岡本安晴　2001.2.1；2001.8

関数の立体図[1]

数値計算では関数の積分値や微分方程式の解を求めたり、あるいは関数の根や最小値（極小値）を求めたりというように関数についての計算が行われる。このとき、関数の変化の様子をビジュアルに表示してみると、その関数の大局的な性質を直感的に把握することができる。２変数の関数の場合、ビジュアルな表示方法として立体図あるいは等高線図がよく用いられる。ここでは、立体図の描き方について説明する。等高線図は別に説明する。

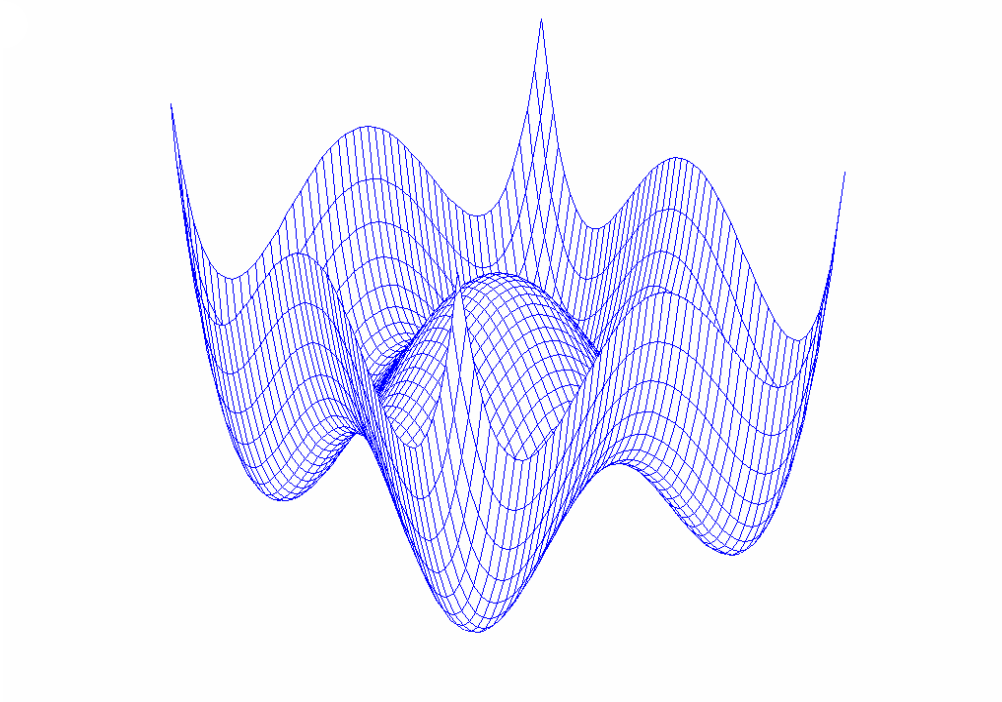

図１　曲面の立体図

立体図の例

図１は関数 $z = f(x,y) = (x/2)^4 + (y/2)^4 - 2((x/2)^2 + (y/2)^2)$ の立体図である。このように、２変数の関数は、その曲面の立体図を描くと、２変数 x と y の値の変化に伴う関数値の変化の様子が良くわかる。

図１の立体図は、手続き DrawPerspec を

---

[1]本解説は、TRY！PC、2000 年３月号「Delphi による数値計算プログラミングのすすめ：第１回関数の立体図を描く」の原稿をもとにしたものである。

```
DrawPerspec( CheckF, -3.0, 3.0, -3.0, 3.0 );
```

のように呼出して描いたものである。

手続き DrawPerspec の第１パラメータには、立体図を描く関数を指定する。上の場合、CheckF が指定されている。この関数は、以下のように宣言されています。

```
function CheckF( x, y : Extended ) : Extended;
  begin
    CheckF:=sqr(sqr(0.5*x))+sqr(sqr(0.5*y))-2*(sqr(0.5*x)+sqr(0.5*y));
  end;
```

