グラフィクスとCAD 76－6
（１９９５．８．１７）

リアルタイム集合演算表示に関する研究

床井浩平
和歌山大学経済学部
kohe-t@eco.wakayama-u.ac.jp

本稿では，会話的な形状モデリング作業の支援を目的とした，集合演算処理の結果をリアルタイムに陰影画像表示する手法について報告する．一般に多面体に対するスキャンライン法による隠れ面消去処理では，物体形状の一様性を用いて可視面の判定を効率的に行うことができる．本手法では，スキャンライン法による集合演算表示アルゴリズムにおいて，奥行き方向の集合演算処理の際にも物体形状の一様性を利用する．これにより，集合演算表示を通常の隠れ面消去処理と同等のオーダで実行することが可能になった．

Real-Time Display of Boolean Combined Objects

Kohe Tokoi
Faculty of Economics
Wakayama University

In this report, I describe a procedure to display Boolean combined objects in a real-time. This procedure aims to support interactive geometric modeling. Scan-line hidden surface removal procedure is very efficiently because it uses the coherencies of shape of the objects to decide the visible surface. My procedure applies the shape coherency to execute Boolean operation besides hidden surface removal. Therefore, it can display Boolean combined objects as fast as normal scan-line procedure without Boolean operation.

1 はじめに

境界表現を内部表現に持つソリッドモデラにおいても，集合演算による形状定義は，会話的な形状モデリング作業において複雑な形状を定義するのに有効な手段として使われている．一般にこの操作では，ブール結合された対象形状あるいはプリミティブを空間中で位置決めした後，集合演算処理を実行して目的形状を得る．集合演算による形状変更の結果は，集合演算処理によって得られた目的形状の形状データをもとに，画面表示などを行って確認する．その際，得られた形状が意図したものと異なる場合には，一度集合演算処理を取り消して，形状変更前のデータを回復しなければならない．

そこで，形状データに対して実際に集合演算処理を適用する前に，ブール結合関係の情報を用いて，直接集合演算表示を行うことが考えられる．これには一般に視線探索法が用いられる他，一旦空間格子表現や八分木表現に変換して表示する方法などが考えられるが，いずれの方法も形状の確認だけを目的とした場合には計算コストが高い．これに対しスキャンライン法による集合演算表示は，これらに比べて効率的に陰影画像を生成でき，形状把握という目的に適した手法だと考えられる．

一方，多面体に対するスキャンライン法による隠れ面消去アルゴリズムは，ソフトウェアによってリアルタイムに隠れ面消去処理を行おうとする場合にも有効な手法である．リアルタイム3次元コンピュータグラフィックスの実現には，従来専用ハードウェアが用いられてきており，今日ではパソコン等においても3次元グラフィックス用のハードウェアが普及の兆しを見せている．しかし，マイクロプロセッサやバスの性能向上により，専用ハードウェアを用いなくとも，ソフトウェアによってある程度リアルタイム3次元グラフィックスが実現可能になりつつある．特に集合演算表示では，可視面を単純な奥行き比較だけでは決定できないため，一般に普及している単一のZバッファしか持たない専用ハードウェアにおいてこれを実現することは難しい．その点，スキャンライン法を用いれば，物体形状の一様性に基づく各種の高速化手法が集合演算処理にも適用できるため，これを通常の隠れ面消去処理と同程度の計算量で処理できる可能性がある．

本研究は，プリミティブの位置決めの際，プリミティブの移動に伴う対象形状の変形をアルタイムに画面に反映させることを，通常のパソコン等でソフトウェアにより実現することを目標とする．

2 スキャンライン法による集合演算表示

スキャンライン法による集合演算表示は，隠れ面消去アルゴリズムにおけるサンプルスパン単位の前後判定に代えて，奥行き方向の一次元の集合演算処理を行うことによって実現できる[1]．

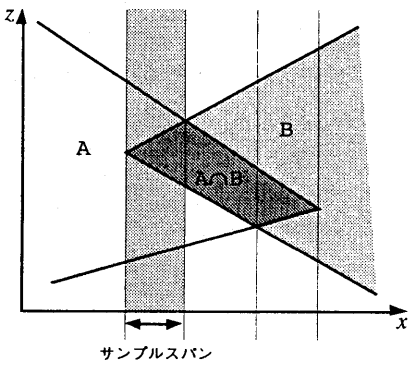


図1 サンプルスパン単位の集合演算処理

スキャンライン法による隠れ面消去処理では，物体形状の一様性を利用して可視面の決定を高速化することができるが，これは集合演算表示の際にも利用できる．すなわち，サンプルスパン単位に行う集合演算処理も，物体形状の一様性を利用して高速化することができる[1][3]．

