A Real-Time Transfer Matrix Implementation of Layered Materials

Soren Saville Scott

University of Sydney

Introduction

”A Real-Time Transfer Matrix Implementation of Layered Materials”

Breaking this down...

Realtime Rendering

Rendering: Displaying 3D scenes to the screen, simulating lighting, shadows and other physical phenomena.

Realtime: Doing the above *very fast*. Usually within 16.66ms, in order to attain persistence of vision and fluid motion.

This constraint typically excludes general light-transport techniques such as ray tracing and path tracing.

Materials

A model that defines how a given surface interacts with light.

characterised by a Bidirectional 'x' Distribution Function.

For input direction ω_i and output direction ω_o, returns ratio of input to output energy after interacting with the surface.

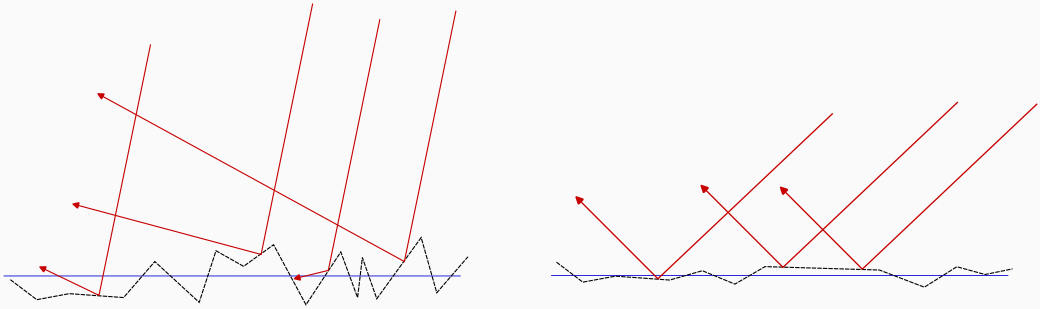
BxDF

Many flavours of BxDF:

- Bidirectional Reflectance Distribution Function (BRDF). Most common. Accounts for reflectance at surface boundary only.
- Bidirectional Transmission Distribution Function (BTDF). Accounts for transmission *through* a surface.
- Bidirectional Scattering Distribution Function (BSDF). Composition of both BRDF and BTDF. General form for interactions with a surface.

Canonical Examples

The microfacet BRDF is the model used by almost all contemporary renderers.



(a) Rough Microfacet Distribution

(b) Smooth Microfacet Distribution

Figure 1: Noisier distribution of facets produces a rougher reflected image.

Models surfaces as field of undulating perfect mirrors (facets). Surface appearance defined by the distribution of facets.

Microfacet Theory

Mathematical definition of the microfacet BRDF:

$$f(\omega_i, \omega_o, \alpha) = \frac{F(\omega_o, \omega_h) G_2(\omega_o, \omega_i, \omega_h, \alpha) D(\omega_h, \alpha)}{4|\omega_g \cdot \omega_o||\omega_g \cdot \omega_i|} \tag{1}$$

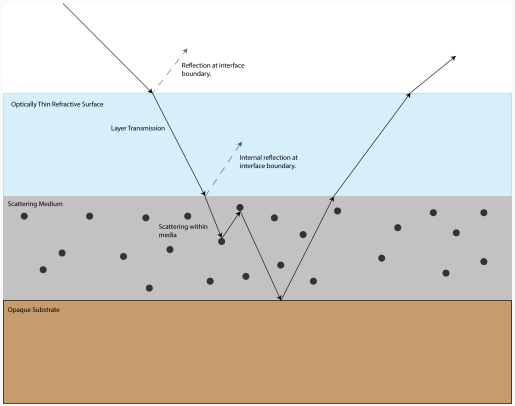
For incident direction ω_i, outgoing direction ω_o and surface roughness α

- F - Fresnel Term. Describes the reflected light for the given surfaces index of refraction.
- G - Geometry Term. Attenuates energy by the proportion of facets visible from a viewpoint.
- D - Distribution of Visible Normals (vNDF) term. Models the facet field's facing directions.

(Heitz, 2014)

Material Layering

A framework that allows artists to design materials by mixing and stacking individual materials together.

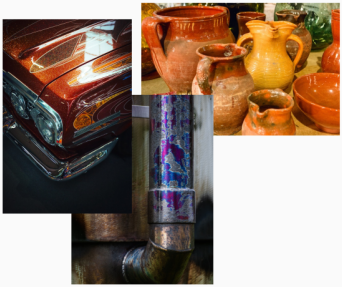


Interactions between incoming light and the whole layer stack is computed. Resulting BSDF is a composition of each layer's BSDF.

Material Layering

Useful for simulating common surfaces such as:

- paint
- varnish
- dirt
- patinas.



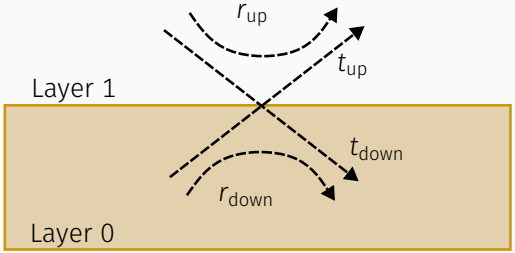
In general, transfers more artistic **freedom** and **flexibility** from engineers to artists.

This is the goal!

Photos by Sindy Süßengut, Aleksi Partanen, Eric Prouzet on Unsplash

Transfer Matrices

A model that encodes light transmission and reflection at an interface as a matrix of fluxes.



$$M_{01} = \frac{1}{t_{\text{down}}} \begin{bmatrix} 1 & -r_{\text{down}} \\ r_{\text{up}} & t_{\text{down}} t_{\text{up}} - r_{\text{up}} r_{\text{down}} \end{bmatrix} \quad (2)$$

Layers can be composed together by multiplying their respective transfer matrices.

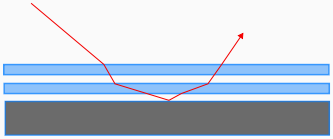
$$M_{ij} = M_{ii+1} M_{i+1i+2} \dots M_{j-1j} \quad (3)$$

(Randrianandrasana, Callet, & Lucas, 2021).

Transfer Matrices

Randrianandrasana et al. use this as a basis for material layering. They develop:

- a 2x2 (2-flux) model, accounting for optically thin dielectric coatings with roughness.
- a 6x6 (6-flux) model, simulating the above, plus multiple scattering within a medium.



Ray transmitting through optically thin layers.

Ray scattering through an optically thick medium.

Layer Parameterisation

Layers are defined as a tuple of the following parameters:

- $\eta + i\kappa$: Complex Index of Refraction. $\kappa > 0$ means the medium is opaque to light.
- α: The microfacet roughness of the layer interface.

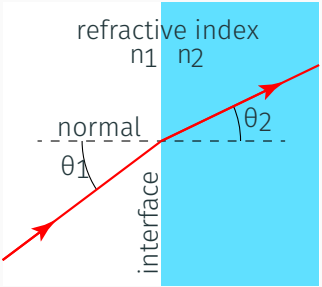


Figure 2: Refraction of a ray at interface boundary.

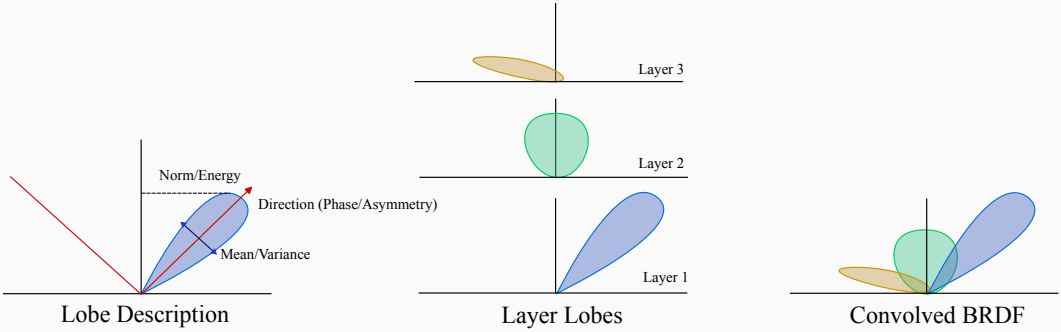
The 6-flux model includes additional parameters for medium depth and scattering behaviour.

Diagram by Ulflund - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=30460545

Output

Summarises resulting BSDF as set of Henyey-Greenstein Lobes.

Exploit convolutional property of HG phase function that is not possible with microfacet model.



Approximate transform from convolved HG representation to microfacet model is given.

Transfer Matrices

The transfer matrix approach is attractive as it is conceptually simple and unifies many effects together into a single operator.

Contrast with other layering approaches, such as Belcour's: 9 distinct operators for handling each form of light interaction.

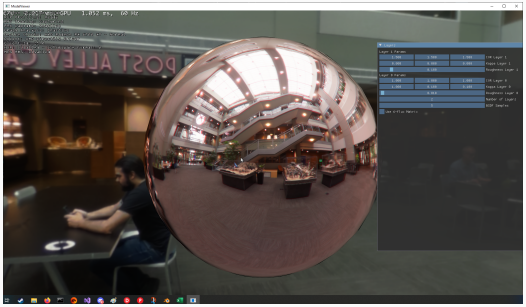
Research & Results

What This Research Achieves

- Fit the Transfer Matrix Model to real-time rendering. This required:
  - Implementing the 2-flux and 6-flux model in a real-time forward rendered rasteriser.
  - Rewriting the sampling scheme for pre-integrated lighting.
  - Replacing FGD LUT with implementation of de Dinechin and Belcour's split-sum approximation.
- Profiling and analysis of performance.
  - Comprehensive analysis of the model's performance characteristics.
  - Applying common optimisations in the literature.
  - Development of a naive analytical estimator for Total Internal Reflection.

Real-Time Transfer Matrices

Transfer Matrix BSDF extension added to Microsoft's 'MiniEngine' sample project. Uses DirectX 12.



UI Interface for experimenting with layer parameters in realtime.

Forward shaded. Includes profiling tools and a library of environment maps for image-based lighting.

Preintegrated Lighting

Solve the 'Light Transport Equation' (Pharr & Humphreys, 2023):

$$L_o(\omega_o) = \int_\Omega f(\omega_o, \omega_i) L(\omega_i) |\cos\theta_i| d\omega_i \tag{4}$$

'The integral of all incoming light from all incoming directions, attenuated by the surface BxDF.'

There is no closed form for this integral, typically solve it via Monte Carlo path tracing.

Preintegrated Lighting

Cannot afford to path trace in real-time!

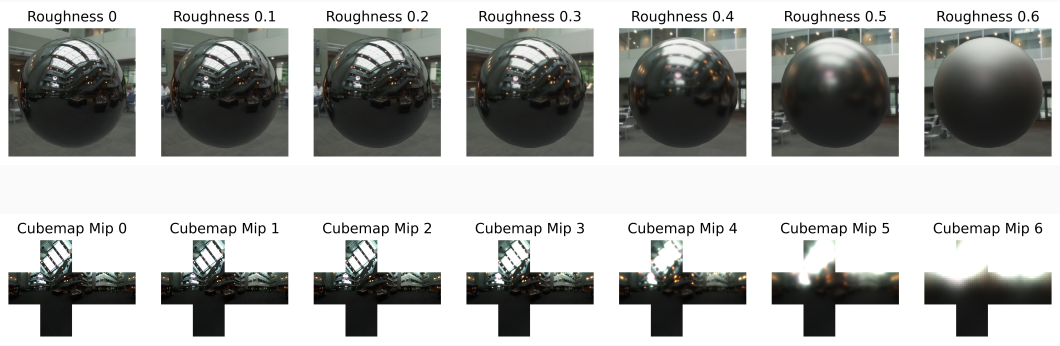
Sometimes there are convenient analytical approximations. Trivial, so won't be discussed here.

No analytical form exists for environment lights, so we must pre-integrate it.

Reflect many rays off surface by importance sampling BxDF, and sample light source at their hit locations. Store the result in a table.

Preintegrated Lighting

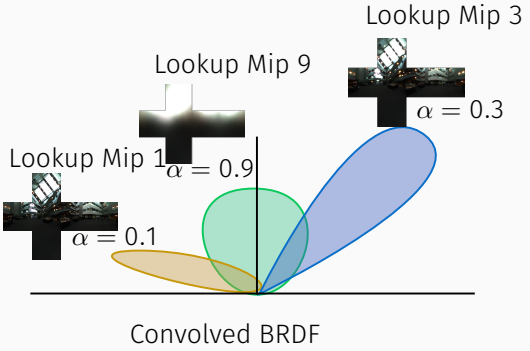
An example with microfacet BRDF pre-integration:



The result of the pre-integration for each roughness level is stored in the mip-chain for the environment light's cubemap.

Transfer Matrix Preintegration

- The transfer matrix model outputs several microfacet lobes. These are composed together by a weighted scheme for the final throughput.
- When sampling each lobe, look-up the preintegrated cubemap for the light energy, weighted by the number of lobes.



Lighting Solved.

On to the FGD LUT.

The FGD LUT

Recall the microfacet BRDF:

$$f(\omega_i, \omega_o, \alpha) = \frac{F(\omega_o, \omega_h) G_2(\omega_o, \omega_i, \omega_h, \alpha) D(\omega_h, \alpha)}{4|\omega_g \cdot \omega_o||\omega_g \cdot \omega_i|} \tag{5}$$

(Heitz, 2014)

Integrating over the *FGD* term describes the directional albedo of the surface.

Randrianandrasana et al. precompute the integration into a 4D $64 \times 64 \times 64 \times 64$ LUT. This is used to create the layer's transfer matrix.

