Rendering in Game

Tanasai Sucontphunt

Today

- High Dynamic Range Image (HDRI)
- HDR Skybox Lab
- Image Space Rendering
- Image Effect Lab
- Lab 2

High Dynamic Range Image (HDRI)

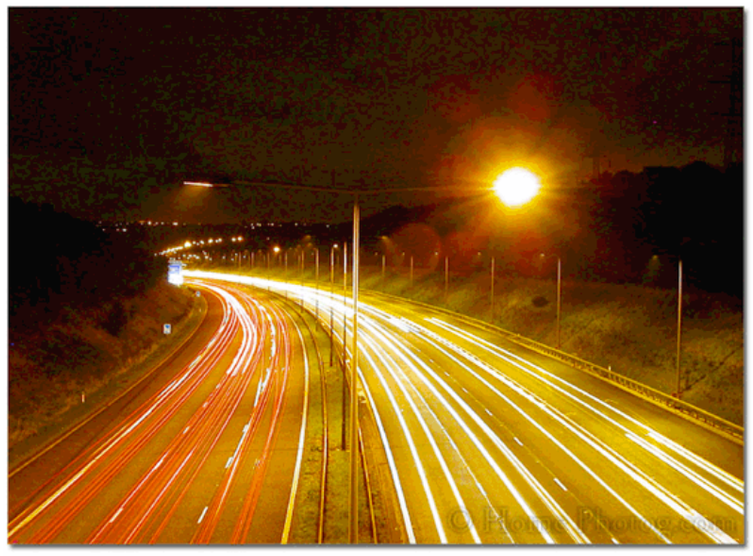
- High Definition (HD) vs High Dynamic Range (HDR)?
- Low Dynamic Range Image = 8 bits/channel
  - BMP (lossless), JPEG (lossy), etc.
- Exposure = amount of light capturing at a camera sensor which depending on:
  - Shutter Speed
  - Aperture
  - ISO Speed

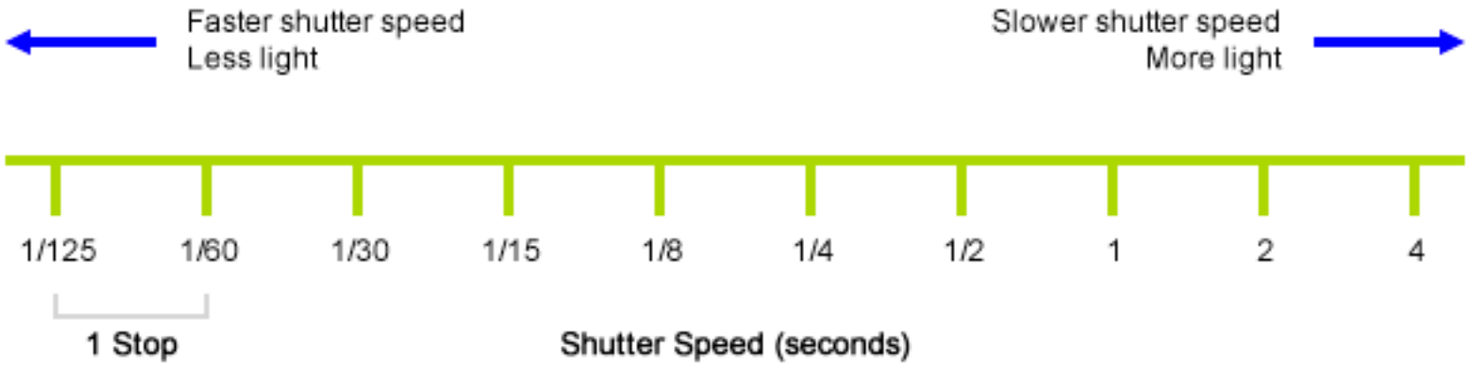
http://www.toptechdaily.com/hdr-

- +1 Stop = capturing twice as much light from the previous stop

Shutter Speed

- How long your camera's shutter is left open
  - longer = more light
- 1 stop = 2 x shutter speed

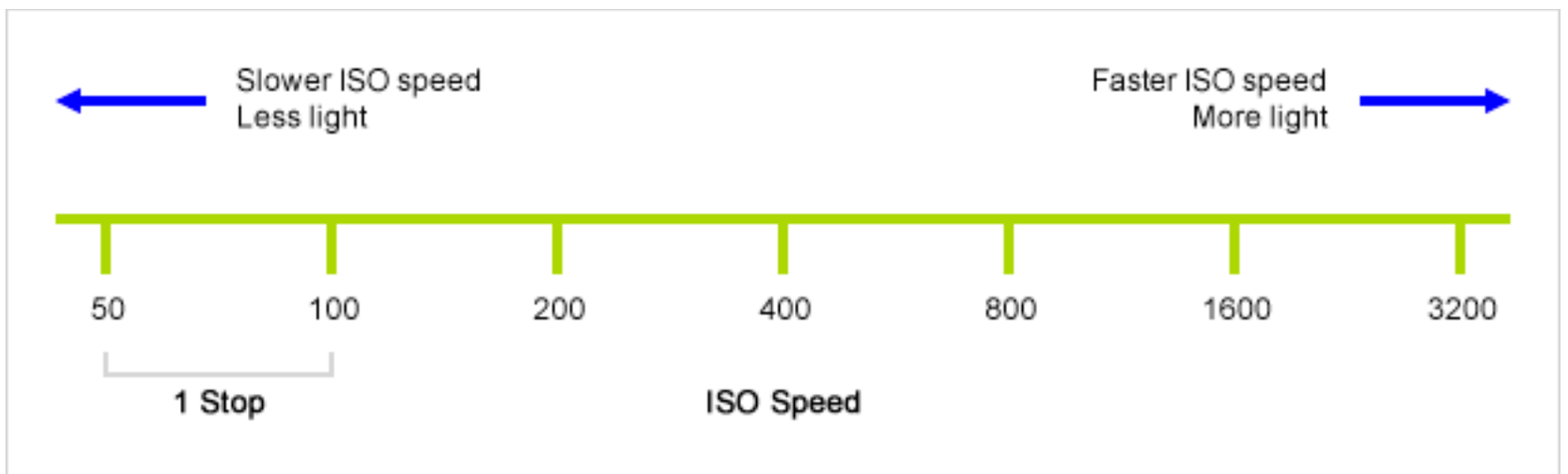




http://www.photographymad.com/pages/view/what-is-a-stop-of-exposure-in-photography

ISO Speed

- Camera Sensor's Sensitivity to Light
  - Faster = more sensitive = more light captured
- 1 stop = 2 x ISO speed



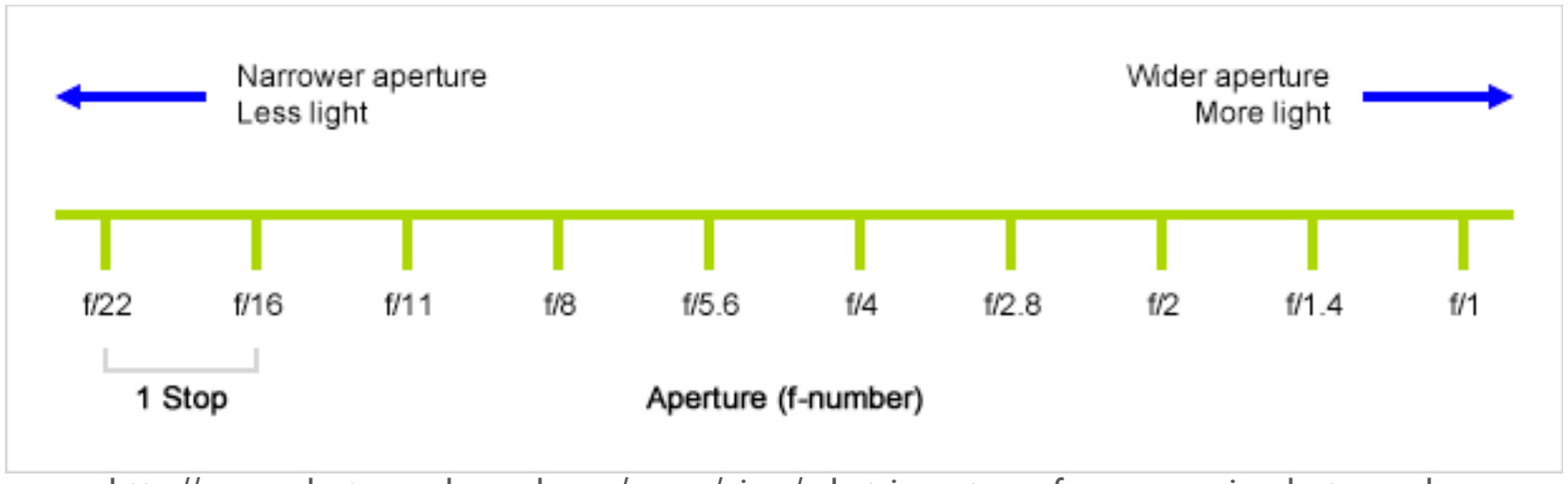
http://www.photographymad.com/pages/view/what-is-a-stop-of-exposure-in-photography

Aperture Diameter

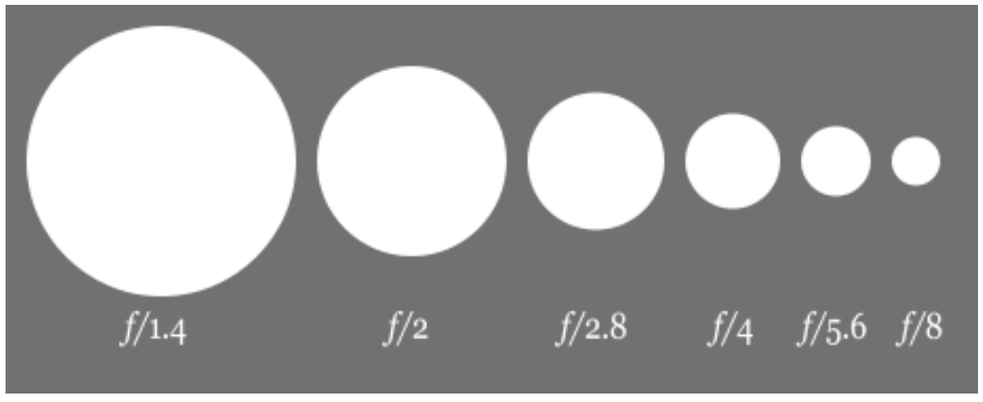
- Camera's aperture diameter
  - Wider = more light
- f-number
  - f/N where N = focal length / aperture diameter
    - Lower N = Wider aperture diameter = More light
- 1 stop = twice area = √2 diameter

f/2.8

f/16



http://www.photographymad.com/pages/view/what-is-a-stop-of-exposure-in-photography

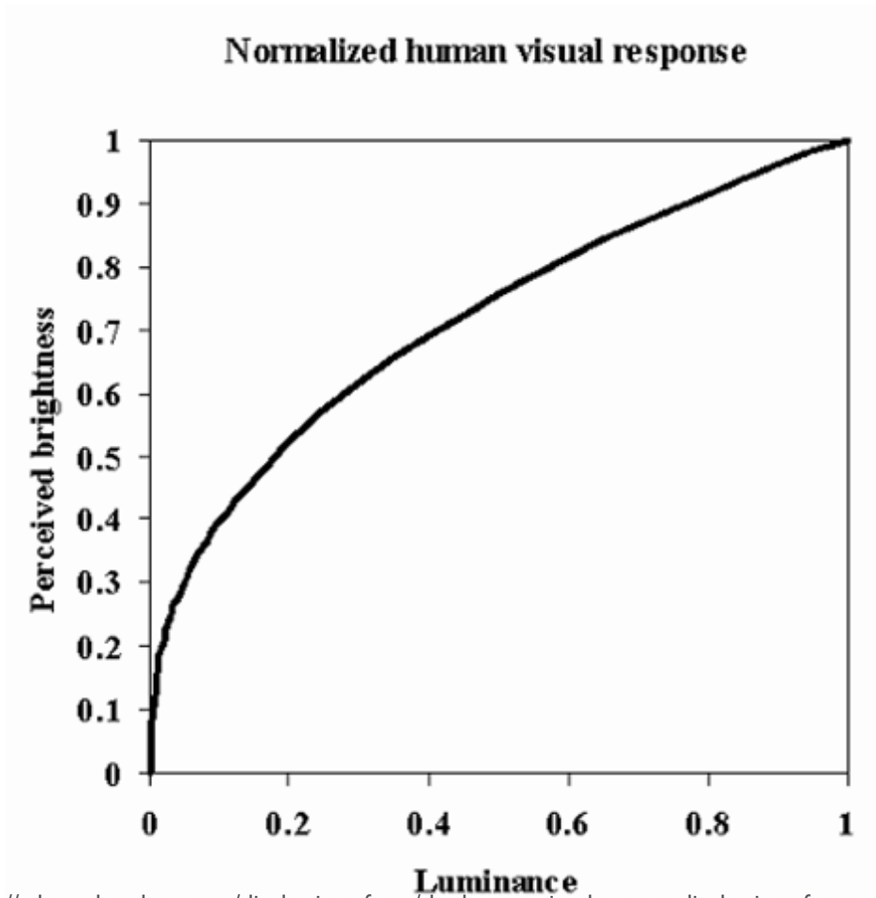


1 Stop

- Stop is used to compare these 3 components together
- Each component's stop in different result
  - Shutter Speed - blur
  - ISO - noise
  - Aperture - depth of field

Dynamic Range

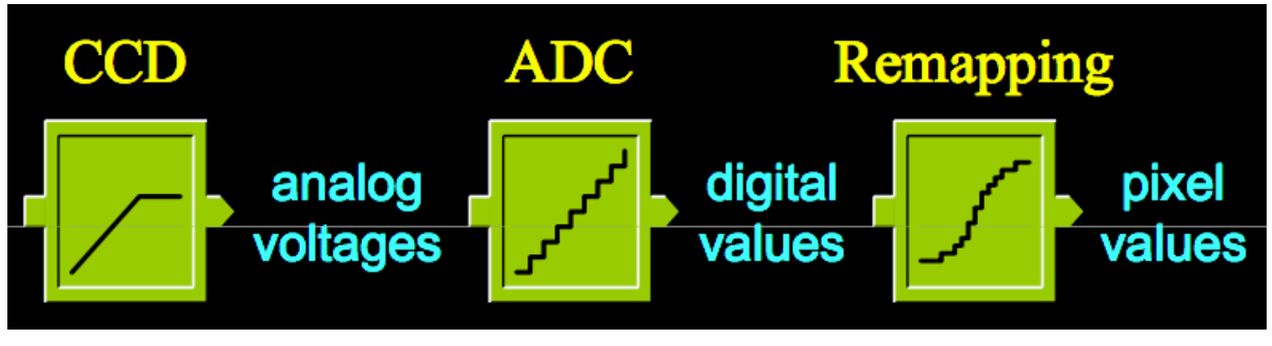
- Ratio between the brightest and darkest of the image
- Typical Natural Scene = 18 stops (200,000:1)
- Good digital camera can capture up to 10 stops (1,000:1): how?
  - By approximating eye/film response curve (non-linear)
  - Gamma Correction



http://what-when-how.com/display-interfaces/the-human-visual-system-display-interfaces-part-2/

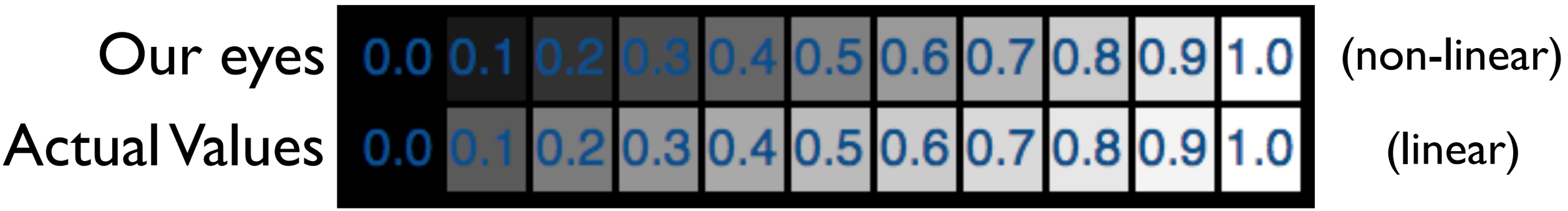
The Response Curve

- Digital Camera = linear response to light originally
- Human eyes and Film Camera = nonlinear response
- So, digital camera tries to approximate this non-linear behavior
  - Many LDR images applied gamma correction to fix this before save it to JPEG, TIFF, etc. to save space
    - RAW images still linear (directly from camera sensor) = big file size
- Basically, the response curve of typical non-RAW digital camera is no longer linear!



