Vol. 37 No. 11　　情　報　処　理　　Nov. 1996

解　説

計算機システムを支える最新技術（装置編）

5. グラフィックスアクセラレータ†

滝澤哲郎††　石田博文††

1. はじめに

Windowsの普及にともない，グラフィックス表示を高速化するためのハードウェアであるグラフィックスアクセラレータが広く普及し，ここ数年にわたり性能の競争が繰り広げられてきた．しかし，最近では，一般のグラフィックス表示の性能向上は鈍化しつつあり，代わってビデオ機能と3D機能と呼ばれる新しい機能が登場し，注目を浴びている．

ビデオ機能はPC上でビデオデータが扱われるようになってからの新しい機能であるが，急速に普及しすでに多くのアクセラレータに取り込まれている．3D機能はWS用のハイエンドアクセラレータには何年も前から搭載されているが，半導体の高集積化，CPU性能の向上，APIの策定という条件が整い始めたことでPCの世界にも進出してきた．

本稿では，グラフィックスアクセラレータの基本的なしくみから，グラフィックス用メモリ，ビデオ機能，3D機能について解説する．

2. グラフィックスアクセラレータとは

本章では，グラフィックスが画面に表示されるしくみについて述べたあと，グラフィックスアクセラレータの基本機能と性能を決定する要因について述べる．

2.1 画面に絵が出るまで

一般的なPCのグラフィックスボードは主に，グラフィックスアクセラレータ（GA），グラフィックスメモリ（GRAM），D/Aコンバータ（DAC）からなる．通常，バスインタフェースはGAが備えており，直接バスに接続されている．ディスプレイにはDACの出力が接続されるが，最近のGAではDACを内蔵したものも多い．

画面に表示される絵の情報はGRAM上に保持されている．GRAM上で絵の情報が格納されている領域はとくにフレームバッファ（FRB）と呼ばれる．FRB上のデータは画面上の画素に1対1に対応しており，画面の左上の画素を起点として，右方向へ昇順に配置されていることが多い（画面の右端に達したら，次の行の左端につながる）．これをラスタ順と呼ぶ．

各画素の色は，通常は赤(R)，緑(G)，青(B)の光の三原色で表現されており，画面上の1画素がFRB上で何ビットを占めるかによって，画面上に表示できる色数が決まる．この1画素あたりのビット数はBit Per Pixel（BPP）と呼ばれる．BPPが多い場合（多くの場合，16 BPP以上）はFRB上にRGB値がそのまま格納されている．これが少ない場合（多くの場合，8 BPP以下）はFRB上にはRGB値ではなくテーブルへのインデックス値が格納されている．テーブルには各インデックスに対応したRGB値が格納されており，画面にはこのテーブルを引いた結果のRGB値の色が表示される（**図-1**）．このテーブルはルックアップテーブル（LUT）と呼ばれる．

FRBに格納されている画面情報は，一定の周期でGRAMからラスタ順で読み出され，DACでアナログ信号に変換されて画面上に表示される．LUTを使用する場合はアナログ変換の前にテーブル変換が行われる．

画面上にグラフィックスを表示するためには，何らかの手段でFRBの内容を上記の規則に従って絵の情報で埋め尽くしてやればよい．最も一般的な方法は，GRAMをCPUのメモリ空間あるいはI/O空間に配置し，CPUからGRAMに直

† Graphics Accelerator by Tetsuro TAKIZAWA and Hirofumi ISHIDA (NEC Corporation C & C Research Laboratories).
†† NEC C&C研究所

