Runtime Implementation of Modular Radiance Transfer

Brad Loos 1, Lakulish Antani 2, Kenny Mitchell 3, Derek Nowrouzezahrai 4, Wojciech Jarosz 4, Peter-Pike Sloan 4

1 University of Utah 2 UNC Chapel Hill 3 Disney Interactive Studios 4 Disney Research Zürich

Direct Lighting  Indirect Lighting (with Multi-Bounce)  Direct & Indirect Lighting (with Multi-Bounce)

16.3 ms = 61 FPS  Our Method (1.7ms = 588 FPS)  Ground Truth  Our Method  Ground Truth

Figure 1: Indirect light computed for a 19 block maze-like cave scene constructed completely at run-time. Our algorithm can model color bleeding and indirect light from surfaces with detailed normal variation and real-time performance. Timings are on an NVIDIA 480 GTX.

1 Introduction

Real-time rendering of indirect lighting significantly enhances the sense of realism in video games. Unfortunately, previously including such effects often required time consuming scene dependent precomputation and heavy runtime computations unsuitable for low-end devices, such as mobile phones or game consoles. Modular Radiance Transfer (MRT) [Loos et al. 2011] is a recent technique that computes approximate direct-to-indirect transfer [Hasan et al. 2006; Kontkanen et al. 2006; Lehtinen et al. 2008] by warping and combining light transport, in real-time, from a small library of simple shapes. This talk focusses on implementation issues of the MRT technical paper, including how our run time is designed to scale across many different platforms, from iPhones to modern GPUs.

2 Modular Radiance Transfer Overview

MRT is a novel technique for approximate direct-to-indirect transfer. The key contributions are the use of a lighting prior to reduce the dimension/entropy of the one-bounce light transport operator, approximating a scene with simpler shapes, and efficiently computing indirect lighting on these simpler shapes by expressing both self transfer and propagation between shapes using carefully constructed reduced-dimensional spaces. We achieve high performance, across a wide range of platforms, with dynamically generated scenes (see [Loos et al. 2011] for more details). We compute indirect lighting as

\[ l_{\text{ind}} = U_b b + U_r r, \]

where \( b = T_{d \rightarrow b} l_d \) are the spectral indirect lighting coefficients, and \( r = T_{b \rightarrow r} b \) are reduced lightfield response coefficients. The \( T_{d \rightarrow b} \) operator maps direct light \( l_d \) to indirect light \( l_d \) and \( T_{b \rightarrow r} \) maps indirect light at all the scene blocks to reduced dimensional lightfields at the interfaces of these blocks. \( U_b \) is a compact basis for indirect light inside a shape, and \( U_r \) is a compact basis for mapping light flowing through a lightfield to response from a shape to its neighbors.

3 Run Time Algorithm Overview

The core run time steps for MRT are as follows:

1. Compute direct light at a reduced resolution in each block \( l_d \).
2. Compute per-block spectral coefficients \( b = T_{d \rightarrow b} l_d \).
3. Compute indirect light within a block into a lightmap \( U_b b \).
4. Compute response coefficients at interfaces \( r = T_{b \rightarrow r} b \).
5. Blend response from "external" blocks into a lightmap \( U_r r \).
6. Pad the lightmap texture to avoid texture mapping artifacts.
7. Render scene using the dynamic lightmaps of indirect lighting.
8. [optional] Compute indirect light volume (from b’s and r’s).
9. [optional] Render dynamic objects the volume lighting.

We describe two run times: a DX11 version entirely on the GPU, and a CPU version for low-end platforms (iPhones and iPads).

4 Implementation Details

There are several ways to map the computation to the GPU, involving trade-offs in the number of passes, data layout in memory, execution kernels scheduling, etc. On low-end devices the GPUs are underpowered (e.g., a single unshadowed point-light evaluation costs 25ms on the iPad) and often missing important features such as high precision texture formats. This motivated our CPU-focused implementation for those platforms.

The differences between GPU and CPU architectures lead to different solutions. In particular, we transformed the GPU data layout used on higher-end platforms to a CPU friendly one, optimizing for better cache usage and proper exploitation of CPU vector instructions.

5 Conclusion

MRT is a compelling, real-time approximate global illumination technique. This talk discusses implementation details that allow the algorithm to scale across different hardware platforms with wildly varying capabilities. We believe some of the techniques employed by MRT may be useful to developers facing similar scaling challenges.

References