Gamma Correction

- Why we need Gamma Correction?
  - To store color intensity in 8 bits space efficiently because human eye perception is non-linear values
  - Arrange 8 bits wisely by using human eye perception knowledge
  - Stored intensity will no longer an actual value

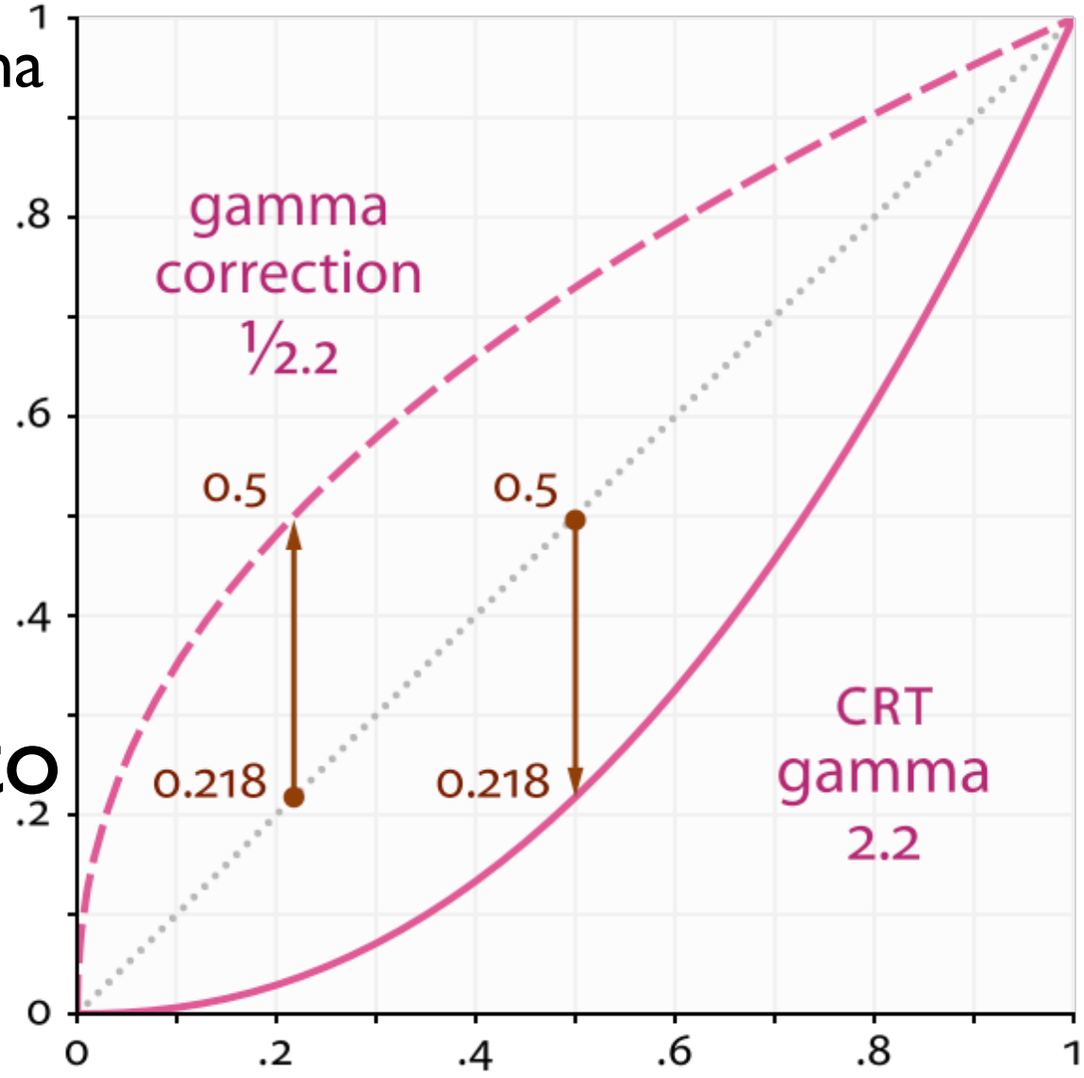


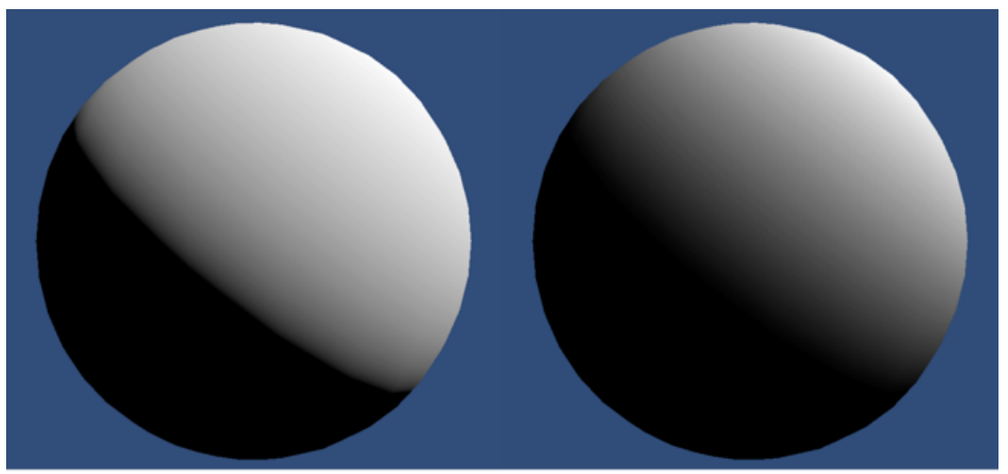
From this example, from our eye, what actual value represents 0.5 (our eyes)?

So, how about we make this actual value as a middle value to be stored?

Gamma Correction

- Stored-Value = Real-Value $^{1/gamma}$
- Typically gamma value = 2.2
- E.g. To store 0.218 grey, it uses $0.218^{1/2.2} = 0.5$
- So, we have half of the storage to store 0 - 0.218 (dark values)
- And, we have another half to store 0.218 - 1.0 (bright values)
- Display-Value = Store-Value ^{gamma}



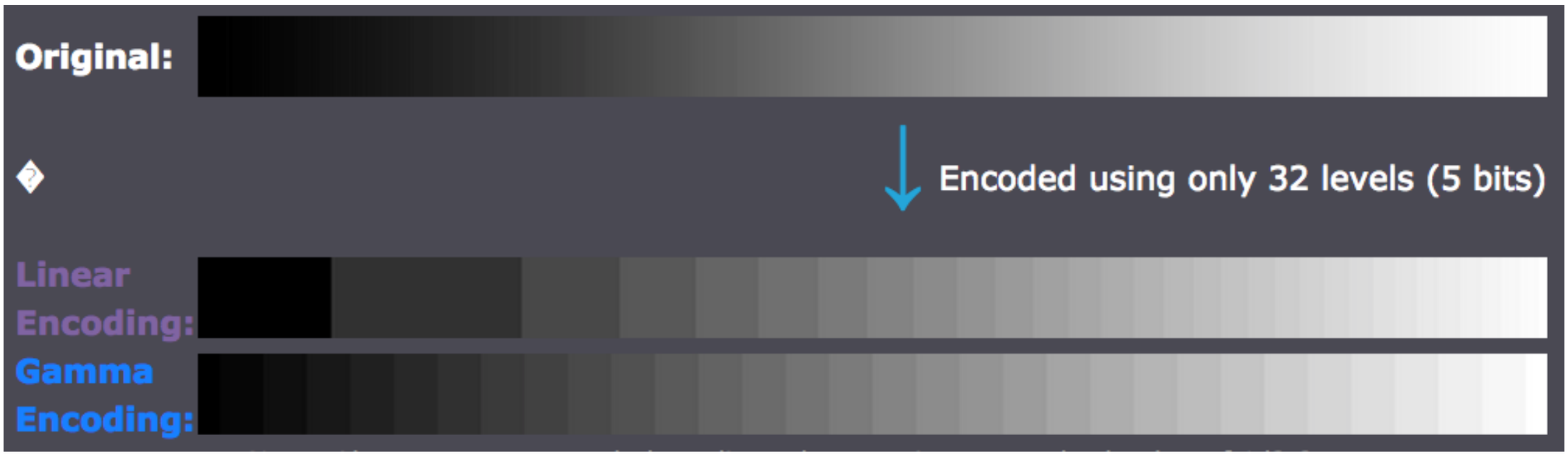


Linear Space

Gamma Space

Gamma Correction

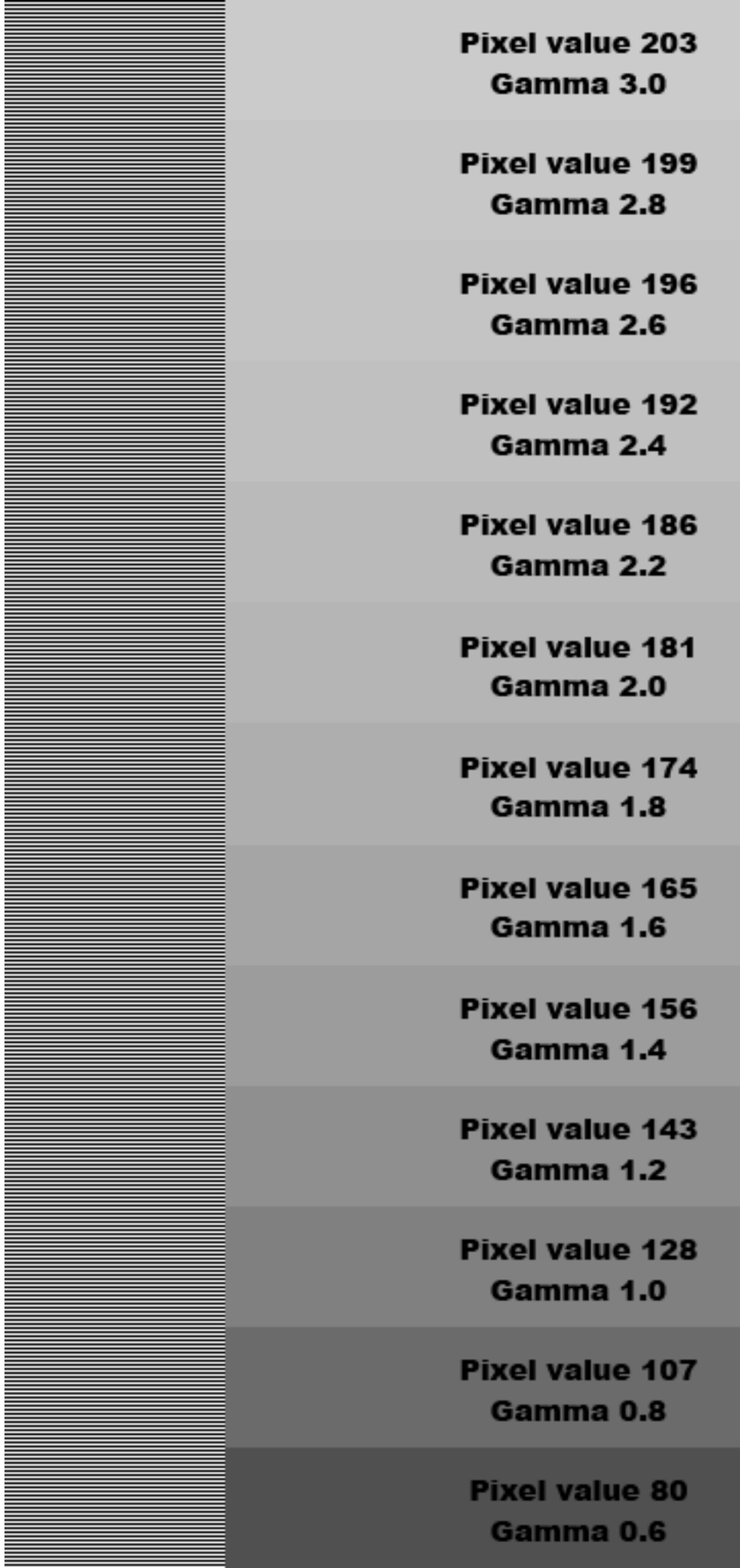
- Gamma Correction helps human differentiate darker values better but will reduce the difference of brighter values (our eyes don't need that - really?)
- sRGB
  - Use more than 1 gamma values (each for each type of color value)
  - More information: https://en.wikipedia.org/wiki/SRGB



http://www.cambridgeincolour.com/tutorials/gamma-correction.htm

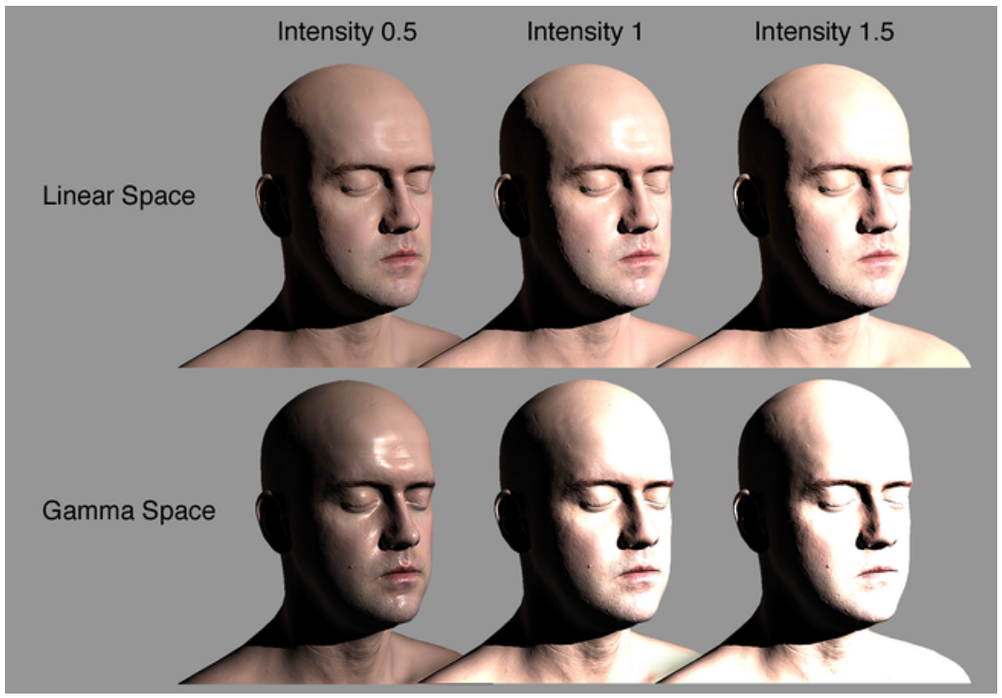
Monitor Test

- Monitor also makes use of our non-linear eye perception
- Here are the strip pattern of 0 (black) and 1 (white)
- Our eyes should blend the color to be 0.5 if the gamma correction of our monitor/projector is correctly set.
- On Mac, we can set gamma correction of the monitor by going to System Preferences -> Displays -> Color -> Calibrate



Problem with Gamma Correction

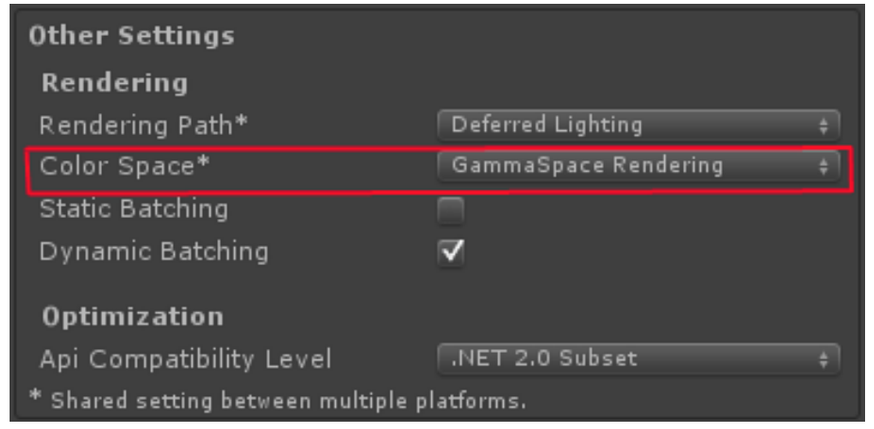
- How to present higher range colors in 1 picture? And more importantly how to interpolate the gamma corrected colors?
  - E.g. 0.5 gradually move to 1.5 (it will be super bright quickly)

