ソフトウェア クリッピングの効率化

5D−5

奥山 健一　小野 眞　室橋 茂雄
日本アイ・ビー・エム株式会社
東京基礎研究所

1 はじめに

グラフィック処理に, 一定の領域内にのみ表示を行ないその領域の外部には描画しないというクリッピング処理がある. クリッピング処理はハードウェアで実現されることが多いが, 近年計算機の性能向上に伴い, ローエンドシステムにおいてもX-Windowをはじめとする Window system をサポートしており, 多くの場合クリッピングはソフトウェアで実現されている.

矩形領域に対する線分のクリッピング処理をソフトウェアで実現する場合広く用いられている手法に Cohen-Sutherland の手法[2](以下CS法と略) と, Liang-Barsky の手法[1](以下LB法と略)がある. CS法は線分端点を符合化するため, 座標値の比較を8回行なう必要がある. またLB法は4回から8回の乗除算を行なう必要がある.

本報告では, 連続して与えられる線分の座標は一般にあまり離れていない点に着目して, 比較の手順を動的に変化させることで矩形領域に対する線分のクリッピング処理を高速化する手法について述べる.

2 線分端点の符合化

CS法では, 図1で示すように, 矩形ウィンドウの枠を元に二次元平面を9つの領域に分割, 線分の両端点がどの領域に入っているかで符合化し, 次にその符合化結果を元にウィンドウ枠との交差判定を行なっている. 表1に端点コードの組合せと線分の可視性を示す.

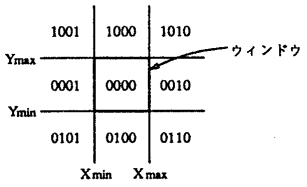


図1: 線分の端点の符合化

クリッピング対象である線分が $(x_0, y_0) - (x_1, y_1)$ で与えられたとする。CS法では, 端点の符合化のために式1に示す8つの比較を行なっている.

$$\begin{array}{llll} x_0 < x_{min}, & x_0 > x_{max}, & x_1 < x_{min}, & x_1 > x_{max}, \\ y_0 < y_{min}, & y_0 > y_{max}, & y_1 < y_{min}, & y_1 > y_{max}. \end{array} \quad (1)$$

しかし, 実際の符合化において, 必ずしも8回の比較は必要ない. 例えばある端点 (x, y) が $x < x_{min}$ を満たした場合, $x > x_{max}$ との比較を行なう必要はない.

また, 線分によっては比較の順番を適切に行なうことで完全な符合化をすることなく不可視線分がわかる場合もある. 例えば, $x_0 < x_{min}$ が成立することがわかった場合, 次に $x_1 < x_{min}$ の比較を行なえば, 条件が成立した場合は表1にあるように, その段階で線分は不可視であるとわかるし, 成立しなかった場合でも, 符合化に必要な比較回数は最大でもCS法と変わらない.

そこで本手法では不可視線分の除去を高速に行なうことでクリッピング処理全体を高速化するために, 同一領域内にある線分を優先的に判定するように符合化手順を決定する.

表1: 端点コードの組合せと線分の可視性

	0000	0001	0010	0100	0101	0110	1000	1001	10
0000	○	△	△	△	△	△	△	△	△
0001	△	×	△	△	×	△	△	×	△
0010	△	△	×	△	△	×	△	△	×
0100	△	△	△	×	×	×	△	△	△
0101	△	×	△	×	×	×	△	×	△
0110	△	△	×	×	×	×	△	△	×
1000	△	△	△	△	△	△	×	×	×
1001	△	×	△	△	×	△	×	×	×
1010	△	△	×	△	△	×	×	×	×

○: 線分はウィンドウ内にあり可視
△: 線分はその一部がウィンドウ内にあり
　部分的に可視か, あるいは角と交差しており不可視
×: 線分は完全にウィンドウ外にあり不可視

3 符合化手順

前章で述べたように式1の評価を全て行なうのは冗長である. そこで, 本手法では図2に示すような4つの評価木を用意する. 評価木 A と評価木 C, 評価木 B と評価木 D はそれぞれウィンドウ枠のX軸およびY軸方向の評価木であり, 互いに置換えが可能である. これらを互いの"配遇木"と呼ぶことにする.

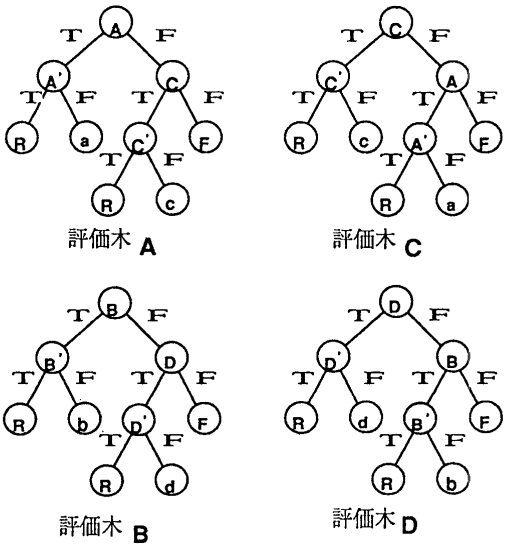
これらの評価木 A,Cから1つ,B,Dから1つ選び, それを組合わせることでクリッピングを行なう. 例えば評価木 A と Cを選び, さらに評価木 A を先に用いることにした場合,

1. 評価木 A を評価した結果 Rに到達したら, その線分は不可視
2. a,c または F に到達した場合,
   - i 評価木 Cを評価した結果 Rに到達したらその線分は不可視
   - ii b,d または Fに到達した場合, ウィンドウの指定された枠との交点を計算し, 線分の可視部分を算出

となる. 他の組合せでも同様に評価を進める。

可視部分の算出法はCS法を元に行なう. 評価木の通過パスによって『未評価』の箇所はわかっている. よって可視部分の端点計算には最大でも4回の乗除算で処理できる.

