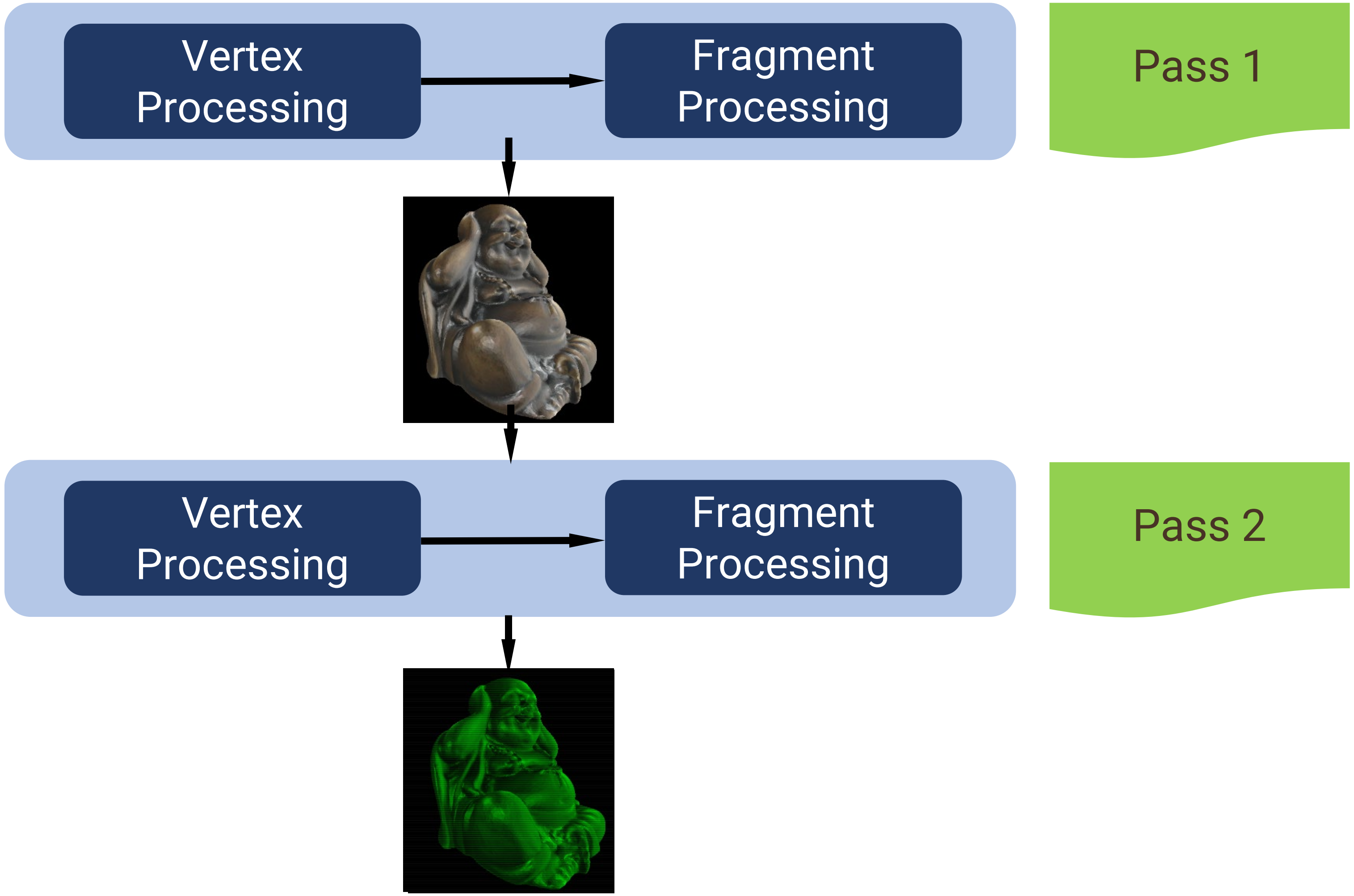
Postprocessing / GPU-based Image Processing

David Schedl

david.schedl@fh-hagenberg.at

Overall goal



Postprocessing

- Download: 09-postpro-deferred.zip
- Let’s get some examples working: 09a-postprocessing
- We only need to edit the shaders
  (textures are already bound, uniforms already set)

First, we define four shaders (2 programs)

```
Shader modelShader("model.vs.glsl", "model.fs.glsl");
Model myModel("objects/buddha2/buddha2.obj");
modelShader.use();

Shader screenShader(“screenshader.vs.glsl", “screenshader.fs.glsl");
screenShader.use();
screenShader.setInt("screenTexture", 0);
```

Creating a Framebuffer Object

```
unsigned int framebuffer;
glGenFramebuffers(1, &framebuffer);
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);

// create a color attachment texture
unsigned int textureColorBuffer;
glGenTextures(1, &textureColorBuffer);
glBindTexture(GL_TEXTURE_2D, textureColorBuffer);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, WIDTH, HEIGHT, 0, GL_RGB,
             GL_UNSIGNED_BYTE, NULL);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
                       GL_TEXTURE_2D, textureColorBuffer, 0);
```

Creating a Framebuffer Object

```
unsigned int framebuffer;
glGenFramebuffers(1, &framebuffer);
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);

// create a color attachment texture
unsigned int textureColorBuffer;
glGenTextures(1, &textureColorBuffer);
glBindTexture(GL_TEXTURE_2D, textureColorBuffer);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, WIDTH, HEIGHT, 0, GL_RGB,
             GL_UNSIGNED_BYTE, NULL);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
                       GL_TEXTURE_2D, textureColorBuffer, 0);
```

Creating a Framebuffer Object

```
unsigned int framebuffer;
glGenFramebuffers(1, &framebuffer);
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);

// create a color attachment texture
unsigned int textureColorBuffer;
glGenTextures(1, &textureColorBuffer);
glBindTexture(GL_TEXTURE_2D, textureColorBuffer);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, WIDTH, HEIGHT, 0, GL_RGB,
             GL_UNSIGNED_BYTE, NULL);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
                       GL_TEXTURE_2D, textureColorBuffer, 0);
```