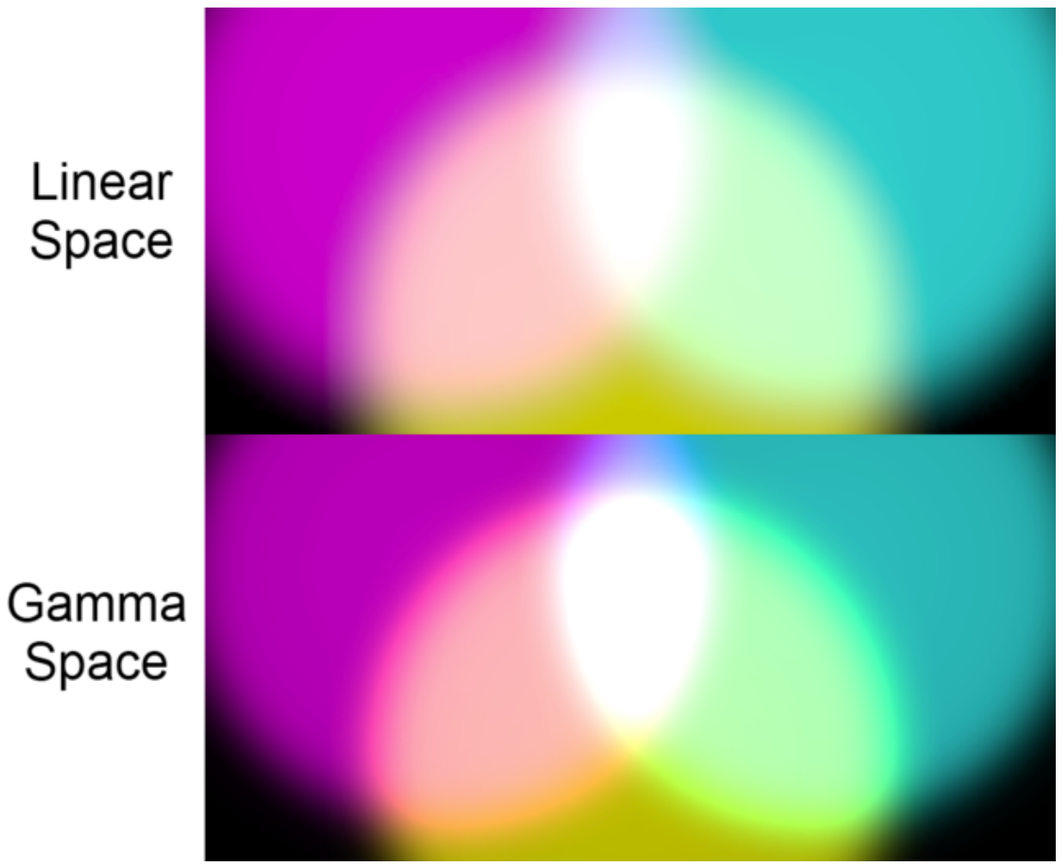


- Many new devices can store a lot more information than 8 bits/channel already
  - For example, HDR image contains 32-64 bits/channel
- Why don’t we use linear (real) value as is? = HDR

Unity Color Space

- Lighting calculation
  - Linear space
  - Gamma space
- Edit -> Project Settings -> Player -> Other Setting
- Need to check your target platform
  - Linear Color Space is NOT normally supported by some mobile hardwares
  - Wrong assumption leads to wrong color presentation





Dynamic Range in the Real World

http://www.pauldebevec.com/

(Sony VX2000 video camera)

Office interior

Indirect light from window

1/60th sec shutter

f/5.6 aperture

0 ND filters

0 dB gain

Dynamic Range in the Real World

<u>Outside in the shade</u>

1/1000th sec shutter

f/5.6 aperture

0 ND filters

0 dB gain

(Real scene = 16 times the light as inside)

Dynamic Range in the Real World

<u>Outside in the sun</u>

1/1000th sec shutter

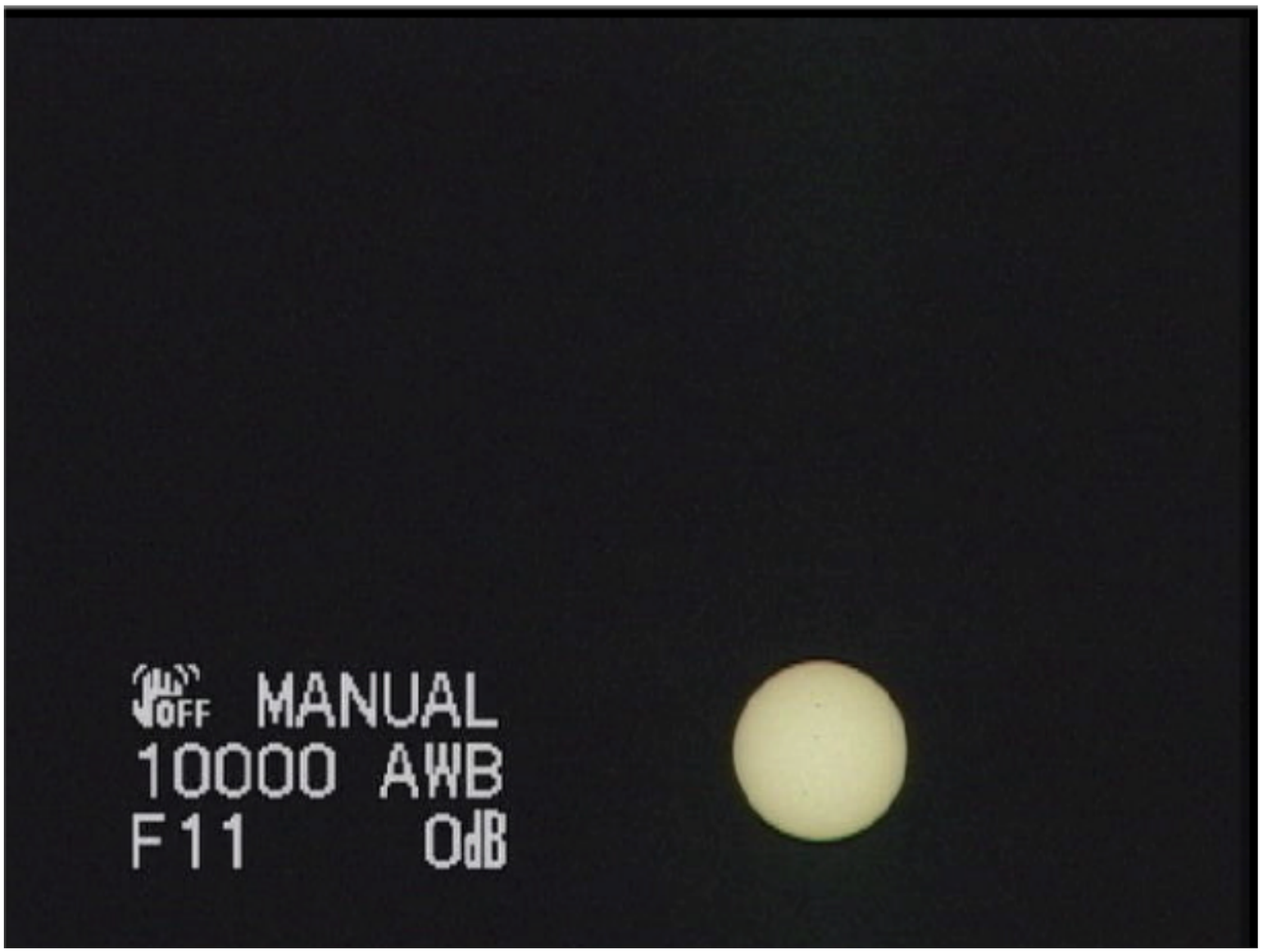
f/11 aperture

0 ND filters

0 dB gain

(Real scene = 64 times the light as inside)

Dynamic Range in the Real World



<u>Straight at the sun</u>

1/10,000th sec shutter

f/11 aperture

13 stops ND filters

0 dB gain

(Real scene = 5,000,000 times the light as inside)

Dynamic Range in the Real World

<u>Very dim room</u>

1/4th sec shutter

f/1.6 aperture

0 ND filters

18 dB gain

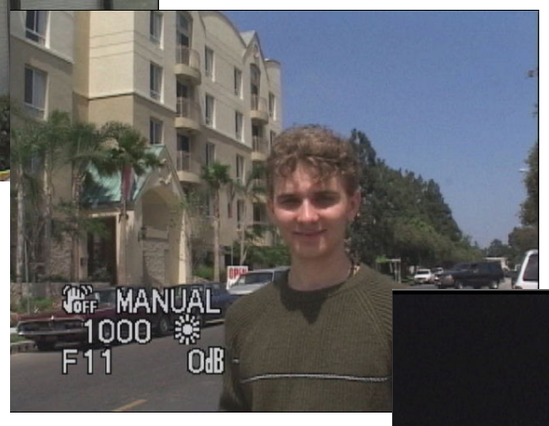
(Real scene = 1/1500th the light than inside)

Dynamic Range in the Real World

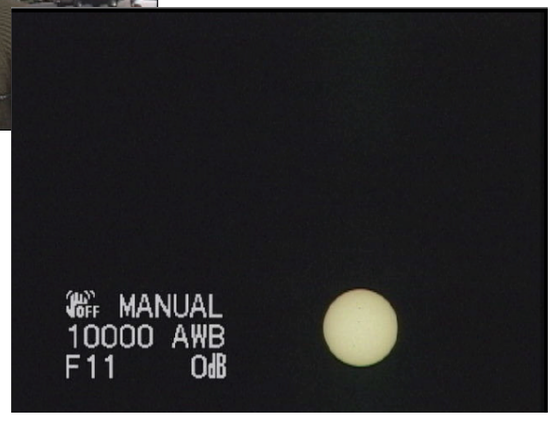
1

1500

25,000



400,000

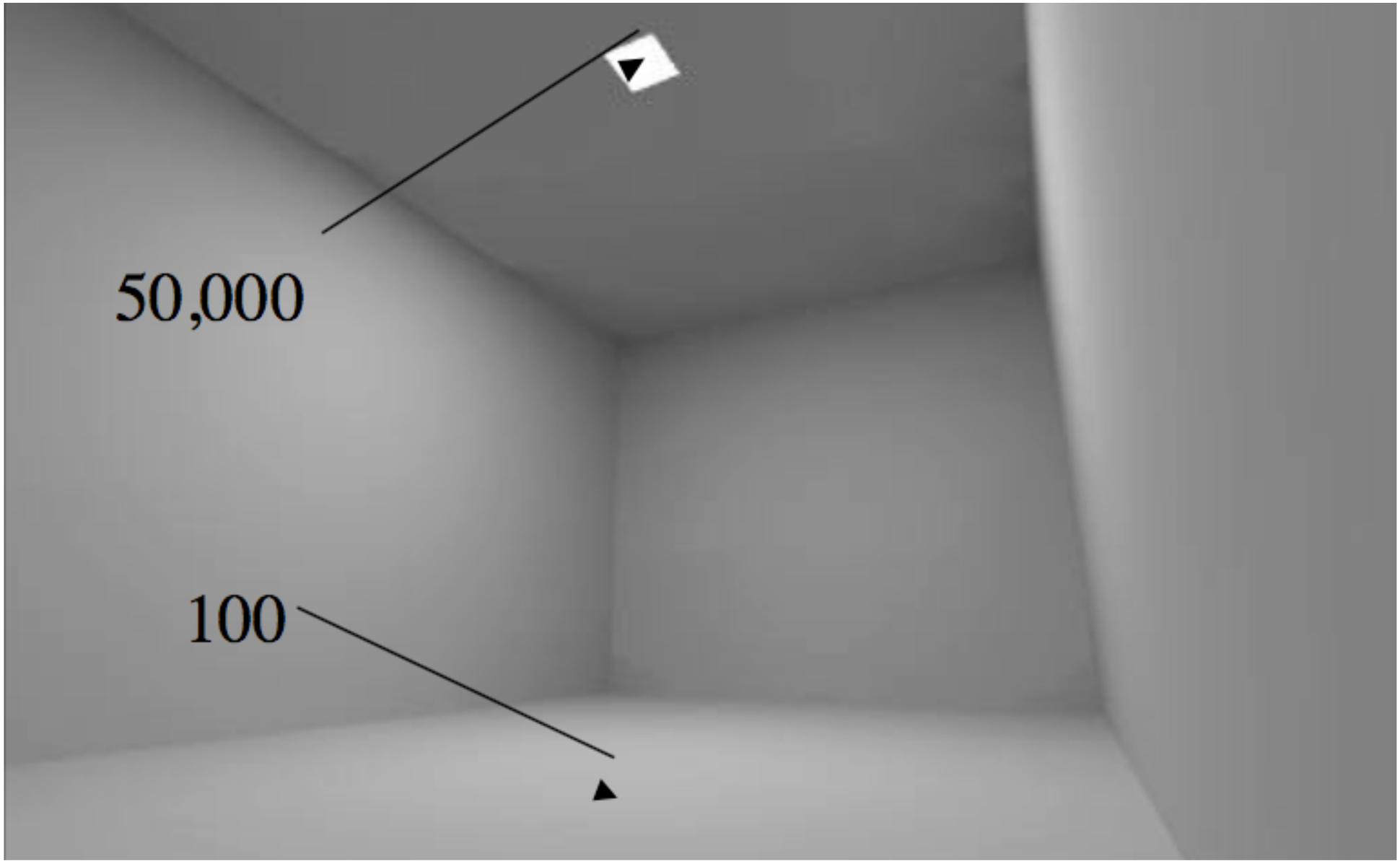


2,000,000,000

The real world is

<u>High Dynamic Range</u>

Light Sources generate High Dynamic Range



10’ x 15’ x 9’ room, 9” by 9” light, 50% reflective walls

Light Sources generate High Dynamic Range



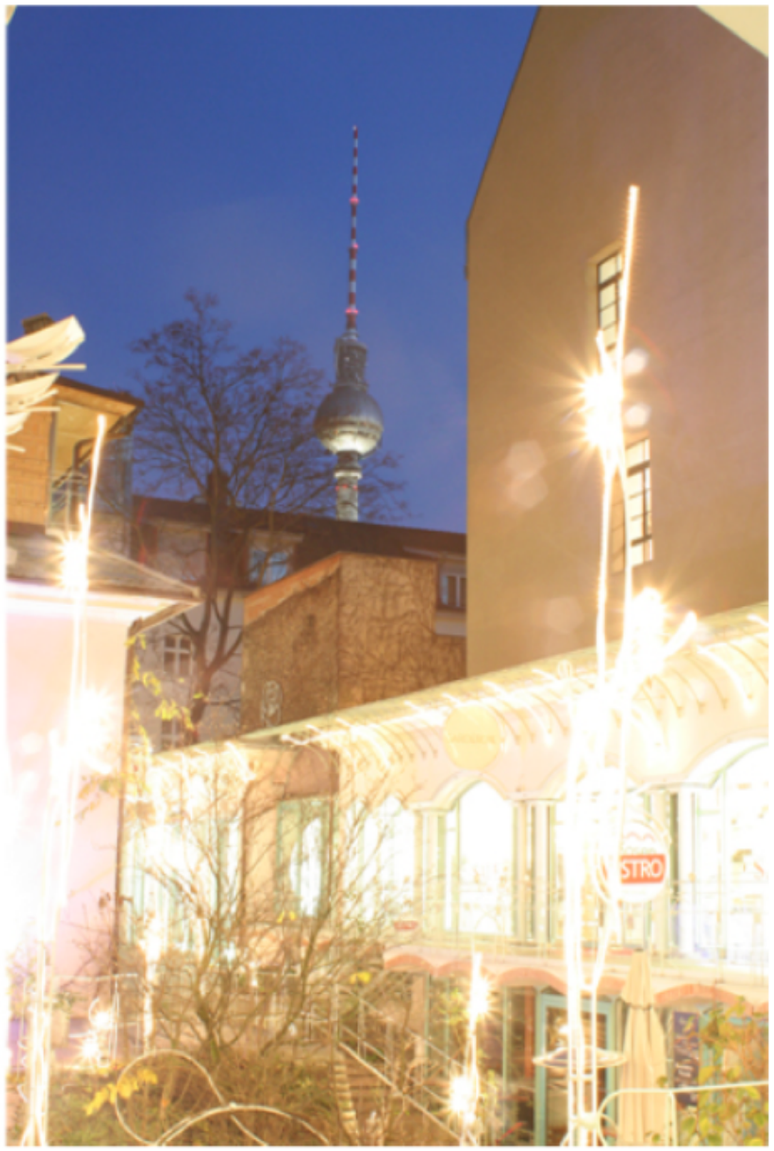


Figure: From left to right: underexposed, correct exposure, overexposed

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

High Dynamic Range Photography

Debevec and Malik. SIGGRAPH 97
Recovering High Dynamic Range Radiance Maps from Photographs.

