コンピュータグラフィックスＩ（蒔苗）　授業資料（2005）

コンピュータグラフィックスとは

コンピュータグラフィックス(computer graphics)

「コンピュータを使って図形や画像を作成，処理すること」

日経 BP 社　エンジニア's ハンドブックより

コンピュータグラフィックスの発達 <<コンピュータ技術の飛躍的進歩

情報伝達の手段の変革

テキスト情報から画像情報へ

例えば新聞からラジオ，そしてテレビへ

つまり画像情報はより多くの情報を伝達することが可能.

アナログ情報からデジタル情報へ

レコードから CD へ

カメラ（フィルムからデジタルカメラへ)

アナログ放送からデジタル放送へ

アナログ携帯電話からデジタル携帯電話へ

>>デジタル化による情報の伝達の効率化

（必要な情報のみ選択的に伝達することができる.）

アナログ情報の方がより多くの情報量を含んでいる場合が多い.

（例えば写真，デジタルカメラとフィルム現像との比較）

>>デジタルによる情報伝達では，情報の劣化が生じない.

（現状の A/D 変換，D/A 変換の積み重ねにおける情報の劣化が生じない）

コンピュータグラフィックスの応用技術

ベクタグラフィックスの発達 >> CAD の発達と大きく関連している.

1952 年　MIT　NC(numerical control)工作機の開発

1959 年　MIT　CAD の概念

1963 年　MIT　SKETCHPAD の開発

以降，1960 年代，CAD/CG の基礎的技術が構築される.

1970 年代－1980 年代　２次元 CAD の普及期

1990 年代　3 次元 CAD の普及期

2000 年代　3 次元 CAD の実用期と高度化（CAM，ロボット等との連携）

ラスタグラフィックスの発達>>宇宙・軍事技術から発展

1960 年代後半から 1970 年代

人工衛星からのデジタル情報の可視化>>例えばリモートセンシング

1970 年代後半から 1990 年代

より精度の高い情報の伝達，画像処理技術の高度化

ベクタグラフィックスとラスタグラフィックスの融合　>>　いわゆる「ＣＧ」

ベクタグラフィックスとラスタグラフィックスの融合

1990 年代－

マルチメディア化

3 次元 CG 技術の高度化及びその応用が進む→VR(virtual reality)技術へ

2000 年代－

MR(Mixed Reality)への発展

コンピュータグラフィックスの応用分野

ゲーム・エンターテイメント

各種産業（機械・医療・建築・土木などその応用はさまざまである）

最新のＣＧ技術の紹介

２次元 CG の基礎(ラスタとベクタ)

１．ラスタグラフィックス raster graphics

ラスタグラフィックス

ディスプレイを構成する１つ１つの画素について属性を持たせることにより，それを画像として表現する．

画素（pixel）

ラスタスキャンディスプレイ上に表現される画像情報の最小単位であり，画像の解像度を左右する．

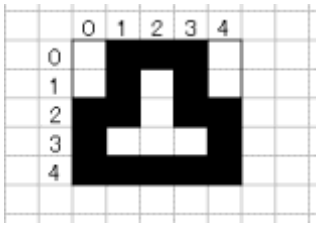
VGA ディスプレイ：640 * 480pixel (=307200 pixel)
XGA ディスプレイ：1024*768 pixel(=786432 pixel)
SXGA ディスプレイ：1280*1024 pixel (=1310720 pixel)

ef. ビデオカメラ等の有効画素数

ラスタグラフィックスの基本的な考え方（白黒２値化）

２値化によるグラフィックス　　例：ビットマップフォント

例えば以下のような格子を定義する．



これを2値情報（0か1か）としてデジタル化（符号化）すれば，

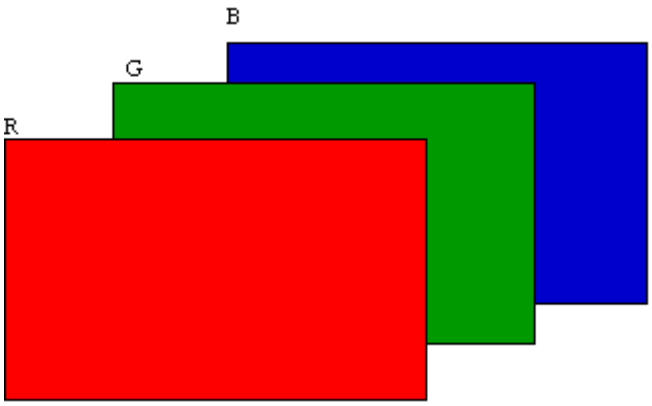
01110
01010
11011
10001
11111

となり，つまり 5*5 ビット＝25 ビット(bit)のデータとして表現される．

コンピュータ上でのカラー表現

コンピュータ上でのカラー表現は，光の３原色（赤(R)，緑(G),青（B)）を合成して得られ

る.



８色カラー表現

1 pixel　に対して，R に 1 bit(0 か 1)，G に 1 bit(0 か 1)，B に 1 bit(0 か 1)を与える.

RGB : 000 (2進数) = 0 (10 進数)　黒
RGB : 001 (2進数) = 1 (10 進数)　青
RGB : 010 (2進数) = 2 (10 進数)　緑
RGB : 011 (2進数) = 3 (10 進数)　シアン
RGB : 100 (2進数) = 4 (10 進数)　赤
RGB : 101 (2進数) = 5 (10 進数)　マジェンタ
RGB : 110 (2進数) = 6 (10 進数)　黄
RGB : 111 (2進数) = 7 (10 進数)　白

となる.

必要となるデータ量は，3 bit/pixel であるから，
VGA : 640 * 480 * 3 =921600 bit = 115200 byte
XGA : 1024 * 768 * 3 = 2359296 bit = 294912 byte
となる.

16 色表現（グレースケールあるいはカラー表現）

1　pixel に対して 4 bit のデータを与えるとする.
2 ^ 4 = 16　(10 進数)
したがって 0 から 15 に対応する色を決定すれば，
16 色の表現が可能となる.
　>>コンピュータ上に Look Up Table (LUT;カラーパレット) を定義する.

640*480pixel のディスプレイ(VGA)で2値化表現する場合のデータ量は,
640 * 480 * 4 (bit) = 1228800 bit = 153600 byte
となる.

256 色表現（グレースケールあるいはカラー表現）

1 pixel に対して 1 byte(8 bit) のデータを与えるとする.
1byte = 8 bit なので, 2^8 = 256 （10 進数）
したがって 0 から 255 に対応する色を決定すれば,
255 色の表現が可能となる.
>>コンピュータ上に Look Up Table (LUT;カラーパレット) を定義する.

そのデータ量は,
VGA: 640 * 480 * 1 byte = 307200 byte
XGA: 1024 * 768 * 1 byte = 786432 byte
となる.

1677 万色のフルカラー表現

1pixel に対し, R,G,B それぞれに対して 1 byte (0-255 の 256 段階)に分離したとすれば, 1pixel の表現に対し, 3byte(24bit)が必要となる.

表現可能な色数は,
R の色数* G の色数 * B の色数 256 * 256 * 256 = 16777216
したがって, 1677 万色の表現が可能となる.

そのデータ量は,
VGA: 640 * 480 * 3 byte = 921600 byte
XGA: 1024 * 768 * 3 byte = 2359296 byte
SXGA:
となる.

その他, 一般的に用いられている階調数として, 4096 色, 32768 色, 65536 色, 262144 色などがある.

注)上述のデータ量は，グラフィックスボードにおいて必要となるメモリ（VRAM）の量となる．

問題：
ノートパソコンに一般的に用いられている 1024×768 の解像度(XGA)において，1677 万色フルカラー表現する場合のデータ量を求めなさい．

アナログ画像のデジタル化

デジタルデータの特徴

1．コンピュータ上での処理が可能である．（加工，再利用等が容易）
2．データの劣化が少ない．
　（ノイズに強い，コピーしても劣化しない）
3．データ圧縮，暗号化が可能

デジタル化の問題

無段階に表現されるアナログ画像に対して，デジタル化による画像の劣化は否めない．

デジタル化によって処理可能な情報量はコンピュータ性能に制約される．現状の技術レベルからすれば，その情報量が膨大なものとなり，実質的にデジタル画像の分解能はアナログ画像より低い場合が多い．

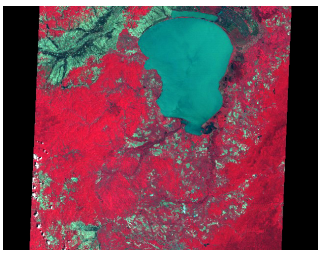
ただし画像による情報の伝達という本来の目的において，実際にそこまで精細な情報が本当に必要であるのかを考えてみれば，デジタル化された画像で十分に対応できるケースがほとんどである．

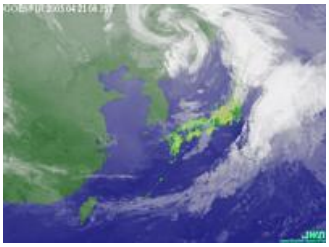
画像のデジタル化（A/D 変換）

標本化（サンプリング），量子化，符号化の過程を経てデジタル化が行なわれる．

ラスタグラフィックスの応用技術

リモートセンシング技術（ランドサット，気象衛星画像，等）





（http://www.tenki.or.jp/及び http://www.restec.or.jp/より）

２．ベクタグラフィックス vector graphics

ベクタグラフィックス

座標値に基いたデータ系列をもとに，図形情報を格納しておき，それをコンピュータを用いて画像として表現する．

ラスタスキャンディスプレイに表現する場合には，コンピュータ上での処理により，ベクタからラスタへ変換し，表示することになる．

結局，ラスタとベクタの違いは画像情報としてのデータ構造の違いということである．

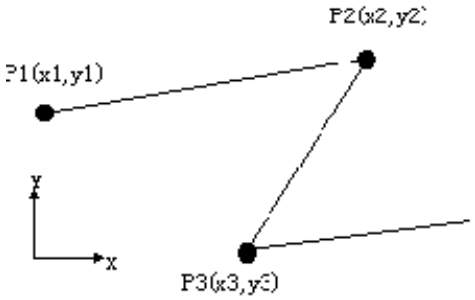
CAD (Computer Aided Design）あるいはドローソフトにおいて一般的に用いられている技術である．

ベクタグラフィックスにおける図形描画

平面上に描かれた線について，その線上の座標を，

P1(x1,y1)，P2(x2,y2)，P3(x3, y3).....

と定義すれば，以下のように線(P1-P2-P3-.....）を描くことができる

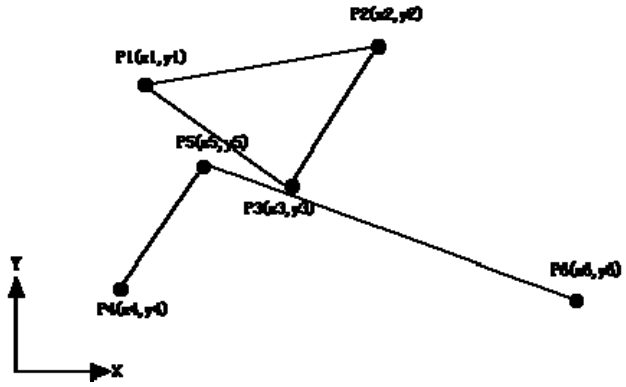


P1, P2, P3, P4, P5, P6　という点の座標を定義した場合に，

P1 と P2 と P3 を接続し，図形を閉じる．

P4 と P5 と P6 を接続し，図形は閉じない．

というように各点の接続条件を定めていくことにより，図形の描画が可能である．



例えば，データファイルの形式を下記のように定義し，コンピュータ上でそれを図形に再現することになる．（3 次元データの場合，CG ソフト PRISMS/ACTION における POLY Format）

POINTS
1:0.0 0.0 2.5
2:2.5 2.3 3.3
3:5.5 3.5 4.0
4:2.2 3.3 4.0
5:2.1 2.1 3.2
6:5.1 3.3 4.2
POLYS
1:1 2 3 <
2:4 5 6
POINTS の以下に座標値を置く．
POLYS の後に図形の接続条件を置く．
閉じる場合には，末尾に < を置く．

基本図形（プリミティブ）

線，円，曲線，多角形などについて，その形状を属性により決定する．
例えば，円の場合には中心座標と半径などにより，描画できる．

ベクタグラフィックスの利点と応用

図形情報をベクタ情報として有することにより，画像サイズ（解像度）にとらわれない描画が可能である．
設計におけるベクタデータの活用はさまざまな応用（CAM, CAE 等）が可能である．

ベクタグラフィックスの問題点

図形描画のためにコンピュータ上でのベクタ>>ラスタ変換の処理が必要であり，データ量が多くなればなるほど，処理に時間を要する．
曲線の描画における問題.

画像情報のデジタル化（色の基礎とデジタル化）

１．人間はどのように色を認識するか？

人間の網膜は光を認識する．（眼球の網膜>>錘体によって色覚，桿体による光の明暗が信号化され，大脳に伝達される） 錘体は，赤(R)，緑(G)，青(B)に特に反応する．>> 赤，緑，青の組合わせによって色が決定される．

可視光

光は電磁波の一種であり，人間が認識できる光の波長はおおよそ 400－700nm の範囲であり，これを可視光という．その波長により色が決定され，おおむね 500nm 以下は青，500-600nm は緑，600nm 以上は赤となる。可視光域すべての波長を等しく含む光は白色光とよばれる（例えば太陽光など）。

光の演色性

物の色は、蛍光灯の下と白熱灯の下とでは違って見える。これを光源の演色性といい、見える色はそれ自体では決まらず、光源の分光特性に左右される。

２．色の表現方式

色の表現は，強弱をもつ RGB の合成により可能である．いくつかの基本色を合成してさまざまな色を作り出すことが可能である（混色）．

加法混色

RGB の強弱ある組み合わせにより色を表現する方法である．

CRT ディスプレイには，RGB 各々の蛍光塗料が塗られており，それらに電子ビームをあてて光らせることで合成した色を作り出している．

減法混色

印刷で使われるインクの基本色は C（シアン；Cyan），M（マゼンタ；Magenta），Y（黄；Yellow）の３つであり，C は R を，M は G を，Y は B をそれぞれ吸収する．CMY のそれぞれの濃度により，RGB としての反射色が表現される．

（ただし，印刷においては CMYK が用いられる（K は黒)．インクの特性から CMY のみ重ねても黒にならないため）

３．カラーシステム

色の３属性

色相(hue)

スペクトルで分光された色の成分　赤，橙，黄，緑，青，藍，紫といった色合いの特徴をいう．

彩度(saturation)

色の鮮やかさをいう．

明度(lightness)

色の明るさをいう.

カラーシステム

マンセルカラー表色系

アメリカの画家であり美術の教師であったアルバート・H・マンセル(1858-1918)が 1905 年に発表した表色系．その後，アメリカ光学会(OSA)により改良され、現在のマンセル表色系が確立される.

マンセル表色系は色を３つの属性，すなわち色相，明度およびクロマ（Chroma；(C)；色の鮮やかさ）で表現する.

マンセル記号では H V/C の順に書き表す。

例　有彩色： 5R 8.0/2.0 、5GY 5/8

　　無彩色： N5(明度 5) 、N8(明度 8)

マンセル記号が示す色 http://www.aist.go.jp/RIODB/ssrdoc/

日本工業規格 JIS Z8721 もマンセルカラー表色系に準拠したカラーシステムが用いられる.

RGB システム

RGB の補色である CMY の値をもとに加法混色により色を定めるシステム.

CMY システム・CMYK システム

RGB の補色である CMY の値をもとに減法混色により色を定めるシステム.
黒を加えたものは CMYK システムである.

HSV システム

色相（hue)，彩度（Saturation)，明度(Value)により色を数値的に定義するカラーシステム.
（彩度を操作すると明度も変化する欠点がある）

HLS システム

色相（hue)，明度(Lightness)，彩度（Saturation）により色を数値的に定義する.

Lab カラースペース

彩度と色相を a(R-G 軸)，b（Y-B 軸）の２つのパラメータで表わし，それに明るさ（K-W 軸)を加えてできたカラースペースを言う.

CIE 表色系

国際照明委員会(CIE)が 1931 年に勧告したカラースペースで，デバイスによらない色空間として広く利用されている．CIE 表色系は，網膜の RGB に強く反応する錐体の特性を考慮してつくられた，心理物理的な表現空間である.
ここでは RGB のかわりに架空の３原色 XYZ が使われる.

X=0.478R+0.299G+0.1758

Y=0.263R+0655G+0.0818

Z=0.020R+0.160G+0.9099

(X,Y,Z)のかわりに(Y,x,y)を用いる場合もあり，この場合 Y は明るさ，(x,y)は色相を示す.

