勉強会レジュメ
並列画像処理について

吉谷 直樹

平成８年５月１日

1　序論

図学の目的は, 物体の形状属性を他人に伝えるための表現技術である。この分野にコンピューター処理を導入することによってより複雑な形状を他人に伝えることが可能となり, その代表的な応用である設計生産応用はＣＡＤ／ＣＡＭと名づけられ画期的な有用性を発揮している。コンピューターで単純な図形処理をするのにも強力な計算能力を必要とするが, このように自然科学を中心とする世界をシュミレートし, それを可視化して人が容易に理解できる技術をコンピューターで実現することは, 新しいアルゴリズムを必要とする, と同時にコンピューター自体に対してより強力な計算能力を要求する。この要求を満たす技術として並列処理が知られている。

1.1　図形処理の歴史

- 第１世代 ：(真空管）
- 第２世代 ：(トランジスタ）ワイヤフレームモデル, スキャンライン法
- 第３世代 ：(ＩＣ）サーフェスモデル, スムーズシェーディング法
- 第４世代 ：(ＬＳＩ／ＶＬＳＩ）ソリッドモデル, レイトレーシング法, ラジオシティ法
- 第５世代 ： (マルチプロセッサ）実時間での４次元モデル

1.2　図形処理の応用

- 工業生産 :設計支援 （ＣＡＤ／ＣＡＭ）, 高精度が要求される。
- 図形, 画像合成 :雑誌, アニメーション, 映像生成, 作成者の概念, 感性を十分表現するに足る画像品質が要求される。
- ヒューマンインターフェース :技能向上, 自然科学全般の計算結果の可視応用, ＧＵＩ, 応答特性が重要となる。

2　図形処理の基礎

2.1　２次元グラフィックス処理

2.1.1　直線と円の描画

ラスタスキャン型図形処理装置では、図形処理アルゴリズムによって得られたピクセルデータをフレームメモリに格納し、ディスプレイコントローラが画面走査に合わせ, フレームメモリから読み出したピクセルデータに対応する信号をディスプレイに供給する

- ピクセル :フレームメモリに格納されるデジタル画像の１画素に対応し, データとして輝度や色を表すデジタル値を持つ。
- スクリーン座標系 :スクリーン上の位置を２次元 x、y 座標系で表現

Bresenham のアルゴリズム

1. 直線描画 :始点 (x,y) と終点 (x+dx,y+dy) によって定義される線分を描画
    - アルゴリズム１ :始点から終点に向かい,x が１増加するごと

に y を計算しそれを四捨五入した値を求め、１ピクセル描画する。

```
/* 2 端点 \\(x,y),(x+dx,y+dy) をもつ線分を描画する関数\\
  (dx>dy>0 とする )*/
Draw_line(x,y,dx,dy)
int x,y,dx,dy;
{int i;float d,e;
 d=(float)dy/(float)dx;
/* x を１増したときの y の増分 */
 e=d-0.5;
 for(i=0;i<=dx;i++){
  point(x,y);
  if(e>0){y+=1;e-=1.0}
  x+=1;
  e+=d;
  }
}
```

リスト１. 描画アルゴリズム１

- アルゴリズム２ :アルゴリズム１で普通に y を求めると浮動小数点の除算が必要となり、効率が悪い。それを改善したもの。

```
Draw_line(x,y,dx,dy)
int x,y,dx,dy;
{int i,e;
 e=2*dy-dx;
 for(i=0;i<=dx;i++){
  point(x,y);
  if(e>0)
     {y+=1;e+=2*(dy-dx)}
  else
     {e+=2*dy;}
  x+=1;
  }
}
```

