社会システム分析のための統合化プログラム２２

ー 幾何アニメーション ー

福井正康

福山平成大学経営学部経営学科

概要

我々は教育分野での利用を目的に社会システム分析に用いられる様々な手法を統合化したプログラム College Analysis を作成してきた。今回は図形を描き、動かすことによって、数学的な関数や座標の概念を学ぶことのできる、2 次元と 3 次元の幾何アニメーションについて紹介する。これは、グラフィックを動かして楽しむだけでなく、物理学のシミュレーションにも応用可能で、将来はシミュレーション言語に発展させる予定である。

キーワード

College Analysis，数学，グラフィック，3D，アニメーション

URL:　http://www.heisei-u.ac.jp/ba/fukui/

１．はじめに

これまで我々は、社会システム分析ソフトウェア College Analysis において、統計分析、数学、経営科学、意思決定手法などを中心にプログラムを作成してきたが、今回は数学的な関数や座標の概念をグラフィックの形状や動きによって学ぶことのできる 2 次元と 3 次元の幾何アニメーションについて紹介する。これらは将来、物理シミュレーションと結合させて、環境を含めて描画できる物理シミュレーション言語に発展させる予定である。2 次元のアニメーションは特に高等学校の物理を意識して開発しており、このままでも物理の教材作成に利用できる。3 次元のアニメーションは、空間図形の把握に役に立ち、将来は大学レベルの物理教材の作成に繋げて行く予定である。

幾何アニメーションは簡易的なマクロ言語によって描画されるが、言語仕様の作成には可能な限りの簡潔さを心掛けた。また、コマンド数もサンプルを考えながらできるだけ減らし、中学生程度の知識で、かなりの描画が可能になるように努めた。今後 2 次元、3 次元のアニメーションとも機能向上を図って行くが、マクロ言語の仕様はできるだけ上位互換になるように発展させる予定である。またその変更箇所については、ホームページとレファレンスマニュアルで解説する。

プログラムデータは、これまでの物理シミュレーション用のデータと同様に、1 つのプログラムを 1 ページとして、複数のページに保存する。しかし、これまでは最初に計算して結果を保存し、後で表示していたのに対し、ここでは計算しながら結果を表示する方法を採用している。このようにすることで、過程を記憶させなければ、長時間の連続シミュレーションが可能となり、待ち時間を気にせずに、実行結果を表示させることができる。また、複数のデータページに含まれるプログラムを順番に実行するデモンストレーション機能も加え、作成した成果を時間的な遅延なく、プレゼンテーションできるようにした。

質点の配置など実データを利用する場合もあると考え、予め保存したデータを読み込む機能も加えたが、シミュレーションと結合させる際には、もう少し読み込み方法を改良しなければならない。我々は最初に 3 次元幾何アニメーションに取り組み、その後に 2 次元幾何アニメーションを作成したが、説明は直観的に分かり易い後者を先に行う。また、頁の都合で、機能について同じものは 2 次元アニメーションに出てきた際に説明し、3 次元アニメーションのところでは説明を省略することもある。

２．２次元幾何アニメーション

メニュー［分析－数学－2 次元幾何アニメーション］を選択すると、図 2.1 のような実行メニューが表示される。

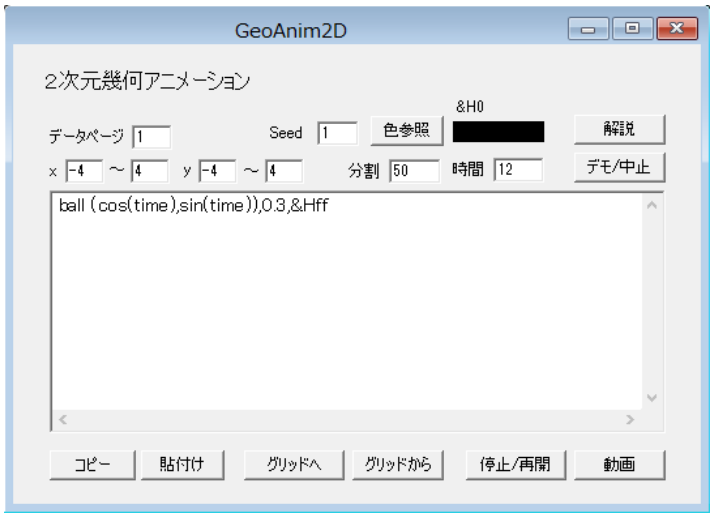

図 2.1　2 次元幾何アニメーション実行画面

デフォルトで描画範囲は、-4≦x≦4, -4≦y≦4 、デモ時間は 1 つのプログラム当たり 12 秒（描画スピードの遅れによりずれることもある）、乱数発生は Seed=1、関数等の描画の分割数は 50 である。

プログラムは中央のテキストボックス内に記述するが、「解説」ボタンをクリックすると、図 2.2 のような、記述法が表示される。

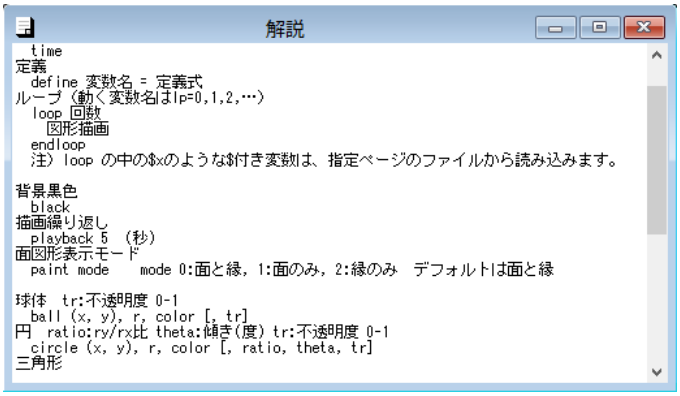

図 2.2　プログラム記述法画面

利用者はこの画面を参照しながらプログラムを作成する。コマンド内での数式の使い方については、メニュー［ヘルプ－数式内利用可能関数］を選択して表示される関数群を参照する。

ここでは各コマンドについて、少し詳しく説明する。コマンドは大文字と小文字を区別しないので、どちらを使ってもよい。また、パラメータ間の空白は、あってもなくてもよい。

描画範囲変更

rangex xmin, xmax　（rangey ymin, ymax）

range min, max
描画範囲を一時的に変更する。メニューの描画範囲を表すテキストボックスは変化しない。2 番目の例のように、range だけだとすべての軸の描画範囲を同一に変更する。

時間変数（time=0, 0.1, 0.2, … 計算機の実行が追いつく限り実時間）
time
時間を表す予約変数で、0, 0.1, 0.2, … （秒）と増加して行くアニメーションの基本となる変数である。描画は 0.1 秒ごとなので、簡単な動画で描画が追いつけば、実時間となる。

定義
define 変数名=定義式
変数名を定義式で置き換えることができる。定義式に乱数が含まれる場合は、描画の度に乱数が発生するので、同じ値は保持しない。プログラムの実行の最初に置換処理を実行するので、描画スピードのロスは少ない。

初期定義
set 変数名=定義式
最初に定義式を計算して、変数名を置き換える。乱数を用いて初期位置を決めることができるので、define が利用できないところで有効である。定義式は数値として評価可能なものに限られる。変数の置換処理はその都度行われるので、これが含まれると実行速度が遅くなる。次で説明するループと連動して使用する。