４．デジタル化

アナログ画像のデジタル化　（A/D 変換）

標本化　>>　量子化　>>符号化

デジタル化のためのツール

イメージスキャナ

デジタイザ

デジタルカメラ・デジタルビデオ

（補足）人間の立体認識機能

両眼による立体認識機能（立体視）　>>　バーチャルリアリティへ応用

1)水晶体調節　距離 2m

2)両眼輻輳　距離 20m まで

3)両眼視差　距離 600m まで　それ以上は単画像からの立体認識機能による

単画像（２次元画像）からの立体認識機能

1)視対象の見えの大きさによる判断

2)視対象の直線的遠近（パース効果）

3)視対象の重なり

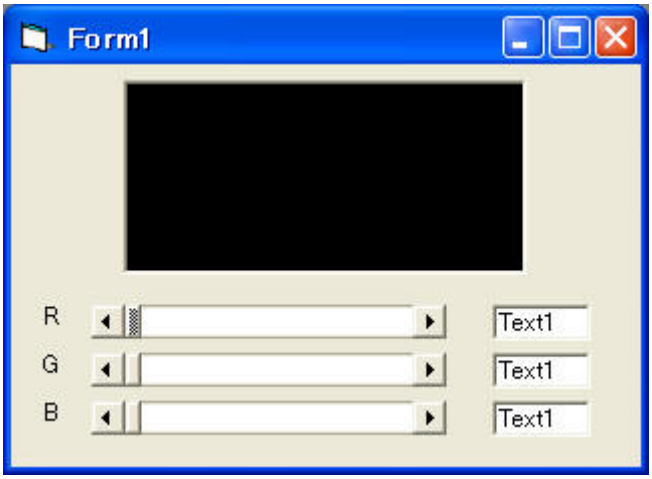
4)視対象の明瞭さ

5)陰による凹凸

6)視対象の肌理の密度勾配

7)視対象の相対運動

(演習)カラーシステム変換プログラム



1. RGB カラーシステム

```
'RGB システムによるカラー表現プログラム
Option Explicit
Private Sub Form_Load()
    Picture1.BackColor = RGB(0, 0, 0)

    Label1(0).Caption = "R"
    Label1(1).Caption = "G"
    Label1(2).Caption = "B"

    HScroll1(0).Max = 255
    HScroll1(1).Max = 255
    HScroll1(2).Max = 255

End Sub

Private Sub HScroll1_Change(Index As Integer)
    Dim R As Byte  '赤(0～255)
    Dim G As Byte  '緑(0～255)
    Dim B As Byte  '青(0～255)

    R = HScroll1(0).Value
    G = HScroll1(1).Value
    B = HScroll1(2).Value

    Text1(0).Text = R
    Text1(1).Text = G
    Text1(2).Text = B

    Picture1.BackColor = RGB(R, G, B)
End Sub
```

2. CMY カラーシステム(RGB への変換)

```
'CMY システムによるカラー表現プログラム
Option Explicit
Private Sub Form_Load()
    Picture1.BackColor = RGB(255, 255, 255)

    Label1(0).Caption = "C"
    Label1(1).Caption = "M"
    Label1(2).Caption = "Y"

    HScroll1(0).Max = 255
    HScroll1(1).Max = 255
    HScroll1(2).Max = 255

End Sub

Private Sub HScroll1_Change(Index As Integer)
    Dim R As Byte  '赤(0～255)
    Dim G As Byte  '緑(0～255)
    Dim B As Byte  '青(0～255)

    Dim C As Byte  'シアン(0～255)
    Dim M As Byte  'マジェンタ(0～255)
    Dim Y As Byte  'イエロー(0～255)


    C = HScroll1(0).Value
    M = HScroll1(1).Value
    Y = HScroll1(2).Value

    Text1(0).Text = C
    Text1(1).Text = M
    Text1(2).Text = Y

    R = 255 - C
    G = 255 - M
    B = 255 - Y

    Picture1.BackColor = RGB(R, G, B)
End Sub
```

３．HSV カラーシステム（RGB への変換）

```
'HSV システムによるカラー表現プログラム
Option Explicit
Private Sub Form_Load()
    Picture1.BackColor = RGB(0, 0, 0)
    Label1(0).Caption = "H"
    Label1(1).Caption = "S"
    Label1(2).Caption = "V"

    HScroll1(0).Max = 359
    HScroll1(1).Max = 255
    HScroll1(2).Max = 255

End Sub
```

```
Private Sub HScroll1_Change(Index As Integer)
    Dim R As Byte  '赤(0～255)
    Dim G As Byte  '緑(0～255)
    Dim B As Byte  '青(0～255)


    Dim H As Integer  '色相(0～359)
    Dim S As Byte '彩度(0～255)
    Dim V As Byte  '明度(0～255)

    Dim t1 As Integer
    Dim t2 As Integer
    Dim t3 As Integer
    Dim ht As Integer
    Dim d As Integer


    H = HScroll1(0).Value
    S = HScroll1(1).Value
    V = HScroll1(2).Value

    Text1(0).Text = H
    Text1(1).Text = S
    Text1(2).Text = V

    If S = 0 Then
        R = V
        G = V
        B = V
    Else
        ht = H * 6
        d = ht Mod 360
        t1 = CInt((255 - S) / 255 * V)
        t2 = CInt((255 - d / 360 * S) / 255 * V)
        t3 = CInt((255 - (360 - d) / 360 * S) / 255 * V)
    Select Case ht ¥ 360
        Case 0
            R = V: G = t3: B = t1
        Case 1
            R = t2: G = V: B = t1
        Case 2
            R = t1: G = V: B = t3
        Case 3
            R = t1: G = t2: B = V
        Case 4
            R = t3: G = t1: B = V
        Case Else
            R = V: G = t1: B = t2
        End Select
    End If
    Picture1.BackColor = RGB(R, G, B)
End Sub
```

画像処理アルゴリズム(1)

１．画像処理のモデル

M 行 N 列の画素からなるディジタル画像は,

$F = \{f_{ij}\}$

f_{ij} =第 i 行 j 列の画素の濃度値

$i = 1, 2, 3, \cdots, M \quad j = 1, 2, 3, \cdots, N$

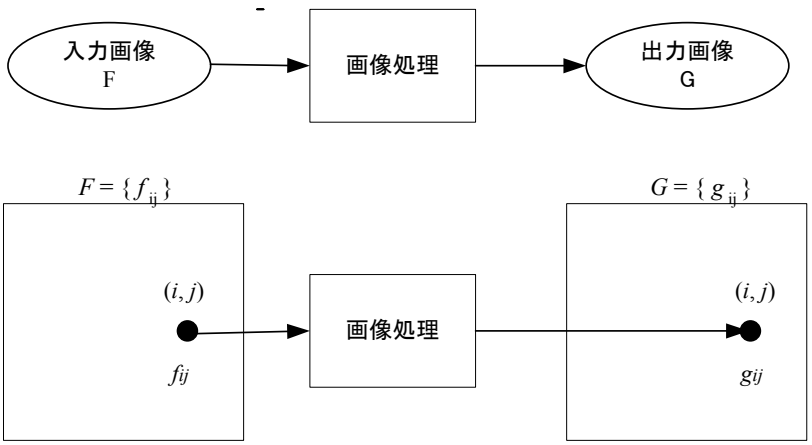
画像処理（狭義）とは,

- 画像の改善
- 画像の強調
- 図形の切り出し，セグメンテーション
- 画像認識

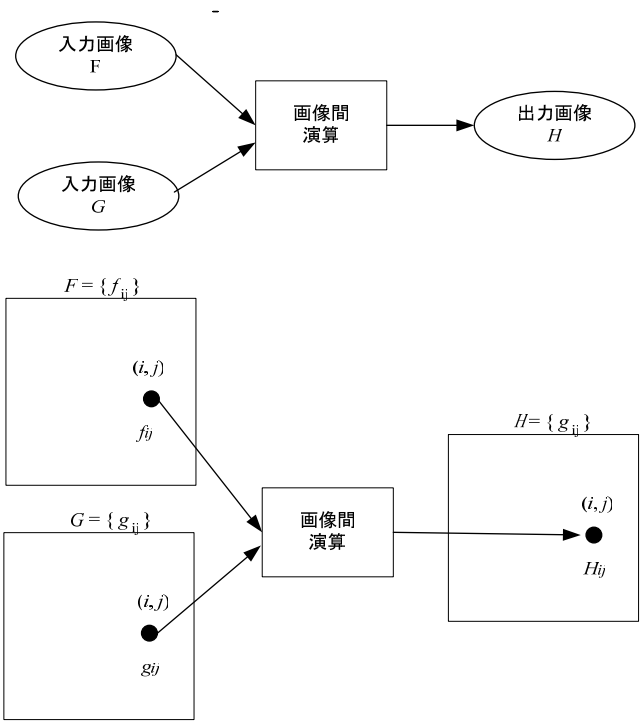
等の処理を行うこと.

画像処理は，ディジタル画像（入力画像）を基に別の画像（出力画像）をつくり出す作業である．画像処理アルゴリズムは，画像処理において，入力画像を基にして出力画像の各画素の濃度値の計算の仕方を言う.

(1)　１入力１出力の画像処理モデル

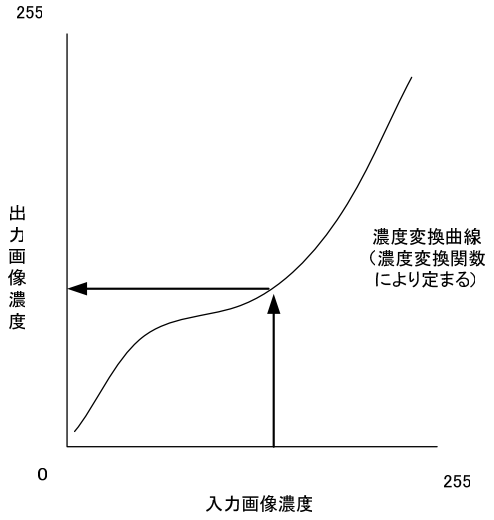


(2) ２入力１出力の画像処理モデル

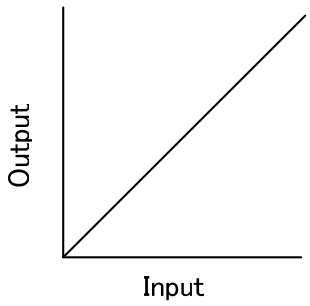


２．濃度変換

入力画像の各画素の濃度値に対する出力画像の濃度値を求める濃度変換関数を定め，それに基づき処理を行う．

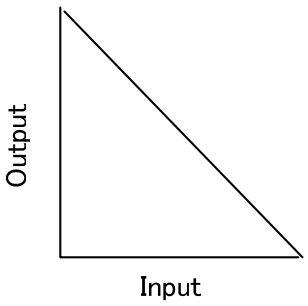


(1) リニア



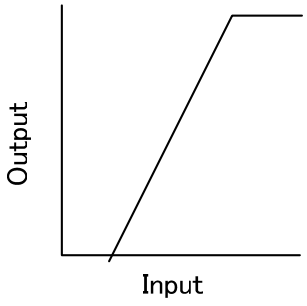
等値出力

(2)　レベルスライス



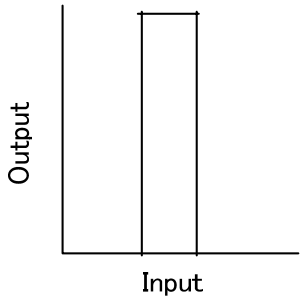
色調反転

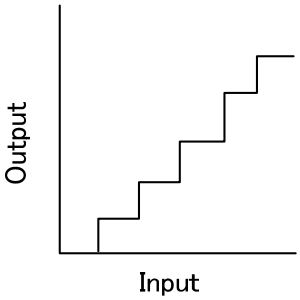
(3)階調化

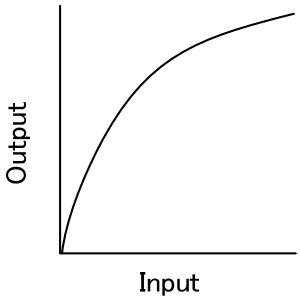


コントラスト調整

(4)ガンマ補正　$y=x^k$





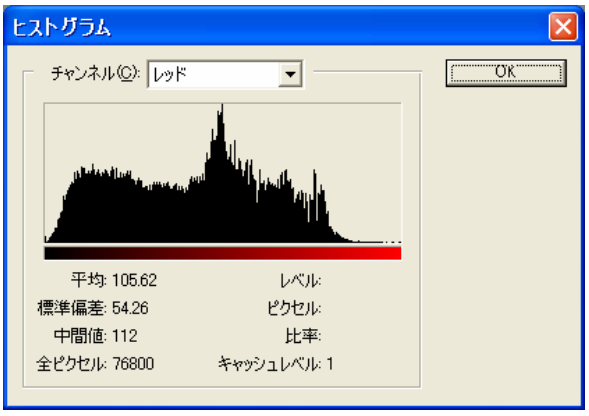


３．画像統計量

画像処理を行うにあたり，画像のもつ統計的な性質を調べ，それを基づいた処理を行う必要が生じる．ここでは代表的な画像統計量について示す．

(1) ヒストグラム（度数分布図）

画像中に各濃度の画素がどの程度の割合で存在するか（頻度）を調べ，グラフとして示したものである．



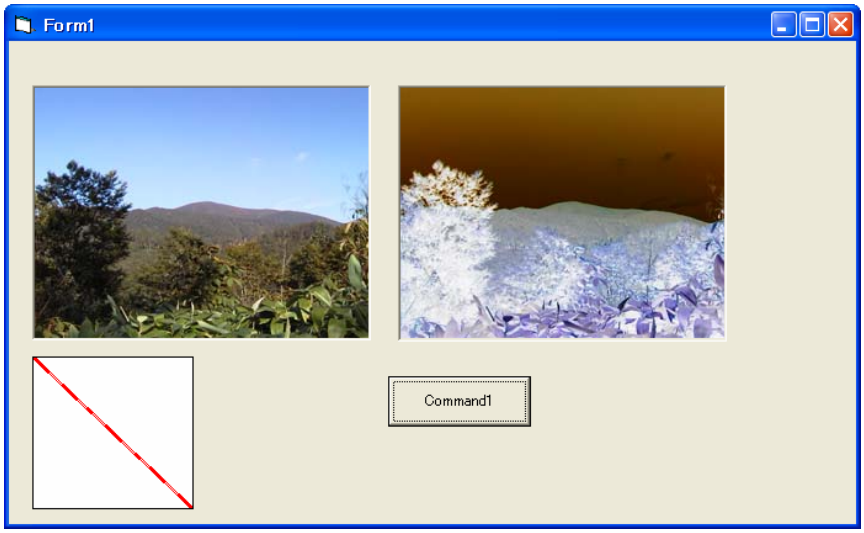
(Adobe Photoshop のヒストグラム画面)

(2) 最大・最小値

(3) 平均値
(4) 中央値
(5) 最頻度
(6) 分散
(7) 標準偏差

４．画像処理アルゴリズム（プログラミング例）

(1)濃度変換プログラム

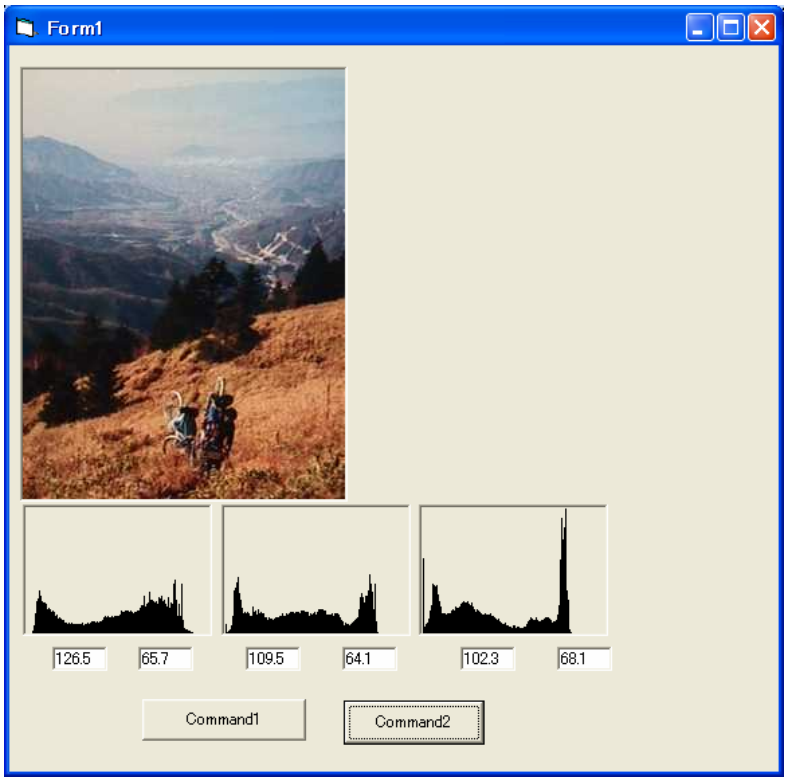



```
Option Explicit
Const linear = 0
Const slice = 1
Const gamma = 2


Private Sub Command1_Click()
    Dim orgColorL As Long
    Dim r As Integer, g As Integer, b As Integer
    Dim m As Integer, n As Integer

    Picture1.Cls
    Picture2.Cls
    Picture3.Cls

    Set Picture1.Picture = LoadPicture("c:¥maka¥cg¥pic¥funagata.jpg")

    For m = 0 To Picture1.ScaleWidth - 1
        For n = 0 To Picture1.ScaleHeight - 1
            orgColorL = Picture1.Point(m, n)


            r = imageFunc(linear, getR(orgColorL), -1, 255)
```

```
                g = imageFunc(linear, getG(orgColorL), -1, 255)
                b = imageFunc(linear, getB(orgColorL), -1, 255)



'               r = imageFunc(gamma, getR(orgColorL), 0.5, 0)
'               g = imageFunc(gamma, getG(orgColorL), 0.5, 0)
'               b = imageFunc(gamma, getB(orgColorL), 0.5, 0)

'               r = imageFunc(slice, getR(orgColorL), 128, 150)



                If r > 255 Then r = 255
                If g > 255 Then g = 255
                If b > 255 Then b = 255
                If r < 0 Then r = 0
                If g < 0 Then g = 0
                If b < 0 Then b = 0

                Picture2.PSet (m, n), RGB(r, g, b)
            Next
        Next
End Sub

Public Function getR(color As Long) 'Long の整数から R 成分を抜き出す関数を設定
        getR = color And &HFF&
End Function

Public Function getG(color As Long)    'Long の整数から G 成分を抜き出す関数を設定
        getG = (color And &HFF00&) / &H100&
End Function

Public Function getB(color As Long) 'Long の整数から B 成分を抜き出す関数
        getB = (color And &HFF0000) / &H10000
End Function

Public Function imageFunc(functype As Integer, x As Integer, k As Single, l As Single) As Integer
        Select Case functype
        Case linear
            imageFunc = k * x + l

        Case slice
            imageFunc = x
            If imageFunc > k And imageFunc < l Then
                imageFunc = 255
            Else
                imageFunc = 0
            End If
        Case gamma
            Dim a As Single
            a = x / 255
            imageFunc = (a ^ k) * 255
        End Select
```

```
        If imageFunc > 255 Then
            imageFunc = 255
        ElseIf imageFunc < 0 Then
            imageFunc = 0
        End If

        Picture3.Circle (x, imageFunc), 2

End Function

Private Sub Form_Load()
        Picture1.ScaleMode = vbPixels
        Picture2.ScaleMode = vbPixels
        Picture3.ForeColor = vbRed
        Picture3.Scale (0, 255)-(255, 0)
End Sub
```