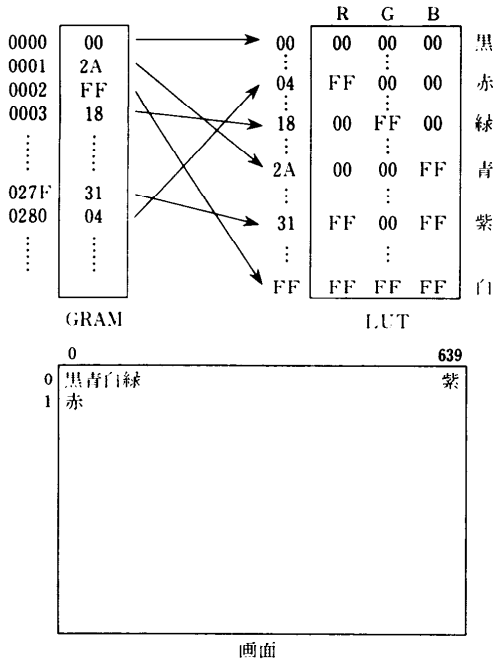

図-1 画面とメモリの対応

接アクセスできるようにすることである．実際，すべてのGAは例外なくこの機能を備えている．

2.2 アクセラレーションの基本

グラフィックス表示の高速化に大きく影響を与え，GAの基本ともいうべき機能が，線分描画と領域転送（BitBLT）である．

線分描画はその名前が示すとおり，線分の両端点の座標を与えるだけで線分を描画する機能である．さらに，すでに画面上に描かれているグラフィックスと論理演算を行った結果を描画できる機能や，一定のパタンを参照することで破線などが描画できる機能を備えたGAが多い．線分の軌跡を求めるアルゴリズムにはいくつかあるが，Bresenhamのアルゴリズムが広く用いられている[1]．線分描画は，とくにCADアプリケーションでその性能が重視される．

BitBLTは，GRAM上のある矩形領域の情報を別の同じ大きさの矩形領域にコピーする機能である．さらに，コピー先にすでにある情報や一定のパタンとの間でビットごとに論理演算を行う機能をもつものが多い．この機能はとくにラスタオペレーション（ROP）と呼ばれる．BitBLTは，ウィンドウの移動，スクロール，領域の塗りつぶしなどに対応し，ウィンドウシステムで最も多用される処理である．また，1BPPのビットパタンを画面のBPPに展開してコピーする機能がある．この機能はカラー拡張と呼ばれ，文字描画の高速化に寄与する．

2.3 アクセラレータの性能

GAの性能は，前節で述べた線分描画とBitBLTの処理性能によって決まることはいうまでもないが，その他にも以下にあげるような要因がある．

メモリバンド幅

GAの処理では，GAとGRAM間でデータの入出力が頻繁に行われる．そのため，メモリバンド幅が狭いと性能を発揮することができない．メモリバンド幅は，メモリの種類（後述）とメモリバス幅によって決まる．

バス

CPUからGAへの描画命令の転送や，CPUからGRAMへの直接アクセスはバスを経由する．現在では，PCIバスが主流であるが，より速いバスも提案されている（Accelerated Graphics Port）[2]．

デバイスドライバ

デバイスドライバは，ウィンドウシステムの描画命令をGAが直接理解できる描画命令に変換する役割を担っている．同じGAでもドライバによって性能が大きく違うことがある．

GAの性能を評価する目安としてはベンチマークテストがよく用いられる．Windowsでは，WinBenchというベンチマークテストのグラフィックスWinMarkの値を参照することが多い．

3. グラフィックス用メモリ

従来，GRAMには2種類のメモリが主に使われてきた．1つは一般的なDRAMで，もう1つはDual Port DRAMあるいはVideo RAM（VRAM）と呼ばれるメモリである．VRAMは通常の入出力ポートのほかに，シリアルアクセスのみが可能な入出力ポートを備え，互いに非同期な2つの入出力が同時に可能なメモリである．

通常，VRAMのシリアルポートは表示読み出しに用いられ，入出力ポートはグラフィックスの描画に使用する．高い解像度とBPPでは表示読み出しが占める割合が非常に大きいため，VRAMはDRAMに比べて高い性能を出すこと

1012

表-1 グラフィックス用メモリ

メモリ	DRAM	VRAM	EDO DRAM	SDRAM	SGRAM	RDRAM	WRAM	MDRAM
ピーク性能 (MB/s)	300	300 +400	400	800	800	533	400 +400	800
実効性能 (MB/s)	240	240 +400	280	630	630	400	280 +400	720
特殊機能	×	○	×	×	○	×	◎	×
価格	1	2〜3	1	1〜1.1	1.2〜1.5	1.2〜1.5	1.5〜2	1.5
供給元	多	多	多	多	多	多	1	少

性能は 64 bit 接続時の性能(RDRAM のみ 8 bit 接続時の性能)
性能欄の下段はシリアルポートの性能
実効性能は 16 ページバーストアクセス時の性能
SDRAM/SGRAM のピーク性能は 100 MHz 動作時，実効性能は 83.3 MHz 動作時
MDRAM は 100 MHz 動作時の性能
価格は対 DRAM 比(変動が激しいため参考値)

