Distributed High-Fidelity Graphics using P2P

Daniel D'Agostino

In the field of three-dimensional computer graphics, rendering refers to the process of generating images of a scene from a particular viewpoint. There are many different ways to do this, from the highly interactive real-time rendering methods [1] to the more photorealistic and computationally intensive methods [5].

This work is concerned with *Physically Based Rendering* (PBR), a class of rendering algorithms capable of achieving a very high level of realism. This is achievable thanks to physically accurate modelling of the way light interacts with objects in a scene [7], together with the use of accurately modelled materials and physical quantities.

Unfortunately, this realism comes at a cost. PBR computations are very expensive, and it may take several hours to render a single image. Hence, it is no surprise that most of the research in this field attempts to find ways to reduce that cost, and produce images in less time.

In spite of their diversity, all physically-based rendering algorithms attempt to solve the same problem: imitating the visual appearance of an environment as it would look in real life. This problem is formulated as the *rendering equation* [6]. Based on the principle of conservation of energy, this equation states that the light leaving a surface comprises both the light emitted from the surface, and that being reflected from other surfaces.

Due to factors relating to geometric complexity of the scene, it is not possible to solve the rendering equation analytically [7]. This difficulty mandates the use of numerical techniques, the most popular of which are Monte Carlo techniques. These were introduced to PBR with distributed ray tracing [3], a technique based on the ray tracing method [11] which could collect light arriving at a surface from many different directions in order to include fuzzy phenomena (such as soft shadows).

Irradiance caching [10] is a particularly useful technique that can be used in conjunction with distributed ray tracing (as well as other algorithms). Based on the observation that indirect diffuse lighting varies slowly over a surface [10], it stores irradiance (view-independent lighting data) in a data structure (usually an octree) and uses fast interpolation in order to speed up computation.

While one side of PBR research has been formulating more efficient algorithms to render physically-based images faster, another has been exploiting the embarrassingly parallel nature of ray tracing [4] in order to distribute the rendering load on many processors or interconnected machines [9].

Research into parallel PBR has so far almost exclusively used the client/server model. Although some rendering research based on the peer-to-peer (P2P) model has recently emerged [8, 2, 12], at present (to the best of our knowledge) there is none that is designed to improve physically-based rendering systems.

The method we are developing is based on the irradiance caching algorithm. Given that irradiance stored by a renderer is view-independent, it is easy to share with other machines, who can then use it directly without having to recompute it locally. This makes it particularly suitable for use over a P2P network.

Aside from the obvious benefits of free computation resulting in overall speedup, this novel way of sharing irradiance between peers is expected to provide new and interesting scenarios in which PBR can be used for collaboration between several peers in the same scene, as opposed to the traditional client/server model where the clients would do the rendering work and the server would be the only one to see the final images.

References

1. Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA, 2008.
2. Azzedine Boukerche and Richard Werner Nelem Pazzi. A peer-to-peer approach for remote rendering and image streaming in walkthrough applications. In *ICC*, pages 1692–1697. IEEE, 2007.
3. Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 137–145, New York, NY, USA, 1984. ACM.
4. Thomas W. Crockett. Parallel rendering. *Parallel Computing*, 23:335–371, 1995.
5. P. Dutré, K. Bala, and P. Bekaert. *Advanced global illumination*. Ak Peters Series. AK Peters, 2006.
6. James T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.
7. M. Pharr and G. Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann. Elsevier Science, 2010.
8. J. A. Mateos Ramos, C. Gonzlez-Morcillo, D. Vallejo Fernndez, and L. M. Lpez-Lpez. Yafrid-NG: A Peer to peer Architecture for Physically Based Rendering. pages 227–230.
9. I. Wald and P. Slusallek. State of the art in interactive ray tracing. *STAR, EUROGRAPHICS 2001*, pages 21–42, 2001.
10. Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 85–92, New York, NY, USA, 1988. ACM.
11. Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.
12. Minhui Zhu, Sebastien Mondet, Géraldine Morin, Wei Tsang Ooi, and Wei Cheng. Towards peer-assisted rendering in networked virtual environments. In *Proceedings of the 19th ACM international conference on Multimedia*, MM '11, pages 183–192, New York, NY, USA, 2011. ACM.

Fig. 1. The Sibenik Cathedral, rendered using our Irradiance Cache

Fig. 2. The Kalabsha Temple, rendered using our Irradiance Cache