Lenticular Lens Effects (3D Effects on 2D Surfaces)

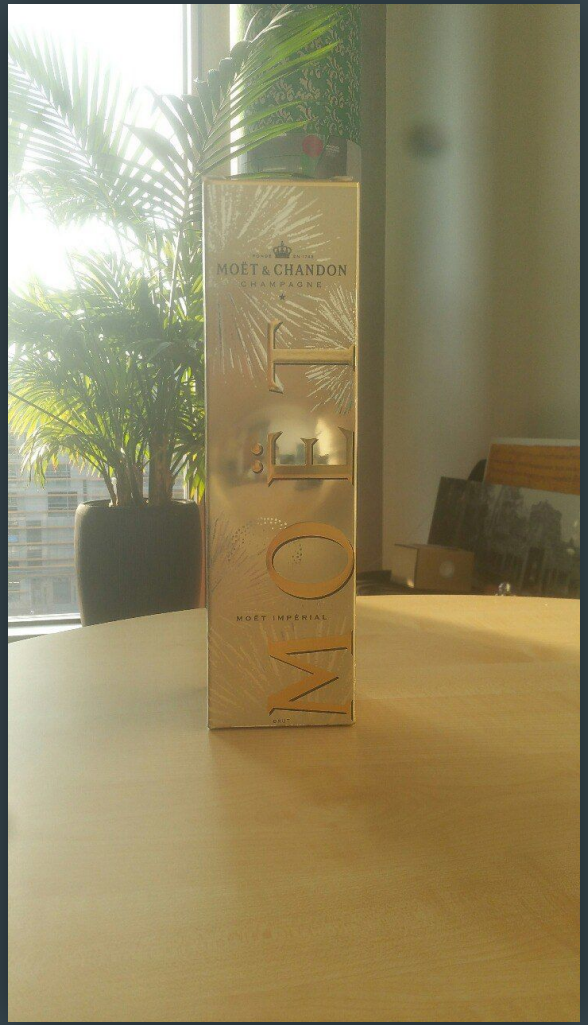
...

A beginner’s practicum by Michael Pronkin

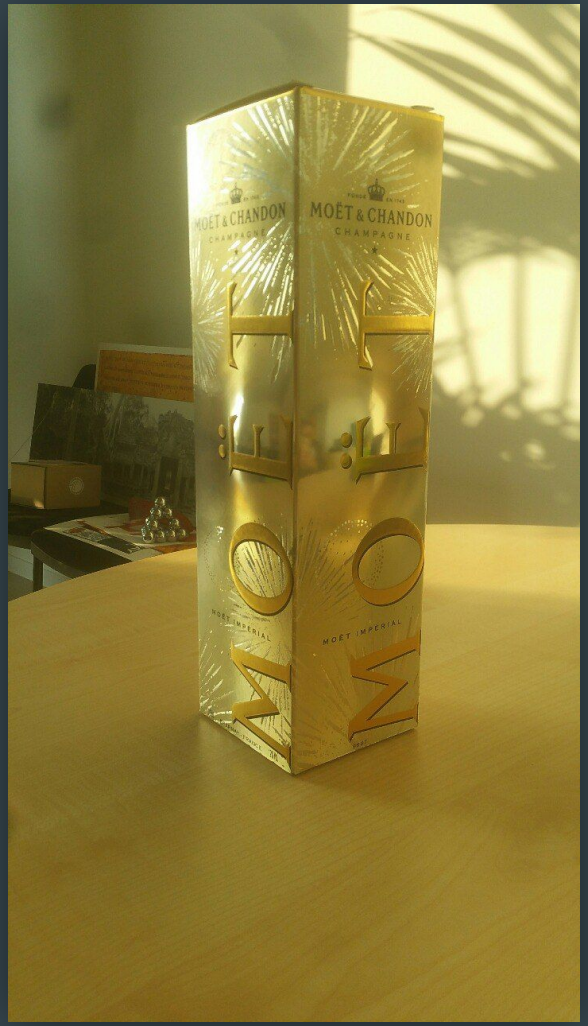
Overview

- Lenticular Lenses
  - Target Effect
  - Background (Lenticular Printing)
  - Simplification for Computer Graphics
  - Possible Solutions
- Implementation
  - UV Transformations
  - Normal Baking
  - Shaders
- Demo

