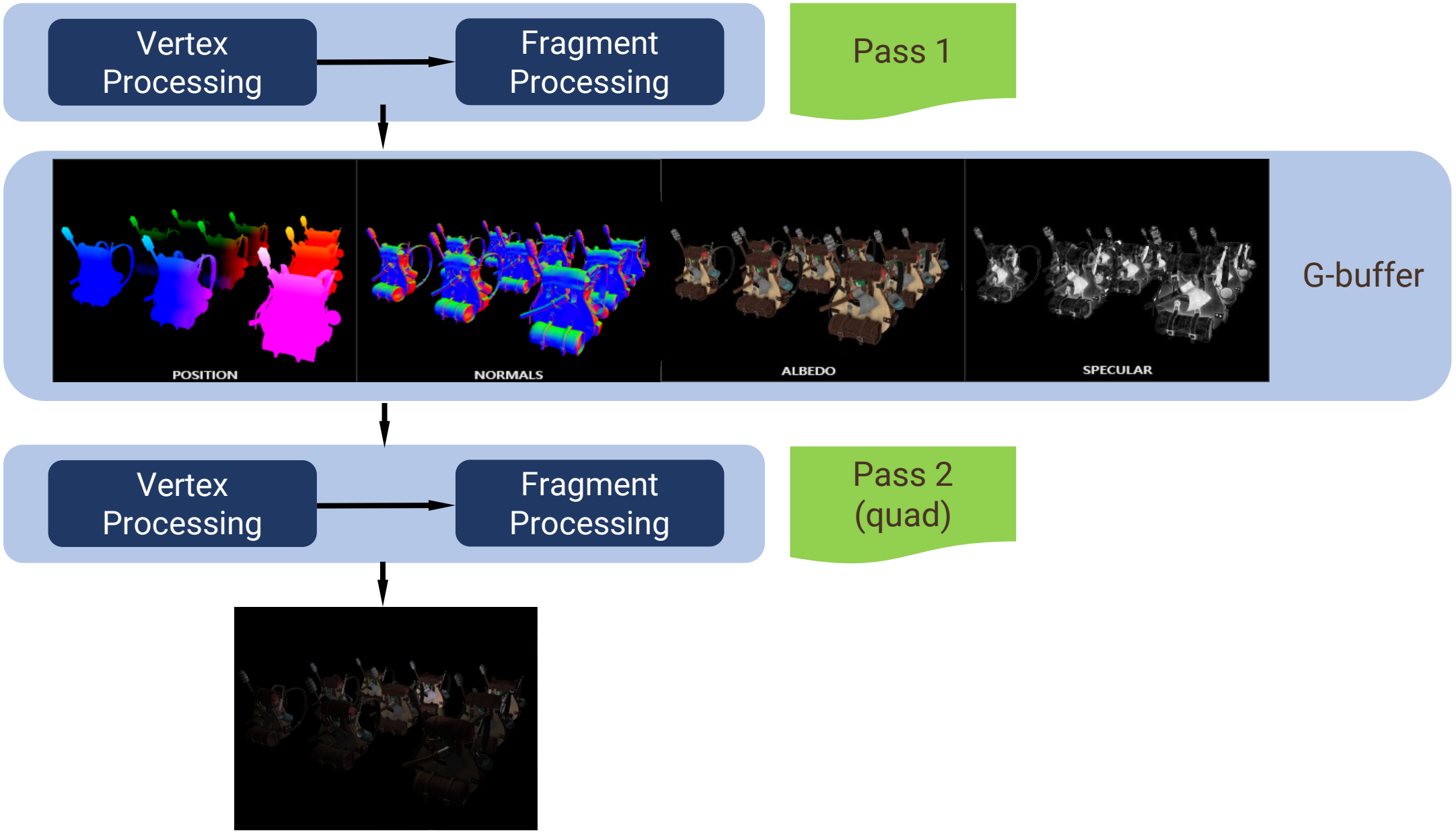
Postprocessing Part 2 - Deferred Shading

