RI3004A   3D Graphics Rendering

Discussion 6 (Answers)

For **Lecture 9: Ray Tracing**

Please attempt the following questions before you go to your discussion class. Some of the questions may be quite open-ended and some may be even ambiguous. In those cases, you are encouraged to make your own (reasonable) assumptions.

(1) Suppose there is an opaque sphere in an enclosed environment. All the surfaces of the environment are opaque and have materials that have only diffuse component. However, the sphere's material has both diffuse and specular components. There are two point light sources in the scene. Assuming we want to render a 100x100 pixels image of the scene using Whitted Ray Tracing, with two levels of recursion, what would be the total number of rays that have to be shot? Assume the camera is within the environment but outside the sphere, and the sphere occupies 3000 pixels in the rendered image. Show your workings clearly.

If all objects are opaque, then there is no need to spawn refraction rays. If the scene is enclosed, then all primary rays and reflection rays will hit some object.

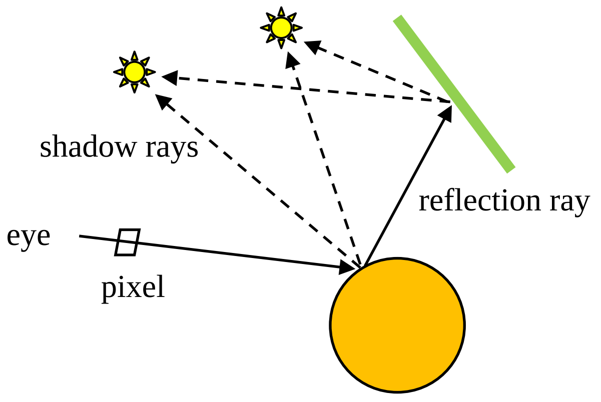
Rays shot per pixel on sphere = 1 primary ray + 1 reflection ray + 4 shadow rays = 6

Total number of rays shot for pixels on sphere = 3000 * 6 = 18,000

Rays shot per pixel not on sphere = 1 primary ray + 2 shadow rays = 3

Total number of rays shot for pixels not on sphere = (100*100 - 3000) * 3 = 21,000

Total rays shot = 18,000 + 21,000 = 39,000



(2) Given a plane, whose equation is $2x + 4y + 4z - 6 = 0$, and a ray, whose origin is $[2, 0, -5]^T$ and its direction is $[1, 2, 3]^T$, **(i)** calculate the location where the ray intersects the plane. **(ii)** Compute the normalized surface normal vector at the intersection point.

$[2, 4, 4]^T \bullet ([2, 0 , -5]^T + t\,[1, 2, 3]^T) - 6 = 0$
$\Rightarrow 2\,(2 + t) + 4\,(2t) + 4\,(-5 + 3t) - 6 = 0$
$\Rightarrow 4 + 2t + 8t - 20 + 12t - 6 = 0$
$\Rightarrow 22t - 22 = 0$
$\Rightarrow t = 1$

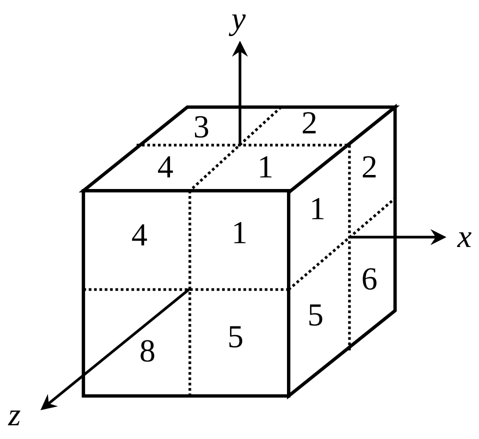
(i) Intersection point is $[2, 0 , -5]^T + (1)\,[1, 2, 3]^T = [3, 2, -2]^T$.

(ii) Normal vector is $[2, 4, 4]^T / \,|\, [2, 4, 4]^T \,| = [2, 4, 4]^T / 6 = [1/3, 2/3, 2/3]^T$.

(3) When using spatial subdivision for ray tracing acceleration, why is it better to use an octree than a 3D uniform grid?

Octree's memory usage and cell resolution are adaptive to the density of the distribution of the objects in the scene. In contrast, usually many uniform grid cells are empty (bad memory efficiency), and many cells contain too many objects (bad ray intersection efficiency).

(4) Given that an octree is used to subdivide the 3D scene. Suppose the octree is centered at the origin and a primary ray originates from the location (30, −40, −55) and is directed towards the octree. The octree has 8 subcells. In what order should we check for intersection between the subcells and the ray to determine the first hit? The subcells have been labeled 1 to 8 as shown in the figure (the subcell not seen is labeled 7).



The subcells should be checked in the following order:

6, {2, 5, 7}, {1, 3, 8}, 4.

where items inside { } can be in any order.

(5) If a scene has a small number of very small and extremely complex objects (i.e. expensive to compute ray-object intersection) evenly distributed in space, which of the following two octree subdivision schemes is more efficient for ray tracing? Why?

Scheme 1: Subdivide a cell if it is occupied by more than one object.

Scheme 2: Subdivide a cell if it is occupied by any object unless the maximum allowable subdivision depth is reached.

Scheme 2 is better.
Scheme 1 will create many large cells with only a small object inside, and if a ray intersects one of these cells (very likely because the cells are large), the small object inside will need to be checked for intersection with the ray, even though the ray most likely intersects nothing.

(6) Describe how you would use Distribution Ray Tracing to produce motion blur for a moving object in the scene.

1. Parameterize the position of the object w.r.t. time *t*.
2. During rendering, for each subpixel, randomly choose a *t* and generate the corresponding 3D scene.
3. Render the 3D scene normally to get the result for the subpixel.
4. Average the subpixels to get the result of the pixel.

——— End of Document ———