

# PSTricks

---

## 3D plots: PST-3dplot

A PSTricks package for drawing 3d objects, v1.83

November 24, 2008

Package author(s):  
**Herbert Voß**

## Contents

<b>1 The Parallel projection</b>	<b>5</b>
<b>2 Options</b>	<b>6</b>
<b>3 Coordinates and Axes</b>	<b>6</b>
3.1 Ticks and labels . . . . .	10
3.2 Experimental features . . . . .	12
<b>4 Rotation</b>	<b>15</b>
<b>5 Plane Grids</b>	<b>21</b>
<b>6 Put</b>	<b>24</b>
6.1 pstThreeDPut . . . . .	24
6.2 pstPlanePut . . . . .	24
<b>7 Nodes</b>	<b>27</b>
<b>8 Dots</b>	<b>27</b>
<b>9 Lines</b>	<b>28</b>
<b>10 Triangles</b>	<b>30</b>
<b>11 Squares</b>	<b>31</b>
<b>12 Boxes</b>	<b>32</b>
<b>13 Ellipses and circles</b>	<b>36</b>
13.1 Options . . . . .	37
13.2 Ellipse . . . . .	37
13.3 Circle . . . . .	38
<b>14 \pstIIIDCylinder</b>	<b>40</b>
<b>15 \psCylinder</b>	<b>42</b>
<b>16 \pstParaboloid</b>	<b>45</b>
<b>17 Spheres</b>	<b>47</b>
<b>18 Mathematical functions</b>	<b>48</b>
18.1 Function $f(x, y)$ . . . . .	48
18.2 Parametric Plots . . . . .	49
<b>19 Plotting data files</b>	<b>54</b>
19.1 \fileplotThreeD . . . . .	55
19.2 \dataplotThreeD . . . . .	57
19.3 \listplotThreeD . . . . .	58

---

<b>20 Utility macros</b>	<b>59</b>
20.1 Rotation of three dimensional coordinates . . . . .	59
20.2 Transformation of coordinates . . . . .	61
20.3 Adding two vectors . . . . .	61
20.4 Subtract two vectors . . . . .	61
<b>21 PDF output</b>	<b>61</b>
<b>22 FAQ</b>	<b>62</b>
<b>References</b>	<b>62</b>

The well known `pstricks` package offers excellent macros to insert more or less complex graphics into a document. `pstricks` itself is the base for several other additional packages, which are mostly named `pst-xxxx`, like `pst-3dplot`. There exist several packages for plotting three dimensional graphical objects. `pst-3dplot` is similar to the `pst-plot` package for two dimensional objects and mathematical functions.

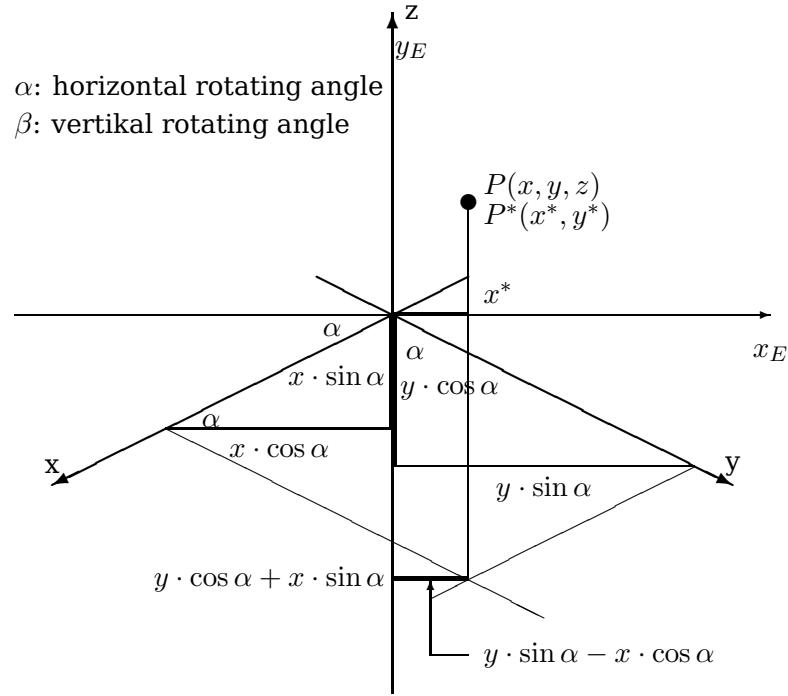
This version uses the extended keyval package `xkeyval`, so be sure that you have installed this package together with the special one `pst-xkey` for PSTricks. The `xkeyval` package is available at [CTAN:/macros/latex/contrib/xkeyval/](http://CTAN:/macros/latex/contrib/xkeyval/). It is also important that after `pst-3dplot` no package is loaded, which uses the old keyval interface.

Thanks for feedback and contributions to:

Bruce Burlton | Christophe Jorssen | Chris Kulewicz | Darrell Lamm | Rolf Niepraschk  
| Michael Sharpe | Uwe Siart | Thorsten Suhling

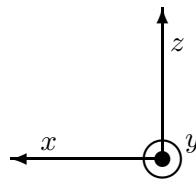
## 1 The Parallel projection

Figure 1 shows a point  $P(x, y, z)$  in a three dimensional coordinate system  $(x, y, z)$  with a transformation into  $P^*(x^*, y^*)$ , the Point in the two dimensional system  $(x_E, y_E)$ .



**Figure 1:** Lengths in a three dimensional System

The angle  $\alpha$  is the horizontal rotation with positive values for anti clockwise rotations of the 3D coordinates. The angle  $\beta$  is the vertical rotation (orthogonal to the paper plane). In figure 2 we have  $\alpha = \beta = 0$ . The y-axis comes perpendicular out of the paper plane. Figure 3 shows the same for another angle with a view from the side, where the x-axis shows into the paper plane and the angle  $\beta$  is greater than 0 degrees.



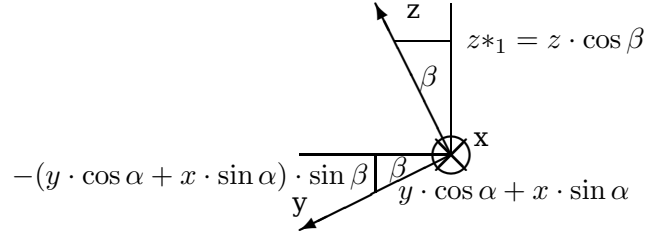
**Figure 2:** Coordinate System for  $\alpha = \beta = 0$  (y-axis comes out of the paper plane)

The two dimensional x coordinate  $x^*$  is the difference of the two horizontal lengths  $y \cdot \sin \alpha$  und  $x \cdot \cos \alpha$  (figure 1):

$$x^* = -x \cdot \cos \alpha + y \cdot \sin \alpha \quad (1)$$

The z-coordinate is unimportant, because the rotation comes out of the paper plane, so we have only a different  $y^*$  value for the two dimensional coordinate but no other

$x^*$  value. The  $\beta$  angle is well seen in figure 3 which derives from figure 2, if the coordinate system is rotated by  $90^\circ$  horizontally to the left and vertically by  $\beta$  also to the left.



**Figure 3:** Coordinate System for  $\alpha = 0$  and  $\beta > 0$  ( $x$ -axis goes into the paper plane)

The value of the perpendicular projected  $z$  coordinate is  $z^* = z \cdot \cos \beta$ . With figure 3 we see, that the point  $P(x, y, z)$  runs on an elliptical curve when  $\beta$  is constant and  $\alpha$  changes continuously. The vertical alteration of  $P$  is the difference of the two "perpendicular" lines  $y \cdot \cos \alpha$  and  $x \cdot \sin \alpha$ . These lines are rotated by the angle  $\beta$ , so we have to multiply them with  $\sin \beta$  to get the vertical part. We get the following transformation equations:

$$\begin{aligned} x_E &= -x \cos \alpha + y \sin \alpha \\ y_E &= -(x \sin \alpha + y \cos \alpha) \cdot \sin \beta + z \cos \beta \end{aligned} \quad (2)$$

or written in matrix form:

$$\begin{pmatrix} x_E \\ y_E \end{pmatrix} = \begin{pmatrix} -\cos \alpha & \sin \alpha & 0 \\ -\sin \alpha \sin \beta & -\cos \alpha \sin \beta & \cos \beta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (3)$$

All following figures show a grid, which has only the sense to make things clearer.

## 2 Options

All options which are set with `\psset` are global and all which are passed with the optional argument of a macro are local for this macro. This is an important fact for setting the angles `Alpha` and `Beta`. Mostly all macro need these values, this is the reason why they should be set with `\psset` and not part of an optional argument.

## 3 Coordinates and Axes

`pst-3dplot` accepts cartesian or spherical coordinates. In both cases there must be three parameters:  $(x, y, z)$  or alternatively  $(r, \phi, \theta)$ , where  $r$  is the radius,  $\phi$  the longitude angle and  $\theta$  the latitude angle. For the spherical coordinates set the option `SphericalCoord=true`. Spherical coordinates are possible for all macros where three dimensional coordinates are expected, except for the plotting functions (math functions and data records). Maybe that this is also interesting for someone, then let me know.

Unlike coordinates in two dimensions, three dimensional coordinates may be specified using PostScript code, which need not be preceded by `!`. For example, assuming `\def\nA{2}`, `(1,0,2)` and `(90 cos, 100 100 sub, \nA\space 2 div 1 add)` specify the same point. (Recall that a `\space` is required after a macro that will be expanded into PostScript code, as  $\text{\TeX}$  absorbs the space following a macro.)

The syntax for drawing the coordinate axes is

`\pstThreeDCoor [Options]`

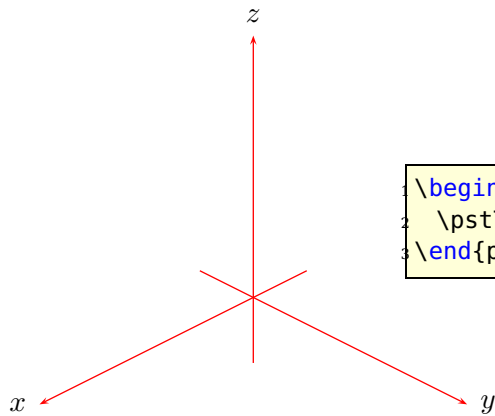
The only special option is `drawing=true|false`, which enables the drawing of the coordinate axes. The default is `true`. In nearly all cases the `\pstThreeDCoor` macro must be part of any drawing to initialize the 3d-system. If `drawing` is set to `false`, then all ticklines options are also disabled.

Without any options we get the default view with the in table 1 listed options with the predefined values.

**Table 1:** All new parameters for `pst-plot`

Name	Type	Default
Alpha	<angle>	45
Beta	<angle>	30
xMin	<value>	-1
xMax	<value>	4
yMin	<value>	-1
yMax	<value>	4
zMin	<value>	-1
zMax	<value>	4
nameX	<string>	$x$
spotX	<angle>	180
nameY	<string>	$y$
spotY	<angle>	0
nameZ	<string>	$z$
spotZ	<angle>	90
IIIDticks	false true	false
IIIDlabels	false true	false
Dx	<value>	1
Dy	<value>	1
Dz	<value>	1
IIIDxTicksPlane	xy xz yz	xy
IIIDyTicksPlane	xy xz yz	yz
IIIDzTicksPlane	xy xz yz	yz
IIIDticksize	<value>	0.1
IIIDxticksep	<value>	-0.4
IIIDyticksep	<value>	-0.2
IIIDzticksep	<value>	0.2
RotX	<angle>	0
RotY	<angle>	0
RotZ	<angle>	0

Name	Type	Default
RotAngle	<angle>	0
xRotVec	<angle>	0
yRotVec	<angle>	0
zRotVec	<angle>	0
RotSequence	xyz xzy yxz yzx zxy zyx quaternion	xyz
RotSet	set concat keep	set
eulerRotation	true false	false



```

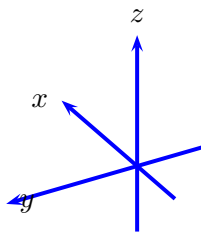
1 \begin{pspicture}(-3,-2.5)(3,4.25)
2   \pstThreeDCoor
3 \end{pspicture}

```

There are no restrictions for the angles and the max and min values for the axes; all `pstricks` options are possible as well. The following example changes the color and the width of the axes.

The angles Alpha and Beta are important to all macros and should always be set with `psset` to make them global to all other macros. Otherwise they are only local inside the macro to which they are passed.

Alpha is the horizontal and Beta the vertical rotation angle of the Cartesian coordinate system.

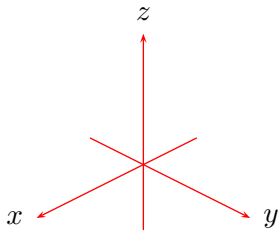


```

1 \begin{pspicture}(-2,-1.25)(1,2.25)
2   \pstThreeDCoor[linewidth=1.5pt,linecolor=blue,
3     xMax=2,yMax=2,zMax=2,
4     Alpha=-60,Beta=30]
5 \end{pspicture}

```

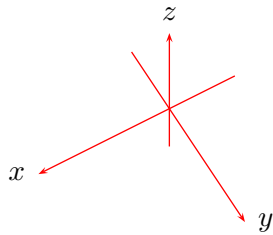




```

1 \begin{pspicture}(-2,-2)(2,2)
2   \pstThreeDCoor[xMax=2,yMax=2,zMax=2]
3 \end{pspicture}

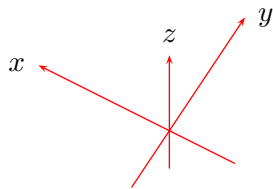
```



```

1 \begin{pspicture}(-2,-2)(2,2)
2   \pstThreeDCoor[xMax=2,yMax=2,zMax=2,
3     Alpha=30,Beta=60]
4 \end{pspicture}

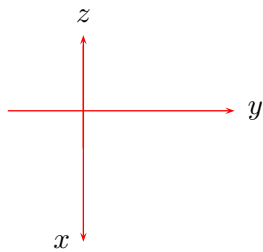
```



```

1 \begin{pspicture}(-2,-2)(2,2)
2   \pstThreeDCoor[xMax=2,yMax=2,zMax=2,
3     Alpha=30,Beta=-60]
4 \end{pspicture}

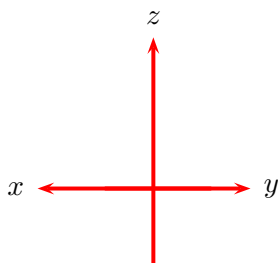
```



```

1 \begin{pspicture}(-2,-2)(2,2)
2   \pstThreeDCoor[
3     xMax=2,yMax=2,zMax=2,
4     Alpha=90,Beta=60]
5 \end{pspicture}

```



```

1 \begin{pspicture}(-2,-2)(2,2)
2   \pstThreeDCoor[linewidth=1.5pt,
3     xMax=2,yMax=2,zMax=2,
4     Alpha=40,Beta=0]
5 \end{pspicture}

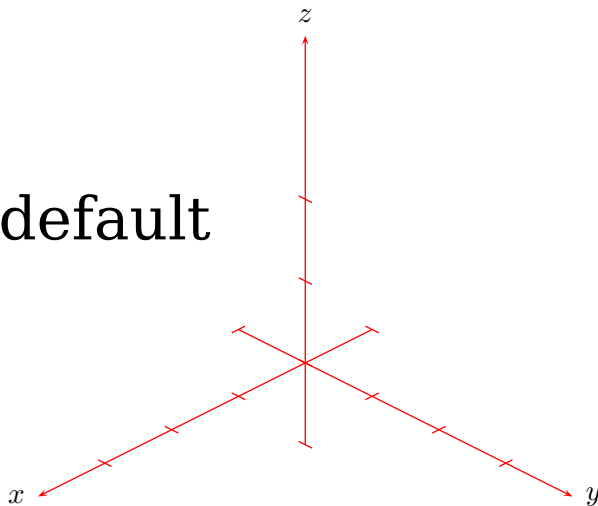
```

