Spacekc リファレンスマニュアル

CinderellaJapan

2019 年 1 月 2 日

目次

1　はじめに

Spacekc は，KeTCindy のための SpaceCindy である。KeTCindy の 3 D グラフィクスに，簡易レイトレーシングで色付けをすることができる。また，KeTCindy で提供していない，空間に関する計算のためのいくつかの関数を用意している。配布はライブラリではなく，スクリプトを記述したテキストファイルあるいは Cinderella のファイルで行っている。

なお，SpaceCindykc を用いた作図【例】と配布は，KeTCindy のページに掲載している。アドレスは以下の通り。

https://sites.google.com/site/KETCindy/home/cindy3d2

2　定数

正多面体の頂点座標　Rpolydata

説明　正多面体の頂点のリスト。内容は，$\phi=\dfrac{1+\sqrt{5}}{2}$ を黄金比として次のようになっている。

正四面体：$(0,0,1),\left(\dfrac{2\sqrt{2}}{3},0,-\dfrac{1}{3}\right),\left(-\dfrac{\sqrt{2}}{3},\dfrac{\sqrt{6}}{3},-\dfrac{1}{3}\right),\left(-\dfrac{\sqrt{2}}{3},\dfrac{\sqrt{6}}{3},-\dfrac{1}{3}\right)$

正六面体：$(-1,-1,1),(1,-1,1),(1,1,1),(-1,1,1),(-1,-1,-1),(1,-1,-1),(1,1,-1),(-1,1,-1)$

正八面体：$(0,0,1),(0,-1,0),(1,0,0),(0,1,0),(-1,0,0),(0,0,-1)$

正十二面体：$(0,-1,-\phi^2),(0,1,-\phi^2),(0,-1,\phi^2),(0,1,\phi^2),(-1,-\phi^2,0),(1,-\phi^2,0)$
$(-1,\phi^2,0),(1,\phi^2,0),(-\phi^2,0,-1),(-\phi^2,0,1),(\phi^2,0,-1),(\phi^2,0,1),$
$(-\phi,-\phi,-\phi),(-\phi,-\phi,\phi),(-\phi,\phi,-\phi),(-\phi,\phi,\phi),$
$(\phi,-\phi,-\phi),(\phi,-\phi,\phi),(\phi,\phi,-\phi),(\phi,\phi,\phi)$

正二十面体：$(0,-1,-\phi),(0,1,-\phi),(0,-1,\phi),(0,1,\phi),(-\phi,0,-1),(-\phi,0,1)$
$(\phi,0,-1),(\phi,0,1),(-1,-\phi,0),(1,-\phi,0),(-1,\phi,0),(1,\phi,0)$

光源の方向ベクトル　Lightpoint

説明　簡易レイトレーシングを行う場合の光源の方向ベクトル。初期設定は [-1,1,1]

コントラスト　Contrast

説明　簡易レイトレーシングを行う場合の，光のあたり方のコントラスト。標準は 0 以上 1 以下の実数。1 を超えてもよいが，黒い部分が多くなる。描画関数の option で設定することもできる。

3　値の取得

関数　angle3pt(座標１, 座標２, 座標３)
機能　平面上で角を求める
戻り値　点 p1,p2,p3 に対し，∠ p1p2p3 を返す

関数　pointindomain(座標１, 点リスト)
機能　平面上で点リストの閉曲線内に座標１の点があるかどうかの判定
戻り値　領域内なら１，領域外なら０，境界線上なら２を返す。

関数　crosssd(座標１, 座標２, 座標３, 座標４)
機能　座標平面において２本の線分か交わるかどうかを判断する
説明　座標１, 座標２を結ぶ線分と，座標３，座標４を結ぶ線分が交わるかどうか（共有点を持つかどうか）を判定する。
戻り値　交点がある場合は true ，ない場合は false を返す。

関数　interll(座標１, 座標２, 座標３, 座標４)
機能　座標空間において２本の直線の交点の座標を求める
戻り値　座標１, 座標２を通る直線と，座標３，座標４を通る直線との交点の座標を返す。交点がない場合は [i,i,i] を返す。

関数　interss(座標１, 座標２, 座標３, 座標４)
機能　座標空間において２本の線分の交点の座標を求める
戻り値　座標１, 座標２を結ぶ線分と，座標３，座標４を結ぶ線分との交点の座標を返す。交点がない場合は [i,i,i] を返す。

関数　interpl(座標１, 座標２, 座標３, 座標４, 座標５)
機能　座標空間において平面と直線の交点の座標を求める
戻り値　座標１, 座標２, 座標３で決まる平面と，座標４，座標５を通る直線との交点の座標を返す。交点がない場合は [i,i,i] を返す。

関数　interps(座標１, 座標２, 座標３, 座標４, 座標５)
機能　座標空間において平面と線分の交点の座標を求める
戻り値　座標１, 座標２, 座標３で決まる平面と，座標４，座標５を通る線分との交点の座標を返す。交点がない場合は [i,i,i] を返す。

関数　distlp(座標１, 座標２, 座標３)

機能　直線と点の距離を求める。(distance of line to point)

説明　座標 1，座標 2 を通る直線と，座標 3 の点との距離を求める。

戻り値　距離。

平面上の点でも空間の点でもよい。座標 1 と座標 2 が等しいときは虚数単位 i を返すとともに，コンソールに警告「Warning:p1 is same to p2」を表示する。2 点を結ぶ直線のベクトル方程式 $x = x_1 + t(x_2 - x_1), y = y_1 + t(y_2 - y_1)$ を用い，内積＝0 から t を求め距離を計算している。垂線の足の位置ベクトルを $h(x, y)$ とすると，