リスト２. 描画アルゴリズム２

2. 円の描画 :円の１／８の円弧 ($x^2+y^2=r^2$かつ $0\leq x<y$) を描画するアルゴリズム。他の部分は対象性により拡張できる。
   - アルゴリズム３ :(0,r) から (r/$\sqrt{2}$,r/$\sqrt{2}$) まで x 座標を１増す毎に１ドット描画する。y 座標としては$\sqrt{r^2-x^2}$の四捨五入した整数値を選ぶ。(x_p,y_p) まで描画した時、$x=x_p+1$ での y 座標は、点 $(x_p+1,y_p-1/2)$ が円の内部なら y_p, そうでないなら y_p-1 である。評価関数としては $F(x,y)=x^2+y^2-r^2$を使い、$d_p=F(x_p+1,y_p-1/2)$ が負なら y_p, そうでないな

ら $y_p - 1$ である。

- アルゴリズム 4 :アルゴリズム 3 から浮動小数点の除算を除くよう改善したもの。

2.1.2 図形の移動, 変形

2 次元図形に対する基本的な幾何学操作として，平行移動，回転，スケーリングがある。これらの操作は，同次座標を用いた 1 次変換の行列演算で行なう。

2.1.3 ビューイング変換

ビューイング変換とはワールド座標系で定義された図形をスクリーン座標へ写像する変換である。ビューイング変換の中でも特に良く使われるのがウィンドウ変換である。ウィンドウ変換とは，ワールド座標系に定義されたウィンドウをスクリーン座標系に定義されたビューポートへ写像する変換であり，回転操作を伴わない。

- ワールド座標系 :一般に図形を定義している座標系で，スクリーン上の表示位置を表すスクリーン座標系と区別される
- ビューポート :スクリーン上の表示位置

2.1.4 クリッピング

クリッピングとは，定義された図形に対して，ウィンドウ内部の表示されるべき部分を求める処理である。ウィンドウを 4 本の直線 $x = x_{left}, x = x_{right}, y = y_{bottom}, y = y_{top}$ で囲まれる領域としたとき，これら境界の直線をクリッピングエッジという。

1. Cohen&Sutherland のアルゴリズム :前処理として線分の端点がウィンドウとどのような位置関係かを表すコードを各端点について計算。コードは 4 ビットで表され、各ビットはそれぞれウィンドウの上下左右を表す。次に線分の 2 端点のコードを求め、ともに 0000 なら完全にウィンドウ内にあるので線分を出力する。そうでないなら、ビットごとの論理積を求め、0000 なら 完全にウィンドウ外にあるので終了。そうでないなら、各クリッピングに対するクリッピングを実行する。もし、交差することが分かれば、交点を求め、新しい端点としてその端点を持つ線分に別のクリッピングとのクリッピング条件を判定する。

2. ポリゴンのクリッピングアルゴリズム :頂点列で表されたポリゴンを入力とし、そのポリゴンを各クリッピングエッジに対し順番に処理する。キューに格納されたポリゴンの頂点列 $\{P_0, ..., P_i, ..., P_n\}$ から頂点 P_i を 1 つずつ取り出し、現在着目しているクリッピングエッジ e_j に関して頂点 P_i が表示される側にあるか調べ、表示される側にあれば P_i をポリゴン Q の頂点として出力。さらに P_i と P_{i-1} がクリッピングエッジに関して同じ側かを調べ、異なる側にあれば、辺 P_iP_{i-1} と e_j との交点を求め、Q の頂点として出力。これを各エッジに対して行なう。

2.1.5 ポリゴンのスキャンコンバージョン

内部領域を持つ閉じた 2 次元図形を描画する方法

- Y-X アルゴリズム :前処理として、y バケットリストと呼ばれるポリゴンの辺を y 座標の最大値でソーティングした連結リストを作成する。
  - 各辺のデータ :y 座標の最大値 y_{max} とその時の x 座標 x_{top},y が 1 減少するときの x の増分, その辺の交差するスキャンライン数
  - アクティブエッジリスト :現在処理中のスキャンラインと交差する辺のリスト

  y 座標の大きい順に走査し、アクティブエッジリスト中の交点の x 座標を更新して、そのスキャンラインと交差しない辺を除去し、そこから始まる辺を付加する。その後、辺を x 座標の小さい順にソートし、その順で走査して交点に出合う度にポリゴンに出入りすると判断してポリゴン内部の区間のピクセルを描画する。

