〔解　説〕

CG ソフトのアルゴリズム

江戸川大学メディアコミュニケーション学部情報文化学科　　ザ ン　ピ ン

筆者は今まで主に組み合わせ最適化の領域を専門として研究活動を行ってきた。ここ数年は，CG 関連の授業を担当し，CG の基礎理論と演習（制作）に携わり始めた。ここでは，新しい分野に入るにあたって感じたこと，さらに，最適化分野との違いについて述べる。

1.　アルゴリズムを意識する例

写真を修正して見栄えを良くする画像処理，説明図やイラストを作成したいばあいなど，一般的に 2 次元ソフトを使う。筆者も以前から，ペイントのほか，Adobe 社の Photoshop や，Illustrator などを使ってきた。これらのソフトに関しては，コンピュータが離散的なディジタル装置であることを忘れさせるほど，操作と人間の感覚がマッチしていた。そのため，ソフトのアルゴリズムの特徴など余り気にせずに使用できた。

しかし，最近筆者が CG ソフトを使うようになって，アルゴリズムを強く意識するケースに何回も出会った。

① JW-CAD の例

一番目は CAD ソフトの包絡処理である。**図 1.1** に示すように，クリックとドラッグだけで違う形への編集結果が得られる。思ったより簡単な操作で，目的とする編集結果が得られるのが意外であった。

② 3D ソフト Shade の例

図 1.2 に示すように円と長方形を描き，次に，図面（キャンバス）には何も手を加えず，**図 1.3** のようにブラウザ（ライブラリ）でパーツを移動する。

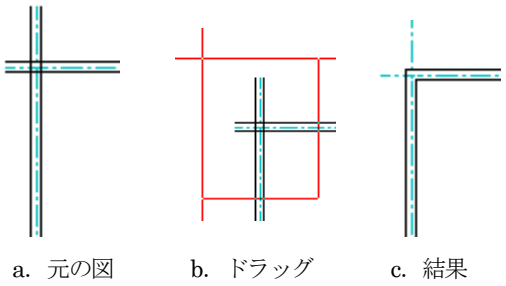

図 1.1　CAD の包絡処理

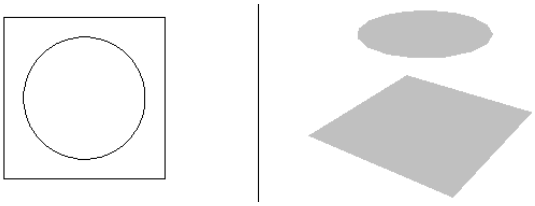

図 1.2　上面図と透視図

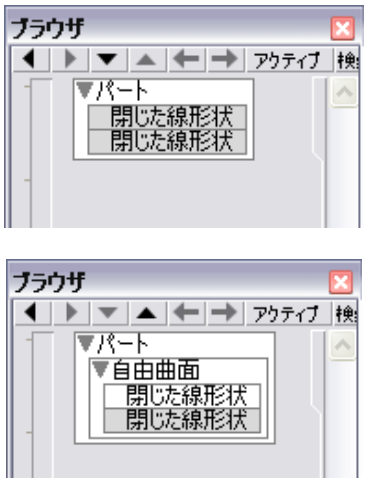

図 1.3　ブラウザウインドウの操作

すると，**図 1.4** のような曲面ができる。

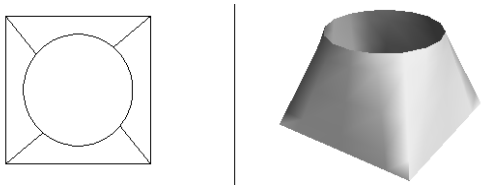

図 1.4　ブラウザ操作後の上面図と透視図

この操作で得られた結果は，前述の「人間の感覚」と少しズレがある。つまり，始めてこの操作をする者が,操作の結果を前もって想像できない。

このような感覚のズレは，筆者だけではなく，授業を受けている学生も同じような意見を述べていた。特に 3D のばあい，レンダリング（見えない隠線，隠面の消去，材質感などを出す）の結果が思ったことと違うばあいなどである。

2.　コヒーレンス

ここから，CG アルゴリズムの特徴を述べる。

代表的な特徴はアルゴリズムを考案する段階で効率性を取り入れ，貫くことであろう。その特徴をよく表わす言葉にコヒーレンス（coherence）がある。つまり，近傍の連続性および類似性を利用し，計算の効率を図ることである。