3 スキャンライン法による集合演算表示アルゴリズム

3.1 スクリーンに関する処理

[1] 多面体を構成する面分や稜線を，それぞれ線形リストの一つのセルとして表す．また面分ごとに，その面分が処理中のサンプルスパン

上にあることを示す活性フラグを用意し，その内容を偽に設定する．

[2] スクリーン上の個々のスキャンラインに対応したバケットを用意し，その内容をすべて空に設定する．

[3] 頂点の座標をスクリーン座標系に投影した後，個々の稜線の上端のy座標値に対応するバケットに，その稜線のセルをつなぐ（バケットソート）．

[4] 処理中のスキャンラインと交差する稜線を保持するXソートリストを用意し，その内容を空に設定する．

[5] スクリーンの上端のスキャンラインから始めて，下端のスキャンラインまで，スキャンラインに関する処理を繰り返す．

3.2 スキャンラインに関する処理

[1] 現在のスキャンラインに対応するバケットから稜線リスト（侵入稜線リスト）を取り出し，個々の稜線の上端の点のX座標値を，その稜線と現在のスキャンラインとの交点のX座標値に設定する．

[2] 侵入稜線リストを，現在のスキャンラインとの交点のX座標値の順に，クイックソートを用いて並べ替える．

[3] 侵入稜線リストを，マージソートを用いて現在のXソートリストと併合する．

[4] Xソートリストが空なら，現在のスキャンラインに背景色を塗る．

[5] Xソートリストが空でないなら，以下の処理を行う．

[5-1] Xソートリストの要素の位置を示すポインタを用意し，その内容にXソートリストの先頭の稜線を設定する．

[5-2] 処理中のサンプルスパンと重なる面分を保持するZソートリストを用意し，その内容を空に設定する．

[5-3] スクリーンの左端の位置を，最初のサンプルスパンの左端の位置に設定する．

[5-4] サンプルスパンに関する処理を，ポインタが空になるまで繰り返す．

[5-5] 最後のサンプルスパンの右端とスクリーンの右端の間に背景色を塗る．

[5-6] Xソートリストに登録されている稜線に対して，下端のY座標値が現在のスキャンラインと一致するものを削除する．また残った稜線の現在のスキャンラインにおけるX座標値に，その稜線の傾きを加える．

[5-7] Xソートリストを現在のスキャンラインにおけるX座標値の順に並べ替える．Xソートリストの順序は以前と変わっていないか，あるいはその一部の順序が反転しているだけだと考えられるため，この並べ替えにはバブルソートを用いる．

3.3 サンプルスパンに関する処理

[1] サンプルスパンの右端の値に，ポインタが示している稜線のX座標値を設定する．

[2] Zソートリストが空なら，そのサンプルスパンに背景色を塗る．

[3] Zソートリストが空でないなら，以下の処理を行う．

[3-1] サンプルスパンの右端の位置において，Zソートリストの順序に変化がないかどうかを調べる．

[3-2] 変化がある場合には，サンプルスパンの右端の位置を，その位置とサンプルスパンの左端の位置の中点にとり，もう一度同じ処理を繰り返す．

[3-3] 変化がなければZソートリストに登録されている面分について，奥行き方向の集合演算を行い，可視面分を決定する．

[4] ポインタが示している稜線が，現在のサンプルスパンの右端に一致するするなら，以下の処理を繰り返す．

[4-1] その稜線が属する面分の活性フラグを反転する．

[4-2] 活性フラグが真になった面分をZソートリストに追加し，偽になったものをZソー

トリストから削除する．Zソートリストへの追加は，その面分のその位置におけるZ座標値および傾きをもとに，単純挿入法を用いてZソートリストの順序が維持されるようする．

[4-3] Xソートリストの次の稜線が空でなければ，その稜線をポインタに設定する．

[5] 現在のサンプルスパンの左端のX座標値に，右端のX座標値を設定する．

4 集合演算表示の高速化

4.1 パターンマトリクスによる集合演算の記述

集合演算の記述には通常木構造が用いられるが，集合演算表示の場合は可視面の決定の度にこの木を探索する必要があり，目的形状が多数のプリミティブから構成されている場合は，この木の探索自体に多くの処理時間が費やされる．また隠れ面消去処理アルゴリズムによっては，それ自身の特性として面同士の交差を処理できるものがある．これは隠れ面消去処理自身が和集合演算の機能を持つことを意味し，この特性を活用することによって，より効率よく集合演算処理を実行することができる．