Target Effect


MOËT & CHANDON
CHAMPAGNE
MOËT IMPÉRIAL


2018
MOËT & CHANDON
CHAMPAGNE
MOËT
MOËT IMPÉRIAL


MOËT & CHANDON
CHAMPAGNE
MOËT IMPERIAL


simple boar
MATH & COMPSCI
APPLE JUICE
MATH
MATHEMATICAL
MATH & COMPSCI
APPLE JUICE
MATH
MATHEMATICAL

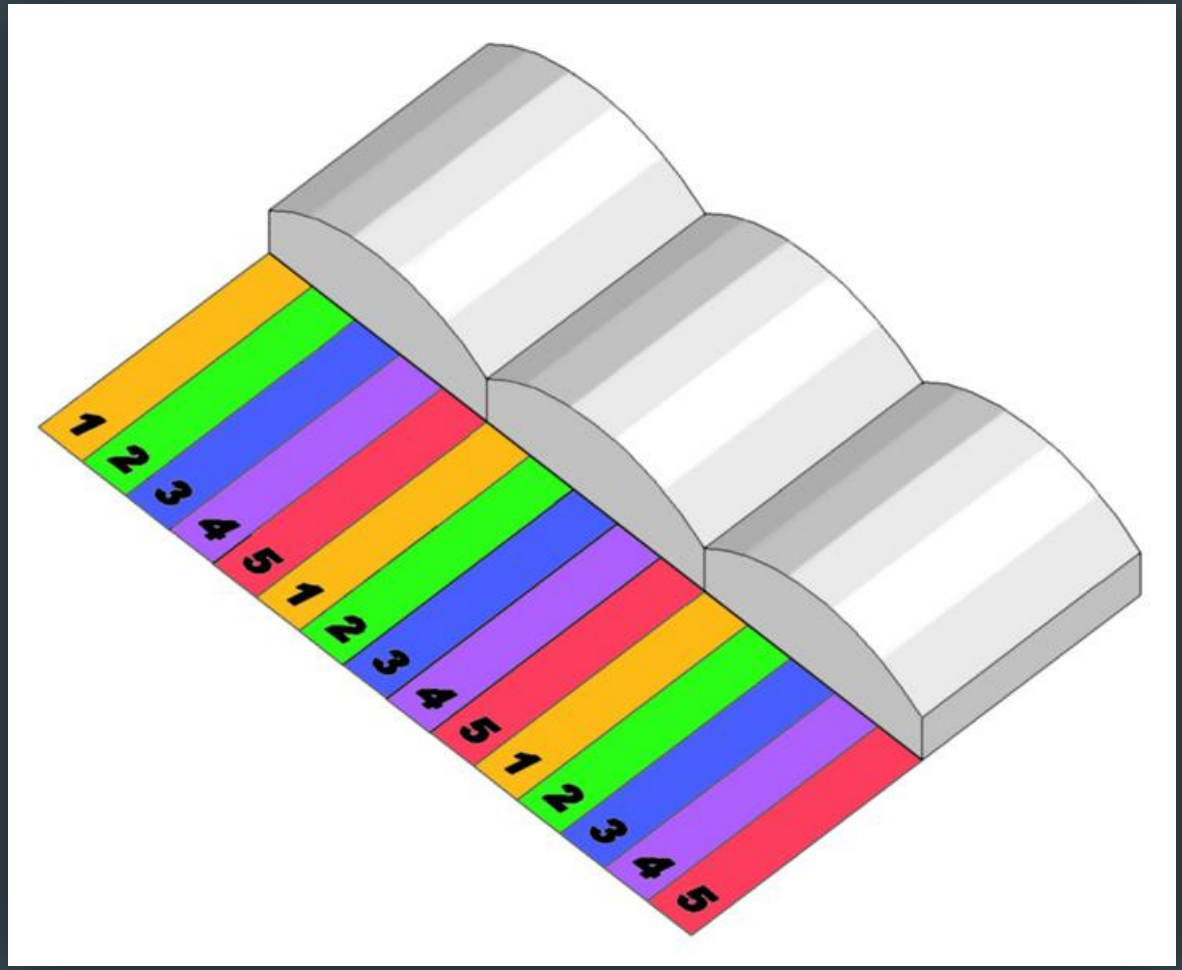
Background (Lenticular Printing)

