第2章
グラフィックツールの製作

本章では，コンピュータグラフィックスに必要な各種の描画アルゴリズムを解説するとともに，多色モードでのグラフィックサブルーチンを提供します．グラフィックサブルーチンは，アルゴリズムの理解を目的としているためすべてBASICで記述しています．その結果，実行速度は決して速いとはいえませんが，IN,OUT文を多用するなど機械語化しやすいコーディングになっています．従って，アルゴリズムを理解された読者は，機械語化を試みるのもよいでしょう．

2-1 各種描画アルゴリズム

2-1-1 直線描画のアルゴリズム

直線描画のアルゴリズムには，いろいろありますが，ここではBresenhanのアルゴリズムについて説明します．直線を引くアルゴリズムにおいては，いかにしてもっとも効率よくとぎれない線を引くかが重要になってきます．このBresenhanのアルゴリズムでは，直線を描画する方法が，x方向の変化量とy方向の変化量によって大きく分けられます．

これは，変化量の大きい方を基準にした方が，分割数が増え，きれいな線が，描けるからです．

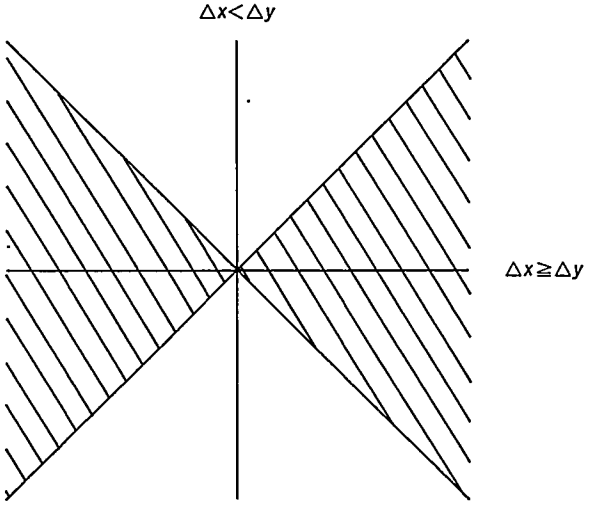


図2-2

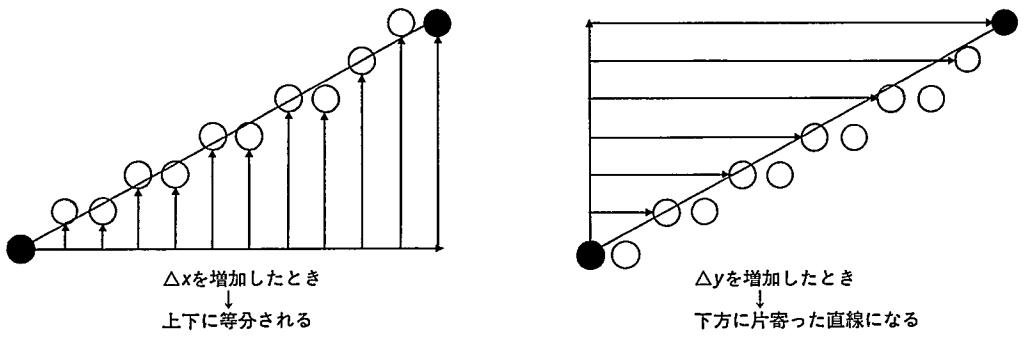


図2-3

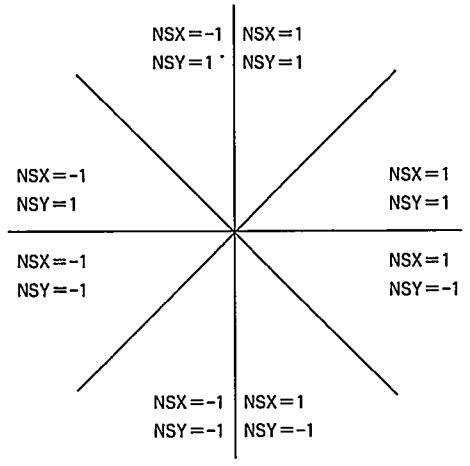


図2-4　xとyの増分値

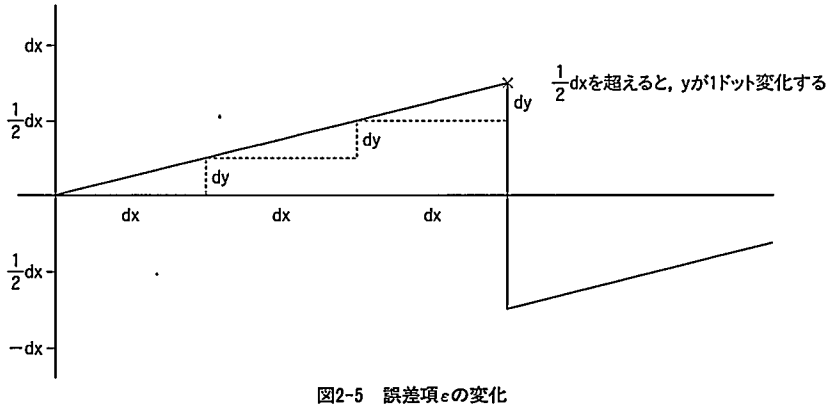


図2-5　誤差項εの変化

直線を描画しようとするとき，変化量の大きい方を１ドット増加または減少させた場合に，もう片方が１ドット増加または，減少するのか，変化しないのかを，誤差項NEで判定します．つまり，誤差項NEが，NDXより大きいか小さいかによって１ドット変化させるかどうかが決定されるわけです．

