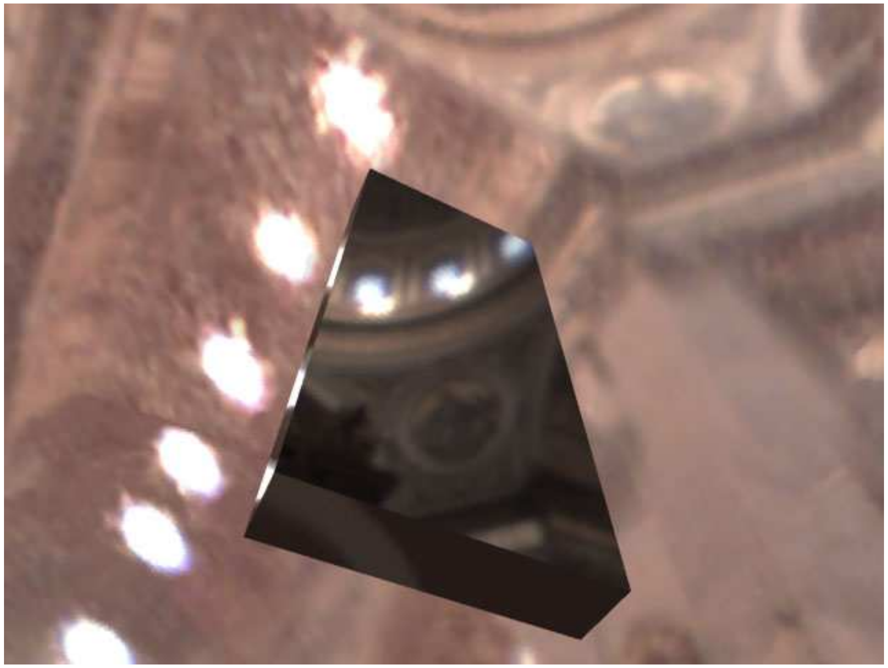
Signature Devices, inc.



High Dynamic Range Images

Kenneth Hurley - CEO

What we're going to cover

- Introduction to High Dynamic Range (HDR)
- DX7 implementation
- DX8 implementations
  - Fake HDR
  - Using HDR for Image Based Lighting
- DX9 Implementations
  - Fake HDR
  - Encoding Formats
  - HLSL implementations
- More Information

HDR Intro

- Developed by Paul E. Debevec and Jitendra Malik
  - http://www.debevec.org
- Radiance can vary beyond precision of 8 bits
- Encodes radiance in floating point values
- Demo at site uses Geforce2
- Commercial Licensing Required

Signature Devices, inc.

HDR Intro

- The human visual system adapts automatically to changes in brightness
- In photography, shutter speed and lens aperture are used to control the amount of light that reaches the film
- HDR imagery attempts to capture the full dynamic range of light in real world scenes
- Measures *radiance = amount of energy per unit time per unit solid angle per unit area* $W / (sr * m^2)$
  - W = *Radiant flux*
  - sr = *solid angle*
  - m^2 = area
- 8 bits is not enough!

Signature Devices, inc.

Why do we need HDR

- It effectively allows us to change the exposure *after* we've taken/rendered the picture
- Dynamic adaptation effects – e.g. moving from a bright outdoor environment to indoors
- Allows physically plausible image-based lighting
- BRDFs may need high dynamic range
- Enables realistic optical effects – glows around bright light sources, more accurate motion blurs

HDR Terminology

- Gaussian (Blur)
  - Blurs image
    - averages pixels around a pixel by sampling
- Exposure
  - Similar to photograph chemical process
  - Digitial photographs clamp captured light values
  - Multiple photographs are taken (exposures)
  - Recombined with software for fuller range of luminance values

HDR Terminology Continued

Tone Mapping

- Tone mapping scales the RGB values of an image, which might be too bright or too dark to be displayed
  - Techniques used to map HDR images to RGB 8 bit monitor images
- “key value” or “neutral value
  - The log-average luminance of the scene
  - DX9 Demos allow changing this value

Signature Devices, inc.

