
MARCH 5-9
2007
SAN FRANCISCO
MOSCONE CENTER
TAKE CONTROL
www.gdconf.com

Post-Processing Pipeline

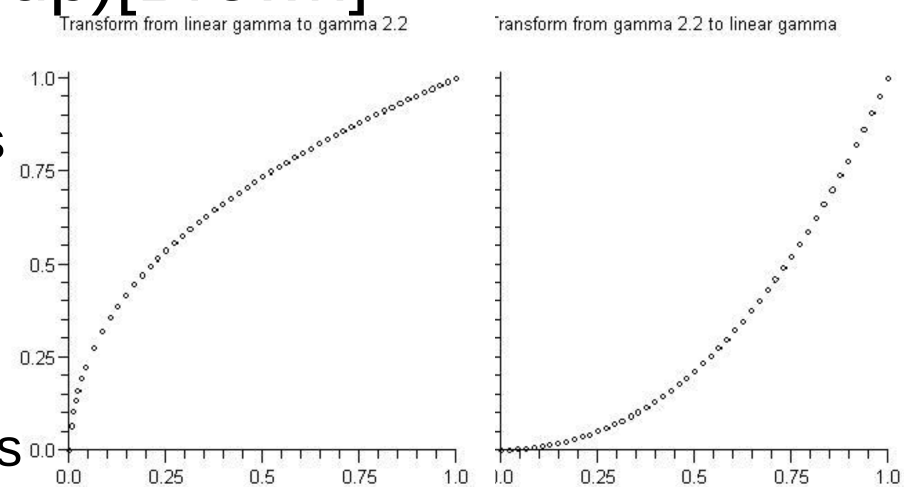
- by Wolfgang Engel (Rockstar Games)
  - GDC San Francisco
  - March 5th, 2007

Agenda

- Gamma control
- Contrast
- High-Dynamic Range Rendering
- Depth of Field

Gamma Control

- Gamma 2.2 purpose: make RGB look good with 8-bits per channel
- Disadvantage: RGB color operations do not look right (adds up)[Brown]
    - dynamic lighting
    - several light sources
    - shadowing
    - texture filtering
    - alpha blending
    - advanced color filters



Gamma Control

- We want: renderer without gamma correction == gamma 1.0
- Art Pipeline is most of the time running gamma 2.2 everywhere
- ->convert from gamma 2.2 to 1.0 while fetching textures and color values and back to gamma 2.2 at the end of the renderer

Gamma Control

- Converting to gamma 1.0 [Stokes]

  Color = ((Color <= 0.03928) ? Color / 12.92 : pow((Color + 0.055) / 1.055, 2.4))

- Converting to gamma 2.2

  Color = (Color <= 0.00304) ? Color * 12.92 : (1.055 * pow(Color, 1.0/2.4) - 0.055);

- Hardware can convert textures and the end result… but some hardware uses linear approximations here
- Vertex colors still need to be converted “by hand”

Gamma Control

- Problem: you need more precision than 8-bit per channel
- Solution: shown in HDR slides

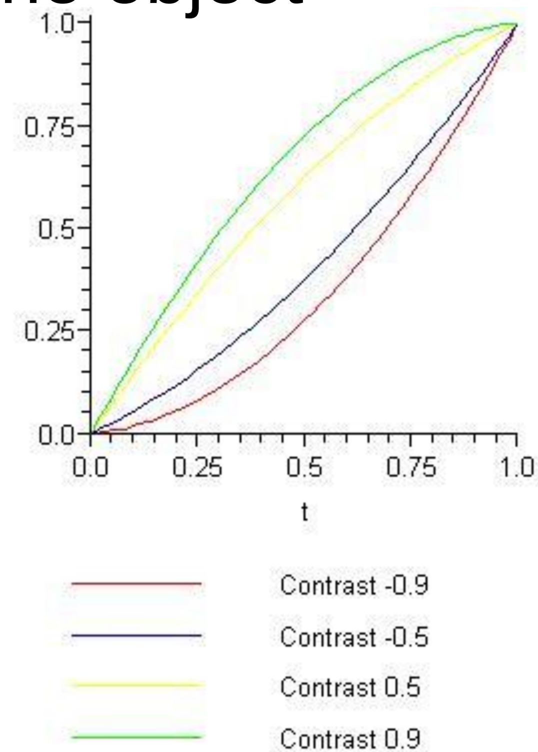
Contrast

- Brain determines color of objects with color of the surrounding of the object
- Cubic Polynomial