Implementation Detail

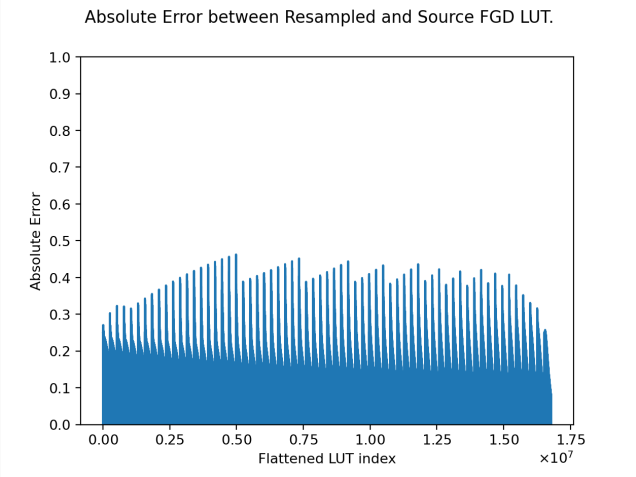
The 4D LUT is too big for DirectX 12! Supports up to 3D resources, with a maximum resolution of 2048 on any axis.

Average the 4th axis down to half-resolution, stack onto the 3rd and rebuild the index in software?

$$64 \times 64 \times 64 \times 64 \rightarrow 64 \times 64 \times 2048 \tag{6}$$

Resampling Error

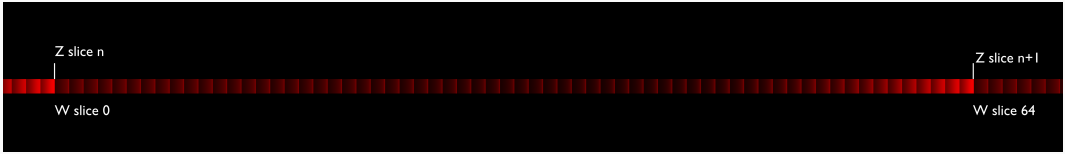
Introduces resampling error. Up to a maximum of 46%.



Naive box filter was most effective, other filters had significantly higher error rates.

Resampling Error

Additional error introduced at runtime when sampling. GPU texture hardware will trilinearly interpolate between incorrect values when on axis boundaries:



Validation tests showed this to be unacceptable. A different approach was needed.

Split-Sum FGD

Real-time renderers precompute the directional albedo integral for use with preintegrated lighting (Karis, 2013).

LUT precomputed as an $n \times n$ texture with 2 channels, where each entry is of the form:

$$\begin{bmatrix} X \\ Y \end{bmatrix}_{\omega_i,\alpha} = \int_\Omega \begin{bmatrix} S_0(\omega_i,\omega_o) \\ S_1(\omega_i,\omega_o) \end{bmatrix} G(\omega_i,\omega_o\alpha)D(\omega_i,\omega_o\alpha)d\omega_o \tag{7}$$

where S_i is the Schlick Fresnel basis functions.

The *FGD* term is reconstructed at runtime by the following formula:

$$FGD_{\omega_i,\alpha} \approx F_0 X_{\omega_i,\alpha} + (1 - F_0)Y_{\omega_i,\alpha} \tag{8}$$

Where F_0 is the Schlick Fresnel reflectance parameter. This is (very) *roughly* analogous to the medium's Index of Refraction.

Split-Sum FGD

Can't use Karis' Split-Sum LUT, as it depends on Schlick's Fresnel.

Use Belcour, Bati, and Barla (2020) form of the split-sum LUT.

Accounts for exact Fresnel, decomposed into 4 basis functions.

Precomputed into an $n \times n$ texture with 4 channels. Same precomputation scheme as previous slide.

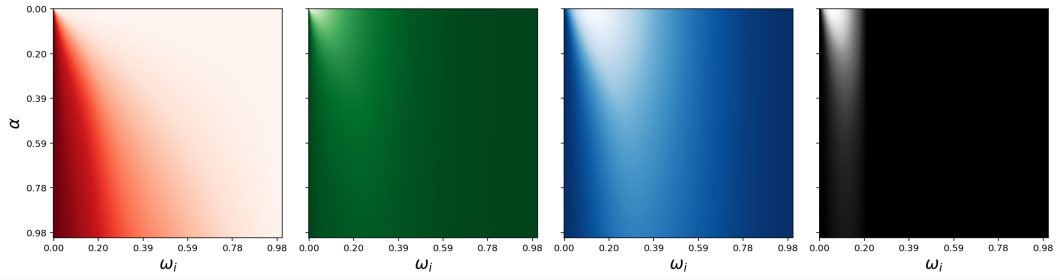


Figure 3: My implementation of the Belcour FGD LUT

Belcour FGD

Reconstructing the *FGD* term at runtime is very similar:

$$FGD_{\omega_i,\alpha} \approx c_1(\eta + i\kappa)X_{\omega_i,\alpha} + c_2(\eta + i\kappa)Y_{\omega_i,\alpha} + c_3(\eta + i\kappa)Z_{\omega_i,\alpha} + c_4(\eta + i\kappa)W_{\omega_i,\alpha} \tag{9}$$

For each X, Y, Z, W channel in the LUT.

c_i is the ith coefficient for the Fresnel basis function, computed at runtime via a projection operator developed in Belcour et al.'s paper.

