

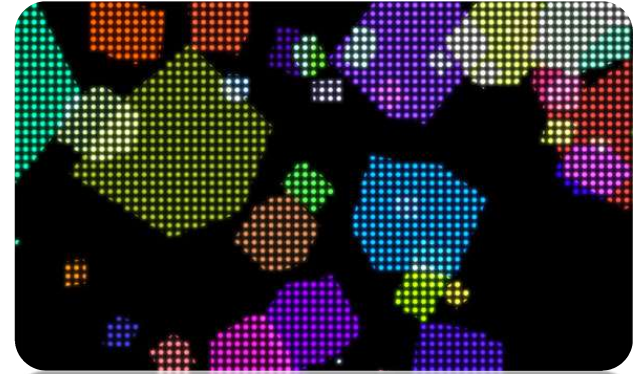
# Multipass Rendering

# Multipass Rendering: Why?

- Many effects possible
  - Dynamic environment maps
  - Dynamic shadow maps
  - Reflections/mirrors
  - Dynamic impostors
  - Screen space effects
  - Every texture can be dynamic

# Multipass Rendering: How?

- Render to texture
  - E.g.: environment maps, shadow maps
  - Shaders render into up to 8 textures in parallel
- Render while using this texture
  1. Second pass maps textures to a quad in screenspace
    - Fragment shaders process textures
  2. Or form different points of view/time ...
    - E.g.: reflections, mirrors
- Can use depth, stencil, alpha, ... tests



# Screenspace Effects

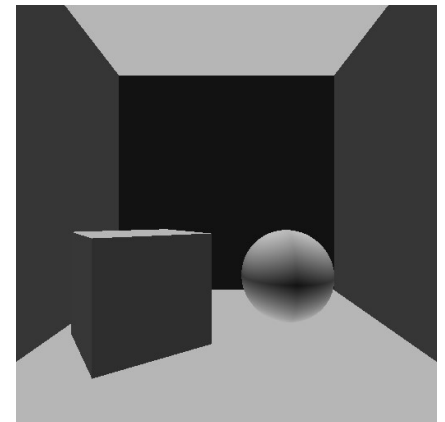
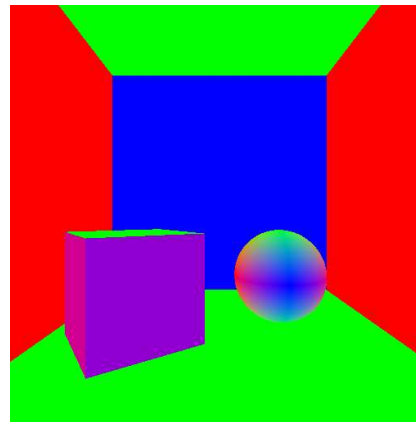


# Introduction

- General idea:
  - Render all data necessary into textures
  - Process textures to calculate final image
- Achievable Effects:
  - Glow/Bloom
  - Depth of field
  - Distortions
  - High dynamic range compression (HDR)
  - Edge detection
  - Cartoon rendering
  - Lots more...

# Grayscale

```
uniform sampler2D image;  
in vec2 uv;  
float grayScale(vec3 color)  
{  
    return 0.2126 * color.r + 0.7152 * color.g + 0.0722 * color.b;  
}  
  
void main()  
{  
    vec3 image = texture(image, uv).rgb;  
    image = vec3(grayScale(image));  
    gl_FragColor = vec4(image, 1.0);  
}
```