CG で代表的な処理である隠線，隠面消去について考えよう。

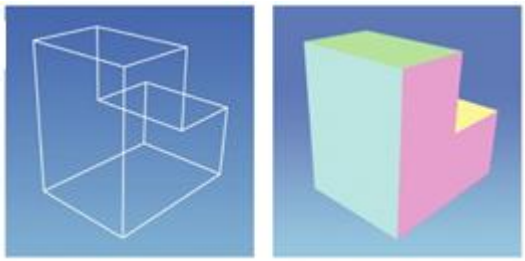

図 2.1　隠面消去

隠面消去の代表的なアルゴリズムの一つZバッファ法は，ポリゴン（画素）までの視点からの距離，すなわちZ値を，すでにZバッファ（スクリーンに画像を表示するための一時的な保存場所）に格納されているZ値と比較する。Z値が小さければ（すなわち手前であれば）バッファのZ値を書き換え，物体のZ値を記憶する。

このような手順で，すべての画素に対して同じ作業をするのは効率が悪い。そこで，スキャンライン法が提案されている。つまり，スキャンラインと交差する交点を求め，その間の画素をまとめて求めることにより効率を上げる方法である。

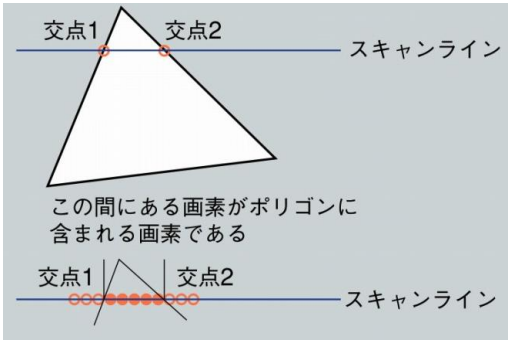

図 2.2　スキャンラインごとの変換法

さらに，交点を求める際には，一般には除算を行う必要があるが，ここにもコヒーレンスを利用し，増分法を用いている。増分法では，加算のみで済ませることができ，計算時間を節約することができる。

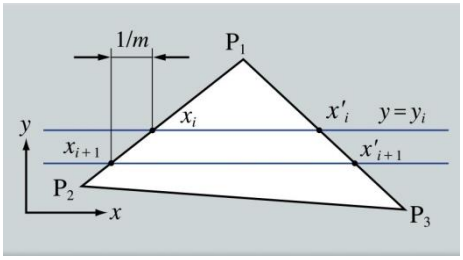

図 2.3　増分法による交点と奥行きの計算

図 2.3 のように, 上から下へと処理するばあい，あるスキャンラインと三角形の 1 辺との交点を x_i とすると，その次のスキャンラインと辺の交点は次式によって求められる。

$$x_{i+1} = x_i - 1/m$$

ここで，m は辺の傾きである。

3.　広義コヒーレンス

以上は狭義のコヒーレンスである。著者は以下

の処理もコヒーレンスの一種と考え，広義コヒーレンスと呼ぶことにする。

まず，同じく隠面処理を例にする。

平面状のスキャンラインではなく，視線（レイ）により線分上の隠“点”消去として隠面消去を実行するのがレイトレーシング法である。

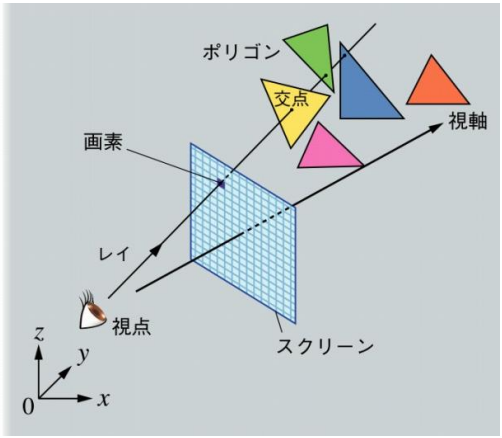

図 3.1　レイトレーシング法

レイはスクリーンを構成するすべての画素に対して放射され，そのそれぞれのレイについて，すべてのポリゴンとの交差判定を行わなければならず，計算時間は膨大である。ここで持ち込まれた高速化方法にはバウンディングボリュームと空間分割法がある。

バウンディングボリュームでは，各物体を，球や直方体のようなバウンディングボリュームとよばれる簡単な形状で囲っておく。レイとバウンディングボリュームが交点を持つときだけ，そのバウンディングボリュームに含まれている物体について交差判定を行う。

空間分割法は，あらかじめ 3 次元空間を格子状に分割し，どの空間にどの物体が存在するかをボグセル空間とよばれる 3 次元配列により記憶しておく。レイとの交差判定においては，そのレイが通過する領域（ボグセル）に存在する物体のみと交差判定すればよい。

