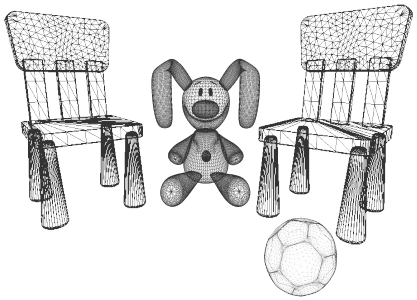
Shadow Maps

Evangelou Iordanis

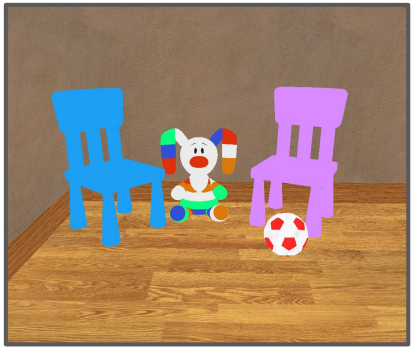
Presentation co-author Anastasios Gkaravelis

Recap



Forward pass

Custom framebuffer

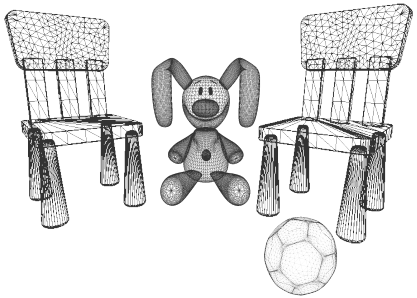


Post process

Default framebuffer

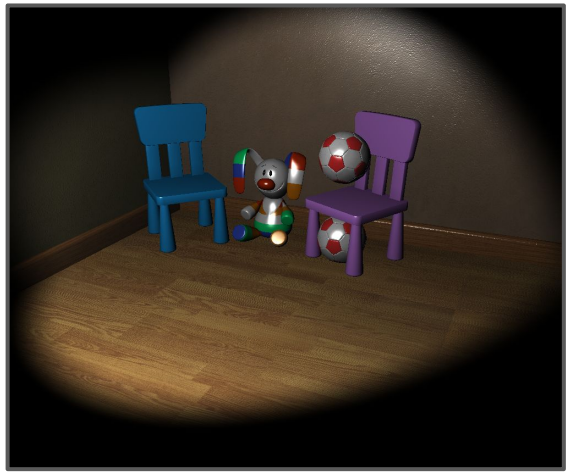
Screen

Recap



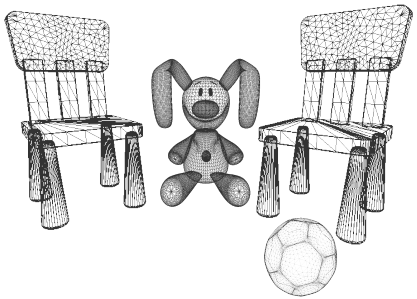
Forward pass

Custom framebuffer

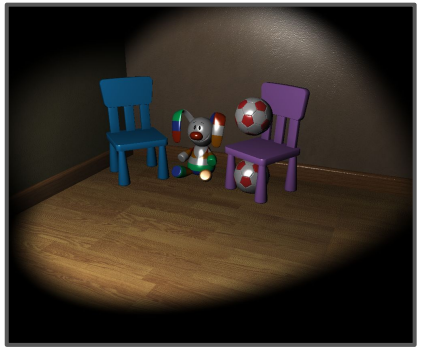


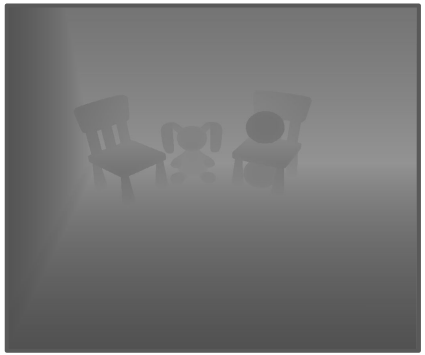
Shadows

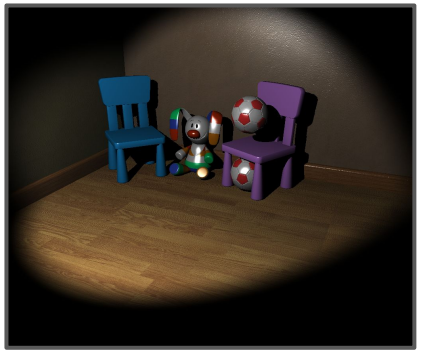
Custom framebuffer



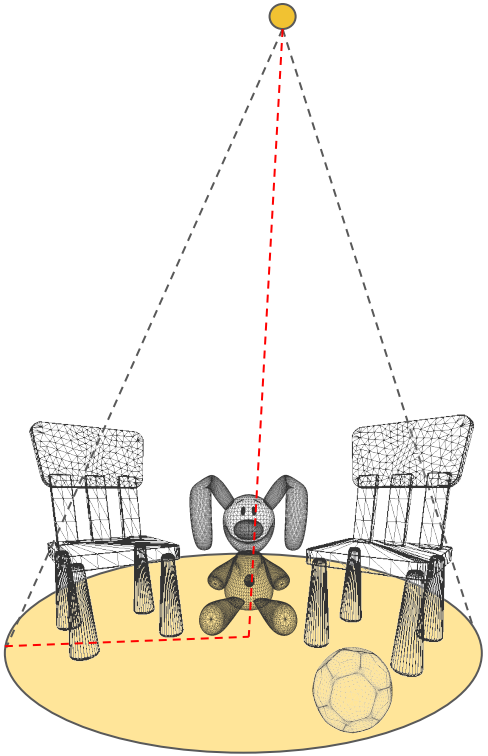
Forward pass



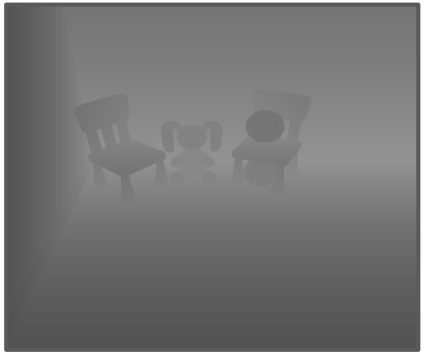




Shadows

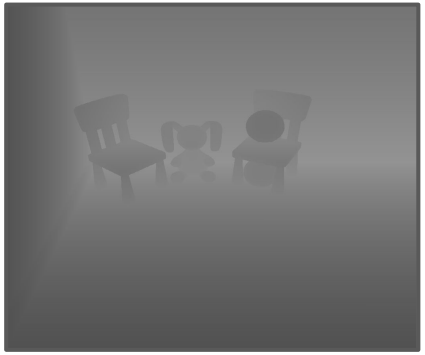


```
glm::mat4 view_matrix = glm::lookAt(
       light_position,
       light_target,
       glm::vec3(0, 1, 0));


float near = 0.1f;
float far = 10.f;

float h = near * glm::tan(
       glm::radians(penumbra * 0.5f));

glm::mat4 projection_matrix = glm::frustum(
