■2 群（画像・音・言語）－3 編（コンピュータグラフィックス）

1 章 コンピュータグラフィックス理論

（執筆者：張　英夏・中嶋正之）［2012 年 10 月 受領］

■概要■

コンピュータグラフィックス（Computer Graphics：CG）とは，コンピュータ及びその周辺機器を用いて作成された画像もしくは映像を意味する．作成方法には，2 次元画像を入力とする手法，3 次元モデルを入力とする手法，ユーザがインタラクティブに描画する手法，入力されたデータを用いて画像を生成する手法など，大きく分けて四つの手法がある．

本章では，3 次元モデルを入力とし，その幾何情報から画像を作成する方法について述べる．

【本章の構成】

実際に画像を生成するためには，コンピュータ内に 3 次元幾何形状を保存するモデリング（Modeling），3 次元モデルに対しカメラや光源情報を考慮して画像に描画するレンダリング（Rendering）の二つの過程が必要になる．本章ではモデリング手法について 1-1 節で，レンダリング手法については 1-2 節で説明する．

■2 群 － 3 編 － 1 章

1-1 モデリング手法

（執筆者：張　英夏・中嶋正之）［2012 年 10 月 受領］

3 次元コンピュータグラフィックスにおけるモデリングとは，対象物体（モデル：Model）の 3 次元幾何形状を作成することを意味する．モデリングを行うフリーソフトウェアとしては，Blender[1]（**図 1･1**）や POV-RAY（Persistence of Vision Raytracer）[2] などが有名であり，有償のソフトウェアでは Autodesk 社の 3ds Max や Maya[3], NewTek 社の Lightwave 3D[4] などが有名である．

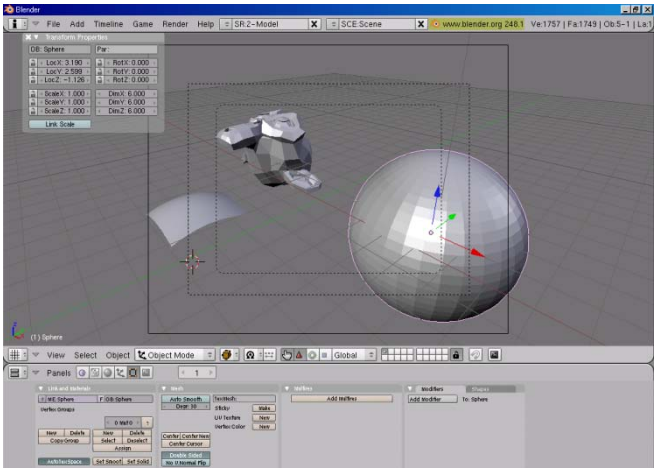


図 1･1　レンダリングソフトウェアの例：Blender

作成された幾何形状をコンピュータ内に表現する方法も様々にあり，用途によって適した表現手法を用いることができる．本節では，コンピュータ内におけるモデルの表現方法について述べる．

1-1-1 線による立体の表現

線のみで立体を表現する方法であり，ワイヤフレームモデル（Wireframe Model）と呼ばれる．データ量が少ないため，高速に描画することができる．そのため，サーフェスモデルを作成する前のシミュレーションとしても利用される．

ワイヤフレームモデルは，頂点テーブルとエッジテーブルの二つによって表現される．例えば，立方体をワイヤフレームモデルで表現したい場合，まず八つの頂点の 3 次元座標を頂点テーブルに定義する（**表 1･1**）．次に，立方体のエッジをエッジテーブルとして保存する．エッジテーブルには，**表 1･2** に示す通り，各エッジに対しエッジの始点となる頂点番号と終点となる頂点番号を保存する．

表 1・1　頂点テーブル

	1	2	3	4	5	6	7	8
X	0	1	1	0	0	1	1	0
Y	0	0	0	0	1	1	1	1
Z	0	0	1	1	0	0	1	1

表 1・2　エッジテーブル

	1	2	3	4	5	6	7	8	9	10	11	12
始点#	1	2	3	4	5	6	7	8	5	6	7	8
終点#	2	3	4	1	6	7	8	5	1	2	3	4

図 1・2 に，このワイヤフレームモデルを描画した結果を示す．ワイヤフレームモデルはデータ構造が単純でデータ量も少ない一方で，面情報をもっていないため，本来なら隠れて見えないはずの部分がどこかを計算することはできない．そのため，写実的でないという問題がある．

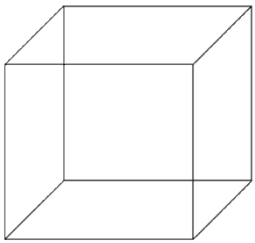


図 1・2　立方体のワイヤフレーム表現

1-1-2 面による立体の表現

3 次元平面や曲面の集合によって立体を表現することができ，このような表現方法をサーフェスモデル（Surface Model）と呼ぶ．本項ではポリゴンモデル及び曲面表現について説明する．

(1)　ポリゴンモデル

3 次元モデルの最も汎用的な表現方法として知られるポリゴンモデル（Polygon Model）は，立体を構成するポリゴンの各頂点情報と，それら頂点の組合せによる多角形（ポリゴン：Polygon）を定義することによって立体を表現する方法である．ポリゴンの形状は主に三角形もしくは四角形ポリゴンを用いるが，五角形以上の形状のポリゴンを用いることもある．ポリゴンを複数つなぎ合わせてモデルを形成するため，ポリゴンメッシュモデル（Polygon Mesh Model）と呼ばれることもある．ある物体をポリゴンモデルで近似する際に，より多くのポリゴンを用いればより精細な表現が可能となるが，後のレンダリングに時間がかかるため，用途に合わせて適切なポリゴン数を設定する必要がある．

ポリゴンモデルを保存するためのファイル形式は数多く存在する．3D-Studio ファイルフォーマットである.3ds，スタンフォード大学のポリゴンフォーマットである.ply，Alias

wavefront オブジェクト形式の.obj，Autodesk 社の AutoCAD 用ファイルフォーマットである.dxf などが代表的なファイル形式である．

いくつかの大学，ウェブサイトでポリゴンモデルを無償で公開しており，スタンフォードのレポジトリーが最も有名である [5]．

(2) 曲面による表現方法

ポリゴンを用いた形状表現では，ポリゴン数を増やせばそれだけ滑らかな表面を表現することは可能ではあるが，頂点の数が膨大になってしまう．そこで，曲面によって形状を表現する方法が存在する．曲線や曲面を生成するためには，陰関数表現（Implicit Function Representation），パラメータ形式（Parameter Representation），陽関数表現（Explicit Function Representation）によって，形状を計算する必要がある．

陽関数表現（Explicit Representation）では，曲線は $y = f(x)$，曲面は $z = f(x, y)$と表現する．例えば，原点を中心とする半径 r の球は，

$$z = \pm\sqrt{(r^2 - x^2 - y^2)}$$

のように表現される．曲線をある間隔でサンプリングするなどの場合には好都合であるが，閉曲線や閉局面の表現には不都合であるといえる．

陰関数表現（Implicit Representation）では，曲線は $f(x, y) = 0$，曲面は $f(x, y, z) = 0$ と表現する．例えば，原点を中心とする半径 r の球は

$$f(x, y, z) = x^2 + y^2 + z^2 - r^2 = 0$$

と表現される．この手法は交点計算が容易にできるという利点がある．したがって，例えば後に述べるレイトレーシングなど，交点計算が多く必要となる技術では，このような表現が適しているといえる．しかしながら，曲線や曲面状の点をある一定の間隔でサンプリングする場合には不都合であるといえる．

