Assisted Environment Map Probe Placement

M. G. Chajdas[1], A. Weis[1] and R. Westermann[1]

[1]Technische Universität München

Abstract

Placing probes for environment maps is a tedious and time-consuming task for current game developers. In particular, locating the significant places for reflection requires the artists to manually scan through a whole game level. Even then, probes may wind up being too similar and have to be cleaned up manually later on. Furthermore, the whole process needs to be repeated when the content changes.

We propose a novel algorithm which assists the artist when placing environment map probes. Based on the original game content, we produce a set of candidate locations for inspection by an artist. By setting a few parameters like sampling density and aggressiveness of probe elimination, the overall number of generated sample probes can be easily adjusted. All of this requires only a short pre-process which can be done using the available in-game renderer. From the generated sample set, the artist can decide which locations are important, add or remove probes manually and adjust the placement to account for supplemental, content-specific parameters. Our initial probe placement, together with additional information computed by our algorithm, allows the artist to place probes more efficiently. Early user testing indicates that the total time can be reduced by up to half a day for a single game level.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction and related work

Environment mapping [BN76, Gre86] is a widely used technique in real-time rendering to simulate diffuse and specular reflections. Lately, games have also started to use light probes – which are essentially low-resolution environment maps – for image-based lighting. Environment maps are easy to generate, can be pre-computed, and render efficiently due to hardware support. However, they are still relatively expensive to create in real-time, especially on consoles with limited vertex throughput. Hence, a lot of games use fixed environment map probes, which are placed by an artist during the scene creation [McT04]. At runtime, the game engine selects one or multiple probes close to the affected object and uses them for reflection or illumination.

As content complexity has increased, placing environment map probes has become a time-consuming process. Typically, one artist is responsible for setting up all probes in a game scene. In order to do this, she manually places probes and checks all possible player locations using a special view mode which lets her examine the reflections [Cry11] (see also figure 1.) If a place has too few probes, resulting in wrong reflections, the artist has to create a new probe at an appropriate location.

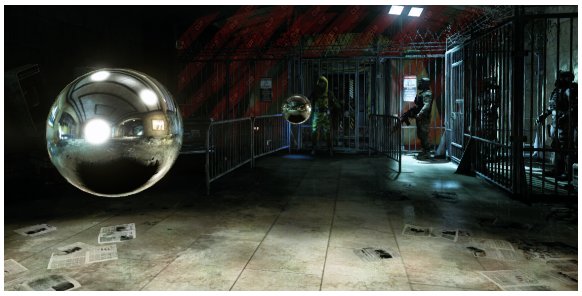
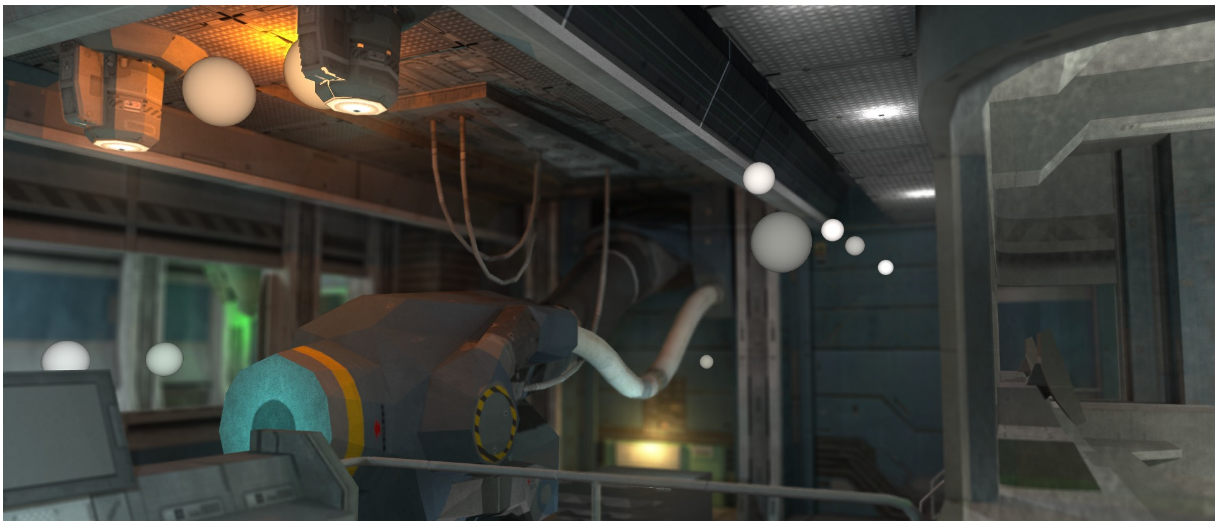


Figure 1: Environment map probes placed in the *Terminal* level in Crysis 2. Scene courtesy of Crytek from Crysis 2.