What is Lenticular Printing?

- Multiple images in one medium
- Visibility selection based on viewing angle
- Used in postcards or stickers
- Illusion of depth

From https://design.whitney.org/post/152879609284/lenticular-postcard-photographs-by-jens

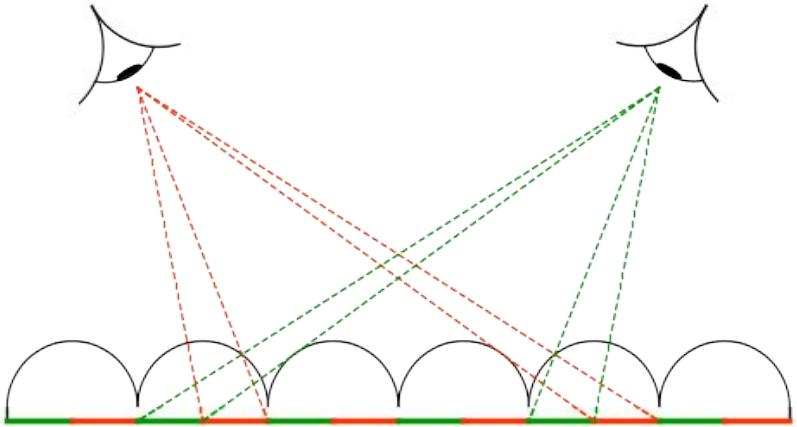
From https://sydneys18.weebly.com/blog/lenticular-printing



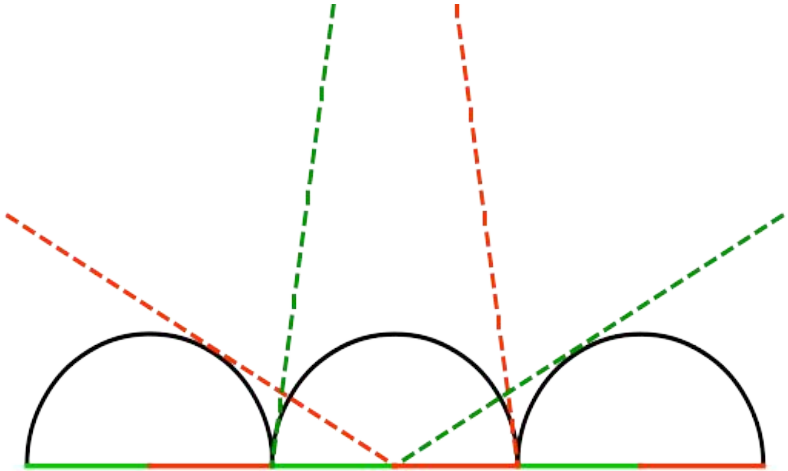
From http://printbrain.webmartuk.com/print-technology/lenticular-printing-and-the-brave-new-world-of-3d-printing/

Lenticular Lenses

Lenticular Lenses



Lenticular Lenses



Simplification for Computer Graphics

How can Lenticular Lenses be rendered?

- Very fine geometry?

