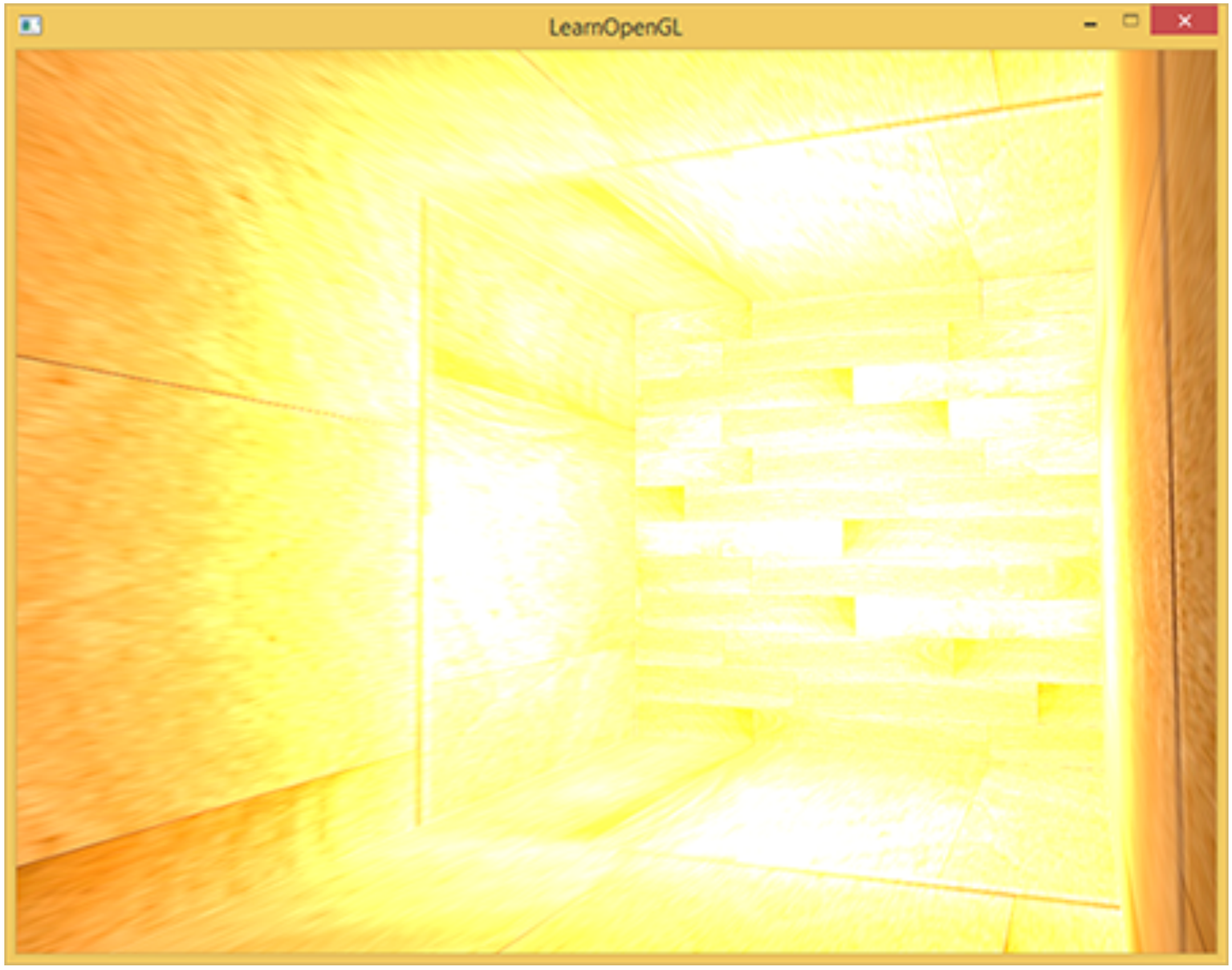
CS 300: Advanced Computer Graphics
High Dynamic Range Rendering

Pushpak Karnick

Problem: Too many lights are bright



What is HDR?

Computer Graphics Pipeline

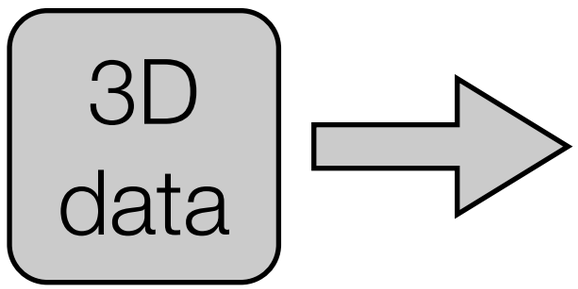
What is HDR?

Computer Graphics Pipeline

3D data

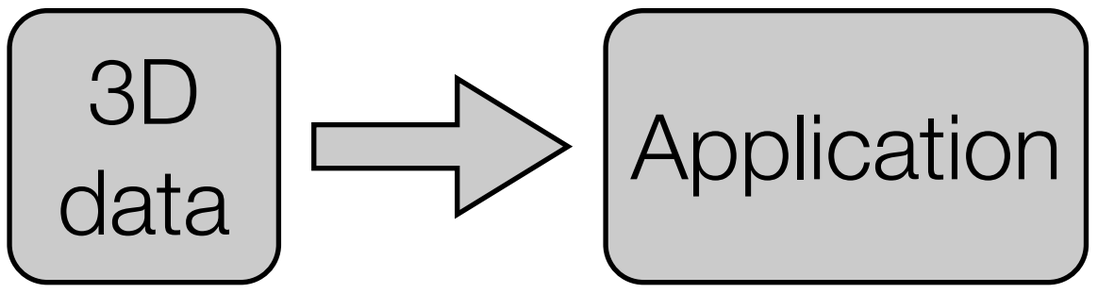
What is HDR?

Computer Graphics Pipeline



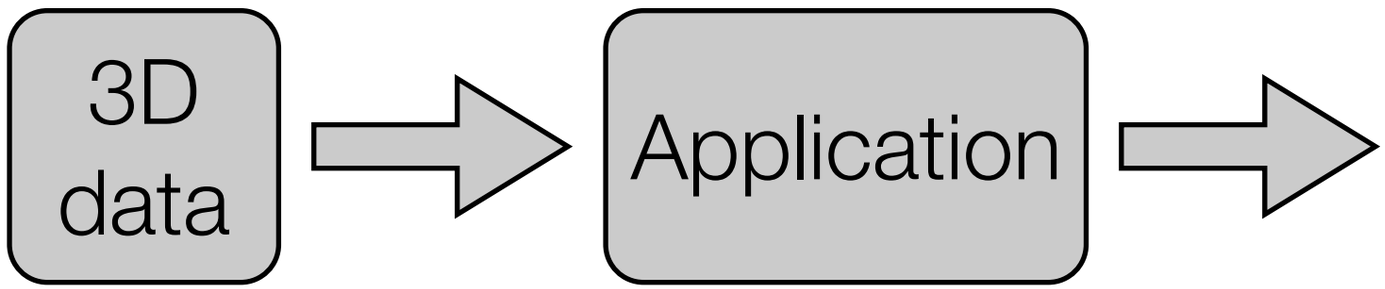
What is HDR?

Computer Graphics Pipeline



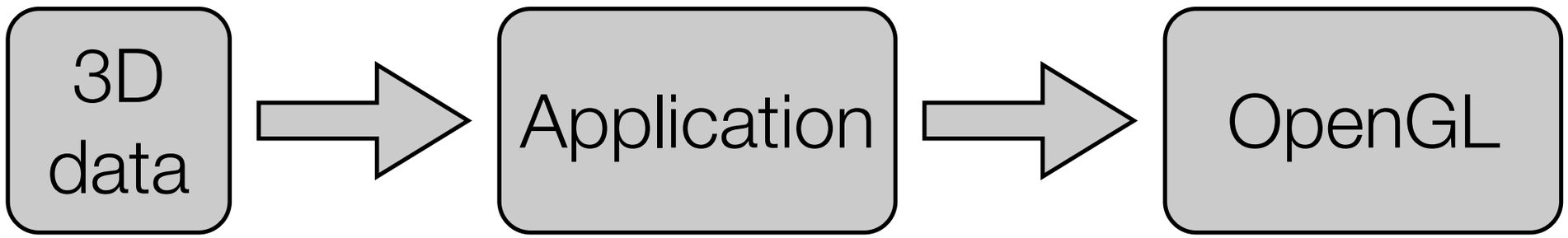
What is HDR?

Computer Graphics Pipeline



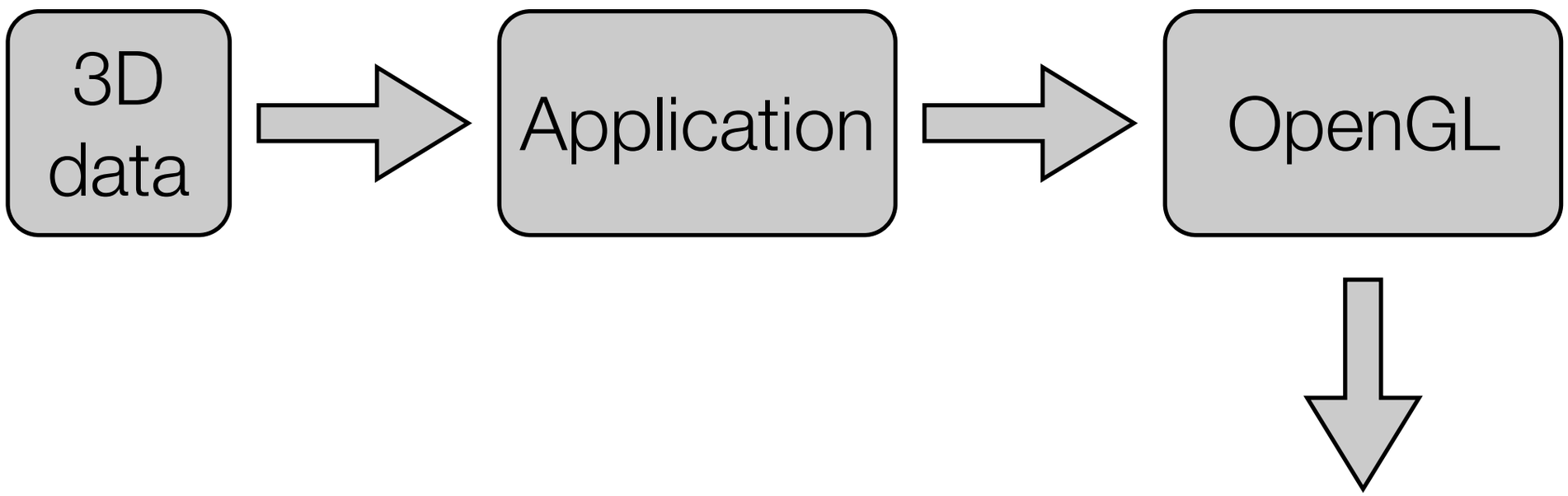
What is HDR?

Computer Graphics Pipeline



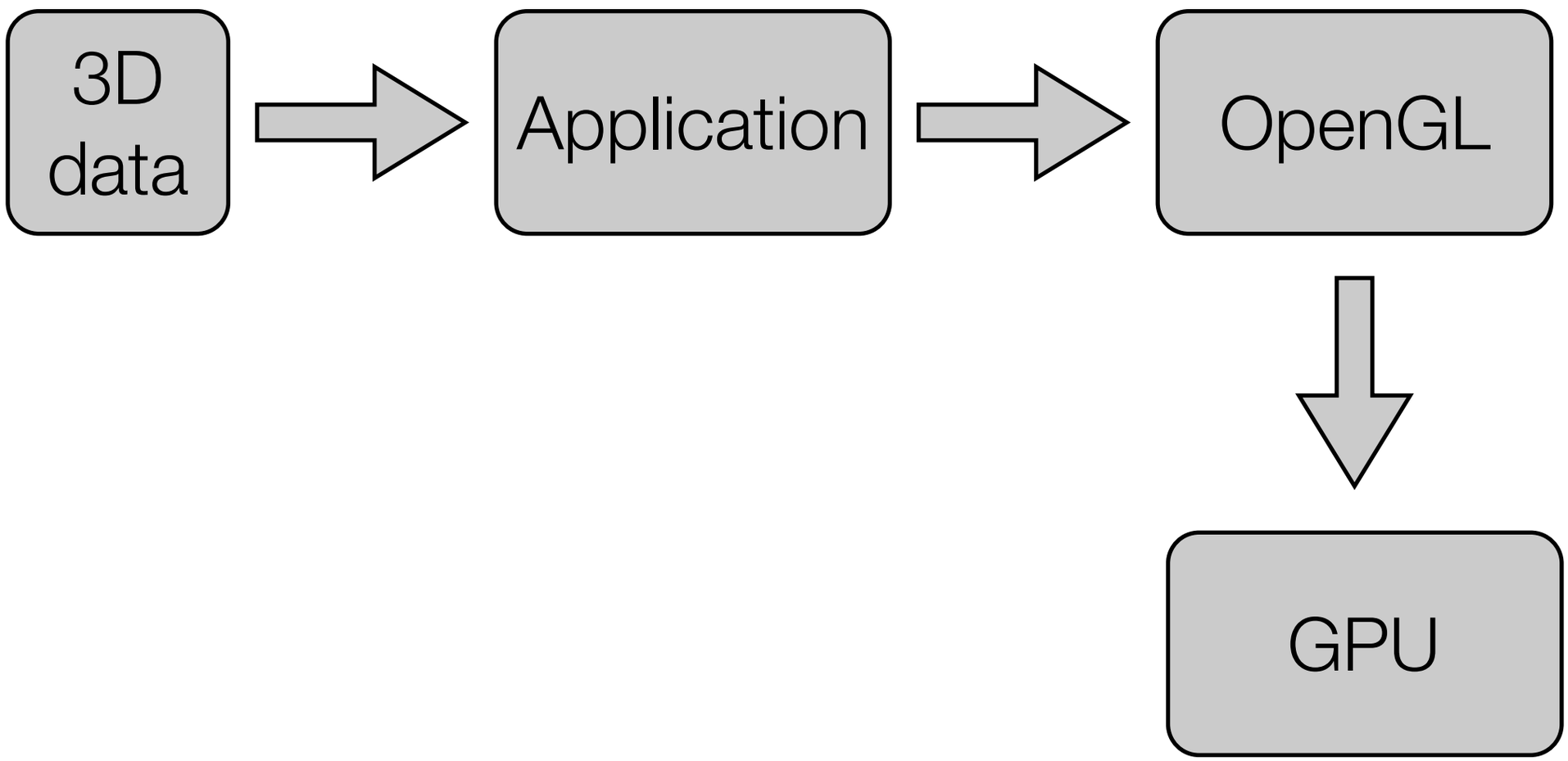
What is HDR?

Computer Graphics Pipeline



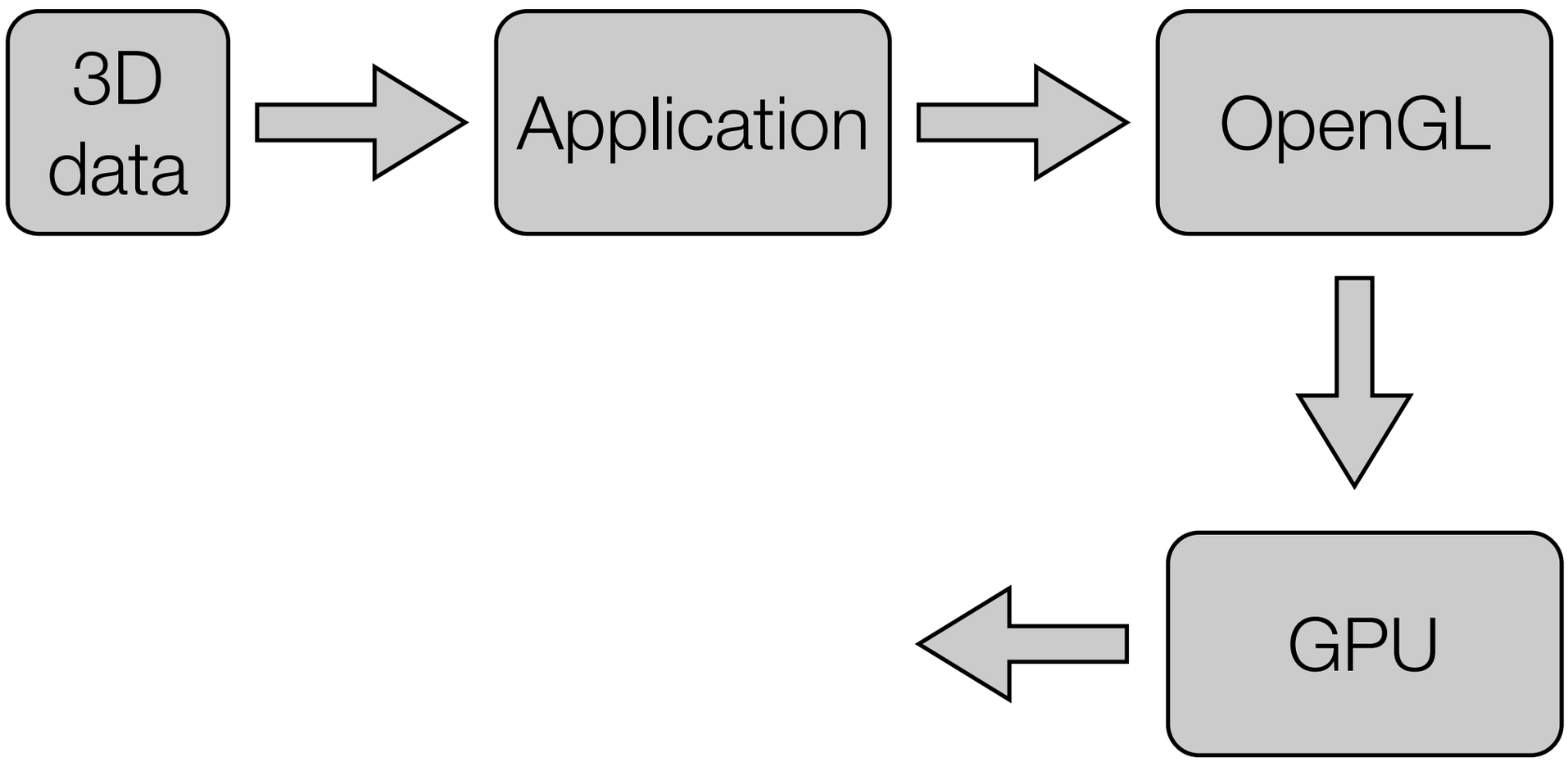
What is HDR?

Computer Graphics Pipeline



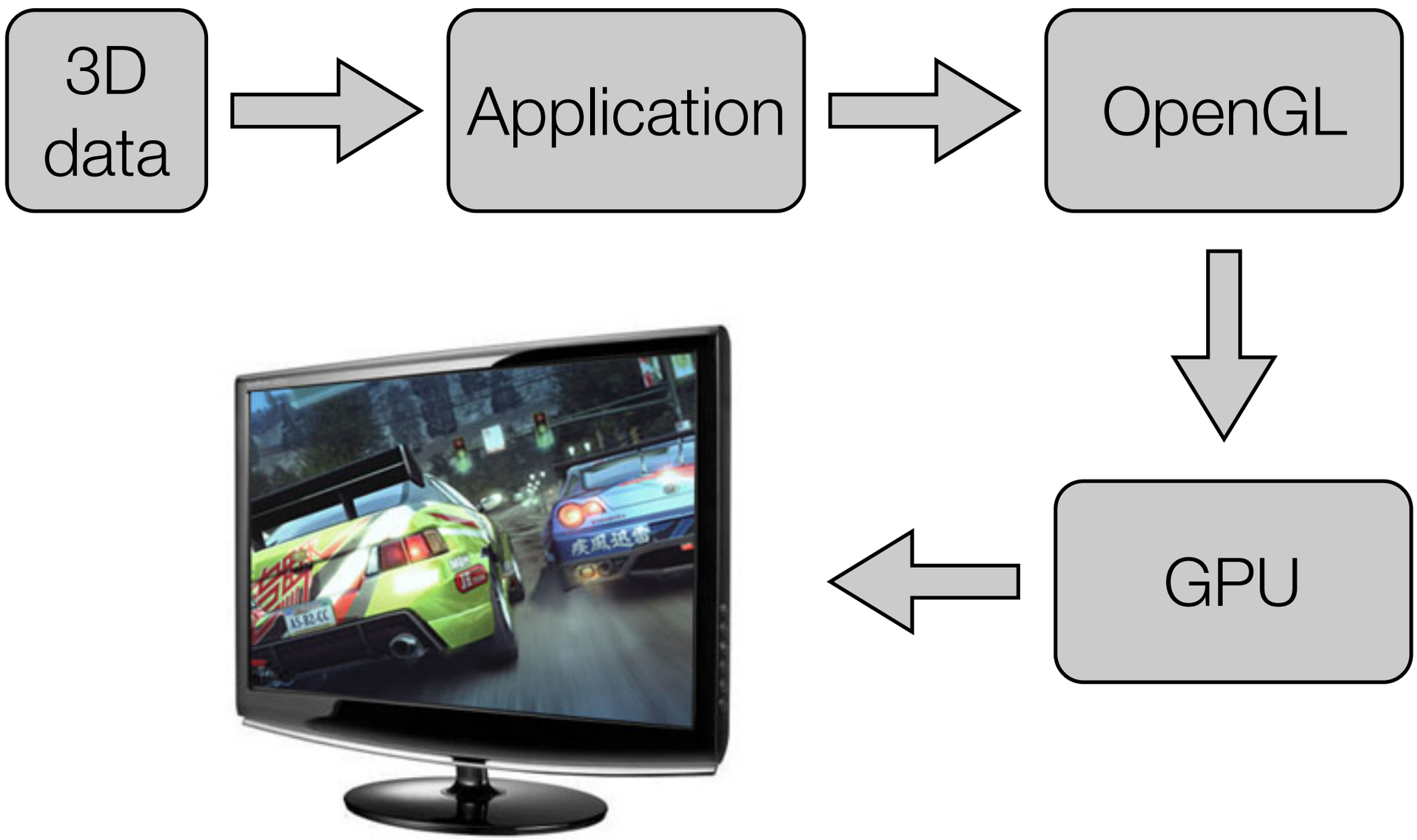
What is HDR?

