アニメーションシステム

画像の図形が少しずつ変化しながら連続的に速い速度で再生されると、人間の目には動いているように見える。これを残像効果という。この画像の再生回数のことをフレームレートといい、単位は１秒間に再生されるフレーム数という意味で fps（Frames Par Second）が用いられる。人間は動画が滑らかであると感じるには最低でも 8～10fps が必要であるとされており、例えば映画やテレビの画面更新間隔は十分に滑らかに見えるよう、元のフィルムはそれぞれ 24fps や 30fps で作成されている。当初、テレビで放映された手書きのアニメーションは経済的な理由から 12fps、8fps、もしくはそれ以下のフレームレートで作成されていた。CG アニメーションの登場により、24fps あるいは 30fps で作成された高品位のアニメーションが実現された。

「元のフィルム」と断りを入れたのは理由がある。実際にはテレビやゲーム機、映画の映写機といった装置では 60fps や 48fps といった高い割合で画面を更新している。元のフィルムは最終的にこの装置による画面の更新の割合に合わせるように変換を行った上で目にすることになる。ただし、変換を行うということは、同じ画像を複数回表示して調整するということになるので、映像の滑らかさは結局、元のフィルムのフレームレートに依存することになる。

今回の授業ではアニメーションシステムに関する概要について述べていく。

1. 3 次元 CG 作成のための技術

3 次元で CG を作成､アニメーションを行うためには､いくつかの技術に対しての知識が必要となる。以下に代表的なものを示す。

・形状モデリング
・レンダリング
・投影変換
・幾何変換
・etc.

「形状モデリング」は物体の幾何情報（データ）を定義する方法であり、このデータを元に「レンダリング」という輝度計算処理を行うことになる。レンダリングを行う際には、3 次元形状の（頂点などの)位置データを元に､仮想的に設定するスクリーン平面である 2 次元面へと変換する必要が出てくる。この変換処理を「投影変換」という。また、形状モデリングで定義されたデータに対して、平行移動、回転、拡大・縮小などの幾何的な操作が行われる。これを「幾何変換」という。アニメーションを行う際には図形の位置を少しずつ変化させた画像を複数用意し、連続して再生することで行う。画像の作成に幾何変換を用いると、手書きで画像を１枚ずつ作成する方法に比べ、大幅に手間を省くことになる。それでは、これらの技術の概要を以下に示していく。

2. 形状モデリング

形状モデリングとは物体の幾何情報を定義する方法である。例えば、３次元形状を定義するには、コンピュータ内に３次元空間と座標系を定義し、その座標系に点の集合として３次元形状を定義することになる。CG で主に使用される座標系は直行座標系 O-xyz である。これには左手系と右手系があり、CG では一般的に右手系が利用され、これから実習に利用するグラフィックスライブラリである OpenGL

でも右手系が採用されている。また、この座標系においてはz方向が視点から奥への方向となる。また、図2.1では回転方向を示す矢印を示している。この回転方向は回転処理を行う際には正負の方向として利用する。回転方向は右手系ならば右手を、左手系ならば左手を用いて、矢印方向と親指を一致させるように矢印を握ると、手首側から指側への方向が回転方向として決まる。各軸周りの回転をそれぞれ、ロール（x軸周り）、ピッチ（y軸周り）、ヨー（z軸周り）という。

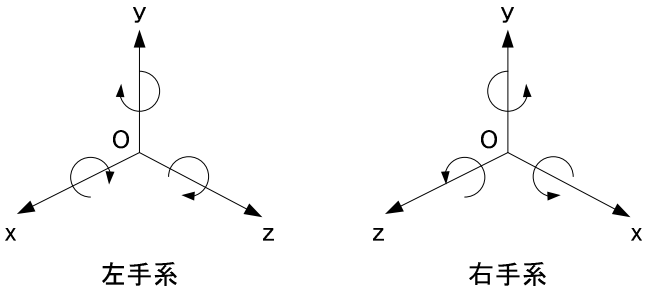


図2.1　右手系と左手系

コンピュータ内での形状の表示方法は大きく分けて次の3つに分類される。

・ワイヤーフレームモデル
・サーフェイスモデル
・ソリッドモデル

以下のような正六面体形状を例に取ってこれらのモデルについて述べていく。物体の形状を表示するためには、コンピュータ上で形状を定義しなければならない。物体の形状を定義する基本的な方法は次の情報を定義する方法である。それは以下の図に示すように頂点、稜線、面の情報であり、各頂点には座標値の情報がある。座標値に頂点番号を割り付け、これらが、稜線、面として、どのように接続しているかを定めることで、コンピュータ内で形状を定義していく。

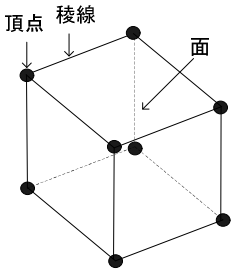


図2.2　頂点、稜線、面

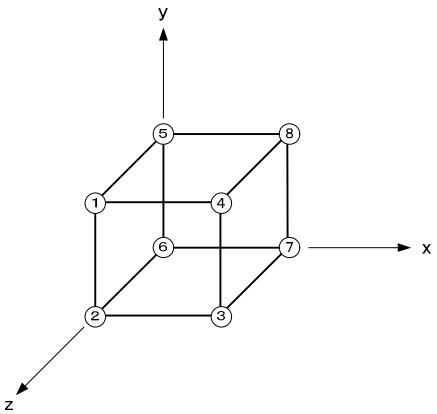


図2.3　ワイヤーフレームモデル

頂点番号	座標値		
	x	y	z
1	0	1	1
2	0	0	1
3	1	0	1
4	1	1	1
5	0	1	0
6	0	0	0
7	1	0	0
8	1	1	0