パラメトリック表現（Parametric Representation）とは，曲線上の点の座標をパラメータ（Parameter：媒介変数）の関数として表す形式である．例えば，原点を中心とする半径 r の球は

$$x = r\cos\theta\cos\varphi,\ y = r\cos\theta\sin\varphi,\ z = r\sin\theta$$

と表現される．パラメトリック表現は，曲線や曲面上で点列を求めることが容易であるという利点がある一方，交点計算は陰関数表現に比べると複雑になるという欠点がある．

3 次元曲面表現において最も一般的な手法はパラメトリック表現であり，エルミート（Hermite），ベジエ（Bézier），スプライン曲面などがある．

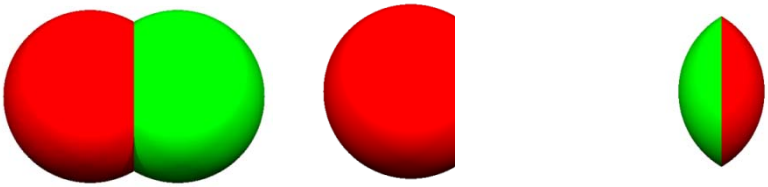
1-1-3 中の詰まった表現による立体表現

ソリッドモデル（Solid Model）は外郭面の情報のみでなく，物体の内外を区別するための情報をもっているので，物体の中が詰まっているものを表現できる．各立体に対して拡大，縮小，分割などの局所操作を繰り返して複雑な形状の立体を構成することができる．代表的なソリッドモデルとして，CSG 表現（Constructive Solid Geometry）やスイープ表現（Sweep），境界表現（B-reps：Boundary Representations）などがある．

(1) CSG 表現

CSG 表現では，直方体や円柱，多面体などの基本立体を和，積，差などの集合演算によって組み合わせることで立体を表現する．

図 1･3 に例を示す．ここでは基本立体として赤と緑の球二つを用いて，それらの和，差，積演算を施すことによって新たな立体を生成した例を示す．

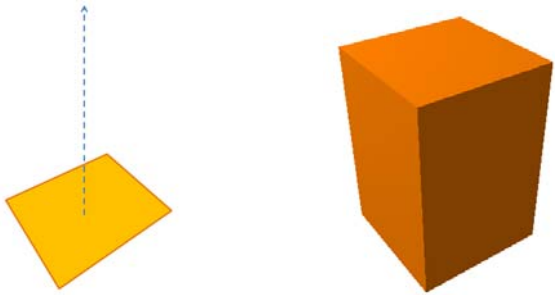


(a) 物体の和　　(b) 物体の差　　(c) 物体の積

図 1・3　プリミティブに対する演算例

(2) スイープ表現

スイープ表現（Sweep Representation）は，あるオブジェクトを軌道に沿ってスイープ（Sweep）することによって新しいオブジェクトを生成する手法を意味する．スイープ法としては，平行移動（エクストルード(Extrude)とも呼ばれる）や回転（施盤とも呼ばれる）などがあり，それらの方法によって作成されたオブジェクトを平行移動スイープ，回転移動スイープと呼ぶ．**図 1･4** に例をあげる．この例では，2 次元の四角形を矢印方向に平行移動するスイープの例を示している．この結果，右図のような 3 次元形状を得ることができる．



(a) 基本図形とスイープ方向の指定　　(b) スイープによって作成された物体

図 1・4　スイープによる 3 次元生成例

1-1-4 その他の表現方法

(1) フラクタル

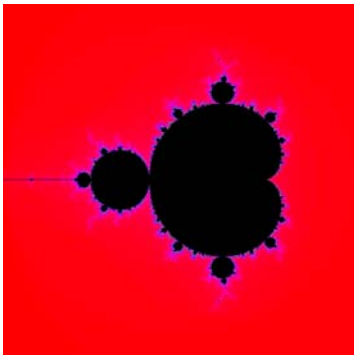
フラクタル（Fractal）とは，自己相似性（Self Similarity）をもつ形状を指す．自己相似性（Self Similarity）とは，ある図形の部分を取り出したときに，その部分形状と図形全体の形状が相似であることを意味する．このような自己相似性をもつ形状は，1900 年代始めにルイス・フライ・リチャードソン（Lewis Fry Richardson）によって初めて指摘された．後にマンデルブロット（Benoit B. Mandelbrot）によってフラクタルと名付けられ，注目され始めた．

本来のフラクタル図形はどれだけ小さい部分を取っても自己相似性をもっている必要があり，無限の再帰によって生成される図形を意味するが，CG 界では有限の再帰で生成される図形をフラクタルとしている．海岸線や雲など多くの自然形状が自己相似性をもっており，フラクタルを用いてモデリングすることができる．

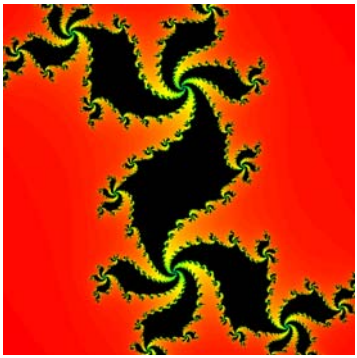
フラクタルの代表的な例としてコッホ（Koch）曲線を**図 1･5** に示す．図 1･5 左図は同じ長さの四つの線分で構成されている．各々の線分を 3 等分し，それぞれの分割されたセグメント（1/3 の長さになった線分）を元の形と同じ図形に置き換えることによって自己相似性をもつ図形を作っていくことができる．

図 1・5　コッホ曲線の例

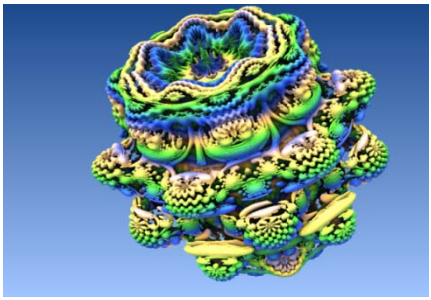
その他の有名なフラクタル図形としてマンデルブロ（Mandelbrot）集合，ジュリア（Julia）集合などがある．また，近年は 3 次元フラクタルなども存在する．**図 1･6** にその描画例を示す．



(a) マンデルブロ集合例



(b) ジュリア集合例



(c) 3 次元フラクタル

図 1・6　フラクタルの例

(2) ボクセル

空間を格子状に分割し，小さい立方体の集合として表すことができる（**図 1･7**）．この構成の最小単位となる立方体をボクセル（Voxel）と呼び，ボクセルで表された表現方法をボクセル表現と呼ぶ．ボクセルデータはデータサイズが膨大になることが問題であるが，データ構造が簡単なので集合演算を簡単に行えるという特徴がある．

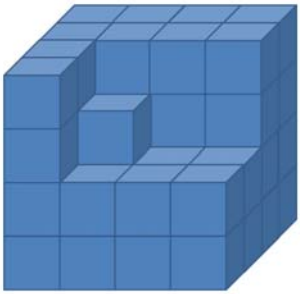
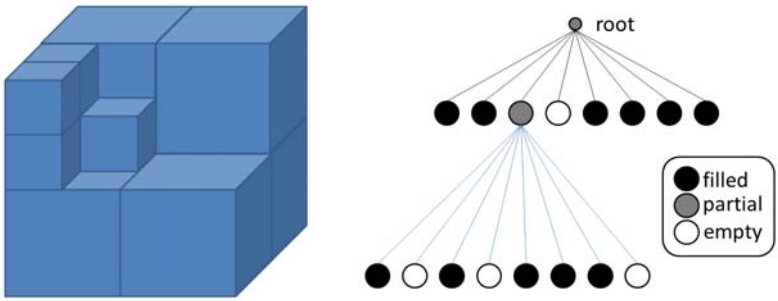


図 1・7　ボクセルの例

(3) 木表現

8 分木（Octree）は，形状が含まれている空間を縦横高さ半分に分割し八つの領域に分ける．形状にすべて占められている領域は塗りつぶし，全く占められていない領域は何もしない．部分的に形状を含んでいる空間は再度分割し，上記のような塗りつぶし・無視・分割処理を繰り返す．ボクセル表現に比べると，広い領域をまとめて表現できるので，データ容量を削減することができる．**図 1･8** に，図 1･7 に示した物体と同じ物体を 8 分木表現で表してものを示す．



(a) 8 分木による 3 次元表現　　(b) (a)に対応する 8 分木

図 1・8　8 分木の例

(4) メタボール

メタボール（Metaball：濃度球）は球の集合を用いて立体を表現する方法であり，ブロブ（Blob）とも呼ばれている．ブロブで生成されたものをブロッビーサーフェイス（Blobby