How can Lenticular Lenses be rendered?

- Very fine geometry? ✘

How can Lenticular Lenses be rendered?

- Very fine geometry? ✘
- Physically Based anisotropic materials?

How can Lenticular Lenses be rendered?

- Very fine geometry? ✘
- Physically Based anisotropic materials? ✘

How can Lenticular Lenses be rendered?

- Very fine geometry? ✘
- Physically Based anisotropic materials? ✘
- Using different textures based on viewing angle?

How can Lenticular Lenses be rendered?

- Very fine geometry? ✘
- Physically Based anisotropic materials? ✘
- Using different textures based on viewing angle? ✔
  - Simple, efficient, flexible

Lenticular Lens effect

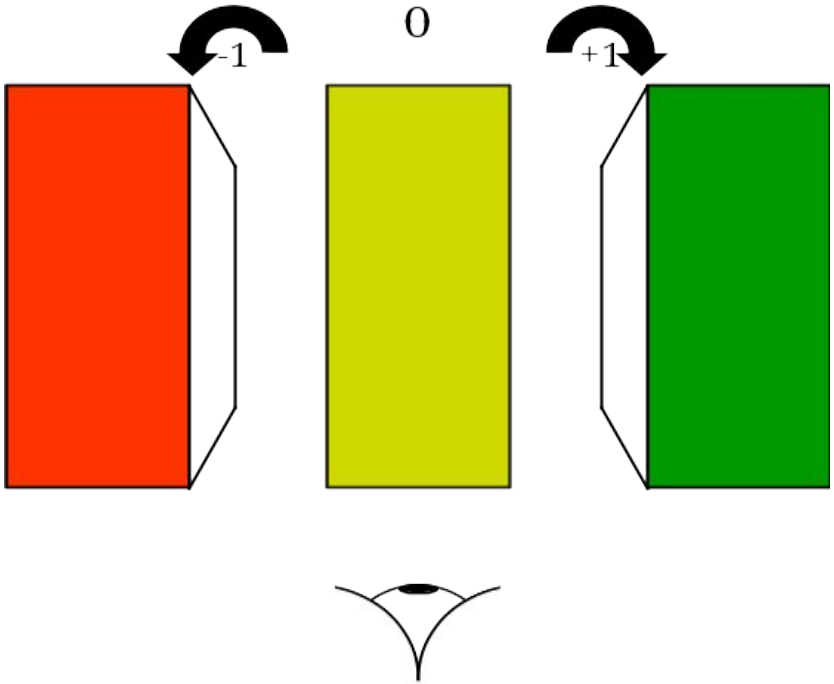
Given

- Viewing angle
- Normal
- Textures for given angle

Needed

- The multiplier for each texture
- A blending function?