(2)画像統計量を求める（ヒストグラム・平均・標準偏差）




```
Option Explicit
Const Red = 0
Const Green = 1
Const Blue = 2

Private Sub Command1_Click()
        Set Picture1.Picture = LoadPicture("c:¥maka¥cg¥pic¥top.jpg")
End Sub
```

```
Private Sub Command2_Click()
    Dim stat(0 To 2, 0 To 255) As Long
    Dim color As Long
    Dim m As Integer, n As Integer
    Dim ic As Integer
    Dim sum As Single
    Dim w As Integer, h As Integer
    Dim max As Long
    Dim r As Byte, g As Byte, b As Byte
    Dim average As Single
    Dim var As Single
    Dim stdev As Single

    For m = 0 To 2
        For n = 0 To 255
            stat(m, n) = 0
        Next
    Next


    w = Picture1.ScaleWidth
    h = Picture1.ScaleHeight
    max = 0
    For m = 0 To w - 1
        For n = 0 To h - 1

            color = Picture1.Point(m, n)

            r = getR(color)
            g = getG(color)
            b = getB(color)

            stat(Red, r) = stat(Red, r) + 1
            stat(Green, g) = stat(Green, g) + 1
            stat(Blue, b) = stat(Blue, b) + 1

            If stat(Red, r) > max Then max = stat(Red, r)
            If stat(Green, g) > max Then max = stat(Green, g)
            If stat(Blue, b) > max Then max = stat(Blue, b)
       Next
    Next

    For ic = 0 To 2
        Picture2(ic).Scale (-1, max)-(256, 0)
        For n = 0 To 255
            Picture2(ic).Line (n, 0)-(n, stat(ic, n))
        Next
    Next

    For ic = 0 To 2
        '平均を求める
        sum = 0
        For n = 0 To 255
            sum = sum + stat(ic, n) * n
        Next
```

```
            average = sum / h / w

            '分散を求める
            sum = 0
            For n = 0 To 255
                sum = (n ^ 2 * stat(ic, n)) + sum
            Next
            var = sum / w / h - average ^ 2

            '標準偏差を求める
            stdev = Sqr(var)

            '結果の出力
            Text1(ic).Text = Format(average, "##0.0")
            Text2(ic).Text = Format(stdev, "##0.0")

        Next
End Sub

Public Function getR(color As Long) 'Long の整数から R 成分を抜き出す関数を設定
        getR = color And &HFF&
End Function

Public Function getG(color As Long)    'Long の整数から G 成分を抜き出す関数を設定
        getG = (color And &HFF00&) / &H100&
End Function

Public Function getB(color As Long) 'Long の整数から B 成分を抜き出す関数
        getB = (color And &HFF0000) / &H10000
End Function

Private Sub Form_Load()
        Picture1.ScaleMode = vbPixels
        Picture1.AutoSize = True
        Picture1.AutoRedraw = True
End Sub
```

画像処理アルゴリズム(2)

１．空間フィルタによる画像処理

画像のある一部分のピクセル（３×３；３近傍）を下図の通り，座標(m, n)として扱う．$V_{x,y}$はピクセルの濃度値である．

(-1,-1) $V_{-1,-1}$	(0,-1) $V_{0,-1}$	(1,-1) $V_{1,-1}$
(-1, 0) $V_{-1,-0}$	(0,0) $V_{0,0}$	(1, 0) $V_{1,0}$
(-1,1) $V_{-1,1}$	(0,1) $V_{0,1}$	(1, 1) $V_{1,1}$

(a)処理前のピクセル

ここで中央のピクセル，つまり(0,0)のピクセルに対する演算処理を行うとする．

近傍処理との演算においては，フィルタを設定する．ここでフィルタ行列を次のようにおく．

(-1,-1) $f_{-1,-1}$	(0,-1) $f_{0,-1}$	(1,-1) $f_{1,-1}$
(-1, 0) $f_{-1,-0}$	(0,0) $f_{0,0}$	(1, 0) $f_{1,0}$
(-1,1) $f_{-1,1}$	(0,1) $f_{0,1}$	(1, 1) $f_{1,1}$

(b)フィルタ行列

(a)の各ピクセルの濃度値と空間フィルタを基にし，下式による演算を行い，(a)の中心(5)に対する処理値($V'_{0,0}$)を得る．

$$V'_{0,0} = V_{-1,-1}\cdot f_{-1,-1} + V_{-1,0}\cdot f_{-1,0} + V_{-1,1}\cdot f_{-1,1} + V_{0,-1}\cdot f_{0,-1} + V_{0,0}\cdot f_{0,0} + V_{0,1}\cdot f_{0,1} + V_{1,-1}\cdot f_{1,-1} + V_{1,0}\cdot f_{1,0} + V_{1,1}\cdot f_{1,1}$$

$$= \sum_{m=-1}^{1}\sum_{n=-1}^{1} V_{m,n} f_{m,n}$$

この処理を画像上の周辺部を除く全ての画素（周辺部は計算できない）に行い，処理後の画像を得る．（カラー画像の場合にはRGB各チャンネルについて行う．）

フィルタ値の設定，フィルタの大きさ（５近傍，７近傍など）により，様々な画像処理が可能となる．

（元の画像）

例１）シャープ化

0	-1	0
-1	5	-1
0	-1	0

元画像のピクセル成分が以下の通りであったとすれば,

12	17	18
11	22	33
9	25	35

その中央のピクセルの新しい成分は,

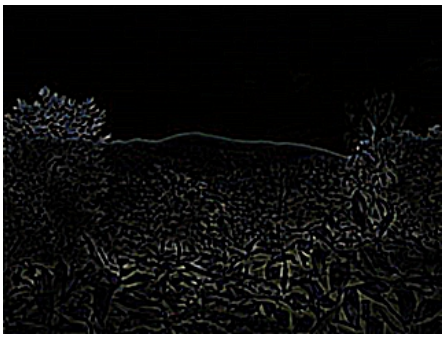
0×12＋(-1)×17＋0×18＋(-1)×11＋5×22＋(-1)×33＋0×9＋(-1)＋25＋0×35＝24 となる.

例２）ぼかし（デフォーカス）の例

0.08	0.12	0.08
0.12	0.2	0.12
0.08	0.12	0.08

例３）輪郭（エッジ）抽出

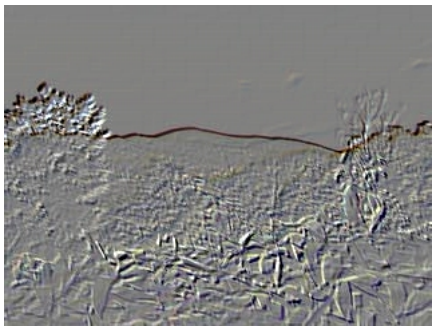
1	1	1
1	-8	1
1	1	1



例４）エンボス

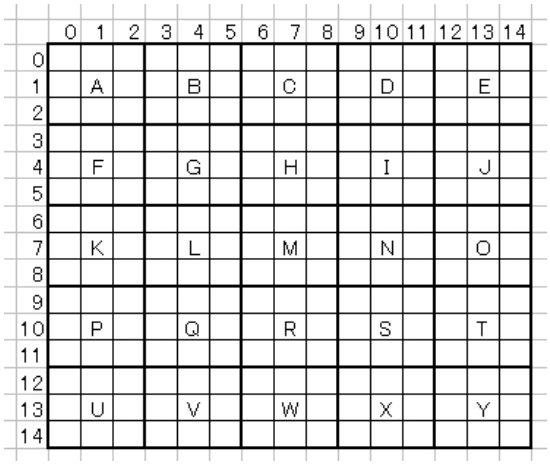
-1	0	0
0	0	0
0	0	1

さらに計算値に 128（グレー）を加える.



２．モザイク処理

画素の各ブロック（下図の例の場合，A，B，C・・の３×３ピクセルのブロック）毎に色の平均化を行う.



３．画像ファイルの形式

GIF (Graphic Interchange Format)

Compuserve で開発されたファイル形式である．(Unisys に特許料を払う必要がある)
インターレース描画，背景の透明化が可能であり，ページタイトル，ボタンなどに使われる． ただし 256 色まで.

JPEG Joint Photographic Experts Group)

人間の目は高い周波数成分に対してほど鈍感であり，また輝度に関する解像度に敏感で，色度に関する解像度については鈍感である．これらの特性を利用し，人間が劣化を感じることが少ないような方法で圧縮を行う．
1677 万色カラー画像が扱えることから，ギャラリーなど大きな画像に利用される．

PNG (Portable Network Graphics)

JPEG や GIF に変わり，web 上での使用を目指して開発され，W3C により推奨されている．
ライセンス料不要の圧縮方式を採用し，フルカラーの自然画を劣化無しで圧縮可能．
１ピクセルあたり 48bit（RGB それぞれ 16 ビットずつ）までの情報を持たせることができる．

PICT：Apple 社 Macintosh シリーズでの画像ファイル形式

BMP：Windows での標準画像ファイル形式

EPS,TIFF:

版下作成の場合，画像のカラーモードは CMYK で，ファイル形式は EPS(Encaplsulated PostScript)もしくは TIFF(Tagged Image File Format)である必要がある．EPS はベクタデータの保存も可能である．

注)PostScript　Adobe 社の開発したページ記述言語

４．プログラムの例（モザイク）

モザイクを計算するためのサブルーチンと RGB 値抽出関数のソース

```
Private Sub ComMozaic_Click() 'コマンドボタン ComMozaic を配置する.
    Dim Xgrid  As Integer, Ygrid  As Integer, X  As Integer, Y  As Integer, m  As Integer, n As Integer, Size As Integer
    Dim sumR  As Long, sumG As Long, sumB As Long
    Dim mozaicR As Integer, mozaicG As Integer, mozaicB As Integer

    Size = 10  'モザイクを行うピクセルサイズ

    'モザイクの格子数計算
    Xgrid = Picture1.ScaleWidth ¥ Size '¥は整数除算
    Ygrid = Picture1.ScaleHeight ¥ Size

    For X = 0 To Xgrid - 1 '半端になるピクセルは除外
        For Y = 0 To Ygrid - 1

            sumR = 0: sumG = 0: sumB = 0
```

```
                For m = X * Size To X * Size + Size - 1
                    For n = Y * Size To Y * Size + Size - 1
                        sumR = getR(Picture1.Point(m, n)) + sumR
                        sumG = getG(Picture1.Point(m, n)) + sumG
                        sumB = getB(Picture1.Point(m, n)) + sumB
                    Next
                Next
                mozaicR = sumR / Size / Size
                mozaicG = sumG / Size / Size
                mozaicB = sumB / Size / Size

                For m = X * Size To X * Size + Size - 1
                    For n = Y * Size To Y * Size + Size - 1
                        Picture1.PSet (m, n), RGB(mozaicR, mozaicG, mozaicB)
                    Next
                Next
            Next
        Next
End Sub

Public Function getR(color As Long)    As Long
    getR = color And &HFF
End Function

Public Function getG(color As Long) As Long
        getG = (color And &HFF00&) / &H100&
End Function

Public Function getB(color As Long) As Long
    getB = (color And &HFF0000) / &H10000
End Function
```

【課題】
任意の画像を読み込み，モザイク，エンボス，ぼかしの処理を行うプログラムを作成しなさい．（その他の任意の処理を加えても構わない，）

提出期限：５月１９日（木）　１２：５０まで
提出先：makarepo@myu.ac.jp
（..vbp と.frm のファイルを提出する．送信エラーが生じる場合は ZIP あるいは LZH 形式で圧縮してください．）

５．プログラムの例（フィルタ）

```
‘Picture1, Picture2 とも Scalemode はピクセル(vbPixels)

Private Sub ComFilter_Click()
    Dim W As Integer
    Dim H As Integer
    Dim flt(-1 To 1, -1 To 1)
    W = Picture1.ScaleWidth
    H = Picture1.ScaleHeight
    flt(-1, -1) = -1
    flt(0, -1) = 0
    flt(1, -1) = 0
    flt(-1, 0) = 0
    flt(0, 0) = 0
    flt(1, 0) = 0
    flt(-1, 1) = 0
    flt(0, 1) = 0
    flt(1, 1) = 1
    offset = 128
    For X = 1 To W - 1
        For Y = 1 To H - 1
            r = offset: g = offset: b = offset
            For m = -1 To 1 Step 1
                For n = -1 To 1 Step 1
                    If flt(m, n) <> 0 Then
                        r = flt(m, n) * getR(Picture1.Point(X + m, Y + n)) + r
                        g = flt(m, n) * getG(Picture1.Point(X + m, Y + n)) + g
                        b = flt(m, n) * getB(Picture1.Point(X + m, Y + n)) + b
                    End If
                Next
            Next
            If r < 0 Then r = 0
            If g < 0 Then g = 0
            If b < 0 Then b = 0
            Picture2.PSet (X, Y), RGB(r, g, b)
        Next
    Next
End Sub
```

画像処理アルゴリズム(3)

１．画像合成

ピクチャコントロールを用いて，PicOrg1, PicOrg2, PicOut の３つの PictureBox オブジェクトを作成する．なお，オブジェクト名はプロパティウィンドウの(オブジェクト名）を変更する．

1)PicOrg1 と PicOrg2 の画像を読み込む．

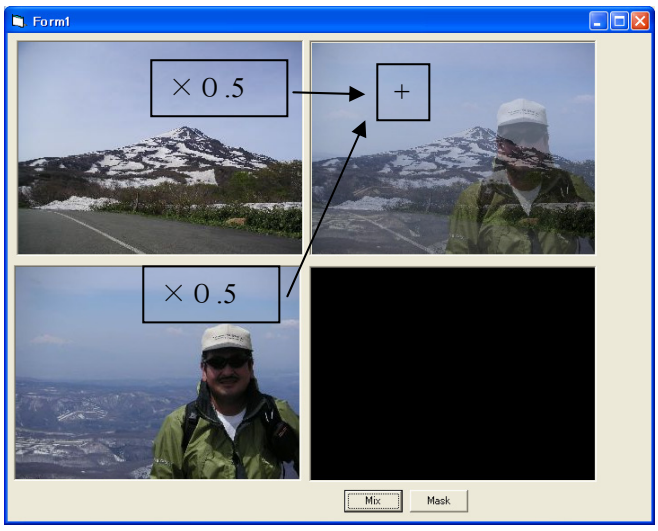
2)R,G,B それぞれについて，設定した比率で 2 枚の画像のピクセルの値を合成し，それを（それぞれの比率の和が 1 になるようにする．）

例えば，

Ｒの合成値＝PicOrg1 のＲの値*.5+PicOrg2 の R の値

とする．

RGB それぞれについて計算した後に，その値を PicOut に Pset すればよい．



２．マスクによる画像合成

マスクという白黒（あるいはグレースケール）の画像を作成し，それをもとに合成処理を行う．

（白の部分で処理，黒の部分は何もしない）

1)さらにもう１枚の PictureBox オブジェクトを追加する（PicMask）というオブジェクトにする．

2)マスク画像の初期色（BackColor）を黒に設定し，マウスをドラッグすることにより，白で描画できるように設定する．（MouseMove イベントを利用する）

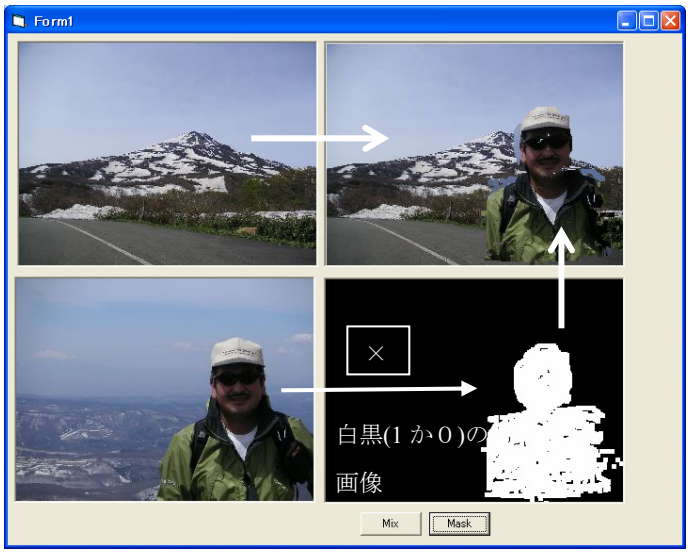
3)マスクの演算は以下の式により行う．

なお，マスクは白黒なので R,G,B の値は等しい．

そこで R 値のみ抽出し，255 で１になるように設定する．

画像の計算値＝(マスクの R 値/255)*PicOrg2＋（1-マスクの R 値/255)*PicOrg1

これを RGB それぞれの成分について行い，その計算値を PicOut に出力する．



２次元ベクタグラフィックス

１．画像処理からベクタグラフィックスへ　（点から線の描画へ）

コンピュータグラフィックスの最も基本となる技術は，画面上を構成するひとつひとつのピクセルに対して，発色するか否か，また発色する場合には RGB をどのような割合で発色させるか．．ということを命令 ・実行するものである．しかし，コンピュータ上により容易に絵を描画しようという場合に，1つ1つのピクセルに対する操作を用いることは極めて効率が悪い．