2.2　３次元グラフィックスの幾何学処理

2.2.1　形状モデル

1. ３次元形状モデル
    - ワイヤフレームモデル :点と線により構成
    - サーフェスモデル :点と線と面により構成
    - ソリッドモデル :中身がつまった３次元物体を取り扱うモデル
2. ソリッドモデルの入力法 :３面図, モデラ（モデル構成ソフト）の提供する形状プリミティブ, その集合演算を指定したもの, 対話的指定
3. B-reps モデルと CSG モデル
    - B-reps モデル :表面を定義して物体記述
        - ウィングドエッジデータ構造 :体ノード, 辺ノード, 面ノード, 頂点ノードからなり、同種のノードは環状の双方向連結リストでリング構成
    - CSG モデル :簡単な形状プリミティブの集合演算を繰り返すことで複雑な形状を持つ物体を定義する。集合演算の階層構造を表現する木構造を持つ。

2.2.2　図形の移動, 変形

３次元図形の基本操作として平行移動，回転，スケーリングが有る。これらは２次元の場合と同様に同次座標を用いて簡潔に表現できる。

2.2.3　座標系の変換

物体が複雑な階層構造を持ち，部品を記述するのにローカルな座標系（モデリング座標）が使用されている場合には，これらの部品をワールド座標に座標変換する必要がある。この変換をモデリング変換とよぶ。またワールド座標系から視点座標系への変換を視野変換と呼ばれる。

- ワールド座標系 :３次元図形を定義する空間の座標系（右手系）
- 視点座標系 :視点を原点とし、z 軸の正方向が視線方向と一致し、y 軸の正方向がスクリーンの上方向と一致する座標系（左手系）

視点座標系の点 $P(X_e, Y_e, Z_e)$ が投影されるスクリーン上の点 (X_s, Y_s) は同次座標で次のようになる。

$$(X, Y, Z, w) = (X_e, Y_e, Z_e, 1)\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & S/\{1-(D/F)\} & S/D \\ 0 & 0 & -S/\{1-(D/F)\} & 0 \end{pmatrix}$$

D :視点とスクリーンの距離
F :視点と後方クリッピング面との距離
S :スクリーンサイズの１／２

$X_s = X/w, Y_s = Y/w, Z_s = Z/w$
３次元スクリーン座標系 (X_s, Y_s, Z_s)
正規装置スクリーン座標系 (X_s, Y_s) → ビューポート変換 → ＣＲＴ画面表示
３次元のクリッピング :２次元のときのコード４ビットをビューボリュームの６面に対応する６ビットに拡張する。

2.3　レンダリング処理

- シェーディングモデル :形状だけでなく物体が本来有している表面の光学的性質や光源を記述したモデル
- レンダリング処理
    1. 隠れ面処理 :可視となる物体面決定
    2. 影処理 :影をつける
    3. 輝度計算 :色や明るさの算出
    4. アンチエリアシング :画質改善
    5. テクスチャマッピング :2 次元のテクスチャパターンを３次元の表面に貼り付ける

2.3.1　隠れ面処理

1. Z バッファ法
ハードウェアとして物体の z 値を保持するための Z バッファを用意
ポリゴンの各面 F について F を xy 平面上に並行投影した場合の投影領域に含まれるピク

セルを列挙（スキャンコンバージョン操作）し、(x,y) に投影される F 上の点の z 値を平面方程式から $z = -(ax + by + d)/c$ により計算する。スキャンライン y の走査では、１画素右に進むと x が１増すので z 値が$-a/c$ ごとの増分計算で求まり、高速化が図れる。こうして z 値を求め、ある面で求めたピクセルの z 値があ以前の面で求めた z 値より小さければ、その点の方が視点に近いので z バッファ値および輝度を更新する。
メモリ使用量大。
スキャンライン-Z バッファ法：1 スキャンライン分の Z バッファのみを用意し、ポリゴンのスキャンコンバージョンに Y－ X アルゴリズムを使用したもの。