A sample image showing the probe locations found by our algorithm on the *Alphalabs1* level. Our algorithm places probes at important locations, like the lit control room on the right or the warning lamp in the top left. The small glyphs indicate candidates, while the large glyphs correspond to actually placed probes. Scenes from Doom 3, © id Software

Typically, an engineer will have to re-check the probe placement later in the development process to ensure performance and memory targets are met. For instance, some areas of the level might already stress the memory budget and only allow for a few probes. Once the content changes, both the artist and engineer have to re-evaluate the placed probes to make sure they are still at valid locations.

It is clear that a significant amount of time can be saved by assisting the artist in probe placement, for instance, by guiding her to possible candidate locations. Moreover, by taking performance or gameplay-specific data into account, placing undesired probes can be avoided in areas which are unreachable or where performance issues might arise. The final probe selection and/or re-positioning, however, is always performed by the artist, since one can neither predict the way the game is actually played nor how the artist envisioned the experience for a specific scene.

1.0.0.1. Our contribution: The primary goal of our work is a robust approach which assists the game artist in accomplishing the task of environment map probe placement. Therefore, we present an algorithm that automatically computes a set of important locations. Similar to [War94], finding these locations is based on a search along the directions of major change in a 3D irradiance distribution. Contrarily, however, in our case this distribution is given by a precomputed irradiance volume [GSHG98]. Here, changes can be with respect to the overall incoming radiance or the distribution of this radiance.

Once our algorithm has determined candidate probe location, clusters consisting of environment maps with similar irradiance distribution are determined and replaced by a single representant. The similarity measure we use builds upon the earth-mover's distance [RTG00], a histogram based method for determining the similarity between images.

Together with a game artist we have performed a qualitative assessment of our results, by providing our sampling as an initial guess and letting the artist manually refine this sampling using her expert intuition. The evaluation has been done using game levels from Doom3. As Doom3 is known for its unusual darkness and contrast, we used global illumination for all of our rendering. In this way, our results are more representative for current-generation content.

Our approach is vastly different to previous approaches for automated placement of environment maps, which aimed for maximum visibility coverage, i.e. they tried to place a minimum number of probes such that any part of the surrounding scene is visible from at least one probe [ML03]. When dealing with games, however, maximizing coverage is neither feasible nor desirable. First of all because the budget for environment maps is strictly limited in modern game engines. For example, a typical Cryengine 2 outdoor area with 1 km^2 might use only a single probe [Kap11], meaning that complete coverage is usually impossible to achieve in the first place. Furthermore, coverage only considers the level geometry while completely ignoring lighting and shading effects. In modern game scenarios, lighting is an integral part of level-design and is used extensively to guide the player's attention towards specific spots in a level. When using an approach that is purely based on geometry, many of the implicit importance hints given by lighting are lost.

For instance, a single probe can be usually used for all dark rooms in a game.

2. Environment maps placement

Our algorithm revolves around the assumption that placement of a new probe becomes necessary once the variation in either intensity, hue or direction of incoming light exceeds a certain threshold. However, as long as there is no large, sudden change, we want to reuse the same probe as long as possible. In particular, we assume that a probe placement does not have to capture all of the scene geometry.

2.1. Irradiance field search

We start by constructing an irradiance volume, similar to those used in current games for lighting [GSHG98, Tat05]. In an irradiance volume, probes are placed at the vertices of a regular grid, and each probe captures the irradiance distribution function for all directions at the corresponding vertex. We are interested in specular reflections, which correspond to irradiance moment volumes with $n = \infty$. These can be easily computed by rendering a HDR spherical environment map for each grid vertex.

One notable consequence of using irradiance volumes in our approach is that the computation becomes scene and geometry independent as it only works on the probes, but the final placement is also partially restricted due to the fixed sampling. This is explained in more detail in section 2.3.