# Bad TV

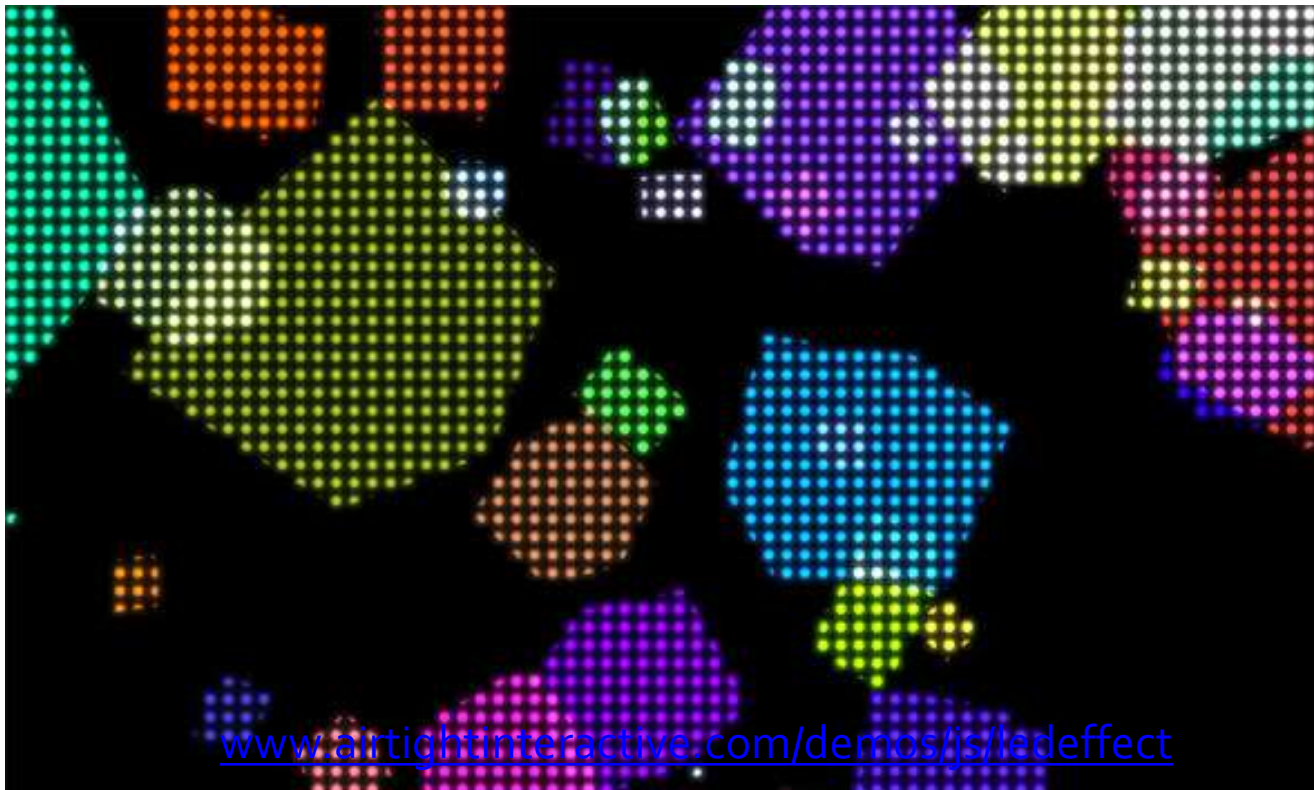
- Vertical distortions
- Chroma displacement



[www.airtightinteractive.com/demos/js/badtvshader](http://www.airtightinteractive.com/demos/js/badtvshader)

# Dot Matrix Shader

- Render texture as a grid of dots
- Glow pass





# Standard Image Filters



- Image is filtered with 3x3 kernel:
  - Weighted texture lookups in adjacent texels
  - Edge detection through laplacian:

0	1	0
1	-4	1
0	1	0



- Emboss filter

2	0	0
0	-1	0
0	0	-1



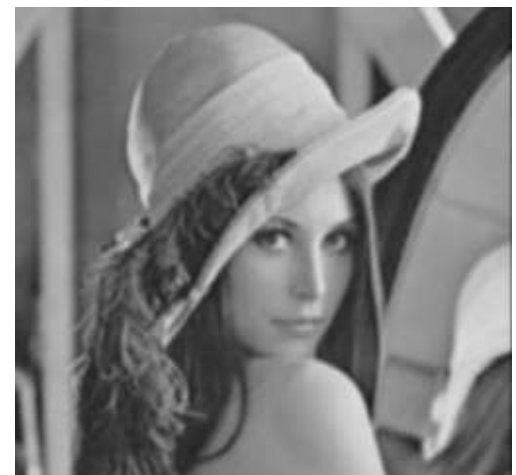
# Gaussian Filter

- Many effects based on gaussian filter
- 5x5 gaussian filter requires 25 texture lookups:

1	4	6	4	1
4	16	26	16	4
6	26	41	26	6
4	16	26	16	4
1	4	6	4	1

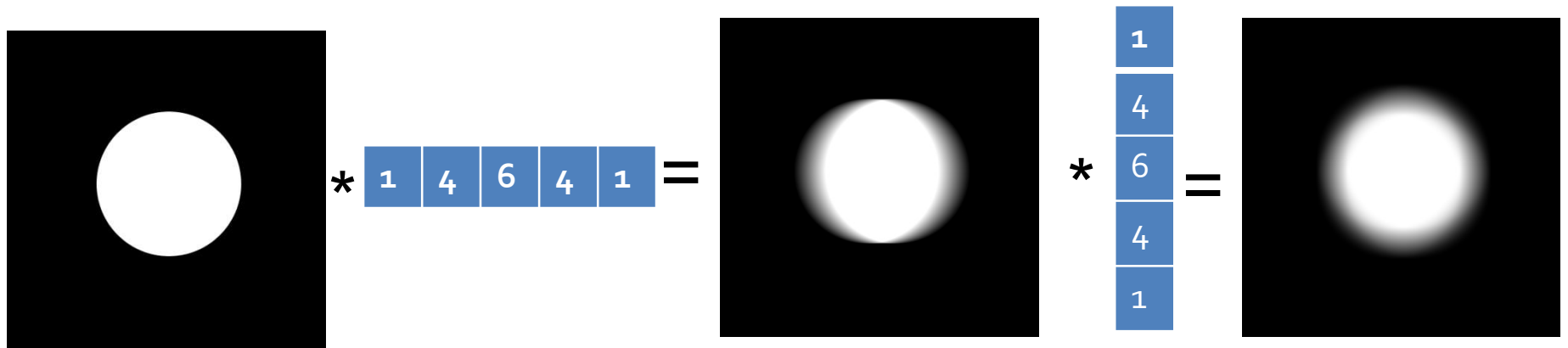
\* 1/256

- Too slow and too expensive



# Gaussian Filter

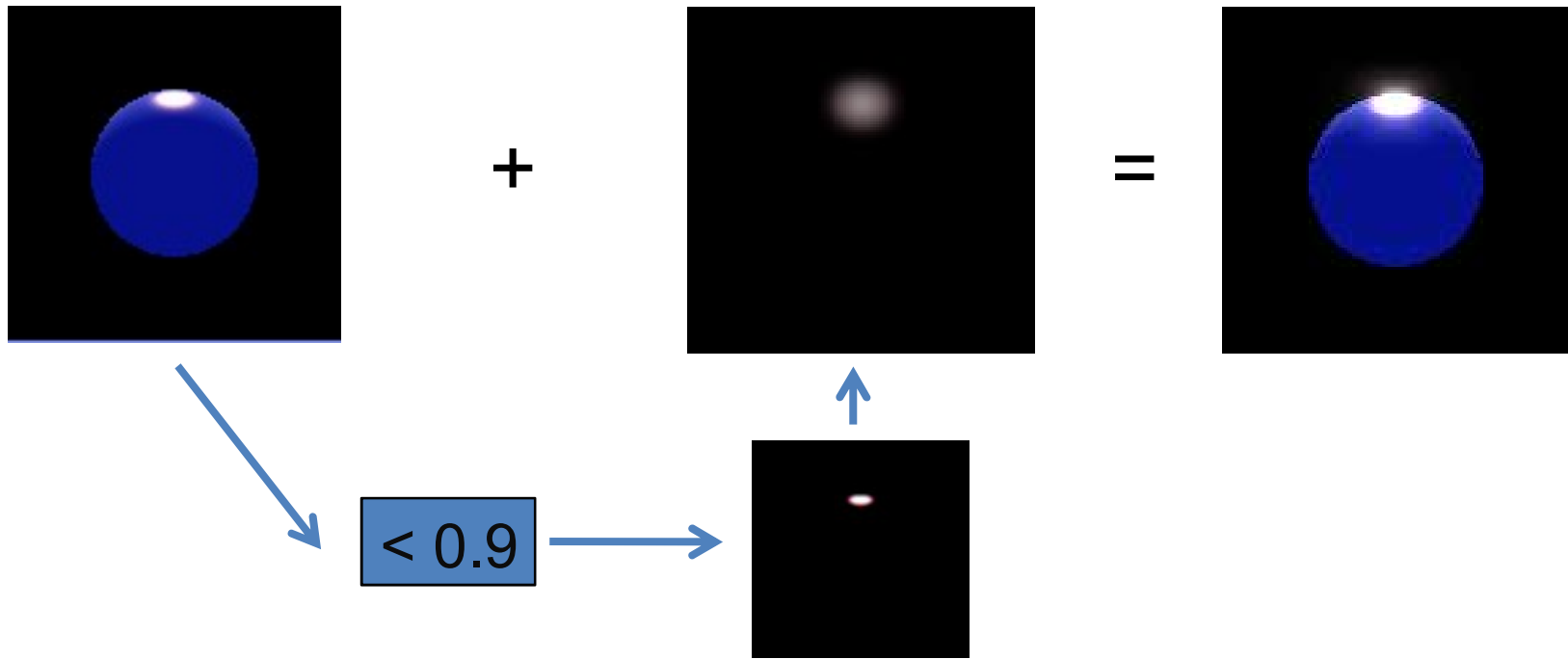
- Separate 5x5 filter into 2 passes
- Perform 5x1 filter in u
- Followed by 1x5 filter in v



