Introduction to Computer Graphics and Animation

Exercise 3 of 5

Prof. Dr. Dennis Allerkamp – December 4, 2024

3.1 Loading Wavefront OBJ

So that you can display more interesting objects in your programs, you need a function to load geometry data.

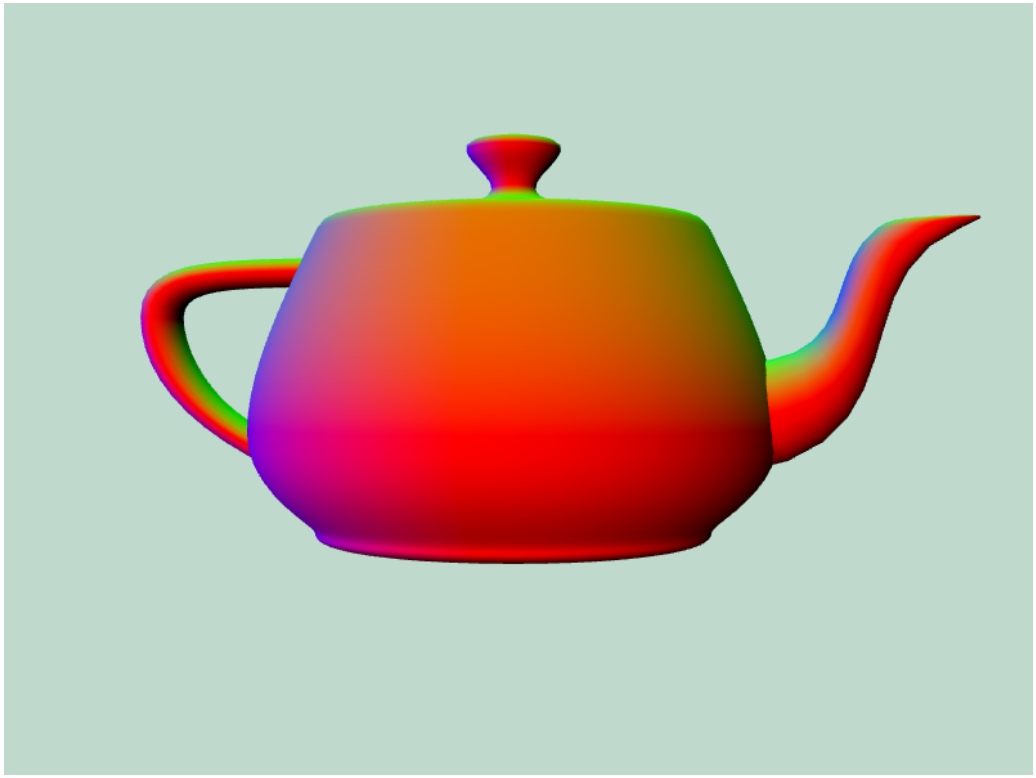
The OBJ format is a very simple format for geometry data, which can be written by many programs (for example, Blender also has a corresponding export).

- Get an overview of the OBJ format in the Wikipedia article Wavefront .obj file. A sample file (teapot.obj) can be found in the course materials.
- Write a function that reads an OBJ file and generates a vertex buffer with the vertex data contained in it. Since OBJ files can have different indices per vertex, it is easiest to completely dispense with index arrays. Your function should:
  - Read the OBJ file line by line
  - Break each line down into its individual parts
  - For lines that start with `v`, `vn` or `vt`, store the data in separate lists
  - For lines that begin with `f`, read the data from the separate lists with the respective indices and write it into a vertex buffer
- Draw the read object with OpenGL. Pass the values of the vertex normals as a color attribute to the shader.

Note that the geometry data must be triangulated for use in OpenGL, i.e., the faces must all consist of exactly three vertices, so that they can be drawn with `GL_TRIANGLES`. (This is given in the sample file teapot.obj.)

If you encounter problems programming this function, you will find a simple implementation in the course materials.

If you did everything correctly, your result should look like the following figure.



3.2 Shading

In this task, you are to implement Phong shading. The previous task serves as the base.

- Calculate a normal matrix from the model and the view matrix and also pass this as uniform to the vertex shader.
- Add an attribute variable for the normals to the vertex shader and link this to the corresponding data from the OBJ file.
- Transform the normals in the vertex shader with the normal matrix from object coordinates to eye point coordinates.
- Pass the transformed normals in the vertex shader as varying variables to the fragment shader. Don't forget to normalize the normal vector in the fragment shader.
- Implement the Phong lighting model in the vertex shader. You can pass the material constants as additional uniform variables to the shader. Make sure that all vectors used for the calculation are in eye point coordinates and are normalized.

Your result may look like this: