CSE40166/60166 — Computer Graphics
Assignment 8 — Rendering Realistic Orange

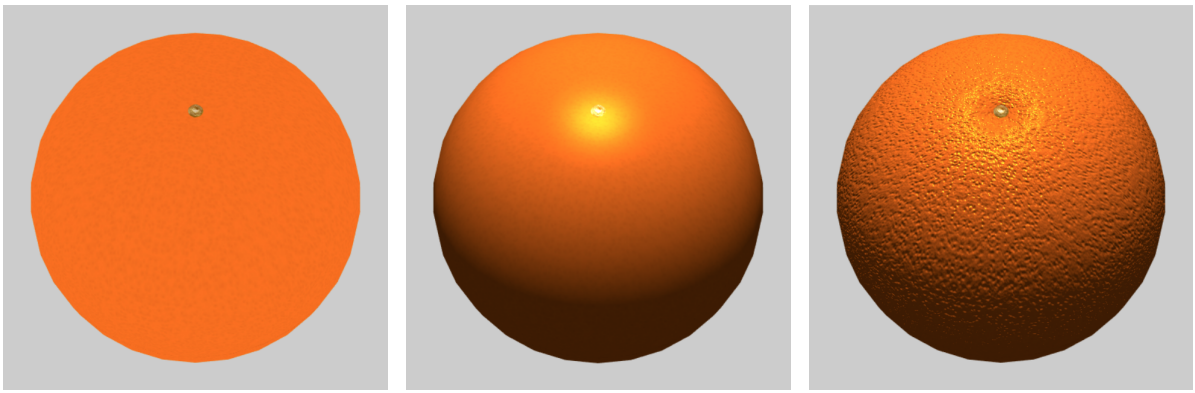
Assigned: 11/14/2018 Due: 11:59PM 11/26/2018

The purpose of this assignment is to get you familiar with advanced lighting and texturing using normal map. The mesh representation is captured in the well-known Wavefront ".obj" file format.

Here's a Wikipedia page that documents the file format:

`http://en.wikipedia.org/wiki/Wavefront_.obj_file`

We expect you to understand the .obj file format to the degree you need to work with mesh geometry, with texture coordinates referring to a single texture image, and with surface normals. Specially, you need to understand vertices ('v'), vertex normals ('vn'), vertex texture coordinates ('vt'), and faces ('f').



(a) (b) (c)

Figure 1: 3D orange rendering. (a) texture mapping only. (b) texture mapping and lighting. (c) texture mapping, lighting, and normal map.

In this assignment, you are asked to implement a WebGL program that realizes 3D mesh model rendering, featuring lighting and texturing with the use of normal map. See Figure 1.

You are given a complete code `orange-objParser.js` that implements the obj file reading for you. Specifically, this parser reads in vertices, vertex texture coordinates, and faces. Because the model we read in represents a sphere, vertex normals are computed as normalized vertex positions. **Note that vertex normals computed here are used in eye-space lighting computation in Task 1, and in forming the TBN tangent space in Task 2.** The parser also breaks down polygonal faces into triangles. For each triangle, the parser computes the per-vertex tangent and bitangent vectors for used in tangent-space lighting computation. The main reason for us to go to the tangent space is because in the normal map, normals are stored in the tangent space. **Note that normals stored in the normal map are used in tangent-space lighting computation in Task 2.** Please go through and understand the template code (i.e., to the extent that you know how to use the information that has been read out or computed).

You are asked to complete the following tasks:

- Task 1 (*35 pts*): The template code `orange-objViewer-template.js/html` has implemented texture mapping for you. Based on this code, further implement texture mapping with lighting. Assume the light position in the eye space is at `(0.0, 500.0, 0.0)`. In the vertex shader, compute vertex normal, eye direction, and light direction in the **eye space**. Passing down these three vectors (eye-space light direction L_e, eye direction E_e, and normal N_e) to fragment shader using `varying` qualifiers. In the fragment shader, normalize the three vectors. Implement **per-fragment lighting** in the **eye space**. Compute the per-fragment Phong lighting: including ambient, diffuse, and specular components. For the specular component, compute reflector direction R_e and use the *original* Phong model. Using per-fragment light to modulate fragment color. Refer to the example `code/09/shadedSphere2` for implementing per-fragment lighting in the eye space.

- Task 2 (*40 pts*): Further implement texture mapping, lighting, and normal map. Specifically, in the vertex shader, compute vertex tangent T, bitangent B, and normal N vectors in the **eye space**. Note that since we only use uniform scaling in the transformation, we can simply use the 3×3 **modelview matrix** to transform T, B and N from object space to eye space. Then use these three vectors to construct the **tangent space** TBN (a 3×3 matrix):

```
                    mat3 TBN = mat3(
vec3(vertexTangentES.x, vertexBitangentES.x, vertexNormalES.x),
vec3(vertexTangentES.y, vertexBitangentES.y, vertexNormalES.y),
 vec3(vertexTangentES.z, vertexBitangentES.z, vertexNormalES.z)
                              );
```

  where `vertexTangentES`, `vertexBitangentES`, and `vertexNormalES` are the tangent, bitangent, and normal vectors in the eye space. After that, transform light direction and eye direction from the eye space to the tangent space:

```
v_LightDirectionTS = TBN * lightDirectionES;
  v_EyeDirectionTS = TBN * eyeDirectionES;
```

  where `v_LightDirectionTS` and `v_EyeDirectionTS` are the light and eye directions in the tangent space. Passing down tangent-space light direction L_t and eye direction E_t to fragment shader using `varying` qualifiers. In the fragment shader, texture lookup the tangent-space normal N_t using the same texture coordinates given for color lookup. Note that each component c in the normal (x, y, z) looked up from the normal map is in $[0, 1]$, for which you need to rescale it to $[-1, 1]$ (i.e., $2.0 \times c - 1.0$) in order to get the correct value. Normalize the three vectors (tangent-space light direction, eye direction, and normal). Implement **per-fragment lighting** in the **tangent space**. Compute the per-fragment Phong lighting: including ambient, diffuse, and specular components. For the specular component, compute reflector direction R_t and use the *original* Phong model. Using per-fragment light to modulate fragment color. Refer to the following online tutorial for understanding normal mapping and tangent space computation:

  `http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-13-normal-mapping/`

- Task 3 (*20 pts*): Create three radio buttons for interacting with three rendering options: Texture (refer to Figure 1 (a)), Texture + Lighting (refer to Figure 1 (b)), and Texture + Lighting + Normal Map

(refer to Figure 1 (c)). Refer to the example `code/04/rotatingSquare4` for implementing radio buttons and the example `code/10/earth-texture-lighting-specular` for sending uniform qualifier to the shader (check code for the "Use lighting" checkbox).

You will receive the remaining *5 pts* if your program can run correctly and the `README` file provides sufficient information for the grading. Please zip or tar your code and the auxiliary files (if any, such as the ones in the `Common` folder as used in class), along with a `README` file into `assignment-8.zip` or `assignment-8.tar.gz`. Submit this single package to your Notre Dame Box (`https://box.nd.edu`) submission directory.