2. スキャンライン法 ポリゴンのコヒーレンシーを最大限利用し、処理速度の向上と記憶領域の削減を図るアルゴリズム
   上から下にスキャンライン単位の走査を行なう。現在走査中のスキャンライン平面とポリゴンの交差により決定される線分をアクティブセグメントという。アクティブセグメントの端点を x 座標の小さい順にソーティングし、端点の列を x_i としたとき、各区間 $[x_i, x_{i+1}]$ をスパンと呼ぶ。スキャンライン走査においては, 各スパンを左から順に走査し、スパン単位に可視セグメントを決定する。可視かどうかはスパンの端点での z 値を比較することで判定するが、スパンの端点での比較ではわからないときは交点を求め、可視となる部分を計算する。それで求まらないときはスパンを分割していく。

3. レイトレーシング法：スクリーン上の各ピクセルにおいて、そのピクセルを通して視点に入ってくる光線を逆追跡し光線と各物体との交差判定演算を実行して、光線と交点を持つ物体の中で一番視点に近い物体を求める。反射、透過、影処理を含む輝度計算を物体と光線との交差判定という単純な処理の繰り返しで扱える。ただし、すべての物体との交差判定にはかなり時間がかかる。
   点 (x_0, y_0, z_0) から発し、点 (x_1, y_1, z_1) に向かう光線は次のように表せる。(ただし $t \leq 0$)

   $$\begin{cases} x = x_0 + t\Delta x & \Delta x = x_1 - x_0 \\ y = y_0 + t\Delta y & \Delta y = y_1 - y_0 \\ z = z_0 + t\Delta z & \Delta y = z_1 - z_0 \end{cases}$$

   交差判定するポリゴンの平面の方程式が $ax + by + cz + d = 0$ なら
   $t(a\Delta x + b\Delta y + c\Delta z) + (ax_0 + by_0 + cz_0 + d) = 0$
   となる。t の係数が 0 なら交差しない。それ以外なら t についてとき、交点が求まる。交点がポリゴン内部に含まれるか否かは主軸方向に平行投影して 2 次元の問題に帰着して解く。

4. プライオリティ法：すべてのポリゴンを z 値によってソーティングしたリストを作成し、次に視点から見たポリゴンの順序関係をリストの奥から順に調べ、必要に応じてポリゴンの順序を入れ換えたり、分割したりして各ポリゴンに優先順位をつける。これに基づき遠いポリゴンから順にフレームバッファに書き込む。よって最終的なピクセル値は最後に書き込まれたポリゴンの輝度となる。

5. 領域分割法：スクリーン座標系でスクリーン分割を行ない分割して得られる各領域単位に隠れ面処理を行なう。一般に小領域程投影される物体は少なく問題は単純化される。
   (a) Warnock のアルゴリズム：簡単に隠れ面処理が行なえるまで１つの矩形領域を４つの矩形領域に分割する操作を繰り返す。
   (b) Weiler-Atherton のアルゴリズム：投影されたポリゴンの境界に沿ってスクリーン分割

2.3.2 光源モデル

1. 平行光線
2. 点光源：放射状に放出
3. スポットライト：有限の底面積を持つ平行光線の円筒
4. その他：線光源, 面光源, 多面体光源など

2.3.3　輝度計算

光源から出た光が物体表面で反射、透過を行ない、その中で可視物体の表面から出て目に達する光の量を計算する。
ラスタスキャンＣＲＴではＲ、Ｇ、Ｂの３本の電子銃の電圧を制御してスクリーン管面の蛍光体の発光量を調整するためのＲ、Ｇ、Ｂデータを計算する。

1. 直接光
   - (a) 反射光
     - i. 拡散反射 :不規則な粒子面に対する、さまざまな方向に入射光が反射される反射
     - ii. 鏡面反射 :金属面など規則的に配列している面に対する、入射光が法線に関して対称に反射される反射
       鏡面反射光 :反射方向と多少ずれていても表面の凹凸により目に入る反射光の一部
   - (b) 透過光
2. 間接光
   厳密には物体の相互反射や大気中の散乱光を計算
   環境光 :上の方法では時間がかかるのでどの面も一様に間接光を受けると考える。

