Graphics programming – three.js: reflections and refractions

You will in this exercise get started with three.js and implement reflection and refraction effects.

Getting started

Create a project where you implement a graphics program with the features below. Use relevant examples from threejs.org.

- Object loading from OBJ-MTL files
- Camera to navigate through the scene from user input. If you want to start with something easy, then experiment with the OrbitControls class
- Lights (with shadows if you like, use cast shadow attribute)
- Experiment with transformations on objects using object properties

Cube maps

Set up a cube map and render it. You can yourself create a cube map texture if you like, but it is easier to use a standard one that you can find from one of the examples.

A cube map can be added to the scene with the background property of the scene object. See the materials/cubemap example for how to do this.

Reflections and refractions

Add material properties to your object or objects so that they reflect and refract “light” from the cube map (i.e. the background environment). See the materials/cubemap example for this as well.

Ensure that you understand the refractionRatio property. Check out the refractive indices for various types of materials here:

https://en.wikipedia.org/wiki/List_of_refractive_indices

For the Fresnel effect (different refractive indices per color channel), three.js´s Fresnel shader can be used. Have a look at the source code of this shader to understand how this is done, and also how three.js shaders are written in general. Include the Fresnel effect in your program (look at example materials/shaders/Fresnel).

Bonus: Scene with objects

Create a more complicated/realistic scene with ground and moving objects. Light should cast shadows on the ground and objects should have different material properties, including reflections and refractions.