そこで本研究では，隠れ面消去処理の和集合演算機能を活用することが容易な集合演算の記述方法として，パターンマトリクスと呼ばれる形式を用いる[2][3]．

パターンマトリクスによる集合演算の記述は次のようになる．

いま形状要素E_{ij}（これをエレメントと呼ぶ）の積集合演算による，次のような組み合わせS_jをセグメントと呼ぶ．

$$S_j = \bigcap_{i=1}^{m} E_{ij} \tag{1}$$

物体形状Tはこのセグメントの和集合演算によって定義される．

$$T = \bigcup_{j=1}^{n} S_j \tag{2}$$

$$T = \begin{pmatrix} E_{11} & E_{21} & \cdots & E_{m1} \\ E_{12} & E_{22} & \cdots & E_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ E_{1n} & E_{2n} & \cdots & E_{mn} \end{pmatrix} \tag{3}$$

(3)式は(2)式をマトリクス上に表現したものであり，これをパターンマトリクスと呼ぶ．一つの行が一つのセグメントに対応している．

なお，差集合演算はエレメントに正負を与えて表現する．

目的形状が任意の集合演算式で表現されている場合，これを加法標準型に変形することによって，パターンマトリクス上に展開できる．これは記号的処理によって容易に行えるため，目的形状が木構造で与えられた場合も本手法を適用することができる．

パターンマトリクスによる集合演算の記述から陰影画像を生成する場合，和集合演算は隠れ面消去処理自身の特性により実現できる．従って，パターンマトリクスのうち参照されるのは，式(1)によるセグメントの記述のみである．本研究ではこれをセグメント表として管理する．

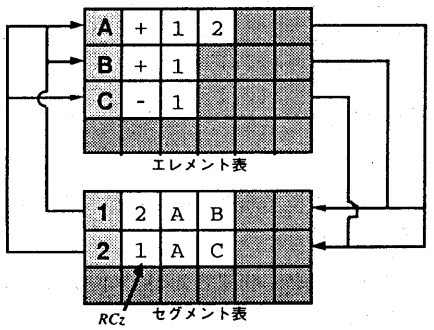


図2 （A∩B）∪（A∩C）を表す
エレメント表とセグメント表

4.2 リファレンスカウンタを用いた集合演算処理

セグメントによって表される形状は，そのセグメントに属するすべての正のエレメントの内部に存在する．この特徴を利用すれば，サンプルスパンにおける奥行き方向の一次元の集合演算処理を，単純な計数処理によって行うことができる．

リファレンスカウンタはセグメント表において一つのセグメントに属する正のエレメントの数を保持する変数(*RCz*)である．

エレメント表はそのエレメントの空間中の位置や回転角，色などの属性情報の他に，そのエレメントが属しているセグメントへのポインタを持つ．

サンプルスパンにおける奥行き方向の一次元の集合演算処理は，次のように行う．Zソートリストに保持されている面分を先頭から一つ取り出し，それがスクリーンに対して表面を向けていれば，その面分の属するエレメントが属するリファレンスカウンタをデクリメントし，裏面を向けていればインクリメントする．この処理の結果リファレンスカウンタが0になれば，その面分をそのサンプルスパンにおける可視面とする．このとき負のエレメントについては，それに属する面分の表裏を反転して処理する必要がある．

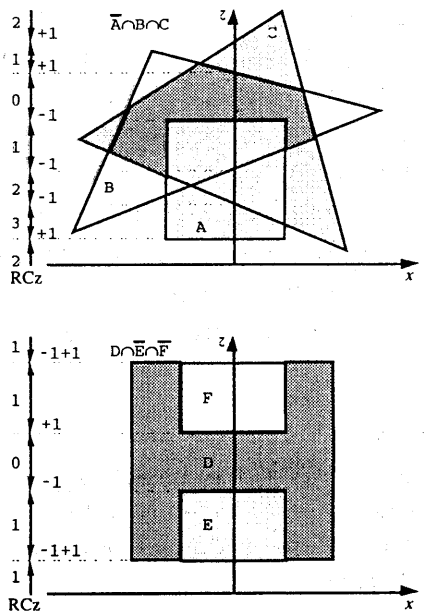


図3 リファレンスカウンタを用いた集合演算処理

なお半透明表示を行う場合は，Zソートリストを逆順（スクリーンから遠い順）に維持し，上の処理によってリファレンスカウンタが1→0に変化したとき及び0→1に変化したときのすべての面分の色を合成して表示すればよい．

4.3 サンプルスパンの類似性の利用

現在のサンプルスパンから次のサンプルスパンに処理が移る際，次の条件を満たしていれば，現在のサンプルスパンにおける可視面分が，次のサンプルスパンにおいても可視となる．従ってこの場合は奥行き方向の集合演算処理を省略できる．

