Vol.2015-CG-160 No.6
2015/8/29

確率的な出力サブサンプル数制御に基づく描画順序非依存な半透明描画

今給黎 隆[1,a)]

概要：ゲームなどのリアルタイムタイムアプリケーションにおいて，半透明物体の描画はいまだ難しい問題である．半透明の物体を描画した結果は，その後ろに描かれている物体の描画結果に影響を受けるため，描画するフラグメントについて，視点から遠い順番に描画して正しい結果を得る必要がある．しかしながら全フラグメントを視点からの距離でソートして描画することは処理負荷が高い．我々は，マルチサンプリング手法において，書き込むサブサンプル数を不透明度に応じてフラグメント毎にランダムに変化させることで，描画順序に依存しない半透明描画を実現する手法を提案する．本手法は，フラグメント毎の追加のメモリや事前計算を必要とせず，広く用いられている 3D API で実装可能で高速なアルゴリズムである．

1. はじめに

ゲームなどのリアルタイムアプリケーションにおいて，半透明物体の描画は特定の手法が定まっていない分野である．半透明の物体の描画結果は，その後ろに描かれる物体の影響を受けるため，描画しようとする全フラグメントについて，視点から遠い順番に描画する必要がある．しかしながら全フラグメントをソートして描画することは簡単ではない．GPU を効率的に使うことで高速化された Per-Pixel Linked Lists [24] を利用したソートによる正確な半透明物体の描画手法は存在するが，描画するフラグメントを保存するための多くのメモリが必要であり，メモリの書き込みについても他の画素の情報を考慮する必要があるため処理のストールが発生しやすいなど，実用的な観点からリアルタイムアプリケーションでの採用は難しい．Stochastic Transparency [6] は，マルチサンプリング手法において書き込むサブサンプルパターンをフラグメント毎に変更することで描画順序に依存しない半透明描画を実現するが，フラグメント毎のサンプリングパターンの制御は，広く用いられている 3D API の標準的な機能では難しい．

我々は，書き込むサブサンプル数をフラグメント毎にランダムに制御することで，描画順序に依存しない半透明描画手法を実現する．我々の手法は，フラグメントの記録に関する追加のメモリや事前計算を必要とせず，広く用いられている 3D API で容易に実装可能なアルゴリズムである．

[1] 株式会社セガゲームス
SEGA Games Co., Ltd.
[a)] Imagire_Takashi@sega.co.jp

2. 関連研究

半透明のオブジェクトの描画に関する研究は，古くから行われている．Porter ら [19] は，半透明物体を描画した結果は，その背景の物体の色に依存することから，それぞれの半透明物体について，背景にあるオブジェクトを事前に描画することで正確な画像を作成する手法を提案した．しかしながら，物体の描画を奥行きの順に効率よく描画することは容易ではない．

Everitt [7] は，複数回の描画によって半透明物体を正確に描画するデプスピーリングと呼ばれる手法を提案した．デプスピーリングは，シーン全体をレンダリングして描画されていない最奥の深度値を抽出し，その深度値にあるフラグメントを描画する手順を繰り返すことで，物体を奥から順番に描画していく．半透明物体が重なる数に比例した描画回数を必要とするため，デプスピーリングは，レンダリングの回数が多く，処理負荷は高い．デプスピーリングを高速化する手法 [2], [13] も提案されているが，複数回の描画が必要であり，リアルタイムレンダリングには不十分である．

物体を複数回描画するのではなく，描画する全フラグメントを記録しておき，後にフラグメントを画素ごとに奥行きでソートして，正しい半透明描画を実現する方法も存在する．Per-Pixel Linked List (PPLL) [24] は，画素ごとのリストを構築することで，この方法を実現するためのデータ構造である．PPLL は，リスト構造の構築にランダムなメモリアクセスを必要とするために，処理負荷が高い．GPU を用いた並列描画で効率的に実行するアルゴリ

