Spacekc Reference

CinderellaJapan

January 2, 2019

Contents

1 Introduction 2

2 Constant 2

3 Value 2

4 Drawing 7

1 Introduction

Spacekc is a function library on Ketcindy. To KeTCindy's 3D graphics, color with simple ray-tracing be able to. In addition, several functions for calculating relating to space are prepared.

2 Constant

Direction vector of light source Lightpoint

Description Direction vector of light source for simplified ray tracing. Default is [-1,1,1].

Contrast Contrast

Description Contrast in the direction of light when performing simple ray tracing. Standard is a real number between 0 and 1.

3 Value

Function angle3pt(coordinate1,coordinate2,coordinate3)

Description Find an angle on a 2D plane.

Return value This function is return p1p2p3 for point p1,p2,p3.

Function pointindomain(coordinate1,list of point)

Description Judgment whether or not there is a point in the closed curve.

Return value The judgment that this function has a point of coordinate1 in the closed curve of the list of points on a plane. The case in the domain return 1 , out of domain return 0, on a boundary line return 2.

Function crosssd(coordinate1,coordinate2,coordinate3,coordinate4)

Description determine whether two segments cross in 2D plane

Description judge it whether a segment of links coordinate 3, coordinate 4 to the segment of linking coordinate 1, coordinate 2 has a common point.

Return value When there is a common point, true is returned, and false is returned when there is not it.

Function interll(coordinate1,coordinate2,coordinate3,coordinate4)

Description Demand the point of intersection of the two straight lines

Return value Demand the coordinate of the point of intersection with the straight line via coordinate 1, coordinate 2 and coordinate 3, coordinate 4. When there is not a point of intersection, It return [i,i,i].

Function interss(coordinate1,coordinate2,coordinate3,coordinate4)

Description Demand the point of intersection of the two segments.

Return value Demand the coordinate of the point of intersection with the segment via coordinate 1, coordinate 2 and coordinate 3, coordinate 4.

When there is not a point of intersection, It return [i,i,i].

Function interpl(list of coefficients ,coordinate1,coordinate2)

Description Demand a plane and the point of intersection of the straight line.

Return value Demand the coordinate of the point of intersection with the plane via coordinate 1, coordinate 2, coordinate 3 and line via coordinate 4 , coordinate 5.

When there is not a point of intersection, It return [i,i,i].

Function interps(list of coefficients ,coordinate1,coordinate2)

Description Demand a plane and the point of intersection of the segment

Return value Demand the coordinate of the point of intersection with the plane via coordinate 1, coordinate 2, coordinate 3 and segment to link coordinate 4 to coordinate 5.

When there is not a point of intersection, It return [i,i,i].

Function distlp(coordinate1,coordinate2,coordinate3)

Description Distance of a line to a point .

Return value It find a straight line via coordinate1 and coordinate2 and the distance with the point of coordinate3. The return value is distance. When coordinate 1 and coordinate 2 is equal, return imaginary unit i and display warning "Warning:p1 is same to p2" to a console.

Example println(distlp([1,0],[0,1],[0,0]));
In addition, 0.71 is displayed to a console by plintln(), but is displayed with 1/2*sqrt(2) by

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

Example println(distlp([1,1,0],[0,0,1],[0,0,0])); In addition, 0.82 is displayed to a console by plintln(), but is displayed with 1/3*sqrt(6) by

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

Function distpp(coordinate1,coordinate2,coordinate3,coordinate4)

Description Distance of a plene and a point in 3D space

Return value It find a plane via coordinate1 ,coordinate2 and coordinate2 and the distance with the point of coordinate4. The return value is distance.

Example distpp([2,0,0],[0,2,0],[0,0,2],[0,0,0]);
return value is $\frac{2\sqrt{3}}{3}$

Function Coordinate of the point that projection of space

Description In a current screen set with a circular slider, It find the coordinate which performed a projection of a point on the space on a plane.

Return value coordinate

Example : pt=map2d([1,2,3]);

It is exactly the same as KeTindy's Parapt ().

Function normalvec(coordinate1,coordinate2,coordinate3)

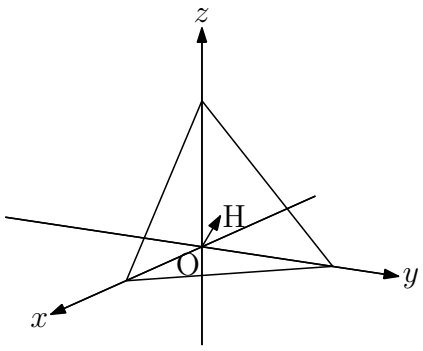
Description It demand a plane unit normal vector via coordinate 1, coordinate 2, coordinate 3.