ループ（ループ変数名は lp=0,1,2,…）
loop 回数
　図形描画
endloop
複数回の処理を連続で行うときに利用する。ループの回数を表す予約変数は「lp」で、これは 0, 1, 2, … ,回数-1 まで変化する。現在のバージョンではループは 1 重までである。1 重ループを多重ループとして機能させるために、整数の割り算の余りを求める演算 % が用意されている。

$付き変数（データページ参照）
$x, $y 等
ループの中の$付き変数は、実行メニュー中のデータページにある（$を取った）同名変数の値を読み込む表記法である。ループの lp 予約変数+1 に相当する行番号のものが利用される。但し、$付き変数がなく、実行メニューの「データページ」が 0 以下ならば、データページは必要ない。

実行の繰り返し
playback 時間（秒）
実行メニューの「動画」ボタンをクリックした際の描画の実行時間を決める。その描画時間が終了したら、再描画が繰り返される。

ペイントモード
paint mode
mode 1：面と縁 2：面のみ 3：縁のみ
以下に述べる、box, circle, tri, poly, param2 コマンドにおいて、面と縁、面のみ、縁のみを表示させるモードを指定する。描画の前に指定し、次に変更して指定されるまでは、そのままの状態になっている。デフォルトは面と縁である。

描画開始・終了時間
starttime 時間（秒）
stoptime 時間（秒）

描画コマンドの開始時間と終了時間を設定する。複雑な描画では時間のずれが生じる。一度設定すると、その情報はその後の描画で保持される。

グリッド

grid [widths, widthl]

小さな間隔を widths、大きな間隔を widthl として、グリッドを引く。図を描くガイド用として利用し、後で消すことを想定している。

背景黒色

black

これは描画の背景を黒色にするコマンドである。

球体

ball (x, y), r, color [, tr]

座標(x,y)を中心にした半径 r の立体的な球を描画する。色を指定する color は&Hffaa80 のように 16 進数で表示するか、10 進数で表示する。値がよく分からない場合は、実行メニューの「色参照」ボタンで調べることができる。tr は不透明度（0～1）を与える。tr は省略可能でデフォルトは 1（不透明）である。color と tr については、他でも同じように使用する。一般に、[　] 内のパラメータは省略可能である。

長方形

box (x, y), wx, wy, color [, theta, tr]

座標(x,y)を中心にした、横幅 wx、高さ wy、色 color の直方体を描く。theta は図形中央を中心とした回転角（単位は度）で、デフォルトは 0 度、tr は不透明度である。

正 n 角形

poly (x, y), r, n, color [, theta, tr]

座標(x,y)を中心にした、半径 r、色 color の n 角形を描く。theta は回転角（度）、tr は不透明度である。

円・楕円

circle (x, y), r, color [, ratio, theta, tr]

座標(x,y)を中心にした、横半径 r、色 color、縦横比 ratio の楕円を描く。縦横比は、縦／横で与え、デフォルトは 1 である。theta は回転角（度）、tr は不透明度である。

三角形

tri (x1, y1)-(x2, y2)-(x3, y3), color [, tr]

3 点の座標(x1, y1), (x2, y2), (x3,y3) を繋ぐ、色 color の 3 角形を描く。tr は不透明度である。

連結

connect (x1, y1)-(x2, y2), color [, mode, r, dr]

mode 1-2：線，3-4：矢印，5-6：バネ，7-8：抵抗，9-10：波線，11-12：点線，13-14 点矢印，15-16：バネ（持ち手なし），17-18：コイル，19-20：コイル半分

21-22：左半円，22-23：右半円，31-32:左半波円，33-34：右半波円

41-42：左半 gluon，43-44：右半 gluon　　注）偶数は太線

2 点の座標(x1,y1), (x2,y2)を繋ぐ、色 color の、線分、矢印、バネ等を描く。描画の mode は上に書かれた通りである。r は、バネ/コイルの場合は半径、波線/抵抗の場合は厚みである。dr は、バネ/コイルの場合は巻き数、波線/抵抗の場合は山と谷数、矢印の場合は矢印の位置（始点 0 終点 1）である。

関数

func y=f(x), color [, mode, min, max, div]

func f(x), color [, mode, min, max, div]
mode 1：細線 2：太線
色 color の y=f(x)の関数形を描く。x=f(y)の関数形を描くこともできる。mode は上に示した通りである。min と max は関数を描く領域の最小と最大で、デフォルトは実行メニューで定めた描画範囲である。div は描画間隔を決めるための分割数で、デフォルトは実行メニューで定めた分割数である。

関数値による色付け

colorz f(x, y) [, mode, xmin, ymin, xmax, ymax, div, tr]
関数 f(x,y) の大きさにより、領域に虹色の色付けを行う。最小が紫、最大が赤である。mode=1 は 1 回の描画の値による描画、mode=2 はそれまでの最大値を保持させるものとする。後者は、例えば波の干渉などで、振幅の大きな領域を見つけるのに役に立つ。デフォルトは mode=1 である。xmin, ymin, xmax, ymax は描画領域（固定）、div はその領域の x,y 方向の分割数である。

パラメータ曲線［0<=u<=1］

param1 (x(u), y(u)), mode, color [, div, r]
mode 1：細線 2：太線 3：矢印 4：太矢印
予約パラメータ「u」で指定するパラメータ関数を色 color で描く。パラメータが動く範囲は 0≦u≦1 である。mode は上に示した通りである。矢印は u=1 の側に付く。r は矢印の長さである。

パラメータ面［0<=u,v<=1］

param2 (x(u,v), y(u,v)), color [,div1, div2, tr]
2 つの予約パラメータ「u, v」で指定するパラメータ関数を色 color で描く。パラメータ u, v は 0 から 1 までの間を分割数 div1 と div2 で動くものとする。tr は不透明度である。通常は 3 次元空間内に描くものだが、ここでは 2 次元平面内で面を描くのに利用する。

陰関数

impfunc f(x,y), color [, mode, div]
mode 1：細線 2：太線
f(x,y)=0 で与えられる陰関数のグラフを色 color で描く。mode は上に示した通りである。div は表示領域の x,y 方向をいくつに分割して描画するかを決める分割数である。

クリップボード中の画像貼り付け

clip (x, y), wx [, theta]
クリップボードにある画像を、座標(x,y)を中心に、wx を横幅として貼り付ける。その際、画像の縦横比は保持する。theta は画像の回転角度（度）である。

文字列

string (x, y), ”文字列”, color [, size]
座標(x, y)の位置に指定された文字列（” ” でくくる）を色 color で描く。文字のフォントサイズは size で指定するが、デフォルトは 12 point である。string の代わりに print も使える。

コマンド解説の中でも述べたが、描く図形の色は RGB の 16 進表示（10 進表示でもよい）で指定する。なじみの薄い利用者もいるので、実行メニューに「色参照」ボタンを加え、色を選びながら、数字を指定できるようにしている。描く図形によって図形表示の最大数が決まっており、指定した数が多い場合は、例えば「ball 数<=最大数」のようにエラーメッセージが表示される。これを変更する場合は、例えば「maxball 20000」のように、max の後に描画コマンド名を続けて、必要な数値を書く。図形描画コマンドのパラメータは、時間変数 time や繰り返し変数 lp などを使って帰ることがで