- Lookups can use linear filtering
  - 5x1 filter with 3 lookups

# Bloom

- Use only bright parts for blur



# Bloom

- Bloom usually applied to downsampled render textures
  - 2x or 4x downsampled
    - Effectively increases kernel size
  - But: Sharp highlights are lost
  - Combination of differently downsampled and filtered render textures possible
    - Allows high controllability of bloom
- Filter in  $u$  and  $v$  and separate addition leads to star effect

# Bloom



Picture: Oblivion

# Bloom remarks

- Disguises aliasing artifacts
- Works best for shiny materials and sun/sky
  - Only render sun and sky to blur pass
  - Only render specular term to blur pass
- A little bit overused these days
  - Use sparsely for most effect
- Can smudge out a scene too much
  - Contrast and sharp features are lost (fairytale look)



# Bloom remarks

- Extreme example





# Motion blur

- Keep previous frames as textures
  - Blend weighted frames to final result
- Render camera space speed of each pixel in texture
- Blur along motion vector
  - Harder to implement, but looks very good
  - Faster than blending



# Other filters

- Use precomputed noise maps
  - Modulate Color with noise:
    - TV snow emulation
  - Modulate texture coordinates:
    - glass refractions
    - TV distortions
    - Warping
  - Remap intensity:
    - Heat vision
    - Eye adaptation

# HDR Rendering

- Up to now, colors are in  $[0..1]$
- Real world
  - Dynamic Range is about 1:100 000
  - 1: dark at night
  - 100 000: direct sunlight
  - Eye adapts to light intensities
- Current hardware allows to calculate everything in floating point precision and range
  - Use lights/environment maps with intensities of high dynamic range

