

Lesson 6

HDR: Tone Mapping, Bloom effect

PV227 – GPU Rendering

Jiří Chmelík, Jan Čejka
Fakulta informatiky Masarykovy univerzity

23. 10. 2017

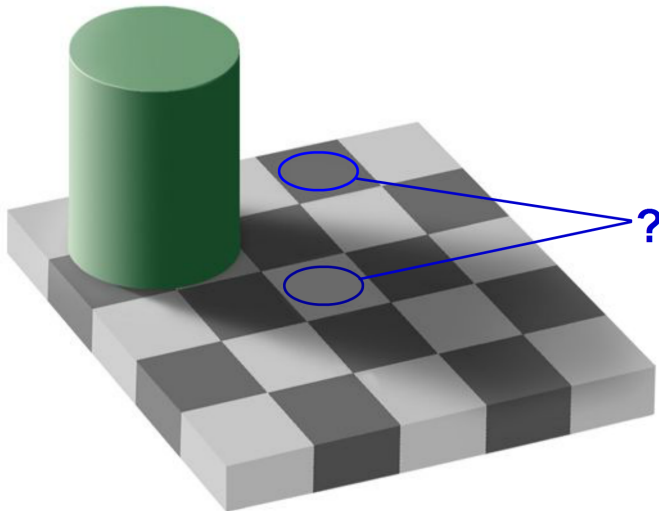
- High Dynamic Range
 - ▶ HDRI (“high dynamic range imaging”)
 - ▶ HDRR (“high dynamic range rendering”)
- Developed to make on-screen rendering more natural (human-eye like)
- Range of intensities:
 - ▶ software: often only 8 bits \Rightarrow only 256 steps
 - ▶ hardware: quite limited (black is not black, bright white is much less brighter than sun light)
 - ▶ human eye. . .

Human Eye – Range of perceptible intensities

illumination condition	illuminance (lux)
Full moon	1
Street lighting	10
Home lighting	30 to 300
Office desk lighting	100 to 1 000
Surgery lighting	10 000
Direct sunlight	100 000

Vast range of perceptible intensities. But not **concurrently!**

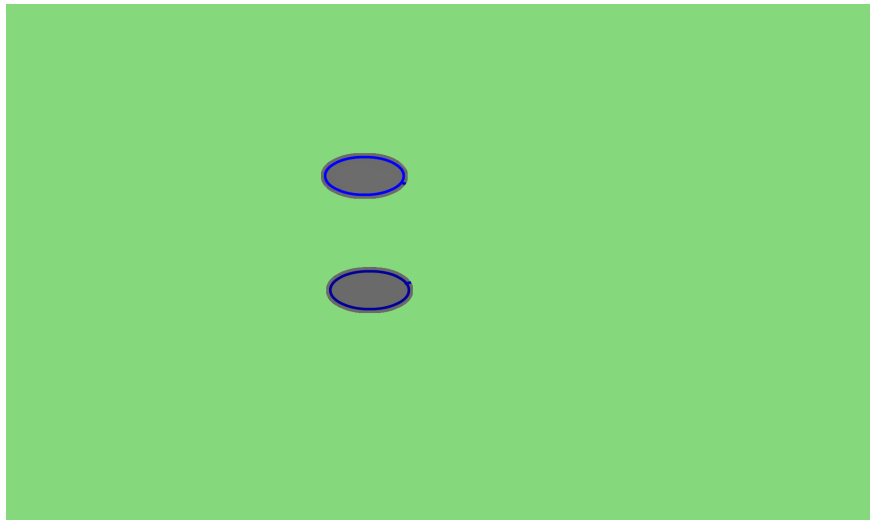
Human Eye Perception Imperfection



©1995 Edward H. Adelson

What are the colors of marked fields?

Human Eye Perception Imperfection



What are the colors of marked fields? The SAME!

Capturing, Rendering

- We can capture (or model) and “realistically” render both the dark scenes and the bright scenes without HDR.

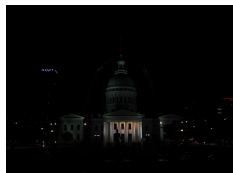


- The problem is with scenes with high dynamic range...

