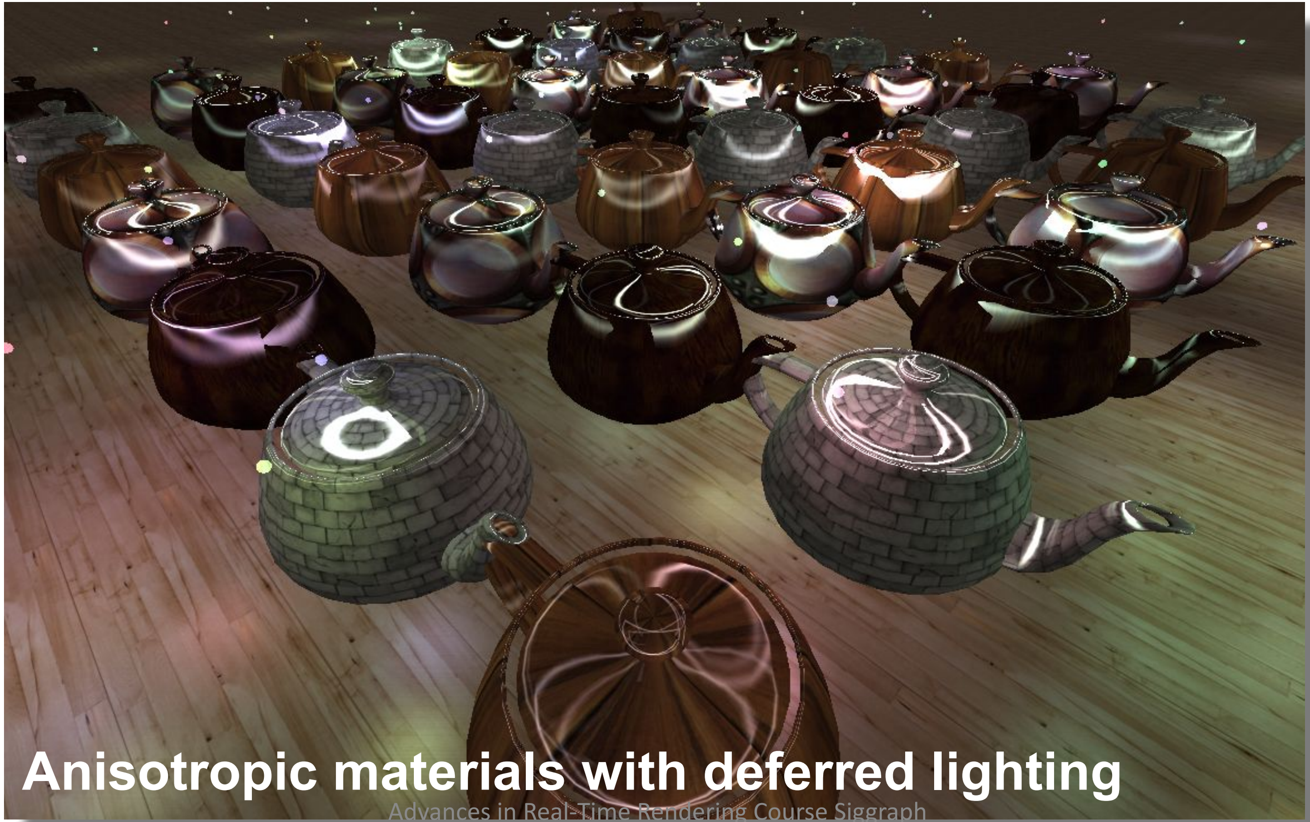
Extracting the principal Phong lobe

- CPU: prepare SG lighting representation per object
- Vertex shader:
  - Rotate SG representation of BRDF to local frame
  - Cut down number of lighting SG lobes to ~7 by hemisphere
- Pixel shader:
  - Rotate SG-represented BRDF wrt tangent space
  - Convolve the SG BRDF with SG lighting
  - Compute the principal Phong lobe and output it