紙上に鉛筆で絵を描く場合を考えてみれば，それは点ではなく，線により構成される．線はいくら細分化しても線であり，それは必ずしも点の集合ではない．しかし，コンピュータ上での表現はディスプレイに制約される点，すなわちピクセルの集合である．コンピュータ上で線を描くためには，ベクトルとして表現される線を，ディスプレイ上の画素の点情報に変換するという必要がある．　そこで，まずディスプレイ上で線をいかに表現するかということから学ぶ．

２．図形描画のしくみ

（1）ピクセル単位での線の描画

①水平線を描画する．

コンピュータディスプレイ上で一本の水平線を描画することを考える場合，ディスプレイ上において水平線に対応するピクセルに対して発色するか否かの指示を与えれば，水平線を描画することができる．

準備

1. ピクチャコントロールボックスを Form1 上に貼り付ける
2. 貼り付けたピクチャコントロールを選択し，**ScaleMode プロパティを 3-ピクセルに変更**する．
3. ピクチャコントロールの ScaleWidth と ScaleHeight がそれぞれ 300 以上であることを確認しておく．（足りない場合には，ピクチャコントロールのサイズを大きくすること）

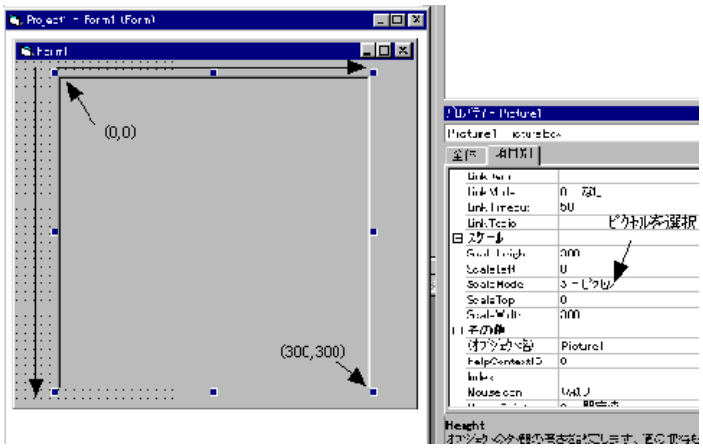


図-1　Picture1 の座標系

●座標(100,100)から(200,100)の直線を描く.

【コード1】

```
Option Explicit
Private Sub Picture1_Click()  'ピクチャボックスをクリックすると描画される.
   Picture1.Pset(100,100), RGB(255,0,0)     '赤の線を描く (&HFF)
    Picture1.Pset(101,100), RGB(255,0,0)
    Picture1.Pset(102,100), RGB(255,0,0)
    Picture1.Pset(103,100), RGB(255,0,0)
                        :
    Picture1.Pset(199,100), RGB(255,0,0)
    Picture1.Pset(200,100), RGB(255,0,0)
End Sub
```

これに For..Next ループを用いて表現したのがコード2である.

【コード2】

```
Option Explicit
Private Sub Picture1_Click()  'ピクチャボックスをクリックすると描画される.
    Dim X  As Integer
    For X = 100 To 200
       Picture1.PSet (X, 100), RGB(255, 0, 0)  '赤で点を描く
    Next
End Sub
```

②斜めの線の描画

斜めの線を描く場合には，その直線の方程式を求め，その式に基づいて発色させるピクセルを求める必要がある.

●座標(100,100)から(200,300)を結ぶ直線を描く

直線の方程式は $y = 2x - 100$ であり，この式を用いて点列を描画する.

【コード３】

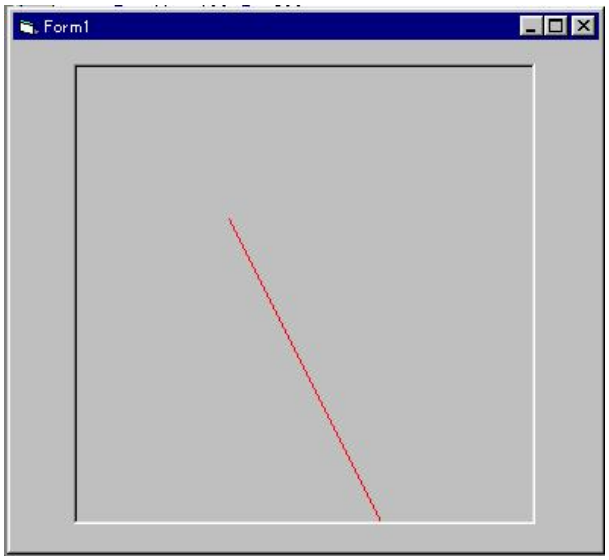
```
Option Explicit
Private Sub Picture1_Click()  'ピクチャボックスをクリックすると描画される.
    Dim X As Integer
    For X = 100 To 200
       Picture1.PSet (X, X*2-100), RGB(255, 0, 0)  '赤で点を描く
    Next
End Sub
```

このプログラムを実行して描かれるのは点線である．Ｘに比べＹの増分が大きいことから，

直線として描画されない．そこで増分の多いほうから少ないピクセル値を計算して描画することが必要である．

描画のための計算式：　x ＝ 1/2 * x + 50

```
Option Explicit
 Private Sub Picture1_Click()  'ピクチャボックスをクリックすると描画される．
    Dim X As Integer, Y As Integer
    For Y = 100 To 300
      Picture1.PSet (Y/2+50, Y), RGB(255, 0, 0)   '赤で点を描く
    Next
 End Sub
```



(2)グラフィックス命令による描画

（１）のように点（ピクセル単位）で描画することにより，直線や曲線の描画が可能であるが，実際のグラフィックスプログラミングを行う上では極めて効率が悪い．そのため，多くのプログラミング言語では，グラフィックスのための命令が組み込まれている．

①直線の描画

VB では直線は以下の命令により描画することができる[ ]省略可能

直線の描画命令

```
object.Line   [Step](x1, y1)- [Step](x2, y2), [color], [B][F]
```

(x1,x1) は始点の座標，(x2,y2)は終点の座標
color は色（省略時は ForeColor プロパティの色で塗りつぶされる）
[B]はそれを対角線とする四角形を描画する．
[F]は塗りつぶし．
[Step]は直前の座標に対する相対座標により指示する場合

例えば

```
Picture1.Line (10,10)-(100,100)
Picture1.Line -Step (100,100)
Picture1.Line (10,10)-(100,100), RGB(255,255,0), B
Picture1.Line (10,10)-(100,100), RGB(255,255,0), BF
```

演習

1. Picture1 のピクセル座標上(20,50)と(250, 300)を結ぶ直線を描きなさい．
2. 1. で描いた直線をを対角線とする四角形を描画しなさい．またその四角形を緑色で塗りつぶしなさい．
3. ピクセル座標上の点 A(100,100)から相対的に(20,50)移動した点 B を，さらに点 B から相対的に(−40, 20)移動した点を点 C としたとき，点 A,B,C を Line メソッドの STEP オプションを利用して描きなさい．

３．線から曲線を描く（円の描画）

ここでは円の描画を例に，関数既知の曲線の描画について説明する．
原点(0,0）を中心とする半径 r の円の方程式は

$$x^2+y^2 = r^2$$

で与えられる． この式を用いて，グラフィックス表示を行うことを考える．
y を x を変数とする方程式に置きかえれば，

$$y = \pm sqr(r^2 - x^2)$$

注）sqr は平方根を求める関数(VB)
これを第一象限の 1/4 円のみについて考えれば，y は常に正の値をとるから，

$$y = sqr(r^2 - x^2)$$

半径は自ら設定すべき値であるから，仮に半径を 100 として考えれば，

$$y = sqr(100^2 - x^2)$$

となる．

上の式に対し，x の値を 0 から 100（つまり半径）の範囲で変えることにより，円の描画が可能となる．

【コード４】 あらかじめ Picture コントロールを配置しておくこと

```
Option Explicit
 Const PAI = 3.141592   '円周率 PAI を定義しておく．次のプログラムで利用します.
 Private Sub Form_Load()
   Picture1.ScaleMode = 3
End Sub
 'Command1 を配置する.
Private Sub Command1_Click()
   Dim x As Single, y As Single, r As Single
   Picture1.Cls 'picture1 をクリアする
   r = 100
   For x = 0 To 100
      y = Sqr(r ^ 2 - x ^ 2)    'y の値を計算
      If x = 0 Then
         Picture1.PSet (x, y)    '最初の 1 点を点として打ち込む
      Else
         Picture1.Line -(x, y)    '直前の描画点との間で直線を引く
      End If
      Picture1.Circle (x, y), 2, &HFF    '実際のプロット点を表示
   Next
End Sub
```

コード４を実行すると半円が表示される．しかし，実際の円を構成する点の配置は，円周上で均一ではなく，きれいな円を描くことができていない．

このような問題に対して，媒介変数（パラメータ）を基にし，それぞれの値を別々に計算するパラメトリック曲線の考え方がある．円の描画については，角度をパラメータとして扱うことになり，下式により表現できる．

$$x = r * \cos(t)$$
$$y = r * \sin(t)$$

【コード５】

```
'Command2 を配置する.
Private Sub Command2_Click()
```

```
    Dim t As Single, x As Single, y As Single, r As Single
    Picture1.Cls  'picture1 をクリアする
    r = 100
    For t = 0 To 90 Step 15
        x = Cos(t * PAI / 180) * r     't*PAI/180 で度をラジアンに変換
        y = Sin(t * PAI / 180) * r
        If t = 0 Then
            Picture1.PSet (x, y) '最初の 1 点を点として打ち込む
        Else
            Picture1.Line -(x, y) '直前の描画点との間で直線を引く
        End If
        Picture1.Circle (x, y), 2, &HFF '実際のプロット点を表示
    Next
End Sub
```

コード5のプログラムを実行すれば明らかであるが，より少ないプロット点で円を描画することが可能である． ここで $x^2+y^2 = r^2$ のように，求めるべき x, y が相互依存した式の表現をノンパラメトリック表現といい，一方，パラメータを介して x, y の値を個別に計算する式の表現（例えば x = r * cos(t), y = r * sin(t) ）を パラメトリック表現という．

上述したようにパラメトリック表現はコンピュータ上での曲線の描画を行なう上で非常に有効な手法であり，**また非常に重要な考え方である．**

【補足】円を描画する命令（VB）

object.Circle [Step] (x, y), radius, [color, start, end, aspect]

[Step]　省略可：相対座標で中心座標を指示

radius　半径

color　省略可：色

start, end　省略可：それぞれ円の開始角度，終了角度（半円などを描画する場合）
ラジアン（360 度はラジアン表記で 2π，つまり 2*3.1415...）

aspect　省略可：楕円を書く場合に指定

例えば，

```
Picture1.Circle (100,128), 50
Picture1.Circle Step (10, 10), 50
Picture1.Circle (100,128), 50, RGB(0,255,255)
Picture1.Circle (100,128), 50, ,0,1.57
Picture1.Circle (100,128), 50, , , ,2
```

４．ベクタグラフィックスのプログラミングの実際

マウス操作に応じて線を描く簡単なグラフィックスソフトウェアの作成を行う．

1)クリックした点を結ぶプログラム

直前の描画点とマウスでクリックした座標とを結ぶ

```
'まず，PictureBox オブジェクトを Form1 上に貼り付けておく．
Option Explicit
Private Sub Form_Load()
      Picture1.ScaleMode = vbPixels   'ピクセルモードを指定しておく
End Sub
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
   Picture1.Line -(X, Y), &H0     'マウスを押した場所の X,Y 座標とその直前の
                                  ' 描画点とを結ぶ直線を描く（&H0=黒）
End Sub
```

2)始点と終点を設定する．

カウンタ変数を設定し，描画点の数を記録する．

・カウンタ変数が０であれば点のみを描画する．

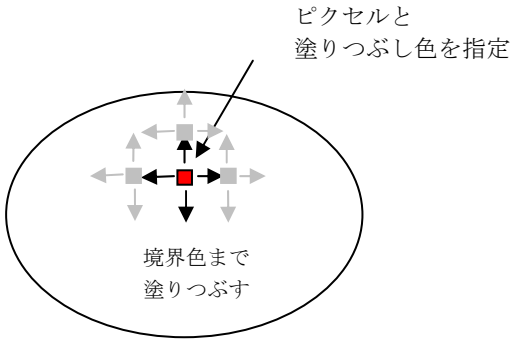
・ダブルクリックした場合には，カウンタ変数を０に戻す．

```
Option Explicit
Dim pcount As Integer 'カウンタ，複数の Sub ルーチンで使うので広域的に宣言しておく
Private Sub Form_Load()
     Picture1.ScaleMode = vbPixels   'ピクセルモードを指定
     pcount = 0   'カウンタを初期化
End Sub
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
          If pcount = 0 Then
               Picture1.PSet (X, Y), &H0        'とりあえず黒で描画（&H0=RGB(0,0,0)
          Else
                Picture1.Line -(X, Y) , &H0
          End If
          pcount = pcount + 1   'カウンタに 1 加える
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
        End      'フォームが閉じられたら終わり.
End Sub

'ダブルクリックイベントの追加
Private Sub Picture1_DblClick()
    'ダブルクリックされたら..
    'まず MouseDown イベントを感知して MouseDown ルーチンを実行します.
    'その後にこのルーチンが呼び出されます.
    pcount = 0 'ダブルクリックされた場合はカウンタを 0 に
End Sub
```

3)図形の塗りつぶし

ディスプレイ上に描画されている図形内の１つのピクセルを指定し，境界線（ピクセル値が異なる範囲）までのピクセルを指定した塗りつぶし色に変更する．完全に図形が閉じていない場合には，図形外に色が溢れる．
（この機能は Windows では API として標準で組み込まれている.）



①FloodFill の呼び出し

プログラムの１行目

```
Private Declare Sub FloodFill Lib "GDI32" (ByVal hDC As Long, ByVal X As Long, ByVal Y As
  Long, ByVal crColor As Long)
```

注)改行を入れずに一行で打ち込むこと.

この 1 行で Windows の API(アプリケーションプログラミングインターフェース)を呼び出す.

②Form1 の MouseDown イベントに，塗りつぶしの色の設定及びコマンドを加える．

```
'Form モジュールに追加するコード（太字）
 Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y _
 As Single)
    If Button = 1 Then     'If Button = vbLeftButton Then でもよい
       If pcount = 0 Then
          Picture1.PSet (X, Y), &H0
       Else
          Picture1.Line -(X, Y), &H0
       End If
       pcount = pcount + 1   'カウンタに 1 加える
    ElseIf Button = 2 Then     'Button = vbRightButton でもよい
       Picture1.FillStyle = 0            ' FillStyle プロパティを塗りつぶし(0)に設定
       Picture1.FillColor = &HFF   ' FillColor プロパティ(&HFF=RGB(255,0,0)を
                                          '設定します。
      FloodFill Picture1.hDC, X, Y, &H0
                                 'X,Y から境界線の色（&H0 黒）までの範囲を塗りつぶす
    End If
 End Sub
```

4)塗りつぶし色の設定

CommonDialog コントロールによる色の設定

色の設定を行うダイアログボックスを表示するコマンド　CommonDialog1.ShowColor

```
Private Sub Command1_Click()
    CommonDialog1.ShowColor
    Picture1.FillColor = CommonDialog1.Color
End Sub
```

5)AutoRedraw の設定

4)により色のダイアログボックスを表示した場合，その背景にある画像情報が失われるという問題がある．これは，これまで VB でプログラムして描かれた画像が一時的なグラフィックス用のメモリ（現在，表示されている画面のみを記憶している；VRAM）に対する操作のみを行なっており，それ以前に表示されていた画像を一切記憶していないということを意味する．これを回避するために，Picture1 の AutoRedraw というプロパティを操作し，グラフィックスをメモリ上に記憶できるようにする．

①Picture1 の AutoRedraw プロパティを True にする．

②AutoRedraw を設定した場合，塗りつぶし命令が直ちに行われないので，強制的な描画更新を行うコマンド(Refresh)を追加する．

```
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If Button = 1 Then     'If Button = vbLeftButton Then でもよい
        If pcount = 0 Then
            Picture1.PSet (X, Y), &H0
        Else
            Picture1.Line -(X, Y) , &H0
        End If
        pcount = pcount + 1   'カウンタに 1 加える
    ElseIf Button = 2 Then     'Button = vbRightButton でもよい
        Picture1.FillStyle = 0
        FloodFill Picture1.hDC, X, Y, &H0
                              'X,Y から境界線の色(&H0  黒)までの範囲を塗りつぶす
        Picture1.Refresh
  End If
End Sub
```

【参考】

Shape コントロールを貼り付け，以下のようなコードを Command1_Click 及び Form_Load プロシージャに記載しておけば，設定色が確認できるようになる．

```
Shape1.FillStyle = 0
Shape1.FillColor = Picture1.FillColor
```

演習

１．直線の描画色を変えることができるようにプログラムを変更しなさい．

２．これまで直線を描くプログラムを作成してきましたが，これに円の描画を加え，円と直線とを切り替えて描画できるようなプログラムを考え，作成しなさい．

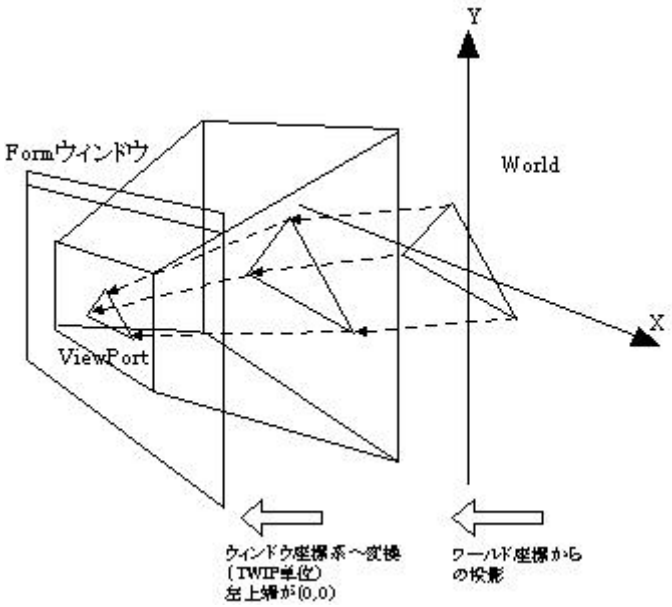
３．さらにボックス(四角形)を描くことができるようなプログラムを作成しなさい．

ウィンドウ座標の変換

１．ウィンドウ座標の変換

これまで画面上の座標はピクセル単位で扱ってきた. しかし,科学技術計算等を図示する場合には, 一般に用いられているデカルト座標系に置き換えることができれば, より容易に描画が可能となる. そこで, VB のピクチャボックス内での座標系を再設定する方法を学ぶ.