Computer Graphics Pipeline



What is HDR?

Computer Graphics Pipeline



Framebuffer Format

- Brightest and darkest portions in a scene
- Highlights and shadows in real world
- 24-bit RGB, 8-bits per channel
- Min value : 0
- Max value : 255

Framebuffer Format

- Brightest and darkest portions in a scene
- Highlights and shadows in real world
- 24-bit RGB, 8-bits per channel
- Min value : 0
- Max value : 255

Framebuffer Format

- Brightest and darkest portions in a scene
- Highlights and shadows in real world
- 24-bit RGB, 8-bits per channel
- Min value : 0
- Max value : 255

Contrast Ratio

- Ratio of maximum intensity to minimum intensity of **displayed** color values
  - Usually, the minimum is 1, not zero
- Hence, for 8-bit images
  - Contrast ratio per channel (intensity) is 256:1
- Compare to
  - Real world - 100,000 : 1

Capping out

- How to display “real-world” color and contrast on a display device with limited contrast ratio?

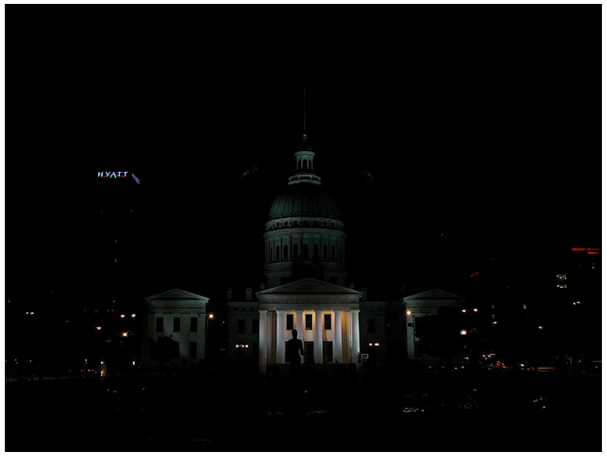
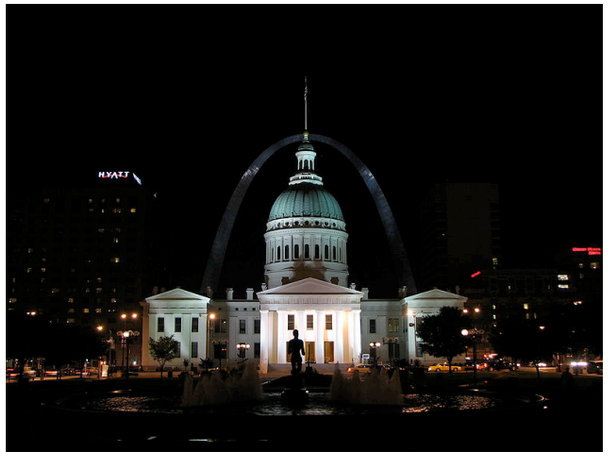
- High Dynamic Range (HDR) methods
  - High Dynamic Range Imaging - creating real-world images
  - High Dynamic Range Rendering - displaying real-world contrast ratio on commodity monitors

High Dynamic Range (HDR) Imaging




Images courtesy
wikipedia.org

High Dynamic Range (HDR) Imaging


Images courtesy
wikipedia.org

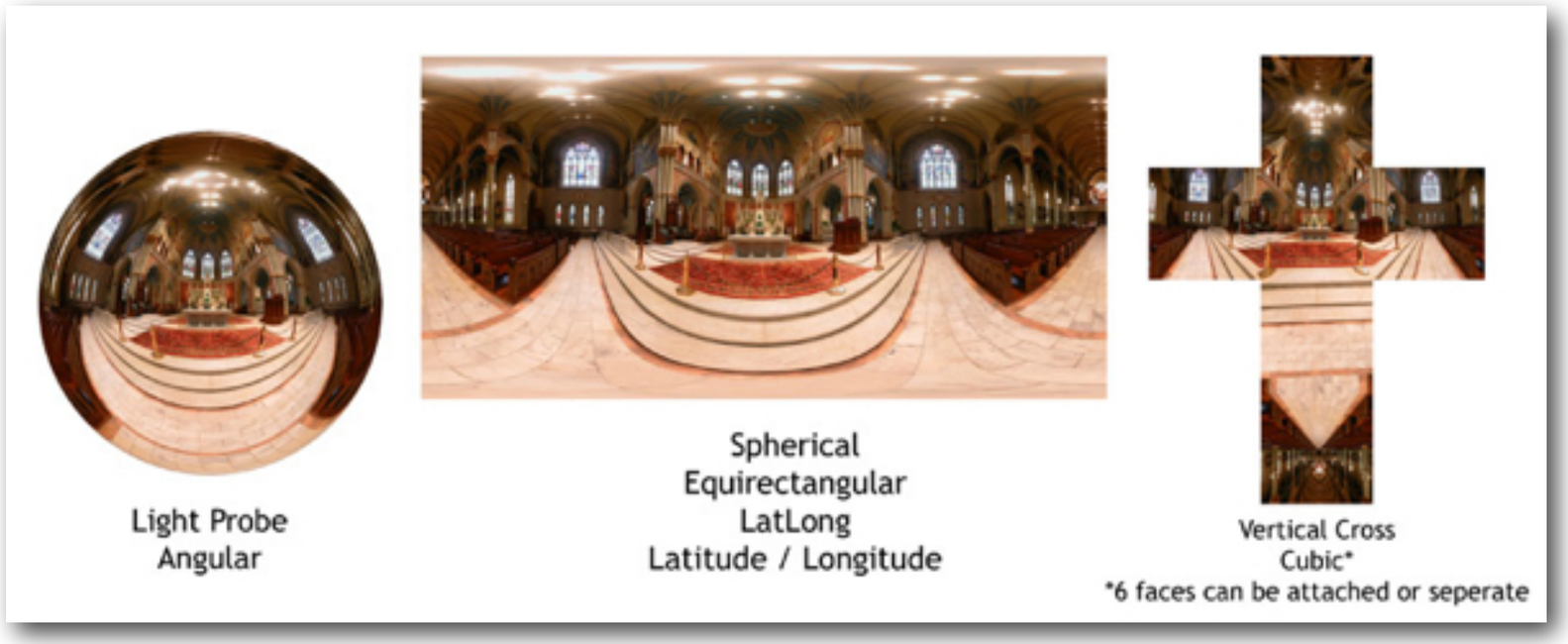
HDR output

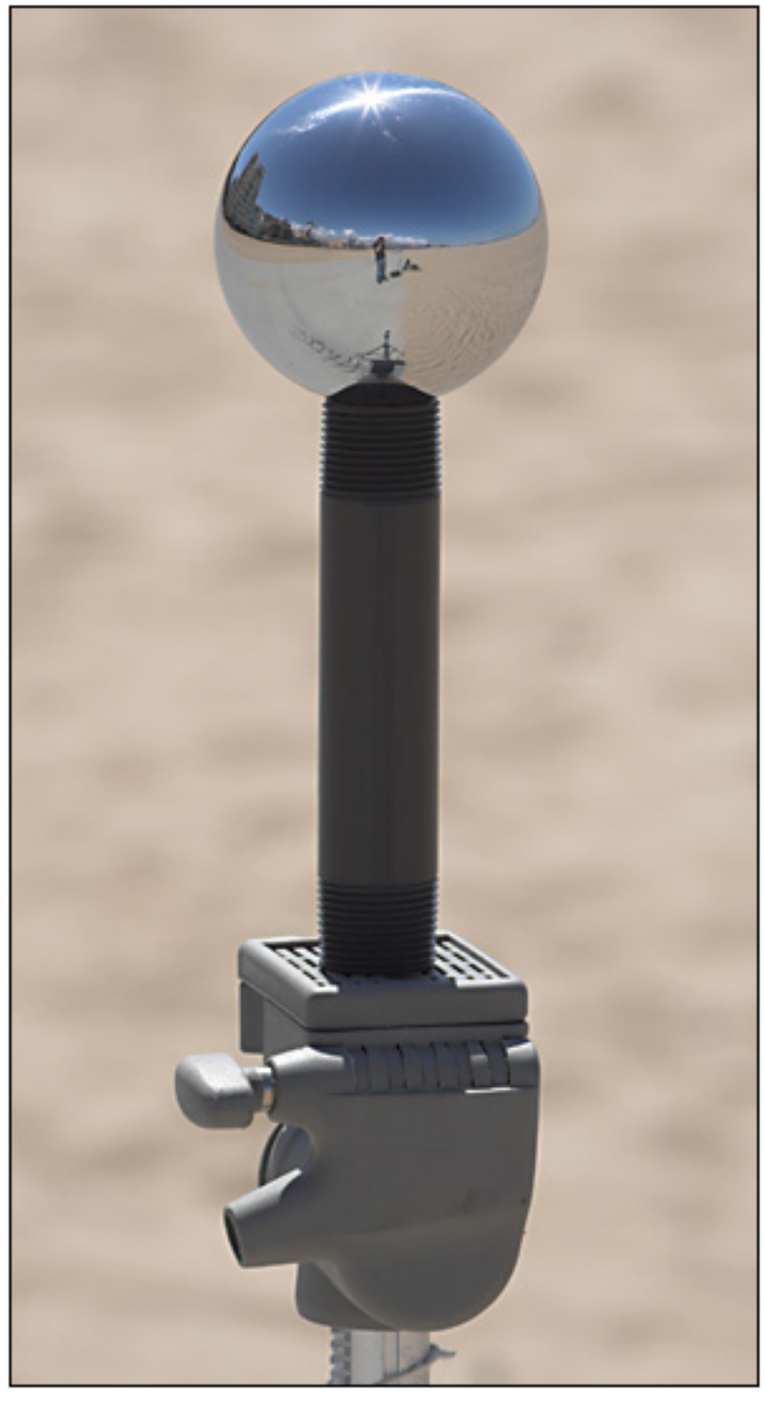
HDR output

Light Probes

- Data collected from real-world
- Similar to earlier approach (merge multi-exposure photographs)
- Combined into an HDR image (Radiance image, *.hdr)