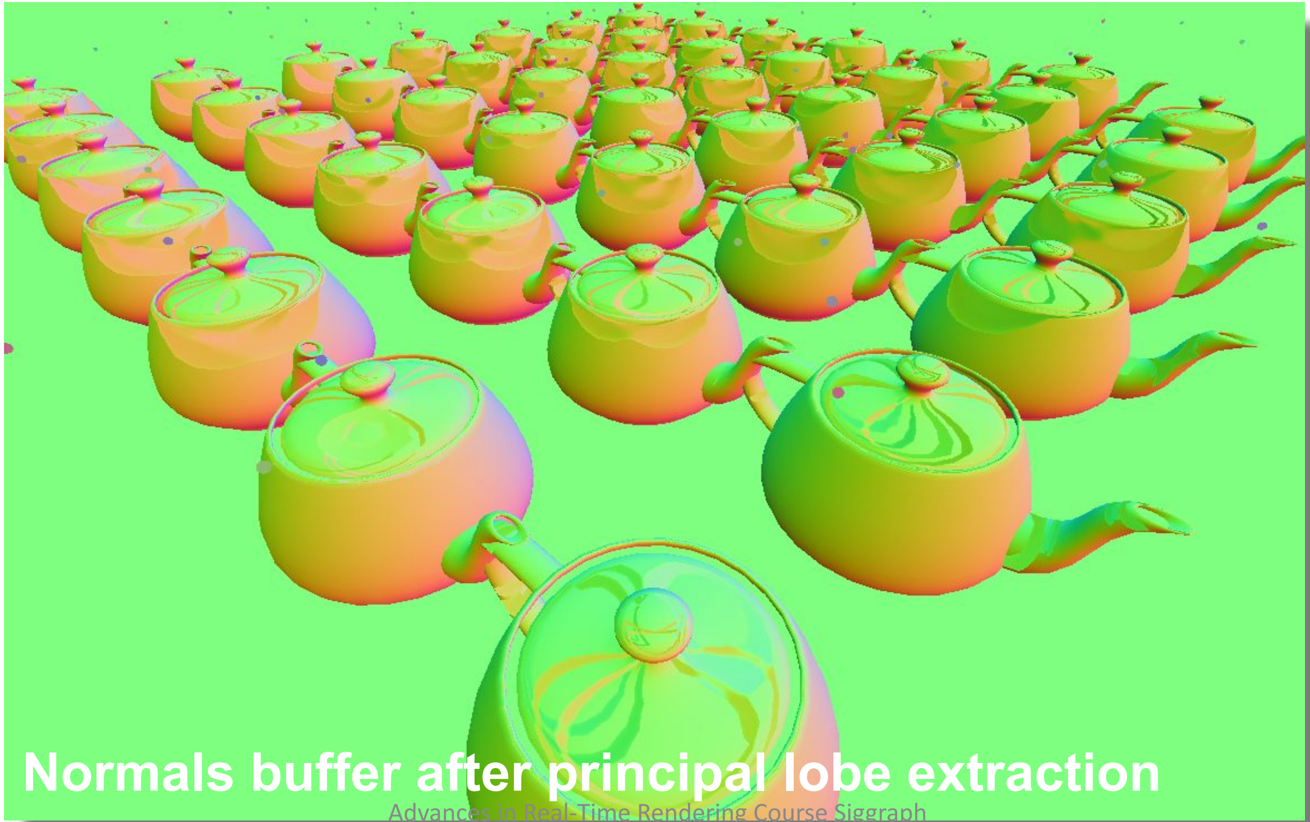
Anisotropic deferred materials



Anisotropic deferred materials



Anisotropic deferred materials



Anisotropic deferred materials: why?

- Cons:
  - Imprecise lobe extraction and specular reflections
    - But: see [RTDKS10] for more details about perceived reflections
  - Two lighting passes per pixel?
    - But: hierarchical culling for prelighting: Object → Vertex → Pixel
- Pros:
  - No additional information in G-Buffer: **bandwidth preserved**
  - Transparent for subsequent lighting pass
  - **Pipeline unification**: shadows, materials, shader combinations

DEFERRED LIGHTING AND ANTI-ALIASING

Aliasing sources

- Coarse surface sampling (rasterization)
  - Saw-like jaggy edges
  - Flickering of highly detailed geometry (foliage, gratings, ropes etc.) because of sparse sampling
    - Any post MSAA (including MLAA) won't help with that
- More aliasing sources
  - Sparse shading
    - Sudden spatial/temporal shading change
  - Sparse lighting etc.etc.