Creating a Framebuffer Object

```
unsigned int framebuffer;
glGenFramebuffers(1, &framebuffer);
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);

// create a color attachment texture
unsigned int textureColorBuffer;
glGenTextures(1, &textureColorBuffer);
glBindTexture(GL_TEXTURE_2D, textureColorBuffer);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, WIDTH, HEIGHT, 0, GL_RGB,
             GL_UNSIGNED_BYTE, NULL);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0,
                       GL_TEXTURE_2D, textureColorBuffer, 0);
```

Creating a Framebuffer Object

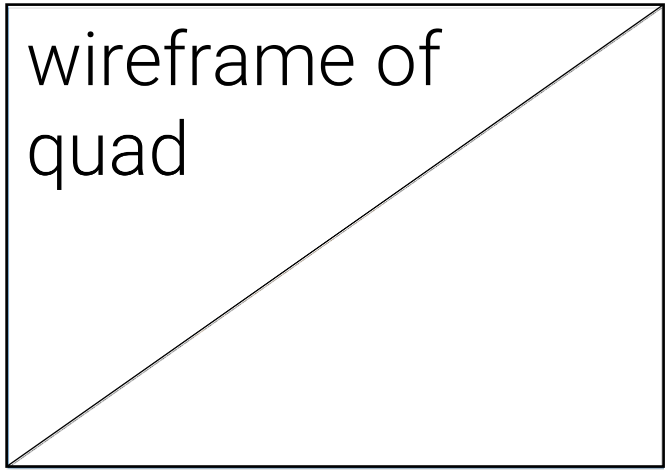
```
// create a renderbuffer object for depth and stencil attachment
unsigned int rbo;
glGenRenderbuffers(1, &rbo);
glBindRenderbuffer(GL_RENDERBUFFER, rbo);
glRenderbufferStorage(GL_RENDERBUFFER, GL_DEPTH24_STENCIL8, WIDTH, HEIGHT);

glFramebufferRenderbuffer(GL_FRAMEBUFFER, GL_DEPTH_STENCIL_ATTACHMENT,
                          GL_RENDERBUFFER, rbo);
```

In the Loop

```
...
processInput(window);

...
// bind to framebuffer and draw scene as we normally would to color texture
glBindFramebuffer(GL_FRAMEBUFFER, framebuffer);
glEnable(GL_DEPTH_TEST);

glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

mat4 model = mat4(1.0f);
model = translate(model, vec3(0.0f, -0.25f, 0.0f));
model = scale(model, vec3(0.2f, 0.2f, 0.2f));

modelShader.use();
```

In the Loop (2)

```
...

modelShader.use();
myModel.Draw(modelShader);

screenShader.use();
// use the color attachment texture as the texture of the quad plane
glBindTexture(GL_TEXTURE_2D, textureColorbuffer);
renderQuad();
```

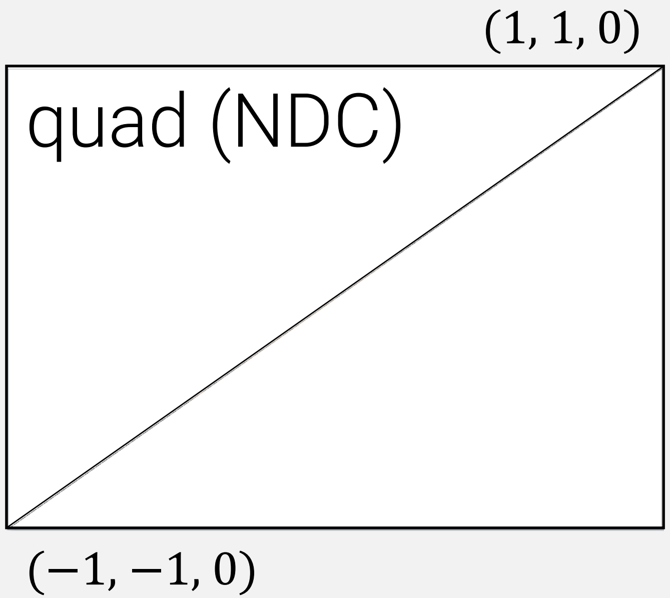


wireframe of quad

The Screen-Shader (VS)

screenshader.vs.glsl

```
#version 330 core
layout (location = 0) in vec2 aPos;
layout (location = 1) in vec2 aTexCoords;

out vec2 TexCoords;

void main()
{
    gl_Position = vec4(aPos.x, aPos.y, 0.0, 1.0);
    TexCoords = aTexCoords;
}
```