ができる．さらに，VRAM はグラフィックス処理を高速化する機能を 2 つ備えている．1 つはビット単位でライトデータをマスクする機能（ライトパービット)，もう 1 つは同一データを連続して高速にライトする機能（ブロックライト）で領域の塗りつぶしに効果が高い．

ただし，VRAM の価格は DRAM に比べて 2 倍近く高いため，ハイエンドの GA では VRAM，ローエンドの GA では DRAM という棲み分けが長く行われてきた．現在ではさまざまな種類のメモリが登場し選択の幅は広がったが，絶対の解となるメモリはいまだ現れていない．その中でも，性能や価格などのバランスがよい SDRAM，SGRAM，RDRAM を採用するケースが今後増えていくと予想される．

表-1 に代表的なグラフィックス用メモリの比較結果をまとめた．以下では各メモリの特徴を簡単に紹介する．

Extended Data Out（EDO）DRAM

通常の DRAM では CAS をインアクティブにするとデータ出力も終了するが，次に CAS がアクティブになるまでデータ出力を保持するようにした DRAM．ページサイクルを短くできる．

Synchronous DRAM（SDRAM）

制御信号やデータの入出力をクロック同期化するとともに，動作をパイプライン化し，ページサイクルを短くした DRAM．

Synchronous Graphics RAM（SGRAM）

SDRAM に VRAM の特殊機能の一部をもたせ，グラフィックス性能を高めたもの．

Rambus DRAM（RDRAM）

実装を細かく規定することで，高速クロックでの動作を可能としたメモリ．現状ではランダムアクセス性能に難があるが，これを改善したコンカレント RDRAM も提案されている．

Windows RAM（WRAM）

VRAM の入出力ポートを EDO DRAM と同様に高速化するとともに，内部に BitBLT の一部機能を集積し，条件次第で BitBLT を高速に実行できるようにしたメモリ．

Multibank DRAM（MDRAM）

内部に多数の DRAM バンクをもち，ページを変更してもバンクが異なれば途切れなく連続アクセスできるようにしたメモリ．

4. ビデオ機能

本章では，ビデオ処理の流れとビデオアクセラレータの役割について説明する．とくに，画質を左右するスケーリング処理とビデオ表示方式について詳しく述べる．

4.1 ビデオアクセラレータとは

通常，ハードディスクや CD などに収められているビデオデータは圧縮されており，これを画面に表示するまでには，**図-2** に示したような処理を経る必要がある．従来の GA でビデオ表示を行う場合には，これらすべての処理を CPU で行っていたため，CPU の負荷率も高く，性能も低かった．

結論から先に述べると，図-2 中の下線部の処

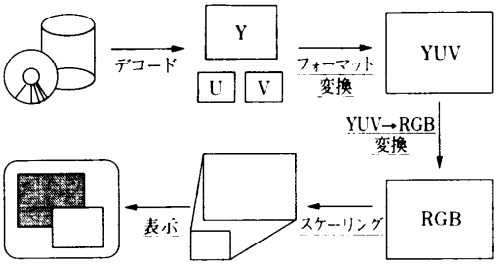

図-2 ビデオ処理の流れ

理をGA側に受け持たせたのがビデオアクセラレータと呼ばれるものである．これらの処理をハードウェア化した理由には以下のようなことがあげられる．

1. ハードウェア量の割に効果が大きい
2. さまざまなビデオ圧縮アルゴリズムで共通に利用できる
3. ソフトウェアとのインタフェースがとりやすい

ビデオアクセラレータの普及には3.が大きく寄与している．Microsoftが Display Control Interface (DCI) と呼ばれるインタフェースを策定し，これを利用したすべてのアプリケーションでビデオアクセラレータの恩恵を受けることができるようになった．

以下では，各処理について簡単に説明する．

フォーマット変換

ビデオデータは通常，輝度(Y)と色差 (U/V) で表現され (YUV 空間)，U と V は間引かれているのが普通である．これは，人間の目が輝度の変化には敏感だが，色の変化には鈍感であることに起因している．フォーマット変換では，間引かれているU/Vデータを補い，ビデオデータを画素単位に再構成する．