$$R_{Gamma1.0} = R_{Gamma1.0} - Contrast * (R_{Gamma1.0} - 1) * R_{Gamma1.0} * (R_{Gamma1.0} - 0.5)$$

$$G_{Gamma1.0} = G_{Gamma1.0} - Contrast * (G_{Gamma1.0} - 1) * G_{Gamma1.0} * (G_{Gamma1.0} - 0.5)$$

$$B_{Gamma1.0} = B_{Gamma1.0} - Contrast * (B_{Gamma1.0} - 1) * B_{Gamma1.0} * (B_{Gamma1.0} - 0.5)$$



High-Dynamic Range Rendering

- Ansel Adam’s Zone System [Reinhard]
- Requirement list:
  - Data with higher range than 0..1
  - Tone mapping operator to compress HDR to LDR
  - Light adaptation
  - Glaring under intense lighting
  - Blue shift and night view -> low lighting conditions

High-Dynamic Range Rendering

- Data with higher range than 0..1
  - Storing High-Dynamic Range Data in Textures
    - RGBE - 32-bit per pixel
    - DXGI_FORMAT_R9G9B9E5_SHAREDEXP - 32-bit per pixel
    - DXT1 + quarter L16 – 8-bit per pixel
    - DXT1: storing common scale + exponent for each of the color channels in a texture by utilizing unused space in the DXT header – 4-bit per-pixel
    - -> Challenge: gamma control -> calc. exp. without gamma
  - Keeping High-Dynamic Range Data in Render Targets
    - 10:10:10:2 (DX9: MS, blending, no filtering)
    - 7e3 format XBOX 360: configure value range & precision with color exp. Bias [Tchou]
    - 16:16:16:16 (DX9: some cards: MS+blend others filter+blend)
    - DX10: 11:11:10 (MS, source blending, filtering)

High-Dynamic Range Rendering

- HDR data in 8:8:8:8 Render Targets

Color Space	# of cycles (encoding)	Bilinear Filtering	Blur Filter	Alpha Blending
RGB	-	Yes	Yes	Yes
HSV	~34	Yes	No	No
CIE Yxy	~19	Yes	Yes	No
L16uv*	~19	Yes	Yes	No
RGBE	~13	No	No	No

  - *based on Greg Wards LogLuv model

- RGB12A2 for primary render target:
  - Two 8:8:8:8 render targets
  - 1-8 bits in render target 0 / 4 – 12 bits in render target 1
  - Overlap of 4 bits for alpha blending

High-Dynamic Range Rendering

- Tone mapping operator to compress HDR to LDR
  - Luminance Transform
  - Range Mapping

High-Dynamic Range Rendering

- Convert whole screen to an average luminance

$$Lum_{avg} = \exp\left(\frac{1}{N}\sum_{x,y}\log(\delta + Lum(x,y))\right)$$

- Logarithmic average not arithmetic average -> non-linear response of the eye to a linear increase in luminance

High-Dynamic Range Rendering

- To convert RGB to Luminance [ITU1990]
- RGB->CIE XYZ->CIE Yxy

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124, 0.3576, 0.1805 \\ 0.2126, 0.7152, 0.0722 \\ 0.0193, 0.1192, 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$Y = Y$$

$$x = X / (X + Y + Z)$$

$$y = Y / (X + Y + Z)$$

- CIE Yxy->CIE XYZ->RGB

$$X = x * (Y / y)$$

$$Y = Y$$

$$Z = (1 - x - y) * (Y / y)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.2405, -1.5371, -0.4985 \\ -0.9693, 1.8760, 0.0416 \\ 0.0556, -0.2040, 1.0572 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