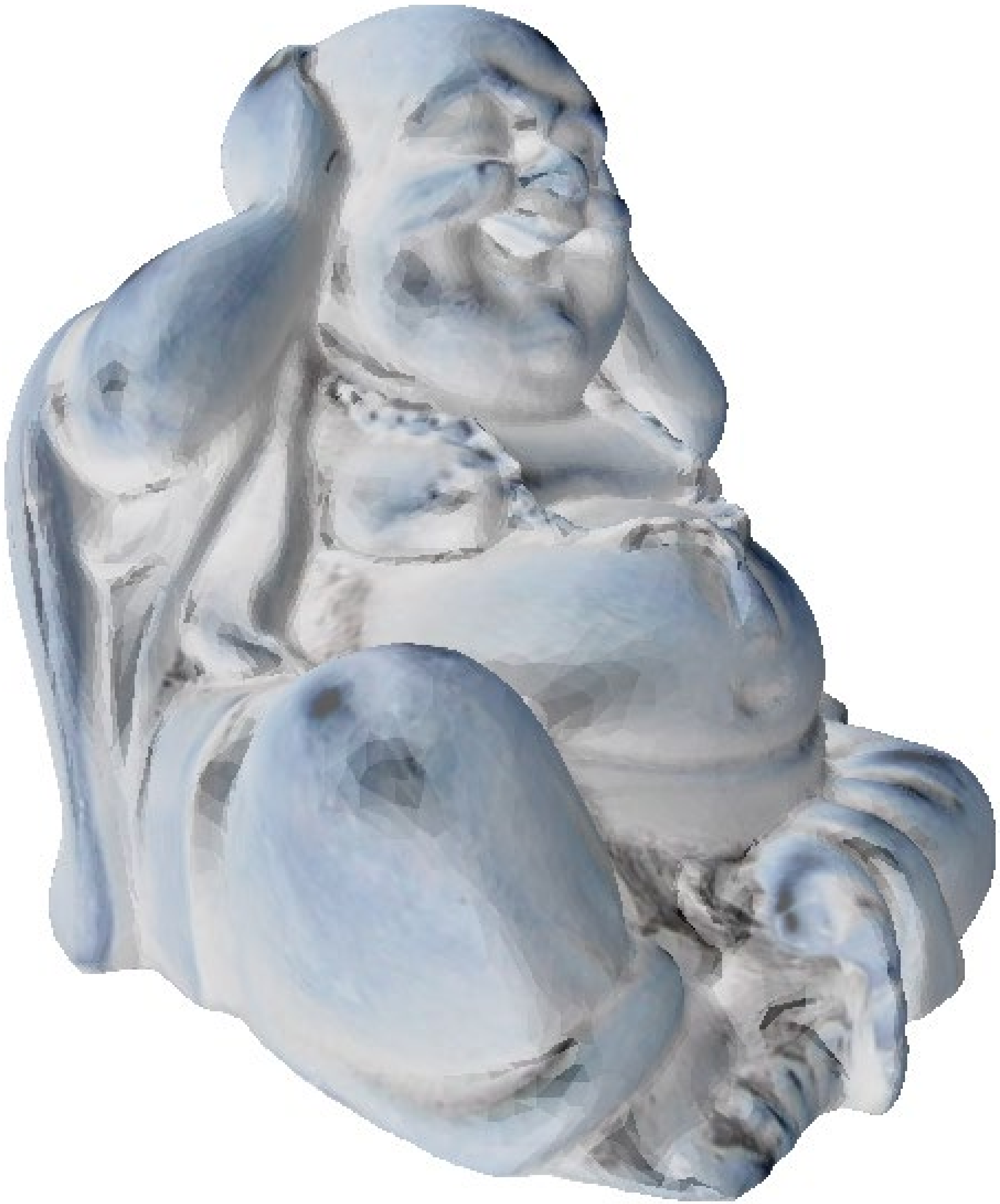
- We’ll reuse that vertex shader for multiple examples!

The Screen-Shader

screenshader.fs.glsl

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

uniform sampler2D screenTexture;

void main()
{
    FragColor = texture(screenTexture, TexCoords);
}
```

Invert Color Shader



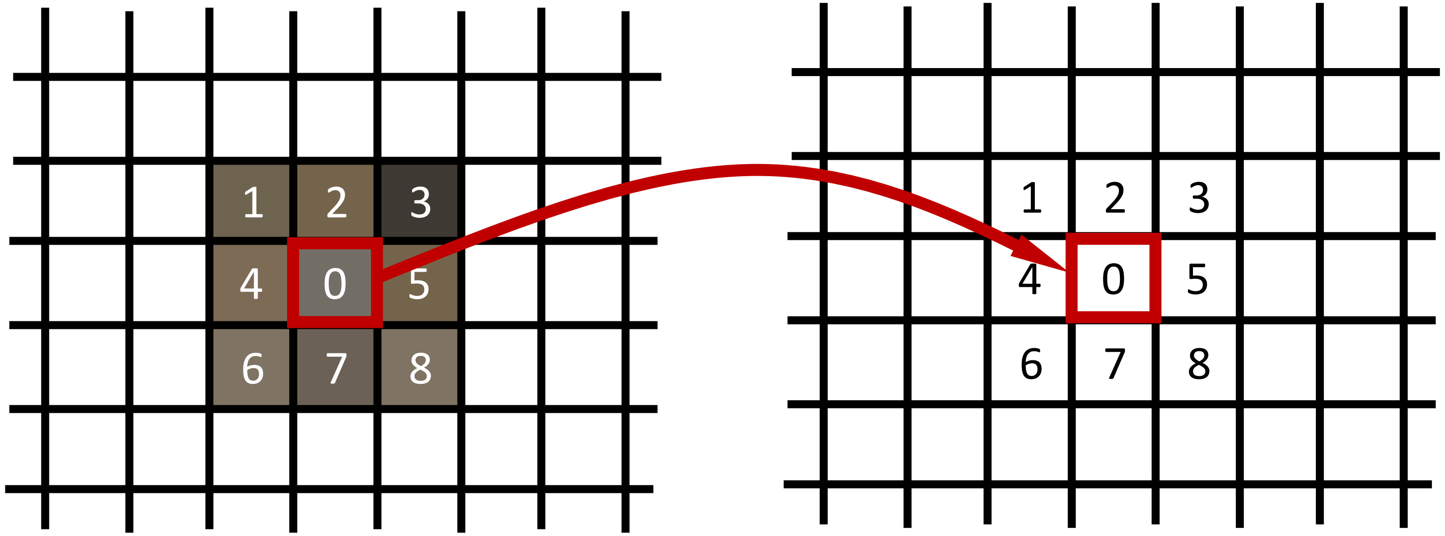
Invert-Color Shader

invertcolor.fs.glsl

```glsl
#version 330 core
out vec4 FragColor;

in vec2 TexCoords;

uniform sampler2D screenTexture;


