Efficient Ray Tracing without Auxiliary Acceleration Data Structure

Alexander Keller*
NVIDIA

Carsten Wächter†
NVIDIA

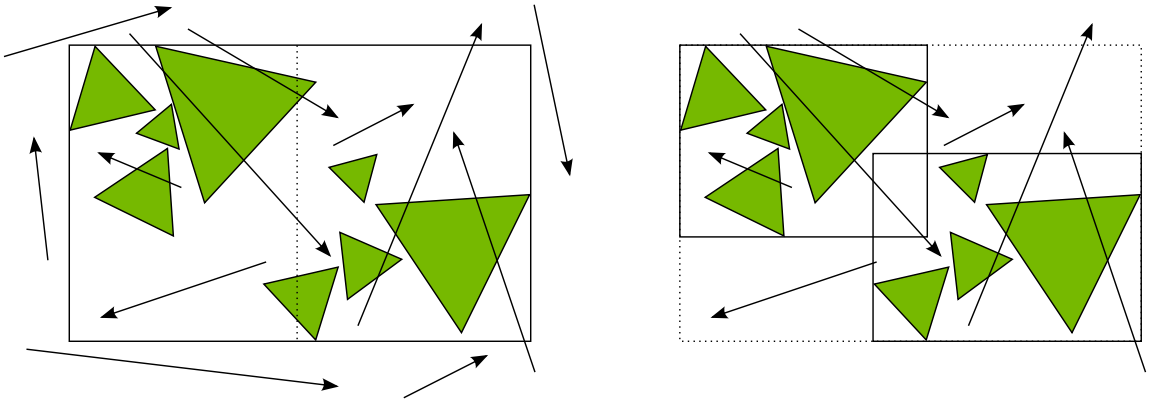

Figure 1: *Instead of tracing single rays, a set of rays is intersected with a set of objects. The procedure first discards all the rays not intersecting the bounding box of the objects. If a sufficient number of rays and objects remains, the set of objects will be partitioned and the procedure will be applied recursively to the resulting sets of objects.*

1 Light Transport Simulation by Ray Tracing

Light transport simulation consists of summing up the contribution of all transport paths that connect cameras and light sources. The vertices of such paths are identified by tracing straight rays, which opposite to rasterization, allows for simulating physical reflection and refraction, as well as arbitrary shadows and global illumination, without approximation artifacts.

2 Implicit Bounding Volume Hierarchy

Simultaneously considering all rays and all objects results in a divide-and-conquer algorithm. As illustrated in Figure 1, this Algorithm 1 [Keller and Wächter 2008] first discards the rays not intersecting the bounding box of the objects. Upon a suitable termination criterion, for example if the numbers of active rays and objects are sufficiently small, all active rays are intersected with all objects, while the closest intersection is recorded with each ray. Otherwise the objects are partitioned and the procedure is recursively called for the resulting sets.

Computing the partition according to a heuristic, for example by separating the objects along the middle of the longest side of their bounding box, can be replaced by using a given partition, as for example the one implied by a scene graph hierarchy. In addition, the presence of bounding volumes of known extent enables the generation of objects on demand.

In fact the recursive procedure traverses a bounding volume hierarchy, which is never explicitly stored. This is useful, whenever explicitly computing and storing an auxiliary acceleration data structure cannot be amortized, for example, in order to augment rasterized images of dynamic scenes by reflections and refractions or multiple shadows.

The efficiency of determining the active rays can be increased by exploiting inherent structure, as for example rays originating from a single point like the eye, the center of an environment map, a point light, or ambient occlusion rays. This includes rasterization and the Reyes architecture as special cases.

Local ray statistics can be used to determine partitions, the order of recursion, and allow for selecting an appropriate level of detail.

Spatial partitioning can be realized by traversing the voxels of a space partition hierarchy and determining the rays and objects intersecting the current voxel in analogy to Algorithm 1.

As illustrated in Figure 1, processing is restricted to regions where both rays and objects intersect. Furthermore, the algorithm takes advantage of caches, because the working set becomes more localized with recursion depth, and intersections are enumerated by spatial proximity.

Algorithm 1: Hierarchical intersection of rays and objects.

```
Intersect(Rays, Objects)
if Rays ≠ ∅ and Objects ≠ ∅ then
    ActiveRays ← {r ∈ Rays ∧ r ∩ BBox(Objects) ≠ ∅} ;
    if ActiveRays ≠ ∅ then
        if Terminate(ActiveRays, Objects) then
            DirectlyIntersect(ActiveRays, Objects);
        else
            (Objects1, Objects2) ← Partition(Objects);
            Intersect(ActiveRays, Objects1);
            Intersect(ActiveRays, Objects2);
        end
    end
end
```

References

KELLER, A., AND WÄCHTER, C. 2008. Efficient ray tracing without acceleration data structure. Internal report, NVIDIA.

*e-mail: keller.alexander@gmail.com
†e-mail: toxie@ainc.de