The RADIANCE Photon Map Manual
Version 3.6

Roland Schregle (schregle@ise.fhg.de)
Fraunhofer Institute for Solar Energy Systems

August 15, 2002

Contents

1 Introduction

The RADIANCE rendering system [Ward94] is a very versatile and powerful tool for lighting simulation, and one of the very few physically based renderers with available source code. However, developments in light redirecting materials have presented new challenges in their simulation; their specular nature makes them difficult to simulate with RADIANCE, often merely resulting in a noisy mess. Some are downright impossible to simulate correctly with RADIANCE or bog the system down with high computational overhead.

Particularly the phenomenon of caustics poses a serious problem for RADIANCE. Caustics are the bright, iridescent highlights on diffuse surfaces caused by specular reflection or refraction. A typical example is the cardioid pattern seen inside a cup or metal ring (figure 1). To quote chapter 13 of *Rendering with RADIANCE* [Ward98],

> "Other cases involving curved, specular reflectors pose similar difficulties for `mkillum`, and the only long-term solution seems to be the creation of a forward raytracing module for computing these kinds of illums."

The problem is actually far more fundamental due to RADIANCE's backward raytracing methodology, and not restricted to `mkillum` alone.

Figure 1: Photograph of metal ring caustic.

RADIANCE's difficulties with sophisticated light redirecting materials necessitated the development of the forward raytracing module mentioned above as a supplementary algorithm. The photon map [Jens96, Jens97, Jens00] was chosen for a number of reasons:

- Apart from handling indirect diffuse illumination, it handles caustics efficiently and has become the algorithm of choice for this purpose.
- It is straightforward to integrate into existing ray tracing environments.
- It is decoupled from the scene geometry and thus can handle arbitrarily complex objects, including procedural and even fractal ones. This also enables using the photon map to simulate indirect illumination in volumes, such as participating media [Jens98].
- It represents a view independent solution to the indirect illumination and thus opens up the possiblity of interactive walkthroughs as with the Radiosity method.

The photon map is based on a (light) particle transport simulation, which lends its name. Each photon interacts with the objects it strikes and is either reflected, transmitted, or absorbed, depending on the characteristics of the material. A Monte Carlo sampling method is used to generate the reflected or transmitted directions of a photon if it survives. Eventually a particle is terminated either by absorbtion or leakage (if it leaves the scene), and a new photon is emitted.

2 Overview of the Implementation

The photon map algorithm comprises two steps:

Photon distribution: as a preprocess, photons are emitted from all light sources and probabilistically reradiated or absorbed upon striking a surface or passing through a volume, depending on its properties. The result is a view independent representation of the indirect illumination (figure 2).

Photon gathering: during the actual rendering, the indirect illumination for a point on a surface or within a volume is determined by finding a number of nearest photons to the point. The irradiance is then proportional to the photon density (figure 3).

The distribution step is performed by the `mkpmap` utility. It is responsible for building the photon map and saving it to a file for subsequent reuse by `rpict`[1] for the actual rendering and photon gathering.

Three photon types are available:

Global photons account for all indirect (ambient) illumination incident on surfaces with a diffuse component.

[1] Unless noted otherwise, `rpict` also implies its siblings `rtrace` and `rview`.

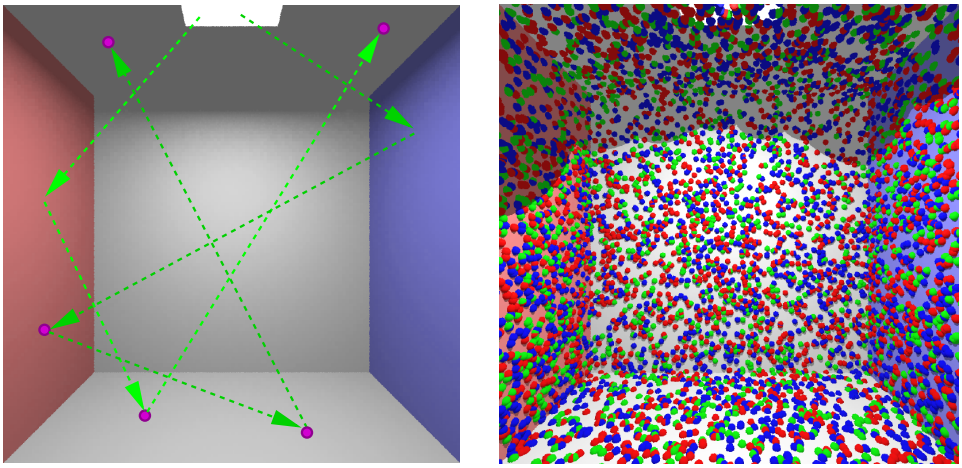

Figure 2: Global photon distribution in the diffuse Cornell box. Left: individual photon paths. Dots indicate stored photons. Right: photon distribution after completion of forward pass.

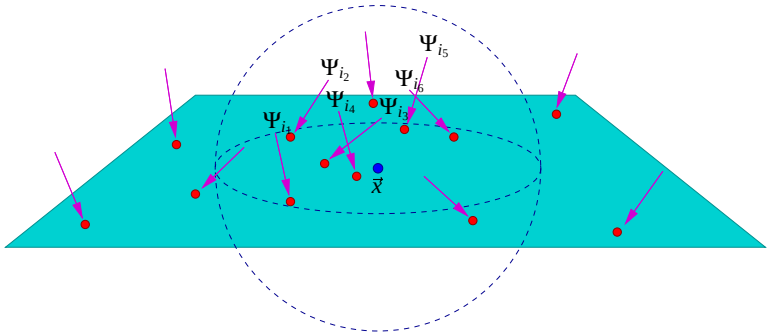

Figure 3: Gathering photons via nearest neighbour search.

Caustic photons account for all indirect specular to diffuse transfers, i.e. caustics. These transport paths are also accounted for by global photons, but not visualised directly like caustics.

Volume photons account for indirect illumination within participating media using the *mist* primitive.

Each photon type is stored in a separate photon map and visualised differently.

The stock RADIANCE functionality (referred to here as “RADIANCE Classic”) is preserved in `rpict` if no command line arguments pertaining to the photon map are given. When rendering with the photon map, the ambient calculation is augmentented by a photon map density estimate, but the direct and indirect specular components are still delivered by RADIANCE Classic.

2.1 Supported Primitives

Material types in RADIANCE require additional code to probabilistically generate outgoing directions for incident photons using monte carlo methods. This has been done for those materials with fixed BRDFs / BTDFs:

Gaussian: *plastic, metal, trans, plastic2, metal2, trans2*

Refracting/Reflecting: *glass, dielectric, interface, mirror*

Mixtures: *mixfunc, mixdata*

Patterns: *colorfunc, brightfunc, colordata, brightdata, colorpict*

Textures: *texfunc, texdata*

Volumes: *mist, antimatter*

Materials for user defined BRDFs / BTDFs such as *plasfunc, metfunc, transfunc, brtdfunc, plasdata, metdata,* and *transdata* are currently **not** supported due to the high sampling overhead in conjunction with the photon map. A wavelet representation is being developed for these. (But don't hold your breath...)

In order for volume photons to function correctly with materials which modify the coefficient of extinction and scattering albedo (i.e. *dielectric* and *interface*), it is necessary to either set the global mist parameters or to surround the objects in question with a mist boundary.

Light sources require additional code for photon emission. Currently supported primitives are *light, spotlight, glow,* and *source.* To ensure accurate distribution light source geometries are partitioned to a user specified resolution, and photons emitted from each partition. In the case of the *source* primitive there is no geometry to partition; instead, the scene cube faces are partitioned and act as emitters, unless photon *ports* are used (see section 2.2.4). The *glow* primitive acts as a regular light source. Virtual sources are disabled with the photon map, since caustic photons already account for these effects.