void main()
{
    FragColor = vec4( vec3(1.0) - texture(screenTexture, TexCoords).rgb, 1.0);
}
```

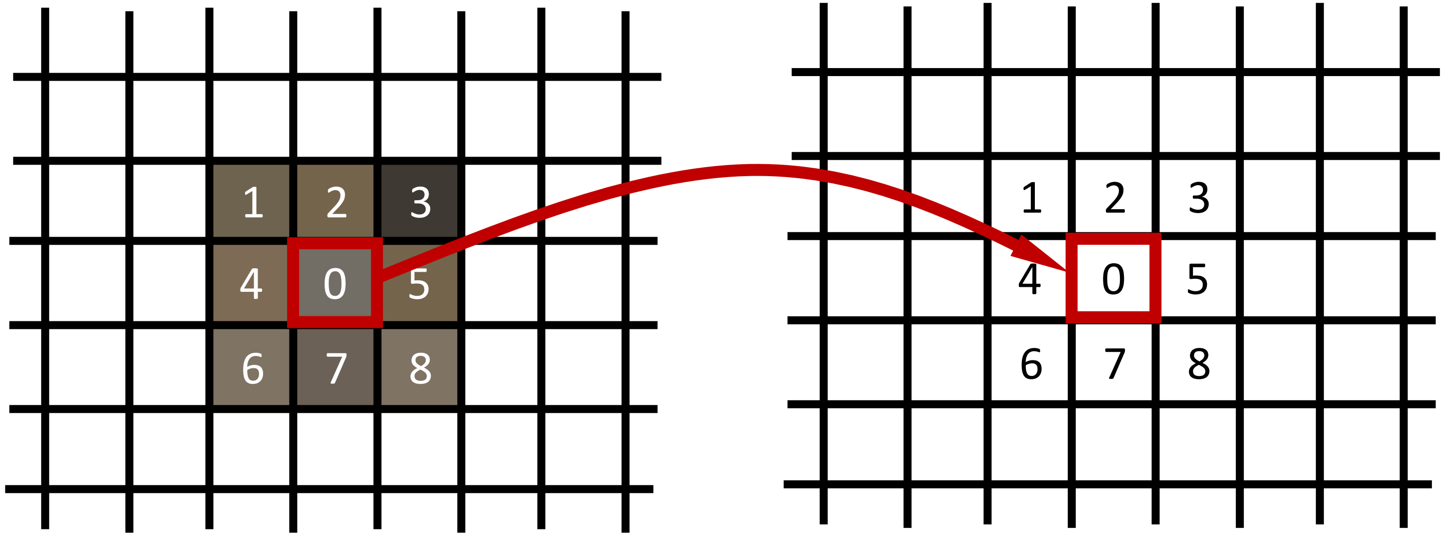
Filters

How to filter an image?



$$
\begin{aligned}
I'(s,t) = \frac{1}{9}[& I(s-1,t-1) + I(s,t-1) + I(s+1,t-1) + \\
& I(s-1,t) \quad + I(s,t) \quad + I(s+1,t) \quad + \\
& I(s-1,t+1) + I(s,t+1) + I(s+1,t+1)]
\end{aligned}
$$

