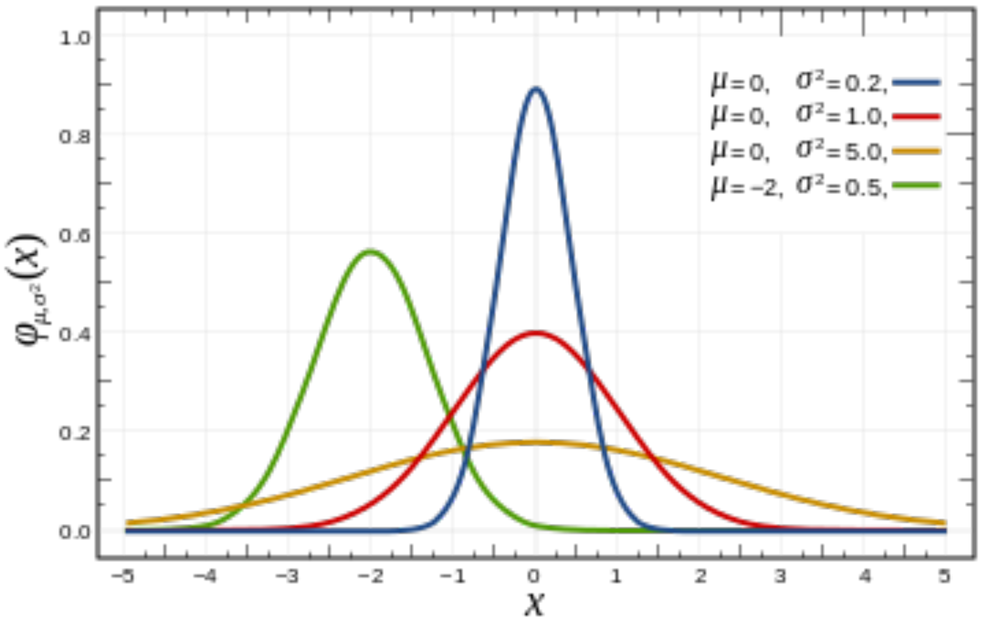
Color and post-processing.

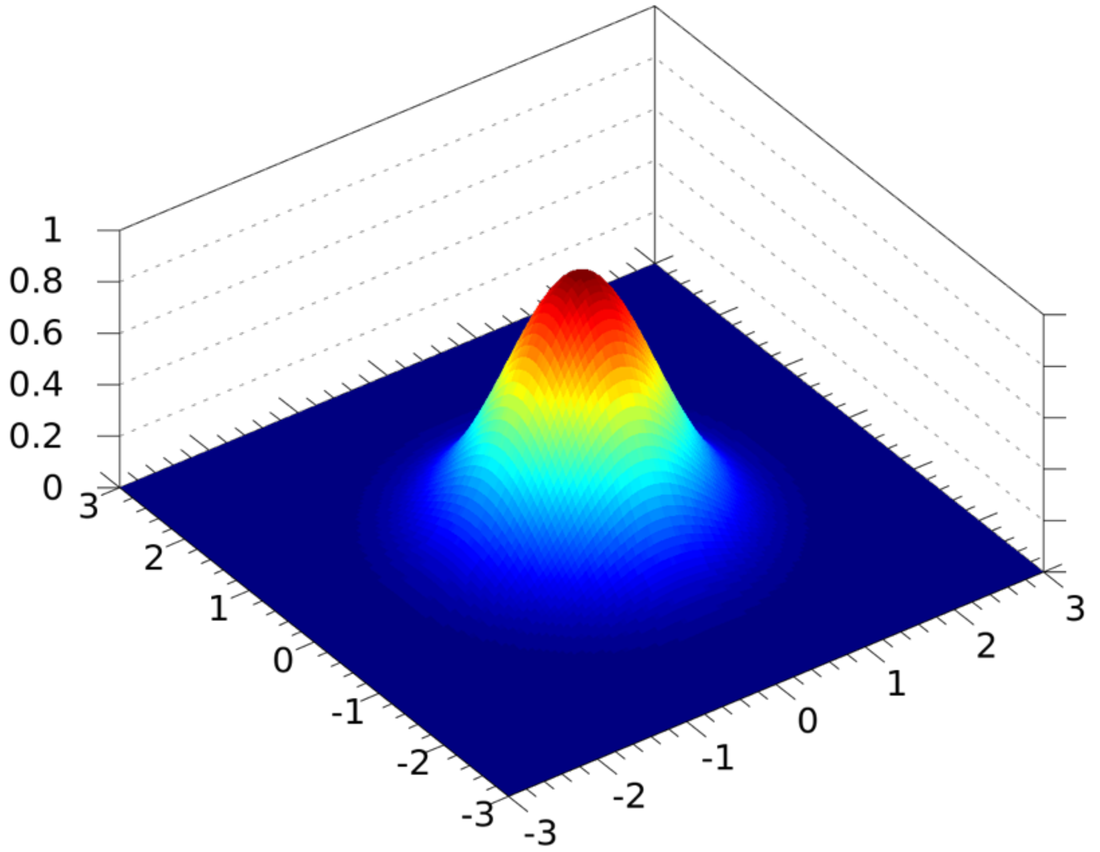
CS GY-6533 / UY-4533

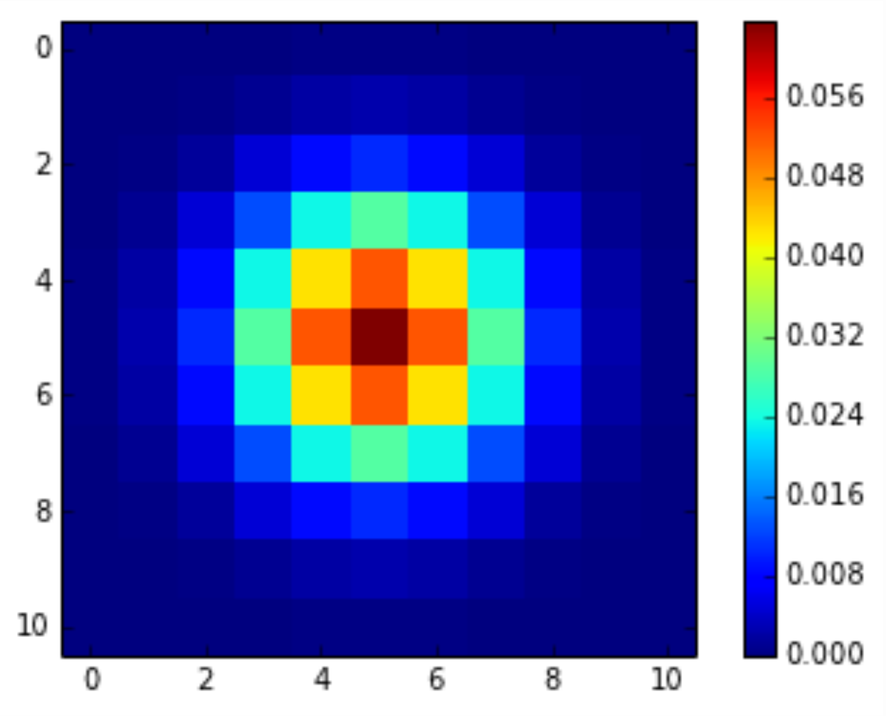
Review: post-processing shaders.

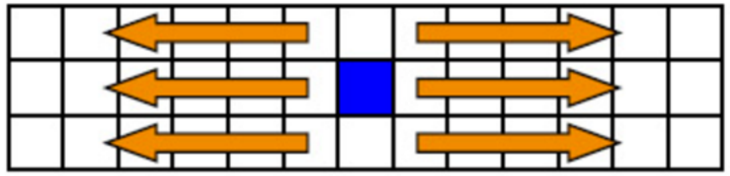
Multi-step post-processing.

Example: Gaussian blur

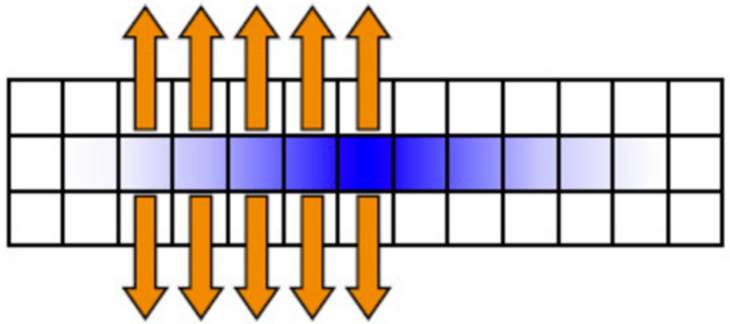

$\mu=0, \quad \sigma^2=0.2,$
$\mu=0, \quad \sigma^2=1.0,$
$\mu=0, \quad \sigma^2=5.0,$
$\mu=-2, \quad \sigma^2=0.5,$
$\varphi_{\mu,\sigma^2}(x)$
x



