Qt questions?

A5G6: photon mapper

Idea: (2-pass ray tracer)

1. shoot photons (A5G4)
   → emit photons from light source(s)

2. trace rays
   → each ray accumulates light "power" from knn photons

Step 1:

- shoot photons
- result is list of 3D "points" each with power & direction, scattered on surfaces in 3D world

cbox.txt : AS6 6

yourmodel.txt : bonus
↳ make your own scene

email me your pic

Step 1a:

loop thru list of n photons, scale power of each by $\frac{1}{n}$

(n will be smaller than what you start with, e.g. 50,000)

Step 1b:

loop thru n photons

find min, max

(x,y,z) (x,y,z)

pos pos

bounding volume for kd-tree

Step 1c:

- insert photons into kd-tree:

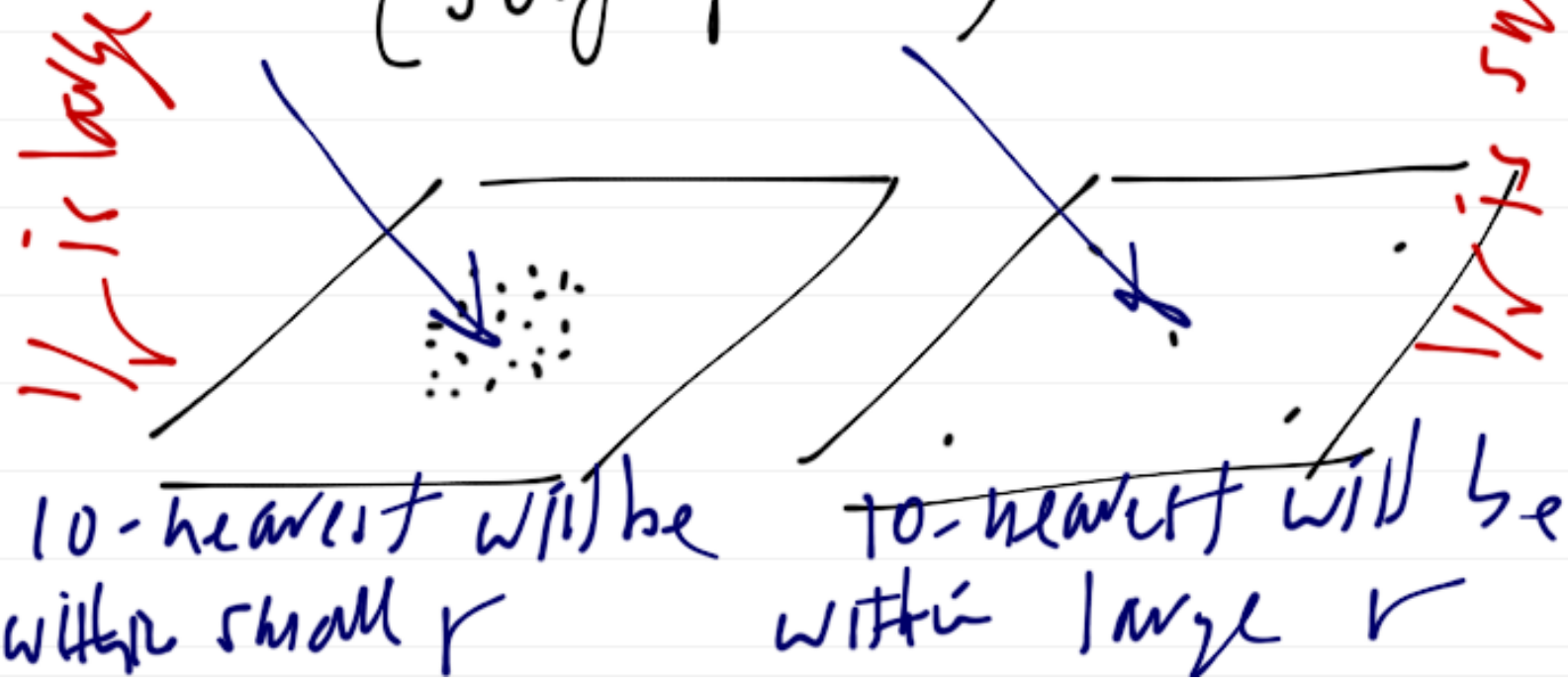
```
kdtree_t<photon_t,
         photon_t *,
         photon_c> kdtree;
```

Comparator functor

```
kdtree.insert(photons,
              min, max);
```

Step 2 : ray trace

- each ray will query atlas for k nearest photons (k = 10)
- estimate power density (flux) at ray intersection (surf. point)



- code (in main.cpp)

use

```
ray->trace(model, kdtree, #pragma omp statement (shared or private), color, 0)
```

instead of

```
ray->trace(model, color, 0)
```

don't forget to include kdtree into your

overloaded trace() fun. just copy old trace() fun & add argument

(ray.cpp)

```
ray_t:: trace (..., kdtree, ..)
{
  // same as before — don't
  forget to make this trace
  routine call trace(..., kdtree, ..)
  instead of old trace(...)
  // add at end "contribution"
     from photon power flux
```

Contributions (to color at surface point):

ambient +

diffuse +

reflection +

transmission + } Ray tracing

specular highlights +

power flux } photon gathering

⋮

```
dis = model.find_closest(...)
if (dis > 0) {
    // get material properties
    // ambient constants.
    color += 1.0/dis * ambient
    color.clamp(...)
```

```
// diffuse contrib
if (!diffuse.iszero()) {
    color += (Id * diffuse)
    color.clamp(...)
}
// reflection contrib
if (!specular.iszero()) {
    // shoot reflection ray
    color = (diffuse * color) +
(specular * ref_color)
    color.clamp(...)
}
```

```
// transmission coefficient
if (alpha > 0)
    // shoot refraction ray
    color = (1 - alpha) * color +
        alpha * ((1 - R) * refr
            + R * rcol))
    color.clamp(...)
}
```

// spot light

⋮

// photon power
(see Henrik Wann
Jensen)

a) knn query ($k = 10$)
get r
=
(radius)

5) estimate (& smooth) flux

(try 6 wall filter)

for each of kNN photons

if ($\vec{N}$ · photon.dir < 0) ← surface normal

dp = distance from hit point to photon

(use photon_t::distance())

$$W_{pc} = \frac{1 - \frac{dp}{kr}}{1 - \frac{2}{3k}}$$

See RSGG Web page

cone filter

flux += photon.pwr * wpc

// end of for k photons loop

c) color += flux * $\left(\frac{1}{\pi r^2}\right)$

color.clamp(...)