### 3.1 Ticks and labels

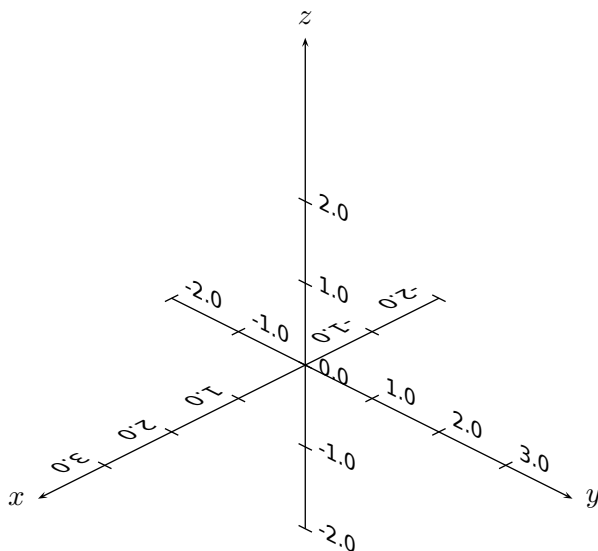
With the option `IIIDticks` the axes get ticks and with `IIIDlabels` labels. Without ticks are not possible. There are several options to place the labels in right plane to get an optimal view. The view of the ticklabels can be changed by redefining the macro

```
\def\psxyzlabel#1{\bgroup\footnotesize\textsf{#1}\egroup}
```

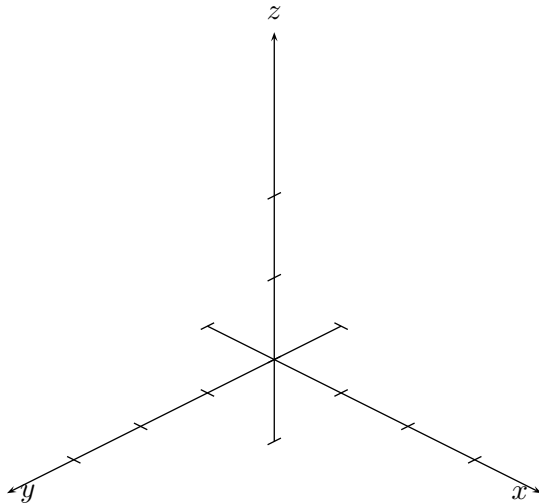
default



```
\begin{pspicture}(-3,-2.5)(3,4)
\pstThreeDCoor[IIIDticks]%
\pstThreeDPut(3,0,3){\Huge default}
\end{pspicture}
```



```
\begin{pspicture}(-3,-2.5)(3,4)
\pstThreeDCoor[linecolor=black,%
IIIDticks,IIIDlabels,
xMin=-2,yMin=-2,zMin=-2]%
\end{pspicture}
```

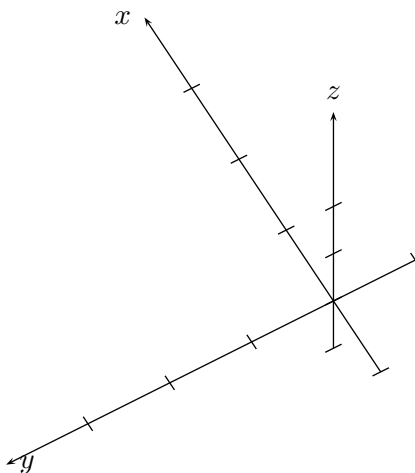


```

1 \begin{pspicture}(-3,-2.5)(3,4)
2   \pstThreeDCoor[linecolor=black,%
3     IIIDticks,IIIDzTicksPlane=yz,IIIDzticksep
4     =-0.2,%
5     IIIDxTicksPlane=yz,,IIIDxticksep=-0.2,%
6     IIIDyTicksPlane=xy,,IIIDyticksep=0.2,%
7     Dx=2,Dy=1,Dz=0.25,Alpha=-135,Beta=-30]%
8 \end{pspicture}

```

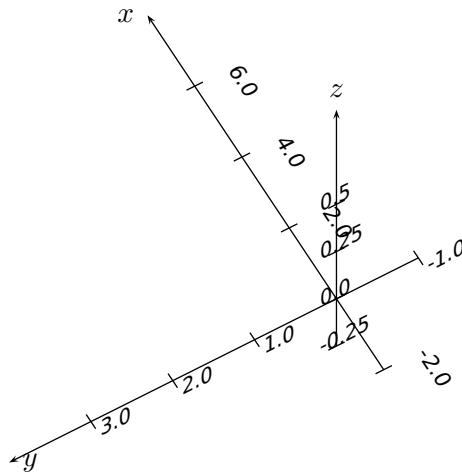
The following example shows a wrong placing of the labels, the planes should be changed.



```

1 \psset{Alpha=-60,Beta=60}
2 \begin{pspicture}(-4,-2.25)(1,3)
3   \pstThreeDCoor[linecolor=black,%
4     IIIDticks,Dx=2,Dy=1,Dz=0.25]%
5 \end{pspicture}

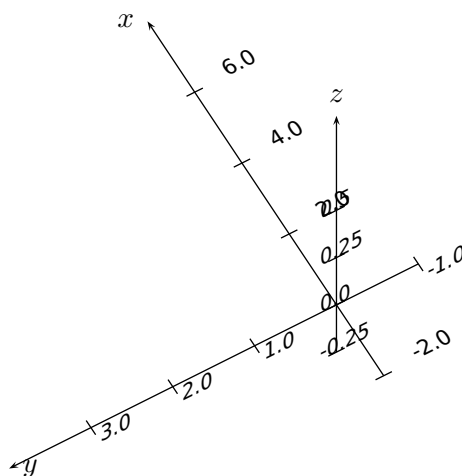
```



```

1 \psset{Alpha=-60,Beta=60}
2 \begin{pspicture}(-4,-2.25)(1,3)
3   \pstThreeDCoor[linecolor=black,%
4     IIDticks,IIDlabels,
5     plane corr=normal,
6     Dx=2,Dy=1,Dz=0.25]%
7 \end{pspicture}

```



```

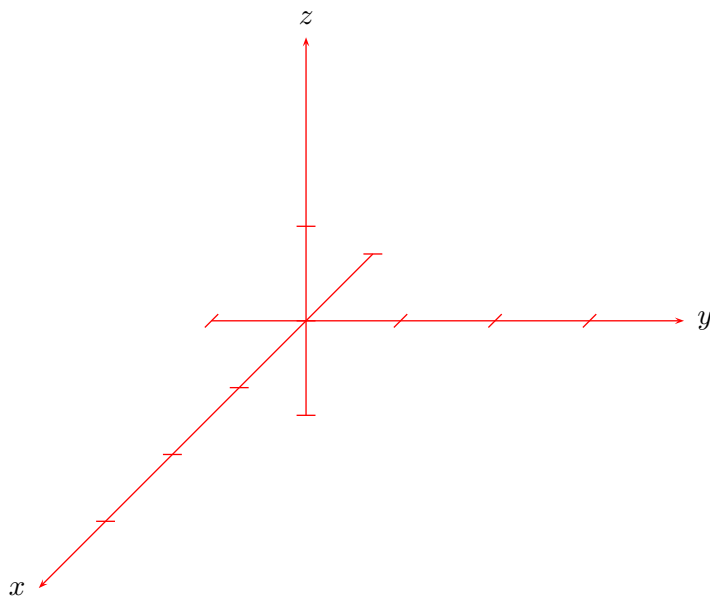
1 \psset{Alpha=-60,Beta=60}
2 \begin{pspicture}(-4,-2.25)(1,3)
3   \pstThreeDCoor[linecolor=black,%
4     IIDticks,IIDlabels,
5     plane corr=xyrot,
6     Dx=2,Dy=1,Dz=0.25]%
7 \end{pspicture}

```

## 3.2 Experimental features

All features are as long as they are not really tested called experimental. With the optional argument `coorType`, which is by default 0, one can change the the viewing of the axes and all other three dimensional objects.

With `coorType=1` the y-z-axes are orthogonal and the angle between x- and y-axis is Alpha. The angle Beta is not valid.

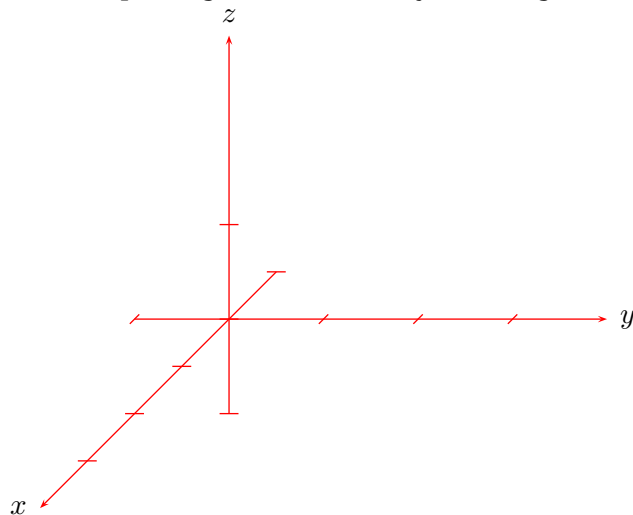


```

1 \psset{coorType=1,Alpha=135}
2 \begin{pspicture}(-2,-3)(3,3)
3 \pstThreeDCoor[IIIDticks,zMax=3]%
4 \end{pspicture}

```

With `coorType=2` the y-z-axes are orthogonal and the angle between x- and y-axis is always 135 degrees and the x-axis is shortened by a factor of  $1/\sqrt{2}$ . The angle Alpha is only valid for placing the ticks, if any. The angle Beta is not valid.

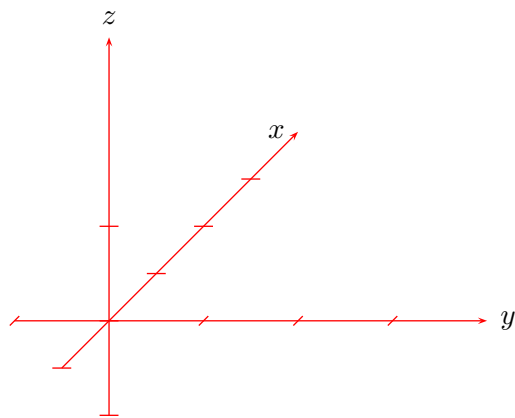


```

1 \psset{coorType=2,Alpha=90,
2       IIIDxTicksPlane=yz}
3 \begin{pspicture}(-2,-2)(3,3)
4 \pstThreeDCoor[IIIDticks,zMax=3]%
5 \end{pspicture}

```

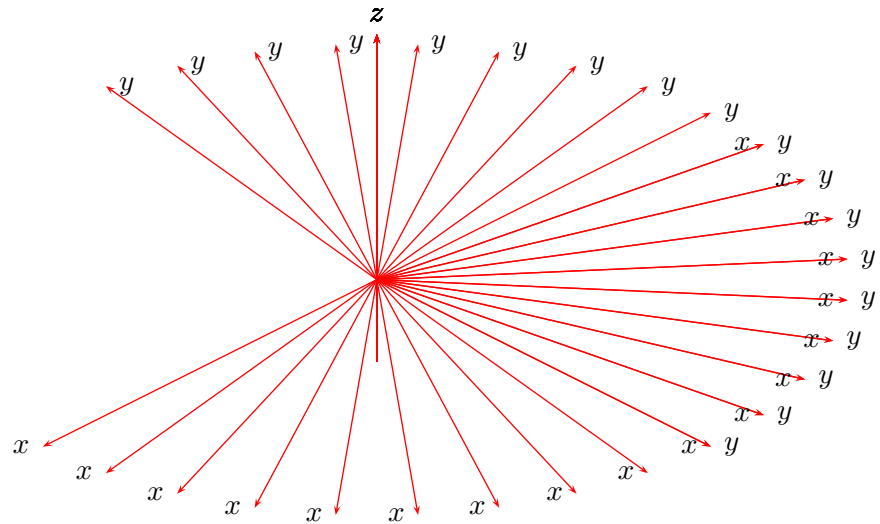
With `coorType=3` the y-z-axes are orthogonal and the angle between x- and y-axis is always 45 degrees and the x-axis is shortened by a factor of  $1/\sqrt{2}$ . The angle Alpha is only valid for placing the ticks, if any. The angle Beta is not valid.



```
1 \psset{coorType=3,Alpha=90,  
2   IIIDxTicksPlane=yz}  
3 \begin{pspicture}(-2,-2)(3,3)  
4 \pstThreeDCoor[IIIDticks,zMax=3]%  
5 \end{pspicture}
```

## 4 Rotation

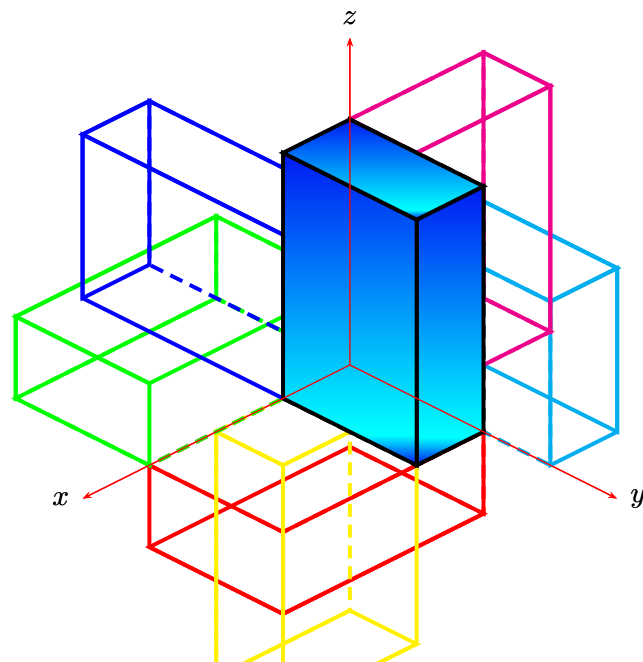
The coordinate system can be rotated independent from the given Alpha and Beta values. This makes it possible to place the axes in any direction and any order. There are the three options RotX, RotY, RotZ and an additional one for the rotating sequence (LkeywordrotSequence), which can be any combination of the three letters xyz.



```

1 \begin{pspicture}(-6,-3)(6,3)
2   \multido{\iA=0+10}{18}{%
3     \pstThreeDCoor[RotZ=\iA,xMin=0,xMax=5,yMin=0,yMax=5,zMin=-1,zMax=3]%
4   }
5 \end{pspicture}

```

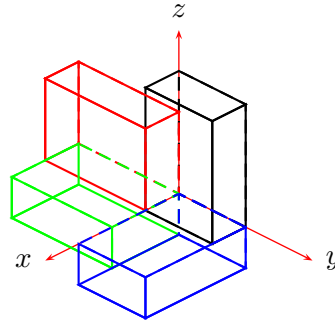


```

1 \psset{unit=2,linewidth=1.5pt,drawCoor=false}
2 \begin{pspicture}(-2,-1.5)(2,2.5)%
3   \pstThreeDCoor[xMin=0,xMax=2,yMin=0,yMax=2,zMin=0,zMax=2]%
4   \pstThreeDBox[RotX=90,RotY=90,RotZ=90,%
5     linecolor=red](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
6   \pstThreeDBox[RotSequence=xzy,RotX=90,RotY=90,RotZ=90,%
7     linecolor=yellow](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
8   \pstThreeDBox[RotSequence=zyx,RotX=90,RotY=90,RotZ=90,%
9     linecolor=green](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
10  \pstThreeDBox[RotSequence=zxy,RotX=90,RotY=90,RotZ=90,%
11    linecolor=blue](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
12  \pstThreeDBox[RotSequence=yxz,RotX=90,RotY=90,RotZ=90,%
13    linecolor=cyan](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
14  \pstThreeDBox[RotSequence=yzx,RotX=90,RotY=90,RotZ=90,%
15    linecolor=magenta](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
16  \pstThreeDBox[fillstyle=gradient,RotX=0](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
17  \pstThreeDCoor[xMin=0,xMax=2,yMin=0,yMax=2,zMin=0,zMax=2]%
18 \end{pspicture}%

```



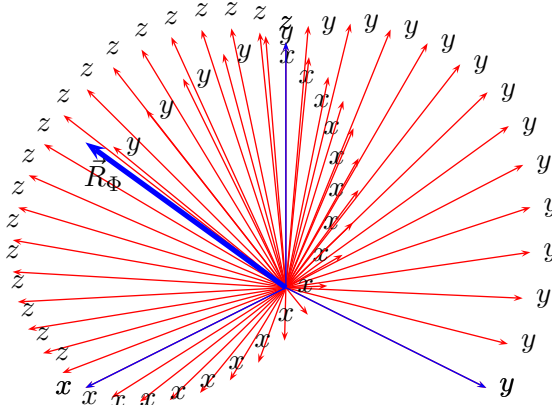


```

1 \begin{pspicture}(-2,-1.5)(2,2.5)%
2 \pstThreeDCoor[xMin=0,xMax=2,yMin=0,yMax=2,zMin=0,zMax=2]%
3 \pstThreeDBox(0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
4 \pstThreeDBox[RotX=90,linecolor=red](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
5 \pstThreeDBox[RotX=90,RotY=90,linecolor=green](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
6 \pstThreeDBox[RotX=90,RotY=90,RotZ=90,linecolor=blue](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
7 \end{pspicture}%