Lenticular Lens effect



Lenticular Lens effect

$$uv\_mod(n, v) = n \cdot v$$

$$uv\_move_{+}(x, y) = (x + uv\_mod(n, v), y)$$

$$uv\_scale_{+}(x, y) = (x \times tex\_width \times uv\_mod(n,v), y)$$

n := normal vector
v := view direction

Lenticular Lens effect

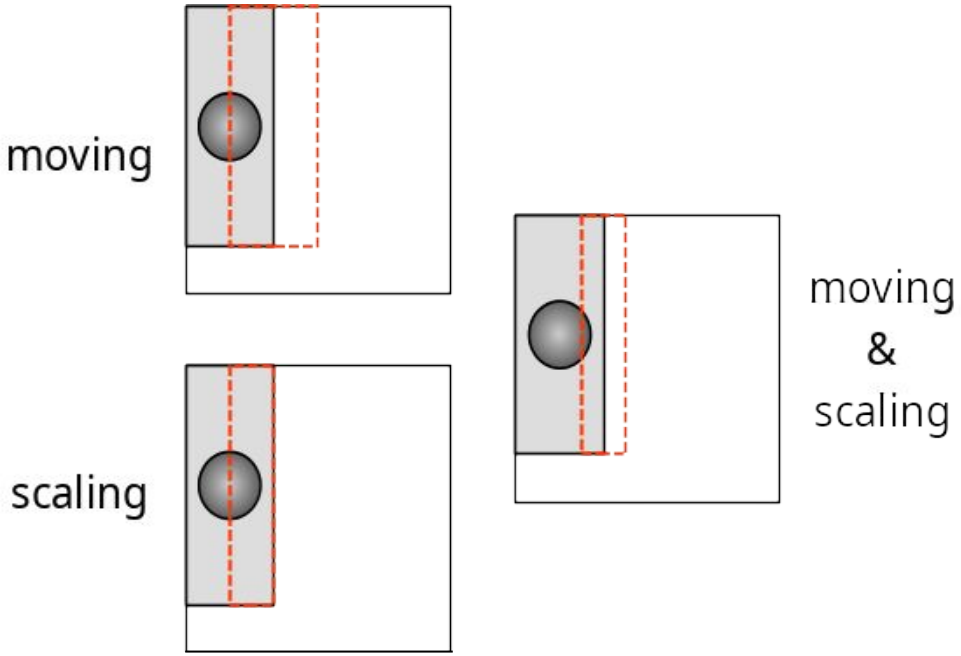
$$uv\_mod(n, v) = n \cdot v$$

$$uv\_move\_(x, y) = (x + tex\_width + uv\_mod(n, v), y)$$

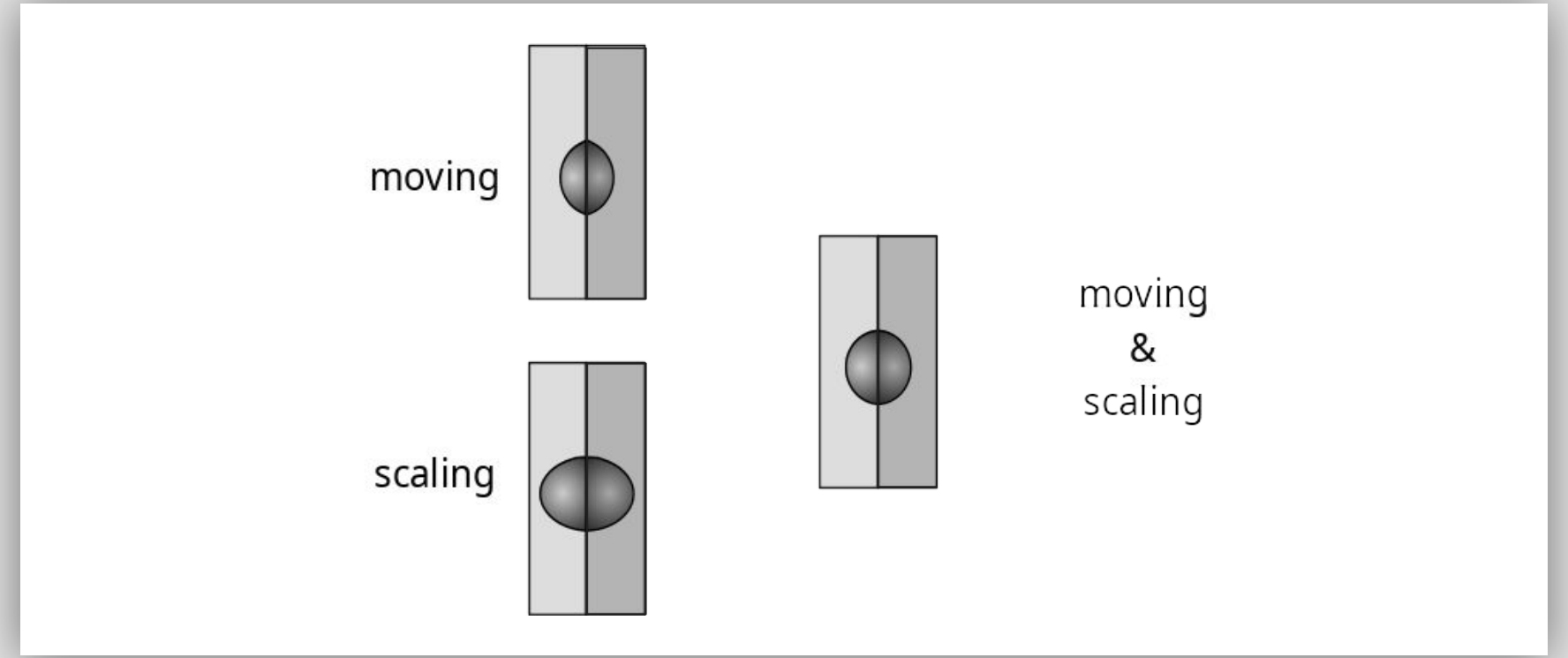
$$uv\_scale\_(x, y) = (x \times tex\_width \times (1 - uv\_mod(n,v)), y)$$