Idea

- Take multiple exposures of the same scene
- Vary exposure time from small to long to capture complete dynamic range
- Compute HDR image from those exposures

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

High Dynamic Range Photography

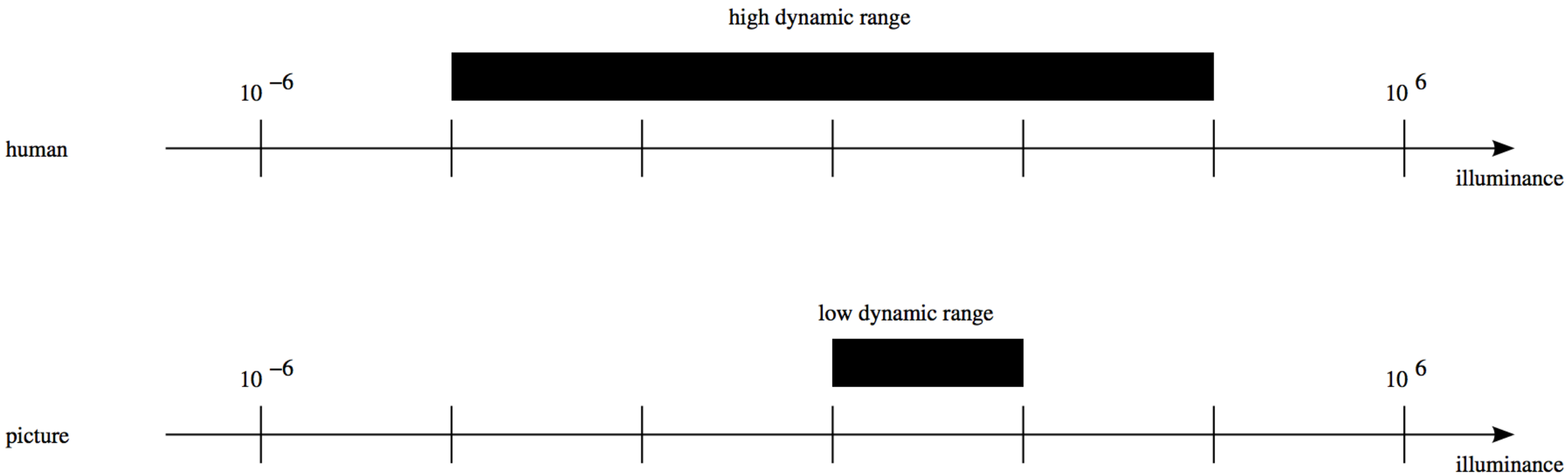
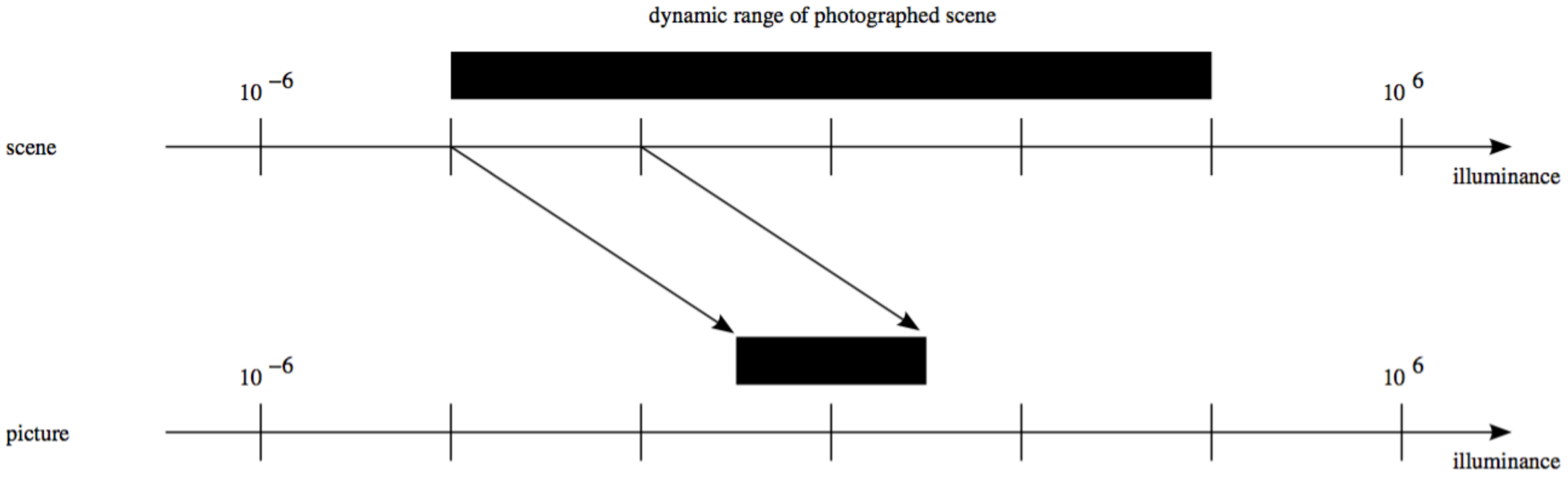


Figure: High dynamic vs. low dynamic range

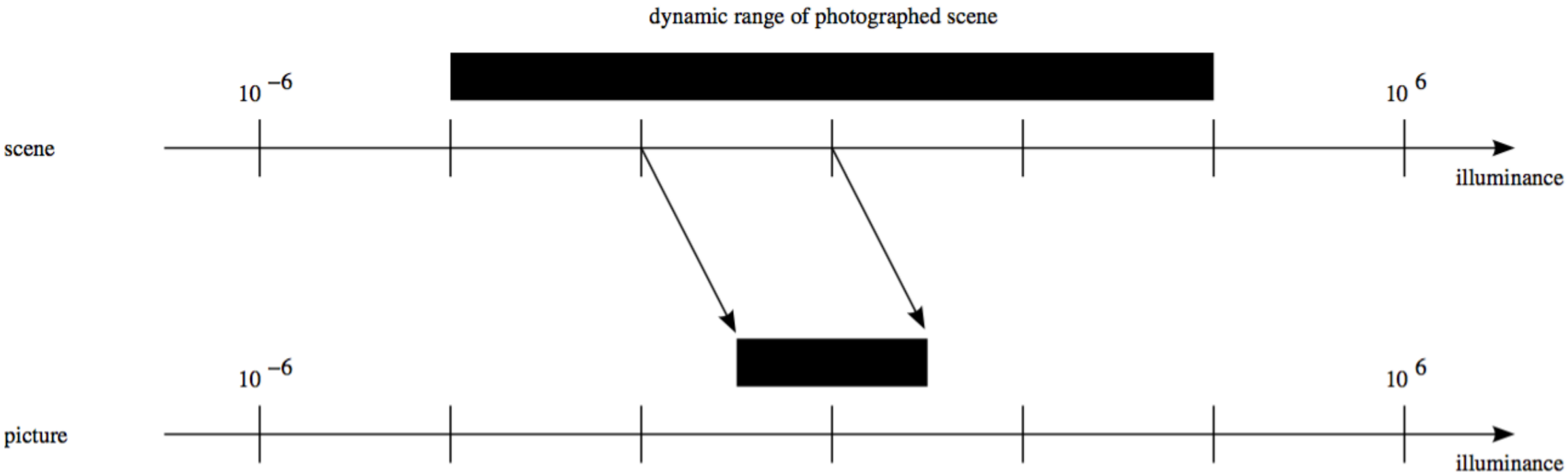
http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

High Dynamic Range Photography



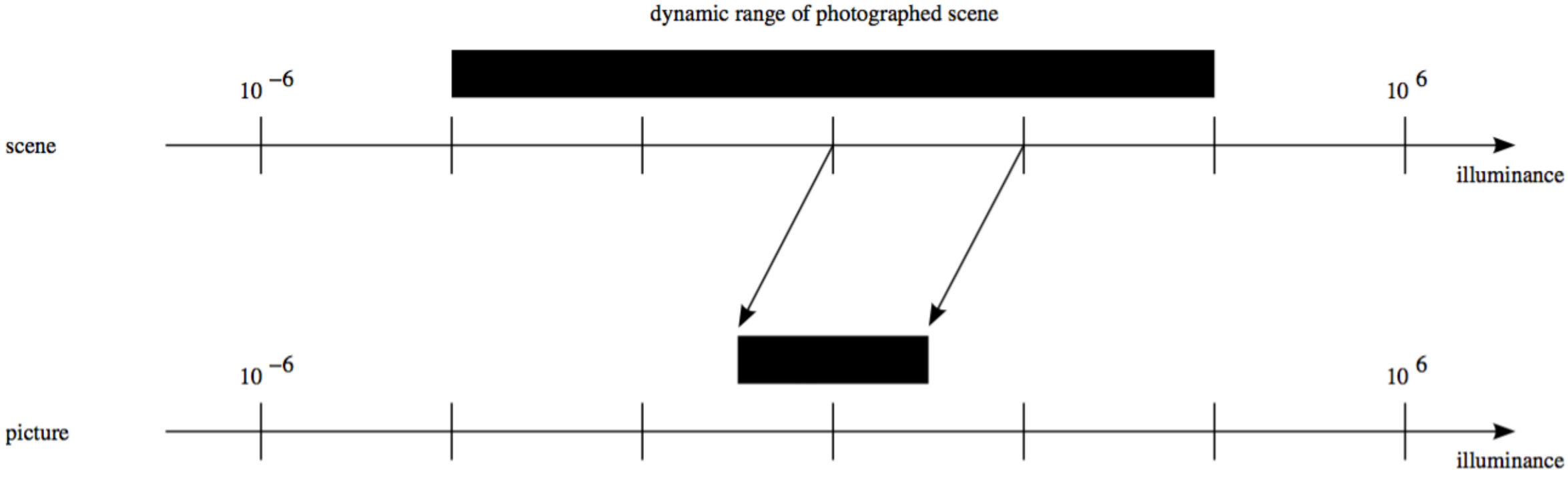
http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

High Dynamic Range Photography



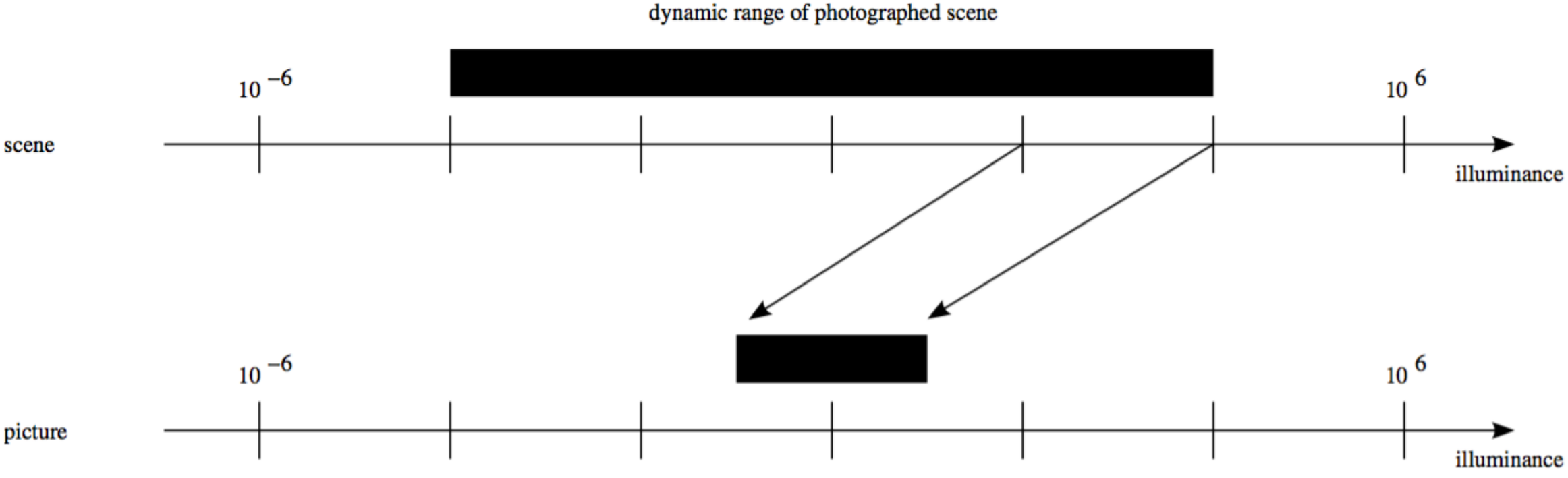
http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

High Dynamic Range Photography



http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

High Dynamic Range Photography



http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Recovering the Response Curve

- Camera response curve f tells us, how scene radiance E is mapped to pixel brighness Z
- Using the inverse of f allows us to reproduce actual scene radiance E

- f is different for each camera compute f from a series of exposures

Camera response curve

- Non-linear mapping f from scene radiance E_i and exposure time Δt_j to pixel brightness Z_i
- $Z_{ij} = f(E_i \Delta t_j)$
- We want to determine f

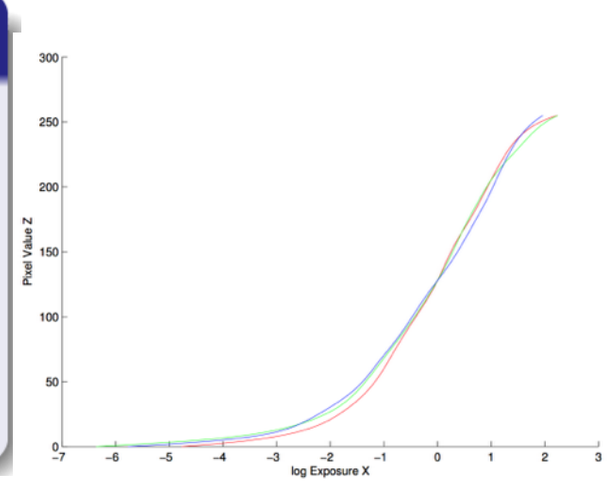


Figure: Recovered response curve f

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Recovering the Response Curve

Define equation system

- $Z_{ij} = f\left(E_i \Delta t_j\right)$
- $f^{-1}\left(Z_{ij}\right) = E_i \Delta t_j$
- $\ln f^{-1}\left(Z_{ij}\right) = \ln E_i + \ln \Delta t_j$
- Set $g = \ln f^{-1}$

Solve for g

- Solve overdetermined system of equations
  $g\left(Z_{ij}\right) = \ln E_i + \ln \Delta t_j$
- Solved in a least-squared error sense
- Result: Camera response function f

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Recovering the Response Curve

- Computed from a series of 9 exposures Exposure time from 1/4000 to 15
- Each exposure two stops (shutter speed) apart (i.e. 1/4000 , 1/1000 , 1/250 ,...,15)

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Recovering the Response Curve