$(\vec{h} - \vec{p3}) \cdot (\vec{p2} - \vec{p1}) = 0$

から

$$t = \frac{(\vec{p2} - \vec{p1}) \cdot (\vec{p3} - \vec{p1})}{|\vec{p2} - \vec{p1}|^2}$$

を得る。

【例】 println(distlp([1,0],[0,1],[0,0]));

で $\frac{\sqrt{2}}{2}$ が表示される。なお，plintln() により，コンソールには 0.71 が表示されるが，内部的には四捨五入した値ではないので

println(guess(distlp([1,0],[0,1],[0,0])));

とすると，1/2*sqrt(2) と表示される。

【例】 println(distlp([1,1,0],[0,0,1],[0,0,0])); で $\frac{\sqrt{6}}{3}$ が表示される。なお，plintln() により，コンソールには 0.82 が表示されるが，内部的には四捨五入した値ではないので println(guess(distlp([1,1,0],[0,0,1],[0,0,0]))); とすると，1/3*sqrt(6) と表示される。

関数　distpp(座標 1 , 座標 2 , 座標 3 , 座標 4)

機能　座標空間で 3 点を通る平面と点の距離を求める。

説明　座標 1，座標 2，座標 3 を通る平面と，座標 4 の点との距離を求める。

戻り値　距離

【例】 3 点 (2,0,0),(0,2,0),(0,0,2) を通る平面と，原点との距離

distpp([2,0,0],[0,2,0],[0,0,2],[0,0,0]); 戻り値は $\frac{2\sqrt{3}}{3}$

関数　map2d(座標)

機能　座標空間の点を現在の座標平面に射影した点の座標を求める。

説明　「現在の」とは，スライダで設定された軸設定（視点）による画面のことを指す。

【例】 pt=map2d([1,2,3]);

なお，KeTCindy の Parapt() と全く同じ。

戻り値　平面座標

⇒ 関数一覧

関数　normalvec(座標１, 座標２, 座標３)

機能　３点を通る平面の単位法線ベクトルを求める。

説明　座標１，座標２，座標３を通る平面の単位法線ベクトルを求める。

戻り値　法線ベクトルを表すリスト。

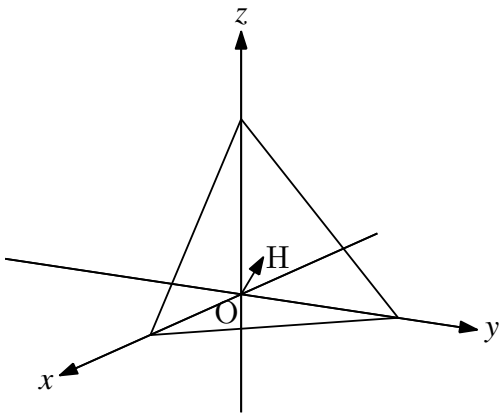
ベクトルの向きは，点の順序による。

【例】 normalvec([2,0,0],[0,2,0],[0,0,2]); の結果は $\left(\frac{\sqrt{3}}{3},\frac{\sqrt{3}}{3},\frac{\sqrt{3}}{3}\right)$

normalvec([2,0,0],[0,0,2],[0,2,0]); の結果は $\left(-\frac{\sqrt{3}}{3},-\frac{\sqrt{3}}{3},-\frac{\sqrt{3}}{3}\right)$

平面と点の距離を求める distpp() を組み合わせると，平面に下した垂線のベクトルを表示できる。

```
nv=normalvec([2,0,0],[0,2,0],[0,0,2]);
dd=distpp([2,0,0],[0,2,0],[0,0,2],[0,0,0]);
poly3d("1",[[2,0,0],[0,2,0],[0,0,2]]);
arrow3d("1",[[0,0,0],dd*nv]);
Letter3d([dd*nv,"e","H"]);
```



⇒ 関数一覧

関数　onsameplane(pa,pb,pc,pd)

機能　４点が同一平面上にあるかどうかの判定

説明　引数は判定する４点の座標。外積と内積を用いた判定法による 。

戻り値　true / false

関数　perp(座標１, 座標２, 座標３)

機能　２点を通る直線への垂線の足の座標を求める。

戻り値　座標１，座標２の２点を通る直線に，座標３の点から下した垂線の足の座標を返す。

関数　perpvec(ベクトル,flag)

機能　ベクトルに垂直な平面の基本ベクトルを作る。

説明　第 1 引数のベクトルに垂直な平面上に 2 つの直交するベクトルを作り，リストにして返す。ひとつは，z 成分を 1 として計算する。flag はオプションで，0 にすると単位ベクトルにしない。デフォルトは単位ベクトル。

戻り値　ベクトル。

【例】perpvec([1,1,1]) の結果は [[-0.8165,0.4082,0.4082],[0,-0.7071,0.7071]]
perpvec([1,1,1],0) の結果は [[-2,1,1],[0,-1,1]]

関数　planecoeff(座標 1 , 座標 2 , 座標 3)

機能　3 点を通る平面の方程式の係数を求める。

説明　座標 1 ，座標 2 ，座標 3 の 3 点を通る平面の方程式 $ax + by + cz = 1$ の係数 a, b, c を求める。

戻り値　リスト [a,b,c]。係数 a, b, c が存在しない場合はコンソールに「Warning! Cannot decide a coefficient.」を表示し，[i,i,i] を返す。

関数　reflect3d(dlist,mirror)

機能　dlist の鏡像を得る

説明　第 1 引数は，点，プロットデータ，面データのいずれか。第 2 引数の座標リストが，1 つの点だけなら，点対称，2 つの点なら，その 2 点を結ぶ直線に関する対称点，3 つの点なら，その 3 点で決まる平面に関する対称点となる。

戻り値　第 1 引数と同じタイプのデータ

KeTCindy の Refrectdata3d() との違いは，描画をしないこと，引数が，点，プロットデータ，面データのいずれでもよいことである。第 2 引数の形は同じ。