n := normal vector
v := view direction

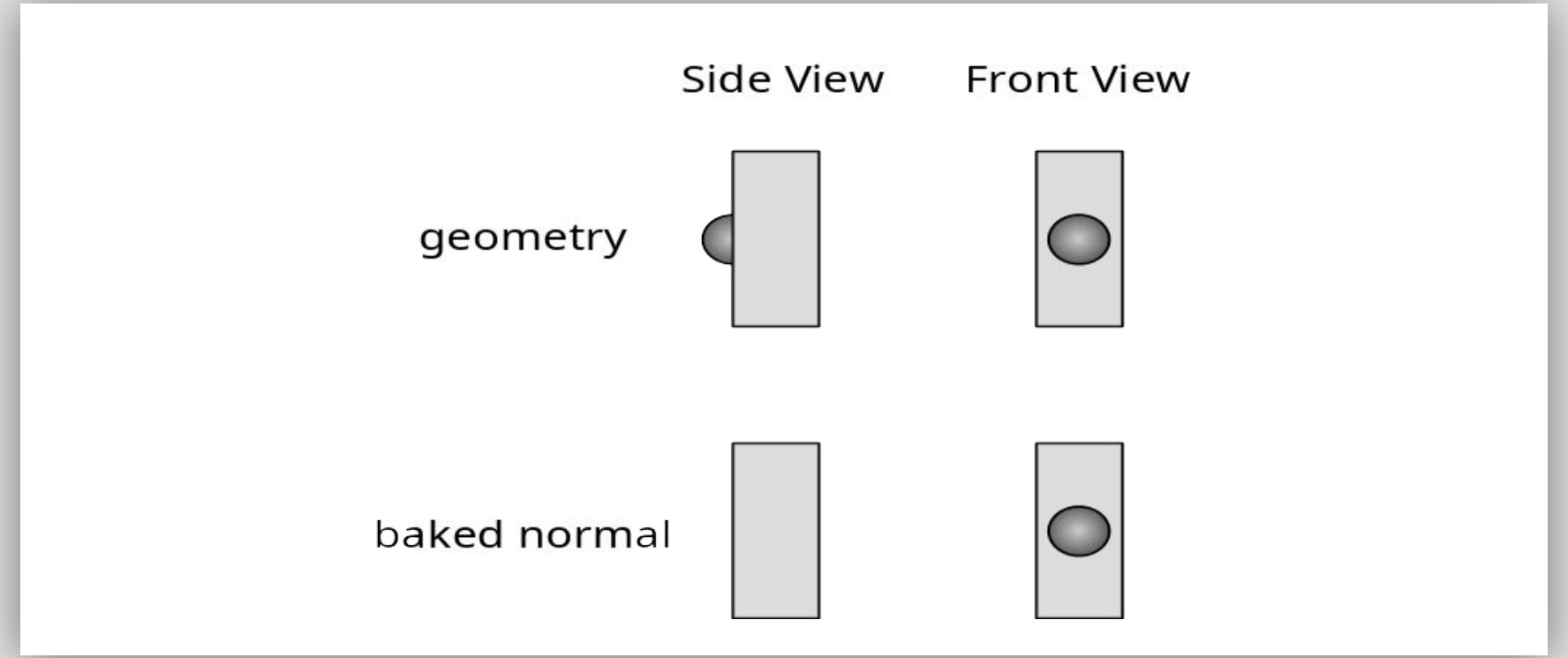
UV transformations



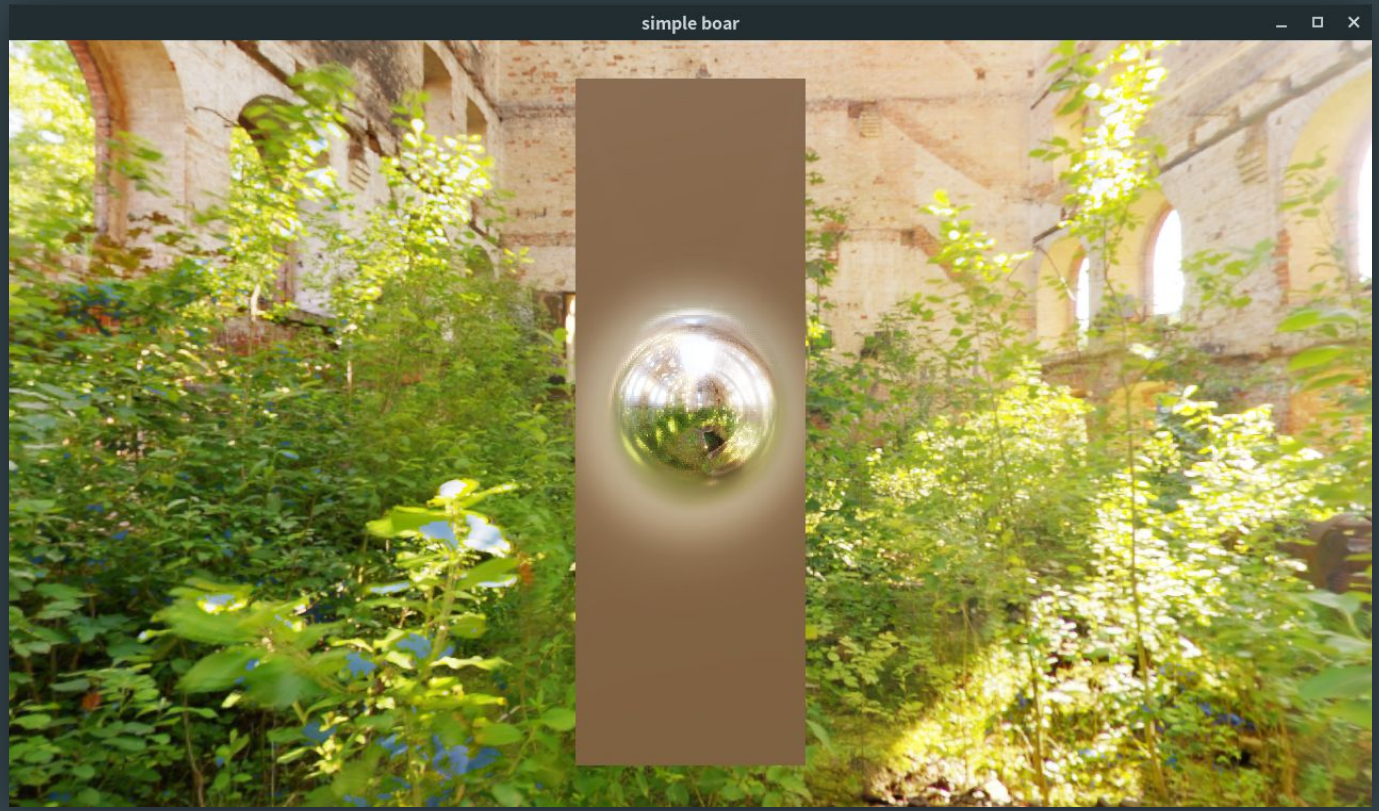
UV transformations



Normal Baking



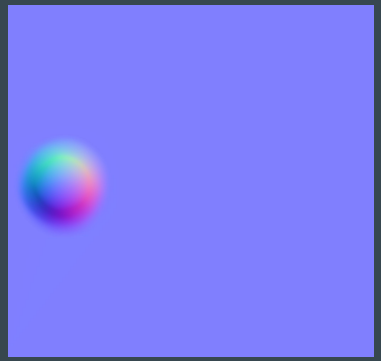
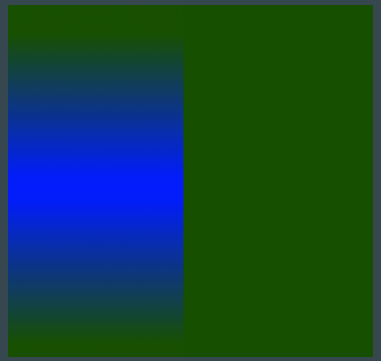
Normal Baking



Shader (vert)

```
#version 330 core

// input data
layout(location = 0) in vec3 pos;
layout(location = 1) in vec3 normal;
layout(location = 2) in vec2 uv;

// output data
out vec2 frag_uv_lenticular;
// ...

// uniforms
uniform mat4 mvp_mat;

void main() {
    gl_Position    = mvp_mat * vec4(pos, 1.0f);
    float mdot = dot((mvp_mat * vec4(normal, 0.0f)).xyz, vec3(1,0,0));
    // ...
}

```

Overlay Textures

	albedo	normal	roughness-metallic