Once we have computed the irradiance volume, we want to discover the sinks of this field. We do so by analyzing the irradiance gradient field, which approximates the gradient of the irradiance field at each grid point of the irradiance volume. Instead of using a numerical solver to compute the derivatives of the irradiance distribution as proposed in [WH92], however, we use the irradiance probes directly to compute a single vector $\vec{f}$ that approximates the change in overall irradiance. Recall that each probe consists of a spherical environment map, where the pixel (u_i, v_i) captures the irradiance I in the direction $\vec{\omega}_i$. We compute the result vector by summing up all direction vectors for a single grid point, weighted by the luminance of the corresponding direction:

$$\vec{f} = \sum_{\omega_i \in \Omega} \vec{\omega}_i * \text{luminance}(I(\vec{\omega}_i))$$

An example of a so constructed gradient field is shown in figure 2.

Besides its efficiency, the proposed method for estimating the irradiance gradient has another desirable property. Since game levels often consist of tight hallways and unreachable space, the computation of derivatives becomes quite cumbersome due to the frequent occurrence of boundary cases. The vector $\vec{f}$, on the other hand, can be computed at every location where a probe can bee placed. Having these gradients at hand, we can now perform a search for sinks in the irradiance volume.

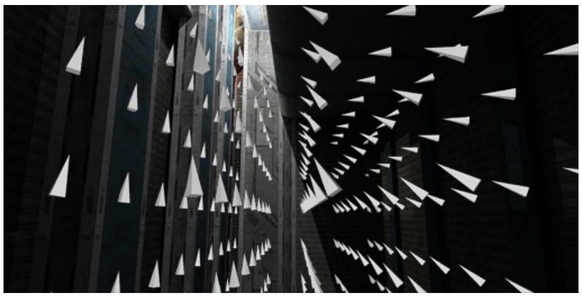
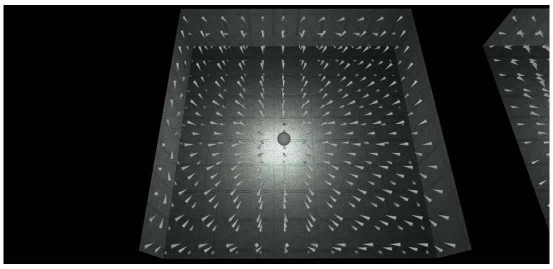


Figure 2: The irradiance gradient field in a dark area of *Alphalabs1*. Each glyph represents the direction of the gradient at a grid cell. The gradients are pointing towards the bright areas near the ceiling.


(a) Empty room with single light source

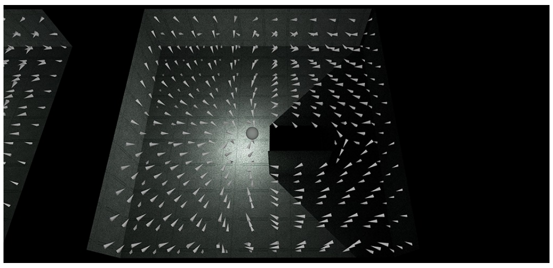

(b) Same room with a pillar at the center

Figure 3: An empty room with a single light source gets assigned a single probe near the floor in the middle of the room. An additional pillar in the room triggers an error case: While a probe in front of the pillar is placed correctly, no probe is set behind the pillar.

The reason why we search for the sinks can be explained with some intuition for what we are really searching for. Let us therefore assume a large, empty room with a single light in the middle, as can be seen in Figure 3. We want to place a single probe in this room, minimizing the visible error for all locations in the room. Clearly, if we place the probe in one of the corners, it will be wrong for most of the room as both the total reflected energy, as well as the direction change significantly. A search for sources of the gradient field would give this result. Recall that our gradient vectors point towards the light, so a source is local light minimum and hence darker than the surrounding area. In order to minimize the error, we need to place the probe somewhere close to the light – in the center of the room. Searching for the maxima, or in this case, equivalently the sinks of the gradient field, gives exactly the desired placement.

To find the sinks in the irradiance volume we use classical particle tracing in vector fields. From each grid point we start a single particle and integrate it through the gradient field using Runge-Kutta scheme of second order. As soon as we discover an area where multiple particle paths end – that is, the particles fall into a closed loop – we mark the probes as candidates. Conceptually, we perform a discretized gradient ascent which is robust and works well on incomplete volumes.

The tracing results in small pockets, or clusters, of probes. All probes of such a cluster are usually very similar, but some may still exhibit significant variance. For instance, probes around a corner can be partially in shadow and partially lit, so even though they are spatially close the contents will differ significantly.

While we could use each cluster as a probe location, but we can prune the candidate set further and make our approach more robust by using a post-process for cleaning up the clusters, which will be described in the next section.