Hybrid anti-aliasing solution

- Post-process AA for near objects
  - Doesn‘t supersample
  - Works on edges
- Temporal AA for distant objects
  - Does temporal supersampling
  - Doesn‘t distinguish surface-space shading changes
- Separate it with stencil and non-jitterred camera

Post-process Anti-Aliasing

- Two h/w color spaces for free: linear/gamma
  - Solution to the equation

$$x_\gamma{}' = x'$$

  - Median (**linear space**):

$$x = \frac{5}{11} = 0.(45) \approx \frac{\mathbf{116}}{255}$$

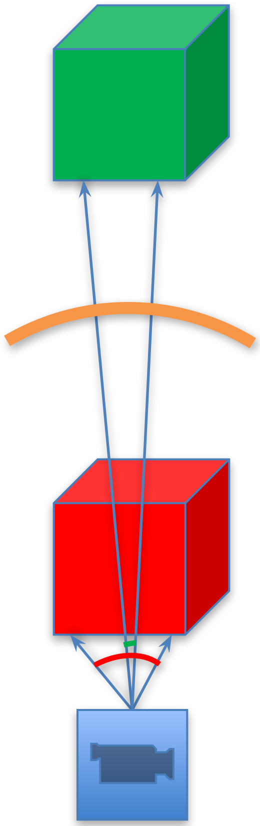
  - Choose the right color space based on histogram:
  - <u>Rule of thumb:</u> use linear if >75% of pixels are above the median

Temporal Anti-Aliasing

- Use temporal reprojection with cache miss approach
  - Store previous frame and depth buffer
  - Reproject the texel to the previous frame
  - Assess depth changes
  - Do an accumulation in case of small depth change
- Use sub-pixel temporal jittering for camera position
  - Take into account edge discontinuities for accumulation
- See [NVLTI07] and [HEMS10] for more details

Hybrid anti-aliasing solution

- Separation by distance guarantees small changes of view vector for distant objects
  - Reduces the fundamental problem of reverse temporal reprojection:
    view-dependent changes in shading domain
  - Separate on per-object base
    - Consistent object-space shading behavior
    - Use stencil to tag an object for temporal jittering