関数　rotate3d(dlist,vec,angle,center)

機能　dlist を指定した角だけ center を始点とするベクトル周りに回転する。

説明　dlist は点，プロットデータ，面データのいずれか。

戻り値　dlist と同じ形式のデータ。

KeTCindy の Rotatedata3d() との違いは，描画をしないこと，引数が，点，プロットデータ，面データのいずれでもよいことである。その他の引数の形式は同じ。

【例】 angle=[0,0,pi/4]; p=[1,0,0]; drawpt3d(rotate3d(p,angle));
により，点 (1,0,0) を z 軸回りに $\frac{\pi}{4}$ だけ回転した位置に点を打つ。

⇒ 関数一覧

関数　translate3d(dlist, ベクトル)

機能　dlist を指定したベクトルだけ平行移動。

説明　dlist は点，プロットデータ，面データのいずれか。

戻り値　dlist と同じ形式のデータ

KeTCindy の Translatedata3d() との違いは，描画をしないこと，引数が，点，プロットデータ，面データのいずれでもよいことである。その他の引数の形式は同じ。

関数　rotmatrix(vec)

機能　法線ベクトルから回転行列を作る

説明　引数は法線ベクトル。 初め，法線ベクトルは [0,0,1] であるとする。これを回転して，引数の vec になるような回転行列を求める。y 軸周り，z 軸周りの順に回転する行列。

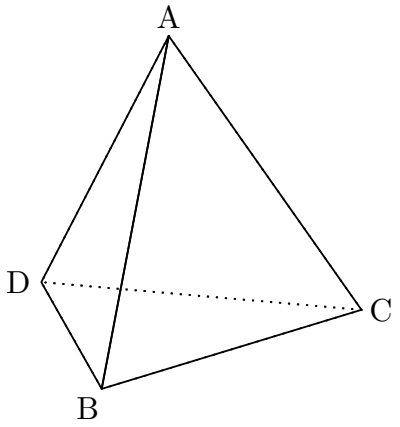
戻り値　行列

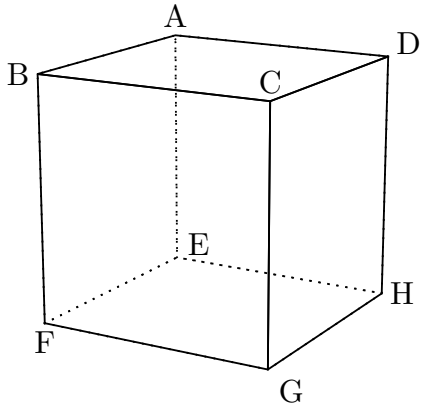
関数　vertexrpolyhedron(n)

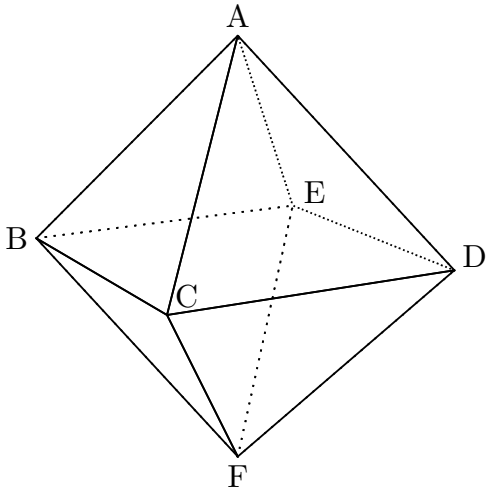
機能　正 n 面体の頂点リストを取得する

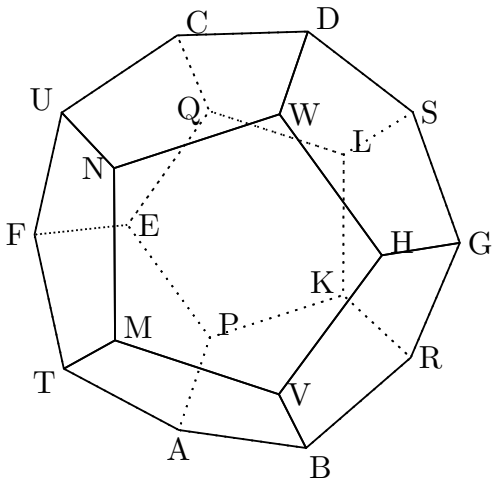
説明　n の値は 4,6,8,12,20 のいずれか。

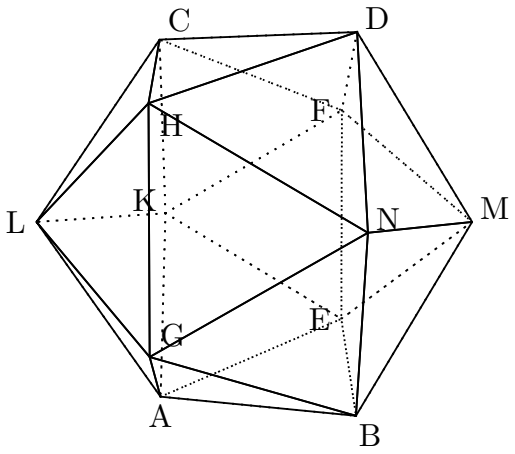
戻り値　半径 1 の球面に内接する正多面体の頂点リスト。頂点の順番は次のようになっている。それぞれアルファベット順。











なお，スクリプト内では，次のように定義されている。これを，半径 1 の球面に内接するように変換して戻り値としている。

$\phi = \dfrac{1+\sqrt{5}}{2}$ は黄金比。

正四面体：$(0,0,1),\left(\dfrac{2\sqrt{2}}{3},0,-\dfrac{1}{3}\right),\left(-\dfrac{\sqrt{2}}{3},\dfrac{\sqrt{6}}{3},-\dfrac{1}{3}\right),\left(-\dfrac{\sqrt{2}}{3},\dfrac{\sqrt{6}}{3},-\dfrac{1}{3}\right)$