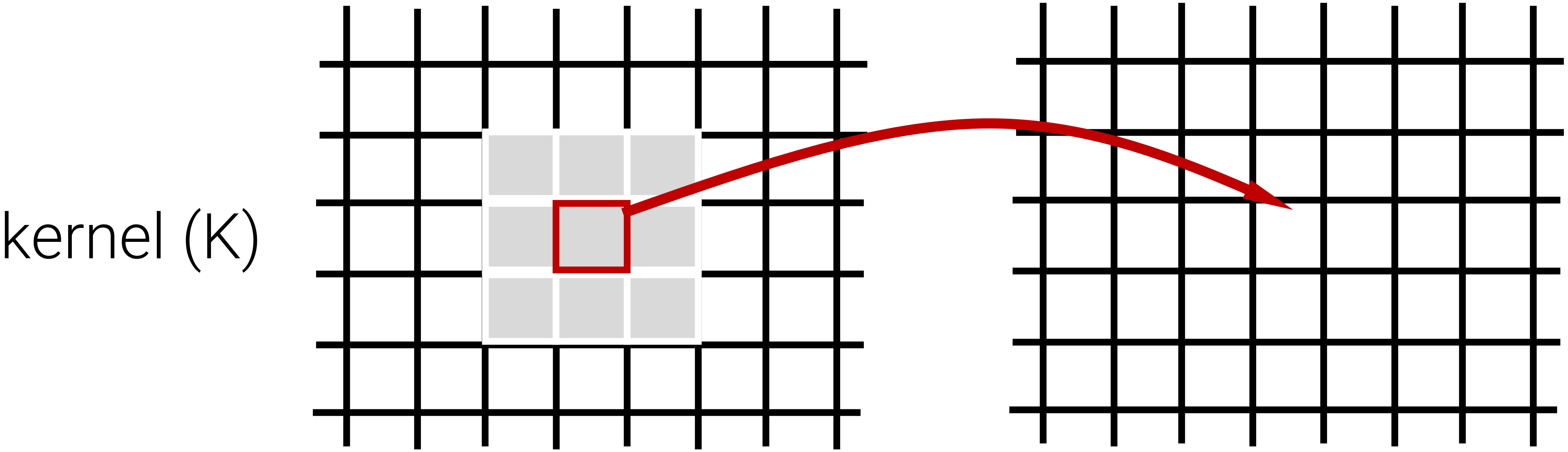
How to filter an image?



$$I'(s,t) = \frac{1}{9} \sum_{j=-1}^{1} \sum_{i=-1}^{1} I(s+i, t+j)$$

Using a filter matrix (kernel)

- New pixel value is a weighted sum (linear combination) of the original pixel values
- Filter matrix or “kernel” = two-dimensional function of weights (coefficients)
- Convolution matrix

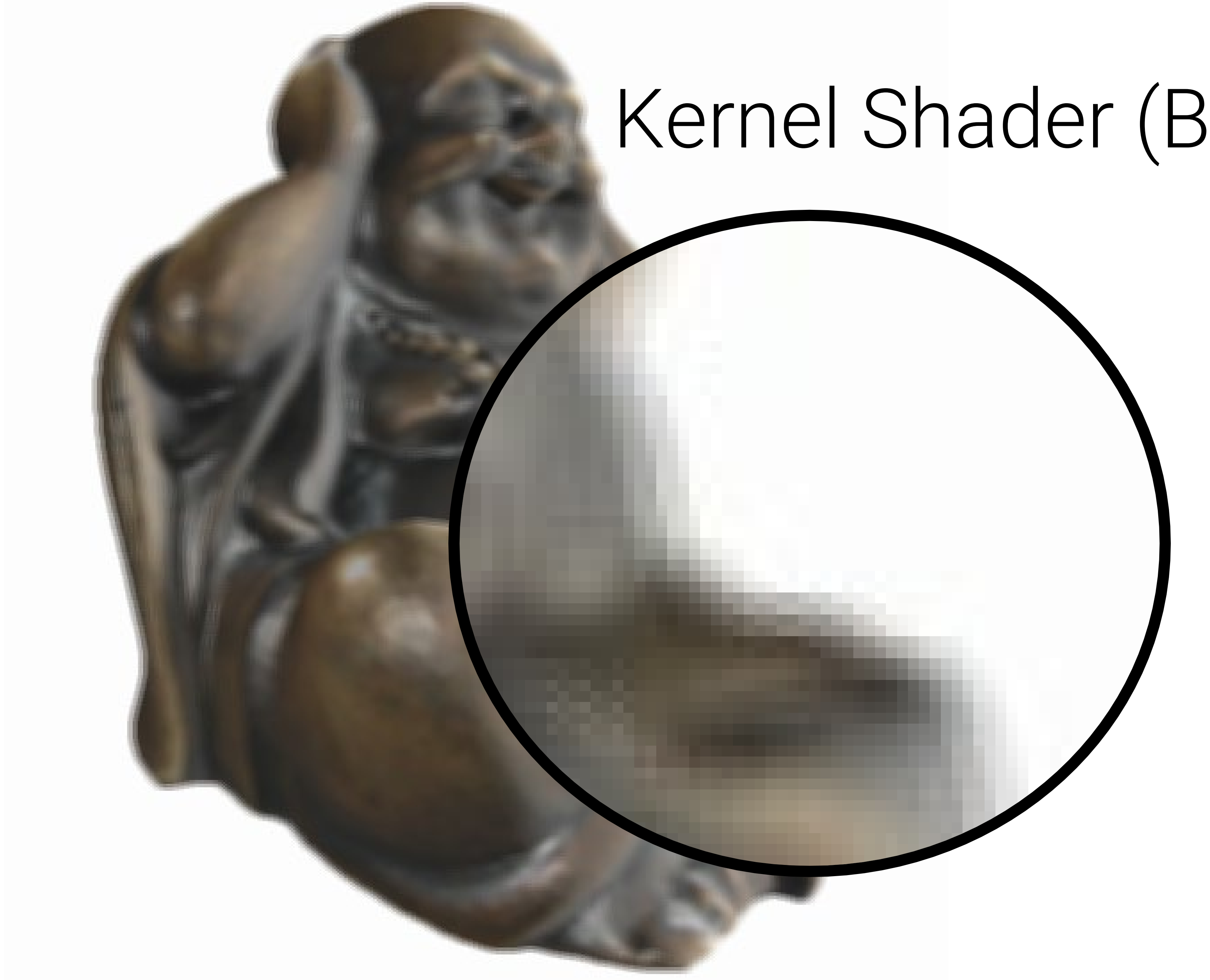