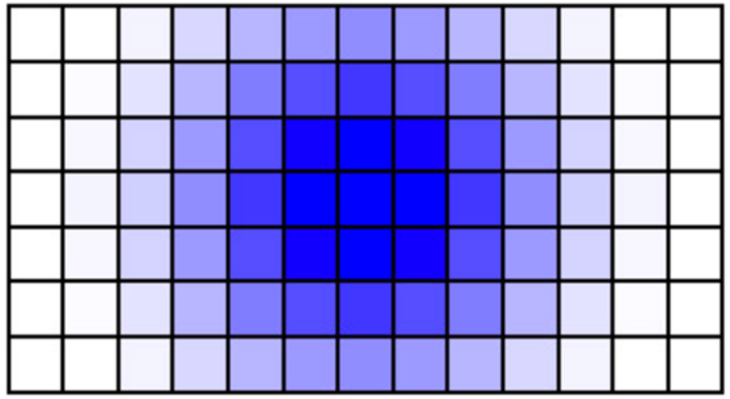




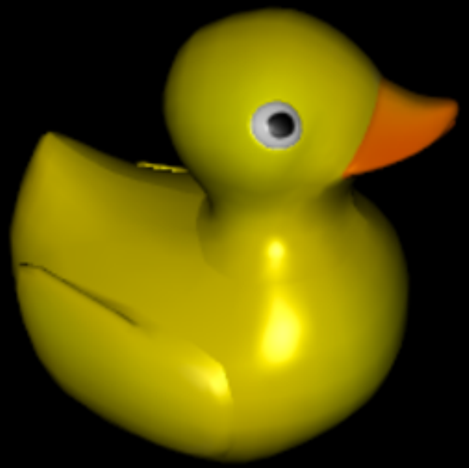
Blur the source horizontally



Blur the blur verticaly



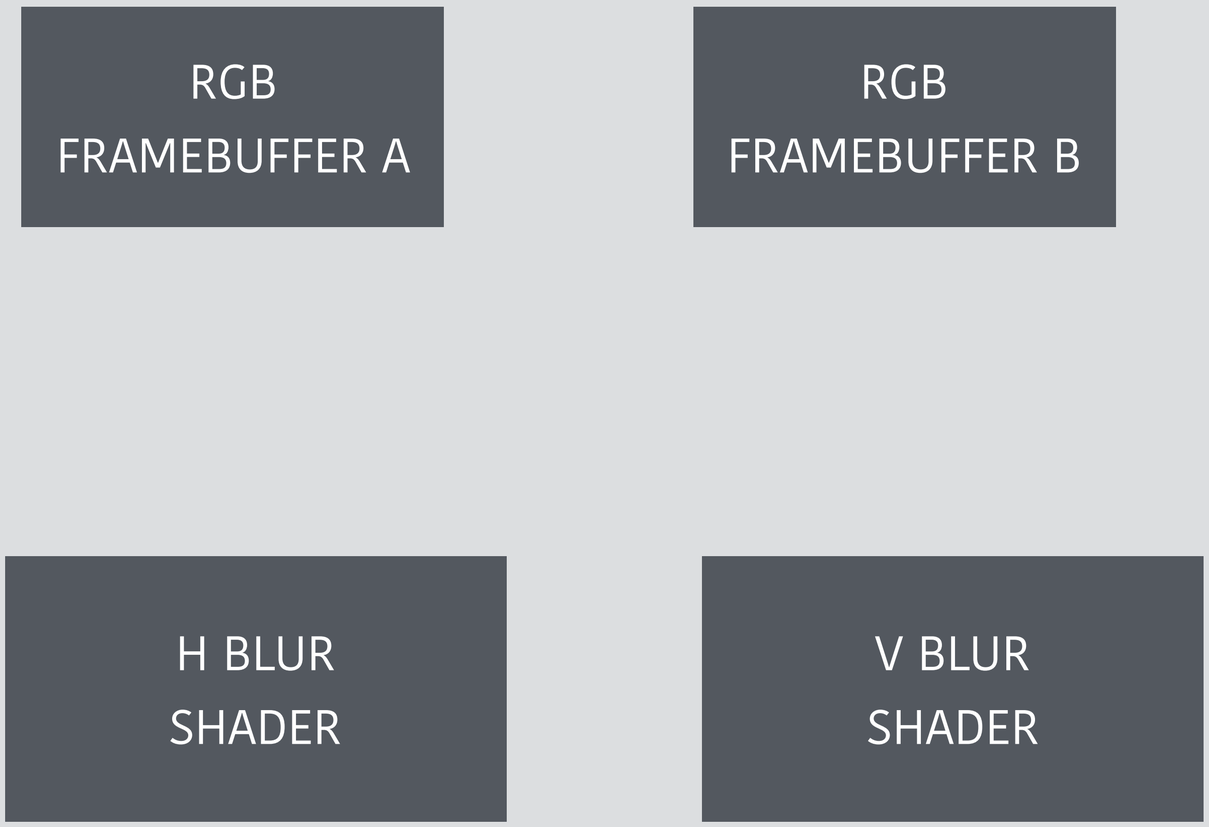
Result



Basic Scene

First Blur Pass (Horizontal)

Second Blur Pass (Vertical)


RGB
FRAMEBUFFER A
RGB
FRAMEBUFFER B
H BLUR
SHADER
V BLUR
SHADER

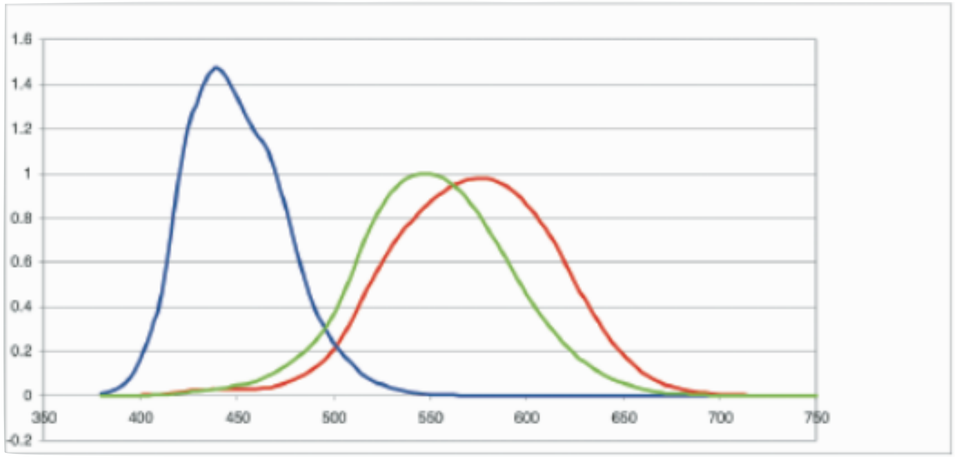
```
uniform sampler2D screenFramebuffer;
varying vec2 texCoordVar;

void main(void)
{
    const float blurSize = 0.004;
    vec4 sum = vec4(0.0);

    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 16.0*blurSize, texCoordVar.y)) * 0.000004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 15.0*blurSize, texCoordVar.y)) * 0.000012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 14.0*blurSize, texCoordVar.y)) * 0.00004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 13.0*blurSize, texCoordVar.y)) * 0.00012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 12.0*blurSize, texCoordVar.y)) * 0.000331;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 11.0*blurSize, texCoordVar.y)) * 0.000841;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 10.0*blurSize, texCoordVar.y)) * 0.001971;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 9.0*blurSize, texCoordVar.y)) * 0.004258;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 8.0*blurSize, texCoordVar.y)) * 0.008483;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 7.0*blurSize, texCoordVar.y)) * 0.015583;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 6.0*blurSize, texCoordVar.y)) * 0.026396;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 5.0*blurSize, texCoordVar.y)) * 0.04123;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 4.0*blurSize, texCoordVar.y)) * 0.059384;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 3.0*blurSize, texCoordVar.y)) * 0.07887;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 2.0*blurSize, texCoordVar.y)) * 0.096593;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x - 1.0*blurSize, texCoordVar.y)) * 0.109084;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y)) * 0.113597;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 16.0*blurSize, texCoordVar.y)) * 0.000004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 15.0*blurSize, texCoordVar.y)) * 0.000012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 14.0*blurSize, texCoordVar.y)) * 0.00004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 13.0*blurSize, texCoordVar.y)) * 0.00012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 12.0*blurSize, texCoordVar.y)) * 0.000331;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 11.0*blurSize, texCoordVar.y)) * 0.000841;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 10.0*blurSize, texCoordVar.y)) * 0.001971;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 9.0*blurSize, texCoordVar.y)) * 0.004258;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 8.0*blurSize, texCoordVar.y)) * 0.008483;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 7.0*blurSize, texCoordVar.y)) * 0.015583;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 6.0*blurSize, texCoordVar.y)) * 0.026396;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 5.0*blurSize, texCoordVar.y)) * 0.04123;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 4.0*blurSize, texCoordVar.y)) * 0.059384;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 3.0*blurSize, texCoordVar.y)) * 0.07887;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 2.0*blurSize, texCoordVar.y)) * 0.096593;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x + 1.0*blurSize, texCoordVar.y)) * 0.109084;

    gl_FragColor = sum;
}
```

```
uniform sampler2D screenFramebuffer;
varying vec2 texCoordVar;

void main(void)
{
    const float blurSize = 0.005;
    vec4 sum = vec4(0.0);

    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 16.0*blurSize)) * 0.000004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 15.0*blurSize)) * 0.000012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 14.0*blurSize)) * 0.00004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 13.0*blurSize)) * 0.00012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 12.0*blurSize)) * 0.000331;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 11.0*blurSize)) * 0.000841;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 10.0*blurSize)) * 0.001971;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 9.0*blurSize)) * 0.004258;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 8.0*blurSize)) * 0.008483;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 7.0*blurSize)) * 0.015583;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 6.0*blurSize)) * 0.026396;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 5.0*blurSize)) * 0.04123;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 4.0*blurSize)) * 0.059384;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 3.0*blurSize)) * 0.07887;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 2.0*blurSize)) * 0.096593;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y- 1.0*blurSize)) * 0.109084;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y)) * 0.113597;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 16.0*blurSize)) * 0.000004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 15.0*blurSize)) * 0.000012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 14.0*blurSize)) * 0.00004;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 13.0*blurSize)) * 0.00012;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 12.0*blurSize)) * 0.000331;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 11.0*blurSize)) * 0.000841;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 10.0*blurSize)) * 0.001971;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 9.0*blurSize)) * 0.004258;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 8.0*blurSize)) * 0.008483;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 7.0*blurSize)) * 0.015583;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 6.0*blurSize)) * 0.026396;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 5.0*blurSize)) * 0.04123;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 4.0*blurSize)) * 0.059384;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 3.0*blurSize)) * 0.07887;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 2.0*blurSize)) * 0.096593;
    sum += texture2D(screenFramebuffer, vec2(texCoordVar.x, texCoordVar.y+ 1.0*blurSize)) * 0.109084;

    gl_FragColor = sum;
}
```

1. Bind first frame buffer
2. Render scene normally.
3. Bind second framebuffer.
4. Render first framebuffer as screen shader using horizontal blur shader.
5. Unbind framebuffer.
6. Render second framebuffer to the screen using vertical blur shader.