HDR Encoding

- Eyes sensitivity to luminance suggests we must encode 9,900 values if we use linear steps for luminance
- If not linear then only 460 values are requires (9 bits)
- Eye is very sensitive to luminance changes
- Less sensitive to color changes
- OpenEXR Format

HDR on DX7

- "Real-Time High Dynamic Range Imagery", Cohen, Tchou, Hawkins, Debevec, Eurographics 2001
- Splits HDR images into several 8-bit textures
  - Recombines using register combiners on DX7 capable hardware
- Doesn't automatically adjust exposure
  - Requires different combiner setups for different exposure ranges, so exposure can only be changed on a per-primitive basis

HDR on DX7

HDR on DX8 class hardware

- Developed by Simon Green at NVIDIA
- DX8 that supports a 16-bit format known as HILO can be used
- Stores 2 16-bit components: (HI, LO, 1)
- Filtered by hardware at 16-bit precision
- We can also use this format to store high(er) dynamic range imagery
- Remap floating point HDR data to gamma encoded 16-bit fixed-point range [0, 65535]
- HILO only stores two components so we need two HILO textures to store RGB

HDR on DX8 class hardware

- To display the image, we need to multiply the HDR radiance values by the exposure factor, and then re-map them to the displayable [0,255] range
- This can be achieved using the texm3x2tex pixel shader operation
- Exposure is sent as texture coordinates, the dot product performs the multiply for both channels
- We create a 2D texture that maps the result back to displayable values

HDR on DX8 class hardware

- Psuedo Code

```
0: hilo = texture_cube_map(hdr_texture, s0, t0, r0)
1: dot1 = s1*hi + t1*lo + r1*1.0;   // = r_exposure*r + 0 + r_bias
2: dot2 = s2*hi + t2*lo + r2*1.0;   // = 0 + g_exposure*g + g_bias
   color = texture_2d(lut_texture, dot1, dot2)
```

- Pixel Shader code

```
ps_1_1
tex t0                    // Grab hilo data from cubemap
texm3x2pad  t1, t0   // = r_exposure*r + 0 + r_bias
texm3x2tex  t2, t0   // 0 + g_exposure*g + g_bias
mov r0, t2
```

Signature Devices, inc.

HDR on DX8 class hardware

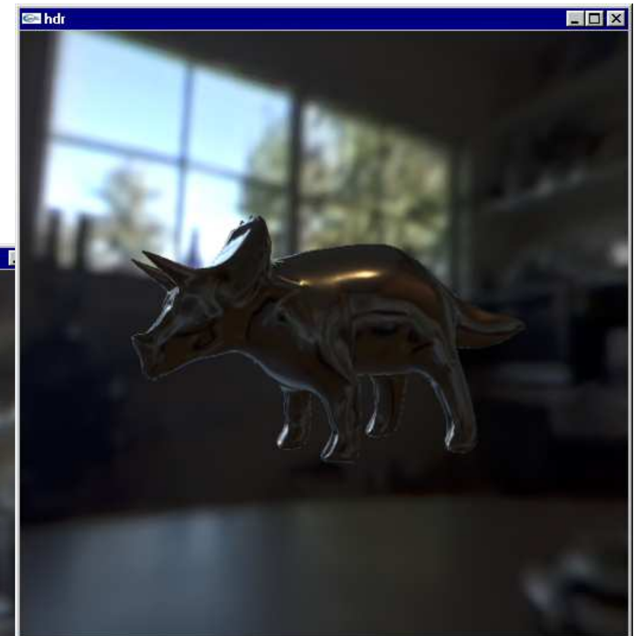
- Requires 2 passes to render RGB, using D3DRS_COLORWRITEENABLE to mask off color channels
- First pass renders R and G:
  - texcoord1 = (r_exposure, 0.0, r_bias)
  - texcoord2 = (0.0, g_exposure, g_bias)
- Second pass renders B:
  - texcoord1 = (0, 0, 0)
  - texcoord2 = (b_exposure, 0.0, b_bias)