Using a filter matrix (kernel) in action

- The kernel is moved over the original image such that its origin coincides with the current image position *(s, t)*.
- All filter/kernel coefficients *H(i, j)* are multiplied with the corresponding image element *I(u+i, v+j)*, and summed.

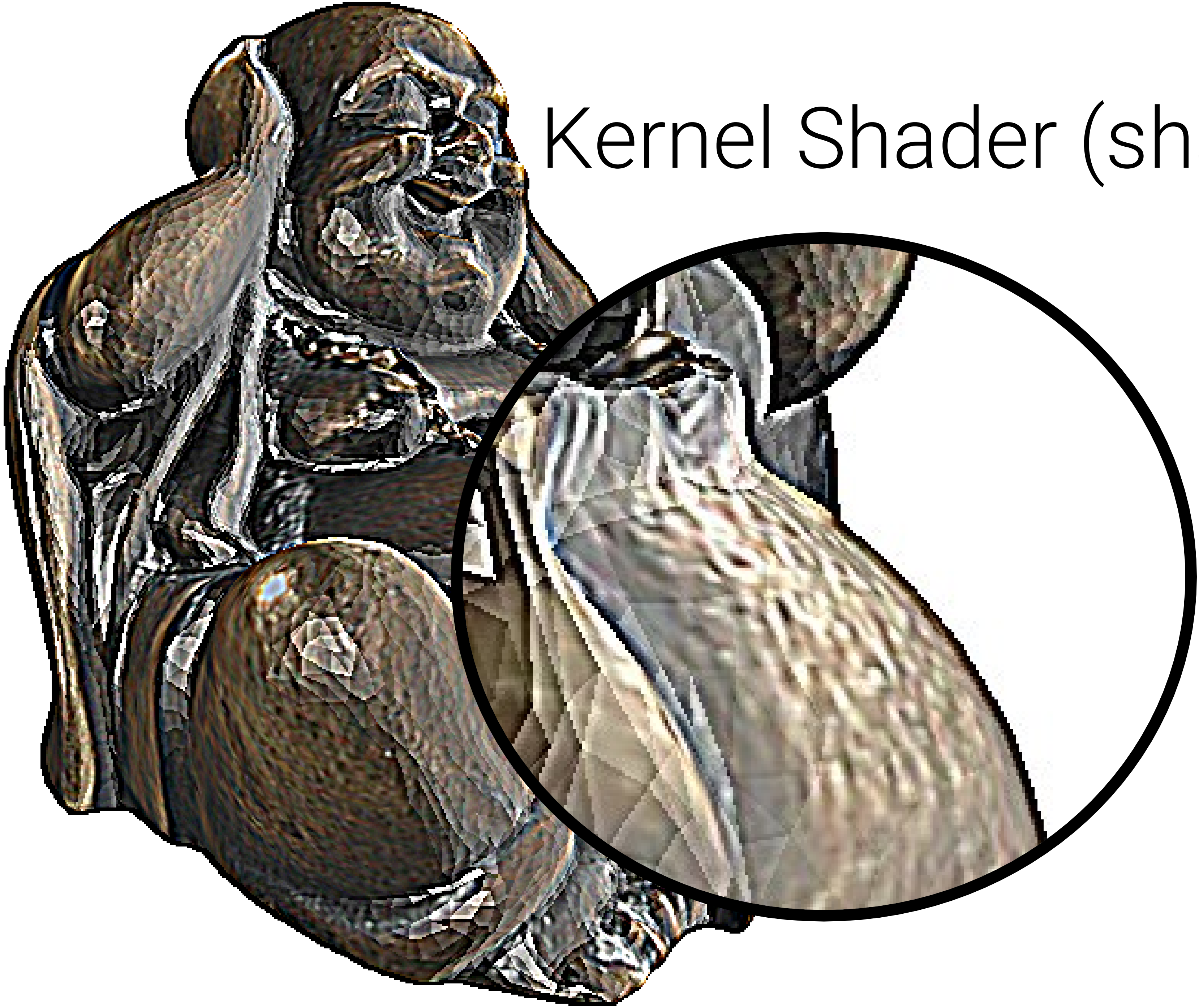
$$I'(s,t) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} I(s+i, t+j) \cdot H(i,j)$$