(1)ワールド座標系とウィンドウ座標系



（2）VB での座標系の設定

VB のデフォルトの座標系：

スケール：Twip (1 論理インチ 1440 Twip）

原点(0,0)は左上端，下方向＋，右方向＋

一般のデカルト座標系を用いる場合には座標変換が必要であるが, VB の PictureBox では, 任意の座標系を定義できる.

【原点（0,0)を中心し，左上隅が(-1,1)，右下隅が（1,-1）となる座標系を定義する】

①プロパティの変更による方法

1)ピクチャコントロールボックスを Form1 上に貼り付ける.

2)ピクチャコントロールボックスの大きさを正方形になるように設定する.

（プロパティ　Picture1.Width と Picture1.Height が同じ値になる）

3)以下のように Picture1 のプロパティを修正する.（コード内からの変更も可能である）

```
Picture1.ScaleMode = 0
Picture1.ScaleHeight = -2
Picture1.ScaleWidth = 2
```

Picture1.ScaleTop = 1

Picture1.ScaleLeft = -1

これにより原点を中心とする座標系が定義できる.

②メソッドを用いた方法

Scale メソッド　Picture1.Scale (左上隅の座標)-(右下隅の座標)

1)ピクチャコントロールボックスを Form1 上に貼り付ける.

2)ピクチャコントロールボックスの大きさを正方形になるように設定する.

3)以下の命令をプログラム内で呼び出す.

Picture1.Scale (-1,1)-(1,-1)

これにより原点を中心とする座標系が定義される.

(3)円の描画

例えば, この座標系で円を描画してみよう.

（以前にピクセルモードで書いた円のプログラムを改変）

```
Option Explicit
Const PAI = 3.1415
Private Sub Form_Load()
    Picture1.Width = 3000
    Picture1.Height = 3000
     Picture1.Scale (-1, 1)-(1,-1)
      ’または. .
    ‘Picture1.ScaleMode = 0
    ‘Picture1.ScaleHeight = -2
    ‘Picture1.ScaleWidth = 2
    ‘Picture1.ScaleTop = 1
    ‘Picture1.ScaleLeft = -1
End Sub
Private Sub Command1_Click()
    Dim n As Integer
    Dim x As Single, y As Single
   For n = 0 To 360
        x = Cos(n / 180 * PAI)
        y = Sin(n / 180 * PAI)
        If n = 0 Then
```

```
        Picture1.PSet (x, y)
      Else
        Picture1.Line -(x, y)
      End If
    Next
End Sub
```

(4)多角形の描画

上述のプログラムの

```
For n = 0 To 360
```

の部分に 例えば,

```
For n = 0 TO 360 Step 60
```

のように, Step として設定する数値を 90, 120 等と変えれば, 任意の正多角形の描画が可能となる.

VisualBasic による２次元アニメーション

１．タイマーコントロール

ＶＢではタイマーコントロールはイベントを一定の間隔で生じさせるコントロールであり，アニメーションの速度を制御することができる.

例えば，時計を表示したいという場合に，

```
Do
    Text1.Text = Time        'Time は現在の時間を返す関数である.
Loop
```

という処理によりプログラムは常に動きつづけるが，秒単位で表示する時計の場合には，1 秒毎に時刻を読取り表示するだけで良い．このような処理を行うために，タイマーコントロールを用いる.

【１秒毎に時刻を表示するプログラム】

```
‘タイマーコントロールとラベルコントロール（Ａと表示されたコントロール）を Form1
’に貼り付ける.
Option Explicit
Private Sub Form_Load()
    Label1.Caption = ""
    Timer1.Interval = 1000
    Timer1.Enabled = True
End Sub
Private Sub Timer1_Timer()
    Label1.Caption = Time
End Sub
```

３．図形の移動アニメーション

タイマーコントロール（Timer1）により，設定した時間に応じてイベントを発生させる.

【例】タイマーコントロールを選択し，Picture1 上に貼り付ける.

```
Dim n As Integer        'n を広域宣言しておく.
Private Sub Command1_Click()
   Timer1.Interval = 100
          ‘イベントの発生する単位をミリ秒で設定（ただしその精度は 1/18 秒程度）
   Timer1.Enabled = True
       ‘True あるいは False を設定する．(初期は False になっている）
End Sub
Private Sub Timer1_Timer()
   Picture1.Cls       '画像のクリア
```

```
    Picture1.Line(n,20)-Step(40,40),&HFF,BF
    n = n + 1    'n に 1 を足しておく（呼び出される毎に 1 増える）
End Sub
```

２．アニメーション速度とダブルバッファリング

(1)アニメーションの速度

n = n + 1 を　n = n + 5 というように n の増分を大きくすること, あるいは Timer1.Interval の値を変えてやれば，アニメーションの速度を変えることが可能である.

(2)ダブルバッファリング

画像のちらつきは，描画とクリアとを繰り返すことにより，生じるものである.

そこで，コンピュータ上に表裏の 2 枚の画像メモリを準備しておき，裏の画面で描画作業を行ない，描画が終ったら，表裏の画像を入れ替えるという方法が用いられる．これをダブルバッファリング(double buffering)という.

VB では同様の機能が以下の設定により実現できる.

```
    Piture1.AutoRedraw = False     'Picture1 を設定する場合
```

ただし，これにより描画速度は低下する.

3. 物理的現象のシミュレーション

(1)シミュレーション（Simulation) :

コンピュータシミュレーションとは，コンピュータを用いて, 複雑な自然現象や経済現象などについてモデルを作り，予測結果をはじきだす手法である. .

物理に基づいて，物体の落下や移動をシミュレートする２次元アニメーションを作成するためには，単位時間あたりにおける物体の移動方向と移動量を数値的に得て，それをシミュレートする必要がある．そのためには，若干の数学的知識や物理に関する知識が必要とならざるを得ない.

例えば，投げたボールについてアニメーションしてみよう．（ただし空気抵抗等は考慮していない．実際にはより複雑なシステムとなる.）

時間 t に対するボールの変位 y は，落下運動の式をもとにすれば，以下の式により表現される.

$y = v_{y0}t - gt^2/2$　　v_{y0}:y 方向への初速度, g:重力加速度(9.8m/s^2)

右辺第2項が負となるのは，ボールを上に投げることにより, 負の重力加速度が働くからである.

一方, x 方向への変位は,

$x = v_{x0}t$　　v_{x0}:x 方向への初速度

さらに投げたボールが再び地面に達したときには，そこで衝突による力の減衰が働くから，ここで初速度を再設定するとともに，再び t の値を初期化して計算を行う必要がある．

これらを実際にプログラミングすると以下のようになる．

```
Option Explicit
Dim t As Single , x0 As Single , y0 As Single , vx0 As Single , vy0 As Single , reduce As Single
Const g = 9.8  '重力加速度の設定
Private Sub Form_Load()
    Timer1.Enabled = False
    Timer1.Interval = 100
    With Picture1
       .AutoRedraw = True
        .BackColor = &HFFFFFF
        .ScaleMode = 0
        .Width = 6000
        .Height = 3000
        .ScaleWidth = 200
        .ScaleHeight = -100
        .ScaleTop = 98
        .ScaleLeft = -50
    End With
    x0 = -50  'ホールの初期位置(X 方向)
    y0 = 0  'ボールの初期位置(y 方向)
    reduce = 0.7  '衝突時の速度減衰率
End Sub
Private Sub Command1_Click()
    If Timer1.Enabled = True Then
        Timer1.Enabled = False
    Else
        t = 0       '時間の初期化
        vx0 = 10    'x 方向初速度(m/s)
        vy0 = 40    'y 方向初速度(m/s)
        Timer1.Enabled = True
    End If
End Sub
Private Sub Timer1_Timer()
    Dim x As Single, y As Single
```

```
    x = vx0 * t + x0        'x 方向の位置計算
    y = vy0 * t - g * t ^ 2 / 2    'y 方向の位置計算
    Picture1.Cls
    Picture1.FillStyle = 0
    Picture1.FillColor = &HFF
    Picture1.Circle (x, y), 2, &HFF
    If t > Timer1.Interval / 1000 And y <= 0 Then    '地面に達したら
        t = 0                    '時間を初期化
        x0 = x                   '現在の位置を初期位置に再設定
        y0 = y
        vy0 = vy0 * reduce           '速度の減衰を行なう.
        vx0 = vx0 * reduce
    End If
    t = t + Timer1.Interval/1000     '次の計算時間 t を求める.
End Sub
```

レポート課題

(1)これまでに学習した機能を組み合わせ, VB 上で稼動するアナログ時計のプログラムを作成しなさい.

時針, 分針, 秒針を有し, かつ現在の時刻を表示することができるものであることが最低条件.

機能, デザインは自由に設定すること.

(2)先週までの授業で学習した機能を用いたドローイングソフト(任意の図形の描画, 色の塗りつぶしが可能なソフトウェア)を作成しなさい.

提出期限は 6 月 23 日(木)まで 12:50 まで, makarepo@myu.ac.jp に送付すること.

(.vbp, .frm(必要があれば画像ファイル)を含めて添付ファイルとして送付する.)

他人の作品や書籍等からのコピーは絶対不可. 自分で考えて作成すること.

(参考)

線の太さを変える	事前に Picture ボックスのプロパティ .DrawWidth を変更しておく.	Picture1.DrawWidth = 2 Picture1.Line (0,0)-(100,100)
線の色を変える	Line メソッドのオプションで変更するか, 事前に Picture ボックスのプロパティ ForeColor を変更しておく.	Picture1.Line -(100,199),&HFF または Picture1.ForeColor = RGB(0,255,0) Picture1.Line (0,0)-(100,100)
ダブルバッファリング	Picture1.AutoRedraw プロパティを True に設定しておく.	Picture1.AutoRedraw = True
Picture1 の背景色を変える.	Picture1.BackColor プロパティを設定する.	Picture1.BackColor = RGB(255,0,0)
画面のクリア	.Cls メソッドを用いる.	Picture1.Cls

基本図形の描画	シェープ(Shape)コントロールを用いる手段もある．形状，大きさは Shape のプロパティで設定する．	
背景への画像ファイルの読み込み	Picture プロパティで設定	Set Picture1.Picture = LoadPicture(″画像ファイル名″)
警告音を鳴らす	Beep メソッド （設定音は Windows で設定した警告音）	Beep
日付	Date メソッド	Date 年月日の抽出は， Year(Date), Month(Date),Day(Date)など．． 引数は Now でも可
複数のオブジェクトの重ね合わせ	オブジェクトを選択し，右クリックで，どのオブジェクトを最上部，あるいは最下部に表示するかを設定できる．	

(参考 1）アラーム機能

例えばテキストボックスを追加し，以下のようなコードを追加すれば，アラーム機能が追加可能

例１）コード内で時刻を指定する場合

```
Private Sub Timer1_Timer()
    Label1.Caption = Time
    If Time > TimeValue("09:40:00") Then Beep
End Sub
```

例２） テキストボックスの値を用いる場合

```
Private Sub Timer1_Timer()
    Label1.Caption = Time
    If Time > TimeValue(Text1.Text) Then Beep
End Sub
```

（参考 2）TIme 関数からの時間 H，分 M，秒 S の抽出

VB の標準関数を用いれば，HMS の抽出は容易である．

```
H = Hour(Now)
M = Minute(Now)
S = Second(Now)
```

ＣＡＤとコンピュータグラフィックス

１．CAD とは

CAD(Computer Aided Design)：コンピュータによる設計支援

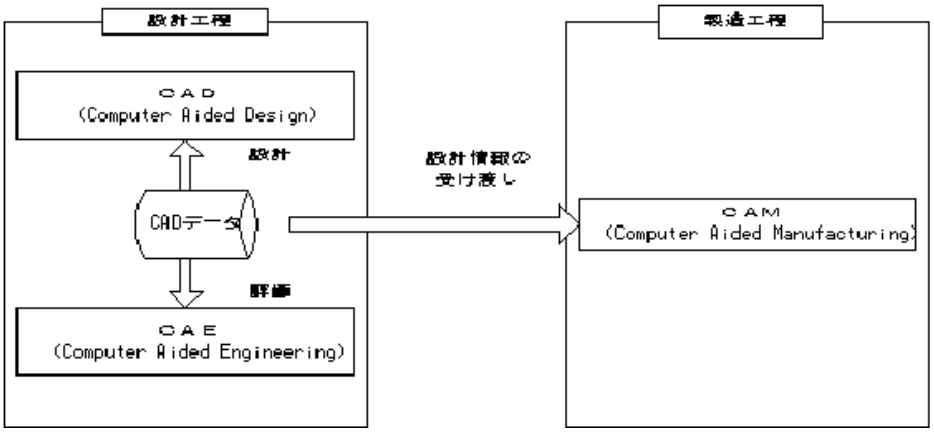
関連する技術

CAM（Computer Aided Manufacturing)

コンピュータによる製造支援技術.

CAE(Computer Aided Engineering)

製品開発時に行う数値解析，シミュレーション技術を指す.



＊広義の CAD システムは CAE も含む.

２．２次元 CAD と３次元 CAD

1)２次元 CAD

設計情報を２次元情報として有する.

Computer Aided Drafting とも言われ，実際には単なる製図ツールである）

2)３次元 CAD

設計情報を３次元情報として有する.

CAE や CG，CAM へのデータ転用が容易となり，設計・製造工程の効率化が実現される.

（しかし，現状ではその適用には問題がないわけではない.）

現在は２次元 CAD から３次元ＣＡＤへの移行期にあたる.

