コンピュータグラフィックス（３Ｄ図形作成）　澤見英男

１．　はじめに

Ｍａｐｌｅは数式処理，数値計算，グラフィックスの機能を備えているが，この資料ではこの中から主としてグラフィックス機能を紹介する。実際に確かめる場合は，配布された資料と担当者の説明をもとにして，各自で考察しながら実験室にて試してみるのも良いでしょう。その際に得られた結果は，その時には良く憶えているつもりでも，後になって来ると思い出せないことがあります。実験室で試したこのと経過や結果の要点などを，忘れないうちに資料やノートなどに記録しておくことをお勧めします。

２．　２Ｄ図形作成について

ではこれからＭａｐｌｅを用いた２Ｄ図形作成について試してみましょう。まず，Ｍａｐｌｅの使える実験室では担当者の指示に従ってパーソナルコンピュータの操作を行ってください。パーソナルコンピュータ本体の電源投入ボタンを押すことにより電源が入ります。しばらくするとログオン画面が表示されますので，アカウント情報（ユーザ名とパスワード）を入力してください。アカウント入力操作を経由しない限り目の前の（ローカル）パーソナルコンピュータはもちろん，（リモート）プリンタやファイルサービスなども使えませんので正しいアカウント情報を入力してください。電源を切らずにコンピュータの利用者が入れ替わる場合には，スタートボタンをマウスで指示し左ボタンのクリックでメニュー表示してログオフを選び，一度ログオフしてから今度は別の利用者のアカウント情報を入力しログオンし直すようにしてください。通常の終了では，スタートボタンをマウスで指示し左ボタンのクリックでメニュー表示してウインドウの終了を選び，電源を切るを選んでください。システム側で必要な処理を行った後に自動で電源が切れます。この際に処理途中のアプリケーションが在ります云々のメッセージが表示されて処理が進まなくなった場合には，画面の案内に従って適切な対応をしてください。

さあ準備は良いですか，ではログインの作業を完了し見慣れたデスクトップ画面が現れたらＭａｐｌｅを起動しましょう。マウスを操作して，スタートボタン，プログラムと順次辿って行き，Ｍａｐｌｅを選ぶと，次のような画面が現れます。この画面内の＞プロンプトが表示されている部分にＭａｐｌｅプログラム文を順次入力していきます。ところで，Ｍａｐｌｅのバージョンによって表示画面の差異が在りますが，基本的な操作に関しては同じなので気にかける必要はありません。

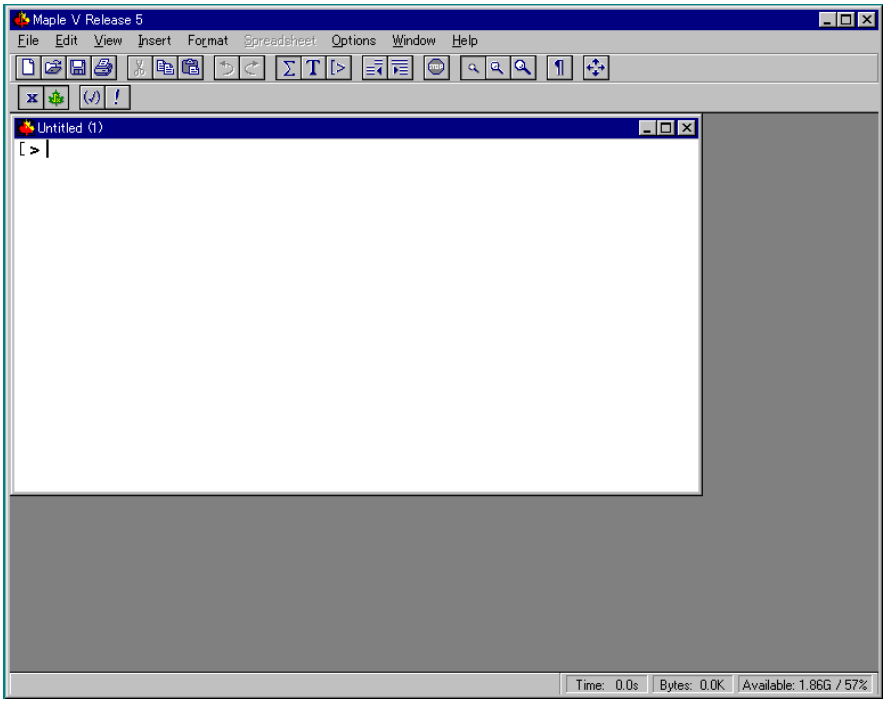

図１　プロンプト

プログラム文の終了はセミコロン（；　言い換えればの意味で用いることが多い）で示すようになっています。そして，例えば以下のようにして，一行に複数個のプログラム文

を書き表すことができます。早速試してみることにしましょう。

```
>4!;  1*2*3*4;    sin(Pi/4);  (1/2)^(1/2);
```

最初のプログラム文４！は４の階乗を，次はこれを明示的に表したもの，そして三角関数および平方根の計算と三つのプログラム文を続けて表しています。そして，リターン入力と同時にそれぞれについての記号処理による計算が順次行なわれます。ところで，変数Ｐｉは円周率πを表す予約語の一つです。ところでプログラム文の終了をコロン(：　すなわちの意味で用いることが多い)で示した場合には，計算処理は行ったとしても結果の表示は行われません。そのことから，中間結果を求める処理をこのコロンを用いて書き表すと，表示結果が随分と整理されてすっきりしたものになります。

次は，皆さんよくご存じの関数を描いてみましょう。例えば，以下のように書き表すことにより，三角関数(sin)を表示するためのグラフィックス処理を行うことができます。

```
>plot( sin(x), x=0..Pi);
```

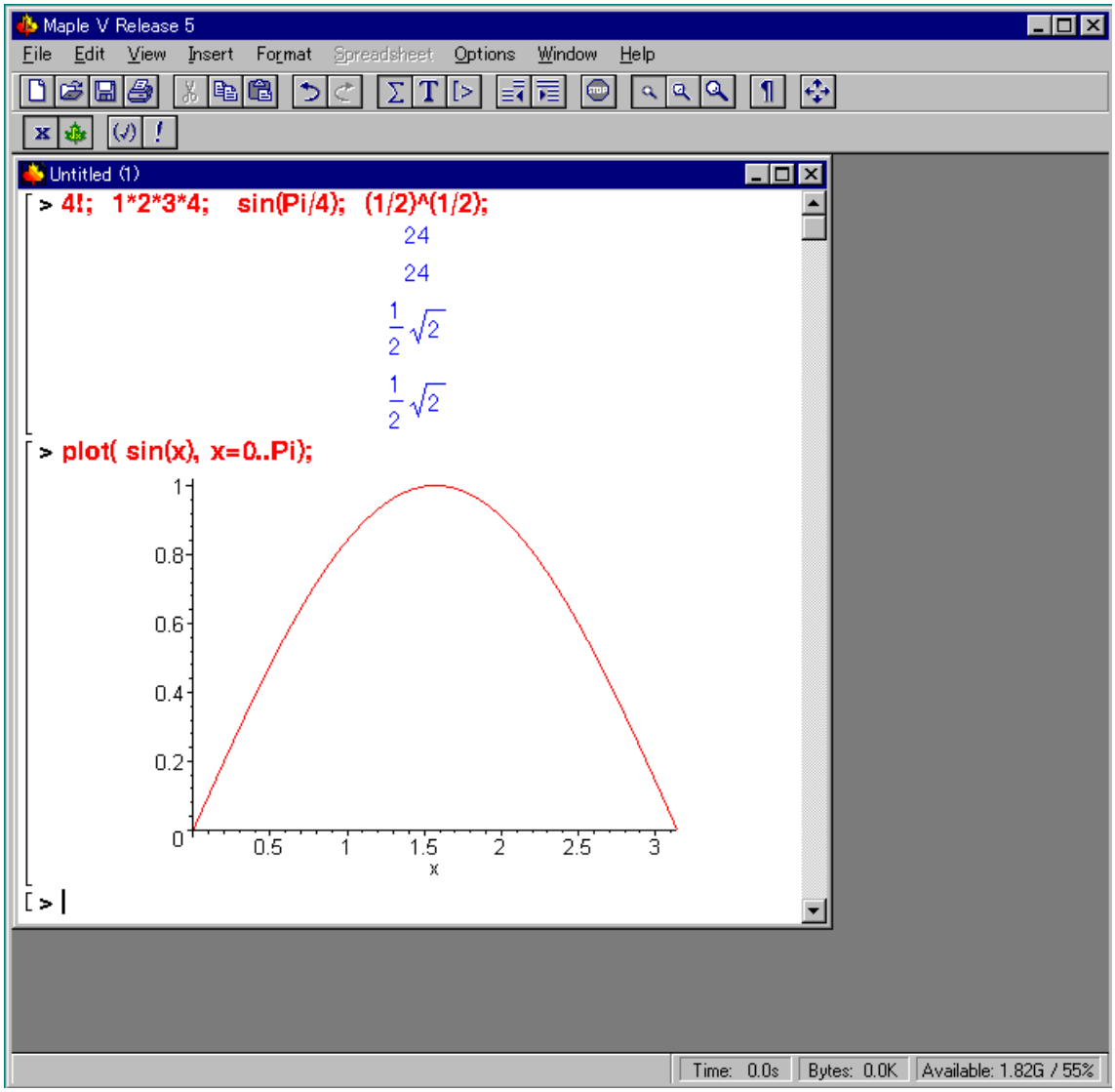

図２　正弦関数の描画（範囲は０からπまで）

このウィンドウ内の上辺にあるメニューバーやその下に置かれているツールバーの中から，必要に応じた機能を選び出し利用することができます。メニューバーやツールバーの表示内容は処理内容に対応して変化するので注意してください。よく使う機能についてはツールバー内にあるアイコン（絵文字）を経由しても利用できます。また，メニューバーをマウスの左ボタンでクリックするとプルダウンメニューが表示されるので，この中から

必要な機能を選んで利用しても良いでしょう。そしてこのメニューは，段瀑(cascade，カスケード)のように項目を選ぶ毎にさらに新しいメニューが現れるようになっているので，項目全てを見なくても目的の項目まで辿り着けるようになっています。このようなメニューは，各種キイワードを階層化して整理し憶えておくと便利に利用することができるものです。また何をどう行えばどんな結果が出るのかといったことを順序だって考えるような習慣を着けておくと，メニューを使う上での効率が上がってきますので心がけてください。

マウスで Help を指示すると次の図のようなプルダウンメニューが表示されます。この中にある Introduction はいわゆるオンラインヘルプであり，この中に諸君らが必要とする資料の殆どがここに網羅されています。説明は英文によるものが多いみたいですが，キイワードとそのキイワードを含む概念や，逆にそのキイワードに付随する概念などの階層構造を思い浮かべながら読み進んでいけば，特に問題なく必要な項目まで辿り着けるはずです。また，順次解説して行きますが，このオンラインヘルプの中には例題として多くのプログラムを記載していることから，これをコピーしてプログラム内にペーストし実行させることにより，比較的簡単にＭａｐｌｅの使い方をマスターできるようになっています。ところで，対象としている部分をマウスの右ボタンでクリックすると，その対象物に関する処理内容に応じたメニューが表示されことから，メニューバーやその下に表示されるツールバーに表示されている盛りだくさんの項目から必要なものを探し出して選ぶよりは随分と効率的です。操作に慣れてきたらどちらも大いに利用しましょう。

まず２次元座標上に関数を描くことから始めて，グラフィックスを主としたＭａｐｌｅの使い方を理解していくことにしましょう。資料としては，先に述べたオンラインヘルプを用いることにします。マウスを使ってＨｅｌｐを指示し左ボタンをクリックして次に示す図のようにプルダウンメニューを表示させ，このヘルプ・メニューから段瀑(cascade)を滝下りのように順次辿るような感じで必要な項目を探してみましょう。

