幾何アニメーション講習会資料

College Analysis の幾何アニメーションシリーズは、2 次元幾何アニメーション、3 次元幾何アニメーション、幾何シミュレーションと 3 種類あります。そのうち幾何シミュレーションは物理学用のプログラムですが、3 次元幾何アニメーションの殆どの機能を含んでいるので、ここでは 2 次元幾何アニメーションと幾何シミュレーションを使って数学の世界を楽しんでみたいと思います。

2 次元のアニメーションは直観的に分かり易く、高校生がいろいろな数学プログラムをつくるには最適です。しかし基本は数学なので、数学的な知識がなければ使えません。逆に言えば数学の知識を再確認するためのツールと思ってもよいでしょう。このプログラムを使うためにどうしても必要なことは、直交座標と極座標の概念とその間の変換です。

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} r &= \sqrt{x^2+y^2} \\ \theta &= \tan^{-1}(y/x) \end{aligned}$$

パソコンではこの関係を以下のように書きます（ほぼ Excel 準拠)。θ は打てないので th としておきましょう。

```
x=r*cos(th)          r=(x^2+y^2)^0.5
y=r*sin(th)          th=atan(y/x)
```

atan はアークタンジェントと言ってタンジェントの逆関数（値から角度を求める）です。ここで注意することは 0 で割れないということです。

演算の基礎が分かれば、プログラムが可能です。いろいろな例を使ってプログラムを作っていきましょう。

例１　ボールの描画

```
ball (3,0), 0.2, &Hff
```

ここで 0.2 はボールの半径です。色については、&Hxxyyzz の形式で表します。xx,yy,zz の部分はそれぞれ、赤、緑、青の部分で 2 桁の 16 進数（00～ff の 256 諧調）で表します。また xx の部分が 0 の場合は、yyzz だけで、xxyy の部分が 0 の場合は、zz だけでも表せます。例えば、&Hff00 とした場合は、xx の部分は 0 とみなされ、&H00ff00 と同じになります。また、&Hff とした場合は、xxyy の部分は 0 とみなされ、&H0000ff と同じになります。色の値はメニューの「色参照」で調べることもできます。画面の広さは、メニューで x,y: -4～4 になっています。この範囲を-10～10 にするには、メニューを変えるか、プログラムで以下のようにします。

例２　描画範囲の変更

```
range -10,10
ball (3,0), 0.2, &Hff
```

これによって、ボールの相対的な大きさも変わっています。次に座標軸を追加してみます。

例３　座標軸

```
axis
ball (3,0), 0.2, &Hff
```

次は、三角関数を使って x 軸から 45 度の位置に描いてみましょう。

例４　三角関数による描画

```
axis
ball (3*cos(pi/4), 3*sin(pi/4)), 0.2, &Hff
```

さらに、time 変数を使ってこのボールを原点の周りに回してみましょう。

例５　三角関数による描画

```
axis
ball (3*cos(time), 3*sin(time)), 0.2, &Hff
```

問題　ボールの回る向きを逆にせよ。

定義を与えて例５と同じものを作ります。定義はよく使う機能です。

例６　定義式

```
axis
define x1=3*cos(time)
define y1=3*sin(time)
ball (x1, y1), 0.2, &Hff
```

定義

```
define 変数名 = 定義式
```

ball と同種類の命令としては以下のものがあります。

球体　tr:不透明度 0-1

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

円　ratio:ry/rx 比 theta:傾き(度) tr:不透明度 0-1

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

三角形

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

四角形

```
squ (x1,y1)-(x2,y2)-(x3,y3)-(x4,y4), color [, tr]
```

正多角形

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

直方体　theta:傾き(度) tr:不透明度 0-1

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

最後に connect 命令でボールと中心をつないでみます。

例７　中心からの糸での連結

```
axis
define x1=3*cos(time)
define y1=3*sin(time)
connect (0,0)-(x1,y1), &Hff0000, 1
ball (x1, y1), 0.2, &Hff
```