Capturing HDR Content

To capture HDR content with LDR camera:

- 1 capture more shots with different exposure settings
- 2 Compose final image with “tone mapping” technique



EV -4



EV -1



EV +1



EV +4

Source: https://en.wikipedia.org/wiki/High-dynamic-range_imaging

Rendering HDR Content

Various methods to convert HDR content into LDR image exists:



Contrast reduction



Local tone mapping

- Bloom (blooming, glow) – “overflowing” of light to surrounding objects
- Other buffers, techniques:
 - ▶ light maps,
 - ▶ skybox,
 - ▶ ...
- Advanced technique: “Temporally Coherent Local Tone Mapping”:
<https://youtu.be/6yItM8UB7k4>

Implementation of HDR Rendering

- Common color buffer format: **R8G8B8**
 - ▶ Don't forget: values are clamped to range $< 0.0, 1.0 >$
- We need high dynamic range buffer
 - ▶ We can simply switch to **R16G16B16A16F**
 - ▶ No clamping of values

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA16F, win_width,  
win_height, 0, GL_RGBA, GL_FLOAT, nullptr);
```

HDR Tone Mapping – Overview

- Set-up HDR buffer (TASK 1)
- First pass: compute lighting of scene into HDR buffer
 - ▶ The SAME computations, just no clamping of values
- Second pass: use one of algorithms to tone map HDR buffer to (LDR) frame-buffer (TASK 2)
- Following passes: reuse existing HDR buffer to other effects. . .

Tone Mapping – “Reinhard” technique

Among the simplest tone mapping techniques – simple remapping of HDR values to range $< 0.0, 1.0 >$ by division.

```
// read color from HDR texture
vec3 hdrColor = texture(hdrBuffer, TexCoords).rgb;

// simple reinhard mapping
vec3 result = hdrColor / (hdrColor + vec3(1.0));
```


Tone Mapping – “Adjustable Exposure” technique

Allow us to render scene with different exposures:

$$rgb = 1 - 2^{-hdr * exposure}$$

```
// read color from HDR texture
vec3 hdrColor = texture(hdrBuffer, TexCoords).rgb;

// Exposure tone mapping
vec3 mapped = vec3(1.0) - exp2(-hdrColor * exposure);
```

Tone Mapping – Gamma Correction

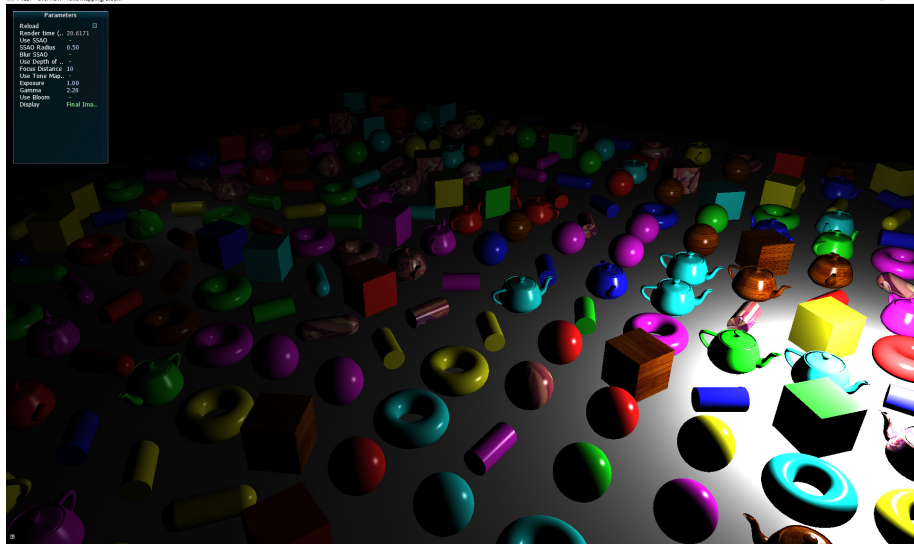
Could be combined with all tone mapping techniques.

```
// read from texture
// Do a tone mapping
vec3 mapped = ...

