Assignment on Advanced Computer Graphics - Sheet 2

Due Date 23.5.2019 (during class)

Exercise 1 (Acceleration Data Structures, 10+1 Credits)

In this assignment you should implement one of the following data structures to accelerate the raytracer provided in RayTracingTemplateADS:

- Uniform grid (Prototype in UniformGrid.h/cpp)
- Kd-Tree (Prototype in KDTree.h/cpp)
- Octree (Prototype in Octree.h/cpp)
- AABB-Tree (Prototype in AABBTree.h/cpp)

All acceleration data structures (ADS) are derived from the common base class Accelerator. You have to implement the constructor, the destructor, and the intersect() function, respectively.

Inside the constructor, you have to construct the ADS. To do that, the scene will be passed to it in form of a SurfaceList object. The SurfaceList object contains a list with all geometric objects in the scene. An example in the constructor class shows how you can traverse the objects. Each geometric object has a function getAABB() that returns an axis aligned bounding box of the object. Probably this feature will be helpful. But be careful with objects of infinite size like the plane.

The intersect() function of the acceleration data structures is called for the whole scene instead of the currently used intersect() from the SurfaceList class. As expected, it should compute the first intersection between the input ray and the closest object. In order to get the intersection with the actual geometric object in your ADS you can use the existing intersect() for the geometric objects.

Finally, the destructor should free potentially allocated memory.

If you want to activate your ADS, you can simply choose the respective ADS in the dialogue of the raytracer’s GUI.

You can load 3D triangle mesh files in obj format by using meshfile in the XML scene description. The scenes teapot.xml and chicken.xml show some examples.

Bonus: In the tutorial class, we will honor the fastest ADS. You are allowed to define a scene that favours your own and stresses the other acceleration data structures. The solution with the lowest overall render time for all scenes wins.

For reference: a regular grid renders teapot.xml (at 1000² Pixels with 3 recursions) in 8271 ms.