ズム [10], [16] も提案されているが，メモリの最大使用量の見積もりの難しさや処理負荷の高さから，リアルタイムアプリケーションでの採用は難しい．

ランダムなメモリアクセスを行うことなくフラグメントをソートする技術も存在する．A-buffer [5] は，1 画素内にあらかじめ決められたサイズの多くの情報を描画時に保存しておき，後から合成する方法である．A-buffer として，記録された複数のサブサンプルの情報のうち，透明度に応じた割合で画素数を書き込むことで，半透明を含む物体を描画できる．また，Jouppi ら [8] は，フラグメントの深度情報と深度勾配を持つことで，サブサンプルの情報を圧縮して格納する手法を提案した．しかしながら，処理方法が複雑であり，リアルタイムアプリケーションに用いることは難しい．Callahan ら [4] や Bavoil ら [3] は，k 枚の画面と等しいサイズのバッファを用意し，k 枚の重なりまでの半透明を正確に描画する k-buffer の手法を提案した．k-buffer による半透明描画に関しては，数多くの高速化の研究 [9], [14], [23], [25] や，品質を改善するための研究 [15], [20], [21] が行われているものの，リアルタイムアプリケーションにおいて，十分な枚数での半透明描画を行うことは難しい．

負荷が高い正確な描画ではなく，近似的にでも高速に半透明を描画する方法も提案されている．Akeley [1] は，各画素の描画として，画素よりも小さなサブサンプルの情報を記録し，最終的に出力する画像において，サブサンプルの情報を統合することで，高品質な画像を生成する手法を提案した．提案手法において，物体の不透明度に比例して書き出すサブサンプルの数を抑制することで，描画順序に依存しない半透明描画手法 (アルファトゥカバレッジ) が実現される．しかしながら，アルファトゥカバレッジは，表現できる不透明度がサンプリング数と同じ段階しか持てず，同じ半透明度を持つ物体を描画する時に，片方の物体の描画結果しか影響しない等，描画結果を期待される結果の違いが判別しやすいケースが存在する．Meshkin [18] は，順序依存項を取り込まないことで，順序非依存の半透明描画を実現した．しかしながら，半透明物体が多く重なると，その誤差は大きくなる．Stochastic Transparency [6] やその改善手法 [12] は，サブサンプルをフラグメントごとに変えることで，順序非依存な半透明をリアルタイムに実現する．しかしながら，これらの手法は，GPU の拡張機能を必要とし，多くの GPU や 3D API で必ずしも実現可能ではない．McGuire ら [17] は，遠方の物体の寄与を正確に見積もることで，より正確な半透明描画の近似手法を提案した．しかしながら，半透明物体が重なる際の色の描画依存性を考慮していないため，半透明物体が重なる際に，本来の結果からの差が大きくなる場合が存在する．

3. 提案アルゴリズム

我々は，半透明を描画するためにアルファトゥカバレッジ [1] によるマルチサンプリングを利用する．アルファトゥカバレッジに用いられるサブサンプルは，不透明度に応じて，書き込まれる数やパターンが自動的に決定される．提案法では，フラグメント毎にアルファ値を変化させることで書き込むサブサンプル数の制御を行う．通常のアルファトゥカバレッジでは，同じ不透明度の場合には一方のフラグメントの情報が欠落するが，提案手法ではフラグメントごとにアルファ値が異なるため，この問題は生じない．

アルファ値の決定に関して，完全に透明な状態ではサブサンプルに書き込む必要はなく，逆に完全に不透明な状態ではサブサンプルは全て書き込む必要がある．このことから，不透明度に応じて 0 から 1 へ変化する単調増加関数として，塗りつぶすサブサンプルの割合，すなわちアルファ値を決定する．提案法では，塗りつぶすサブサンプルを変化させるために，$[0, 1]$ の一様乱数 f を導入し，単調増加関数として指数関数を使用する．つまり，アルファトゥカバレッジのアルファ値として f^γ の値を出力する．

指数 γ は，描画したフラグメントの平均をした結果が本来の不透明度 α と等しくなる値という条件により求められる．この条件は，乱数を一様乱数とすると，f^γ の確率平均が不透明度 α となること等しい．したがって，γ は，下記の式を満たす．

$$\alpha = \int_0^1 f^\gamma df.$$

この式を解くことにより，$\gamma = 1/\alpha - 1$ が導出される．

乱数 f には画素の位置と，描画するフラグメントの奥行き値を用いる．乱数のために画面解像度と等しい大きさの 2 枚のバッファ (乱数テクスチャ) を用意する．乱数テクスチャは，他方のテクスチャを乱数のシード値として，フレーム毎にピンポン形式で片方ずつ更新することで，毎フレーム異なる画素ごとの乱数を構築する．最終的な乱数は，フラグメントのスクリーンにおける位置での乱数テクスチャの参照と，描画するフラグメントの奥行き値を乱数のシード値として，フラグメント計算時に疑似乱数を生成することで，フラグメントごとに異なる乱数を実現する．