きるが、数値で表し、固定値として扱うものもある。例えば、描画モードを表す mode や分割数を表す div、colorz コマンドの描画領域などである。また、図形描画以外のコマンドのパラメータは殆ど固定である。

以後は例を用いてコマンドの利用法を説明する。行の最初の数字は説明用の行番号であり、実際には入力しない。図は通常正方形の描画領域内に表示されるが、画面の都合により、必要部分を切り取って示す。

例１　バネ

```
1. box (0,3.5),6,1,&H804000
2. define y=sin(2*time)
3. connect (0,3)-(0,y),&Hff0000,5
4. ball (0,y-0.2),0.3,&Hff
```

1 行目はバネをつるす天井を描いている。(0,3.5)を中心として、幅 6、高さ 1 の直方体を色番号&H804000 で描く。2 行目は運動するおもりの位置を y として与える定義文である。運動は位相 θ =2t（秒）で与えられる。3 行目は、コマンド connect の mode が 5 で、赤色のバネを表している。4 行目は中心を y-0.2 とした、青色のボールを表している。図 2.3 に描画結果を示す。

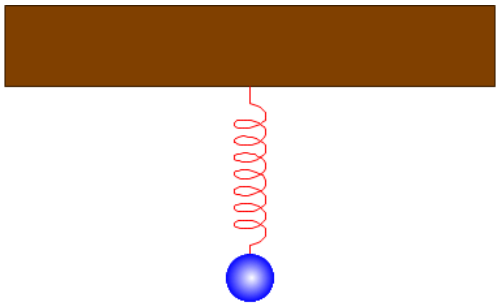

図 2.3　バネ

例２　ボックスとボールの回転

```
1.    circle (0,0),2,&Hff,0.5,90*time
2.    box (0,0),1,0.5,&Hff00,90*time
```

1 行目は座標(0, 0)を中心とした長半径 2、縦横比率 0.5 の楕円を表す。楕円は 1 秒間に 90 度の角度で回転をさせている。但し、パソコンの描画速度によって遅れる場合もある。2 行目は(0, 0)を中心とした、幅 1、高さ 0.5 の緑色の長方形を描く。長方形は楕円と同じ速さで回転させている。描画はプログラムの順番に描かれるので、circle の上に box が描かれている。図 2.4 に描画結果を示す。

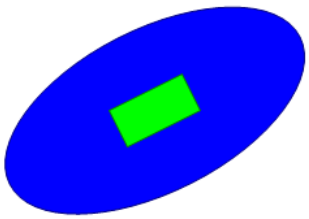

図 2.4　ボックスとボールの回転

例３　多角形の回転

```
1.  poly (-1,0),2,6,&Hff,-90*time,0.2
2.  poly (1,0),1.5,5,&Hff00,90*time,0.2
```

1 行目は半径 2 の青色で半透明な（不透明度 0.2）六角形を描くコマンドで、2 行目は半径 1.5 の緑色で半透明な（不透明度 0.2）五角形を描くコマンドである。2 つの多角形の回転速度は 1 秒間に 90 度で、逆方向に回っている。図 2.5 に描画結果を示す。

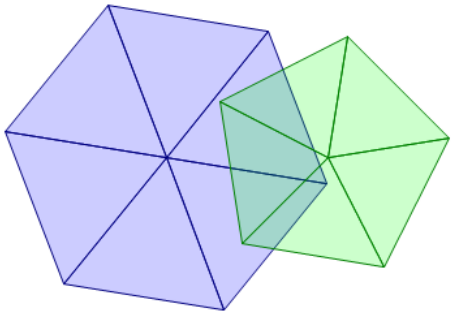

図 2.5　多角形の回転

例４　円運動

```
1.  define x=3*cos(time)
2.  define y=3*sin(time)
3.  ball (x,y),0.2,&Hff
4.  connect (x,y)-(0,0),&H606060,1
5.  connect (x,y)-(2*x/3,2*y/3),&H8080,4
6.  connect(x,y)-(x-y/3,y+x/3),&Hff0000,4
```

1,2 行目は円運動の位置の定義文である。3 行目はその位置に青色のボールを描くコマンドである。4 行目では、connect コマンドの mode=1 を使って、中心とボールを繋ぐ線、5 行目は加速度の方向、6 行目は速度の方向を mode=4 の太矢印で描画している。図 2.6 に描画結果を示す。

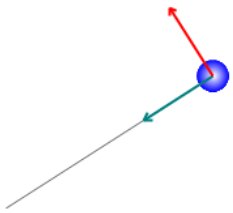

図 2.6　円運動

例５　坂と滑車

```
1.  playback 3
2.  define l=(time/1.5)^2
3.  define x1=-2.2+l*cosd(30)
4.  define y1=-2.11+l*sind(30)
5.  define x2=3.2
6.  define y2=-0.5-l
7.  connect (x1,y1)-(3,1/3^0.5+0.4),&H404040,1
8.  connect (x2,y2)-(3.2,1/3^0.5+0.2),&H404040,1
9.  tri (-3.2,-3)-(2.8,-3)-(2.8,-3+6*tand(30)),&H804000
10. box(x1,y1),0.7,0.5,&Hff0000,30
11. box(x2,y2),0.7,0.5,&H800000
12. circle(3,1/3^0.5+0.2),0.2,&Hff8040
13. connect (3,1/3^0.5+0.2)-(2.8,-3+6*tand(30)),&Haa0000,2
```

1 行目では、描画の繰り返し時間を 3 秒に設定している。3,4,9,13 行目の cosd(), sind(), tand()関数は、引数を度単位で表した三角関数である。9 行目の tri は台の三角形を描くコマンドである。図 2.7 に描画結果を示す。

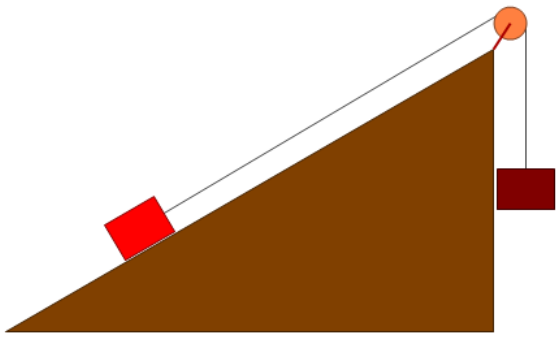

図 2.7　坂と滑車

例６　クリップボードからの描画

```
1.  loop 10
2.  define x=2*cos(vth(time-(10-lp)/2.5))
3.  define y=2*sin(vth(time-(10-lp)/2.5))
4.  clip (x,y),2-(10-lp)*0.1
5.  endloop
```

1行目から5行目のloop 10 ～ endloopはその間の描画を10回繰り返すコマンドである。その間、予約変数 lp は 0 から 9 まで値が動く。define はループの間でも前後でも同じ働きをする。4 行目の clip はクリップボードにコピーした画像を表示するコマンドである。大きさは描画幅で指定するが、縦横比は保持される。2,3 行目で使われた vth()は引数が負の時は 0、正の時はその値となる関数である。必要に応じて図を回転させることもできる。図 2.8 に描画結果を示す。