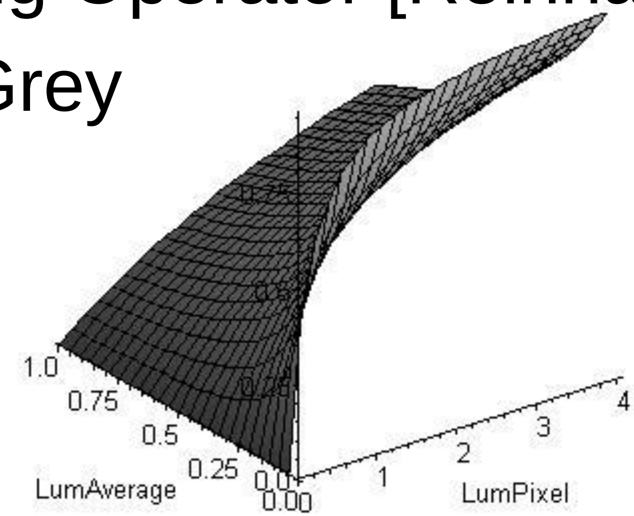
High-Dynamic Range Rendering

- Simple Tone Mapping Operator [Reinhard]
- Scaling with MiddleGrey

$$Lum_{Scaled} = \frac{Lum_{Image} * MiddleGrey}{Lum_{Average}}$$

- Map range from 0..1

$$Lum_{Compressed} = \frac{Lum_{Scaled}}{1 + Lum_{Scaled}}$$

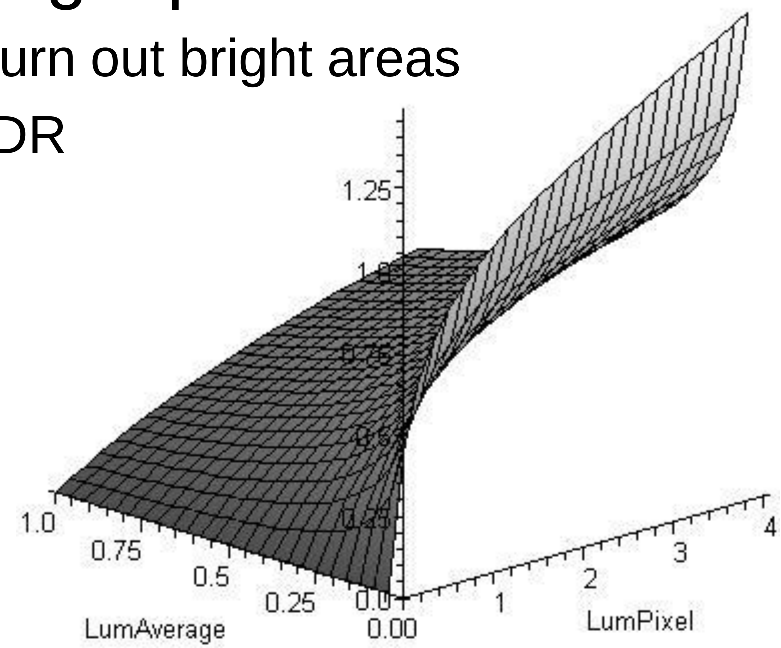


High-Dynamic Range Rendering

- Advanced Tone Mapping Operator
  - Artistically desirable to burn out bright areas
  - Source art not always HDR

$$Lum_{Compress} = \frac{Lum_{Scaled}\left(1 + \frac{Lum_{Scaled}}{L^2_{White}}\right)}{1 + Lum_{Scaled}}$$

  - Leaves 0..1



High-Dynamic Range Rendering

- Light Adaptation
  - Re-use luminance data to mimic light adaptation of the eye -> cheap
  - Temporal changes in lighting conditions
    - Day -> Night: Rods ~30 minutes
    - Outdoor <-> Indoor: Cones ~few seconds
  - Game Scenarios:
    - Outdoor <-> Indoor
    - Weather Changes
    - Tunnel drive

High-Dynamic Range Rendering

- Exponential decay function [Pattanaik]

$$Lum_{Adapted(i)} = Lum_{Adapted(i-1)} + (Lum_{Average} - Lum_{Adapted})(1 - e^{-dt * \tau})$$