- Zソートリストにおいて，現在の可視面分より前方に挿入される面分がない．
- 現在の可視面分がZソートリストから削除されない．

これにより，特に面分の交差によってサンプルスパンの分割が繰り返される場合も，集合演算処理は交差の両側においてそれぞれ1回だけですむ．さらに，交差を含むサンプルスパンの左端における可視面分をあらかじめ知ることができるため，その面分の背後にある面分の交差によるサンプルスパンの分割を回避できるほか，面分の順序の検査も可視面分の前方だけですむ．

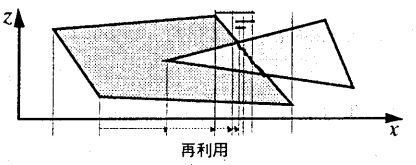


図4 サンプルスパン類似性の利用

4.4 面分の接続関係の利用

現在のサンプルスパンから次のサンプルスパンに処理が移る際，次のサンプルスパンでZソートリストに挿入される面分が，Zソートリストから削除される面分とサンプルスパンの境界となっている稜線を共有しているなら，その面分は現在の削除される面分の位置に挿入すればよい．従って，この場合は挿入位置の決定のための奥行き比較が不要になる．

さらに削除される面分が可視なら，挿入される面分も可視となるため，奥行き方向の集合演算処理を省略できる．

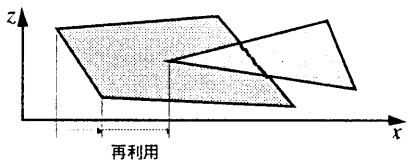


図5 面分の接続関係の利用

4.5 スキャンライン類似性の利用

現在のスキャンラインから次のスキャンラインに処理が移る際，次の条件を満たしていれば，現在のサンプルスパンにおける可視面分は，次のスキャンラインの対応するサンプルスパンにおいても可視となる．

- Xソートリストにマージされる稜線がない．
- Xソートリストから削除される稜線がない．
- Xソートリストの順序が変化しない．
- 対応するサンプルスパンがどちらのスキャンラインにおいても分割されていない．

従って，現在のスキャンラインにおいて分割によって作られたもの以外のサンプルスパンにおける可視面分を保存しておけば，次のスキャンラインにおいて対応するサンプルスパンにおける奥行き方向の集合演算処理を省略できる．

これらによって，1回の集合演算処理の結果をスクリーン上の2次元領域に拡大できるため，集合演算表示の速度を大きく向上させることができる．

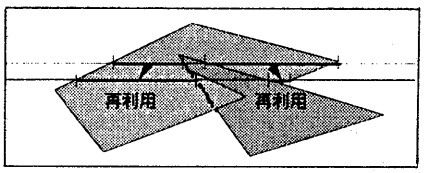
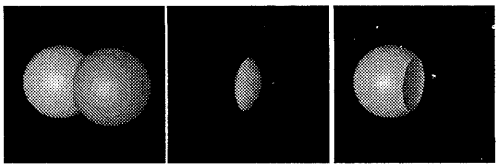


図6 スキャンライン類似性の利用

5 実験結果

以下の3つのタイプの形状について，処理時間を求めた．実験にはSONY製ワークステーションNEWS NWS-5000VI (CPU R4000/133MHz, 1MB Cache Memory, 64MB Main Memory) を用いた．



3.23秒 (10032ポリゴン/秒)　3.13秒 (10353ポリゴン/秒)　3.10秒 (10453ポリゴン/秒)

図7テスト図形と処理時間

画像の大きさは1024×1024画素で，いずれの形状も1個16202枚の面分からなる球状の多面体2個をスムーズシェーディング表示したものである．

処理時間には座標変換，法線ベクトルの算出及び陰影計算，スムーズシェーディングのための稜線の色の補間を含むが，フレームバッファへの転送時間は含まない．

6 まとめ

実験結果から，集合演算表示の場合も，ほぼローエンドのグラフィックワークステーション程度の表示速度が得られている．また，実験に使用したプログラムには不効率なコードが含まれており，この処理速度にも改善の余地が残されている．より最適化した場合の実験結果については，別の機会に報告する．

また本手法は原理的に幾何情報と位相情報の矛盾による暴走を起こさないため，集合演算表示をみながら微妙な位置合わせを行うことが可能である．この点でも，本手法は会話的な用途に適している．

参考文献

[1] Atherton, P. R.: A Scan-Line Hidden Surface Removal Procedure for Constructive Solid Geometry, *Computer Graphics*, Vol. 17, No. 3, pp. 73-82 (1983)

[2] 床井, 北橋: 凸な立体の集合演算によって定義された形状のスキャンライン法による陰影画像生成, 情報処理学会論文誌, Vol. 30, No. 1, pp. 81-90 (1989).

[3] 床井, 北橋: スキャンライン法による多面体の集合演算表示の高速化, 情報処理学会論文誌, Vol. 34, No. 11, pp. 2412-2420 (1993).