**Graphics 2018 – P3: Rasterization**

**Amber Elferink (5491525) & Miriam Sterl (5664187)**

In this file, we will discuss the architecture of our Rasterizer, along with the **bonus** features we implemented.

*Data structure for storing a hierarchy of meshes: Node*To store a hierarchy of meshes in our SceneGraph, we made the class Node. Each Node has a parent-Node (except if it is the root), (possibly) children-Nodes, a Mesh and Texture, a SceneGraph it belongs to, and a matrix defining its position with respect to its parent. The position of the planets are adapted in runtime. In the SceneGraph class, we can transform the nodes to the camera by multiplying their matrix with their parent’s matrix recursively; we do this from the root all the way down.

*Interactive camera*The camera can be controlled by the arrow keys to move in the XZ-plane, Space/Shift in the Y plane, and WASD to look around by rotations. The camera starts inside the sun. To view the space scene, zoom out with the bottom arrow.

*Phong shading model*  
In the fragment shader, we implemented the Phong shading model, with a Phong exponent equal to 1. The shader can handle **multiple lights**, for which we used uniform variables to pass light positions and colors. For this, we first made a Light class; we can use this to define a Light object with a specified position, color and specular color, all of which are vectors of length 3. In the game class, we now have a method passLights() that passes the lights to the shader (in principle, it can handle an arbitrary number of lights; there are 8 lights in our demo). This is done in the following way:

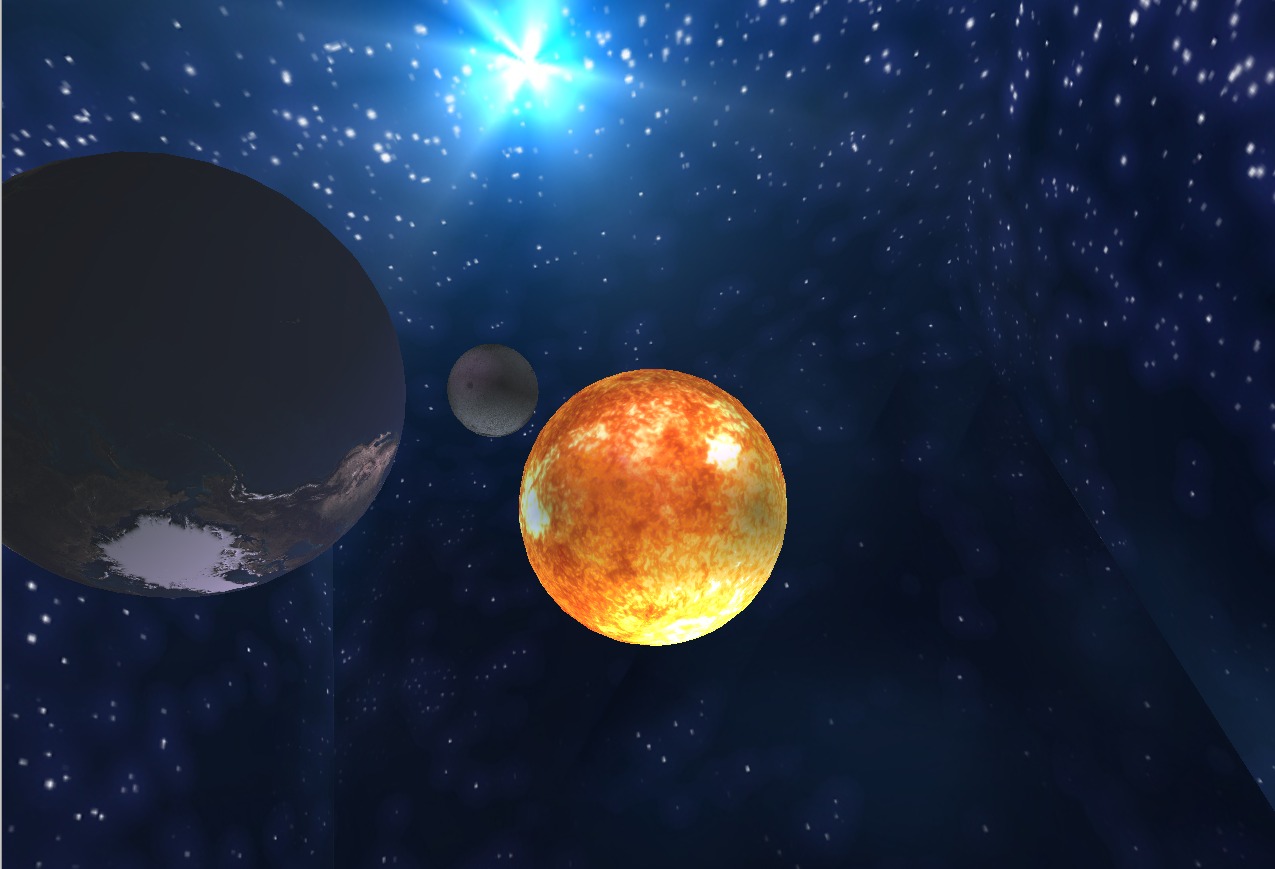
* A Light object is created
* A 3x3 matrix is created, in which the position, color and specular color of the light are stored as rows
* This matrix is transposed, so that the first column of the matrix now represents the position, the second column gives the color, and the third column gives the specular color
* This matrix is passed to the fragment shader, where we retrieve the position, color and specular color of the light by multiplying the matrix with the unit vector in the x-, y- and z-direction, respectively (this yields the first, second, and third column, respectively). With this information, the contribution of the light to the Phong shading is calculated. This is done for all the lights in a for-loop.

