3次元ユーザーインタフェースのための高速ベクトル表示

Interactive Vector Rendering for 3D User Interfaces

爰島　快行　　杉田　馨
■ KOKOJIMA Yoshiyuki　■ SUGITA Kaoru

東芝は，パソコン（PC）や携帯電話に搭載されているGPU（Graphics Processing Unit）を活用して，Flashアニメーションや TrueTypeフォントなどのベクトルデータを解像度に依存せずに3次元表示する技術を開発した。従来技術で不可欠であった前処理を不要にしたことにより，動的に変化するベクトルデータを高速に表示することが可能になった。

この技術を応用して，大量のテキストを3次元表示する電子番組表ブラウザを試作し，ハードディスクに録画されている大量のテレビ映像を効率良く閲覧するための新しいユーザーインタフェースを実現した。

Toshiba has developed a new method for graphics processing unit (GPU)-accelerated rendering of vector graphics such as Flash and TrueType characters embedded in a three-dimensional space. Our method requires no expensive preprocesses, allowing it to render dynamically deformable vector objects with high efficiency.

We have implemented a prototype 3D electronic program guide (EPG) browser using this method. This browser provides an easy way for users to select their favorite TV contents from among a large number of videos stored on hard disk.

1　まえがき

われわれが日常的に目にしているFlashアニメーションやTrueTypeフォントの画像は，直線やベジエ曲線などのベクトルデータを組み合わせて構成されている。このような画像はベクトルグラフィックスと呼ばれ，PCや携帯電話の画面にユーザーインタフェースを表示するために使われている。

しかし，近年のディスプレイの高解像度化に伴い，それに見合った高解像度のユーザーインタフェース画面が求められるようになったため，ベクトルグラフィックス表示の負荷が飛躍的に増大している。特に携帯電話などの組込み機器の分野では，消費電力の制約からCPUの処理能力が低く，負荷の増大は大きな問題になる。そのため，CPU以外のハードウェアを用いてベクトルグラフィックスの表示をアクセラレートする技術に注目が集まっており，OpenVGと呼ばれる業界標準のソフトウェア仕様が策定されている。

東芝は，PCや携帯電話に搭載されているGPU（Graphics Processing Unit）を活用してベクトルグラフィックスを高速に3次元表示する技術を開発した。従来技術で不可欠であった前処理を不要にしたことにより，状況に応じて動的に変化する高解像度のユーザーインタフェース画面を高速に表示することが可能になった。

ここでは，GPUを活用してベクトルグラフィックスの表示を高速化する技術について述べるとともに，この技術を応用して試作したユーザーインタフェースの画面例として，3次元電子番組表ブラウザを紹介する。

2　GPUの基本処理

GPUの代表的なアーキテクチャを**図1**に示す。3次元グラフィックスは，一度図形をメッシュ状の三角形に分解し，GPUが図形を再構築して表示される。GPUを構成する各演算ユニットは，次のような手順で協調動作する。

(1)　頂点シェーダ　　三角形を3次元空間から画像面に投影する
(2)　ラスタライザ　　投影された三角形の内部を画素の集まりに分解する
(3)　テクスチャユニット　　テクスチャ（画像）を読み込む
(4)　ピクセルシェーダ　　テクスチャを参照して各画素の色を計算する
(5)　ラスタ処理ユニット　　各画素の色をメモリに書き込む

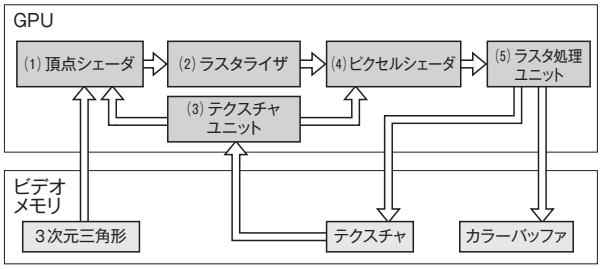

図1. GPUの代表的なアーキテクチャ — 三角形をメッシュ状にはり合わせてグラフィックスを表示することに特化して設計されている。
Overview of GPU architecture

このように，GPUはたくさんの三角形をメッシュ状にはり合わせてグラフィックスを表示することに特化したプロセッサである。したがって，ベクトルグラフィックスのような直線やベジエ曲線が組み合わされた図形を表示するためには，その図形をなんらかの方法で三角形メッシュに変換してからGPUに入力する必要がある。当社は，この変換を効率良く行うことができる新しいアルゴリズムを提案した。