正六面体:$(-1,-1,1),(1,-1,1),(1,1,1),(-1,1,1),(-1,-1,-1),(1,-1,-1),(1,1,-1),(-1,1,-1)$

正八面体：$(0,0,1),(0,-1,0),(1,0,0),(0,1,0),(-1,0,0),(0,0,-1)$

正十二面体：$(0,-1,-\phi^2),(0,1,-\phi^2),(0,-1,\phi^2),(0,1,\phi^2),(-1,-\phi^2,0),(1,-\phi^2,0)$

$(-1,\phi^2,0),(1,\phi^2,0),(-\phi^2,0,-1),(-\phi^2,0,1),(\phi^2,0,-1),(\phi^2,0,1),$

$(-\phi,-\phi,-\phi),(-\phi,-\phi,\phi),(-\phi,\phi,-\phi),(-\phi,\phi,\phi),$

$((\phi,-\phi,-\phi),(\phi,-\phi,\phi),\phi,\phi,-\phi),(\phi,\phi,\phi)$

正二十面体：$(0,-1,-\phi),(0,1,-\phi),(0,-1,\phi),(0,1,\phi),(-\phi,0,-1),(-\phi,0,1)$

$(\phi,0,-1),(\phi,0,1),(-1,-\phi,0),(1,-\phi,0),(-1,\phi,0),(1,\phi,0)$

⇒ 関数一覧

関数　val2tex(x)

機能　数値を guess() で解析して，分数，平方根の TeX の文字列にして返す

説明　分数，平方根でない場合は guess() の結果をそのまま返す。計算結果が分数や平方根になる場合に，この関数を適用し，戻り値の前後にドルマークをつけて drawtext() で表示すれば分数や平方根の形で表示される。letter3d() でも利用できる。ただし，精度によっては変換できない場合もある。

戻り値　TeX の文字列

【例】：原点から平面に下した垂線の足の座標を計算して表示する。

```
pa=[2,0,0];
pb=[0,2,0];
```

```
pc=[0,0,2];
nv=normalvec(pa,pb,pc);
hv=distpp(pa,pb,pc,[0,0,0])*nv;
hvstr=apply(hv,val2tex(#));
plate3d("1",[pa,pb,pc],["Color=skyblue","Rayoff"]);
poly3d("1",[pa,pb,pc]);
arrow3d("1",[[0,0,0],hv],["size=2"]);
letter3d([pa,"s2",text(pa_1),pb,"s2",text(pb_2),pc,"w2",text(pc_3),
   hv,"ne2","H$\left( "+hvstr_1+","+hvstr_2+","+hvstr_3+" \right)$"]);
```

⇒ 関数一覧

4　描画関数

options

直線・曲線を描く関数の引数には座標の他，オプションがある。KeTCindy のオプションの他，メッシュ密度，濃度，コントラスト，レイトレーシングの有無が指定できる。

＜メッシュ密度＞

曲面を描くとき，網目状にして描く。このときの経緯線の数。

メッシュ密度指定は，”Mesh=[10,15]” のように与える。

＜濃度＞

濃度は 0 から 1 までの実数。”Alpha=n” で，n は 0 以上 1 以下。たとえば，”Alpha=0.4”。初期設定は 0.3。

なお，「Alpha」は，CindyScript の「透明度」と同じ語であるが，ここでは，濃淡を表し，1 のときが指定した色で，0 に近いと白に近くなる。向こう側が透けて見えるようになるわけではない。

＜レイトレーシング＞

レイトレーシングで陰影をつけるかどうかの指定。”Rayon” のとき陰をつけ，”Rayoff” のときは陰影をつけない。初期設定は ”Rayon”

＜非表示＞

戻り値がプロットデータの描画関数で，プロットデータだけを取得したい場合に，KeTCindy と同様，”nodisp” オプションをつける。

描画関数の戻り値

描画関数の幾つかには戻り値がある。それぞれの関数の説明を参照のこと。

関数　grid(範囲 1，範囲 2，本数，option)

機能　xy 平面に方眼を表示する。

説明　範囲 1 は x 軸の範囲，範囲 2 は y 軸の範囲。

戻り値　なし

本数は，単位 (1 目盛) あたりの本数。

色，太さなど，通常の option が有効。

引数，option ともなしで，grid() とすることも可能で，その場合は，範囲を setaxis() で指定した範囲とし，単位あたり 1 本と 10 本の方眼を描く。

方眼は描画面だけで，TeX には出力されない。

⇒ 関数一覧

関数　line3d(name, 座標リスト ,option）

機能　２点を結ぶ直線を描く

説明　２点の座標をリストで与えて，３次元空間に直線を表示する。

戻り値　２点の座標リストをそのまま返す。

⇒ 関数一覧

関数　arrow3d(name, 座標リスト ,option)

機能　２点を結ぶ矢線を描く

説明　２点の座標のリストを与えて，３次元空間に矢線を表示する 。option は，color 指定以外は２次元の KeTCindy の Arrowdata と同様。ただし，矢じりについては，Cinderella の画面上では大きさ以外のオプションは反映されない。また，矢じりについてのオプションは初めに書く必要がある。

戻り値　２点の座標リストをそのまま返す。

【例】　２点 $(1,3,2),(-2,-1,-2)$ を結ぶ矢線を表示する。

```
arrow3d("1",[[1,3,2],[-2,-1,-2]])
```