  - Adapted luminance replaces average luminance in previous equations
  - Frame-rate independent
  - Adapted luminance chases average luminance
    - Stable lighting conditions -> the same
- tau interpolates between adaptation rates of cones and rods

$$\tau = p * \tau_{Rods} + (1 - p) * \tau_{Cones}$$

- 0.2 for rods / 0.4 for cones

High-Dynamic Range Rendering

- Luminance History function [Tchou]
  - Even out fast luminance changes (flashes etc.)
  - Keeps track of the luminance of the last 16 frames

$$Lum_{Adapted(i)} = \begin{pmatrix} for\left(\sum_{i=1}^{16} Lum_{Adapted(i)} >= Lum_{Adapted}\right) == 16 \,\|\, 0 \\ Lum_{Adapted(i-1)} + (Lum_{Average} - Lum_{Adapted})(1 - e^{-dt * \tau}) \\ otherwise \\ Lum_{Adapted(i-1)} \end{pmatrix}$$

  - If the last 16 values >= || < current adapted luminance -> run light adaptation
  - If some of the 16 values are going in different directions
    -> no light adaptation
- Runs only once per frame -> cheap

High-Dynamic Range Rendering

- Glaring

  Intense lighting -> optic nerve of the eye overloads

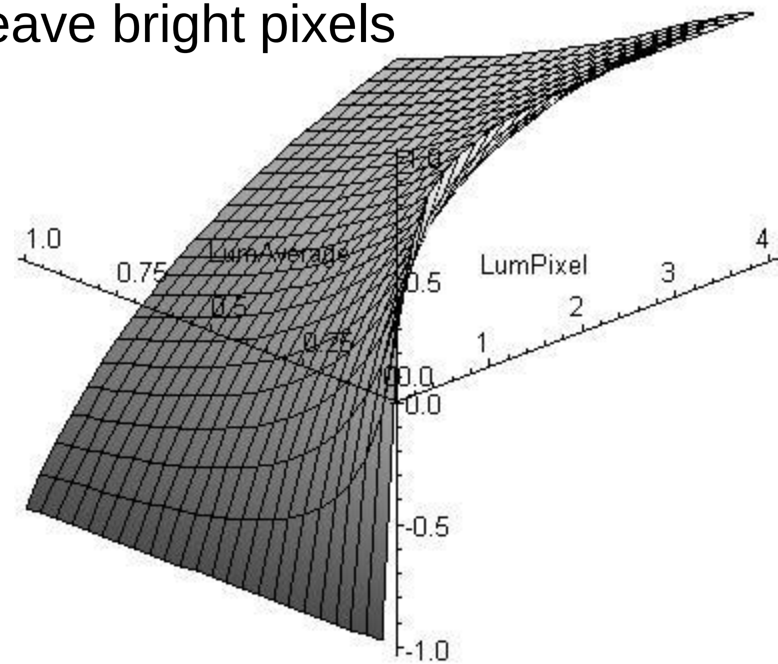
  - Bright pass filter
  - Gaussian convolution filter to bloom

High-Dynamic Range Rendering

- Bright pass filter
  - Compresses dark pixels leave bright pixels

$$Lum_{Threshold} = \max(Lum_{Scaled}(1.0 + \frac{Lum_{Scaled}}{White^2_{BrightPass}}) - T, 0.0)$$