2.3.4　スムーズシェーディング

平行光線を完全拡散面に照射した時を考える。このとき環境光と拡散反射光のみ考えればよい。
輝度計算は次式で与えられる。

$$I = I_a + I_{in} \times (\mathbf{N} \cdot \mathbf{L}) \times k_d$$
I_a :間接光による輝度成分
I_in :入射光による輝度成分
$\mathbf{N}$:法線方向の単位ベクトル
$\mathbf{L}$:入射方向の単位ベクトル
k_d :物体表面の拡散反射率

物体面が平面のときは一定の輝度値になる。曲面のときはポリゴンのパッチで近似するので輝度補間が必要となる。これを行なうのがスムーズシェーディングで、グローシェーディングとフォンシェーディングがある。

1. グローシェーディング :頂点に隣接するポリゴン面の法線の平均により頂点の法線ベクトルを求め、これにより頂点の輝度を計算する。次に頂点の輝度によってスクリーン空間でポリゴン内部の点の輝度を線形補間する。
2. フォンシェーディング :頂点の法線の補間によってポリゴン内部の点における法線を求め、それによって内部の点の輝度計算を行なう。

2.3.5　影処理

- 本影 :全く光が当たらない
- 反影 :一部の光のみ到達

1. レイトレーシング法 :各光源から可視物体面上交点に向けて光線を出し、光線を遮る物体があるか無いかを物体と光線の交差判定で判定する。存在すれば、影であるから直接光の輝度計算において光源を無視する。
2. シャドウポリゴン :光源から放射される光束と一つの多面体、またはポリゴンによって生じる半無限の空間をシャドウボリュームといい、その面をシャドウポリゴンという。シャドウポリゴンに隠れ面処理を実行し、可視点の輝度計算において視点からいくつのシャドウポリゴンを横切ったかでその可視点が影の領域であるかを判断する。
3. スキャンライン単位の影処理 :現在走査中のスキャンライン上に影を落す可能性のあるポリゴンを光源を投影中心として、スキャンライン平面と交差しているポリゴン上に投影し、投影されたポリゴンの辺（シャドウエッジ）を物体のデータとして加えておく。シャドウエッジと交差したときに影の領域に入るものとして輝度計算を実行する。
4. ２段階処理の影処理法 :第一段階で光源から見た隠れ面処理を実行し、そこでの不可視領域を影のデータとする。第二段階では視点から見た隠れ面処理を実行し、影のデータを考慮して輝度計算を行なう。

2.3.6 テクスチャマッピング

- テクスチャ :物体の表面の図柄や模様、きめ細かさ、光学的属性など
- テクスチャマッピング :2 次元のテクスチャパターンを 3 次元の物体表面に貼り付ける

テクスチャデータ (s,t) 座標↔ 3 次元表面 (u,v) 座標↔スクリーン座標 (x_s,y_s,z_s)
可視点 (x_s,y_s,z_s) に対応する u,v の値を求め、u,v に写像される s,t の値を算出し、そのテクスチャデータで輝度計算する。

2.3.7 アンチエリアシング

シャノンの標本化定理によれば、解像度が原画像の持つ最高周波数成分の 2 倍より大きくなければ原画像を忠実に再現できない。
エリアシングの現象 :シャギー (境界がギザギザの階段状になる。), ピクセル間隔より小さい物体、テクスチャが正確に表現できない。
アンチエリアシング :

- スーパサンプリング :各ピクセルについて複数サンプリング値をとり、その荷重平均で輝度決定
- エリアサンプリング :1 画素中に占める各図形の面積と位置付けから荷重平均して輝度決定