【例】　矢じりの大きさを２にして，全体を赤で描く。

```
arrow3d("1",[[2,0,0],[-1,4,1]],[2,1,1,"dr,2","Color=red"])
```

⇒ 関数一覧

関数　poly3d(name, 座標リスト,option)

機能　多角形を描く

説明　リストで与えた点を結ぶ多角形を描く。リストで与えた点が同一平面上になくてもそれらの点を結んだ図形を描く。リストで与える点は閉じていなくてもよい。自動的に閉じて描く。

戻り値　描画した多角形のプロットデータ。

【例】　３点 $(1,1,1),(2,2,1),(0,1,-1)$ を頂点とする三角形を描く。

```
pd=[[1,1,1],[2,2,1],[0,1,-1]];
poly3d("1",pd);
```

⇒ 関数一覧

関数　plate3d(name, 座標リスト,option)

機能　多角形を色塗りする

説明　リストで与えた点を結ぶ多角形を色塗りする。縁取りはしない。

戻り値　引数に与えた座標リストをそのまま返す

Rayoff オプションをつけると，レイトレーシングを行わない。

【例】: ３点 $(1,1,1),(2,2,1),(0,1,-1)$ を頂点とする三角形を赤で塗る

```
pd=[[1,1,1],[2,2,1],[0,1,-1]];
plate3d("1",pd,["Color=Red"]);
```

⇒ 関数一覧

関数　circle3d(name, 中心, 法線ベクトル, 半径,option)

機能　円を描く

説明　中心, 半径と法線ベクトルを与えて円を描く。

戻り値　円のプロットデータ

【例】: 中心 (1,1,1), 半径 2，法線ベクトルが (1,1,1) である円を太さ 2，赤で描く。

```
circle3d("1",[1,1,1],[1,1,1],2,["dr,2","Color=Red"]);
```

⇒ 関数一覧

関数　drawarc3d(name, 中心, 法線ベクトル, 半径, 範囲,option)

機能　弧を描く

説明　中心, 半径と法線ベクトルを与えて描く円の一部として弧を描く。

円は，$x = r\cos\theta, y = r\sin\theta$ で描き，法線ベクトルで決まる回転行列で回転している。

戻り値　弧のプロットデータ

【例】: 中心 (1,1,1), 半径 2，法線ベクトルが (-1,1,1) である弧を描く。

```
drawarc3d("1",[1,1,1],[-1,1,1],2,[0,2*pi/3]);
```

⇒ 関数一覧

関数　disc3d(name, 中心, 法線ベクトル, 半径,option)

機能　中を色塗りした円盤を描く

説明　中心, 半径と法線ベクトルを与えて円盤を描く。縁取りはしない。

戻り値　円のプロットデータ

Rayoff オプションをつけると，レイトレーシングを行わない。

【例】: 中心 (1,1,1), 半径 2，法線ベクトルが (1,1,1) である円盤を赤，透明度 0.4 で描く。

```
disc3d("1",[1,1,1],[1,1,1],2,["Color=Red","Alpha=0.4"]);
```

⇒ 関数一覧

関数　drawsphere(name, 中心, 半径,opton)

機能　球面を描く

説明　中心と半径を指定して球面を描く。球面を簡易レイトレーシングを用いてグラデーションをかけて描く。 球面を網目状に分けて色塗りをしている。

引数は，中心の座標，半径，option。

半径は，x 軸，y 軸，z 軸方向についてリストにして指定することもできる。

描画には時間がかかる。あまりにも時間がかかりすぎる場合は，”Mesh=[10,10]”をオプションにつけて描画してみる。これで描ければ編み目を細かくする。初期値は”Mesh=[30,20]”。

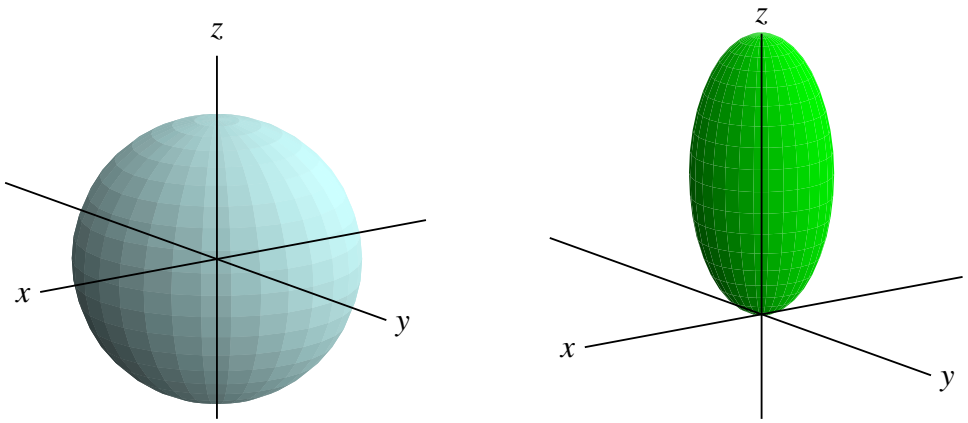
戻り値　なし

【例】原点中心，半径 2 の球面（下図左）

```
drawsphere("1",[0,0,0],2)
```

中心が (0,0,2)，半径が [1,1,2] ：楕円形の球を緑で描く（下図右）

```
drawsphere("1",[0,0,2],[1,1,2],["Color=green","Mesh=[20,20]"])
```



⇒ 関数一覧

関数　quasisphere(name, 中心, 半径,fill,option)

機能　疑似球面を描く

説明　中心と半径を指定して疑似球面を描く。座標軸を動かしても常に法線ベクトルが視点を向いた円を描くので，球面同様に見える。しかし実際には面ではない。 したがって，陰線処理はしない。

