

OpenGL
Shader
Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

Shaders

EDAF80: Computer Graphics

Rikard Olajos



OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Tessellation
- Interpolation
- Any questions?

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

1 OpenGL Shader Language

2 Textures

3 Shading theory

4 Assignment 3

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

OpenGL Shader Language

OpenGL Shader Language (GLSL)

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

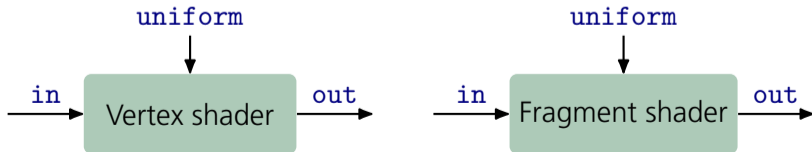
- C like language for programming GPUs
- Vertex & fragment shaders
- Labs: GLSL 4.10 (OpenGL 4.1)
 - Have geometry & tessellation shaders as well
- Compute shaders since OpenGL 4.3
- Newest OpenGL 4.6

- Types

```
float, int, bool, vec2, vec3, vec4, mat3 ...  
sampler2D, samplerCube ...
```

- Type qualifiers

```
const           // compile-time constant  
uniform        // constant per primitive  
in, out        // attributes passed to/from, and between,  
                // the vertex & pixel shader
```



GLSL OPERATORS AND FUNCTIONS

- Structures and arrays with built-in operators

```
vec2, vec3, vec4, mat3, mat4
vec3 v = vec4(1.0f).xyz;           // Swizzle
vec4 u = mat4(1.0f) * vec4(1.0f); // Operator overloading
```

- User defined functions
- Built-in functions

```
sin(), cos(), pow(), normalize(), min(), max(), clamp(),
reflect(), refract(), sqrt(), noise1(), noise2(), ...
```

GLSL FLOW CONTROL AND PREPROCESSOR

- Flow control

```
if, if-else, for, while, do-while  
discard    // Fragment shader only
```

- Preprocessor directives

```
#define, #undef, #if, #else, #endif, ...
```

- Comments

```
// Like this  
/* Or, like this */
```

DEVELOPMENT TOOLS

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Shadertoy: <https://www.shadertoy.com/>
- NVIDIA Nsight Visual Studio Edition
- GLSL-Debugger
- This course: by hand 😊
 - Text-editor in Visual Studio
 - Compiler messages (from the GPU) written to ImGui log window
- Check out <https://docs.g1/> and [GLSL 4.10 Specification](#)

DIFFUSE VERTEX SHADER

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

```
layout (location = 0) in vec3 vertex;  
layout (location = 1) in vec3 normal;
```

```
uniform mat4 vertex_model_to_world;  
uniform mat4 normal_model_to_world;  
uniform mat4 vertex_world_to_clip;
```

```
out VS_OUT {  
    vec3 vertex;  
    vec3 normal;  
} vs_out;
```

```
void main()  
{  
    vs_out.vertex = vec3(vertex_model_to_world * vec4(vertex, 1.0));  
    vs_out.normal = vec3(normal_model_to_world * vec4(normal, 0.0));  
  
    gl_Position = vertex_world_to_clip * vertex_model_to_world * vec4(vertex, 1.0);  
}
```

VERTEX SHADER INPUTS

```
layout (location = 0) in vec3 vertex;  
layout (location = 1) in vec3 normal;
```

```
uniform mat4 vertex_model_to_world;  
uniform mat4 normal_model_to_world;  
uniform mat4 vertex_world_to_clip;
```

- layout specifies the location of the input data in the vertex buffer
 - Sent from `node.cpp`: `Node::render()`
 - Other `mat4` also comes from there
- Matrix data is loaded with `glUniformMatrix4fv(...)`

```
glUseProgram(program);  
GLint i = glGetUniformLocation(program, "vertex_model_to_world");  
glUniformMatrix4fv(i, 1, GL_FALSE, glm::value_ptr(world));
```

- More in [reference pages](#)