Kernel Shader (Blur) - results



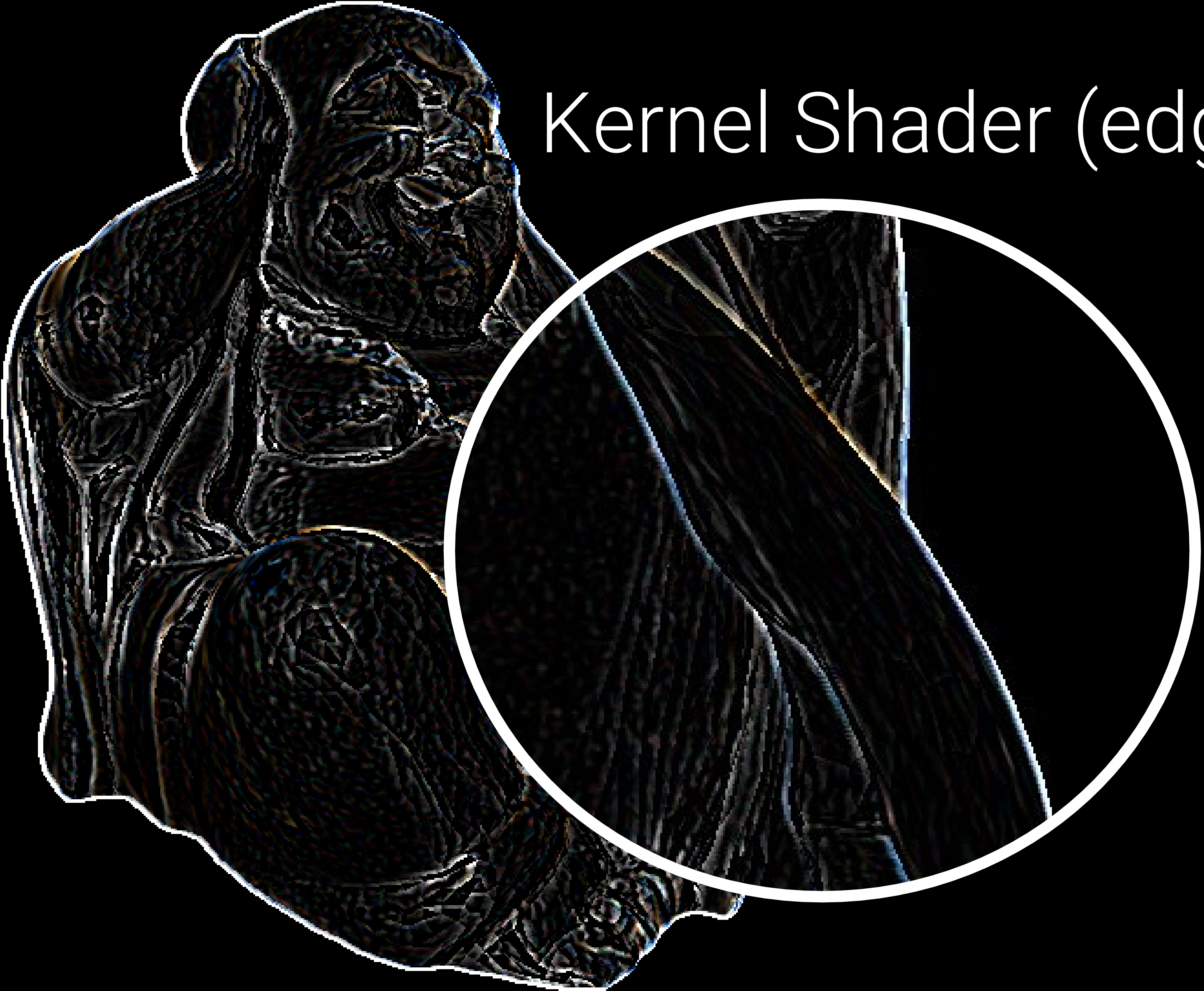
$$H = \begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix}$$

Kernel Shader (sharpen) - results



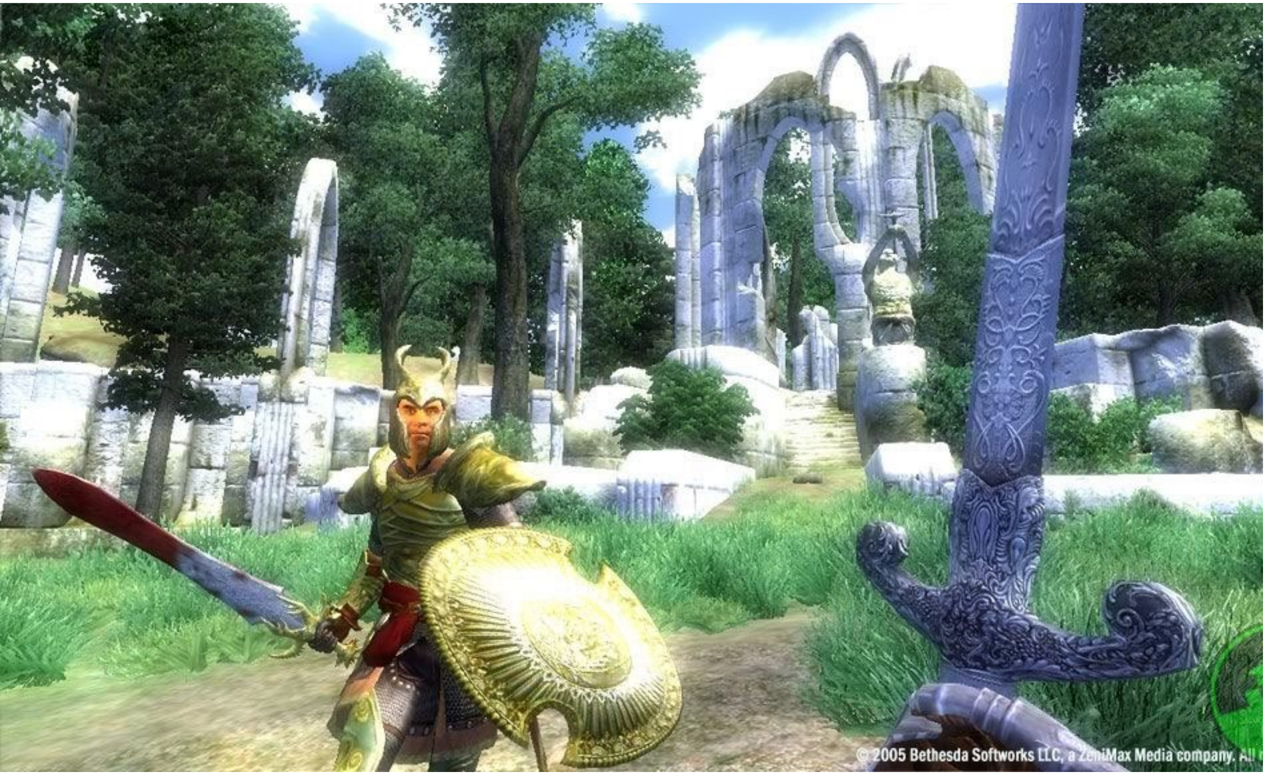
$$H = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Kernel Shader (edge) - results

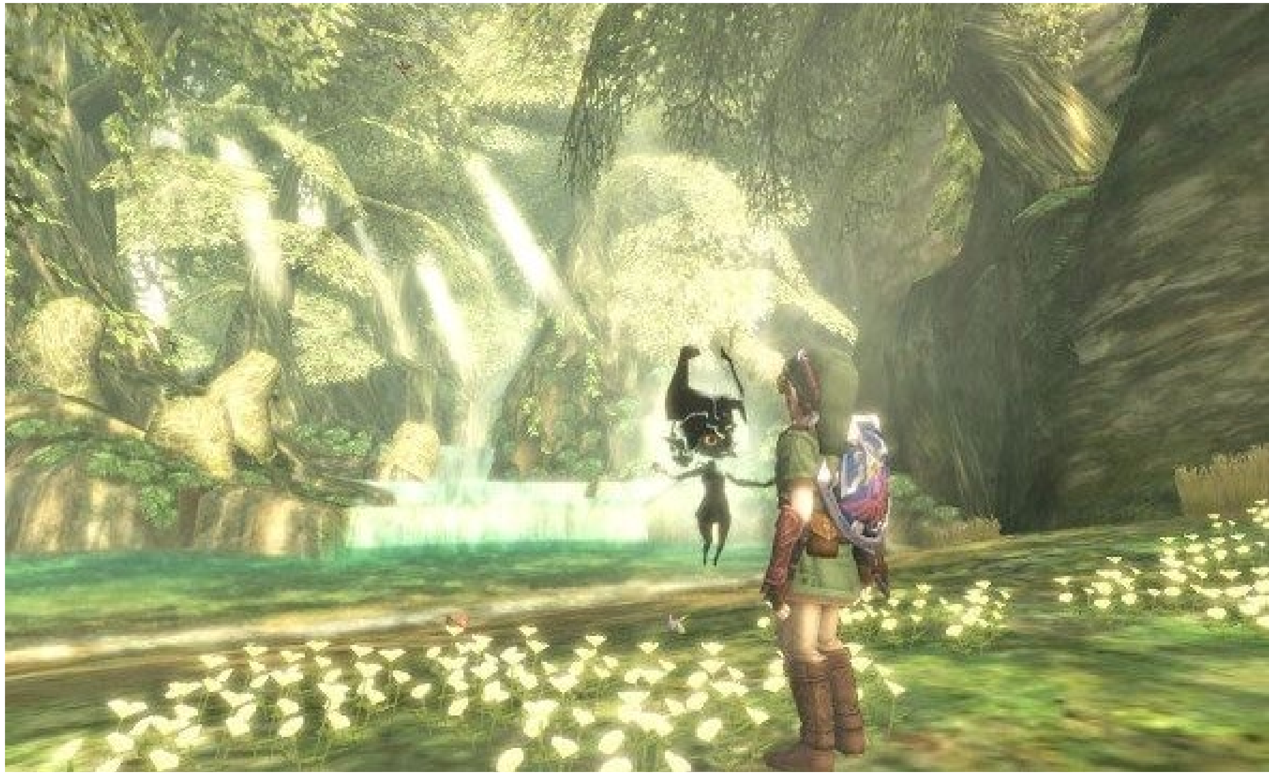