$$Lum_{BrightPass} = \frac{Lum_{Threshold}}{O + Lum_{Threshold}}$$



Threshold 0.5 Offset = 1.0

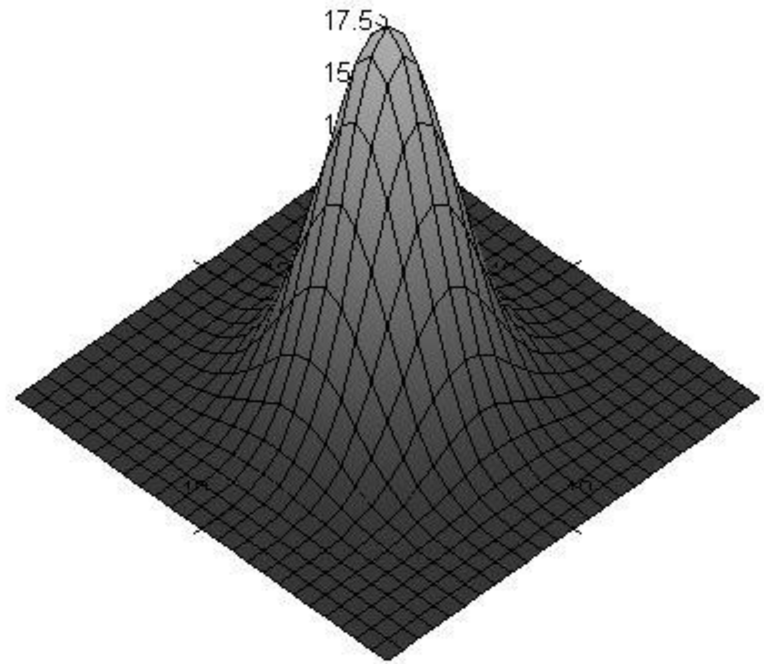
  - Same tone mapping operator as in tone mapping -> consistent

High-Dynamic Range Rendering

- Gauss filter

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} * \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}$$

- σ - standard deviation
- x, y coordinates relativ to center of filter kernel



High-Dynamic Range Rendering

- Scotopic View
  - Contrast is lower
  - Visual acuity is lower
  - Blue shift
- Convert RGB to CIE XYZ
- Scotopic Tone Mapping Operator [Shirley]

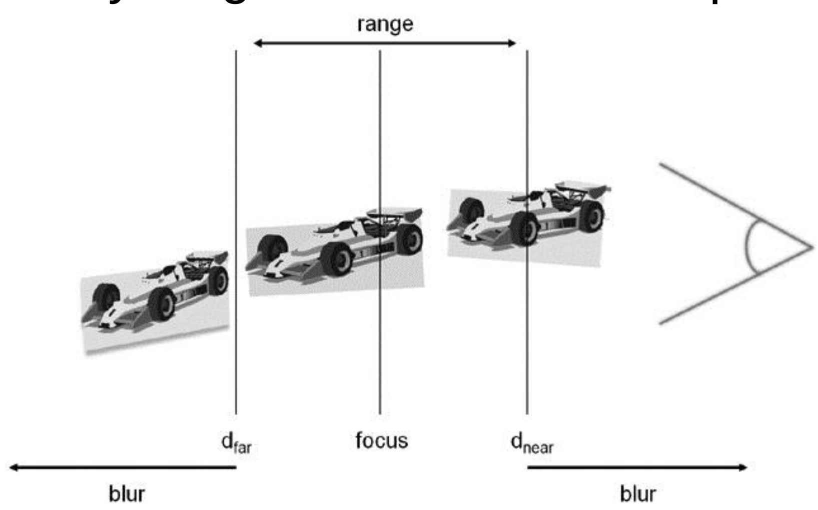
$$V = Y[1.33(1 + \frac{Y + Z}{X}) - 1.68]$$

- Multiply with a grey-bluish color

$$NightColor_{Red} = V1.05$$
$$NightColor_{Blue} = V0.97$$
$$NightColor_{Green} = V1.27$$

Depth of Field

- Range of acceptable sharpness == Depth of Field see [Scheuermann] and [Gillham]
  - Define a near and far blur plane
  - Everything in front of the near blur plane and everything behind the far blur plane is blurred