2.2 Optimisations

A number of optimisations have been incorporated into the implementation to either improve performance or accuracy. The user must be aware of the fact that most of these optimisations incur trade-offs; increasing performance will inevitably compromise accuracy, and vice versa. Such is the harsh reality of Monte Carlo.

2.2.1 Final Gather

Per Christensen's *final gather* scheme [Chri00] has been integrated into the implementation and is available as an option to accelerate the global photon gathering step. Global photon irradiance is precomputed for a fraction of the photon positions

and stored with the photons. This reduces the overhead of global photon lookups during the gathering phase; only the single closest global photon is retrieved, and its precomputed irradiance can be used directly (figure 4). However, final gather has the disadvantage of introducing bias to the solution for scenes with nonuniform indirect illumination. When accurate simulation is required it's best to forego this option. Final gather may also be combined with the bias compensating operator explained below.

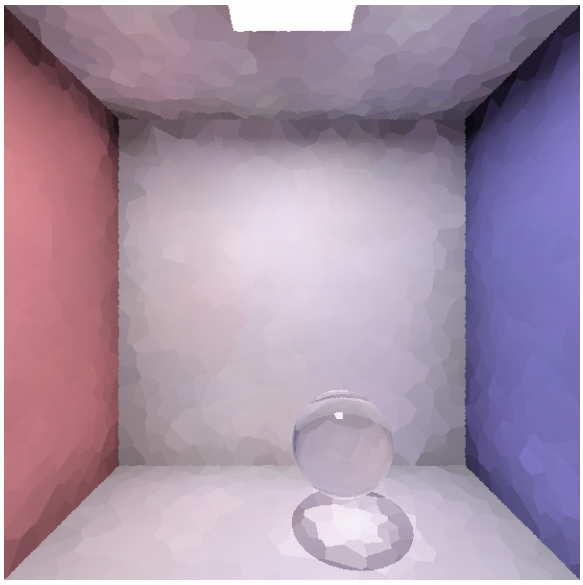

Figure 4: Radiance derived from precomputed global photon irradiance for final gather. Note the caustic from the glass sphere.

2.2.2 Bias Compensation

The number of photons used for an irradiance estimate (also referred to as the estimate *bandwidth*) affords the user control over the quality and accuracy of the rendering by trading off between noise and bias. Setting this too low may result in excessive noise. Setting this too high may result in excessive blurring and local bias in the solution.

To counter the bias/noise tradeoff, a *bias compensating operator* was developed and incorporated into the irradiance estimate as an option. Its purpose is to adapt the bandwidth within a user specified lower and upper bound. The operator employs a binary search for an optimal bandwidth based on the likelihood that deviations from a running average irradiance are due to noise or bias. This results in substantially improved contours in nonuniform indirect illumination (particularly caustics) with a slight increase in noise in these regions (figure 5). The operator also compensates boundary bias (darkening at polygon edges) and chromatic bias.

However, this sophistication comes at a hefty price; performance will suffer severely under this operator.

