Lab 7 - Texture and normal mapping, environmental and refraction

Laura Mazzuca - matr. 0000919489

16/09/2021

0 Assignments

1. Torus mappings
   (a) texture mapping 2D on the torus
   (b) texture mapping 2D + Phong shader on the torus
   (c) procedural mapping on torus
2. Normal mapping
3. Environment mappings
   (a) skybox
   (b) reflection
   (c) refraction
4. semi-transparent objects

1 Torus mappings

1.1 Texture Only

To apply a Texture Only Shader to the torus the mapping of the texture coordinates was added to the $computeTorusVertex()$ function. This mapping is done assigning to each vertex texture coordinate of the torus $(\theta/\pi, \phi/\pi)$.

1.2 Phong Texture Shader

To apply a Phong Texture Shader the **f_texture_phong.glsl** was implemented. This implementation only required to copy-paste the lab-03 Phong shader and change the material.diffuse parameter with the operation to obtain the rgb of the texture in this fragment.

1.3 Procedural

The procedural texture was generated as Perlin noise on the Red Brick color (203, 65, 84) in the function **compute_torus_procedural_texture()**.

2 Normal

To implement the normal mapping, the Tangent, Bi-tangent and Normal inverse matrix, defined in the **v_normal_map.glsl** and used to translate world space vectors into tangent space, then forwarded to the fragment shader **f_normal_map.glsl**. In the ladder, the phong algorithm was implemented by sampling the normal from the Normal Map and sampling the color from the Texture Map. The normal also needed to be converted from the [0,1] space to [-1,1]. The two variables were then substituted into the algorithm computing the fragment color. The visual effect was then applied to the column and the rock.

3 Environment mappings

3.1 Skybox

The only change that was made was to center the skybox in (0,0,0) and uncomment the *reverse* function applies to $surface.vertices$.

3.2 Reflection

The reflection shader was implemented in **v_reflection.glsl** and **f_reflection.glsl**. In the first one the normal and the position are computed. Note that the Normal is computed only with respect to M, so in World Coordinate System, and only after the View and Projection matrices are applied. Finally, in the fragment shader we need to compute the reflection ray and assign the color to the fragment with respect to the point hit on the cube map.

3.3 Refraction

The reflection shader was implemented in **v_reflection.glsl** and **f_reflection.glsl**. The vertex shader is the same as the reflection one, while in the fragment shader the ray is computed by ging in input to the $refract$ function the Snell ratio for materials.

4 Semi-transparent objects

To render semi-transparent objects, the blending in fragment rendering was enabled. To do so the $glEnable(GL\_BLEND)$ was added in the *init* function.