YUV → RGB 変換

前記の YUV 空間上の画素値を，RGB 空間上の画素値に変換する．

スケーリング

ビデオを指定されたウィンドウの大きさに合わせるために，拡大/縮小が必要となる．詳細については後述する．

表示

最終的に，ビデオはウィンドウの1つとしてグラフィックスと合成されて表示される．詳細については後述する．

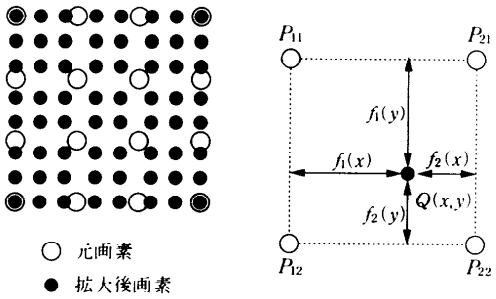

図-3 スケーリング（拡大）処理の例

4.2 スケーリングと画質

スケーリングは元々のビデオの大きさが表示すべき大きさと異なる場合に，足りない画素を補う，あるいは余分な画素を間引く処理である．ウィンドウシステムではウィンドウの大きさは自由に変えられるので，スケーリングも任意の大きさに変換できなければならない．

スケーリング処理は幾何学的変換の一種で，2次元空間で標本化されている元画像の各画素間の間隔を変更し，再標本化することに相当する．**図-3**に拡大の例を示すが，この図のように標本点の変換を行うとともに変換後の画素値を求めるのがスケーリング処理である．このとき，変換後の画素値に最も近い元の画素の値をとるのが最も単純な方法で，最近傍（零次補間）法と呼ばれる．最近傍法は最も少ないハードウェアで実現できるが，結果の画質は悪い．また，ソフトウェアでは，処理速度の関係で最近傍法をとらざるを得ない．

一方，変換後の画質を向上するためには画素値の補間を行うことになる．最も少ないハードウェアで効果が認められるのが，4点線形補間である[3]．4点線形補間は，近傍の4点の画素値から線形に内挿して変換後の画素値を求める方法である．その変換式は以下のとおりである．

$$Q=[f_1(y)f_2(y)]\begin{bmatrix} P_{11}P_{21} \\ P_{12}P_{22} \end{bmatrix}\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix}$$

$$f_1(t)=1-(t-[t])$$

$$f_2(t)=t-[t]$$

各画素値は3つの8ビット精度で表されているのが普通であるので，上式をそのまま完全に実装するとかなり大きな回路となる．実際には，演算をある程度近似することでハードウェア量を抑えたものが多い．また，2次元の補間を効率よく行

1014

うためには最低でも1ライン分のバッファメモリがGAの内部に必要となるため，これを省き，X方向のみの2点線形補間しか行わないものも多い．

4.3 ビデオ表示方式

ビデオの表示方式には以下の2方式がある．いずれの方式でもビデオデータはGRAM上のFRBを除いた残りの領域（オフスクリーン）に置かれている．

インレイド方式

オフスクリーンのビデオデータをFRBに転送することでビデオとグラフィックスをFRB上で合成する方式．拡大/縮小などの処理は転送時に行われ，FRB上にはビデオを含んだ表示イメージがそのまま格納される．ビデオが非矩形の場合は，マスク情報に基づいて転送を行う．以下の特徴をもつ．

- ビデオの表示品質がFRBのBPPに依存する（FRBが8BPPではビデオの画質が低い）
- 少ないハードウェアで複数のビデオを同時に表示することができる

オーバレイ方式

オフスクリーンのビデオデータを表示時にFRB上のグラフィックスと合成する方式．拡大/縮小などの処理は表示時に行われる．FRB上にはビデオは格納されない．ビデオが非矩形の場合は，ビデオが表示される領域に対応するFRB上の領域を特定色で塗りつぶし，その部分にのみビデオを合成する．以下の特徴をもつ．

- FRBのBPPに依存せず，ビデオは常に高画質で表示できる
- 複数のビデオを表示するためには，ハードウェア量が大きくなる

初期のビデオアクセラレータはVRAMを用いたハイエンド指向のものが多かったため，インレイド方式によるものがほとんどであった．これは，VRAMを使用すると表示データがGA内部を経由せずDACに直接送られるので，表示時にビデオとグラフィックスを合成することが難しいためである．現在では，DRAMを使用したGAでもビデオアクセラレータを備えているが，こちらの主流はオーバレイ方式である．