Return value normal vector.

The direction of the vector is decided in order of a point.
Exsample normalvec([2,0,0],[0,2,0],[0,0,2]);
Result is $\left(\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}\right)$
normalvec([2,0,0],[0,0,2],[0,2,0]);
Result is $\left(-\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}, -\frac{\sqrt{3}}{3}\right)$

The vector of the perpendicular line which you gave to a plane is provided when you use distpp()

```
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([[2,0,0],[0,2,0],[0,0,2]]);
arrow3d([[0,0,0],dd*nv]);
Letter3d([dd*nv,"e","H"]);
```



Function planecoeff(coordinate1,coordinate2,coordinate3)

Description It demands coefficient a, b, c of plane equation $ax + by + cz = 1$ going along three points of coordinate 1 and coordinate 2 and coordinate 3.

Return value List [a,b,c]. When coefficiets not exist, "Warning! Cannot decide a coefficient." is displayed and return [i,i,i].

Function reflect3d(dlist,mirror)

Description Reflection of dlist. dlist is point or plot data or face data.

Return value Data of the same type as the first argument.

Function rotate3d(dlist,vec,angle,center)

Description Rotaate of dlist. dlist is point or plot data or face data.

Return value Data of the same type as the first argument.

Function rotate3d(dlist,vec,angle,center)

Description Translate of dlist. dlist is point or plot data or face data.

Return value Data of the same type as the first argument.

Function rotmatrix(vec)

Description Make rotation matrix from normal vectors.

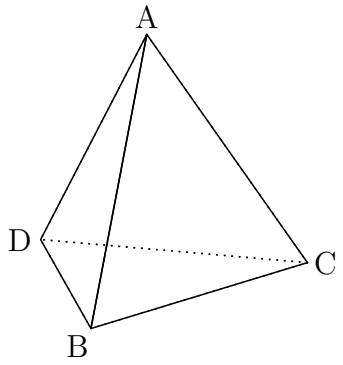
Return value List.

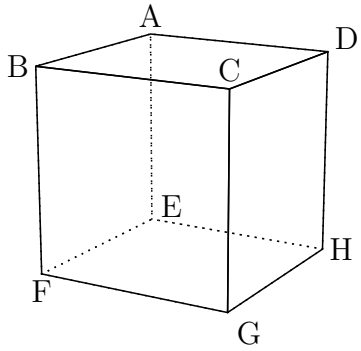
Function vertexrpolyhedron(n)

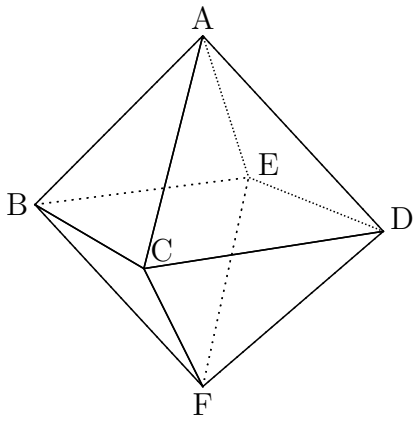
Description Acquire a list of tops of the regular polyhedron

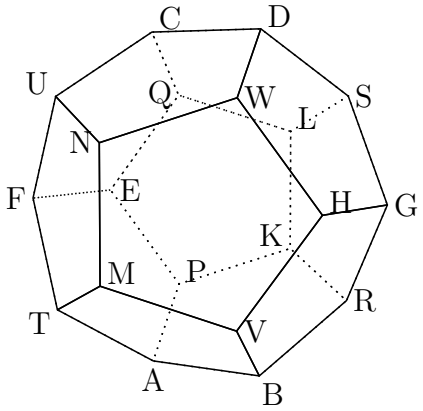
Description The value of n is one of 4,6,8,12,20.

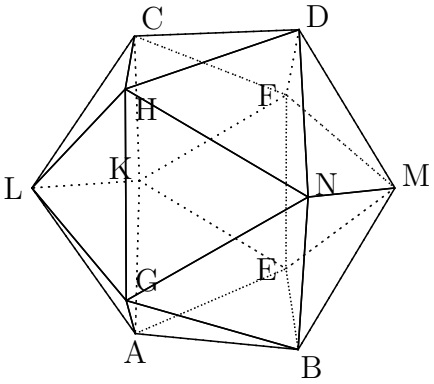
Return value list of vertexs of the regular polyhedron that touches the spherical surface of radius 1 internally. The turn of the vertexs are as follows. It is alphabetical order each.