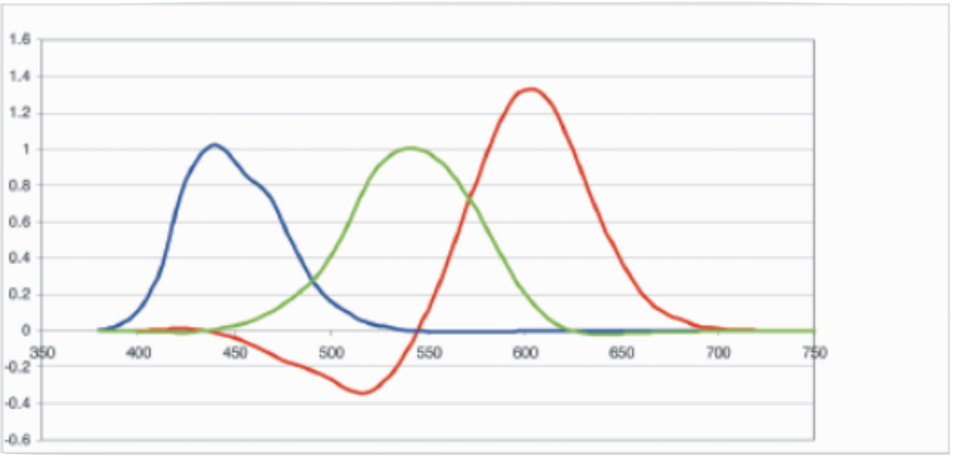
Color

$k_s(\lambda)$ $k_m(\lambda)$ $k_l(\lambda)$



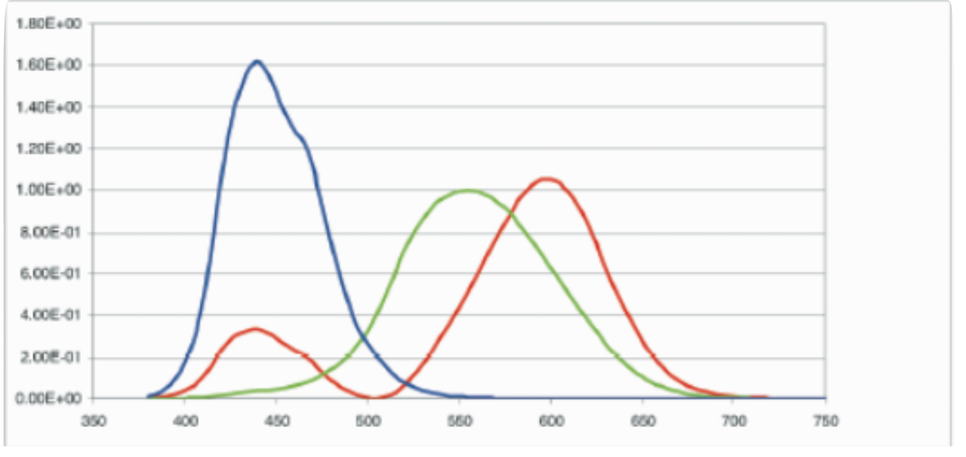
LMS senstivity functions

$k_{435}(\lambda)$ $k_{545}(\lambda)$ $k_{625}(\lambda)$



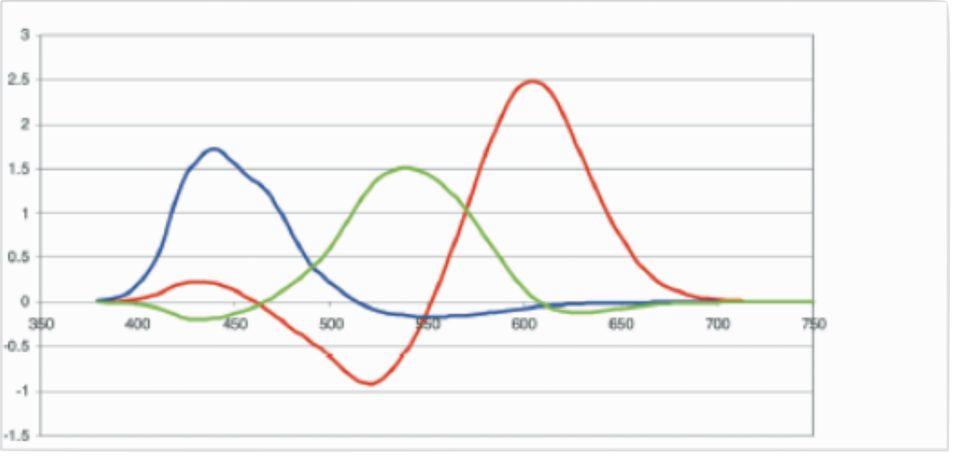
Matching experiment matching functions

$k_z(\lambda)$ $k_y(\lambda)$ $k_x(\lambda)$



XYZ matching functions

$k_b(\lambda)$ $k_g(\lambda)$ $k_r(\lambda)$