なお，インレイド方式ではFRBが8BPPのときにビデオの画質が低いと述べたが，この欠点を克服しているGAもある[4]．

5. 3D 機 能

3D機能は正確には3Dグラフィックス処理のアクセラレーション機能である．一般に3Dグラフィックス処理（以下3D処理と呼ぶ）にはレイトレーシングやラジオシティといった高品質な画像を生成する手法も含まれるが，GAのようなハードウェアではZバッファ法と呼ばれる低コストで高速化しやすい手法が使われている．本章では，まずこのZバッファ法による3D処理のおおまかな流れについて述べ，次に現在のGAに搭載されている，あるいは今後搭載されるであろう機能を中心に説明し，最後に3D性能と処理のボトルネックについて言及する．

5.1 3D処理の流れ

3D処理のおおまかな流れを**図-4**に示す．入力は3D形状（以下物体と呼ぶ）データと視点や光源を表すパラメータなどであり，物体はポリゴン（多角形）やワイヤーフレーム（ライン）で表現される．3D処理の中心はポリゴンであることから，ここではポリゴンの処理に絞って説明する．

3D処理は大きく分けて3Dの広い空間内で演算を行うジオメトリ処理と，2Dの狭い表示画面内で演算を行うレンダリング処理に別れる．ジオメトリ処理では，物体をそれ固有のオブジェクト座標系から視点を原点とした視点座標系に変換する座標変換，物体表面で反射して視点に到達する

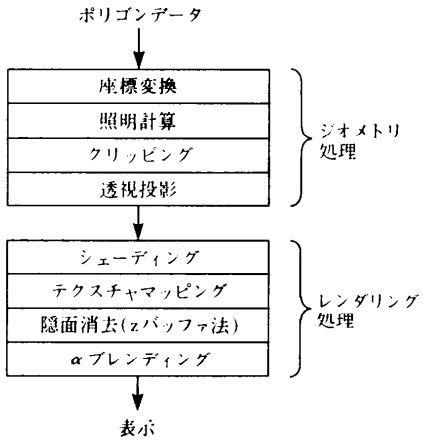

図-4 3D処理の流れ

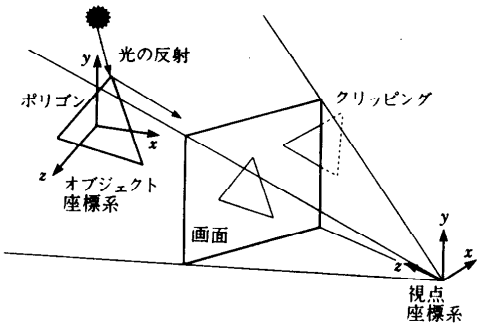

図-5　ジオメトリ処理

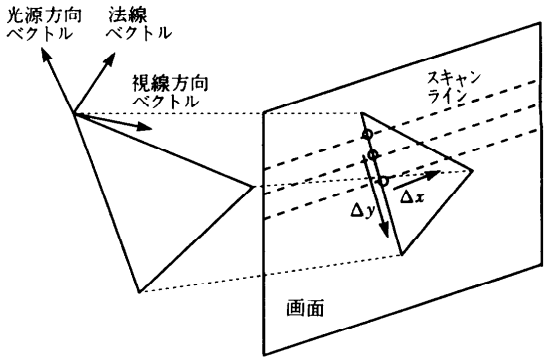

図-6　照明計算とシェーディング

光量を計算する照明計算，表示画面からはみ出す部分を切断するクリッピング，物体を表示画面に投影する透視投影を行う（図-5）．また，レンダリング処理では物体表面に陰影をつけるシェーディング，物体表面に画像を張りつけた効果を出すテクスチャマッピング，近くの物体が遠くの物体を隠す隠面消去，半透明の効果を出すαブレンディングなどを行う．

現在のGAはレンダリング処理の一部しか実行しておらず，ジオメトリ処理はすべてCPUが行う分担になっている．このため，以下ではレンダリング処理を中心に若干ジオメトリ処理について述べる．