```

It is sometimes more convenient to rotate the coordinate system by specifying a *single* angle of rotation `RotAngle` (in degrees) about a vector whose coordinates are `xRotVec`, `yRotVec`, and `zRotVec` using the quaternion option for `RotSequence`.



```

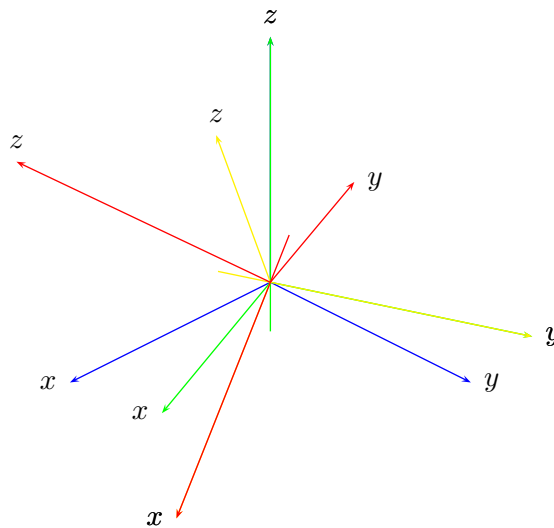
1 \begin{pspicture}(-3,-1.8)(3,3)
2 \multido{\iA=0+10}{18}{%
3 \pstThreeDCoor[linecolor=red, RotSequence=quaternion, RotAngle=\iA, xRotVec=3,yRotVec
   =0,zRotVec=3,
4 xMin=0,xMax=3, yMin=0,yMax=3, zMin=0,zMax=3]}
5 \pstThreeDCoor[linecolor=blue, RotSequence=quaternion, RotAngle=0, xRotVec=0, yRotVec
   =0, zRotVec=1,
6 xMin=0,xMax=3, yMin=0,yMax=3, zMin=0,zMax=3]
7 \pstThreeDLine[linecolor=blue, linewidth=2pt, arrows=->](0,0,0)(3,0,3)
8 \uput[0](-2.28,1.2){$\vec{R}_{\Phi}$}
9 \end{pspicture}

```

Rotations of the coordinate system may be “accumulated” by applying successive rotation sequences using the `RotSet` variable, which is set either as a `pst-3dplot`

object's optional argument, or with a `\psset[pst-3dplot]{RotSet=value}` command. The usual  $\text{\TeX}$  scoping rules for the value of `RotSet` hold. The following are valid values of `RotSet`:

- `set`: Sets the rotation matrix using the rotation parameters. This is the default value for `RotSet` and is what is used if `RotSet` is not set as an option for the `pst-3dplot` object, or if not previously set within the object's scope by a `\psset[pst-3dplot]{RotSet=val}` command.
- `concat`: Concatenates the current rotation matrix with a the new rotation that is defined by the rotation parameters. This option is most useful when multiple `\pstThreeDCoor` calls are made, with or without actual plotting of the axes, to accumulate rotations. A previous value of `RotSet=set` must have been made!
- `keep`: Keeps the current rotation matrix, ignoring the rotation parameters. Mostly used internally to eliminate redundant calculations.



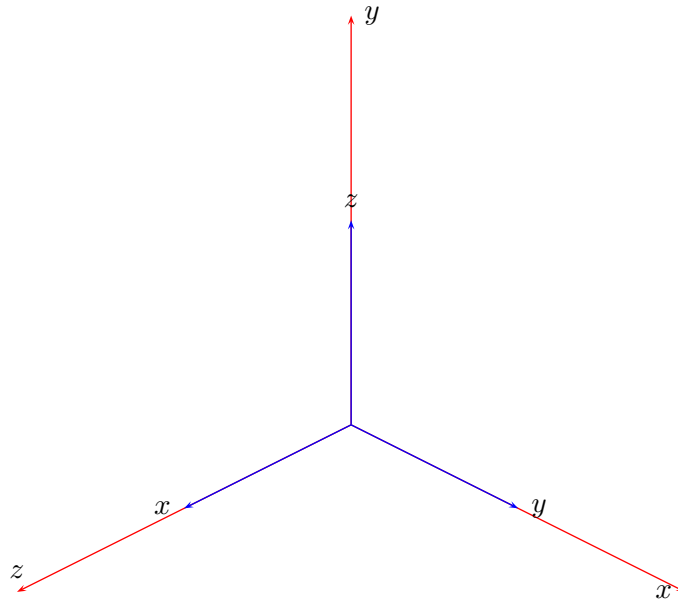
```

1 \begin{pspicture}(-3,-3)(3.6,3)
2 \pstThreeDCoor[linecolor=blue, RotSequence=quaternion, RotAngle=0, RotSet=set, xRotVec
   =0,yRotVec=0,zRotVec=1,
3 xMin=0,xMax=3, yMin=0,yMax=3, zMin=0,zMax=3]
4 \pstThreeDCoor[linecolor=green, RotSequence=quaternion, RotSet=concat, RotAngle=22.5,
   xRotVec=0,yRotVec=0,zRotVec=1,
5 xMin=0,xMax=3, yMin=0,yMax=3, zMin=-0.6,zMax=3]
6 \pstThreeDCoor[linecolor=yellow, RotSequence=quaternion, RotSet=concat, RotAngle=30,
   xRotVec=0,yRotVec=1,zRotVec=0,
7 xMin=0,xMax=3,yMin=-0.6,yMax=3, zMin=0,zMax=3]
8 \pstThreeDCoor[linecolor=red, RotSequence=quaternion, RotSet=concat, RotAngle=60,
   xRotVec=1,yRotVec=0,zRotVec=0,
9 xMin=-0.6,xMax=3, yMin=0,yMax=3, zMin=0,zMax=3]%
10 \end{pspicture}

```

By default, the rotations defined by `RotX`, `RotY`, and `RotZ` are rotations about the *original* coordinate system's,  $x$ ,  $y$ , or  $z$  axes, respectively. More traditionally, however,

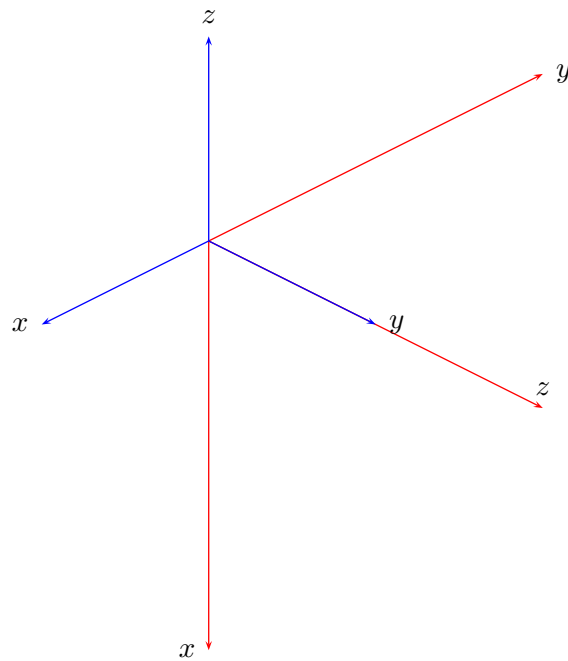
these rotation angles are defined as rotations about the rotated coordinate system's *current*,  $x$ ,  $y$ , or  $z$  axis. The `pst-3dplot` variable option `eulerRotation` can be set to `true` to activate Euler angle definitions; i.e., `eulerRotation=true`. The default is `eulerRotation=false`.



```

1 \begin{pspicture}(-4,-5)(6,5)
2   \pstThreeDCoor[linecolor=red, RotSequence=zyx, RotZ=90, RotY=90, RotX=0,
3     xmin=0, xmax=5, ymin=0, ymax=5, zmin=0, zmax=5]
4   \pstThreeDCoor[linecolor=blue, RotSequence=zyx, RotZ=0, RotY=0, RotX=0,
5     xmin=0, xmax=2.5, ymin=0, ymax=2.5, zmin=0, zmax=2.5]
6 \end{pspicture}

```



```

1 \begin{pspicture}(-3,-5)(7,5)
2   \pstThreeDCoor[eulerRotation=true, linecolor=red, RotSequence=zyx, RotZ=90, RotY=90,
3     RotX=0,
4     xmin=0,xmax=5, ymin=0,ymax=5, zmin=0,zmax=5]
5   \pstThreeDCoor[linecolor=blue, RotSequence=zyx, RotZ=0, RotY=0, RotX=0,
6     xmin=0,xmax=2.5, ymin=0,ymax=2.5, zmin=0,zmax=2.5]
7 \end{pspicture}

```

## 5 Plane Grids

`\pstThreeDPlaneGrid [Options] (xMin,yMin)(xMax,yMax)`

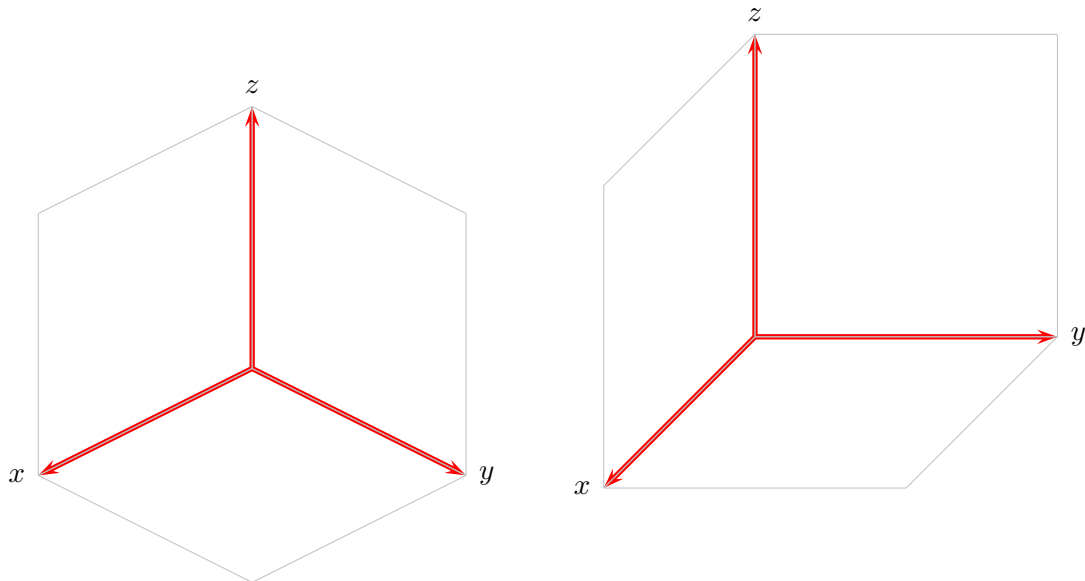
There are three additional options

**planeGrid** can be one of the following values: xy, xz, yz. Default is xy.

**subticks** Number of ticks. Default is 10.<sup>1</sup>

**planeGridOffset** a length for the shift of the grid. Default is 0.

. This macro is a special one for the coordinate system to show the units, but can be used in any way. `subticks` defines the number of ticklines for both axes and `xsubticks` and `ysubticks` for each one.



```

1 \begin{pspicture}(-4,-3.5)(5,4)
2   \pstThreeDCoor[xMin=0,yMin=0,zMin
3     =0,linewidth=2pt]
4   \psset{linewidth=0.1pt,linecolor=
5     lightgray}
6   \pstThreeDPlaneGrid(0,0)(4,4)
7   \pstThreeDPlaneGrid[planeGrid=xz
8     ](0,0)(4,4)
9   \pstThreeDPlaneGrid[planeGrid=yz
10     ](0,0)(4,4)
11 \end{pspicture}

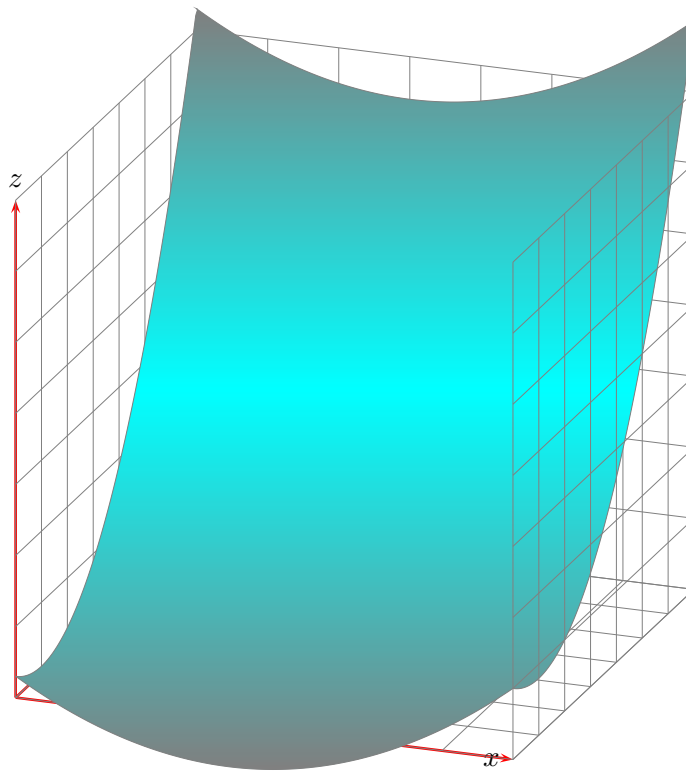
```

```

1 \begin{pspicture}(-3,-3.5)(5,4)
2   \psset{coorType=2}% set it
3   globally!
4   \pstThreeDCoor[xMin=0,yMin=0,zMin
5     =0,linewidth=2pt]
6   \psset{linewidth=0.1pt,linecolor=
7     lightgray}
8   \pstThreeDPlaneGrid(0,0)(4,4)
9   \pstThreeDPlaneGrid[planeGrid=xz
10     ](0,0)(4,4)
11 \pstThreeDPlaneGrid[planeGrid=yz
12     ](0,0)(4,4)
13 \end{pspicture}

```

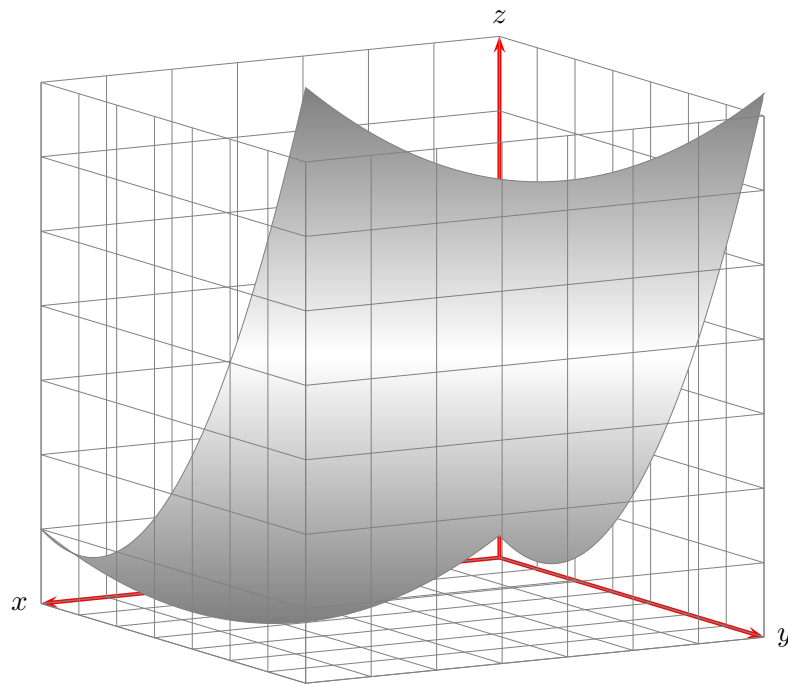
<sup>1</sup> This options is also defined in the package `pstricks-add`, so it is nessecary to to set this option locally or with the family option of `pst-xkey`, eg `\psset[pst-3dplot]{subticks=...}`



```

1 \begin{pspicture}(-1,-2)(10,10)
2   \psset{Beta=20,Alpha=160,subticks=7}
3   \pstThreeDCoord[xMin=0,yMin=0,zMin=0,xMax=7,yMax=7,zMax=7,linewidth=1pt]
4   \psset{linewidth=0.1pt,linecolor=gray}
5   \pstThreeDPlaneGrid(0,0)(7,7)
6   \pstThreeDPlaneGrid[planeGrid=xz,planeGridOffset=7](0,0)(7,7)
7   \pstThreeDPlaneGrid[planeGrid=yz](0,0)(7,7)
8   \pscustom[linewidth=0.1pt,fillstyle=gradient,gradbegin=gray,gradmidpoint=0.5,
9     plotstyle=curve]{%
10     \psset{xPlotpoints=200,yPlotpoints=1}
11     \psplotThreeD(0,7)(0,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
12       10 div }
13     \psset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
14     \psplotThreeD(7,7)(0,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
15       10 div }
16     \psset{xPlotpoints=200,yPlotpoints=1,drawStyle=xLines}
17     \psplotThreeD(7,0)(7,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
18       10 div }
19     \psset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
20     \psplotThreeD(0,0)(7,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
21       10 div }}
22   \pstThreeDPlaneGrid[planeGrid=yz,planeGridOffset=7](0,0)(7,7)
23 \end{pspicture}