GPUを用いた3次元ベクトル表示の，従来のアルゴリズムと提案したアルゴリズムを**図2**に示す。以下に，それぞれのアルゴリズムについて説明する。

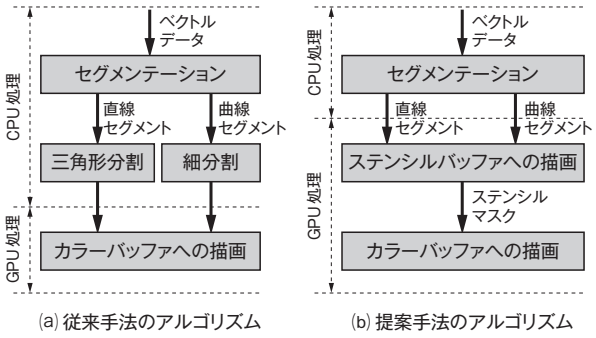

図2．GPUを用いた3次元ベクトル表示のアルゴリズム — いずれのアルゴリズムも，CPU処理とGPU処理の大きく二つの処理に分かれている。

Process flows of GPU-accelerated vector rendering

3　従来手法のアルゴリズム

提案手法の詳細を説明する前に，そのベースとなる従来手法のアルゴリズム[1]を説明して問題点を明らかにする。以降の説明では，**図3**(a)に示すような漢字“次”のTrueTypeフォントのアウトラインをベクトルデータの例として用いる。図中，黒丸はベクトルデータに含まれる直線及びベジエ曲線(注1)の始点と終点を表し，白丸はベジエ曲線の制御点を表す。

図2(a)に示すように，従来手法のアルゴリズムはCPU処理とGPU処理の大きく二つの処理に分かれている。

3.1　CPU処理

CPU処理では，最初に，与えられたベクトルデータが直線セグメントと曲線セグメントに分割される。後者の曲線セグメントとは，ベクトルデータに含まれる各ベジエ曲線の始点，制御点，及び終点を結ぶ三角形である。例えば，図3の(a)のベクトルデータからは(b)に示すような三角形の集まりが得られる。

次に，曲線セグメントどうしが重複するかどうかがチェックされる。重複する曲線セグメントが見つかった場合は，重複がなくなるまで該当する曲線セグメントが再帰的に細分割される。例えば，(b)の曲線セグメントは，(c)に示すように“次”の左払いと右払いの部分の三角形がそれぞれ二つに分割される。

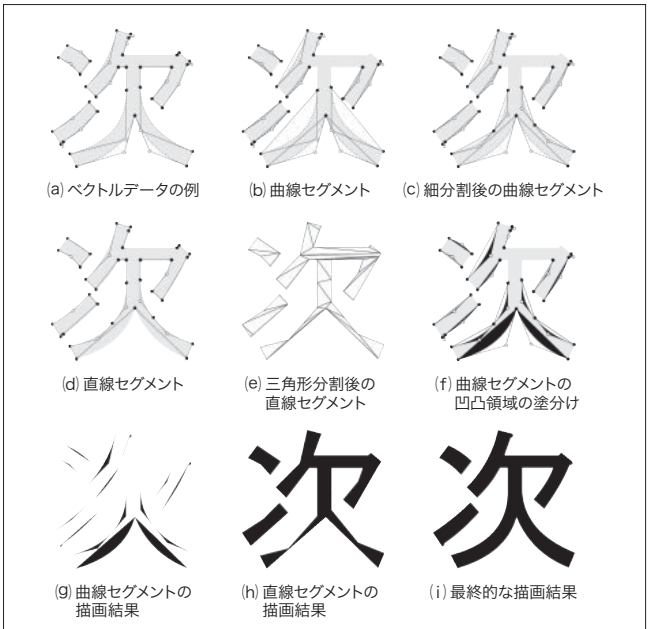

図3．従来手法の処理例 — ベクトルデータは，CPUで直線セグメントと曲線セグメントに分けられ，それぞれ更に分割される。GPUで最終的に描画される。

Example of vector rendering using conventional method

もう一方の直線セグメントとは，ベクトルデータに含まれる直線の始点と終点，凹領域に図形の内部があるベジエ曲線（凹ベジエ曲線）の始点と制御点，終点，及び凸領域に図形の内部があるベジエ曲線（凸ベジエ曲線）の始点と終点をそれぞれ結ぶ多角形である。例えば，(a)のベクトルデータからは(d)に示すような三つの多角形が得られる。

続いて，直線セグメントの内部が三角形に分割される。例えば，(d)の直線セグメントは(e)に示すような三角形の集まりに分解される。

3.2　GPU処理

GPU処理では，曲線セグメント内部の画素ごとに，ベジエ曲線の凹領域に属するか，あるいは凸領域に属するかが判定される。そして，判定された領域が図形の内側領域に相当する場合だけ，その画素がカラーバッファに描画される。一方の直線セグメントについては，常にその全領域がカラーバッファに描画される。