Surface）と呼ぶこともある．まず，ブロブの中心からの距離関数として密度分布を定義する．複数のメタボールがある場合には，それぞれの密度分布の和を計算し，全体の密度分布とする．そして，あるしきい値を設け，密度がしきい値のところを表面とする．こうすることで密度の等値面によって曲面が生成される．**図 1･9** にメタボールの例三つを示す．

メタボールを使うとなめらかな表面を作ることができることから液体などの表現に優れている．

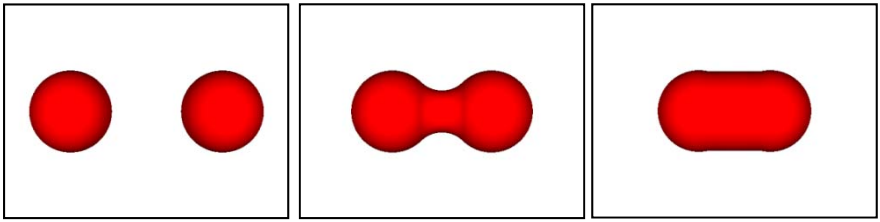


図 1・9　メタボール．左図ではメタボール同士が離れているため，二つのメタボールが独立している．次第に近づくにつれ，メタボールの密度分布が干渉し，ボール同士が融合されていく．

(5)　パーティクルシステム

パーティクル（Particle：粒子）とは位置や色などの属性をもつ微粒子を意味し，パーティクルシステム（Particle System）とは，パーティクルの集合体によって物体を表現する手法である．例えば，火の粉の集合体である炎や水滴の集まりである水などのように，境界がはっきりしないオブジェクトや形が不定なものを表現するのに効果的である（**図 1･10**）．

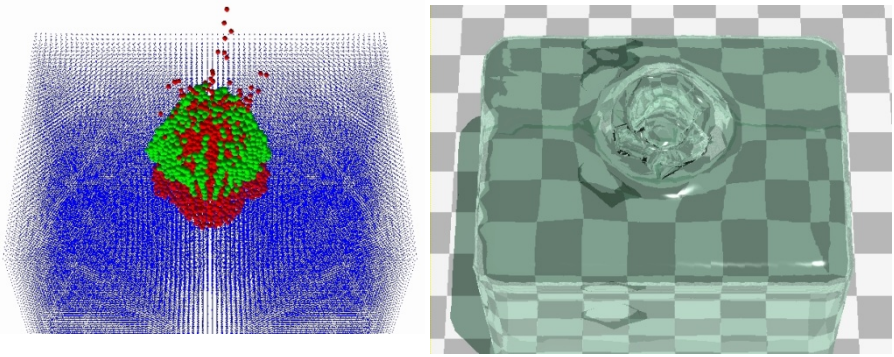


図 1・10　パーティクルによる気泡及び流体の表現．（東京都市大学画像工学研究室　向井信彦氏提供）

■2 群 － 3 編 － 1 章

1-2 レンダリングの基礎

（執筆者：張　英夏・中嶋正之）［2012 年 10 月 受領］

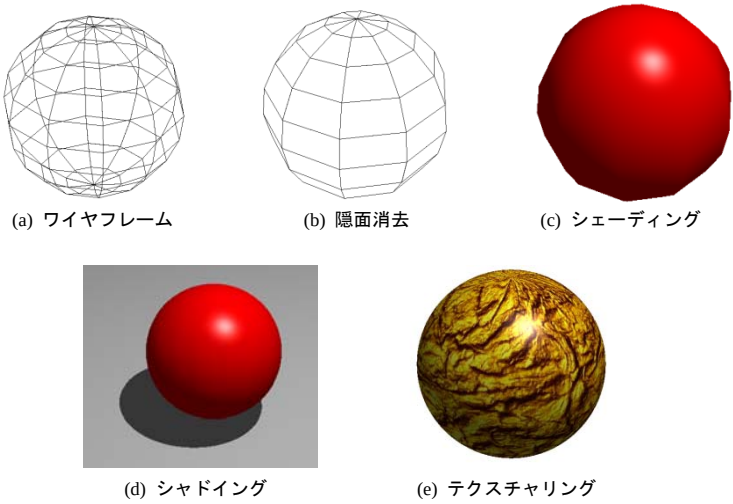
レンダリング（Rendering）は，入力されたデータを用いて画像を生成する可視化の過程である．本節では 3 次元モデルを入力とした場合のレンダリング手法について述べる．

3 次元モデルを用いてレンダリングを行うためには，以下のような手順をふむ．

① 投影面上での形状の決定：3 次元空間上で視点，視野範囲を決定した後，3 次元空間に定義されているモデルを 2 次元の画面（投影面）に投影したときの形状を決定する．

② 描画色の計算：物体を描画するために，物体各箇所の描画色を計算する．なお，描画の度合いは簡易的な描画からより詳細な描画まで以下のように分けられる（図 1･11）．

a. ワイヤフレーム表現
b. 陰線・隠面消去
c. シェーディング付加
d. シャドウ付加
e. テクスチャリング



(a) ワイヤフレーム　(b) 隠面消去　(c) シェーディング

(d) シャドイング　(e) テクスチャリング

図 1・11　描画の度合い

1-2-1 投　影

人間は目を通して世の中を見ている．物体から反射された光が目の角膜を通じて入っていき，網膜上に像を成すことによって世界を認識することができる．コンピュータ内部にも世界があり，その世界にはユーザによって配置されたモデルとカメラがある．カメラの位置，視線方向，視野範囲によって投影される範囲を決め，投影面に描画される形状を計算することができる．

投影方法（Projection Method）には大きく分けて，透視投影（Perspective Projection）と平行投影（Parallel Projection）の 2 種類がある．

透視投影は**図 1･12** に示すように，カメラの位置が比較的投影面の近くにあると仮定する方法である．透視投影されたものは，近くのものが大きく，遠方のものが小さく表示される．このため，遠近感を表現できる．

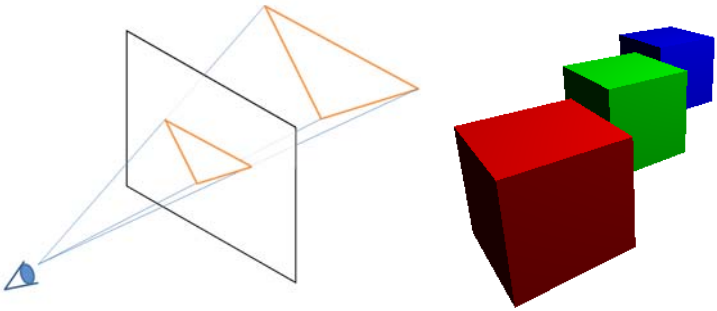


図 1・12　透視投影の概念図及び透視投影によって作成された図形

図 1･13 に示す平行投影は，視点が無限遠にあると仮定する．そのため，投射線が平行になり，近くのものも遠くのものも同じ大きさで描画される．

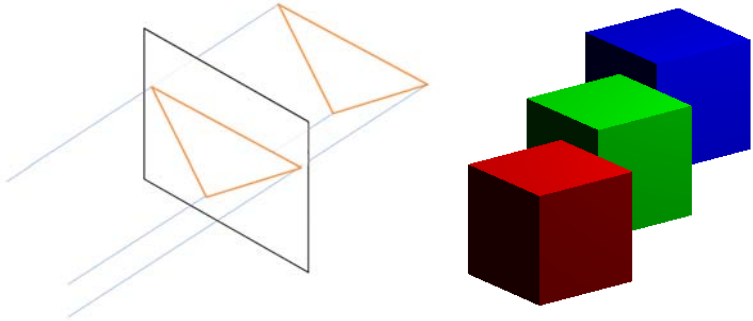


図 1・13　平行投影の概念図及び透視投影によって作成された図形

本項では，一般的に用いられている透視投影に焦点を当て，説明する．

まず，モデルが配置されているコンピュータ内の 3 次元世界を考える．このとき，物体の配置を表現さるための 3 次元座標系が必要となる．この座標系をワールド座標系（World Coordinate）と呼ぶ．3 次元座標系の表現方法として，頂点の位置を各軸に対する位置（x, y, z）で表現する直交座標系（Orthogonal Coordinate System）と，原点からの距離 r と二つの角度（θ, ϕ）で表現する極座標系（Polar Coordinate System）が存在するが，ここでは直交座標系の形式を採用する．ただし，投影変換では行列演算を多く行うため，便意上 3 次元座標を（wx, wy, yz, w）というように 4 値によって表す．通常 $w = 1$ として，（x, y, z, 1）と表すことが多い．