3 図形処理アルゴリズムと並列化手法

3.1 並列化に対するニーズ

一般に，最終的な画像が表示されるまでの過程において，次に示す 3 つの段階の図形処理が含まれるものと考えられる。

1. 形状モデルとシェーディングモデルを計算機上に構築するモデリング処理
2. 構築されたモデルに対する移動，回転，変形，色変更などの操作と光源設定及びビューイングパラメーター設定などの処理
3. 各種変換などの幾何学処理とレンダリング処理を行って最終的にラスタスキャン型ディスプレイに画像として出力する処理

このうち 1 と 2 の処理に関しては対話的速度が要求される。それに対して 3 の処理には実時間処理が望まれる。図形データ量の増大やディスプレイの高解像度などで処理能力への要求はさらに強まると思われる。上述の一連の処理とデータの流れは一方向が多く、各処理にハードウェアを用意し、パイプライン構成にすることが可能である。

3.2 図形処理から見た並列アーキテクチャ

1. 命令レベルの並列処理 :VLIW 方式、スーパスカラ方式、スーパパイプライン方式
2. マルチプロセッサ :
    - SIMD 方式 :単一命令流、条件分岐少なくプロセス間同期を命令ごとにとる
    - MIMD 方式 :複数命令流、条件分岐多く並列処理単位は大きい方が良い
3. パイプライン :ポリゴンの図形データを 1 つずつ取り出し、いくつかの座標変換、クリッピング、透視変換などを施していく。
4. システリックアレイ :演算ユニット (セル) に同期クロックに従って入力ポートからデータを流し込み、出力ポートからデータを取り出す。
5. ロジックインメモリ :フレームメモリを複数の演算機能つきメモリで構成

3.3 2 次元グラフィックスの並列処理

並列処理を採用した VLSI グラフィックプロセッサ
例) AGDC(NEC) :パラメータ生成、描画前処理、描画が CPU、プリプロセッサ、描画プリプロセッサによってパイプライン的に処理される。

3.4 3次元グラフィックスの巨視レベル並列処理

図形処理 :モデリング処理, 幾何学処理, レンダリング処理, ディスプレイ処理

- EWS (エンジニアリングワークステーション) :幾何学処理, レンダリング処理を受け持つグラフィックプロセッサを持つ。
- GWS (3次元グラフィックワークステーション) :物体をポリゴンパッチに分解し、それらをZバッファ法グローシェーディングで処理するのに並列処理技術を採用。幾何学処理やZバッファ法の物体に関するforループやピクセルについてのforループは並列処理可能

3.5 3次元グラフィックスの幾何学処理の並列化

例) ジオメトリシステム (IRIS) :マトリックスエンジン4個、クリッピングエンジン6個、スケーリングエンジン2個のパイプライン

3.6 Zバッファ法の並列処理

ピクセルを分担し、各単位ごとに並列処理を行なう。

1. ブロック単位分割 :画面を水平、垂直方向に分割。MIMDでの高速処理可能だが、領域ごとに計算負荷が異なり、早く処理の終ったプロセッサの割り付け必要。
2. スキャンライン単位インタリーブ :負荷分散はほぼ均等。
3. ピクセル単位 :各ピクセルプロセッサが単独でスキャンコンバージョンを行なうとコヒーレンスの利用がなく効率が悪い。担当するピクセルは現在処理中の形状の投影領域にあるかを直接判定するようにする。

3.6.1 スキャンコンバージョンの並列処理

1. ピクセルレベルのレベルの並列処理 :1つのピクセルに対するx,y,z,R,G,B計算とメモリアクセスを並列実行
2. 形状プリミティブレベルの並列処理 :辺関数(値が正か零か負かで辺の右側、直線上、辺の右側にあるかが判断できる関数) を形状プリミティブ毎に作成し、ピクセルの辺関数値を計算していく。
3. 物体レベルの並列処理 :複数の形状プリミティブに対するスキャンコンバージョンを複数のプロセッサで並列処理
4. スクリーン分割の並列処理 :前節参照

3.7 スキャンライン法の並列処理

パイプライン処理過程

1. y座標によりソーティングしたyバケットリスト作成
2. スキャンライン走査ごとにアクティブセグメントをx座標でソートし、アクティブセグメントリスト作成
3. スパン作成
4. スパン走査して可視セグメント決定

