Assignment 5

Placing the objects in the scene

In the function `update_body_positions()` we created a new helper functions called `compute_planet_position()`, which simply helps us placing a planet by rotating it and translating it, using its angle and distance.

We do this repetitavely for every planet, even the moon. But in the case of the moon we also translate it by the earth position so that it's on the earth's orbit.

Placing the eye in the scene

Here we defined another helper function, `rotate_around()`, that applies one or two rotation matrices to an homogeneous vector and, if needed, applies a translation matrix too.

We use the position of the ship and a small offset to place the eye slightly behind and above it. The direction of the eye will be the same as the direction of the ship.

If the camera is looking at a planet, we begin by offsetting the z coordinate of the eye and then rotating it depending on the `x_angle_` and `y_angle_` parameters. The center of the view should be the planet we're looking at, so `center = pl->pos_`. We rotate the `up` vector the same way as the eye, but without translating it.

Rendering texturing objects

For each planet we scale, rotate and then translate it to the corresponding values, before adding its shading and texture in the same way it was done with the Sun but with the corresponding variables.

Responding to keyboard events controlling the eye

To listen to keyboard inputs for keys 8 and 9, we simply respectively add or subtract 1 to `dist_factor_` and then check that it is still inside the desired bounds [2.5, 20].

To rotate the eye in the case of looking at a planet, we use `rotate_around()`, so that we move the planet at (0, 0, 0), apply our rotations (depending on 'x_angle' and 'y_angle') and go back to where it was. The vector 'up' is updated just by adding the rotation matrices, because we don't care about where it is in our scene, we only need its direction.

In the case of the ship, we compute the rotation of the eye only around the y-axis, in a similar way than before.

Workload

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