This form is both smaller and more accurate than the resampled 4D LUT. The reconstruction step adds some small runtime overhead.

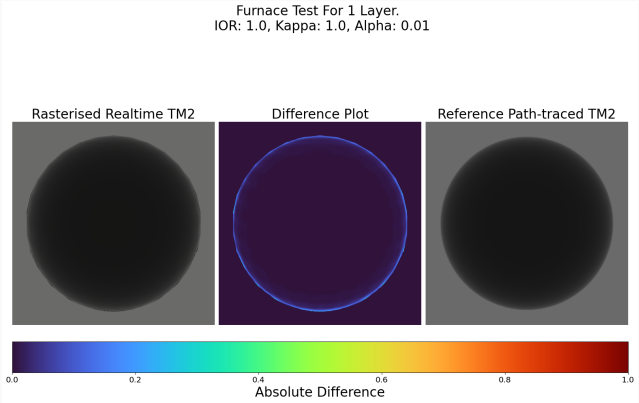
Validation

Validation

Compare renders from my implementation against reference implementation provided by Randrianandrasana et al..

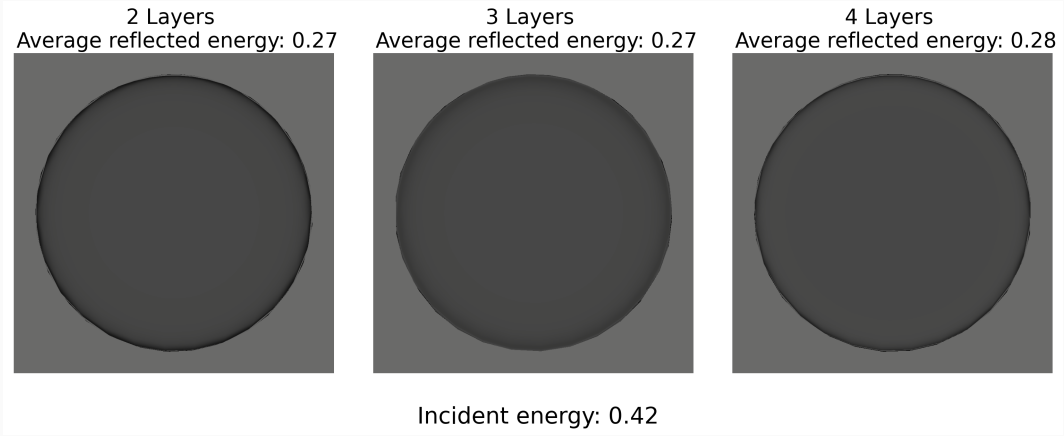
Use a ’furnace test’ to detect energy loss or gain that breaks conservation.

Validation



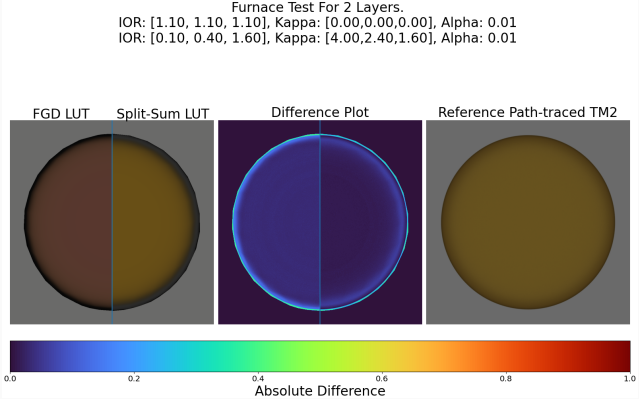
Overall, quite close to the reference implementation. Generally less than 10% error!

Energy Conservation



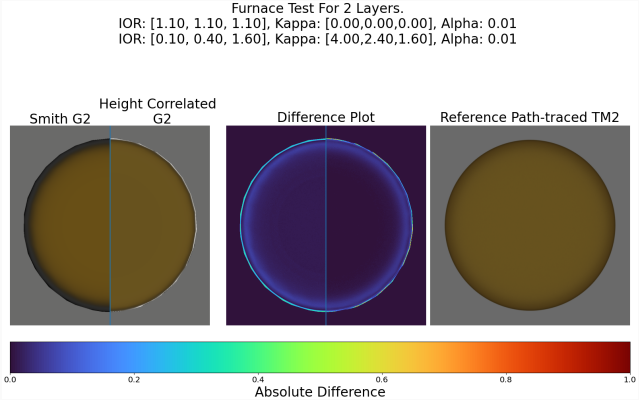
Can also see that the model conserves energy over several layers.

Split-Sum Validation



The split-sum LUT is significantly more accurate than the resampled FGD LUT.

