グラフィクスとCAD　109－6
（２００２．１１．２２）

事前領域特定手法を用いた環境マッピング

小池 崇文*
E-mail: koike@sdl.hitachi.co.jp

概 要

近年、グラフィックスカードの高性能・高機能化により、環境マッピングを用いることによって鏡面反射効果のみならず、各種照明効果を施した画像を実時間でレンダリングできるようになっている。また、Chromium 等の実時間分散レンダリング技術を利用することで、容易に並列ディスプレイ環境を構築する事が出来るようになった。こうした状況の下で高画質画像をレンダリングするには、高解像度でかつ複数の環境マップが必要とされ、環境マップ生成負荷が大きくなると予想される。そこで、本研究では、最終レンダリングに必要な環境マップ領域を事前に求めることにより必要部分のみの環境マップを生成し、環境マップ生成に関する計算量を削減する手法を開発した。

Environment Mapping with Priorly Identifying Region

Takafumi Koike†
E-mail: koike@sdl.hitachi.co.jp

abstract

In recent years, we can render images with reflection effect and lighting models in real-time because graphics cards enhance its speed and functions. And we can easily construct parallel display environments to use with real-time distributed rendering technology for example Chromium. In this situation, to render high quality images, we need high quality and many environment maps, and it is estimated that a load of generating environment maps is increasing. So we present a new environment mapping method which reduces computational quantity with priorly identifying environment regions to render.

*日立製作所システム開発研究所
†Hitachi, Ltd., Systems Development Laboratory

1 はじめに

近年の VR(Virtual Reality) 研究の進展により、大画面、高解像度の特徴を持つ高臨場感ディスプレイや VR 環境を構築することが増えてきた。また、コンピュータとグラフィックスカードの高性能化・低価格化により、高価なワークステーションではなく、PC を利用して高臨場感ディスプレイを構築するようになりつつある。1 台の PC やディスプレイでは解像度に限界があり、必然的に複数台の構成が必要となり、それにともなって、分散レンダリング技術が必要となる。

PC を利用した環境は、比較的低コストで構築できるが、PC 間の同期処理や、通信速度の制限等、新たな問題を抱えることとなる。そこで、Chromium のような一般的な手法を用いるが、VR 用途の場合、遅延やフレームレートが臨場感に対して大きな要因をしめている。そのため、多少特殊技術を利用してもより高フレームレートが達成できる手法を開発する必要がある。筆者の研究グループでは、高臨場感ディスプレイシステムの開発と研究の中で、特にその必要性を感じてきた。

並列分散レンダリング技術は、Sort Level[4] で分類されるが、構築可能な環境を柔軟にするために、Sort Level によらないアルゴリズムを開発する必要がある。本論文では、今日の CG 技術に不可欠な環境マッピングの効率化に注目した。それには、環境マップ生成自体の手法を効率化する必要がある。

これまで、並列分散レンダリング環境で、環境マッピングを利用するには、通常、環境マップをネットワークで通信するか、各映像 PC が独自に環境マップ作成する必要があった。本来、各 PC は自分のレンダリング領域のみを描画すればよく、環境マップも本来自分たちが使用する部分についてのみ生成すればよいはずである。

本論文ではこうした分散レンダリング環境で効果がある、環境マッピングを紹介する。本手法は、こうした実時間分散レンダリング環境と相性が良く、通信量を削減する効果をもつ。本手法は、現状のハードウェアアーキテクチャでも、十分効果があるが、本手法により適したハードウェアアーキテクチャの提案も行った。そして最後に、ネットワーク接続された高臨場感ディスプレイシステムを構築し、その環境の紹介と、通信・計算時間の評価を行う。

2 研究の背景

Blinn と Newell によって開発された環境マッピング [2] は、改良が重ねられ、球面 [7] や、双対放物面 [8] を用いた環境マッピングが開発された。また、専用回路を必要とせず、球面環境マッピングより歪みが少ない立方体環境マッピング [6] も開発された。立方体環境マッピングは、Xbox*、NINTENDO GAMECUBE†といったゲーム機で容易に利用でき、鏡面反射効果だけでなく、透過効果や、Non-Photorealistic Rendering[5] で使用されていて、現在の 3 次元 CG には欠かせない技術である。