このように余分な実数 w を用いて表す座標を同次座標（Homogeneous Coordinate）と呼ぶ．

まずは，必要な用語を**図 1･14** を用いて説明する．

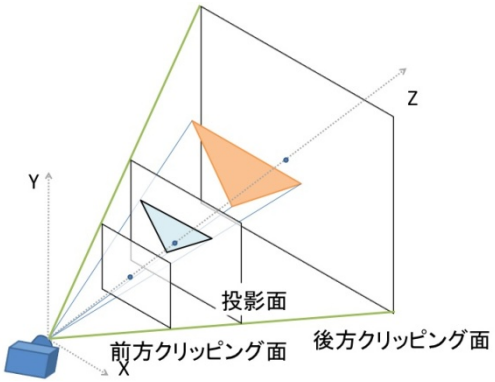


図 1・14　透視投影

いま，仮想世界の中に 3 角ポリゴンがある（図 1･14 のオレンジの三角）．これをレンダリングするためには，まずカメラを設置し，カメラを原点とするカメラ座標を考える．次に，カメラとポリゴンの間にポリゴンを投影するための平面を置く．この平面を投影面と呼ぶ．投影面はカメラ座標系の xy 平面に対し平行に，またカメラ座標の z 軸が投影面の中心点を通るようにする．カメラからポリゴンの頂点に線（投影線）を延ばし，この線と投影面が交わる点が描画される点になる．また，奥行方向に対し，2 枚のクリッピング面（Clipping Plane）を用意する．カメラの近くに配置されるものが前方クリップ面，遠くに配置されるものが後方クリッピング面と呼ばれるものである．いずれも，投影面に対し平行に配置される．投影面に描画される範囲は，前方クリッピング面を上面，後方クリッピング面を底面とする四角錐台の範囲内となる．

実際に透視投影は，この四角錐台をクリッピング空間に写像することによって求まる．写像のための行列は以下のように定義できる [6]．

$$v' = Pv$$

$$P = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

ここで，v はワールド座標系における頂点の座標，n はカメラ位置からビューボリュームの前方クリッピング面までの距離，f は後方クリッピング面までの距離であり，t, b, l, r は前方クリッピング面と投射線との交点からの上辺，下辺，左辺，右辺の位置となる．行列計算をすると，v' の 4 番目の要素が $-z$ となる．$v' = (x', y', z', -z)$ は同次座標系なので，

$$x'' = -\frac{x'}{z}$$
$$y'' = -\frac{y'}{z}$$
$$z'' = -\frac{z'}{z}$$

と計算する．2 次元座標系での x, y 値は x'', y'' を投影面の座標系にマッピングする．z'' は各頂点の奥行であるため，Z バッファ法を使うときなどに用いる．

1-2-2 隠れ面処理

モデルをそのまま出力すると，隠れて見えない面や線も描画されてしまい，形状を把握することが困難となる．そこで，現在の視線から隠れて見えない部分や裏向きの面は描かないことが一般的である．このような処理を隠線・隠面消去と呼ぶ．ここでは簡単な隠面処理方法である法線ベクトル法，Z バッファ法，スキャンライン法について述べる．

(1) 法線ベクトル法

最も単純な隠面処理法である．あるポリゴンが現在のカメラ位置から見えるか否かを求めるために，ポリゴンの法線ベクトルを用いる手法である．カメラ位置からポリゴンが見えるためには，ポリゴンの表面がカメラの方向に向いている必要がある．ポリゴンの法線ベクトルはポリゴン表面から垂直に外側に伸びたベクトルであるため，この法線ベクトル $\boldsymbol{N}$ とカメラの方向ベクトル $\boldsymbol{V}$ の内積をとることで，可視不可視の判定ができる．

- $\boldsymbol{V} \cdot \boldsymbol{N} > 0$ の場合，表の面（可視）
- $\boldsymbol{V} \cdot \boldsymbol{N} < 0$ の場合，裏の面（不可視）
- $\boldsymbol{V} \cdot \boldsymbol{N} = 0$ の場合，ポリゴンはカメラ方向に対して水平である．

このように，法線ベクトル法ではオブジェクトを構成する各ポリゴンに対し可視不可視の判定を素早く行うことで，隠面消去が可能となる．しかしながら，可視ポリゴンが本当に見えているのか，例えば可視と判定されたポリゴンの一部もしくは全部が他のポリゴンによって遮られているか否かを判定する手立てがないので，複雑な物体の隠面消去には向かない．

(2) Z バッファ法

Z バッファ法は，Z バッファと呼ばれる 2 次元配列を利用して画素ごとに処理を行い，隠面消去を行う．Z バッファを depth(x, y)，画像用フレームバッファを intensity(x, y) として，以下にアルゴリズムを示す．

1) 画面上の全画素について，depth(x, y) の値を無限に，intensity(x, y) の値を背景色の色 B に初期設定する．
   depth(x, y) = ＋∞
   intensity(x, y) = B
2) 各多角形が画面に投影されたとき，多角形の領域内に含まれる画素集合 A を計算する．
3) 集合 A に含まれるすべての画素（x_i, y_i）について以下を計算する．
   3-1) 画素（x_i, y_i）の深度 Z を計算する．

3-2) もし，$Z \leqq \text{depth}(x, y)$ ならば，この画素（x_i, y_i）を含むこの多角形が，それまでに扱われた多角形の中で，最もカメラに近いことを表している．その場合，depth(x, y) に Z を代入し，intensity(x, y) には多角形の濃淡値を代入する．一方 $Z > \text{depth}(x, y)$ ならば，既にカメラにより近い多角形が存在しているので何もしない．例えば**図 1･15** 左に示すように，最初に Z 値を計算した緑の四角形は Z 値が 20 と計算されたとする．他の物体が存在しないので，デプスバッファにはこの Z 値を保存する．次の青い四角形の Z 値を計算したところ，Z 値が 8 であったとする．すると，二番目に計算された四角形の Z 値がより大きいので，最も観測者に近い物体となり，デプスバッファ値の更新を行い，この物体を手前に描く（つまり，先ほどの四角形の上に描き重ねる）．このように，次の物体の Z 値を計算しその Z 値がデプスバッファの値より大きければ手前にあると判断する手法が Z バッファ法である．

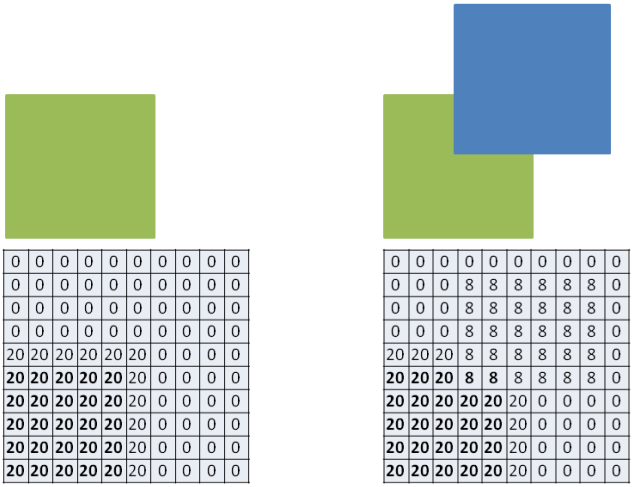


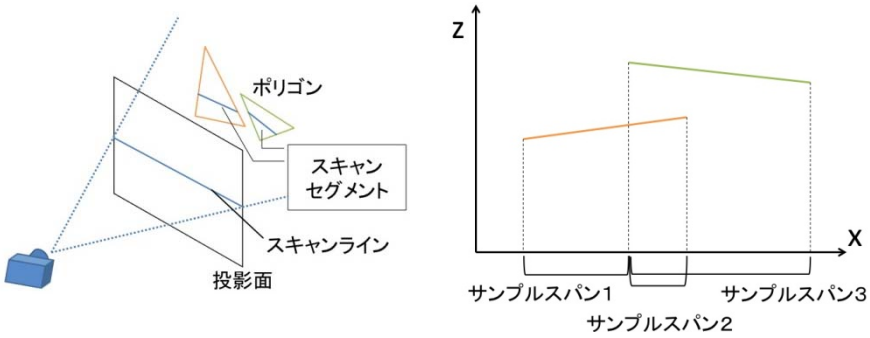
図 1･15　Z バッファ法

(3)　スキャンライン法

スキャンライン法（Scanline Algorithm）は，走査線単位に陰面消去を行う方法である．スキャンラインとは走査線を意味し，投影面の上から下へ順番に移動させながら隠面処理を行う．