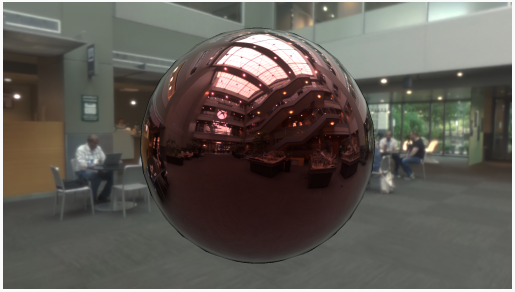
Edge Darkening



The over-darkening at grazing angles can be corrected by using a height-correlated *G* geometry term described by Heitz, rather than the Smith form.

Performance

Test Setup



Average, min and max timings over 64 frames for just the pixel shader draw call. 1920x1080 resolution. Transfer matrix pixels occupy 100% of screen area.

Figure 4: Reference Scene Used for Profiling

Profiled on an AMD 6700XT GPU and Ryzen 5 CPU.

1024 VGPRs per SIMD. 16 wave slots per SIMD.

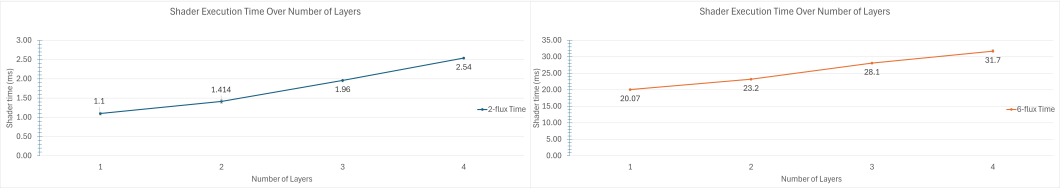
Baseline Timings

	Average(ms)	Minimum (ms)	Maximum (ms)
2-flux Model	1.433	1.401	1.486
6-flux model	23.275	23.167	23.536

Runtime does not vary over roughness or index of refraction.

A non-zero depth in the 6-flux model makes the layer no longer optically thin, introducing multiple scattering.

Layer Dependence



Runtime is dependent on the number of layers in a stack.

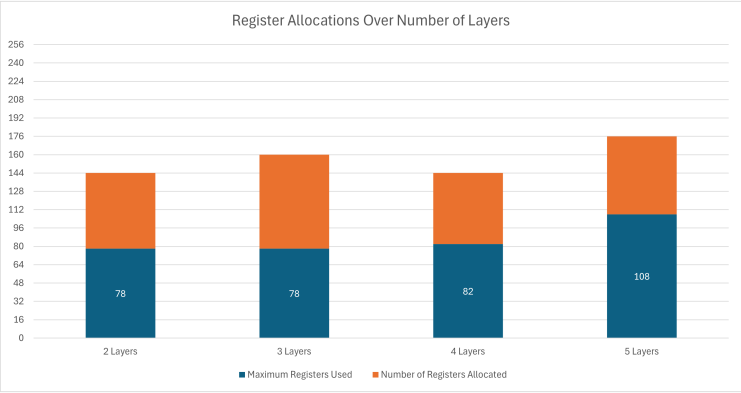
Growth is roughly linear. Corresponds to the number of matrix multiplications (2 per layer).

Dominating Factor

Size of the model is dominating factor. Generally too large to fit into cache/register limits.

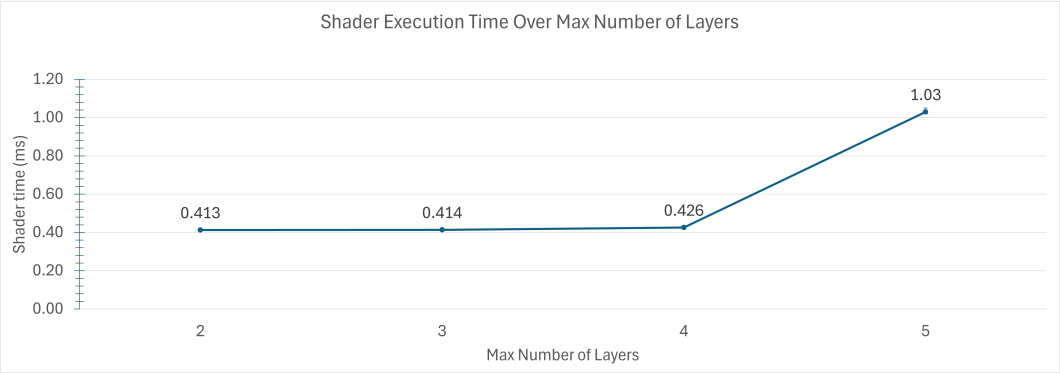
Layers allocated statically - high overhead.

Low occupancy, cache thrashing and use of high latency global memory.



Static Maximum Layers

Reducing size of static layer stack arrays had outsized impact on performance.