一方で、実時間分散レンダリングについても、その研究がすすんでおり、OpenGL‡をベースとし、プログラムの書き換えが発生しない WireGL[9] や Chromium[10] といった技術が開発されている。

また、分散レンダリングした画像を一つの大画面や高解像度の画像に結合する技術も研究されており、画像をリアルタイムに結合するハードウェアには複数の映像信号を結合する Lightning-2[13] や、任意の色・幾何変形を行う PA ボード [14] がある。それらを利用した並列ディスプレイ環境システムについては、ディスプレイ間のつなぎ目を、計測により削減しシームレスに接続する Projector Array[14] がある。

また、レンダリングの負荷を削減する技術としては、スクリーンのピクセル数の範囲内にテクスチャマッピングの計算量を抑える Deferred Shading[11] がある。

---

*Xbox は米国 Microsoft Corporation の米国及びその他の国における商標です。

†NINTENDO GAMECUBE は任天堂株式会社の商標です。

‡OpenGL は米国 SGI の登録商標です。

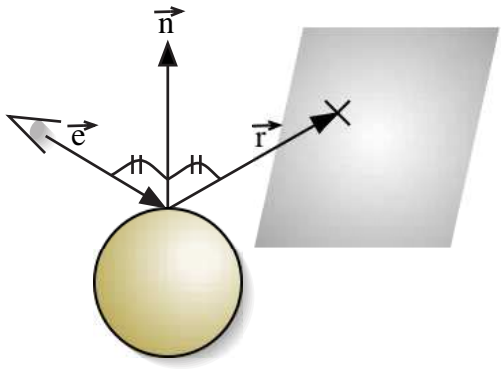

図 1: 環境マップ

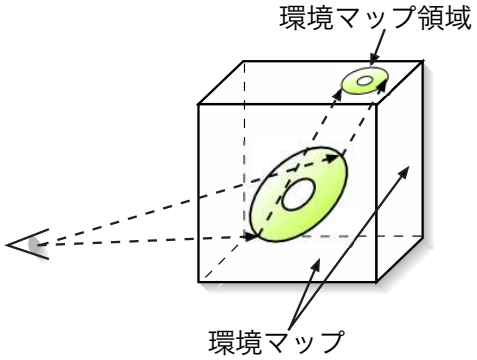

図 2: 環境マップ領域

3 アルゴリズム

3.1 立方体環境マッピング

環境マップは、レンダリングする物体を囲むように配置されたテクスチャ面に対して、背景シーンをレンダリングし、そのテクスチャをレンダリングするピクセルに入射する光線として利用するテクスチャマップで、通常、鏡面反射を表現するのにもちいる (図 1)。背景が十分にレンダリングする物体から離れている時には、レイレーシング手法の近似となる。

視線ベクトル $\vec{e}$、法線ベクトル $\vec{n}$ から、反射ベクトル $\vec{r}$ が求まり、3 つのベクトル間の関係は次式で表される。

$$\vec{r} = \vec{e} - 2(\vec{n} \cdot \vec{e})\vec{n} \qquad (1)$$

立方体環境マップは、6 枚の平面からなる立方体で、レンダリングする物体を包囲する環境マップである (図 2)。以下では、簡単のために、すべて立方体環境マップを用いて考える。他の環境マップでも、歪み方や面数以外は同じで、本提案手法を適用できる。

3.2 事前領域特定手法

本手法は、環境マップ生成に関する計算量をいかに減らすかという事を念頭において開発した。通常の球であれば、すべての環境マップが必要であるが、薄い円盤であったり、マルチディスプレイで、オブジェクトの一部を描画する時には、環境マップの一部しか使用されていない事に注目した (図 2)。そこで、事前に使用するテクスチャ領域を調べる環境マップ手法 (事前領域特定手法) を考案した。