HDR on DX8 class hardware

Exposure .25

Exposure 0.0625

Exposure 0.015625

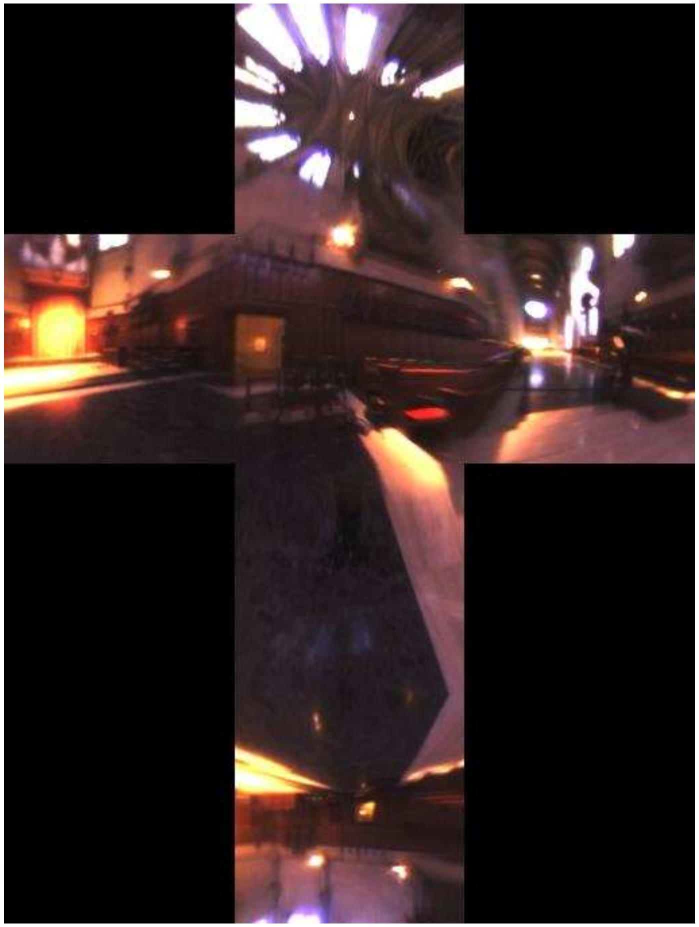


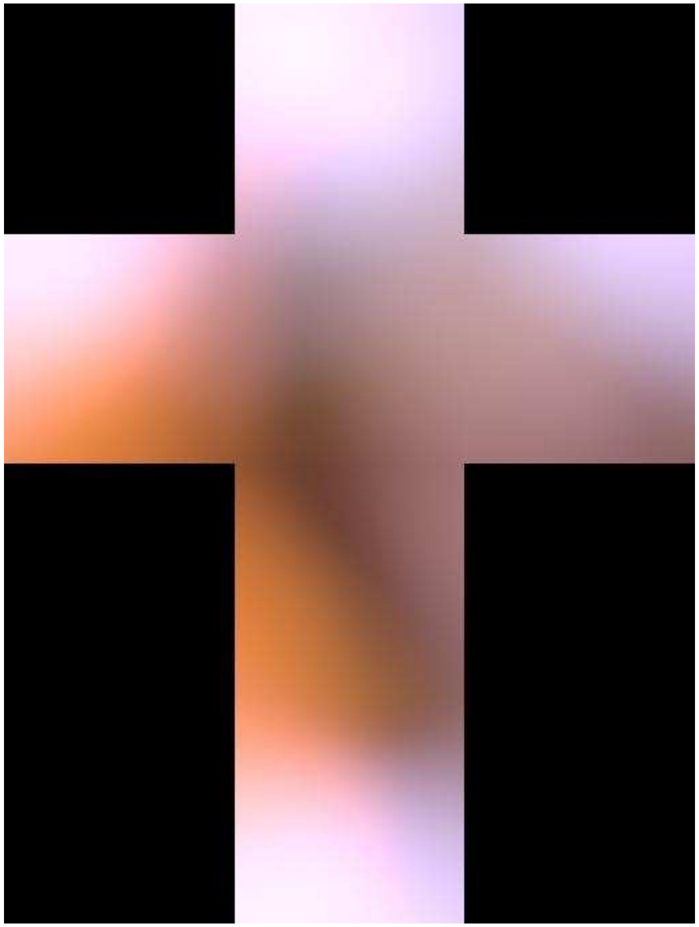
Signature Devices, inc.