Depth of Field

- Convert depth buffer values into camera space

$$[x, y, z, 1]\begin{bmatrix} Zoom_x & 0 & 0 & 0 \\ 0 & Zoom_y & 0 & 0 \\ 0 & 0 & Q & 1 \\ 0 & 0 & -Z_n Q & 0 \end{bmatrix} = [x', y', z', z]$$

where

$$Q = \frac{Z_f}{Z_f - Z_n}$$

Z_f = far clip plane

Z_n = near clip plane

- Multiply vector with third column of proj. matrix

$$z' = zQ - Z_n Q \qquad (x.1)$$

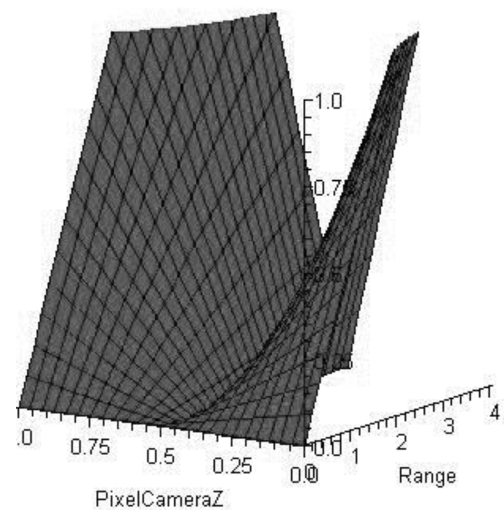
$$Z_d = -\frac{zQ + (-Z_n Q)}{z} \qquad (x.2)$$

$$z = \frac{-Z_n Q}{Z_d - Q} \qquad (x.3)$$

  - x.2 shows how to factor in / w here w = z
  - x.3 result

Depth of Field

- Applying Depth of Field
  - Convert to Camera Z == pixel distance from camera
    float PixelCameraZ = (-NearClip * Q) / (Depth - Q);
  - Focus + Depth of Field Range [DOFRM]
    lerp(OriginalImage, BlurredImage, saturate(Range * abs(Focus - PixelCameraZ)));
    -> Auto-Focus effect possible



  - Color leaking: change draw order or ignore it

Summary

- Use gamma control
- Reinhard's tone mapper was not meant for games … do your own [Reinhard05]
- Depth of field is great … smoother blend would be good - adjust filter kernel based on distance

Thank you

- wolf@shaderx.com
- ShaderX^{6} Call for Proposals
  - Deadline April

References

- [Brown] Simon Brown, “Gamma-Correct Rendering”, http://www.sjbrown.co.uk/?article=gamma
- [DOFRM] Depth of Field Effect in RenderMonkey 1.62
- [Gilham] David Gilham, "Real-Time Depth-of-Field Implemented with a Post-Processing only Technique", ShaderX5: Advanced Rendering, Charles River Media / Thomson, pp 163 - 175, ISBN 1-58450-499-4
- [ITU1990] ITU (International Tlecommunication Union), Geneva. ITU-R Recommendation BT.709, Basic Parameter Values for the HDTV Standard for the Studio and for International Programme Exchange, 1990 (Formerly CCIR Rec. 709).
- [Pattanaik] Sumanta N. Pattanaik, Jack Tumblin, Hector Yee, Donald P. Greenberg, "Time Depen dent Visual Adaptation For Fast Realistic Image Display", SIGGRAPH 2000, pp 47-54
- [Reinhard] Erik Reinhard, Michael Stark, Peter Shirley, James Ferwerda, "Photographic Tone Reproduction for Digital Images", http://www.cs.utah.edu/~reinhard/cdrom/
- [Reinhard05] Erik Reinhard, Kate Devlin, “Dynamic Range Reduction inspired by Photoreceptor Physiology”, https://www.cs.bris.ac.uk/~reinhard/papers/tvcg2005.pdf
- [Scheuermann] Thorsten Scheuermann and Natalya Tatarchuk, ShaderX3: Advanced Rendering With DirectX And OpenGL, Charles River Media, pp 363 - 377, ISBN 1-58450-357-2
- [Shirley] Peter Shirley et all., "Fundamentals of Computer Graphics", Second Edition, A.K. Peters pp. 543 - 544, 2005, ISBN 1-56881-269-8
- [Stokes] Michael Stokes (Hewlett-Packard), Matthew Anderson (Microsoft), Srinivasan Chandrasekar (Microsoft), Ricardo Motta (Hewlett-Packard) “A Standard Default Color Space for the Internet - sRGB“ http://www.w3.org/Graphics/Color/sRGB.html
- [Tchou] Chris Tchou, HDR the Bungie Way, http://msdn2.microsoft.com/en-us/directx/aa937787.aspx