David Schedl

david.schedl@fh-hagenberg.at

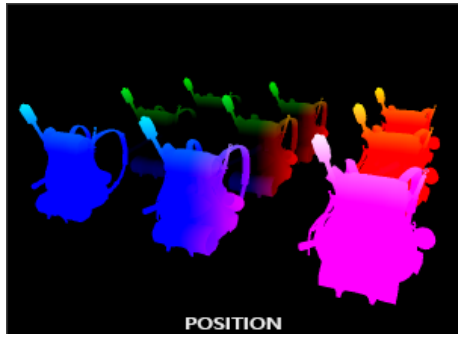
Deferred Shading

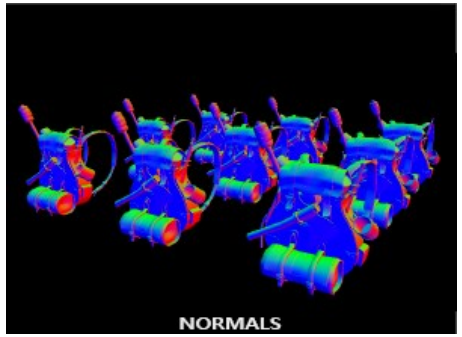
- Main Idea: Treat shading as a postprocess
  - Needs 2 passes
- For each pixel, store surface data require for lighting computation
  - Rendered textures: position, normal, material, …
- 2nd pass: Use pixel/fragment shaders for lighting computation
  - Use textures as input
  - Good for:
    - complex shaders
    - many (small) light sources

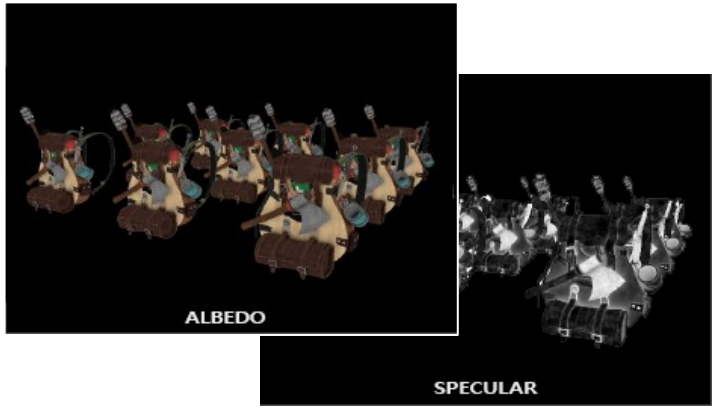

Vertex Processing
Fragment Processing
Pass 1
POSITION
NORMALS
ALBEDO
SPECULAR
G-buffer
Vertex Processing
Fragment Processing
Pass 2 (quad)

The G-Buffer

- Framebuffer with multiple color buffers and a single depth buffer
- Store properties needed for shading:
  - Position (16-bit float)
  - Normal (16-bit float)
  - Combined albedo (RGB) & specular (alpha)
    (using 8-bit precision)





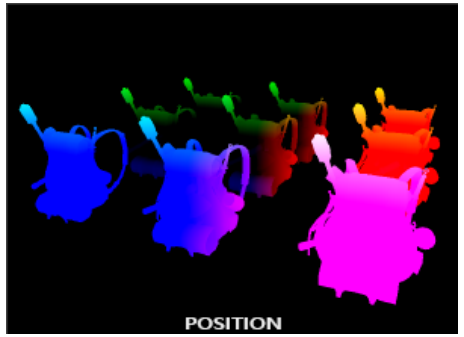


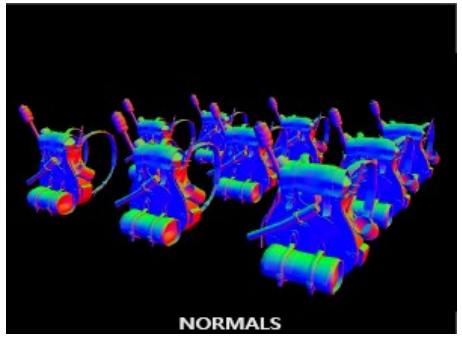
The G-Buffer

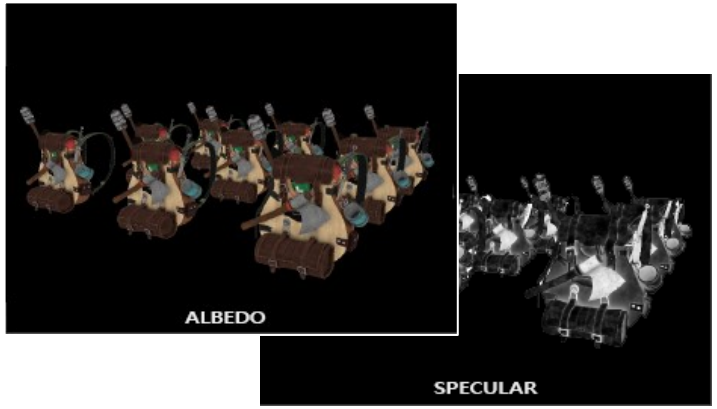
```
unsigned int gBuffer;
glGenFramebuffers(1, &gBuffer);
glBindFramebuffer(GL_FRAMEBUFFER, gBuffer);
unsigned int gPosition, gNormal, gAlbedoSpec;
// position color buffer
glGenTextures(1, &gPosition);
glBindTexture(GL_TEXTURE_2D, gPosition);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA16F, SCR_WIDTH, SCR_HEIGHT, 0, GL_RGBA, GL_FLOAT, NULL);
…
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, GL_TEXTURE_2D, gPosition, 0);
// normal color buffer
glGenTextures(1, &gNormal);
glBindTexture(GL_TEXTURE_2D, gNormal);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA16F, SCR_WIDTH, SCR_HEIGHT, 0, GL_RGBA, GL_FLOAT, NULL);
…
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT1, GL_TEXTURE_2D, gNormal, 0);
// color + specular color buffer
glGenTextures(1, &gAlbedoSpec);
glBindTexture(GL_TEXTURE_2D, gAlbedoSpec);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, SCR_WIDTH, SCR_HEIGHT, 0, GL_RGBA, GL_UNSIGNED_BYTE, NULL);
…
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT2, GL_TEXTURE_2D, gAlbedoSpec, 0);
```

