Hierarchical Structuring with a Scenegraph

The solar system is a good example case for the hierarchical organisation of a set of objects. The absolute position and orientation of a solar object is determined by its orbital movements around a reference (solar) object (see Figure 1) and also the local rotation about ist own axis (see Figure 2).

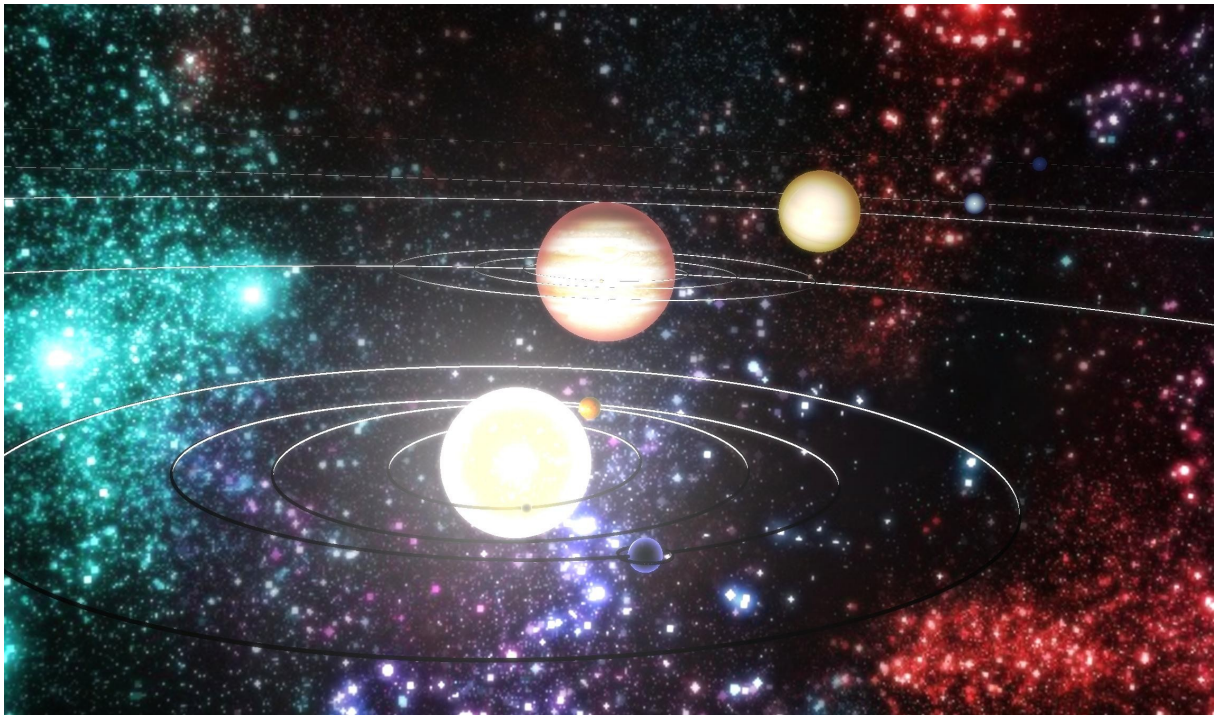

Figure 1: solar system model using multiple SolarObject instances (not true to scale).