HDR Panoramic Light-Image

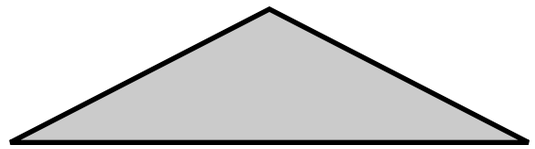




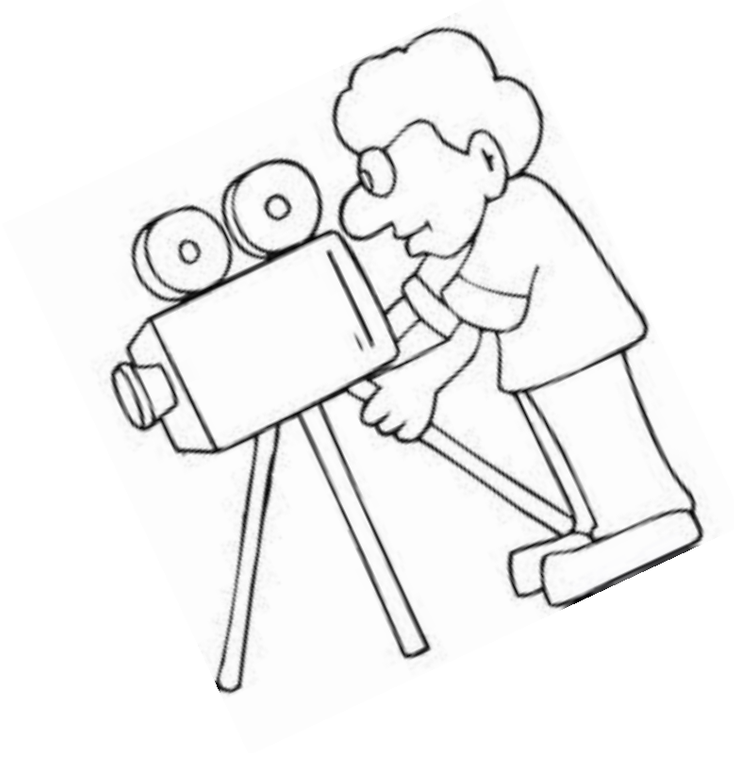
www.hdrlabs.com

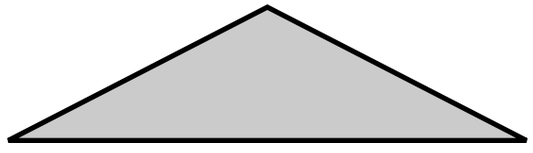
CG Pipeline Revisited

CG Pipeline Revisited

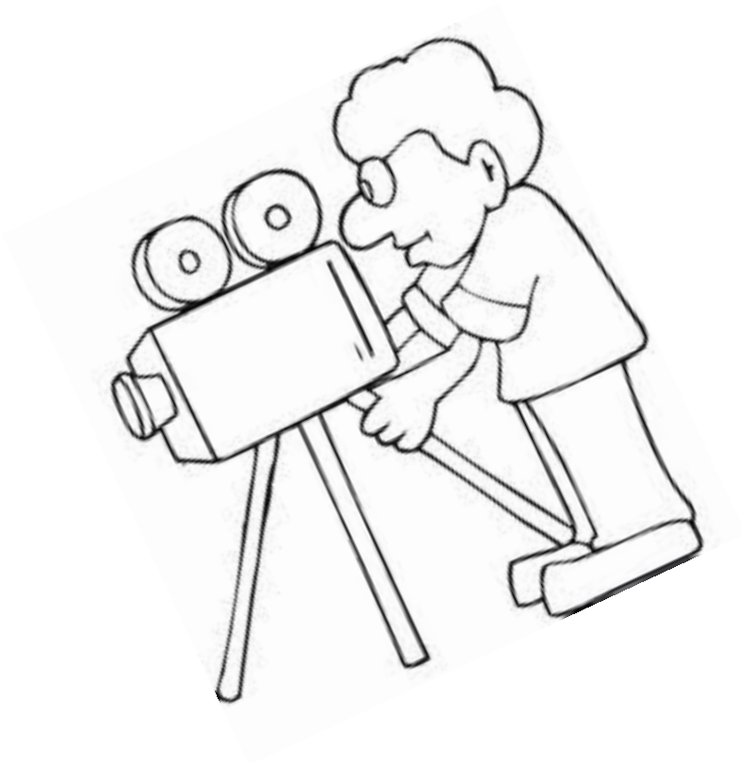


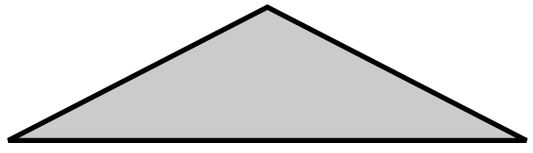
CG Pipeline Revisited



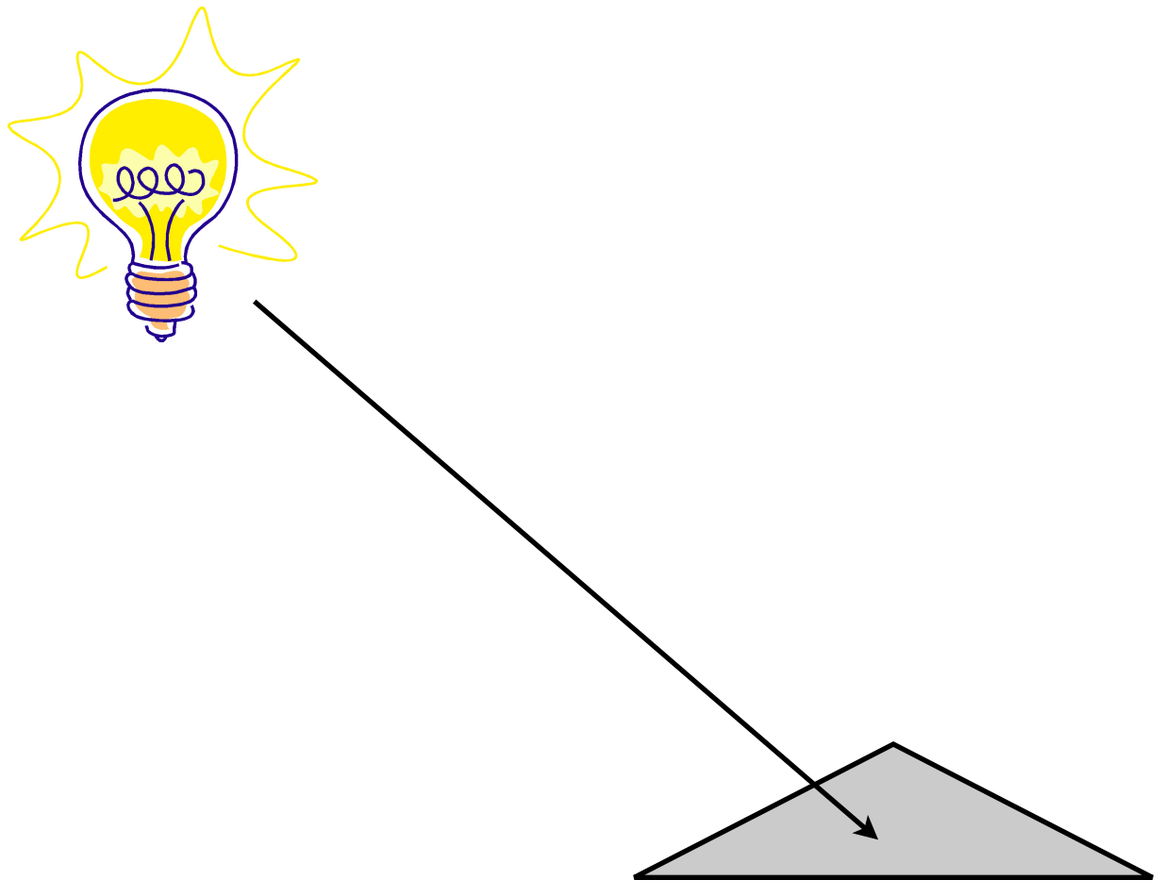


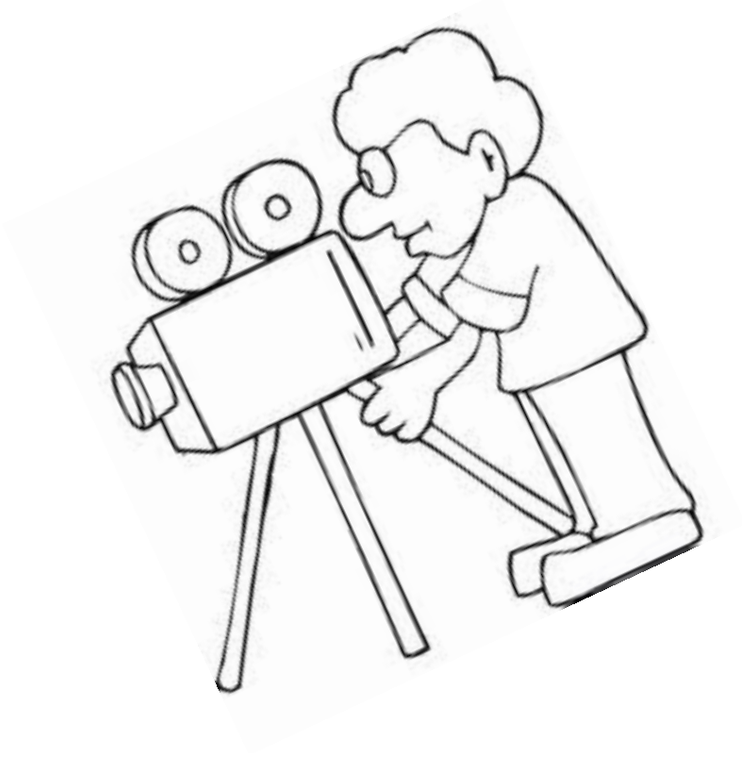
CG Pipeline Revisited



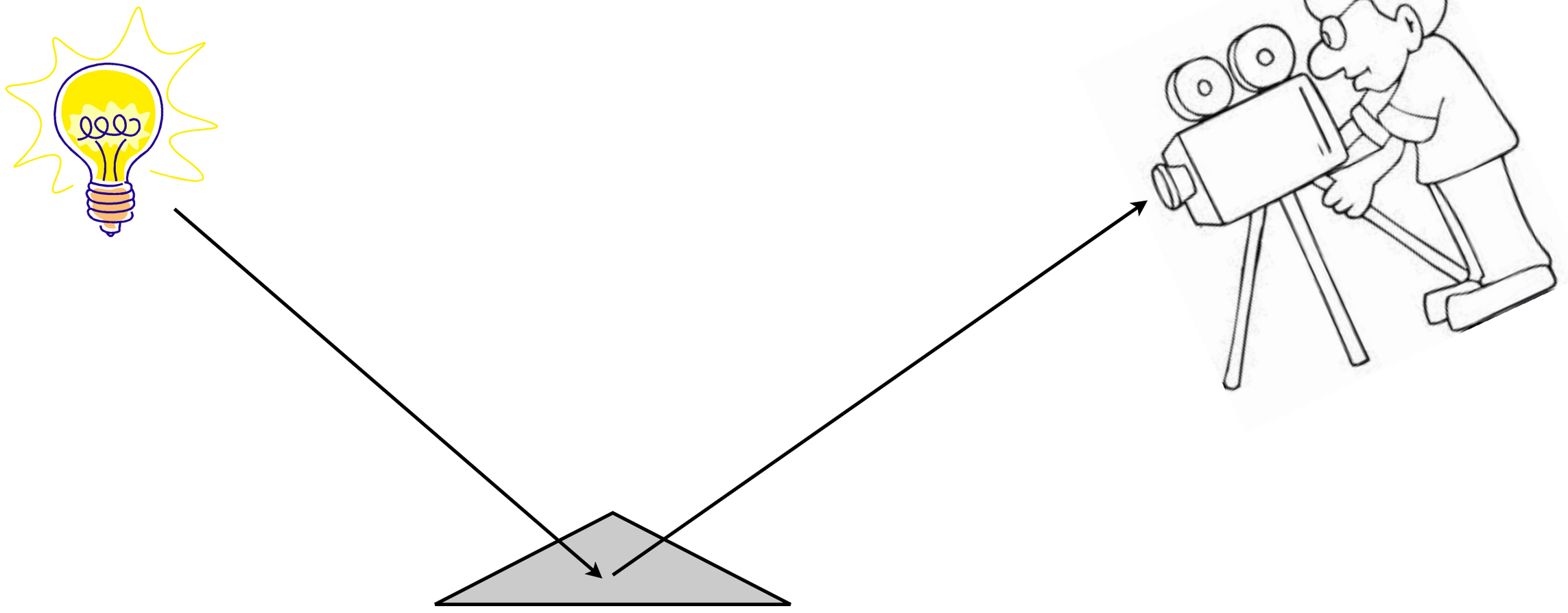


CG Pipeline Revisited

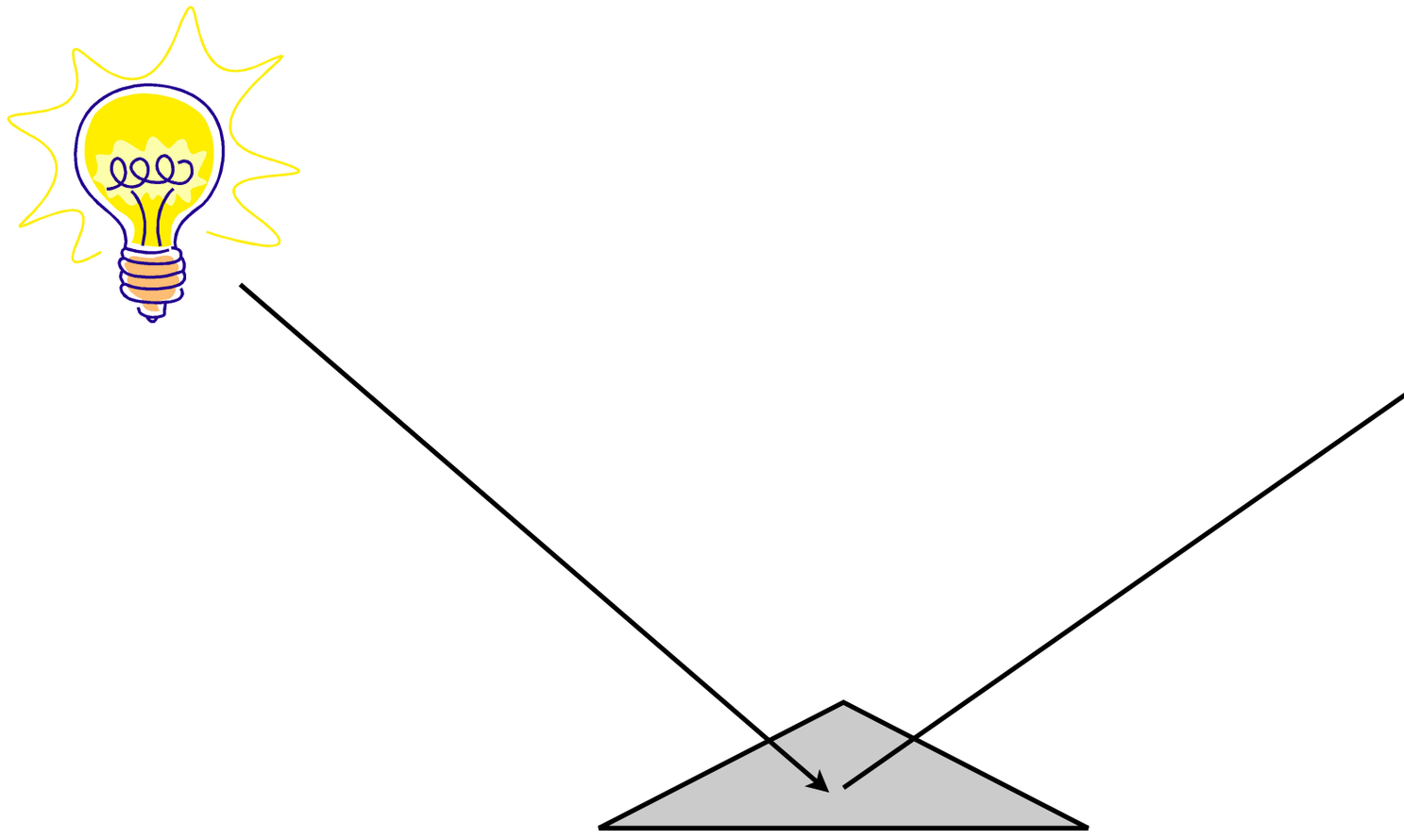


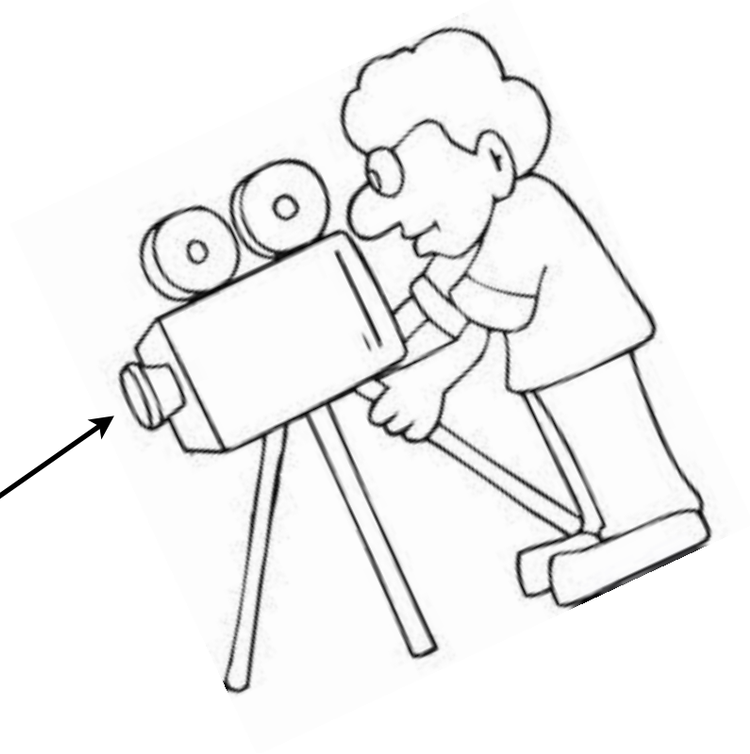


CG Pipeline Revisited



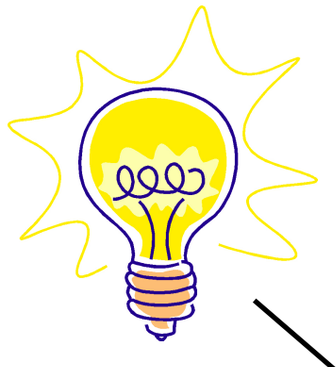
CG Pipeline Revisited



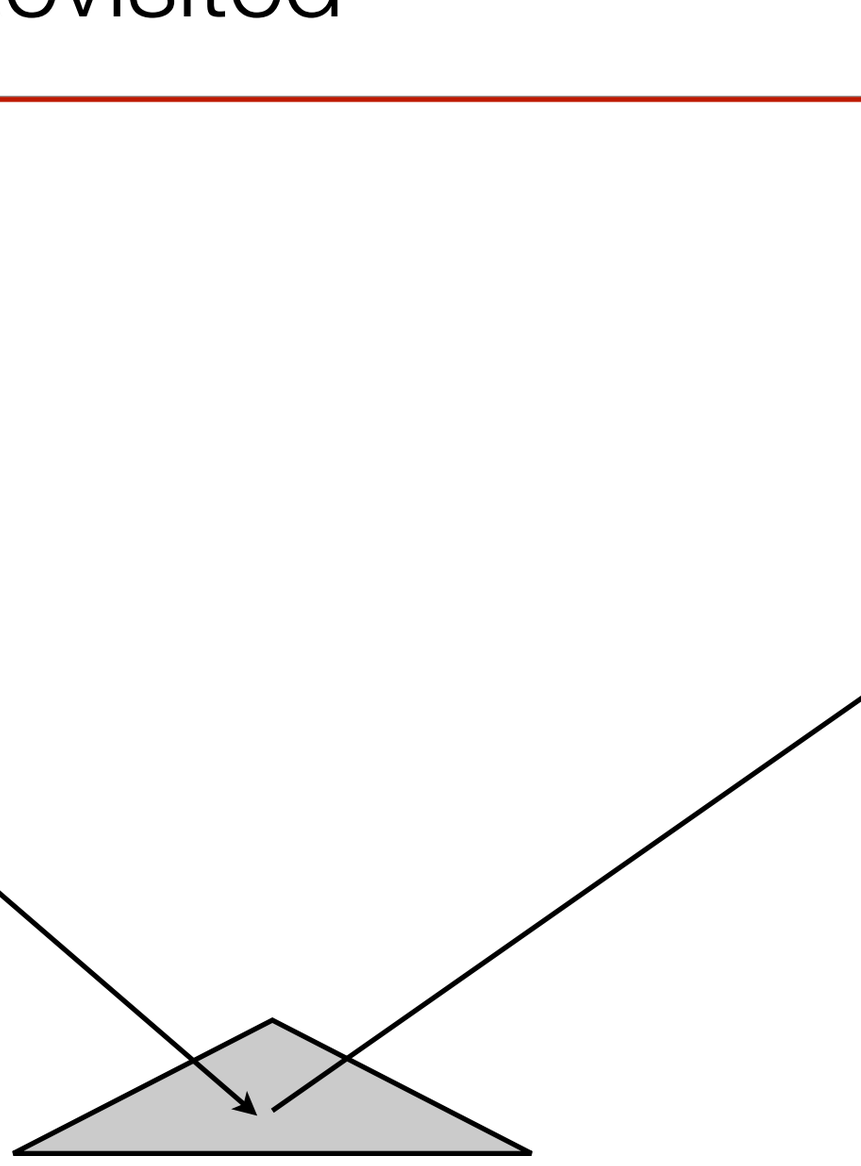


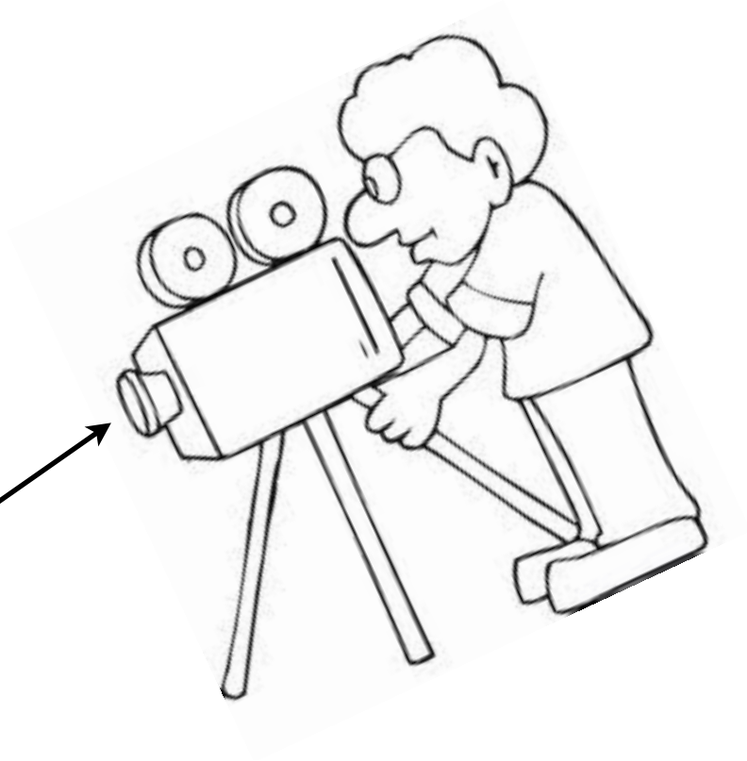
Film
Analog Medium
Higher contrast ratio

CG Pipeline Revisited



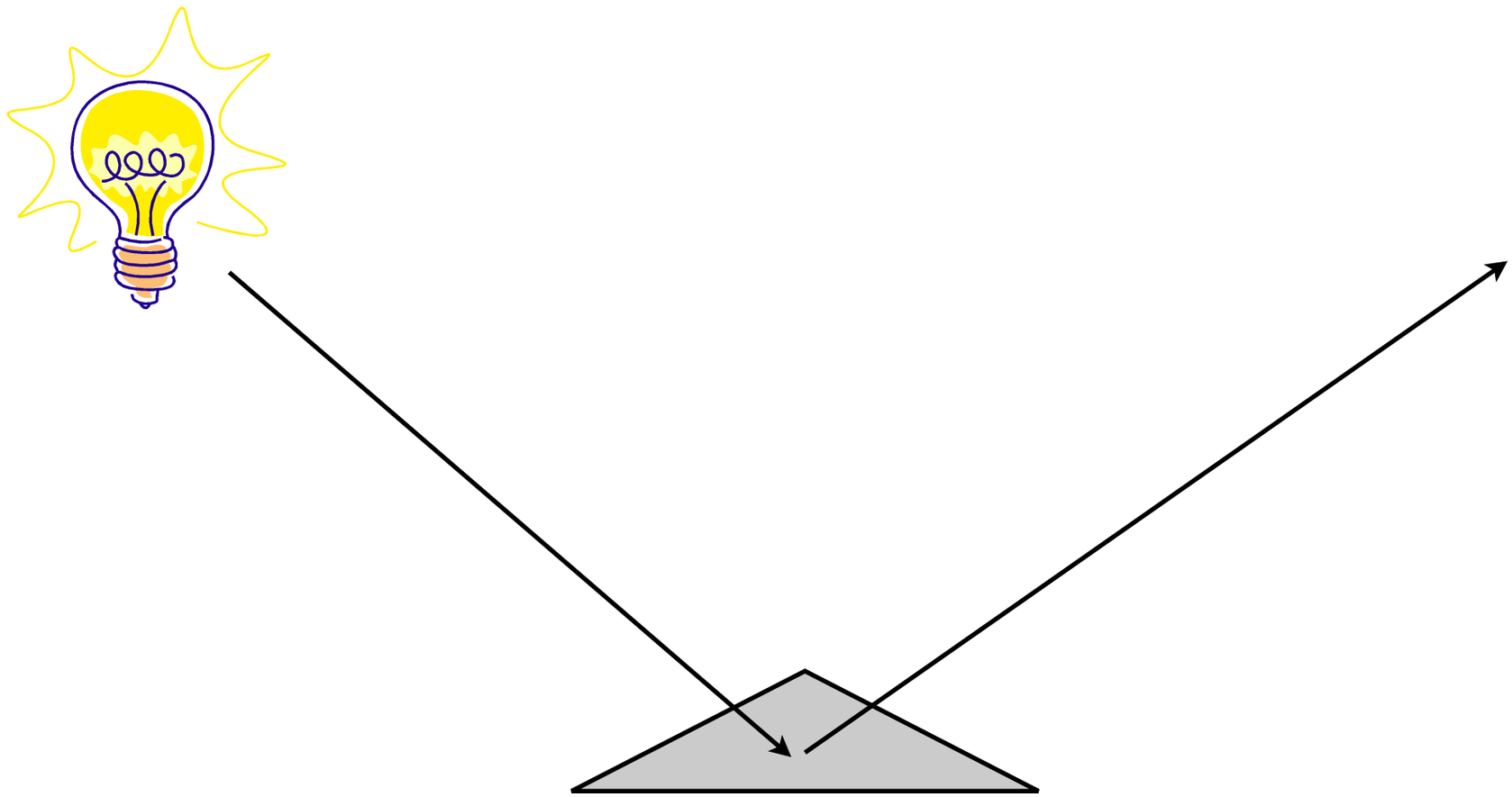
Radiance
“Real” values
floating point
representation



