

“Real-Time Polygonal-Light Shading with Linearly Transformed Cosines”の実装報告

森重 伸也^{†1,a)}

概要：筆者は、ACM SIGGRAPH 2016 で Eric Heitz らが発表した技術論文 “Real-Time Polygonal-Light Shading with Linearly Transformed Cosines” を Unreal Engine 4^{b)}上 に実装し、リアルタイムゲームへの導入を考慮して、追実験を行っている。この技術論文の目的は、エリアライトなど、面積を持つポリゴン形状の光源と BRDF で、リアルタイムに物理ベースの照明を実現する、というものだ。本来、面光源の照明には、球面方程式を解く必要があり、リアルタイムの照明は困難だった。この技術論文では、リアルタイムで実現するために、オリジナルの光源分布から事前積分で導出した少数パラメタの Linearly Transformed Cosine (LTC) を導入することで、面積を持った光源でのリアルタイム照明を実現した。

1. Real-Time Polygonal-Light Shading

近年、リアルタイムゲーム分野では、物理ベースレンダリングが導入されており、職人的な技術に頼らずとも、誰でも手軽に自然な照明と質感を表現できるようになってきた。その際、レンダリング方程式にもとづいた照明と材質の相互作用計算が生じる。具体的には、半球空間での BRDF と照明の積を積分することだ。この積分は、従来のゲームで扱っていた点光源で計算しても、自然な照明にはならない。物理にもとづく現実世界では、点の光源は存在せず、全ての光源は、微小な面積を持つからだ[1]。つまり、面光源をリアルタイムに近似計算する手法が必要になってくる。具体的には、陰影付け (Shading) する点 r に対する面光源での照明は、Irradiance を計算すればよい。

本稿では、Heitz [2] らが提案した Linearly Transformed Cosines (LTCs) をもとに、実在するゲームエンジン、Unreal Engine 4 上で設計実装を検討して追実験している内容を提供する。

2. Linearly Transformed Cosines (LTCs)

半球空間で光の分布を表現する方法は、解析的な方法は Arvo の方法[3]があり、計算コストは $O(n*s)$ でリアルタイムでの計算は困難である。 n は頂点数で、 s はスペキュラ指数。自然な質感表現には、光の分布表現で異方性と歪みの表現が必須になる。そこで、欲しい光の分布を、余弦重み付けされた球面分布 (Clamped Cosine Distribution) を基底に用い、その球面上の方向ベクトルに対して、SRT など適当な 3×3 Matrix の線形変換を適用することで、欲しい分布を表現することを考える。

(1) 欲しい分布 D (線形変換後) と基底の分布 D_o の関係

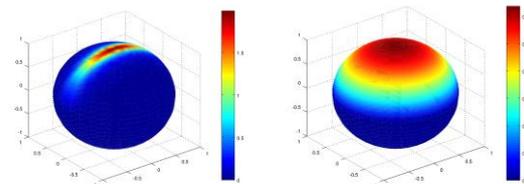


図 1 Clamped Cosine Distribution

図 1 は、GNU Octave を用いて分布を可視化したものである。左からそれぞれ、欲しい分布 D (ここでは異方性とする)、基底の分布 D_o (Clamped Cosine) である。分布の関係は、

$$D(\omega) = D_o(\omega_o)$$

$$\omega = M\omega_o$$

であり、図 1 の異方性への線形変換行列 M は、

$$M = \begin{bmatrix} 0.8 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

である。この場合に、欲しい分布 D は、

$$D(\omega) = D_o(\omega_o) \frac{\partial \omega_o}{\partial \omega}$$

で計算できる。 $\frac{\partial \omega_o}{\partial \omega}$ は、線形変換の Jacobian で、変換のスケール項だ。具体的には、 $\frac{\partial \omega_o}{\partial \omega} = \frac{|M^{-1}|}{\|M^{-1}\omega\|^3}$ で計算する。

3. LTCs を用いた面光源のリアルタイム照明

BRDF には、GGX を使い、4 頂点の平面から構成される、面光源のリアルタイム照明について説明する。

3.1 The GGX Microfacet BRDF

Unreal Engine 4 をはじめとしてゲーム分野では、Microfacet BRDF に GGX が良く使われる[5]。理由は、大きく 2 つある。1 つは、物体表面の粗さ (roughness) のみで、物理ベースの質感を表現できること。もう 1 つは、照明計算の数値積分に Importance Sampling を使い、その結果を 2 次元の LUT (Look Up Table) に落とし込めるからだ。

$$D \approx f(\omega_i, \omega_o) \cos \theta_i = \frac{D_{ggx}(\omega_h) G(\omega_i, \omega_o) F(\omega_i, \omega_o)}{4 \cos \theta_o}$$

F は、Fresnel であり、1 とする。

^{†1} エヌディーキューブ株式会社
NDCUBE Co., Ltd.

a) morishige@ndcube.co.jp

b) Unreal Engine 4 は、米国およびその他の地域における Epic Games, Inc の商標または登録商標です。

$$D_{ggx}(\omega_h) = \frac{\alpha^2}{\pi((\langle n, \omega_h \rangle)^2(\alpha^2 - 1) + 1)^2}$$

$$G(\omega_i, \omega_o) = G_1(\omega_i)G_1(\omega_o)$$

$$G_1(\omega) = \frac{\langle n, \omega \rangle}{(\langle n, \omega \rangle)(1 - k) + k}$$