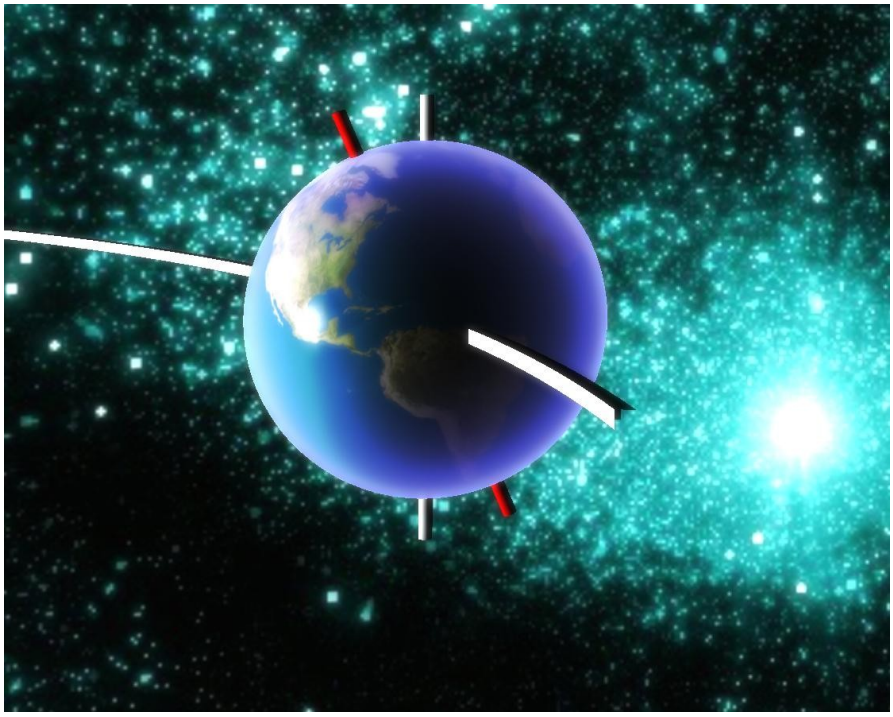

Figure 1: Inclined rotation axis (red cylinder) of a planet.

The *hierarchical_structuring* example contains the class *SolarObject*, which is used to represent all objects in the solar system. The sun, the planets and moons are all instances of *SolarObject* and have to be attached to each other in the correct manner. Therefore *SolarObject allready* already has two internal scenegraph nodes. The first node, *object_geometry*, is used to load external geometries (tri-meshes) from a file. Despite the loader supports multiple 3D file formats, we recommend using the „Wavefront OBJ“ format. In this application a default sphere geometry (1m diameter) is used to represent all solar objects. The varying appearances of the planets and moons are realized through parametrization of the size of the sphere geometry and by applying respective textures to the material of the geometry node. The available textures can be found in the *data/texture* folder.

The second node, *orbit_radius_node*, contains the relative geometric transformation (distance in km) regarding the higher-level solar object. *object_geometry* is a child node of *orbit_radius_node*. The function *update_orbit* is called every frame and accumulates the respective movement input to the local matrix of *orbit_radius_node*. For simplicity the orbit is not implemented as an ellipse but as a circular motion.

A simple navigation interface is already provided in the application. The WASD keys are used for translations of the virtual camera on two axes. PageUp and PageDown enable translation on the third axis. The arrow keys rotate the virtual camera on two axes. Alternatively to the keyboard navigation interface the Spacemouse device can also be used for navigation input. Therefore change the global parameter NAVIGATION_MODE in the *main.py* file.

How to start?

- Copy the *01_hierarchical_structuring* folder from */opt/vr_exercises/WS_16_17* to a local repository
- Execute the application by running ./start.sh in a terminal
- Proceed with the assignments

Assignment Tasks (no grading):

1. Init the following planets and moons in the class *SolarSystem*: Mercury, Venus, Earth, Earth-Moon, Mars, Jupiter and three Jupiter moons (see Figure 1). Respective parameters for size, orbit radius and orbit duration of the planets and moons can be found at: http://www.astronomia.de/index.htm? http://www.astronomia.de/sonnensystem.htm or http://en.wikipedia.org/wiki/Planet#Solar_System (see subpages) Keep in mind that some values might have to be scaled down to fit into the visual scale of this example.
2. The class *OrbitVisualization* is provided by the application, but is not used so far. Create an instance of *OrbitVisualization* in the *SolarObject* class as a class member with the correct constructor parameters. Choose the parent node carefully.
3. Extend the orbit behavior of a solar object in such a way that the inclination of the orbit is considered as well. For this purpose integrate the *orbit_inclination_node* into the node structure of *SolarObject*. Keep in mind that all matrix operations along the path in the scenegraph are accumulated to define the absolute geometric properties of a geometry node.
4. Extend the class *SolarObject* in order to provide functionality for rotation around the planetary axis (see Figure 2). Therefore include a new node *rotation_inclination_node* into the node structure of the class. Implement the body of the function *update_rotation* by accumulating the respective rotation input of the solar object around its own axis. Keep in mind that the order of matrix operations affects the resulting transformation
5. Sketch the final scenegraph up to the third solar object level (sun = first level, planets = second level, moons = third level). Do it on the fly. To this end you can use the function *print_graph()* in the interactive Guacamole console in the shell.