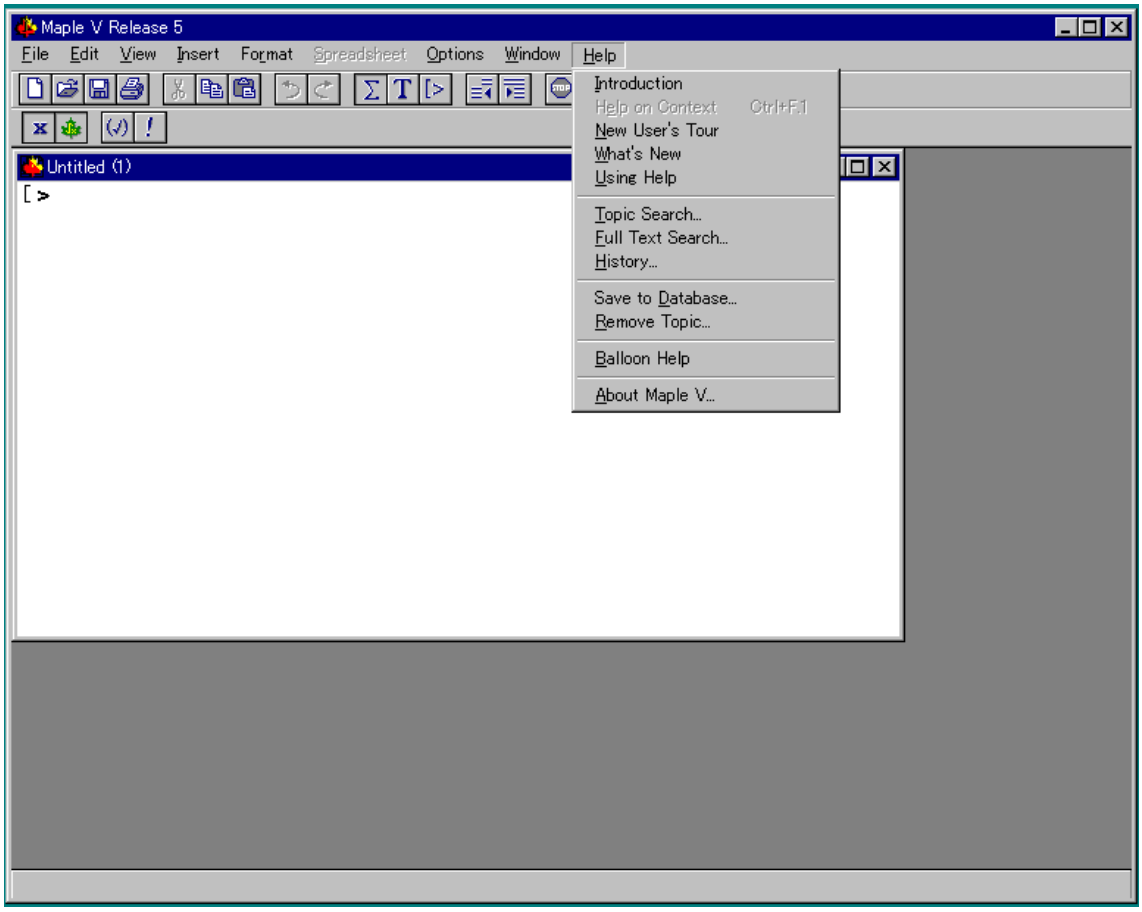

図３　ヘルプメニュー（英文の場合）

このメニューの中から Introduction を選び，以後順次 Graphics，2D，plot と辿って行ってみましょう。そうすると次の図に示したように，先程例示した plot についての説明が表示されます。この表示画面の右側にあるスクロールバーを操作して読み進んでいくと，

詳しい説明と各種サンプルが記載されているので，このサンプルをプログラム作成画面へコピー・ペーストして試すことができます。説明が英文でも，気にすることはないでしょう。キィワードに注目して読めば直ぐに意味が分かると思います。また，サンプル部分は目立つように赤色で表示されていますので，このサンプル部分の赤色の表示が現れるまでスクロールダウンして見るのが手っ取り早くて良い方法かもしれません。

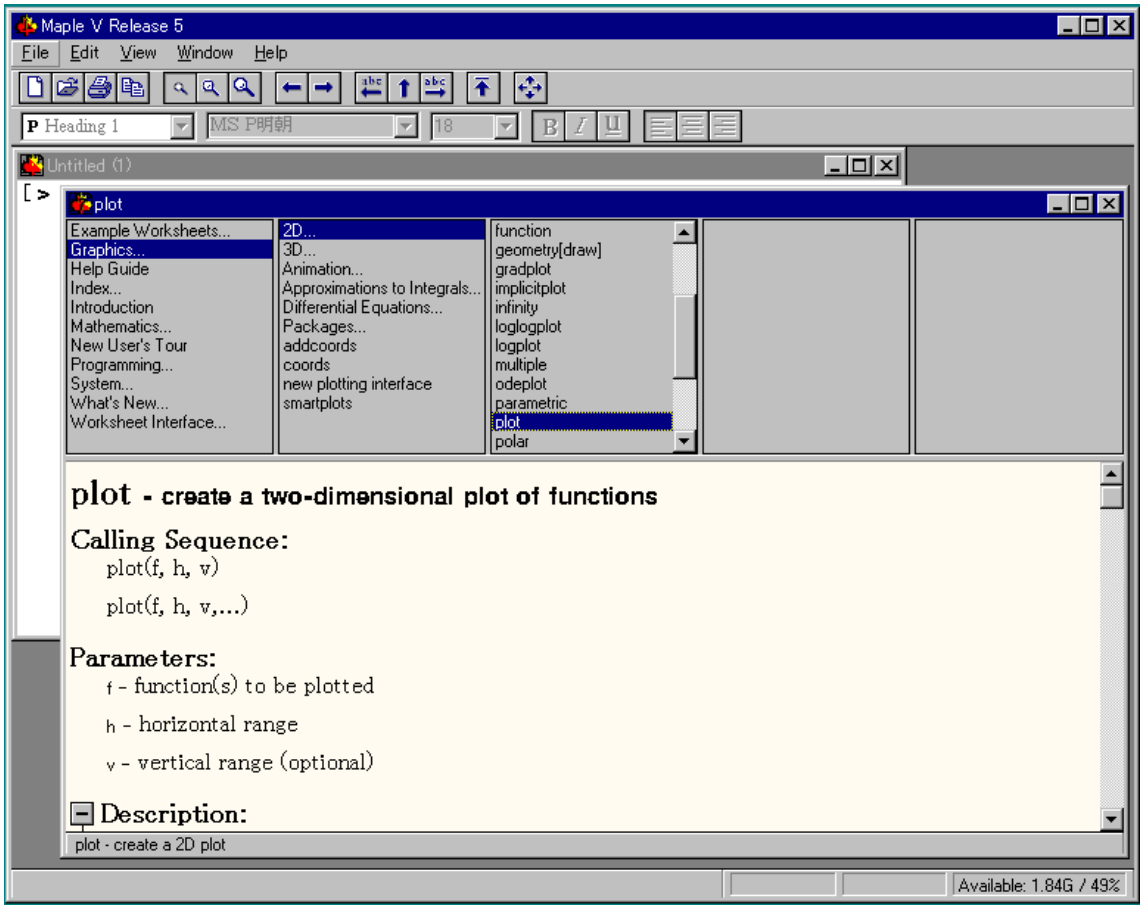

図４　ヘルプメニューから辿ってプログラム例を探す（英文の場合）

上記ヘルプ画面の垂直スクロールバーを操作して読み進んでいくと，赤色の文字で表示されている Examples(例題)に辿り着けます。何個かのプログラム文が数行にわたって書かれているものと思います。この中から以下に示す様な例題をコピー・ペーストして実行すると，画面に関数が図示されます。どうです，随分と簡単だったでしょう。

```
>plot(cos(x)+sin(x), x=0..Pi);
```

このプログラム文では，三角関数　cos(x)+sin(x)　を範囲ゼロからπまでについて描画表示(plot)することを指定しています。

座標軸として，Ｘ軸とＹ軸とに目盛を付けた通常(Normal)のものが標準として書き加えられます。これの変更は，表示したグラフをマウスで左クリックしてから範囲を指定し，メニューバーの中から Axes(軸)項をプルダウンメニュー表示させると Boxed(箱形)，Framed(枠付)，Normal(通常)，None(軸無し)の中から選べるようになっています。別の方法として，表示したグラフをマウスで右クリックしてメニュー表示をし，この中から順次辿ることにより同様のことを行うこともできますし，メニューバーの下に表示されるツールバー中のアイコンをマウスで指示してクリックし表示方法を選ぶこともできます。