An efficient software clipping method
Kenichi OKUYAMA, Makoto ONO, Shigeo MUROHASHI
Tokyo Research Laboratory, IBM Japan, Ltd.



A: $x_0 < x_{min}$, A': $x_1 < x_{min}$
B: $y_0 < y_{min}$, B': $y_1 < y_{min}$
C: $x_0 > x_{max}$, C': $x_1 > x_{max}$
D: $y_0 > y_{max}$, D': $x_1 > x_{max}$
a: x_{min}と交点計算 b: y_{min}と交点計算
c: x_{max}と交点計算 d: y_{max}と交点計算
R: 線分は不可視
F: 可視線分だが符合化が完全に済むまでは不明

図 2: 冗長な判定を省くための 4 つの評価木

4 評価木の組合わせ手法

一般に CAD(Computer Aided Design) などの世界では線分の数は数万本以上で, それぞれが連続している場合が多い. このため与えられる線分はあまり離れていない点が連続すると想定できる. 従って, 連続した線分は評価木の同じパスを通りやすく, 線分を評価する度に評価するパスが短くなるように評価木の順序を変更すればクリッピング処理全体では評価回数を減らすことができる.

変更には次の 2 つの規則を用いる.

規則 1 R に到達した場合, その評価木を次からは先頭にする. 例えば, 評価木を $A \rightarrow B$の順に用いており, 評価木 Bで Rに達した場合, 次からは評価木を $B \rightarrow A$ の順に用いる.

規則 2 各木に 1 つずつ存在する 3 つ目のノードで評価を行なった場合, その木の配偶木と入れ換える. 例えば, 評価木 A を用いていて, C'を評価した場合, 次回からは評価木 Cを用いる.

これらの変更は, 関数へのポインタを付け変えるという処理によって容易に実現できる.

5 実験結果

本手法を CS 法, 及び LB 法 と比較した結果を図 3 に示す. 線分の数は 16384 本とし, 1, 2, 4, 8, 16, 32 本の連続する線分が合計で 16384 本になるようにデータを乱数で発生させ, 処理時間を計測した. 実験は全て, 弊社の PS-5570 を用いて行なった.

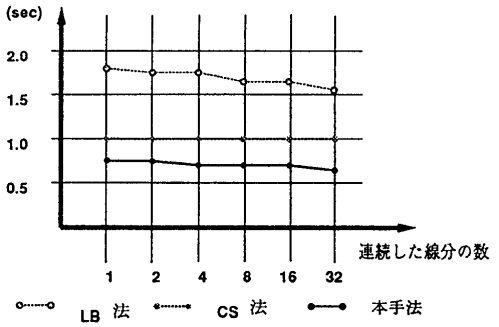


図 3: 線分のクリッピング処理に必要とした時間比較

6 考察

図 3からわかるように, 本手法は大量の線分が存在する場合, CS 法に比べ約 7 割りの時間でクリッピング処理を行なうことができる. また, 連続した線分が存在する場合は, その連続性が高いほど処理時間は減少している.

一方, 同じデータと処理プログラムを 弊社の RS/6000-320 で処理した結果を 表 2 に示す. 本機は複数の浮動小数点比較処理を並列で処理することができる. この場合は比較順序を最適化する本手法はそのメリットを失い逆に評価木を変更する手間の分だけ CS 法よりも処理時間が必要となった, と考えられる. そのため, 連続した線分を与えた場合でも処理時間に変化は現れていない. 両機とも浮動小数点除算処理は他の処理に比べて時間がかかるので除算回数の多い LB 法は他の手法に比べ処理時間が多くなったと予想される.

表 2: LB 法,CS 法と本手法を RS/6000 320 上で比較した結果

線分の連続数	LB 法 (sec)	CS 法 (sec)	本手法 (sec)
1	0.15	0.08	0.09
2	0.15	0.07	0.09
4	0.15	0.07	0.09
8	0.15	0.09	0.09
16	0.17	0.06	0.08
32	0.14	0.07	0.10

7 おわりに

本報告では比較の手順を動的に変化させることで矩形領域に対する線分のクリッピング処理を CS 法の約 1.4 倍の処理速度で行なう手法について述べた.

本手法はクリッピングだけでなくソフトウェアピックにも利用できる. ピック処理は “大量の線分から小数の線分を選択する” 操作なので, 与えられる線分のほとんどが符合化結果だけから除去できる. 従って不要な符合化を減らす本手法を用いることで, ピック処理全体を大幅に高速化できる.

参考文献

[1] Liang,Y. and Barsky,M.: “A New Concept and Method for Line Clipping”, ACM TOG, Vol. 3, No. 1(Jan. 1984) pp.1-22.

[2] Foley,J. et al.: “Computer graphics: principles and practice – 2nd ed.”, Addison-Wesley Publishing Co., 1990.