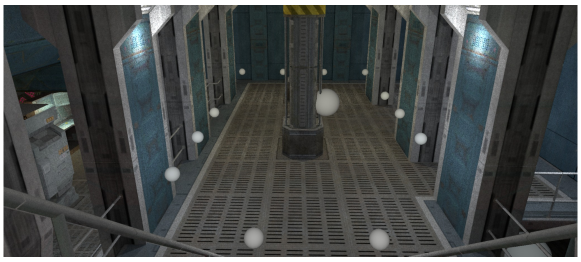


Figure 4: 1D sink manifold formed inside a room. The probes on the 1D line around the room are part of a single cluster, and our algorithm correctly generates a single probe in the center to represent this cluster.

Another common case of clustering is when the sources form 1D or 2D manifolds. For instance, rooms may end up with a line or plane of closely adjacent sources. An example for such a configuration can be seen in Figure 4. In this case, we need a post-process to determine whether the probes can indeed be merged and replaced by a single representative probe.

2.2. Probe matching

As our search process usually results in clusters of probes, we have devised an image-based metric to quickly prune such clusters and to identify those with low confidence. Our metric is based on two separate parts. First of all, we compute the histogram of each probe and use the earth-mover's distance to compare them. In order to reduce the dimensionality of our problem, we use only the luminance and hue instead of three histograms for RGB. By computing luminance and hue histograms upfront, we are able to quickly separate probes with significantly different colors, as well as probes containing bright highlights.

However, it is not enough to discern between probes where only the dominant light direction changes. For instance, the histograms in Figure 5 match each other very well, while the image content is clearly different.

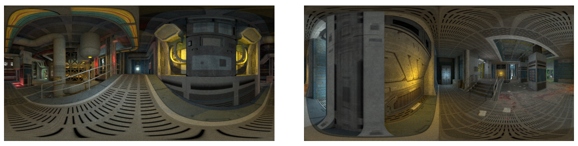


Figure 5: An example case where the histogram metric finds a match. Both images have similar brightness and color, and hence the histogram distance considers them as similar. The FFT metric allows us to distinguish such cases.

We thus employ a second criterion, which is a FFT-based distance metric. For each probe, we compute a low-order FFT and compare the coefficients using

$$dist(\{f_1^0, f_1^1, \dots\}, \{f_2^0, f_2^1, \dots\}) = \sum_i \left((\|f_1^i\| - \|f_2^i\|)^2 * c^{M(i)} \right)$$

with the image frequencies $f_j^i \in \mathbb{C}$ of image j and the Manhattan norm $M(i)$ from the DC component of the 2D FFT. Typically, we use the first 25-100 frequencies only. Comparing probes based on their FFT is similar to comparing the spherical harmonic representation; we used the FFT as it is trivial to include additional frequencies and it can be efficiently computed on the GPU.

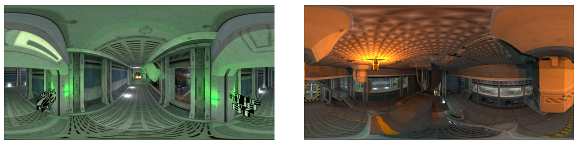


Figure 6: An example case where the FFT distance metric finds a match. Both images have a similar luminance distribution. The histogram metric however distinguishes this images due to the different colors.

2.3. Optimization

The probe matching procedure as described can be used to prune candidate probes, but it has several failure cases which make it unreliable. The main problem is that this method is very sensitive, so it produces many probes and moving a light source slightly can result in vastly different clusters.

On the other hand, perceptually different probes may wind up being classified as similar if there is low frequency variation in luminance and the colors are close under the earth-mover's distance.

Furthermore, probe matching is inherently order-dependent. This can easily be shown by considering a long row of probes: If a new probe is to be placed whenever the difference exceeds a threshold, the order in which the probes are traversed influences the result. For instance, starting from the middle might result in a single probe, while starting from one end could wind up with two probes being created. Finally, the probe matching procedure has no spatial knowledge and hence cannot guarantee good coverage everywhere. Even if spatial distance is taken into account, it is difficult to adapt it to the surrounding scenes – outdoor areas for instance can get away with far less probes than interior scenes.

The gradient field search alone, on the other hand, is not suitable either as it produces large clusters; in particular, it can produce a 1D or 2D manifold with potentially tens or hundreds of probes which are very similar. In such cases, using the gradient field alone is not sufficient to decide which probes can be safely merged and which require the formation of a new cluster.