Function val2tex(x)

Description Analyze numerical values with guess ()

Return value TeX character string.

Example

```
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$\left( "+hvstr_1+","+hvstr_2+","+hvstr_3+" \right)$"]);
```

4 Drawing

options

Density of the mesh The curved surface is described in mesh. This options appoints the number of process lines of this time.

The density of the mesh is appointed like "Mesh=[10,15]".

Concentration The concentration is appointed with real numbers from 0 to 1.

Ray tracing

Specify whether to add shadows in ray tracing. When shadowing "Rayon", do not shade when "Rayoff". The default setting is "Rayon"

Function grid(range1,range2,ne,option)

Description Display grid on xy plane.

Description Range 1 is the range of the x axis, and range 2 is the range of the y axis.

Return value none.

Function line3d(name,list ,option

Description This function draws a lint linking two points in list of the argument.

Return value list of 2nd argument.

Function arrow3d(name,list,option

Description This function draws an arrowed line by list of two points of coordinates.

Return value list of 2nd argument.

Example The next script draws an arrowed line linking two points of $(1, 3, 2), (-2, -1, -2)$.

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

Example The next script draw a straight line via two points of $(1, 3, 2), (-2, -1, -2)$ with 2 thickness red.

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

Function poly3d(name,list,option)

Description This function draws a polygon to link the point that I gave in list. The point to give in list does not need to be closed. It is closed automatically and is drawn.

Return value Plot data of the drawn polygon.

Example The next script draws a triangle to assume three points of $(1, 1, 1), (2, 2, 1), (0, 1, -1)$ a top.

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

Function plate3d(name,list,option)

Description This function applies a polygon to link the point that I gave in list.

Return value Plot data of the drawn polygon.

With the Rayoff option, ray tracing is not performed.

Example : The next script applies a triangle to assume three points of $(1, 1, 1), (2, 2, 1), (0, 1, -1)$ a top with red.

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

Function circle3d(name,center,normal vector,radius,option)

Description This function gives the center, a radius and a normal vector and draws the circle.

Return value Plotdata of circle.

Example : The next script describes the circle that central $(1, 1, 1)$, radius 2, a normal vector are (1,1,1) in thickness 2, red.

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

Function drawarc3d(name,center,normal vector,radius,range,option)

Description Draw an arc as a part of the circle drawn by giving the center, radius and normal vector.

Return value Plotdata of arc.

Example : Draw arc with center $(1, 1, 1)$, radius 2, normal vector (-1, 1, 1).

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

Function disc3d(center,nomal vector,radius,options)

Description This function gives the center, a radius and a normal vector and draws a disk.

Return value Plotdata of circle.

With the Rayoff option, ray tracing is not performed.

Example : The next script draws the disk that central $(1, 1, 1)$, radius 2, a normal vector are (1,1,1) at red

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

Function drawsphere(name,center,radius,opton)

Description This function describes the origin center, draw spherical surface of radius r.

Draw a spherical surface with gradation by using simple ray tracing. The spheres are divided into meshes and coloring is done.

The radius can also be specified as a list for the x axis, y axis, z axis direction.

Drawing takes time. If it takes too much time, try drawing with "Mesh = [10, 10]" as an option. If you can draw it, make the mesh finer. The initial value is "Mesh = [30, 20]".

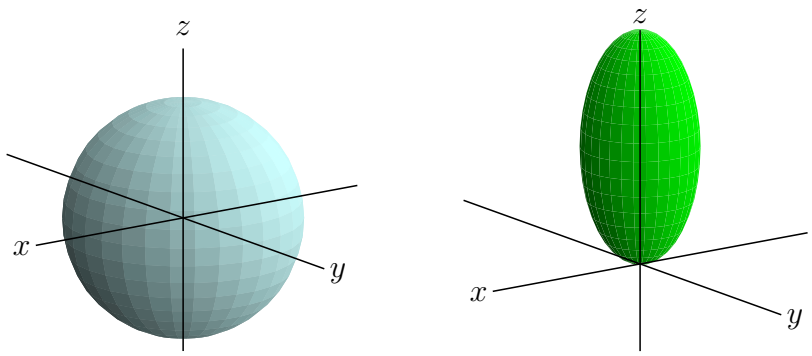
Return value none

Example : Origin center, spherical surface with radius 2 (left figure)

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

Draw a sphere with an ellipse shape in green (0, 0, 2) and a radius [1, 1, 2] in green (right figure)

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



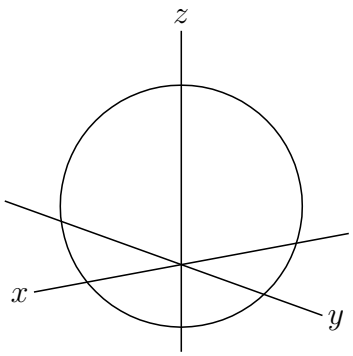
Function quasisphere(name,center,radius,fill,opton)

Description This function describes the origin center, draw a quasi spherical surface of radius r.

fill is a flag indicating whether or not to color. If it is 1, it paints. If 0 it does not paint. This argument can be omitted. The default is 1.