```



```

1 \begin{pspicture}(-6,-2)(4,7)
2   \psset{Beta=10,Alpha=30,subticks=7}
3   \pstThreeDCoor[xMin=0,yMin=0,zMin=0,xMax=7,yMax=7,zMax=7,linewidth=1.5pt]
4   \psset{linewidth=0.1pt,linecolor=gray}
5   \pstThreeDPlaneGrid(0,0)(7,7)
6   \pstThreeDPlaneGrid[planeGrid=xz](0,0)(7,7)
7   \pstThreeDPlaneGrid[planeGrid=yz](0,0)(7,7)
8   \pscustom[linewidth=0.1pt,fillstyle=gradient,gradbegin=gray,gradend=white,
9     gradmidpoint=0.5,
10    plotstyle=curve]{%
11    \psset{xPlotpoints=200,yPlotpoints=1}
12    \psplotThreeD(0,7)(0,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
13      10 div }
14    \psset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
15    \psplotThreeD(7,7)(0,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
16      10 div }
17    \psset{xPlotpoints=200,yPlotpoints=1,drawStyle=xLines}
18    \psplotThreeD(7,0)(7,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
19      10 div }
20    \psset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
21    \psplotThreeD(0,0)(7,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add
22      10 div }}
23   \pstThreeDPlaneGrid[planeGrid=xz,planeGridOffset=7](0,0)(7,7)
24   \pstThreeDPlaneGrid[planeGrid=yz,planeGridOffset=7](0,0)(7,7)
25 \end{pspicture}

```

The equation for the examples is

$$f(x,y) = \frac{x^2 + 2y^2 - 6x - 4y + 3}{10}$$

## 6 Put

There exists a special option for the put macros:

`origin=lt|lB|lb|t|c|B|b|rt|rB|rb`

for the placing of the text or other objects.

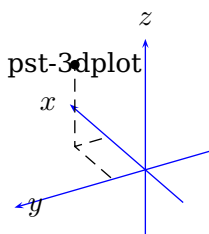


This works only well for the `\pstThreeDPut` macro. The default is `c` and for the `\pstPlanePut` the left baseline `lB`.

### 6.1 pstThreeDPut

The syntax is similar to the `\rput` macro:

`\pstThreeDPut[options](x,y,z){<any stuff>}`



```

1 \begin{pspicture}(-2,-1.25)(1,2.25)
2   \psset{Alpha=-60,Beta=30}
3   \pstThreeDCoor[linecolor=blue,%
4     xMin=-1,xMax=2,yMin=-1,yMax=2,zMin=-1,zMax=2]
5   \pstThreeDPut(1,0.5,1.25){pst-3dplot}
6   \pstThreeDDot[drawCoord=true](1,0.5,1.25)
7 \end{pspicture}

```

Internally the `\pstThreeDPut` macro defines the two dimensional node `temp@pstNode` and then uses the default `\rput` macro from `pstricks`. In fact of the perspective view of the coordinate system, the 3D dot must not be seen as the center of the printed stuff.

### 6.2 pstPlanePut<sup>2</sup>

The syntax of the `\pstPlanePut` is

`\pstPlanePut[plane=<2D plane>,planecorr=<Correction of plane's alignment>](x,y,z){Object}`

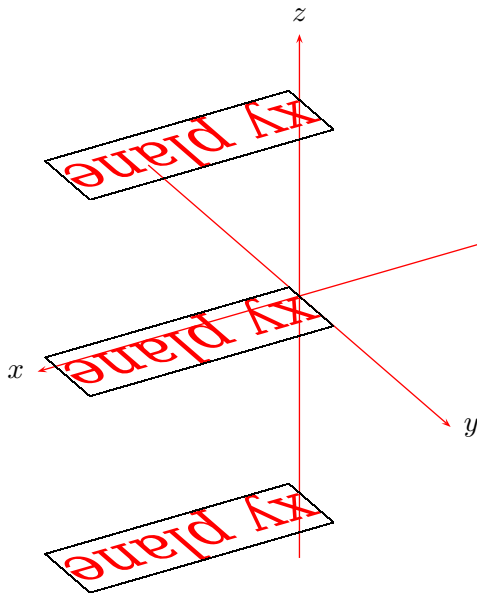
We have two parameters, `plane` and `planecorr`; both are optional. Let's start with the first parameter, `plane`. Possible values for the two dimensional plane are `xy` `xz` `yz`. If this parameter is missing then `plane=xy` is set. The first letter marks the positive direction for the width and the second for the height.

The object can be of any type, in most cases it will be some kind of text. The reference point for the object is the left side and vertically centered, often abbreviated as `lB`. The following examples show for all three planes the same textbox.

---

<sup>2</sup> Thanks to Torsten Suhling

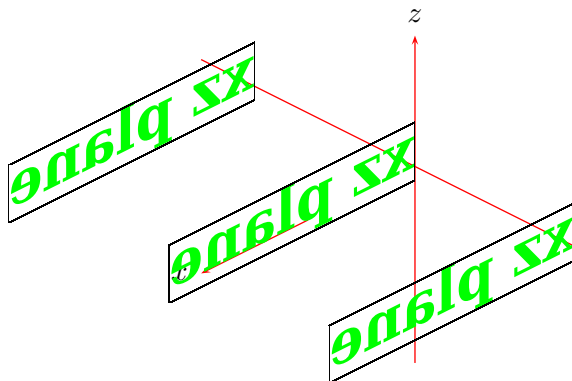




```

1 \begin{pspicture}(-4,-4)(3,4)
2   \psset{Alpha=30}
3   \pstThreeDCoor[xMin=-4,yMin=-4,zMin=-4]
4   \pstPlanePut[plane=xy](0,0,-3){\fbox{\Huge\red
5     xy plane}}
6   \pstPlanePut[plane=xy](0,0,0){\fbox{\Huge\red
7     xy plane}}
8   \pstPlanePut[plane=xy](0,0,3){\fbox{\Huge\red
9     xy plane}}
10 \end{pspicture}

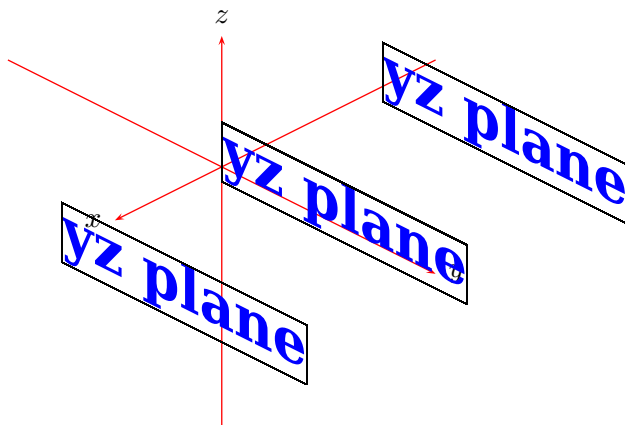
```



```

1 \begin{pspicture}(-5,-3)(2,3)
2   \pstThreeDCoor[xMin=2,yMin=-4,zMin=-3,zMax=2]
3   \pstPlanePut[plane=xz](0,-3,0){\fbox{\Huge\green\textbf{xz plane}}}
4   \pstPlanePut[plane=xz](0,0,0){\fbox{\Huge\green\textbf{xz plane}}}
5   \pstPlanePut[plane=xz](0,3,0){\fbox{\Huge\green\textbf{xz plane}}}
6 \end{pspicture}

```



```

1 \begin{pspicture}(-2,-4)(6,2)
2   \pstThreeDCoor[xMin=-4,yMin=-4,zMin=-4,xMax=2,
3     zMax=2]
4   \pstPlanePut[plane=yz](-3,0,0){\fbox{\Huge\blue\textbf{yz plane}}}
5   \pstPlanePut[plane=yz](0,0,0){\fbox{\Huge\blue\textbf{yz plane}}}
6   \pstPlanePut[plane=yz](3,0,0){\fbox{\Huge\blue\textbf{yz plane}}}
7 \end{pspicture}

```

The following examples use the origin option to show that there are still some problems with the xy-plane. The second parameter is planecorr. As first the values:

*off* Former and default behaviour; nothing will be changed. This value is set, when parameter is missing.

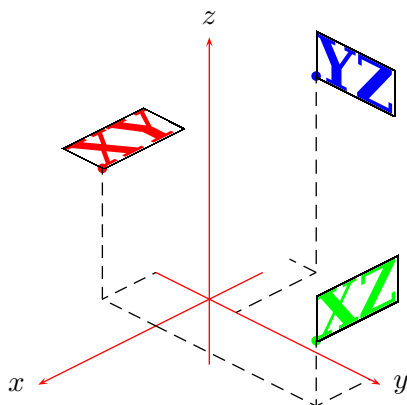
*normal* Default correction, planes will be rotated to be readable.

*xyrot* Additional correction for *xy* plane; bottom line of letters will be set parallel to the *y*-axis.

What kind of correction is meant? In the plots above labels for the *xy* plane and the *xz* plane are mirrored. This is not a bug, it's ... mathematics.

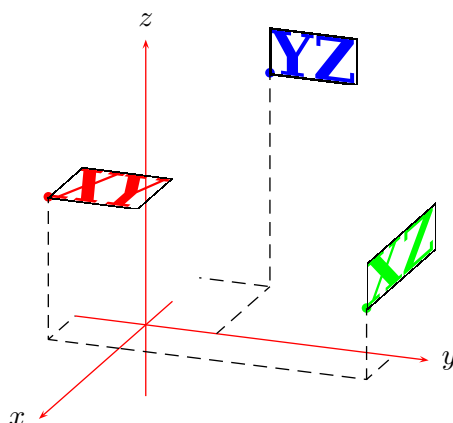
`\pstPlanePut` puts the labels on the plane of its value. That means, `plane=xy` puts the label on the *xy* plane, so that the *x* marks the positive direction for the width, the *y* for the height and the label *XY* plane on the top side of plane. If you see the label mirrored, you just look from the bottom side of plane ...

If you want to keep the labels readable for every view, i. e. for every value of Alpha and Beta, you should set the value of the parameter `planecorr` to `normal`; just like in next example:



```
\begin{pspicture}(-3,-2)(3,4)
\psset{origin=lb}
\pstThreeDCoor[xMax=3.2,yMax=3.2,zMax=4]
\pstThreeDDot[drawCoor=true,linecolor=red](1,-1,2)
\pstPlanePut[plane=xy,planecorr=normal](1,-1,2)
{\fbox{\Huge\red\textbf{XY}}}
\pstThreeDDot[drawCoor=true,linecolor=green](1,3,1)
\pstPlanePut[plane=xz,planecorr=normal](1,3,1)
{\fbox{\Huge\green\textbf{XZ}}}
\pstThreeDDot[drawCoor=true,linecolor=blue](-1.5,0.5,3)
\pstPlanePut[plane=yz,planecorr=normal](-1.5,0.5,3)
{\fbox{\Huge\blue\textbf{YZ}}}
\end{pspicture}
```

But, why we have a third value `xyrot` of `planecorr`? If there isn't an symmetrical view, – just like in this example – it could be useful to rotate the label for *xy*-plane, so that bottom line of letters is parallel to the *y* axis. It's done by setting `planecorr=xyrot`:



```
\begin{pspicture}(-2,-2)(4,4)
\psset{origin=lb}
\psset{Alpha=69.3,Beta=19.43}
\pstThreeDCoor[xMax=4,yMax=4,zMax=4]
\pstThreeDDot[drawCoor=true,linecolor=red](1,-1,2)
\pstPlanePut[plane=xy,planecorr=xyrot](1,-1,2)
{\fbox{\Huge\red\textbf{XY}}}
\pstThreeDDot[drawCoor=true,linecolor=green](1,3.5,1)
\pstPlanePut[plane=xz,planecorr=xyrot](1,3.5,1)
{\fbox{\Huge\green\textbf{XZ}}}
\pstThreeDDot[drawCoor=true,linecolor=blue](-2,1,3)
\pstPlanePut[plane=yz,planecorr=xyrot](-2,1,3)
{\fbox{\Huge\blue\textbf{YZ}}}
\end{pspicture}
```

## 7 Nodes

The syntax is

```
\pstThreeDNode(x,y,z){node name}
```

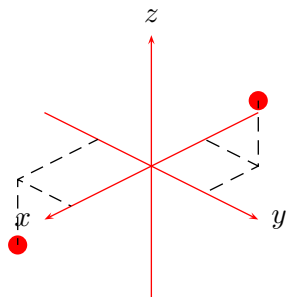
This node is internally a two dimensional node, so it cannot be used as a replacement for the parameters  $(x,y,z)$  of a 3D dot, which is possible with the `\psline` macro from `pst-plot`: `\psline{A}{B}`, where A and B are two nodes. It is still on the to do list, that it may also be possible with `pst-3dplot`. On the other hand it is no problem to define two 3D nodes C and D and then drawing a two dimensional line from C to D.

## 8 Dots

The syntax for a dot is

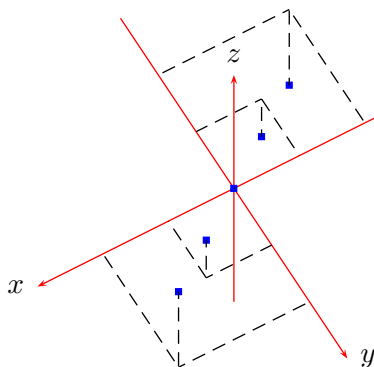
```
\pstThreeDDot [Options] (x,y,z)
```

Dots can be drawn with dashed lines for the three coordinates, when the option `drawCoor` is set to true. It is also possible to draw an unseen dot with the option `dotstyle=none`. In this case the macro draws only the coordinates when the `drawCoor` option is set to true.



```
\begin{pspicture}(-2,-2)(2,2)
2 \pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
3 \psset{dotstyle=*,dotstyle=2,linewidth=red,drawCoor=true}
4 \pstThreeDDot(-1,1,1)
5 \pstThreeDDot(1.5,-1,-1)
6 \end{pspicture}
```

In the following figure the coordinates of the dots are  $(a, a, a)$  where  $a$  is  $-2, -1, 0, 1, 2$ .



```
\begin{pspicture}(-3,-3.25)(2,3.25)
2 \psset{Alpha=30,Beta=60,dotstyle=square*,dotsize=3pt,%
3 linewidth=blue,drawCoor=true}
4 \pstThreeDCoor[xMin=-3,xMax=3,yMin=-3,yMax=3,zMin=-3,zMax=3]
5 \multido{\n=-2+1}{5}{\pstThreeDDot(\n,\n,\n)}
6 \end{pspicture}
```

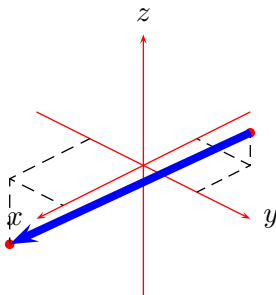
## 9 Lines

The syntax for a three dimensional line is just like the same from `\psline`

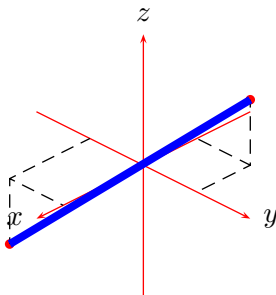
```
\pstThreeDLine [Options] [ $\langle$ arrow $\rangle$ ] (x1,y1,z1) (...) (xn,yn,zn)
```

The option and arrow part are both optional and the number of points is only limited to the memory. All options for lines from `pstricks` are possible, there are no special ones for a 3D line. There is no difference in drawing a line or a vector; the first one has an arrow of type `"'-"` and the second of `"'->"`.