       -h,           // left
       h,            // right
       -h,           // bottom
       h,            // top
       near, far);
```



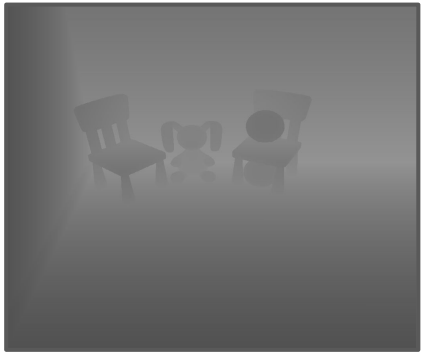
Shadows

```
glm::mat4 view_matrix;
glm::mat4 projection_matrix;

glGenTextures(1, &shadow_map_texture);

glBindTexture(GL_TEXTURE_2D, shadow_map_texture);

glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT24,
      shadow_map_resolution, shadow_map_resolution, 0,
      GL_DEPTH_COMPONENT, GL_FLOAT, NULL);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);

glGenFramebuffers(1, &shadow_map_fbo);
glBindFramebuffer(GL_FRAMEBUFFER, shadow_map_fbo);

glFramebufferTexture(GL_FRAMEBUFFER,
      GL_DEPTH_ATTACHMENT, shadow_map_texture, 0);

glDrawBuffer(GL_NONE);
glReadBuffer(GL_NONE);
glBindTexture(GL_TEXTURE_2D, 0);
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```



Shadows

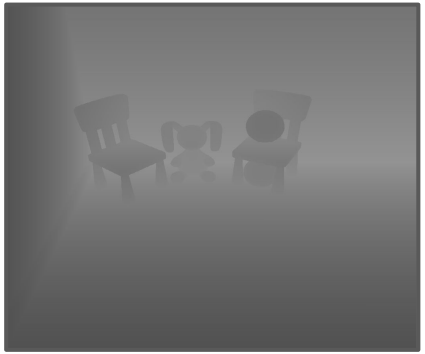
```
glm::mat4 view_matrix;
glm::mat4 projection_matrix;
GLuint shadow_map_fbo;