第２～５パラメータの－3.0、・・・、3.0 は、変数ｘとｙの変域を指定するものである。

手続き DrawPerspec はユニットファイル UMyPerspec1.pas に宣言されている。

図１は、このユニット UMyPerspec1 を用いたプロジェクト PCkPerspec.dpr の実行によって描かれている。手続き DrawPespec は、このプロジェクトのユニットファイル UCkPerspec.pas（リスト１）に示されているように、簡単に使用できる。

リスト１ ユニット UCkPerspec の一部

```
uses
  UMyPerspec1;       //  立体図を描くためのユニット

procedure TForm1.ExitButtonClick(Sender: TObject);
begin
            Close;
end;

function CheckF( x, y : Extended ) : Extended; //  立体図を描く曲面の関数
  begin
        CheckF:=sqr(sqr(0.5*x))+sqr(sqr(0.5*y))-2*(sqr(0.5*x)+sqr(0.5*y));
  end;

procedure TForm1.GOButtonClick(Sender: TObject);
begin
         ExitButton.SetFocus;  UpDate;

         DrawPerspec( CheckF, -3.0, 3.0, -3.0, 3.0 );
end;
```

立体図の描き方

ユニット UMyPerspec1 **では、透視図の画面を図２のように設定して立体図を描いている。ここでの説明は森**[1]（1974、**第** 4.2 **節）に従っているが、直方体と透視図の関係を統一する、角度はすべて正の値で表わす、などの変更を行っている。**

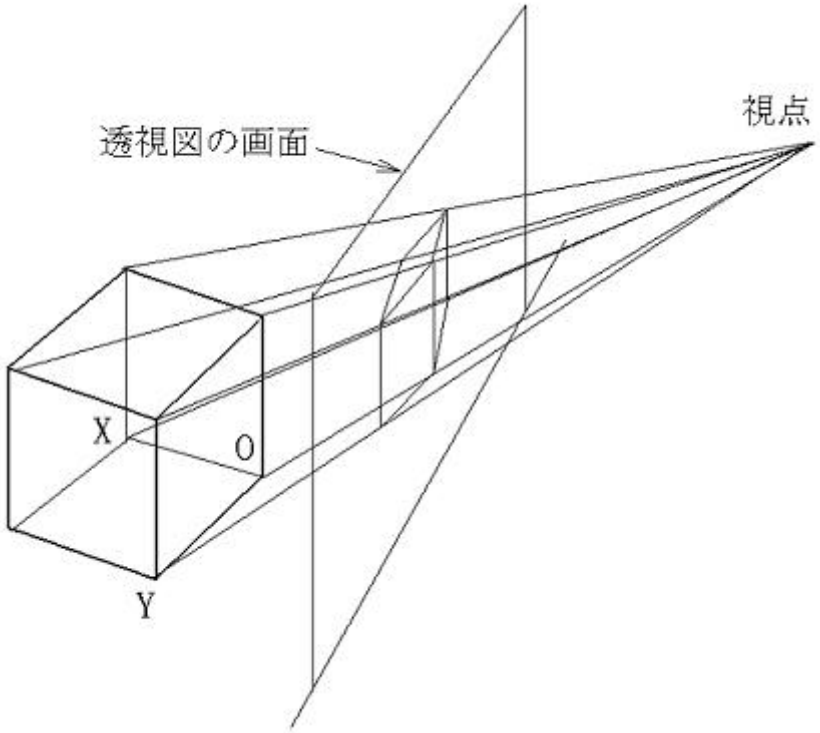

図２　立体図を透視図で描く

X **軸と** Y **軸を直方体の辺** OX **と** OY **にとり、関数の曲面が直方体内に納まるように設定して透視図の画面に立体図を描く。２変数** x **と** y **の変域が** $[a,b]$ **と** $[c,d]$ **のとき、一次式による変数変換によって、それぞれの変域が** $[0,X_w]$ **と** $[0,Y_w]$ **であるようにできる。また、値域が** $[u,v]$ **のきは、一次式による変換によって値域が** $[0,Z_h]$ **であるようにできる。これらの変換によって、変域が長方形である関数の曲面は、大きさが** $X_w \times Y_w \times Z_h$ **の直方体に納めることができる（図３）。**