また，Temporal Anti-Aliasing (TAA) [11] と Bilateral フィルタ [22] によるポストフィルタリングを導入することで，乱数に伴うアーティファクトを軽減する．まず，前フレームの結果と描画する色が近い場合に前フレームの色との合成を行う．次に周辺の画素との色差の違いに応じた重みで平滑化を行う．これらポストフィルタリングにより，乱数の分布に伴うごま塩ノイズが発生するものが，時間及び空間的な平均により確率を平均化する効果が付与されるため，本来期待される結果と近い描画結果が生成される．

4. 結果

検証として，赤，緑，青，白の不透明度 0.6，0.6，0.6，0.2 の各板を左上，右上，左下，右下に半分ずらして，手前から奥に配置したシーンを，各技法で手前から奥の順序で描画した．マルチサンプリングを用いる場合には，8 サブサンプルのマルチサンプリングで計算を行っている．図 1 は，描画順序の問題を解消した場合に期待される結果である．通常の手法で描画した場合には，深度テストを有効にするか，しないかに応じて 図 2 及び，図 3 の結果が得られる．重なりが生じている部分で，一方の色しか反映されていない場合や，正しくない色が描画されている．図 4 は，アルファトゥカバレッジを用いたレンダリング結果である．不透明度が等しい赤と緑のポリゴンに関して，後から描かれた緑色の板により赤色の情報が消されている．図 5 は，提案手法による結果である．ノイズが生じているが，描画結果は平均として期待される結果と等しい．図 6 は，提案手法においてポストフィルタリングを行わない場合の結果である．ごま塩状のノイズが粗く観察される．図 7 は，マルチサンプリングを行わない場合の結果である．フラグメントごとに乱数が変わるため，ノイズが粗くはなるが，期待される結果と近い色味が再現される．図 8 と図 9 は，Meshkin [18] 及び，McGuire ら [17] による手法による結果である．共に，順序依存項が省かれているために，特に多く重なり合う中心の領域の色は不正確である．

図 1 望まれる結果

図 2 深度テストあり

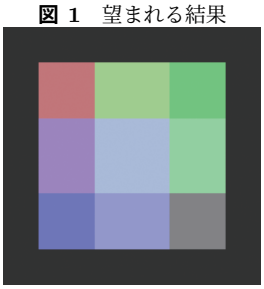

図 3 深度テストなし

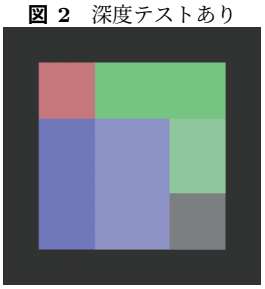

図 4 アルファトゥカバレッジ

図 5 提案法

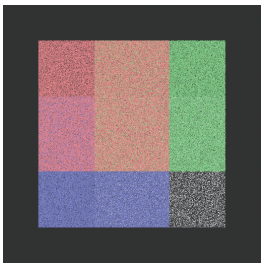

図 6 ポストフィルタリングなし

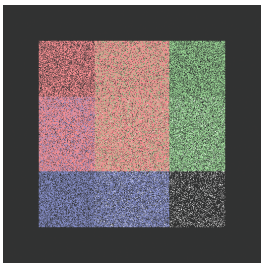

図 7 MSAA なし

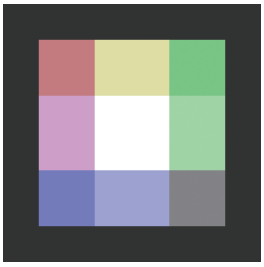

図 8 Meshkin

図 9 McGuire ら

5. まとめと今後の課題

アルファトゥカバレッジにおけるアルファ値をランダムに変化させることで，順序に依存しない新しい半透明描画手法を提案した．今回の手法は，モバイルフォンなど，性能が比較的低いデバイスに対しても適応可能な柔軟性の高い手法である．

本手法はごま塩状のノイズが発生する．時間的及び空間的な平滑化によりノイズを低減させたが，完全には消去はできていない．また，書き込むサブサンプルは，画素ごとのアルファ値でのみ制御されるため，効率的なサブサンプルの指定を行うことにより，色の誤差をより少なくする手法が望まれる．

参考文献

[1] Akeley, K.: Reality Engine Graphics, *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pp. 109–116 (1993).
[2] Bavoil, L. and Myers, K.: Order independent transparency with dual depth peeling, Technical report, NVIDIA Developer SDK 10 (2008).
[3] Bavoil, L., Callahan, S. P., Lefohn, A., Comba, J. a. L. D. and Silva, C. T.: Multi-fragment Effects on the GPU Using the K-buffer, *Proceedings of the 2007 Sym-*

posium on Interactive 3D Graphics and Games, I3D '07, pp. 97–104 (2007).

