3F−03 天空光に照射された3次元物体のグラフィックスハードウェアを利用した高速表示*

高田 信太郎† 土橋 宜典† 山本 強†
北海道大学工学部‡

1. はじめに

近年、様々な屋外景観画像がコンピュータグラフィックスを用いて作成されている。リアルな屋外景観画像を作成するためには、太陽光だけではなく天空光によって照射された物体の輝度計算を行う必要がある。しかし、天空光による照度計算は非常に多くの計算時間を必要とする点が問題となっている。

一方、近年はグラフィックスハードウェアの高性能化が進んでおり、これを利用したリアルな画像生成手法の開発が多く行われている。本稿では、グラフィックスハードウェアを利用することで高速に天空光による照度を計算する手法を提案する。提案手法を用いれば天空光によって照射された物体の輝度だけではなく、天空の一部が物体によって遮られることにより生じる半影も考慮した画像を高速に作成できる。

2. 天空光による照度計算のモデル

天空光は地上を覆う極めて大きな半径を持った半球状の光源と考えることができる[1]。ただし、半球状光源の輝度分布は一様ではない。図1に示すように、天空を小さな領域（天空要素と呼ぶ）に分割する。それぞれの天空要素は一定の輝度とする。光の逆二乗則を適用すると、天空要素 P_e による計算点 P の照度 dE は次式で計算できる。

$$dE = H(\theta,\phi)L(\theta,\phi)\frac{\cos\theta}{r^2}dA \qquad (1)$$

ここで、L は天空要素の輝度、θ は天頂方向と線分 PP_e とのなす角、ϕ はX軸から天空要素までの方位角、r は P と P_e 間の距離、dA は天空要素の面積である。また、$H(\theta,\phi)$ は影の有無を表す関数であり、点 P から光源が可視であれば1を返し、そうでなければ0を返す。

天空要素の面積 dA は $dA = (rd\theta)\times(r\sin\theta d\phi)$ で表せる。よって半球状光源による点 P の照度は θ と ϕ で積分することにより、

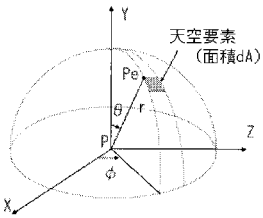

図1: 天空要素の位置関係

$$E = \int_\theta\int_\phi H(\theta,\phi)L(\theta,\phi)\cos\theta\sin\theta d\theta d\phi \qquad (2)$$

となる。以降はこの E のグラフィックスハードウェアを利用した計算方法について述べる。

3. グラフィックスハードウェアを用いた物体の表示

3.1 レンダリング

式(2)で表される照度 E を解析的に計算するのは困難であるため、数値積分によって計算する。提案手法では、この数値積分を実行するために、それぞれの天空要素を一つの光源で近似しそれらの総和で E を求める。すなわち、点 P の照度 E は次式で計算される。

$$E = \sum_i\sum_j H(\theta_i,\phi_j)L(\theta_i,\phi_j)\cos\theta_i\sin\theta_i\Delta\theta\Delta\phi \qquad (3)$$

ここで、$\Delta\theta$、$\Delta\phi$ は積分間隔を表す。式(3)から、計算点の照度を求めるためには、影の有無を表す H の値を求める必要がある。この値は各光源からの光により物体が落とす影を計算することで求めることができる。これは、グラフィックスハードウェアを用いたシャドウマッピングが利用できる[2]。この方法では、リアルタイムで影を計算することが可能であるため、任意

*A Fast Method for Rendering Three-Dimensional Objects Illuminated by Skylight on Graphics Hardware
†Shintaro Takada, Yoshinori Dobashi, Tsuyoshi Yamamoto
‡Faculty of Engineering, Hokkaido University

の点での H の値を高速に求めることができる。また、L の値は経験式が定められている [1]。

以上のことから各天空要素による照度は計算可能である。よって、それぞれの天空要素に光源を置いて画像を描画し、アルファブレンディングの機能によりその色をフレームバッファに足しこむことで、式 (3) の総和の計算を行うことが可能となる。

