02561 COMPUTER GRAPHICS IMM . DTU

Exercise 02561-10	**Environment Mapping and Bump Mapping**
Readings	Angel: chap. 8.10, 9.12, 9.13
Purpose	The purpose of this exercise is to become familiar with the concepts behind environment mapping and bump mapping. We will use environment mapping to simulate mirrors and glass and in the process learn to use cube maps. Finally we will apply bump mapping to a sphere to add small scale details. The file `02561-10-2008.zip` contains some files to help you get started. These files include the textures used in the exercise and some C++ source files and GLSL shaders. Place these files in the `exercises` folder in the framework you downloaded for exercise 8 (`02561-08-2008-stud.zip`).
Part 1	Load a cube map using OpenGL. Create a cube map texture and fill the six faces with the PPM files from the `textures` folder. Setup texture filtering and wrapping modes.
Part 2	Draw a skybox using the cube map. To do this, draw a box using `glutSolidCube` centered on the origin using the cube map as the texture. You will need to use shaders to draw the box (use `background.vert` and `background.frag`.) In the fragment shader you can use the object space position to do the cube map texture lookup.
Part 3	Draw a mirror sphere. To do this, draw a sphere using `glutSolidSphere` and using `glass.vert` and `glass.frag` as shaders. In the fragment shader compute the reflection direction using GLSL' `reflect` function and do a lookup in the cube map in this direction.
Part 4	Draw a glass sphere. To transform the mirror sphere into a glass sphere we just need to add refraction. In the fragment shader compute the refraction direction using GLSL' `refract` function and do a lookup in the cube map in this direction. Use Schlick's Fresnel approximation to blend between reflection and refraction (see slides or Angel p. 489).

Part 5	Load the normal map. Load the normal map from the `textures` folder and generate an OpenGL texture. Optionally create mipmaps.
Part 6	Index the normal map. Modify the glass shader to compute the spherical coordinates (θ, φ) of the object space position, and use these as texture coordinates into the normal map. Be sure to scale them to the [0; 1] range. Do a lookup in the normal map and transform the resulting color to a vector using the procedure described in class. The result is the shading normal in tangent space.
Part 7	Compute the tangent vectors. The tangent vectors and the geometric normal form an orthonormal basis in which the shading normal is defined. Compute the tangent vectors in the fragment shader by taking the partial derivative of the object space position with respect to each of the spherical coordinates. Organize these basis vectors in a 3×3 matrix. Finally, compute the bump mapped normal as the product of this matrix and the shading normal.
Optional	Add chromatic dispersion. Dispersion is the optical effect responsible for rainbows. It is caused by a wavelength dependant index of refraction. To simulate this, do three lookups in the cube map, one for red, green, and blue, each with different index of refraction, when computing the refraction color.

AWK.12.11.2008