To overcome these limitations we combine the gradient field search and probe matching process in the following way: We start by computing the probe set, which is an initial dense sampling of the accessible area with low-resolution probes. From these probes, we compute the gradient field and form the "initial cluster set", which contains all candidate locations. Typically, the initial cluster set is already extremely sparse compared to the sampling grid. We clean up the cluster set using our probe matching to form the "final candidate set", where each remaining probe has been assigned to one cluster. For each of these clusters, we eventually estimate a representative probe and store those. The data flow can be seen in Figure 7. Notice in particular that once the probes have been generated, the rest of the algorithm does no longer use the source scene geometry any more.

As we mentioned previously, the initial probe candidates are placed on a fixed grid. This limits the possible locations where a probe can be placed. In order to partially lift this limitation, we compute the average position for each cluster when we are about to place the final probe. As we have no geometric knowledge, an additional image based comparison is used to guarantee that the newly created probe is indeed a good representative for the cluster. This allows us to efficiently resolve cases like the manifold in Figure 4, which requires a new probe on a non-grid location in the middle of the room.

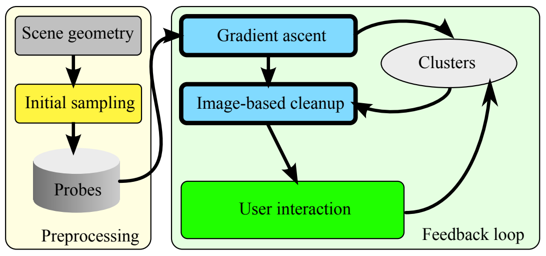


Figure 7: Data flow in our algorithm. After the preprocessing, we only work on the initial probe set. The gradient ascent generates a set of clusters, which is further refined by probe matching. User input is fed back by creating and/or deleting clusters.

In addition to the automatically placed probes, we also provide two additional hints to the artist. First of all, we show the irradiance gradient to the artist so she can quickly understand how the light flows through the scene and in which direction to move. Second, we also show the luminance distance to all surrounding probes. Typically, artists try to minimize the luminance change between adjacent probes. As we know the luminance throughout the whole volume, we can show the locations where the distance becomes too large. This allows artists to resolve the corner cases of our algorithm which are explained in section 3.

2.4. Cluster control

As the gradient field search produces clusters of probes, additional cluster constraints can be integrated into the process at this point quite easily. Clusters which lie inside "importance volumes" can be weighted higher, clusters outside reachable space can be directly pruned, and already existing probes can be compared to nearby clusters. This allows the artist to retain full control where the environment map probes are placed. We can also trivially guide the cluster size by game-data: For instance, if a player spends a significant amount of time in a certain location, we can reduce the spatial extents at which we form clusters and build more clusters to better cover the area. Similarly, we can increase the cluster size for parts where performance is critical, for example, areas where multiplayer profiling shows high player density.

3. Results

To validate our approach, we have tested our algorithm on actual game levels from Doom3 as well as specially crafted test geometry. Even though the Doom3 source data is not as complex as current-generation games, it is a good representative as our algorithm is independent of the actual scene complexity. On the other hand, the level size and the resulting number of probes in Doom3 is still representative for modern games.

We placed probes with 128×64 pixels resolution on a uniform grid throughout the reachable area in the scene. The probe density was set so high that even in tight areas like hallways we would still get 9-12 probes for each cross-section. As the probes are of very low resolution, they can be quickly computed using the in-game renderer or generated during the light baking. Since the time for sample generation highly depends on the game engine and content, we did not include it in our timings and instead focused on the particle tracing and clean-up times.

Figure 3a shows a simple synthetic test case consisting of a single room with one light source at the center. The glyph indicators point into the direction a particle would move during particle tracing. As expected in this case, all particles got drawn towards the light source, resulting in a single probe being placed in the middle of the room.

In Figure 3b we have a similar situation, but now a pillar in the middle of the room occludes parts of the level. In this case, our algorithm still places only one probe in the middle of the room, because particles starting in the pillar's shadow are immediately drawn to the light. This behavior is generally undesirable, as the reflections behind the pillar will be incorrect. Fortunately, such situations are very rare in real-world data, where there are usually at least some smaller ambient lights visible from any point in a level, resulting in additional probes being placed even in darker areas. Nonetheless, completely dark areas are currently not considered by our algorithm and will require the manual placement of additional probes by an artist.

A real-world example from *Alphalabs1* shows a more typical case. At one place in the game level a single bridge leads the player over a deep pit that is completely dark. As can be seen from Figure 2, the computed gradient field leads any particle that starts inside the pit directly to the bridge level. In this particular case, "turbulence" in the gradient field at the bottom of the pit cause a single probe to be placed in the center of the pit near the bottom, resulting in an optimal placement for this area.