問題

以下の連結の定義を使って、糸をバネに変えてみよ。

連結 mode: 1-2:太線, 3-4:矢印, 5-6:バネ, 7-8:抵抗, 9-10:波,
11-12:点線, 13-14:点矢印, 15-16:gluon, 17-18:コイル, 19-20:半コイル
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

```
connect (x1,y1)-(x2,y2), color [, mode, バネ/コイル半径, 巻き数／山谷数]
```

次は動く関数グラフについて見てみましょう。まずは、普通の関数グラフです。

例８　関数グラフの表示

```
axis
func y=2*sin(x), &Hff
```

関数　mode 1:細線 2:太線　func x=f(y), func f(y) も可

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

以下のようにすると、右へ流れて行くグラフが描けます。

例９　動くグラフ

```
axis
func y=2*sin(x-time), &Hff
```

問題１　流れを左にしましょう。

問題２　流れの速さを２倍にしてみましょう。

次はリサージュ図形をパラメータ関数を使って表示してみます。

例 10　リサージュ図形

```
axis
define pu=2*pi*u
define x1=3*cos(3*pu)
```

```
    define y1=3*sin(4*pu)
    param1 (x1, y1), &Hff, 1, 200
```

ここにｕは0～1の間を動きます。200は図形を書くときの分割数です。少なすぎると図形に角ができます。

パラメータ曲線 (0<=u<=1)　mode 1-2:線, 3-4:矢印　r:矢印長さ

```
  param1 (x(u),y(u)), color[, mode, div, r]
  param1r (r(u), θ(u)), color[, mode, div, r]
```

次にこのグラフを次第に伸びて行くように変更してみます。伸びすぎると図に角が出てくるので、4秒経ったら元に戻るようにしました。

例11　次第に伸びるリサージュ図形

```
    axis
    playback 4
    define pu=time*pi/2*u
    define x1=3*cos(3*pu)
    define y1=3*sin(4*pu)
    param1 (x1, y1), &Hff, 1, 200
```

次に陰関数を使った例を紹介します。

例12　陰関数を用いた双曲線

```
axis
impfunc x^2-y^2-1, &Hff
func y=x, &Hff0000
func y=-x, &Hff0000
```

ここで陰関数の命令ではf(x,y)=0のf(x,y)部分だけを記述します。

陰関数 mode 1：細線 2：太線

```
  impfunc f(x,y), color [, mode, div]
```

最後に文字列を円に沿って動かしてみます。

例13　文字列の表示

```
    define x1=2*cos(time)
    define y1=2*sin(time)
    print (x1,y1),"現在は <time> 秒", &Hff0000, 20
```

文字列　文字列の中に<変数名>も可

string (x,y), "文字列" , color [, size, "format1", "format2", …]

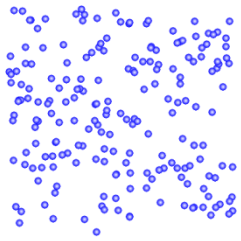
または print (x,y), "文字列" , color [, size, "format1", "format2", …]

次は多くの同じ図形を描く場合についてです。これをいちいち描いていては大変です。ここでは初期位置を乱数で決めて多くのボールを描いてみます。初期位置を乱数で決める命令は、set で、define と同様に使えます。define との違いは、define が実行中に乱数を与えるのに対して、set は最初に与えます。そのため、一度決めた値は実行中変わりません。

多くのボールを表示するには loop 数　プログラム　endloop です。例は 200 個のボールを乱数を使って表示する例です。

例 14　多くの図形をまとめて書く機能

```
loop 200
  set x=6*rnd-3
  set y=6*rnd-3
  ball (x,y), 0.1, &Hff
endloop
```