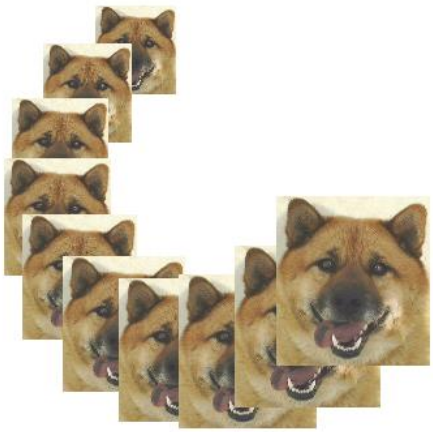

図 2.8　クリップボードからの描画

例７　定常波

```
1.  define y1=0.5*sin(2*pi*(time/5-x))
2.  define y2=-0.5*sin(2*pi*(time/5-3+x))
3.  func y1,&Hff,,-3,3,100
4.  func y2,&Hff00,,-3,3,100
5.  func y1+y2,&Hff0000,2,-3,3,100
6.  connect (-0.5,1.5)-(0.5,1.5),&Hff,4
7.  connect (0.5,-1.5)-(-0.5,-1.5),&Hff00,4
8.  box (-3.2,0),0.4,2,&H804040
9.  box (3.2,0),0.4,2,&H804040
```

この例の 3,4,5 行の func は、y=f(x)または x=f(y)の形の関数を描画するコマンドである。この場合は y1,y2 の中に変数 x が含まれているため、y=f(x)の描画となる。描画の mode を指定しない場合は mode=1 となり細線、2 の場合は太線となる。ここでは 5 行目で記述された赤色の定常波が太線である。図 2.9 に描画結果を示す。

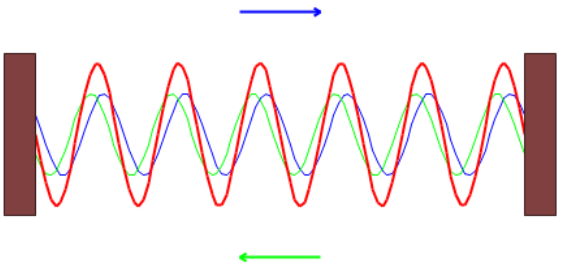

図 2.9　定常波

例８　超音速

```
1.  playback 15
2.  define x1=(lp*0.04)^2-2.5
3.  define x2=(0.2*time)^2-2.5
4.  paint 3
5.  loop 100
6.  circle (x1,0),vth(time/2-lp/10), &Hff
7.  end loop
8.  paint 1
9.  tri (x2,0)-(x2-0.2,0.1)-(x2-0.2,-0.1),&Hff0000
```

これは音速を超える際の音波の伝搬を描画したものである。1 行目は 15 秒ごとに再描画するコマンドである。4 行目と 8 行目の paint は、それぞれ描画を縁のみと縁と面にするコマンドである。図 2.10 に描画結果を示す。

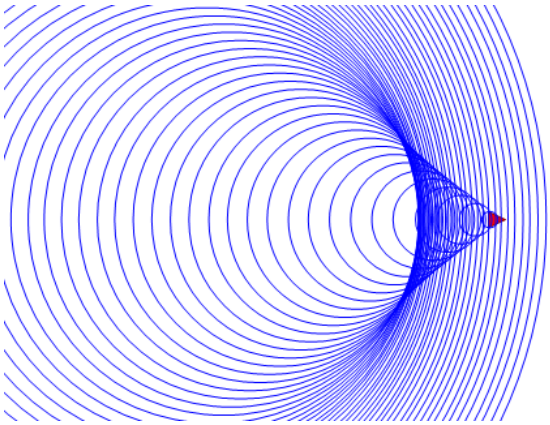

図 2.10　超音速

例９　リサジュー図形

```
1.  define x1=2*cos(2*pi*u*0.9+time)
2.  define y1=sin(4*pi*u*0.9-pi/4+time)
```

3. param1 (x1,y1),rainbow(u),4,100

これは時間とともに動くリサジュー図形の例である。3 行目の param1 は 2 次元のパラメータ関数を表すコマンドである。パラメータは、予約変数として u が使われ、0≦u≦1 の間の値を取る。パラメータ u は通常、実行メニューの分割数で指定された数で分割され描画されるが、描画を滑らかにする場合など、3 行目の最後の 100 のように、分割数を増やすこともできる。3 行目の rainbow()関数は引数の値が 0 から 1 の範囲で、紫色から赤色の虹色の数値を与える関数である。また、mode の値は太矢印の mode=4 が与えられている。図 2.11 に描画結果を示す。

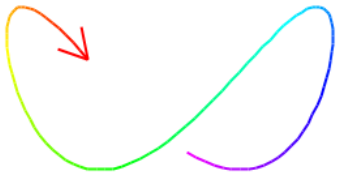

図 2.11　リサジュー図形

例 10　等電位面

```
1.  define x1=2*sin(time)
2.  define x2=-2*sin(time)
3.  define y1=2*cos(time)
4.  define y2=-2*cos(time)
5.  define r1=((x+x1)^2+y^2)^0.5
6.  define r2=((x+x2)^2+y^2)^0.5
7.  define r3=(x^2+(y+y1)^2)^0.5
8.  define r4=(x^2+(y+y2)^2)^0.5
9.  define p=1/(r1+0.001)+1/(r2+0.001)+1/(r3+0.001)+1/(r4+0.001)
10. loop 3
11. impfunc p-1.1*lp-2,&Hff,2,100
12. end loop
13. ball(x1,0),0.2,&Hff00
14. ball(x2,0),0.2,&Hff0000
15. ball(0,y1),0.2,&Hffff
16. ball(0,y2),0.2,&Hffff00
```

ここでは 4 つの等しい電荷が作る等電位面を陰関数を用いて描いている。11 行目の impfunc は f(x,y)=0 で表される図形を描画する。ここで f(x,y)に相当する p-1.1*lp-2 の p は 9 行目に定義されている。また impfunc コマンドで描画を滑らかにするために、領域を x,y 方向で 100 分割にしている。図 2.12 に描画結果を示す。

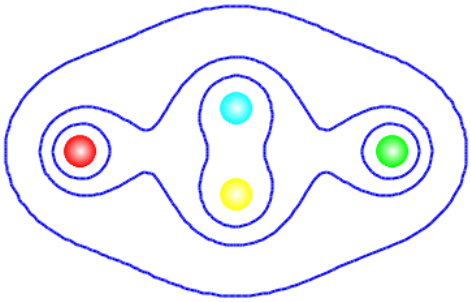

図 2.12　等電位面

例 11　重ね合わせの原理

```
playback 10
maxcircle=200
define d=1
define f=2
define r1=((x+d)^2+y^2)^0.5
define r2=((x-d)^2+y^2)^0.5
define z1=sin(2*pi*f*vth(time-r1))+sin(2*pi*f*vth(time-r2))
colorz z1+z2,2,,,,,101
loop 40
paint 3
circle(d,0),vth(time-lp/f-0.25/f),&H0
circle(-d,0),vth(time-lp/f-0.25/f),&H0
endloop
```