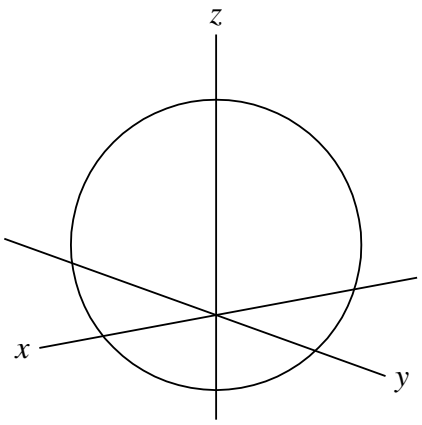
引数は，中心の座標，半径，fill , option。

fill は塗りつぶすかどうかで，1 なら塗りつぶす（初期値），0 なら塗らない。

fill は省略可能。

戻り値　円のプロットデータ

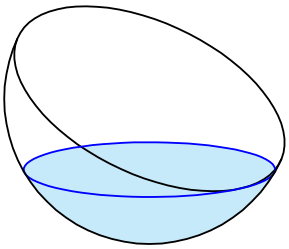
【例】 `quasisphere("1",[0,0,1],1,0]);`

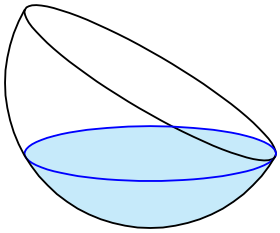


陰線処理をしない分，Sfbdparadata() と ExeccmdC() で描くより速い。これを用いて，次のようなスクリプトを作ると，疑似球面で十分であることがわかる。

```
pd1=quasisphere("1",[0,0,0],2,0,["nodisp"]);
pd2=circle3d("1",[0,0,0],[0,1,sqrt(3)],2,["nodisp"]);
pd3=circle3d("2",[0,0,-1],[0,0,1],sqrt(3),["nodisp"]);
sp=apply(pd1,map2d(#));
su1=apply(pd2,map2d(#));
su2=apply(pd3,map2d(#));
Listplot("1",sp,["nodisp"]);
Listplot("2",su1);
Listplot("3",su2,["Color=blue"]);
int1=Intersectcrvs("sg1","sg2");
int2=Intersectcrvs("sg1","sg3");
println(int1);
println(int2);
p1=int1_1;
p2=int1_2;
p3=int2_1;
p4=int2_2;
Partcrv("1", p2, p1, "sg1");
Partcrv("2", p3, p4, "sg3",["nodisp"]);
Partcrv("3", p4, p3, "sg1",["nodisp"]);
Joincrvs("1",["part2","part3"],["nodisp"]);
Shade(["join1"],["Color=[0.2,0,0,0]"]);
```

スライダを動かして視点を変えたとき，次のようになる。





なお，このスクリプトを実行したとき，視点によって交点 int1 , int2 の数が変わるので，コンソールに表示されたものを見て，`p1,p2,p3,p4` を設定する必要がある。

⇒ 関数一覧

関数　polyhedron(name, 面データ,option)

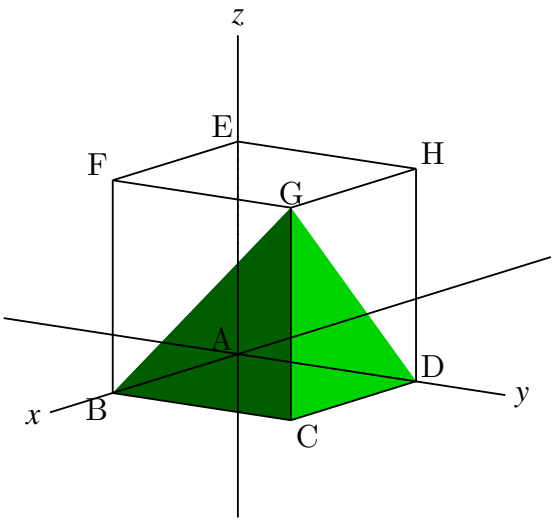
機能　面を色塗りした多面体を描く

説明　縁取りはしない。

戻り値　引数の面データをそのまま返す

Rayoff オプションをつけると，レイトレーシングを行わない。

【例】 立方体 ABCD-EFGH を，B,D,G を通る平面で切ってできる錐体を緑で描く。



頂点 B,C,D,G に対し，面リストを次のように作る。

まず，頂点 B,C,D,G に番号 1,2,3,4 をつける。

面 BCG は外側から見て反時計回りに B,C,G なので [1,2,4]

面 CDG は同様に C,D,G なので [2,3,4]

面 DGB は同様に D,B,G なので [3,1,4]

底面 BCD は外側から見て反時計回りに B,D,C なので [1,3,2]

そこで，B,C,D,G の座標を p1,p2,p3,p4 として，次のようにして面データ `fd` を作る。

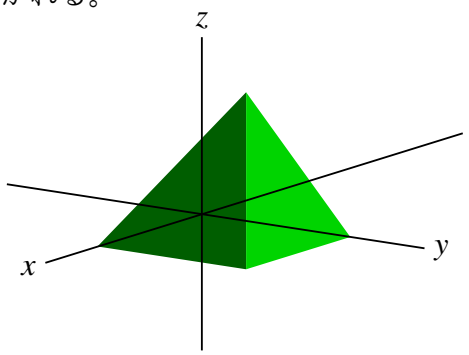
```
p1=[2,0,0];
```

```
p2=[2,2,0];
p3=[0,2,0];
p4=[2,2,2];
fd=[[p1,p2,p3,p4],[[1,2,4],[2,3,4],[3,1,4],[1,3,2]]];
```

この面データを使って

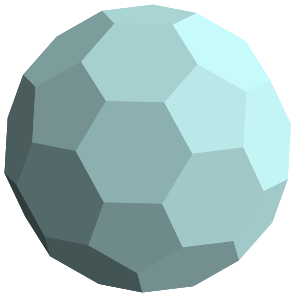
```
polyhedron("1",fd,["Color=Green"])
```