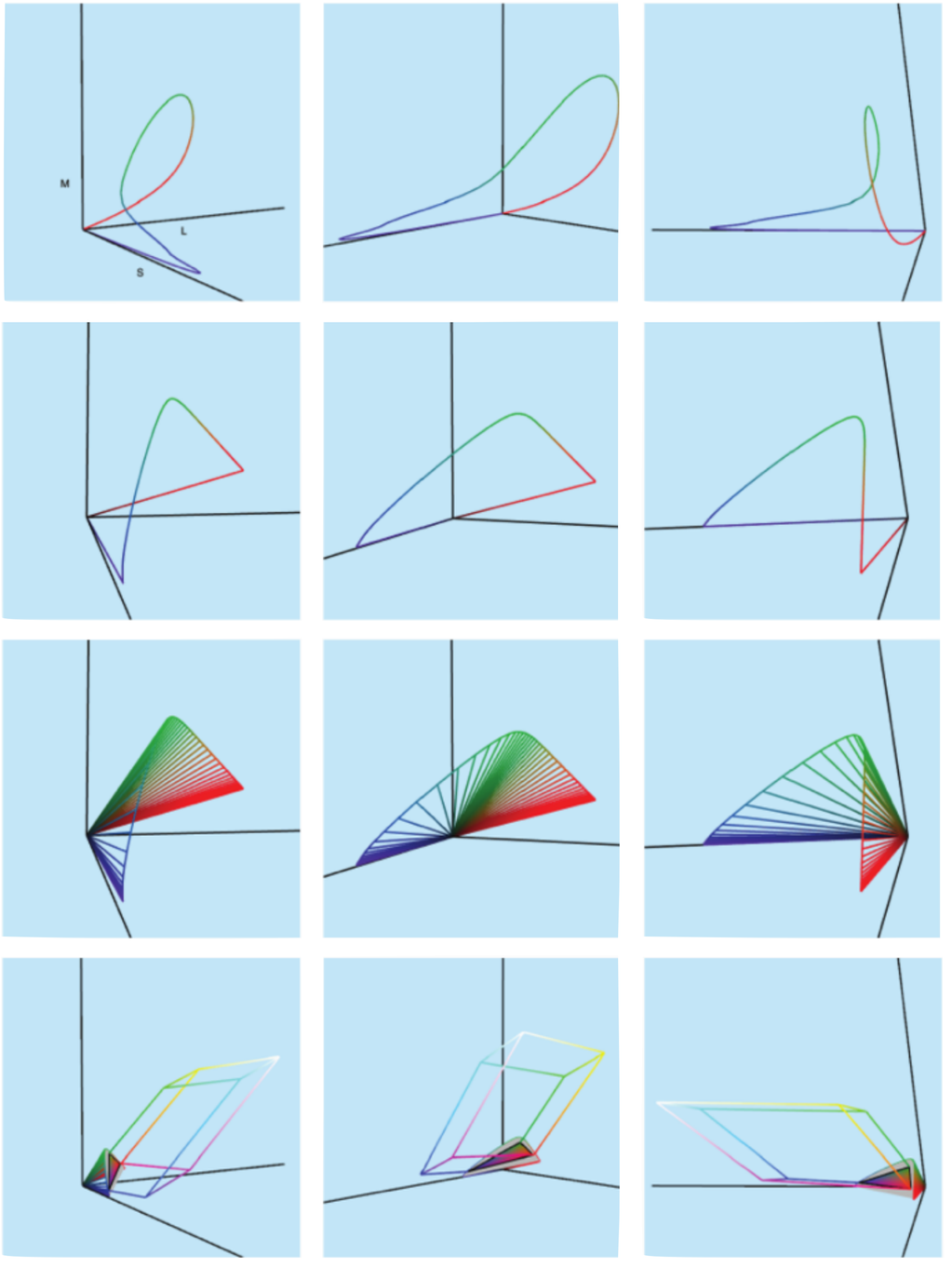


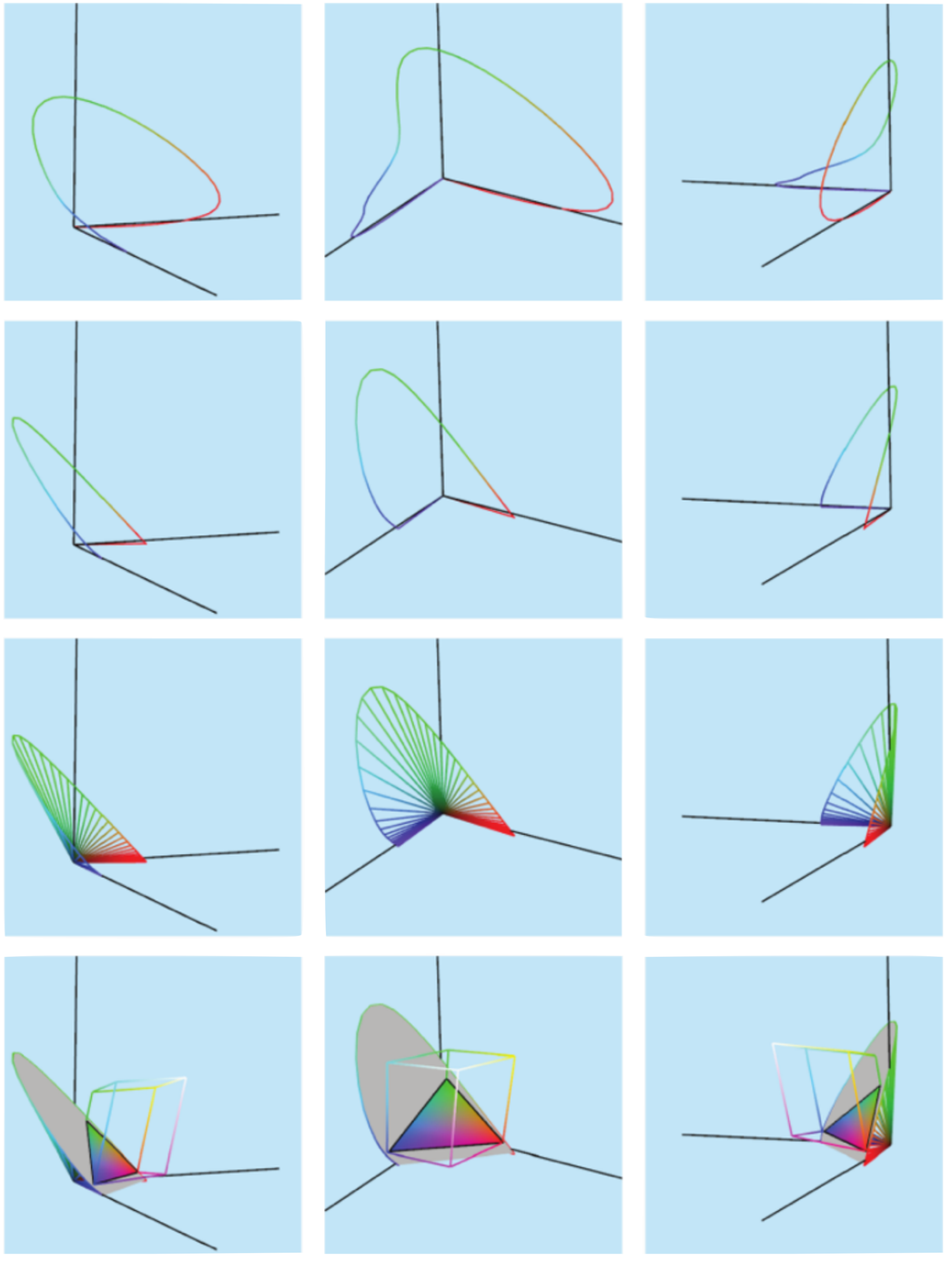
RGB matching functions

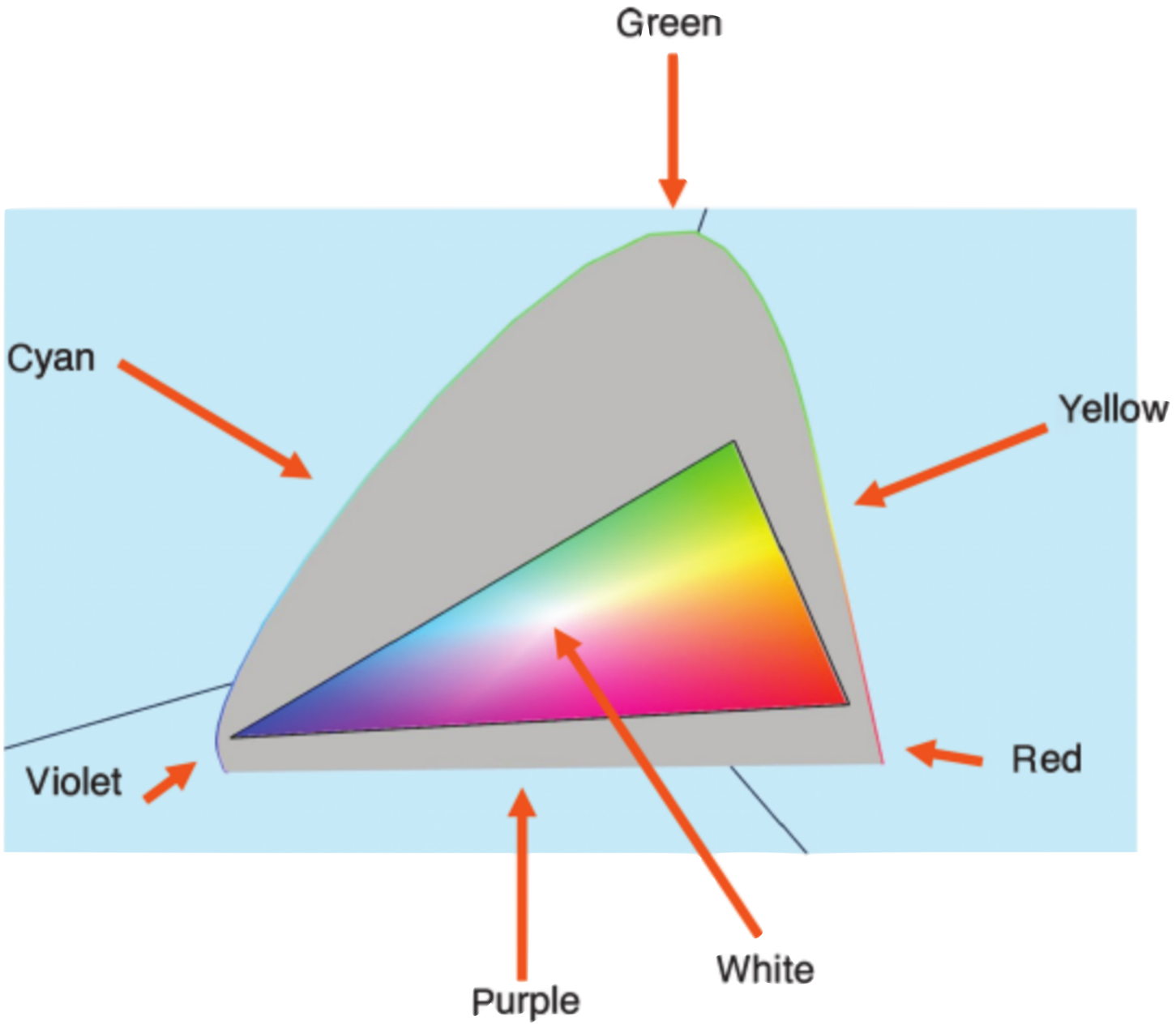
$$L = \int_{\Omega} d\lambda \;\; l(\lambda)\, k_l(\lambda)$$

$$M = \int_{\Omega} d\lambda \;\; l(\lambda)\, k_m(\lambda)$$

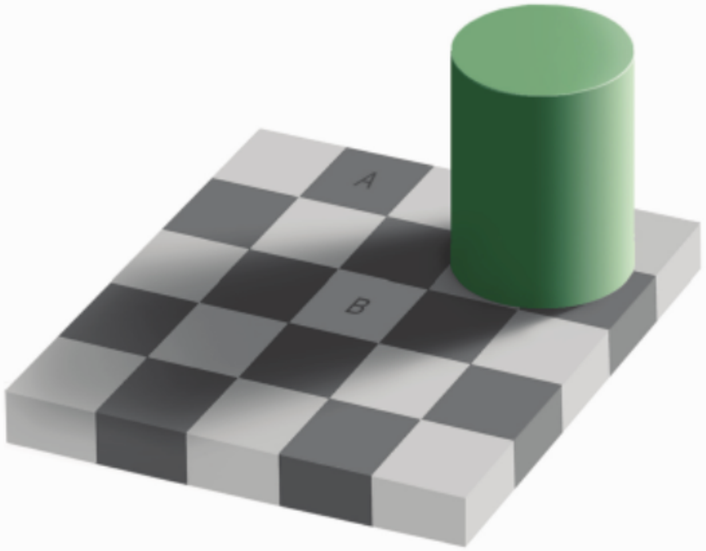
$$S = \int_{\Omega} d\lambda \;\; l(\lambda)\, k_s(\lambda),$$