三角形ポリゴンの集合をスキャンライン法で隠面処理する方法について述べる．**図 1･16**(a) に概念図を示す．まず，カメラと投影面，ポリゴンを考える．投影面の一番上からスキャンラインによって走査を行う．まず，カメラ位置とスキャンラインによって定義される平面を考える．この平面をスキャンライン面（走査面）と呼ぶ．この平面が各ポリゴンと交わる線分（スキャンセグメント）を計算する．分かり易いように，スキャンライン平面を図 1･16(b) に示す．スキャンライン平面には二つのスキャンセグメントが存在する．それらの x 値の範囲を調査すると，オレンジのセグメントのみの部分と緑とオレンジのセグメントが重なる部

分，緑のみの部分の三つに分けることができる．これら三つの部分をそれぞれサンプルスパンと呼ぶ．一つのスキャンセグメントのみが存在する部分に対してはそのまま描画すればよい．一方で，スキャンセグメントが重なっているサンプルスパンについてはどちらのセグメントの z 値が小さいか（カメラ位置に対し手前に存在するか）を算出し，小さい方のセグメントを描画することで，隠面消去を行うことが可能となる．



(a) スキャンライン法の概要　　(b) サンプルスパン

図 1・16　スキャンライン法

1-2-3 陰影処理

立体を，ある光源の下で観察すると，その表面の明るさは一定ではない．例えば**図 1・17** に示すように，球を机上に置いて斜め上方から光を当てると，光が直接当たらない面は暗く，光が当たる方向の面も，光源との方向の関係から明るさが変化して観測される．また机上には，球によってさえぎられて光が当たらない部分が生じる．このとき，光が直接当たらずに物体表面が暗くなる部分を陰（Shade），球により光がさえぎられてできた光が当たらない机上の領域を影（Shadow）と呼ぶ．

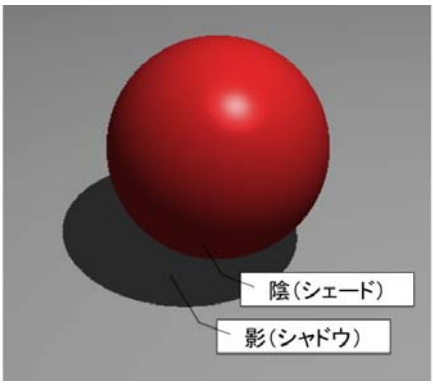


図 1・17　陰と影

CG の分野においては，立体を構成する各面に対して，光源位置や周囲環境条件により定まる明るさを決定する方法をシェーディング（Shading）または陰影付けと呼び，影を付与する作業を影付け（Shadowing）と呼ぶ．

以下，まずは基本的な光学的モデルを述べ，次にシェーディング方法を説明し，最後にシャドイングについて説明する．

(1) シェーディング法

シェーディングを表現するためには，拡散反射，鏡面反射，環境光，透過光の四つの光の成分を考える．

鏡面反射（Specular Reflection）とは，入射した光がある一方向のみに強く反射する成分を意味する．光沢をもっている物体は鏡面反射を考慮することで，艶を表現することができる．拡散反射（Diffuse Reflection）とは，入射した光があらゆる方向に均等に反射する成分を意味する．環境光（Ambient Light）とは，一箇所の光源ではなく，物体を取り巻く周辺環境からの反射光を意味する．透過光（Transmitted Light）とは，透明もしくは半透明な物体内部を透過して表面から発する光の成分である．

物体上のある 1 点の明るさ L を，無限遠点にある光源からの入射光による散乱の輝度 L_d，入射光の正反射による輝度 L_s，立体内部からの透過光 L_b 及び周囲環境の散乱 L_e の四つの要素の和として次式で求める．

$$L = L_d + L_s + L_b + L_e$$

上式のように，物体上のある点の輝度を求める式を陰影モデル（Shading Model）と呼ぶ．右辺の各光量を求める各種の方式が提案されている．

例えば，散乱反射 L_d は，

$$L_d = k_d I \cos \alpha$$

として計算できる．ここで，k_d は立体の拡散反射係数，I は入射強度，α は物体表面の法線と入射光源の角度である．

鏡面反射 L_s については

$$L_s = k_s I \cos^n \beta$$

として計算できる．ここで，k_s は立体の鏡面反射係数，n は鏡面反射の強度係数，β は光源の正反射成分とカメラ方向ベクトルとの角度である．

環境光成分 L_e については，

$$L_e = k_e$$

として計算できる．ここで，k_e は環境反射係数である．

透過光 L_b は

$$L_b = k_b I_o$$

と計算できる．ただし，k_b は透過係数，I_o は物体後方の光の屈折方向における輝度である．

環境光と拡散反射光成分のみを考慮したランバートモデル（Lambert Model），環境光と拡散反射光と鏡面反射光を考慮したフォンモデル（Phong Model）などが広く知られている．

以上で，カメラ方向と光源位置と法線ベクトルが既知の場合のある一点に対する反射光の強さ（明暗：シェーディング）を求めることができる．この反射計算方法を用いて，物体全体のシェーディングを行う方法として，フラットシェーディング法とスムーズシェーディン

グ法がある．

フラットシェーディング法（Flat Shading）は，ポリゴンごとに反射色を求め，ポリゴンを同じ色で塗りつぶす方法である．**図 1･18** に，フラットシェーディング法による描画結果を示す．ポリゴン内は同じ色が塗られているため，球の丸みが表現できていない．

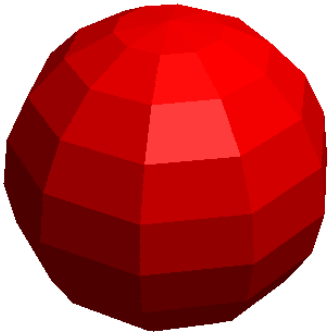


図 1・18　フラットシェーディング法

このような問題に対応するために，スムーズシェーディング法（Smooth Shading）が提案されている．ここでは代表的なグーローシェーディング法とフォンシェーディング法を説明する．

グーローシェーディング（Gouraud Shading）は，輝度の補間によって滑らかな描画を可能にする手法である．補間の方法は以下のようである．**図 1･19** に示すようなポリゴンがあったとする．いま，ポリゴンの各頂点における輝度強度 I_1, I_2, I_3 が求まっている．このとき，ポリゴン内部の点 I_p の輝度強度を求めたい．そのためには，I_p を通り y 軸に垂直なスキャンライン（$y = y_p$）を考え，そのスキャンラインとポリゴンとの交点二つを求める．交点における輝度強度 I_a, I_b は，I_1, I_2, I_3 を用いて補間することによって以下のように求まる．

$$I_a = I_1 - (I_1 - I_2)\frac{y_1 - y_p}{y_1 - y_2}$$

$$I_b = I_1 - (I_1 - I_3)\frac{y_1 - y_p}{y_1 - y_3}$$

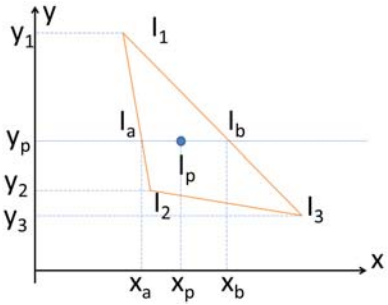


図 1・19　グーローシェーディング方法

次に，求まった l_a, l_b を用いて l_p における強度計算を以下のように行う．

$$l_p = l_b - (l_b - l_a)\frac{x_b - x_p}{x_b - x_a}$$

この方法は輝度の補間値を求めているので，ポリゴン内部に鏡面反射部分があってもそれが表現できないという欠点がある．**図 1･20** にグーローシェーディングによる描画例を示す．フラットシェーディングに比べるとなめらかな表面になっているが，鏡面反射がうまく表現できていない．