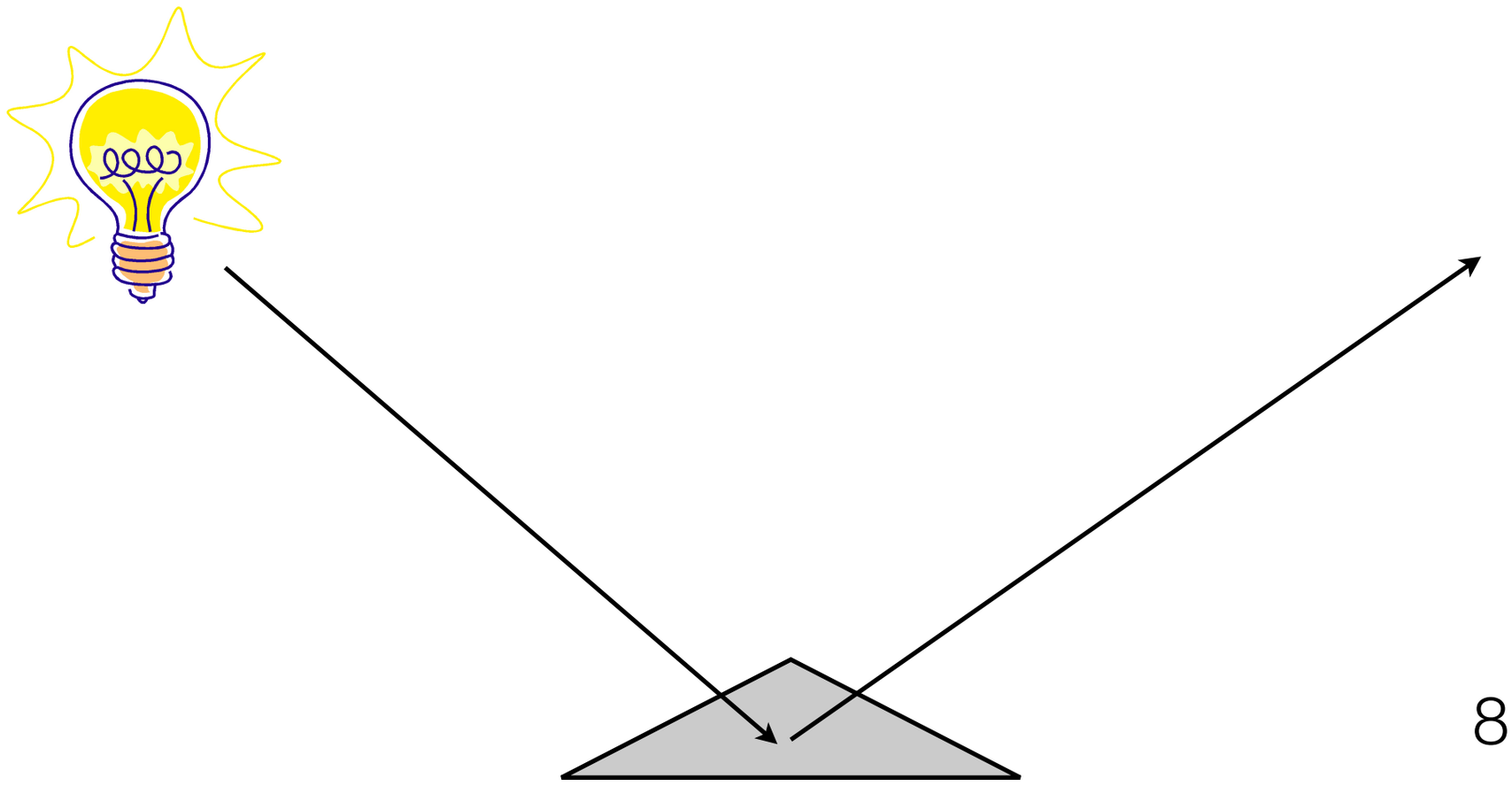


Film
Analog Medium
Higher contrast ratio

CG Pipeline Revisited

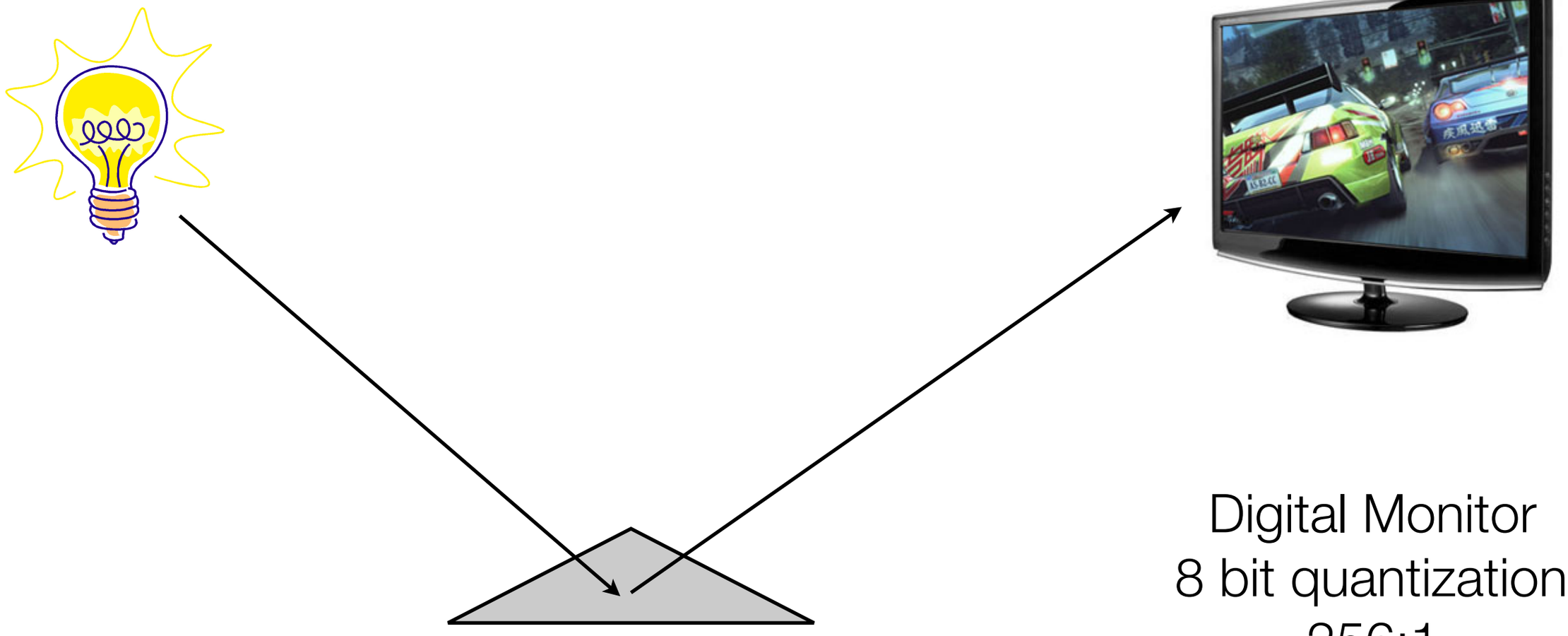


CG Pipeline Revisited



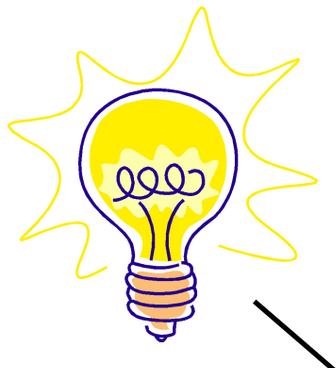
Digital Monitor
8 bit quantization
256:1

CG Pipeline Revisited



Digital Monitor
8 bit quantization
256:1

CG Pipeline Revisited



Digital Lights
8 bit quantization
3 wavelengths
256:1

Digital Monitor
8 bit quantization
256:1

HDR Pipeline Motivation

- How to use real-world data (i.e. floating point values) for our lighting calculations?

- Ordinary framebuffer is integer-based representation with 256:1 contrast ratio

- What if our output of the fragment shader could be floating point instead of integers?

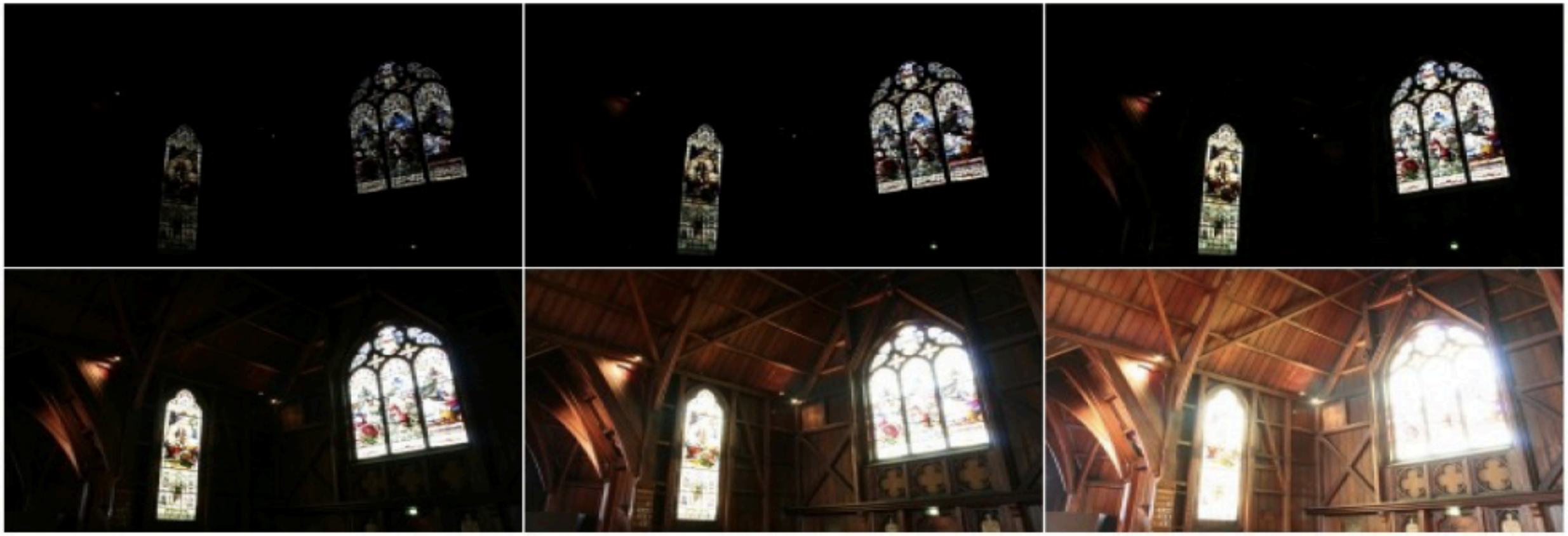
  - Floating point render targets

  - Framebuffer Objects (FBOs)

HDR

- Two types of operations in practice
- HDRI - High Dynamic Range Imaging
  - Applied in photography and image processing
- HDR(R) - High Dynamic Range Rendering
  - Used to display larger contrast ratio data on a lower contrast ratio medium
  - For e.g. FBO to monitor screen

HDR Imaging

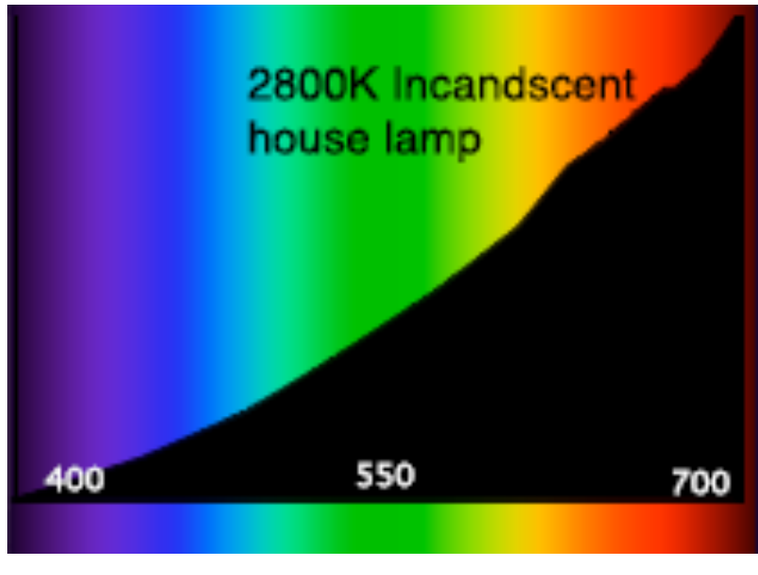


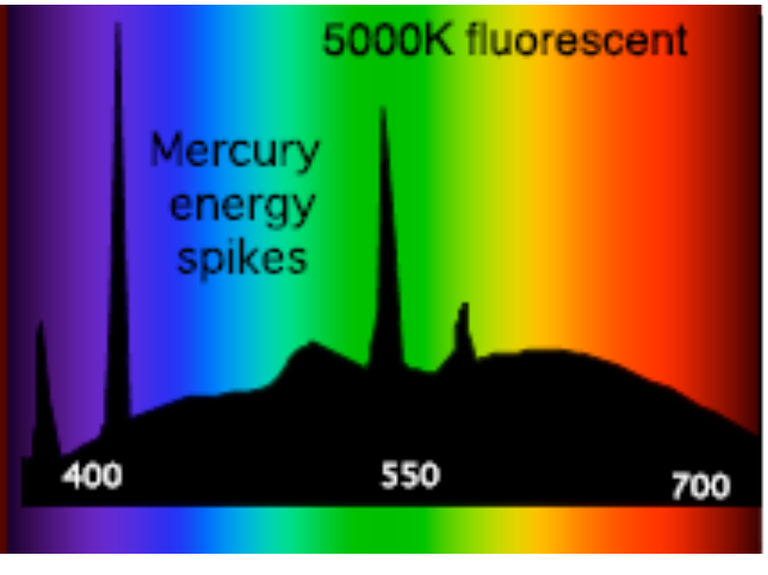

3
138
331
62


3
138
331
62

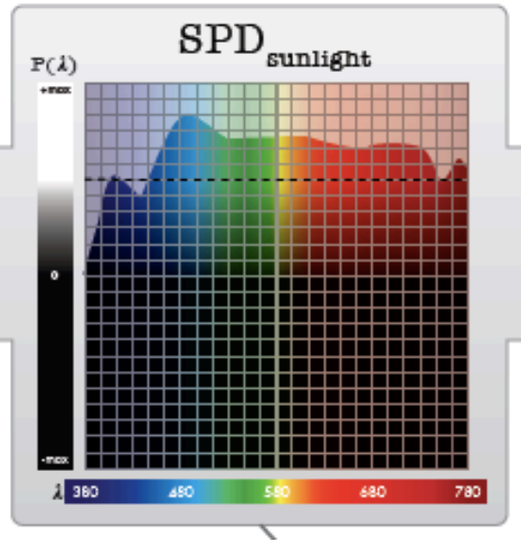
How to obtain HDR lights?