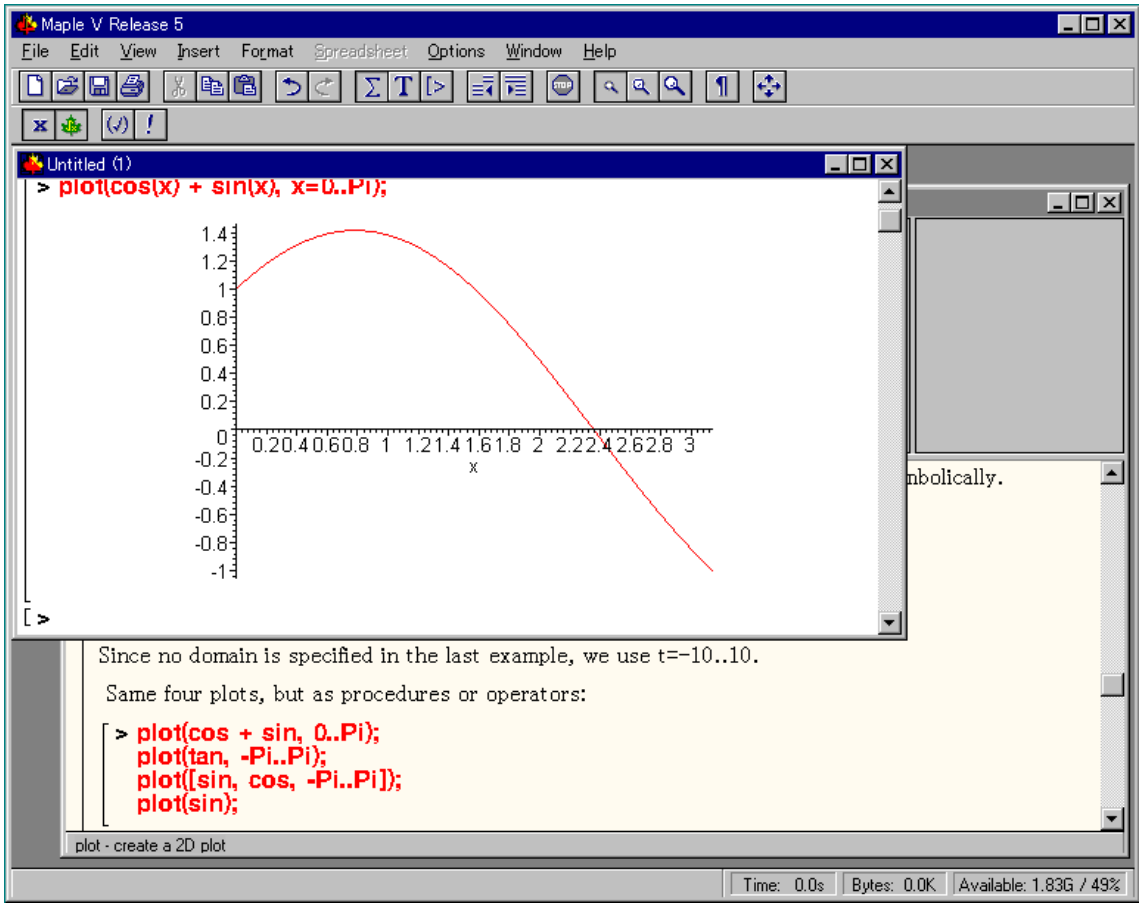

図５　サンプルをコピー・ペーストして実行

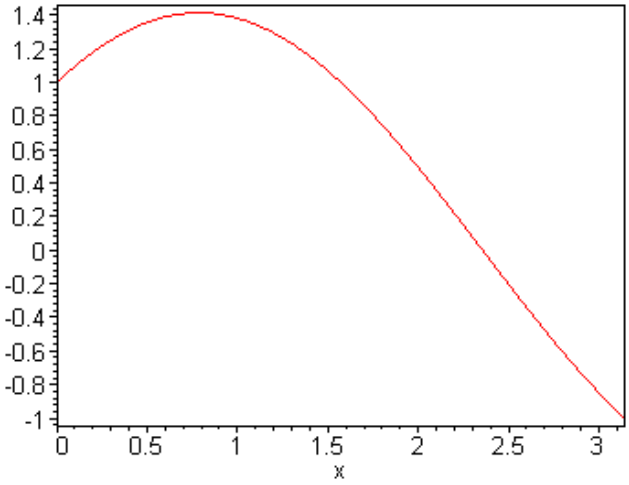

図６（a）Boxed
（箱形）

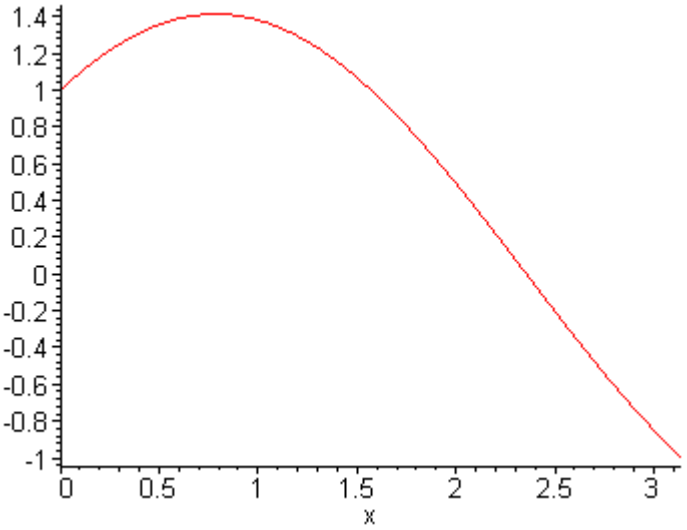

図６（b）Framed
（枠付）

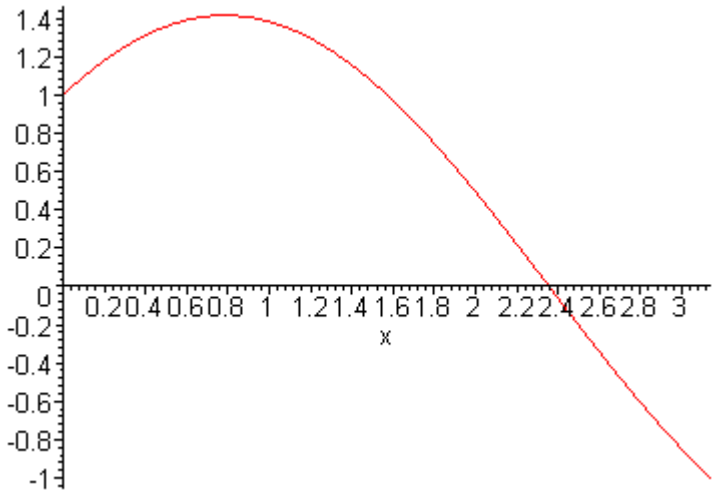

図６（c）Normal
（通常）

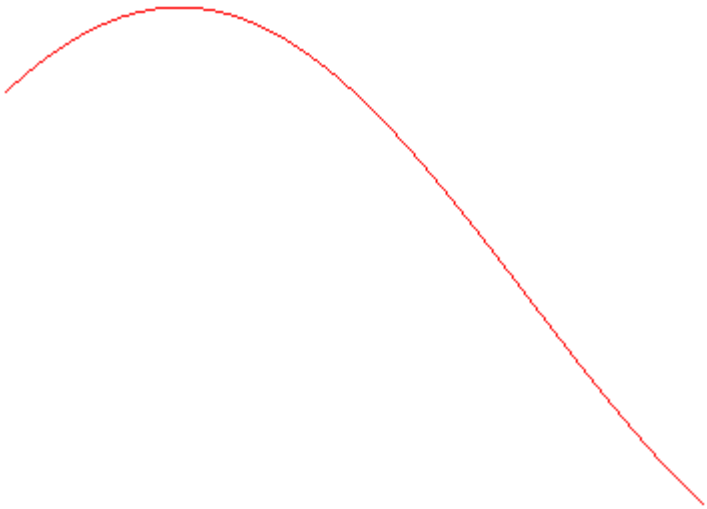

図６（d）None
（軸無し）

ここまでは通常の２次元空間上でのＭａｐｌｅを用いた作図方法についてその概略を見てきました。例として，諸君らの知っている関数の幾つかについて作図処理を行ってきました。ところで，複数個の関数を同じ座標上に描く場合には括弧を用い集合の要素を列記

するようにします。例えば，二つの関数ｆ（ｘ）とｇ（ｘ）を同じ座標に描きたい場合には｛ｆ（ｘ），ｇ（ｘ）｝などとしてまとめます。さらに，これらそれぞれを区別するために，線の種類や色などの指定も行うことができます。作図方法や各種オプションの指定について工夫することにより多様な作図ができるので，色々と試してみましょう。オプション指定はプログラム文中で行うこともできますし，描画後，メニューを表示してその中から選ぶことによっても行うことができます。

３．　３Ｄ図形作成について

先の２Ｄ図形作成と同じ手順を辿って３Ｄ図形を作成しましょう。ヘルプからIntroduction, Graphics, 3D, plot3d と辿り，例題から関数 x*exp(-x^2-y^2)をＸ軸とＹ軸それぞれの範囲を－２から２までとし，標準の格子，横２５×縦２５よりも精細な格子数４９＊４９をオプション指定して以下のようなプログラムを入力して描いてみます。

```
>plot3d(x*exp(- x^2 - y^2), X=2..2, y=-2..2, grid=[49,49]);
```

例示したような表示が現れましたか。

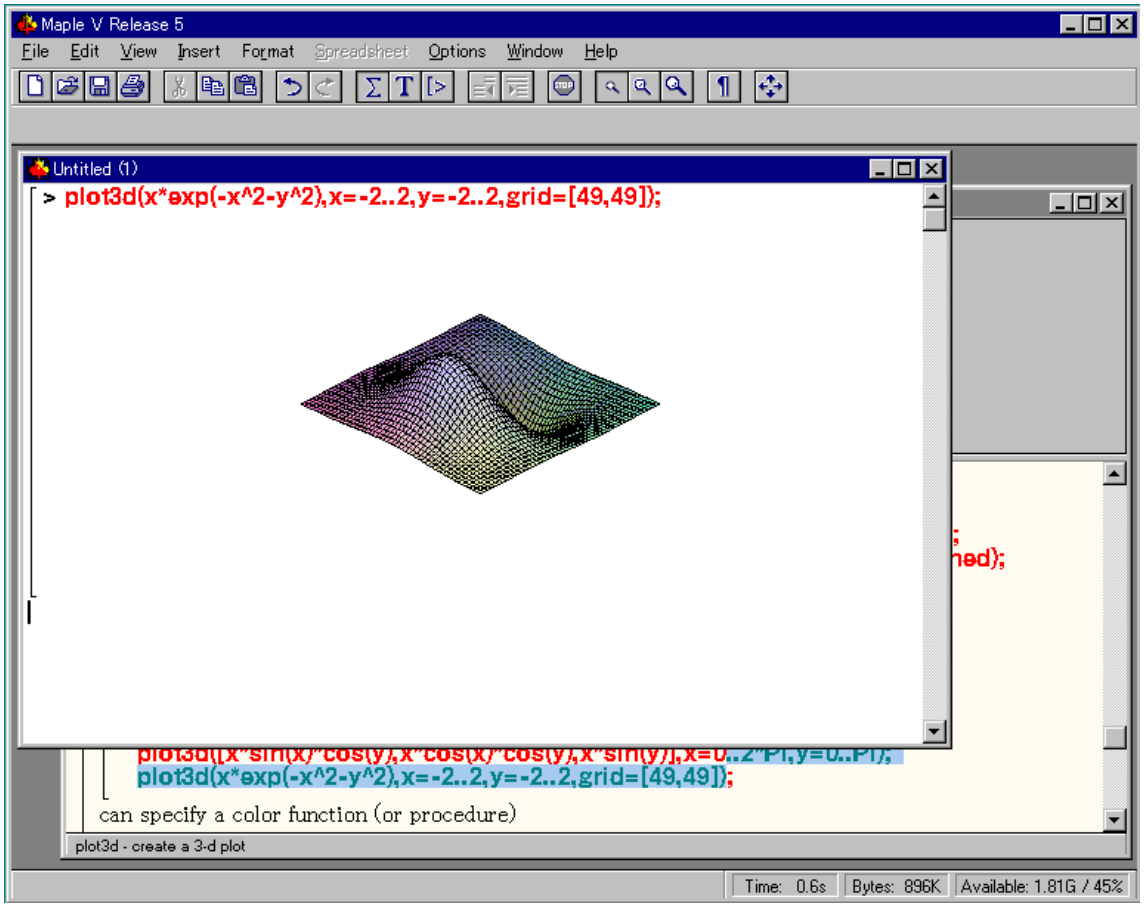

図７　関数の三次元表示

このグラフは，ｘとｙに関する関数の値をＺ軸座標として，この値をＸ軸とＹ軸上に設けた４９×４９個の格子点上で計算し，これら座標点(x,y,z)の集合を関数の形に対応させて線分で連結しレンダリングすることにより表現した３Ｄ図形です。ディスプレイ画面は２次元なので，この３Ｄ図形を隠面消去処理しそれらしく投影したものが描かれます。

投影の際の方向は，次に示すように正のＺ軸から正のＸ軸方向への回転角度φと，正のＸ軸から正のＹ軸方向への回転角度θとにより与えられ，標準ではそれぞれ４５度になっています。なお，この角度はツールバーの中に数値表示（工学単位）されております。

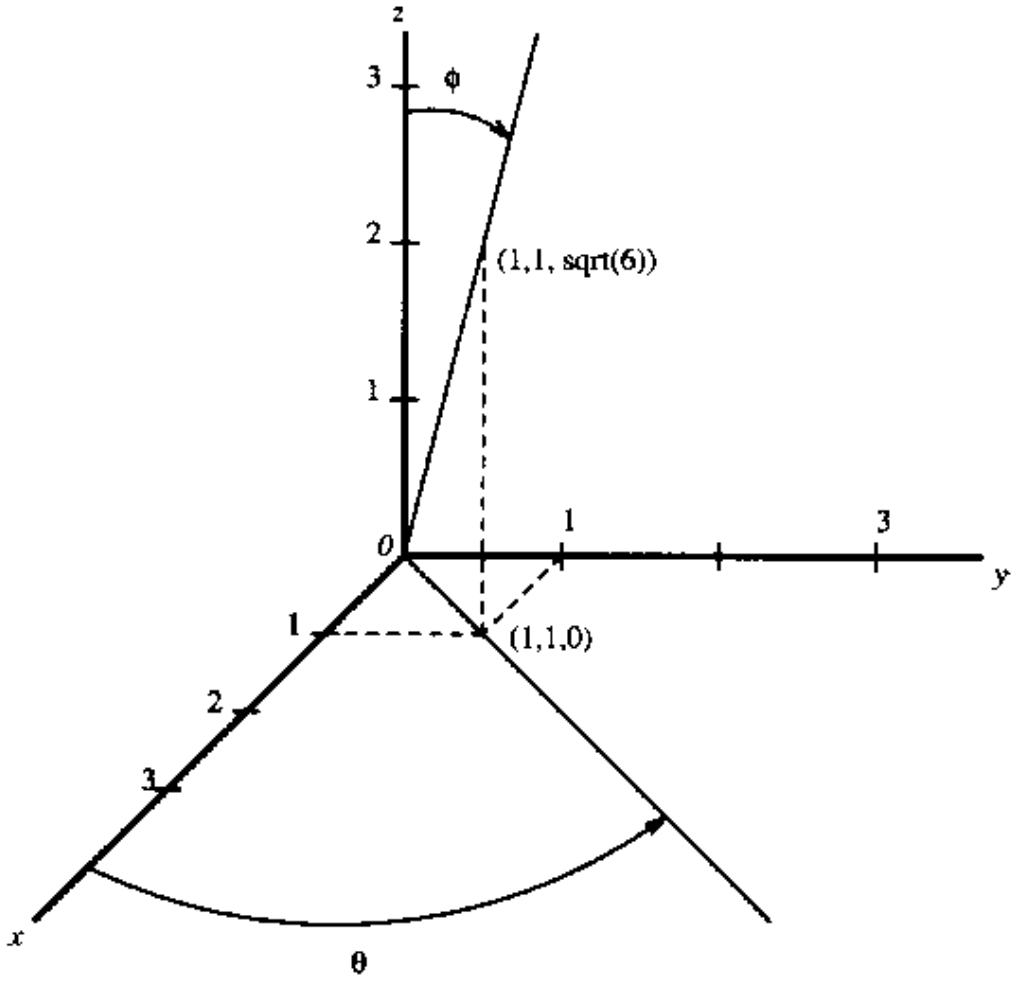

図８　角度θとφ

視点を確認するために，ここで表示した図形をマウスで指示し左クリックしてみましょう。そうすることにより，アイコンの並んでいるのと同じメニュー段に角度θとφの値が表示されているのを確認することができます。どちらも４５度にするのが良いでしょう。