TASK: The Shaders

- Download: 09-postrpo-deferred.zip
- Let's get deferred shading working in `09b-deferred`
- We only need to edit the shaders
  (textures are already bound, uniforms already set)







The G-Buffer Shader

g_buffer.fs

```
#version 330 core
layout (location = 0) out vec3 gPosition;
layout (location = 1) out vec3 gNormal;
layout (location = 2) out vec4 gAlbedoSpec;

in vec2 TexCoords;
in vec3 FragPos;
in vec3 Normal;

uniform sampler2D texture_diffuse1;
uniform sampler2D texture_specular1;

void main(){
    gPosition = FragPos;
    gNormal = normalize(Normal);
    gAlbedoSpec.rgb = texture(texture_diffuse1, TexCoords).rgb;
    gAlbedoSpec.a = texture(texture_specular1, TexCoords).r;
}
```

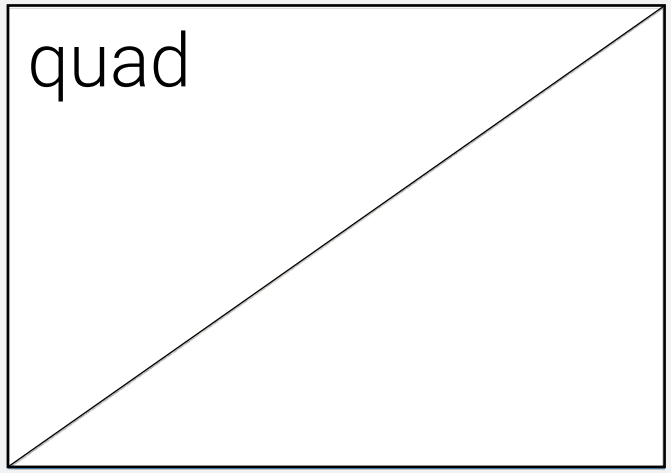
2nd Pass

```
shaderLightingPass.use();
shaderLightingPass.setInt("gPosition", 0);
shaderLightingPass.setInt("gNormal", 1);
shaderLightingPass.setInt("gAlbedoSpec", 2);

shaderLightingPass.use();
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, gPosition);
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, gNormal);
glActiveTexture(GL_TEXTURE2);
glBindTexture(GL_TEXTURE_2D, gAlbedoSpec);

…

renderQuad();
```