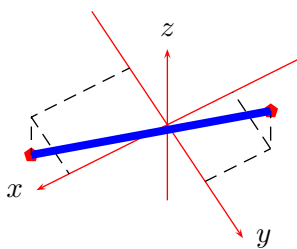
There is no special polygon macro, because you can get nearly the same with `\pstThreeDLine`.



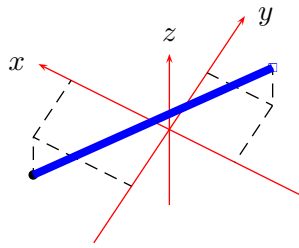
```
1 \begin{pspicture}(-2,-2.25)(2,2.25)
2   \pstThreeDCoord[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
3   \psset{dotstyle=*,linecolor=red,drawCoord=true}
4   \pstThreeDDot(-1,1,0.5)
5   \pstThreeDDot(1.5,-1,-1)
6   \pstThreeDLine[linewidth=3pt,linecolor=blue,arrows=->]%
7     (-1,1,0.5)(1.5,-1,-1)
8 \end{pspicture}
```



```
1 \begin{pspicture}(-2,-2.25)(2,2.25)
2   \pstThreeDCoord[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
3   \psset{dotstyle=*,linecolor=red,drawCoord=true}
4   \pstThreeDDot(-1,1,1)
5   \pstThreeDDot(1.5,-1,-1)
6   \pstThreeDLine[linewidth=3pt,linecolor=blue](-1,1,1)
7     (1.5,-1,-1)
8 \end{pspicture}
```



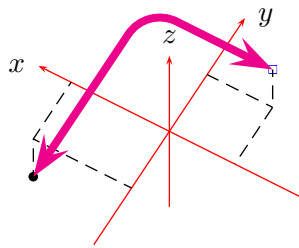
```
1 \begin{pspicture}(-2,-2.25)(2,2.25)
2   \psset{Alpha=30,Beta=60,dotstyle=pentagon*,dotsize=5pt,%
3     linecolor=red,drawCoord=true}
4   \pstThreeDCoord[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
5   \pstThreeDDot(-1,1,1)
6   \pstThreeDDot(1.5,-1,-1)
7   \pstThreeDLine[linewidth=3pt,linecolor=blue](-1,1,1)
8     (1.5,-1,-1)
9 \end{pspicture}
```



```

1 \begin{pspicture}(-2,-2.25)(2,2.25)
2   \psset{Alpha=30,Beta=-60}
3   \pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
4   \pstThreeDDot[dotstyle=square,linecolor=blue,drawCoor=true]
5     (-1,1,1)
6   \pstThreeDDot[drawCoor=true](1.5,-1,-1)
7   \pstThreeDLine[linewidth=3pt,linecolor=blue](-1,1,1)
8     (1.5,-1,-1)
9 \end{pspicture}

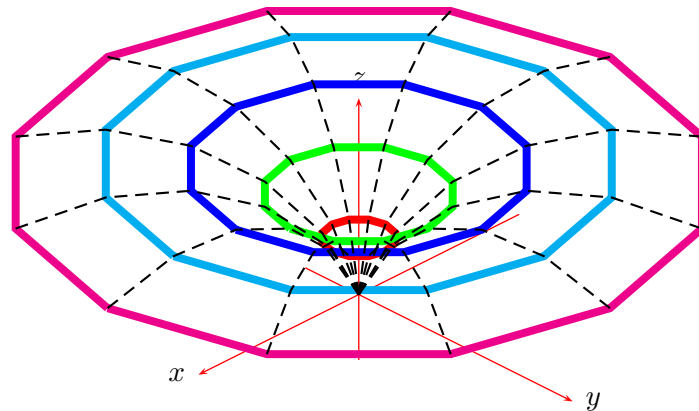
```



```

1 \begin{pspicture}(-2,-2.25)(2,2.25)
2   \psset{Alpha=30,Beta=-60}
3   \pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
4   \pstThreeDDot[dotstyle=square,linecolor=blue,drawCoor=true]
5     (-1,1,1)
6   \pstThreeDDot[drawCoor=true](1.5,-1,-1)
7   \pstThreeDLine[linewidth=3pt,arrowscale=1.5,%
8     linecolor=magenta,linearc=0.5]{<->}(-1,1,1)(1.5,2,-1)
9   (1.5,-1,-1)
10 \end{pspicture}

```



```

1 \begin{pspicture}(-3,-2)(4,5)\label{lines}
2   \pstThreeDCoor[xMin=-3,xMax=3,yMin=-1,yMax=4,zMin=-1,zMax=3]
3   \multido{\iA=1+1,\iB=60+-10}{5}{%
4     \ifcase\iA\or\psset{linecolor=red}\or\psset{linecolor=green}
5     \or\psset{linecolor=blue}\or\psset{linecolor=cyan}
6     \or\psset{linecolor=magenta}
7     \fi
8     \pstThreeDLine[SphericalCoor=true,linewidth=3pt]%
9     (\iA,0,\iB)(\iA,30,\iB)(\iA,60,\iB)(\iA,90,\iB)(\iA,120,\iB)(\iA,150,\iB)%
10    (\iA,180,\iB)(\iA,210,\iB)(\iA,240,\iB)(\iA,270,\iB)(\iA,300,\iB)%
11    (\iA,330,\iB)(\iA,360,\iB)%
12  }
13  \multido{\iA=0+30}{12}{%
14    \pstThreeDLine[SphericalCoor=true,linestyle=dashed]%
15    (0,0,0)(1,\iA,60)(2,\iA,50)(3,\iA,40)(4,\iA,30)(5,\iA,20)}
16 \end{pspicture}

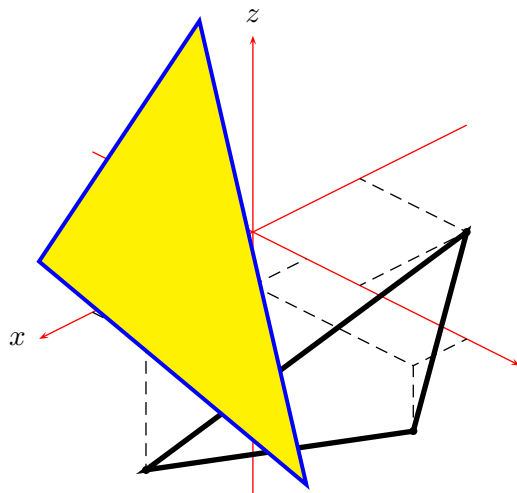
```

## 10 Triangles

A triangle is given with its three points:

```
\pstThreeDTriangle[<options>](P1)(P2)(P3)
```

When the option `fillstyle` is set to another value than `none` the triangle is filled with the active color or with the one which is set with the option `fillcolor`.

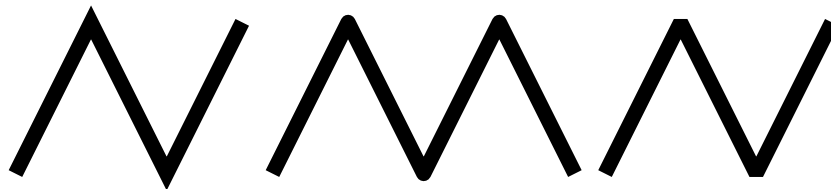


```

\begin{pspicture}(-3,-4.25)(3,3.25)
\pstThreeDCoor[xMin=-4,xMax=4,yMin=-3,yMax=5,zMin=-4,zMax=3]
\pstThreeDTriangle[drawCoor=true,linecolor=black,%
linewidth=2pt](3,1,-2)(1,4,-1)(-2,2,0)
\pstThreeDTriangle[fillcolor=yellow,fillstyle=solid,%
linecolor=blue,linewidth=1.5pt](5,1,2)(3,4,-1)
(-1,-2,2)
\end{pspicture}

```

Especially for triangles the option `linejoin` is important. The default value is 1, which gives rounded edges.

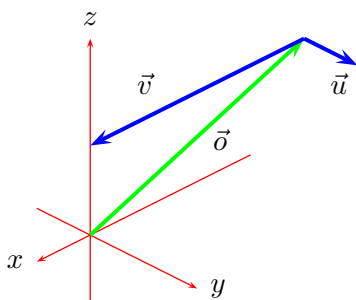


**Figure 4:** The meaning of the option `linejoin=0|1|2` for drawing lines

## 11 Squares

The syntax for a 3D square is:

```
\pstThreeDSquare [Options] (vector o) (vec u) (vec v)
```

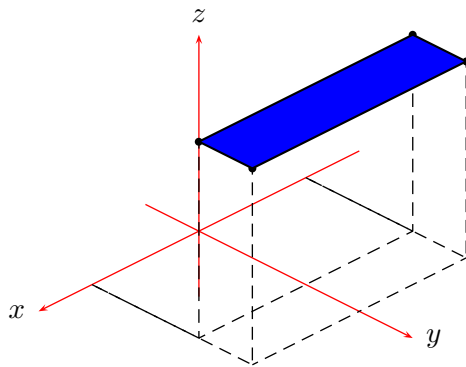


```

\begin{pspicture}(-1,-1)(4,3)
\pstThreeDCoor[xMin=-3,xMax=1,yMin=-1,yMax=2,zMin=-1,zMax=3]
\psset{arrows=->,arrowsize=0.2,linecolor=blue,linewidth=1.5pt}
\pstThreeDLine[linecolor=green](0,0,0)(-2,2,3)\uput[45](1.5,1){$\vec{o}$}
\pstThreeDLine(-2,2,3)(2,2,3)\uput[0](3,2){$\vec{u}$}
\pstThreeDLine(-2,2,3)(-2,3,3)\uput[180](1,2){$\vec{v}$}
\end{pspicture}

```

Squares are nothing else than a polygon with the starting point  $P_o$  given with the origin vector  $\vec{o}$  and the two direction vectors  $\vec{u}$  and  $\vec{v}$ , which build the sides of the square.



```

1 \begin{pspicture}(-3,-2)(4,3)
2   \pstThreeDCoor[xMin=-3,xMax=3,yMin=-1,yMax=4,
3     zMin=-1,zMax=3]
4   {\psset{fillcolor=blue,fillstyle=solid,
5     drawCoor=true,dotstyle=*}
6     \pstThreeDSquare(-2,2,3)(4,0,0)(0,1,0)}
7 \end{pspicture}

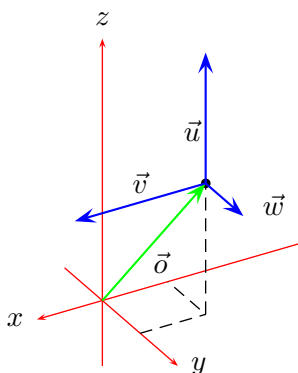
```

## 12 Boxes

A box is a special case of a square and has the syntax

`\pstThreeDBox [Options] (vector o) (vec u) (vec v) (vec w)`

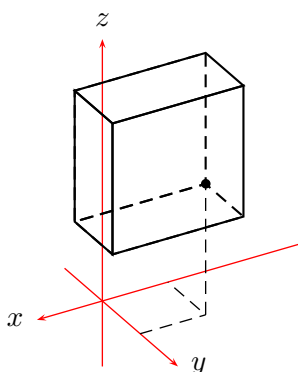
These are the origin vector  $\vec{o}$  and three direction vectors  $\vec{u}$ ,  $\vec{v}$  and  $\vec{w}$ , which are for example shown in the following figure.



```

1 \begin{pspicture}(-2,-1.25)(3,4.25)
2   \psset{Alpha=30,Beta=30}
3   \pstThreeDCoor[xMin=-3,xMax=1,yMin=-1,yMax=2,zMin=-1,zMax=4]
4   \pstThreeDDot[drawCoor=true](-1,1,2)
5   \psset{arrows=->,arrowsize=0.2}
6   \pstThreeDLine[linecolor=green](0,0,0)(-1,1,2)
7   \uput[0](0.5,0.5){$\vec{o}$}
8   \uput[0](0.9,2.25){$\vec{u}$}
9   \uput[90](0.5,1.25){$\vec{v}$}
10  \uput[45](2,1){$\vec{w}$}
11  \pstThreeDLine[linecolor=blue](-1,1,2)(-1,1,4)
12  \pstThreeDLine[linecolor=blue](-1,1,2)(1,1,2)
13  \pstThreeDLine[linecolor=blue](-1,1,2)(-1,2,2)
14 \end{pspicture}

```

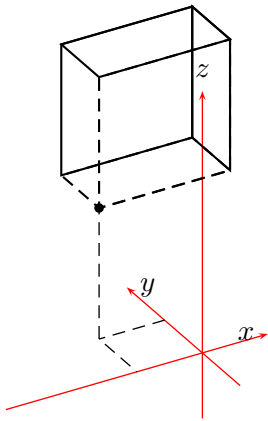


```

1 \begin{pspicture}(-2,-1.25)(3,4.25)
2   \psset{Alpha=30,Beta=30}
3   \pstThreeDCoor[xMin=-3,xMax=1,yMin=-1,yMax=2,zMin=-1,zMax=4]
4   \pstThreeDBox[hiddenLine](-1,1,2)(0,0,2)(2,0,0)(0,1,0)
5   \pstThreeDDot[drawCoor=true](-1,1,2)
6 \end{pspicture}

```

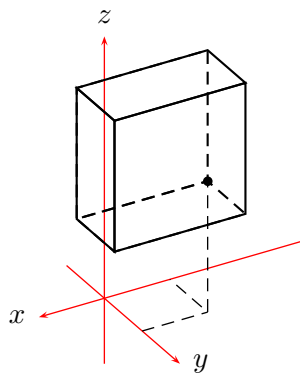




```

\begin{pspicture}(-2,-1.25)(3,4.25)
\psset{Alpha=210,Beta=30}
\pstThreeDCoor[xMin=-3,xMax=1,yMin=-1,yMax=2,zMin=-1,zMax=4]
\pstThreeDBox[hiddenLine](-1,1,2)(0,0,2)(2,0,0)(0,1,0)
\pstThreeDDot[drawCoor=true](-1,1,2)
\end{pspicture}

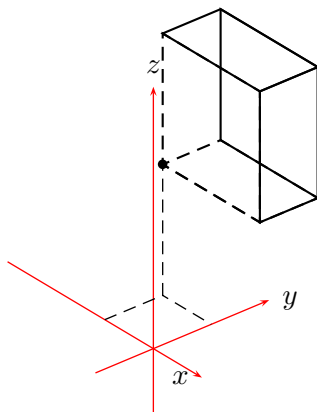
```



```

\begin{pspicture}(-2,-1.25)(3,4.25)
\psset{Alpha=30,Beta=30}
\pstThreeDCoor[xMin=-3,xMax=1,yMin=-1,yMax=2,zMin=-1,zMax=4]
\pstThreeDBox[hiddenLine](-1,1,2)(0,0,2)(2,0,0)(0,1,0)
\pstThreeDDot[drawCoor=true](-1,1,2)
\end{pspicture}

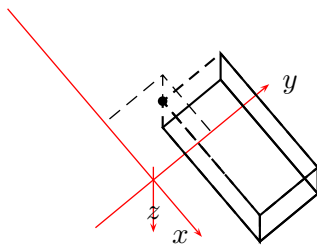
```



```

\begin{pspicture}(-2,-1.25)(3,4.25)
\psset{Alpha=130,Beta=30}
\pstThreeDCoor[xMin=-3,xMax=1,yMin=-1,yMax=2,zMin=-1,zMax=4]
\pstThreeDBox[hiddenLine](-1,1,2)(0,0,2)(2,0,0)(0,1,0)
\pstThreeDDot[drawCoor=true](-1,1,2)
\end{pspicture}

