ETH zürich

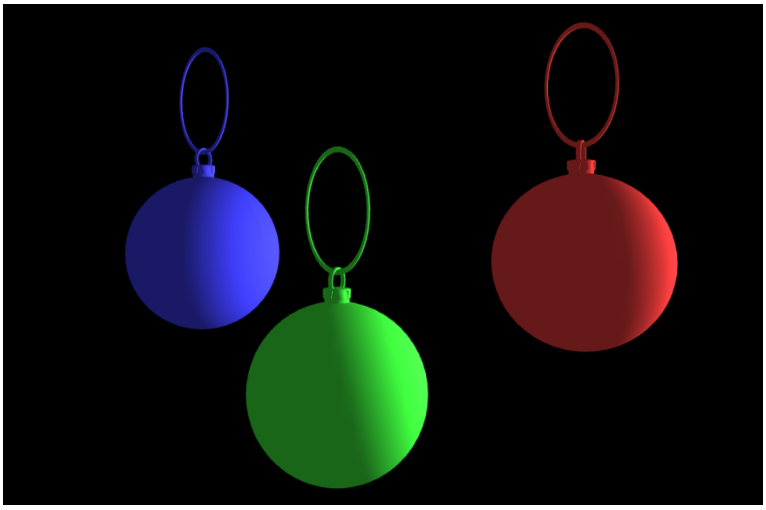
cgl computer graphics laboratory

Prof. Dr. Markus Gross

Visual Computing

Exercise 06: Introduction to WebGL

Hand-out Date: 05 November 2024



Goals

- Getting acquainted with the WebGL graphics library API.
- Learning about the GL graphics pipeline, shaders, and buffers.
- Understanding how to apply colors and simple transformations.

Resources

The lecture slides and exercise slides are accessible via the Moodle.

Tasks

1. Mesh Setup and Initialization

In this exercise, we will set up a mesh (Christmas decoration) for rendering. All code needed for this project is contained in the main file `ex06.js` and the shader files `vertexShader.js` and `fragmentShader.js`.
First, get acquainted with the program structure. Then, have a look at the shader code. Note which attributes and uniforms are used in the fragment and vertex

shader. Once you familiarized yourself with the code structure, complete the following two tasks:

- **Binding the Data**: Locate the part of the code marked as "Task 1a". Insert code to create buffers for all the necessary vertex attributes (i.e., vertex positions and normals) and an index buffer for the mesh faces. Next, identify the per-vertex attribute inputs of the vertex shader and bind them to the correct array pointer (the procedure is covered in the exercise slides).
- **Uniform Variables**: Locate the part of the code marked as "Task 1b". Identify the uniform variable inputs of the vertex shader and bind them to the correct pointer (the procedure is covered in the exercise slides). Finally, draw the mesh using `gl.drawElements(...)`. When implemented correctly, the mesh will be displayed on the screen (still all grey as no lighting is performed yet).

2. Normals for Lighting

In this part, we will add simple lighting in order to see the details of the 3D mesh. In the vertex shader, a simple diffuse lighting method is already implemented. This code, however, requires per-vertex normals to calculate the lighting. Since the vertex normals cannot be computed directly, we approximate them as follows (see code marked as "Task 2"):

- **Face normals**: First, calculate the face normal for each triangle based on the vertex positions that correspond to each face by following the steps given as comments in the code.
- **Vertex Normals**: Next, approximate the vertex normals by averaging all the face normals around that vertex.
- **Weighting (optional)**: Instead of uniformly averaging the normals, we can achieve a better approximation using weighted averaging. For this, multiply each face normal with the angle it covers around that vertex (see the exercise slides for details).
- **Normalization**: In a final step, we iterate through the calculated vertex normals and normalize them to unit length. When implemented correctly, you should see a light coming from the right.

3. Color and Rotation

In this part, we will color the mesh and rotate it. Note that the vertex color is currently set to `vec4(0.3, 0.3, 0.3, 1.0)` in the vertex shader. We want to extend the vertex shader to take a custom vertex color as attribute, which should be precomputed in the main code as follows:

- **Coloring**: Extend the mesh with an additional attribute called `vertexColors` for storing the color in the code marked as "Task 3a". Instead of giving the same color to all the Christmas baubles, color each of them differently (Hint: the mesh can be split up into three regions that separate the baubles using two planes going through $y = -8$ and $y = 9$). Then, add a new attribute `aVertexColor` to the vertex shader and extend the code in "Task 1a" to create a buffer for the new vertex colors and bind them analogously to the other attributes used in the vertex shader (see the procedure in 1a). Finally, do not forget to use the new `aVertexColor` instead of the constant color inside the vertex shader. If implemented correctly, the three baubles will appear in different colors.
- **Rotation**: In order to rotate the mesh, apply a rotation around the z-axis to the `modelViewMatrix` in the code marked as "Task 3b".