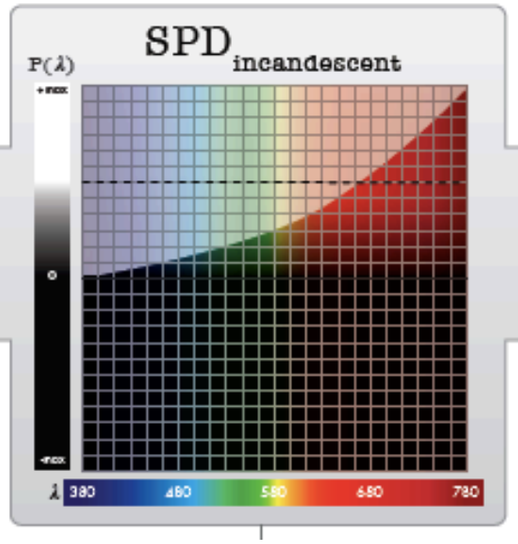
- Spectral Distribution Function

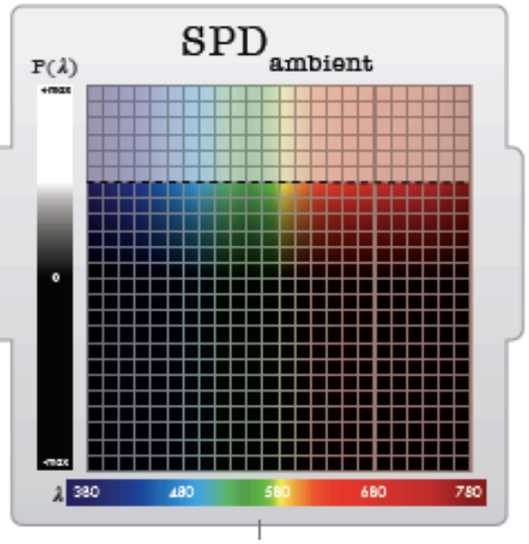




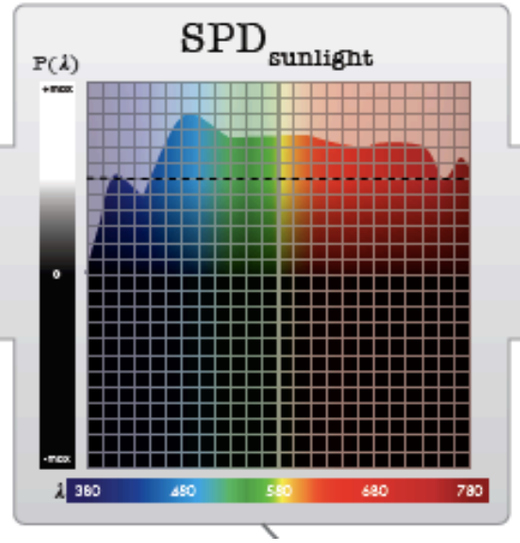
Sample Light Sources

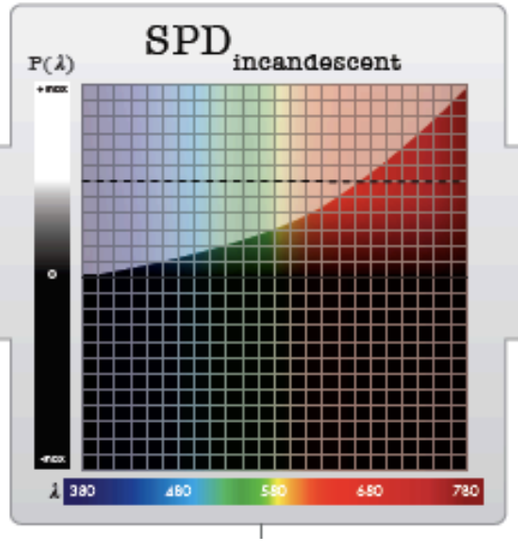


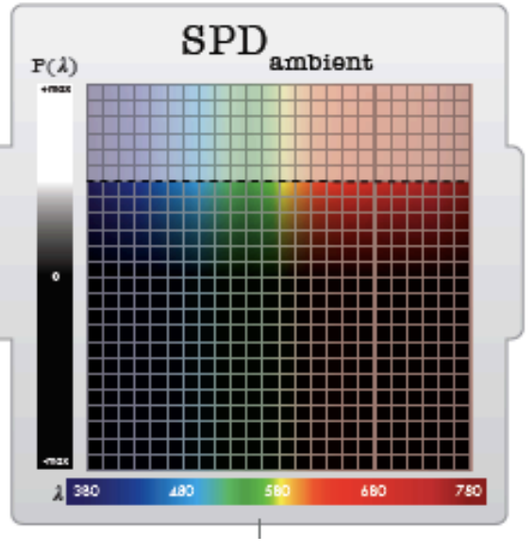




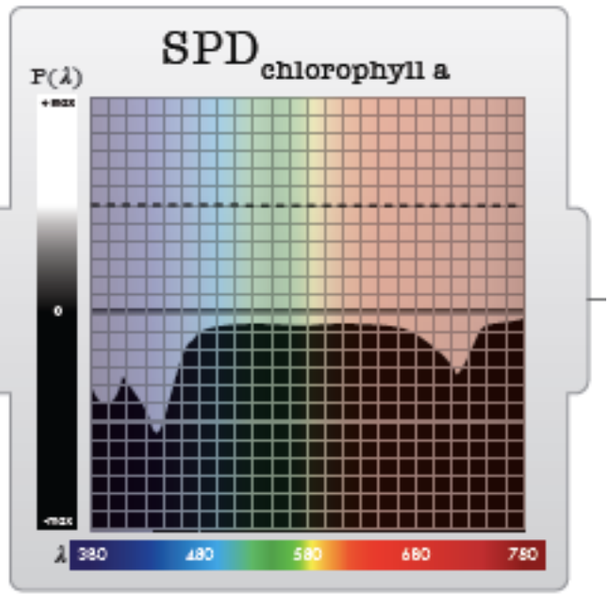
Sample Light Sources



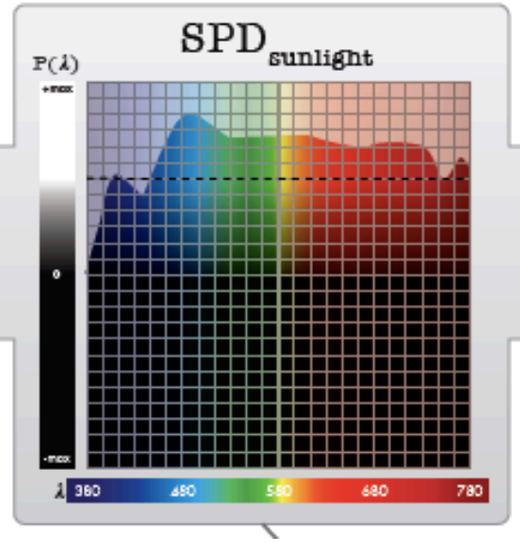


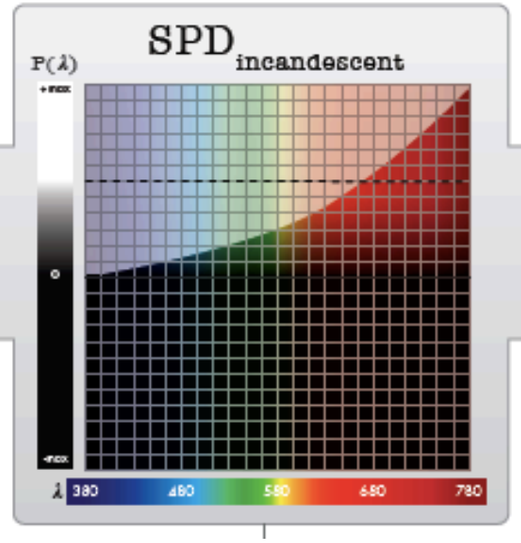


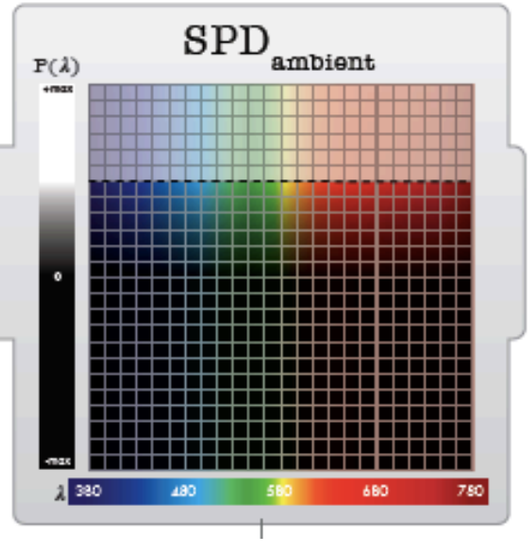
Sample Material



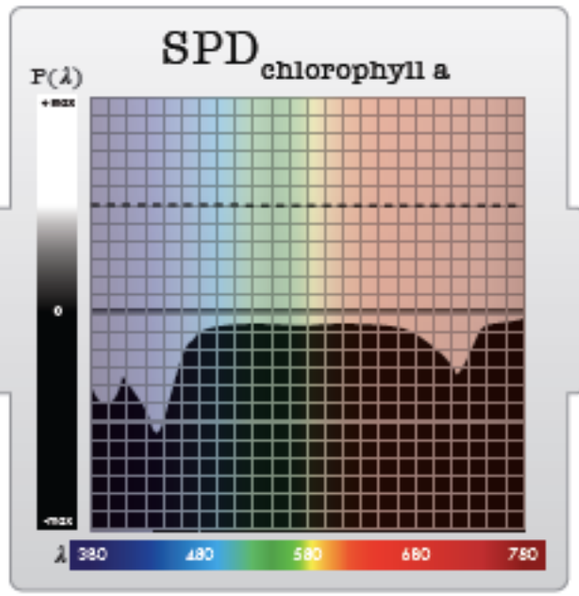
Sample Light Sources



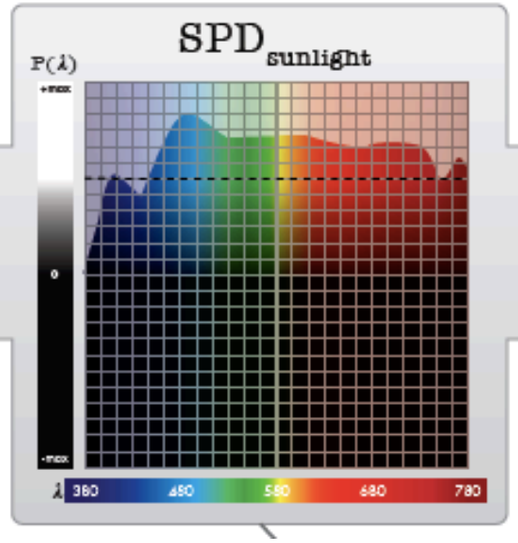


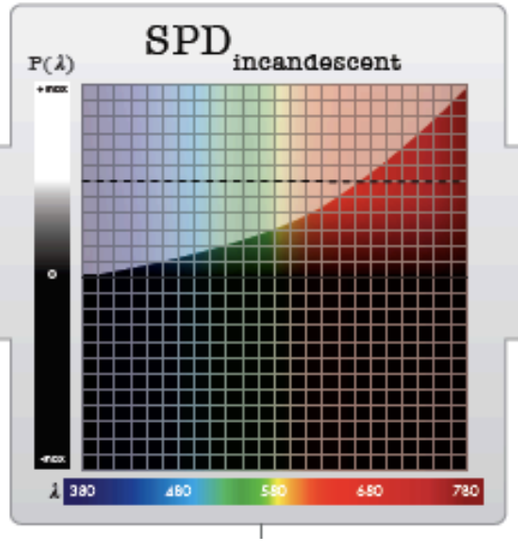


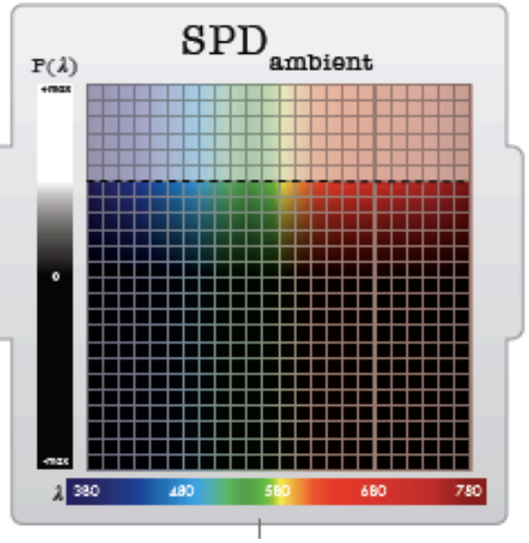
Sample Material



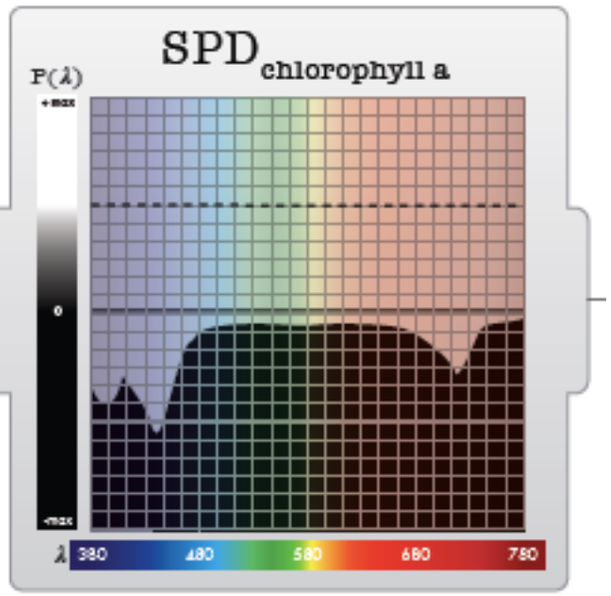
Sample Light Sources



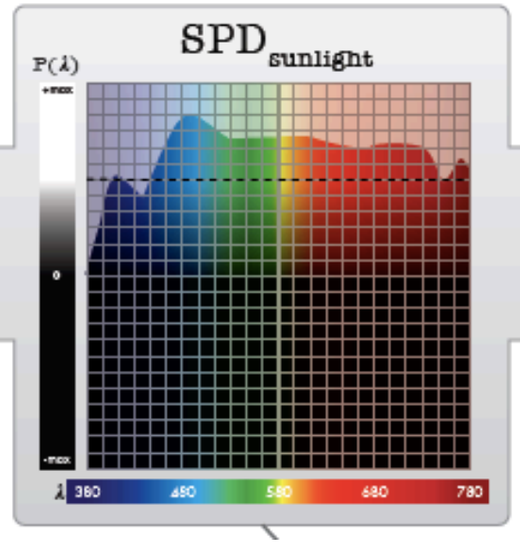


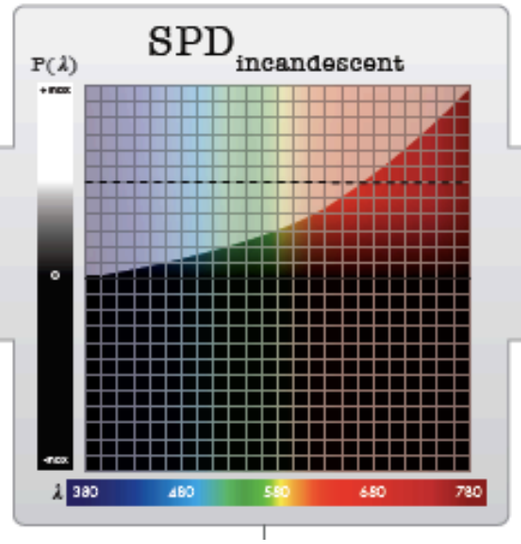


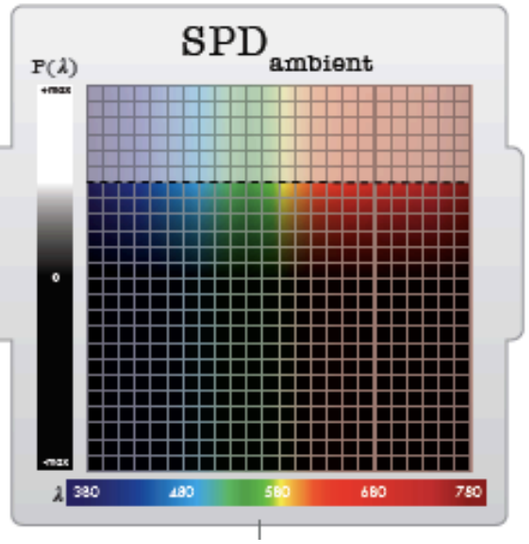
Sample Material



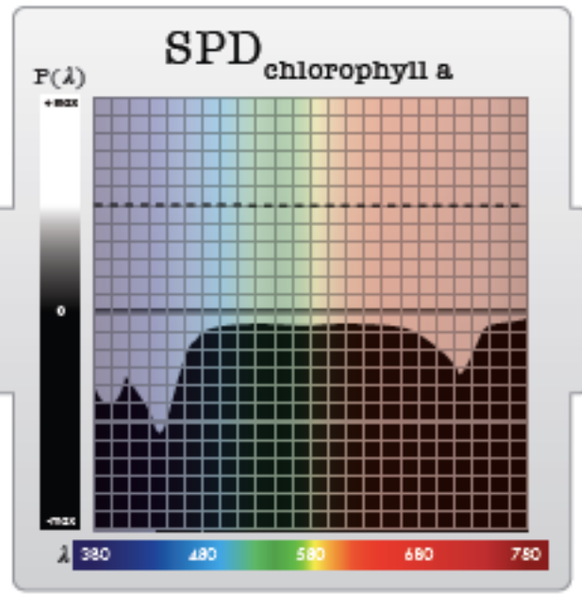
Sample Light Sources





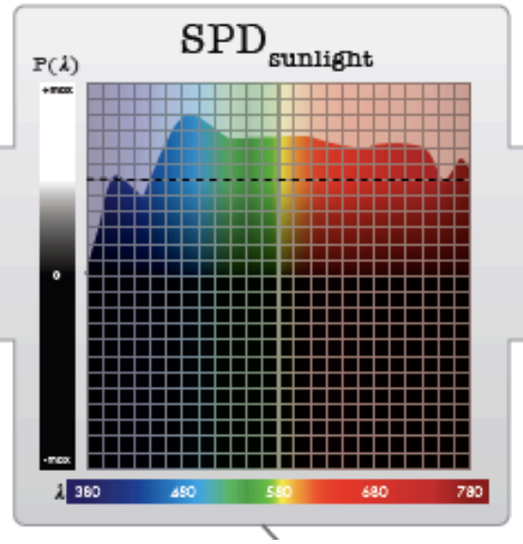


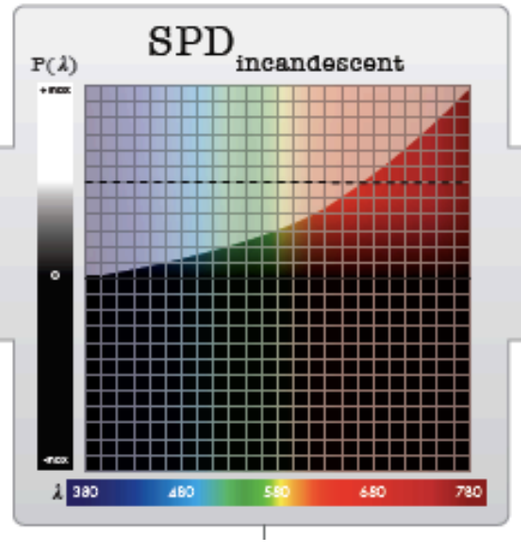
Sample Material

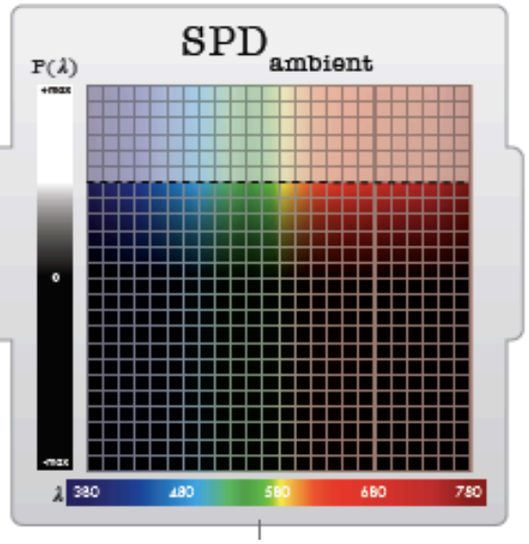


Illumination shader

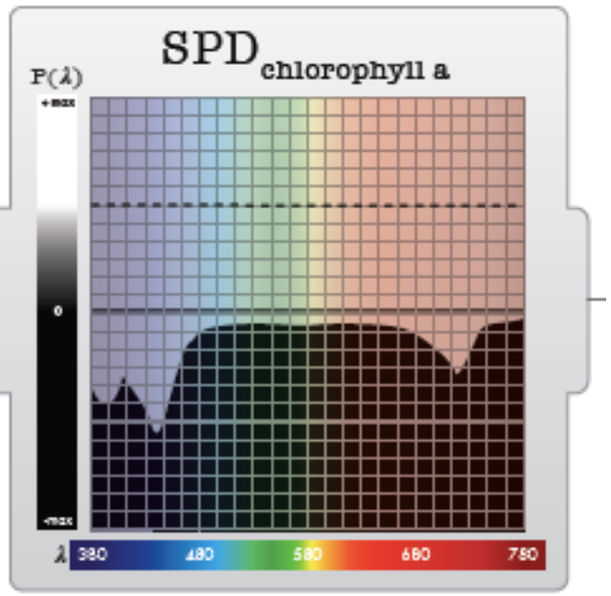
Sample Light Sources



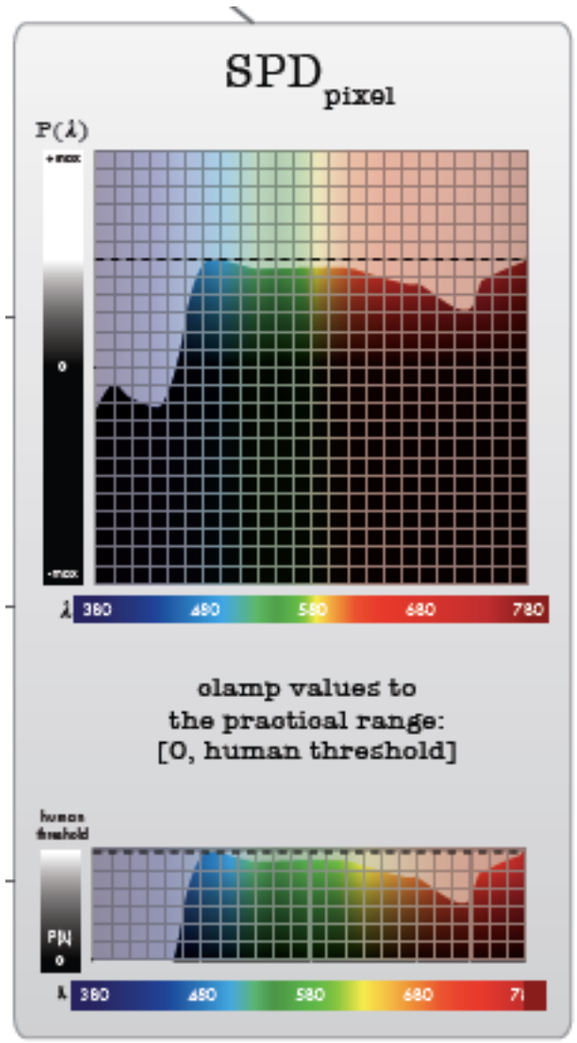


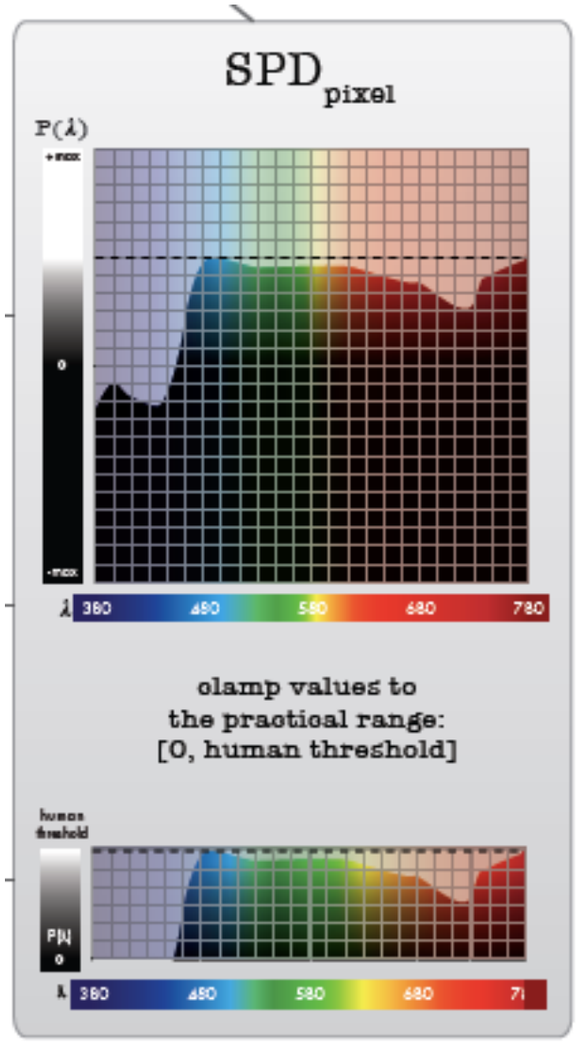


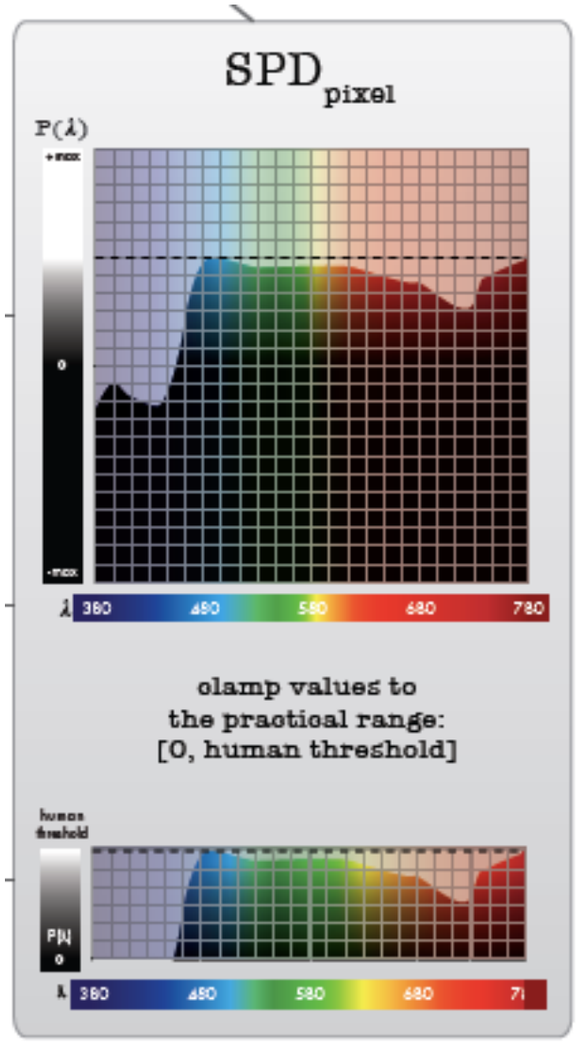