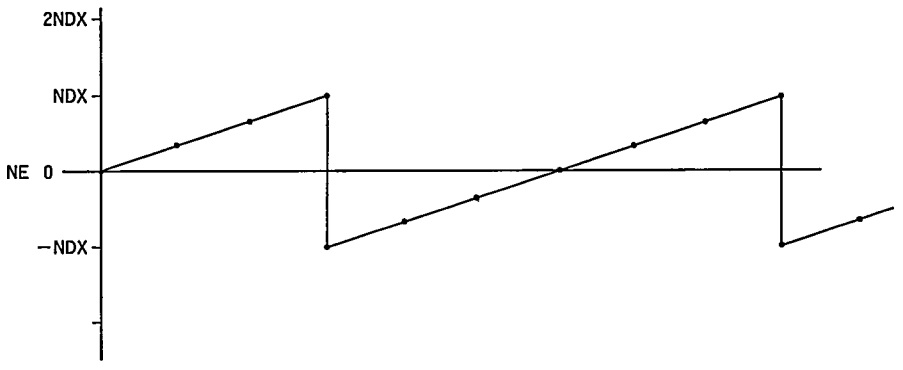


図2-6 整数型誤差項NEの変化

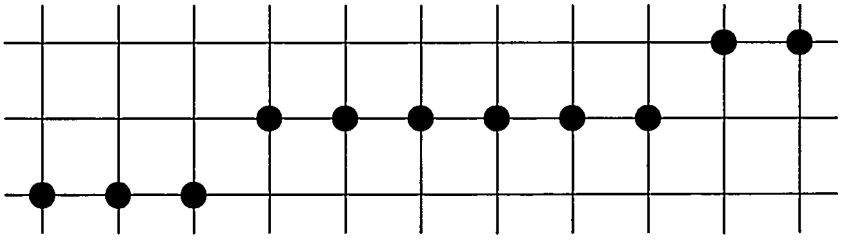


図2-7 画面上のドット表示

2-1-2 グラデーション付きの直線描画

直線を描画するとき，始点と終点の座標値と，直線の色を指定すると，その直線は同じ色で表示されます．ここでは，始点と終点に対して別の色を指定することにより，その間の色を補間し，グラデーションをつけて描画する方法について説明します．

直線を描画するとき，ｘとｙの変化量が大きい方の値を１ドットずつ変化させた方が，きれいな線が描けることはすでに説明しましたが，色についても同じことが言えるため，同様の方法で補間します．本サブルーチンでは，色は，RGB で表すような仕様になっていますので，RGB それぞれに対して，補間していきます．HSV やカラーコードで表現している場合も，同様にできます．ただし，HSV の場合，HUE は 359° の次が 0° になるため，注意が必要です．

2-1-3 グラデーション付きの 3 角形表示アルゴリズム

３角形の頂点の座標値と色を与えることにより，グラデーションを付けて表示する方法は，曲面の多面体表示など３次元グラフィックスではよく使われています．しかしイラストを描くときにも，３点の座標と，色を指定するだけで，その内部にグラデーションをつけて塗りつぶす機能があれば，非常に使いやすいものとなります．そこで，そのアルゴリズムについて説明します．このアルゴリズムは，グーローシェーディングと呼ばれる手法を参考にしています．