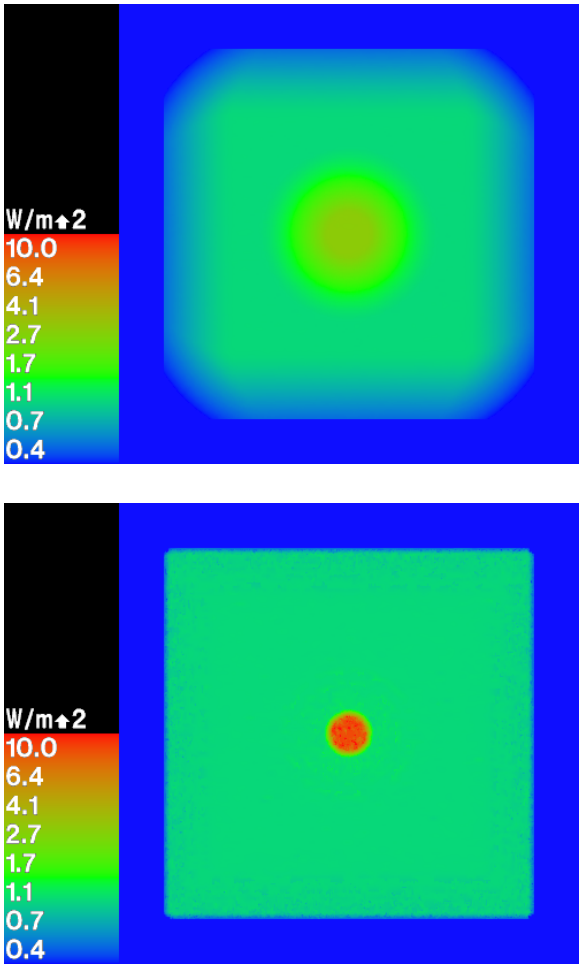

Figure 5: Top: falsecolour rendering of severely biased caustic with a bandwidth of 5000 photons. The caustic should have an irradiance of 10 W/m^2. Bottom: falsecolour rendering of the same caustic with bias compensation using a bandwidth of 50–5000 photons.

2.2.3 Irradiance Thresholds

Performance can be tweaked with the *irradiance threshold* parameter, which can be specified separately for each photon type. This is assumed to be the minimum visible irradiance contributed by the photon map. A maximum search radius for photon gathering is derived from this threshold based on the photon flux and the number of photons to gather. No photons beyond this radius will be searched,

which can speed up the search dramatically. This of course presupposes some knowledge of the lighting levels in the scene on behalf of the user. Unfortunately, setting the threshold too high will result in frequent “short” gathers, finding fewer photons than requested. While this isn’t tragic, it can introduce noise or banding. However, grossly overestimated thresholds can lead to the complete omission of low indirect illumination, which is unacceptable, except possibly for crude previews. With reasonable settings the artifacts are usually quite subtle while still yielding a performance gain. Caustics benefit most from this parameter, since they are only visible where the photons cluster, while the rest of the scene is sparse and therefore contributes negligible illumination.

2.2.4 Photon Ports

Emitting photons from the *source* primitive presents a challenge; there is no local geometry associated with these distant sources. While the default behaviour is to emit photons from the scene cube faces, this can become highly inefficient for sources with large solid angles. These are infact situations for which a forward raytracer is fundamentally inappropriate. RADIANCE Classic’s backward raytracer fares better here, since sending a ray to a large source is trivial. With the forward raytracer, the majority of emitted photons will be lost and won’t contribute to the photon map. To aggravate the situation, distant sources are often used to render interior daylit spaces. With the viewer positioned inside, the few photons that actually strike the geometry must also pass through windows or skylights before they can even contribute visibly to the photon map. Needless to say, this amounts to abysmal performance.

A simple workaround to the problem is the concept of photon *ports*. The ports are basically apertures through which photons can enter the space containing the viewpoint. These can be windows, skylights, but also invisible polygons within the scene (figure 6). There is a price to pay however, since this scheme requires user intervention. The user must define the port geometry and specify them according to their modifiers. If the user places the ports intelligently, the photon loss factor will be reduced dramatically, resulting in a substantial increase in performance.

All objects using the specified port modifiers act as ports during photon distribution. Photons are emitted directly from the ports rather than from the scene cube faces. The ports must be defined in the scene description such that the surface normals point *outside*. The incident flux from distant sources is evaluated at each port and the sources are checked for occlusion. However, no interreflection calculation is made; ports differ from *illums* in this respect. Photons are scattered by the port upon emission as if they had passed through it.

In some cases ports may not coincide with any scene geometry. These situations call for invisible ports. Photons should be emitted from them, but they should not be affected by scattering through the port, and the port should not be visible during rendering. The simple solution is to define photon ports using the *antimatter* modifier, with *void* as its only string argument.