The Lighting Shader

deferred_shading.fs

```
#version 330 core
out vec4 FragColor;
in vec2 TexCoords;
uniform sampler2D gPosition;
uniform sampler2D gNormal;
uniform sampler2D gAlbedoSpec;
…
void main()
{
    // retrieve data from gbuffer
    vec3 FragPos = texture(gPosition, TexCoords).rgb;
    vec3 Normal = texture(gNormal, TexCoords).rgb;
    vec3 Diffuse = texture(gAlbedoSpec, TexCoords).rgb;
    float Specular = texture(gAlbedoSpec, TexCoords).a;

    // then calculate lighting as usual
    …

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

Light Attenuation

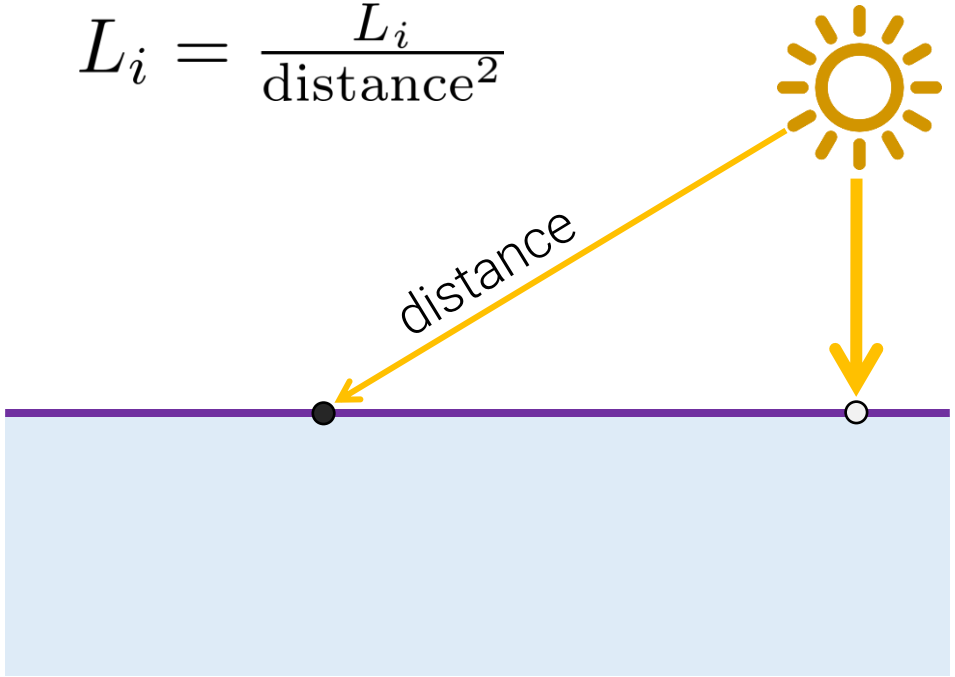
deferred_shading.fs

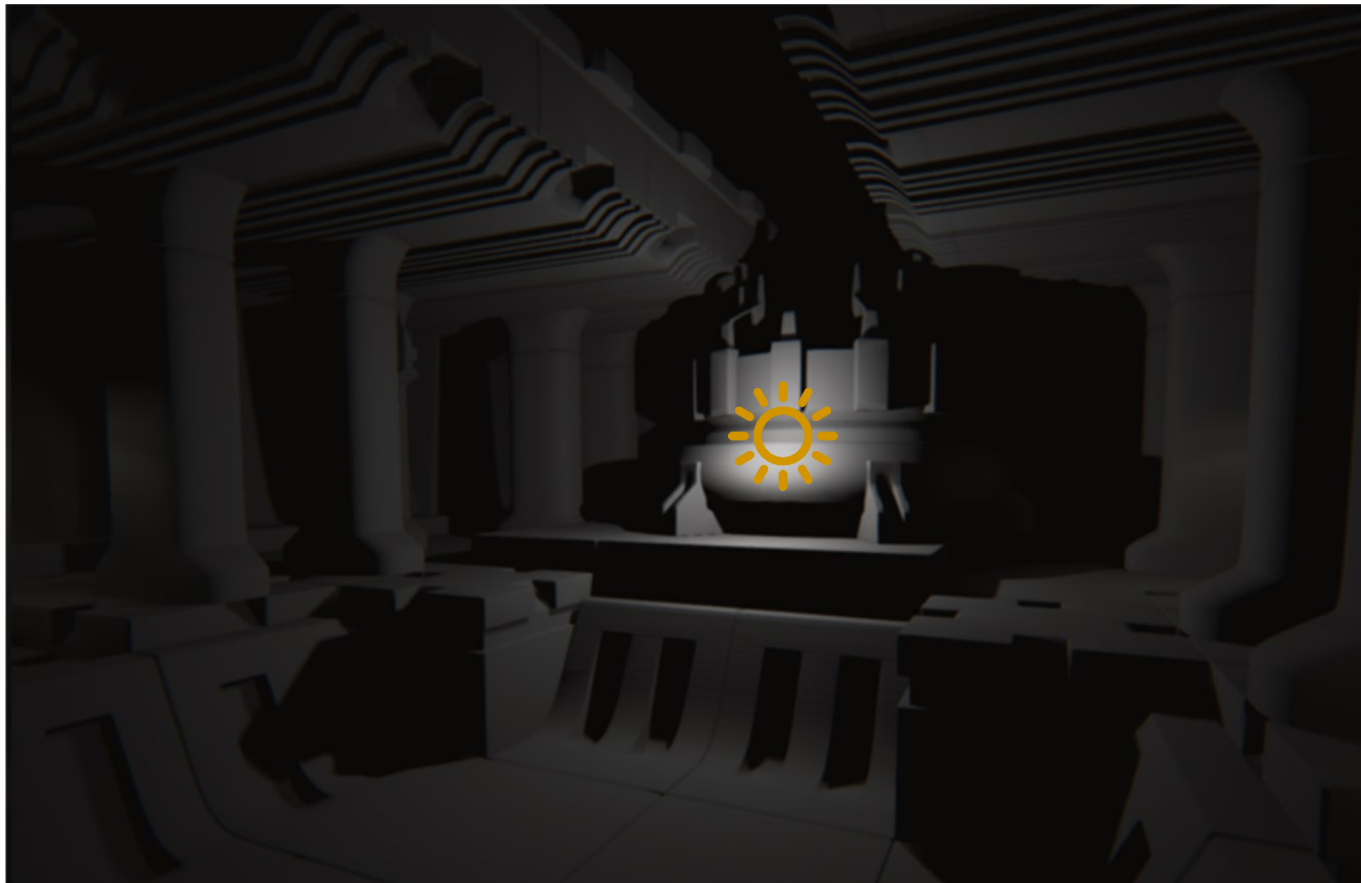