2 点から発せられる円形波の干渉縞を振幅の大きい部分と小さい部分で塗り分けた図である。波の重なった部分は振幅が大きく、赤くなっている。8 行目の colorz が関数の値によって領域を塗り分けるコマンドである。この場合は過去の最大振幅を記憶する mode=2 となっている。また、分割は結果をきれいに表示するために、101 とデフォルトより多くしている。7 行目によると、この波は速さ 1、周期 0.5、速さ×周期である波長が 0.5、円形波の中心間の距離が 2 の設定になっており、波の線は 11,12 行目から 1 波長ごとに描かれている。図 2.13 に描画結果を示す。

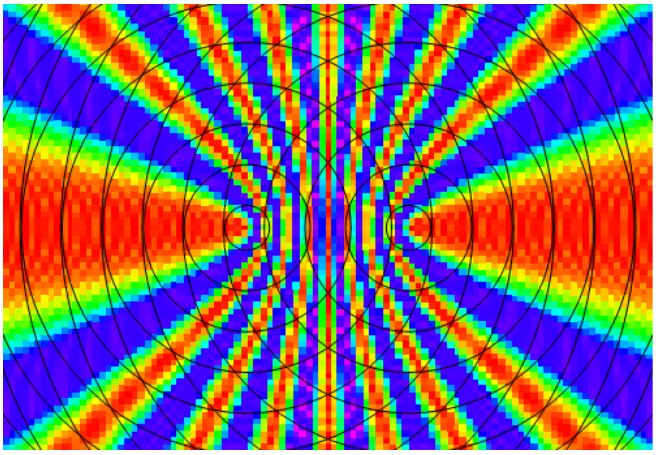

図 2.13　干渉縞

３．３次元幾何アニメーション

これまで 3 次元物理シミュレーションで、質点系や静電磁場の問題を扱ってきたが、その際に環境を含めて描くシミュレーションを目指すことにした。ここではその準備のために種々の描画に必要な要素を開発している。シミュレーションとの連携は今後の課題であるが、この図形の描画法だけでも数学関数や空間座標概念の把握に役に立つ。

メニュー［分析－数学－3 次元幾何アニメーション］を選択すると図 3.1 のような実行メニューが表示される。

図 3.1　実行メニュー

利用法は 2 次元幾何アニメーションとほぼ同じである。コマンドについて、基本的な事項は 2 次元

のコマンドと同様であるので、重複する部分は簡単な説明に留める。結果はこれまでに開発した 3D ビューアに表示する[1), 2)]。以下のコマンドについて、利用法は 2 次元の場合と同じである。

描画範囲変更
rangex xmin, xmax （rangey ymin, ymax / rangez zmin,zmax）
range min, max

時間変数（time=0, 0.1, 0.2, … 計算機が追いつく限り実時間）
time

定義
define 変数名=定義式

初期定義
set 変数名=定義式

ループ（ループ変数名は lp=0,1,2,…）
loop 回数
　図形描画
endloop

$付き変数（データページ参照）
$x, $y 等

実行の繰り返し
playback 時間（秒）

ペイントモード
paint mode
mode 1：面と縁 2：面のみ 3：縁のみ

描画開始・終了時間
starttime 時間（秒）
stoptime 時間（秒）

背景黒色
black

これ以後のコマンドは 2 次元の場合とは異なるので、少し詳しく説明する。また 2 次元の場合と同じく、コマンド等は大文字と小文字を区別しない。

切り取りモード
cutmode
これは図形がある距離まで近づくと、近づいた部分の表示が消えるモードで、内部構造などを見るときに利用する。3D ビューアの機能によって、実行中に設定することもできる。

座標軸表示

axis

これは実行メニューで与えられる描画領域を囲む座標軸を描くコマンドである。

初期角度

angle 角度（度）

これは図形が最初に表示されるときの見る角度を指定するコマンドである。0（度）の場合は正面から、90（度）の場合は真上から見た図になる。

小さな球体

ball (x, y, z), r, color [, tr]

座標(x,y,z)を中心にした、半径 r、色 color の、立体的に見える球体を描く。不透明度 tr を指定すると、立体表示は無くなり通常の半透明な球になる。デフォルトは不透明な立体的球体である。

大きな球体（分割は指定分割）

spher (x, y, z), r, color [, theta, phi, rot, tdiv, pdiv, tr]

座標(x, y, z)を中心とした、半径 r、色 color の球体を描く。その際、中心軸の z 軸からの傾きを theta（度）、軸の旋回角度を phi（度）とし、軸の周りの回転角度を rot（度）とする。また tdiv と pdiv はそれぞれ theta 方向と phi 方向の分割数、tr は不透明度である。デフォルトは theta=0, phi=0, rot=0, tr=1 であり、tdiv と pdiv は実行メニューに与えられた分割数である。これらの分割数を少なくすることによって多面体を表現することもできる。回転を分かり易くするために、予約変数「phi」を利用した経線方向の色付けが可能である。

三角形

tri (x1, y1, z1)-(x2, y2, z2)-(x3, y3, z3), color [, tr]

3 点の座標(x1, y1, z1), (x2, y2, z2), (x3, y3, z3)を繋ぐ、色 color の三角形を描く。tr は不透明度である。

正 n 多角形

poly (x, y, z), r, n, color [, theta, phi, rot, rdiv, tr]

座標(x,y,z)を中心にした、半径 r、色 color の正 n 角形を描く。正 n 角形と垂直な軸の z 軸からの傾きを theta（度）、軸の旋回角度を phi（度）とし、軸の周りの回転角度を rot（度）とする。回転を分かり易くするために、予約変数「phi」を利用した経線方向の色付けが可能である。rdiv はデフォルトが 1 の半径方向の分割数、tr は不透明度である。

小さな直方体（描画に多少の乱れが生じる）

box (x, y, z), wx, wy, wz, color [, theta, phi, rot, tr]

座標(x,y,z)を中心にした、x 方向 wx、y 方向 wy、z 方向 wz の大きさ、色 color の直方体を描く。z 軸との傾きを theta（度）、軸の旋回角度を phi（度）とし、軸の周りの回転角度を rot（度）とする。

連結

connect (x1, y1, z1)-(x2, y2, z2), color [, mode, r]

種類 1-2：線 3-4：矢印 5-6：バネ　　注）偶数は太線

2 点の座標(x1, y1, z1), (x2, y2, z2)を繋ぐ、色 color の、線分、矢印、バネ等を描く。描画の mode は上に書かれた通りである。r はバネを描いた場合の半径である。

床・天井・平面波

func z=f(x, y), color [, min1, min2, max1, max2, div, tr]

func f(x, y), color [, min1, min2, max1, max2, div, tr]

色 color の z=f(x, y)の関数形を描く。x=f(y, z), y=f(z, x) の関数形を描くこともできる。描画領域は、座標(min1,min2)と(max1,max2) を対角線とする四角形である。分割は 1 方向当たり div で指定する。

デフォルトは実行メニューの分割数である。

パラメータ曲線［0<=u<=1］

param1 (x(u), y(u), z(u)), mode, color [, div, r]

mode 1-2：太線 3-4：矢印　　注）偶数は太線

予約パラメータ「u」で指定するパラメータ関数を色 color で描く。パラメータ u は 0 から 1 までの間を分割数 div で動くものとする。描画 mode は上に示した通りである。r は矢印の長さである。

