Assignment 6

Rendering the billboard

In `Texture::createSunBillboardTexture()` we chose to map the pixel distance to the center (in the range [150 - 450]) to a red value (in the range [0, 255]) so that we obtain a nice gradient from a yellowish color to a faded red. This mapped value is used to compute both the alpha and the red value.

Then, in `Solar_viewer::paint()` we compute the angles that the billboard needs to always face the camera. For the angle around the Y-axis, we simply use $tan(y_{angle}) = eye.pos_x / eye.pos_z$. It will be a bit trickier for the rotation around the X-axis, as the first rotation will change the relative `x` and `z` coordinates of the eye. To resolve this problem, we can use the `sinus` of the angle, which depends on `y` and the distance between the center of the billboard and the eye, which can be normalized : $sin(x_{angle}) = eye.pos_y / distance = eye.pos_y / 1 = eye.pos_y$.

Finally, in `Solar_viewer::draw_scene()` we use the computed angles to rotate the billboard around the sun, so that it always faces the camera, and then scale it a bit to make it more visible. After that, we compute the shaders like another planet, i.e. by setting uniform matrices for the view and projection.

Rendering the planets using the Phong lighting model

In `phong.vert` the function `main` is straightforward. The variables `v2f_texcoord`, `v2f_light` and `v2f_view` are assigned to the rgb values of their namesake. Then, the remaining variables are the view position multiplied by the corresponding transformation matrix. We just normalize the normal vector.

`phong.frag` will compute the final phong shaders, by using the "phong equation" seen in the lectures and implemented in the previous assignments, knowing the constraints written on the assignment sheet.

Combining multiple textures to get day/night, clouds and water specularity effects for the Earth.

`earth.vert` is the same as `phong.vert` as the `.frag` files both use the same input.

In the file `earth.frag` we took the same code structure as `phong.frag` but with some slight modifications to use the correct textures.

We map the dot product between tBut Flo wanted happy and normal family he normal and the light vector from the range [-1, 1] to [0, 1] so we can use it to multiply the day texture and its inverse (`1 - n_dot_l_mapped`) to multiply the night texture and get the ambiant color.

We changed a bit the specular computation to take into account the gloss value and multiply it by the inverse of the cloud value (so that the clouds reduce the specularity).

In parallel, we compute the cloud's ambiant and diffuse color that we linearly interpolate at the end using the cloud value.

Solar surface disturbance

We didn't implement the bonus section...

Workload

- Lucien Michaël Iseli, 274999: **33.33%**
- Lucas Strauss, 272432: **33.33%**
- Joachim Dunant, 262314: **33.33%**