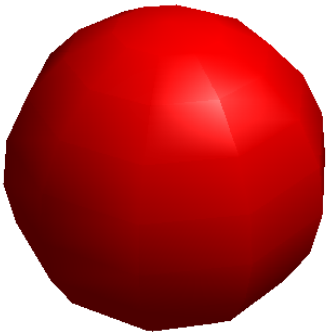


図 1・20　グーローシェーディングによる描画結果

鏡面反射を表現できる方法として，フォンシェーディング（Phong Shading）法がある．これは輝度の代わりに法線ベクトルを補間することによってスムーズシェーディングを行う．このために，ポリゴン頂点で法線ベクトルを求め，ポリゴン内部の各点における法線ベクトルを補間する．この方法は鏡面反射成分を表現できるだけでなく，比較的精度の高いレンダリングが可能であるが，計算が複雑になり，処理時間が長くなるという問題がある．**図 1･21** にフォンシェーディング法による描画結果を示す．表面が非常になめらかに描画されており，更に鏡面反射もうまく表現できている．

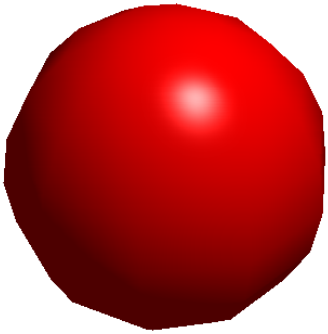


図 1・21　フォンシェーディングによる描画結果

以上で示した陰影モデルでは，他の物体からの反射光を再び入射光として扱うことが不可能であった．こうした問題を解決したのがレイトレーシング法（Ray Tracing：光線追跡法）

である．以下に，レイトレーシング法のアルゴリズムを示す．

なめらかな平面に光線が照射されると，一部は反射し一部は内部へ向かって進む．これらの表面で分離した光は，更に他の表面に照射され分離を繰り返す．このように，多数の反射を繰り返して視点へ向かった光が実際に観測されることになる．もし，光源から出発して反射分離の繰り返し演算を行った場合，視点まで到着する光は少なく，多くの演算が最終的な CG 画像作成には不要なものとなる．そこで逆に，視点方向から出発して，最終的に得たい 2 次元ディスプレイ平面上の各画素点だけの明度情報を得るのがレイトレーシング法である．すなわち，カメラから出発しディスプレイ平面の各画素点を通過する単位光源を仮定し，その光源を追跡して明度値を求める．例えば，ある光源が対象物体と一度も交差しなければ背景色（例えば黒）とする．もし物体と交差した場合には，対象物が不透明な場合はその表面の明度（色相も考慮）とし，ガラス球などの透明な場合は反射光と透過光を考え，更に光線を追跡する．その際，単位光量がある明度以下となるか，または反射回数が設定した回数を超えた場合に追跡を打ち切る．このような計算をすべての画素点に対して施せば，レイトレーシングの画像が求められる．**図 1･22** に，レイトレーシング法により求めた画像を示す．

図 1・22　レイトレーシング法によるレンダリング結果

レイトレーシング法は直接光を用いた鏡面反射を計算するには向いているが，その他の間接光などの効果を表現することはできない．そのため，間接光を考慮したレンダリング手法がいくつか存在する．これらのモデルでは閉じた環境で光エネルギーが保存されると仮定しており，各表面から反射される光は他の表面に吸収され，更に反射されると仮定している．このように，拡散光を正確に取り扱うレンダリング方法をグローバルイルミネーション（Global Illumination：大域照明）と呼ぶ．この大域照明モデルの代表的な手法の一つに，ラジオシティ法（Radiosity Method）がある．

ラジオシティとは，ある表面から放出されるエネルギー率を意味する．ラジオシティ法とは，ある環境における各表面のラジオシティを計算する方法を指す．

ラジオシティ法の基本的なアルゴリズムの流れを以下に示す．

(1) 物体表面を細分化し，細かいメッシュを作成する．

(2) 作成したそれぞれのメッシュに，他のメッシュから分配される光の量を計算する．

(3) 計算された値をメッシュに保存する．

ラジオシティ法では拡散光を考慮しているため，ぼやけた影を表現できたり，間接光を考慮することによって反射面の色が隣接する面に影響するといった現象を再現することができるなど，非常に写実的なレンダリングが可能である．例えば**図 1･23** では，立方体の影の境界部分がぼやけていることが分かる．また，左側に立っている立方体を見ると左側の側面に壁の赤色が反射されていることが確認できる．また，ラジオシティ法はレイトレーシング法とは異なり，一度計算を行っておけば，カメラ設定を変更しても再計算の必要がない点が利点としてあげられる．

一方で，所要メモリや計算時間が膨大になることが難点とされている．また，物体表面の形状が複雑な場合，細かく分割する必要があるが，その場合，データ量や計算量も膨大になる．また，ラジオシティ法は拡散反射にしか適用できないという問題がある．このため，レイトレーシングで再度レンダリングを行って，スペキュラー反射の影響を付加していることがある．

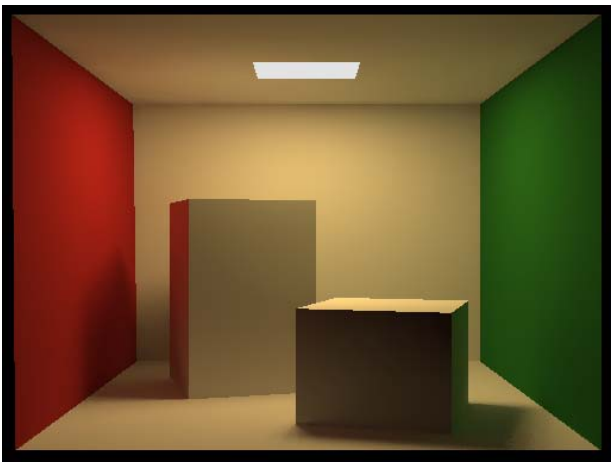


図 1・23　ラジオシティによるレンダリング

(2)　シャドイング

これまで，陰の生成方法について述べた．次に，影の生成方法（Shadowing）について述べる．影は，光源条件によって本影と半影に分類することができる．例えば**図 1･24** 左に示すような点光源の場合，物体の背後部分には全く光が当たらないか当たるかのどちらかしかない．このように全く光が当たっていない部分を本影と呼ぶ．しかし，図 1･24 右に示すように，光源が面光源だった場合，光が全く当たらない，すべての光が当たるという二つのみでなく，一部の光源のみが到達する部分が存在する．このように，一部の光のみが到達している部分を半影と呼ぶ．面光源の場合，本影と半影が混在する．

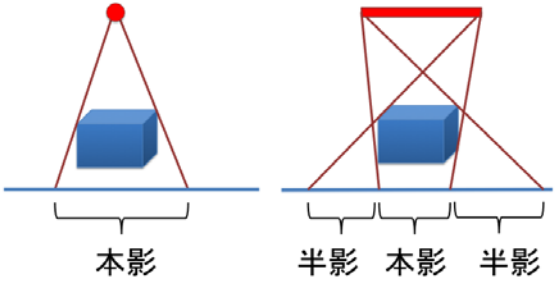


図 1・24　本影と半影

本影の計算は比較的簡単に行うことができる．例えば，レイトレーシングを利用して本影を計算することができる．まず，視点位置から物体表面のある点 P へ光線を飛ばす．P から光源までの道のりの間に物体が存在すれば，P は影領域となる．半影の方は計算がより複雑である．例えば，光源を点光源の集合として近似し，物体表面が，各点光源から影になっているか否かを計算する方法がある．これらの計算結果を足し合わすことによって半影を生成することができる．しかしながら，よりリアルな半影を生成するためにはより多くの点光源で近似する必要があり，計算時間がかかるなどの問題がある．