（大手の製造業では３次元 CAD の導入が進んでいる.）

主な CAD ソフト

- AutoCAD
- MicroStation
- SolidWorks
- IDEAS
- CADAM
- GDS

・VectorWorks (旧 MiniCAD)
・JW-CAD（フリーウェア）　　など多くの CAD ソフトがある．

特に AutoCAD，MicroStation 等の CAD はマクロプログラムを作成することにより，
自分の目的とする業務に特化するアプリケーションを作成することができる．
アドオンソフトとしてこれらのソフトウェアが販売されている．
これらのソフトウェアはいわば CAD の OS であるということができる．

３．CAD の基本的な仕組み

ここでは２次元 CAD を例として，そのシステム構造について VB によるプログラミングを通して学ぶ．

CAD に求められる基本機能は，以下の通りである．

・図形の描画機能
・図形の編集機能
・図形の出力機能（ファイル，プリンタ，プロッタ等）

1)描画機能の実現

CAD 上で描かれる描画スケールは，基本的には現実のスケールが与えられる．
（ドローソフトウェアとの大きな差である．）
ex. 10m×10m の図形を描画する場合に，そのデータも 10m×10m の構造を有する．
そのため，ビューポート（描画ウィンドウ）のスケールを，設計物にあわせて変更する必要がある．
ここでは左下隅を原点とする 10m×10m のスケール(単位は m)でウィンドウを定義する．
ここでは，Picture1 をビューポートとする．
まずビューポートの初期設定を，Form_Load()のサブルーチンで行う．

```
Private Sub Form_Load()
    Form1.ScaleMode = vbTwips
    Picture1.Width = 5000
    Picture1.Height = 5000
    Picture1.ScaleHeight = -10
    Picture1.ScaleWidth = 10
    Picture1.ScaleTop = 10
    Picture1.ScaleLeft = 0
End Sub
```

2)座標値の格納

基本的な描画方法については，これまでの授業で学んでいるが，描かれた図形に対する編集は困難であった．それは，ピクチャボックスをクリックした座標値を直接的に Line コ

マンドに受け渡しており，コンピュータ内にではその座標値を記憶していないためである．
　図形の編集を可能とするためには，座標値をコンピュータ上に記憶しておく必要がある．そこで，ここでは配列を持ちて，プロットした座標を格納する手法について説明する．

構造体を用いた座標の格納
　配列を Dim X(100), Y(100) As Single として定義し，それを用いて座標を格納していけばよいのであるが，X,Y は常にペアであるにも関わらず，それぞれが別個に定義され，プログラムでの使い勝手がよくない．そこで，ここで構造体を用いて 座標を定義する．ここでは，以下のように構造体を定義する．

```
Private Type Point2D 'Form のコード上では Private をつけなければ構造体は定義できない.
        x As Single
        y As Single
End Type
Dim PointOnLine(100) As Point2D
```

　上記のように定義することにより，例えば 10 番目の X 座標は PointOfLine(10).x，Y 座標は PointOfLine(10).y と表すことが可能となる．
　あとは，ウィンドウ上でのプロットに応じて座標点を描画するようにしておけばよい．

```
Dim cPointNum As Integer   '現在のポイント番号を覚えておくための変数を定義
Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
'    If cPointNum = 0 Then Picture1.PSet (X, Y) Else Picture1.Line -(X, Y)
   cPointNum = cPointNum + 1
   PointOnLine(cPointNum).x = x
   PointOnLine(cPointNum).y = y
   DrawLine
End Sub
Private Sub DrawLine()
    Picture1.Cls
    Dim n As Integer
    Picture1.PSet (PointOnLine(1).x, PointOnLine(1).y)
    For n = 2 To cPointNum
       Picture1.Line -(PointOnLine(n).x, PointOnLine(n).y)
    Next
End Sub
```

これで点の座標を格納することが可能である.

3)点の編集

すべての描画点について，その座標を格納しているから，その編集を行うことが可能である.

ここで，最も簡単な例として，Delete キーを押すことにより，直前の描画点を消去する例を示す.

```
Private Sub Picture1_KeyDown(KeyCode As Integer, Shift As Integer)
        'debug.print KeyCode '押されたキーの番号を確認する場合
      If KeyCode = 46 And cPointNum > 0 Then   '46 は Del キー
            cPointNum = cPointNum - 1
            DrawLine
      End If
End Sub
```

格納された座標値を変更するプログラムを書くことにより，点の移動等も可能である.

4)ファイルへの入出力

描かれた座標値をファイルとして出力する方法，またファイルを読み取り，描画を再現する方法を示す.

ファイルから読み込み描画を再現することを考えると，どのような情報が必要だろうか.

2.0, 1.0

22.2, 12.0

18.0, 17.0

.....

のように書き出すことは容易であるが，複数の線を扱う場合には，どれが始点でどれが終点であるかは明らかにしておく必要がある.

そのために２つの方法がある.

1)For　..Next　による読み込みを想定した方法

　For ..Next を利用する場合には，線を構成する点の数があらかじめ明らかになっていなければならない.

2)Do...Loop　による読み込みを想定した場合

　Do ...Loop を利用する場合には，座標値以外の数値や文字を読み込んだら終点であるという条件判断を行う必要がある.

ここでは，1)の方法を用いたプログラムの例を示す.

なお，事前に

CommonDialog コントロールを追加する.

Picture1 の AutoRedraw プロパティを True にしておく（上にウィンドウがかぶっても消えないように..）.

ファイルへの出力

```
Private Sub Command1_Click()
    SaveToFile
End Sub
Private Sub SaveToFile()
    Dim n As Integer
    CommonDialog1.ShowSave
    If CommonDialog1.filename = "" Then Exit Sub
    Open CommonDialog1.filename For Output As #1
    Write #1, cPointNum        'ポイント数を書く
    For n = 1 To cPointNum
        Write #1, PointOnLine(n).x, PointOnLine(n).y
    Next
    Close #1        'ファイルを閉じる
End Sub
```

ここで情報が適切に出力されているか，メモ帳などで開いてみること..

ファイルからの入力

```
Private Sub Command2_Click()
    LoadFile
End Sub
Private Sub LoadFIle()
    Dim n As Integer
    CommonDialog1.ShowOpen
    If CommonDialog1.filename = "" Then Exit Sub
    Open CommonDialog1.filename For Input As #1
    Input #1, cPointNum     'ポイント数を読む
    For n = 1 To cPointNum
      Input #1, PointOnLine(n).x, PointOnLine(n).y
    Next
    Close #1     'ファイルを閉じる
    DrawLine
End Sub
```

5)ファイルフォーマット統一の必要性

ソフトウェアによって，独自にフォーマットを定めて図形を定義することになるが，このことにより，他のソフトウェアで作成したデータが読み込めないと問題がある．

例えば，建築，土木分野では特に多くの業者が介在し，そのデータ交換も必要となる場合が多く，何らかの共通仕様が必要となる．

一般に普及している交換フォーマットとして，

DXF　（AutoCAD が開発した仕様だが，AutoCAD のバージョンアップにより頻繁に仕様が変わる）

IGES　(米国で定めた仕様）

などがある．また近年では，ネットワーク上での情報交換を目的とした DWG などがある．

また現在，オブジェクト指向のプロダクトモデルの開発が進んでいる．

STEP (Standard for the Exchange of Product Model Data)

IFC(Industrial Foundation Classes; 建築）

４．自由曲線を描く（パラメトリック曲線）

CAD で設計する物体は必ずしも直線や円などから構成されるわけではない．自動車のように曲線・曲面から構成される物体も多い．このような物体を CAD 上でいかに扱うかは大きな課題であり，多くの研究とその開発が行われている．（現在も進行中である．）

現在，一般に普及している自由曲線として

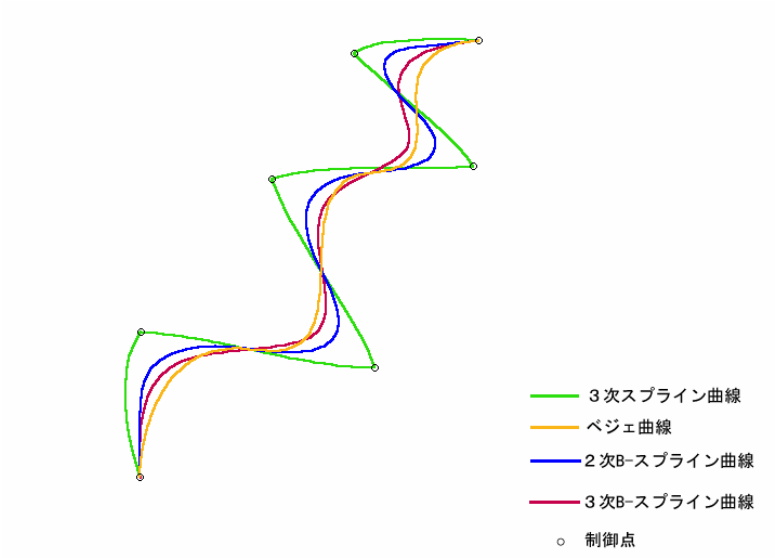
スプライン曲線

ベジェ曲線

B-スプライン曲線

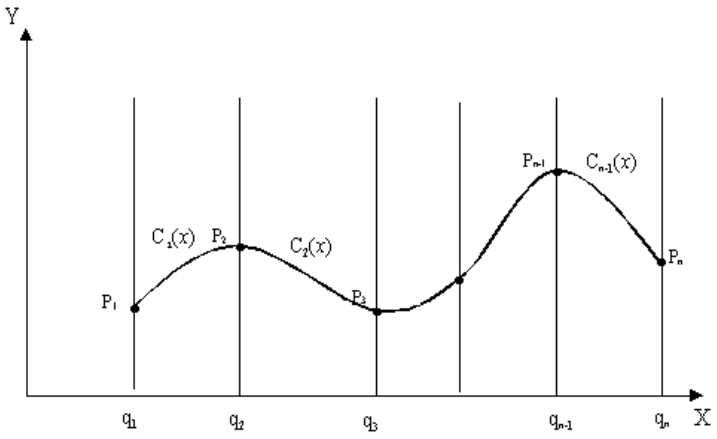
NURBS 曲線

などがある．



これらの曲線は，X,Y（3 次元の場合には Z も含む）の曲線上の座標を 1 つのパラメタをもとにして表現するものであり，**パラメトリック曲線**と呼ばれる．

曲線の描画には，任意の制御点を定義し，それに対して，パラメタを用いて，曲線を定義していく．



空間内に定められた P1(x1, y1, z1), P2(x2, y2, z2), P3(x3, y3, z3), P4(x4, y4, z4)の 4 点を境界条件として描く 3 次元の 3 次スプライン曲線上の座標値(x, y, z)は下式によって与えられる．

$$\begin{cases} x = \sum_{i=1}^{4} x_i B_i(t) = x_1 B_1(t) + x_2 B_2(t) + x_3 B_3(t) + x_4 B_4(t) \\ y = \sum_{i=1}^{4} y_i B_i(t) = y_1 B_1(t) + y_2 B_2(t) + y_3 B_3(t) + y_4 B_4(t) \\ z = \sum_{i=1}^{4} z_i B_i(t) = z_1 B_1(t) + z_2 B_2(t) + z_3 B_3(t) + z_4 B_4(t) \end{cases} \quad (t1 \leqq t \leqq t2)$$

ただし, t はパラメータであり, t1, t2 は始点と終点のパラメータ値である. Bi(t)は係数である.

それぞれの曲線の描画手法には, 特徴(短所を伴う)があり, 場合によって使い分けられている.

2)スプライン曲線の描画

ここでは, スプライン曲線を描くためのプログラムを作成する. その理論はともかく, 計算自体はそれほど難しいものではない. これは3次元の計算にもそのまま応用できる

スプライン曲線の計算

パラメータ t の範囲を 0≦t≦1, すなわち t1 = 0, t2 = 1 として正規化した場合, 3 次スプラインの係数 Bi(t)は, 下式により求められる).

$$\begin{cases} B_1(t) = -t(t-1)(t-2)/6 \\ B_2(t) = (t+1)(t-1)(t-2)/2 \\ B_3(t) = -t(t+1)(t-2)/2 \\ B_4(t) = t(t+1)(t-1)/6 \end{cases} \quad (0 \leqq t \leqq 1)$$

スプライン曲線描画のプログラム

```
Private Sub drawSpline()
    Dim x As Single , y As Single , b1 As Single , b2 As Single , b3 As Single , b4 As Single , t As Single
    Dim n As Integer

    If cPointNum < 4 Then
        MsgBox "Can't draw spline curve"
        Exit Sub
    End If
     'Picture1.Cls
    For n = 4 To cPointNum
        For t = 0 To 1.05 Step 0.2
            Debug.Print t
```

```
            b1 = -t * (t - 1) * (t - 2) / 6
            b2 = (t + 1) * (t - 1) * (t - 2) / 2
            b3 = -(t + 1) * t * (t - 2) / 2
            b4 = (t + 1) * t * (t - 1) / 6

            '以下の__は改行せずに打ち込むの意
            x = b1 * PointOnLine(n - 3).x + b2 * PointOnLine(n - 2).x + b3 * PointOnLine(n - 1).x__
            + b4 * PointOnLine(n).x
            y = b1 * PointOnLine(n - 3).y + b2 * PointOnLine(n - 2).y + b3 * PointOnLine(n - 1).y__
            + b4 * PointOnLine(n).y
            If t = 0 And n = 4 Then
                Picture1.PSet (x, y)
            Else
                Picture1.Line -(x, y), &HFF
            End If
        Next
    Next
End Sub
```

【補足】

(1)3 次のベジェ曲線

$$B_{n,i}(t) = \frac{n!}{(n-1)!i!} \cdot t^i \cdot (1-t)^{n-i} \qquad (0 \leqq t \leqq 1)$$

$$\begin{cases} B_{3,0}(t) = (1-t)^3 \\ B_{3,1}(t) = 3t(1-t)^2 \\ B_{3,2}(t) = 3t^2(1-t) \\ B_{3,3}(t) = t^3 \end{cases} \qquad (0 \leqq t \leqq 1)$$

(2)B-スプライン曲線

$$R(t) = \sum_{i=1}^{n+1} N_{i,k}(t) P_i \qquad (t_{min} \leqq t \leqq t_{max} \quad 2 \leqq k \leqq n+1)$$

・2 次

$$\begin{cases} N_{1,3}(t) = (1-t)^2 / 2 \\ N_{2,3}(t) = t(1-t) + 1/2 \\ N_{3,3}(t) = t^2 / 2 \end{cases} \quad (0 \leqq t \leqq 1)$$

・3 次

$$\begin{cases} N_{1,3}(t) = (1-t)^2 / 2 \\ N_{2,3}(t) = t(1-t) + 1/2 \\ N_{3,3}(t) = t^2 / 2 \end{cases} \quad (0 \leqq t \leqq 1)$$

他の曲線（NURBS を除く）の描画に関しては，下記書籍が参考となる．

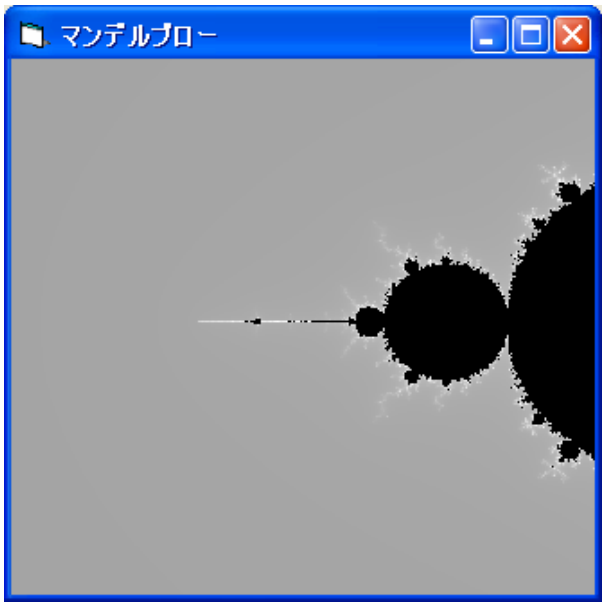
佐藤義雄「実習グラフィックス」アスキー出版局　1850 円　但し言語は N88-BASIC

フラクタル

１．フラクタル(Fractal)

Mandelbrot, B. 「FLACTRALS.form.chance and dimension」による造語で，「不規則な断片ができる」等の伊未．図形の部分と全体が自己相似になっているものを言う．

フラクタルの例としては海岸線の形が上げられ，海岸線はマクロ的には複雑に入り組んだ形状をしているが，これを拡大しても同様に入り組んだ形状（部分）により構成される．つまり，自己相似性を有する図形であるということが言うことができる．



マンデルブロー集合

このような概念に基づくと海岸線には長さが存在せず，無限大となる．

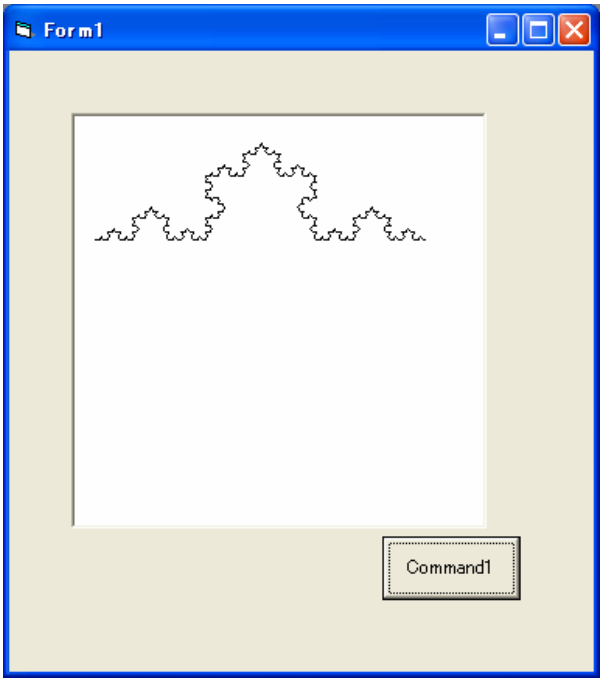
フラクタル図形における複雑さを表す定量的に表す指標がフラクタル次元である．自己相似性のある図形において，図形が 1/n に縮小した相似形 m 個により構成されているとき，そのフラクタル次元は，下式により表される．

$$D = \log m / \log n$$

となる．（後述のコッホ曲線の場合のフラクタル次元は， $\log_3 4 = \log 4 / \log 3 = 1.26$）

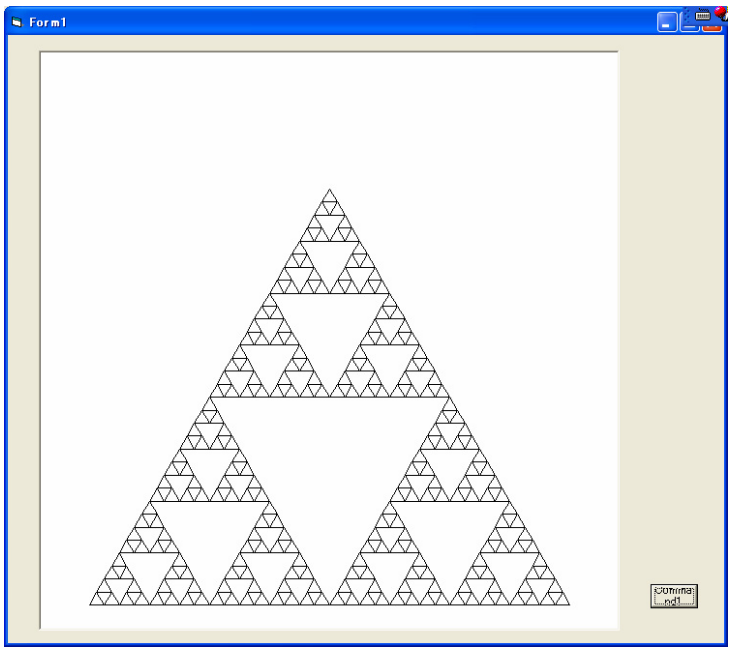
２．コッホ曲線を描画するプログラム

```
Private Type xy
        x As Single
        y As Single
End Type
Dim pointer As xy
Dim angle As Integer

Private Sub Command1_Click()
        Dim n As Integer
        Dim length As Integer

        n = 4
        length = 4
        Picture1.Scale (0, 0)-(400, 400)
        pointer.x = 20: pointer.y = 120
        angle = 0
        kochcurve n, length
End Sub

Sub kochcurve(n As Integer, length As Integer)
        If n = 0 Then
                fmove length, 3
        Else
                kochcurve n - 1, length
                turn 60
                kochcurve n - 1, length
                turn -120
                kochcurve n - 1, length
                turn 60
                kochcurve n - 1, length
        End If
End Sub

Private Sub fmove(length, col)
        x = length * Cos(3.1415 / 180 * angle)
        y = length * Sin(3.1415 / 180 * angle)
        Picture1.Line (pointer.x, pointer.y)-(pointer.x + x, pointer.y - y), col
        pointer.x = pointer.x + x
        pointer.y = pointer.y - y
End Sub

Sub turn(deg)
        angle = (angle + deg) Mod 360
```



