Prof. G. Zachmann
P. Lange

University of Bremen
School of Computer Science
CGVR Group
May 28, 2014

Summer Semester 2014

Assignment on Advanced Computer Graphics - Sheet 4

Due Date 19. 06. 2014

Exercise 1 (Acceleration Data Structures, *9 Credits*)

In this assignment you should implement **one** of the following data structures to accelerate your raytracer:

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

Exercise 2 (Performance Comparison, *1 Credits*)

In the tutorial class, we will honor the fastest ADS with one additional point. We will compare all implemented ADS with respect to the `steinbach.xml`.