111　グラフィックシステムプロセッサを利用したＶｏｘｅｌ処理システム（第一報）

○菊地慶仁、深谷健一（北海学園大学）、岸浪建史（北海道大学）

１．はじめに

形状表現の手法としては、連続的な３次元幾何モデルによるものが一般的である。一方非連続的なＶｏｘｅｌデータによる形状表現は、３次元のモデリング対象の内部構造を表現する目的で、医療用や科学計算、形状操作などの分野で用いられつつある。（図１）

Ｖｏｘｅｌデータによる形状表現は、データ操作そのものは単純なアルゴリズムで行えるが、大量のメモリと、操作に必要とされる時間が問題とされており、ハードウェアによる実現が望まれている。

近年、ワークステーションのビットマップディスプレイ用等で、２次元のメモリ空間に対する操作命令を備えたＣＰＵが開発されてきており、グラフィックシステムプロセッサ（以下ＧＳＰ＝Graphic System Processor）の名称で使用されている。

本研究では、Ｖｏｘｅｌ操作と操作結果の画像生成の高速化を目的として、これらのＧＳＰの機能をソフト的に３次元のメモリ空間に拡張し、Ｖｏｘｅｌ操作システムとして実現することを試みるものである。

２．ＧＳＰの機能の特徴

ＧＳＰの機能の例として、米ＴＩ社製ＴＭＳ３４０１０に関して述べる。

（１）命令キャッシュ付き３２ビット汎用プロセッサをコアに持ち、汎用命令とグラフィックス固有の命令を使用することが出来る。

（２）グラフィック命令の操作アドレスとしてスクリーンの座標系のように、２次元のＸＹ座標によってピクセルの位置を指定することが出来る。（図２）

命令実行はＧＳＰ側でＸＹアドレスからリニアアドレスに変換して行なわれる。この際、グラフィック命令で操作するピクセル当りのビット数を設定することが出来る。

（３）ラスタ演算、ピクセル処理を含む、２次元メモリアレイに対する高速な転送命令、ＰＩＸＢＬＴ（Pixel Block Transfer）とＦＩＬＬ（塗りつぶし）命令を持っている。一般的にｂｉｔｂｌｔ（BIT BLock Transfer)と呼ばれる。

（４）ＤＲＡＶ（Draw And Advance）命令とＬＩＮＥ命令を持っており、直線描画を高速に行なうことが出来る。

（５）ピクセル内の特定ビットに対する保護マスクを設定することが出来る。

（６）グラフィック命令実行時にウィンドウクリッピングをＧＳＰ側で自動的に行なうことが出来る。

３．テストシステムの構成

本研究では、上記のＧＳＰを登載したグラフィックシステム開発用ボードを用いてシステムの開発を行なう予定である。（図３）

このボードはＧＳＰとローカルのＲＡＭ、ＶＲＡＭを持っており、ＰＣ／ＡＴの拡張スロットに挿入して使用する。

ＧＳＰはホストからは独立して動き、ＶＲＡＭを操作した結果を直接ディスプレイに出力することが出来る。

Ｖｏｘｅｌ操作のソフトはＰＣ上のクロス開発ソフトでコンパイルする。その後デバッガソフトでボードにダウンロードし、ＧＳＰとローカルメモリの状態をモニタしながら、実行状態を監視する。

ソフトウェアの構成は、最も下位の部分にＧＳＰのグラフィック命令を実行するアセンブラルーチンを置き、これをもとに２次元のＶｏｘｅｌに対する操作ルーチン、さらに３次元Ｖｏｘｅｌに対する操作ルーチンをＣ言語で開発する。（図４）

実行結果の表示は、開発ボードに設けられているＶＲＡＭをＧＳＰで直接操作して行なう。

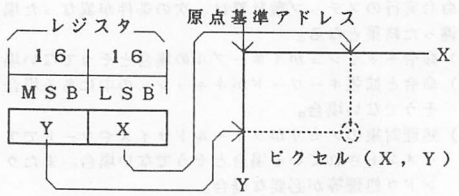

図２　Ｘ－Ｙアドレッシングの概念

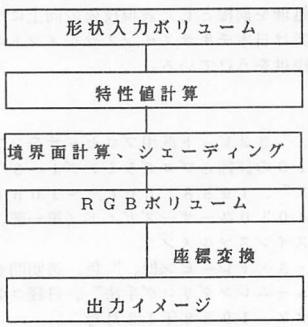

図１　Ｖｏｘｅｌデータからの画像生成手順

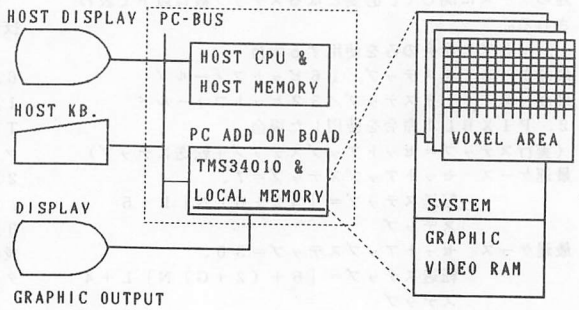

図３　テストシステムの構成（ハードウェア）

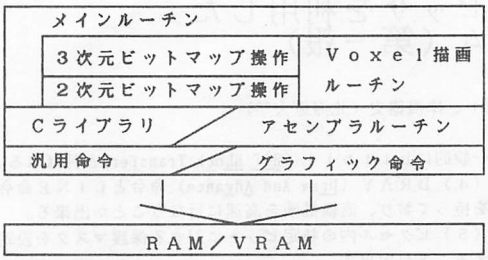

図４　テストシステムの構成（ソフトウェア）

Ｖｏｘｅｌ空間用のメモリは開発ボードのローカルＲＡＭ（約１Ｍバイト）の一部を複数個のメモリプレーンとし、１２８×１２８×６４（４ビット／１ピクセル）で５２０Ｋバイト程度を使用する。またワーク用として小規模のＶｏｘｅｌ空間を使用する。