パラメータ曲面［0<=u,v<=1］

param2 (x(u,v), y(u,v), z(u,v)), color [, div1, div2, tr]

2 つの予約パラメータ「u,v」で指定するパラメータ関数を色 color で描く。パラメータ u,v は 0 から 1 までの間を分割数 div1 と div2 で動くものとする。tr は不透明度である。

文字列

string (x, y, z), ”文字列”, color [, size]

座標(x, y, z)の位置に指定された文字列（” ” でくくる）を色 color で描く。文字のフォントサイズは size で指定するが、デフォルトは 12（point）である。string の代わりに print も使える。

以後は 2 次元の場合と同様、例を用いてコマンドの利用法を説明する。行の最初の数字は説明用の行番号であり、実際には入力しない。図は通常正方形の描画領域内に表示されるが、印字の都合により、必要部分を切り取って示す。

例 1　銀河のような形状

```
1.  black
2.  ball(0,0,0),0.3,&Hffffff,0.5
3.  loop 800
4.  define x=$r*cos(time/($r+1)+$a)
5.  define y=$r*sin(time/($r+1)+$a)
6.  ball(x,y,$z),0.05,&H8888ff
7.  endloop
```

1 行目では背景を黒色にし、2 行目は原点に白い球形を描いている。3 行目から 7 行目は繰り返しで、800 個の小さな球形を描く。4,5 行目は、loop の前にあっても中にあっても、機能は同じである。$r, $a には実行メニューのデータページにある r, a 変数のデータの値をループ予約変数 lp の値に応じて、lp+1 行目から読み出して行く。動きは time 変数で与える。データさえ与えておけば星の数は 10000 個程度でも問題なく動く。図 3.2a に描画結果を示す。

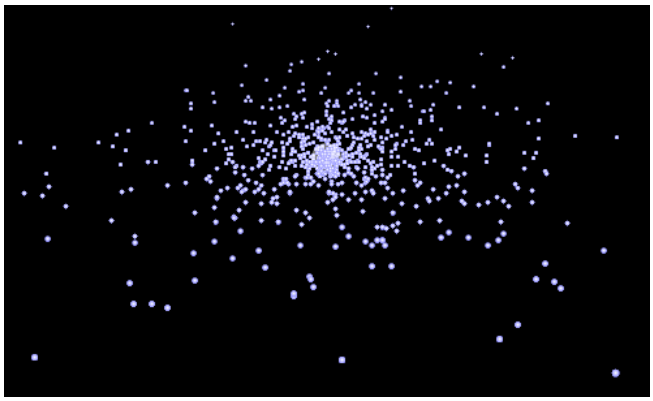

図 3.2a　銀河のような形状

これは力学的法則を用いてシミュレーションを実行しているわけではない。あくまで単なる点の回転である。またデータページにデータを与えない限り、サンプルとして実行することはできない。

同様な処理を ball の数を増やして、データページを使わずに示しておく。これならデータなしにすぐに実行できる。

```
1.  black
2.  angle 10
3.  ball(0,0,0),0.3,&Hffffff,0.5
4.  set r=3*abs(nrnd)
5.  set a=2*pi*rnd
6.  set z1=rnd-0.5
7.  loop 9000
8.  define x=r*cos(time/(r+1)+a)
9.  define y=r*sin(time/(r+1)+a)
10. define z=exp(-r^2)*z1
11. ball(x,y,z),0.03,&Haaaaff
12. endloop
```

ここでは ball の数を 9000 個にしている。4, 5, 6 行目の set コマンドは、要素 1 個 1 個の定義データ（計算式）を作る際に乱数に予め数値を割り当て、描画の段階ではその数値を利用する方法で描画する。描画スピードは落ちるが、データページを使わず似た効果を出せる利点がある。2 行目の angle コマンドでは、最初の見る角度を水平から 10 度に設定している。図 3.2b に描画結果を示す。

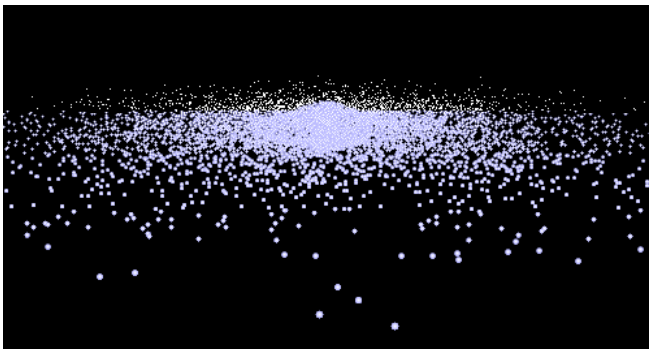

図 3.2b　データページを使わない銀河のような形状

例２　2 つの球体

```
1.  define col1=&Hffff*theta(phi-pi*0.01)+&Hff*theta(pi*0.01-phi)
2.  define col2=&Hffff00*theta(phi-pi*0.01)+&Hff00*theta(pi*0.01-phi)
3.  spher(2.5*cos(time),2.5*sin(time),0),2,col1,,,90*time,20,,0.25
4.  spher(2.5*cos(time),2.5*sin(time),0),0.5,&Hff0000,,,90*time,20,,0.5
5.  spher(2.5*cos(time+pi),2.5*sin(time+pi),0),1,col2,-30,,-180*time,20
```

ここでは、spher コマンドを使って 2 つの球体を描いているが、大きい方の中にはもう 1 つ球体がある。5 行目の小さい方の球体は軸を 30 度傾けている。軸方向の回転速度は大きい球体で 90 度／秒、小さい球体で、-180 度／秒である。分割数は、theta 方向を 20、phi 方向はデフォルトの 30 に設定している。また回転が分かるように予約変数 phi が 0 の近傍で色を変えるようになっている。1,2 行目の theta()は引数が 0 以上の時に 1、負のときに 0 となる関数である。図 3.3 に描画結果を示す。

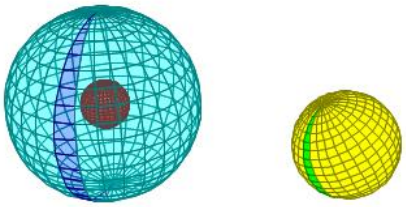

図 3.3　2 つの球形

例３ 波とプール

```
1.  playback 10
2.  define z1=0.2*sin((x^2+y^2)^0.5*3-4*time)
3.  define def=theta(4*time-(x^2+y^2)^0.5*3)
4.  func z=z1*def,&Hffcc,-5,-5,5,5,,0.5
```

```
5.  define dz=0.2*sin(-4*time)
6.  ball(0,0,dz),0.3,&Hff
7.  connect (0,0,3)-(0,0,dz+0.2),&Hff0000,5
8.  box (0,0,0),10,10,5,&Hffff,,,,0.1
```

これは円形波の例で、3 行目で定義された変数 def で最初の波の到達を与える。波の形は 4 行目の func コマンドで、z1 の中に x と y が time 変数と共に含まれることで与えられる。波の描画範囲は x,y の範囲として、(-5,-5)と(5,5)を対角とする正方形で与えられる。6 行目でボールを描き、7 行目で connect コマンドを用いてバネを描いている。8 行目はプール自体を半透明に描いている。図 3.4 に描画結果を示す。