```
…
        // attenuation
        float distance = length(lights[i].Position - FragPos);
        float attenuation = 1.0 / (1.0 + distance * distance);
        diffuse *= attenuation;
        specular *= attenuation;
…
```

Light Attenuation

- If an object is further away from a light it receives less light.
- Inverse-square law:

$$L_i = \frac{L_i}{\text{distance}^2}$$





Gamma Correction

- Map in the range [0, 1) :
- Gamma correction (increase contrast):

$$c_{\text{final}} = \frac{c_{\text{final}}}{c_{\text{final}}+1}$$

$$c_{\text{final}} = c_{\text{final}}^{1/\gamma}$$

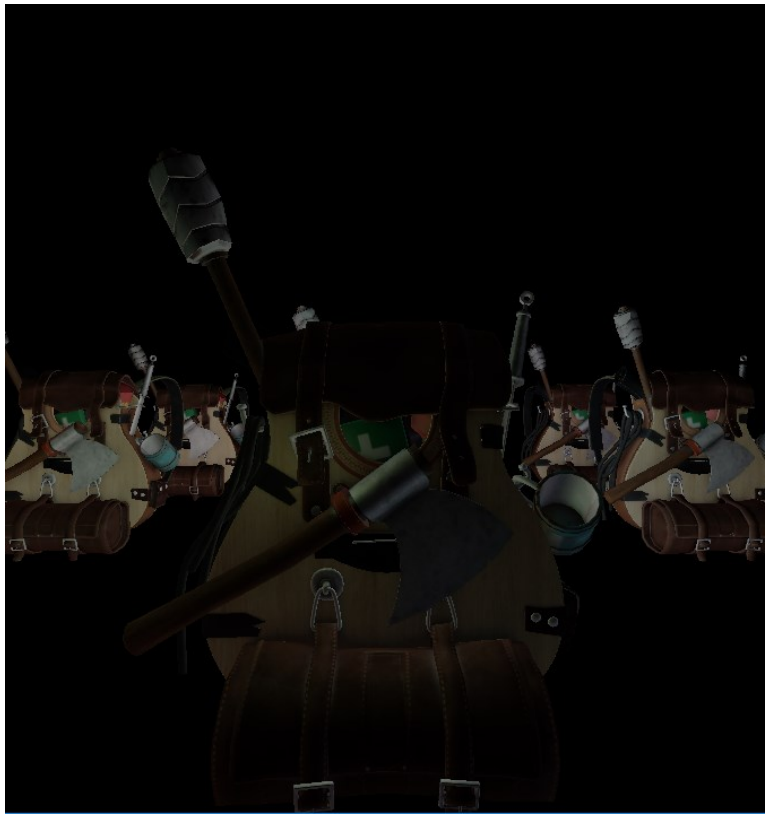
deferred_shading.fs