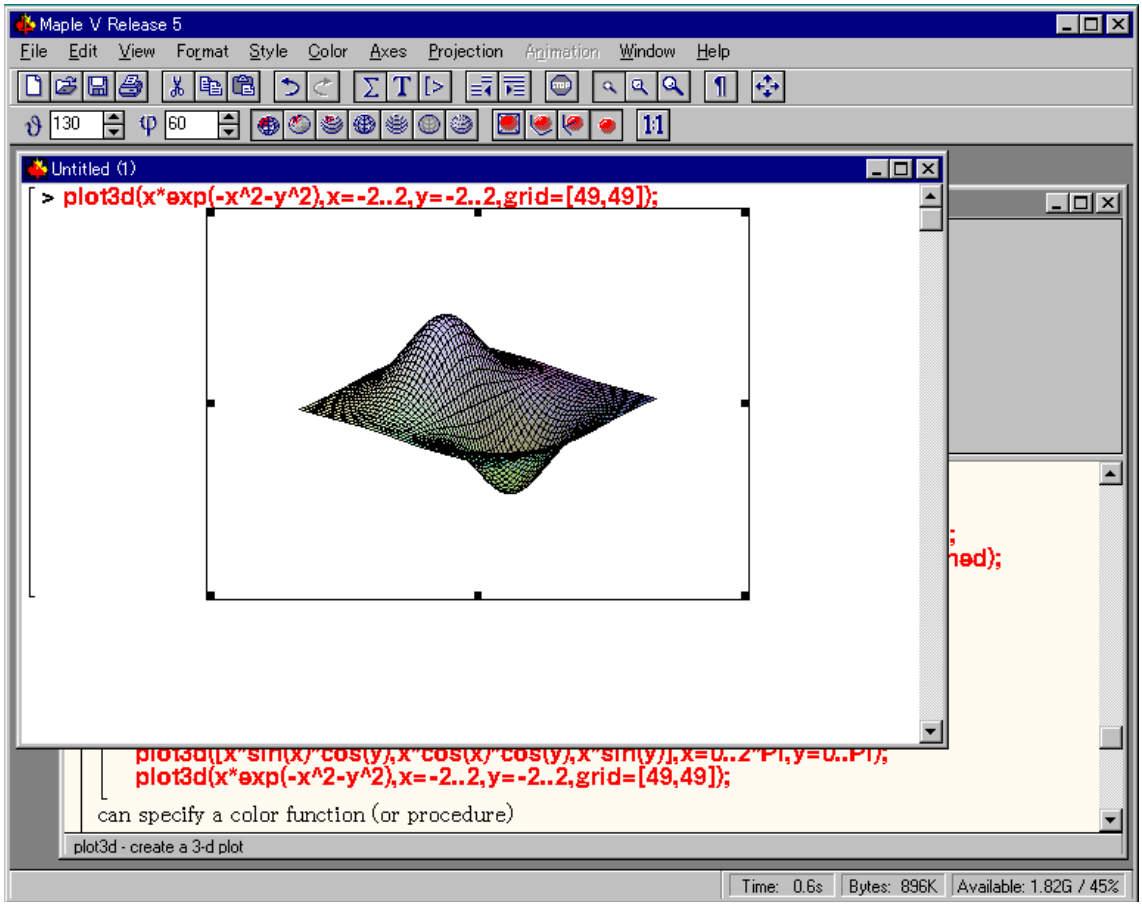

図９　マウス操作により角度を変えてみる

次にこの視点を変更してみましょう。上の表示例ではキィボード入力により角度θを１３０度，φを６０度にして再描画させたものです。ところで視点を示す角度をマウス操作により指定することができます。　３Ｄ図形をマウスで指示し，左クリックしたまま色んな方向にドラッグしてみてください。表示している図形が変化していきますが，これに見とれていないで先程説明した角度表示部分も見ておきましょう。角度θとφの値もマウス操作に応じて変化していきますね，確認できましたでしょうか。

ところで，メニューバーやその下のツールバーに２Ｄの場合とは異なる項目が並んでいるのに気がつきましたか。このメニューバー中の Projection（投影）には並行投影や遠近法による投影などから投影方法を選ぶ項目と，図形が縦横ほぼ同じ寸法になるようスケーリングして枠に入るよう制約(Constrained)するか，枠に入れるような制約を加えない(Unconstrained)かを選ぶ項目とがあります。Style(描画方法)には，パッチ(patch；標準の描画方法)，格子枠なしのパッチ(Patch w/o grid)，等高線付きのパッチ(Patch and contour)，隠線消去(Hiddenline)，等高線(Contour)，ワイヤフレーム(Wireframe)，点(Point)などの項目もあります。また，ツールバー中にはこれらの項目がアイコン表示されています。以上取り上げた各項目の意味を理解するため色々試してみましょう。

４．　座標データによる２Ｄ作図

関数と変域を与えることにより作図できることを見てきました。ここでは，座標点データのリストを与えて作図します。座標点データを並べるとこれらの点を通過する一筆書きのようにして描かれるので，この折れ線グラフを描くための座標を数値として入力していきます。このようにすることにより，関数と変域により表すのが困難な図形を描くことができます。この座標点データのリストをプログラムを作成し，これを実行することにより座標データのリストを生成する手法についても触れて行きますが，ここでは直接座標入力することにより作図する方法を実験しながら修得していくことにします。

まず最初に以下の２Ｄ作図例を取り上げることにします。この例で用いている機能については，Help から Topic　Search を選んで Topic のダイアログボックスを開きこれから項目

を指定してその項目についての説明文などを見ることもできます。ここで取り上げた例は，このカスケード形式のオンラインマニュアル中から選んだものですが，先ず正方形を描き，表題を書き，座標軸の形式を指定してこれに目盛を記入しています。

```
>PLOT(CURVES([[0,0],[0,1],[1,1],[1,0],[0,0]]),TITLE(SQUARE),COLOUR(HUE,0.5),
    AXESSTYLE(BOX),AXESTICKS(10,[0=`0`,0.5=`1/2`,1=`1`]));
```

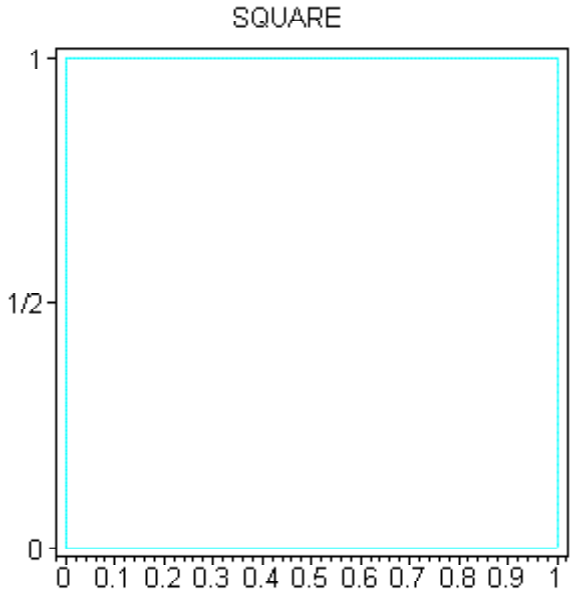

図１０　関数ＣＵＲＶＥを用いて正方形を描く

オプション CURVES により折れ線近似した曲線で描くことを指定し，そして折れ線を描くための座標値の並びを与えています。折れ線近似以外に POINTS，POLYGONS，TEXT などを指定してそれぞれ点，多角形，文字を使って描くことができます。オプション CURVES では，折れ線１本をベクトル・データ[[x1,y1],[x2,y2],...[xn,yn]]により表します。オプション TITLE により図の表題を書き込むことができます。例では SQUARE としています。またオプション AXESSTYLE により座標軸の形式を指定することができます。さらにオプション AXESTICKS により，Ｘ座標およびＹ座標の目盛の付け方を指定することができます。例ではＸ軸は等間隔に１０分割した目盛を，Ｙ軸に関しては３箇所の指定した座標値に対して対応した文字列（分数表記）を記入しています。

オプション項目 COLOUR(または COLOR))の中で指定しているパラメータ HUE とは色合いのことです。指定できる値の範囲は０から１までで，この範囲外の場合の色は赤になります。このような色指定に用いる値は通常に０から１までの範囲から選ぶようになっています。色合いによる色指定 COLOUR(HUE,x)とＨＳＶ形式による色指定 COLOUR(HSV,x,0.9,1.0)は同じ色を指しています。またＲＧＢ形式による色指定 COLOUR(RGB,r,g,b)を用いて３原色（赤緑青）の成分毎に値を与えて色指定することもできます。

ところで項目 AXESTICKS での細かな指定はＣＵＩ(character user interface))ならではのものでしょう。これをＧＵＩ(graphics user interface)により指定しようとするとＣＵＩによるよりも作業量が多くなることもあります。作業結果は，ＧＵＩでは画像情報も含めてファイルにして保存しておく必要がありますが，ＣＵＩの場合には文字情報のみファイルに保存しておけば良く，同じ処理を行えばいつでも同じ結果を再現できます。しかし思うような結果を得るためには，オプション指定する項目を調べたり決めたりするための作業時間がＧＵＩによるよりも長くなるかも知れません。これらのことから，各オプション項目はＣＵＩもしくはＧＵＩのどちらによっても指定することができるが，とりあえずＣＵＩによる指定を行っておき，その後表示結果を見ながらＧＵＩ指定するのが良いかも知れません。各自の気に入った方法を選んで試して見てください。

もう一例２Ｄ作図をしてみましょう。この例では，原点の位置にダイアモンド型の記号を記入し，その右下に位置(align，整列)合わせして文字列 Origin（原点の意）を表示し，さらに一点鎖線(１は実線，２は点線，３は破線，４は一点鎖線)，フォント指定したギリシャ文字，多角形などを描いています。また表示領域をＶＩＥＷにより指定しています。

```
PLOT(POINTS([0,0],SYMBOL(DIAMOND)),
   TEXT([0,0],``Origin``,ALIGNBELOW,ALIGNRIGHT,FONT(HELVETICA,OBLIQUE,10)),
     CURVES([[-3,0.5],[3,0.5]],THICKNESS(3),LINESTYLE(4)),
      TEXT([0,0.5],``Dotted``,ALIGNBELOW),
      TEXT([3.1415,0],'p',FONT(SYMBOL,12)),
      TEXT([-3.1415,0],'P',FONT(SYMBOL,12)),
      POLYGONS([[-2,-0.25],[-2,-0.5],[2,-0.5],[2,-0.25]],COLOUR(HUE,0.5)),
      TEXT([0,-0.37],``Red``,COLOUR(RGB,1,0,0)),
      AXESSTYLE(FRAME),VIEW(-4..4,-1..1) );
```

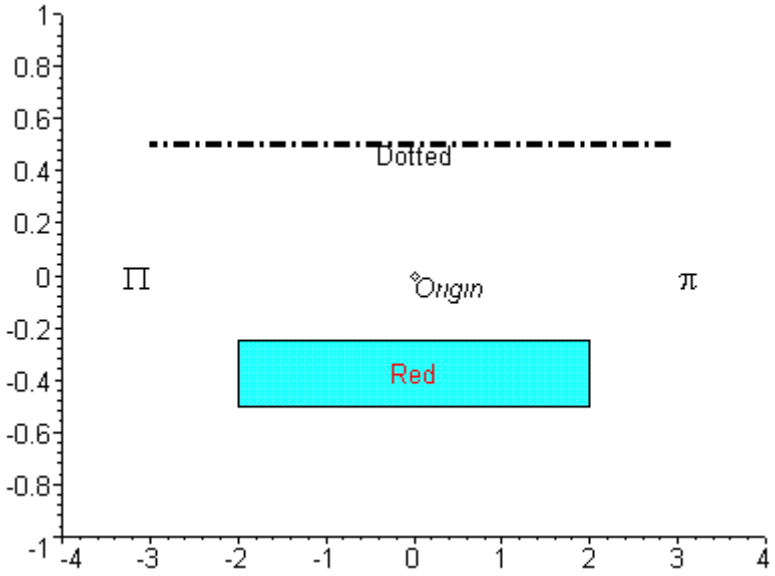

図１１　関数ＴＥＸＴとＣＵＲＶＥを用いて文字と図を描く

指定した座標を中心にして記号を書き込む場合には PLOT のオプションの一つ SYMBOL により中心記号を指定することができます。中心記号として BOX, CROSS, CIRCLE, POINT, DIAMOND, DEFAULT の中から選ぶことができます。SYMBOL オプションを省略した場合には DEFAULT を選んだものとして扱われます。
フォント指定は FONT により行い，文字種として TIMES, COURIER, HELVETICA, SYMBOL の中から何を用いるかを選ぶことができます。また字体については TIME に関しては ROMAN, BOLD, ITALIC , BOLDITALIC の中から選ぶことができる。HELVETICA と COURIER に関しては BOLD, OBLIQUE, BOLDOBLIQUE の中から選べるが, SYMBOL に関しては指定する事ができないので注意してください。また文字の大きさも指定することができるが，その単位にはポイント（約 0.35mm）を用いなくてはいけません。
オプション VIEW により長方形の表示領域(view port)を指定できる。この例では長方形領域のＸ座標の範囲を－４から４まで, Ｙ座標の範囲を－１から１までと指定しています。