図2.4　頂点番号と座標値

ワイヤーフレームモデルでは直線や曲線のみを用いて立体の形状を表現したモデルである。頂点の座標値、稜線、頂点との関係を定義する。その結果、モデルの形状は針金細工のような状態となり、形状の視覚的な把握は困難である。

サーフェイスモデルでは3次元形状を構成する面情報のみを用いて立体の形状を表現したモデルである。頂点の座標値と面を構成する頂点との関係を定義することになり、その結果、モデルには面情報が付加された状態となる。面情報を持つのでレンダリングを行うことができるが、物体の内外に関する情報をもたないため、物体形状を集合演算に利用することはできない。

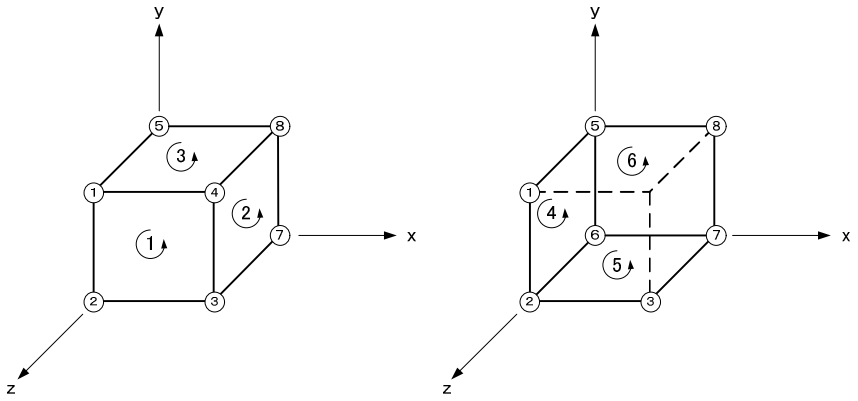


図 2.5　サーフェイスモデル

面番号	頂点番号			
1	1	2	3	4
2	4	3	7	8
3	5	1	4	8
4	2	6	5	1
5	2	3	7	6
6	6	7	8	5

図 2.6　面番号と頂点番号

ソリッドモデルではサーフェイスモデルに面の表・裏の概念を加えたものとなる。中身が詰まった立体としての表現が可能であり、立体どうしの集合演算ができる。面に表・裏の概念を加えるためには、接続している頂点を一定の規則で接続すればよい。例えば、定義しようとする面を外側から見て、反時計回りに頂点番号を並べて記述し、その接続情報を描画や集合演算を行う際の判断材料にする。

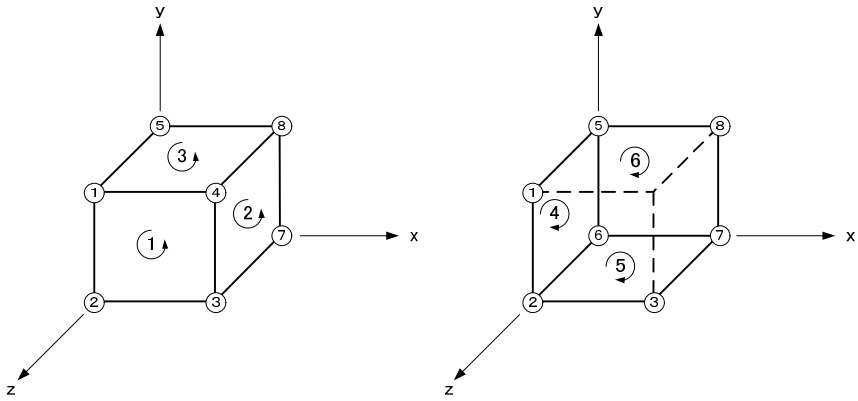


図 2.7　ソリッドモデル

面番号	頂点番号			
1	1	2	3	4
2	4	3	7	8
3	5	1	4	8
4	2	1	5	6
5	2	6	7	3
6	6	5	8	7

図 2.8　面番号と頂点番号

ここまでに紹介したモデルはすべて、立体形状を幾何情報と接続情報で表現したものであり、幾何情報については座標値に限っている。座標値と接続情報で定義したソリッドモデルの具体的な例としては**ポリゴン表現**というものがあり、OpenGL でも採用されている。これは形状をポリゴンと呼ばれる多角形の集合で表現したものである。

この他、形状の幾何情報として数式を利用したものがある。たとえば、プリミティブと呼ばれる基本立体を数式で表し、その集合演算で形状を表現する **CSG 表現**や、中心点とその周囲に、球状に密から疎へと広がる濃度分布を定義し、一定の濃度（しきい値）をもつ等濃度面を立体表面とし、しきい値より濃度が高い部分を立体内部として扱う**メタボール**などがある。