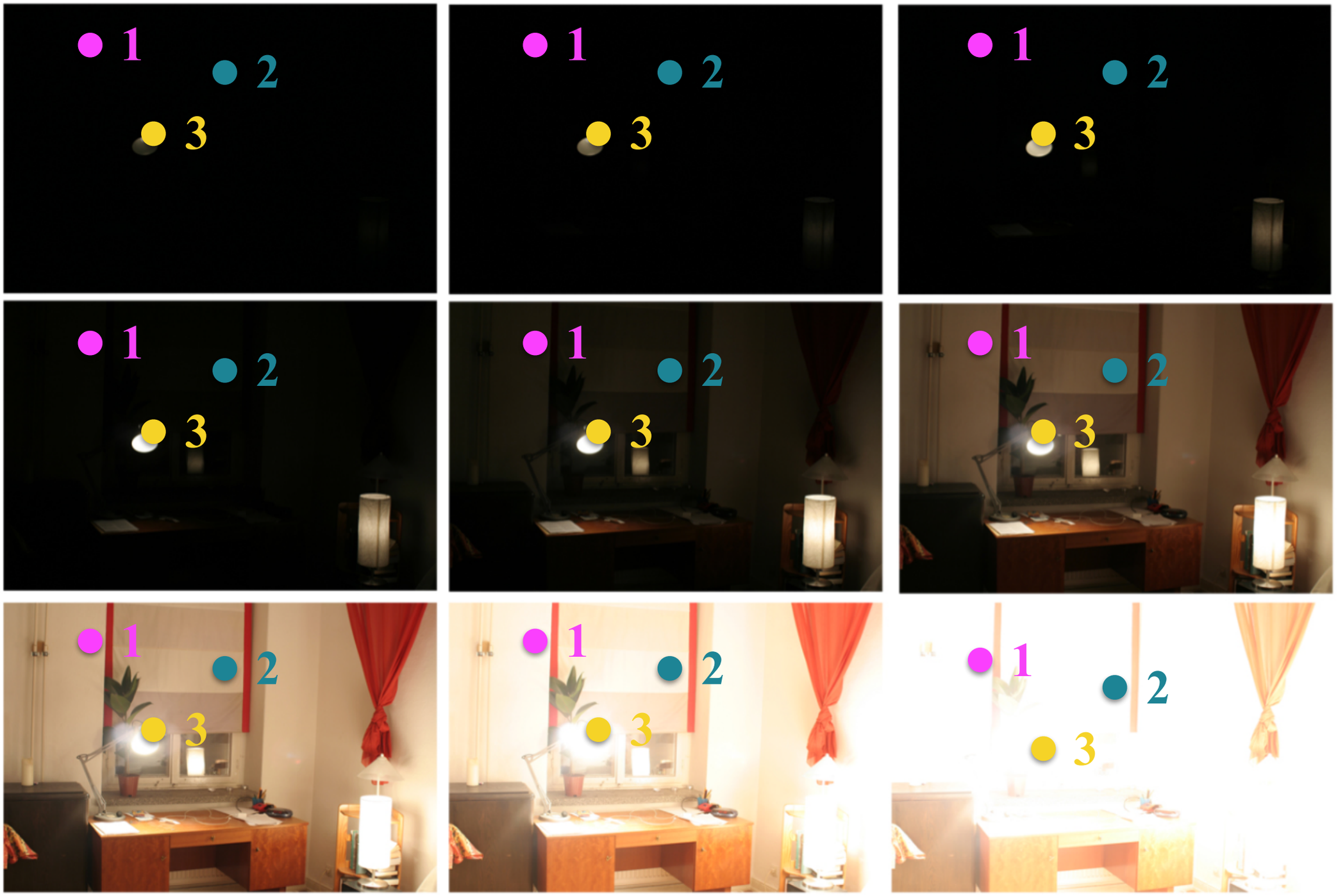
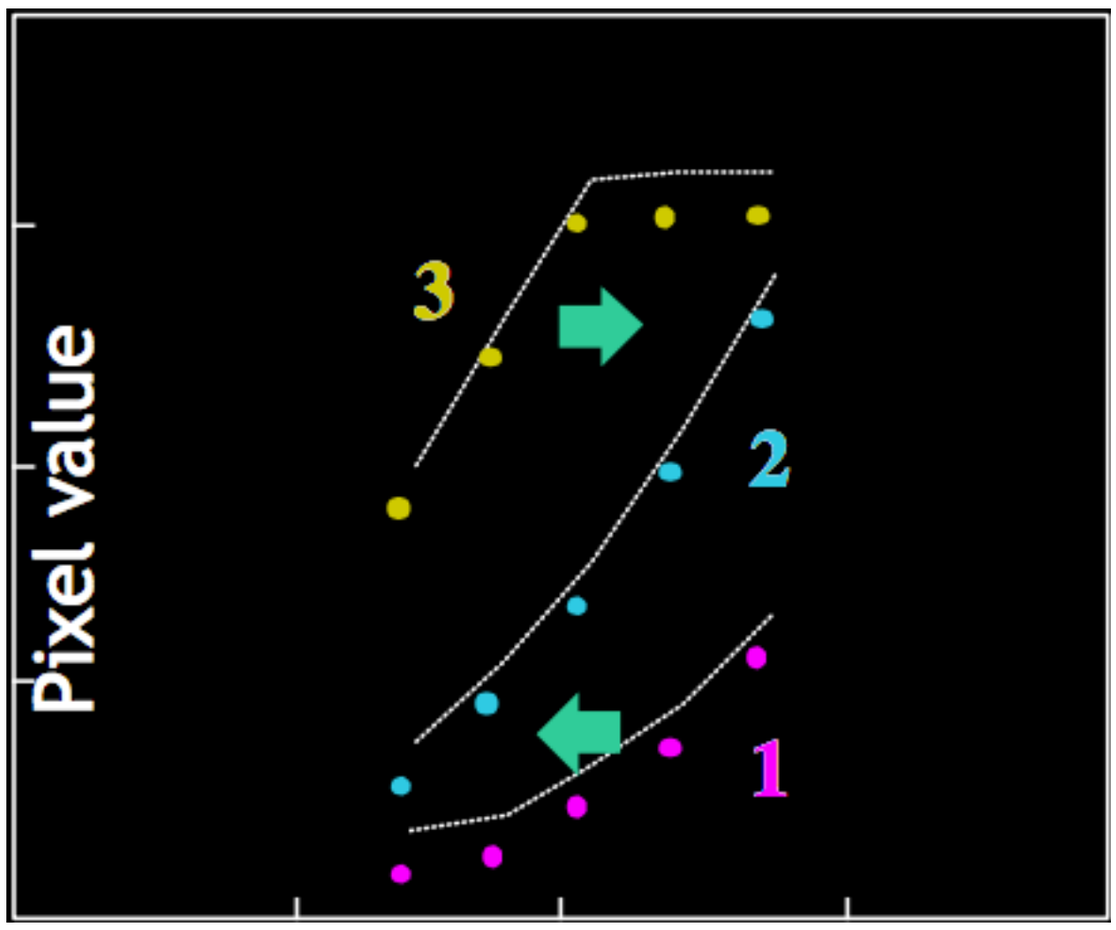


Figure: Exposure series used to compute camera response curve

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Recovering the Response Curve

Assuming unit radiance
for each pixel

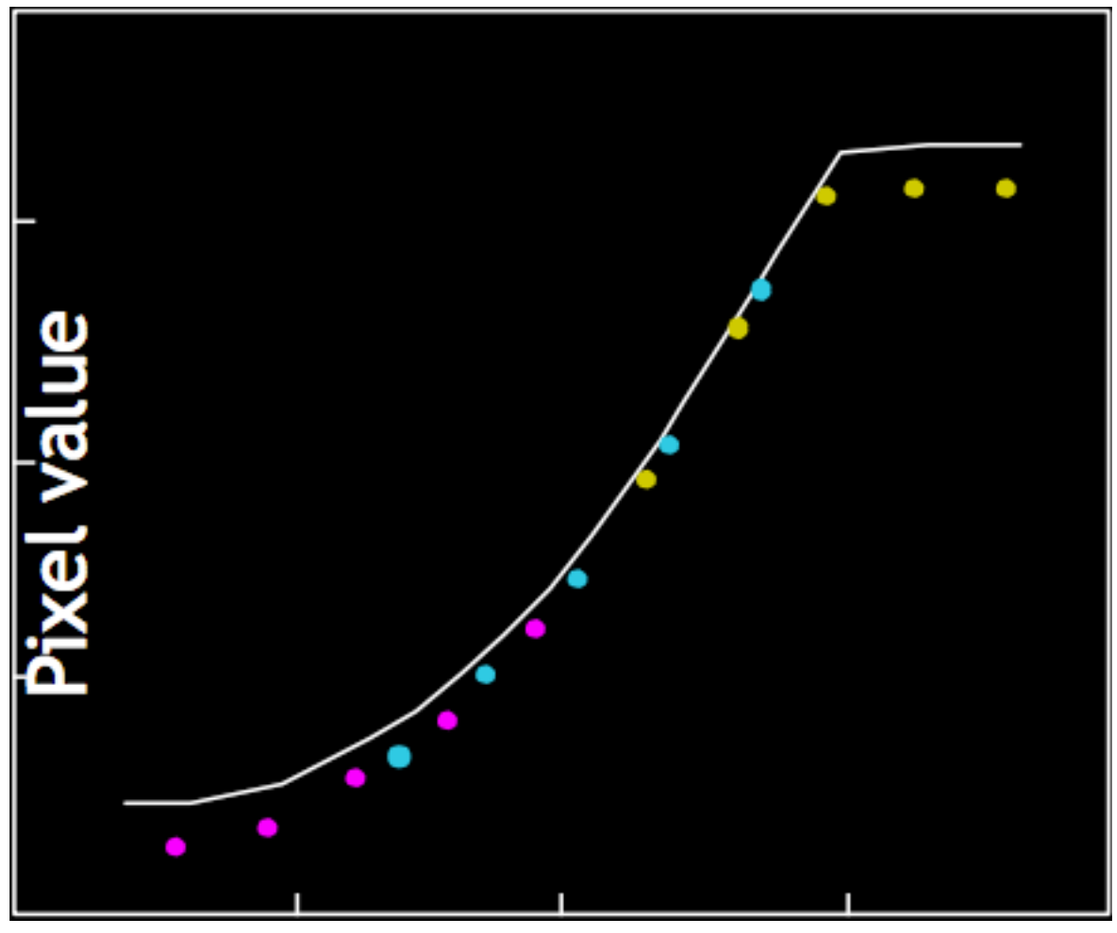


log Exposure

log 1 + log Δt

L.S.

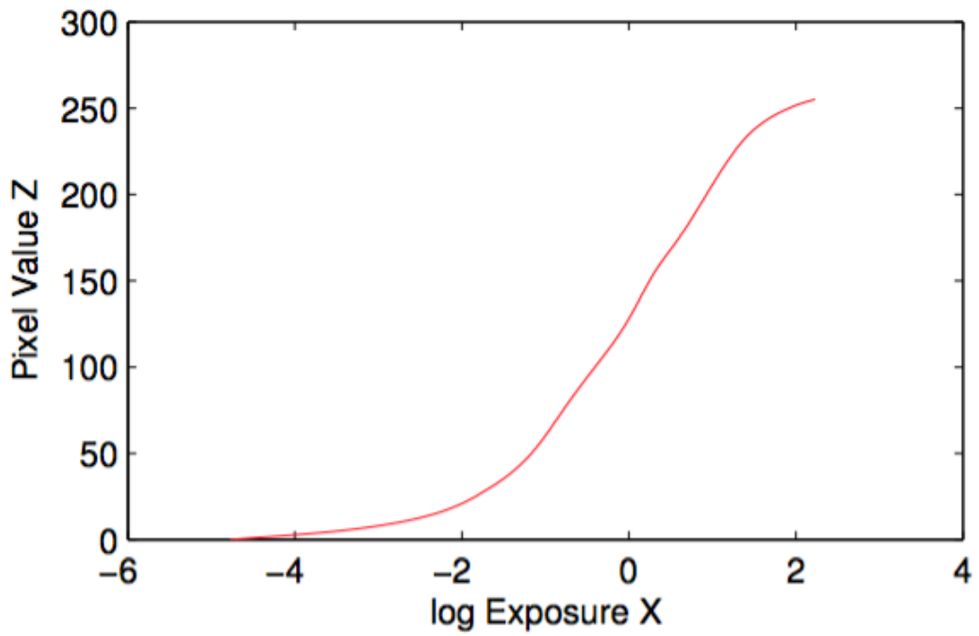
After adjusting radiances
to obtain a smooth curve

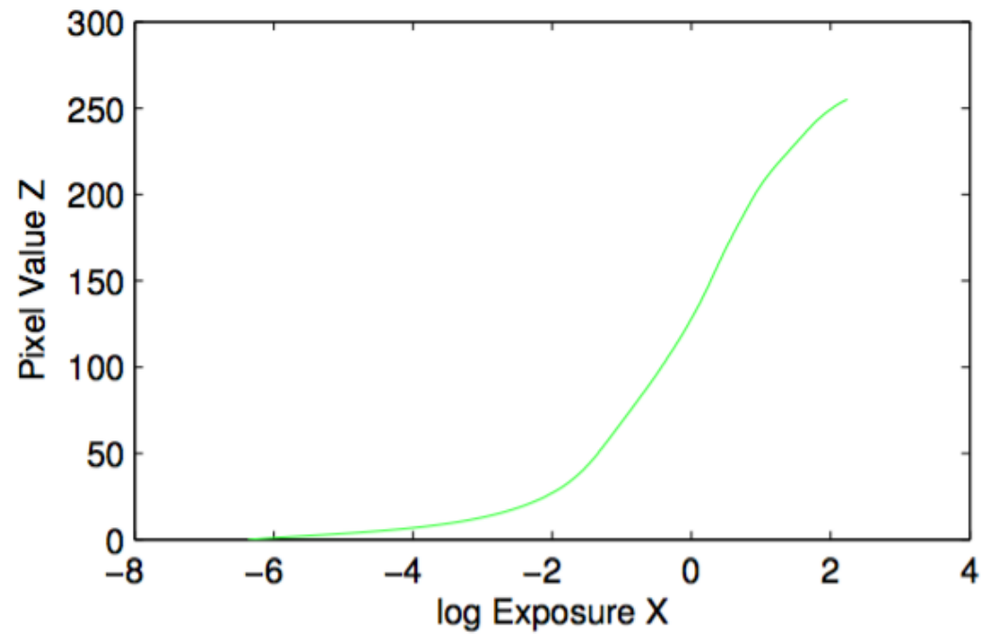


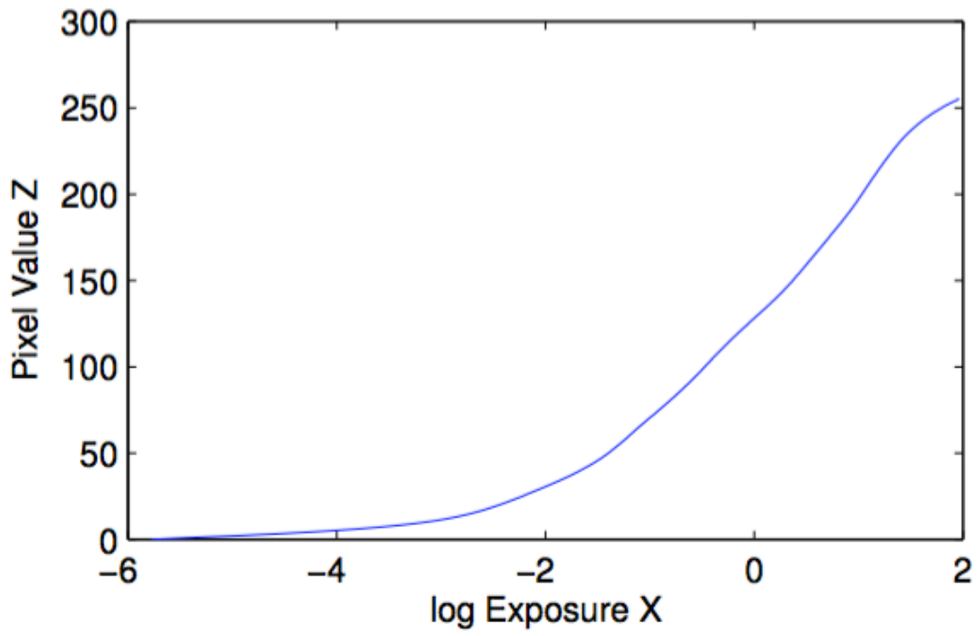
log Exposure

log adjusted-radiance + log Δt

Recovering the Response Curve







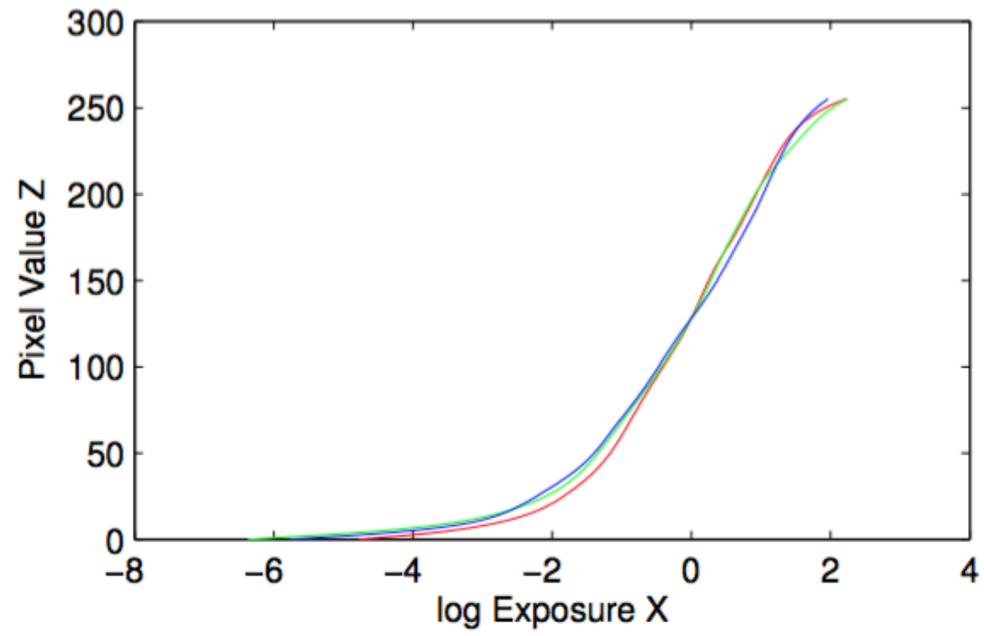
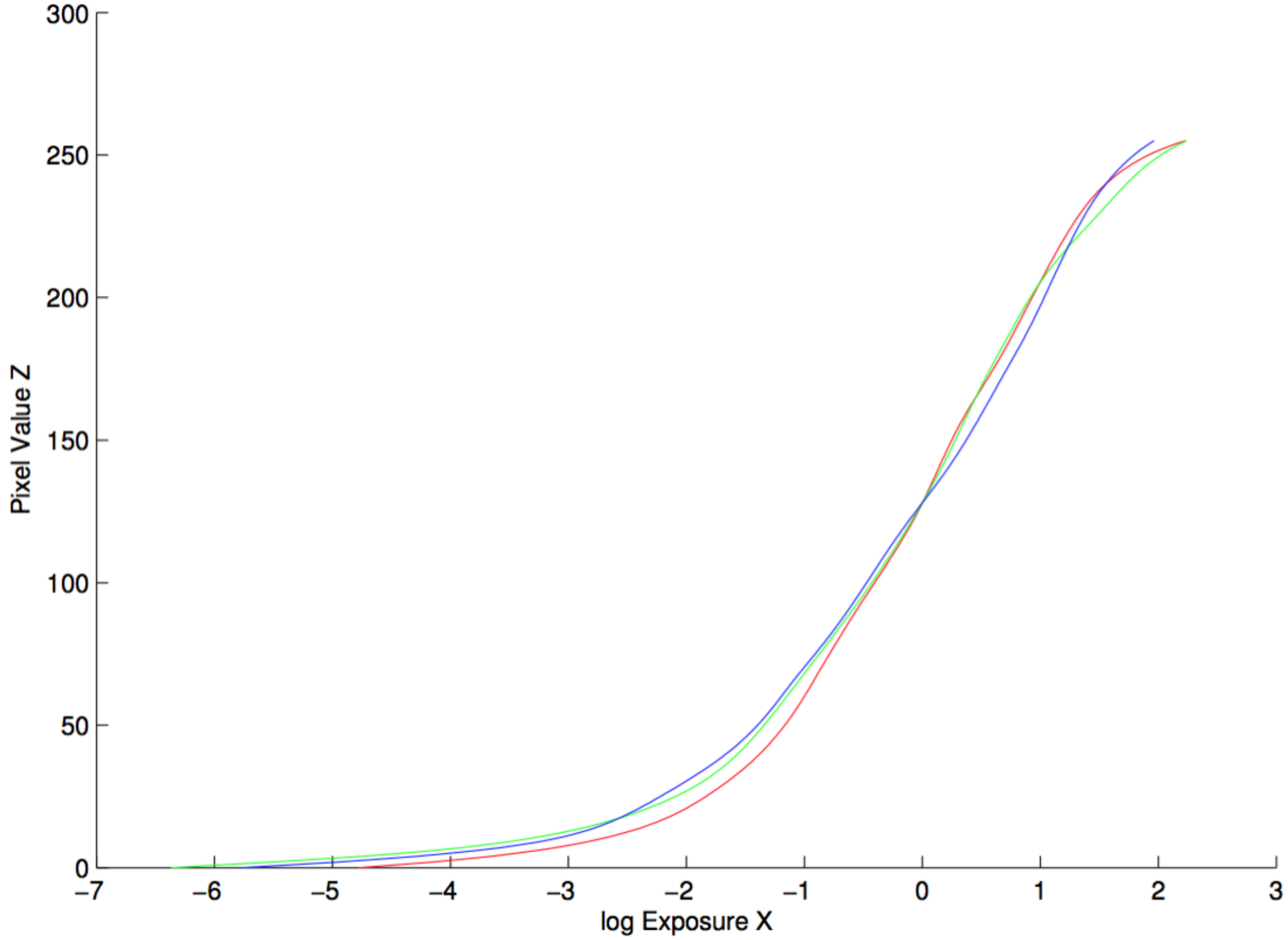