画面座標系での3点$P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$とそれらの点の色$C_1(R_1, G_1, B_1)$, $C_2(R_2, G_2, B_2)$, $C_3(R_3, G_3, B_3)$が与えられているとします．ここで，3角形$P_1P_2P_3$の内部にある任意の点P(x,y)の色を求めてみます．まず，y座標について，ソートすると，次のようになります．

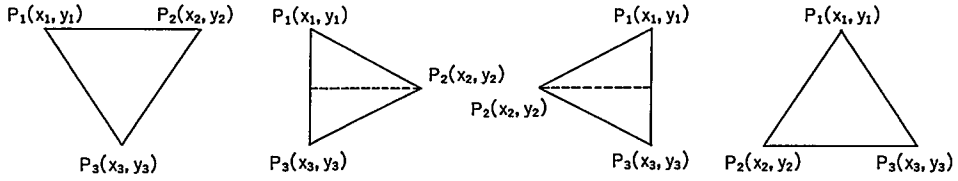


図2-8　三角形の種類

点P_2を通る水平線と辺P_1P_3との交点を$P_4(x_4, y_4)$とすると3角形$P_1P_4P_2$と3角形$P_3P_2P_4$に分けられます．

点Pを通る水平線と3角形との交点を$P_5(x_5, y_5)$, $P_6(x_6, y_6)$とします．点P_5の色とx座標は，P_1P_3を線形補間することにより求めることができます．点P_6の色は，$y_6 \leqq y_2$のときはP_1P_2を線形補間し，$y_6 > y_2$のときは，P_2P_3を線形補間することによって求めることができます．P_5P_6の色が決まれば，グラデーション付のラインと同じ方法で水平線を描画していけばよいでしょう．

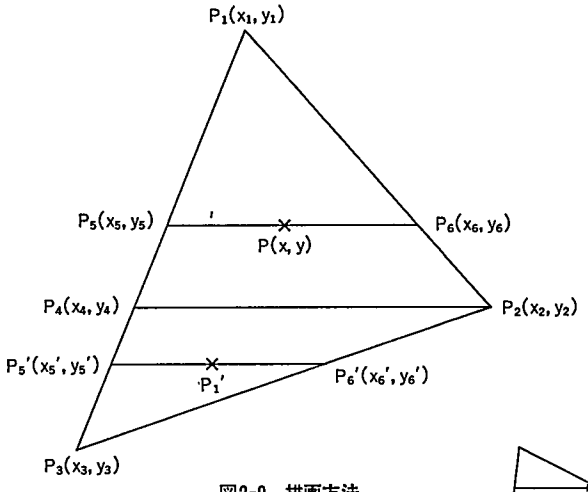


図2-9　描画方法

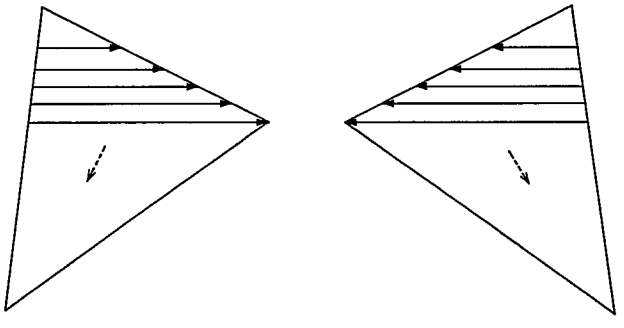


図2-10　描画順序

2-2 グラフィックサブルーチン

2-2-1 本グラフィックサブルーチンの特徴

- MERGE コマンドによるモジュール結合
- GOTO 文と REM 文によるブロック IF 文
- 変数名の命令規則によるモジュール変数の分離
- 他言語への１対１対応の移植性
- コメント行によるドキュメント化
- ラベル付きサブルーチンによる機能モジュール提供
- DEFINT，etc による，変数の型宣言

(1) MERGE コマンドによるモジュール結合

本グラフィックサブルーチンは行番号が10000行から始まっており，ユーザーの作成するメインルーチンの行番号と重複しないようになっています．ですからメインルーチンは，１～9999で作成し，必要なモジュールを MERGE すればプログラムが完成するようになっています．

MERGE したままセーブすれば，次回からはそのまま実行できます．

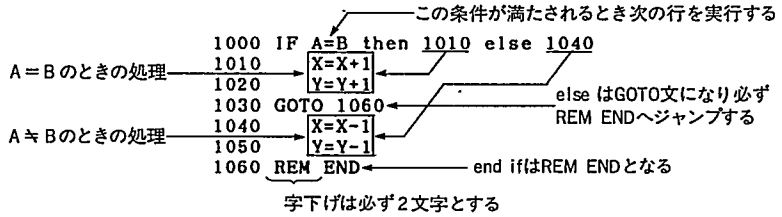
変数名規則さえ守っていれば MERGE したサブルーチンのモジュール変数は，全く意識しなくて構いません．