1-2-4 テクスチャマッピング

テクスチャマッピングは，複雑な模様や質感をもつ物体を表示する技術である．テクスチャマッピングには様々な手法があり，目的に応じて使い分けられる．

(1)　テクスチャマッピング

狭い意味でのテクスチャマッピング（Texture Mapping）は，写真や絵などの 2 次元平面の画像を 3 次元物体の表面にマッピングする手法である（**図 1･25**）．2 次元テクスチャを張り付けるためには，物体の表面上の各点に対して，画像のどの点を対応させるかという問題があり，画像を球状にマッピングする方法，画像を円筒状にマッピングする方法，画像を変形せずそのままマッピングする方法の三つの方法がある．

図 1・25　テクスチャマッピングの例

(2) ソリッドテクスチャマッピング

ソリッドテクスチャマッピング（Solid Texture Mapping）とは，オブジェクト表面のみにテクスチャを張り付けるのではなく，金太郎飴のように 3 次元物体の内部にまでテクスチャを浸透させる方法を意味する（**図 1･26**）．

図 1・26　ソリッドテクスチャマッピングの例

(3) バンプマッピング

バンプマッピング（Bump Mapping）とは，表面形状を変えずに凹凸を表現するために利用される（**図 1･27**）．まず，バンプマップという明度情報のみをもつ画像を用意する．この画像の明度値によって窪んでいるか出っ張っているかを表す．次に，マップの濃淡変化に応じて法線方向に微小距離だけ移動させたと仮定し，その点での法線ベクトルを求める．このような方法によって複雑なテクスチャを効率良く表現できるが，実際に形状が変化するわけではないので，カメラに近く輪郭がはっきり見えるオブジェクトや，バンプマップによって大きな凹凸を表現する場合には，輪郭が平らなままであるため，不自然さが際立つ．それを解決する方法として，ディスプレイスメントマッピング（Displacement Mapping）がある．バンプマッピングでは法線の方向のみを変えていたのに対し，ディスプレイスメントマッピングでは，ディスプレイスメントマップの情報に応じて実際に表面の構造を変化させる．

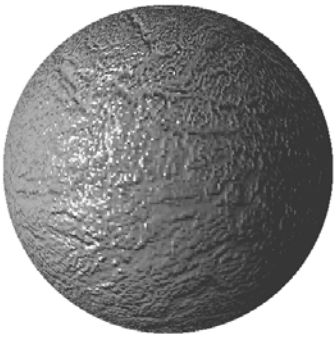


図 1・27　バンプマッピングの例

(4) 環境マッピング

よりリアルな描画のためには周辺物体からの反射や屈折を考慮しなければならない．しかしながら，周辺環境が複雑な場合，計算コストがかかってしまう．そこで，高速にかつ疑似的に背景を物体に映り込ませる方法として，周囲の環境をマッピング画像として利用することで，物体表面に書き込む手法が環境マッピング（Environment Mapping）である（**図 1･28**）．

図 1・28　環境マッピングの例

■2 群 － 3 編 － 1 章

1-3 ノンフォトリアリスティックレンダリング

（執筆者：張　英夏・中嶋正之）［2012 年 10 月 受領］

CG における画像や映像生成の方向としては，長年より現実感の高い映像の生成を目的にアルゴリズムの提案がなされてきた．しかし，リアリズムの追求もある程度の成果が得られる状況になり，CG の新しい流れの方向として，必ずしもリアル性を追求しない非写実的画像生成手法（ノンフォトリアリスティックレンダリング : Non-Photorealistic Rendering，以下，NPR と略す）が活発に議論されるようになってきた（**図 1・29**）．

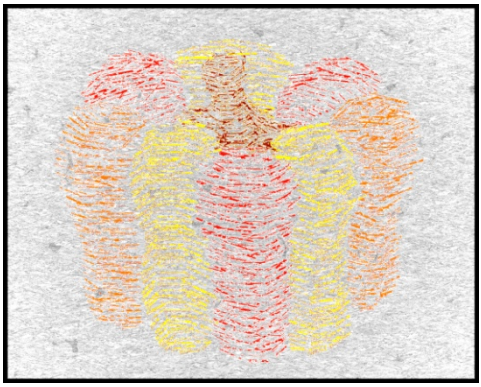


図 1・29　NPR 画像の例 [24)]

NPR は，直訳すると「写実的レンダリングでないもの」と，いささか不明瞭なネーミングである．そのため，近年はアーティスティックレンダリング（Artistic Rendering）と呼ばれることもある．

一般に画像や映像表現を考えた場合，写実的方法以外にも多くの表現方法があることが理解できる．例えば，観光地で売られている絵はがきは，風景写真の他に風景を水彩画で描いたものが並んでいる．また，機械や医療用の技術解説書では手で描いたイラストが用いられている．

これらの手書きやイラストが好まれるのは，細部にこだわったり，重要な部分を強調したり，不要な部分は省略したりする情報の操作をユーザが行えるためである．また，画像全体を通して観測すると写真とは異なり，描き手の主観をより伝えやすい．代表的な NPR 例を列挙すると以下のようになる．

・産業，工学分野：テクニカルイラストレーション
・芸術分野：水彩画調，油絵調，水墨画調レンダリング
・娯楽映像分野：CG 誇張表現，3 次元 CG のセル画風変換など

手書き風画像の生成にはそのデータ入力方法により以下の三つに分類される．

1. マウスなどのポインティング装置を利用する方法
2. 写真などの 2 次元画像を利用する方法

3. 3 次元幾何情報を入力する方法
以下に，それぞれについて述べる．

1-3-1 直接入力による方法

マウスやペンタブレットなどのポインティングデバイスによって計算機内の仮想画材を操り，絵を描く手法である．自由度が非常に高く，ユーザの意図を最も反映させやすい入力方法である．

Lansdown らによれば, インタラクティブペイントシステムの中で初めて画材によって描かれたような描画効果をもつアルゴリズムが使われたのは Quantel の PaintBox の“chalk”機能である [7]．この機能は，ストロークを描く際に簡単なルックアップテーブルを用いてピクセルを描画するかしないかを決定するものであった．現在，実際の画材を用いたような描画をするための商業パッケージでは，MetaCreations 社の Painter[8] が有名である．更に，よりリアルな絵の具の特性の再現を試みたものに，水っぽいものから粘性のある物質まで様々な特徴をもつ絵の具のモデル化をセルオートマトンを用いて行い，実際の画材の特性に似せた描画を可能とした研究 [9] や，水彩での滲みまで再現した研究 [10] がある．しかしながら，これらは処理が重たいため，ストローク入力時には仮描画しか行われず，本描画は後処理で行われていた．

このように，直接入力法においては，より画材に似せるための処理とインタラクションできる処理速度がトレードオフになっていた．一方で，近年 GPGPU が出現し，高速なシミュレーションが可能になってきたため，絵の具の流体シミュレーションを行うことによってよりリアルな画材の表現が可能になってきている [11)~13)]．

1-3-2 2 次元画像を入力とするもの

この手法では，画像処理により得られる画像上の局所的な特徴量と描画方法との対応付けが重要となる．対応付けが人間の感覚と異なる場合，機械的な描画になったり，対象物の判読不可能な描画になる可能性がある．

Haeberli は，画像の輝度勾配から描画ストロークの方向を決定して描画を行う手法 [14] について発表した．この絵は，機械的描画であるともいえるが，高度な画像認識を行わずに手描き風の画像の生成可能性を示した初期の例である．

その他の研究でも輝度勾配を基にストローク方向を決定して描画を行う手法 [15] が多いが，FFT を画像処理に用いた描画方法 [16]，画像モーメントを画像処理に用いた描画方法 [17]，K-平均アルゴリズムを用いた領域分割法を画像処理に用いた描画法 [18] の提案がなされている．