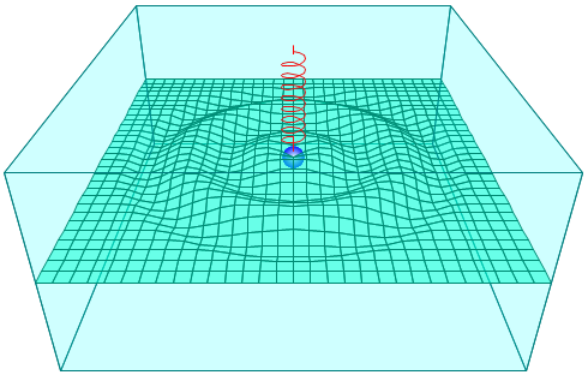

図 3.4　波とプール

例 4　力学（バネ）

```
1.  axis
2.  define z=2*sin(2*time)
3.  ball(0,0,z),0.5,&Hff00
4.  connect (0,0,4)-(0,0,z+0.51),&H8888,5
5.  connect (0,-0.2.,z)-(0,-0.2,z-2),&Hff,4
6.  connect (0,-0.2.,z)-(0,-0.2,2),&Hff0000,4
7.  connect (1,-0.2,z)-(1,-0.2,0),&Hff00ff,4
8.  string (1.7,-0.2,z/2), "合力",&Hff00ff,20
```

これは今後応用して行く物理シミュレーション用に作られたものである。バネとおもりの力の向きをベクトルで表し、その中に文字を入れている。また、動きが分かり易いように、1 行目の axis コマンドで座標軸も加えている。8 行目の string コマンドは文字列を描くコマンドで、フォントサイズは 20 であるが、表示の大きさは文字の位置により変わってくる。図 3.5 に描画結果を示す。

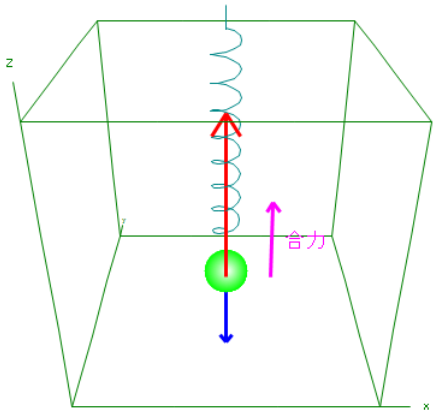

図 3.5　力学（バネ）

例５　斜面

```
1.  playback 3.5
2.  func z=tan(pi/6)*x,&Hccffff,-10,-10,10,10
3.  define x1=-time^2+4
4.  define z1=x1*tan(pi/6)+0.5
5.  box (x1,-3,z1),1,1,0.5,&Hff0000,-30,0,-180*time
6.  ball (x1,0,z1),0.4,&H8800
7.  box (x1,3,z1+0.1),1,2,0.5,&Hff,-30,0,90*time
```

これは 2 つのボックスとボールが斜面を回転しながら滑り落ちるプログラムで、2 行目で斜面を与え、3, 4 行目でボックスとボールの位置を定義している。それぞれの動きは、5, 6, 7 行目で与えるが、ボックスにはそれぞれ-180, 90（度／秒）の回転を与えている。図 3.6 に描画結果を示す。

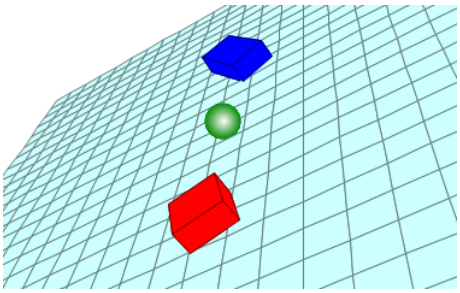

図 3.6　斜面

例６　矢印

```
1.  define x1=3*cos(2*pi*u+2*time)
2.  define x2=3*sin(2*pi*u+2*time)
3.  define x3=4*u-2+sin(2*time)
4.  param1 (x1,x2,x3),rainbow(u),4
5.  param1 (x2,x3,x1),rainbow(u),4
6.  param1 (x3,x1,x2),rainbow(u),4
```

これは param1 コマンドを使った矢印の動きを表す例で、1,2,3 行目で与えた定義により、4,5,6 行目で座標の役割をサイクリックに変えて表示したものである。色は rainbow 関数で予約パラメータ u の値に応じて付けられている。また、param1 コマンドの mode は矢印・太線を与える 4 である。図 3.7 に描画結果を示す。

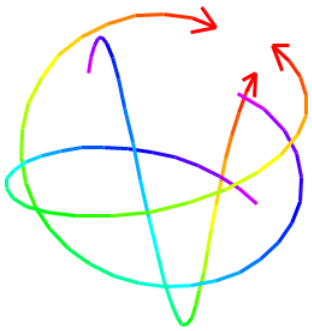

図 3.7　矢印

例 7　トーラス

```
1.  define x=(3+1.5*sin(2*pi*u))*cos(2*pi*v)
2.  define y=(3+1.5*sin(2*pi*u))*sin(2*pi*v)
3.  define z=1.5*cos(2*pi*u)+2*sin(sin(time)*u)
4.  param2(x,y,z),rainbow(u)
```

これは param2 コマンドの使用例で、3 次元パラメータ表示のトーラスに少し時間的な動きを加えたものである。3 行目の第 2 項によって動きが得られる。図 3.8 に描画結果を示す。

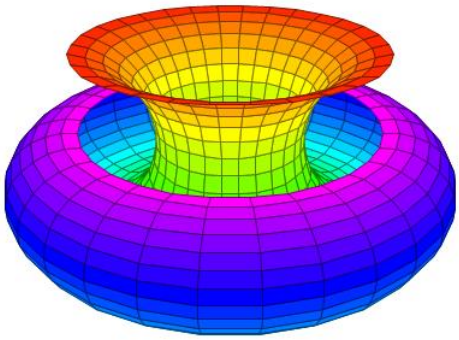

図 3.8　トーラス

例 8　コマ

```
1.  define col=&Hff*theta(phi-175)+&Hffff*theta(175-phi)
2.  define x1=2*sind(30)*cosd(90*time)
3.  define y1=2*sind(30)*sind(90*time)
4.  define z1=2*cosd(30)-2
5.  define x2=4*sind(30)*cosd(90*time)
```

```
6.  define y2=4*sind(30)*sind(90*time)
7.  define z2=4*cosd(30)-2
8.  define th=30
9.  define alpha=-(90*time) mod 360
10. poly (x1,y1,z1),3,6,col, th, alpha, 270*time, 3
11. connect (x1,y1,z1)-(x2,y2,z2), &Hff0000,2
12. connect (x1,y1,z1)-(0,0,-2), &Hff0000,2
13. connect (0,0,-2)-(0,0,-4),&H6600,2
14. poly (0,0,-4),2,30,&Hff00, , , , 6
```

これは支柱の上で回転する 6 角形のコマを表現している。10 行目の poly は 6 角形のコマを描画するコマンドで、軸の傾きが th、軸の旋回角度が alpha である。これらはそれぞれ 8 行目と 9 行目で定義されている。コマ自身の回転は軸の周りに 270 度／秒であり、コマの半径方向の分割数は 3 となっている。13, 14 行目が台座の描画である。図 3.9 に描画結果を示す。