3.2 LTCs を用いた陰影付け (Shading)

均一な面光源 L を構成する平面ポリゴン P を、球面上に射影して、球面上で、BRDF と照明の積を積分する。

$$I(\mathbf{r}) = \int_P L(\omega_i)D(\omega_i) d\omega_i,$$

この積分結果は、平面 P の Irradiance E(P) であり。

$$\int_P D(\omega)d\omega = \int_{P_o} D(\omega_o)d\omega_o = E(P_o)$$

で計算できる。平面だけでなく、任意形状の微小面の積分計算は、リアルタイムでは困難である。理由は、陰影付けしようとする点 r と面光源 L の間に、遮蔽物がある場合とない場合があり、点 r に応じて、Irradiance の計算結果が変わるからだ。

そこで、平面ポリゴンを構成する頂点集合 $P = \{p_1, p_2, p_3, p_4\}$ に対する、Irradiance E(P) 計算を、シェーディング位置 r を原点とする座標系で、r から見える頂点ベクトル集合 $U = \{u_1, u_2, u_3, u_4\}$ に対する Vector Irradiance E(U) 計算に置き換える[3]。見えない頂点は、Clipping する。

$$u_k = p_k - r$$

Irradiance は、Light Field [3] であり、Light Field は、Vector Fields だからだ。2 頂点ベクトル u_k, u_{k+1} が作るポリゴンエッジに対して、Irradiance を計算すればよい。Irradiance の値は、2 頂点ベクトルの角度 θ_k 、エッジの長さに比例して、拡散反射するので、

$$E(U) = \frac{1}{2\pi} \sum_{k=1}^4 \theta_k(u_k, u_{k+1}) \Gamma_k(u_k, u_{k+1})$$

で計算できる。k=5 の場合、k=1 の閉領域とする。

2 頂点ベクトルのなす角度は、

$$\theta_k = \cos^{-1}(\langle u_k, u_{k+1} \rangle)$$

拡散反射は、位置 r の法線との余弦重み付け

$$\Gamma_k = \langle u_k \times u_{k+1}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \rangle$$

とする。

4. 結果

Unreal Engine 4 (UE4) 上で、GGX BRDF と 4 頂点の平面ポリゴン光源の実装を行った。照明の分布を表す LTC パラメータは、FP16 テクスチャの RGBA 成分に事前書き込んでおき、リアルタイムシェーディング時に、LUT として参照する。

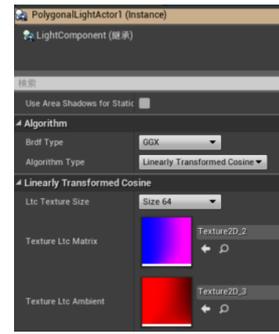


図 2 UE4 上で、LTC のパラメータをランタイムベイク

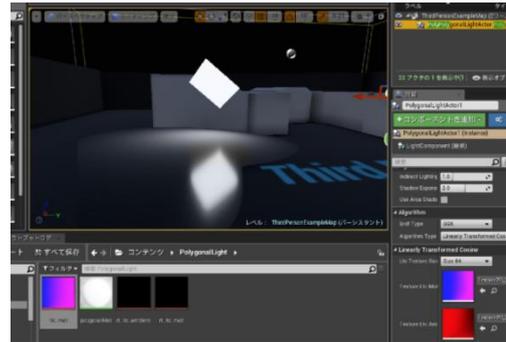


図 3 UE4 上での面光源

5. 今後の取り組み

2 点ある。1 つは、頂点数が多くなった場合の頂点ベクトルの Clipping 高速化だ。シェーディング時に、照明計算に寄与する頂点ベクトルだけを、高速に取得する。もう 1 つは、LTC に対して、線形や畳込みの演算オペレーターを定義して、IBL や、アーティストが意図したキャラクタ IBL の生成に活用することだ。

今回の実機デモやソースコードは、2016 年 9 月 21 日開催の研究会発表後に、Github にて公開して、引き続き開発を進めていく予定である。要望や意見があれば、是非 morishige@ndcube.co.jp までお寄せいただきたい。

参考文献

- [1] [Hill] HILL, S., MCAULEY, S., BURLEY, B., CHAN, D., FASCIONE, L., IWANICKI, M., HOFFMAN, N., JAKOB, W., NEUBELT, D., PESCE, A., AND PETTINEO, M. 2015. Physically based shading in theory and practice. In ACM SIGGRAPH Courses 2015.
- [2] [Heitz] Eric Heitz, "Real-Time Polygonal-Light Shading with Linearly Transformed Cosines". <https://eheimresearch.wordpress.com/415-2/>
- [3] [Arvo94] Arvo, James, "The Irradiance Jacobian for Partially Occluded Polyhedral Sources," Siggraph 94, July 1994, 343-350.
- [4] [Walter] WALTER, B., MARSCHNER, S. R., LI, H., AND TORRANCE, K. E. 2007. Microfacet models for refraction through rough surfaces. In Proc. Eurographics Symposium on Rendering, 195-206.
- [5] [John] John Hable, "Optimizing GGX Shaders with dot(L,H)." FilmicWorld. <http://www.filmicworlds.com/2014/04/21/optimizing-ggx-shaders-with-dotlh/>