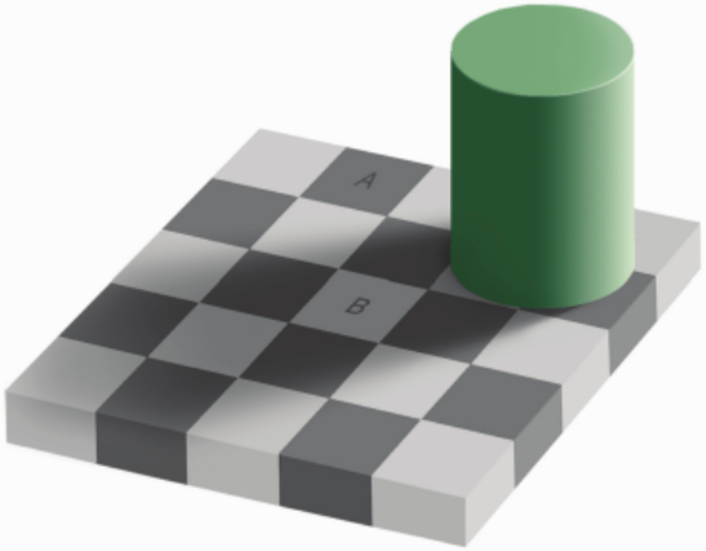

M
L
S

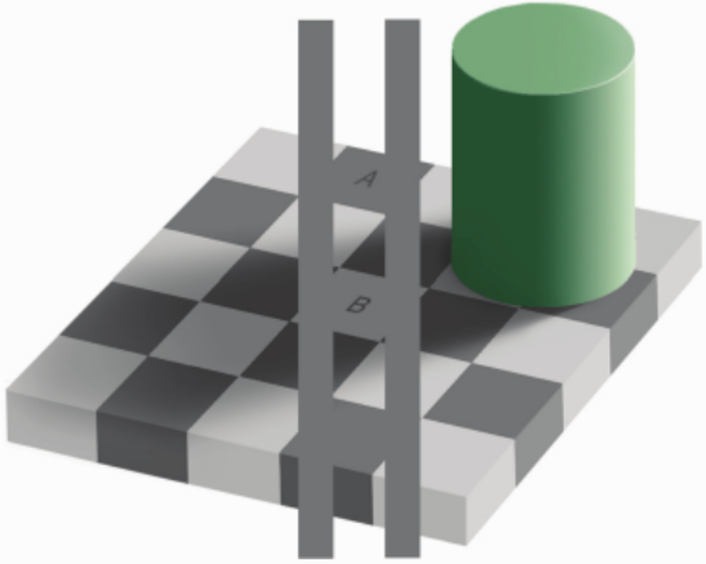



Green
Cyan
Yellow
Violet
Red
Purple
White

Color perception and adaptation.


A
B


A
B


A
B

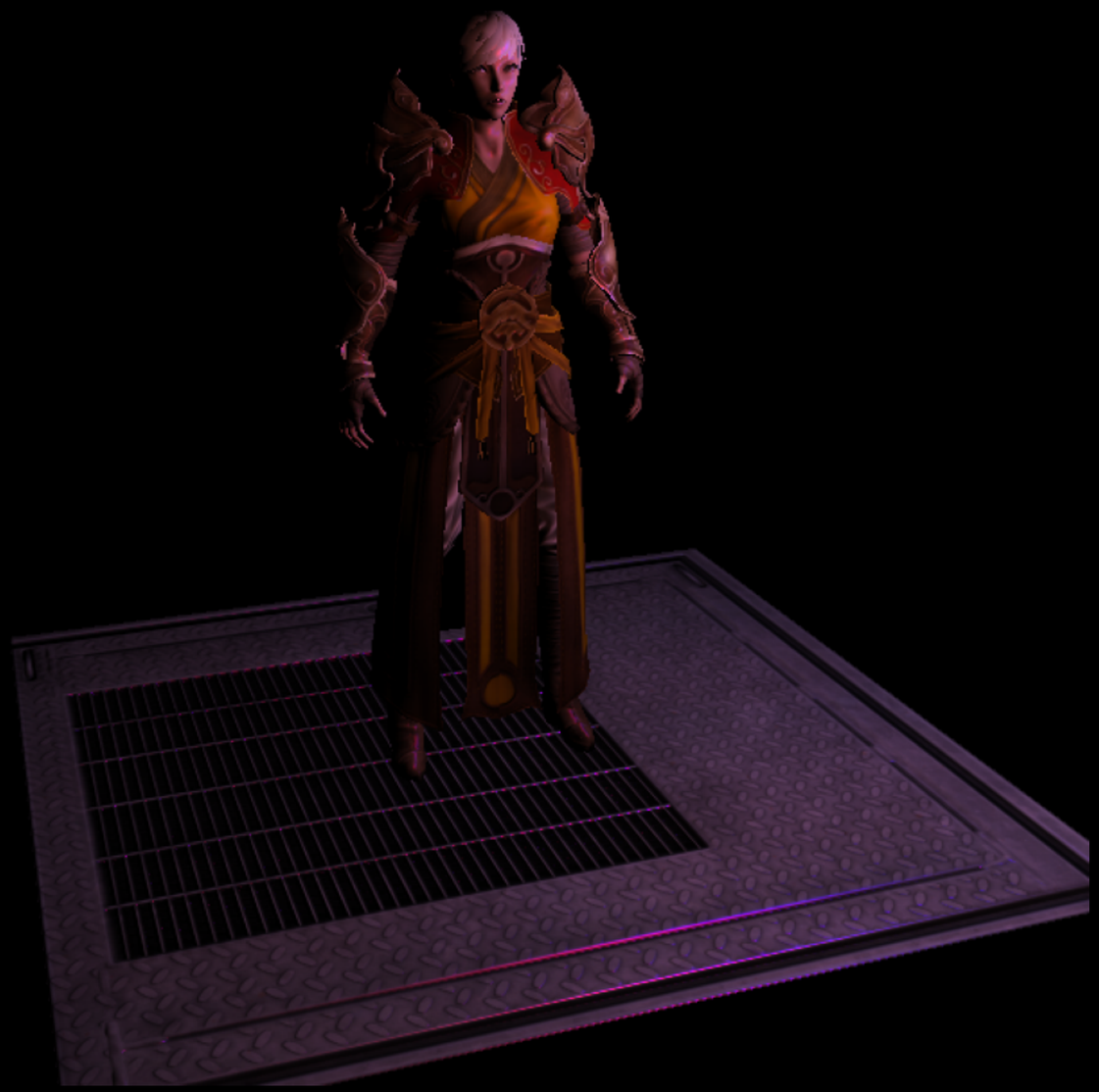
Gamma correction and sRGB.

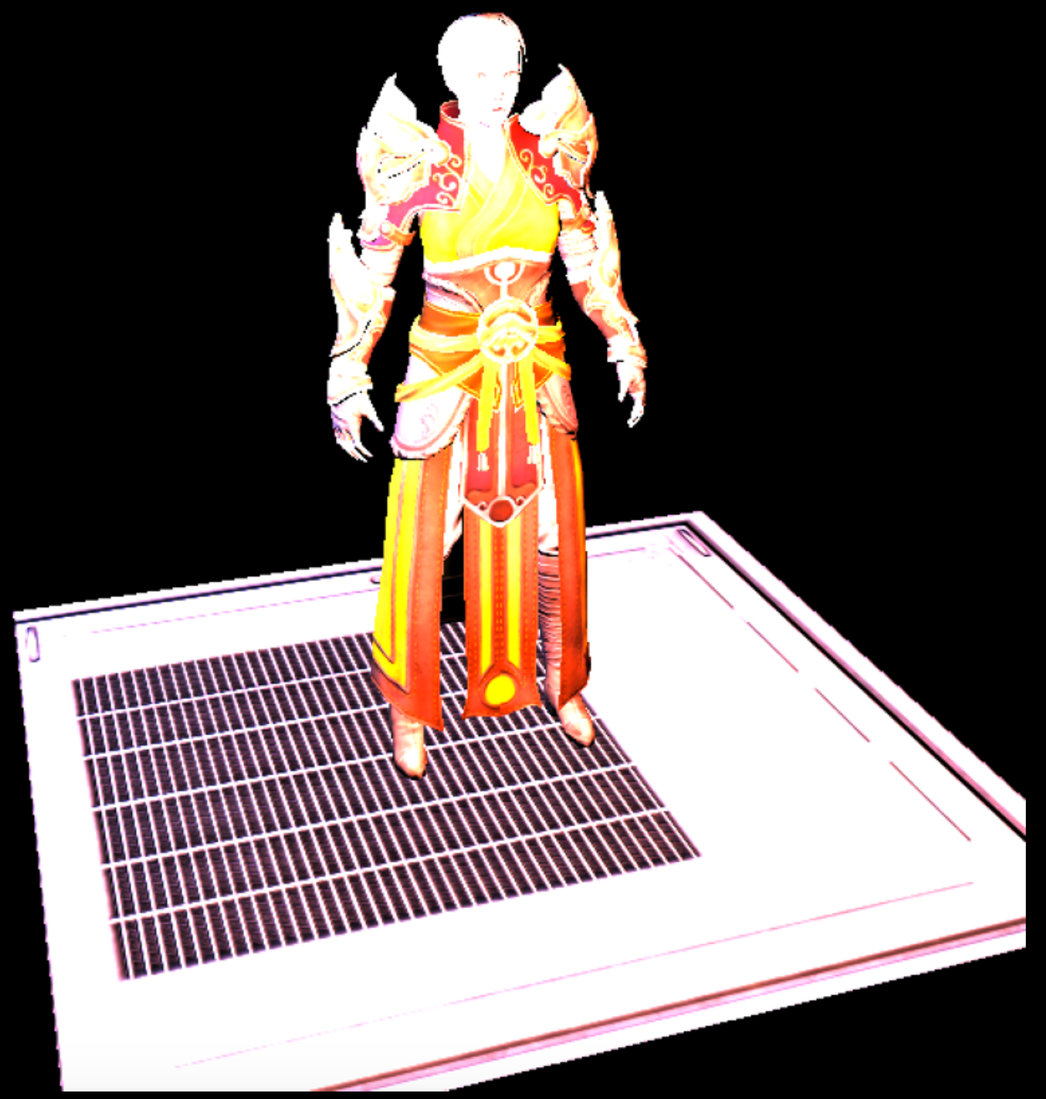
$$R = (R')^{\frac{1}{0.45}} \qquad R' = R^{0.45}$$