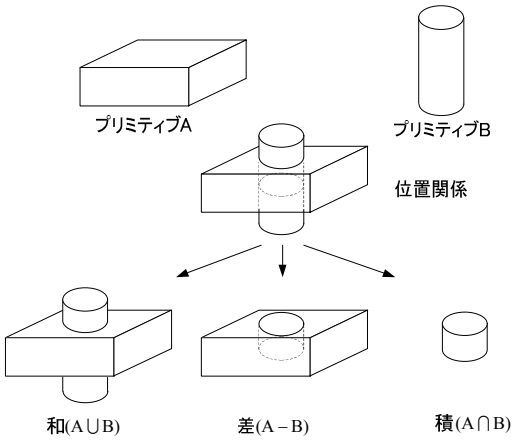


図 2.9　CSG 表現

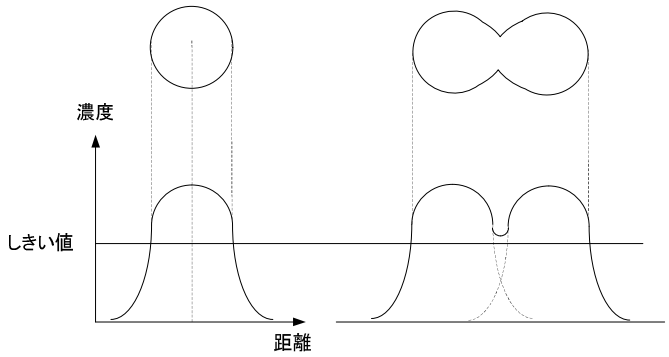


図 2.10　メタボール

また、稜線を中心とした隣接する形状要素を記述する**ウイングドエッジデータ構造**、立体を小さな立方体の集合で表現する**ボクセル表現**、ボクセル表現のデータ量を減らすよう考案され、立体を異なる大きさの立方体の集合で表現した**オクトリー表現**、２次元形状を一定方向に移動させることで立体を生成する**スイープ表現**がある。

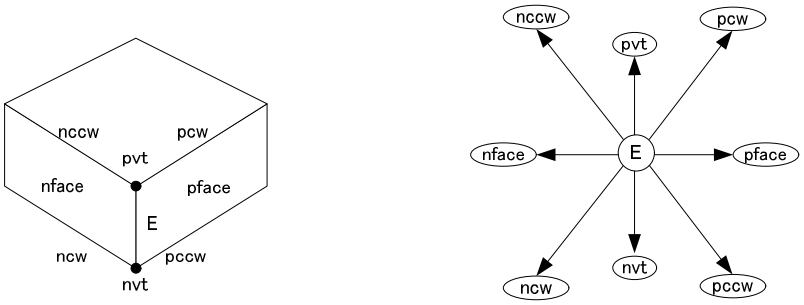


図 2.11　ウイングドエッジデータ構造

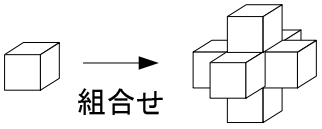


図 2.12　ボクセル表現

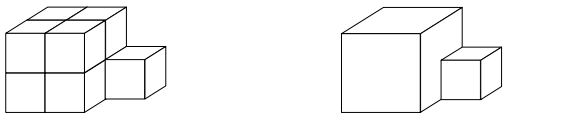


図 2.13　オクトリー表現

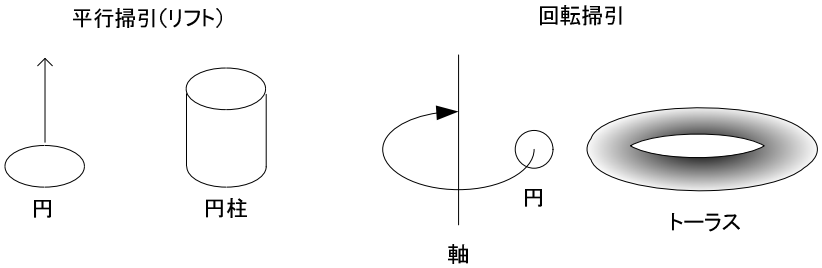


図 2.14　スイープ表現

3. レンダリング

レンダリングとはモデルに色付けをして表示を行う技術であり、その主要とされる処理には「シェーディング」や「テクスチャマッピング」がある。「シェーディング」とは物体の陰影を表現する方法であり、視線の方向と光源からの光の方向との関係で生じる物体の陰影を数学的に求める方法である。また、「テクスチャマッピング」とは 3 次元形状に画像を張り付けることで質感を高める方法である。

3.1 シェーディング（shading）

光源（太陽や蛍光灯など）から出た光は物体において反射や透過などを行い、人間の目に届くことになる。その際の視点に到達する光の強度をモデル化し、計算を行う方法をシェーディングという。この手法によれば、物体表面の色の濃淡（陰影）や表面反射などの質感を表現することができ、一般的には次の４つの光の成分を考慮する。

- 拡散反射
- 鏡面反射
- 環境光
- 屈折光

3.1.1 拡散反射（diffuse reflection）

拡散反射成分とは物体表面であらゆる方向に均等に反射（散乱する）光の成分であり、この成分を考慮するとざらついた光沢のない物体、例えば布や砂地などが表現できる。物体の面の明るさは面の傾きと光源の位置によって決定し、視点の位置に依存しない。物体の持つ色を反映しながら物体自身が作る陰ができるため、物体の濃淡が現れる。

3.1.2 鏡面反射（specular reflection）

鏡面反射成分とは鏡や磨かれた金属の表面で起こる一定方向への反射である。面の法線ベクトルに対する光の正反射方向（入射角と反射角が等しくなる方向）に対して強い反射が発生する。正反射方向から見るとハイライト（光源の色を反映した明るいスポット）が見える。