Return value Plotdata of circle.

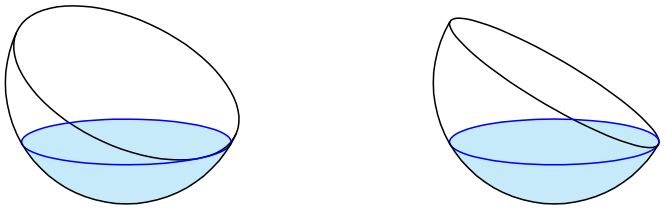
Example `quasisphere("1",[0,0,1],1]);`



It is faster than drawing with Sfbdparadata () and ExeccmdC () because it does not handle hidden lines. By using this, if you make a script like the following, you can see that pseudo sphere is sufficient.

```
pd1=quasisphere("1",[0,0,0],2,["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]"]);
```



When this script is executed, the number of intersection points int 1 and int 2 changes according to the viewpoint, so it is necessary to look at what is displayed on the console and set |p1, p2, p3, p4 |.

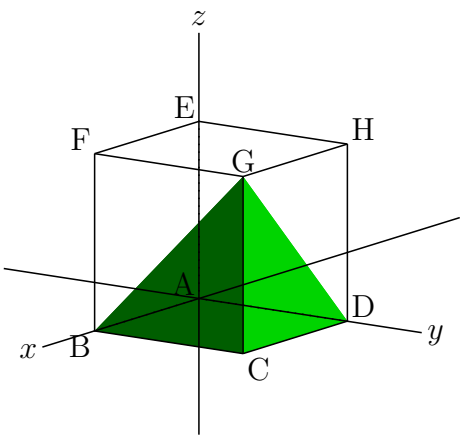
Function polyhedron(name,face data,option)

Description Drawing a polyhedron with colored surfaces.

Return value face data.

With the Rayoff option, ray tracing is not performed.

Example : Cube ABCD - EFGH is drawn in green with the plane cut through B, D and G.



For the vertices B, C, D, G, make a surface list as follows.

First, add numbers 1, 2, 3, 4 to vertices B, C, D and G.

The plane BCG is B, C, G counterclockwise as seen from the outside, so [1, 2, 4]

Since surface CDG is similarly C, D, G, [2, 3, 4]

Since the planes DGB are similarly D, B, G, [3, 1, 4]

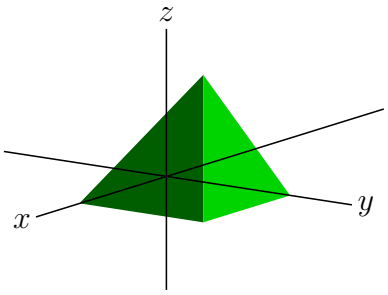
Bottom BCD is B, D, C counterclockwise as seen from the outside [1, 3, 2]

Therefore, with the coordinates of B, C, D, G as p1, p2, p3, p4, make surface data |fd | as follows.

```
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]]];
```

Using this surface data, the cone is drawn as follows

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

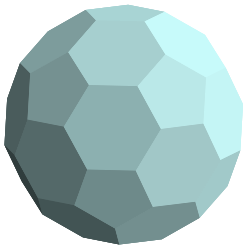


Example : Draw polyhedron using polyhedrons obj polyhedron data by Kobayashi, Suzuki, Mitani. Data is a regular polyhedron, semi-regular polyhedron, Johnson's solid,

http://mitani.cs.tsukuba.ac.jp/polyhedron/index.html

Specify the path to the folder polyhedrons obj storing this data with `Setdirectory ()` , and read it with |Readobj () |. For example, when placed in a work directory (fig folder)

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



Note that `fd = [2 * polydt - 1, polydt - 2]` doubles the vertex coordinates.

If you are drawing an edge, add the following.