Sample Material

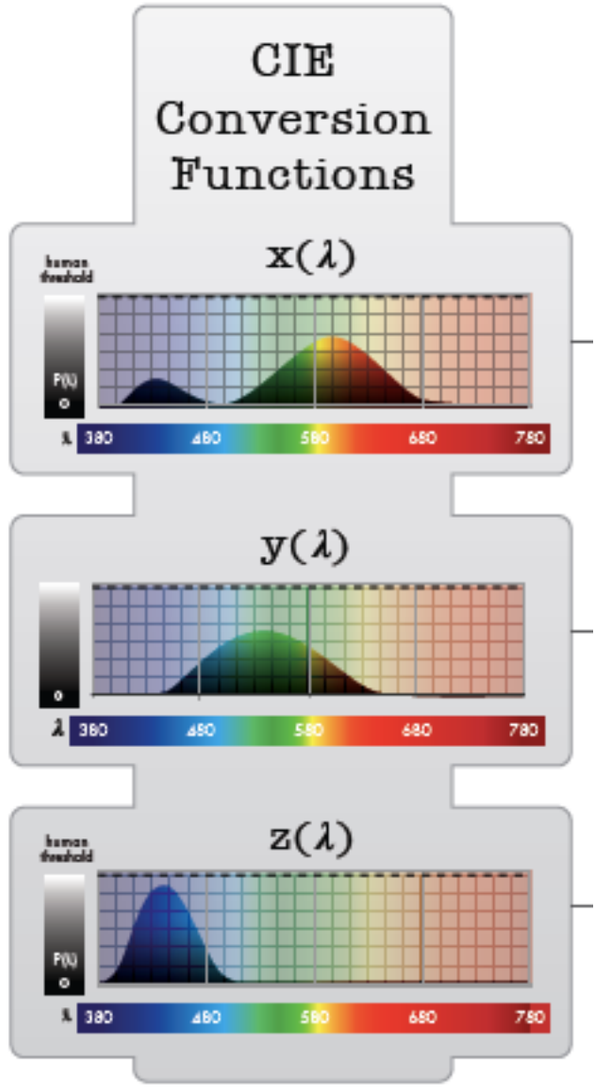


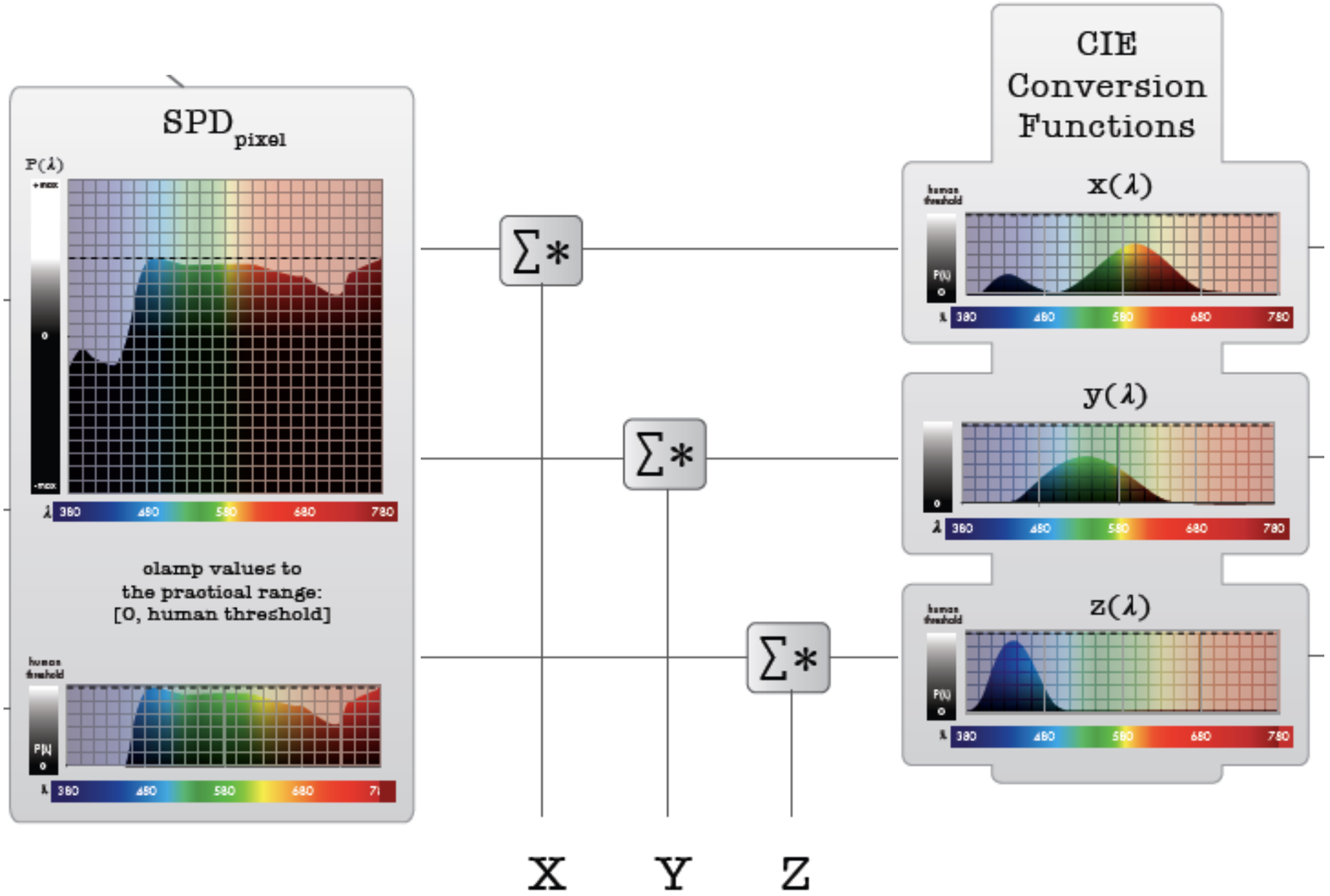
Illumination shader

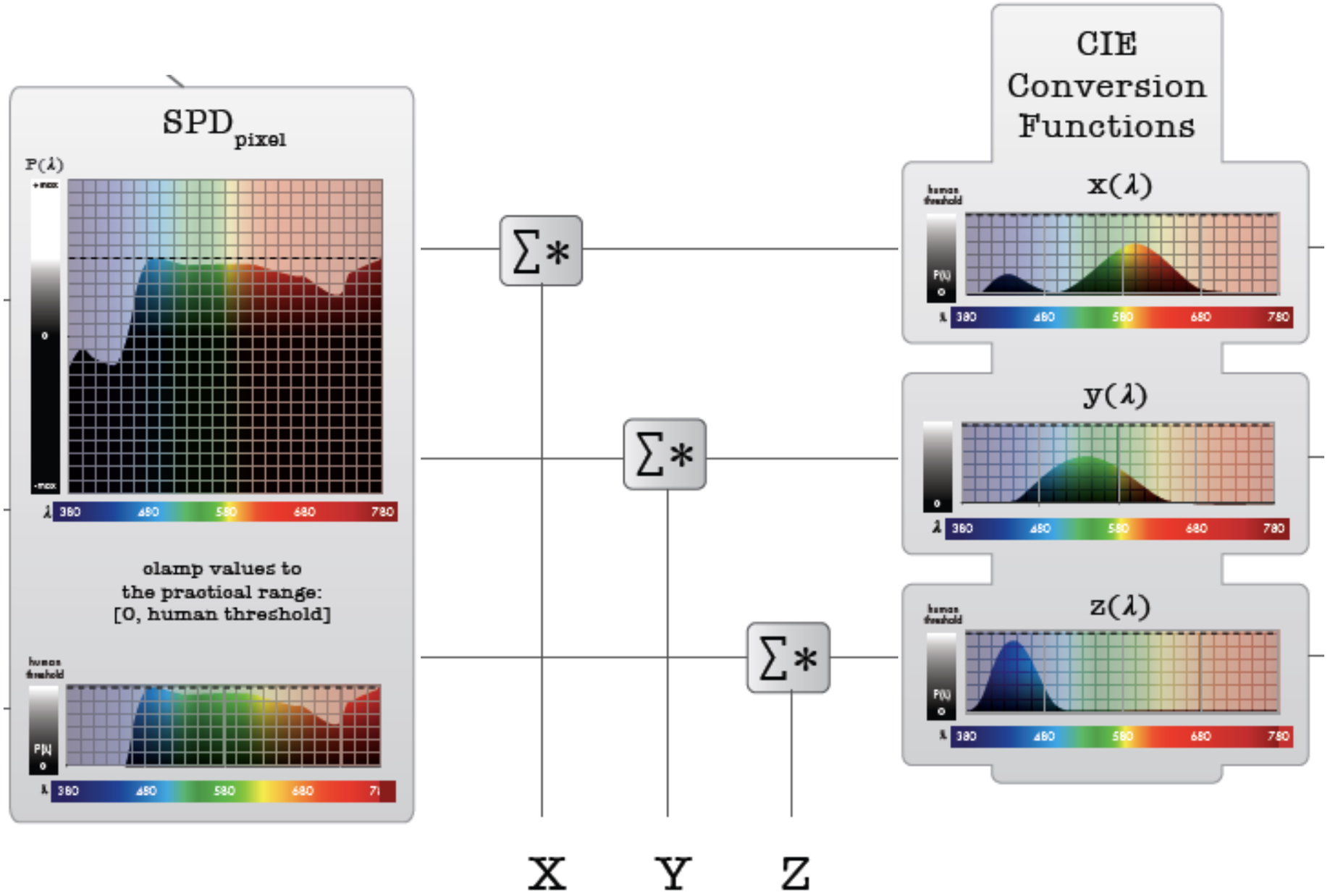



SPD pixel
P(λ)
+max
0
-max
λ 380 480 580 680 780
clamp values to
the practical range:
[0, human threshold]
human threshold
PW
0
λ 380 480 580 680


SPD pixel
P(λ)
+max
0
-max
λ 380 480 580 680 780
clamp values to
the practical range:
[0, human threshold]
human threshold
P(λ)
0
λ 380 480 580 680 7


CIE
Conversion
Functions
x(λ)
human threshold
P(λ)
0
λ 380 480 580 680 780
y(λ)
0
λ 380 480 580 680 780
z(λ)
human threshold
P(λ)
0
λ 380 480 580 680 780

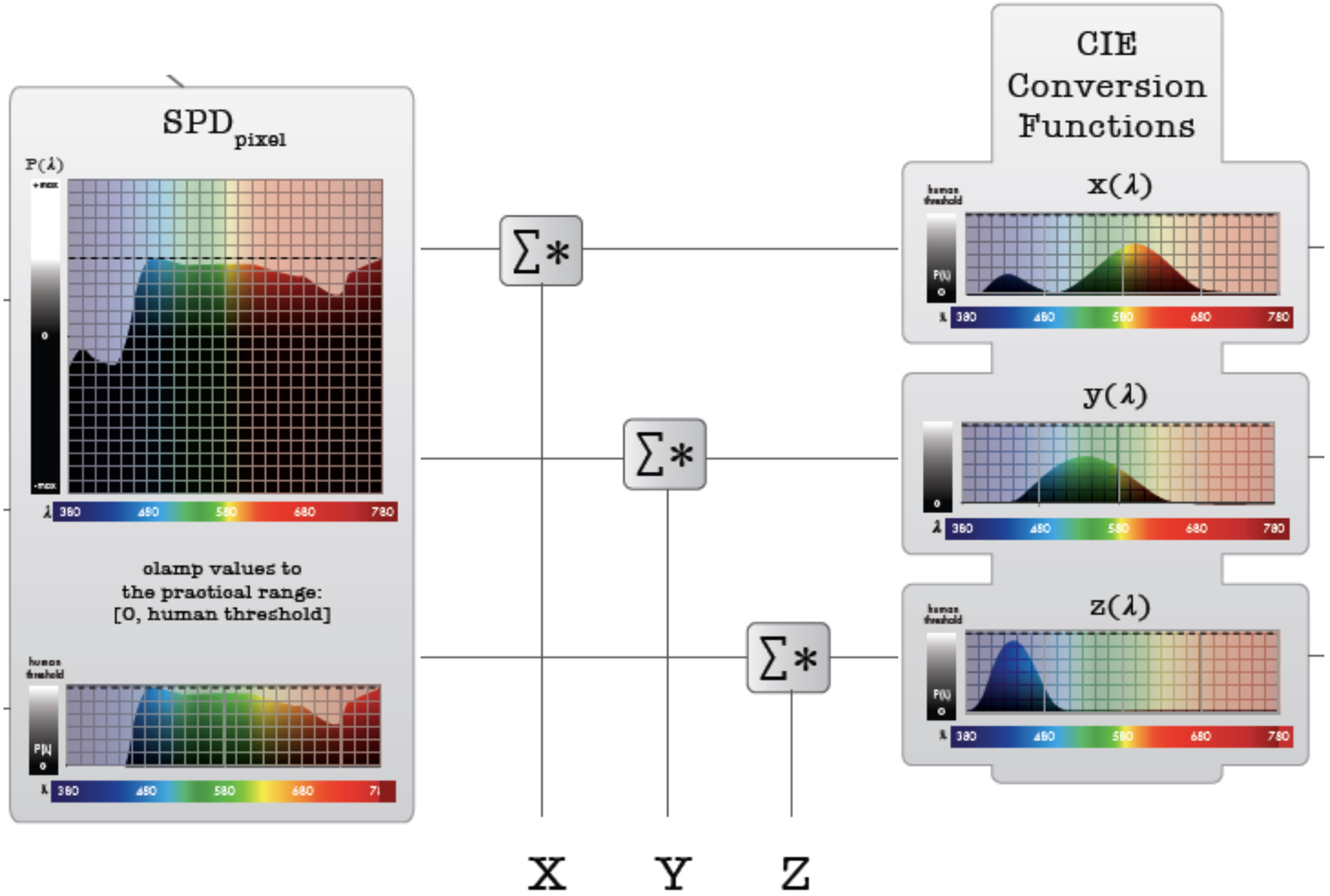

SPD pixel
P(λ)
+max
0
-max
λ 380 480 580 680 780
clamp values to
the practical range:
[0, human threshold]
human threshold
P(λ)
0
λ 380 480 580 680
∑*
∑*
∑*
X
Y
Z
CIE
Conversion
Functions
x(λ)
human threshold
P(λ)
0
λ 380 480 580 680 780
y(λ)
0
λ 380 480 580 680 780
z(λ)
human threshold
P(λ)
0
λ 380 480 580 680 780



$$x = \frac{X}{X + Y + Z}$$

$$y = \frac{Y}{X + Y + Z}$$

$$z = \frac{Z}{X + Y + Z}$$



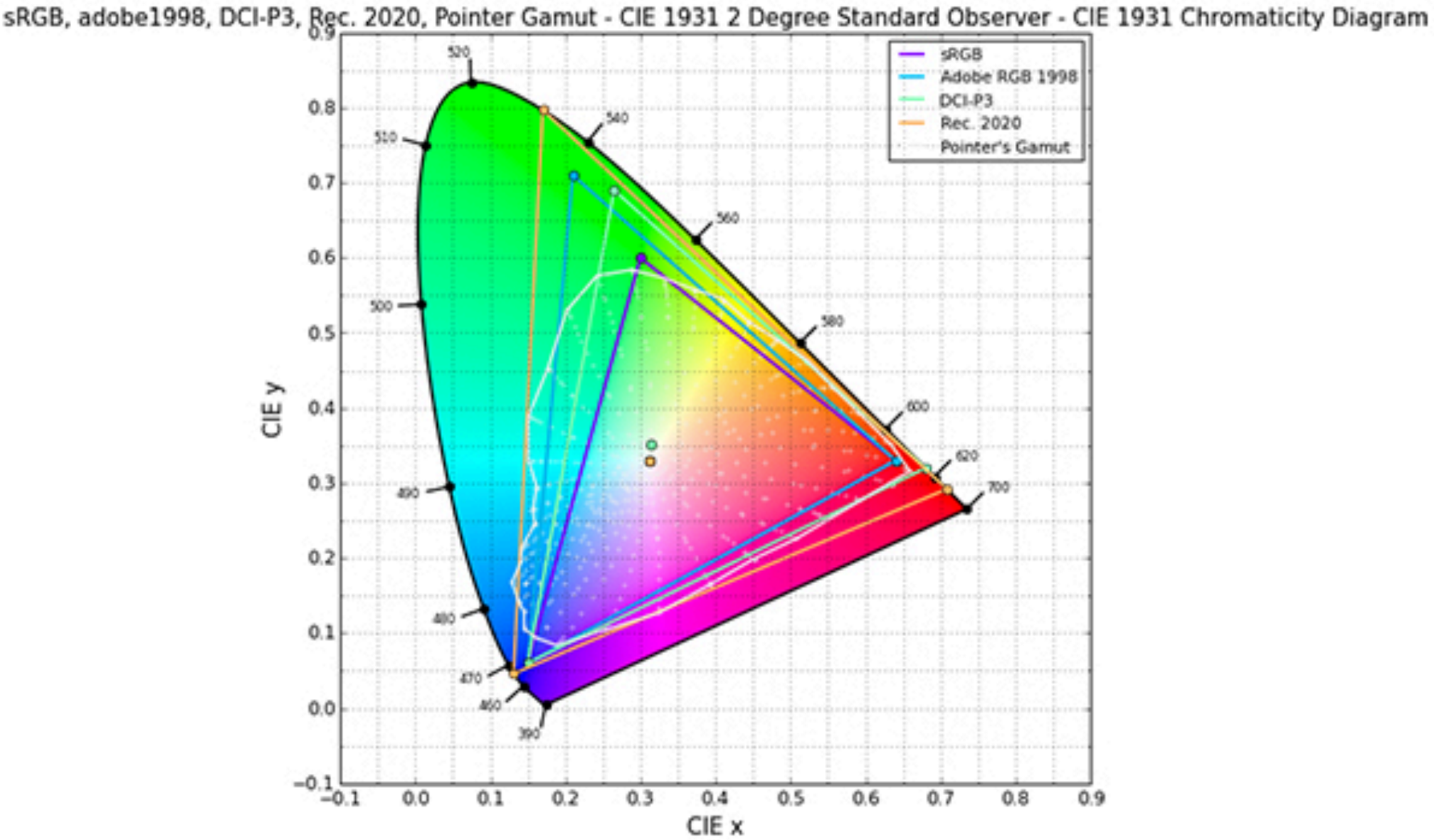
Pixel
RGB values

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

$$z = \frac{Z}{X+Y+Z}$$

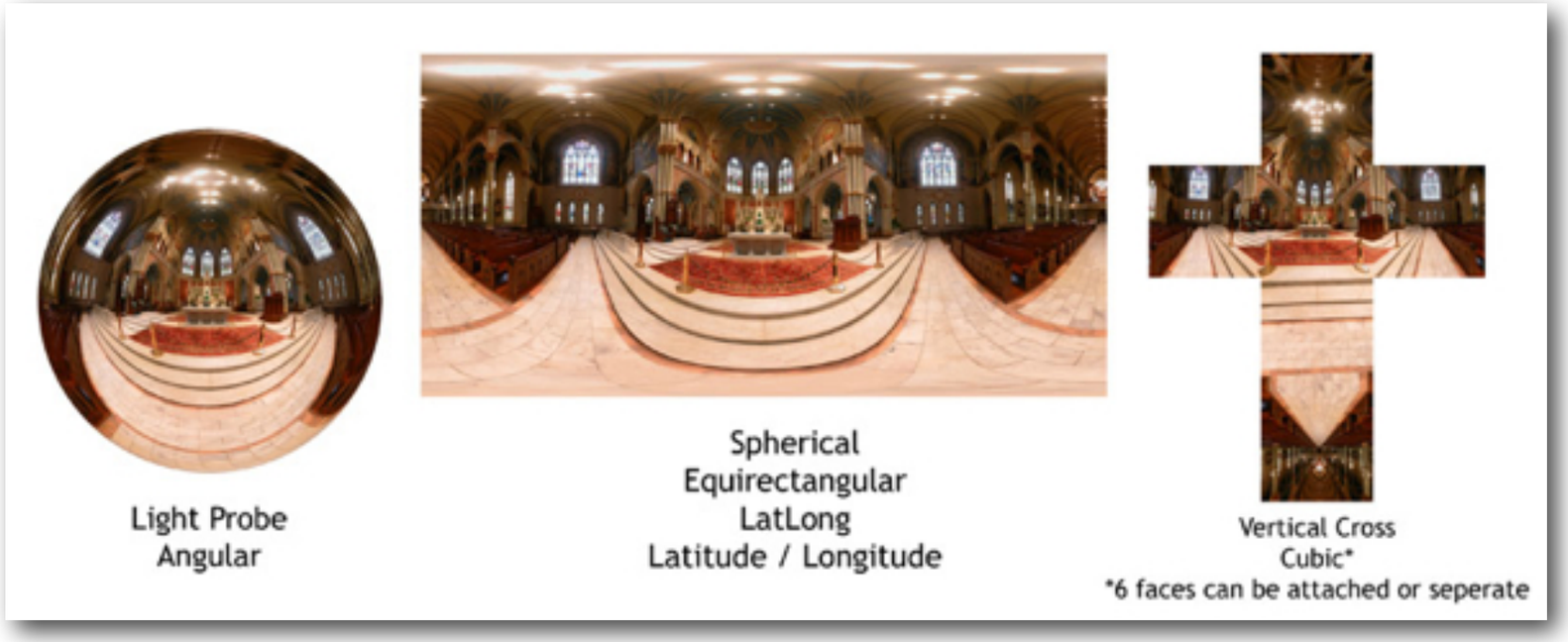
How many colors can we represent?