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

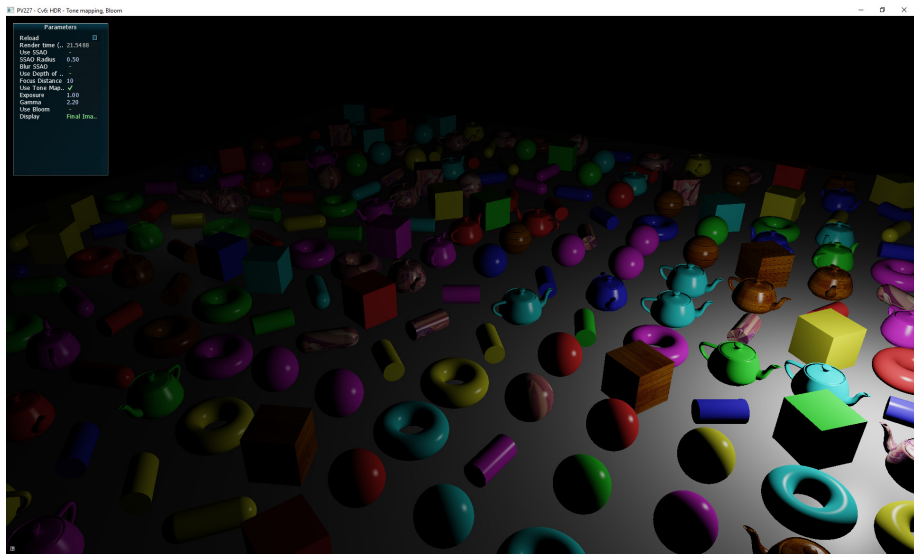
Tone Mapping – Results

PV227 - C++ HDR - Tone mapping, Bloom



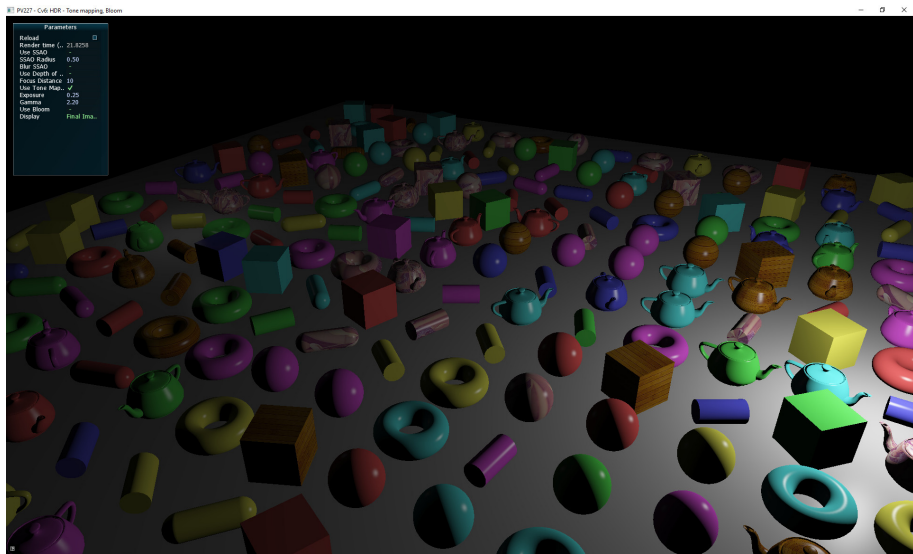
Tone mapping: none

Tone Mapping – Results



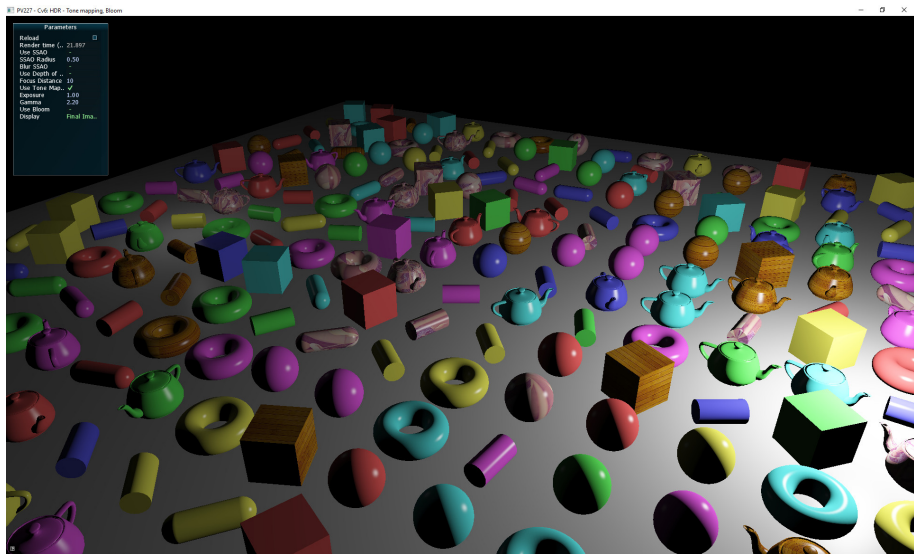
Tone mapping: “Reinhard” technique

Tone Mapping – Results



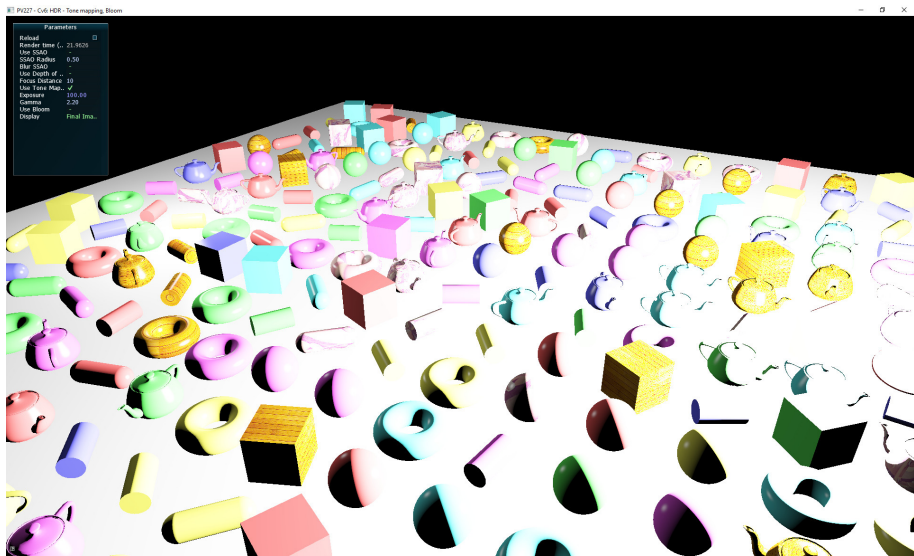
Tone mapping with exposure: 0.25

Tone Mapping – Results



Tone mapping with exposure: 1.0

Tone Mapping – Results



Tone mapping with exposure: 100

Bloom Effect (Blooming, Glow)

- Imperfection of human eye (or camera sensor) which is overwhelmed by bright light. Light is “overflowing” to surrounding cells (pixels).
- In CG – added artificially to increase realism.
- Bloom can be used also in LDR, but with HDR make more sense.



Figure Source: <http://www.nationalgeographic.com/photography/photo-tips/overexposure-on-purpose-richardson/>

Bloom Effect – CG example