ところでオプション TEXT と組併せて SYMBOL でギリシャ文字を指定しています。アルファベットの大文字Ｐはギリシャ文字のΠを，小文字ｐはπに対応します。その他の文字の対応についても試して確かめてみると良いでしょう。また，複数の作図結果を一纏めにする場合には display(３次元描画の場合には display3d)を用います。以下の例では（セミコロンを用いると多量の座標データがリスト出力されるので）コロンと組み合わせて F,G,H に座標データリストを割り当て，その後で display を用いまとめて作図しています。座標データリストの作成方法については，後ほど詳しく述べることにします。

```
>with(plots):F:=plot(cos(x),x=-Pi..Pi,y=-Pi..Pi,style=line):
 G:=plot(cos(x),x=Pi..2*Pi,y=-Pi..Pi,style=point):
 H:=plot(tan(x),x=-Pi..2*Pi,y=-Pi..Pi,style=point):
 display({F,G,H},axes=boxed,scaling=constrained,title=`Cosine and Tangent`);
```

５．　座標データによる３Ｄ作図

関数と変域を与えることにより複雑な３Ｄ作図ができました。ここでは，座標点データのリストを与えて作図する方法について説明します。オンラインマニュアルの中から以下の３Ｄ作図例をとりあげます。この例では正６面体を描いています。多面体(polygon)の一つ正６面体を，その構成面でもある正方形を３次元空間中の４個の座標点の連結として表し，これらを順次６個定義することにより表すのです。なお多面体の構成面は凸である必要があります。この多面体に対する照明は角度 ϕ と θ およびＲＧＢ形式により色指定された平行光線 light=[phi,theta,r,g,b]により行われ，さらには環境光も ambientlight=[r,g,b]により指定できます。また立体感を出すための明暗処理を shading=s により行うこともでき，ｓとしてどの方向に関して明暗の区別を付けるかを XYZ, XY, Z, ZGREYSCALE, ZHUE, NONE により指定します。例えば XYZ と指定すれば全方向に関して色／明暗が変化しますが，この機能に関してはシステムに依存しています。とりあえず適当に指定して表示し，その後に変更を加えながら再表示して確かめることもできるので，各自で試みてください。その他の項目についても指定可能ですが，ツールバーを利用して指定したり，図形をマウスで指示して右クリックして得られるメニュー中から選ぶのが手っ取り早いかも知れません。またＧＵＩでは指定したい項目が見つからなかった場合には，オンラインヘルプを使ってキィワードを入力し(Topic Search)調べるのも良いでしょう。

```
>PLOT3D(POLYGONS([[0,0,0],[1,0,0],[1,1,0],[0,1,0]],    [[0,0,0],[0,1,0],[0,1,1],[0,0,1]],
                 [[1,0,0],[1,1,0],[1,1,1],[1,0,1]],    [[0,0,0],[1,0,0],[1,0,1],[0,0,1]],
                 [[0,1,0],[1,1,0],[1,1,1],[0,1,1]],    [[0,0,1],[1,0,1],[1,1,1],[0,1,1]]),
       LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0), LIGHT(100,-45,0.0,0.0,0.7),
       AMBIENTLIGHT(0.4,0.4,0.4),
       TITLE(CUBE),STYLE(PATCH),COLOUR(ZHUE));
```

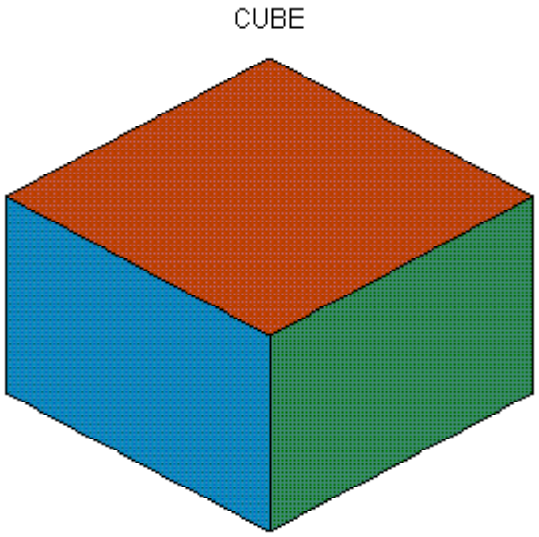

図１２　関数ＰＯＬＹＧＯＮＳにより正方形６面を指定して立方体を描く

６．　その他の３Ｄ作図について

これまでは主として plot3d を例として取り上げ，三次元空間におけるプロット方法に関する説明をしてきました。ここでその他の三次元プロットの方法と併せてまとめ，一覧できるようにしておくことにします。それぞれの方法についてはオンラインマニュアルを使って調べたり，その中にあるサンプルを利用してこれらの機能などを調べてみましょう。

パラメータ曲線プロット
plot3d([x-表式,y-表式,z-表式], var=a..b, var2=c..d, options);

球座標形を使ったプロット
plots[sphereplot](d-表式, θ=角度範囲 1, ϕ=角度範囲 2, options);

球座標形を使ったパラメータプロット
sphereplot([d-表式, θ=角度範囲 1, φ=角度範囲 2], var1=領域 1, var2=領域 2, options);

円筒座標形を使ったプロット
plots[cylinderplot](r-表式, θ=角度範囲 1, z=領域, options);

円筒座標形を使ったパラメータプロット
cylinderplot([r-表式, θ=角度表式 1, z=領域], var1=領域 1, var2=領域 2, options);

曲線プロット
plots[spacecurve]([x-表式,y-表式,z-表式], var=領域, options);

複数曲線のプロット
plots[spacecurve](曲線データのリスト, var=領域, options);

チューブプロット
plots[tubeplot](曲線データのリスト, var=領域);

複数のチューブプロット
plots[tubeplot](曲線データのリスト, var=領域, options);

点リストのプロット
plots[pointplot]({[x1,y1,z1],...,[xn,yn,zn]}, options);

パラメータ曲線から得られる点のプロット
plots[pointplot]]([x-表式,y-表式,z-表式], var=領域, options);

７．　プロシージャについて

式を途切れなく書き続けて行くだけでは望む処理内容を定義できない場合があります。そのような場合にはプロシージャ(procedure)により処理したい内容を書き表して用いるのが良いでしょう。プロシージャの一般形は次のようになっています。

```
proc(引数列)
  local 変数列 e;
  options オプション列;
    statement 1;
    statement 2;
       ・
       ・
    statement n;
end;
```

例として，二つの正の数値から最大値を求めるための関数 saidai を以下のように定義してみました。実際にはライブラリ関数 max を利用できるますが，こちらはパラメータの個数を数えたりする部分などが，もう少しこみ入った処理内容になっています。以下のプロシージャを理解してから，その内容についても少し触れてみましょう。

```
>saidai := proc(a,b)
    print(`Find the maximum of` , a,b);
    if a>b then a;   else b;
    fi;
 end;
```

それでは定義したプロシージャとライブラリ関数 max とを使い比較してみよう。

```
>saidai(8,7);   max(8,7);
                        Find the maximum of, 8, 7
                                    8
                                    8
```

どうやらうまく動いたようですね。次にもう少し複雑な手続きの例を示します。今度はライブラリ関数とほぼ同じ処理ができる関数 saidaiN について見てみましょう。ここでは

いわゆる予約語 type, args, nargs, list, numeric を用いています。なお args は引数を表し, nargs はこの引数の個数を表します。そして type により引数全体[args]が数値(numeric)のリスト(list)から成り立っているかどうかを list(numeric)との比較により調べ，この引数の中から第 i 番目の項目を取り出して使用する場合には args[i]と書き表すことになります。

```
>saidaiN := proc()local result,i;
  if not( type( [args], list(numeric) ) ) then 'procesname(args)';
     elif nargs>0 then result := args[1];
     for i from 2 to nargs do
        if args[i]>result  then  result := args[i]; fi;
     od;  result;
  fi;    end;
```

それでは同様にして，プロシージャ saidaiN とライブラリ関数 max とを比較してみます。

```
>saidaiN(5,3,8,7);    max(5,3,8,7);
                                    8
                                    8
```

同じ結果が得られましたね。ところでかなり複雑なプロシージャの書き表し方もオンラインヘルプを参考にすればできるようになります。使いながら学習してみましょう。

７．１　プロシージャによる２Ｄ座標データの作成と描画

関数を用いて座標データを生成できますが，プロシージャを用いることにより，関数単独だけによるよりも自由度の高い座標データ生成が可能となります。以下にそのような手順を紹介します。例として，プロシージャを用いてテント型の周期関数を定義してみます。周期は２，高さは１です。整数や多項式について剰余を求めるライブラリ関数は用意されていますが，実数の剰余関数は用意されていない様なので，変数(k)を使った繰り返し文により剰余を求めるようにしています。この関数では，引数が正の場合には０から２の範囲に入るまで２づつ減算を行い，その後でさらに１を減算してから絶対値を求めています。引数が負の場合にも同様の処理を行って，鋸の歯のような形をした関数 sawf を定義しています。このように範囲を分けて計算する場合には以下のようにすれば良いでしょう。

```
>sawf := proc（x）local k, t;    t := x;
     if x>0 then for k from 0 by 1 while t>2   do t := t-2; od;   abs(t-1);
             else for k from 0 by 1 while t <= -2   do t := t+2;   od;   abs(t+1);   fi
 end;
```

ところで，この関数は偶関数なので正負の判別を省略できるよう引数の絶対値(absolute value)を abs により求め利用すると，以下のように簡単に表すこともできます。

```
>sawfn := proc（x）local k, t;   t := abs(x);
       for k from 0 while 2 <= t   do t := t-2 od;   abs(t-1);
  end;
```

ではプロシージャによる関数 sawf を使って(x,y)座標データのリストを作ってみましょう。このためには seq と組み合わせて用います。以下のようにしてデータ並び(list, sequence)を生成することができるので，手入力によるよりも効率的ではないでしょうか。

```
>PLIST:=[seq([i/5,sawf(i/5)],i=-5..15)];
```

この結果次のような座標データのベクトル・データによるリストが得られます。

```
PLIST := [[-1, 0], [-4/5, 1/5], [-3/5, 2/5], [-2/5, 3/5], [-1/5, 4/5], [0, 1], [1/5, 4/5],
[2/5, 3/5], [3/5, 2/5], [4/5, 1/5], [1, 0], [6/5, 1/5], [7/5, 2/5], [8/5, 3/5], [9/5, 4/5],
[2, 1], [11/5, 4/5], [12/5, 3/5], [13/5, 2/5], [14/5, 1/5], [3, 0]]
```

ではこの座標データリストを用いて２Ｄ図形を描いてみましょう。以下のように PLOT に対して曲線の描画 CURVES をオプション指定し，これに対して座標データリスト名 PLIST を与えてやれば以下のような２Ｄ図形が得られます。

> PLOT(CURVES(PLIST));

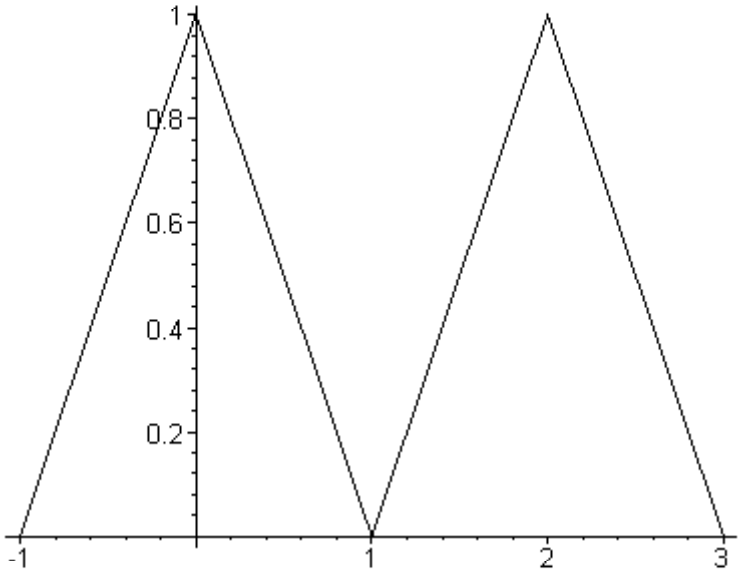

図１３　関数ＣＵＲＶＥに座標データのリストを代入

７．２　プロシージャによる３Ｄ座標データの作成と描画

先に作成したプロシージャ sawf を用い３次元空間内に折れ線を描いてみることにします。次の手順のようにして(x,y,z)座標のデータ並び PLIST を求め，これを plots[spacecurve]とするか with(plots)とした後 spacecurve を用いて PLIST の描画をすればよいでしょう。ただし，データ並びの作成には seq(sequence ； 並びの意)を用います。以下の例では x=i/5, y=sawf(i/5)，z=0 として２１個(i=-5..15)の座標値からなるデータ並びを作成しています。