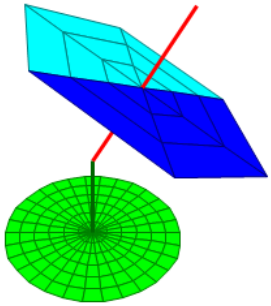

図 3.9　コマ

例９　スイカ

```
1.  paint 2
2.  cutmode
3.  angle 0
4.  define y1=2-5.5*abs(sin(time))
5.  define col=&Hff00*(phi % 36)/18)
6.  spher (0,y1,0), 3, col, , , , , 20
7.  spher (0,y1,0), 2.9, &Hffffff, , , , , 20
8.  spher (0,y1,0), 2.5, &Hff0000, , , , , 20
9.  loop 100
10. define x=$r2*sin($theta2)*cos($phi2)
11. define y=2+$r2*sin($theta2)*sin($phi2)
12. define z=$r2*cos($theta2)
13. define y2=y-5.5*abs(sin(time))
14. ball (x,y2,z),0.1,&H0,1
15. endloop
```

これはスイカが近づいたり遠ざかったりする描画である。2 行目の cutmode コマンドで、近づいた際、中が見えるようになっている。スイカらしくするために、1 行目の paint 2 コマンドでフレームを表示しないようにしている。また、cutmode を効果的にするために 3 行目の angle 0 で、最初の見

る角度を水平面からにしている。スイカの位置は、4 行目で y 方向に振動させている。また色の区分けは 5 行目で予約変数 phi の値が飛び飛びになることを利用して付けられている。図 3.10 に描画結果を示す。

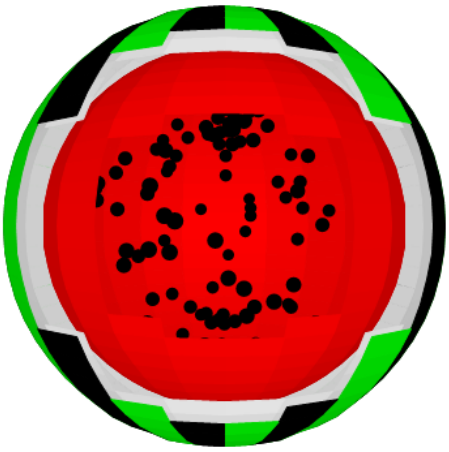

図 3.10　スイカ

４．おわりに

我々はいくつかの例を用いて、2 次元及び 3 次元幾何シミュレーションのプログラムを説明してきたが、ここでの使用例はほんの一例であり、応用範囲は広い。数学の関数や座標概念の把握、1 次変換やベクトルについての学習、物理の教材作り、簡単な物理シミュレーション、さらに将来的には本格的な物理シミュレーションと可能性は大きく広がっている。

我々はマクロ言語の作成原理を、次のように考えている。コマンドのオプション部分（コマンド一覧の[ ]で囲まれている部分）を除いても一応の作図ができること。コマンドはできるだけ簡単にし、コマンド間に関連を持たせること。例えば、ball コマンドと circle コマンドの半径や色など、同じ順番に並べて、関連付けている。2 次元アニメーションと 3 次元アニメーションの間でも、コマンドやパラメータに関連性を持たせている。

コマンドを細かく複雑にすることは、それほど難しいことではない。一番難しいのは、使い勝手と機能とのバランスである。例えば、circle コマンドに開始角度とスイープ角度の機能を持たせるかどうか現在も迷っている。これは param2 コマンドで代用することもできるので、組み込まないでおくこともできる。使用頻度と使い勝手のことを考えて決めるべきであろう。

この論文の副題中の「幾何」とは直接関係がないが、我々は図形に画像をマッピングする機能を 3 次元の spher 命令と param2 命令に実装してみた。これは、予めファイルを図 4.1 の画像読み込み機能で読み込んでおき、これらの命令の color 指定の場所で、例えば sphere (0,0,0), 1, image1 のよう

に、「image」と番号で画像を指定することで利用可能となる。描画の例を図 4.2 に示す。

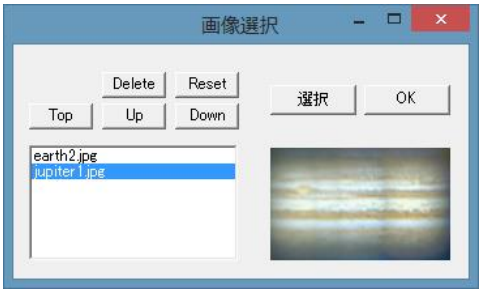

図 4.1　マッピング用画像

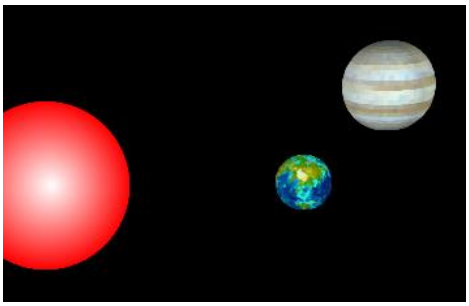

図 4.2　描画例

きれいな画像として貼り付けると描画スピードが落ちるので、妥協も必要であるが、今後他のコマンドにも適用する予定である。

2 次元幾何アニメーションでは、物理教材の作成を第 1 目標にしているため、よく利用される記号を含めておく必要がある。例えば電流計や電圧計、電池、コンデンサー、スイッチなどは、作成予定である。これらは今後の課題である。

このプログラムでは、これまで仕様を主体に考えてきたので、デバッグ機能は殆ど付けていない。このままでは初心者にはマクロ作成が難しいであろう。まず簡単にできるところからデバッグ機能を追加し、実際に使用して利用者の間違えるところを見極め、順次整備して行かなければならない。

さらに、今後も描画機能の拡張を進めて行くであろうが、これからは気軽に変更を加えることが難しくなる。可能な限り上位互換を守るようにしたい。また、これは単なる独立したプログラムではなく、物理シミュレーションの汎用ライブラリにする必要もある。今後プログラムのライブラリ化を進め、どんなプログラムからも呼び出せるようにしておかなければならない。

参考文献

1)　社会システム分析のための統合化プログラム 16 －３Ｄビューアとその応用－, 福井正康, 尾崎誠, 石丸敬二, 福山平成大学経営研究, 7 号, (2011) 111-127.

2)　数理系教育ソフトウェアにおける汎用グラフィックインターフェースの提案, 福井正康, 石丸敬二, 尾崎誠, 日本教育情報学会誌「教育情報研究」, 第 27 巻, 第 4 号, (2012) 55-66.

Multi-purpose Program for Social System Analysis 22

- Geometric Animation -

Masayasu FUKUI

Department of Business Administration, Faculty of Business Administration,
Fukuyama Heisei University

Abstract

We have been constructing a unified program on the social system analysis for the purpose of education. In this article we introduce 2D and 3D geometric animation that students can learn mathematical function and concept of coordinate system by drawing and moving graphics. This program is also applicable to simulations of physics and it is expected to develop to the simulation language.

Keywords

College Analysis, mathematics, graphic, 3D, animation

URL: http://www.heisei-u.ac.jp/ba/fukui/