3.1.3 環境光（ambient light）

環境光とは空気による散乱光や他の物体からの二次反射光などによって間接的に物体に届く光の成分である。環境光は正確にシミュレートすると多大な計算コストとなるため、全方向から物体が弱く照らされているとみなす。これによって物体の影の部分を含め、ある程度の明るさを持った状態として表現される。例えば球をＣＧで表し、拡散反射成分のみを考慮して環境光を考慮しないならば、夜空に浮かぶ月のように真っ暗な部分（月が欠けた状態）が見える。

3.1.4 透過光（transmitted light）

透過光とはガラスなどの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。物体に入射した光はその表面で屈折し、透過中には強度が減衰する。そのため、透明物体を通して見た場合、背景がゆがんでぼんやりと見える。簡略的な表現では物体を半透明で表現し、物体の輝度と背景の輝度を適当な割合で合成する。その際には屈折の効果を考慮しない。

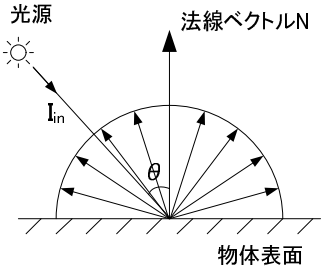


図 3.1 拡散反射成分

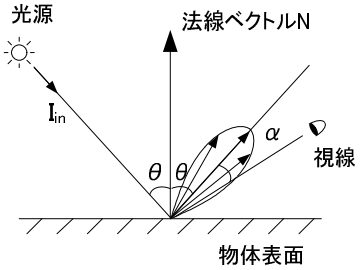


図 3.2 鏡面反射成分

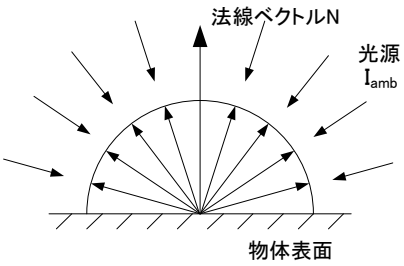


図 3.3 環境光成分

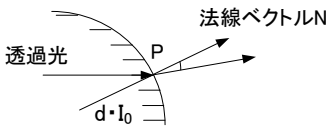


図 3.4 透過光成分

3.2 シェーディング技術

CG では物体表現をポリゴンの集合で近似した多面体モデルとして表すことが多い。そのため単一のポリゴンでは法線ベクトルはどの場所でも等しくなるため、シェーディングの結果、ポリゴンが単色で塗りつぶされる。これを**フラットシェーディング**（flat shading）または**コンスタントシェーディング**（constant shading）という。

これに対して多面体モデルのシェーディングを滑らかにし、擬似的に曲面に見せる手法を**スムーズシェーディング**（smooth shading）といい、**グーローシェーディング**（Gourand shading）と**フォンシェーディング**（Phong shading）の 2 種類がある。OpenGL ではフラットシェーディングとグーローシェーディングを利用することができる。グーローシェーディングは物体を構成するポリゴンが比較的大きくてもある程度は滑らかに見えるため、高速に画像を生成することができる。ただし、間接光や屈折光の追跡については OpenGL のライブラリには含まれないので別途プログラムを書く必要がある。スムーズシェーディングについて、以下に概要を示す。

3.2.1 グーローシェーディング

グーローシェーディングではポリゴンの頂点のみについて輝度計算を行う。内部の各点では頂点の輝度を線形補間することで対応する。計算量が比較的少ないが、ハイライトの見え方が不自然になる場合がある。以下に手順を示す。

(1)隣接ポリゴンによって共有される頂点の法線ベクトルを各ポリゴンの法線ベクトルの平均値とする。
(2)(1)の法線ベクトルを用いて輝度計算を行う。
(3)(2)で求めた頂点の輝度を用いて、多角形内部の各点の輝度の補間で求める。これを輝度補間法という。

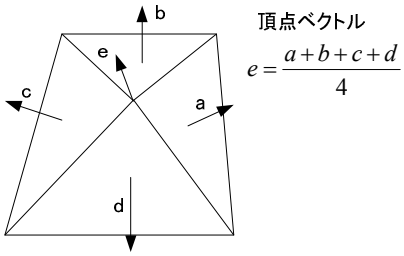


図 3.5 法線ベクトルの平均化

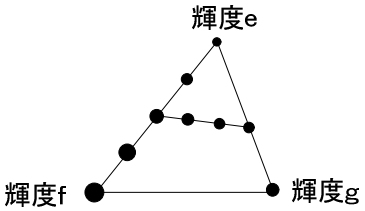


図 3.6 輝度の補間

3.2.2 フォンシェーディング

フォンシェーディングではポリゴンの頂点の法線ベクトルを用いてポリゴン内部の各点の法線ベクトルを線形補間してから輝度計算を行う。そのため、グーローシェーディングよりも自然な表現となる。ただし、多角形内部の点全てに対して輝度計算を行うため計算量が多くなる。

手順
(1)隣接ポリゴンによって共有される頂点の法線ベクトルを各ポリゴンの法線ベクトルの平均値とする。
(2)(1)で求めた法線ベクトルを用いて多角形内部の各点の法線ベクトルを補間で求める。
(3)(2)で求めた法線ベクトルを用いて輝度計算を行う。これを法線補間法という。