>PLIST:=[seq([i/5,sawf(i/5),0],i=-5..15)]; with(plots): spacecurve({PLIST},axes=BOX);

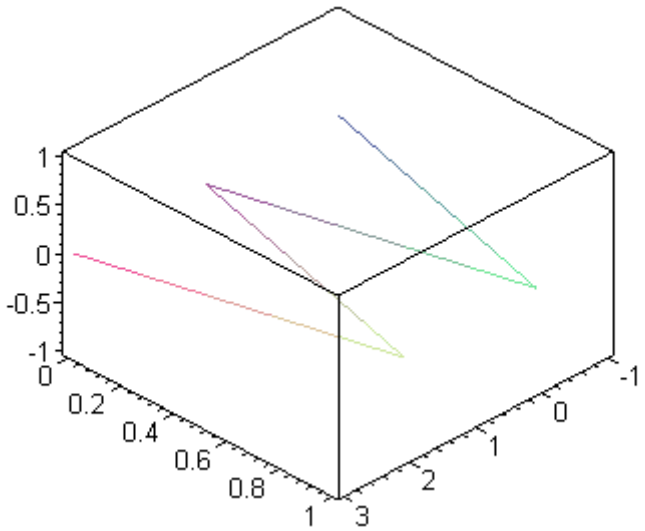

図１４　関数ＳＰＡＣＥＣＵＲＶＥに座標テータリストを代入

また，z=t とし，以下の様に seq を用い t を０から１まで計２回繰り返すことにより，

座標値 z だけが異なる同じ図形を２個描くこともできます。

```
>PLIST2:=seq([seq([i/5,sawf(i/5),t],i=-5..15)],t=0..1);
                    plots[spacecurve]({PLIST2},axes=BOX);
```

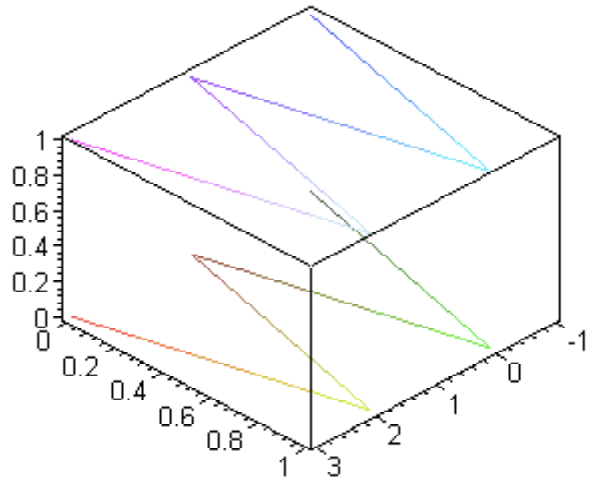

図１５　繰り返し操作により同じ図形を位置をずらして２回描く

ここでの例では図形として曲線を取り上げこれを seq([i/5,sawf(i/5),t],i=-5..15)のようにして２１個の座標点を用いた折れ線により定義しました。ここでは，同じプロシージャを用いてＸ，Ｙ座標成分を seq([i,sawf(i),t],i=0..2)のように与えて３角形を定義し，これにＺ座標を加えて seq でまとめ，同じ３角形を複数個描くことにします。以下のようにすれば３角形を３個描くことができます。

```
>PLIST3:=seq([seq([i,sawf(i),t],i=0..2)],t=0..2);
PLOT3D(POLYGONS(PLIST3),
   LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),LIGHT(100,-45,0.0,0.0,0.7),
   AMBIENTLIGHT(0.4,0.4,0.4), TITLE(TRIANGLES),STYLE(PATCH),COLOUR(ZHUE));
```

さらに先に述べたように，描画結果を示す座標データリストを適当な変数名で定義しておき，これを display を用いて一纏めにして描くことができます。例えば，以下のようにして座標データリスト DLIST1 と DLIST2 を定義しまとめて描画することができます。少し込み入っていますが，視点を変えて見ると描画内容の確認が簡単になります。

```
>DLIST1:=PLOT3D(POLYGONS(PLIST3),
     LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),LIGHT(100,-45,0.0,0.0,0.7),
     AMBIENTLIGHT(0.4,0.4,0.4),ORIENTATION(-150,60),
     STYLE(PATCH),COLOUR(ZHUE)):
>DLIST2:=plot3d(cos(Pi*x*y),x=0..2,y=0..2):
>display({DLIST1,DLIST2});
```

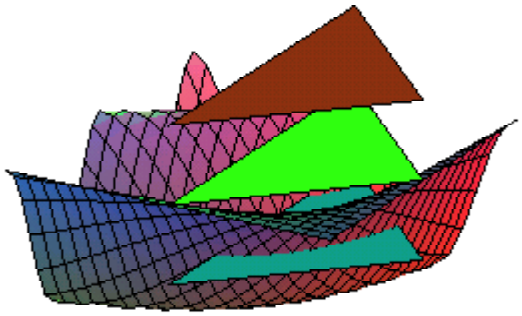

図１６　複数の座標データリストを同じ画面内に描画する

７．３　プロシージャによる３Ｄ作図応用

球面を３角形の集合として描いてみましょう。球面を緯度方向に関して変数 nv で，経度方向に関しては nh などで分割することにします。このため，緯度方向の分割は北極点に相当する場所を角度ゼロ南極点に相当する場所を角度πとして nv により等分割し，経度方向には nh で等分割してこの格子状の分割により合計 2nh 個の３角形と(nv-2)*nh 個の４角形を得ます。３角形は４角形を構成する隣り合う２点が縮退したものと考えると，この分割により得られる４角形の数は nv*nh 個となりますね。これより格子点座標からこの４角形の座標[[xhv00,yhv00,zhv00],[xhv10,yhv10,zhv10],[xhv11,yhv11,zhv11],[xhv01,yhv01,zhv01]]を生成するプロシージャ bodyfn を以下のように定義します。

```
>nv:=9; nh:=18;
bodyfn := proc（nv,nh,h,v)
local　ht,vt,xhv00,yhv00,zhv00,xhv01,yhv01,zhv01,xhv11,yhv11,zhv11,xhv10,yhv10,zhv10,h1,v1;
    ht:=2*evalf(Pi)/nh; vt:=evalf(Pi)/nv;　h1:=h+1; v1:=v+1;
    xhv00:=sin(vt*v)*cos(ht*h);yhv00:=sin(vt*v)*sin(ht*h);zhv00:=cos(vt*v);
    xhv01:=sin(vt*v1)*cos(ht*h);yhv01:=sin(vt*v1)*sin(ht*h);zhv01:=cos(vt*v1);
    xhv11:=sin(vt*v1)*cos(ht*h1);yhv11:=sin(vt*v1)*sin(ht*h1);zhv11:=cos(vt*v1);
    xhv10:=sin(vt*v)*cos(ht*h1);yhv10:=sin(vt*v)*sin(ht*h1);zhv10:=cos(vt*v);
 [[xhv00,yhv00,zhv00],[xhv10,yhv10,zhv10],[xhv11,yhv11,zhv11],[xhv01,yhv01,zhv01]]; end;
```

プロシージャ bodyfn の中で円周率πを evalf(Pi)により浮動小数点数で計算評価することを指定しているのは，以降の描画処理で必要となる座標データが記号ではなく数値でなくてはならないからです。さて，このプロシージャ bodyfn を用い，以下のようにして座標データのリスト BLIST を作成してみましょう。

```
>BLIST:=seq(seq(bodyfn(nv,nh,h,v),v=0..nv-1),h=0..nh-1):
```

文末をセミコロン（;）ではなくコロン（:）にしているのは，座標データの量が多いからです。デバッグの際にこれらを使い分けると効率的です。ではこの座標データリストを用いて曲線と曲面を描いてみましょう。曲線を描くためには以下のようにします。

```
>with(plots): spacecurve({BLIST},axes=BOX);
```

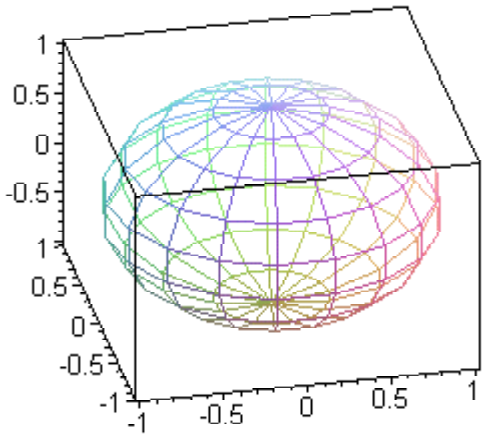

図１７　プロシージャを用いて作成した座標データリストの描画

同様に，曲面を描くためには以下のようにすればよいでしょう。

```
>PLOT3D(POLYGONS(BLIST),
  LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),LIGHT(100,-45,0.0,0.0,0.7),
  AMBIENTLIGHT(0.4,0.4,0.4), TITLE(BALL),STYLE(PATCH),COLOUR(ZHUE));
```

BALL

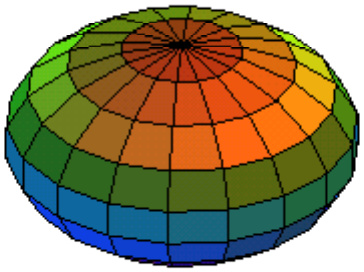

図１８　球面上の格子点座標データリストを用いた多面体の描画

それでは，同じプロシージャ bodyfn を用い，範囲指定を変えて面の数を少なくして描いてみましょう。そのためには，例えば，以下のようにすると良いでしょう。

```
>BLIST1:=seq(seq(bodyfn(nv,nh,h,v),v=1..nv-2),h=0..nh-4):
  PLOT3D(POLYGONS(BLIST1),
    LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),LIGHT(100,-45,0.0,0.0,0.7),
    AMBIENTLIGHT(0.4,0.4,0.4), ORIENTATION(-30,30) ,
    TITLE(CRACKED_BALL),STYLE(PATCH),COLOUR(ZHUE));
```

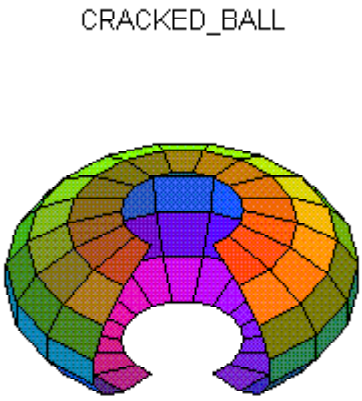

図１９　描画面の表と裏の様子

プログラムで BLIST1:=seq(seq(bodyfn(nv,nh,h,v),v=1..nv-2),h=0..nh-4):とあるのは，プロシージャ bodyfn で北極相当(v=0)および南極相当(v=nv-1)周辺を省き，さらに裂け目に相当する部分 h=nh-3..nh-1 も省いているからです。また ORIENTATION(-30,30)により省略による裂け目が見易い位置にくるような視点を指定しています。

８．　２Ｄアニメーション

アニメーション(animation)とは命を与える(animate)という意味からきております。同じ様な図形であっても，アニメーションにより動きを伴う表示にすれば，見る人により強い印象を与えることができます。また，内容説明を短時間で行なったり，直感的な意味の理解を容易にすることもできます。ここではこのアニメーションについての解説を行います。最初に，描画環境を定義し，次に関数を範囲指定してアニメーション表示してみます。

```
>with(plots):
 animate(sin(3*x*t),x=-Pi..Pi,t=0..1,view=-0.8..1);
```

座標軸などが表示された後，図形部分をマウスで指示して左クリックし，切り替え表示されたツールバーの中から再生ボタンを選んで左クリックしてみましょう。三角関数を用いたアニメーション表示を見ることができましたね。処理内容の詳細については ALIST:=animate(sin(3*x*t),x=-Pi..Pi,t=0..1,view=-0.8..1);などとすれば見ることができます。