例えば，図3の(c)の曲線セグメントを(f)のように描画すると(g)が得られる。また，(e)の直線セグメントを描画すると(h)が得られる。これら二つの結果を合わせると，最終的に(i)の描画結果が得られる。

3.3　問題点

このように，従来手法では，重複する曲線セグメントを見つ

（注1）　曲線の表現方式の一つで，始点と終点を固定し，制御点によって曲がり具合を調節する。

けて細分割する処理と，直線セグメントの内部を三角形に分割する処理をCPUで行う必要がある。表示するベクトルデータがあらかじめわかっている場合はこれらの処理を一度だけ行えばよいが，ベクトルデータが動的に与えられる場合は，そのつどこれらの処理を行って三角形メッシュに変換し直さなければならないため，CPU処理に手間取ってしまい，表示速度が遅くなる問題が生じてしまう。

4　提案手法のアルゴリズム

次に，提案手法のアルゴリズム[2]を説明して，従来手法の問題点を解決できることを示す。図2(b)に示すように，提案手法のアルゴリズムも，従来手法と同じようにCPU処理とGPU処理の大きく二つに分かれている。

4.1　CPU処理

CPU処理では，与えられたベクトルデータが直線セグメントと曲線セグメントに分割される。ただし，提案手法における直線セグメントは，ベクトルデータに含まれる各閉路において，黒丸で示した点の中から任意に選択した基点と残りの黒丸の点を結んでできる，扇状の三角形群（三角形ファン）で表される。

もう一方の曲線セグメントは，従来手法と同じように各ベジエ曲線の始点，制御点，及び終点を結ぶ三角形である。例えば，**図4**の(a)のベクトルデータは(b)の直線セグメントと(c)の曲線セグメントに分割される。(b)では三角形ファンの基点を白い四角形で表している。

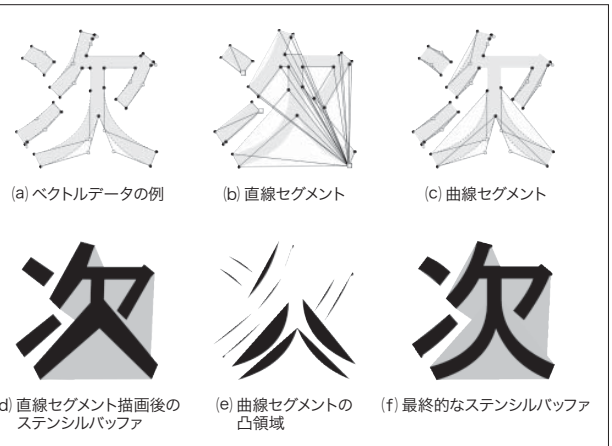

図4．提案手法の処理例 ― ベクトルデータは，直線セグメントと曲線セグメントに分けられるが，それぞれ更なる分割は不要である。GPUで最終的に描画される。

Example of vector rendering using our method

4.2　GPU処理

GPU処理では，最初に，カラーバッファの各画素への書込みの可否を表すマスク値を保持するステンシルバッファを，全画素ゼロに初期化する。その後，直線セグメントの全領域が，このステンシルバッファに描画される。このとき，ステンシルバッファの各画素にはカラー値が書き込まれるのではなく，既に書き込まれていた画素値をビット反転する操作が適用される。この結果，奇数回塗りつぶされた画素値は非ゼロになり，偶数回塗りつぶされた画素値はゼロになる。

例えば，図4の(b)の直線セグメントが描画されるとステンシルバッファは(d)の状態になる。この図では，ステンシルバッファの画素値が非ゼロの領域を黒，ゼロの領域を灰色で塗りつぶしている。

続いて，各曲線セグメントの凸領域が同じようにステンシルバッファに描画される。すると，凹ベジエ曲線の凸領域が非ゼロ（黒）からゼロ（灰色）に変わり，凸ベジエ曲線の凸領域がゼロ（灰色）から非ゼロ（黒）に変わる。例えば，(e)に示す曲線セグメントの凸領域を描画すると，ステンシルバッファの内容は(d)から(f)に変わる。(d)から余分な凸領域が削り取られ，不足していた凸領域が新たに付け加えられている。

最後に，ベクトルデータ全体を覆う大きな2枚の三角形がカラーバッファに描画される。このとき，カラーバッファの各画素への書込みに先立ち，対応するステンシルバッファの画素値がチェックされ，その値が非ゼロである場合だけカラーバッファへの書込みが許可される。この結果，(f)の黒領域だけにカラー値が書き込まれることになり，“次”を正しく表示することができる。