Figure: Recovered response curve f

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Recovering the Response Curve



http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Constructing HDR Radiance Map

What we know

- Camera response curve
- Multiple exposures of same scene at know exposre times
- Each pixel in the scene is correctly exposed at least once

- Use this information to assemble HDR image
- Each pixel is a weighted combination from the corresponding pixels of all exposures

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Constructing HDR Radiance Map

Radiance

- $\ln E_i = g(Z_{ij}) - \Delta t_j$

Weighted average

- Use weighted average of all exposures to create HDR map
- $\ln E_i = \frac{sum_{j=1}^{P}\left(w\left(Z_{ij}\right)\left(g\left(Z_{ij}\right)-\Delta t_j\right)\right)}{sum_{j=1}^{P}\left(w\left(Z_{ij}\right)\right)}$

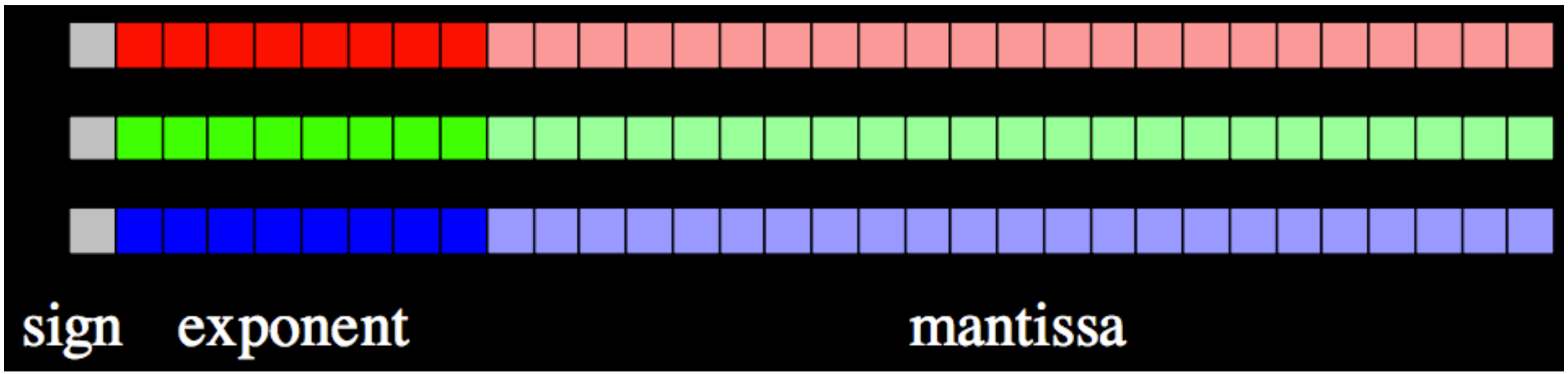
http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

HDRI Formats

- Portable FloatMap (.pfm)
- Floating-Point TIFF (.tif)
- Greg Ward's RADIANCE format (.pic, .hdr)
- Greg Larson's LogLuv TIFF (.tif)
- Newtek Flexible Image Format (.flx)

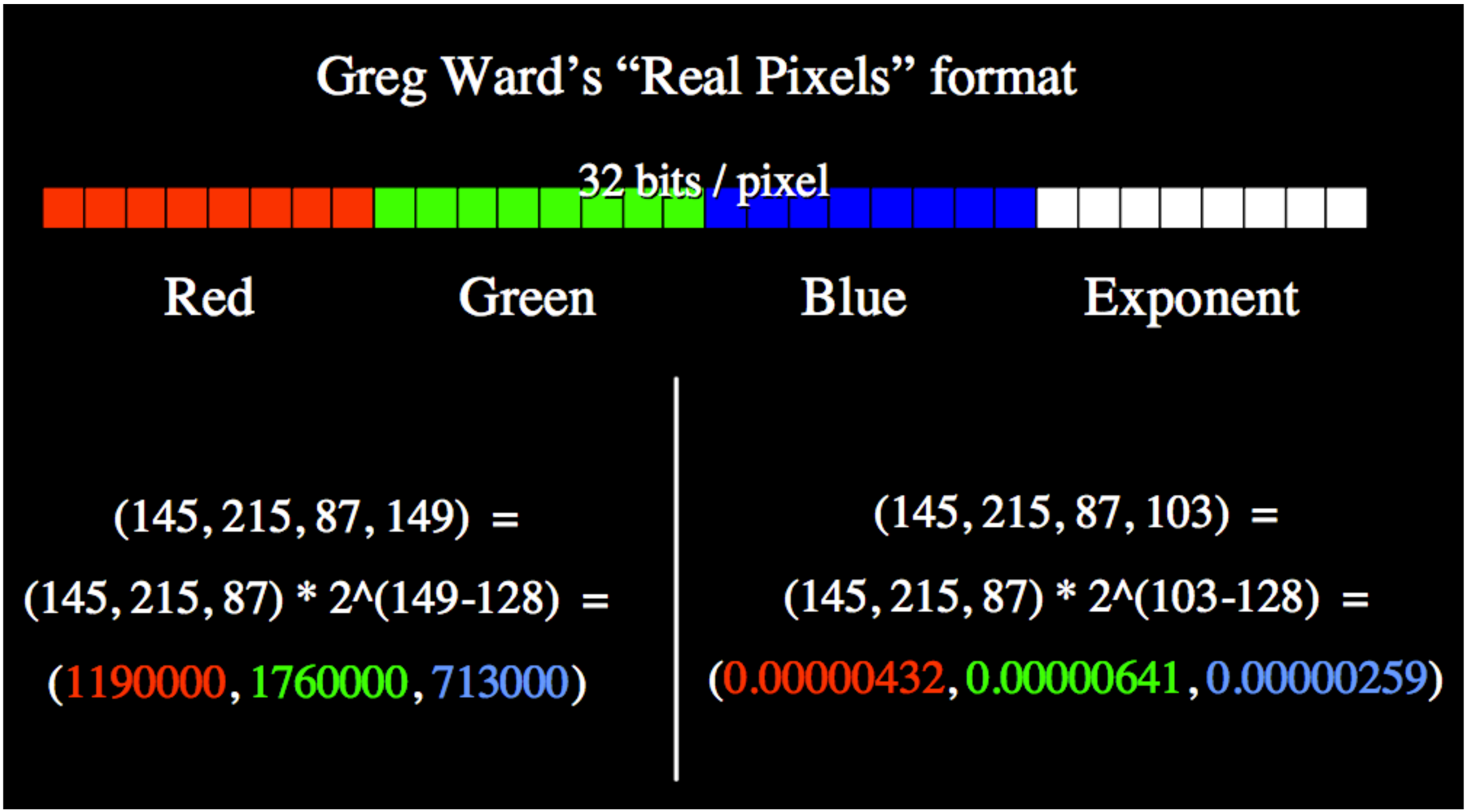
Portable FloatMap

- 12 bytes per pixel, 4 bytes (32 bits) for each color
- 1 byte (8 bits) for each color's exponent



RADIANCE Format

4 bytes (32 bits) for each pixel, 1 byte for each color, 1 byte for exponent



HDR Radiance Map

- We have now generated a HDR image
- Dynamic range of resulting HDR image: $\approx 1 : 100000$
- Dynamic range of computer screen: much lower

Problem We cannot directly display the resulting HDR image

Solution Tone mapping

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Tone Mapping

- Compress high dynamic range to displayable low dynamic range
- Need to preserve details and colors

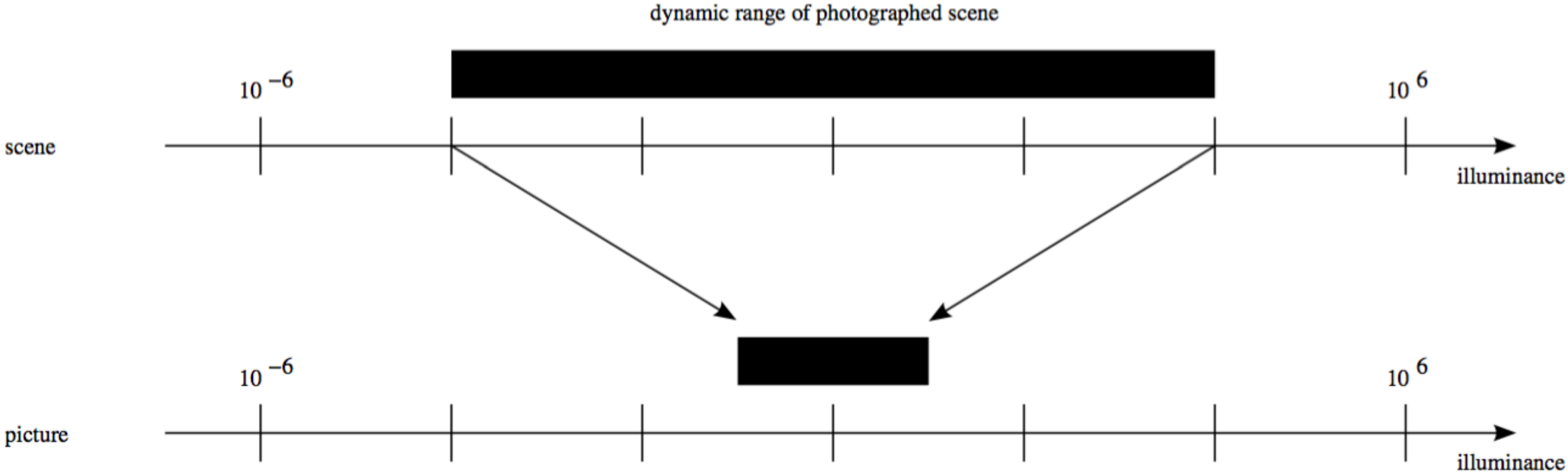


Figure: Tone mapping

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Luminance Map

- Tone mapping is usually done on the luminance map
- Then colors are reapplied

Luminance map

- $L(x, y) = 0.2 \cdot Red(x, y) + 0.7 \cdot Green(x, y) + 0.1 \cdot Blue(x, y)$

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Global Operators

Definition

Tone mapping operator Maps a HDR image into displayable range $[0, 1]$

Linear Linear mapping of HDR image into range $[0, 1]$

Logarithmic Logarithmic mapping of HDR image into $[0, 1]$

Reinhard Global Reinhard operator $L(x, y) = \frac{L(x,y)}{1+L(x,y)}$

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Global Operators

Figure: Linear Mapping

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Global Operators

Figure: Logarithmic Mapping

Global Operators

Figure: Global Reinhard Mapping

http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

Local Operators

Figure: Desk - Reinhard local

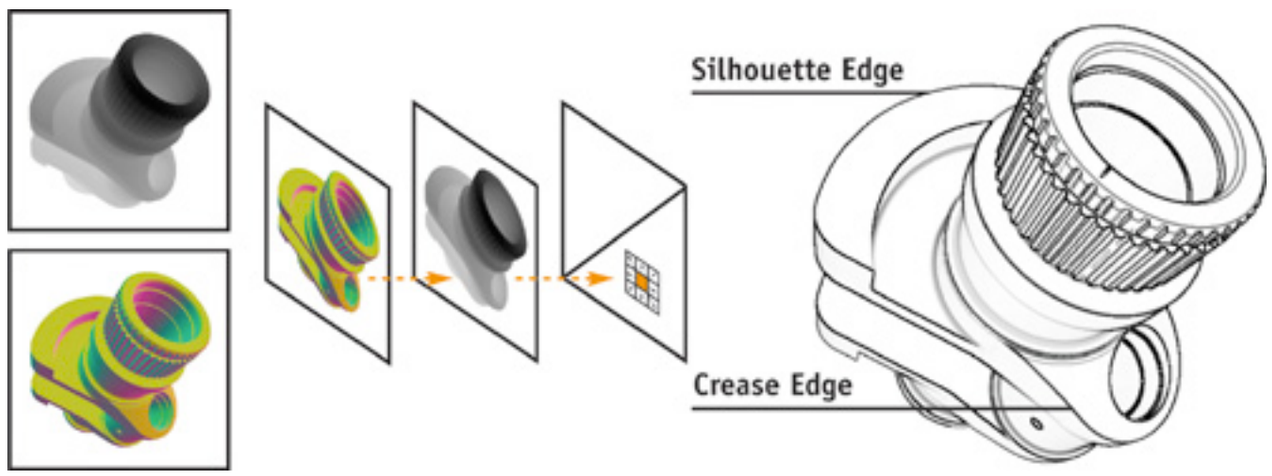
http://cybertron.cg.tu-berlin.de/eitz/hdr/presentation_hdr_tonemapping.pdf

HDR Skybox Lab

- Open Lab 1 project (or just create a new project with some metallic 3D objects)
- Download our light probe: https://dl.dropboxusercontent.com/u/39794258/GameEngine/uffizi_probe.hdr
- Import the light probe
  - Assets -> Import New Assets…
  - Select our uffizi_probe.hdr
- At the new imported HDRI
  - Texture Type -> Cubemap
  - Mapping -> Mirrored Ball (click Apply)
- Create new material
  - Project -> Create -> Material -> Name it “uffizi”
- Add the light probe to the cube map
  - Inspector -> Shader -> Skybox -> Cubemap (HDR) -> Select -> “uffizi_probe”
- Set the new skybox
  - Window -> Lighting -> Scene -> Environment Lighting
    - skybox -> select the new “uffizi”
    - Reflection Source -> Custom -> Cubemap -> select “uffizi_probe”
- Create a very metallic material and add to some objects to see its reflection