とすれば，錐体が描かれる。



【例】小林・鈴木・三谷による 多面体データ polyhedrons_obj を用いた多面体を描く。
データは，正多面体，半正多面体，ジョンソンの立体で，
http://mitani.cs.tsukuba.ac.jp/polyhedron/index.html
にある。このデータを格納したフォルダ polyhedrons_obj へのパスを `Setdirectory()` で指定し，`Readobj()` で読み込む。たとえば，ワークディレクトリ（fig フォルダ）に置いた場合は，

```
Setdirectory(Dirwork+"/polyhedrons_obj");
polydt=Readobj("s06.obj");
Setdirectory(Dirwork);
fd=[2*polydt_1,polydt_2];
polyhedron("1",fd);
```



で右のような図ができる。
なお，`fd=[2*polydt_1,polydt_2]` で，頂点の座標を 2 倍にしている。
稜線を描くのであれば

```
VertexEdgeFace("1",fd);
Nohiddenbyfaces("1","phe3d1","phf3d1");
```

を追加すればよい。

⇒ 関数一覧

関数　convexhedron(name, 頂点リスト, 倍率,option)

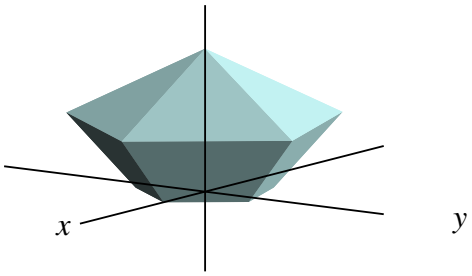
機能　凸型多面体を描く

説明　頂点リストを与えて面を塗った凸型多面体を描く。倍率は，頂点リストに対して実際に描画するサイズへの倍率。1 のときは省略できる。

戻り値　描画した面データ。

【例】底面が五角形の凸型多面体

```
th=2*pi/5;
pd=apply(1..5,[cos(#*th),sin(#*th),0]);
pd=pd++apply(1..5,[2*cos(#*th),2*sin(#*th),1]);
pd=append(pd,[0,0,2]);
println(pd);
fd=convexhedron("1",pd)
```



稜線を描くのであれば，戻り値を利用して

```
fd=convexhedron("1",pd);
VertexEdgeFace("1",fd);
Nohiddenbyfaces("1","phe3d1","phf3d1");
```

とすればよい。

⇒ 関数一覧

関数　rpolyhedron(name，面数, 半径,option)

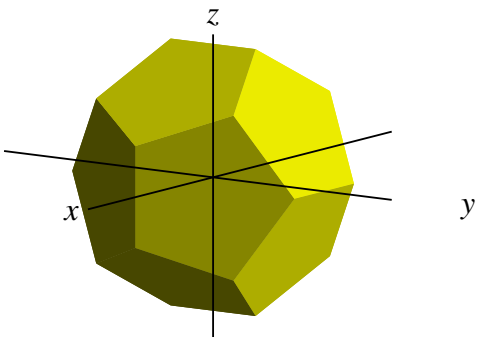
機能　面を色塗りした正多面体を描く

説明　正多面体は 5 種類あり，Spacekc にはデータが組み込まれている。正多面体の頂点座標を参照のこと。これを用いて，面の数と外接球の半径を指定すれば正多面体を描くことができる。

戻り値　描画した面データ

【例】 面を黄色で塗った正十二面体を描く

```
rpolyhedron("1",6,2,["Color=yellow"]);
```



稜線を描くのであれば，戻り値を利用して

```
fd=rpolyhedron("1",12,2,["Color=yellow"]);
VertexEdgeFace("1",fd);
Nohiddenbyfaces("1","phe3d1","phf3d1");
```

とすればよい。

＜参考＞正多面体のサイズ

rpolyhedron() では，指定した半径の球面に内接するサイズで描画する。ここで，その半径と辺の長さの関係をあげておく。ここで，$\phi = \dfrac{1+\sqrt{5}}{2}$

面	辺の長さ	内接球半径
4	$\dfrac{2\sqrt{6}}{3}$	$\dfrac{1}{3}$
6	$\dfrac{2}{\sqrt{3}}$	$\dfrac{1}{\sqrt{3}}$
8	$\sqrt{2}$	$\dfrac{1}{\sqrt{3}}$
12	$\dfrac{2}{\sqrt{3}\phi}$	$\sqrt{\dfrac{\phi^3}{3\sqrt{5}}}$
20	$\dfrac{2}{\sqrt{\sqrt{5}\phi}}$	$\dfrac{\phi^2}{\sqrt{3\sqrt{5}\phi}}$

【例】 一辺の長さが 2 の正六面体を描く

```
rpolyhedron(6,sqrt(3),["dr,2"]);
```

⇒ 関数一覧

関数　frustum(name,n,r1,r2,h,option)

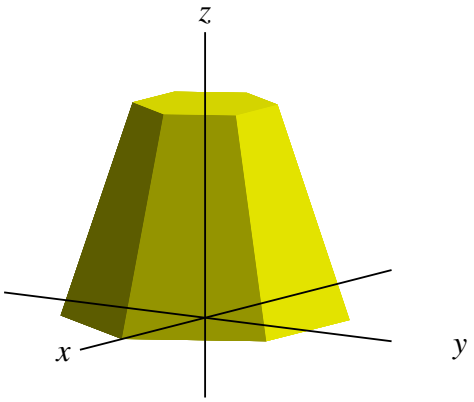
機能　正角錐（台）の面を塗る n：角の数

r1,r2：上底，下底の外接円の半径

h：高さ

【例】 正六角錐台を描く

```
frustum("1",6,1,2,3,["Color=yellow"]);
```



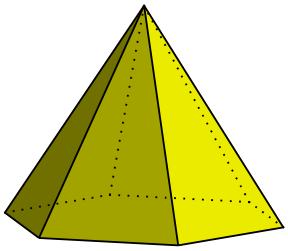
稜線を描くのであれば，戻り値を利用して

```
fd=frustum("1",6,1,2,3,["Color=yellow"]);
VertexEdgeFace("1",fd);
Nohiddenbyfaces("1","phe3d1","phf3d1");
```