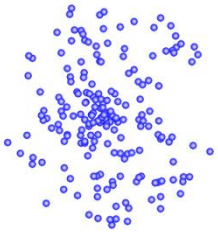


rnd は 0 から 1 の間の乱数を発生させる関数です（参考資料を見て下さい）。そのため x と y には-3 から 3 の乱数が入ります。それを半径 0.1 のボールとして表示しています。実行結果は右の図のようになります。他の図形も同じようにできます。この set 命令は loop の外に出しても同じ結果です。loop の中が分かり易いのでこのような書き方にしました。loop の中の字下げはしなくても構いません。ボールの数は 10,000 個程度まで可能ですが、２次元の場合はびっしりになります。

これを極座標でやってみましょう。

例 15　極座標で設定

```
loop 200
  set r=3*rnd
  set th=2*pi*rnd
  ball (r*cos(th), r*sin(th)), 0.1, &Hff
endloop
```



ball の中は直交座標です。

次にこの図形を回転させてみましょう。ball の式が少し長くなるので、define を使います。

例 16　図形の回転

```
    loop 200
      set r=3*rnd
      set th=2*pi*rnd
      define x=r*cos(th+time)
      define y=r*sin(th+time)
      ball (x, y), 0.1, &Hff
    endloop
```

問題　connect 命令を入れてすべてのボールと原点を緑の糸でつなげ。
極座標は回転に適しています。

最後に複数のバネを loop を使って表示する例を示します。

例 17　いくつかのバネ

```
loop 5
  define x=-3+1.5*lp
  set th=2*pi*rnd
  define y=sin(th+2*time)
  connect (x, 3)-(x, y+0.2), &hff00, 5
  ball (x, y), 0.3, &Hff
endloop
```

この中で２行目に使われている変数 lp は loop の両端の文字を取ったもので、loop の中で、0 から 4 の値を取ります。そのため、x は、-3, -1.5, 0, 1.5, 3 という値になります。後は th で初期乱数角度を与え、ball を動かしています。connect コマンドで、y+0.2 としたのは、バネの端をきれいに見せるためです。

例 18　時間待ち

```
disp 3-time
print (0,0), "描画時間の制限", &Hff, 20
disp time-3.1
ball (cos(time), sin(time)), 0.2, &Hff
```

これはある一定時間対象を表示する例です。disp 引数 の引数が 0 以上のときにそれに続く命令が実行表示されます。

後は、コピー画像やファイル画像を使った命令がありますが、デモの設定のとき操作をしなければいけないので省略しました。幾何アニメーションはプログラマが図形を作り込んでおくとどんなものでも描画に追加できて動かせますので、動く回路図あたり（実際コイルや抵抗は connect 命令に入っている）ばからしくて面白そうかなと思います（後は電圧計や電流計かな)。もし希望があれば追加も可能です。大いに遊んで下さい。

できたデータはグリッドのページを増やしながら、「グリッドへ」ボタンを使ってグリッドへ移し、保存しておいて下さい。

以下のプログラムは授業で用いたサンプルです。活用して下さい。

演習１

```
connect (-4, 0)-(4, 0)
connect (0, -4)-(0, 4)
define a=sin(time)
func y=a*x+b, &Hff, 2
```

演習２

```
connect (-4, 0)-(4, 0)
connect (0, -4)-(0, 4)
define a=sin(time)
func y=a*x^2, &Hff, 2
```

最後の行を以下に変えるとどんな動きをするか。

```
func y=(x-a)^2, &Hff, 2
```

演習３

```
playback 6
connect (-4, 0)-(4, 0)
connect (0, -4)-(0, 4)
define a=time-3
define b=time-3
func y=x^3+a*x^2, &Hff, 2
func y=x^3+b*x, &Hff0000, 2
```

演習４

```
playback 4
rangey 0, 8
func y=0.6^x
define x1=time^2-4
define y1=0.6^x1
ball (x1, y1), 0.2, &H808000
```

4行目を少し変えて、ボールを逆に走らせるにはどうするか。

演習５

```
connect (-4, 0)-(4, 0)
connect (0, -4)-(0, 4)
define a=2.01+sin(time)
func y=a^x, &Hff, 2
func y=log(x)/log(a), &Hff00, 2
func y=x, &Hff0000
```