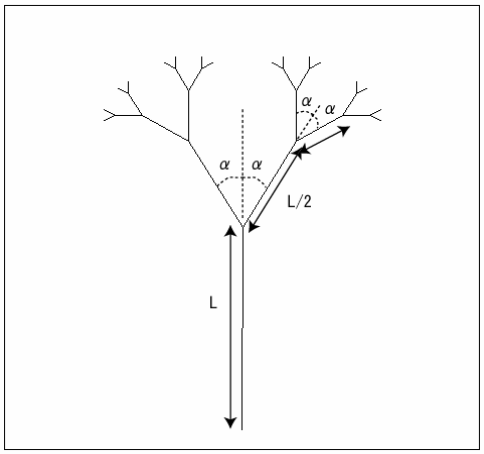
３．シルピンスキーガスケットを作成するプログラム

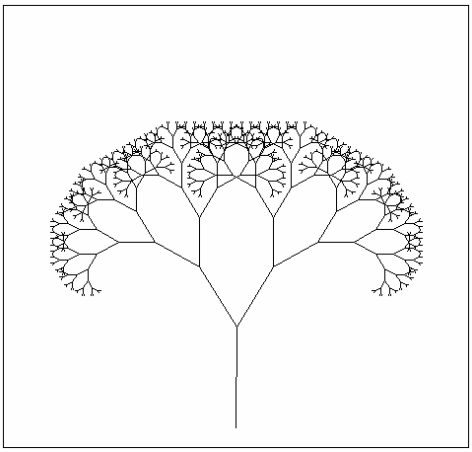
```
Private Type xy
        x As Single
        y As Single
End Type
Dim p(3) As xy


Private Sub Command1_Click()

        Picture1.Scale (-120, 230)-(120, -10)
        p(1).x = 0: p(1).y = Sqr(3) * 100
        p(2).x = -100: p(2).y = 0
        p(3).x = 100: p(3).y = 0

        triangle 5, p(1), p(2), p(3)

End Sub

Private Sub triangle(n As Integer, p1 As xy, p2 As xy, p3 As xy)
        Dim pa As xy, pb As xy, pc As xy

        If n = 0 Then
                Picture1.Line (p1.x, p1.y)-(p2.x, p2.y)
                Picture1.Line (p2.x, p2.y)-(p3.x, p3.y)
                Picture1.Line (p3.x, p3.y)-(p1.x, p1.y)
                Exit Sub
        End If

        pa.x = (p1.x + p2.x) / 2
        pa.y = (p1.y + p2.y) / 2
        pb.x = (p2.x + p3.x) / 2
        pb.y = (p2.y + p3.y) / 2
        pc.x = (p3.x + p1.x) / 2
        pc.y = (p3.y + p1.y) / 2

        triangle n - 1, p1, pa, pc
        triangle n - 1, pa, p2, pb
        triangle n - 1, pc, pb, p3
End Sub
```



４．樹木曲線を描くプログラム

長さＬの主幹を描画した後，一定の手順を数回，繰り返すことにより樹木曲線が作成できる．





```
Private Type xy
    x As Single
    y As Single
End Type
Dim angle As Single '初期角
Dim addangle As Single '分岐角
Dim bscale As Single '枝の伸び率

Private Sub Command1_Click()
    Dim p As xy
    Picture1.Scale (-11, 21)-(11, -1)
    angle = 90
    addangle = 30
    bscale = 0.7
    length = 5
    p.x = 0: p.y = 0
    branch 10, p, length, angle
End Sub

Private Sub branch(n, p0 As xy, l, angle)
    Dim p1 As xy
    If n = 0 Then Exit Sub
    p1.x = p0.x + l * Cos(angle / 180 * 3.1415)
    p1.y = p0.y + l * Sin(angle / 180 * 3.1415)
    Picture1.Line (p0.x, p0.y)-(p1.x, p1.y)
    branch n - 1, p1, l * bscale, angle + addangle
    branch n - 1, p1, l * bscale, angle - addangle
End Sub
```

参考文献：河西朝雄：改訂Ｃ言語によるはじめてのアルゴリズム入門，技術評論社, 2001.

図形の回転と移動　－アフィン変換－

図形の回転や移動，縮小，あるいは視点の移動や回転のための座標変換はアフィン変換（Affine transformation)と呼ばれ，コンピュータグラフィックスの基礎として極めて重要である．ここでは,図形の移動と回転，縮小，視点の移動と回転について学ぶ.

１．座標変換の式

点$(x, y,)$を点(X,Y)に変換する．

①代数方程式

$$X = ax + by + l$$
$$Y = cx + dy + m$$

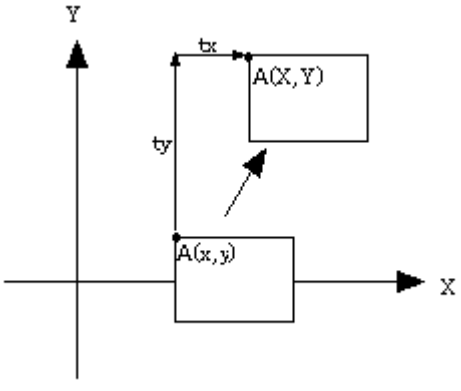
②行列

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = T \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & l \\ c & d & m \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

T を変換行列と呼ぶ

２．図形の平行移動 Translation

点 $A(x,\ y)$を X 方向に tx，Y 方向に ty 移動した点 A'(X, Y)は，下式によって表現される．



$$X = x + tx$$
$$Y = y + ty$$

または,

$$T = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

これを VB 上でのサブプロシージャとして定義すれば，以下のようになる．

```
'図形移動のためのサブプロシージャ
Private Sub Translate(p() As point2D, s As Integer, e As Integer, tx As Single, ty As Single)
    Dim n As Integer
    For n = s To e
        p(n).x = p(n).x + tx
        p(n).y = p(n).y + ty
    Next
End Sub
```

p( ) ： あらかじめ設定された 3 次元の座標配列が受け渡される．
s :座標配列のうち，変換を行う開始番号（インデックス）
e :座標配列のうち，変換を行う終了番号（インデックス）
tx : X 方向の移動量
ty : Y 方向の移動量

３．図形の回転 Rotation

原点を中心に図形を回転すると考える．
図形上の頂点 A（x, y)を Z 軸回りに α だけ回転した頂点 A'(X, Y)は，下式によって表現される．

$$X = x\cos\alpha - y\sin\alpha$$
$$Y = x\sin\alpha + y\cos\alpha$$

または,

$$T = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

原点を中心に回転変換を行う VB のサブプロージャは，以下のように表わされる．

```
'原点を中心とした回転変換プロシージャ
Private Sub Rotate (p() As point2D, s As Integer, e As Integer, r As Single)
    Dim n As Integer
```

```
    Dim x As Single, y As Single, rr As Single

    rr = r / 180 * PAI          'sin,cos の計算にラジアンへの変換が必要
    For n = s To e
        x = p(n).x :   y = p(n).y           '別の変数に受け渡しておく
        p(n).x = x * Cos(rr) - y * Sin(rr)
        p(n).y = x * Sin(rr) + y * Cos(rr)
    Next
End Sub
```

p() ： あらかじめ設定された 3 次元の座標配列が受け渡される．
s :座標配列のうち，変換を行う開始番号（インデックス）
e :座標配列のうち，変換を行う終了番号（インデックス）
r :回転角（度）

４．任意の点を中心とした図形の回転

３．での図形の回転は原点(0,0)を中心としたものであった．しかし実際には，物体（オブジェクト）の中心で回転させるなど，任意の点を中心とした回転が必要となる場合が多い．このような変換はどのようにして行うのだろうか.

物体を任意の中心点においたまま，計算することは非常に厄介なことであり，一般には以下のような方法が取られる..

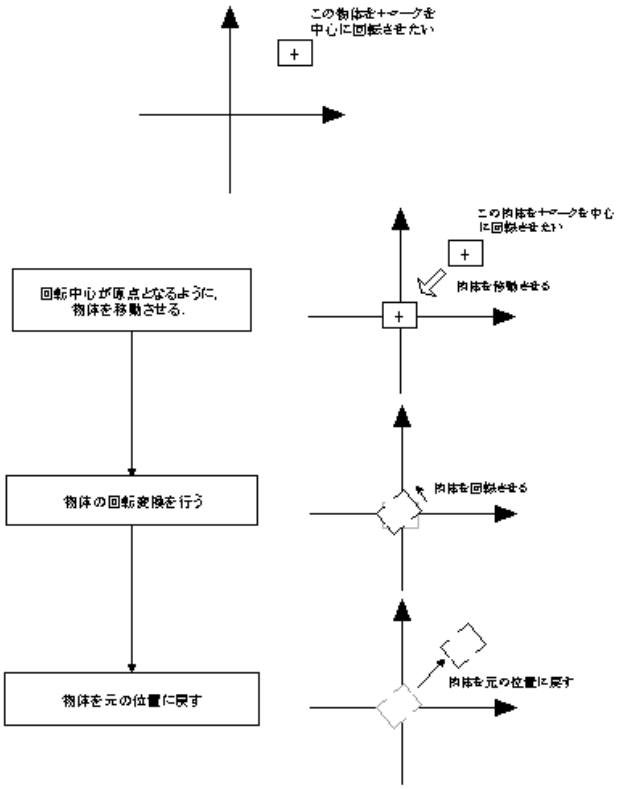


図-3　任意の点を中心とした物体の回転

つまり，Translate>>Rotate>>Translate という 3 の座標変換を行う．この方法により，容易に簡単に物体の回転を行うことが可能となる．

５．図形の拡大と縮小（Scaling）

図形の拡大，縮小は，図形の各頂点の座標値に対して，X 方向・Y 方向の拡大（縮小）率を乗じる．図形上の頂点 A(x, y)を X 方向の拡大率 s_x，Y 方向の拡大率 s_y でスケーリング変換するとすれば，その変換後の点 A'(X, Y)は，下式により表現される．

$$X = s_x \cdot x$$
$$Y = s_y \cdot y$$

または

$$T = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

これを VB 上のサブプロシージャとして定義すれば以下のようになる．

```
'スケーリング変換を行うサブプロシージャ
Private Sub ScaleA(p() As point2D, s As Integer, e As Integer, sx As Single, sy As Single)
    Dim n As Integer
    For n = s To e
        p(n).x = p(n).x * sx
        p(n).y = p(n).y * sy
    Next
End Sub
```

原点(0,0)に対する拡大率となるので，任意の点（例えば図形の中心）で拡大・縮小を行う場合には，４．で行なったような図形の移動処理が必要である．

（重要）
変換の組合わせ順序（例えば移動>>回転>>拡大・縮小とするか，回転>>拡大縮小>>移動とするか）により，変換結果が異なるので，十分に注意すること．

【演習】
プリントに挙げたアフィン変換のサブプロシージャを用いたプログラムを作成しなさい．

６．アフィン変換を用いた図形描画プログラム

【基本プログラム（原点を中心に回転する四角形)】
Form1 上に Picture コントロール（Picture1),Command ボダン(Command1)配置する.
（Picture1 のサイズは Form_Load サブプロシージャで定義されている）

```
Option Explicit
Private Type point2D
    x As Single
    y As Single
End Type

Dim pnt(100) As point2D
Dim rot As Single
Dim trans As point2D

Const PAI = 3.1415926

Private Sub Command1_Click()
    draw
End Sub

Private Sub Form_Load()
    Form1.ScaleMode = vbTwips
    With Picture1 'With...EndWith を用いるとプロパティの設定をブロック化できる.
        .Width = 4000
        .Height = 4000
        .AutoRedraw = True
    End With
    Picture1.Scale (-5, 5)-(5, -5)
End Sub


Private Sub draw()
    Picture1.Cls

    '図形の座標点の定義
    pnt(0) = setpnt(-1, -1)
    pnt(1) = setpnt(1, -1)
    pnt(2) = setpnt(1, 1)
    pnt(3) = setpnt(-1, 1)
    Quad pnt(0), pnt(1), pnt(2), pnt(3), QBColor(0)

    '原点を中心とした図形回転
    Rotate pnt(), 0, 3, 30
    Quad pnt(0), pnt(1), pnt(2), pnt(3), QBColor(9)

    'X 方向に 2，Y 方向に 0 移動
    Translate pnt(), 0, 3, 2, 0
    Quad pnt(0), pnt(1), pnt(2), pnt(3), QBColor(13)

    'さらに X 方向に-1，Y 方向に 2 移動
    Translate pnt(), 0, 3, -1, 2
    Quad pnt(0), pnt(1), pnt(2), pnt(3), QBColor(14)
```

```
End Sub

Private Function setpnt(x As Single, y As Single) As point2D
    setpnt.x = x
    setpnt.y = y
End Function

Private Sub Quad(p0 As point2D, p1 As point2D, p2 As point2D, p3 As point2D, col As Long)
    Picture1.Line (p0.x, p0.y)-(p1.x, p1.y), col
    Picture1.Line -(p2.x, p2.y), col
    Picture1.Line -(p3.x, p3.y), col
    Picture1.Line -(p0.x, p0.y), col
End Sub

Private Sub Translate(p() As point2D, s As Integer, e As Integer, tx As Single, ty As Single)
    Dim n As Integer
    For n = s To e
        p(n).x = p(n).x + tx
        p(n).y = p(n).y + ty
    Next
End Sub

Private Sub Rotate(p() As point2D, s As Integer, e As Integer, r As Single)
    Dim n As Integer
    Dim x As Single, y As Single, z As Single, rr As Single
    rr = r / 180 * PAI
    For n = s To e
        x = p(n).x: y = p(n).y
        p(n).x = x * Cos(rr) - y * Sin(rr)
        p(n).y = x * Sin(rr) + y * Cos(rr)
    Next
End Sub
```

【問題】

1. 点（１，２）を中心とする１辺の長さが２の正方形を描画しなさい.
2. １で描いた正方形をＸ方向に２，Ｙ方向に１移動させなさい.
3. ２で描いた正方形をその中心で＋６０度回転させなさい.
4. タイマーコントロールを用いて，３で描いた図形が回転するアニメーションプログラム作成しなさい.
5. 三角形（あるいはその他の任意の図形）も描画できるようにプログラムを変更し，それを用いた描画を行いなさい.