```



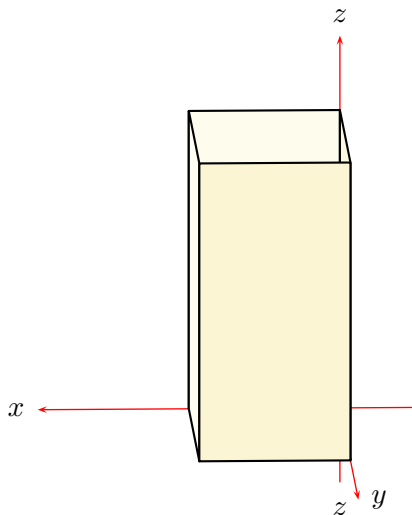
```

1 \begin{pspicture}(-2,-1.25)(3,4.25)
2   \psset{Alpha=130,Beta=100}
3   \pstThreeDCoor[xMin=-3,xMax=1,yMin=-1,yMax=2,zMin=-1,zMax=4]
4   \pstThreeDBox[hiddenLine](-1,1,2)(0,0,2)(2,0,0)(0,1,0)
5   \pstThreeDDot[drawCoor=true](-1,1,2)
6 \end{pspicture}

```

`\psBox [Options] (vector o){width}{depth}{height}`

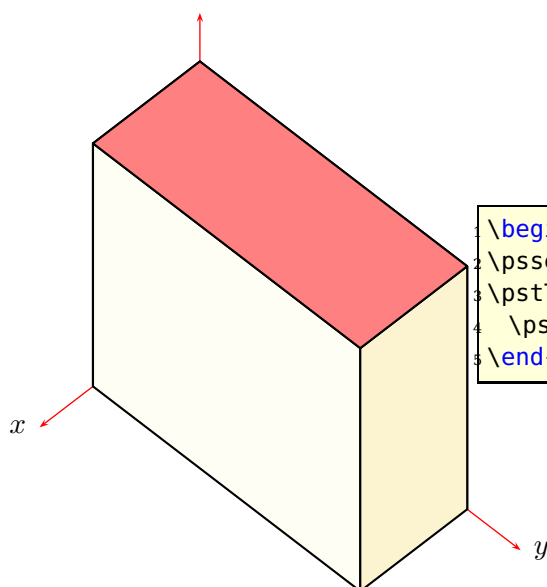
The origin vector  $\vec{o}$  determines the left corner of the box.



```

1 \begin{pspicture}(-3,-2)(3,5)
2   \psset{Alpha=2,Beta=10}
3   \pstThreeDCoor[zMax=5,yMax=7]
4   \psBox(0,0,0){2}{4}{3}
5 \end{pspicture}

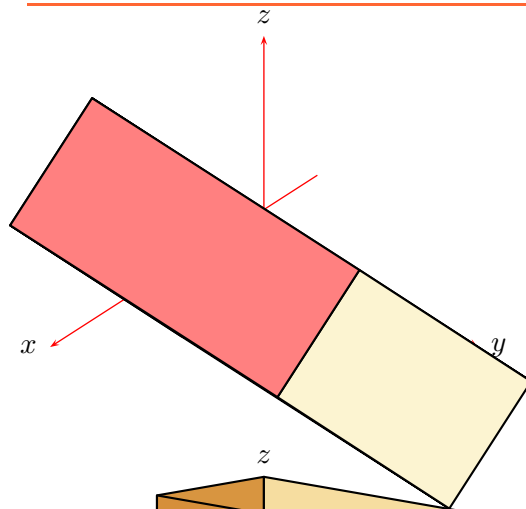
```



```

1 \begin{pspicture}(-3,-3)(3,3)
2   \psset{Beta=50}
3   \pstThreeDCoor[xMax=3,zMax=6,yMax=6]
4   \psBox[showInside=false](0,0,0){2}{5}{3}
5 \end{pspicture}

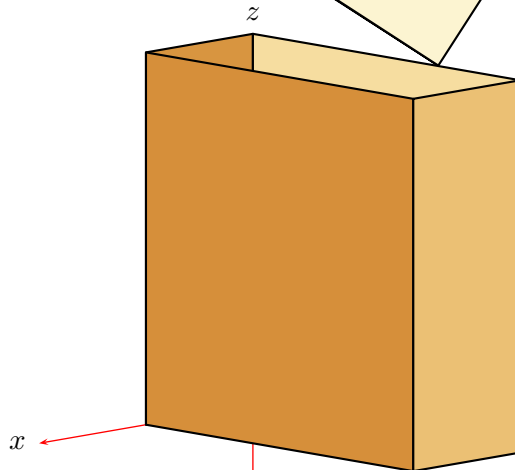
```



```

1 \begin{pspicture}(-3,-4)(3,2)
2 \psset{Beta=40}
3 \pstThreeDCoor[zMax=3]
4   \psBox[RotY=20,showInside=false](0,0,0){2}{5}{3}
5 \end{pspicture}

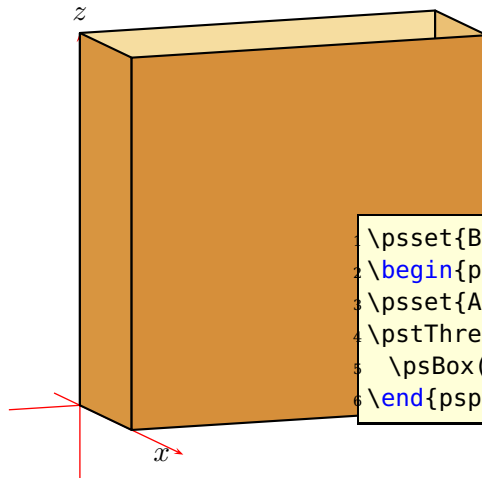
```



```

1 \psset{Beta=10,xyzLight=-7 3 4}
2 \begin{pspicture}(-3,-2)(3,4)
3 \pstThreeDCoor[zMax=5]
4   \psBox(0,0,0){2}{5}{3}
5 \end{pspicture}

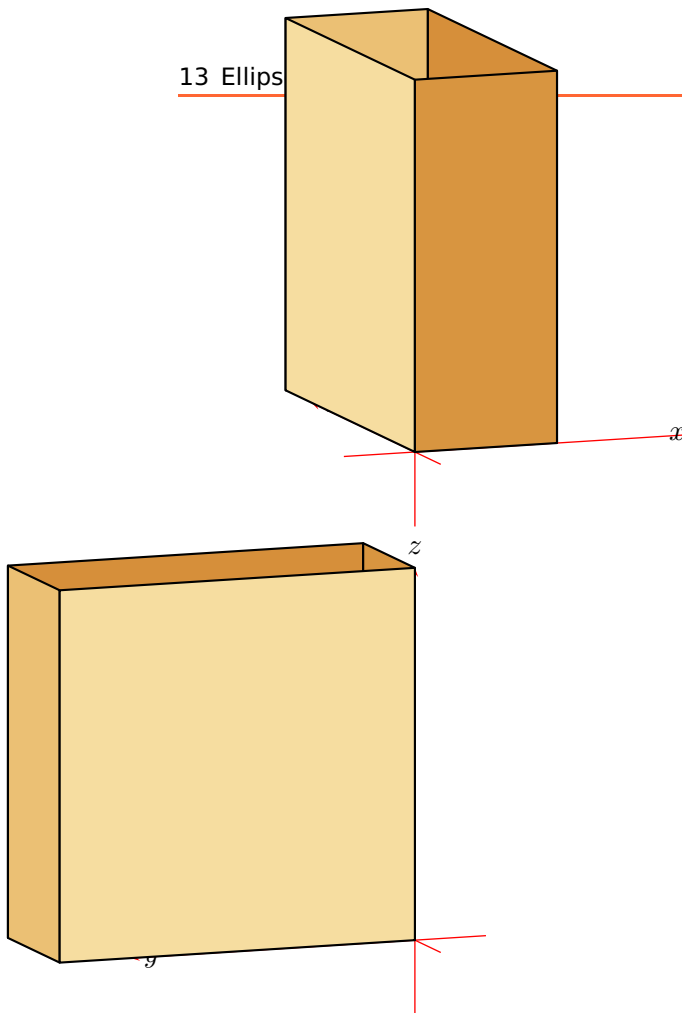
```



```

1 \psset{Beta=10,xyzLight=-7 3 4}
2 \begin{pspicture}(-3,-2)(3,4)
3 \psset{Alpha=110}
4 \pstThreeDCoor[zMax=5]
5   \psBox(0,0,0){2}{5}{3}
6 \end{pspicture}

```



```

1 \psset{Beta=10,xyzLight=-7 3 4}
2 \begin{pspicture}(-3,-2)(3,4)
3 \psset{Alpha=200}
4 \pstThreeDCoor[zMax=5]
5   \psBox(0,0,0){2}{5}{3}
6 \end{pspicture}

```

```

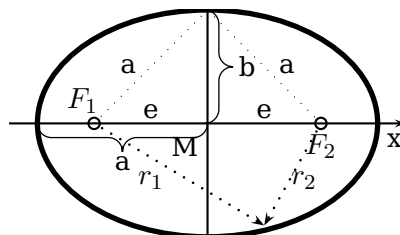
1 \psset{Beta=10,xyzLight=-7 3 4}
2 \begin{pspicture}(-3,-2)(3,4)
3 \psset{Alpha=290}
4 \pstThreeDCoor[zMax=5]
5   \psBox(0,0,0){2}{5}{3}
6 \end{pspicture}

```

## 13 Ellipses and circles

The equation for a two dimensional ellipse (figure 5) is:

$$e : \frac{(x - x_M)^2}{a^2} + \frac{(y - y_M)^2}{b^2} = 1 \quad (4)$$



**Figure 5:** Definition of an Ellipse

$(x_m; y_m)$  is the center,  $a$  and  $b$  the semi major and semi minor axes respectively and  $e$  the excentricity. For  $a = b = 1$  in equation 4 we get the one for the circle, which is nothing else than a special ellipse. The equation written in the parameterform is

$$\begin{aligned}x &= a \cdot \cos \alpha \\y &= b \cdot \sin \alpha\end{aligned}\tag{5}$$

or the same with vectors to get an ellipse in a 3D system:

$$e : \vec{x} = \vec{m} + \cos \alpha \cdot \vec{u} + \sin \alpha \cdot \vec{v} \quad 0 \leq \alpha \leq 360 \tag{6}$$

where  $\vec{m}$  is the center,  $\vec{u}$  and  $\vec{v}$  the directions vectors which are perpendicular to each other.

### 13.1 Options

In addition to all possible options from `pst-plot` there are two special options to allow drawing of an arc (with predefined values for a full ellipse/circle):

```
beginAngle=0
endAngle=360
```

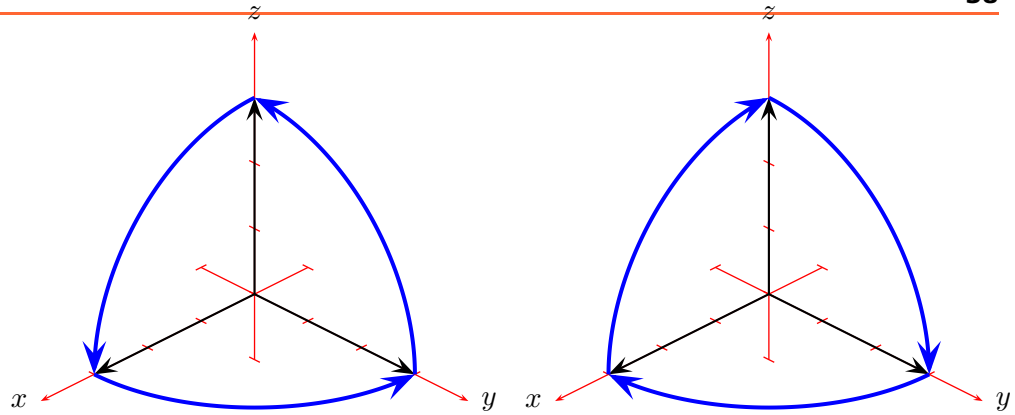
Ellipses and circles are drawn with the in section 18.2 described `parametricplotThreeD` macro with a default setting of 50 points for a full ellipse/circle.

### 13.2 Ellipse

It is very difficult to see in a 3D coordinate system the difference of an ellipse and a circle. Depending to the view point an ellipse maybe seen as a circle and vice versa. The syntax of the ellipse macro is:

```
\pstThreeDEllipse [Options] (cx,cy,cz) (ux,uy,uz) (vx,vy,vz)
```

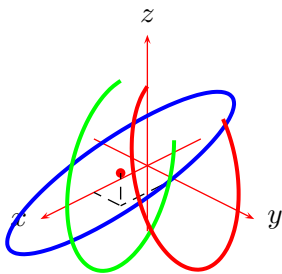
where *c* is for center and *u* and *v* for the two direction vectors. The order of these two vectors is important for the drawing if it is a left or right turn. It follows the right hand rule: flap the first vector  $\vec{u}$  on the shortest way into the second one  $\vec{v}$ , then you'll get the positive rotating.



```

1 \begin{pspicture}(-3,-2)(3,3)
2   \pstThreeDCoor[IIIDticks]
3   \psset{arrowscale=2,arrows=->}
4   \pstThreeDLine(0,0,0)(3,0,0)\pstThreeDLine(0,0,0)(0,3,0)\pstThreeDLine(0,0,0)(0,0,3)
5   \psset{linecolor=blue,linewidth=1.5pt,beginAngle=0,endAngle=90}
6   \pstThreeDEllipse(0,0,0)(3,0,0)(0,3,0) \pstThreeDEllipse(0,0,0)(0,0,3)(3,0,0)
7   \pstThreeDEllipse(0,0,0)(0,3,0)(0,0,3)
8 \end{pspicture}\hspace{2em}
9 \begin{pspicture}(-3,-2)(3,3)
10  \pstThreeDCoor[IIIDticks]
11  \psset{arrowscale=2,arrows=->}
12  \pstThreeDLine(0,0,0)(3,0,0)\pstThreeDLine(0,0,0)(0,3,0)\pstThreeDLine(0,0,0)(0,0,3)
13  \psset{linecolor=blue,linewidth=1.5pt,beginAngle=0,endAngle=90}
14  \pstThreeDEllipse(0,0,0)(0,3,0)(3,0,0) \pstThreeDEllipse(0,0,0)(3,0,0)(0,0,3)
15  \pstThreeDEllipse(0,0,0)(0,0,3)(0,3,0)
16 \end{pspicture}

```



```

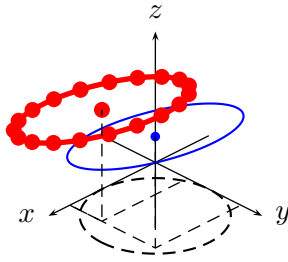
1 \begin{pspicture}(-2,-2.25)(2,2.25)
2   \pstThreeDCoor[xMax=2,yMax=2,zMax=2]
3   \pstThreeDDot[linecolor=red,drawCoor=true](1,0.5,0.5)
4   \psset{linecolor=blue,linewidth=1.5pt}
5   \pstThreeDEllipse(1,0.5,0.5)(-0.5,1,0.5)(1,-0.5,-1)
6   \psset{beginAngle=0,endAngle=270,linecolor=green}
7   \pstThreeDEllipse(1,0.5,0.5)(-0.5,0.5,0.5)(0.5,0.5,-1)
8   \pstThreeDEllipse[RotZ=45,linecolor=red](1,0.5,0.5)
9   (-0.5,0.5,0.5)(0.5,0.5,-1)
10 \end{pspicture}

```

### 13.3 Circle

The circle is a special case of an ellipse (equ. 6) with the vectors  $\vec{u}$  and  $\vec{v}$  which build the circle plain. They must not be orthogonal to each other. The circle macro takes the length of vector  $\vec{u}$  into account for the radius. The orthogonal part of vector  $\vec{v}$  is calculated internally