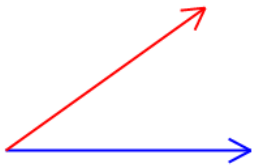
演習６

```
rangex -3*pi, 3*pi
rangey -2, 2
connect (-3*pi, 0)-(3*pi, 0)
connect (0, -2)-(0, 2)
func y=sin(x+time), &Hff,2, , , 100
```

波が右へ動くようにしてみよう。

演習７a　座標の回転と図形の回転

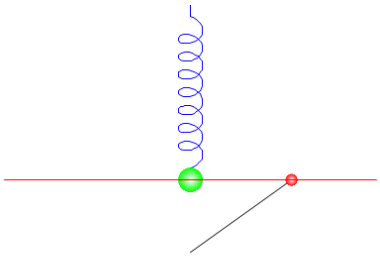
```
define x1=3
define y1=0
connect (0, 0)-(x1, y1), &Hff, 4
define th=time
define x2=cos(th)*x1-sin(th)*y1
define y2=sin(th)*x1+cos(th)*y1
connect (0, 0)-(x2, y2), &Hff0000, 4
```



演習７b　円運動と三角関数

```
define th=time
```

```
define y1=2*sin(th)
connect (0, 4)-(0, y1), &Hff, 5
ball (0, y1), 0.2, &Hff00
define x2=2*cos(th)
define y2=2*sin(th)
connect (0, 0)-(x2, y2)
ball (x2, y2), 0.1, &Hff0000
connect (-3, y2)-(3, y2), &Hff0000
```

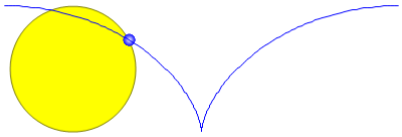


演習７c　位相が違う三角関数を足しても三角関数

```
define a=pi/4
define y1=sin(3*x)
define y2=0.5*sin(3*x+a)
func y=y1, &Hff, 1, , ,100
func y=y2, &Hff00, 1, , ,100
func y=y1+y2, &Hff0000, 2, , , 100
```

演習８　サイクロイド

```
playback 12
define th=time/2
define x0=th-pi
circle (x0, 1), 1, &Hffff00
define x1=th+sin(th)-pi
define y1=1+cos(th)
ball (x1, y1), 0.1, &Hff
define u1=2*pi*u
define x=u1+sin(u1)-pi
define y=1+cos(u1)
param1 (x, y), &Hff, 1, 200
```



演習９

```
playback 30
define n=time
define y=n/(n+1)
print (0, 0), "lim n/(n+1) = <y>", &Hff, 20, "0.0000"
print (-1.9, -0.5), "n=<n>", &Hff, 20, "0.0"
```

```
box (3.5, 2*y-2), 0.4, 4*y, &Hff0000
paint 3
box (3.5,0), 0.4, 4, &H8000
```

演習 10

```
playback 2
range -2, 2
func y=x^2, &Hff, 2
define x1=-1
define y1=x1^2
define a=2*x1
func y=a*(x-x1)+y1, &Hff0000
```

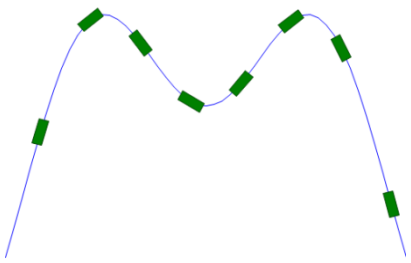
接点 x1 の位置を-1 から時間と共に右に動かして、接線が変化する様子を示せ。

演習 11

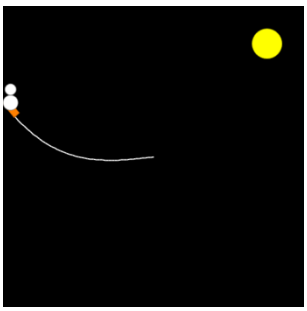
```
playback 10
rangex -6, 6
func y=sin(x), &Hff, 2
define x1=-5+time
define y1=sin(x1)
define a=cos(x1)
func y=a*(x-x1)+y1, &Hff0000
```