【補足】図形を塗りつぶしする場合の追加コード

```
Private Declare Sub FloodFill Lib "GDI32" (ByVal hDC As Long, _ByVal x As Long, ByVal y As Long, ByVal Color As Long)


Private Type UV
    u As Single
    v As Single
```

```
End Type

Private Sub QuadFill(p0 As point2D, p1 As point2D, p2 As point2D, p3 As point2D, bColor As
Long, fColor As Long)
    Dim average As point2D
    Dim wScale As UV
    Dim pixelFill As UV

    Form1.ScaleMode = vbPixels
    wScale.u = Picture1.Width / Picture1.ScaleWidth
    wScale.v = Picture1.Height / Picture1.ScaleHeight

    Picture1.Line (p0.x, p0.y)-(p1.x, p1.y), bColor
    Picture1.Line -(p2.x, p2.y), bColor
    Picture1.Line -(p3.x, p3.y), bColor
    Picture1.Line -(p0.x, p0.y), bColor
    average.x = (p0.x + p1.x + p2.x + p3.x) / 4
    average.y = (p0.y + p1.y + p2.y + p3.y) / 4
    pixelFill.u = (average.x - Picture1.ScaleLeft) * wScale.u
    pixelFill.v = (average.y - Picture1.ScaleTop) * wScale.v

    Picture1.FillStyle = vbFSSolid ' FillStyle プロパティを塗りつぶしに設定します。
    Picture1.fillColor = fColor
    FloodFill Picture1.hDC, CLng(pixelFill.u), CLng(pixelFill.v), bColor
End Sub
```

3 次元コンピュータグラフィックスの基礎(1)
―3 次元データモデルの構築―

１．3 次元コンピュータグラフィックスとは

3 次元コンピュータグラフィックス（3 次元 CG）は，描画対象となるデータを 3 次元情報として定義し，それをコンピュータディスプレイ上で表現する技術である．

3 次元 CG の基礎となる技術は以下の 3 つに分類される．

①3 次元データモデルの定義

②3 次元モデルの 2 次元平面への投影変換

③3 次元モデルの移動と回転（アフィン変換）

③表現手法（隠線・隠面の処理，着色，陰影等）

２．3 次元データモデル

コンピュータ上で 3 次元モデルを構築するためのデータモデルとして，以下の３つがある．

①ワイヤフレームモデル

ワイヤフレームモデルは，立体の骨組み構造を線として表現するモデルである．面のレンダリングはできない．ただし，描画が高速に行なえるメリットがあり，データ量の多い物体を表示する場合には有利である．

②サーフェースモデル

3 次元物体の面に着目し，面（ポリゴン）として定義していく方法である． 面の隠線消去やレンダリングを行なうことができる．（一般的な３次元ＣＧソフトで利用されている）

③ソリッドモデル

物体を中身が詰まった固体として捉える方法である．図形間の演算が行えるメリットがある．ただし，データ生成や処理に多量の時間が必要である．

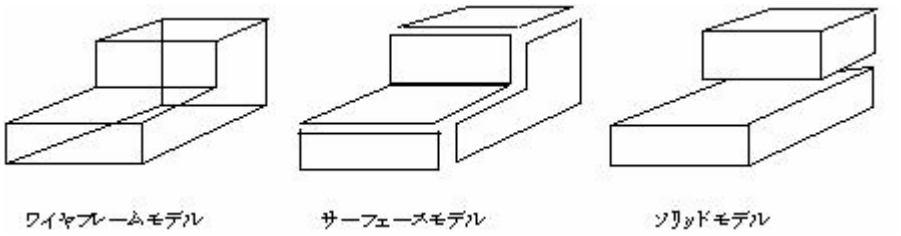


図-1　立体の表現方法

３．サーフェースモデルによる形状定義

ここでは，これらの方法のうち，サーフェースモデルの概念をもとに，3 次元 CG について述べていく．

(2)サーフェースモデルによる形状定義

１つの直方体の形状定義を例として考える．

直方体は，下図に示すように６つの面から構成されている．ここで面０～面５として，以下のように定義する．

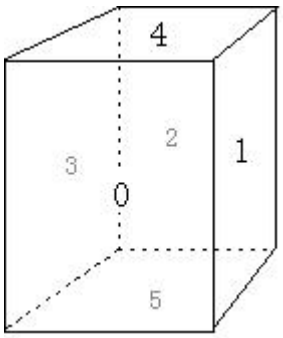


図-2　直方体の面

直方体を構成する面はすべて四角形であり，それぞれの面は４つの頂点を有している．頂点番号を下図のように定義する（図中( )が頂点番号）

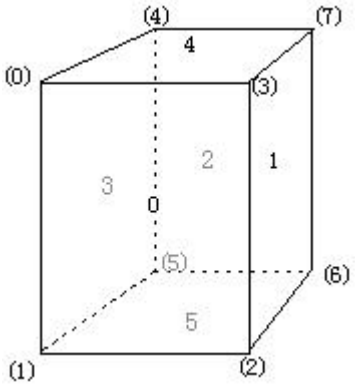


図-3　直方体の頂点

それぞれの面に対して，構成する頂点を整理すると以下のとおりになる．

面０：(0)(1)(2)(3)

面１：(2)(3)(6)(7)

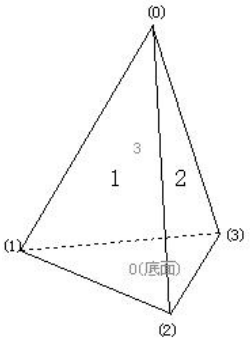
面２：(4)(5)(6)(7)
面３：(0)(1)(4)(5)
面４：(0)(3)(4)(7)
面５：(1)(2)(5)(6)

しかし，これらの情報から面を復元することはできない．それは点の順番と面との関係が統一されていないためである． そこで，面に対する番号付けを，面の表に反時計周りで付けることにする．
（番号付けの方法を定義しておくことは，面の向き（表・裏）を判定する上で極めて重要である.）
これらを再整理すると以下のようになる．

面０：(0)(1)(2)(3)
面１：(3)(2)(6)(7)
面２：(7)(6)(5)(4)
面３：(0)(4)(5)(1)
面４：(0)(3)(7)(4)
面５：(1)(5)(6)(2)

サーフェースモデルでは，面とそれを構成する点により，その 3 次元モデルをコンピュータ上に定義する．ここで，それぞれの面を**ポリゴン(polygon)**と呼ぶ.

問題：
以下のように面０から面３により構成される三角錐がある. これらの面を頂点番号( )により表わしなさい． （三角錐の表面が表である.）



面０：
面１：
面２：
面３：

問題２：
任意の大きさの六角柱を定義し，頂点番号を定めるとともに，各面を頂点番号により表しなさい.

４．３次元の座標系

3 次元モデルの構築を行う場合，図形の頂点を座標，つまり数値で表す必要がある．そのため，空間を定義するための 3 次元座標系が必要となる． 座標系には，右手座標系と左手座標系という2つの種類があり，前者をワールド座標系，後者を視点座標系と呼ぶ場合もある．

どちらの座標系を用いるかは，ソフトウェアや適応分野，また場合に応じて異なるが，ここでは OpenGL や VRML などで採用されている右手座標系を用いることとする．

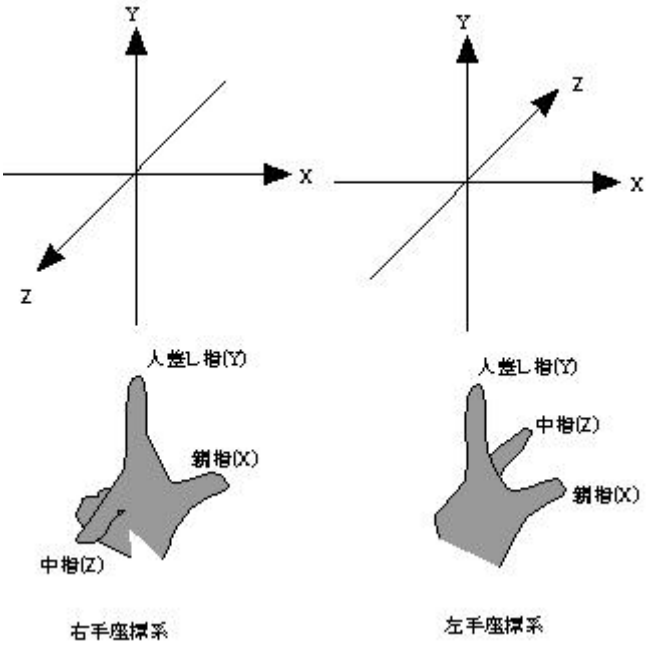


図-4 右手座標系と左手座標系

５．３次元形状モデルの定義

これらの手法により， 3 次元形状モデルは以下の情報により定義できる．

①頂点の座標情報（頂点の x, y, z 座標）

頂点１：x_1, y_1, z_1

頂点２：x_2, y_2, z_2

頂点３：x_3, y_3, z_3

頂点・・・・（以下略）

②面を構成する頂点情報

面１：頂点１，頂点２，頂点５

面２：頂点・・・・（以下略）

例：

Points

(0): -0, 2, -1

(1): -1.5, 0, -1.5

(2): 1.5, 0, -1.5

(3): 0, 0, 2

Polygons

1: (2) (1) (3)

2: (0) (2) (3)

3: (0) (3) (1)

4: (0) (1) (2)

問題３：

任意の直方体を定義し，その形状モデルを作成しなさい.

問題４：

問題２で作成した六角柱に対して頂点座標を定め，形状モデルを完成させなさい.

問題５：

任意の物体の形状モデルを作成しなさい.

３次元コンピュータグラフィックスの基礎(2)
－投影変換―

１．２次元平面への投影

作成した 3 次元モデルを，2 次元平面であるコンピュータディスプレイ上に描画するためには，3D/2D 変換が必要である．この変換は投影変換と呼ばれ，その方法として，

(1)平行投影

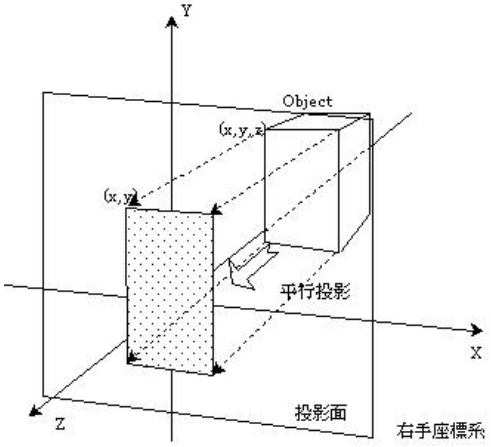
(2)透視投影（中心投影）

の２つの方法がある．

２．平行投影　orthographic projection

平行投影は，空間内の物体を２次元平面に対して平行に投影する方法である．この投影は，物体の無限遠点からの照明に対する影，すなわち太陽のようにはるか遠方にある照明に対する影と考えてもよい．

図－１は，空間内に位置する直方体を XY 平面に対して投影したものである．この投影法は，物体の X,Y,Z 各成分のうち，Z 成分を考慮せずに，X,Y のみで描画することにより，実現される．



図－１　平行投影の概念

平行投影は，製図においては一般的に用いられており， X 軸，Y 軸，Z 軸それぞれについての平行投影により，平面図，立面図，側面図が作成できる．

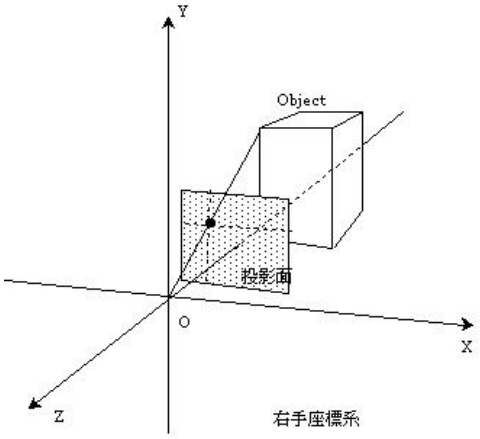
３．透視投影（中心投影）　perspective projection

人間の目を通して見える物体は，近くのものほど大きく，また遠くのものほど小さく見える．これは眼球の水晶体というレンズを通し，網膜に対して投影していることによる．またカメラで撮影した画像も，同様に遠方のものほど小さく写る．これも同様に，カメラのレンズを通して，フィルムに対して投影していることによる．これらの投影法は，透視

投影（中心投影）によるものである．

コンピュータグラフィックスによっても同様の原理を適用すれば，人間の目で見るのと同様の立体感のある画像を得ることが可能である．

図－２に示すように，空間内にある立方体が存在している．この図形の透視投影像を求める．

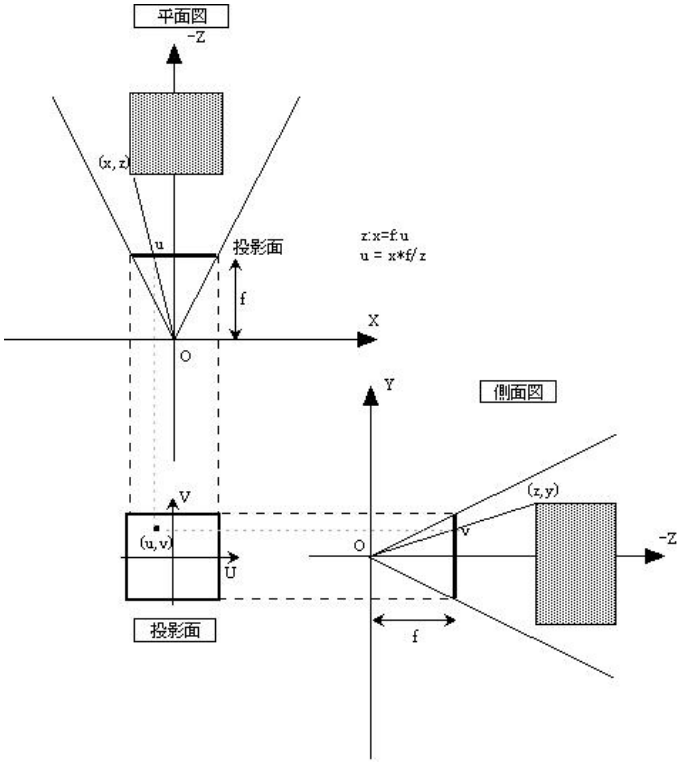


図－２　透視投影の概念

まず視点の位置を原点とする．次に視点から Z 方向に f だけ離れた位置に任意の大きさの投影面を設ける．これは，目の前にガラス板を置いて，その上に描画すると考えればよい. そのガラス面までの距離が f に相当し, その大きさはガラス板の大きさと考えればよい.

この投影面の座標系を横方向に U，縦方向に V という UV という座標系で定義する．ここでは，その中心に原点が位置すると考えれば理解しやすい.

物体の頂点（x, y)が UV 座標上に投影される位置（u, v）について考えてみよう.



図－３　透視投影のしくみ

まず XZ 座標について考える．すなわち，これは U 座標を求めるための計算である．図－３に示すように，u の座標値は，焦点距離 f と x，z 座標により，以下のように表現される．

$$z : x = f : u$$

これを変形すれば，座標 u は，

$$u = f \cdot x / z$$

となる．

一方，XY 座標について考えてみれば，同様に，

$$z : y = f : v$$

これを変形して，

$$v = y * f / z$$

となる．

すなわち，UV 座標系で定義される投影面上において，空間内の点(x, y)の投影点は，

$$(f \cdot x / z,\ y * f / z)$$

と表される．

４．VB 上での透視投影変換プログラム

```
Option Explicit
Dim f As Single        '投影面の z 座標上の位置を示す
Private Type point3D
    x As Single
    y As Single
    z As Single
End Type
Private Type UV
    u As Single
    v As Single
End Type
Dim pnt(100) As point3D

Private Sub Command1_Click()
    Call draw          'コマンドボタンが押されたら draw を呼び出す.
End Sub

Private Sub Form_Load()        'プログラム起動時に設定する初期値
    Form1.ScaleMode = vbTwips
    With Picture1          'With...EndWith を用いるとプロパティの設定をブロック化できる.
        .Width = 3000
        .Height = 3000
        .ScaleHeight = -2
        .ScaleWidth = 2
        .ScaleTop = 1
        .ScaleLeft = -1
    End With
    f = -2          '投影面の位置を-2 に設定する.
End Sub

Private Sub draw()
    pnt(0) = setPnt(-1, 1.5, -5)
    pnt(1) = setPnt(-1, -1.5, -5)
    pnt(2) = setPnt(1, -1.5, -5)
    pnt(3) = setPnt(1, 1.5, -5)
    pnt(4) = setPnt(-1, 1.5, -6)
    pnt(5) = setPnt(-1, -1.5, -6)
    pnt(6) = setPnt(1, -1.5, -6)
    pnt(7) = setPnt(1, 1.5, -6)

    Quad pnt(0), pnt(1), pnt(2), pnt(3)
    Quad pnt(3), pnt(2), pnt(6), pnt(7)
    Quad pnt(7), pnt(6), pnt(5), pnt(4)
    Quad pnt(0), pnt(4), pnt(5), pnt(1)
    Quad pnt(0), pnt(3), pnt(7), pnt(4)
    Quad pnt(1), pnt(5), pnt(6), pnt(2)
```

```
    '     Picture1.Line -(U(pnt(4)), V(pnt(4)))
End Sub

'投影面上の u 座標を計算する関数（重要）
Private Function u(p As point3D) As Single
    u = p.x * f / p.z
End Function

'投影面上の v 座標を計算する関数（重要）
Private Function v(p As point3D) As Single
    v = p.y * f / p.z
End Function


'ポイントを構造体に受け渡すための関数（重要）
Private Function setPnt(x As Single, y As Single, z As Single) As point3D
    setPnt.x = x
    setPnt.y = y
    setPnt.z = z
End Function

'四角形を簡単に描画するためのサブルーチン（重要）
Private Sub Quad(p0 As point3D, p1 As point3D, p2 As point3D, p3 As point3D)
    Picture1.Line (u(p0), v(p0))-(u(p1), v(p1))
    Picture1.Line -(u(p2), v(p2))
    Picture1.Line -(u(p3), v(p3))
    Picture1.Line (u(p3), v(p3))-(u(p0), v(p0))
End Sub
```

(演習）

(1)任意の三角錐を定義し，透視投影するプログラムを作成しなさい.

(2)空間内に任意の図形（何らかのテーマ性をもつもの）を定義し，
透視投影するプログラムを作成しなさい.

１．試験対策について

(1)画像処理アルゴリズム
・濃度変換関数
・空間フィルタの仕組み
・モザイク
・画像合成
(2)パラメトリック曲線
・円の描画
　アフィン変換（プログラム）
　フラクタルとは（コッホ曲線）
(3)３次元 CG
・形状定義
・透視投影（プログラム）

２．コンピュータグラフィックスⅡ（後期）について

・３次元アフィン変換（図形の回転と移動）
・隠線処理
・OpenGL による３次元グラフィックスプログラミング