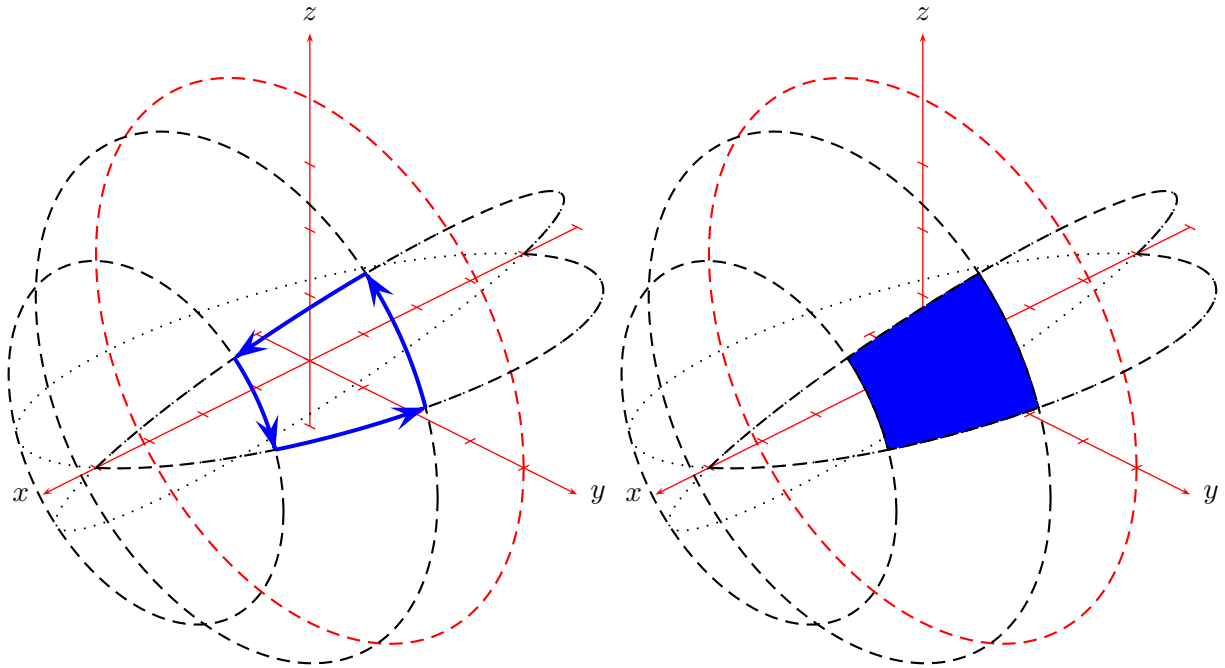
```
\pstThreeDCircle [Options] (cx,cy,cz) (ux,uy,uz) (vx,vy,vz)
```



```

\begin{pspicture}(-2,-1.25)(2,2.25)
  \pstThreeDCoor[xMax=2,yMax=2,zMax=2,linecolor=black]
  \pstThreeDCircle[linestyle=dashed](1,1,0)(1,0,0)(3,4,0)
  \pstThreeDCircle[linecolor=blue](1.6,1.6,1.7)(0.8,0.4,0.8)
    (0.8,-0.8,-0.4)
  \pstThreeDDot[drawCoor=true,linecolor=blue](1.6,1.6,1.7)
  \psset{linecolor=red,linewidth=2pt,plotpoints=20,showpoints=
    true}
  \pstThreeDCircle(1.6,0.6,1.7)(0.8,0.4,0.8)(0.8,-0.8,-0.4)
  \pstThreeDDot[drawCoor=true,linecolor=red](1.6,0.6,1.7)
\end{pspicture}

```



```

1 \def\radius{4 }\def\PhiI{20 }\def\PhiII{50 }
2 %
3 \def\RadIs{\radius \PhiI sin mul}
4 \def\RadIc{\radius \PhiI cos mul}
5 \def\RadIIs{\radius \PhiII sin mul}
6 \def\RadIic{\radius \PhiII cos mul}
7 \begin{pspicture}(-4,-4)(4,5)
8   \psset{Alpha=45,Beta=30,linestyle=dashed}
9   \pstThreeDCoor[linestyle=solid,xMin=-5,xMax=5,yMax=5,zMax=5,IIIDticks]
10  \pstThreeDEllipse[linecolor=red](0,0,0)(0,\radius,0)(0,0,\radius)
11  \pstThreeDEllipse(\RadIs,0,0)(0,\RadIc,0)(0,0,\RadIc)
12  \pstThreeDEllipse(\RadIIs,0,0)(0,\RadIic,0)(0,0,\RadIic)
13 %
14  \pstThreeDEllipse[linestyle=dotted,SphericalCoor](0,0,0)(\radius,90,\PhiI)
15    (\radius,0,0)
16  \pstThreeDEllipse[SphericalCoor,
17    beginAngle=-90,endAngle=90](0,0,0)(\radius,90,\PhiI)(\radius,0,0)
18  \pstThreeDEllipse[linestyle=dotted,SphericalCoor](0,0,0)(\radius,90,\PhiII)
19    (\radius,0,0)

```

```

18 \pstThreeDEllipse[SphericalCoor,
19   beginAngle=-90,endAngle=90](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
20 %
21 \psset{linecolor=blue,arrows=->,arrowscale=2,linewidth=1.5pt,linestyle=
22   solid}
23 \pstThreeDEllipse[SphericalCoor,beginAngle=\PhiI,endAngle=\PhiII]%
24   (0,0,0)(\radius,90,\PhiII)(\radius,0,0)
25 \pstThreeDEllipse[beginAngle=\PhiII,endAngle=\PhiI](\RadIIIs,0,0)(0,\RadIIc
26   ,0)(0,0,\RadIIc)
27 \pstThreeDEllipse[SphericalCoor,beginAngle=\PhiII,endAngle=\PhiI]%
28   (0,0,0)(\radius,90,\PhiI)(\radius,0,0)
29 \pstThreeDEllipse[beginAngle=\PhiI,endAngle=\PhiII](\RadIs,0,0)(0,\RadIc
30   ,0)(0,0,\RadIc)
31 \end{pspicture}
32 \begin{pspicture}(-4,-4)(4,5)
33 [ ... ]
34 \pstThreeDEllipse[linestyle=dotted,SphericalCoor](0,0,0)(\radius,90,\PhiI)
35   (\radius,0,0)
36 \pstThreeDEllipse[SphericalCoor,
37   beginAngle=-90,endAngle=90](0,0,0)(\radius,90,\PhiI)(\radius,0,0)
38 \pstThreeDEllipse[linestyle=dotted,SphericalCoor](0,0,0)(\radius,90,\PhiII)
39   (\radius,0,0)
40 \pstThreeDEllipse[SphericalCoor,
41   beginAngle=-90,endAngle=90](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
42 %
43 \pscustom[fillstyle=solid,fillcolor=blue]{
44   \pstThreeDEllipse[SphericalCoor,beginAngle=\PhiI,endAngle=\PhiII]%
45   (0,0,0)(\radius,90,\PhiII)(\radius,0,0)
46   \pstThreeDEllipse[beginAngle=\PhiII,endAngle=\PhiI](\RadIIIs,0,0)(0,\RadIIc
47     ,0)(0,0,\RadIIc)
48   \pstThreeDEllipse[SphericalCoor,beginAngle=\PhiII,endAngle=\PhiI]%
49   (0,0,0)(\radius,90,\PhiI)(\radius,0,0)
50   \pstThreeDEllipse[beginAngle=\PhiI,endAngle=\PhiII](\RadIs,0,0)(0,\RadIc
51     ,0)(0,0,\RadIc)
52 }
53 \end{pspicture}

```

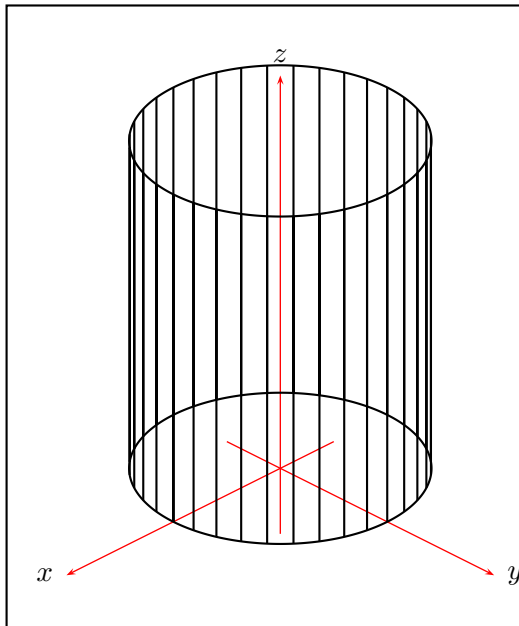
## 14 \pstIIIDCylinder

The syntax is

<code>\pstIIIDCylinder [Options] (x,y,z){radius}{height}</code>
---

$(x,y,z)$  defines the center of the lower part of the cylinder. If it is missing, then  $(0,0,0)$  are taken into account.

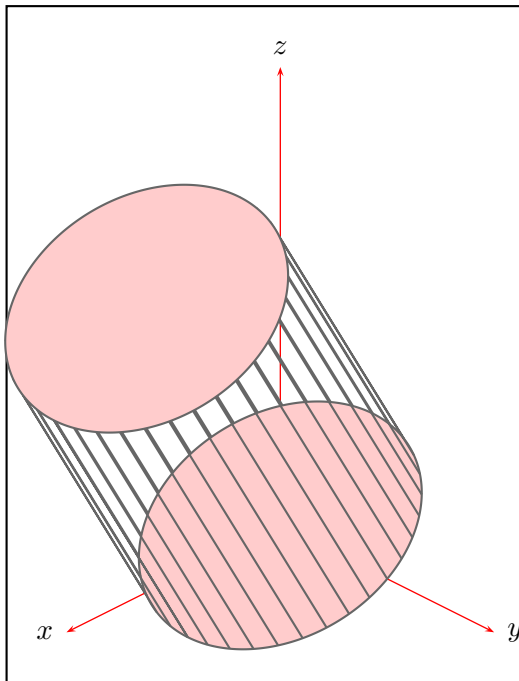




```

1 \psframebox{%
2 \begin{pspicture}(-3.5,-2)(3,6)
3 \pstThreeDCoor[zMax=6]
4 \pstIIIDCylinder{2}{5}
5 \end{pspicture}
6 }

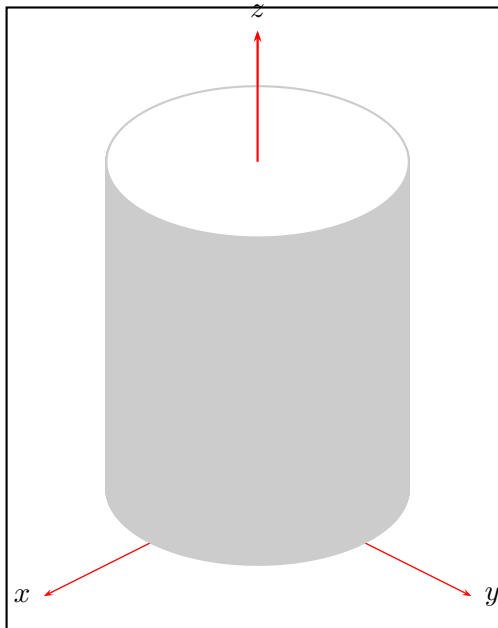
```



```

1 \psframebox{%
2 \begin{pspicture}(-3.5,-2)(3,6.75)
3 \pstThreeDCoor[zMax=7]
4 \pstIIIDCylinder[RotY=30,fillstyle=solid,
5   fillcolor=red!20,linecolor=black!60](0,0,0){2}{5}
6 \end{pspicture}
7 }

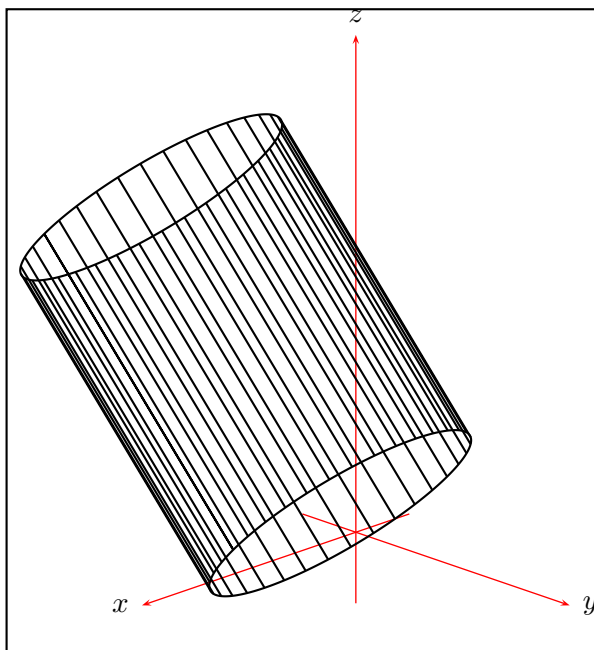
```



```

1 \psframebox{%
2 \begin{pspicture}(-3.2,-1.75)(3,6.25)
3 \pstThreeDCoor[zMax=7]
4 \pstIIIDCylinder[linecolor=black!20,
5   increment=0.4,fillstyle=solid]{2}{5}
6 \psset{linecolor=red}
7 \pstThreeDLine{->}(0,0,5)(0,0,7)
8 \end{pspicture}
9 }

```



```

1 \psframebox{%
2 \begin{pspicture}(-4.5,-1.5)(3,6.8)
3 \psset{Beta=20}
4 \pstThreeDCoor[zMax=7]
5 \pstIIIDCylinder[fillcolor=blue!20,
6   RotX=45](1,1,0){2}{5}
7 \end{pspicture}
8 }

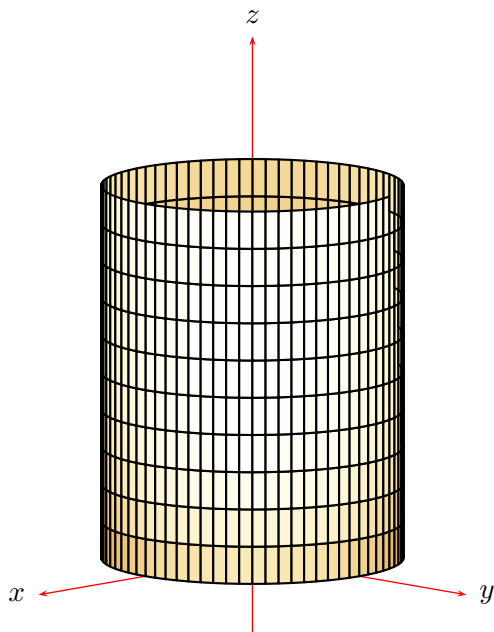
```

## 15 \psCylinder

The syntax is

```
\psCylinder [Options] (x,y,z){radius}{height}
```

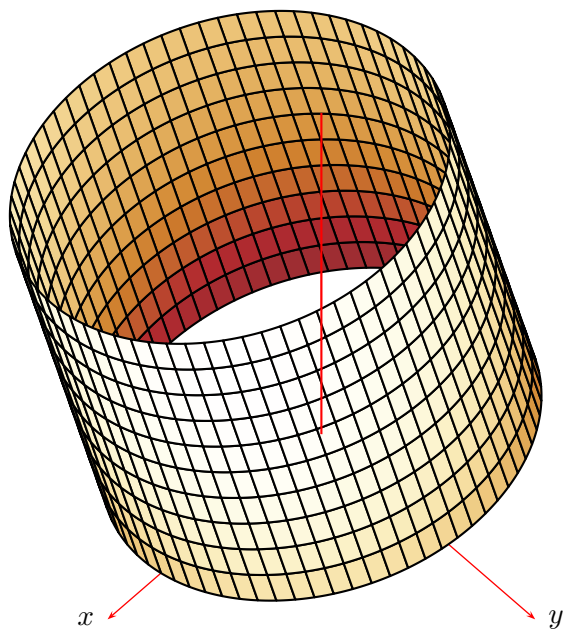
$(x,y,z)$  defines the center of the lower part of the cylinder. If it is missing, then  $(0,0,0)$  are taken into account.



```

1 \begin{pspicture}(-3,-2)(3,7)
2 \psset{Beta=10}
3 \pstThreeDCoor[zMax=7]
4   \psCylinder[increment=5]{2}{5}
5 \end{pspicture}

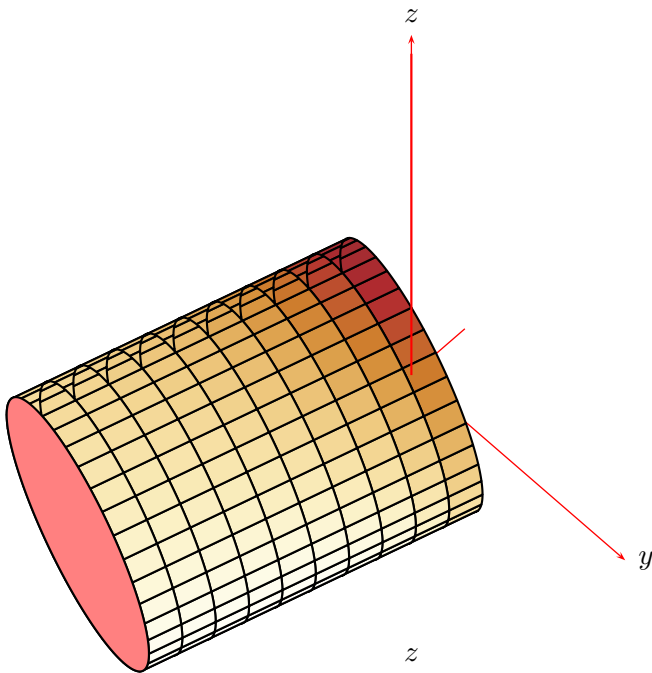
```



```

1 \begin{pspicture}(-3,-2)(3,6)
2 \psset{Beta=60}
3 \pstThreeDCoor[zMax=9]
4   \psCylinder[RotX=10,increment=5]{3}{5}
5   \pstThreeDLine[linecolor=red](0,0,0)(0,0,8.5)
6 \end{pspicture}

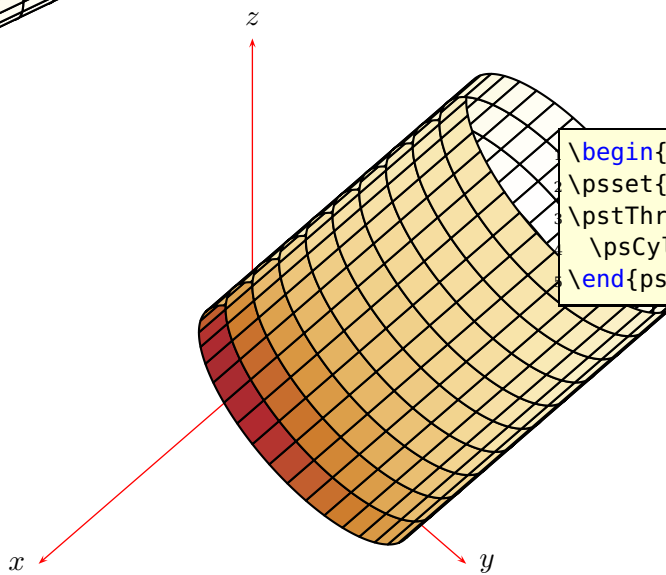
```



```

1 \begin{pspicture}(-3,-2)(3,6)
2 \psset{Beta=60}
3 \pstThreeDCoor[zMax=9]
4 \psCylinder[RotX=10,RotY=45,showInside=false]
5   {2}{5}
6 \pstThreeDLine[linecolor=red](0,0,0)(0,0,8.5)
7 \end{pspicture}