Pros/Cons of This Approach

- Generates very realistic looking images (in theory)
- Cumbersome to set up
  - How about using a “readymade” lighting information
    - Light probes!!
      - Already in floating point format (*.hdr files)

Using Light Probes as Environment Maps



Steps to implement HDR (Rendering) in OpenGL

Steps to implement HDR (Rendering) in OpenGL

- Create an FBO

Steps to implement HDR (Rendering) in OpenGL

- Create an FBO
- Set the FBO as a render target, render object (mesh) in the center

Steps to implement HDR (Rendering) in OpenGL

- Create an FBO
- Set the FBO as a render target, render object (mesh) in the center
- Use the radiance map (*.hdr/ OpenEXR) files as the environment map texture

Steps to implement HDR (Rendering) in OpenGL

- Create an FBO
- Set the FBO as a render target, render object (mesh) in the center
- Use the radiance map (*.hdr/ OpenEXR) files as the environment map texture
- Texture uses GL_RGBA16F (half float) format (*GL_RGBA32F TOOO slow*)

Steps to implement HDR (Rendering) in OpenGL

- Create an FBO
- Set the FBO as a render target, render object (mesh) in the center
- Use the radiance map (*.hdr/ OpenEXR) files as the environment map texture
  - Texture uses GL_RGBA16F (half float) format (*GL_RGBA32F TOOO slow*)
- Implement reflection, refraction, fresnel, dispersion (sounds familiar?) using the FLOATING POINT environment map

Steps to implement HDR (Rendering) in OpenGL

- Create an FBO
- Set the FBO as a render target, render object (mesh) in the center
- Use the radiance map (*.hdr/ OpenEXR) files as the environment map texture
  - Texture uses GL_RGBA16F (half float) format (*GL_RGBA32F TOOO slow*)
- Implement reflection, refraction, fresnel, dispersion (sounds familiar?) using the FLOATING POINT environment map
- The resulting values are also floating point per fragment

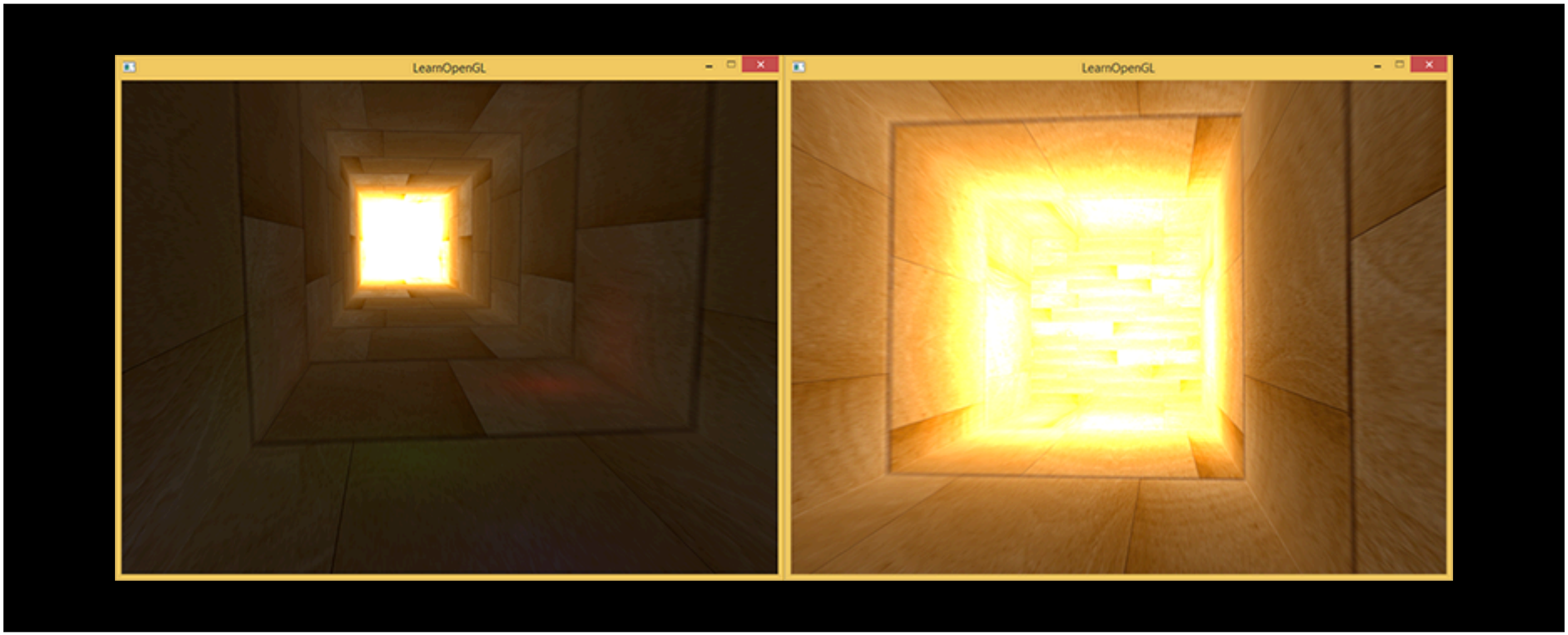
Steps to implement HDR (Rendering) in OpenGL

- Create an FBO
- Set the FBO as a render target, render object (mesh) in the center
- Use the radiance map (*.hdr/ OpenEXR) files as the environment map texture
  - Texture uses GL_RGBA16F (half float) format (*GL_RGBA32F TOOO slow*)
- Implement reflection, refraction, fresnel, dispersion (sounds familiar?) using the FLOATING POINT environment map
- The resulting values are also floating point per fragment
- Are we done? Not yet

Code Example

```
#version 330 core
out vec4 color;
in vec2 TexCoords;

// Output generated using FBO,
// now used as lookup texture

uniform sampler2D hdrBuffer;

void main()
{
    vec3 hdrColor = texture(hdrBuffer, TexCoords).rgb;
    color = vec4(hdrColor, 1.0);
}
```

Sample output



4 point lights
L1= (200.0, 200.0, 200.0)
L2=(1.0, 0.0, 0.0)
L3=(0.0, 0.0, 0.2)
L4=(0.0, 0.1, 0.0)

HDR --> LDR aka Tone Mapping

- Our render target contains HDR values (contrast ratio > 256:1)
- Eventually, we need to display the current frame on screen (256:1)
- Convert HDR values to LDR values using a Tone Mapping Operator

$$L = \frac{Y}{(Y+1)}$$

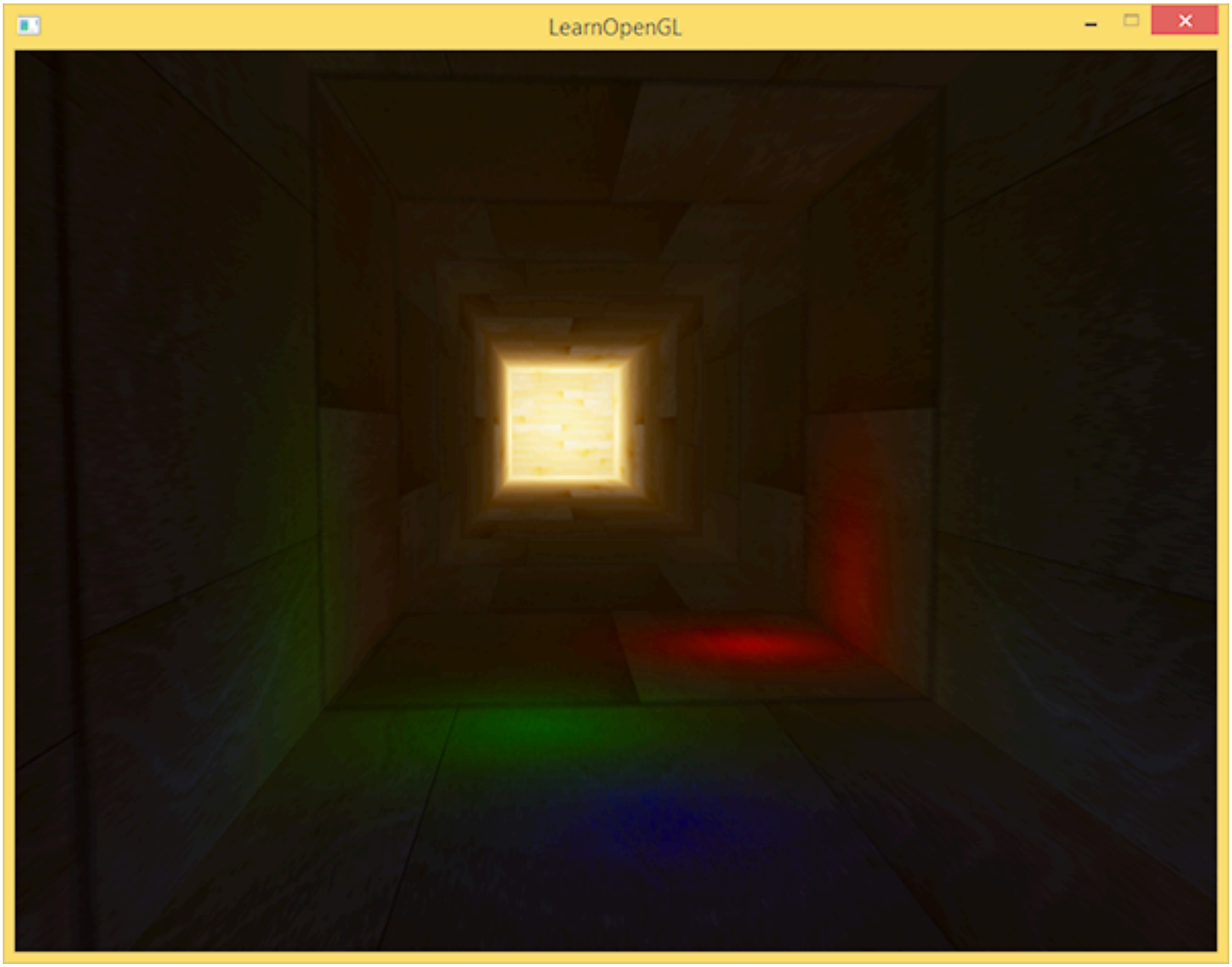
This operator will map value from [0; inf) to [0;1)

http://www.mpi-inf.mpg.de/resources/tmo/

Code Version 2.0

```
#version 330 core
out vec4 color;
in vec2 TexCoords;

void main()
{
    const float gamma = 2.2;
    vec3 hdrColor = texture(hdrBuffer, TexCoords).rgb;

    // Reinhard tone mapping
    vec3 mapped = hdrColor / (hdrColor + vec3(1.0));

    // Gamma correction
    mapped = pow(mapped, vec3(1.0 / gamma));

    color = vec4(mapped, 1.0);
}
```

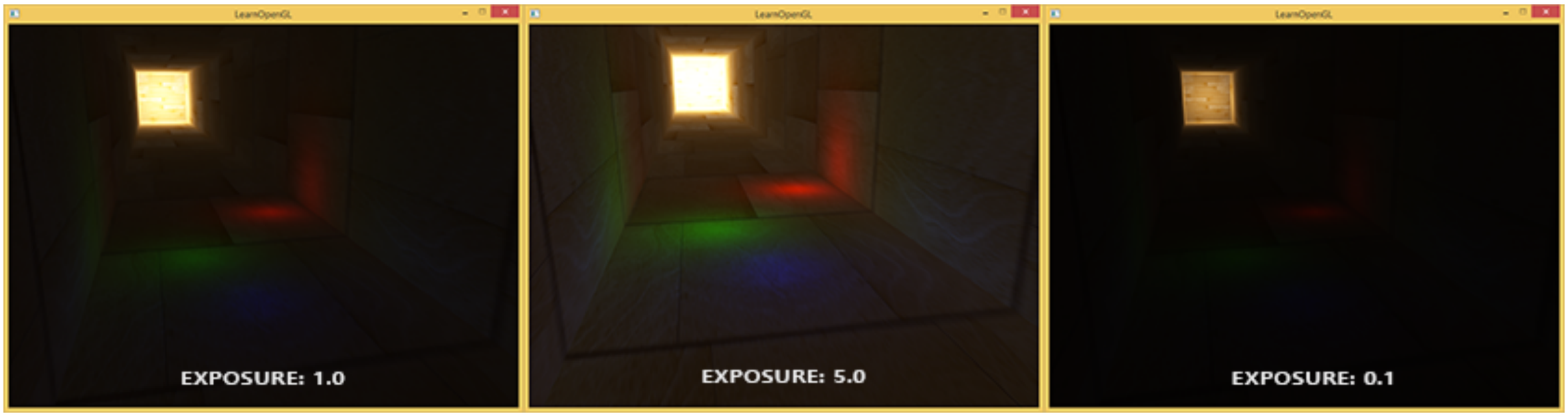
Output



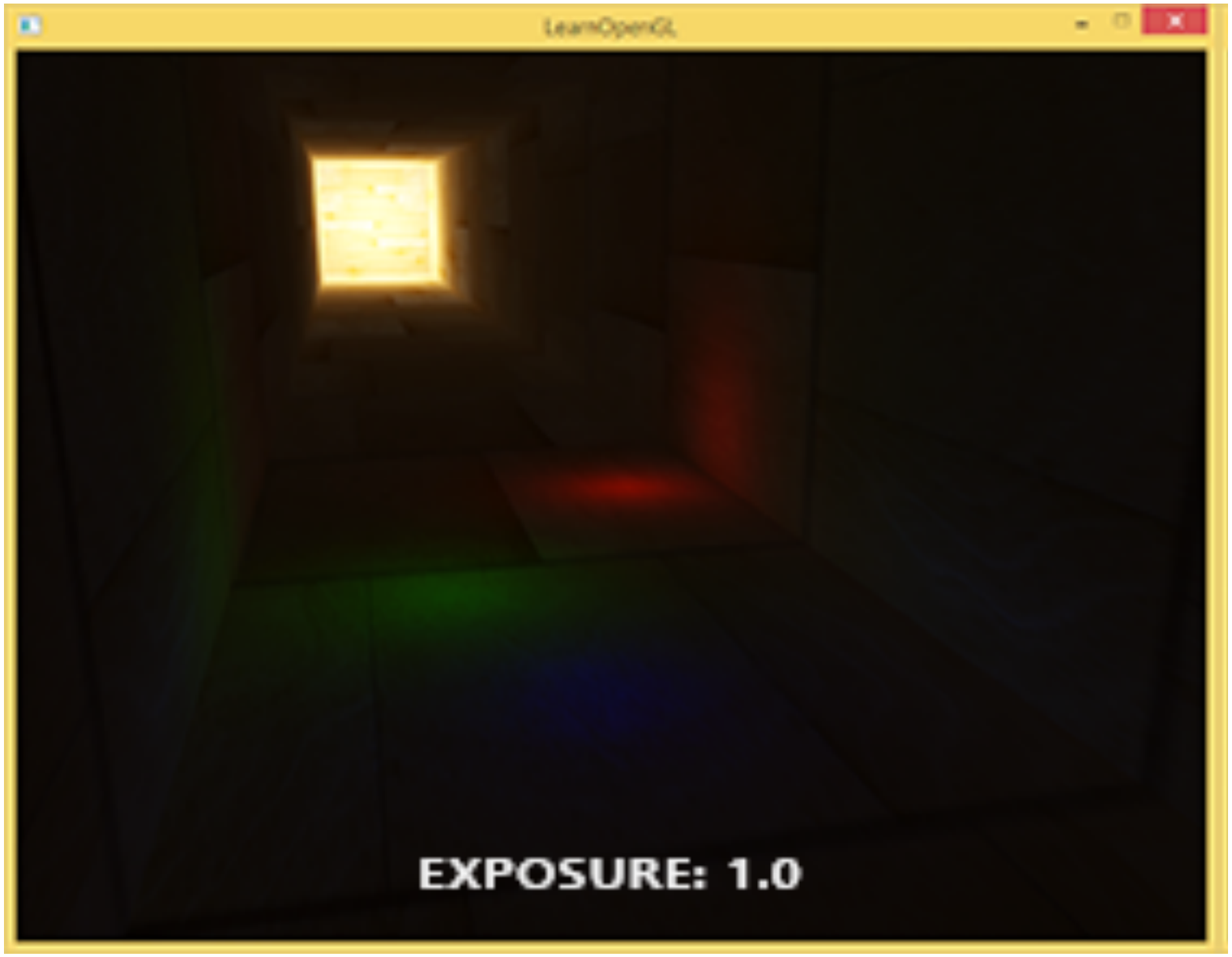
Adding exposure

```
#version 330 core
out vec4 color;
in vec2 TexCoords;
uniform float exposure
void main()
{
    const float gamma = 2.2;
    vec3 hdrColor = texture(hdrBuffer, TexCoords).rgb;

    // Exposure-based tone mapping
    vec3 mapped = vec3(1.0) - exp( -hdrColor * exposure );

    // Gamma correction
    mapped = pow(mapped, vec3(1.0 / gamma));

    color = vec4(mapped, 1.0);
}
```

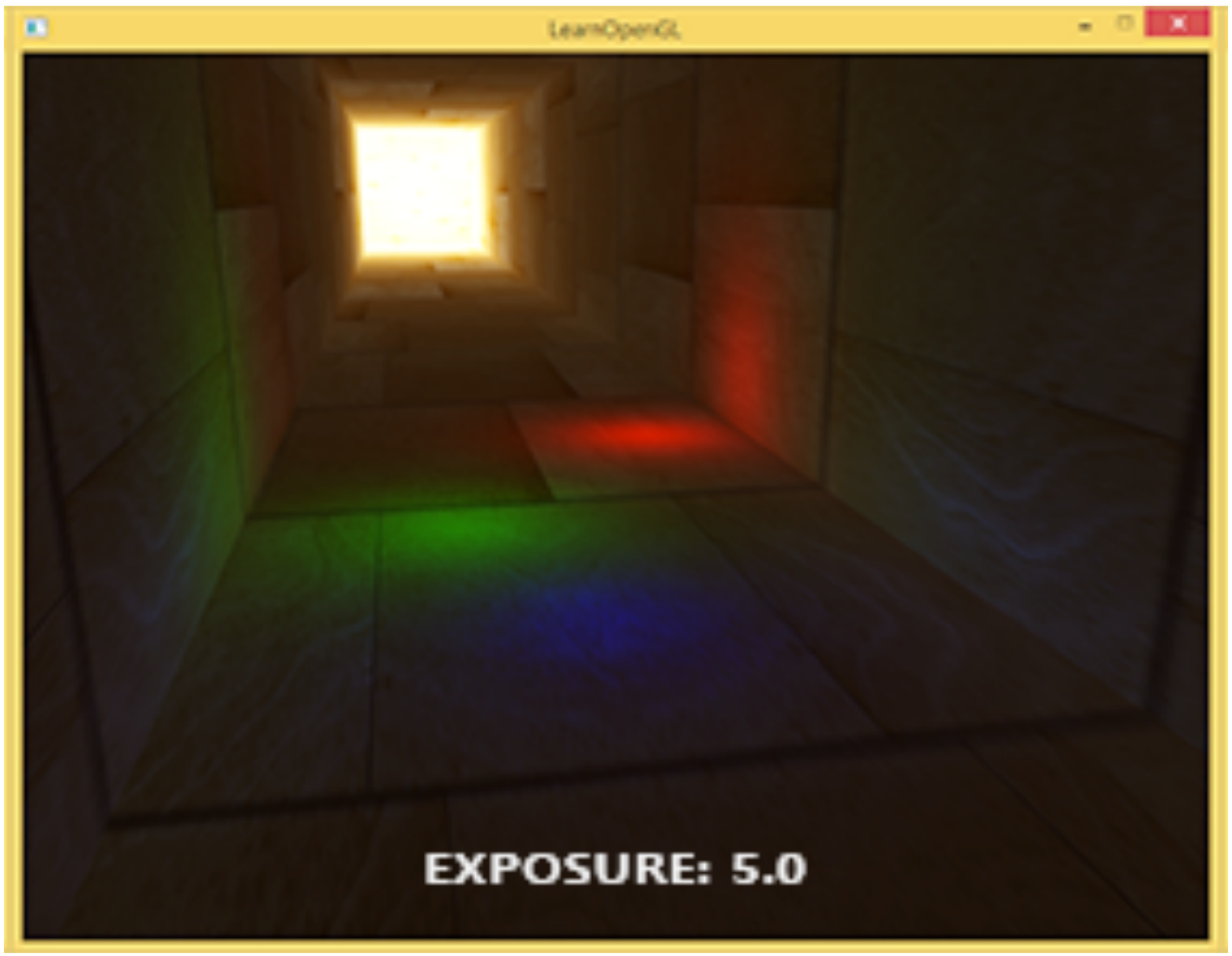
Output


LearnOpenGL
EXPOSURE: 1.0
LearnOpenGL
EXPOSURE: 5.0
LearnOpenGL
EXPOSURE: 0.1

Exposure = 1.0



Exposure = 5.0 (overexposed)



Exposure = 0.1 (underexposed)

More “Optimizations” aka Hacks

- A scene with values between [0,256) looks very flat and dull

More “Optimizations” aka Hacks

- A scene with values between [0,256) looks very flat and dull

More “Optimizations” aka Hacks

- A scene with values between [0,256) looks very flat and dull

More “Optimizations” aka Hacks

- A scene with values between [0,256) looks very flat and dull

Images from Farcry

More “Optimizations” aka Hacks

- A scene with values between [0,256) looks very flat and dull

Bloom Effect!!!