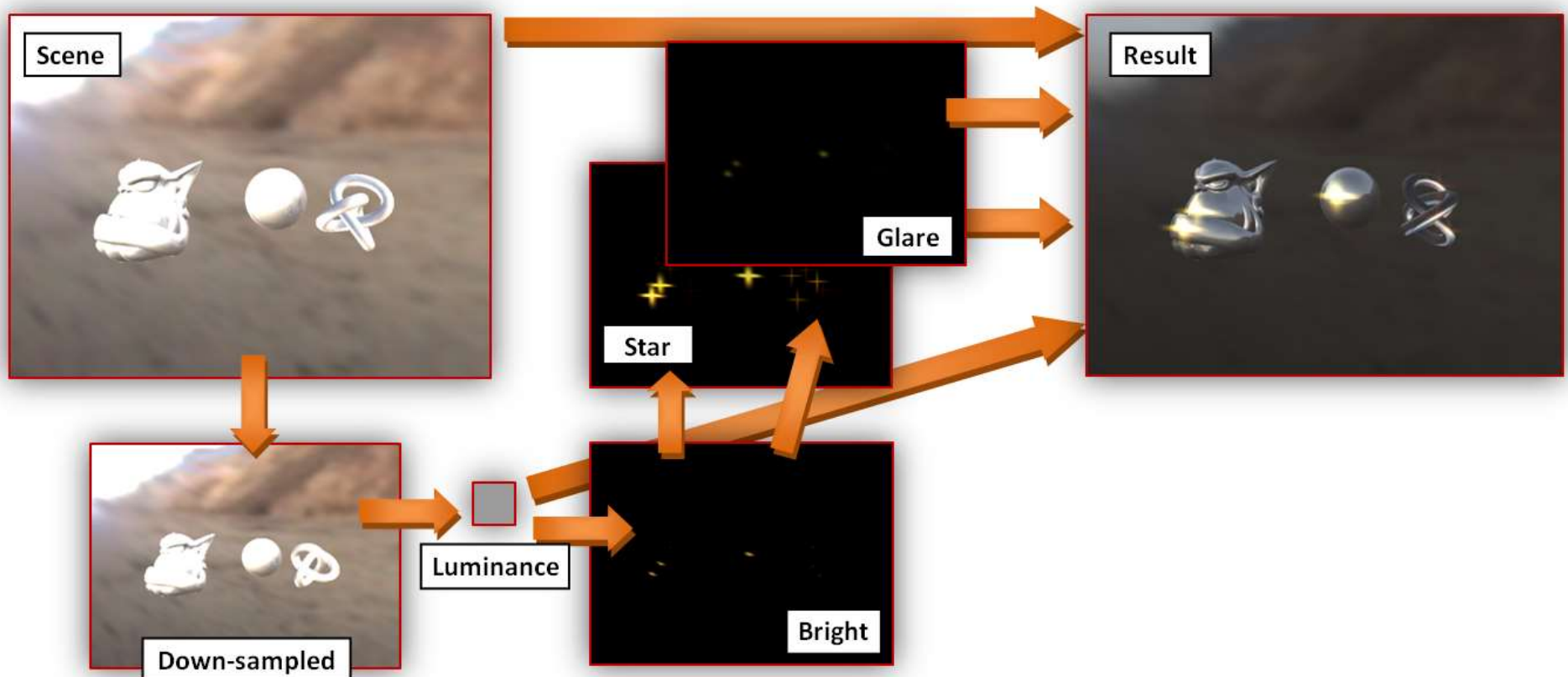
# HDR rendering

- **But:** we cannot display a HDR image!
- Solution: Remap HDR intensities to low dynamic range
- **Tone mapping**
  - Imitates human perception
  - Can mimic time delayed eye adaptation
  - Can mimic color desaturation
  - Can imitate photographic effects
  - Over exposure
  - Glares

# HDR Rendering

- Tone mapping requires information about the intensities of the HDR image
  - Extract average/maximum luminance through downsampling
    - Hardware MIP-map generation
    - Or through a series of fragment shaders
- Feed high intensities to bloom pass
  - Mimics over exposure, glare, streaks...