```
…
    vec3 color = lighting;
    // map to [0,1)
    color = color / (color + vec3(1.0));
    // gamma correct
    color = pow(color, vec3(1.0/gamma));

    FragColor = vec4(color, 1.0);
…
```

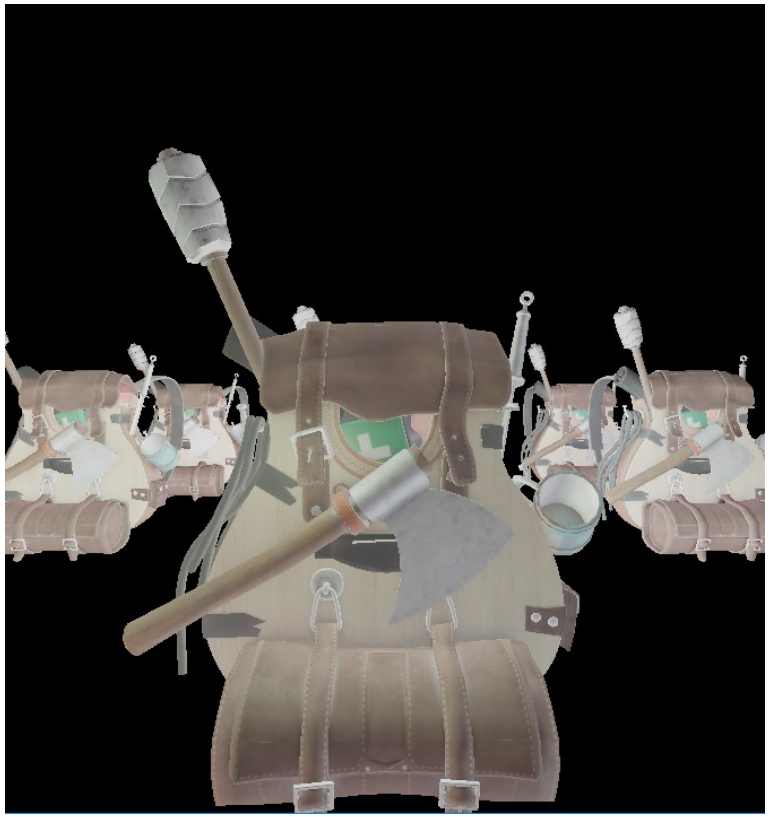
Gamma Correction

gamma = 0.5



gamma = 1.0

gamma = 2.5



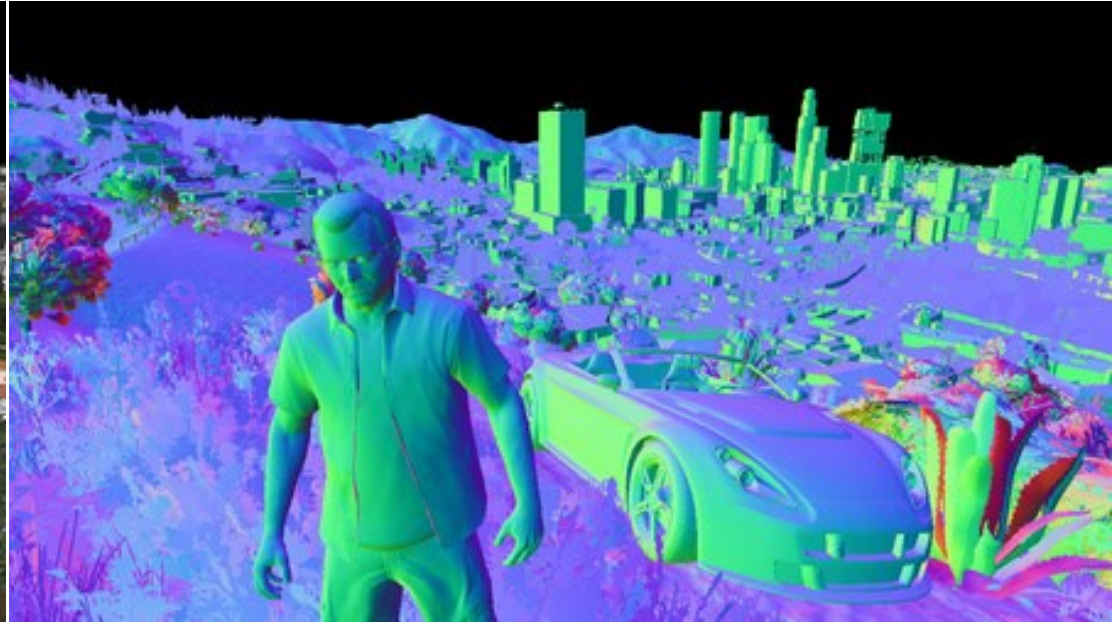
Questions?

Deffered Shading Example: GTA V

www.adriancourreges.com/blog/2015/11/02/gta-v-graphics-study/

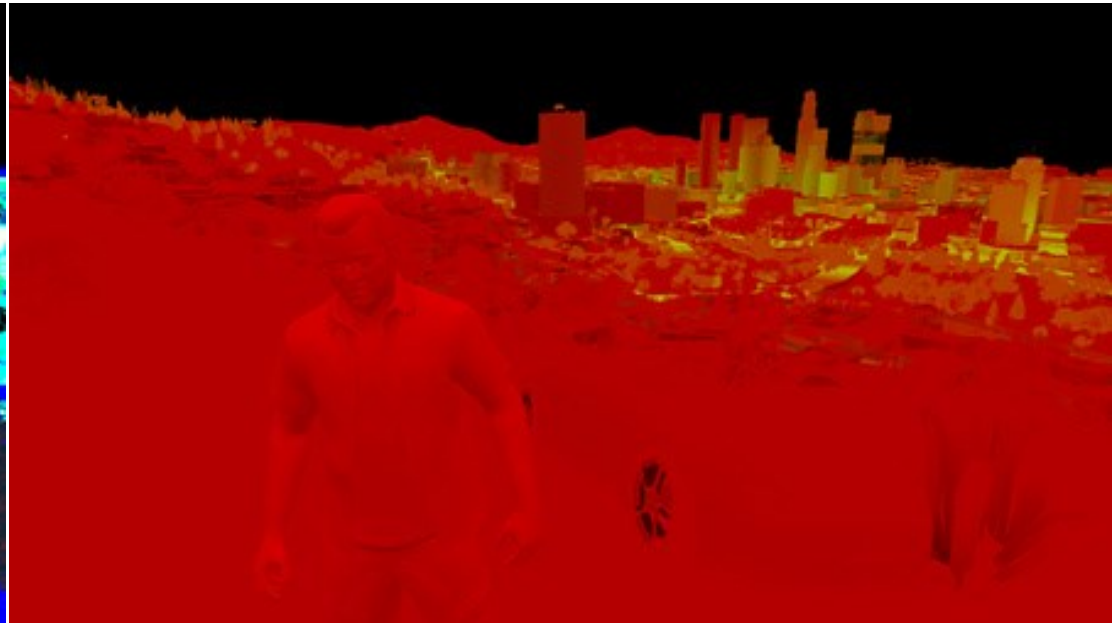
G-Buffer

Diffuse



Normal

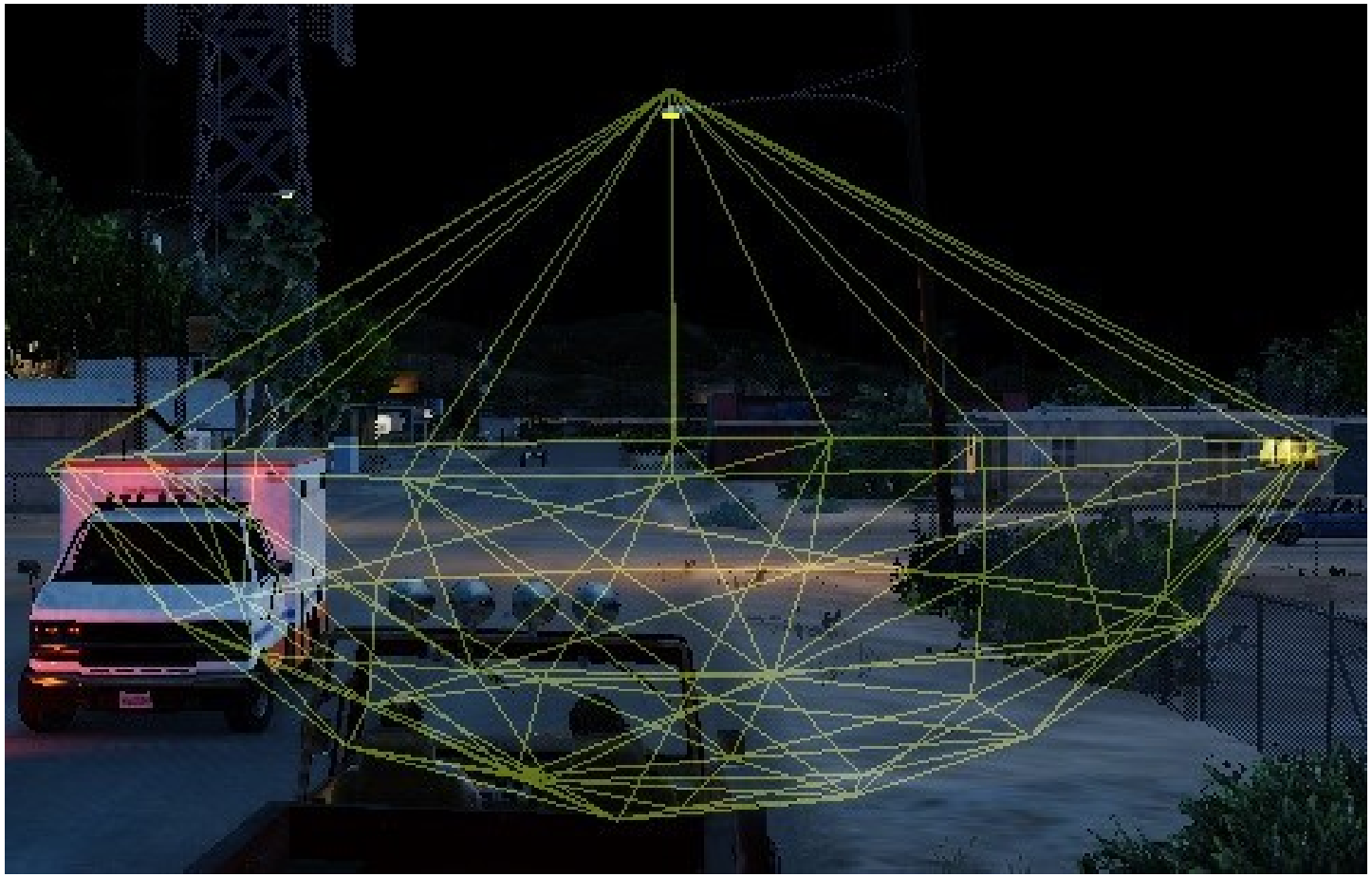
Specular



Irradiance

Light Volumes

Light Volumes



Loads of Lights …

Deferred Shading

- Pros:
  - Perfect batching (no object dependence)