$$G = (G')^{\frac{1}{0.45}} \qquad G' = G^{0.45}$$

$$B = (B')^{\frac{1}{0.45}}. \qquad B' = B^{0.45}.$$

High Dynamic Range color





Really bright lights calculated in RGB colorspace.

Rendering to a high precision framebuffer.

```
glGenTextures(1, &frameBufferTexture);
glBindTexture(GL_TEXTURE_2D, frameBufferTexture);
glTexImage2D(GL_TEXTURE_2D, 0,GL_RGBA16F_ARB, WINDOW_WIDTH, WINDOW_HEIGHT, 0, GL_RGB, GL_FLOAT, NULL);
```

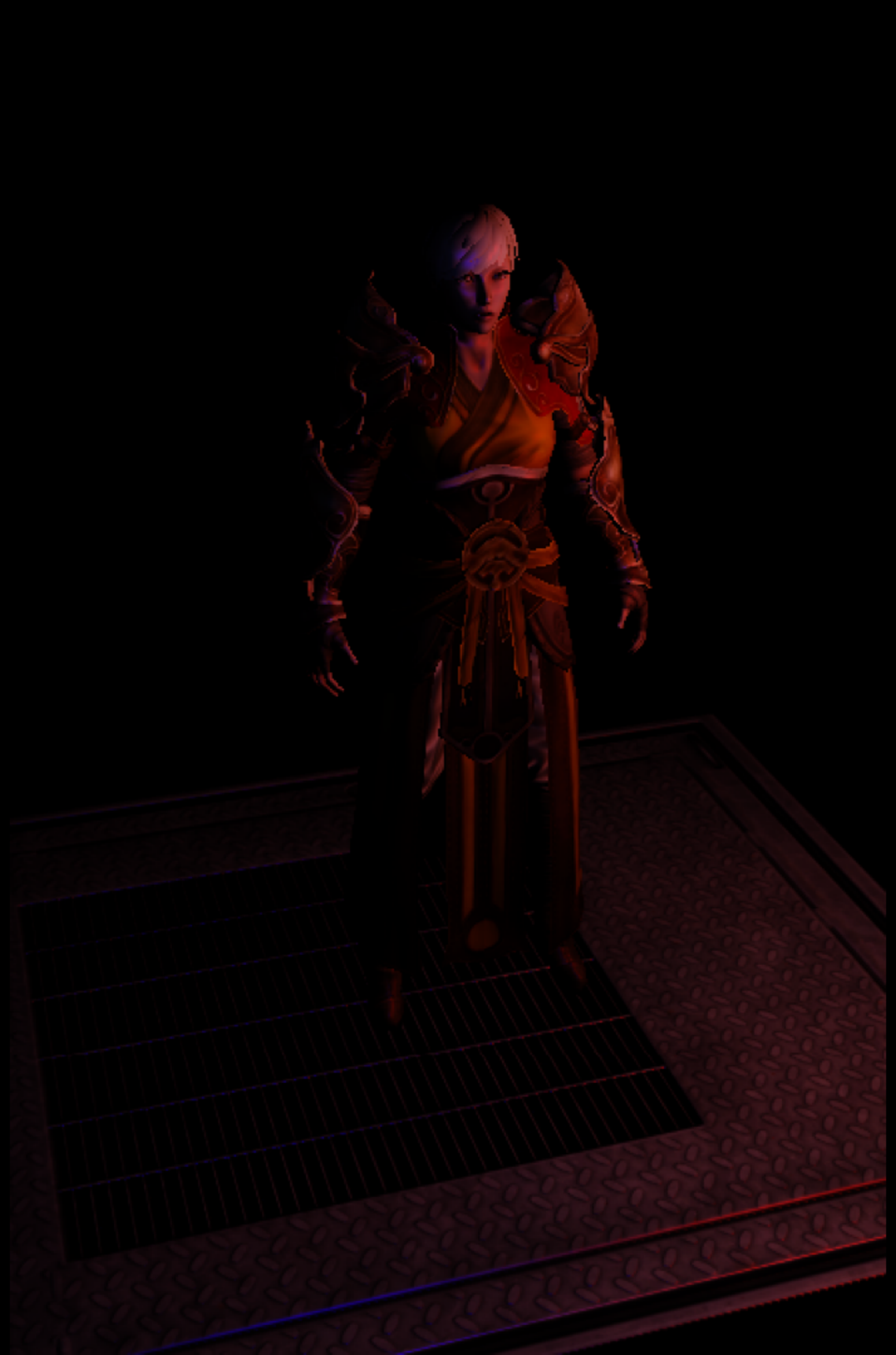
Your internal pixel format needs to be **GL_RGBA16F_ARB** (or **GL_RGBA16F_EXT** or **GL_RGBA16** depending on OpenGL version) and your pixel storage type needs to be **GL_FLOAT**.

HDR tone mapping.

```
uniform sampler2D screenFramebuffer;
varying vec2 texCoordVar;

void main()
{
    const float gamma = 2.2;
    vec3 hdrColor = texture2D( screenFramebuffer, texCoordVar).rgb;
    vec3 mapped = hdrColor / (hdrColor + vec3(1.0));
    mapped = pow(mapped, vec3(gamma));
    gl_FragColor = vec4(mapped, 1.0);
}
```

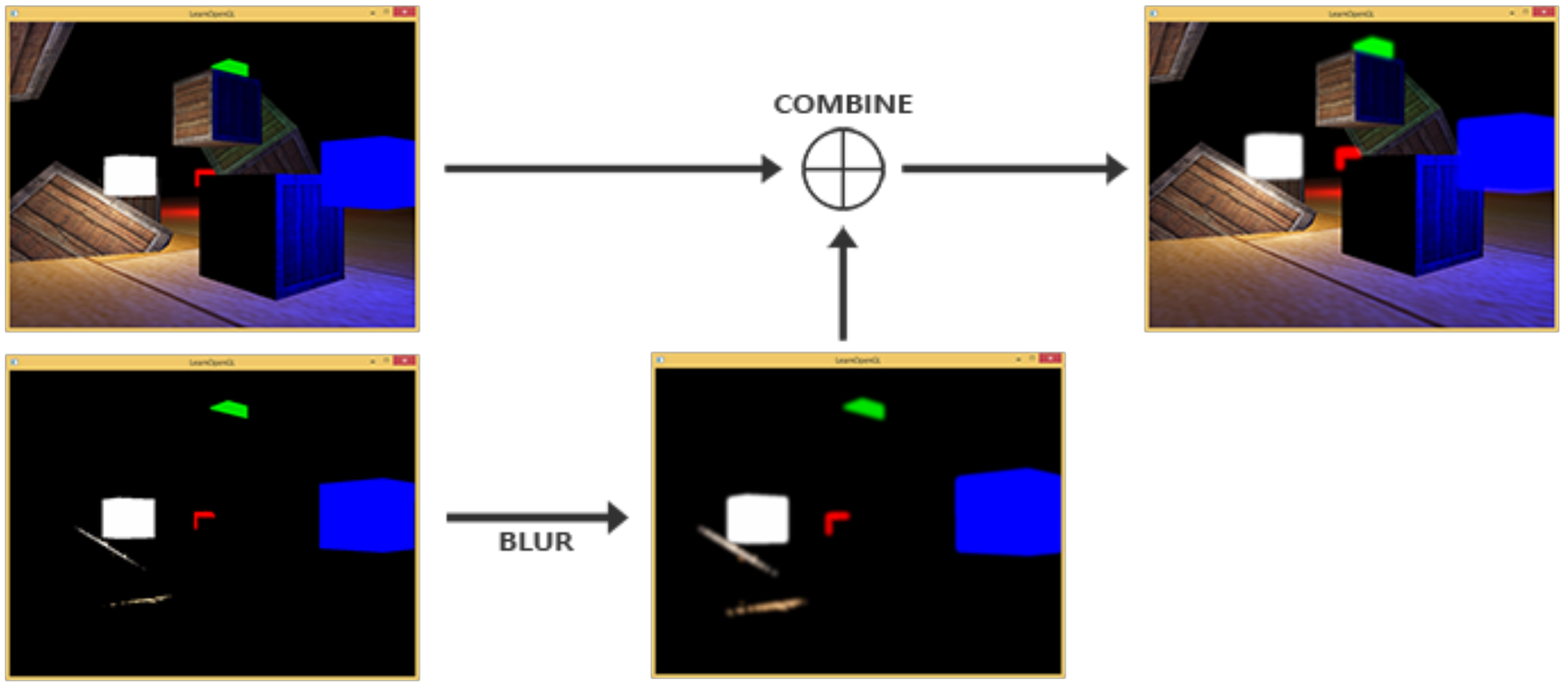
Tone mapping with exposure.

```
uniform sampler2D screenFramebuffer;
varying vec2 texCoordVar;

uniform float exposure;

void main()
{
    const float gamma = 2.2;
    vec3 hdrColor = texture2D( screenFramebuffer, texCoordVar).rgb;
    vec3 mapped = vec3(1.0) - exp(-hdrColor * exposure);
    mapped = pow(mapped, vec3(gamma));
    gl_FragColor = vec4(mapped, 1.0);
}
```



HDR and glare.