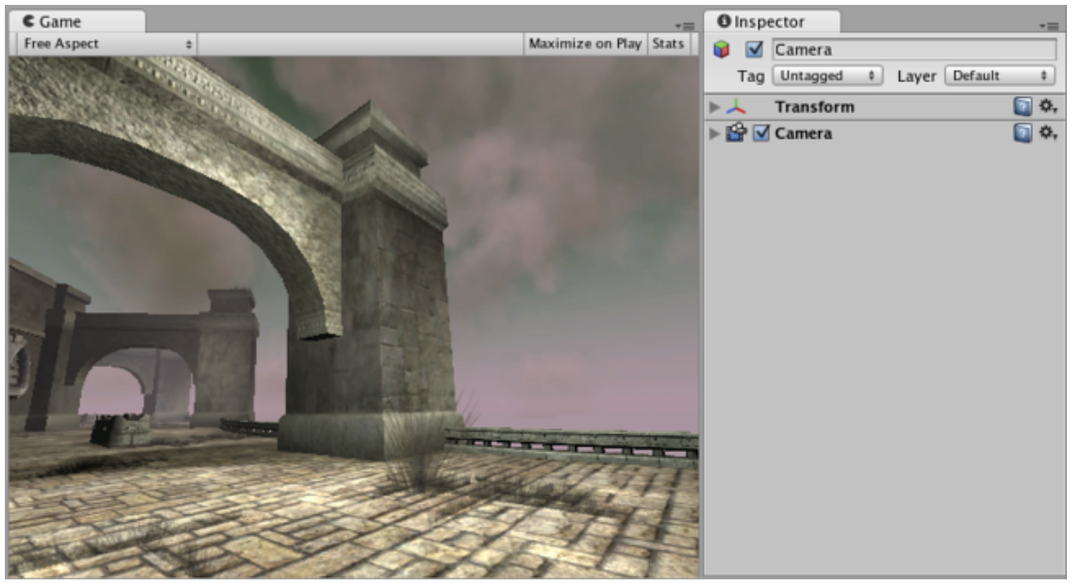
Image Space Rendering

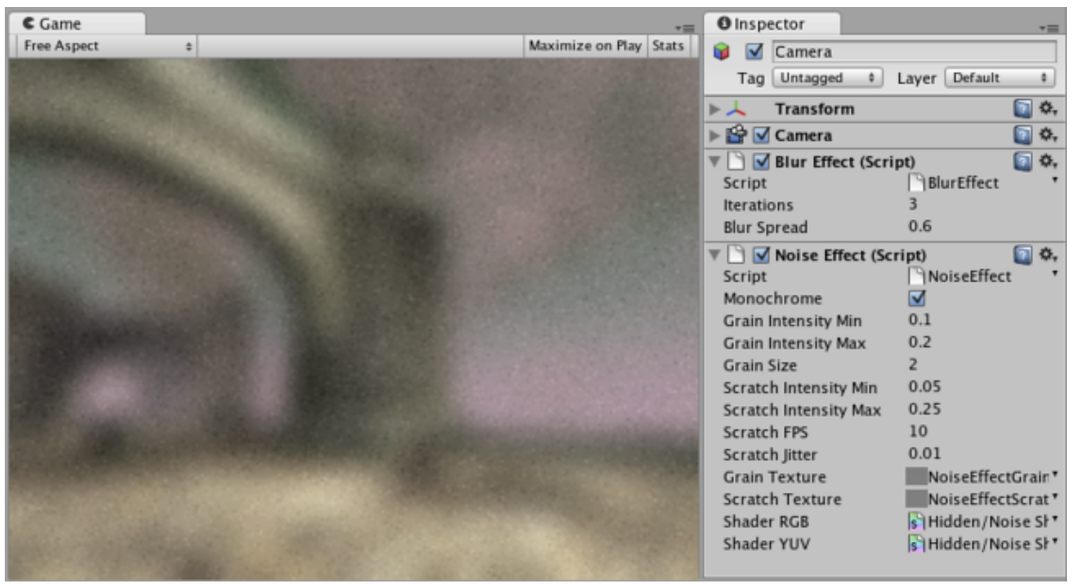
- Use a lot in game to achieve real-time rate
- A lot quicker than 3D rendering
- Post-processing effects
  - Skip the 3D rendering and go right to image/screen space
  - Use pixels and texture information (e.g. depth buffer, normal buffer, etc) - similar to deferred shading
- Similar to Image Processing Techniques as using in Photoshop, After Effects, etc. where only 2D information (pixels) are mainly manipulated



Unity Image Effects

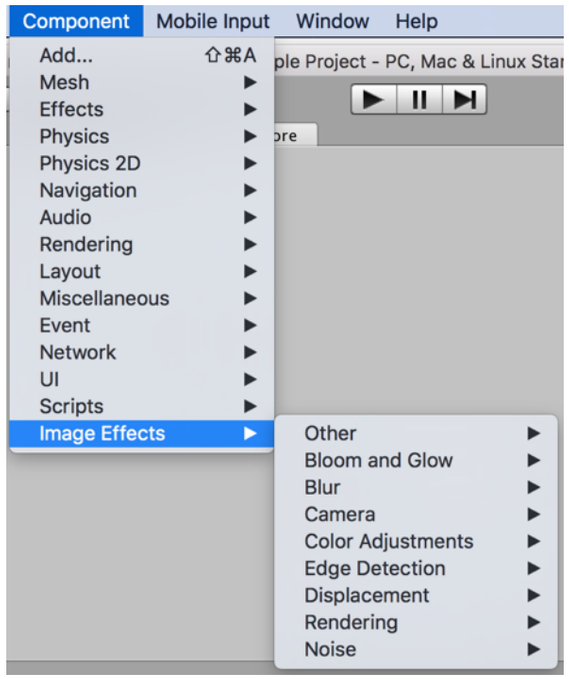
- In Standard Assets
  - Assets -> Import Package -> Effects
  - At Camera -> add Component -> Image Effects
- Image Space Rendering
  - Apply to each image frame after 3D rendering
- Scripts attaching to Camera
- Calling from OnRenderImage() function





Unity Image Effect Labs

- Assets -> Import Package -> Effects
- At any scene, select the main camera
  - Component -> Image Effects
  - Try Blur



- Check Game View
- Try Adjust Blur (Script)'s parameters
- Preview cannot show the effect because this is image-based rendering
- Try other effects

Ambient Occlusion

- Global Illumination Approximation (Simplicity and Efficiency)
  - Increase Realism in the Scene without high computation
- Ambient Light's Shadows (Better depth clue than direct lighting's)
  - It darkens creases, holes and surfaces that are close to each other
- Ambient = Fake
- Ambient Light Shadows = Fake Light Shadows

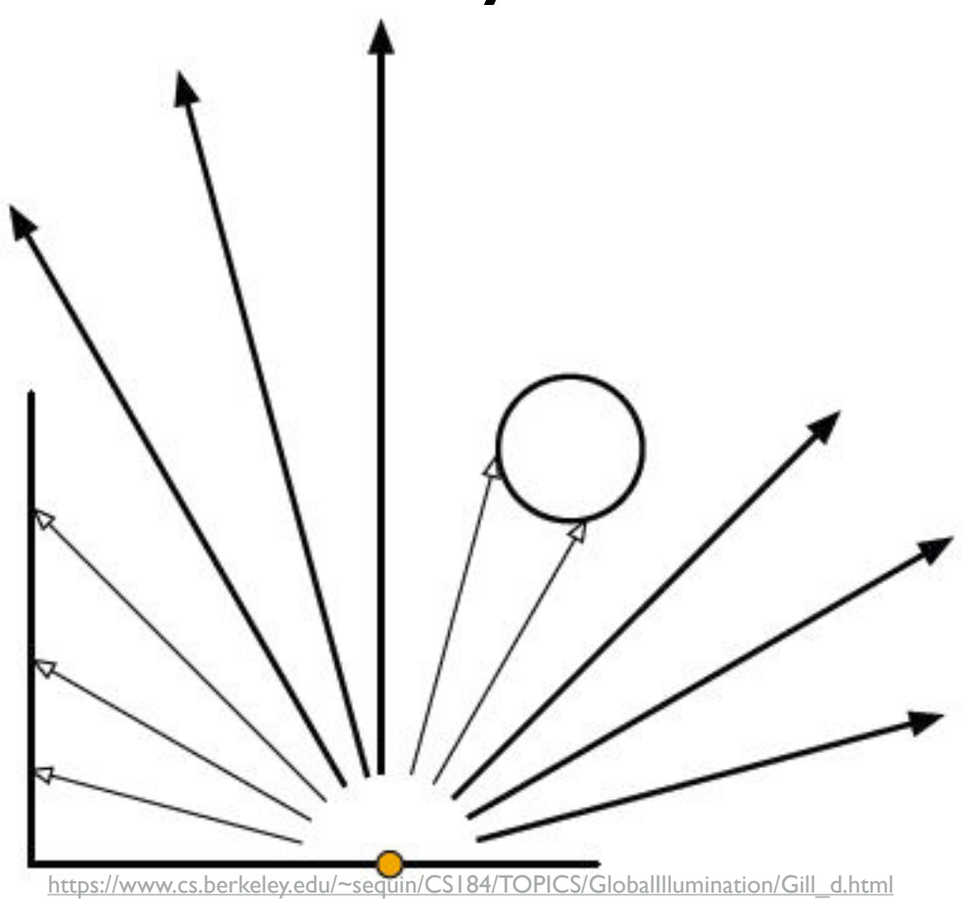
Diffuse Only

Ambient Occlusion

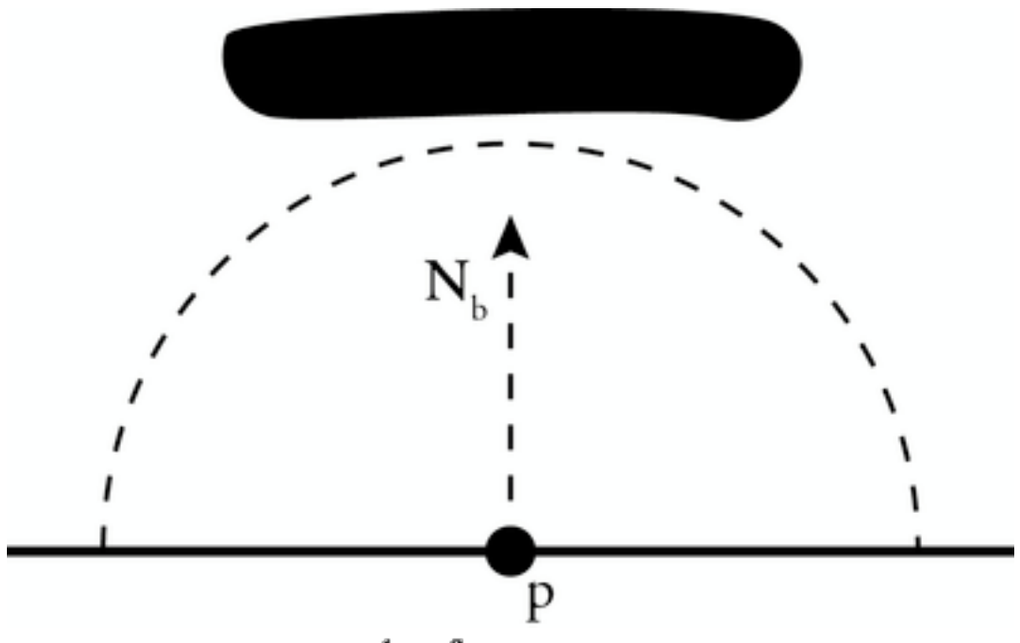
Combined

Ambient Occlusion

- How easy for a surface to be touched by light
- From a point on the surface, cast many rays at random
- Proportion of the sum of light rays that hit the surface
- Still not really efficient



https://www.cs.berkeley.edu/~sequin/CS184/TOPICS/GlobalIllumination/Gill_d.html



$$A_{\bar{p}} = \frac{1}{\pi} \int_{\Omega} V_{\bar{p},\hat{\omega}} (\hat{n} \cdot \hat{\omega}) \, \mathrm{d}\omega$$

Ambient Occlusion

- Examples

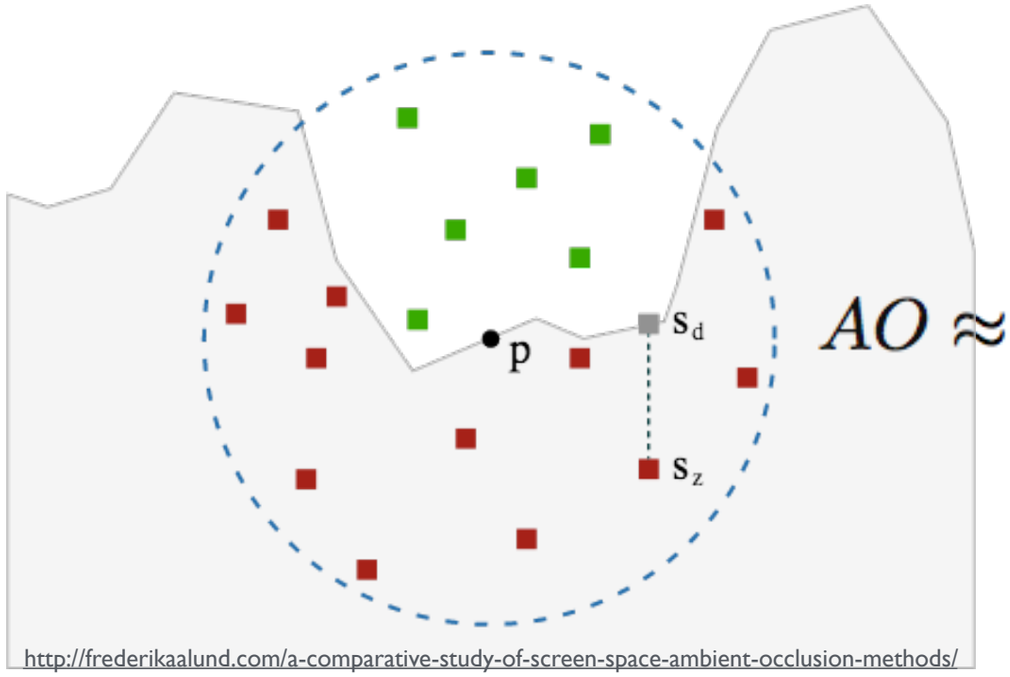
http://ozancicek.com/ambient-occlusion-

http://bobby-parker.com/architectural-rendering-blog/all-you-need-to-know-

http://www.design-corps.co.uk/tag/ambient-occlusion/

Screen Space Ambient Occlusion (SSAO)

- Approximating Ambient Occlusion (for real-time)
  - Developed at Crytek (Crysis Game) in 2007
- Analyzing pixel depth (from Depth Buffer)
  - Depth differences of sampled points around the pixel
- Rather local and view-dependent



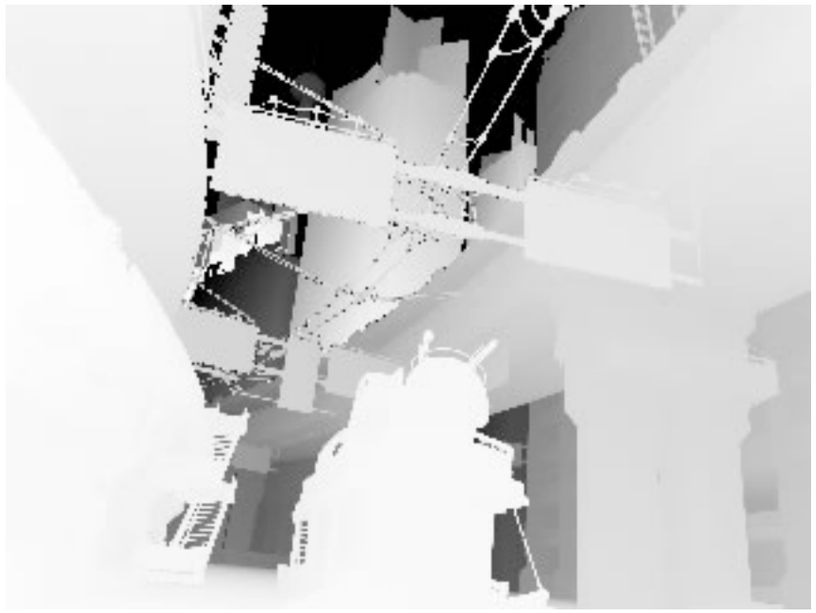
http://frederikaalund.com/a-comparative-study-of-screen-space-ambient-occlusion-methods/

$$AO \approx \frac{1}{N} \sum_{n=1}^{N} V'(\mathbf{s}_n)$$

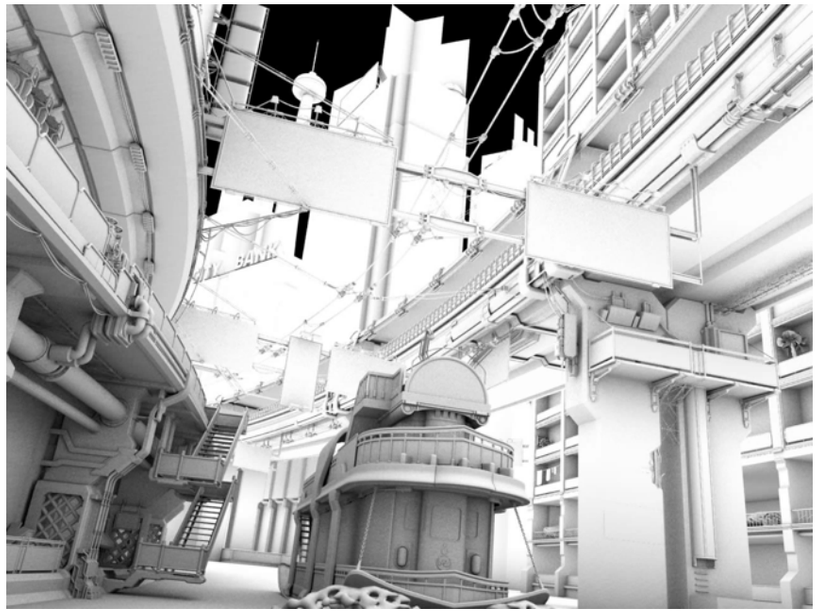
Screen Space Ambient Occlusion (SSAO)