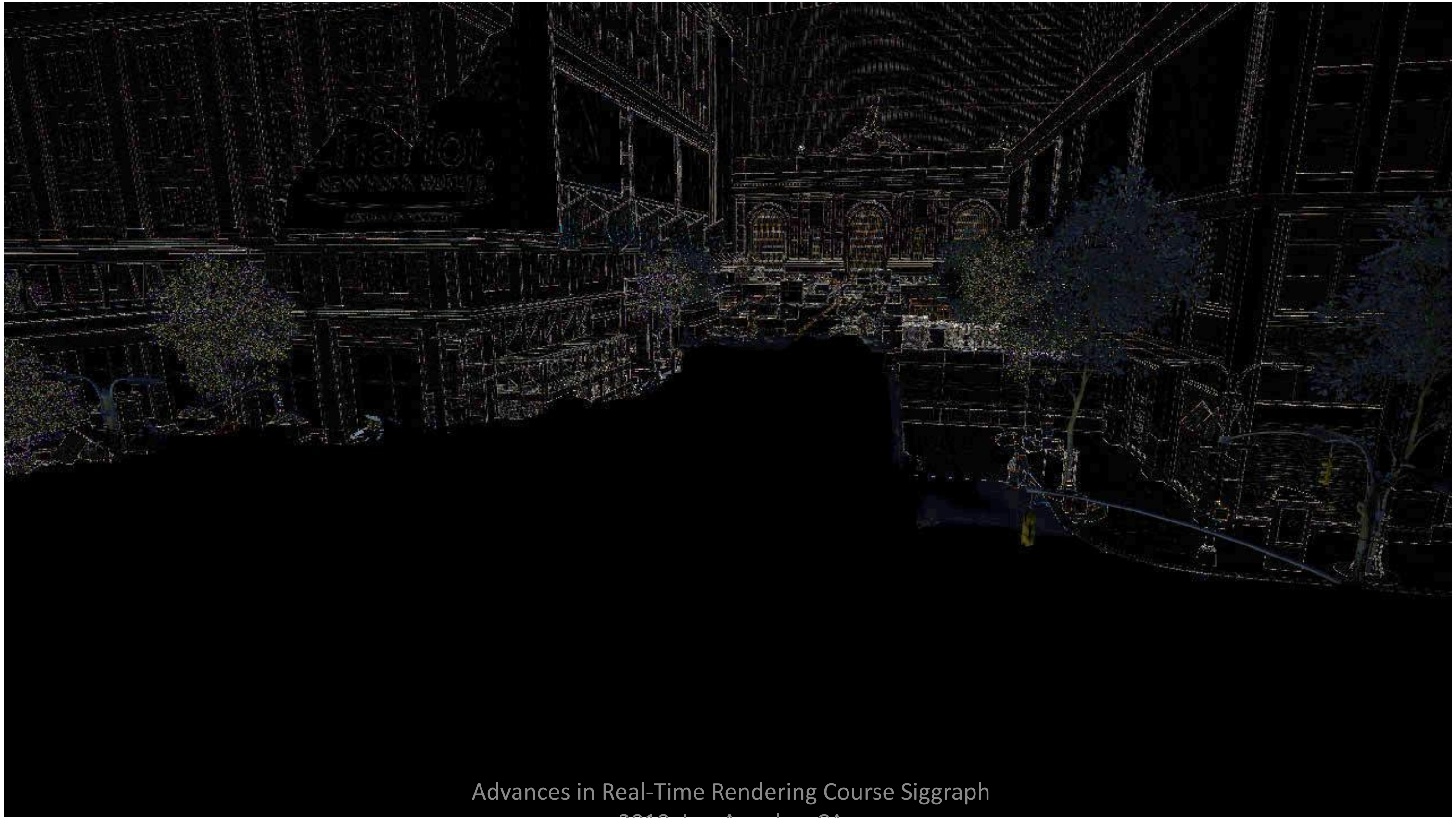


Hybrid anti-aliasing example

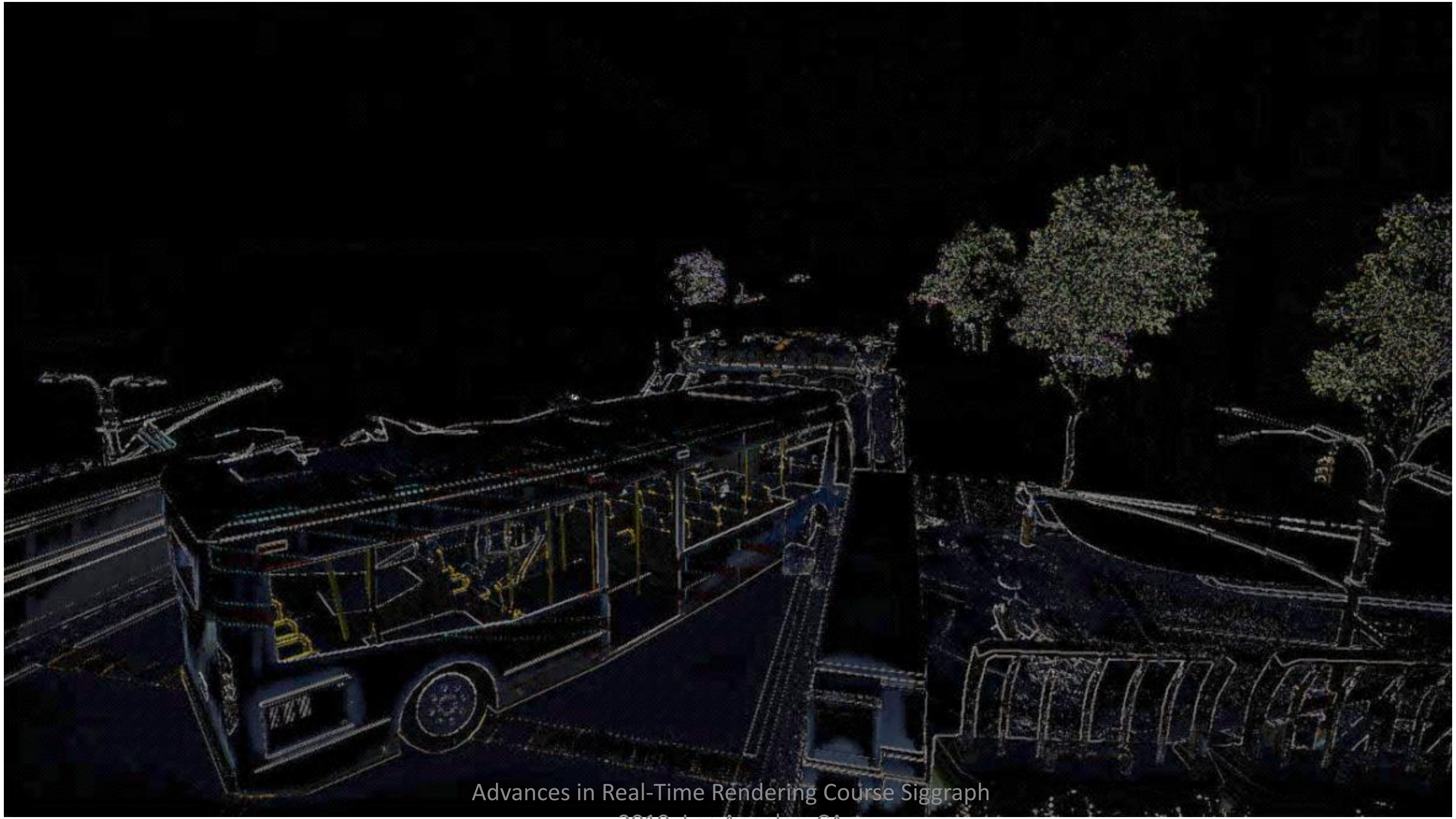
Hybrid anti-aliasing example

Hybrid anti-aliasing example

Temporal AA contribution



Edge AA contribution



Hybrid anti-aliasing video

Conclusion

- Texture compression improvements for consoles
- Deferred pipeline: some major issues successfully resolved
  - √ Bandwidth and precision
  - √ Anisotropic materials
  - √ Anti-aliasing
- Please look at the full version of slides (including texture compression) at: http://advances.realtimerendering.com/

Acknowledgements

- **Vaclav Kyba** from R&D for implementation of temporal AA
- **Tiago Sousa**, **Sergey Sokov** and the **whole Crytek R&D** department
- **Carsten Dachsbacher** for suggestions on the talk
- **Holger Gruen** for invaluable help on effects
- **Yury Uralsky** and **Miguel Sainz** for consulting
- **David Cook** and **Ivan Nevraev** for consulting on Xbox 360 GPU
- **Phil Scott, Sebastien Domine, Kumar Iyer** and the **whole Parallel Nsight team**