Glare in a photograph.


COMBINE
BLUR

GL_RGBA16
FRAMEBUFFERA

RGB
FRAMEBUFFER B

RGB
FRAMEBUFFER C

CLAMP LUMINANCE
SHADER

H BLUR
SHADER

V BLUR
SHADER

FINAL ADDITIVE
SHADER

Luminance clamp shader.

```
uniform sampler2D screenFramebuffer;
varying vec2 texCoordVar;

uniform float exposure;

void main()
{
    const float gamma = 2.2;
    vec3 hdrColor = texture2D( screenFramebuffer, texCoordVar).rgb;
    vec3 mapped = vec3(1.0) - exp(-hdrColor * exposure);

    float luminance = dot(mapped, vec3(0.299, 0.587, 0.144));
    mapped = mapped * smoothstep(0.8, 1.0, luminance) * 2.0;
    mapped = pow(mapped, vec3(gamma));

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

Final additive shader

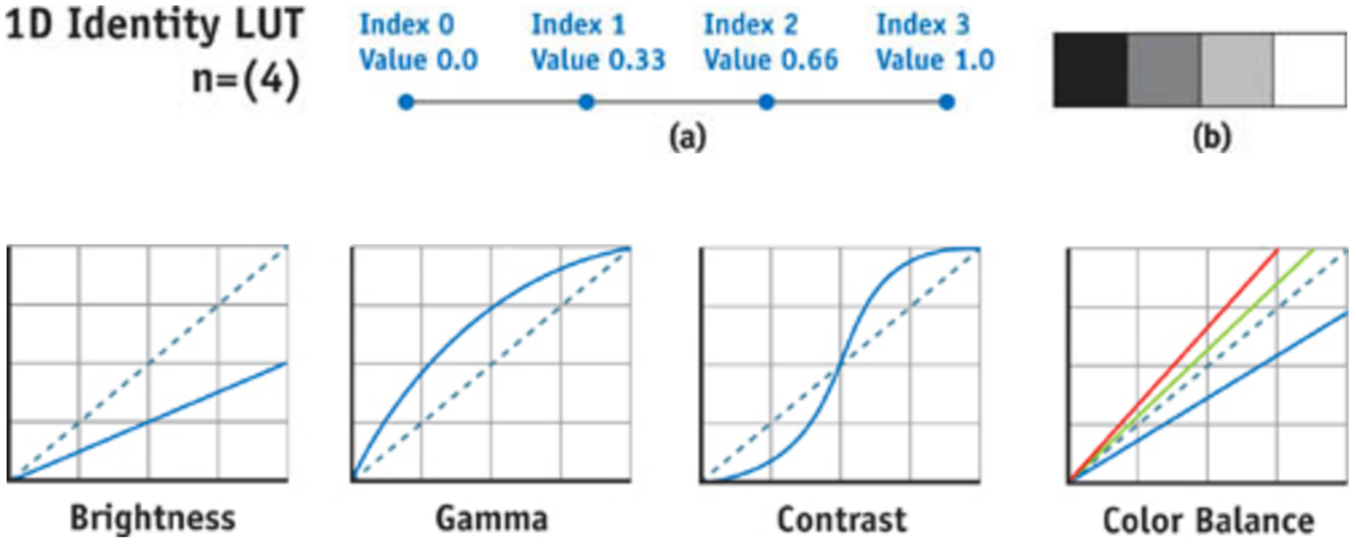
```
uniform sampler2D screenFramebuffer;
uniform sampler2D blurredHighlights;

varying vec2 texCoordVar;

uniform float exposure;

