ETH zürich

computer graphics laboratory

Prof. Dr. Markus Gross

# Visual Computing

## Exercise 11: Lighting & Shading

**Hand-out Date: 3 December 2024**

## Goals

- Understand the Phong reflection model
- Learn how to use textures in the fragment shader
- Implement normal mapping and environment-based lighting

## Resources

The lecture slides and exercise slides are accessible via the Visual Computing Course Web Page. The following resources may additionally help you with solving the exercises:

- Mozilla tutorial on textures
- glmatrix API
- WebGL Skybox Example

ETH zürich

computer graphics laboratory

Prof. Dr. Markus Gross

# Tasks

## 1. Phong Reflection Model

The Phong reflection model computes a diffuse term, which is independent of the position of the viewer, and a viewer-dependent specular term. In this exercise, you will compute these two reflection terms in the fragment shader.

a) Implement the function `computeDiffuseSpecIntens` in `fragmentShader.js`. Given the position of the pixel being rendered, the normal vector at the pixel, the position of the camera, and the position of the light source in world coordinates, compute the amount of diffuse and specular light.

b) Test different values for $n$. What changes visually?

**Hint:** If the rotation around the skybox distracts you during debugging, you can simply comment out lines $404 - 409$, which apply a rotation to the skybox' view matrix.

## 2. Texture Mapping

In this exercise, we replace the baubles' constant color by colors read from a Christmas texture (see `textures` folder). We use one texture image for all three baubles.

a) Identify and understand the lines of code responsible for reading the texture image in `ex10.js` (i.e., function `loadTexture`). Additionally, take a look at lines $330 - 345$. Here, we pass the texture coordinates into the buffer and create a sampler object that stores the sampling parameters for texture access inside of a shader.

b) Implement the missing code to read the texture color `txtColor` from the texture sampler `uSamplerTexture` using the `texture2D` command.

## 3. Normal Mapping

In this exercise, we add micro-geometry to the right bauble. This can be achieved using normal mapping.

a) Implement the code for computing tangents and bitangents in `ex10.js`. We already computed the matrix $M^{-1}$ for you.

b) Fill in the missing code for computing the perturbed normals in the fragment shader by looking up the RGB color values of the normal map and using them to perturb the tangent, bitangent and normal. Be careful: normal maps usually range from $[0, 1]$. For computing the perturbed normal, we need to adjust the range to $[-1, 1]$.

## 4. Reflection Mapping

For reflection mapping, we use a 2D image to produce realistic reflections on the surface of the baubles. We capture the environment surrounding our scene using a cube map. A cube map consists of six images, where each image captures the view into the environment from six different directions corresponding to the sides of a cube (i.e., top, bottom, left, right, back, front). To ensure that you actually have a realistic representation of the environment instead of a black background, we already implemented a skybox. However, using the actual 3D representation of a box causes depth issues since the corners of the box will be further away than the sides. Thus, we realized the skybox in form of a quad, which is also computationally more efficient.

a) Implement reflection mapping in `fragmentShader.js` by reflecting the camera-to-pixel vector at the normal and using the reflected direction for a texture lookup into the cubemap.