glBindFramebuffer(GL_FRAMEBUFFER, shadow_map_fbo);
glViewport(0, 0, depth_texture_resolution, depth_texture_resolution);
glEnable(GL_DEPTH_TEST);
glClear(GL_DEPTH_BUFFER_BIT);

light_program.Bind();

glm::mat4 proj = projection_matrix * view_matrix * world_matrix;

// Draw geometry
// [..]

light_program.unBind();
glDisable(GL_DEPTH_TEST);
glBindFramebuffer(GL_FRAMEBUFFER, 0);
```



Shadows

- Vertex shader

```
#version 330 core
layout(location = 0) in vec3 coord3d;

uniform mat4 uniform_projection_matrix;

void main(void)
{
	gl_Position = uniform_projection_matrix * vec4(coord3d, 1.0);
}
```



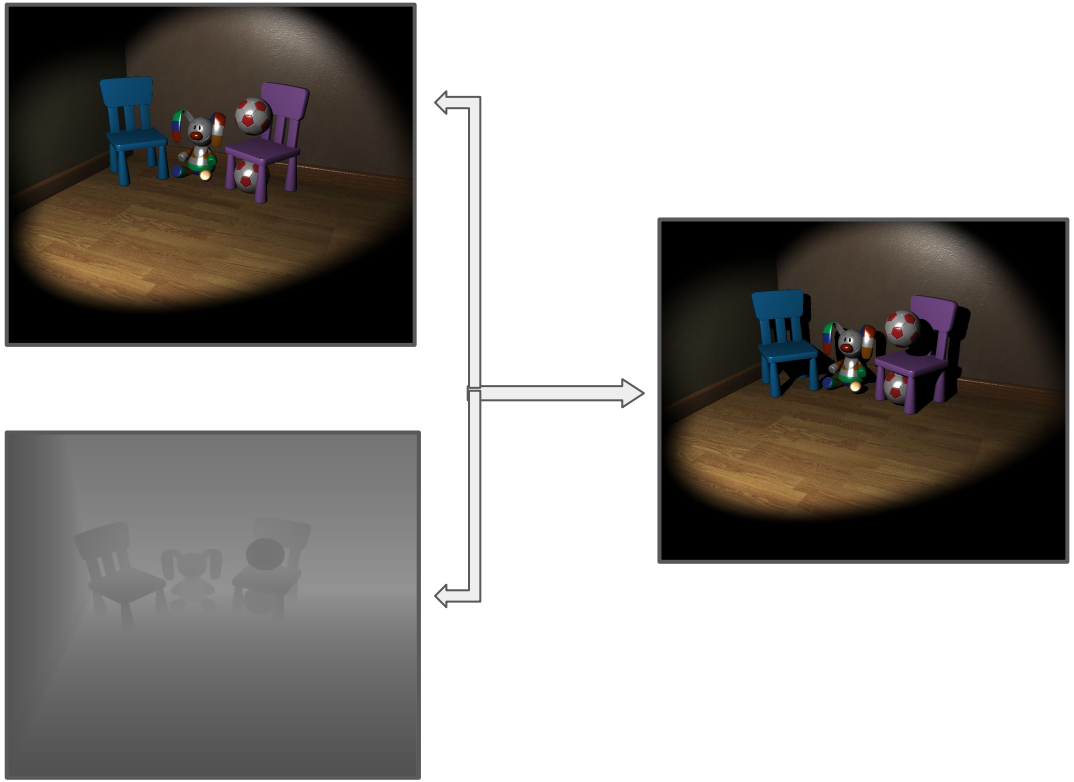
- Fragment shader