5.2　シェーディングと照明計算

シェーディング方法としては，Gouraud shadingとflat shadingの2種類が広く用いられている．Gouraud shadingはポリゴンの各頂点で照明計算した結果の輝度値（通常はR，G，Bの3成分をもつ）を線形に補間して，ポリゴン内部の各画素での輝度値を求める．一方flat shadingは，ポリゴンの代表点（通常は頂点の1つ）で照明計算した結果をポリゴン内部の各画素で同輝度値にする．このため，flat shadingではポリゴンの境界が目立ち，いかにも多面体という画像になる．以下，照明計算の方法とGouraud shadingの手順について述べる．

照明計算ではいくつかのモデルによる計算法が用いられる．比較的簡単なものとしては，光源からの直接光が物体表面で散乱する拡散反射光モデルや空間全体に一様な光があたっているとみなす環境光モデルがある．また複雑なものとしては，光源からの直接光が物体表面で鏡のように鋭く反射する鏡面反射のモデルや光線の中心から離れるにつれて徐々に暗くなって行くスポットライトモデルがある[7]．照明計算はこうしたモデルを複数の光源に適用して，光源数分の反射光の和として計算する．複数の光源を用いたり複雑なモデルを使うと計算量は増えるが，よりリアルな画像が生成できる．

Gouraud shadingは以下の手順で行われる（図-6）．

1. Δ計算：ポリゴンのエッジに沿ってyが1増加したときの輝度値の増分Δyと，x方向に1増加したときの輝度値の増分Δxを求める．
2. スパン計算：ポリゴンとスキャンラインの交線をスパンと呼ぶが，そのスパンの左端点の輝度値にΔyを加算して順次次ラインのスパンの左端点の輝度値を求める．
3. 画素値計算：スパンの左端点の輝度値にΔxを順次加算してスパン内部の各画素の輝度値を求める．

5.3　テクスチャマッピング

テクスチャマッピングは，テクスチャバッファに格納されているテクスチャ画像をポリゴン上に対応させて張り込む処理である．基本的には以下の手順で行う．

1. 各頂点に対応したテクスチャアドレスを補間してポリゴン内部の画素位置(x, y)でのテクスチャアドレス(u, v)を求める．
2. (u, v)が指すテクスチャ画像の輝度値を読み出して(x, y)に張りつける輝度値を求める．
3. シェーディング結果の輝度値と2.で求めた輝度値を混合して(x, y)での輝度値を求める．

1. 2.の具体的な処理方法によって生成画像の品質に違いが生じる．以下ではそれらの違いを説

1016

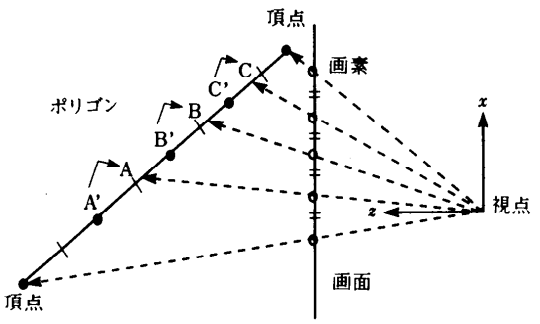

図-7 パースペクティブコレクション

明する．

5.3.1 パースペクティブコレクション

1.における最も簡単な補間は線形補間である．すなわち，頂点に与えられたテクスチャアドレスをシェーディングと同様に線形に補間してポリゴン内部の各画素でのテクスチャアドレスを求める．通常は計算量の点からこの方法を用いることが多いが，この場合透視投影によるテクスチャのずれが問題になることがある．その状況を図-7に示す．これはy軸方向から投影してみた図であるが，たとえばA'の点がAに，B'，C'の点がそれぞれB，Cにあるかのようにずれてみえる．

こうしたずれは，ポリゴンの頂点でのZ値の差が大きい場合に顕著に現れ，とくに動画像にすると歪みがはっきりわかることがある．この歪みを防ぐための手法がパースペクティブコレクションで，ポリゴン頂点のテクスチャアドレスをいったん画面上の座標系で表現し直してから線形補間し，再びポリゴン上の座標系に戻すことで実現できる[6]．ポリゴン上の座標系と画面上の座標系の行き帰は除算による非線型変換が必要になる．このため，パースペクティブコレクションを正確に行うには，画素ごとに除算を行う必要がある．