Thank you for your attention

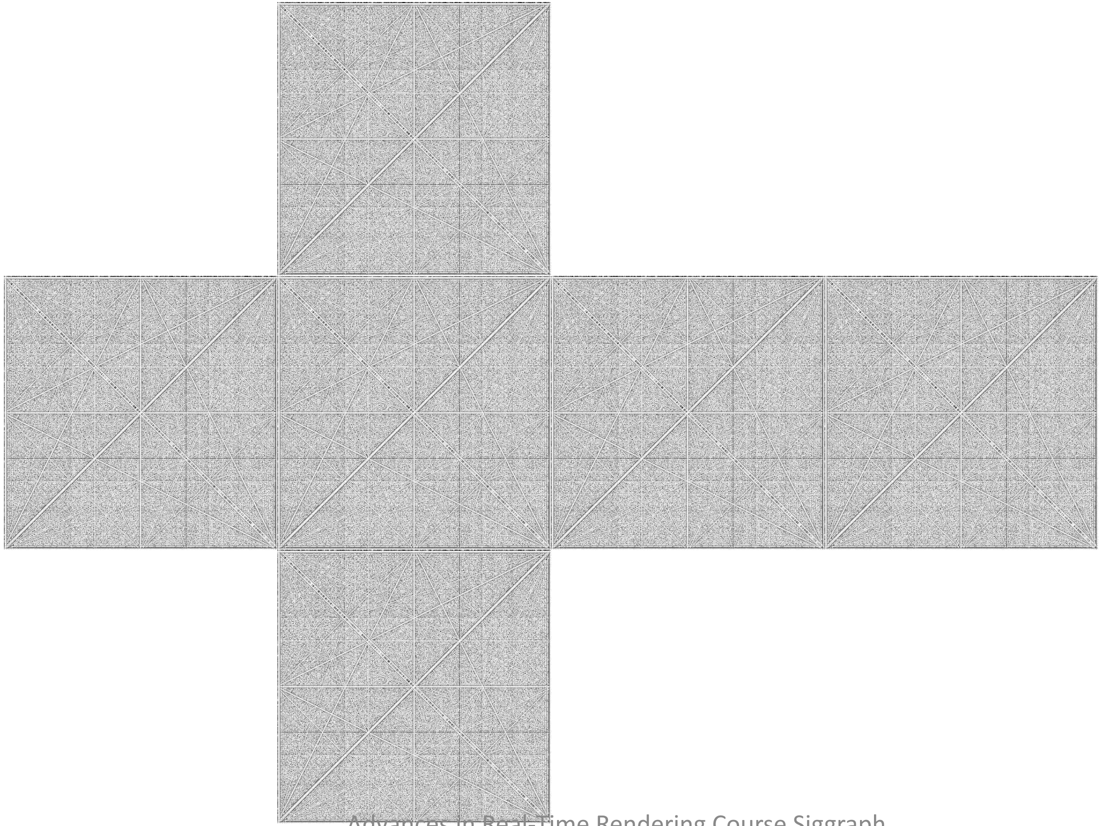
QUESTIONS?

APPENDIX A: BEST FIT FOR NORMALS

Function to find minimum error:

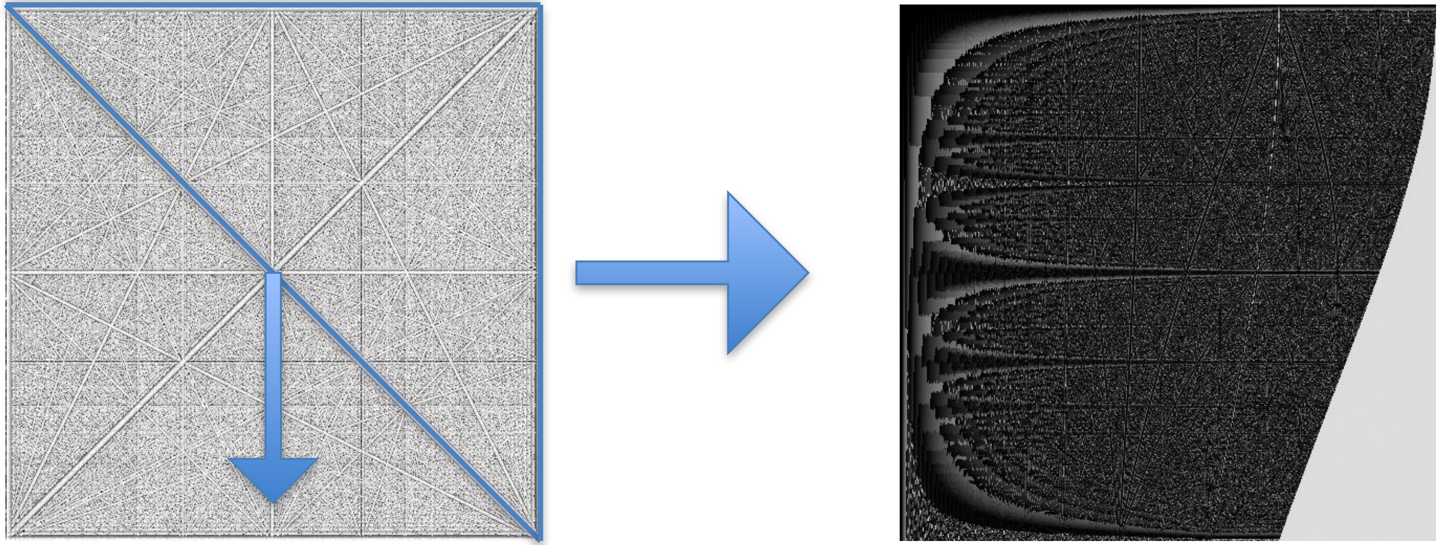
```
float quantize255(float c)
{
        float w = saturate(c * .5f + .5f);
        float r = round(w * 255.f);
        float v = r / 255.f * 2.f - 1.f;
        return v;
}

float3 FindMinimumQuantizationError(in half3 normal)
{
        normal /= max(abs(normal.x), max(abs(normal.y), abs(normal.z)));
        float fMinError = 100000.f;
        float3 fOut = normal;
        for(float nStep = 1.5f;nStep <= 127.5f;++nStep)
        {
                float t = nStep / 127.5f;

                // compute the probe
                float3 vP = normal * t;

                // quantize the probe
                float3 vQuantizedP = float3(quantize255(vP.x), quantize255(vP.y), quantize255(vP.z));

                // error computation for the probe
                float3 vDiff = (vQuantizedP - vP) / t;
                float fError = max(abs(vDiff.x), max(abs(vDiff.y), abs(vDiff.z)));

                // find the minimum
                if(fError < fMinError)
                {
                        fMinError = fError;
                        fOut = vQuantizedP;
                }
        }
        return fOut;
}
```