3.3 テクスチャマッピング

マッピングデータを３次元オブジェクトに表面の曲率を考慮して貼り付ける処理を**テクスチャマッピング**（texture mapping）という。マッピングデータには以下がある。

・画像データ（２次元の表面テクスチャ）
・ソリッドテクスチャ（オブジェクト内部も定義する）
・法線ベクトルのデータ

いくつかの例を挙げたが、狭義のテクスチャマッピングでは画像データをマッピングすることで行い、以下の手順で行う。

（１）ボディ座標系で表現された対象物体に画像データを写像する。
（２）対象物体をワールド座標系に変換する。
（３）ワールド座標系から２次元ディスプレイ平面へ投影する。

ここで、**ワールド座標系**とは形状が全体的（グローバル）に定義される座標であり、**ボディ座標**とは個別の形状において局所的（ローカル）に定義される座標である。ボディ座標系において、形状を定義している座標値に平行移動などの適当な変換を施すことにより、ワールド座標系で記述された形状に変換できる。

具体的には、マッピングしようとする面を２変数のパラメータ（u,v）で表し、マッピングデータの対応する画素の位置を決める処理が必要となる。例えば、**m**、**n** を平面上にある互いに独立なベクトル、**o** は位置ベクトルとし、テクスチャデータの各画素の座標値（u,v）座標系で表すと、平面上の点 **p** は以下で表すことができる。

$$\mathrm{p} = u\mathrm{m} + v\mathrm{n} + \mathrm{o} \qquad (3.1)$$

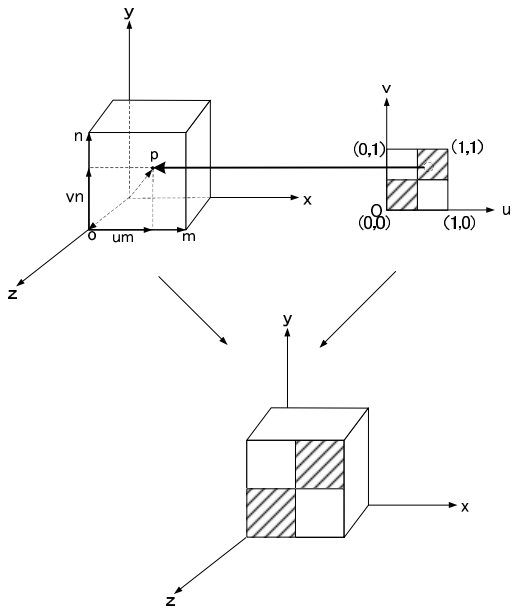


図 3.7　テクスチャマッピング

また、マッピングデータに法線ベクトルを用いる方法を**バンプマッピング**（bump mapping）といい、物体の細かい凹凸を表すのに使う。通常グレースケール画像として与え、輝度の変化を法線ベクトルの傾きとして計算して処理を行うことが多い。

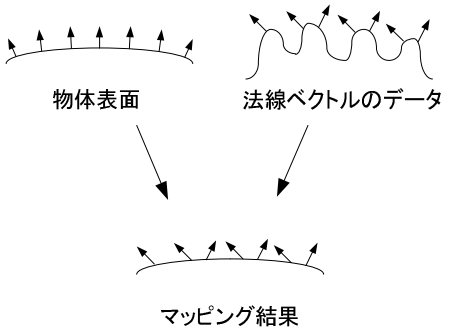


図 3.8　バンプマッピング

また、背景や前景を擬似的に物体にとりこませる方法として、**環境マッピング**（environment mapping）がある。環境マッピングは画像をマッピングデータとするが、貼り付ける画像は背景や前景に利用している画像である。貼り付ける画像データをどの位置から拾ってくるかによって、**反射マッピング**（reflection mapping）と**屈折マッピング**（refraction mapping）の違いがある。

反射マッピングでは、鏡面反射を考慮して画像をマッピングし、屈折マッピングでは、透明物体に対して光の屈折を考慮して画像をマッピングする。

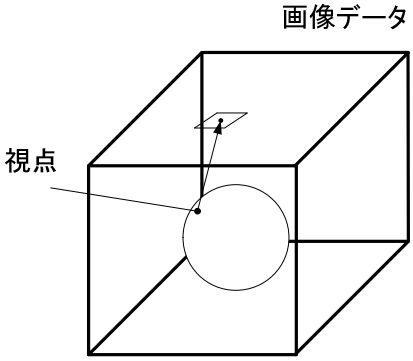


図 3.9　反射マッピング

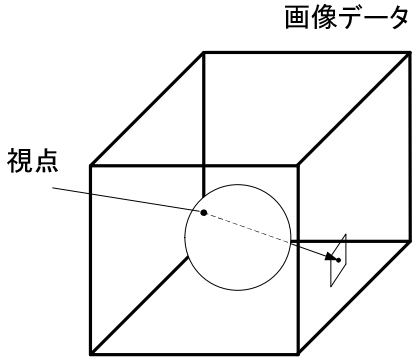


図 3.10　屈折マッピング

ここまでのマッピングデータでは、物体表面にデータを貼り付けただけであるので、物体を切断した形状を表示すると、内部にマッピングデータによる模様は無い。アニメーションに利用するには変形した形状に合わせた画像をそのたびに用意しなくてはならないために手間がかかる。その問題を解決するのが物体の内部空間も含めてマッピングデータを用意する**ソリッドテクスチャ**（solid texture）と呼ばれる手法である。マッピングデータは、通常、数式やアルゴリズムによってパターンを発生させ、木材や岩石のようにテクスチャが内部に連続している物体を容易に表現できる。