lenticular			
overlay			

Shader (frag)

```
#version 330 core

// ouput data
layout(location = 0) out vec4 albedo;
layout(location = 1) out vec4 normal_rough;
layout(location = 2) out vec4 specular_reflect;
layout(location = 3) out vec4 pos_depth;

// input data
in vec2 frag_uv;
in vec2 frag_uv_lenticular;

// uniforms
uniform sampler2D tex_albedo;
uniform sampler2D tex_normal;
uniform sampler2D tex_specular;
uniform sampler2D tex_occlusion_rough_metal;
uniform sampler2D tex_albedo_over;
uniform sampler2D tex_normal_over;
uniform sampler2D tex_ocr_over;

void main() {
    vec4 albedo_over       = texture(tex_albedo_over, frag_uv);
    vec4 albedo_lenticular = texture(tex_albedo, frag_uv_lenticular);
    albedo                 = mix(albedo_lenticular, albedo_over, albedo_over.a);

    vec3 normal_over       = normalize(tbn_mat * ((texture(tex_normal_over, frag_uv).rgb * 2.0f) - 1.0f));
    vec3 normal_lenticular = normalize(tbn_mat * ((texture(tex_normal, frag_uv_lenticular).rgb * 2.0f) - 1.0f));
    normal_rough.rgb       = mix(normal_lenticular, normal_over, albedo_over.a);

    specular_reflect.rgb   = texture(tex_specular, frag_uv).rgb;
    vec4 ocr_over          = texture(tex_ocr_over, frag_uv);
    vec4 ocr_lenticular    = texture(tex_occlusion_rough_metal, frag_uv_lenticular);
    normal_rough.a         = mix(ocr_lenticular.y, ocr_over.y, albedo_over.a);
    specular_reflect.a     = mix(ocr_lenticular.z, ocr_over.z, albedo_over.a);
}

```

thanks!