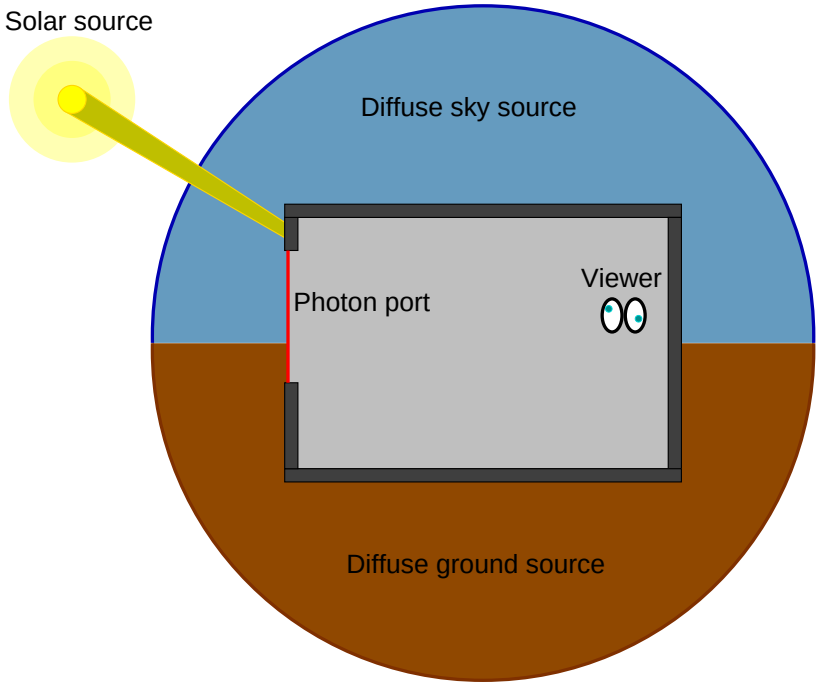

Figure 6: Typical daylighting geometry with window as photon port for distant sources.

Photon ports are the only optimisation in the implementation which do not compromise accuracy for performance, assuming the user knows what he/she/it is doing. It should also be noted that the photon port concept in the current release is a simple hack as an interim to a more sophisticated and flexible distribution method. Future implementations may use importance information for photon paths to yield more intelligent and efficient distribution, possibly even without user intervention. (Or not. Probably not.)

3 Photon Distribution with `mkpmap`

The `mkpmap` utility accepts a RADIANCE octree as input and generates a photon map of each specified type for the scene. The photons are stored in a space subdividing data structure [Bent75] which facilitates nearest neighbour queries required for the gathering step. The data structure is written to a portable binary file for subsequent use by `rpict`.

If a photon map with the specified file name already exists, `mkpmap` aborts without overwriting. The need for this protection stems from the frustration of having photon maps from very long previous distribution passes obliterated by the command line history! There is no “clobber” option; the files must be renamed or removed by hand. This is an annoyance, but better safe than sorry...

3.1 Material Check

`mkpmap` checks the materials in the octree before distributing any photons. This is required to avoid embarrassing infinite loops in the following situations:

- Specifying a global photon map for a scene containing no materials with diffuse component
- Specifying a caustic photon map for a scene containing no materials with diffuse or specular component
- Specifying a volume photon map for a scene lacking the *mist* primitive.

`mkpmap` aborts immediately if one of these conditions is detected.

3.2 Distribution Algorithm

The parametrisation of the distribution step is primarily intended to be user friendly; the user specifies the required number of photons stored in each map. These are treated as "targets" which the photon distribution code tries to attain, since there is no way of knowing the outcome of the distribution a priori. The distribution algorithm makes an educated guess at the number of photons it must emit in order to satisfy the targets. This is based on the results of a distribution prepass in which a small number of photons (a fraction of the minimum specified target) is emitted and distributed. Since the distribution process scales linearly, the remaining number of photons to emit can be predicted and distributed in the main pass. The actual number of photons generated after the distribution is complete will approximate the targets.