- Examples

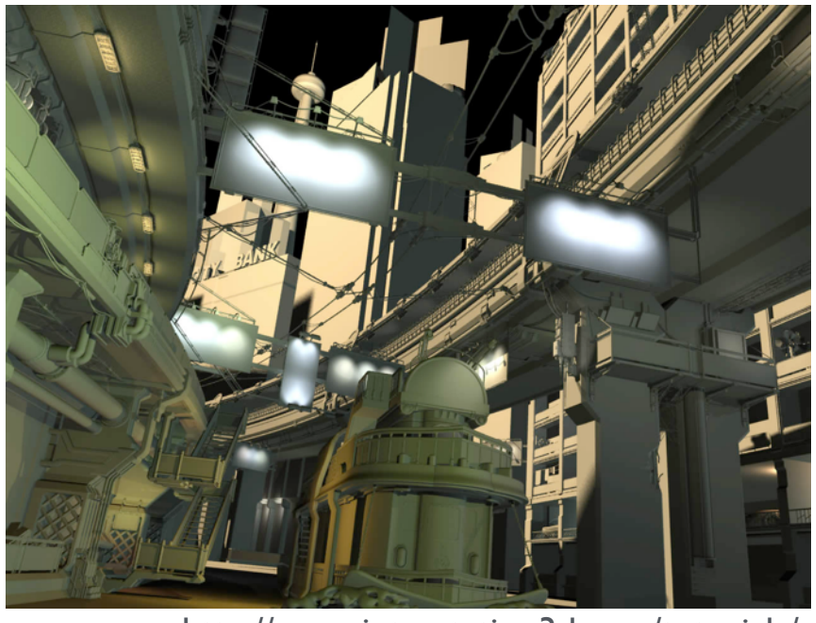
Depth Buffer



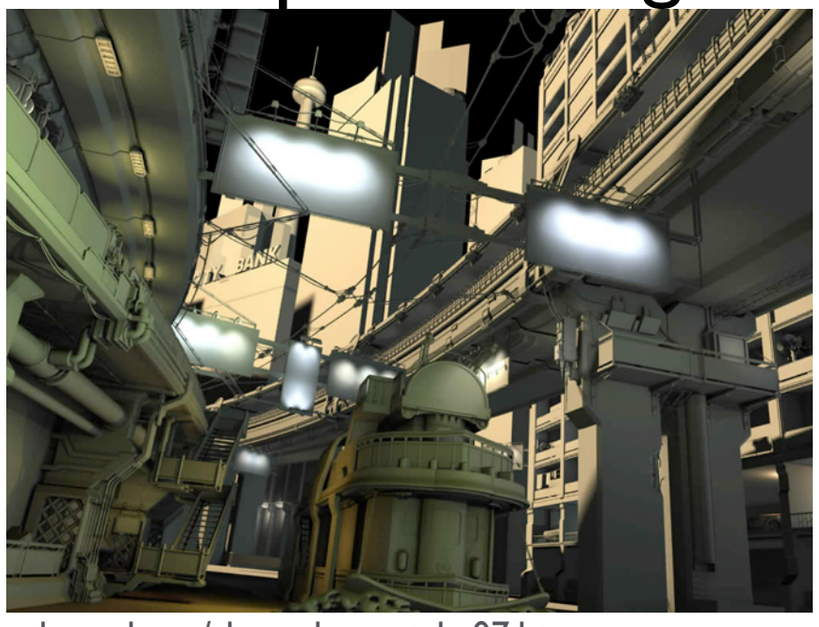
SSAO



First Pass

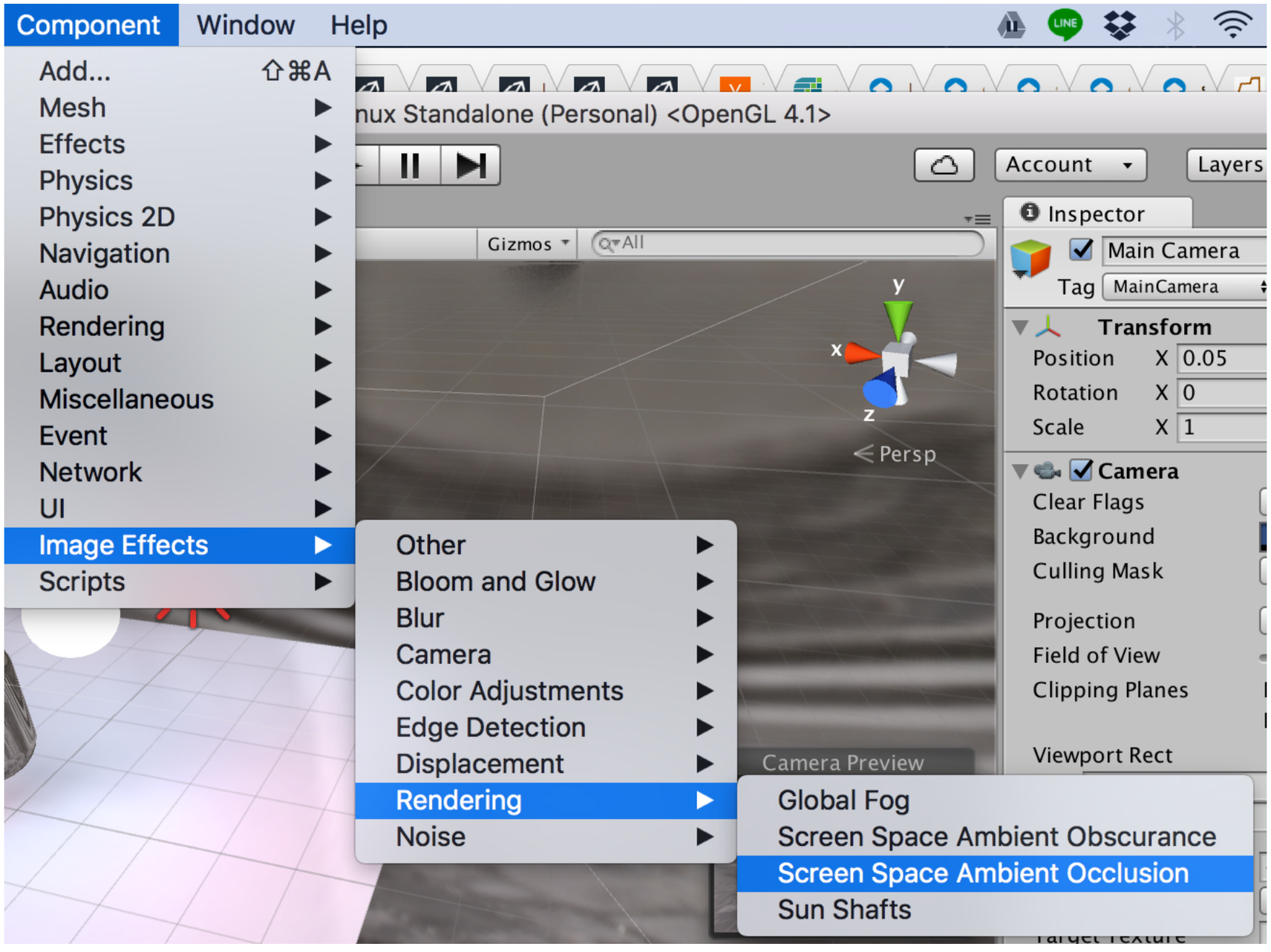


Composite Image



http://www.interstation3d.com/tutorials/making_slow_decay/slow_decay_take07.htm

SSAO in Unity



To see the effect clearly, you need a rather complex scene

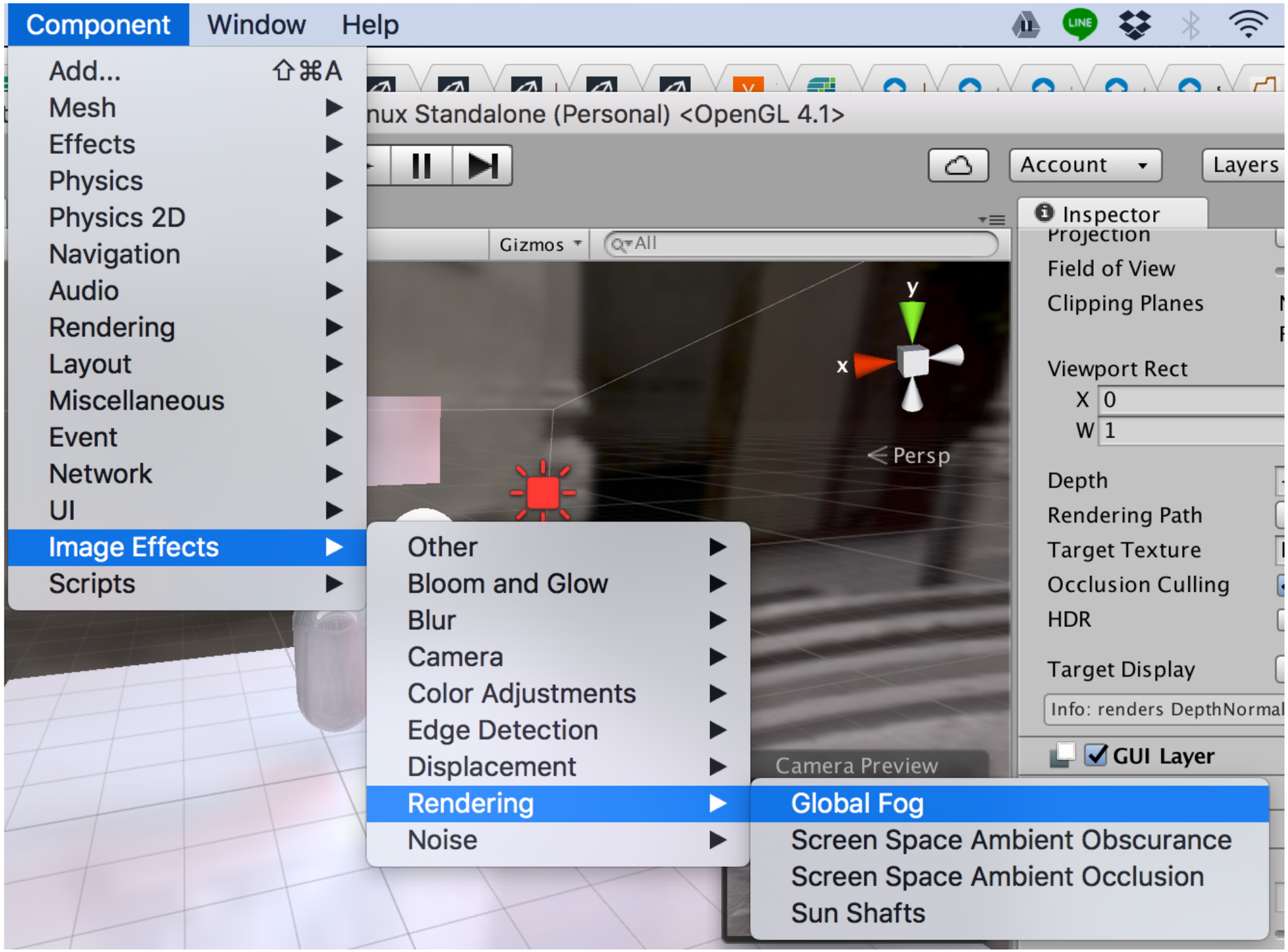
Global Fog

- Fog = Volumetric Light Scattering = Really Expensive

- Image-Based Distance Fog

- Image-Based Height-based Fog

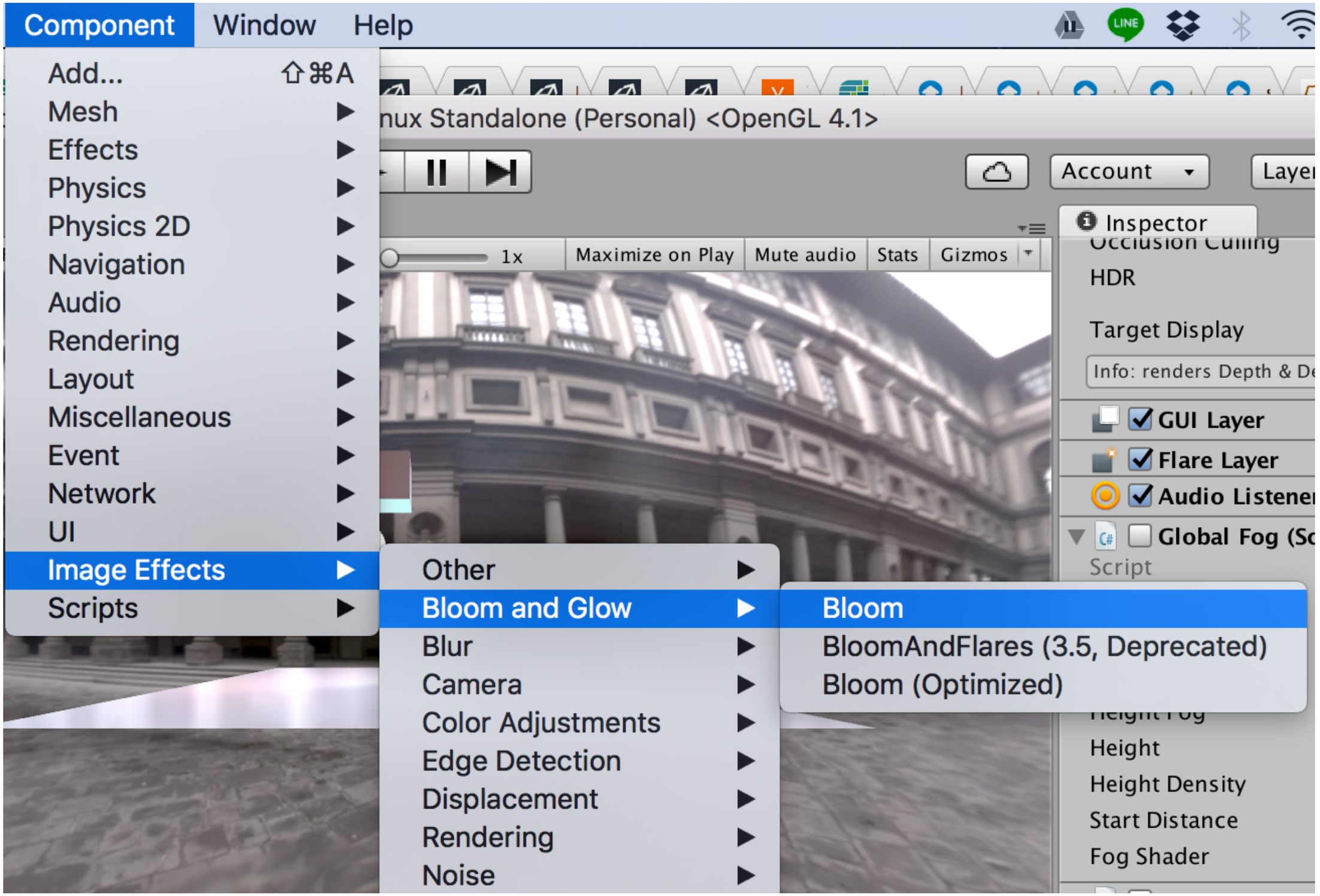
Global Fog in Unity



Bloom

- Camera artifacts
- Simulating extremely bright light where the light extending from the borders of bright areas in an image
- Using of HDR images
- Blurring the highlights

Bloom in Unity



More Camera Artifacts

- Vignetting
  - Decrease in brightness of a photograph around its edges

- Chromatic Aberration
  - Color Distortions from lens

- In Unity: Component -> Image Effects -> Camera -> Vignette and Chromatic Aberration

Camera Artifact Examples

https://www.youtube.com/watch?v=72RqpItxd8M

Image Effect Lab

- Create a new project
- Import “Corridor Lighting Example” to the project
  - Window -> Asset Store
  - At Corridor Light Example -> Import
- Open the Scenes -> CorridorDemo
  - Wait a few seconds for Game view to properly render the scene
- Play it! (check frame rate at Game view -> Stats)

Image Effect Lab

- Click on Cameras -> MainCamera
- There are many image effects under the MainCamera
  - As you can see, they are all scripts (so, you can check their codes)
- Try disable (uncheck) them all and see the real scene
- Enable each image effect individually and try tweaking its parameters to see the effect
- Try move “Bloom” before the “Tonemapping”
  - Order does matter! - it runs from top to bottom
  - Try move other image effect orders

Lab 2

- Make a nice scene/animation using any combination of existing and new image effects on the corridor example
- To add new image effects
  - Component -> Image Effects -> ANY!
- Submission (10 points) - at the end of the class
  - Login to WikiSpace
  - Create a new wiki page with title "your name - lab2" e.g. tanasai - lab2
  - Tag your page with lab2_2016
  - Post your result as an image or youtube link
  - Add text for the name of the new added image effects

Project Alert

Tutorial 1 DUE TODAY!