3.2 アキュムレーションバッファを用いたアンチエイリアシング

前節で提案した手法により天空光に照射された物体の表示は可能になるが、天空要素への分割数が少ない場合は物体の影を十分にサンプリングできない。これを解決する単純な方法は分割数を増加することである。しかし、単純に分割数を増加すると量子化誤差により画質が低下してしまう。多くのグラフィックスハードウェアの性質上、各画素値は8ビットに量子化されてしまうため、分割数に比例して量子化誤差が蓄積されて画質が低下してしまう。

解決策として、アキュムレーションバッファを利用することが挙げられる。アキュムレーションバッファは標準のカラーバッファよりもビット数が多く（一般に 16 ビット）、量子化誤差を抑制することができる。しかし、一般にアキュムレーションバッファを利用すると著しくレンダリング時間が増加してしまう。そこで参考文献 [3] の手法を応用する。すなわち、照度の値を 16 ビットで表現し、上位 8 ビットと下位 8 ビットに分解して画像を描画し、その後にアキュムレーションバッファを用いてそれぞれの画像を足しこむ。

式 (3) の $HL\cos\theta\sin\theta\Delta\theta\Delta\phi$ を $f(\theta,\phi)$ とし、上位 8 ビットと下位 8 ビットに分解すると、照度 E は次式のように計算できる。

$$E=\sum_i\sum_j\left\{Int\left(\frac{f(\theta_i,\phi_j)}{2^8}\right)\times 2^8+R(\theta_i,\phi_j)\right\} \quad (4)$$

ここで、Int は整数化することを示し、R は $f(\theta,\phi)$ を 2^8 で割った余りである。まず第一項の Int の項（上位 8 ビット）の総和を計算し、その値を 2^8 倍してアキュムレーションバッファに足しこむ。次に第二項（下位 8 ビット）の総和を計算して、その値をアキュムレーションバッファに足しこむ。しかし、この方法を用いても各項の総和は 8 ビットの精度であるフレームバッファを利用するため、量子化誤差が蓄積され十分な精度が得られない場合が生じる。そこで、8 ビットでも十分精度よく計算できるようさらに細かく分解する。そして、同様の操作を繰り返し画像を生成する。この方法を用いると、一つ一つの画像をアキュムレーションバッファを利用して加算するより高速で、かつ量子化誤差による画質の低下を抑えることが可能になる。

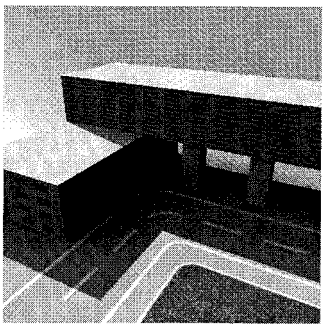

図 2: 作成した画像

4. 適用例

簡単な例を用いて、提案手法の検討を行った。図 2 は提案手法で作成した画像である。積分間隔は $\Delta\theta$、$\Delta\phi$ ともに 5 度に設定し、照度の値は 16 ビットを 4 つに分解した。一つ一つの画像をアキュムレーションバッファを用いて足しこむ方法と、提案手法との比較を行った。計算時間は前者で 343 秒、後者で 15.1 秒であった。差分画像を求め、その平均輝度を求めたところ 5.4 であった。計算機は CPU が Athlon(900MHz) であり、グラフィックスハードウェアとして NVIDIA 社の GeForce3 を搭載した PC を用いた。画像サイズは 512 × 512 である。このことから、提案手法は速度の面で非常に有用である。

5. まとめ

グラフィックスハードウェアを利用して、天空光に照射された 3 次元物体を高速に表示する手法を提案した。提案手法は以下の特徴を持つ。

- 天空の一部が物体によって遮られることにより生じる半影を表示することができる。
- ビット分解をすることで単純にアキュムレーションバッファを用いるよりも高速に画像を作成できる。

今後の課題として、画質の向上が必要である。また、建物のガラス面などを表示するための鏡面反射を扱えるようにすることが挙げられる。

参考文献

1. T.Nishita, E.Nakamae, "Continuous Tone Representation of Three-Dimensional Objects Illuminated by Sky Light", Proc.SIGGRAPH'86, pp.125-132 (August 1986)
2. "Shadow Mapping with Today's OpenGL Hardware", CEDEC 2001
3. J.Cohen, C.Tchou, T.Hawkins, P.Debevec, "Real-Time High-Dynamic Range Texture Mapping", Eurographics Rendering Workshop 2001, London, England, June 2001