Source: <http://learnopengl.com/#!Advanced-Lighting/Bloom>

Bloom Effect – Implementation Overview

- ① Lit the scene (as always)
 - ② Fill buffer of pixels with high brightness (highlights) (TASK 3)
 - ③ Blur the highlights buffer to simulate glow effect (TASK 4)
 - ④ Compose original rendering with blurred highlights (TASK 5)
-
- How many passes we need?
 - Do we need deferred shading? Can we exploit it? How?
 - Where is HDR in this?

Bloom – Highlights Filtering (TASK3)

- In what shader?
- Compare fragment brightness to some threshold
 - ▶ We are working with HDR buffers, threshold could be simply 1.0f
 - ▶ Hint: brightness of pixel... `vec3(0.299f, 0.587f, 0.114f)`

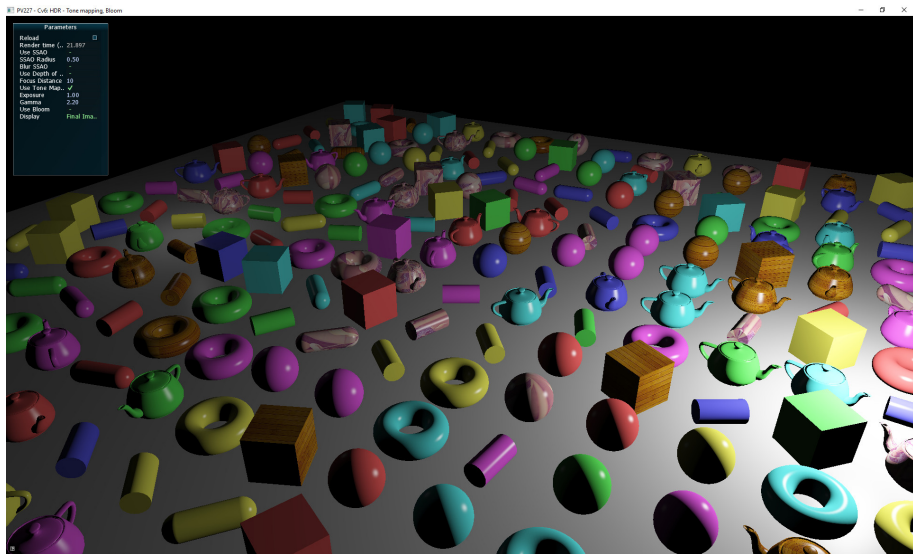
Blurring shader

- What is difference from `blur_SSAO` shader?
- Blur possibilities:
 - ▶ Simple average
 - ▶ Gaussian
 - ▶ Repeated blurring – later
 - ▶ Separable kernels
 - ▶ ...
- Think about effectiveness

Could be complicated in LDR, pretty easy with HDR buffers:

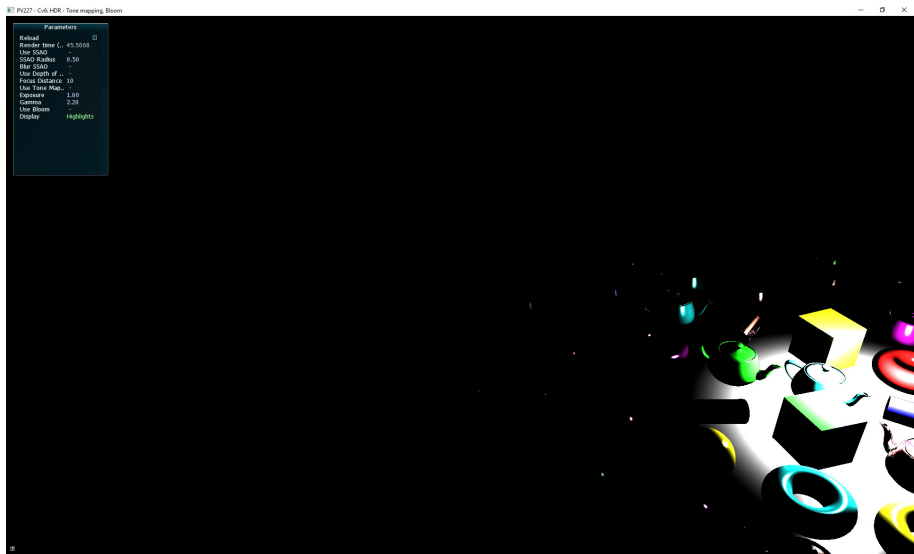
- ① Simply add values of lit scene and blurred highlights
- ② Use tone mapping as before

Bloom – Results



Rendering without bloom

Bloom – Results



Debug: highlights buffer

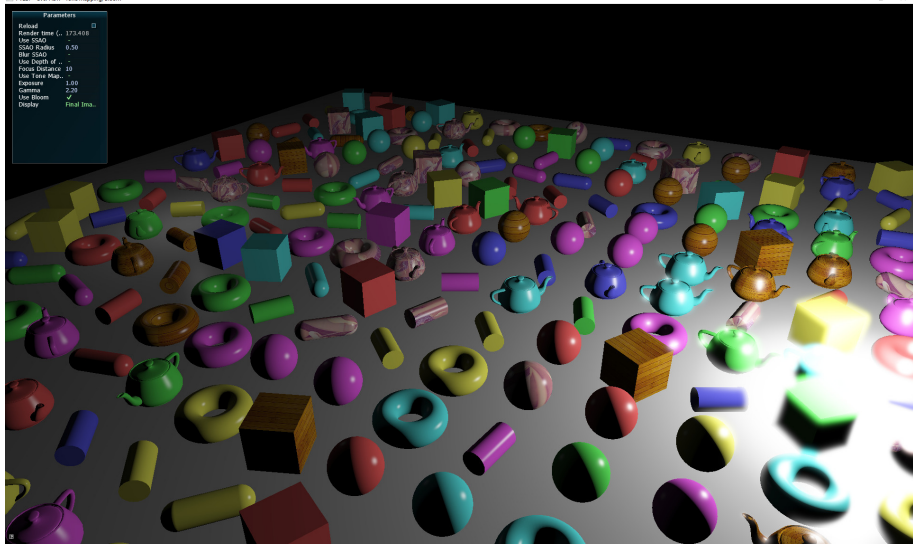
Bloom – Results



Debug: blurred buffer

Bloom – Results

PV227 - C46 HDR - Tone mapping, Bloom



Final image with bloom effect

What Next?

- How to combine effects?
 - ▶ We can have: SSAO, HDR (Tone mapping + Bloom), DoF, Grain, Flares, etc.
 - ▶ Lots of buffers needed – lots of memory needed
 - ▶ Lots of “logic” needed to do it effectively

- Post-effect double buffering trick:
 - ▶ For combining various post-process effects
 - ▶ “Double buffer” for blurring – bonus task (TASK4b)
 - ★ Is it faster than separable kernels blurring?

Assignment 1

- John Hable: *Uncharted 2: HDR lighting*
 - ▶ Extensive description of HDR in context of AAA game:
 - ★ gamma,
 - ★ linear space,
 - ★ filmic tone mapping,
 - ★ A lot more
 - ▶ <http://www.slideshare.net/ozlael/hable-john-uncharted2-hdr-lighting>
- About convolution computing
https://cg.ivd.kit.edu/downloads/GPUComputing_assignment_3.pdf