Images from Farcry

Implementing a Smart Bloom

- Extract bright areas (threshold) from the HDR buffer
- Downsample and filter with a median filter
- Add (superimpose, in signal-speak) the blurred HDR buffers on the original

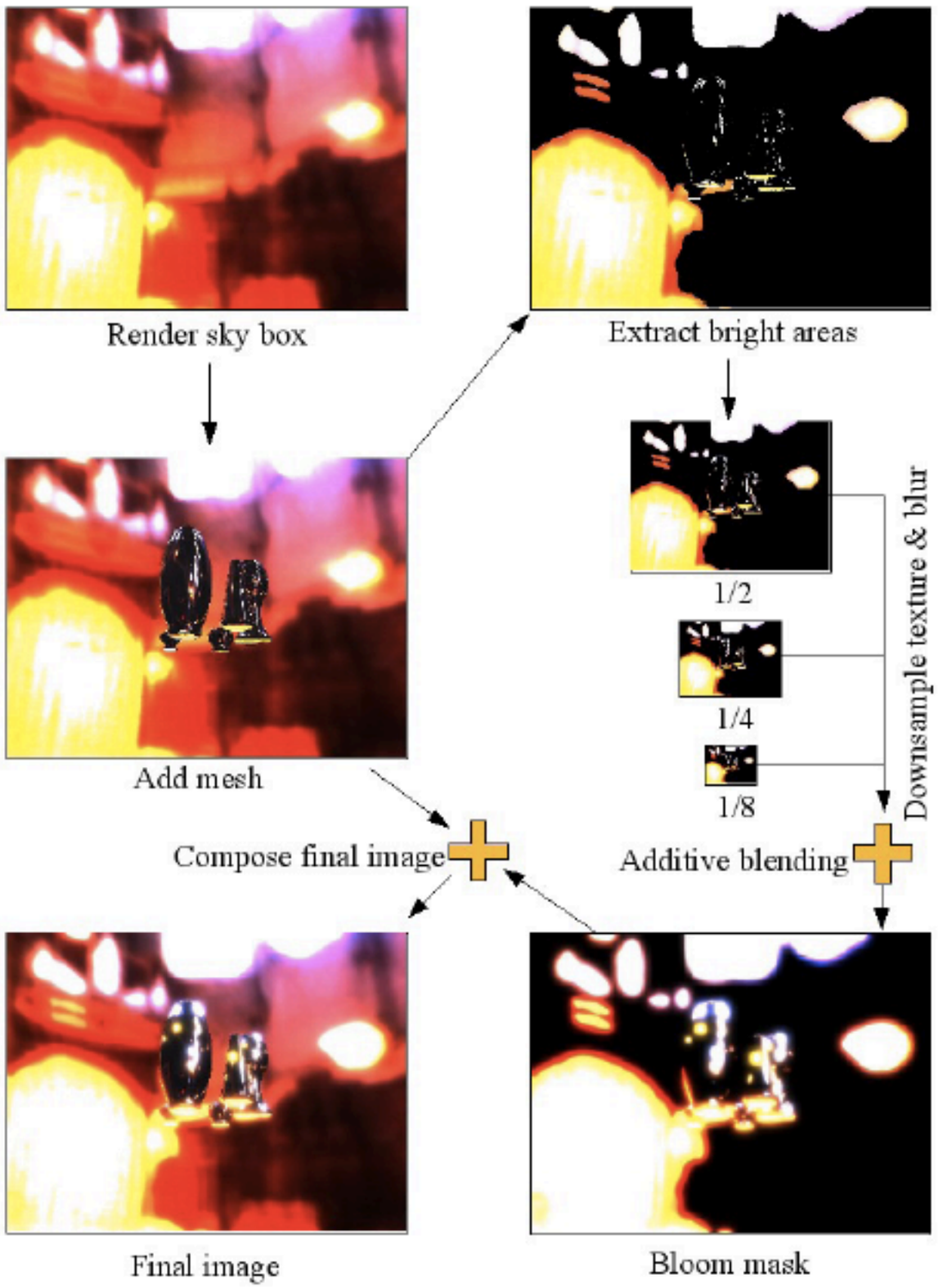
- Apply tone-mapping to this new blurred/bloomed buffer

$$Y = exposure \times \left(\frac{\left(\frac{exposure}{maxLuminance} + 1 \right)}{(exposure + 1)} \right)$$

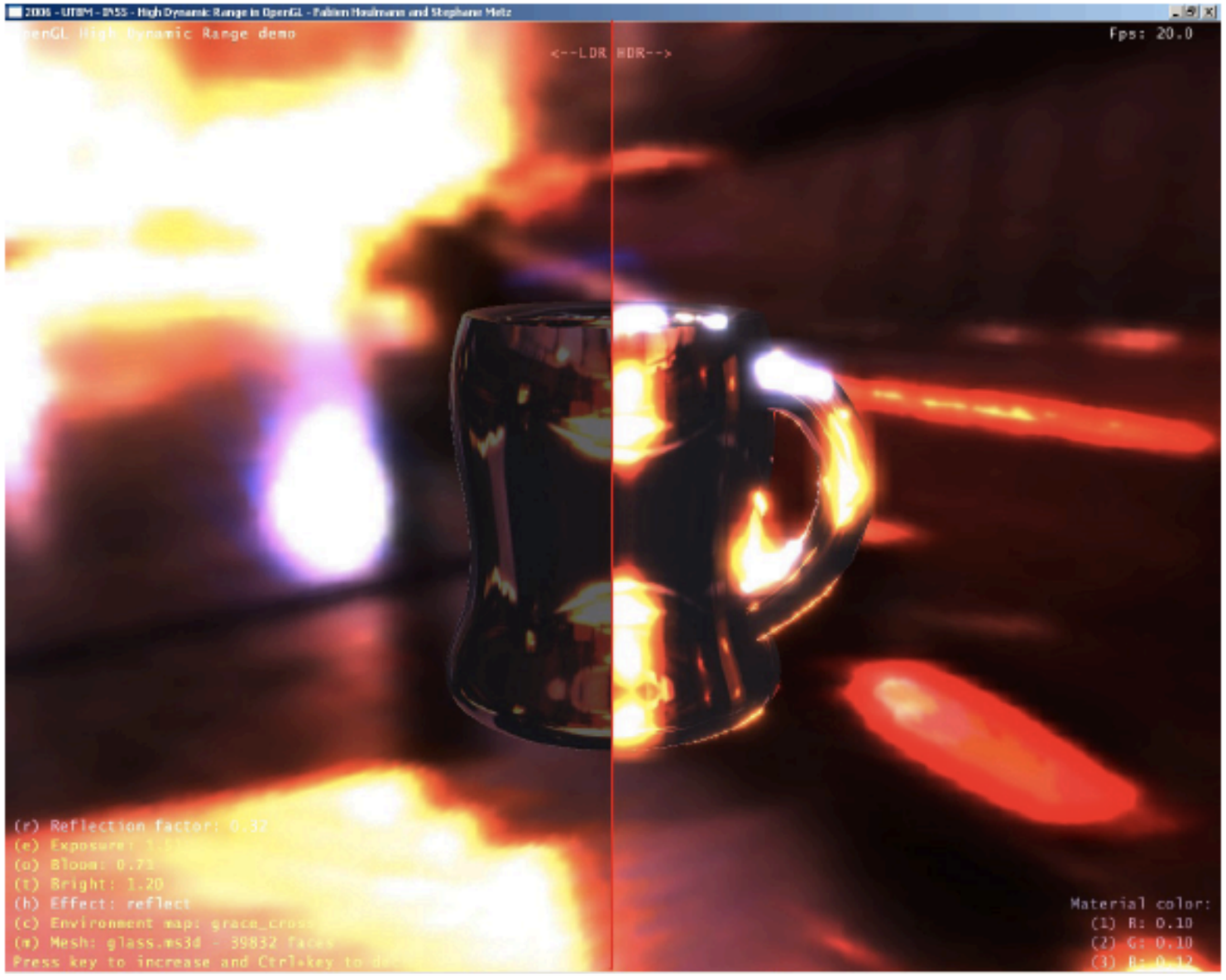
$$Y'_{601} = 0.30R + 0.59G + 0.11B$$

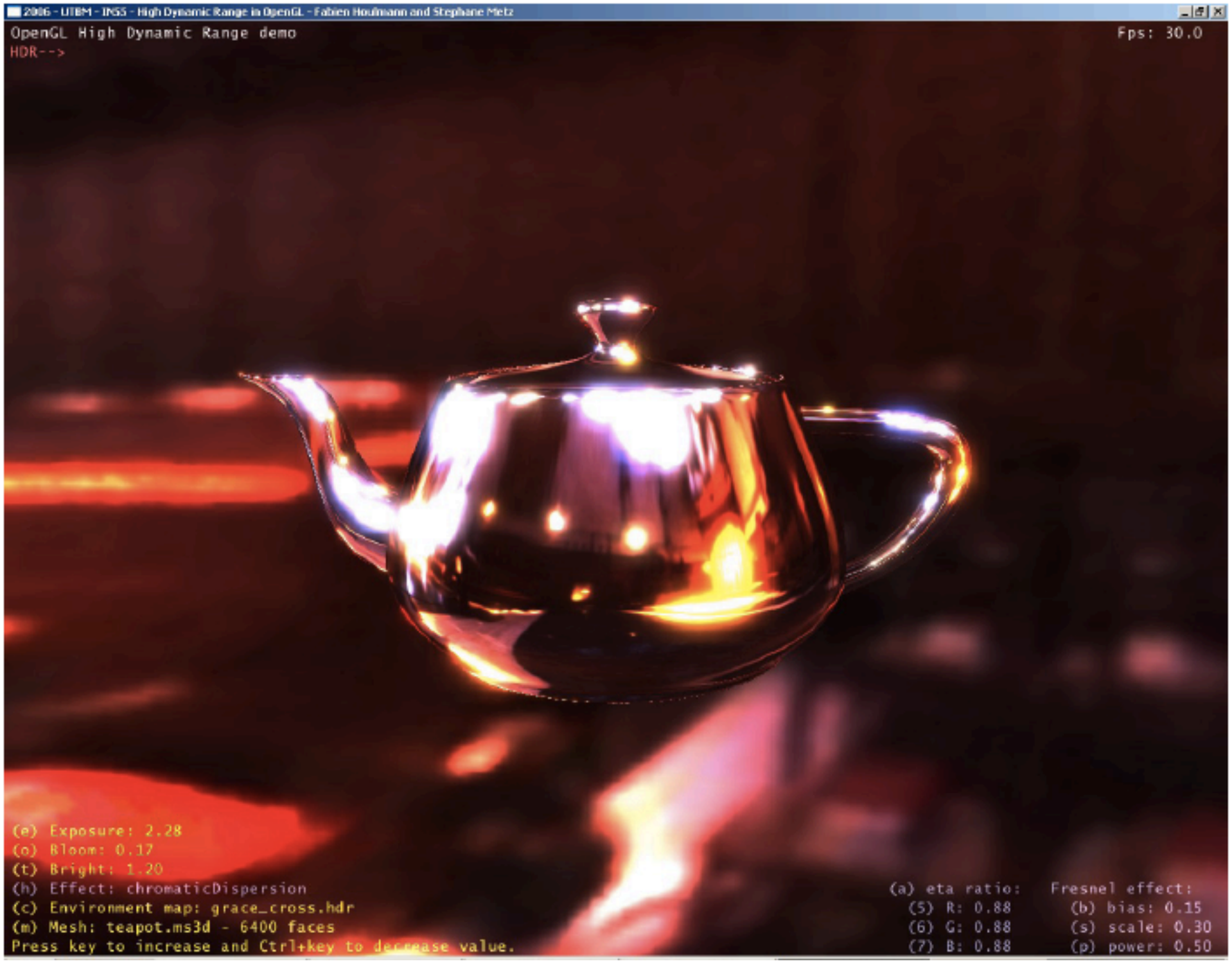
In GLSL

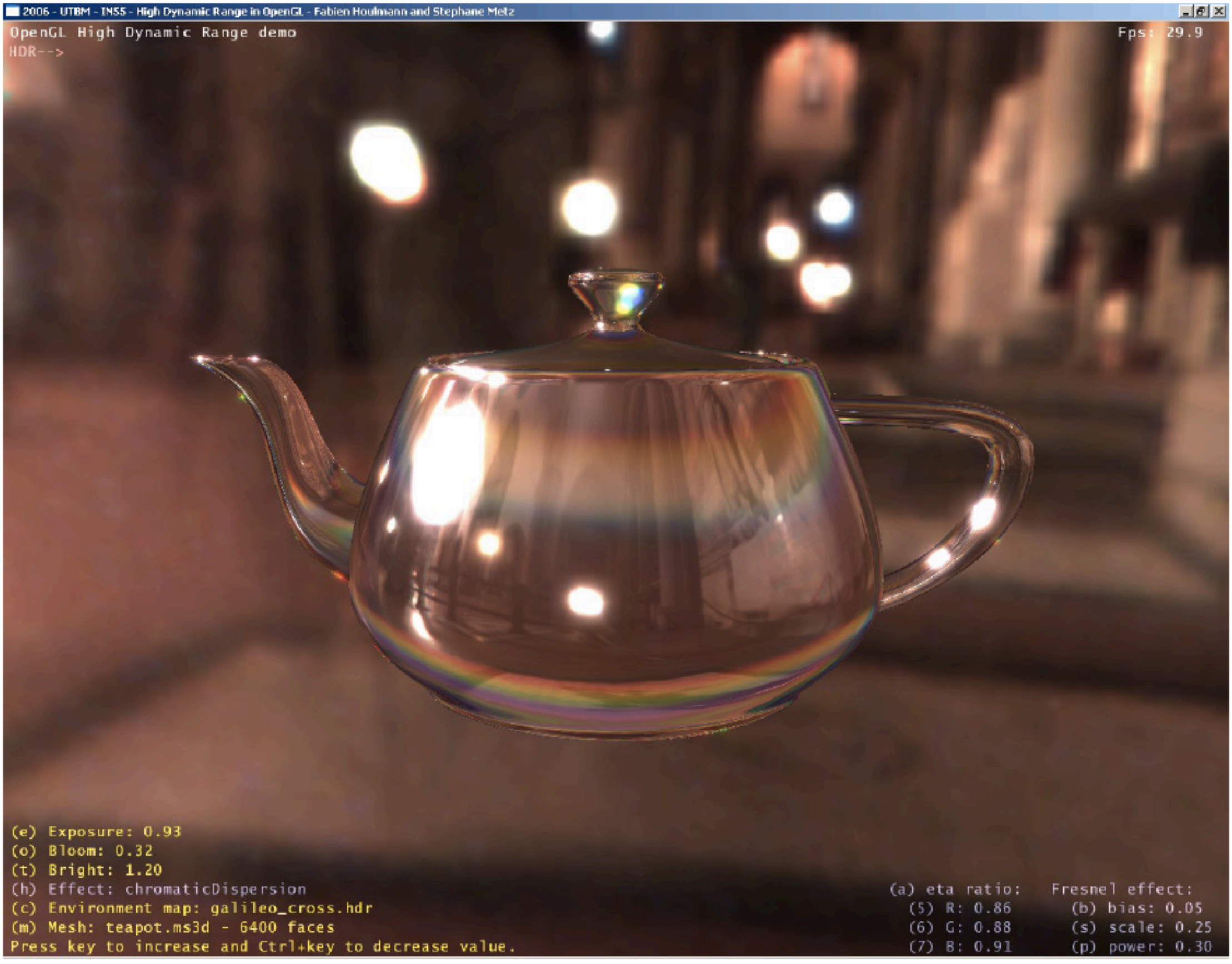
```
// render texture and bloom map
uniform sampler2D tex, bloom;
// Control exposure with this value
uniform float exposure;
// How much bloom to add
uniform float bloomFactor;
// Max bright
uniform float brightMax;

void main()
{
	vec2 st = gl_TexCoord[0].st;
	vec4 color = texture2D(tex, st);
	vec4 colorBloom = texture2D(bloom, st);

	// Add bloom to the image
	color += colorBloom * bloomFactor;

	// Perform tone-mapping
	float Y = dot(vec4(0.30, 0.59, 0.11, 0.0), color);
	float YD = exposure * (exposure/brightMax + 1.0) / (exposure + 1.0);
	color *= YD;

	gl_FragColor = color;
}
```


Render sky box
Extract bright areas
1/2
1/4
1/8
Downsample texture & blur
Add mesh
Compose final image
Additive blending
Final image
Bloom mask


High Dynamic Range demo
Fps: 23.0
<--LDR HDR-->
(e) Exposure: 1.09
(b) Bloom: 0.37
(t) Bright: 1.20
(h) Effect: chromaticDispersion
(c) Environment map: rnl_cross.hdr
(m) Mesh: car2.ms3d - 44000 faces
Press key to increase and Ctrl+key to decrease value.
(a) eta ratio:
(1) R: 0.80
(6) G: 0.80
(7) B: 0.80
Fresnel effect:
(b) bias: 0.20
(s) scale: 1.00
(p) power: 1.00


2006 - UTBM - INSS - High Dynamic Range in OpenGL - Fabien Houlmann and Stephane Metz
OpenGL High Dynamic Range demo
<--LDR HDR-->
Fps: 20.0
(r) Reflection factor: 0.32
(e) Exposure: 1.5
(o) Bloom: 0.71
(t) Bright: 1.20
(h) Effect: reflect
(c) Environment map: grace_cross
(m) Mesh: glass.ms3d - 39832 faces
Press key to increase and Ctrl+key to de
Material color:
(1) R: 0.10
(2) G: 0.10
(3) B: 0.12


2006 - LITBM - INSS - High Dynamic Range in OpenGL - Fabien Houlmann and Stephane Metz
OpenGL High Dynamic Range demo
HDR-->
Fps: 30.0
(e) Exposure: 2.28
(o) Bloom: 0.17
(t) Bright: 1.20
(h) Effect: chromaticDispersion
(c) Environment map: grace_cross.hdr
(m) Mesh: teapot.ms3d - 6400 faces
Press key to increase and Ctrl+key to decrease value.
(a) eta ratio:
(5) R: 0.88
(6) G: 0.88
(7) B: 0.88
Fresnel effect:
(b) bias: 0.15
(s) scale: 0.30
(p) power: 0.50


2006 - UTBM - IN55 - High Dynamic Range in OpenGL - Fabien Houlmann and Stephane Metz
OpenGL High Dynamic Range demo
Fps: 29.9
HDR-->
(e) Exposure: 0.93
(o) Bloom: 0.32
(t) Bright: 1.20
(h) Effect: chromaticDispersion
(c) Environment map: galileo_cross.hdr
(m) Mesh: teapot.ms3d - 6400 faces
Press key to increase and Ctrl+key to decrease value.
(a) eta ratio:
(5) R: 0.86
(6) G: 0.88
(7) B: 0.91
Fresnel effect:
(b) bias: 0.05
(s) scale: 0.25
(p) power: 0.30


Filmic tone mapping
No tone mapping
NAUGHTY DOG

References

- https://learnopengl.com/#!Advanced-Lighting/HDR
- “High Dynamic Range Rendering in OpenGL,” Fabian Houlmann & Stephane Metz
- Digital Image Processing, 3rd Edition, Gonzales & Woods
- www.hdrlabs.com
- Paul Debevec, USC (http://www.pauldebevec.com/)
  - Awarded Oscar for his work in relighting for “Avatar”
- SPD from Robert Campbell Farish, Digipen MSCS student