Bachelor or Master Thesis: Traversal Shaders on Current GPUs

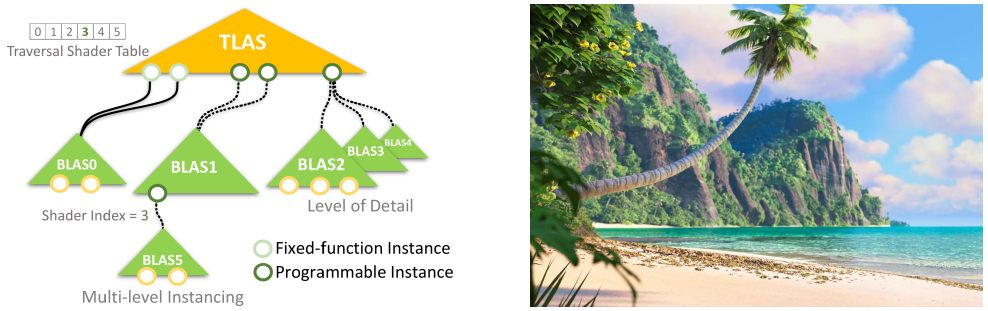

Figure 1: Left: Traversal shaders [2] enable dynamic level of detail and multi-level instancing. Right: Moana island [3] is a 265 GB production asset donated by Disney.

Acceleration structures are indispensable for efficient ray tracing. The simplest variant is a single tree with leaf nodes containing triangles, which is constructed before rendering starts. However, large scenes call for more complicated designs. The same geometry is instanced many times throughout the scene. Often instances are assemblies of instances themselves. This multi-level instancing gives rise to a tree of acceleration structures. Besides, different levels of detail (LoD) may exist for each triangle mesh, which need to be selected based on ray attributes.

Traversal shaders [2] are a programming model satisfying these needs. Leafs can be traversal leafs, which define a bounding box and a programmable traversal shader. This shader has access to ray attributes and uses them to pick an acceleration structure where traversal should continue. Traversal shaders can implement multi-level instancing, LoD selection and more. However, current APIs for ray tracing on GPUs do not support traversal shaders. The hardware enforces the use of two levels of acceleration structures (top and bottom).

The goal of this thesis, is to overcome these hardware limitations to provide a prototypical implementation of traversal shaders on GPUs. To this end, Vulkan ray queries [1] should be used. Some bounding volumes in leafs are marked as traversal leafs. The ray query keeps track of all traversal leafs that are intersected before the first opaque triangle. Afterwards, a common shader loops over this list, implementing functionality of the traversal shader. The result is a list of acceleration structures to traverse next, which is done by another batch of ray queries.

It will not be viable to support traversal shaders in full generality without sacrificing too much performance but for some special cases, efficient implementations should be possible. Correspondingly, tasks include:

- Implement a basic ray tracer on GPU with Vulkan and ray queries [1],
- Flag bounding volumes as traversal nodes and enumerate them with ray queries,
- Implement multi-level instancing with more than two levels [2],
- Implement dynamic selection of levels of detail [2],
- Compare to an optimal reduction of multi-level instancing to two levels or to construction of new top-level acceleration structures instancing suitable LoDs each frame,
- Evaluate with suitable test data, e.g. parts of Moana Island [3].

Contact: Christoph Peters, christoph.peters@kit.edu

References

[1] Daniel Koch, Ashwin Lele, Tobias Hector, Joshua Barczak, and Tyler Nowicki. EXT_ray_query, 2021. https://github.com/KhronosGroup/GLSL.

[2] Won-Jong Lee, Gabor Liktor, and Karthik Vaidyanathan. Flexible ray traversal with an extended programming model. In *SIGGRAPH Asia 2019 Technical Briefs*. ACM, 2019.

[3] Rasmus Tamstorf and Heather Pritchett. The challenges of releasing the Moana island scene. In *Eurographics Symposium on Rendering - Industry Track*, 2019.