Ｖｏｘｅｌに対する操作命令として、平行移動は基本的には各ビットマップごとの移動およびビット演算として考えられるが、回転操作は現段階ではＧＳＰに該当する命令がないのでソフト的に処理し、これにディスプレイ用の表示ルーチンが付属する。

４．パフォーマンスの推定

ここではＧＳＰを使用した場合のパフォーマンスの比較を行なう。例としてＴＭＳ３４０１０を用いた。

実時間の推定はシステムのメモリ回路の性能で転送時間に影響を受けてしまうので、ステップ数で比較を行なう。

比較は、ある特定のビットマップの転送を行なう場合を想定して、通常のメモリ転送命令と、ＧＳＰの特徴であるグラフィック命令のＢｉｔＢｌｔを使用した場合について行う。

命令実行のステップ数計算は、次の条件が異なった場合で違った結果となる。

１）命令キャッシュがイネーブルの場合とそうでない場合。
２）命令と拡張キーワードがキャッシュの中にある場合とそうでない場合。
３）処理対象のメモリがフィールドサイズやワードでアライメントされている場合とそうでない場合。またウィンドウ処理等が必要な場合。

そこで、命令キャッシュに関しては理想的な状態で実行されているとし、メモリのアライメントで決まる最速と最遅のケースに関して、必要となるステップ数は以下で表わされる。

１．ＭＯＶＥ命令のみを使用する場合
最速ケース　４ステップ／１６ビットフィールド
最遅ケース　１８ステップ／３２ビットフィールド
２．ＰＩＸＢＬＴ命令を使用した場合
（実行ステップ＝セットアップステップ＋転送ステップ）
最速ケース　セットアップステップ＝７、
　　転送ステップ＝［（２＋Ｇ）Ｎ］Ｌ＋５
　　ステップ
最遅ケース　セットアップステップ＝３０、
　　転送ステップ＝［６＋（２＋Ｇ）Ｎ］Ｌ＋４
　　ステップ

（Ｌ：転送する行数、Ｎ：一行あたりのワード数、
Ｇ：グラフィック操作で決まる固有値（２≦Ｇ≦８））

ＭＯＶＥ命令では転送するメモリが、バイト単位で区切れるアドレスかどうか、ＰＩＸＢＬＴ命令ではさらにウィンドウ処理内容で必要ステップ数が変わる。

転送するメモリの量は１２８×１２８（４ビット／Ｖｏｘｅｌ）の１プレーンとする。

メモリ全体の転送に必要なステップ数を求めると

	最速ケース	最遅ケース
ＭＯＶＥ命令	１６３８４	３６８６４
ＰＩＸＢＬＴ命令	１６３８９	４１７３２

表１　128*128 ボクセルの転送に必要なステップ数

最も効率のよい場合、差はあまり無いが、実際にＭＯＶＥ命令で転送を行なう場合は、このループの中で命令を繰り返すので、ループが終端かどうかの判定と、ジャンプ命令を実行するため、１ループごとにさらに数十ステップの実行が必要となる。

したがってＧＳＰを使用した場合、メモリ転送では数倍程度の効率アップがはかられると考えられる。

ＴＭＳ３４０１０は命令キャッシュを使用した場合、５０ＭＨｚ駆動時には１命令サイクルを１６０ｎｓｅｃで実行する。

したがって上記のＰＩＸＢＬＴ命令は約３ｍｓｅｃ程度で実行されることになる。

本システムで、他のルーチンを合わせて、一つのオペレーションにおけるＶｏｘｅｌ操作から１画面の描画までが５０ｍｓｅｃ程度で実行できれば、通常のアニメーションで行なわれている２０フレーム／秒が満足できるので、リアルタイム的なシュミレーションが実行できると考えられる。

５．まとめ

今回の発表では、以下の項目について報告を行なった。

１．ＧＳＰを利用したＶｏｘｅｌ操作システムの構造を提案した。
２．上記のシステムを使用した際のパフォーマンスの推定を行い、リアルタイム処理を行なう上での可能性を示した。

今後の課題としては、提案したシステムの実現をはかり、リアルタイム処理を前提とした表現技術の向上につとめる。

最後に本研究は日本テキサスインスツルメントの協力で技術資料等の提供をうけている。

６．参考文献

１）木村光雄、”３２ビット汎用プロセッサをコアにもつＴＭＳ３４０１０の詳細とプログラミング１～３”、”インターフェース”、１９８８　Ａｐｒ．～Ｊｕｎｅ。
２）ＴＭＳ３４０１０ユーザーズガイド（第一部、第二部）、日本テキサスインスツルメンツ
３）ロバート・Ａ・ドレーピン他、”色、透明間を自然に表わせるボリュームレンダリング手法”、日経コンピュータグラフィックス　１９８８年１２月号