# HDR Processing Overview



# Tone mapping Operators (1)

- Reinhard's operator

$$L_{scaled} = \frac{a \cdot L_w}{\bar{L}_w}$$

$a$  ... Key  
 $\bar{L}_w$  ... Average luminance  
 $L_w$  ... Pixel luminance

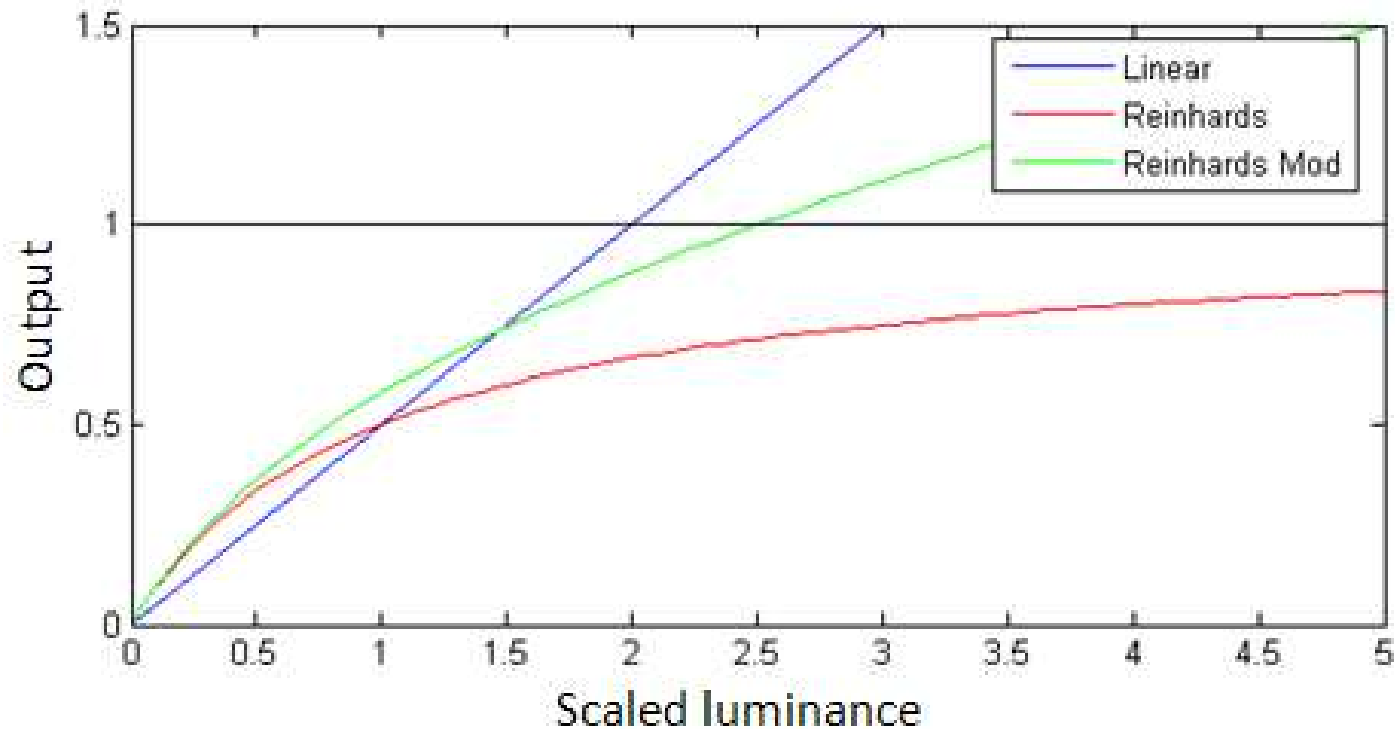
- Original  $Color = \frac{L_{scaled}}{1 + L_{scaled}}$

- Modified  $Color = \frac{L_{scaled} \cdot \left(1 + \frac{L_{scaled}}{L_{white}^2}\right)}{1 + L_{scaled}}$

- Key  $a$  is set by user or some predefined curve  $a(l_a)$  dependent on average luminance  $l_a$
- Calculations need to be done in linear color space! (floating point buffers)