3.3 Runaway Photons

Photons may become trapped inside *dielectric* or *interface* materials, often due to infinite total reflection. These "runaway" photons are terminated after a user defined bounce limit is exceeded, and a warning is issued under the assumption that this is an anomaly of the scene geometry.

3.4 Progress Report

If status reports are enabled, `mkpmap` will report the following during distribution:

$$N_e \texttt{ emitted, } N_g/N_c/N_v \texttt{ stored, } P\texttt{\% after } T \texttt{ hours,}$$

where:

- N_e is the total number of emitted photons
- N_g, N_c, and N_v are the number of global, caustic, and volume photons stored, respectively

- P is the percentage done
- T is the time elapsed.

When precomputing global photon irradiance a report will also be issued with the number of precomputed photons N_p:

N_p `precomputed global photons, P% after T hours`,

If precomputation is combined with bias compensation, error statistics will also be reported (see section 5.3).

3.5 Command line arguments for `mkpmap`

The general synopsis of the `mkpmap` utility is:

```
mkpmap [options] octree
```

The options are as follows:

`-apg <file> <nphotons>`
: The output file name and approximate number of photons for the global photon map.

`-app <file> <nphotons> <nestimate>`
: The output file name for the precomputed global photon map, the approximate number of photons (before precomputation), and the number of photons to use for the precomputed irradiance estimate.

`-appb <file> <nphotons> <nestimate_min> <nestimate_max>`
: Same as above, but applying bias compensation within the minimum and maximum number of photons on the precomputed irradiance estimate.

`-appt <thresh>`
: Irradiance threshold for precomputed global photon map in W/m^2.

`-apc <file> <nphotons>`
: The output file name and approximate number of photons for the caustic photon map.

`-apv <file> <nphotons>`
: The output file name and approximate number of photons for the volume photon map. Volume photons can only be used with octrees containing the *mist* primitive.

`-apd <frac>`
: Fraction of the minimum number of photons specified which are emitted in the distribution prepass. Increase this if the number of stored photons differs substantially from your specified targets. This parameter may exceed 1.

`-apf <frac>`
: Fraction of global photons to precompute for final gather, in the range $]0, 1]$. The remaining photons are discarded.

`-apm <max>`
: Specifies the maximum number of bounces for a photon before termination as runaway photon.

`-apo <mod>`
: Specifies photon port modifier. All objects with this modifier will act as ports for photons emitted from the *source* primitive. This option can be used in mutliple instances with cumulative effect.

`-apO <file>`
: Photon port modifiers are read from the specified file.

`-bv`
: Toggles backface visibility. Enabling this switch causes photons to be stored and possibly reradiated if they strike the back of a surface, otherwise they are ignored and immediately absorbed.

`-dp <res>`
: Resolution for sampling user defined light source emission distributions, in samples per steradian. This is required for numerically integrating the flux emitted by the light source and for constructing a probability density function (PDF) for photon emission. The accuracy of photon emission with user specified distributions depends on this parameter. Increase it if caustics exhibit spurious artifacts.

`-ds <frac>`
: Light source partition size ratio; light sources are partitioned to distribute the photon emission over its surface. This parameter specifies the ratio of the size (per dimension) of each partition to the scene cube.

`-e <file>`
: Redirect diagnostics and progress reports to specified file.

`-i <inc>`
: Photon heap size increment; photon heaps are enlarged by this amount when storage overflows. No need to fiddle with this.

`-ma <ralb> <galb> <balb>`
: Global scattering albedo. See `rpict`.

`-me <rext> <gext> <bext>`
: Global extinction coefficient. See `rpict`.

`-mg <gecc>`
Global scattering eccentricity. See `rpict`.

`-sj <frac>`
Specular jitter applied to outgoing photon directions on gaussian materials.

`-st <frac>`
Specular sampling threshold. See `rpict`.

`-t <sec>`
Generate progress report in specified intervals.

The `-apc` and `-apv` options can be freely combined with either `-apg`, `-app`, or `-appb`. At least one of these must be specified... or nothing happens!