画面をy方向に分割して各々の領域の走査を複数のプロセッサに分担させ並列化する。

3.8 レイトレーシング法の並列処理

3.8.1 レイトレーシング法

1. 1本の光線に関する交差判定の回数を減らす方法
    (a) 物体データを階層型木構造で保持し、各ノードにバウンディングボリュームを付加する。
    (b) 影処理のために光源に向けて出す2次光線の交差判定を減らす方法 光源中心に表面がメッシュ状に分割された立方体を考え、メッシュに投影された物体を光源からの距離でソートしたリストを作成し、影処理ではメッシュから順に交差判定していく。

2. オブジェクト空間を部分空間に分割し、各部分空間ごとに内部に存在する物体を保持する方法
    (a) オクトリー法 :x,y,z 軸に平行な面で空間で空間を2等分 (全体で8分割) する。
    (b) ボクセル法 :ボクセル (大きさの等しい直方体) に分割。メモリをくうが、トラバース処理 (次にどの部分空間に光線が入るかの計算) が高速処理できる。
3. 交差判定処理を行なう光線の数を減らす方法 : Z バッファ法と組み合わせる方法など

3.8.2 レイトレーシング法を使用した並列処理マシン

1. １次光線がピクセル独立に出せることを利用し、画面分割による並列処理を行なう方法
2. 光線追跡が各光線について独立して実行できることに着目し、空間分割法で並列処理を行なう方法
3. 交差判定計算を並列処理する方法
4. レイトレーシングに適したアーキテクチャをもつプロセッシングエレメントを使用する方法

3.9 ラジオシティ法の並列処理

3.9.1 ラジオシティ法

相互乱反射を取り扱うレンダリング法
物体表面をメッシュに区切りパッチに分割し、各パッチが受けた光のエネルギーの一部を拡散反射により放出し、その2次元反射光の一部が他のパッチを照らす。こうして連鎖的な光のエネルギーの放出の過程を経て平衡状態に達した時、次の方程式が成立する。

$$B_i = E_i + \delta_i \sum_{j=1}^{N} B_j F_{ji} A_j / A_i$$

B_i :パッチ i の単位面積当たりのラジオシティ
E_i :パッチ i の単位面積当たりの自発光エネルギー
δ_i :パッチ i の反射率
F_{ji} :フォームファクタ (パッチ j から放出されるエネルギーのうちパッチ i に到達するエネルギーの割合
A_i :パッチ i の面積
N :パッチ総数

上式を近似で解いてすべてのパッチのラジオシティを算出した後、隠れ面処理ラジオシティに基づいた輝度計算を行なう。

3.9.2 ラジオシティ法の並列処理マシン

フォームファクタの計算の並列化を行なう。各プロセッシングエレメント (PE) はポリゴンを分担して処理し、各 PE の局所メモリにポリゴンデータを分散させ、放出を行なうパッチを担当する PE がフォームファクタを計算し、その結果を他の班にブロードキャストする方式

3.10 ボクセルレンダリング法の並列処理

3.10.1 ボクセルレンダリング法

ボリュームレンダリング :物体内部の可視化や科学シミュレーション等応用分野を主眼として開発した技術で、その1つにボクセルレンダリング技術がある。
ボクセルのデータとしては着目している属性のボクセルの中心点においてサンプリングした値を保持する。ボクセルデータの表示は通常主軸方向から見た平行投影によって行なわれる。

3.10.2 並列処理マシン

3次元フレームメモリに3次元のボクセルデータを保持する。3次元フレームメモリを多数の PE により構成し、各 PE に部分空間のボクセルデータを分散させておく。各 PE は並列に自己の部分空間に対する視点から見た2次元画像を生成し、それらの結果をマージして隠れ面処理を実行することにより、最終的な2次元画像を得る。

4 参考文献

「並列図形処理」:鷺島 敬之, 西澤 貞次 共著