Image Based Lighting use HDR on DX8 class hardware

- Lighting synthetic objects with “real” light
- An environment map represents all light arriving at a point for each incoming direction
- By convolving (blurring) an environment map with the diffuse reflection function (N.L) we can create a diffuse reflection map
- Indexed by surface normal N, this gives the sum of N.L for all light sources in the hemisphere
- Low freq - cube map can be small - e.g. 32x32x6
- HDRShop will do this for you

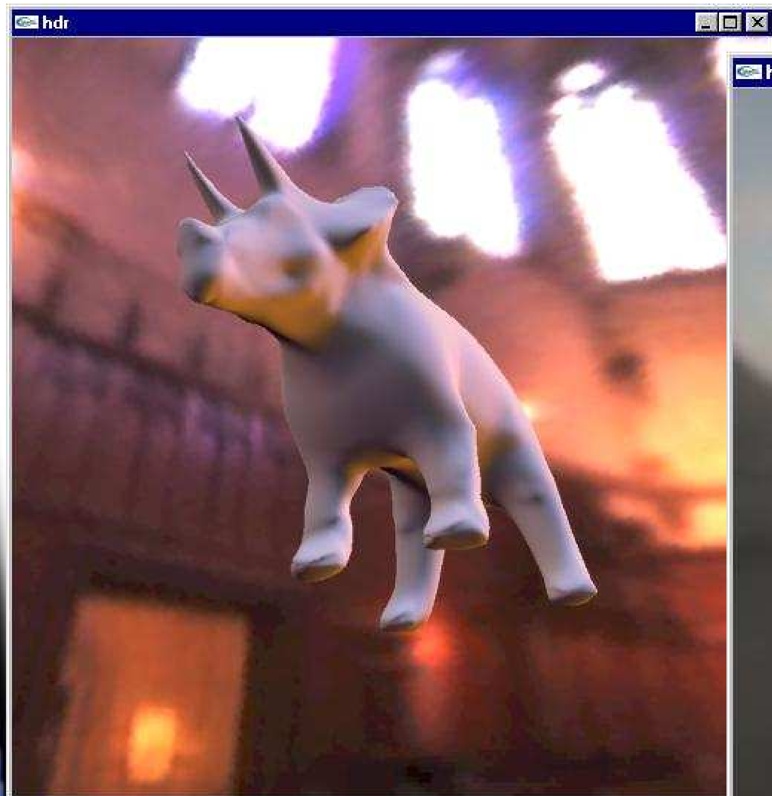
Image Based Lighting use HDR on DX8 class hardware





Signature Devices, inc.

Image Based Lighting use HDR on DX8 class hardware



Signature Devices, inc.

Fake HDR on DX8 class hardware

- Masaki Kawase techinque
  - Used in XBOX Wreckless: Yakuza Missions
  - Can be implemented in 1.1 shader
- Blur filters up to 8 passes
- Simple Tone map
  - LERPS between original and blurred image
- DEMO, RGBA and RGBE

HDR on DX9 class hardware

- Easier to implement
- Floating point buffers
- HLSL available

Signature Devices, inc.

Realtime HDR on DX9 class hardware

- Masaki Kawase is at it again
- Demo

HDR on DX9 class hardware

- Format possibilities
  - RGB16
    - 16-bit per channel integer format
      - decoded.rgb = encoded.rgb dot max_value
  - RGBE
    - Compressed logarithmic values with E being shared exponent calculated from RGB
      - decoded.rgb = encoded.rgb * $2^{encoded.a}$
  - FP16
    - Partial precision floating point values
  - FP32
    - Full Precision floating point values

Signature Devices, inc.

HDR on DX9 class hardware

- Simple Code (ATI RenderMonkey Sample)
  - Render the scene with HDR values into a floating point buffer.
  - Down-sample this buffer to 1/4th size (1/2 width and 1/2 height) and optionally suppress low values to get only brightest parts
  - Blur image (bloom filter) Best to do it X then Y, to reduce texture lookups
  - Tone map the blurred image after compositing it with the original image.