VERTEX SHADER OUTPUTS

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

```
out VS_OUT {
    vec3 vertex;
    vec3 normal;
} vs_out;

void main()
{
    vs_out.vertex = vec3(vertex_model_to_world * vec4(vertex, 1.0));
    vs_out.normal = vec3(normal_model_to_world * vec4(normal, 0.0));

    gl_Position = vertex_world_to_clip * vertex_model_to_world * vec4(vertex, 1.0);
}
```

- VS_OUT is a struct with the output values
- This must match the input to the fragment shader

DIFFUSE FRAGMENT SHADER

```
uniform vec3 light_position;

in VS_OUT {
    vec3 vertex;
    vec3 normal;
} fs_in;

out vec4 frag_color;

void main()
{
    vec3 L = normalize(light_position - fs_in.vertex);
    frag_color = vec4(1.0) * clamp(dot(normalize(fs_in.normal), L), 0.0, 1.0);
}
```

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

DIFFUSE FRAGMENT SHADER

```
uniform vec3 light_position;

in VS_OUT {
    vec3 vertex;
    vec3 normal;
} fs_in;

out vec4 frag_color;

void main()
{
    vec3 L = normalize(light_position - fs_in.vertex);
    frag_color = vec4(1.0) * clamp(dot(normalize(fs_in.normal), L), 0.0, 1.0);
}
```

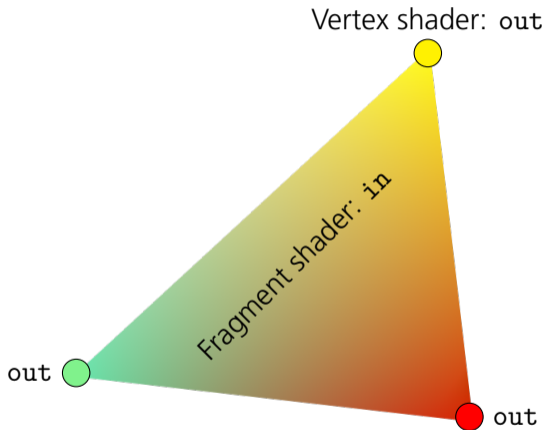
- VS_OUT struct matches the vertex shader output and contains interpolated values
- vertex is the world position of the current pixel
- L is the normalized direction to the light
- light_position set with

```
glUseProgram(program);
GLint i = glGetUniformLocation(program, "light_position");
glUniform3fv(i, 1, glm::value_ptr(light_position));
```

Behaviour can be
changed with

- `smooth` (default)
- `flat`
- `noperspective`

INTERPOLATION OVER TRIANGLE



Seminar Exercise 3-1: In/Out Variables and Provoking Vertex

- 1 Replace the red color in the fragment shader with the varying `in` variable `var_color`.
- 2 Add a second triangle with indices 2, 1, 3.
- 3 Put the keyword `flat` in front of the `var_color` variable in the vertex and fragment shaders. However, do **not** put it in front of the `out` variable in the fragment shader!
 - Try changing the order of the indices used to draw the triangle. What happens?

OpenGL
Shader

Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- 1 See provoking vertex in the [OpenGL wiki](#)

OpenGL
Shader
Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

Textures

OpenGL
Shader
Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

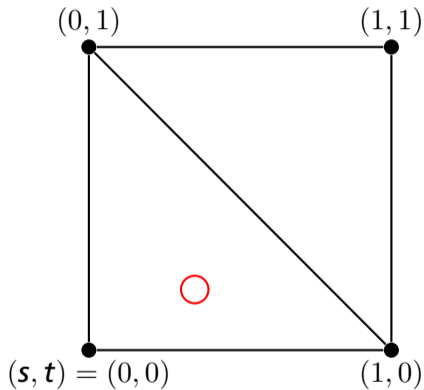
Normal mapping

Roughness mapping

Assignment 3

Hints

Demo



TEXTURES



Recognise the texture?