事前領域特定手法を用いた環境マッピングは下記方法で行う。

1. 1 ビットのステンシルバッファを用意して、すべてのステンシルビットを 0 とする。

2. ステンシルバッファを書き込み状態にして、環境マップ領域を生成する。
   ここで、環境マップとして参照される領域のみのステンシルバッファのビットが 1 となる。このビットが 1 の領域を環境マップ領域と呼ぶ。

3. 全く必要としない環境マップ面を削除。
   OpenGL の HP_occlusion_test[1] 等のテスト関数を用いて、前段で何も書き込まれなかった面を削除する。

4. ステンシルテストをオンにして、通常の環境マップ生成処理を行う。
   ここで、環境マップ領域の範囲のみの環境マップが生成される。

5. 環境マッピング。
   通常の環境マッピングを行う。

6. 視点の移動等、環境マップ領域に変更がある場合には、必要に応じて処理を繰り返す。

環境マップを適用するオブジェクトが時間的に静的な場合には、環境マップ領域を更新する必要がない。そのため、本手法の効果はより大きくなる。一方、動的な場合には、毎回環境マップ領域を生成する必要が生じるため、環境マップ領域生成負荷が大きいと、通常の手法より遅くなる。

今、レンダリングパイプラインを示すのに、次に述べるような表記法を使う [12]。

大文字はバッファを表す事とし、C はカラーバッファ、T はテクスチャバッファとする。ステンシルバッファをあらわす時には T_S を用いる。頂点座標は $\mathbf{x}$ を用い、$\mathbf{c}$ はカラー値、$\mathbf{s}$ は法線ベクトル等のシェーダ変数、$\mathbf{u}$ はテクスチャ座標をあらわす。$\mathbf{us}$ は $\mathbf{u}(\mathbf{s})$ と同じ意味である。また、δ はラスタライズ操作、π は物体座標から世界座標への変換をあらわしている。$\leftarrow$ はルックアップを表し、$\mathbf{x} \leftarrow T$ と表記したら、テクスチャT を頂点座標 $\mathbf{x}$ に対応づけるという意味とする。こうした表記法の元で、上記手法を記述すると次式となる。

$$T_S \delta \pi \mathbf{x} \leftarrow \delta \mathbf{usx}$$

次に、ステンシルバッファが 1 となった領域に対してのみ環境マップを生成し、環境マッピングを行う。この処理は次式で表現できる。

$$C \delta \pi \mathbf{x} \leftarrow T \delta \mathbf{usx}$$

ここでは、$T_S = T_{S=1} + T_{S=0}$ であるので、最悪の場合でも計算量は同じで、通常は計算量が削減されることがわかる。

いま、本手法で用いているレンダリングパイプラインは Texture Shader[3] と同じであるので、他の手法でも使用できる。

4 アーキテクチャ

次に、前節で紹介した環境マップ手法に最適なハードウェアアーキテクチャを提案する。通常のグラフィックスハードウェアでは、図 3(a) に示すように、Pixel Shader の後に Alpha Test や Stencil Test がおこなわれるアーキテクチャであるが、順番を入れ替えるアーキテクチャを提案する (図 3(b))。提案アーキテクチャにより、環境マップ領域が小さい場合には、大部分の Pixel Shader 処理を削減することができ、レンダリングの負荷を大幅に削減する事が可能である。