(2) GOTO 文と REM 文によるブロック IF 文

BASIC では，IF 文を複数行に渡って記述することができません．そのために，プログラムが分かりにくくなったり，他言語への書き換えが難しくなったりしています．そこで GOTO 文と REM 文を用いた，ブロック IF 文により記述します．

ブロック IF に文は次の条件があります．

①フォーマットは次の通りです



C言語での記述

```
if(a==b){
  x=x+1;
  y=y+1;
}
else{
  x=x-1;
  y=y-1;
}
```

FORTRANでの記述

```
IF(A.EQ.B)THEN
  X=X+1
  Y=Y+1
ELSE
  X=X-1
  Y=Y-1
ENDIF
```

② 原則としてブロックIF文の長さは，画面の1ページ(25行)以内とします．画面上でのデバッグ効率と，モジュール構造を分かりやすくするため，1ページ以内で理解できるようにするわけです．1ページを超えるときは，なるべく，サブルーチン化するようにします．

これによりGOTO文のジャンプ先も1画面で見られるため，デバッグしやすくなります．

③ GOTO文は，コメント行へジャンプするようにします．

コメント行以外は，追加削除など，修正が加えられる可能性が大きいためGOTO文はコメント行へジャンプするようにします．コメント行を記述したくないときは行番号の追加，削除のときに要注意です．

(3)変数名の命令規則

本サブルーチンパッケージでは，モジュール変数という概念を用いています．FORTRAN，C言語などでは，グローバル変数とローカル変数という概念があり，プログラム作成時に，ローカル変数として，他のモジュールの変数名を気にせずプログラムを作ることができます．しかし，BASICでは，グローバル変数しかなく，サブルーチンで，どのような変数を使用するのかを常に認識していなければなりません．そこで，モジュール変数という概念を用いて，他モジュールの変数を意識せずにプログラムを作成する方法を説明します．

● BASICでエラーとなる例

@リスト2-2入る

```
FOR I=1 TO 10
  GOSUB "GLINE"
NEXT
END
;
LABEL "GLINE"
  FOR I=1 TO IX
    PSET(IX,IY)
  NEXT
RETURN
```

同じ変数を使用すると
正常に動作しない！

①変数名の命名規則

BASIC独特の%，$，#などによる変数宣言は行わず，変数の開始文字でデータ型を区別します．また，一般変数，モジュール変数により，開始文字を区別します．

	一般変数	モジュール変数
整数型	I(Integer)	N(iNteger)
実数型	R(Real)	E(rEal)
倍精度型	D(Double)	A(dAburu)
文字型	C(Character)	H(cHaracter)

※doubleのOは0と誤り易いため
ローマ字記述のAを用います

1文字目をとる　2文字目をとる

表2－1

また，FORTRANなどへの移植を考えるのならば，変数名は6文字以内にする必要があります．本サブルーチンパッケージは，すべて6文字以内となっています．

さらに，BASICではA$とA%とA(1)は別変数として扱えますが，他言語ではこのような記述はできないため，使わないようにします．

また，C言語などへの移植を考えるならば，配列は0からとっておいたほうが良いでしょう．

つまり，すべてのメインプログラムの初めに

```
DEFINT I-N
DEFSNG R,E
DEFDBL D,A
DEFSTR C,H
OPTION BASE 0
```

と記述しておくわけです．

そして，プログラムを作成するときは，I,R,D,Cで始まるようにすればよいわけです．

②ラベル名の命令規則

本サブルーチンパッケージでは，ラベル名はGraphicのGをとって，必ずGで始まるようになっています．従って，メインルーチンではGで始まるラベルはなるべく使わないようにした方が良いでしょう．

(4)ラベル付きサブルーチンの使用法

実際に使用することを考えると，BASICのサブルーチンではなく，CALL文によるアセンブラプログラムの呼び出しにした方が高速で使い易いのですが，今回は，描画アルゴリズムの理解が，目的であるためサブルーチン形式としました．しかし，サブルーチンでは引数渡しができないため，命令規則に合ったモジュール変数に値を代入してから，サブルーチンを呼び出します．

このように記述しておけば，アセンブラ版のCALL命令ができれば，すぐに置き換えが可能です．また，引数渡しが可能な，他言語への移植も容易です．

点を描画するとき

BASICで

```
PSET( x,y,c )
```

と記述するものを

本サブルーチンパッケージでは

```
NX=x:NY=y:NC=c
GOSUB "GPSET"
```

という形態で表現します．

なお，FORTRANやBASICのCALL文では

```
CALL GPSET( x,y,c )
```

C言語では

gpset(x,y,c)

と記述します.