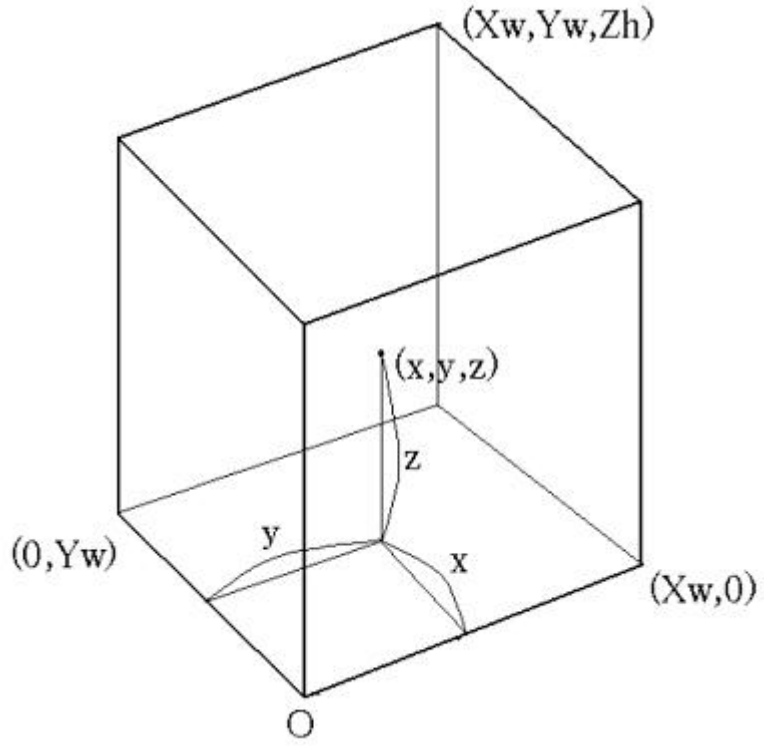

図３ 関数を納める直方体

直方体の内部の点(x, y, z)が、透視図の画面にどのように投影されるか調べる。
直方体を透視画面に対して図４のように傾ける。

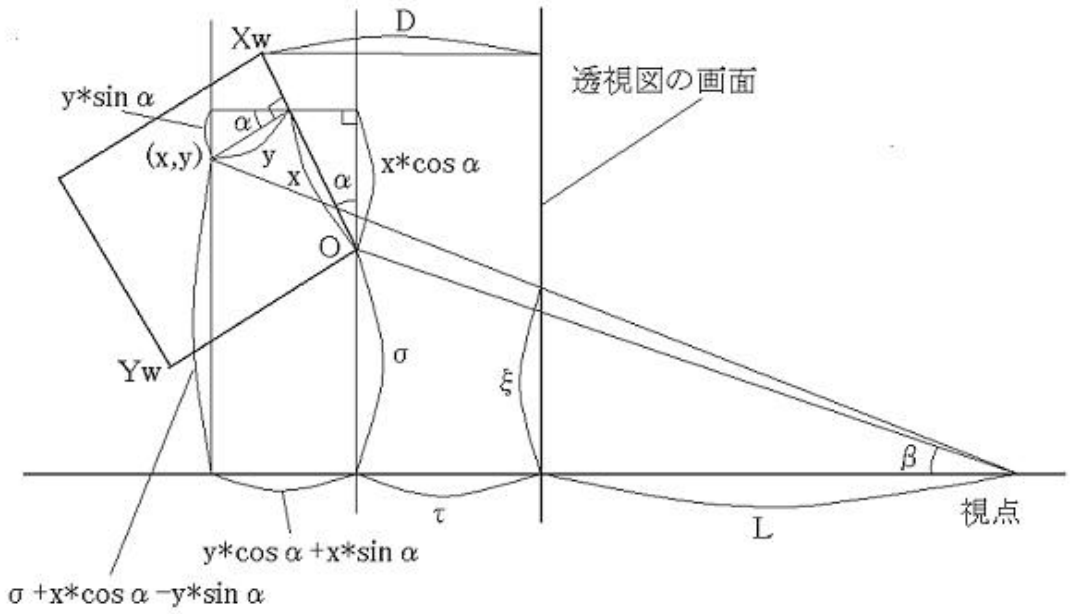

図４ 透視図の画面と点(x, y)の位置

図４は、真上（Z 軸方向）から見た図として、ＸＹ平面上の２次元で直方体と透視画面の関係を表わすものである。直方体の一辺 OX と透視画面との角度が a であるように直方体を置く。点 (x, y, z) は XY 平面状では点 (x, y) として表わされている。

点 (x, y) の透視図の画面上における水平方向の位置は、視点と点 (x, y, z) を結ぶ線が透視図の画面と交わる点として、視点から画面へ下ろした垂線の足からの距離 x で表わされている。x は、以下のように求めることができる。

視点から透視図の画面へ下ろした垂線と点 O との距離を s 、垂線の長さを L、点 O と透視図の画面との距離を t 、点 $(X_w, 0)$ と透視図の画面との距離を D とおく（図４）。直方体は透視図の画面に対して a だけ半時計周りの方向に回転しているものとし、視点から透視図の画面に下ろした垂線と視点と点 O を結ぶ直線のなす角を b とおく。このとき、図４において、視点、点 (x, y)、および視点から透視図の画面へ下ろした垂線へ点 (x, y) から下ろした垂線の足の３点で作られる直角三角形に注目して次式を得る。

$$\frac{x}{s + x\cos a - y\sin a} = \frac{L}{y\cos a + x\sin a + t + L}$$

また、

$$\begin{aligned} t &= D - X_w \sin a \\ s &= (t + L)\tan b \end{aligned} \qquad (1)$$

が成り立つ。

したがって、

$$\begin{aligned} x &= \frac{s + x\cos a - y\sin a}{y\cos a + x\sin a + t + L} \cdot L \\ &= \frac{x\cos a - y\sin a + (D - X_w \sin a + L)\tan b}{y\cos a + x\sin a + D - X_w \sin a + L} \cdot L \end{aligned} \qquad (2)$$

となる。

次に、透視図の画面上の垂直方向について考える。

図５は、直方体を真横、透視図の画面と平行な方向から見たものである。視点と点 (x, y, z) を結ぶ直線が透視図の画面と交わる点の高さを視点の高さから下方向への距離 h で表わす。h の符号は、視線が下方向を向いているので負とする。