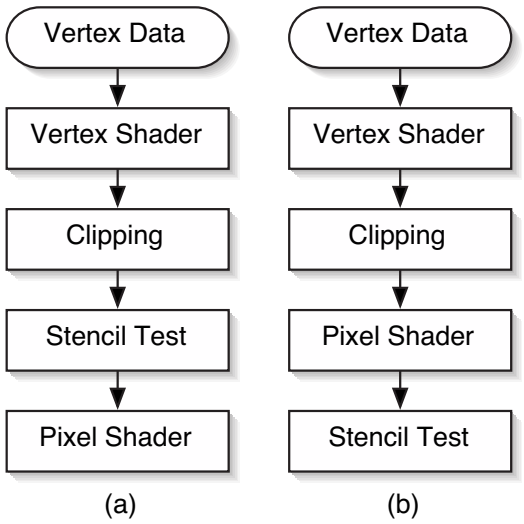

図 3: ハードウェアアーキテクチャ

5 実験

5.1 装置構成

本提案環境マッピング手法の有効性を確認するため、複数台のディスプレイによる並列ディスプレイ環境を構築した (図 4)。

各 PC は、1G Hz の PentiumIII[§] CPU を搭載し、グラフィックスカードは GeForce3[¶] を用いている。

6 台の映像 PC と 1 台の統括 PC を 100BASE-T でネットワーク接続されている。各映像 PC には DLP[‖] プロジェクタが接続されて、正面投射方式でスクリーンに映像を投影している。スクリーンは半球面状である。各映像 PC はプロジェクタが理想位置にあると仮定してスクリーンに投影し、つなぎ目部分の重なりは約 10% ほどとする。ただし、プロジェクタを理論上の位

[§]Pentium は米国およびその他の国における、Intel Corporation またはその子会社の商標または登録商標です。
[¶]GeForce3 は NVIDIA Corporation の商標です。
[‖]DLP はテキサス・インスツルメンツの商標です。

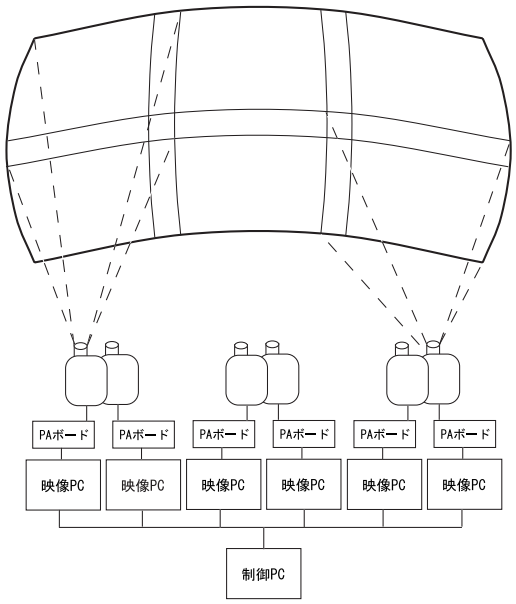

図 4: 環境概要

置に設置しても誤差が生じ、また、プロジェクタ間の輝度・色空間の差異により、通常、重なり部分にはつなぎ目が見える。

そこで、本装置では、つなぎ目を無くすために、事前にテストパターンを用いて色・幾何計測を行い、そのデータを基にシームレスにプロジェクタ間の画像がつながるようにする。グラフィックスカードで色・幾何変形を行うには負荷が高いため、リアルタイムに幾何・色変形を行う PCI ボード (PA99 ボード) を独自開発した。PA99 ボードは入力映像信号に対して、一画素単位で任意の幾何変形・色変形をリアルタイムに処理できるカードで、1024×768 の解像度で、毎秒 60 フレームの変換が可能である。

グラフィックスカードから入力され映像信号を PA99 ボードで変換し、プロジェクタへ映像信号を出力し、この映像を投影することによって、1 枚のシームレスな映像を生成する。本装置では、横 3× 縦 2 面で構成されるため、2868×1460 画素の解像度を持つ映像表示装置として機能する。

本装置で、事前領域特定手法を用いた環境マッピングを実装し、その有効性を確認した。

5.2 計算時間の評価

前記装置構成における、本アルゴリズムと通常アルゴリズムの通信と計算時間について評価する。ネットワークの転送速度 N(Byte/sec)、各 PC での画像生成速度 P(Byte/sec)、環境マップ画像バイト E(Byte) と、e を本手法による効率とする。e は 0 に近づくほど高効率である。s は 1 画素ピクセルの容量とステンシルバッファの容量の比率でで、通常は 1 画素ピクセルが 32bit に対して、ステンシルバッファは 1 ～ 8byte であるので、$s = 4 \sim 32$ である。