とすればよい。

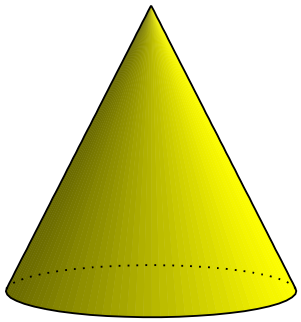
【例】 上底の外接円の半径を 0 にすれば正六角錐になる。（座標軸非表示）

```
fd=frustum("1",6,0,2,3,["Color=yellow"]);
VertexEdgeFace("1",fd);
Nohiddenbyfaces("1","phe3d1","phf3d1");
```



【例】 角数を多くするとほとんど円錐になる。輪郭線は Sfbdparadata で曲面として描く。

```
frustum("1",108,0,2,4,["Color=yellow"]);
fd=[
 "p",
 "x=r*cos(t)","y=r*sin(t)","z=2*(2-r)",
 "r=[0,2]","t=[0,2*pi]","e"
];
Startsurf();
Sfbdparadata("1",fd);
ExeccmdC("1");
```



上底と下底を同じサイズにすれば円柱になる。

⇒ 関数一覧

関数　hatch3d(name, 方向,PD,option)

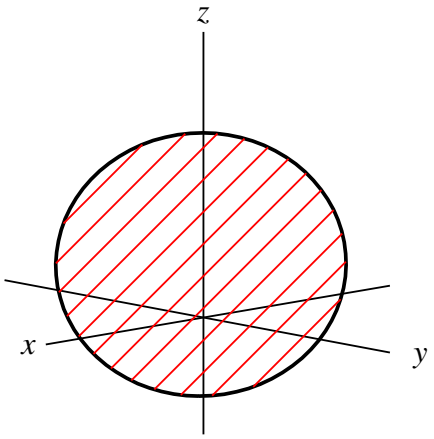
機能　閉曲線にハッチをかける。

説明　閉曲線は，poly3d() , circle3d(); で描いたものなど。KeTCindy の Hatchdata() と異なり，閉曲線だけが対象なので，方向は引数にない。option には色指定を入れることができ，色指定があればその色でハッチをかける。 複数の領域にハッチをかけることはできない。

戻り値　なし

【例】 円にハッチをかける。

```
pd=circle3d("1",[1,1,1],[1,1,1],2,["dr,2"]);
hatch3d("1",pd,["Color=red"]);
```


z
x
y

5　関数一覧

KeTCindy の関数名が大文字で始まるのに対し，Spacekc の関数名は小文字で始まる。ただし，若干例外がある。

【定数】

Lightpoint	光源の方向ベクトル
Contrast	コントラスト
Rpolydata	正多面体の頂点リスト

【値の取得】

angle3pt(座標, 座標, 座標)	平面上の 3 点 p1,p2,p3 に対し角 p1p2p3 を返す
pointindomain(座標, 点リスト)	平面上で点が閉曲線内にあるかどうかの判定
distlp(座標, 座標, 座標)	２点を通る直線と点の距離を取得する
distpp3d(座標, 座標, 座標, 座標)	座標空間で 3 点を通る平面と点の距離を取得する
map2d(座標)	空間座標を座標平面に射影した点の座標を取得する
nomalvec(座標, 座標, 座標)	３点を通る平面の法線単位ベクトルを取得する
perp(座標, 座標, 座標)	座標空間で 2 点を通る直線への垂線の足の座標を取得する
perpvec(ベクトル,flag)	ベクトルに垂直な平面上に直交する基本ベクトルをとる
planecoeff(座標, 座標, 座標)	座標空間で 3 点を通る平面の方程式の係数を取得する
crosssd(座標, 座標, 座標, 座標)	座標平面で 2 つの線分が交わるかどうかの判断
interpl(座標, 座標, 座標, 座標, 座標)	平面と直線の交点の座標を求める
interps(座標, 座標, 座標, 座標, 座標)	平面と線分の交点の座標を求める
reflect3d(リスト, リスト)	データの鏡像を得る
rotate3d(座標, 角リスト)	データを回転する
translate3d(リスト, ベクトル)	データを平行移動する
vertexrpolyhedron(数)	正多面体の頂点リストを取得する

【描画】

grid()	xy 平面の方眼の描画
line3d(name,list,option)	２点を結ぶ直線を描く
arrow3d(name,list,option)	２点を結ぶ矢線を描く
poly3d(name,list,option)	多角形を描く
circle3d(name,center, r,vec,option)	円を描く
circle3d(name,center, r,vec,range,option)	弧を描く
plate3d(name,list,option)	多角形を色塗りする
disc3d(name,center, r, vec,option)	中を塗った円盤を描く

drawsphere(name,center,r,option)	球面を描く
quasisphere(name,center,r,option)	球面のようなものを描く
frustum(name,fn,r1,r2,h, option)	角錐台を描く
polyhedron(name,fd,option)	面を色塗りした多面体を描く
rpolyhedron(name,fn,r,option)	面を色塗りした正多面体を描く
convexhedron(name,pd,size,option)	面を色ぬりした凸型多面体を描く
hatch3d(name,direc,PD,option)	閉曲線にハッチをかける