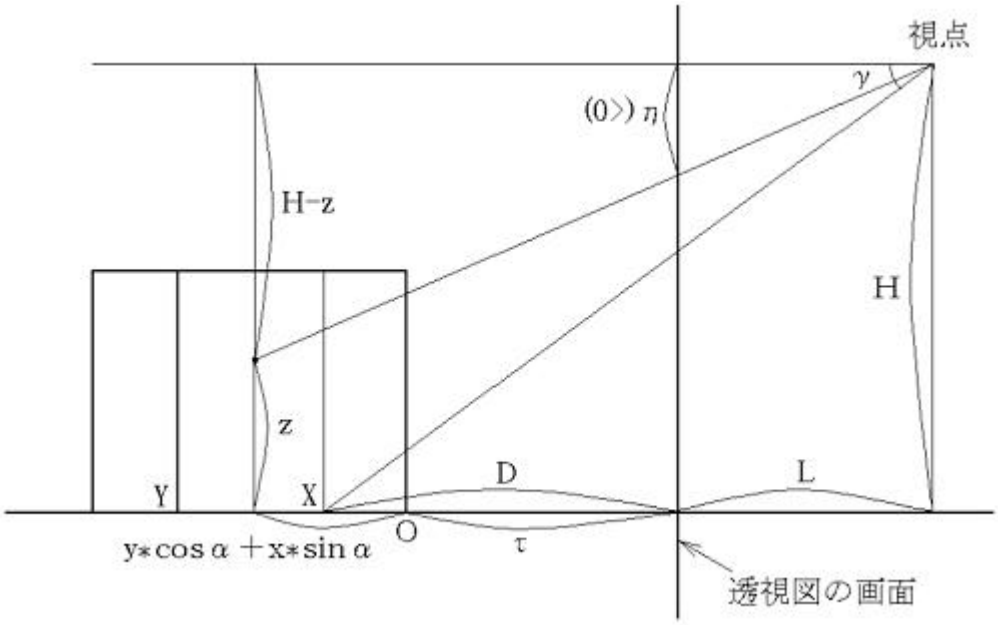

図５ 透視図の画面上での高さ

図５において、直方体の点$(X_w,0,0)$を視線が見下ろす角度をg とし、視点の直方体の置かれている平面からの高さを H とする。このとき、点(x,y,z)と視点、および視点から水平に引いた直線と点(x,y,z)を通る垂直線との交点の３点で作られる直角三角形（真横から見た様子を表わす図５におけるもの）に注目して次式を得る。

$$\frac{-h}{H-z}=\frac{L}{y\cos a+x\sin a+t+L} \qquad (3)$$

また、視点、視点から水平面に下ろした垂線の足、および視点から角度g で見下ろす直線と水平面との交点の３点で作られる直角三角形に注目して次式を得る。

$$\tan g=\frac{H}{D+L} \qquad (4)$$

式（１）、（３）、（４）より次式が導かれる。

$$h=\frac{z-(D+L)\tan g}{y\cos a+x\sin a+D-X_w\sin a+L}\cdot L \qquad (5)$$

曲面上の点(x,y,z)の透視図の画面上の位置は、式（２）および（５）で与えられるにxおよびhによって点(x,h)と表わされる。曲面の透視図は、点(x,y,z)が曲面上を動くときの点(x,h)の軌跡を描くことによって得られる。

隠線の処理

曲面の透視図では、曲面の後ろに隠れて見えない部分（隠線）は描かないようにする必要がある。

曲面の立体図は、点(x,y)を格子点(x_i,y_j)にとって、格子点上の曲面の点$(x_i,y_j,f(x_i,y_j))$を線分で結んでできる網目を透視図の画面上に描くことにより作成する。格子点上の関数値を描画前に計算しておき、描画のときはこの計算済みの格子点での値を使うことにより、曲面の関数の計算回数を減らすことができる。