4. 変換

形状モデルに対して処理前の形状データから処理後の形状データへ数学的に置き換える処理を変換という。変換には大きく分けて、「投影変換」と「幾何変換」が存在する。「投影変換」とは、例えば、

３次元形状モデルをグラフィックディスプレイの２次元面に表示するといった３次元から２次元への変換がある。「幾何変換」とは定められた形状に対して、移動や回転、拡大・縮小といった操作を行う際に利用する。また、視点に移動や回転といった操作を行う視野変換も幾何変換の一例である。

4.1 変換処理

３次元形状の座標データを元に、仮想的に設定するスクリーン平面である２次元面へと変換する処理を投影変換という。また、形状モデリングで定義されたデータに対して、平行移動、回転、拡大・縮小などの幾何的に変換する処理を幾何変換という。また、幾何変換の例として、視線方向の移動回転を行う視野変換やパーツごとに形状モデリングした後、全体を統合するための座標系に変換する座標変換などがある。

4.2 同次座標と射影変換

通常座標（ordinary coordinate）として、３次元空間における点 P の位置について、直交座標系 o-xyz である（x, y, z）が定まるとすると、この点 P に対して、４個の実数の組を（W, X, Y, Z）を用い、

$$x=X/W,\ y=Y/W,\ z=Z/W \qquad (4.1)$$

を満たすものを**同次座標**（homogeneous coordinate）という。この場合、任意の実数$\lambda \neq 0$に対して（λW, λX, λY, λZ）も点Pの同次座標となる。ここで、W＝0 の場合には（0, X, Y, Z）は［X, Y, Z］方向の無限遠点を表すものとしている。同次座標（W, X, Y, Z）で表される点全体からなる空間を**射影空間**（projective space）といい、同次座標で表された点（W, X, Y, Z）を行列[A]を用いて、（W', X', Y', Z'）へ移す変換、

$$[W' \quad X' \quad Y' \quad Z'] = [W \quad X \quad Y \quad Z][A] \qquad (4.2)$$

を**射影変換**（projective transformation）という。この射影変換において、W' = W の場合を**アフィン変換**（affine transformation）という。例として、通常座標（x, y, z）、（x', y', z'）において、W' = W =1 として、

$$[1 \quad x' \quad y' \quad z'] = [1 \quad x \quad y \quad z][A] \qquad (4.3)$$

として用いる。一般的には

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1][A] \qquad (4.4)$$

として用いる。

アフィン変換に対して、通常座標（x, y, z）、（x', y', z'）において、

$$[x' \quad y' \quad z'] = [x \quad y \quad z][A] \qquad (4.5)$$

とした変換を**線形変換**（linear transformation）という。拡大・縮小・反転や回転は線形変換として表現することが可能であるが、平行移動は行列の和として表さなくてはならない。アフィン変換を用いることで平行移動も含め統一的に表現することが可能となる。

4.3 幾何変換

以下ではアフィン変換を用いた 2 次元幾何変換を解説していく。

4.3.1 幾何変換

説明を簡単にするため、2 次元座標系を元に解説する。図形要素（線分や円など）の拡大・縮小、移動を数学的な操作で変換することを**幾何変換**（geometric transformation）という。変換前の点 P を（x, y)、変換後の点 P'を（x', y'）とすると、点の幾何変換における、幾何変換の一般公式は以下となる。

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & c & 0 \\ b & d & 0 \\ m & n & 1 \end{bmatrix} \qquad (4.6)$$

↑

アフィン変換における**変換マトリクス**（transformation matrix）

以下に、点の平行移動、拡大・縮小・反転、回転を示す。

・平行移動

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

x' = x+m　　y'=y+n

m、 n は x 方向、 y 方向への移動量

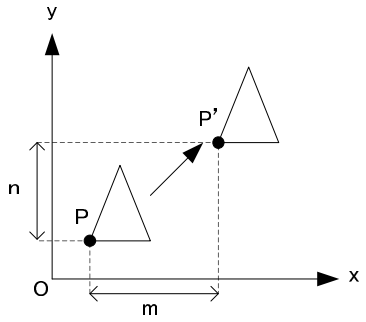


図 4.1　平行移動

・拡大・縮小・反転（スケール変換）

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

x' = ax　　y'=dy

a>1, d>1 の場合　　a, d は x 方向、 y 方向への拡大

0<a<1, 0<d<1 の場合　a, d は x 方向、 y 方向への縮小

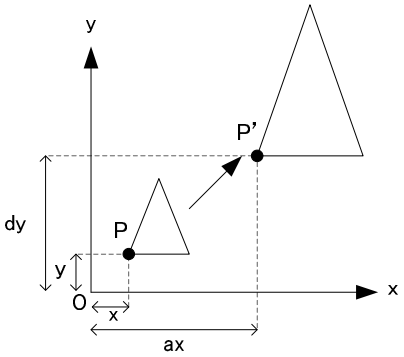


図 4.2　拡大・縮小

反転について

a=1, d=－1 の場合　　x’= x　y’=－y　→　x 軸についての反転　P_1’

a=－1, d=1 の場合　　x’= －x　y’=y　→　y 軸についての反転　P_2’

a=－1, d=－1 の場合　　x’= －x　y’=－y　→　原点についての反転　P_3’