Cubemap produced with this function



Extract unique part

- Consider one face, extract non-symmetric part into 2D texture
  - Also divide y coordinate by x coordinate to expand the triangle to quad
  - To download this texture look at: http://advances.realtimerendering.com/



Function to fetch 2D texture at G-Buffer pass:

```
void CompressUnsignedNormalToNormalsBuffer(inout half4 vNormal)
{
  // renormalize (needed if any blending or interpolation happened before)
  vNormal.rgb = normalize(vNormal.rgb);
  // get unsigned normal for cubemap lookup (note the full float precision is required)
   half3 vNormalUns = abs(vNormal.rgb);
  // get the main axis for cubemap lookup
  half maxNAbs = max(vNormalUns.z, max(vNormalUns.x, vNormalUns.y));
  // get texture coordinates in a collapsed cubemap
  float2 vTexCoord = vNormalUns.z<maxNAbs?(vNormalUns.y<maxNAbs?vNormalUns.yz:vNormalUns.xz):vNormalUns.xy;
  vTexCoord = vTexCoord.x < vTexCoord.y ? vTexCoord.yx : vTexCoord.xy;
  vTexCoord.y /= vTexCoord.x;
  // fit normal into the edge of unit cube
  vNormal.rgb /= maxNAbs;
  // look-up fitting length and scale the normal to get the best fit
  float fFittingScale = tex2D(normalsSampler2D, vTexCoord).a;
  // scale the normal to get the best fit
  vNormal.rgb *= fFittingScale;
  // squeeze back to unsigned
  vNormal.rgb = vNormal.rgb * .5h + .5h;
}
```

References

- [CT81] *Cook, R. L., and Torrance, K. E.* 1981. “A reflectance model for computer graphics”, SIGGRAPH 1981
- [HEMS10] Herzog, R., Eisemann, E., Myszkowski, K., Seidel, H.-P. 2010. “Spatio-Temporal Upsampling on the GPU” I3D 2010.
- [LS10] *Loos, B.J. and Sloan, P.-P. 2010* “Volumetric Obscurance”, I3D symposium on interactive graphics, 2010
- [NVLTI07] Nehab, D., Sander, P., Lawrence, J., Tatarchuk, N., Isidoro, J. 2007. “Accelerating Real-Time Shading with Reverse Reprojection Caching”, Conference On Graphics Hardware, 2007
- *[RTDKS10] T. Ritschel, T. Thormählen, C. Dachsbacher, J. Kautz, H.-P. Seidel,* 2010. “Interactive On-surface Signal Deformation”, SIGGRAPH 2010
- [SELAN07] *Selan, J. 2007.* “Using Lookup Tables to Accelerate Color Transformations”, GPU Gems 3, Chapter 24.
- [WRGSG09] *Wang., J., Ren, P., Gong, M., Snyder, J., Guo, B. 2009.* “All-Frequency Rendering of Dynamic, Spatially-Varying Reflectance”, SIGGRAPH Asia 2009