それでは次に進みましょう。今度は処理内容の理解をするため，プロシージャを用いた２Ｄアニメーションにチャレンジしてみましょう。プロシージャとしては前に定義した sawfn を使うことにします。このプロシージャをファイルに保存している場合には，読み込んだ後その場所まで画面をスクロールして実行してください。または以下を参照して再定義しても構いません。

```
>sawfn := proc（x） local k, t;
    t := abs(x); for k from 0 by 1 while t>2 do t := t-2; od; abs(t-1); end:
>ALIST:=seq( [CURVES([ seq( [t*i,sawfn(i)],i=-3..3 ) ])],t=1..7 ):
  PLOT(ANIMATE(ALIST));
```

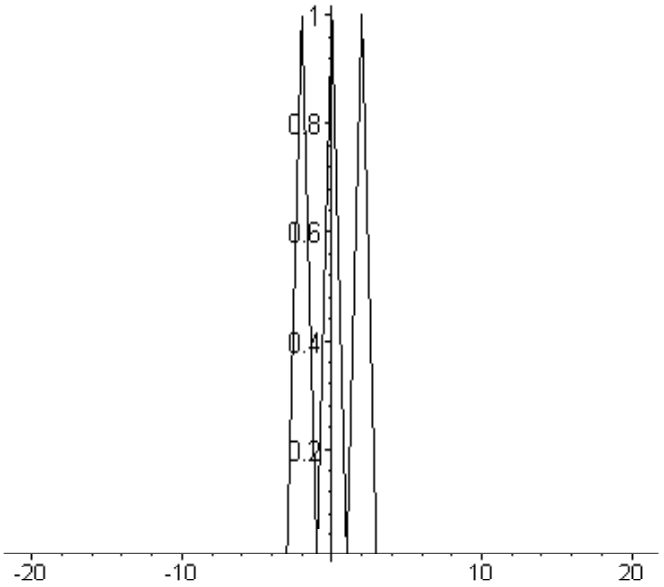

図２０　２次元アニメーションの例

ではプロシージャ sawfn と seq とを用いて座標データのリストを生成することにします。ここでの２Ｄアニメーションは，曲線の集合の中から曲線を１本ずつ選んで順次表示することにより実現しています。そして，CURVE と seq を用い座標点のリストからなる折れ線を１本定義し，これをさらに seq により複数本にして，曲線の集合を表す座標データのリスト ALIST を求めることにします。そして PLOT とオプション ANIMATE を使い実行すれば２Ｄアニメーション表示が得られるはずです。それではこれらのことを考えながら実験してみることにしましょう。上手く表示できましたか。

表示領域の指定は PLOT(ANIMATE(ALIST) ,VIEW(DEFAULT,0.5 .. 1.));などとすればよい。一般にオプションの指定方法は，用いようとしている言語の機能が上位レベルから下位レベルに落ちるにつれて，より厳密に書く必要があります。オンラインヘルプから自分が考えているアニメーション処理と類似のサンプルを探し，これをコピー・ペーストし，さらに表示処理の部分(animate...)を代入処理(例えば ALIST:=animate...)に書き換えて実行し表示すれば，オプション指定についての例題が得られますので利用してください。

９．　３Ｄアニメーション

この資料で扱っているＣＧの最終段階になりました。先の２Ｄアニメーションで学んできたことを基にして，ここでは３Ｄアニメーションの実験をしてみましょう。最初はＭａｐｌｅのオンラインヘルプ中にあるものを参考にして，次の様な関数を３次元アニメーションとして表示してみましょう。

```
>with(plots):
 animate3d(cos(t*x)*sin(t*y),x=-Pi..Pi, y=-Pi..Pi,t=1..2);
```

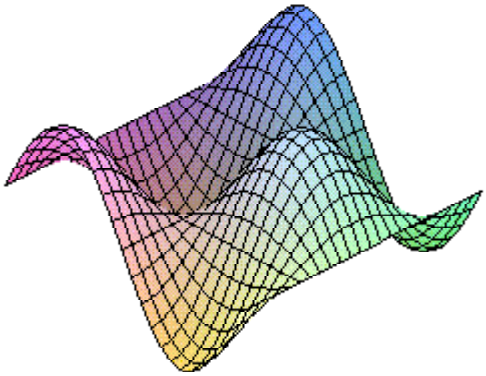

図２１　関数を用いた３次元アニメーションの例

図形部分をマウスで指示して左クリックし，切り替え表示されたツールバーの中から再生ボタンを選んで左クリックしてください。３Ｄアニメーション表示を見ることができましたね。処理内容の詳細については ALIST:=animate3d(cos(t*x)*sin(t*y),x=-Pi..Pi,y=-Pi..Pi,t=1..2);とすれば知ることができます。

この表示内容と同様のものをプロシージャや seq などを組み合わせた処理により得ることができます。２Ｄアニメーションと同様にして，例えば，多面体のデータリストを複数作成すれば，これを順次用い表示することにより３Ｄアニメーションが可能になるのです。では早速取りかかることにしましょう。

簡単な例として，３次元空間中に置いた正方形を回転表示するアニメーションを作成することにしてみましょう。正方形の面は POLYGON([[0,0,0],[1,0,0],[1,1,0]],[0,1,0]])により定義します。この１枚の面をＺ軸の周りで１回転させることにします。そのため，角度 2 π/na を単位として１周を等角度で刻んでいった位置での正方形の４個の頂点座標を計算し，これを用いて各回転位置における正方形を表します。回転角度を刻んでいくための変数を a などとし，seq を用いて回転する正方形の座標データリスト A1 を次のようにして求めます。その後で PLOT3D と ANIMATE を用い光源などのオプション指定とを組み合わせて３Ｄアニメーション表示すれば良いのです。

```
>na:=12: ang:=2*evalf(Pi)/na:
  A1:=seq( [ POLYGONS( [[0,0,0],
  [cos(ang*a),sin(ang*a),0],
  [cos(ang*a)-sin(ang*a),cos(ang*a)+sin(ang*a),0],
  [-sin(ang*a),cos(ang*a),0]] ) ], a=0..na-1 ):
> PLOT3D(ANIMATE(A1),LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),
   LIGHT(100,-45,0.0,0.0,0.7), AMBIENTLIGHT(0.4,0.4,0.4),
   TITLE(ROTATE_SQUARE),STYLE(PATCH),COLOUR(ZHUE));
```

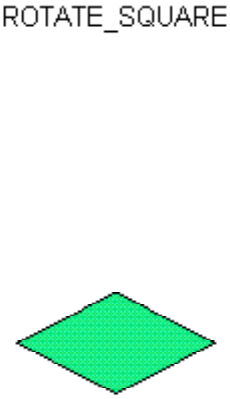

図２２　正方形を１周１２分割し回転させて表示

図形をマウスにより指示して左クリックすればアニメーション用のツールバーが切り替え表示されます。この中から繰り返しボタン，再生ボタンと順に左クリックにより押していけば，回転する正方形が表示されます。うまくいきましたか。

９．１　３Ｄアニメーション応用

応用編です。前に用いたプロシージャ bodyfn を参考にして，円軌道を描いて回る球を多面体により表してみましょう。この多面体表面上の格子点座標を求めるため，プロシージャ bodyfnc を用意します。新たなパラメータとして，球の半径 r1，原点からの距離 r2，１周の分割数 nc などが追加されており，座標を表す変数名は簡略化されています。

プロシージャ bodyfnc と seq とにより半径 r2 の円軌道を描いて回る半径 r1 の球を座標データリストを求めて B1 に代入しておき，さらに先に定義したプロシージャ bodyfn により回転中心に置いた半径１の球を表す座標データリストを B2 に求めておきます。そして，この二つの図形を一方を３Ｄアニメーションで，もう一方は３Ｄグラフィックスにより組み合わせて同一座標上に表示します。この場合，各種オプションは指定順序に従い前の値に上書きされるので最後の値が有効です。では準備ができたら実行してみましょう。

```
>nv:=6: nh:=12: nc:=18: r1:=0.5: r2:=3:
 bodyfnc := proc（c,h,v）
 local ht,vt,ct,xc,yc,x00,y00,z00,x01,y01,z01,x11,y11,z11,x10,y10,z10,h1,v1;
 ht:=2*evalf(Pi)/nh; vt:=evalf(Pi)/nv;
 ct:=2*evalf(Pi)/nc; xc:=r2*cos(ct*c);yc:=r2*sin(ct*c); h1:=h+1; v1:=v+1;
 x00:=r1*sin(vt*v)*cos(ht*h)+xc;y00:=r1*sin(vt*v)*sin(ht*h)+yc;z00:=r1*cos(vt*v);
 x01:=r1*sin(vt*v1)*cos(ht*h)+xc;y01:=r1*sin(vt*v1)*sin(ht*h)+yc;z01:=r1*cos(vt*v1);
 x11:=r1*sin(vt*v1)*cos(ht*h1)+xc;y11:=r1*sin(vt*v1)*sin(ht*h1)+yc;z11:=r1*cos(vt*v1);
 x10:=r1*sin(vt*v)*cos(ht*h1)+xc;y10:=r1*sin(vt*v)*sin(ht*h1)+yc;z10:=r1*cos(vt*v);
 [[x00,y00,z00],[x10,y10,z10],[x11,y11,z11],[x01,y01,z01]]; end:
BLIST1:=seq( [POLYGONS( seq(seq(bodyfnc(c,h,v),v=0..nv-1),h=0..nh-1) )],c=0..nc-1):
B1:=PLOT3D(ANIMATE(BLIST1),
   LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),LIGHT(100,-45,0.0,0.0,0.7),
   AMBIENTLIGHT(0.4,0.4,0.4),
   TITLE(BALLS_AROUND_BALL),STYLE(PATCH),COLOUR(ZHUE)):
r1:=0.3: r2:=4:
BLIST2:=seq([POLYGONS(seq(seq(bodyfnc(nc-c-1,h,v),v=0..nv-1),h=0..nh-1))],c=0..nc-1):
B2:=PLOT3D(ANIMATE(BLIST2)):
B3:=PLOT3D(POLYGONS(BLIST),STYLE(PATCH),
   COLOUR(ZHUE),SCALING(CONSTRAINED)):
```

>display({B1,B2,B3});

図２３　衛星は惑星の周りをそして惑星は恒星の周りを回転する

表示された図形をマウスで指示し左クリックしてツールバーを切り替えましょう。このツールバーの右端にある繰り返しボタンを押してから再生ボタンを押してみましょう，どうです随分と簡単に３Ｄアニメーションを実現することができたでしょう。

１０．　課題とレポート

これまでに学習してきたことを基に，簡単なＣＧ作成に挑戦してみましょう。例えば，以下の課題から一つを選び，作成プログラム，実行結果，解説，考察を加えてレポートにまとめるようなことをするとＣＧの基本が良く理解できます。

課題１．正方形の独楽が回転するような３Ｄアニメーションの作成
　　（軸は面の中心を通る法線）
課題２．正方形の独楽が回転するような３Ｄアニメーションの作成
　　（軸は面の中心を通り法線と成す角はφ）
課題３．多角形で近似した円の独楽が傾いて回転するような３Ｄアニメーションの作成
課題４．軸が面の中心を通る立方体の独楽が回転するような３Ｄアニメーションの作成
課題５．軸が重心を通る傾いた立方体の独楽が回転するような３Ｄアニメーションの作成
課題６．多角形で近似した円盤型の独楽が傾き回転するような３Ｄアニメーションの作成
課題７．プロシージャを用いた立体図形の作成
課題８．プロシージャを用いた立体図形の３Ｄアニメーション作成
課題９．正多面体を３種類以上作成
課題 10．その他３Ｄアニメーションの作成

１１．　参考資料

参考資料として以下に幾つかのサンプルとその解説を示しておきます。

座標計算などを，線型代数(linear　algebra)でお馴染みのマトリックスとベクトルを用い以下のようにして簡単に計算することができます。

１１．１　座標変換

座標軸周りの回転を３次元ベクトル（ｘ，ｙ，ｚ）と行列式が１の３次元正方行列により表すことができる。例えば，Ｙ軸周りでの角度φによる回転とそれに続いたＺ軸周りでの角度θによる回転をＭａｐｌｅにより計算するのは以下のようにします。ここでは３次元空間中の座標点（１，１，０）をベクトルにより表しています。なお，平行移動を考慮した場合には，一般に，行列の次元は４になることを注記しておきます。

```
>with(linalg):                （角度θの値をリセットする場合　theta:='theta':　などとする）
  A := array( [[cos(theta),-sin(theta),0],[sin(theta),cos(theta),0],[0,0,1]] );
>v := vector( [1,1,0] ):   theta:=Pi/4:     （もしくは theta:=evalf(Pi)/4: などとする）
>u:=multiply(A, v);
```

```
>theta:=Pi/3:    u:=multiply(A, v);          （このように計算するのも良いでしょう）
```