この時に各手法での所要時間は下記になる。第一項が通信時間であり、第二項が計算時間である。

1. 1 台で環境マップ生成
   1 台の PC で環境マップを生成し、残りの PC にブロードキャストする。
   $$6E/N + 6E/P$$
2. 6 台で分担して環境マップ生成
   各 PC が 6 面の環境マップのうちの一面分を生成し、お互いにブロードキャストしあう。
   $$6E/N + E/P$$
3. 6 台の各々で環境マップ生成
   各 PC で 6 面分の環境マップを独自に生成する。通信は発生しない。
   $$0 + 6E/P$$
4. 事前環境特定手法を用いた環境マップ生成
   $$0 + (6E/s + 6E)/P \times e$$

通常の環境では $N \ll P$ であり、$s \ll P$ であるので、十分効率が良ければ、事前環境特定手法が一番計算時間が短いことがわかる。

6 まとめ

本研究では、並列ディスプレイ環境に適した環境マッピング手法である、事前領域環境マップ

手法を開発した。ステンシルバッファを持ちいると簡単に実装できるが、より適したグラフィックスアーキテクチャの提案も行った。また、実際の並列ディスプレイ環境を構築し、その有効性も確認した。現在のグラフィックスハードウェアアーキテクチャでも十分有効なアルゴリズムで、特に並列ディスプレイや VR 環境で効果を発揮する。今後は、こうしたアルゴリズムを組み込んだグラフィックスエンジンの開発と、他の手法への応用を目標としている。

参考文献

[1] "GL_HP_occlusion_test". http://oss.sgi.com/projects/ogl-sample/registry/HP/occlusion_test.txt.

[2] J. Blinn and M. Newell. "Texture and Reflection in Computer Generated Images". *Communications of the ACM(SIGGRAPH '76 Proceedings)*, 19(10):542–546, 1976.

[3] M. D.McCool and W. Heidrich. "Texture shaders". In *1999 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pages 117–126, 1999.

[4] M. Eldridge, H. Igehy, and P. M. Hanrahan. "Pomegranate: A Fully Scalable Graphics Architecture". In *Proceedings of ACM SIGGRAPH 2000*, pages 443–454, 2000.

[5] B. Gooch, P.-P. J. Sloan, A. Gooch, P. S. Shirley, and R. Riesenfeld. "Interactive Technical Illustration". In *1999 ACM Symposium on Interactive 3D Graphics*, pages 31–38, 1999.

[6] N. Greene. "Environment Mapping and Other Applications of World Projections". *IEEE Computer Graphics & Applications*, 6(11):21–29, 1986.

[7] P. Haeberli and M. Segal. "Texture Mapping As A Fundamental Drawing Primitive". In *Fourth Eurographics Workshop on Rendering*, pages 259–266, 1993.

[8] W. Heidrich and H.-P. Seidel. "View-independent environment maps". In *SIGGRAPH / Eurographics Workshop on Graphics Hardware.*, pages 39–46, 1998.

[9] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan. "WireGL: A Scalable Graphics System for Clusters". In *Proceedings of ACM SIGGRAPH 2001*, pages 129–140, 2001.

[10] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. Kirchner, and J. Klosowski. "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters". *ACM Transactions on Graphics*, 21(3):693–702, 2002.

[11] S. Molnar, J. Eyles, and J. Poulton. "PixelFlow: High-speed rendering using image composition". In *Computer Graphics (Proceedings of SIGGRAPH 92)*, volume 26, pages 231–240, 1992.

[12] M. Olano, J. C.Hart, W. Heidrich, and M. McCool. *Real-Time Shading*. A K Peters, 2002.

[13] G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt, and P. Hanrahan. "Lightning-2: A High-Performance Display Subsystem for PC Clusters". In *Proceedings of ACM SIGGRAPH 2001*, pages 141–148, 2001.

[14] M. Yamasaki, T. Minakawa, H. Takeda, S. Hasegawa, and M. Sato. "Technology for seamless multi-projection onto a hybrid screen composed of differently shaped surface elements". In *IPT 2002*, 2002.