Figure 8 shows how probe matching using the image-based metric prevents spatially close clusters from being merged. In this case, the probes from two adjacent rooms are spatially close together and thus considered for merging. However, as the brightness distribution of the probes is significantly different, the image-based metric prevents the clusters from merging.

Since most indoor scenes in games use doors to separate different areas, we have also investigated how the presence of doors affects the outcome of the algorithm and whether the level should be sampled with doors opened or closed. Figure 9 shows the result of this investigation, clearly indicating that the effect of a closed door is negligible for the final result. The influence of doors becomes noticeable only if the amount of light reflected on a door's surface is significantly higher than in its environment.

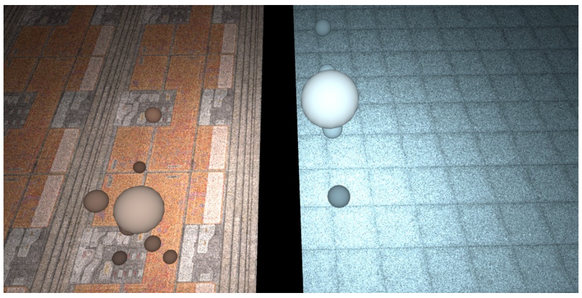


Figure 8: Finding clusters in a simple setup with two different rooms that are separated by walls. The light sources are placed close to the walls. Our algorithm is able to discern between each independent cluster, even though they are spatially close together.

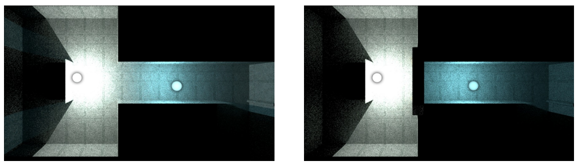


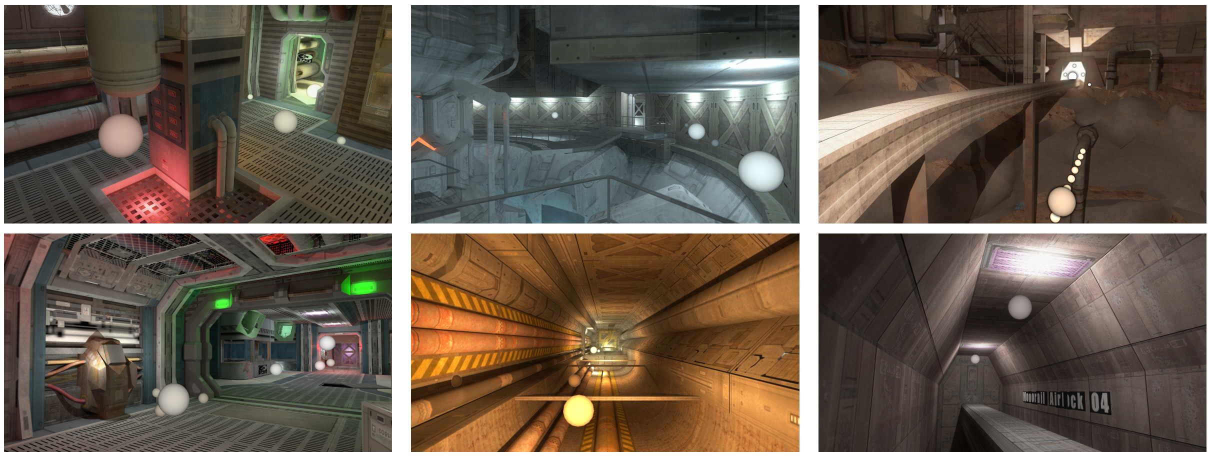
Figure 9: Open or closed doors do not have much impact on the placement, as it mostly depends on the reflected light. The two light sources on both sides of the wall trigger the placement of the probe in every case.

We have tested a representative subset of the Doom3 levels: *EnPro*, *Alphalabs1* and *Monorail*. *Alphalabs1* is a typical claustrophobic space station of tight hallways and corridors. *Monorail*, on the other hand, is dominated by two large outdoor areas, while *EnPro* is a mixture of both tight and open areas.

Some typical placements for these levels are shown in Figure 10. Note that game-specific information was not considered when generating the samples. In particular, probes are usually not generated at the player's eye level and might get placed in unreachable areas. However, this can be fixed easily in a post-processing step.