- Wrapping defines behaviour of texture, outside $(s, t) \in [0, 1]$
- `GL_REPEAT`: Coordinates wrap around the texture $(-0.2 \rightarrow 0.8)$
- `GL_MIRRORED_REPEAT`: Coordinates wraps with mirrored value $(-0.2 \rightarrow 0.2)$
- `GL_CLAMP_TO_EDGE`: Clamp coordinates to $[0, 1]$, edge value get copied
- `GL_CLAMP_TO_BORDER`: Clamp coordinates to $[0, 1]$, border colour is used
 - `GL_TEXTURE_BORDER_COLOR` needs to be set

```
glBindTexture(GL_TEXTURE_2D, texture);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_MIRRORED_REPEAT);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_MIRRORED_REPEAT);
```


OpenGL
Shader
Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

WRAPPING



GL_REPEAT (default)



GL_MIRRORED_REPEAT

WRAPPING



`GL_CLAMP_TO_EDGE`



`GL_CLAMP_TO_BORDER`

MAGNIFICATION

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Magnification filter defines behaviour of texture when a texel covers more than one pixel
- `GL_NEAREST`: Use the nearest texel only
- `GL_LINEAR`: Use linear interpolation of four closest texels

```
glBindTexture(GL_TEXTURE_2D, texture);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
```

- (Minification is discussed in EDAN35 😊)

OpenGL
Shader
Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

MAGNIFICATION



GL_LINEAR (default)



GL_NEAREST

Seminar Exercise 3-2: Texture Mapping

- 1 Inspect the values of texture coordinates by outputting the `var_tex` vector as the red and green channels.
- 2 Use `texture()` to replace the color in the fragment shader with the pixels from the texture image.
- 3 Change the 1.0 values for the `s` texture coordinate in the vertex buffer to 2.0. The flower should repeat horizontally.
- 4 Change the `t` texture coordinate as well so a grid of flowers form.
- 5 Try `bricks_seamless.jpg` and study the behavior of it for different wrapping modes. How does it compare to `flower.jpg`?

OpenGL

Shader

Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

FOLLOW-UP QUESTION

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

1 What happens for non-integer values of s and t ?

OpenGL
Shader
Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

Shading theory

PHONG SHADING

- Phong shading per pixel
- Assuming normalized vectors

$$\text{color} = \text{ambientColor} + \text{diffuseColor} * \max(\mathbf{n} \cdot \mathbf{L}, 0) + \text{specularColor} * (\max(\text{reflect}(-\mathbf{L}, \mathbf{n}) \cdot \mathbf{V}, 0))^{\text{shininess}}$$



