CS 300 | Advanced Computer Graphics I
Assignment 2 | Normal Mapping

Description

In this assignment, the main task to apply the normal mapping to the Phong illumination model implemented on assignment 1. The parsed scene format will have all the content used in assignment 0 with new added data listed below:

- Material data:
  - Objects will have an extra parameter for the material:
    - `normalMap`: specifies the normal map texture applied to this object. Default value is `data/textures/default_normal.png`.

The application should:

1. Read the input file
   a. Load/Generate mesh data
   b. Generate Texture
2. Upload data to OpenGL
3. Accept from the keyboard the user input in order to move the camera
4. Update objects/light with the corresponding animation
5. Build the camera and perspective transformation according to the user input
6. Set shader program and uniforms
7. Draw each mesh
   a. **Vertex Shader**:
      i. Transform N from model to camera space by multiplying it by the normal matrix.
      ii. Transform T and B from model to camera space by multiplying it first by the model matrix, then by the view matrix.
      iii. Output those vectors to the fragment shader.
   b. **Fragment Shader**:
      i. Given the fragment's texture coordinates, read the normal from the normal map texture and transform it to the correct range of values.
      ii. Build the matrix to represent the transformation from tangent space to camera space.
      iii. Use the modified normal, the light and view vectors to implement the Phong Reflection Model (note that all three vectors should be in camera space at this point).
8. Finish frame

Tangent Space

- For each one of the shapes generated or loaded in previous assignments (plane, cube, cylinder, cone, sphere and loaded meshes):
  - For each triangle, generate the tangent and bitangent vectors using the vertex positions and texture coordinates. Incorporate lab 3 to your framework.

- Update the tangent T to make sure that it is orthonormal to the normal to the geometry by implementing the Gram-Schmidt orthonormalization process.
- Transfer to the GPU as per-vertex data the normal, tangent and bitangent. Again, remember that these two vectors are in model coordinates.
- Render N, T and B per vertex using different colors for each line. Normals as blue, tangents as red and bitangents as green.

Material Properties

- Most material properties are hardcoded with the following values:
  - Texture to map on shape (when texturing is disabled use UV as color)
  - Ambient and Diffuse color (texture color)
  - Specular color (always white)
  - Shininess read from the scene file
  - Ambient coefficient will be 1
  - Normal map read from the scene file

Light

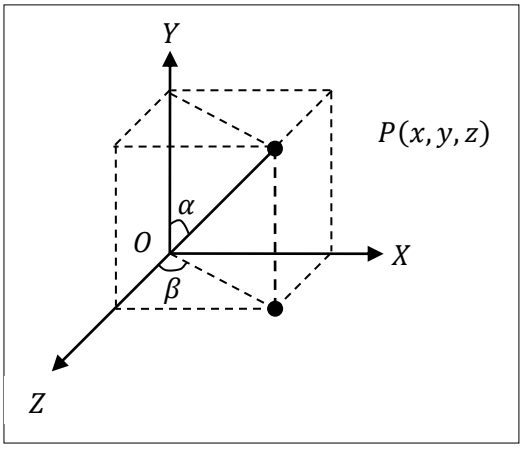
- Requirement:
  - One light properly lighting the scene. If the scene file contains multiple lights and they are not supported use the first one loaded.
  - Extra credit will be rewarded if multiple lights are supported up to a maximum of 8 lights. If the scene file contains more than 8 lights keep only the first 8 listed.
- Properties:
  - Type of light
  - Its position and/or direction.
  - Light color. Specular component will be white.
  - Attenuation coefficients (for the spotlights this includes the angular attenuation parameters).

Shader Management

- Shader code MUST be loaded from an external file, i.e. you must NOT embed the shader code within your C++ code.
- Compile and install the shader code to be used. Should the shader/program fail to compile or link, you must print out the error messages to the console or the main window.
- The error message MUST specify which shader file(s) is causing the problem. (**Hint**: implement a *Shader Manager* class that encapsulates the one or more program objects – a program object contains a vertex and a fragment shader; also think that you might want to support more than one shader program in your application).

Input

- Camera will be controlled using spherical coordinates defined in the diagram. The target will be the origin of that coordinate system. The input should alter r, α and β as follows:
    - W: Make α angle smaller. It should move the camera towards the top of the target.
    - S: Make α angle greater. It should move the camera towards the bottom of the target.
    - A: Make β angle smaller. It should move the camera towards the left rotating around the target.
    - D: Make β angle greater. It should move the camera towards the left rotating around the target.
    - E: Make r greater. It should move the camera away from the target.
    - Q: Make r smaller. It should move the camera closer to the target.



- N: Toggle normal/tangent/bitangent rendering
- T: Change between rendering modes:
    - Normal Mapping: Scene with lighting and normal mapping with checkerboard texture
    - Normal: Geometry normal in camera space as color
    - Tangent: Geometry tangent in camera space as color
    - Bitangent: Geometry bitangent in camera space as color
- F: Toggle face/averaged normal
- M: Toggle wireframe on/off
- +/- or Z/X: Increase/reduce number of slices (4 is the minimum number of slices)

Assignment Submission

Please refer to the syllabus for assignment submission guideline. Failure to the submission guidelines correctly might cause you to lose point.

Grading Rubrics

The following is a rough guideline on how your assignment will be graded and the weight of each part.

Feature	Grade %
Tangent and Bitangent calculation	30%
Correct tangent/bitangent transformation and usage	30%
Normal map load and usage	20%
Rendering modes and presentation	20%