Figure 10a shows two typical indoor scenes from the *Alpalabs1* level. In the upper picture, three probes were placed in the scene: Two in the main room—one on each side of the central pillar—and a third one in the adjacent hallway. The bright red highlight to the left of the pillar is very prominent and thus triggers placement of a separate probe, while the central probe gives a good overall estimate of the reflection experienced in the rest of the room. The third probe is placed right behind the doorway. This is a typical situation, as the lighting situation changes drastically when stepping into the hallway. The algorithm successfully detects this from the irradiance field trace and adds a probe accordingly.

Similar behavior can be observed in the lower picture. The



(a) Alphalabs 1 (b) Energy processing and storage (c) Monorail

Figure 10: Typical probe placements in our sample levels. Pictures were generated with the same renderer that was used for generating the sample probes in our algorithm. Small glyphs indicate candidates for probes before clustering. Large glyphs indicate the final positions of environment probes as generated by our algorithm.

area is roughly divided into three rooms, with one probe placed at the center of each room. Additional probes get placed near the central cabin in the left part of the main room and under the red highlight at the back-door. These additional probes reflect the different lighting situations in the main room. Notice that the algorithm found a number of potential probe candidates for the airlock in the foreground, which get clustered and eventually result in a single probe being placed in the center of the room.

The next set of samples in Figure 10b shows some wider areas from the *EnPro* level. The upper picture was taken in a large cylindrical room. When moving along the outer wall of the cylinder, the generated environment maps change very slowly. This is because the reflected geometry is very homogeneous but the direction changes from which the light (e.g. the reddish light) at the center reflects. Our algorithm accounts for that, by periodically clustering probe candidates together, thus resulting in a sparse, regular placement of probes along the outer walls.

The lower picture shows a very long lift shaft spanning the whole height of the level. Our algorithm captures the very different light situations at both ends while it places only a single probe in the interior part.

The third sample set from the *Monorail* level is shown in Figure 10c. An importance volume has been placed around the tracks, as the player cannot freely move in this level. This restricts the number of probes significantly, as the level consists of huge empty volumes which are unreachable for the player.

The upper picture shows how our algorithm behaves for an open, outdoor environment. A large manifold of probe candidates is found parallel to the railway track. The image-based metric is able to merge those into a single probe. A second set of probes is found near the door in the back wall, which is flanked by additional light sources. This placement effectively models the reflection seen on the windows of a train moving along the railway tracks. The lower picture shows the placement of probes in a tunnel. Again, probes are placed close to areas of prominent changes in lighting, in this case due to the ceiling lights.

The most important parameter of the algorithm is the sampling density of the initial regular grid. Naturally, if the grid is too coarse, tight spaces will not get sampled correctly. On the other hand, sampling huge volumes of empty space with a high density is a careless waste of computational resources, so those should be sampled at lower densities.

In general, the grid sampling should be roughly in the same order as the character's size. Higher sampling densities will eventually result in more probes being generated, as more subtle features in the level will be detected. This can be a desirable effect if a detailed coverage of the scene is required, for instance, when using the environment maps as irradiance estimate for global illumination.

Table 1 shows some statistics of our algorithm for probe placement. The grid density was adjusted manually based on the overall layout of the level. For instance, the *EnPro* level required a relatively dense sampling to be able to capture the tight hallways accordingly, but the majority of probes was placed in two large vertical volumes in the middle of the level. An adaptive placement could resolve such cases.

As can be seen from the table, the irradiance field search is able to isolate a small subset of grid probes as potential

Scene	# Grid	# Candidate	# Final	t (s)
Alphalabs	8925	205 (2.3%)	117	2
EnPro	17640	208 (1.2%)	80	5
Monorail	11143	166 (1.5%)	29	2
AlphalabsHD	67980	1139 (1.7%)	508	121

Table 1: Statistics showing typical probe counts and processing times. From left to right: Number of sampled grid probes for irradiance approximation; Number of candidates found via particle tracing; Remaining number of probes after clustering; Computation time for particle tracing and clustering.

candidates. Since these candidates tend to form clusters of spatially close probes, the image-based metric is able to further reduce this set by a factor of 2-3. Note that although *Alphalabs* started with far fewer initial grid probes than *EnPro*, it ended up having more probes in the final set. This is mainly due to the fact that large empty volumes in *EnPro* tend to contain many grid probes that do not add any information to the irradiance field.