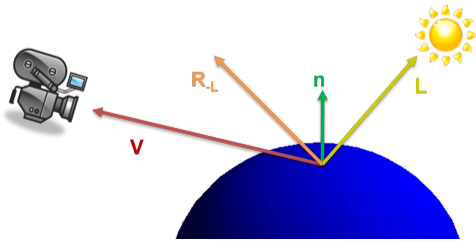
ambient



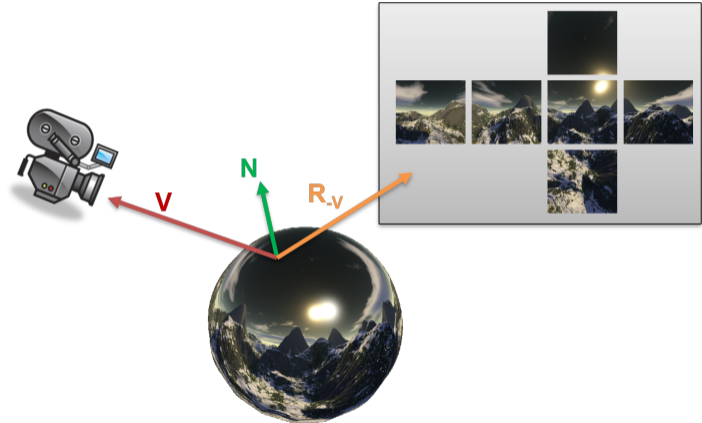
ambient +
diffuse



ambient +
diffuse +
specular



CUBE MAPPING



OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

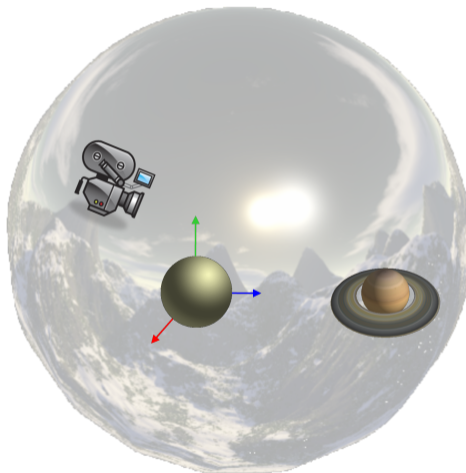
Normal mapping

Roughness mapping

Assignment 3

Hints

Demo



OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Attach cube map to large sphere (large enough to contain all scene content)
- Write a “skybox”-shader that looks up colours from the cube map
 - in the normal direction,
 - or with world position coordinates(?)
- Create illusion of sphere placed at infinity
- Make sure back-face culling is disabled
- Inside of sphere will appear as surrounding sky/landscape

ADDING A CUBE MAP

- Complete `loadTextureCubeMap()` and use it

```
GLuint cubemap = bonobo::loadTextureCubeMap(  
    config::resources_path("cubemaps/NissiBeach2/posx.jpg"),  
    config::resources_path("cubemaps/NissiBeach2/negx.jpg"),  
    config::resources_path("cubemaps/NissiBeach2/posy.jpg"),  
    config::resources_path("cubemaps/NissiBeach2/negy.jpg"),  
    config::resources_path("cubemaps/NissiBeach2/posz.jpg"),  
    config::resources_path("cubemaps/NissiBeach2/negz.jpg"));
```

- Add a cube map to your sphere node

```
sphere.add_texture("cubemap", cubemap, GL_TEXTURE_CUBE_MAP);
```

- `node.cpp` will bind the cube map and set the uniform (check code)

ADDING A CUBE MAP

- In your fragment shader add

```
uniform samplerCube cubemap;
```

- Then use it in your fragment shader

```
frag_color = texture(cubemap, coordinates);
```

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

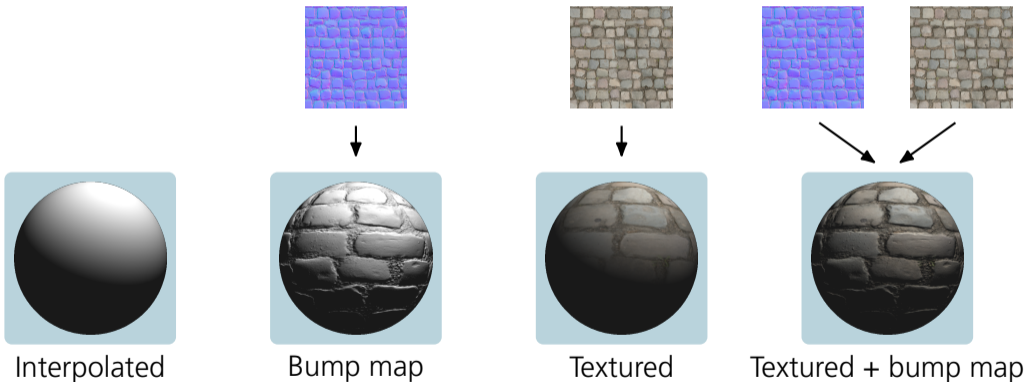
Assignment 3

Hints

Demo

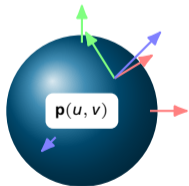
NORMAL MAPPING

- Obtain normal from normal map instead of interpolation from vertices
- Requires a defined *tangent space*



TANGENT SPACE

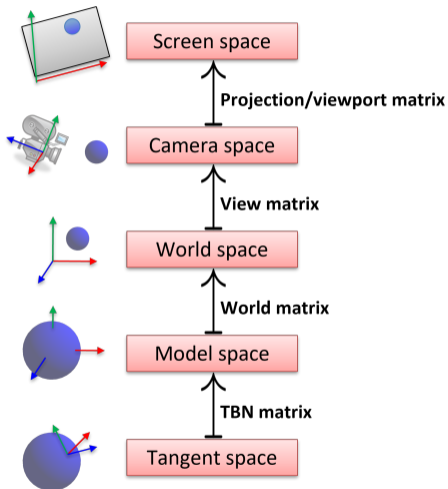
- Basis vectors: tangent \mathbf{t} , binormal \mathbf{b} , normal \mathbf{n}
- Basis matrix: **TBN**
- Derived from the surface equation, $\mathbf{p}(u, v)$
 - Assignment 2!



$$\mathbf{TBN} = \begin{pmatrix} t_x & b_x & n_x \\ t_y & b_y & n_y \\ t_z & b_z & n_z \end{pmatrix}$$

tangent space \mapsto model space

SPACES OVERVIEW



OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

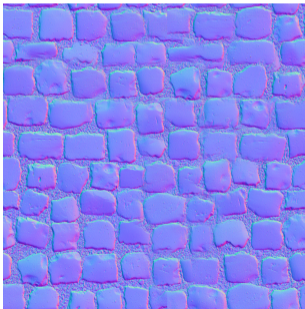
Assignment 3

Hints

Demo

NORMAL MAP LOOK-UP

- Normals stored as (r, g, b) where each component lies in $[0, 1]$
- Map to $[-1, 1]$: $\mathbf{n} = (r, g, b) \times 2 - 1$
- Blue-ish colour since “unmodified” normal, $\mathbf{n} = (0, 0, 1)$, maps to $RGB = (0.5, 0.5, 1.0)$



NORMAL TRANSFORMATIONS

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Normal not ready to use yet; light vector is in world space, so normal must be as well
- Transform normal from tangent to world space:

$$\mathbf{WORLD}^{-T} * \mathbf{TBN} * \mathbf{n}$$

- Available as `normal_model_to_world` in Bonobo shaders (check `node.cpp`)
- Now we can proceed with light calculations

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

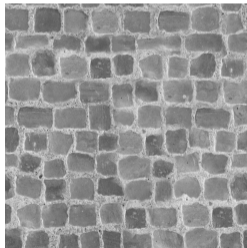
- 1 Look up (r, g, b) from texture
- 2 Create \mathbf{n} by mapping from $[0, 1]$ to $[-1, 1]$
- 3 Transform to world space: $\mathbf{WORLD}^{-T} \cdot \mathbf{TBN} \cdot \mathbf{n}$
- 4 Use this normal when performing light calculations
 - Don't forget to use normalised vectors!

ROUGHNESS MAPPING

- Similar to normal mapping
 - Store material properties in a texture
 - During lighting calculations, do texture look-ups and apply value
- Replace specular component of Phong model
- Many other material properties are also common to store in textures



Diffuse texture



Roughness texture

OpenGL
Shader
Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

Assignment 3

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Implement the following shader techniques:
 - Phong shading
 - Cube mapping using a skybox
 - Normal mapping
- Finish implementation of cube map loading
- Files you have to modify (or add)
 - `src/EDAF80/assignment3.cpp`
 - `src/EDAF80/parametric_shapes.cpp`
 - `src/core/helpers.cpp`
 - Any shader files that you created

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Create new shader programs as needed
 - `skybox.vert`, `skybox.frag...`
- Use **R** key to do hot reload of shaders
 - Performs read from disk and compilation
 - Check log for error messages
- In `shaders/EDAF80/`, look at `diffuse.vert` and `diffuse.frag` for guidance

NORMAL MAPPING IN SHADER

- All tangent space basis vectors are needed (normal, tangent, binormal)
- Complete vertex definition:

```
struct Vertex {  
    float    x, y, z,          /* vertex position */  
            s, t,             /* texture coords */  
            nx, ny, nz,       /* normal */  
            tx, ty, tz,       /* tangent */  
            bx, by, bz;       /* binormal */  
};
```


NORMAL MAPPING IN SHADER

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

- Hint:
 - Set **t**, **b**, **n** as `in`-attributes to the vertex shader and make them vary per fragment using `out`
 - Construct **TBN**-matrix in fragment shader using `mat4`
- Check out `res/textures/` for more textures

OpenGL Shader Language

Types, operators and
flow

Development tools

Example: Diffuse
shader

Interpolation

▲ Exercise 3-1

Textures

Wrapping

Magnification

▲ Exercise 3-2

Shading theory

Phong shading

Cube mapping

Normal mapping

Roughness mapping

Assignment 3

Hints

Demo