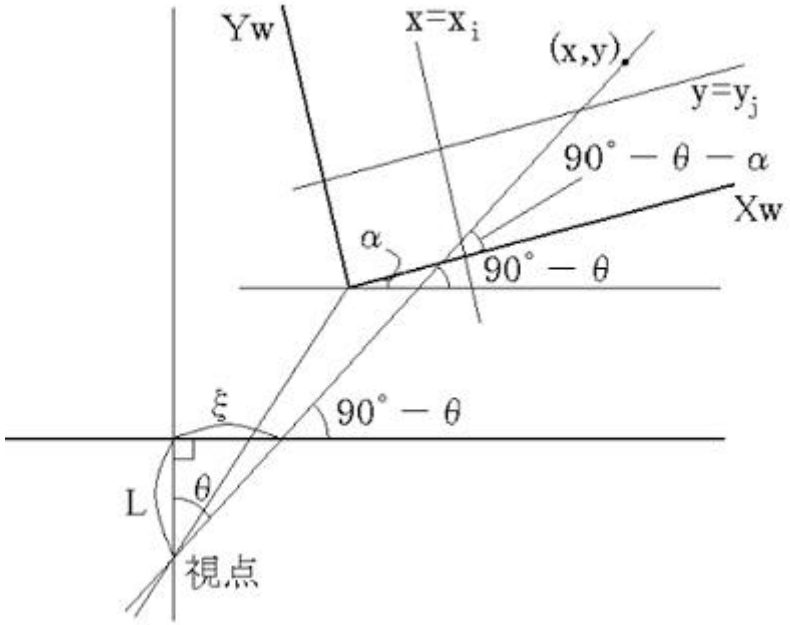

図６　隠線処理のための視点と点(x,y)を結ぶ直線

したがって、隠線のチェックは、格子点上の関数値を結んでできる網目と、視点と点(x,y,z)を結ぶ直線（視線）との比較により行う。視点と点(x,y,z)を結ぶ直線の水平面上への射影（図６）、すなわち視点からの垂直線の足と点(x,y)を結ぶ直線が、点$(0,0)$と点$(X_w,0)$を結ぶ軸をX軸、点$(0,0)$と点$(0,Y_w)$を結ぶ軸をY軸とする平面上の、$x=x_i$、$y=y_j$によって構成される格子と交わる点での関数値（正確には、格子点を線分で結んで一次式によって求めた値）と視線との関係を調べる。

点(x,y)を通る直線の傾きT_xは、直線が視点から透視図の画面へ下ろした垂線となす角をqとするとき、図６より次式で与えられることがわかる。

$$T_x=\tan(90°-q-a)$$
$$=\frac{\cos a-\tan q\sin a}{\sin a+\tan q\cos a}$$

図６より

$$\tan q = \frac{x}{L}$$

なので、

$$T_x = \frac{\cos a - \frac{x}{L}\sin a}{\sin a + \frac{x}{L}\cos a}$$

$$= \frac{L\cos a - x\sin a}{L\sin a + x\cos a}$$

となる。

点(x, y)を通る傾きT_xの直線は、次式で与えられる。

$$(Y - y) = T_x(X - x)$$

従って、格子

$$X = x_i \qquad \text{および} \qquad Y = y_j$$

との交点の座標Y_iおよびX_jは、それぞれ次式で与えられる。

$$Y_i = T_x(x_i - x) + y \qquad \text{および} \qquad X_j = (y_j - y)/T_x + x$$

隠線の判定は、点(x, y)より透視図の画面の方にある点(x_i, Y_i)および(X_j, y_j)での関数値を調べて行う。プログラムでは、関数値は格子点(x_i, y_j)での値が配列に保持されているが、点(x_i, Y_i)あるいは(X_j, y_j)が格子点(x_i, y_j)でない場合は隣接する格子点での関数値から 1 次補間によって関数値を求めている。点(x_i, Y_i)あるいは(X_j, y_j)での関数値がすべて点(x, y)での関数値より小さいか、あるいはすべて大きいとき、点(x, y)での曲面すなわち点$(x, y, f(x, y))$は視点から見えている。しかし、点(x_i, Y_i)あるいは(X_j, y_j)での関数値が、点(x, y)での関数値より小さいものと大きいものの両方があるとき、点$(x, y, f(x, y))$はその前の曲面に隠れて見えない。