# Tone mapping Operators (2)

- Reinhard's operator





# Tone mapping Operators (3)

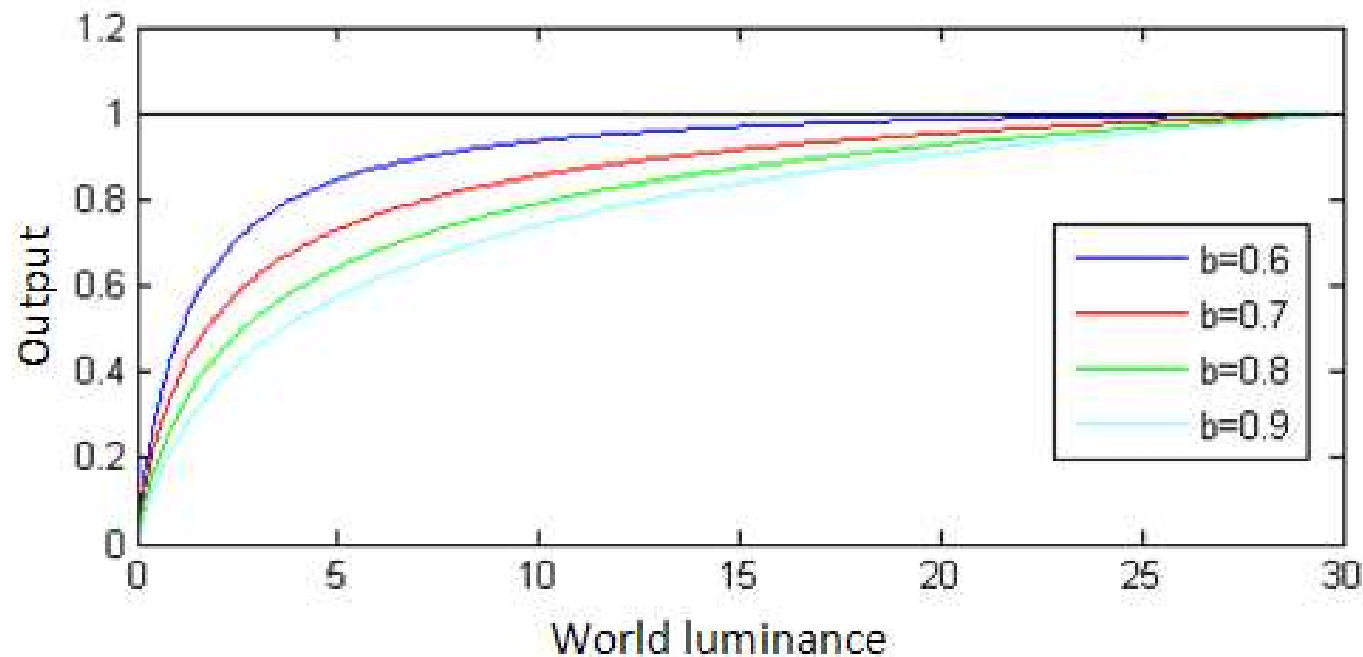
- Logarithmic mapping

$$L_d = \frac{\log_x(L_w + 1)}{\log_x(L_{max} + 1)}$$

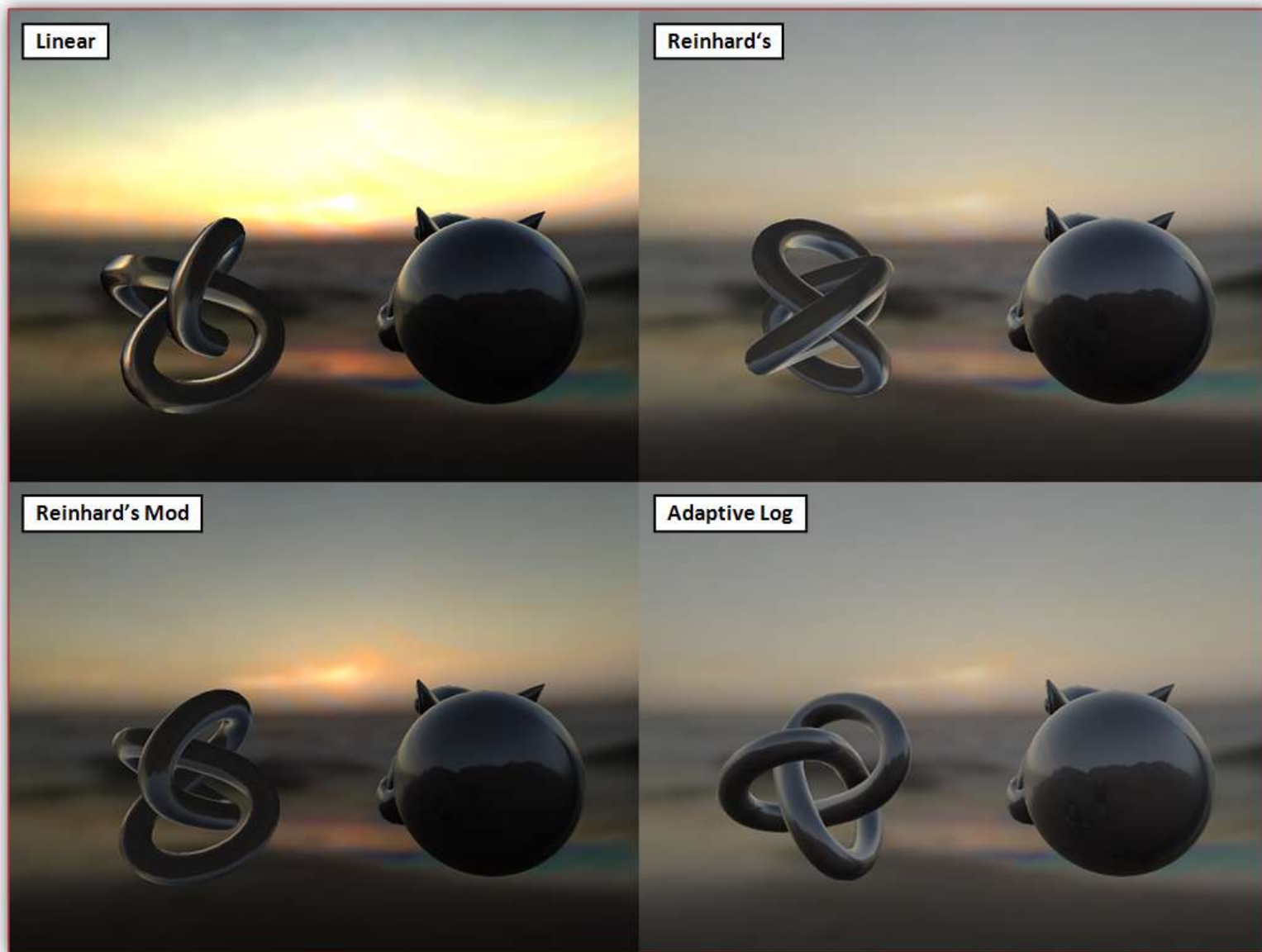
- Improvement: Adaptive logarithmic mapping
- $L_{max}$  causes heavy changes of the output color when moving through the scene