演習 12

```
playback 10
func y=x*sin(x), &Hff
loop 20
  define x1=-10+lp+time
  define y1=x1*sin(x1)
  define a=sin(x1)+x1*cos(x1)
  define d=atand(a)
  box (x1, y1), 0.5, 0.2, &H8000, d
endloop
```



演習 13



```
black
playback 4
circle (3, 3), 0.4, &Hffff00
func y=-0.03*(x^3-4*x), &Hffffff, 2, -4, 0
define x1=-3.8
define y1=-0.03*(x1^3-4*x1)
define a=-0.03*(3*x1^2-4)
define d=atand(a)
box (x1, y1+0.1), 0.7, 0.2, &Hff8000, d
circle (x1, y1+0.25), 0.2, &Hffffff
circle (x1, y1+0.6), 0.15, &Hffffff
```

雪だるまを滑らせてジャンプさせよ。

演習 14　3 次元幾何アニメーションの例

```
range -6, 6
define r=(x^2+y^2)^0.5
func z=sin(r), &Hffff, -10, -10, 10, 10
```

これに時間（time）を入れて、波が中心から周りに伝わるようにせよ。

幾何シミュレーション（3 次元幾何アニメーション）のためのコメント

3 次元の図形を作るためには以下の極座標と円柱座標が基本になります。

極座標

$$\begin{aligned} x &= r\sin\theta\cos\phi \\ y &= r\sin\theta\sin\phi \\ z &= r\cos\theta \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} r &= \sqrt{x^2+y^2+z^2} \\ \theta &= \tan^{-1}\left(\sqrt{x^2+y^2}\,/\,z\right) \\ \phi &= \tan^{-1}(x/y) \end{aligned}$$

円柱座標

$$\begin{aligned} x &= r\cos\theta \\ y &= r\sin\theta \\ z &= z \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} r &= \sqrt{x^2+y^2} \\ \theta &= \tan^{-1}(y/x) \\ z &= z \end{aligned}$$

実数数式内で利用可能な演算・関数・定数

^,　*,/,%(余り),　+,-

sin(), cos(), tan(), atan(), exp(), log(), log10()

sinh(), cosh(), tanh(), int(), abs()

sind(度), cosd(度), tand(度), atand()[度]

fact(), gamma(), beta(,)

theta()：x>=0 のとき theta(x)=1, x<0 のとき theta(x)=0

vth()：x>=0 のとき vth(x)=x, x<0 のとき vth(x)=0

def()：x>=0 のとき def(x)=0, x<0 のとき未定義

pulse(): x=0 のとき pulse(x)=1, x<>0 のとき pulse(x)=0

rainbow(x): 0≦x≦1 で虹色のうちの 1 つの色数値を返す

legen(x,l,m): Legendre の陪多項式 l≧|m|

lague(x,n,k): Laguerre の陪多項式 n≧k≧0

pi[=π], ep[=e], rnd[=rnd()], nrnd[標準正規乱数]

2 次元幾何アニメーションコマンド一覧

［分析－数学－幾何アニメーション－2 次元幾何アニメーション］

描画範囲変更
  rangex xmin, xmax （rangey ymin, ymax）
  range min, max ’すべての軸の変更

時間変数（time=0,1,2,…）
  time

定義
  set 変数名 = 定義式　　（rnd を含む式の初期設定）
  define 変数名 = 定義式

ループ（動く変数名は lp=0,1,2,…）
  loop 回数
    図形描画
  endloop
  注）loop の中の$x のような$付き変数は、指定ページのファイルから読み込みます。