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

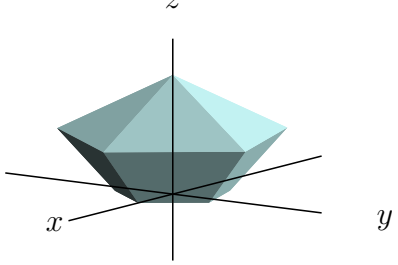
Function convexhedron(name,list ,magnification,option)

Description Draw a convex polyhedron with a vertex list and a surface painted. Magnification is the magnification to the size actually drawn for the vertex list. If it is 1 it can be omitted.

Return value face data

Example : A convex polyhedron whose bottom is a pentagon

```
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)
```



If you are drawing an edge, use the return value as follows.

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

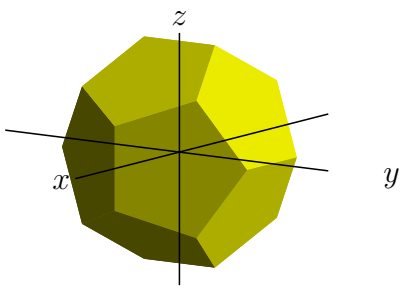
Function rpolyhedron(name, number of face ,radius,option)

Description Draw a regular polyhedron with a color-painted face. There are five kinds of regular polyhedrons, and data is incorporated in Spacekc. vertex coordinates of regular polyhedron. Using this, you can draw a regular polyhedron by specifying the number of faces and the radius of the circumscribed sphere.

Return value face data.

Example : Draw a regular dodecahedron painted with yellow face

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



If you are drawing an edge, use the return value as follows.

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

Reference : Size of regular polyhedron

In rpolyhedron (), draw with the size inscribed in the sphere of the specified radius. Here, the relationship between the radius and the side length is mentioned. $\phi = \dfrac{1+\sqrt{5}}{2}$

	$\dfrac{2\sqrt{6}}{3}$	$\dfrac{1}{3}$
	$\dfrac{2}{\sqrt{3}}$	$\dfrac{1}{\sqrt{3}}$
	$\sqrt{2}$	$\dfrac{1}{\sqrt{3}}$
12	$\dfrac{2}{\sqrt{3}\phi}$	$\sqrt{\dfrac{\phi^3}{3\sqrt{5}}}$
20	$\dfrac{2}{\sqrt{\sqrt{5}\phi}}$	$\dfrac{\phi^2}{\sqrt{3\sqrt{5}\phi}}$

Example : Draw a regular hexahedron with a side length of 2

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

Function frustum(name,n,r1,r2,h,option)

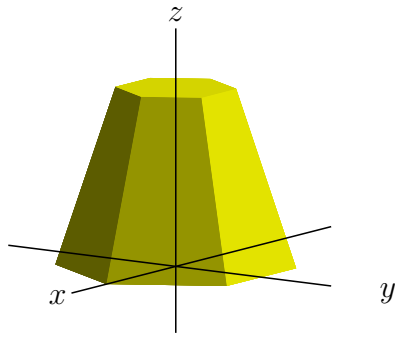
Description Put the face of a regular pyramid. n : Number of corners

r1,r2 : Radius of the circumscribed circle of the upper base and the lower base

h : height

ExampleDraw a regular hexagonal pyramid

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

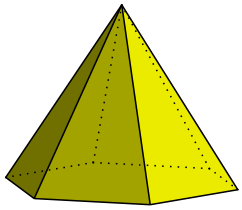


If you are drawing an edge, use the return value as follows.

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

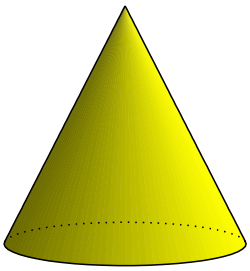
Example :

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



Example : It becomes almost a cone when increasing the number of corners. The contour line is drawn as a curved surface with Sfbdparadata.

```
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");
```



If it is the same size of the upper base and the lower base, it becomes a cylinder.

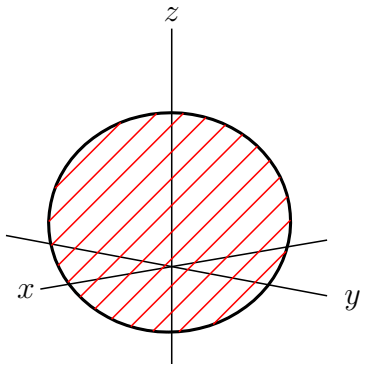
Function hatch3d(name, ,PD,option)

Description Hatch the closed curve. The closed curve is such as poly3d (), circle3d (); Unlike KeTCindy 's Hatchdata (), only the closed curve is the target, so the direction is not in the argument. A color designation can be put in option, and if there is a color designation, hatch is applied with that color. You can not hatch multiple areas.

Return value none

Example : Hatch the circle.

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


z
x
y