5.3.2 ポイントサンプリングとバイリニア補間

2.では，テクスチャアドレス(u, v)が指すテクスチャ画像を読み出すと述べたが，u, vは一般には整数値とはならない．そのため簡単な方法としては，(u, v)に最も近い整数格子点をテクスチャアドレスとする．これをポイントサンプリングという（ビデオ機能のスケーリングにおける最近傍法に相当する）．しかし，ポイントサンプリングは画質が悪いため，スケーリングの項で説明した4点線形補間と同様に(u, v)の近傍4点で輝度値を読み出し，それらを加重平均する処理が行われる．これをバイリニア補間と呼ぶ．

5.3.3 ミップマップ法

バイリニア補間によって通常は十分な品質の画像が得られる．しかし，小さいポリゴンに大きなテクスチャ画像を貼りつけるような画像の縮小をともなう場合，折り返し歪みというモアレが生ずる．この折り返し歪みを少ない計算量で弱める方法がミップマップ法という手法である．ミップマップ法は以下の手順で行う．

1. あらかじめ折り返し歪みの原因となる高周波成分を取り除いて縦横両方向を1/2，1/4，1/8，…に縮小した複数レベルの画像（ミップマップ）を生成しておく．
2. テクスチャ画像をポリゴンに貼りつける際の縮小度を求めて，縮小度に応じたレベルのミップマップを選択する．
3. 選択したミップマップを用いてポリゴンへの張りつけを行う．

ミップマップのレベルは0，1，2，…という整数値で表すのだが，縮小度からレベルを算出する計算式でレベルを求めると一般には整数値とはならない．このため，計算結果に近いレベルのミップマップを選択するか，あるいは両方のレベルを用いて加重平均するかで2とおりの計算法がある．また，1つのレベル内ではポイントサンプリングとバイリニア補間の2とおりがあるため，合計4とおりの計算法が存在する．こうした計算法の中では，2つのレベルを用いレベル内で4点内挿をする方法が最も画質がよく，これをトリリニア補間と呼ぶ．

5.4 隠面消去（Zバッファ法）

Zバッファ法は，FRBと同じ大きさのZバッファ，すなわち奥行き値Zを格納するバッファを用い，1画素ごとに隠面消去を行う手法である．基本的には以下の手順で行う．

1. ポリゴン頂点で与えられたZ値を線形補間してポリゴン内部の各画素でのZ値を計算する．
2. 1.で計算した新Z値とZバッファ中のZ値とを比較して，新Z値の方が小さい場合にのみFRB，Zバッファへの書き込みを行う．

5.5 αブレンディング

αブレンディングは，レンダリングの最終段階で計算結果の輝度値とFRB中の輝度値とを混合係数αで混ぜ合わせる．この混合係数αは，通常は入力時にポリゴンの頂点に与えられ，シェーディングの際に輝度値とともに線形補間で求められる．

5.6 アンチエイリアシングとフォグ

レンダリングの基本的な処理は上で説明したとおりであるが，そのほかにもオプション処理がある．その中でいくつかのGAがサポートしているアンチエイリアシングとフォグについて述べる．

アンチエイリアシングはポリゴンの輪郭を滑らかにみせるスムージング処理である．その基本的な方法は，シェーディング計算時にポリゴンの輪郭部分で画素を覆う割合を求めて混合係数αとする．そして，αブレンディングによってFRB中の輝度値と混合係数αで混ぜ合わすことでスムージングを実現する．

フォグは視点から遠くなると霧が濃くなる効果を出す処理である．その基本的な方法は，奥行き値Zをパラメータとするフォグ関数を計算し，その計算結果であるフォグ係数をテクスチャマッピング後の輝度値に乗算する．

5.7 3D性能と処理のボトルネック

3D性能，すなわち3Dグラフィックスの描画性能を評価する尺度として，1秒間あたり何ポリゴンの描画ができるかという，ポリゴン性能がよく用いられる．しかし，レンダリング処理の性能はポリゴン性能だけで表せるものではなく，1秒間に何画素の描画ができるかという，画素性能が重要になる．というのは，レンダリングのΔ計算まではポリゴン数に依存して処理量が決まるが，それ以降の処理では画素数に依存して処理量が決まるからである．このため，3D性能を議論するにはポリゴン性能のほかにポリゴンの平均的な大きさ，たとえば50画素/ポリゴン，という条件が必要になる．