１１．２　配列を利用した座標データリストの作成

配列を用いて座標データリストを作成する例を示します。プロシージャを使って多面体により近似した球を描きましたが，球面上の格子点に対応する配列(array)に予め座標値を求めておき，これを用いて個々の面の座標データリストを順次書き表し目的の座標データリストを作成しても同様のことができます。 先ず，次のようにして球面上に設定した格子点に関する座標を求め２次元配列 XX,YY,ZZ に記憶します。また，以降の処理を簡単にするため，南極および北極に相当する同じ値を有する格子点座標と，経度ゼロおよび経度 360 度の経線１本相当分の座標値の重複も含めて,用意した２次元配列に代入します。

```
>nv:=9: nh:=18:
 XX:=array(0..nh,0..nv); YY:=array(0..nh,0..nv); ZZ:=array(0..nh,0..nv);
 ht:=2*evalf(Pi)/nh; vt:=evalf(Pi)/nv;
      for ih from 0 to nh do
       for iv from 0 to nv do
         XX[ih,iv]:=sin(vt*iv)*cos(ht*ih);
         YY[ih,iv]:=sin(vt*iv)*sin(ht*ih);
         ZZ[ih,iv]:=cos(vt*iv);
  od; od;
```

球面を覆う格子点の座標が求まったので，これを４個ずつまとめて順次参照して多面体を表す座標データリストを作成し，これを BLIST に代入しておきます。そしてこれを用いて描画すれば，以前の実験で得たのと同じ結果が得られるはずです。処理時間は，この配列を用いる例での座標の計算回数はプロシージャによる場合の約４分の１になるので，こちらによる方が随分と短時間に処理が済むはずです。しかし，このような座標データ作成にプロシージャを使うのか，配列を使うか，それとも両方とも使うのか，さらには別の機能などを使うか，もしくはキィボードにより力ずくで入力するかは，同じ結果が得られる限りにおいて趣味の問題でもあります。プログラム作成の手間とデータ作成の手間は一般にトレードオフの関係になることが多いので，各自で色々試して見ると良いでしょう。

```
>BLIST:=seq( seq( [[XX[h,v],YY[h,v],ZZ[h,v]],
                  [XX[h+1,v],YY[h+1,v],ZZ[h+1,v]],
                  [XX[h+1,v+1],YY[h+1,v+1],ZZ[h+1,v+1]],
                  [XX[h,v+1],YY[h,v+1],ZZ[h,v+1]]],
        v=0..nv-1),h=0..nh-1):
>with(plots):
 PLOT3D(POLYGONS(BLIST),LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),
        LIGHT(100,-45,0.0,0.0,0.7), AMBIENTLIGHT(0.4,0.4,0.4),
        TITLE(BALL),STYLE(PATCH),COLOUR(ZHUE));
```

配列を用いた座標データリスト作成の例をもう二つ示すことにします。先ず，二次元配列中に記憶した座標データから，経線方向に一つ飛びに四辺形を取り出してみます。

```
>BXLIST:=seq( seq(
 [[XX[2*h,v],YY[2*h,v],ZZ[2*h,v]],[XX[2*h+1,v],YY[2*h+1,v],ZZ[2*h+1,v]],
  [XX[2*h+1,v+1],YY[2*h+1,v+1],ZZ[2*h+1,v+1]],
  [XX[2*h,v+1],YY[2*h,v+1],ZZ[2*h,v+1]]], h=0..nh/2-1),v=0..nv-1):
PLOT3D(POLYGONS(BXLIST),LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,60,0.7,0.0,0.0),
        LIGHT(100,-30,0.0,0.0,0.7), AMBIENTLIGHT(0.4,0.4,0.4),
        TITLE(STRIPE_BALL),STYLE(PATCH),COLOUR(ZHUE));
```

次の例では，配列中に記憶してある座標データから，経度方向の指標と緯度方向の指標が四辺形の左上隅でどちらも偶数または奇数になるよう組み合わせ，千鳥格子状になるよう順次四辺形を取り出すことにしました。ただし，nh の値は偶数，nv の値を奇数にしているため，偶数指標の組み合わせでは取り出す四辺形の数は(nh/2)*(nv/2+1)，奇数指標の組み合わせでは取り出す四辺形の数が(nh/2)*(nv/2)となり，少し異なっています。

```
BYLIST:=seq(seq(
 [[XX[2*h,2*v],YY[2*h,2*v],ZZ[2*h,2*v]],
  [XX[2*h+1,2*v],YY[2*h+1,2*v],ZZ[2*h+1,2*v]],
  [XX[2*h+1,2*v+1],YY[2*h+1,2*v+1],ZZ[2*h+1,2*v+1]],
  [XX[2*h,2*v+1],YY[2*h,2*v+1],ZZ[2*h,2*v+1]]],h=0..nh/2-1),v=0..nv/2),
seq(seq(
```

```
 [[XX[2*h+1,2*v+1],YY[2*h+1,2*v+1],ZZ[2*h+1,2*v+1]],
  [XX[2*h+2,2*v+1],YY[2*h+2,2*v+1],ZZ[2*h+2,2*v+1]],
  [XX[2*h+2,2*v+2],YY[2*h+2,2*v+2],ZZ[2*h+2,2*v+2]],
   [XX[2*h+1,2*v+2],YY[2*h+1,2*v+2],ZZ[2*h+1,2*v+2]]], h=0..nh/2-1), v=0..nv/2-1):
PLOT3D(POLYGONS(BYLIST),LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,30,0.7,0.0,0.0),
          LIGHT(100,-30,0.0,0.0,0.7), AMBIENTLIGHT(0.4,0.4,0.4),
          TITLE(MESH_BALL),STYLE(PATCH),COLOUR(ZHUE));
```

１１．３　座標変換とアニメーション

各種の幾何図形に対して連続性のあるような座標変換を施しこれを順次表示すればアニメーションになります。ここではそのような座標変換のうち，Ｚ軸周りでの回転によるアニメーションの例を示します。座標変換を施す対象物として先に取り上げた球を用います。そして一週をｎｃ個に等分割して各回転位置におけるＸＹ座標を計算しその座標データリストを ABLIST1 に求め，これを用いて PLOT3D によりアニメーション表示します

```
>nc:=48: ang:=2*evalf(Pi)/nc:
ABLIST1:=seq( [POLYGONS(seq( seq(
[[cos(ang*c)*XX[h,v]-sin(ang*c)*YY[h,v],
     sin(ang*c)*XX[h,v]+cos(ang*c)*YY[h,v],ZZ[h,v]],
 [cos(ang*c)*XX[h+1,v]-sin(ang*c)*YY[h+1,v],
     sin(ang*c)*XX[h+1,v]+cos(ang*c)*YY[h+1,v],ZZ[h+1,v]],
 [cos(ang*c)*XX[h+1,v+1]-sin(ang*c)*YY[h+1,v+1],
   sin(ang*c)*XX[h+1,v+1]+cos(ang*c)*YY[h+1,v+1],ZZ[h+1,v+1]],
 [cos(ang*c)*XX[h,v+1]-sin(ang*c)*YY[h,v+1],
   sin(ang*c)*XX[h,v+1]+cos(ang*c)*YY[h,v+1],ZZ[h,v+1]]],
 h=0..nh-1),v=0..nv-1)) ], c=0..nc-1):
>with(plots): PLOT3D(ANIMATE(ABLIST1),
     LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),
     LIGHT(100,-45,0.0,0.0,0.7), AMBIENTLIGHT(0.4,0.4,0.4),
     STYLE(PATCH),COLOUR(ZHUE));
```

上記の例では，座標変換の内容を明示的に記述して計算するようにしています。このような計算内容の記述を行うために必要な数学的知識は基礎的なものではあるが，内容を良く理解していたとしても記述作業そのものは結構煩雑です。ところで Maple では座標軸周りの回転および平行移動を対象物に対して施すための手続きが幾つも用意されています。これらを用いると，計算内容について良く理解さえしていれば，煩雑な記述にかける時間を省いてグラフィックスの仕上がりの方に時間をかけることが可能となります。以下の例では，平行移動および座標軸周りでの回転といった座標変換を行うための手続き translate および rotate を用いている。先ず，先に求めた球状の対象物 BLIST に対しＸ軸方向に１だけ平行移動し，次にＸ軸周りでは角度π/6 で回転，Ｚ軸周りでは一週を nh 等分する角度で回転させるための座標変換を順次適用しアニメーション表示で用いる座標データリスト RBALL を求めている。どうです，随分と簡単なので拍子抜けしませんでしたか。

```
>with(plottools):                          (座標変換のためのツール類使用を宣言)
>RBALL:=seq(
 [rotate(translate(POLYGONS(BLIST),1,0,0), evalf(Pi)/6, 0, 2*evalf(Pi)/nh*t)], t=0..nh-1 ):
>PLOT3D(ANIMATE(RBALL),
     LIGHT(0,0,0.0,0.7,0.0), LIGHT(100,45,0.7,0.0,0.0),
     LIGHT(100,-45,0.0,0.0,0.7), AMBIENTLIGHT(0.4,0.4,0.4),
     STYLE(PATCH),COLOUR(ZHUE));
```

座標変換の代表的な手続き translate，rotate および scale は以下のようにして用います。

```
translate(２次元データ,Sx,Sy)                    translate(３次元データ,Sx,Sy,Sz)
rotate(２次元データ，角度，回転の中心点)          rotate(３次元データ ,θ x,θ y,θ z)
rotate(３次元データ,角度,[３次元空間中の２点により回転軸を定義])
scale(２次元データ,Sx,Sy)          scale(２次元データ,Sx,Sy,スケーリング中心点)
scale(３次元データ,Sx,Sy,Sz)      scale(３次元データ,Sx,Sy,Sz,スケーリング中心点)
```

このような手続きを利用した例として，２個の球状物体がそれぞれ別の角速度で回転しながら，互いに追いかけて廻るアニメーションを作成してみましょう。そのためには，例えば以下のようにすれば良いでしょう。

```
>nt:=25:
>BALL1 :=seq([rotate(translate(rotate(POLYGONS(BLIST),0,0,-2*evalf(Pi)/nt*t),
                +1.5,0,0), evalf(Pi)/3, 0, 2*evalf(Pi)/nt*t)],t=0..nt-1):
> BALL2 :=seq([rotate(translate(rotate(POLYGONS(BLIST),0,0,-4*evalf(Pi)/nt*t),
                -1.5,0,0), evalf(Pi)/9, 0, 2*evalf(Pi)/nt*t)],t=0..nt-1):
>RBALL1:=PLOT3D(ANIMATE(BALL1),
    LIGHT(90,60,0.0,0.6,0.0), AMBIENTLIGHT(0.4,0.4,0.4),
    STYLE(PATCH),COLOUR(ZHUE),TITLE(ROUND_BALLS_ROUND)):
 RBALL2:=PLOT3D(ANIMATE(BALL2),
    LIGHT(90,0,0.9,0,0), AMBIENTLIGHT(0.4,0.4,0.4),STYLE(PATCH),COLOUR(ZHUE)):
>display({RBALL1,RBALL2});
```

まず最初に，一週を２５分割して回転させるため nt=25 とします。次に，先に求めた球の座標データリストに対してＺ軸周りで角度刻み-2 π/nt による回転を与え，これをＸ軸方向に+1.5 平行移動し，そしてさらにＺ軸周りで角度刻み 2 π/nt による回転を与え，これを座標データリスト BALL1 として定義・記憶します。同様にして，Ｚ軸周りで角度刻み-4 π/nt による回転，Ｘ軸方向に-1.5 平行移動，そしてＺ軸周りで角度刻み 2 π/nt による回転を与え，座標データリスト BALL2 として定義・記憶します。これら二つの対象物をもとにして，３次元アニメーションのためのデータリスト RBALL1 と RBALL2 をそれぞれ求めます。最後に，これらをまとめて display により表示すれば完成です。グラフィクスパッケージ plottools にはこの他にも多くの作図のための手続きが用意されています。オンラインマニュアルを利用しサンプルなどでチェックしておくことをお勧めします。

色々と例を上げて Maple によるグラフィックスの導入的な説明をしてきました。これまでに提示してきた資料を，オンラインヘルプ機能などにより補足すれば Maple の機能を使いこなした３次元図形作成が行えるものと考えています。