The computation times in the last column include the time for building the irradiance gradient field, perform the particle tracing and clustering the probe candidates using the image-based metric. They were measured on a 2.8 GHz Dual Xeon X5560 machine (8C/16T); as the algorithm is embarrassingly parallel, we used all cores of the machine. As a special case, we included the numbers for the *Alphalabs* level sampled at double density in each direction to show how the algorithm scales. The higher processing time is mostly related to longer particle tracing time through the denser grid.

Again, the initial sampling density turned out to be the key parameter of the algorithm. For scenarios in which a dense sampling is required, e.g. when generating probes for global illumination, an adaptive grid sampling will help reducing computation times considerably.

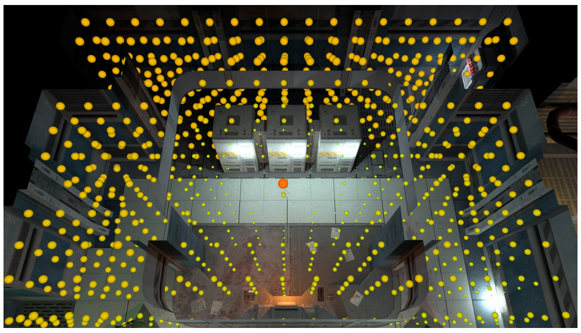


Figure 11: Luminance error field around a single probe in the center of the screen. Probes inside the control room have comparable brightness, while probes behind the boxes are much darker. The visualization helps artists to identify regions where noticeable changes would occur.

We have evaluated the quality of our placement by an artist who worked on Doom3, as well as an artist who was not particularly familiar with the setting. The vast majority of the changes involved moving the probes up to eye-level and away from walls, which we could not do automatically due to lack of player and collusion geometry. The artists indicated that the probe density is very similar to manually placed probes. In the future, we hope to get access to source data with already placed environment map probes for a more thorough validation.

Besides the initial placement, the artists found the gradient field visualization and the luminance distance helpful, which can be see in Figure 11. The latter allows an artist to quickly identify how much of the volume contains probes of similar brightness.

4. Conclusion

We have presented a novel approach to guide artists while placing environment reflection map probes. Our algorithm uses the irradiance gradient field which represents the reflected light as well as an adapted image-space metric which together provide a robust way to place environment map probes in complex game environments.

Our algorithm is an early exploratory approach to solve the problem. In the future, we would like to explore a more adaptive approach which adjusts the initial probe density based on the local geometric complexity. For instance, the sample rate should decrease in large outdoor areas and increase in tight places or situations with complex lighting.

We would also like to resolve the case where extremely dark areas may end up without any samples at all, as all samples have been moved towards the closest light. This failure case can be seen in Figure 3b. While it is possible to get an error indication for each probe by simply comparing the sample to the closest probe, we expect that our algorithm can be modified to directly resolve such cases.

Eventually, we expect that our algorithm could be modified to produce probe locations which are suitable for image-based lighting approaches. This requires a more rigorous treatment of the remaining error.

Acknowledgements

Thanks to Vicarious Visions for helpful comments and result validation. Thanks to Anton Kaplanyan for comments. Special thanks to Alexandru Adrian Radoiu for additional testing and feedback.

References

[BN76] Blinn J. F., Newell M. E.: Texture and reflection in computer generated images. *Commun. ACM 19* (October 1976), 542–547. 1

[Cry11] CRYTEK: CryEngine Documentation: Environment probes (http://freesdk.crydev.net/display/SDKDOC2/Environment+Probes), 2011. 1

[Gre86] GREENE N.: Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl. 6* (November 1986), 21–29. 1

[GSHG98] GREGER G., SHIRLEY P., HUBBARD P. M., GREENBERG D. P.: The irradiance volume. *IEEE Computer Graphics and Applications 18* (1998), 32–43. 2, 3

[Kap11] KAPLANYAN A.: Personal communication with Anton Kaplanyan, 2011. 2

[McT04] MCTAGGART G.: Half-life 2 / valve source shading. Game Developer's Conference, March 2004. 1

[ML03] MEYER A., LOSCOS C.: Real-time reflection on moving vehicles in urban environments. In *Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 2003), VRST '03, ACM, pp. 32–40. 2

[RTG00] RUBNER Y., TOMASI C., GUIBAS L. J.: The earth moverâĂŹs distance as a metric for image retrieval. *International Journal of Computer Vision 40* (2000), 2000. 2

[Tat05] TATARCHUK N.: Irradiance volumes for games. Game Developer's Conference, 2005. 3

[War94] WARD G. J.: The radiance lighting simulation and rendering system. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 459–472. 2

[WH92] WARD G. J., HECKBERT P. S.: Irradiance gradients, 1992. 3