時系列データの読み込み
  timedata
  注）&x のような&付き変数を、指定ページから時間とともに 1 行ずつ読み込みます。

図形表示モード
  paint mode　　mode 1:面と縁, 2:面のみ, 3:縁のみ　デフォルトは面と縁

バネなどのゆとり
  grip mode　　mode 1:あり（標準） 2:なし

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

描画条件
  disp 式　　式≧0 のときそれ以降を描画

背景黒色
  black

描画繰り返し
  playback 秒

座標軸 範囲を利用
  axis

グリッド
  grid［細かい幅, 大きな幅］

球体　tr:不透明度 0-1
  ball (x,y), r, color [, tr]

円　ratio:ry/rx 比 theta:傾き(度) tr:不透明度 0-1
  circle (x,y), r, color [, ratio, theta, tr]

三角形
  tri (x1,y1)-(x2,y2)-(x3,y3), color, tr

四角形
  squ (x1,y1)-(x2,y2)-(x3,y3)-(x4,y4), color [, tr]

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

直方体　theta:傾き(度) tr:不透明度 0-1
  box (x,y), wx, wy, color [, theta, tr]

連結 mode: 1-2:太線, 3-4:矢印, 5-6:バネ, 7-8:抵抗, 9-10:波,
      11-12:点線, 13-14:点矢印, 15-16:gluon, 17-18:コイル, 19-20:半コイル
      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
  connect (x1,y1)-(x2,y2), color [, mode, バネ/コイル半径, 巻き数／山谷数]

関数　mode 1:細線 2:太線　func x=f(y), func f(y) も可
  func y=f(x), color [, mode, min, max, div]
  func f(x), color [, mode, min, max, div]

関数値による色付け（虹色）mode 1:高さによる 2:それまでの最大値をキープ
  colorz f(x,y) [, mode, xmin, ymin, xmax, ymax, div, tr]

パラメータ曲線 (0<=u<=1)　mode 1-2:線, 3-4:矢印　r:矢印長さ
  param1 (x(u),y(u)), color[, mode, div, r]
  param1r (r(u), θ(u)), color[, mode, div, r]

パラメータ面 [0<=u,v<=1]
  param2 (x(u,v),y(u,v)), color [, div1, div2, tr]
  param2r (r(u,v), θ(u,v)), color [, div1, div2, tr]

陰関数 mode 1：細線 2：太線
  impfunc f(x,y), color [, mode, div]

クリップボード中の画像貼り付け
  clip (x, y), wx [, theta]

指定された画像を貼り付け
  image (x, y), image 番号, wx, [透明色, theta]

文字列　文字列の中に<変数名>も可
  string (x,y), "文字列" , color [, size, "format1", "format2", …]
  または print (x,y), "文字列" , color [, size, "format1", "format2", …]

幾何シミュレーションコマンド一覧

［分析－教育・科学他－物理シミュレーション－幾何シミュレーション］

描画範囲変更
  rangex xmin, xmax　(rangey ymin, ymax / rangez zmin,zmax)
  range min, max　'すべての軸の変更

時間変数（time=0,1,2,…）
  time

定義
  set 変数名 = 定義式　　（rnd を含む式の初期設定）
  define 変数名 = 定義式

ループ（動く変数名は lp=0,1,2,…）
  loop 回数
    図形描画
  endloop
  注）loop の中の$x のような$付き変数は、指定ページのファイルから読み込みます。

面図形表示モード
  paint mode　　mode 1:面と縁, 2:面のみ, 3:縁のみ　デフォルトは面と縁

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

描画条件
  disp 式　　式≧0 のときそれ以降を描画

背景黒色
  black

描画繰り返し
  playback 時間(秒)

座標軸描画
  axis

図形カットモード
  cutmode

上下角度指定 角度に式も可能
  angle 角度(度)

左右角度指定 角度に式も可能
  rotation 角度(度)

小さな球体軌跡表示 表示される球体の数は指定された数÷球体数
  plot 数

小さな球体　tr:不透明度　balln で軌道非描画
  ball (x, y, z), r, color [, tr]

大きな球体　theta:傾き(度) phi:向き(度) rot:回転(度) tdiv:theta 方向分割数 pdiv:phi 方向分割数
  spher (x, y, z), r, color [, theta, phi, rot, tdiv, pdiv, tr]