立体図描画用ユニット UMyPerspec1

以上の方法によって関数の曲面の立体図を図１のように描くためのユニットがUMyPerspec1である。このユニットのフォームは、図７のように用意されている。

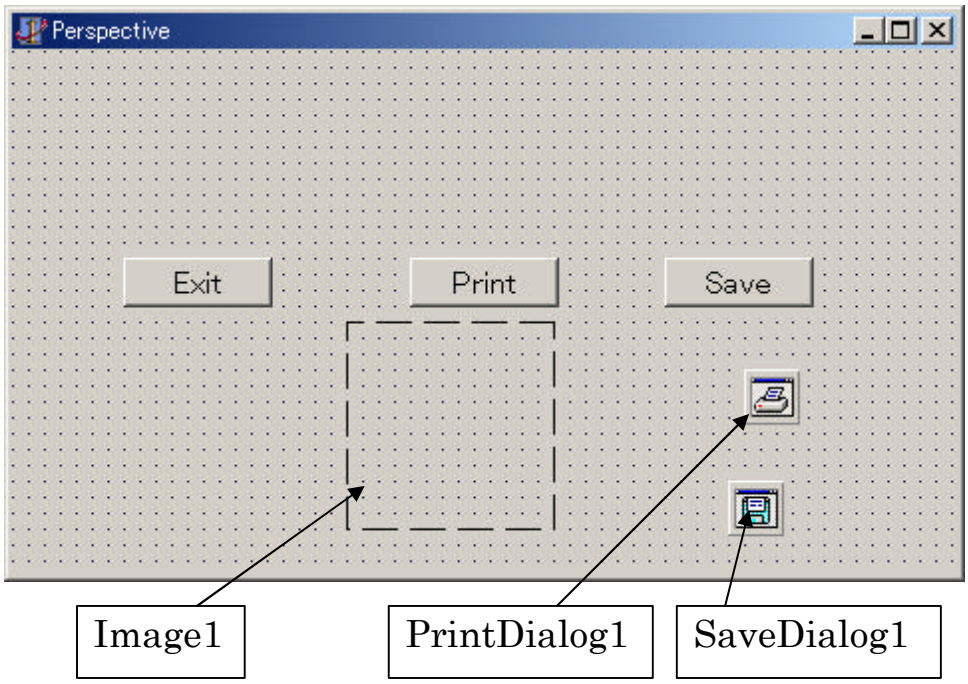

図７ ユニット UMyPerspec1 のフォーム

UMyPerspec1 において宣言されている手続き DrawPerspec を、関数と変域を指定して呼出すと、描画用のフォームが生成、表示される（リスト２）。

リスト２ 手続き DrawPerspec の宣言の始めの部分。

```
procedure DrawPerspec( CheckF : TFunc;  //  立体図を描く関数
                       pxmin, pxmax,    //  xの変域（最小値と最大値）
                       pymin, pymax     //  yの変域（最小値と最大値）
                              : Extended );
var  i, j : Longint;
begin
    //   変数xとyの変域の設定
    MinX:=pxmin; MaxX:=pxmax;
    MinY:=pymin; MaxY:=pymax;

    FPerspec:=TFPerspec.Create(Application); //  フォームの生成

    with FPerspec do
      begin
        //   フォームの準備
        Visible:=True;
        UpDate;
        WindowState:=wsMaximized;
        UpDate;
```

```
with ExitButton do
  begin  Top:=0; Left:=0;  end;
with SButton do
  begin  Top:=0; Left:=ExitButton.Width;  end;
with PButton do
  begin  Top:=0; Left:=ExitButton.Width+SButton.Width; end;
UpDate;
```