  - Many small lights are just as cheap as a few big ones (100 lights and up are no problem)
  - Combines well with other postprocessing effects
- Cons (not a big problem on current hardware ☺):
  - High bandwidth required
    - Not applicable on older hardware
  - Alpha blending needs forward rendering
  - Hardware multisampling not available

Deferred Shading …

- Is standard in most modern game engines.

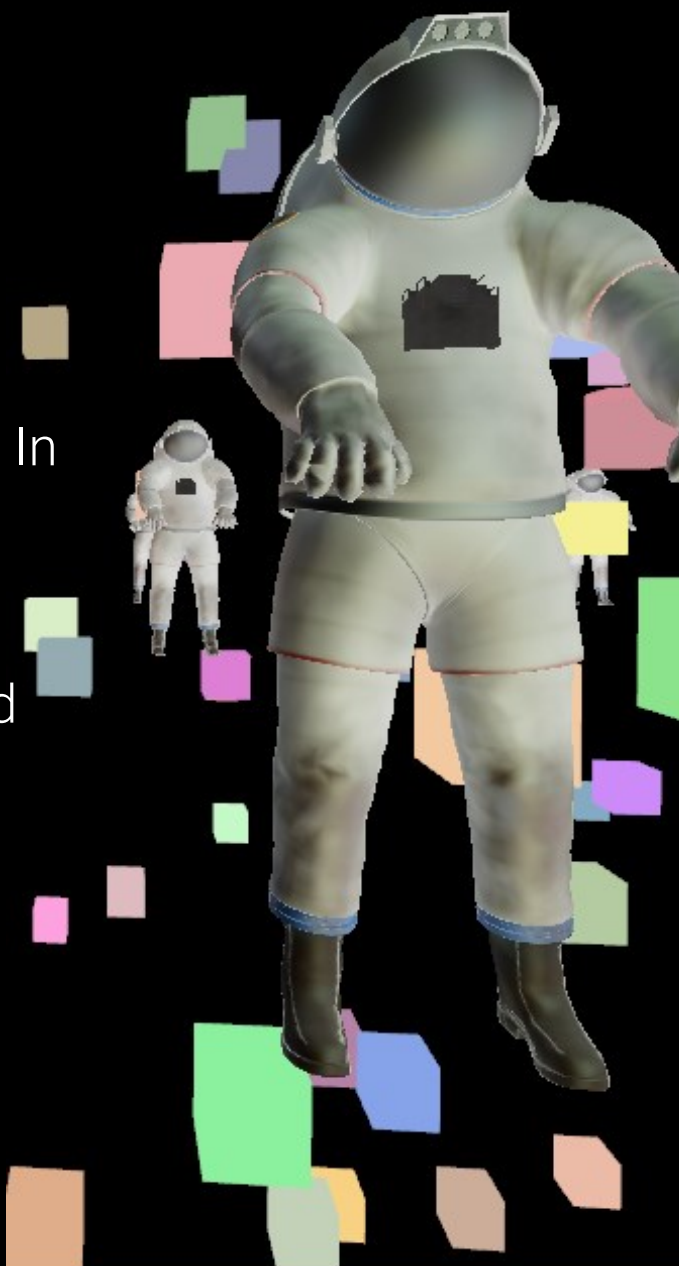
Exercise 03 – Deferred: Normal Mapping & Postpro

Implement Normal Mapping in our deferred shading pipeline. Remember you need Tangents, Bitangents and the normal texture in the shader. They are already available in the shaders, but we have not used them thus far. Think about the different spaces (Tangent space, world space, …) and how you can get a tangent space normal into world space. In deferred shading all lighting computations are done in world space. For normal mapping we did the computations in tangent space (see slides).

Furthermore, implement a nice postprocessing effect on top of the deferred-shading pipeline (e.g., blurring, bloom, water ripples, …). It should be possible to turn postprocessing on and off.

You can use a custom scene. If your model has the texture names in an MTL file, they will be loaded automatically and bound during shading.

Upload a PDF for the progress report (you have ~3 weeks for this exercise).



Questions?

David Schedl

david.schedl@fh-hagenberg.at