[4] Callahan, S. P., Ikits, M., Comba, J. L. D. and Silva, C. T.: Hardware-Assisted Visibility Sorting for Unstructured Volume Rendering, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 11, No. 3 (2005).

[5] Carpenter, L.: The A -buffer, an Antialiased Hidden Surface Method, *SIGGRAPH Comput. Graph.*, Vol. 18, No. 3 (1984).

[6] Enderton, E., Sintorn, E., Shirley, P. and Luebke, D.: Stochastic Transparency, *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pp. 157–164 (2010).

[7] Everitt, C.: Interactive Order-Independent Transparency (2001).

[8] Jouppi, N. P. and Chang, C.-F.: Z3: An Economical Hardware Technique for High-quality Antialiasing and Transparency, *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, HWWS '99, pp. 85–93 (1999).

[9] Kerzner, E., Wyman, C., Butler, L. and Gribble, C.: Toward Efficient and Accurate Order-independent Transparency, *ACM SIGGRAPH 2013 Posters*, SIGGRAPH '13, pp. 109:1–109:1 (2013).

[10] Knowles, P., Leach, G. and Zambetta, F.: Fast Sorting for Exact OIT of Complex Scenes, *Vis. Comput.*, Vol. 30, No. 6-8 (2014).

[11] Korein, J. and Badler, N.: Temporal Anti-aliasing in Computer Generated Animation, *SIGGRAPH Comput. Graph.*, Vol. 17, No. 3, pp. 377–388 (1983).

[12] Laine, S. and Karras, T.: Stratified Sampling for Stochastic Transparency, *Proceedings of the Twenty-second Eurographics Conference on Rendering*, EGSR '11, pp. 1197–1204 (2011).

[13] Liu, F., Huang, M.-C., Liu, X.-H. and Wu, E.-H.: Bucket Depth Peeling, *SIGGRAPH 2009: Talks*, SIGGRAPH '09, pp. 50:1–50:1 (2009).

[14] Liu, F., Huang, M.-C., Liu, X.-H. and Wu, E.-H.: FreePipe: A Programmable Parallel Rendering Architecture for Efficient Multi-fragment Effects, *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '10, pp. 75–82 (2010).

[15] Maule, M., Comba, J. a., Torchelsen, R. and Bastos, R.: Hybrid Transparency, *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, pp. 103–118 (2013).

[16] Maule, M., Comba, J. L. D., Torchelsen, R. and Bastos, R.: Memory-Efficient Order-Independent Transparency with Dynamic Fragment Buffer, *Proceedings of the 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images*, SIBGRAPI '12, pp. 134–141 (2012).

[17] McGuire, M. and Bavoil, L.: Weighted Blended Order-Independent Transparency, *Journal of Computer Graphics Techniques (JCGT)*, Vol. 2, No. 2, pp. 122–141 (2013).

[18] Meshkin, H.: Sort-Independent Alpha Blending, GDC2007 (2007).

[19] Porter, T. and Duff, T.: Seminal Graphics, chapter Compositing Digital Images, pp. 355–361 (1998).

[20] Salvi, M., Montgomery, J. and Lefohn, A.: Adaptive Order Independent Transparency: A Fast and Practical Approach to Rendering Transparent Geometry, Game Developers Conference 2011 (2011).

[21] Salvi, M. and Vaidyanathan, K.: Multi-layer Alpha Blending, *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, pp. 151–158 (2014).

[22] Tomasi, C. and Manduchi, R.: Bilateral Filtering for Gray and Color Images, *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pp. 839– (1998).

[23] Vasilakis, A. A. and Fudos, I.: K+-buffer: Fragment Synchronized K-buffer, *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, pp. 143–150 (2014).

[24] Yang, J. C., Hensley, J., Grün, H. and Thibieroz, N.: Real-time Concurrent Linked List Construction on the GPU, *Proceedings of the 21st Eurographics Conference on Rendering*, EGSR'10, pp. 1297–1304 (2010).

[25] Zhang, N.: Memory-Hazard-Aware K-Buffer Algorithm for Order-Independent Transparency Rendering, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 20, No. 2, pp. 238–248 (2014).