A guide to the photon-mapping code

The changes from the book's code to the example code here very closely parallels that of the path tracer, so I won't go through every detail. But there's one point that isen't as clear in the book as it should be.

When we included final gathering as an option, the book said the code was this:

```
Radiance3 App::estimateDiffuselyReflectedIndirectLight(const Ray& r, const shared_ptr<Surfel>& surfel, int depth, bool
useGather){
    Radiance3 L(0.0f);

    if (!useGather){
        ...use the previous photon-mapping code
    } else { //gathering
        for (int i = 0; i < m_gatherRaysPerSample; i++) {
            // draw a sample, ray, from the BRDF TOWARDS a light source
[...]
            L += pif() * surfel->...
        }
        return L/ m_gatherRaysPerSample;
    }
}
```

..., which might lead you to think that the previously computed radiance value was divided by the number of gather-rays per sample at the end, in the no-final-gather case. That would make no sense. The actual code structure is this:

```
Radiance3 App::estimateDiffuselyReflectedIndirectLight(...
    if (!useGather){
        ...use the previous photon-mapping code
        return L;
    } else { //gathering
        for ...
        return L/ m_gatherRaysPerSample;
    }
}
```

There's another point that's not entirely clear from the discussion of final gathering: suppose you've got a bunch of photons on some surface, all concentrated in a single area, representing a caustic. If your eye-ray happens to hit in the middle of these photons, and the number of final gather rays is not very very large, you're unlikely to sample these caustic rays --- indeed, it's almost impossible if the caustics arise from point lights, but mostly works out if they're from area lights --- so including final-gather tends to make caustics disappear. At this point, it's worth following Jensen's advice and building a *second* photon map, the *caustic map*. You can use exactly the same code, except that there's a slightly new rule: photons bounce around until either they're absorbed, or they hit a diffuse surface. (Perhaps it'd be better to say "until they arrive at a surface that has some diffuse-scattering component, and are either absorbed or, when you try to scatter them, end up in the diffuse-scattering path through your code rather than the specular-scattering part.) When either of these things happens, you place the resulting phon in the "caustics map." Note: you start with the same light-power in constructing the map, and may use either more or fewer photons than are used in the ordinary photon map; the two kinds of photons are entirely independent. Of course, if you use more photons, the power of each one will be smaller, and vice versa.

When you build this caustic map, you must, however, no longer store photons on specular surfaces (or more precisely, that scatter by specular scattering at some point); that prevents you from double-counting specular photons.

Now when it comes to estimating the diffusely scattered indirect light, we consider that light as broken into two classes of paths:

(i) LS+DE (caustic photons)
(ii) L(S|D)*D DE (non-caustic photons)

Final-gather, using the general photon ma, will estimate the latter; note that because we only store photons that have at least one diffuse bounce, and the final diffuse scattering at the eyeRay/world intersection is also a "D", we only account for paths that end in DDE, hence none of the type-i paths. We then must also estimate the radiance (without a final-gather, by looking at photons near the eyeRay/world intersection) in the caustic map to account for the remaining power from the light that's indirectly scattered back to the eye. We leave this additional implementation to you.