また, 動画像への適応にはオプティカルフロー技術を利用した手法の提案 [19] がされており，画像フレーム間での対象物の移動に追従したストロークの配置を行っている．ストロークの大きさに従って描画手順を階層的に行う手法ことにより，描画の誤りを修正しつつ大胆な描画を行う手法 [20] も存在する．

その他の表現方法を目指したものには，版画風表現 [21]，テクニカルペン画風表現 [22]，ストロークの表現力を高めるためにテクスチャを用いた研究 [23]，色鉛筆画のために特化したペイントツールの研究 [24]，テクニカルペンによる描画をマウスオペレーションにより行う手法などがある．テクニカルペンでの描画では細いストロークの集積により濃淡や模様を表現する

ため細かな描画が要求される．しかし上記の手法では，模様を個々のストロークを描画するのではなく，テクスチャを“塗り付ける”ことにより広範囲な書き込み操作を簡便に行うことができ，計算機ならではの描画法の提案として興味深い．

1-3-3 3 次元幾何情報を入力とする方法

3 次元幾何情報を入力データとした場合，視点を移動した場合の画像制作やオブジェクトの配置を変更した画像制作などでデータの再利用が可能である点が最大の利点である．この手法の初期の研究で代表的なものに，Saito らの G バッファを用いた表現法 [25] の研究がある．G バッファは，描画される画面と同等の解像度をもち，射影される各ピクセルに対応する面の距離，法線方向などを格納する領域として定義されている．G バッファを用いた表現例として線画風な画像を斎藤らは示しているが，後にその他の表現法にも応用されている．

多様な表現手法への試みとしては，テクニカルペン画風表現 [26]，樹木のための水墨画風表現 [27]，油絵風表現 [28]，特殊な照明，反射モデルに基づくテクニカルイラスト風表現 [29] などがある．

油絵風表現ではストローク描画の雰囲気を再現しようとするが，動画像を制作する際に個々のストロークの描画に注意を払わないとストロークがノイズのようにちらついて見えてしまう．Meier の研究 [28] ではその点に考慮した描画法を実現している．

以上のような 3 次元幾何情報をもとにした手法では，総じて先に述べた視点移動やオブジェクトの移動に対して描画が容易であるという利点がある一方で，3 次元オブジェクト制作にかかるコストが問題となる．2 次元の画像を制作するには 2 次元に直接描画する方が容易であるとか，3 次元モデル作成よりも描画に習熟しているなどの理由もある．そこで，アニメーションの制作現場では立て看板のような板状の 3 次元モデルに描画した絵をテクスチャとして張ることによる，2.5 次元的な手法も用いられている [30]．

■参考文献

1) Blender. [Online]. http://www.blender.org/
2) POV-RAY. [Online]. http://www.povray.org/
3) Autodesk. [Online]. http://www.autodesk.co.jp/
4) NewTek Lightwave. [Online]. http://www.newtek.com/lightwave
5) The Stanford 3D Scanning Repository. [Online]. http://graphics.stanford.edu/data/3Dscanrep/
6) Eric Lengyel, 智英 狩野, “ゲームプログラミングのための 3D グラフィックス数学,” in ゲームプログラミングのための 3D グラフィックス数学, ボーンデジタル, pp.97-99, 2002.
7) S. Schofield J. Lansdown, “Expressive rendering: A review of nonphotorealistic techniques,” Computer Graphics and Application, vol.15, no.3, pp.29-37, 1995.
8) Painter. [Online]. http://www.metacreations.com/products
9) T. Cockshott, “Modeling the texture of paint,” Computer Graphics Forum, vol.11, no.3, pp.C217-C226, 1992.
10) C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, D. H. Salesin, “Computer-generated watercolor,” SIGGRAPH 97, pp.421-430, 1997.
11) C. L. Tai N. S. H Chu, “MoXi: Real-time ink dispersion in absorbent paper,” Transactions on Graphics, vol.24, no.3, pp.504-511, 2005.
12) W. V. Baxter, V. Scheib, M. C. Lin, “dAb: Interactive haptic painting with 3D virtual brushes,” SIGGRAPH, pp.461-468, 2001.
13) N. Chu, W. Baxter, L. Y. Wei, N. Govindaraju, “Detail-preserving paint modeling for 3D brushes,” NPAR’10, pp.27-34, 2010.

14) P. Haeberli, “Paint by numbers: Abstract image representation,” Computer Graphics, vol.24, no.4, pp.207-214, 1990.
15) T. Agui, H. Do, H. Nagahashi, M. Kaji, “Painting-like image generation,” 日本印刷学会誌, vol.31, no.1, pp.40-48, 1994.
16) 齋藤　豪, 中嶋正之, “計算機による手描き風画像の自動生成に関する研究,” テレビジョン学会誌, vol.50, no.10, pp.1528-1535, 1996.
17) 白石路雄, 山口　泰, “実在する油彩画の特徴分析による油彩画風画像の生成手法,” 情報処理学会研究会報告, vol.98-CG-91, pp.1-6, 1998.
18) 笠尾敦司, 中嶋正之, “テクスチャの方向要素を利用した K-平均アルゴリズムによる自然な領域分割方法,” 電子情報通信学会, vol.J81-D-II, no.7, pp.1528-1535, 1998.
19) P. Litwinowicz, “Processing images and video for an impressionist effect,” SIGGRAPH, pp.407-414, 1997.
20) A. Hertzmann, “Painterly rendering with curved brush strokes of multiple sizes,” SIGGRAPH, pp.453-360, 1998.
21) 大河内俊雄, 水野慎士, 岡田　稔, 鳥脇純一郎, “仮装版木自動生成による版画画像合成の一手法,” 情報処理学会研究報告, vol.97-CG-86, pp.1-4, 1997.
22) M. Salisbury, C. Anderson, D. Lischinski, D. H. Salesin, “Scale-dependent reproduction of pen-and-ink illustrations,” SIGGRAPH, pp.461-468, 1996.
23) S. C. Hsu, H. H. Lee, “Drawing and animation using skeletal strokes,” SIGGRAPH, pp.109-118, 1994.
24) 高木佐恵子, 藤代一成, 中嶋正之, “えんぴつ画のボリュームモデリング,” 情報処理学会論文誌, vol.40, no.3, pp.1177-1187, 1999.
25) T. Saito, T. Takahashi, “Comprehensible rendering of 3D shapes,” SIGGRAPH, vol.24, no.4, pp.197-206, 1990.
26) G. Winkenbach, D. H. Salesin, “Rendering parametric surfaces in pen and ink,” SIGGRAPH, pp.469-476, 1996.
27) 張　青, 高橋淳也, 村岡一信, 千葉則茂, “樹木の水墨画調レンダリング,” 情報処理学会研究報告, vol.95-CG-76, pp.45--50, 1995.
28) B. J. Meier, “Painterly rendering for animation,” SIGGRAPH, pp. 477-484, 1996.
29) A. Gooch, B. Gooch, P. Shirley, E. Cohen, “A non-photorealistic lighting model for automatic technical illustration,” SIGGRAPH, pp.447-452, 1998.
30) E. Guaglione, M. Altman, K. Barshatzky, R. Bekuhrs, B. Cook, MA. Pigora, T. Plett, R. Sluiter, “The art of Disney's mulan,” SIGGRAPH 98 Course note #39, 1998.
31) M. P. Salisbury, M. T. Wong, J. F. Hughes, D. H. Salesin, “Orientable texture for image-based pen-and-ink illustration,” SIGGRAPH, pp.401-406, 1997.
32) Y. Chang, S. Saito, M. Nakajima, “Example-based color stylization of images,” ACM Transactions on Applied Perception, pp.322-345, 2005.
33) H. Y. Shum, S. B. Kang, “A review of image-based rendering techniques,” Proc. SPIE, vol.4067, pp.2-13, 2000.
34) K. Weiler, P. Atherton, “Hidden surface removal using polygon area sorting,” Computer Graphics, pp.214-222, 1977.