2-2-2　汎用グラフィックルーチンと高速グラフィックルーチン

よく高速グラフィックパッケージという言葉を見かけますが，はたして，高速グラフィックパッケージは良いのでしょうか．もし，全く同じ機能を持つのであれば高速の方が良いことはいうまでもありません．しかし，高速であるということは，なんらかの機能が欠けていると考えた方が良いでしょう．

マウスで，画面上の座標値を指示して，直線や，曲線を描くときは，ユーザー座標系などは必要ないため，ウィンドウ，ビューポート変換やクリッピングのロジックは全く必要ありません．しかし，CADなどで実数の座標値のモデルを表示するときには，拡大，縮小がスムーズにできなければいけないため，ユーザー座標系は必須の機能となっています．

このように用途に応じて必要な機能だけを取り出すことにより，高速化したものを，高速グラフィックルーチンと考えた方が良いでしょう．

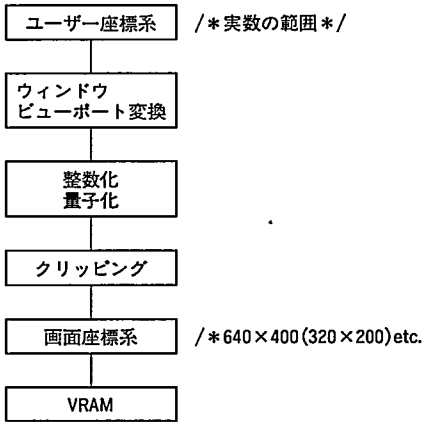


図2-1　汎用グラフィックパッケージの処理

2-2-3　グラフィックスモードの初期化処理

(1)解像度の設定

画面の解像度は，WIDTH文で設定します.

320×200　　WIDTH 40,25,0,1

640×200　　WIDTH 80,25,0,1

(2) V-RAM モードの設定

パレットモードか多色モードかの設定は，&H1FB0 で設定します．

多色モード(4096色×1面)　OUT &H1FB0,&H80
多色モード(64色×2面)　　OUT &H1FB0,&H90

(3)スクリーンモードの設定

スクリーンについての初期設定をします．

```
4096色／64色(バンク0)　OPTION SCREEN 0
                       SCREEN 0, 0
64色(バンク1)          OPTION SCREEN 0
                       SCREEN 2, 2
```

2-2-4 パレットの初期化

カラーパレットの初期化コマンドは，用意されていないため，プログラムによる初期化が必要です．このパレットはリセットしても初期化されないので，注意しなければなりません．

電源 ON の時は，4096色モードとなっているため64色2画面モードにするときは，プログラムで初期化をするしかありません．そこで，64色2画面モード時のパレット初期化ルーチンと4096色同時表示時のパレット初期化ルーチンのプログラムを掲載します．

プログラムの説明をしますと，OUT &H1FB0,&H80 で，多色モードに設定していますが，これは多色モードでないと，パレットの初期化ができないためです．パレットの設定を行うときは，OUT &H1FC5,&H80 を実行してから設定しなければなりません．また，パレットの設定が終了したら，OUT &H1FC5,&H0 を実行しなければなりません．4096色の時は B,R,G の順番にカラーコードと輝度を1つずつ上げていき，16×16×16＝4096色分のカラーコードを設定します．

それに対して，64色2面モードのときは，4×4×4＝64を2回実行すればよいので，128色分のカラーコードを設定するだけでよいのです．

64色2面モードのときに，4096色分のカラーコードを初期化しないのは，おかしいと考えがちですが，64色2面モードのときは，片方のプレーンをマスクして2ビットしか見にいかないため，128色分の初期化で十分なのです．

2-2-5 BASIC で 4096 色同時表示を行う方法

X1turboZ は，4096色同時表示モードをサポートしていますが，BASIC の LINE コマンドや PSET コマンドで，カラーコード4095などと記述すると，Illegal function call というメッセージが出力されてしまいます．Z's STAFF で作成した画面を BASIC 上で表示することはできても，その上に絵を描くことはできません．そこで BASIC で4096色同時表示を可能にするサブルーチンを作成しました．