4.3　特長

図2の(a)と(b)のアルゴリズムを比較するとわかるように，提案手法では，曲線セグメントの細分割と直線セグメントの三角形分割が不要であり，従来手法に比べてCPU処理のコストが格段に低い。したがって，ベクトルデータを三角形メッシュに変換する処理に手間取ることはない。

CPU処理のコストが低くなる分，GPU処理ではステンシルバッファを介した2段階の描画を行わなければならなくなるが，この処理はGPUのハードウェア機能を用いて高速に実行することが可能である。そのため，ベクトルデータが動的に与えられたとしても高速に3次元表示することが可能である。

5　提案手法の3次元ユーザーインタフェースへの応用

提案手法のアプリケーションとして，**図5**に示すような3次元電子番組表（EPG）ブラウザを試作し，ハードディスクに録画されている大量のテレビ映像を効率良く閲覧するための新しいユーザーインタフェースを実現した。

このブラウザでは，テレビ番組のタイトルや出演者などのテキスト情報と動画サムネイルが同時に表示される。ブラウザの縦軸は時間軸を表し，横軸はチャンネルを表している。提案手法を用いて，EPGに含まれている数千個のTrueTypeテキストを3次元表示することにより，あたかも新聞のテレビ欄

(a) ズームアウト映像

(b) ズームイン映像

図5. 3次元EPGブラウザ ― ハードディスクに録画されている大量のテレビ映像を効率良く閲覧するための，新しいインタフェースを実現した。新聞のテレビ欄を斜め上から見下ろす感覚を再現している。

Prototype 3D EPG browser

を斜め上から見下ろしているかのような感覚を再現している。

ユーザーは，図5の(a)のようにズームアウトした状態で番組表の上を自由に動き回り，ハードディスクに録画されているテレビ映像を次々とチェックしていく。気になる番組が見つかれば，(b)のようにズームインしてより高解像度のテレビ映像を視聴することができる。提案手法によって3次元表示されたTrueTypeテキストの品質は解像度に依存しないため，ズームインした場合でもテキストのエッジを美しく保つことができる。

EPGのような大量のテキストデータを従来手法で3次元表示するためには，すべてのテキストのアウトラインに前処理を施し，その結果を保存しておかなければならないため，大きな空きメモリスペースが必要になる。これに対して，提案手法は前処理を行う必要がなく，付加的なメモリスペースを消費せずに大量のテキストを動的に3次元表示することができる。

6 あとがき

グラフィックス専用プロセッサを活用して，Flashアニメーションや TrueTypeフォントなどのベクトルグラフィックスを，解像度に依存せずに3次元表示する技術を開発した。従来技術で不可欠であった前処理を不要にしたことにより，動的に与えられる大量のベクトルデータを高速に表示することが可能になった。また，今回開発した技術を応用して3次元EPGブラウザを試作し，ハードディスクに録画されている大量のテレビ映像を効率良く閲覧するための，新しいユーザーインタフェースを実現した。

今回試作したブラウザは，主にPCなど大画面向けのユーザーインタフェースであるが，今後は，携帯電話をはじめとする組込み機器の小画面向けユーザーインタフェースへの応用に取り組んでいきたい。組込み機器の分野では，消費電力の制約からCPUの処理能力が低いため，CPU以外のハードウェアを用いてベクトルグラフィックスの表示を高速化する技術が注目されており，近年，OpenVGと呼ばれる業界標準のソフトウェア仕様が策定されている。当社の技術を組込み機器に応用することを目指し，OpenVGの要求仕様をすべて満たすことができるようにアルゴリズムの拡張に取り組んでいく。

文 献

(1) Loop, C. ; Blinn, J. Resolution Independent Curve Rendering using Programmable Graphics Hardware. In Proceedings of ACM SIGGRAPH. 2005, p. 1000－1010.

(2) 爰島快行，ほか．“GPUを用いた動的に変形するベクトルデータのレンダリング”．Visual Computing/グラフィクスとCAD合同シンポジウム2006予稿集．船橋，2006-06，画像電子学会及び情報処理学会．p.17－22.

爰島　快行　KOKOJIMA Yoshiyuki
研究開発センター　ヒューマンセントリックラボラトリー。コンピュータグラフィックスに関する研究・開発に従事。情報処理学会会員。
Humancentric Lab.

杉田　　馨　SUGITA Kaoru, Ph.D.
研究開発センター　ヒューマンセントリックラボラトリー，博士（情報理工学）。コンピュータグラフィックスに関する研究・開発に従事。情報処理学会会員。
Humancentric Lab.