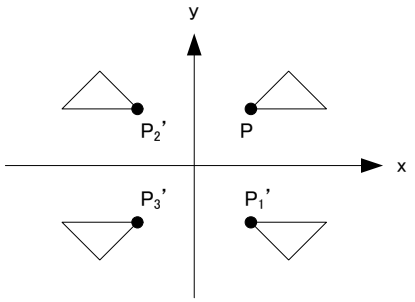


図 4.3　反転

・回転

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

θ は原点に対して反時計回りに回転させる角度

$$\left.\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned}\right\}$$

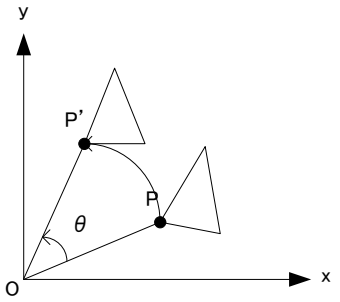


図 4.4　回転

・変換マトリクスの各変数を適宜指定することで、各種の幾何変換を行うことができる。 ・拡大・縮小・反転、回転は線形変換で表現可能、平行移動を含み、統一的に扱うには 　アフィン変換を用いなくてはならない。

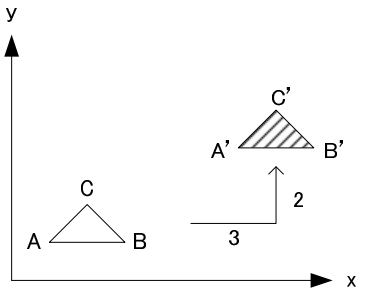
また、図形に対しての幾何変換はそれを構成する点の全てについて幾何変換を施すことになる。例えば、三角形 ABC の各頂点座標を A（x_1, y_1）、B（x_2, y_2）、C（x_3, y_3）とし、変換後の三角形 A’B’C’の各頂点座標を A（x_1, y_1）、B（x_2, y_2）、C（x_3, y_3）とすると、三角形の幾何変換は以下の変換式で表すことができる。

$$\begin{bmatrix} x_1' & y_1' & 1 \\ x_2' & y_2' & 1 \\ x_3' & y_3' & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} \text{変換マトリクス} \end{bmatrix}$$

例　△ABC の各頂点　A（1,1）、B（3,1）、C（2,2）について、x 方向に 3、y 方向に 2 平行移動した変換後の△A’B’C’を求めよ。

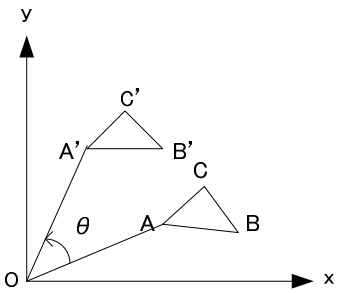
x 方向に 3、 y 方向に 2 平行移動するので

$$\begin{bmatrix}1 & 1 & 1\\ 3 & 1 & 1\\ 2 & 2 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 3 & 2 & 1\end{bmatrix}=\begin{bmatrix}4 & 3 & 1\\ 6 & 3 & 1\\ 5 & 4 & 1\end{bmatrix}$$



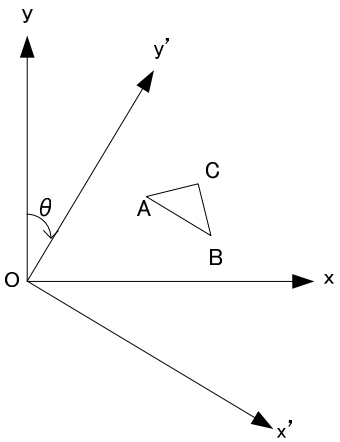
4.3.2 幾何変換と座標変換

座標変換（coordinate transformation）は o-xy 座標系で定義した形状を o'-x'y'座標系で表現する場合に用いる。この変換は与える角度や距離を逆向きに幾何変換を行うことに等しい。回転処理を例にとると、以下の図のように、反時計回り方向を正として、図形を角度 θ で原点を中心として回転を行う処理は、o-xy 座標系を角度- θ で回転した場合の結果に等しいこと意味する。



図形を角度 θ で回転

図 4.5　幾何変換



座標系 o-xy を角度－ θ で回転

図 4.6　座標変換

幾何変換と座標変換の対応表（2 次元）

	平行移動	回転	拡大・縮小	反転	
				x 軸	y 軸
幾何変換	$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ m & n & 1\end{bmatrix}$	$\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}a & 0 & 0\\ 0 & d & 0\\ 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$
座標変換	$\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ -m & -n & 1\end{bmatrix}$	$\begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}1/a & 0 & 0\\ 0 & 1/d & 0\\ 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 0\\ 0 & -1 & 0\\ 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}-1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$

5 投影変換

5.1 投影変換の一般式

以下の図に示すように、$\mathrm{B}(x,y,z)$ を 3 次元空間内の任意の点、$\mathrm{E}(x_e,y_e,z_e)$ を視点、$\mathrm{P}(x',y',z')$ を