サブルーチンには，BASIC のグラフィックコマンドを使用した例と，VRAM を直接操作した

例の2つのサブルーチンがあります．前者は比較的高速(といっても一般のBASICより4倍遅い)で，ステップ数も短く理解しやすいようになっています．後者は，非常に遅く，ステップ数も多いのですが，VRAMを直接操作したり，アセンブラ命令と対応が取れるようになっているため，アセンブラ化すれば非常に高速になります．また，BASICの内部処理も理解できるようになっています．(但し，X1turboのBASICが，これと同じアルゴリズムであるかどうかは，確認していません)

このアルゴリズムでは，まず，SCREEN文で，SCREEN 3から描画していきます．SCREEN 3は，4096色モードの場合，最も，輝度の低いビットが，割り当てられています．そこで，NCで与えられたカラーコードの1ビット目を取り出します．取り出したビットが，BLUEならそのまま，REDなら2倍，GREENなら4倍すると，パレットモードのカラーコード(NCOL)になります．

	G	R	B
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

図2-11　パレットモードのカラーコード(初期値)

SCREEN 2は4096色モードの場合，2番目に輝度の低いビットが割り当てられています．そこで(NC AND 2)／2の演算を実行することにより，2ビット目を取り出します．パレットモードへのカラーコードの変換は，SCREEN 3と同じです．

SCREEN 1,SCREEN 0についても，SCREEN 2の2を4，8に変更するだけで全く同じように変換できます．

このようにして，長方形や，円や，円弧なども，4096色モードで描画することができます．当然ビデオ入力した画像の上に重ねて描画することもできるようになります．

しかし，LINEを1本描画するために，このようなコーディングをするのは大変なので，サブルーチンとしておき，座標値と色コードだけ設定するようにすれば使いやすいでしょう．

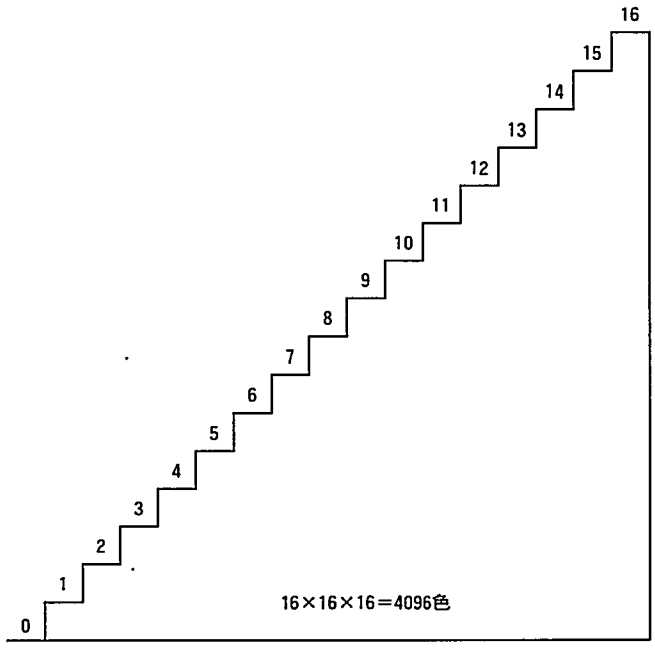


図2-12 4096色の輝度レベル

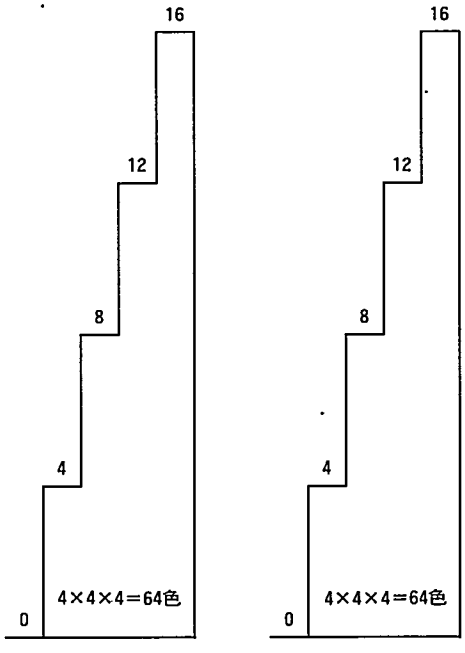


図2-13 64色2面モードのときの輝度レベル