三角形
  tri (x1, y1, z1)-(x2, y2, z2,)-(x3,y3,z3), color, tr

正 n 角形 n:正 n 角形 rdiv:r 方向分割数
  poly (x, y, z), r, n, color [, theta, phi, rot, rdiv, tr]

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]

連結 mode: 1:細線 2:太線 3:矢印 4:太矢印 5:バネ
  connect (x1, y1, z1)-(x2, y2, z2), color [, mode, バネ半径/矢印長さ]

関数　func x=f(y, z), func y=f(z, x) も可
  func z=f(x, y), color [, xmin, ymin, xmax, ymax, div, tr]
  func f(x, y), color [, xmin, ymin, xmax, ymax, div, tr]

パラメータ曲線 [0<=u<=1]　mode 1:細線 2:太線 3:矢印 4:太矢印, r:矢印長さ
  param1 (x(u), y(u), z(u)), color [, mode, div, r]

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]

文字列　文字列の中に<変数名>も可　color=image1-10
  string (x,y,z), "文字列", color [, size, "format1", "format2", …]
  または print (x,y,z), "文字列", color [, size, "format1", "format2", …]

mcmc 乱数発生 a:名前 f(x,y,z):密度関数 (極座標 r,th,ph も可)　n:発生個数
              x0,y0,z0:乱数発生初期値
mcmc a, f(x,y,z), n [,x0,y0,z0]
  mcmc a, f(x,y,z), 10000
  loop 10000
    ball(ax,ay,az), 0.1, &H0
  endloop
    上は利用法です。実行の最初にデータが設定されます。

シミュレーション用コマンド
  simdiv 整数
    1 回の描画の間に何回の計算を行うかをプログラムから指定します。

描画の中央指定
  center 名前
    重力シミュレーションで中心に描く星を名前で指定します。

質点
  particle name, mass, charge, x,y,z, vx,vy,vz
    x,y,z は初期位置, vx,vy,vz は初速度
    xname, vxname 等で質点の x 座標, x 方向速度等が参照可能
    name$, xname$, vxname$ 等で loop の中での名前を指定, $=0,1,2,…
    massname で質量を参照可能(現在は ball や spher の半径で利用可能)
  planet も同じ

運動方程式
  axname=式 等　質量部分は入れないこと(入れても particle での指定が有効)
    axname,ayname,azname, vxname,vyname,vzname で指定

ポテンシャル(１つだけ)
  potential=式

束縛条件(複数可能)
  bind 条件式=0

重力計算を実行(運動方程式は不要)
  gravity
    太陽と地球の距離=1, 地球の速さ=1, 地球の質量=1 の単位系です。
    この単位系だと、 v^2/r = GM/r^2 のため
    GM（重力定数×太陽質量）=1 となります。
    また、地球の公転周期は 2π となります。

電磁力学計算を実行（運動方程式は不要)
  emmode

帯電導体 [0<=u,v<=1]
  conductor (x(u,v), y(u,v), z(u,v)), name, charge, [colormode,
            div1, div2, comb1,comb2,…]
    colormode 0:導体ごとの標準化電荷密度, 1:全導体の標準化電荷密度,
              2:0 を黒とした２色濃淡分解, 3:正と負の２色分解
  comb1, comb2 は導線接続を行う導体の名前です。
  静電誘導の問題は最初の計算に時間がかかります。

電気力線　(x,y,z):起点, mode 0:細線 1:太線
  eline (x,y,z), color, [mode, div]

電流 [0<=u<=1]
  ecurrent (x(u), y(u), z(u)), current, color [mode, div]
  電流の方向は u=0->u-1, curret(A), mode 0:細線 1:太線

磁力線　(x,y,z):起点, mode 0:細線 1:太線
```
mline (x,y,z), color, [mode, div]
```

相対論的視覚効果 -0.99<=speed<=0.99
```
sprel speed
```