Generic Vertex Shader

```
float4x4 matViewProjection;

struct VS_INPUT
{
   float3 Pos:      POSITION;
};

struct VS_OUTPUT
{
   float4 Pos:       POSITION;
   float2 TexCoord : TEXCOORD0;
};

VS_OUTPUT vs_main( VS_INPUT In )
{
   VS_OUTPUT Out;

   Out.Pos.xy = sign(In.Pos);
   Out.Pos.z = 1.0;
   Out.Pos.w = 1.0;

   Out.TexCoord.x = Out.Pos.x * 0.5 + 0.5;
   Out.TexCoord.y = 1.0 - (Out.Pos.y * 0.5 + 0.5);

   return Out;
}
```

HLSL Blur Horizontal Pixel Shader

```
sampler2D Src;

float4 gaussFilter[7] =
{
  -3.0, 0.0, 0.0,  1.0/64.0,
  -2.0, 0.0, 0.0,  6.0/64.0,
  -1.0, 0.0, 0.0, 15.0/64.0,
   0.0, 0.0, 0.0, 20.0/64.0,
   1.0, 0.0, 0.0, 15.0/64.0,
   2.0, 0.0, 0.0,  6.0/64.0,
   3.0, 0.0, 0.0,  1.0/64.0
 };

float texScaler = 1.0/128.0;
float texOffset = 0.0;

struct PS_INPUT
{
   float2 TexCoord : TEXCOORD0;
};
```

Signature Devices, inc.

HLSL Blur Horizontal Pixel Shader (Cont)

```
struct PS_OUTPUT
{
   float4 Color : COLOR;
};

PS_OUTPUT ps_main( PS_INPUT In )
{
   PS_OUTPUT Out;

   float4 color = 0.0;

   int i;
   for (i=0;i<7;i++)
   {
      color += tex2D(Src,float2(In.TexCoord.x + gaussFilter[i].x * texScaler +
    texOffset,
                              In.TexCoord.y + gaussFilter[i].y * texScaler +
    texOffset)) *
                     gaussFilter[i].w;
   } // End for

   Out.Color = color * 4.0;

   return Out;
}
```

Signature Devices, inc.

Final Pixel Shader Tone Mapping

```
float Exposure;
sampler2D SrcHDR;
sampler2D SrcColor;

struct PS_INPUT
{
  float2 TexCoord : TEXCOORD0;
};

struct PS_OUTPUT
{
  float4 Color : COLOR;
};

PS_OUTPUT ps_main( PS_INPUT In )
{
  PS_OUTPUT Out;

  float4 color  = tex2D(SrcColor,In.TexCoord);
  float4 scaler = tex2D(SrcHDR,In.TexCoord) * 2.0;

  Out.Color = color * ( ( 1.0 + scaler.a ) * Exposure );

  return Out;
}
```

Optimizations

- Down-sample image first
  - Reduces the texture samples from 32 pixels to 8 samples
- Blur in X, then in Y
  - 2n texture look-ups rather than n*n

Render Monkey Demo

- DEMO

Final Thoughts

- High Dynamic Range can be accomplished on all current hardware
  - Implementations available for DX7
  - Implementations available for DX8
  - Implementations available for DX9
  - So no excuses.
- IBL or IBR
  - Can make use of HDR tools
  - Look very good
- Precomputed Radiance Transfer

More information on HDR

- Programming Vertex and Pixel Shader, Wolfgang Engel ISBN 1-58450-349-1
- http://developer.nvidia.com
- http://www.ati.com/developer
- DX9 Summer 2004 SDK
- http://www.debevec.org
- Masaka Kawase website http://www.daionet.gr.jp/~masa/rthdribl/

Signature Devices, inc.

Software support for HDR

- HDRShop - http://www.ict.usc.edu/graphics/HDRShop/
- Rendermonkey – http://www.ati.com/developer
- NVSDK – http://developer.nvidia.com
- OpenEXR http://www.openexr.net/
- DX9 Summer 2004 SDK

Signature Devices, inc.

Questions

- klhurley@signaturedevices.com

?

Signature Devices, inc.