4 Querying the Photon Map with `pmapinfo`

The `pmapinfo` tool supplies the following information about a photon map:

- the photon map type as identified by magic number:
  - `Radiance_Global_Photon_Map`,
  - `Radiance_PreComp_Global_Photon_Map`,
  - `Radiance_Caustic_Photon_Map`, or
  - `Radiance_Volume_Photon_Map`
- the `mkpmap` command line used to build the photon map
- the number of photons in the photon map
- the photon flux averaged over the RGB colour channels.

5 Photon Gathering with `rpict`

The modified `rpict` rendering tool takes the photon map generated by `mkpmap` and perform the photon gathering and density estimation. Without photon map functionality enabled, it behaves like RADIANCE Classic.

5.1 Caustic Photon Visualisation

Caustic photons are visualised at points seen directly by the observer, or via specular reflections. Virtual light sources are disabled with the caustic photon map, since these effects are already accounted for. Note that irradiance calculations with `rtrace` using caustic photons are surface bound. Irradiance for points which do not lie on surfaces will be biased.

5.2 Global Photon Visualisation

Global photons are visualised via an intermediate ambient bounce, which reduces noise (figure 7). Ambient values specified with `-ab` are ignored. The global photon irradiance may be optionally precomputed by `mkpmap` to speed things up at the expense of accuracy.

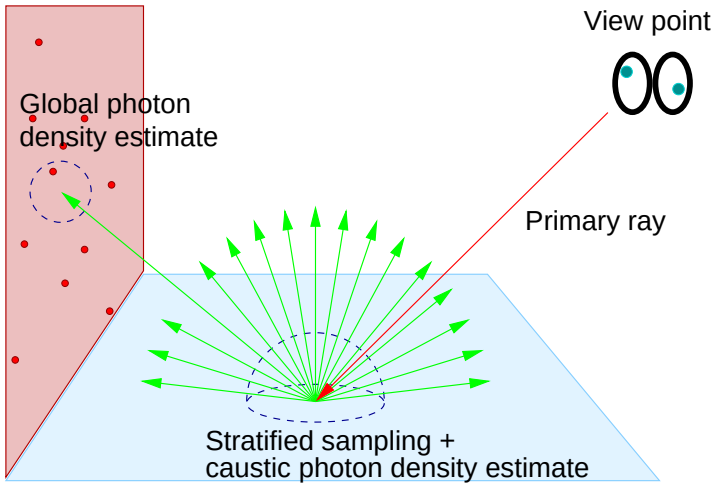

Figure 7: Caustic photons are visualised at the primary ray, while an intermediate stratified sampling step is used between the primary ray and the global photon contribution.

5.3 Progress Report

Bias compensation statistics are provided in the progress report with the standard `-t` option. The statistics reported for each photon type subject to bias compensation are as follows:

$$N_{min}/N_{max}/N_{avg}\ \ ptype\ \texttt{photons/estimate,}\ \ \varepsilon_{min}/\varepsilon_{max}/\varepsilon_{rms}\ \ \texttt{error,}$$

where:

- N_{min}, N_{max}, and N_{avg} are the minimum, maximum, and average number of photons used per irradiance estimate, respectively
- *ptype* is one of `global`, `caustic`, or `volume`.
- ε_{min}, ε_{max}, and ε_{rms} are the minimum, maximum, and RMS (root mean square) estimated relative errors in the irradiance estimates, respectively.

Since bias cannot be determined without knowing the actual illumination there is no reliable way of separating it from noise. As such, the error estimates are crude and only serve as a guide to judge the quality of the operator and its parametrisation.

5.4 Command line arguments for `rpict`

The photon map is disabled by default and must be explicitly enabled using one or more of the `-ap` options. The different photon map types may be renderered independently or in combination. Note that rendering only the global or caustic photon map does not take all light transport paths into consideration, and may omit some indirect illumination. Since modifying the lights or geometry invalidates the photon map, a warning is issued if the octree is newer than the photon map.