投影面上の点とすると、

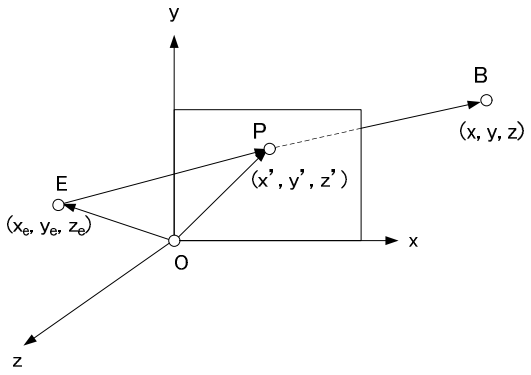


図 5.1　投影変換

ベクトル$\overrightarrow{\mathrm{OP}}$は、$\overrightarrow{\mathrm{OP}}=\overrightarrow{\mathrm{OE}}+\overrightarrow{\mathrm{EP}}$で表され、$\overrightarrow{\mathrm{OP}}=\overrightarrow{\mathrm{OE}}+\alpha\cdot\overrightarrow{\mathrm{EB}}$である。また、各点の座標値でこの式を表すならば、以下となる。

$$[x' \quad y' \quad z']=[x_e \quad y_e \quad z_e]+\alpha[x-x_e \quad y-y_e \quad z-z_e] \quad (5.1)$$

ここで、投影面は xy 平面であるので、$z'=0$より

$$z'=z_e+\alpha(z-z_e)=0 \qquad (5.2)$$

よって、

$$\alpha=-z_e/(z-z_e) \qquad (5.3)$$

式(5.1)、(5.3)より、

$$x'=x_e+\alpha(x-x_e)=\frac{x_e z-z_e x}{z-z_e} \qquad (5.4)$$

$$y'=y_e+\alpha(y-y_e)=\frac{y_e z-z_e y}{z-z_e} \qquad (5.5)$$

となり、式(5.4)、(5.5)は投影の一般式となる。これらの関係式はこのままでは使い勝手が悪い。幾何変換の時と同様に行列の積で演算を行うことができるように、投影の一般式を同次座標で表し、射影変換による表現に変更する。

5.2 射影変換による表現

$\mathrm{B}(W,X,Y,Z)$を 3 次元空間内の任意の点(x,y,z)の同次座標、$\mathrm{E}(W_e,X_e,Y_e,Z_e)$を視点$(x_e,y_e,z_e)$の同次座標、$\mathrm{P}(W',X',Y',Z')$を投影像の点$(x',y',z')$の同次座標とすると、以下の式が成り立つ。

$$x = X/W, \qquad y = Y/W, \qquad z = Z/W \qquad (5.6)$$
$$x_e = X_e/W_e, \qquad y_e = Y_e/W_e, \qquad z_e = Z_e/W_e \qquad (5.7)$$
$$x' = X'/W', \qquad y' = Y'/W', \qquad z' = Z'/W' \qquad (5.8)$$

式(5.4)、(5.5)式に上式を代入すると

$$x' = \frac{X'}{W'} = \frac{ZX_e - Z_eX}{ZW_e - Z_eW} \qquad (5.9)$$
$$y' = \frac{Y'}{W'} = \frac{ZY_e - Z_eY}{ZW_e - Z_eW} \qquad (5.10)$$

となる。また、z'=0 であるので、これを射影変換により表現すると

$$\begin{bmatrix} W' & X' & Y' & Z' \end{bmatrix} = \begin{bmatrix} W & X & Y & Z \end{bmatrix} \begin{bmatrix} -Z_e & 0 & 0 & 0 \\ 0 & -Z_e & 0 & 0 \\ 0 & 0 & -Z_e & 0 \\ W_e & X_e & Y_e & 0 \end{bmatrix} \qquad (5.11)$$

となる。

ここで、$W_e = X_e = Y_e = 0$、$Z_e \neq 0$ の場合を**平行投影**（parallel projection）という。また、平行投影のうち投影面と投射線の角度が垂直であれば**垂直投影**（orthographic projection）という。また、$W_e \neq 0$、$Z_e \neq 0$ の場合を**透視投影**（perspective projection）、あるいは中心投影（central projection）という。平行投影については図 5.2 から確認できるように、奥行き情報を切り捨てることに等しいので、以下の式で直接、座標値 x'y'z'を求めることができる。

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.12)$$

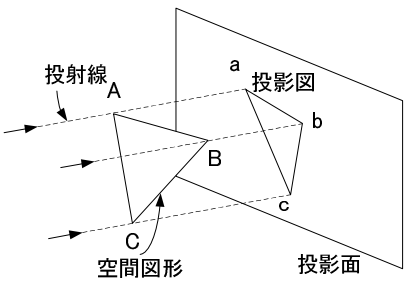


図 5.2　平行投影

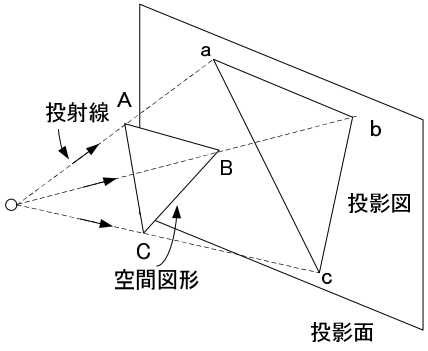


図 5.3　透視投影

参考文献

この文章は以下の文献を参考にしました。

1. コンピュータグラフィックス　前川佳徳　著　オーム社出版局