$$H = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Postprocessing: Bloom

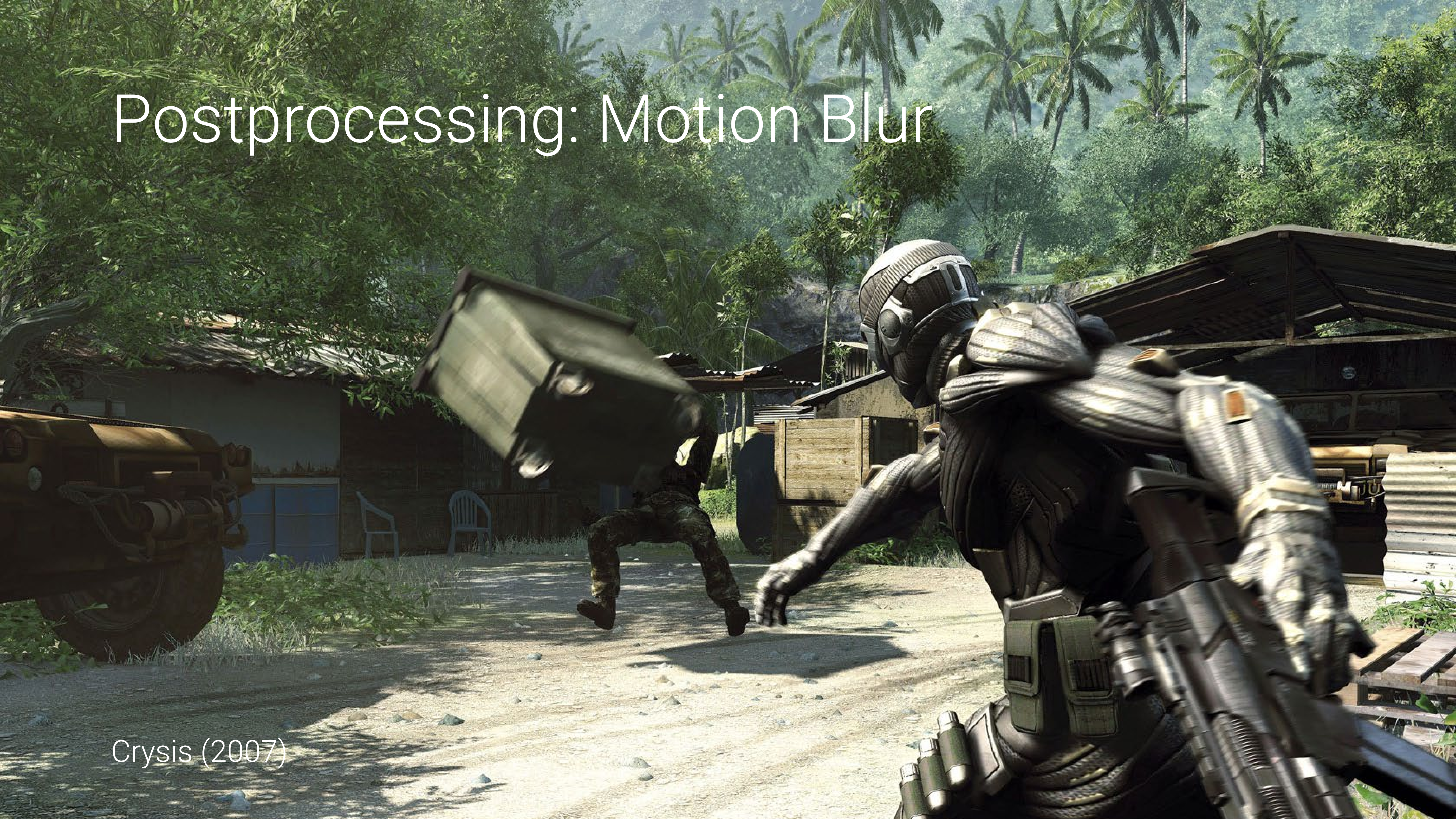


Oblivion



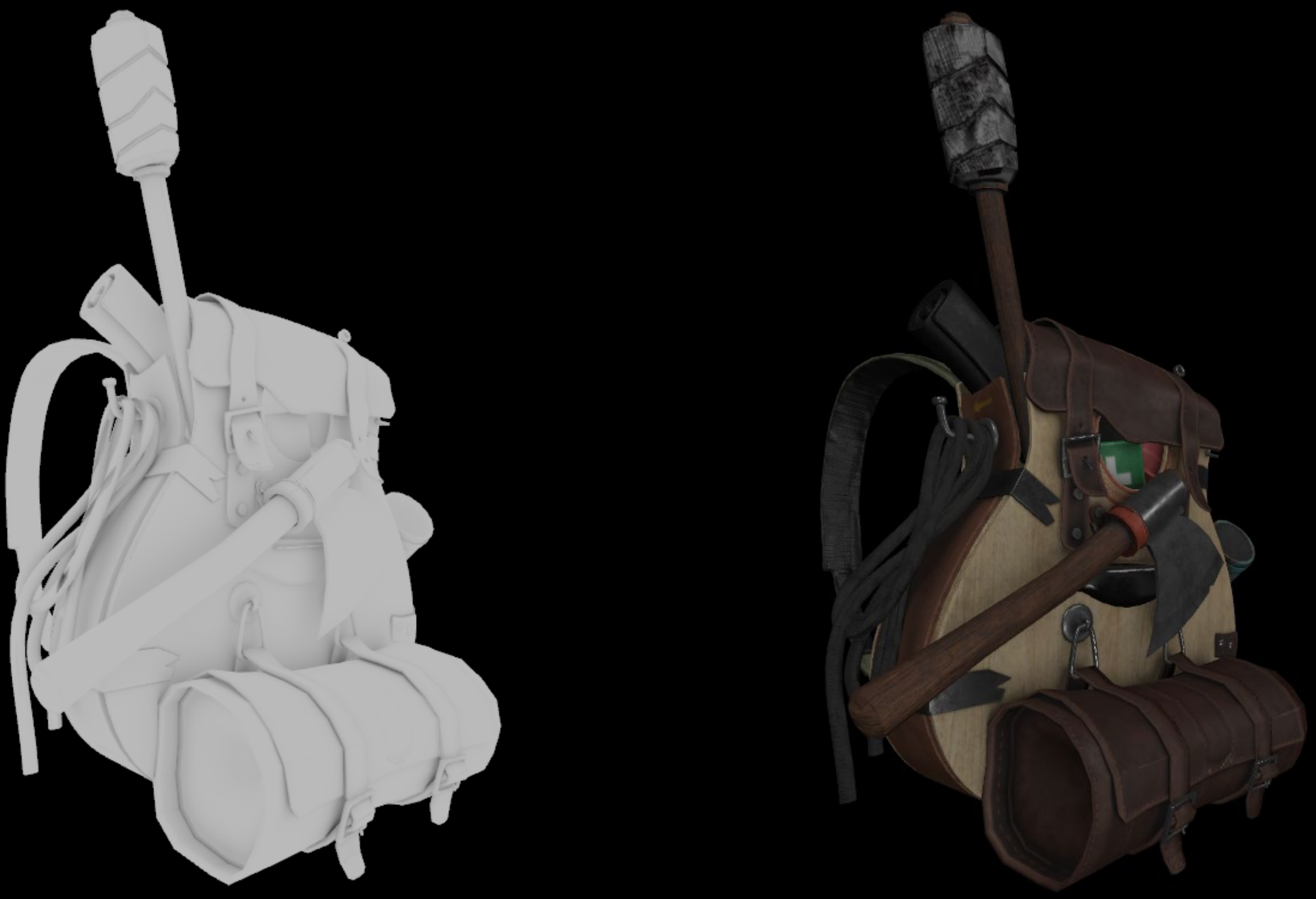
Zelda Twilight Princess

Postprocessing: Motion Blur



Crysis (2007)

Screen-spaced ambient occlusion



and many more ...

GTA V (2013)

Questions?

David Schedl

david.schedl@fh-hagenberg.at