さらに，よく知られている四分木の手法も広義コヒーレンスのひとつと考えられる。すべての画素を記憶せず，領域を縦横それぞれ 2 等分に分割

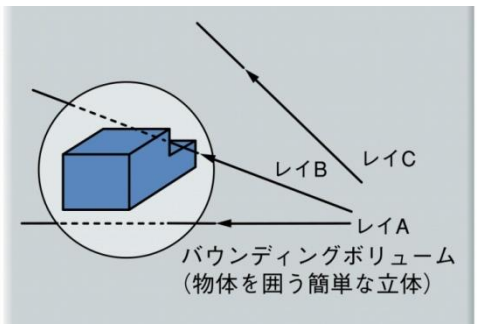

図 3.2　バウンディングボリューム

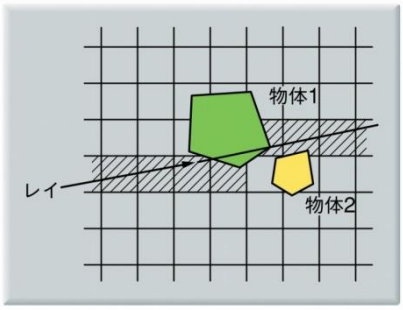

図 3.3　空間分割法

a. 2 次元図形の例　　b. 四分木による表現

図 3.4　四分木による表現の例

し，領域と図形が交差する部分だけを再分割して記憶する。

4.　コヒーレンスの完全融合

最後に，CG アルゴリズムそのものである簡単な線分の描画アルゴリズムを見よう。

一般に，xy 平面上の 2 点 $P_1(x_A, y_A)$，$P_2(x_B, y_B)$ を通る線分は次式で表わされる 。

$$y = \frac{\triangle y}{\triangle x}(x - x_A)$$

ただし、$\triangle x = x_B - x_A, \triangle y = y_B - y_A$

最も直感的なアルゴリズムは，画素の x 値を増加させ，線分の勾配により，誤差が 0.5 以下で画

素の y 値を決める。

しかし，計算処理をより高速に行うために広く利用されているのが次のブレゼンハムのアルゴリズムである。それは実にエレガントで，コンピュータサイエンスならではのアルゴリズムであると筆者は思う。

ブレゼンハムのアルゴリズム

$d_1 = 2\Delta y - \Delta x;$
始点（x_A, y_A）の画素をonにする；
$c_1 = 2(\Delta y - \Delta x); c_2 = 2\Delta y$
x_iを$x_A + 1$からx_Bまで１ずつ増加させて
次に処理を繰り返す；
{もし、誤差$d_i > 0$ならば、
$y_{i+1} = y_i + 1, d_{i+1} = d_i + c_1;$
そうでなければ、
$y_{i+1} = y_i, d_{i+1} = d_i + c_2;$
画素（x_{i+1}, y_{i+1}）をonにする
}

図 4.1 は，始点が（0, 0），終点が（11, 4）の例である。**図 4.2** は比較のために与えられた誤差 0.5 以下の例である。

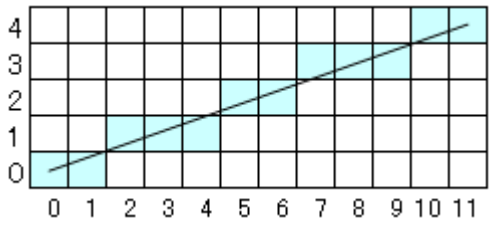

図 4.1　ブレゼンハムのアルゴリズム

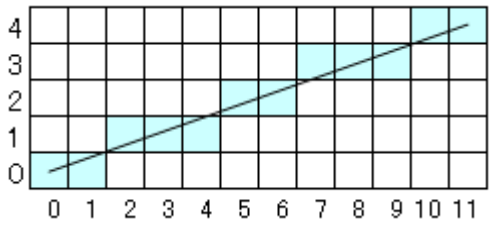

図 4.2　誤差が 0.5 以下のアルゴリズム

ここでの計算結果は同じである。しかし，アルゴリズムが洗練されているという点で，ブレゼンハムのアルゴリズムの方が素晴らしい。

筆者の専攻分野である組み合わせ最適化にも，コヒーレンスのような処置もある。しかし，最適解の条件を証明し，最適解の停止条件などアルゴリズムに入る前の準備が多数あり，多くのばあい，CG アルゴリズムの方が，まだ直感性が高い。また，最初に数値計算レベルのコヒーレンスを考慮に入れることはあまり多くない。

参考文献

1. Computer Graphics, 技術編　CG 標準テキストブック，CG-ARTS 協会，1999 年
2. コンピュータグラフィックス，CG-ARTS 協会，2004 年