`-ab <nbounces>`
: The number of ambient bounces; setting this to a positive value will include global photon contributions via **one** ambient bounce (regardless of the actual number specified). The irradiance cache may be applied to the top level ambient rays, but photon map lookups are not cached. Setting this to a negative value will visualise the global photons directly. With precomputed global photons this results in a pretty Voronoi diagram for those who feel the urge to debug. As with RADIANCE Classic, the default value of 0 will disable the ambient component. Caustic photons, however, are still visualised.

`-apg <file> <nestimate>`
: Loads the global photon map from the file and uses the specified number of photons for a global irradiance estimate.

`-apgb <file> <nestimate_min> <nestimate_max>`
: Same as above, but applying bias compensation within the minimum and maximum number of photons on the global irradiance estimate.

`-apgt <thresh>`
: Irradiance threshold for global photon map in W/m^2.

`-app <file>`
: Loads the precomputed global photon map from the file and uses the precomputed irradiance for final gather.

`-apc <file> <nestimate>`
: Loads the caustic photon map from the file and uses the specified number of photons for a caustic irradiance estimate.

`-apcb <file> <nestimate_min> <nestimate_max>`
: Same as above, but applying bias compensation within the minimum and maximum number of photons on the caustic irradiance estimate.

`-apct <thresh>`
: Irradiance threshold for caustic photon map in W/m^2.

`-apv <file> <nestimate>`
: Loads the volume photon map from the file and uses the specified number of photons for a volume irradiance estimate within the *mist* primitive.

`-apvb <file> <nestimate_min> <nestimate_max>`
: Same as above, but applying bias compensation within the minimum and maximum number of photons on the volume irradiance estimate.

`-apvt <thresh>`
: Irradiance threshold for volume photon map in W/m^2.

6 Bugs

- Although analytical validation has shown the photon map can achieve an accuracy within $\pm 1\%$ for very simple scenes (i.e. under laboratory conditions), the simulated lighting levels can deviate very slightly seriously from Real-Life^{TM} when complex geometry is involved or inappropriate parameters are used. An experimental validation under more realistic conditions is still in the works.
- The spotlight focal point is ignored by the photon map; it coincides with the light source surface. This corresponds to specifying a direction vector of infinitesimal length in RADIANCE Classic.
- `mkpmap`'s distribution algorithm can exceed the specified number of photons with scenes having high reflectance or when excessively high settings of `-apd` are used.
- Caustic photon irradiance will be biased when passing `rtrace` points which do not lie on a surface.
- Pathological scenes containing light sources which see no geometry will leave `mkpmap` in an infinite loop.
- `mkpmap`'s material check isn't foolproof, and may still lead to an infinite loop in some cases. Since only the modifiers are examined, the check will fail if no objects actually use them.

References

[Ward94] Gregory J. Ward: "The RADIANCE Lighting Simulation and Rendering System". Proceedings SIGGRAPH '94, July 1994

[Ward98] Greg Ward Larson and Rob Shakespeare: *Rendering with RADIANCE*. Morgan Kaufmann, San Francisco, 1998

[Jens96] Henrik Wann Jensen: "Global Illumination using Photon Maps". Rendering Techniques '96, Eds. X. Pueyo and P. Schröder, Springer-Verlag, 1996

[Jens97] Henrik Wann Jensen: “Rendering Caustics on Non-Lambertian Surfaces”. Computer Graphics Forum, vol. 16 (1), March 1997

[Jens98] Henrik Wann Jensen and Per H. Christensen: “Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps”. Proceedings SIGGRAPH ’98, July 1998

[Jens00] Henrik Wann Jensen and Niels Jørgen Christensen: “A Practical Guide to Global Illumination using Photon Maps”. SIGGRAPH 2000 Course Notes 8, July 2000

[Bent75] Jon Louis Bentley: “Multidimensional Binary Search Trees Used for Associative Searching”. CACM 18(9), 1975

[Chri00] Per Christensen: “Faster Photon Map Global Illumination”. Journal of Graphics Tools, vol. 4 (3), April 2000