```
#version 330 core
layout(location = 0) out vec4 out_color;

void main(void) { /* Empty */ }
```

Shadows

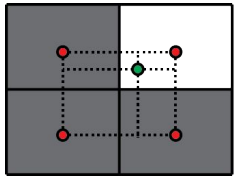
```
void render()
{
	RenderShadowMaps();
	RenderGeometry();
	RenderPostProcess();
}
```

Custom framebuffer



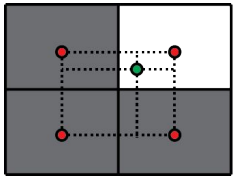
Shadows

Nearest filter



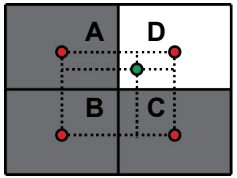
$$s = I(p)$$

Mean filter



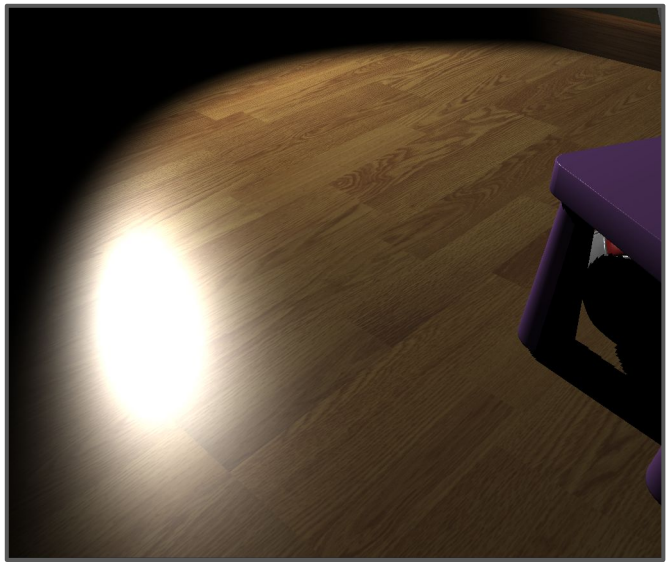
$$s = \frac{I(p-[-1,-1]) + I(p-[-1,0]) + I(p-[0,-1]) + I(p)}{4}$$

Weighted filter

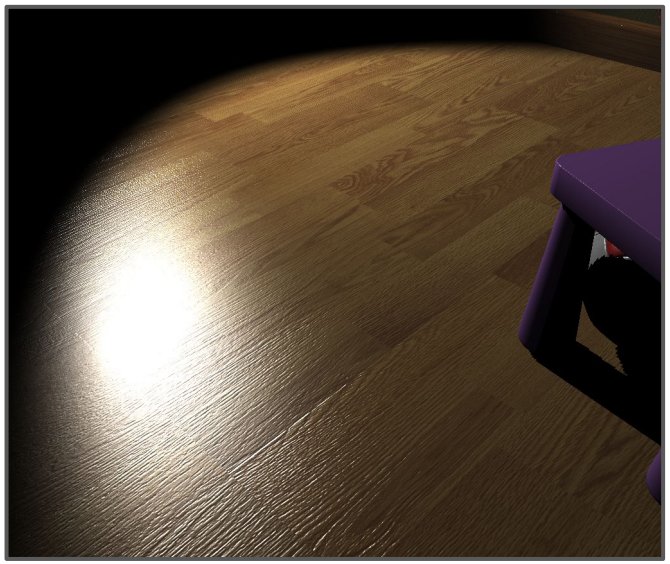


$$s = area(A) + area(B) + area(C) + area(D)$$