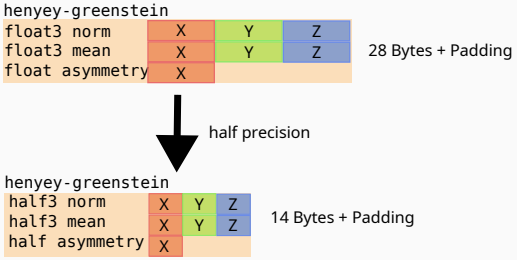


Not unreasonable to fix this at compile time per-material, as *Substrate* does.

Half Precision Types

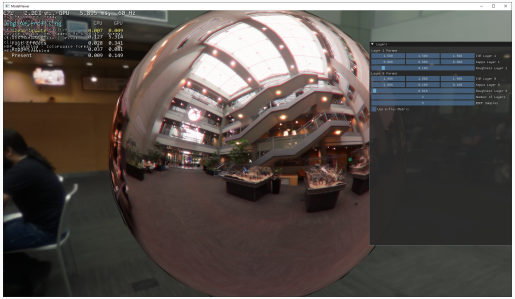
Performance is bottlenecked by model size?

Most hardware supports half precision floating point arithmetic.
In theory, halve the size of *every* structure.
Bonus: profiled hardware benefits from double-rate throughput.



Doesn’t Work In Practice.

Half Precision Results



Precision is important.
Introduces unacceptable banding artefacts.

Slower!

Extra 200 microseconds to execute. Slight drop in register use, but not enough to make any difference to occupancy.

Compiler generated longer program. Conversion instructions were a minority of the extra code.

Total Internal Reflection

Phenomena where all incident light is reflected at interface boundary, rather than being transmitted.

Not simulated in the transfer matrix model. Some proportion of energy is lost over several layers due to this!

Correction

Randrianandrasana et al. and Belcour correct for lost energy by pre-integrating an approximate correction term:

$$\mathrm{TIR} = \int_{\Omega} (1 - F(\omega_t)) D(\omega_t, \alpha_{23}) d\omega_i \tag{10}$$

Integral of Fresnel transmission with respect to distribution of normals, over the hemisphere of view angles.

Profiling

Profiling showed significant time could be saved without this LUT:

32bit TIR LUT	16 bit TIR LUT	Dummy TIR constant	No TIR
1.433ms	1.435ms	0.985ms	0.944ms

Table 1: Average shader invocation times for the 2-flux reference scene with various TIR corrections.

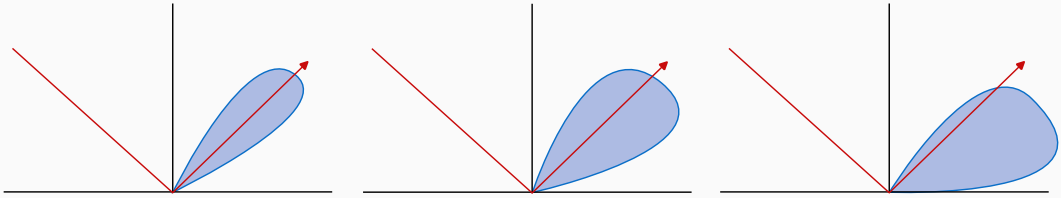
Naive Estimator

Belcour provide a generator for the original decorrelated LUT, as well as a naive estimator.

Naive estimator has a mean squared error of 0.464. Convenient as only dependent on vNDF.

Approximating the Naive Estimator

Need some way of modelling the lobe shift over roughness.



(a) Low Roughness (b) Medium Roughness (c) High Roughness

Figure 5: The blue shaded region indicates the approximate distribution of reflected rays.

Lagarde and De Rousiers `off_specular` function does exactly this!

Naive Analytic TIR

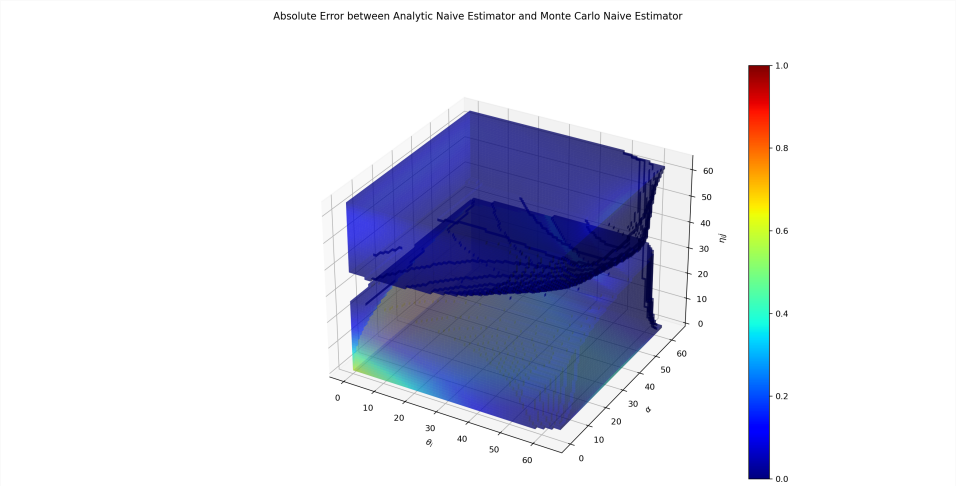
Final Naive estimator:

$$\begin{aligned} \text{TIR} &= (1 - F(\omega_o, \eta_{10}))F(\omega_o \cdot \omega_\alpha, \eta_{12})G_1(\omega_s, \alpha_{23}) \\ \omega_\alpha &= \textsc{OffSpecular}(H, \omega_o, H \cdot \omega_i, \alpha_{23}) \\ \omega_s &= -\textsc{Reflect}(\omega_o, \omega_\alpha) \\ H &= \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{11}$$

η_{01} and η_{12} refer to the IORs of medium above and below the current interface.

Error

Mean squared error of 0.0764, maximum absolute error of 50%.



Profiling

Increases runtime by 200 microseconds!

Fresnel term is expensive to evaluate. Additional 2 evaluations brings count to nearly 6 times per layer.

Most shading models do at most 1.

Discontinued further refinement of the approximation because of this.

Overall

Not really many good tools to combat the size, and thus performance problems.

Interdependence of features such as transmission and scattering makes it difficult to simplify individual pieces without global effects.

This is inherent to the transfer matrix design!

Limitations

Limitations

Implementation is not spatially varying. Potentially hides slowdown due to thread divergence.

static layer stacks introduce overhead that may not reflect realistic production contexts.

Sample size of one (1) makes difficult to generalise benchmarks to large variety of consumer graphics hardware on market.

Future Work

Static analysis for layer simplification such as what's employed by *Substrate*. Pre-compilation step that allows max-layer array to scale to need.

Fit a better parameterisation. Complex IOR is unintuitive, even incorrect for tristimulus renderer (Hoffman, 2019). Could additionally allow for a simpler, faster Fresnel form to be used.

Incorporate diffuse lobes to bring to parity with single-layer models. de Dinechin and Belcour's work in this area is promising for fitting to the transfer-matrix model.

Conclusion

Transfer Matrix approach is largely inappropriate for real-time renderers.

Size is unmanageable, and difficult to scale to specific needs.

Questions?

References

Belcour, L. (2018, jul). Efficient rendering of layered materials using an atomic decomposition with statistical operators. *ACM Trans. Graph.*, *37*(4). Retrieved from https://doi.org/10.1145/3197517.3201289 doi: 10.1145/3197517.3201289

Belcour, L., Bati, M., & Barla, P. (2020, August). *Bringing an Accurate Fresnel to Real-Time Rendering: a Preintegrable Decomposition.* SIGGRAPH'20 Courses. Retrieved from https://inria.hal.science/hal-02883680 doi: 10.1145/3388767.3407325

de Dinechin, H., & Belcour, L. (2022, May). Rendering layered materials with diffuse interfaces. *Proc. ACM Comput. Graph. Interact. Tech.*, *5*(1). Retrieved from https://doi.org/10.1145/3522620 doi: 10.1145/3522620

References ii

Heitz, E. (2014, June 30). Understanding the masking-shadowing function in microfacet-based brdfs. *Journal of Computer Graphics Techniques (JCGT)*, *3*(2), 48–107. Retrieved from http://jcgt.org/published/0003/02/03/

Hoffman, N. (2019). Fresnel Equations Considered Harmful . In R. Klein & H. Rushmeier (Eds.), *Workshop on material appearance modeling*. The Eurographics Association. doi: 10.2312/mam.20191305

Karis, B. (2013). *Real shading in unreal engine 4* (Tech. Rep.). Epic Games. Retrieved 2016-02-05, from http://blog.selfshadow.com/publications/s2013-shading-course/karis/s2013_pbs_epic_notes_v2.pdf

References iii

Lagarde, S., & De Rousiers, C. (2014). Moving frostbite to physically based rendering 3.0. In *Proceedings of physically based shading in theory and practice, siggraph 2014 course notes*. Retrieved from https://seblagarde.wordpress.com/wp-content/uploads/2015/07/course_notes_moving_frostbite_to_pbr_v32.pdf

Pharr, M., & Humphreys, G. (2023). *Physically based rendering, fourth edition: From theory to implementation* (4th ed.). San Francisco, CA, USA: MIT Press.

Randrianandrasana, J., Callet, P., & Lucas, L. (2021, jul). Transfer matrix based layered materials rendering. *ACM Trans. Graph.*, *40*(4). Retrieved from https://doi.org/10.1145/3450626.3459859 doi: 10.1145/3450626.3459859