# Tone mapping Operators (4)

- Adaptive logarithmic mapping [Drago 03]



# Comparison

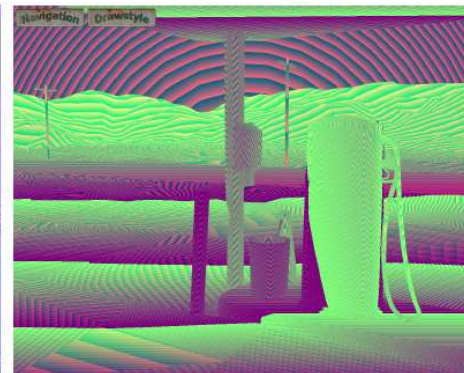
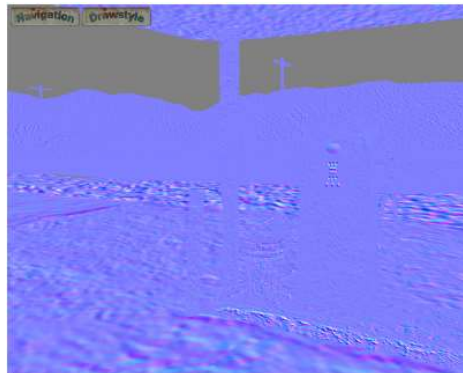


# HDR Rendering (OGRE Beach Demo)



# Deferred Shading

- General Idea: Treat lighting as a 2D postprocess
  - Deferred Shading rendered textures:
    - Normals
    - Position
    - Diffuse color
    - Material parameters
- Execute lighting calculations using the textures as input



# Deferred Shading

## ■ Pros:

- Visibility resolved before expensive shading
- Perfect batching
- Many small lights are just as cheap as a few big ones (32 lights and up are no problem)
- Combines well with screen-space effects

## ■ Cons:

- High bandwidth required
  - Not applicable on older hardware
- Alpha blending hard to achieve
- Hardware multisampling not available

# Deferred Shading



**Picture: Stalker**

# Screen Space Ambient Occlusion (SSAO)

- Popularized by Crysis (Crytec)
- Render textures needed:
  - Depth (as linear z-buffer) or world space
  - Normals
- Approach:
  - Fragment analyses its surrounding
    - Fragment samples z-buffer around screen position
    - Simplest approach: depth difference of fragment and sample



# Screen Space Ambient Occlusion (SSAO)

## ■ Pros:

- Independent from scene complexity
- No preprocessing
- Dynamic scenes

## ■ Cons:

- Not correct
- Only evaluates what is seen
- Only close range shadowing
- Sampling artifacts (needs additional smoothing/blur)



# OGRE SSAO Demo