void main()
{
     const float gamma = 2.2;
    vec3 hdrColor = texture2D( screenFramebuffer, texCoordVar).rgb;
    vec3 mapped = vec3(1.0) - exp(-hdrColor * exposure);
     mapped = pow(mapped, vec3(gamma));
     mapped += texture2D(blurredHighlights, texCoordVar).rgb;
     gl_FragColor = vec4(mapped, 1.0);
}
```

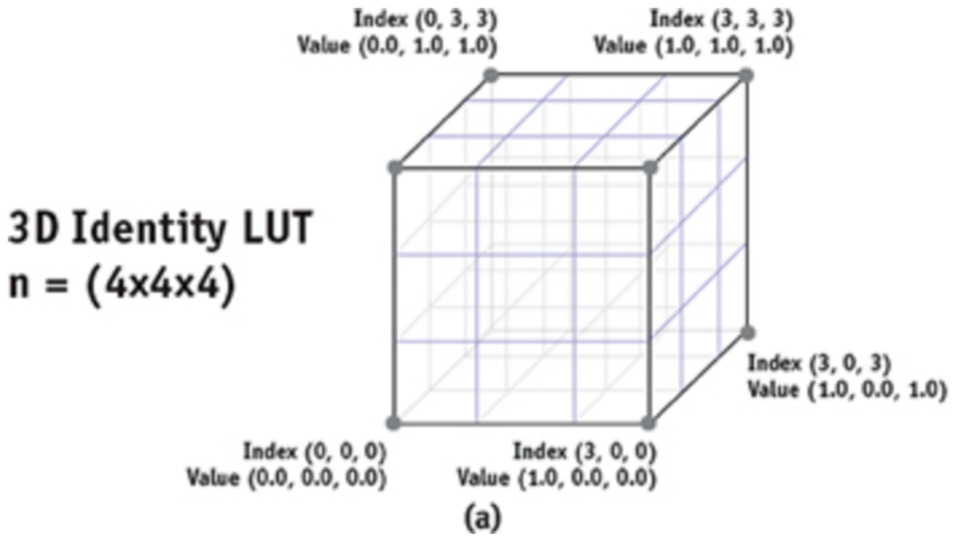
1. Bind Framebuffer A (GL_RGBA16)
2. Render scene normally.
3. Bind Framebuffer B
4. Render Framebuffer A using luminance clamp shader.
5. Bind Framebuffer C
6. Render Framebuffer B using BlurH shader
7. Bind Framebuffer B
8. Render Framebuffer C using Blur V shader
9. Unbind framebuffer.
10. Render Framebuffer A and Framebuffer B using the final additive shader.

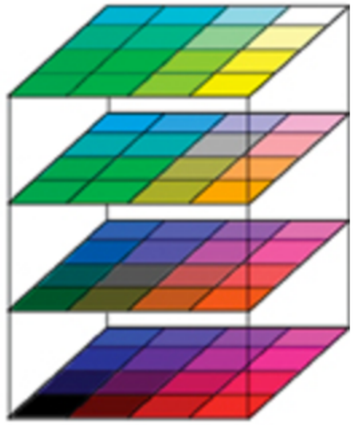
Color correction and LUTs.

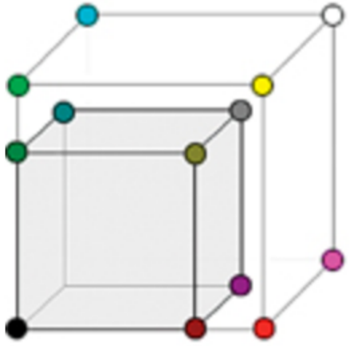

ASSASSIN'S
CREED
UNITY

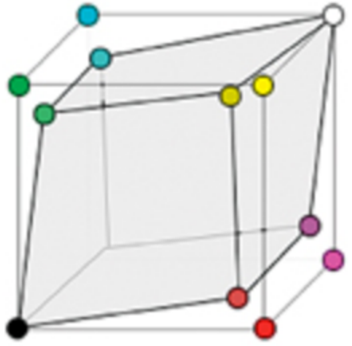
Look Up Tables (LUTs).

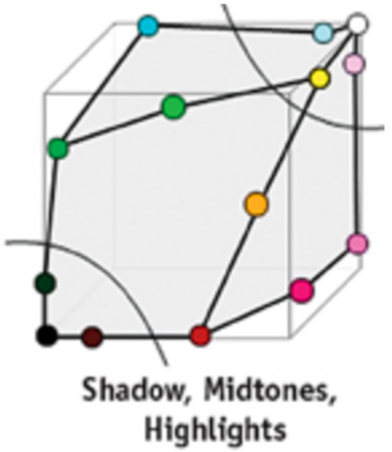

1D Identity LUT
n=(4)
Index 0
Value 0.0
Index 1
Value 0.33
Index 2
Value 0.66
Index 3
Value 1.0
(a)
(b)
Brightness
Gamma
Contrast
Color Balance

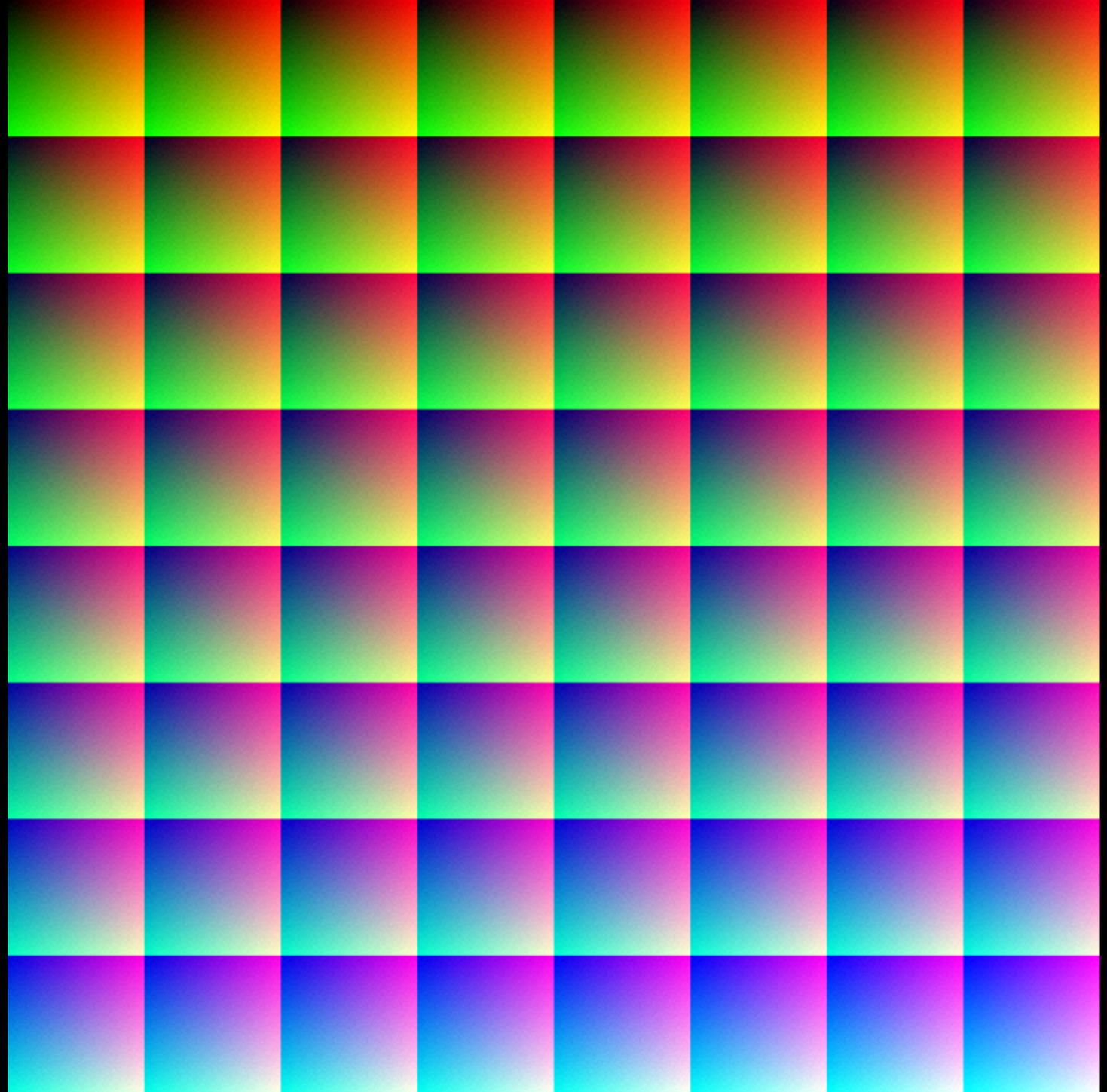

3D Identity LUT
n = (4x4x4)
Index (0, 3, 3)
Value (0.0, 1.0, 1.0)
Index (3, 3, 3)
Value (1.0, 1.0, 1.0)
Index (3, 0, 3)
Value (1.0, 0.0, 1.0)
Index (0, 0, 0)
Value (0.0, 0.0, 0.0)
Index (3, 0, 0)
Value (1.0, 0.0, 0.0)
(a)

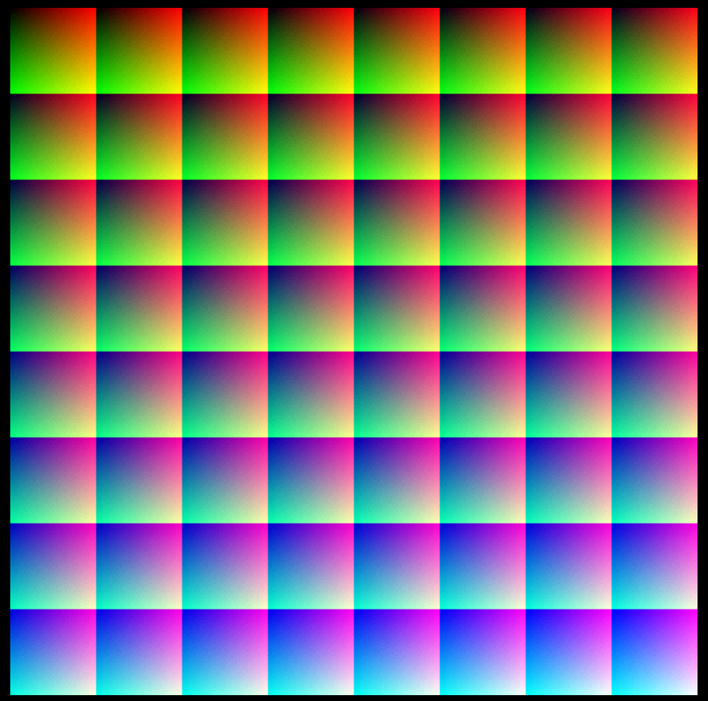

(b)

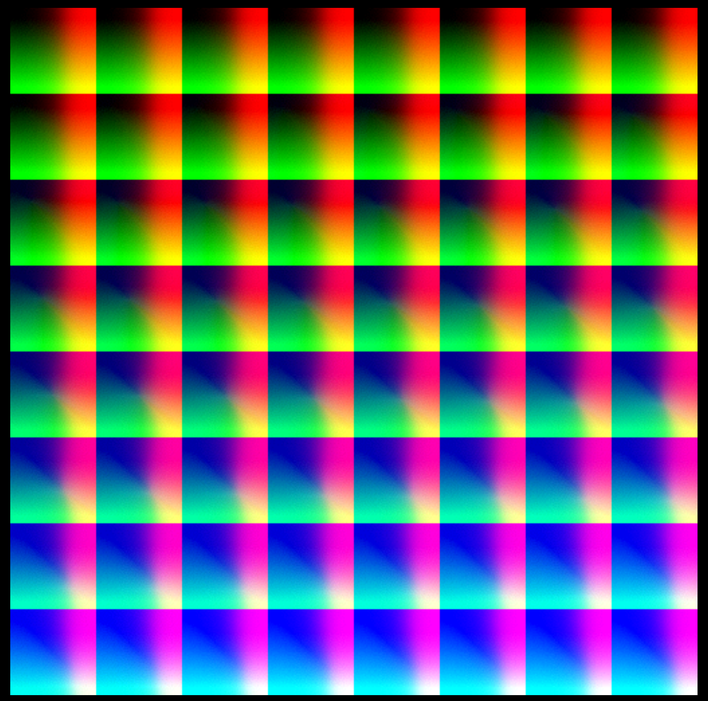

Brightness
(a)


Desaturation
(b)


Shadow, Midtones,
Highlights
(c)







```
vec4 lookup(vec4 textureColor, sampler2D lookupTable) {

    float blueColor = textureColor.b * 63.0;

    vec2 quad1;
    quad1.y = floor(floor(blueColor) / 8.0);
    quad1.x = floor(blueColor) - (quad1.y * 8.0);

    vec2 quad2;
    quad2.y = floor(ceil(blueColor) / 8.0);
    quad2.x = ceil(blueColor) - (quad2.y * 8.0);

    vec2 texPos1;
    texPos1.x = (quad1.x * 0.125) + 0.5/512.0 + ((0.125 - 1.0/512.0) * textureColor.r);
    texPos1.y = (quad1.y * 0.125) + 0.5/512.0 + ((0.125 - 1.0/512.0) * textureColor.g);

    vec2 texPos2;
    texPos2.x = (quad2.x * 0.125) + 0.5/512.0 + ((0.125 - 1.0/512.0) * textureColor.r);
    texPos2.y = (quad2.y * 0.125) + 0.5/512.0 + ((0.125 - 1.0/512.0) * textureColor.g);

    vec4 newColor1 = texture2D(lookupTable, texPos1);
    vec4 newColor2 = texture2D(lookupTable, texPos2);

    vec4 newColor = mix(newColor1, newColor2, fract(blueColor));
    return newColor;
}
```

Assignment #4

- Add a post-processing screen shader to your previous homework.

- It must combine at least two different post processing effects.