また，3D性能を議論するときに忘れてはならないのが，テクスチャマッピングの種類やαブレンディングをするかどうかといった各種の条件である．ポイントサンプリングとバイリニア補間とではテクスチャバッファをアクセスする頻度や演算量が異なり，一般的にはバイリニア補間の方が性能が低くなる．だが，残念ながら現状では，そこまで細かい条件を明記しているGAはない．

なお，3D性能の評価尺度として，OpenGL用のベンチマークであるViewPerfがハイエンドマシンではよく用いられる．ただし，現在のところドライバが提供されていないことなどからOpenGLを使えないGAも多く，客観的な性能評価ができる状況にはない．

最後に3D処理のボトルネック，すなわち3D性能に影響を及ぼす要因をまとめておく．

メモリアクセス

1画素の描画を行うごとにZバッファ，FRB，テクスチャバッファへのアクセスが生じる．ポリゴンサイズが大きくなるなど描画面積が広くなる場合には，メモリアクセス回数が増えてこれがボトルネックになることが多い．

浮動小数点演算

ジオメトリ処理では広い空間で演算が行われるため，浮動小数点演算が必要である．また，その演算量も多い．とくに，照明計算で光源数を増やしたり複雑な照明法を実行すると演算量が膨大になり，浮動小数点演算がボトルネックとなる．

CPU-GA間のデータ転送

CPUが処理した結果のポリゴンデータはGAに転送される．現状のGA程度のポリゴン性能（30万ポリゴン/s以下）の場合，PCIバス程度のデータ転送性能があればCPU-GA間のデータ転送がボトルネックになることはまずない．しかし，今後CPU性能，GA性能が向上するとこのデータ転送がボトルネックになることも考えられる．

3D処理はCPUとGAで仕事を分担して実行される．現状のGAはコスト的な制約から，Δ計算を除いたシェーディング，テクスチャマッピングの一部の機能，それとZバッファ処理のみを搭載しているものが多く，それ以外の処理はCPUに頼っている．このため一般的にはCPUの負荷が高く，CPUが3D処理のボトルネックになることが多い．

1018

6. おわりに

限られたページ数のためにすべてを網羅しているとはいえないが，グラフィックスアクセラレータの基本から最新の情報までを駆け足で紹介してきた．この分野は世代交替が激しく，毎年のように各社から新製品が発表されることが珍しくない．とくに，グラフィックス用メモリ，3D機能を取り巻く状況は急変する可能性が高い．本稿では，できるだけ普遍的な事項を取りあげ紹介したつもりである．

参考文献

1) Bresenham, J.: Pixel-Processing Fundamentals, IEEE Computer Graphics and Applications, Vol. 16, No. 1, pp. 74-82 (1996).
2) Intel Corp.: Accelerated Graphics Port Interface Specification, http: //www. teleport. com/~agfxport/ (1996).
3) 榎並，八巻他：画像信号の幾何学的変換のための補間フィルタと画質に関する一考察，信学論，Vol. J 69-D, No. 11, pp. 1617-1623 (1986).
4) 平沢，滝澤他：ビデオ機能を搭載したグラフィクスアクセラレータの開発，NEC技報，Vol. 49, No. 3, pp. 115-120 (1996).
5) Foley, J., Dam, A., Feiner, S. and Hughes, J.: Computer Graphics Principles and Practice, Addison Wesley (1990).
6) Segal, M., Korobkin, C., Widenfelt, R., Foran, J. and Haeberli, P.: Fast Shadows and Lighting Effects Using Texture Mapping, Proc. of SIGGRAPH '92, pp. 249-252 (1992).
7) Neider, J., Davis, T. and Woo, M.: OpenGL Programing Guide, OpenGL ARB (1993).

(平成8年7月24日受付)

滝澤 哲郎

1965年生．1989年早稲田大学理工学部電子通信学科卒業．1991年同大学院理工学研究科修士課程修了．同年NEC入社．現在同社C&C研究所勤務．グラフィックス/ビデオ処理のアーキテクチャ/アルゴリズムの研究およびグラフィックスアクセラレータの開発に従事．電子情報通信学会会員．

石田 博文（正会員）

1961年生．1985年東京工業大学理学部数学科卒業．1987年同大学院総合理工学研究科システム科学専攻修士課程修了．同年NEC入社．現在同社C&C研究所主任．コンピュータグラフィックスにおける画像生成の高速化およびインタラクティブアニメーションに関する研究に従事．