このフォームは最大化されて、その左上に「Exit」、「Save」、「Print」の３つのボタンが表示される（図８）。

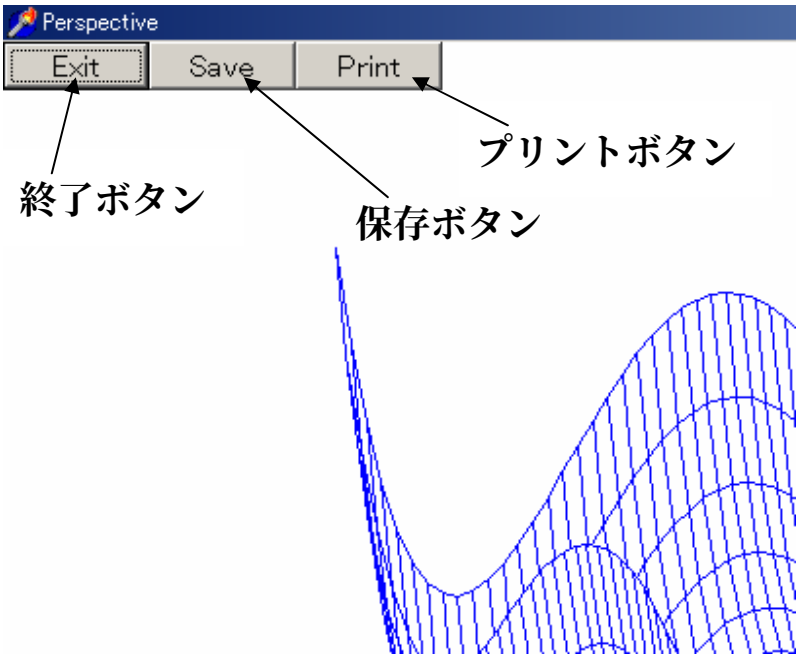

図８　描画終了時の画面

Exit ボタンをクリックすると描画画面が閉じられる。

Save ボタンをクリックすると、描画図形が*.bmp ファイルとして保存される。描画図形は、Image コンポーネントの Canvas に描かれてから、Picture プロパティの SaveToFile メソッドによって保存される。保存先のファイル名の設定は、SaveDialog1 コンポーネントによって行っている。保存したファイルは、お絵描きソフトで読み込んだり、ワープロの文書に貼り付けたりして利用できる。

Print ボタンをクリックすると、立体図がプリンタに出力される。立体図のプリンタへの出力は、Save ボタンのクリックで保存した*.bmp ファイルをワープロの文書に貼り付けて出力することもできるが、この Print ボタンのクリックによる方が、プリンタの Canvas

の解像度で印刷されるのできれいな出力が得られる。

プリンターへの出力は、Printer オブジェクトの BeginDoc メソッドの実行後、Printer の Canvas への描画によって行う。描画終了後、EndDoc を実行する。Printer オブジェクトを利用するときは、Printers ユニットを uses 節に宣言しておく。

ユニット UMyPerspec1 の使用例であるユニットファイル UCkPerpec.pas（リスト１参照）のフォームは、図９のように用意されている。

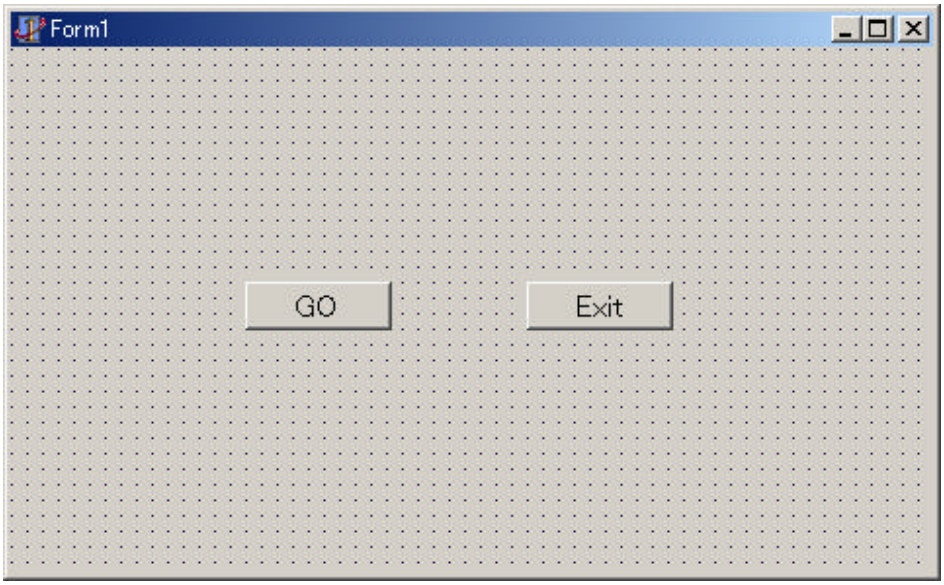

図９　リスト２のユニット UCkPerspec のフォーム

GO ボタンのクリックで手続き GOButtonClick が実行され、手続き DrawPerspec によって関数 CheckF の立体図（図１）が描かれる。

参 考 文 献

（１）森 正武　「曲線と曲面」 Pp.150、 教育出版株式会社、1974.
（２）岡本安晴 「Delphi プログラミング入門」 Pp.207、CQ 出版株式会社、1997.