Normal maps

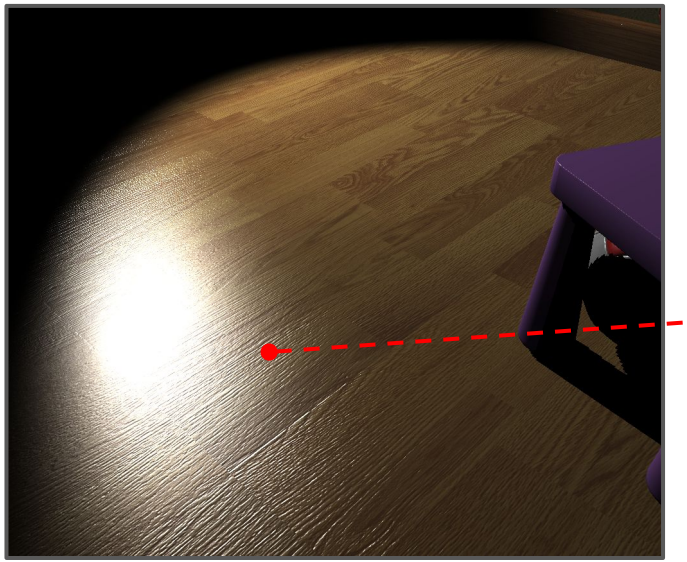


Without bump map

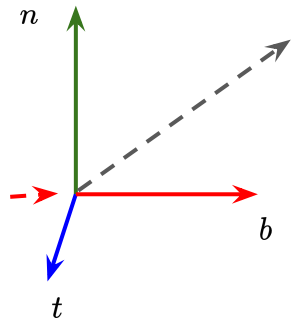


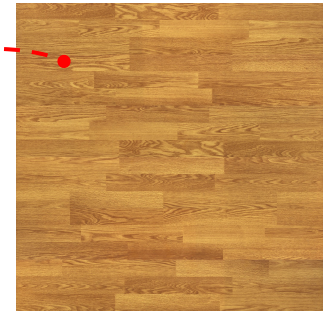
With bump map

Normal maps



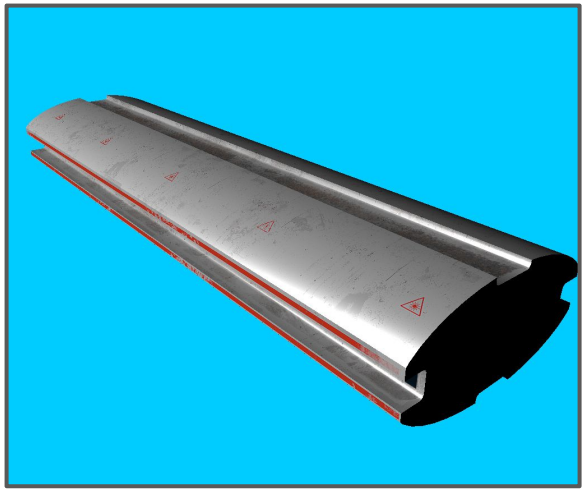
With bump map



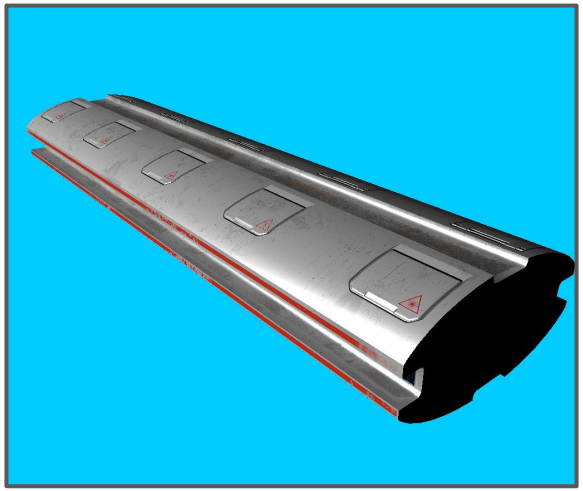


$$n' = n - b\,\frac{\partial b(u,v)}{\partial v} - t\,\frac{\partial b(u,v)}{\partial u}$$

Normal maps

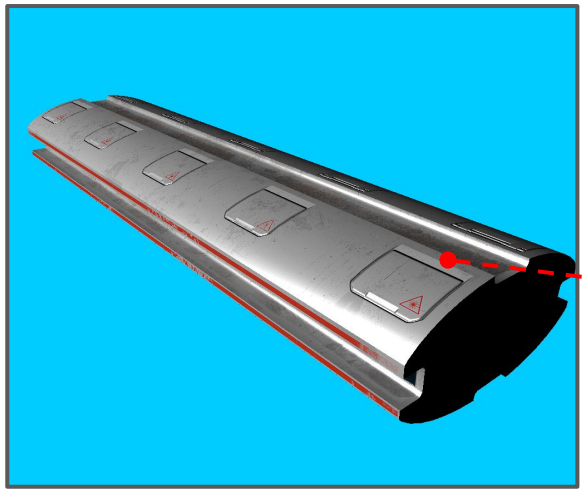


Without normal map

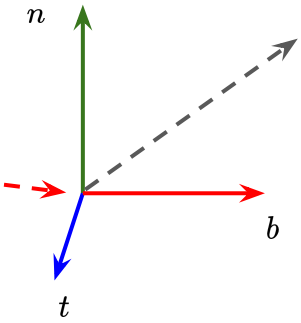


With normal map

Normal maps



With normal map



$$n' = d_x t + d_y b + d_z n$$