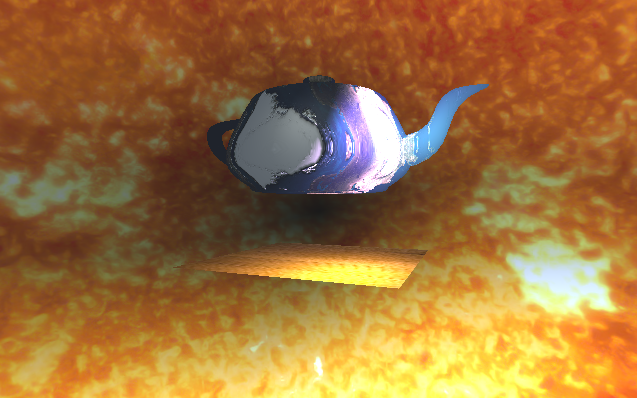
The idea behind this way of passing Lights in matrix form is that we only have to pass one matrix for each light to the shader, as opposed to three vectors per light.

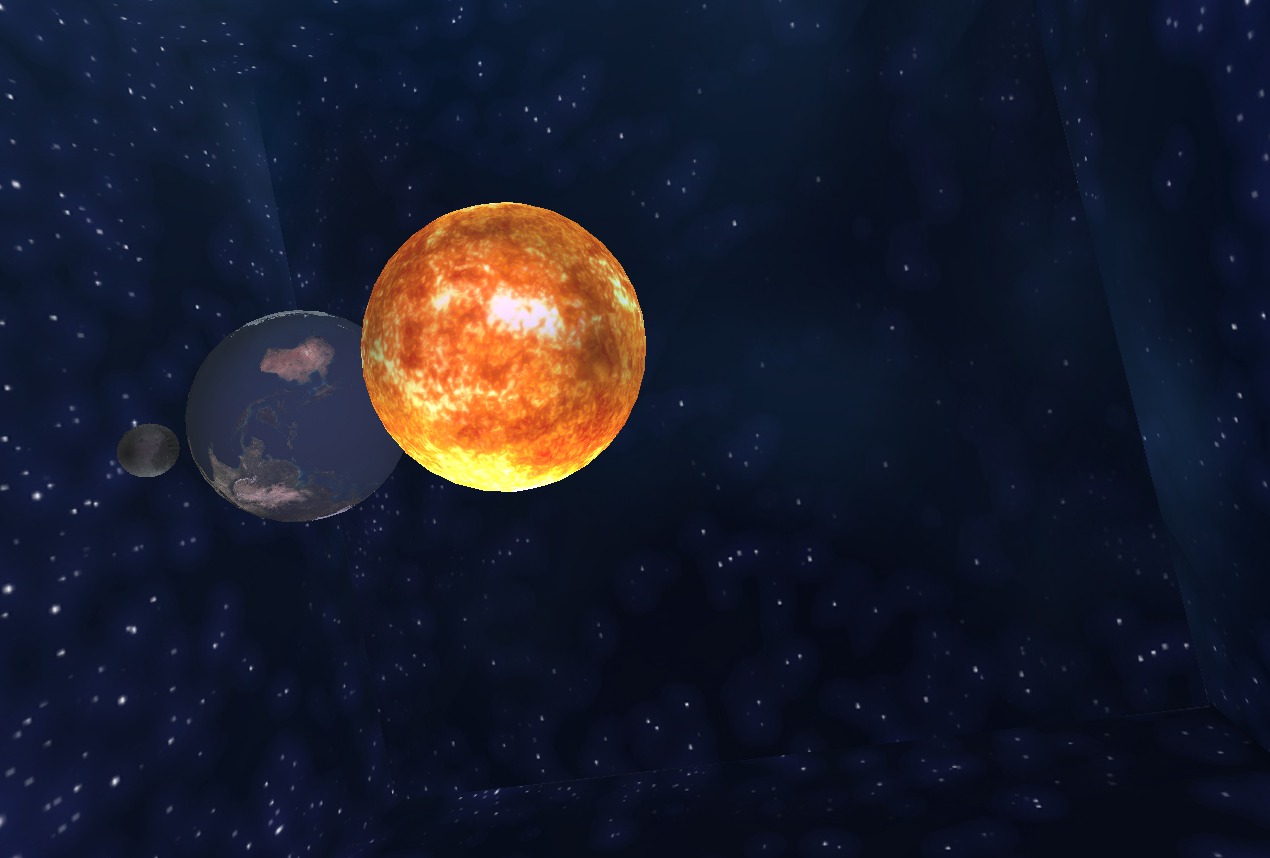
*Normal mapping*  
Every object can (not mandatory) be loaded with a **normal map**. This normal map is then passed as a second texture to the shader in the Render function. The values which are stored in the normal map, will then be used as normals in the shader. This is demonstrated with the teapot and earth inside the scene.

*Skybox*The **skybox** is rendered without the Tworld matrix acting on it, and without shaders, which yields a stunning background of the galaxy.

*Demo*To demonstrate the functionality of our rasterizer, we created a nice scene showing the orbit of the moon around the Earth and of the Earth around the sun, with as extra a rotating teapot and floor inside the sun. Although it might not be entirely physically accurate, it nicely shows the hierarchy of Nodes (the Earth is a child of the sun, so that it rotates around the sun as well as around its own axis; similarly, the moon is the Earth’s child).







*Materials/sources*To implement our rasterizer, we used the following sources:

* The slides from Jacco Bikker’s lectures on rasterization, where some (pseudo)code was provided that we implemented in our rasterizer, in particular for the implementation of the Phong shading model.
* The book used for the course, *Fundamentals of Computer Graphics* by Peter Shirley et al, to understand the concept of the Phong shading model (paragraph 10.2 in the 3rd edition).

3D models used from:

* Earth: <https://www.turbosquid.com/3d-models/earth-clouds-3ds/535678>
* Sun: <https://free3d.com/3d-model/sun-43982.html>
* Moon texture: <https://www.solarsystemscope.com/textures/>
* Skybox
  + image: <http://users.csc.calpoly.edu/~zwood/teaching/csc471/finalW15_1/bclark16/skybox1.jpg>
  + Skybox 3D model: created in Blender ourselves