ADVANCED GRAPHICS – 2017/2018

February 1st – 17.00 – 19.00 – EDUC-MEGARON

PROBLEM 1. Consider the following scene:

An infinite floor plane with a diffuse BRDF fr(..)=1/pi, illuminated by two spherical light sources, and surrounded by a skydome.

The skydome has a uniform color, which is (2,2,2).

On the floor plane we found a point p using a primary ray from the camera.

a) Show that we do not need to know the radius of the skydome when we estimate the light transport from a point on the skydome via point p to the camera.

The problem here is that a sphere of radius 2 with surface color (2,2,2) emits more light than a sphere of radius 1. The question is thus: why is the radius still irrelevant? Many of you stated that the skydome is sampled when a ray leaves the scene, and that radius is not used in this calculation, but that is simply repeating what is stated in the question. One way to show it: the solid angle calculation contains area A, and a division by dist^2. For a growing sphere, this ratio remains constant. Another way: all energy must arrive on a disc (the floor) with radius r and surface pi r^2. Increasing r increases generated energy (sphere area is 4 pi r^2) and floor radius with the same factor.

b) A path tracer is used to render the scene. Next Event Estimation (NEE) is used to sample the illumination from the two spherical light sources. The ray resulting from a random bounce at point p ends up on the skydome. Should this ray return (2,2,2) or (0,0,0)? Why?

(2,2,2). It doesn't matter if you consider the skydome a light or not; it was not sampled by NEE, and therefore we don't have to prevent sampling this part of the integral twice by returning (0,0,0). In fact, if we force NEE to always ignore one of the lights (pdf=0), an implicit connection to that light should return the light color. Likewise, if NEE included the skydome (reducing it to a random bounce!), we would indeed return (0,0,0), but that's not the case here. All about overlap.

c) We remove the two spherical lights, and replace the uniform color of the skydome by a (non-uniform) HDR texture, which is now the only source of illumination for the scene. Describe a method to importance sample the illumination from this skydome. Note: this was not discussed during the course, so I am asking you to be creative.

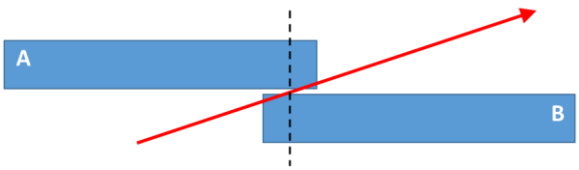
Turns out I did in fact discuss a method for this: the tessellated sphere where triangles become lights with a color based on the texture. That was thus accepted as a correct answer. Another method is to build a CDF over the pixels of the texture (running sum of intensity values). Divide by the sum of all pixel values, and you have a pdf. Sampling it is non-trivial; a binary search is reasonably efficient. Alternatively, you can separate this over x and y: pick an important column, then an important row. One student proposed the RIS approach, which is awesome. :)

PROBLEM 2. When traversing a BVH, doing so in front-to-back order is faster than a random order or back-to-front. We discussed three methods to obtain such an ordering in a (binary) BVH, which rely on different information:

- the first one obtains the distance to each child node on the fly;
- the second one uses information stored with the node about the used split axis;
- the third one guesses the split axis based on the centroids of the child nodes.

Note that the description of the three methods is (deliberately) vague. You were expected to know how they work in slightly more detail to answer the questions.

With that in mind, we have the following interesting situation:



Nodes **A** and **B** are the result of a split along a vertical split plane (the black dotted line).

a) In which order will the nodes be traversed, when applying each of the three methods? Describe for each method briefly how you obtain the answer.

Method 1: B-A. Not A-B: A may be closer to the origin, but not when travelling along the ray.
Method 2: A-B. The method takes the sign of the ray direction over x (because the split is vertical) and concludes that A is to the left and thus 'closer'.
Method 3: A-B. The centroids of A and B are furthest apart over x, therefore the algorithm assumes that the split was vertical. Proceed as with method 2 from here.

b) Describe a situation (if one exists) where traversing node A before node B (with the red ray) is advantageous.

I thought the situation didn't exist: even if B doesn't obstruct the ray while A does, B must still be evaluated, because an intersection in A is further than the enrty point in B. Even if this doesn't yield an intersection in B, we still did the work, therefore this order has no advantage.
What I overlooked is shadow rays. An occluder in A will lead to an early out, which skips B. If B is more work than A, this is advantageous. Both assumptions (shadow or regular ray) were accepted.
Sorry for the trick question, but the hint ('if one exists') makes it acceptable I think.

c) Describe an alternative method to determine which node to visit first. Your method must not rely on calculating the distance to both child nodes. It must visit node A and B in the optimal order in the illustrated case. And finally, it should work well in the general case.

Several creative methods were found and accepted. Personally I hoped you would look for a split plane by looking for the axis with least overlap. Over y, A and B have negative overlap. This would still work reasonably well in the general case.

PROBLEM 3.

The image on the right shows a tree generated by a program called *SpeedTree*. Explain why raytracing or path tracing such an object is relatively expensive.

The problem with trees is that we go deep in the BVH (or kD-tree) several times. A ray may just miss some geometry, crawl all the way up in the BVH, only to go deep again for some other geometry. Sadly no one realized this.

PROBLEM 4.

In the paper "Understanding the Efficiency of Ray Traversal on GPUs", the authors describe several ray traversal kernels in a high-level way. Two of them are called "*while-while*" and "*if-if*".

a) Describe these two traversal loops with some pseudo code. Make sure the difference between the two is clear.

Most of you scored easy points here, check the paper if you still don't know.

b) Explain which approach is faster, and why.

I was surprised to see that most of you are very well aware of lockstep in warps, and the consequences for the while-while loop. Full points for most of you.

PROBLEM 5.

Explain why traversal of an MBVH/QBVH (i.e., a BVH in which each interior node can have more than two child nodes) is generally faster than traversal of a regular BVH (with exactly two child nodes per interior node).

Four reasons: 1) shallow trees get us to the leafs quicker; 2) many children can be intersected with SIMD/AVX/SIMT on the GPU; 3) reading a larger node is not more expensive for the caches; 4) increased arithmetic density for a traversal step is good for GPUs. Full points for just the traversal step reduction.

PROBLEM 6.

Whitted-style ray tracing is a point-sampling algorithm. Is this also true for distributed ray tracing (Cook style)? Explain why / why not.

Technically no. Monte-Carlo methods sample integrals, so any tiny feature has a chance of being sampled. The samples are still point samples, so it depended on your story whether you got points or not.

And, since I forgot: May the Light be with you!