```



```

1 \begin{pspicture}(-3,-2)(3,6)
2 \psset{Beta=60}
3 \pstThreeDCoor[zMax=9]
4 \psCylinder[RotY=-45](0,1,0){2}{5}
5 \end{pspicture}

```

# 16 \pstParaboloid

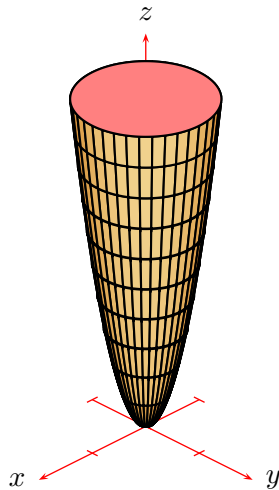
The syntax is

`\pstParaboloid` `[Options]` `{height}``{radius}`

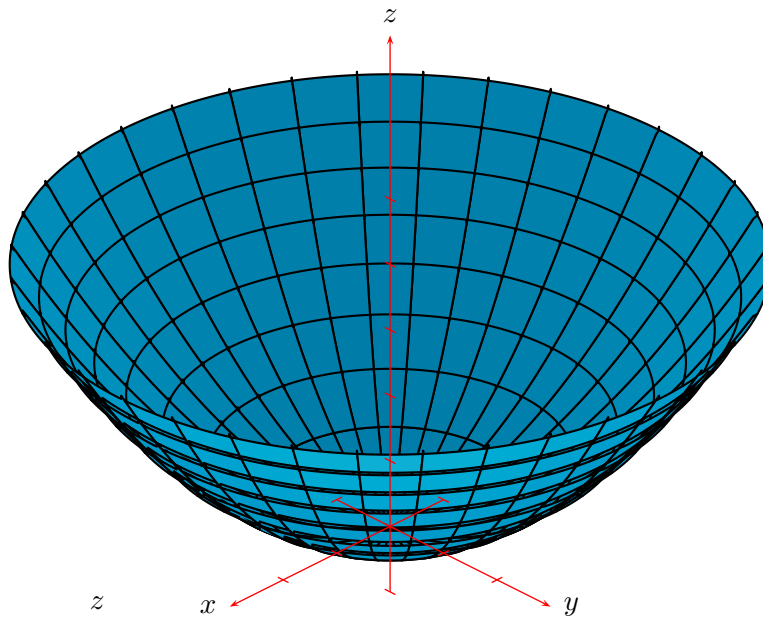
height and radius depend to each other, it is the radius of the circle at the height. By default the paraboloid is placed in the origin of coordinate system, but with \pstThreeDput it can be placed anywhere. The possible options are listed in table 2. The segment color must be set as a cmyk color SegmentColor={ [cmyk]{c,m,y,k}} in parenthesis, otherwise xcolor cannot read the values. A white color is given by SegmentColor={ [cmyk]{0,0,0,0}}.

Table 2: Options for the \pstParaboloid macro

Option name	value
SegmentColor	cmyk color for the segments (0.2,0.6,1,0)
showInside	show inside (true)
increment	number for the segments (10)



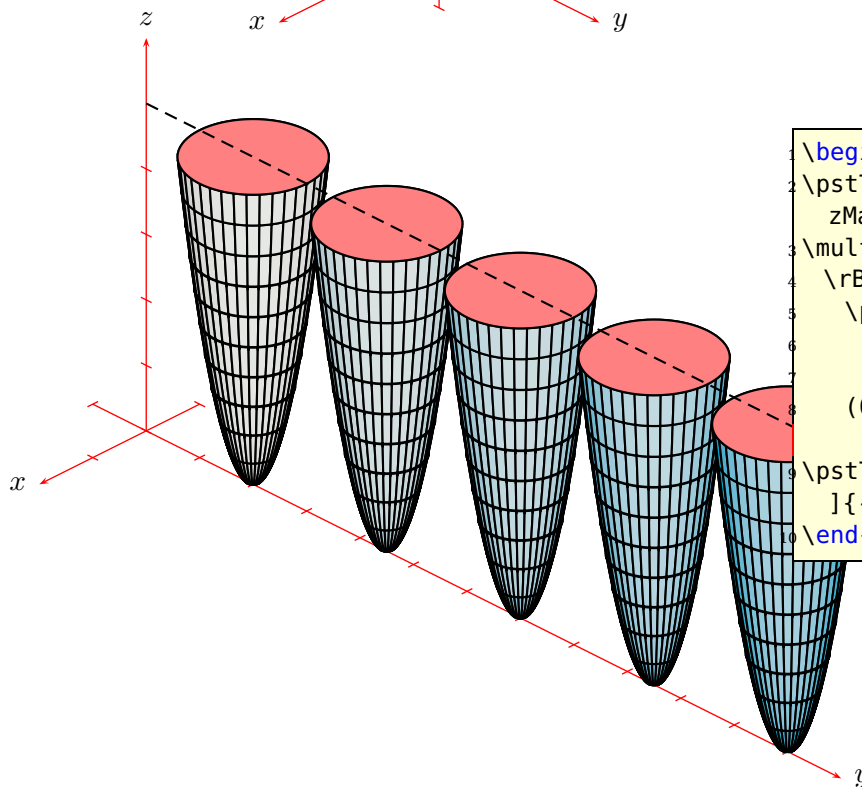
```
1 \begin{pspicture}(-2,-1)(2,5)
2 \pstThreeDCoor[xMax=2,yMax=2,zMin=0,zMax=6,IIIDticks]%
3 \pstParaboloid{5}{1}% Hoehe 5 und Radius 1
4 \end{pspicture}
```



```

\begin{pspicture}(-.25\linewidth
, -1)%
(.25\linewidth,7.5)
\pstParaboloid[showInside=false,
SegmentColor={ [cmyk
]{0.8,0.1,.11,0}}]{4}{5}%
\pstThreeDCoor[xMax=3,yMax=3,
zMax=7.5,IIIDticks]
\end{pspicture}

```

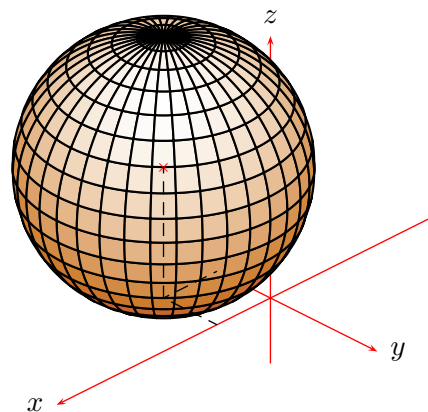


```

\begin{pspicture}(0,-3)(7,5)
\pstThreeDCoor[xMax=2,yMax=13,zMin=0,
zMax=6,IIIDticks]%
\multido{\rA=2.0+2.5,
\rB=0.15+0.20}{5}{%
\pstParaboloid[%
SegmentColor={ [cmyk]%
{\rB,0.1,0.11,0.1}}}%
(0,\rA,0){5}{1}}% height 5 and
radius 1
\pstThreeDLine[linestyle=dashed
]{->}(0,0,5)(0,13,5)
\end{pspicture}

```

# 17 Spheres



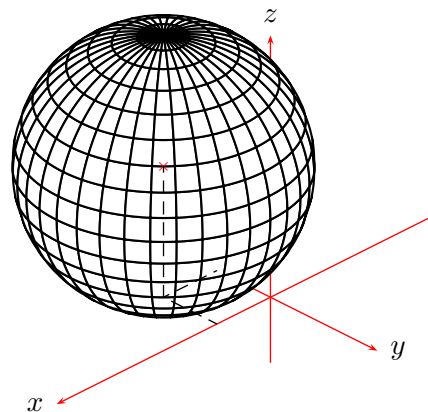
```
\begin{pspicture}(-4,-2.25)(2,4.25)
\pstThreeDCoor[xMin=-3,yMax=2]
\pstThreeDSphere(1,-1,2){2}
\pstThreeDDot[dotstyle=x,linecolor=red,drawCoor=true]
(1,-1,2)
\end{pspicture}
```

`\pstThreeDSphere` [Options] (x,y,z){Radius}

(x,y,z) is the center of the sphere and possible options are listed in table 3. The segment color must be set as a cmyk color `SegmentColor={ [cmyk] {c,m,y,k} }` in parenthesis, otherwise xcolor cannot read the values. A white color is given by `SegmentColor={ [cmyk] {0,0,0,0} }`

Table 3: Options for the sphere macro

Option name	value
SegmentColor	cmyk color for the segments (0.2,0.6,1,0)
increment	number for the segments (10)



```
\begin{pspicture}(-4,-2.25)(2,4.25)
\pstThreeDCoor[xMin=-3,yMax=2]
\pstThreeDSphere[SegmentColor={ [cmyk]
{0,0,0,0} }](1,-1,2){2}
\pstThreeDDot[dotstyle=x,linecolor=red,drawCoor=true]
(1,-1,2)
\end{pspicture}
```

## 18 Mathematical functions

There are two macros for plotting mathematical functions, which work similar to the one from `pst-plot`.

### 18.1 Function $f(x, y)$

The macro for plotting functions does not have the same syntax as the one from `pst-plot`[\[5\]](#), but it is used in the same way:

```
\psplotThreeD [Options] (xMin,xMax) (yMin,yMax) {the function}
```

The function has to be written in PostScript code and the only valid variable names are `x` and `y`, f.ex: `{x dup mul y dup mul add sqrt}` for the math expression  $\sqrt{x^2 + y^2}$ . The macro has the same plotstyle options as `psplot`, except the `plotpoints`-option which is split into one for `x` and one for `y` (table [4](#)).

**Table 4:** Options for the plot Macros

Option name	value
plotstyle	dots
	line
	polygon
	curve
	ecurve
	ccurve
	none (default)
showpoints	default is false
xPlotpoints	default is 25
yPlotpoints	default is 25
drawStyle	default is xLines
	yLines
	xyLines
	yxLines
hiddenLine	default is false
algebraic	default is false

The equation [7](#) is plotted with the following parameters and seen in figure [6](#).

$$z = 10 \left( x^3 + xy^4 - \frac{x}{5} \right) e^{-(x^2+y^2)} + e^{-((x-1.225)^2+y^2)} \quad (7)$$

The function is calculated within two loops:

```
for (float y=yMin; y<yMax; y+=dy)
  for (float x=xMin; x<xMax; x+=dx)
    z=f(x,y);
```



It depends to the inner loop in which direction the curves are drawn. There are four possible values for the option `drawStyle` :

- `xLines` (default) Curves are drawn in x direction
- `yLines` Curves are drawn in y direction
- `xyLines` Curves are first drawn in x and then in y direction
- `yxLines` Curves are first drawn in y and then in x direction

In fact of the inner loop it is only possible to get a closed curve in the defined direction. For lines in x direction less `yPlotpoints` are no problem, in difference to `xPlotpoints`, especially for the `plotstyle` options `line` and `dots`.

Drawing three dimensional functions with curves which are transparent makes it difficult to see if a point is before or behind another one. `\psplotThreeD` has an option `hiddenLine` for a primitive hidden line mode, which only works when the y-intervall is defined in a way that  $y_2 > y_1$ . Then every new curve is plotted over the forgoing one and filled with the color white. Figure 7 is the same as figure 6, only with the option `hiddenLine=true`.

```

1 \begin{pspicture}(-6,-4)(6,5)
2   \psset{Beta=15}
3   \psplotThreeD[plotstyle=line,drawStyle=xLines,% is the default anyway
4     yPlotpoints=50,xPlotpoints=50,linewidth=1pt](-4,4)(-4,4){%
5     x 3 exp x y 4 exp mul add x 5 div sub 10 mul
6     2.729 x dup mul y dup mul add neg exp mul
7     2.729 x 1.225 sub dup mul y dup mul add neg exp add}
8   \pstThreeDCool[xMin=-1,xMax=5,yMin=-1,yMax=5,zMin=-1,zMax=5]
9 \end{pspicture}

```

## 18.2 Parametric Plots

Parametric plots are only possible for drawing curves or areas. The syntax for this plot macro is:

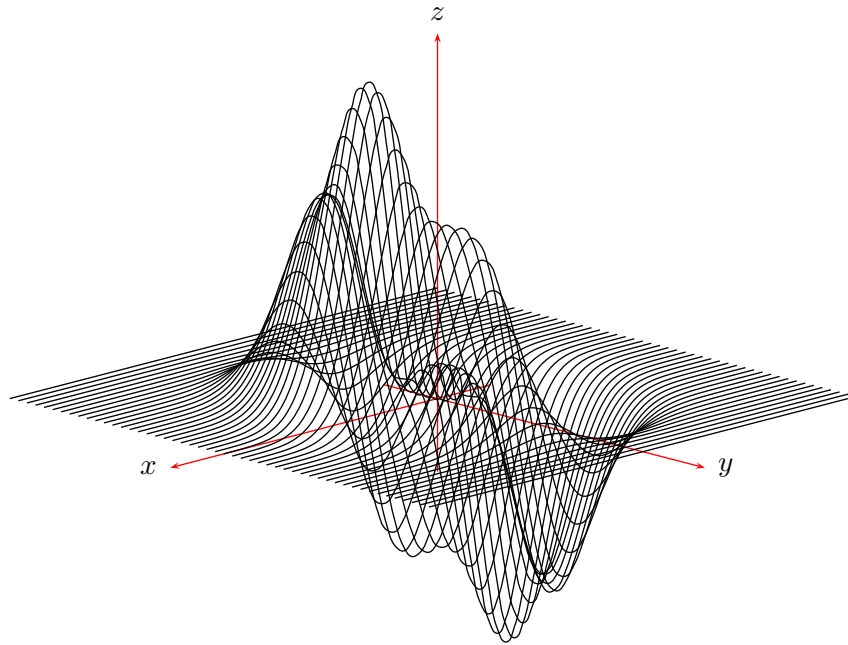
```
\parametricplotThreeD [Options] (t1,t2)(u1,u2){three parametric functions x y z}
```

The only possible variables are `t` and `u` with `t1,t2` and `u1,u2` as the range for the parameters. The order for the functions is not important and `u` may be optional when having only a three dimensional curve and not an area.

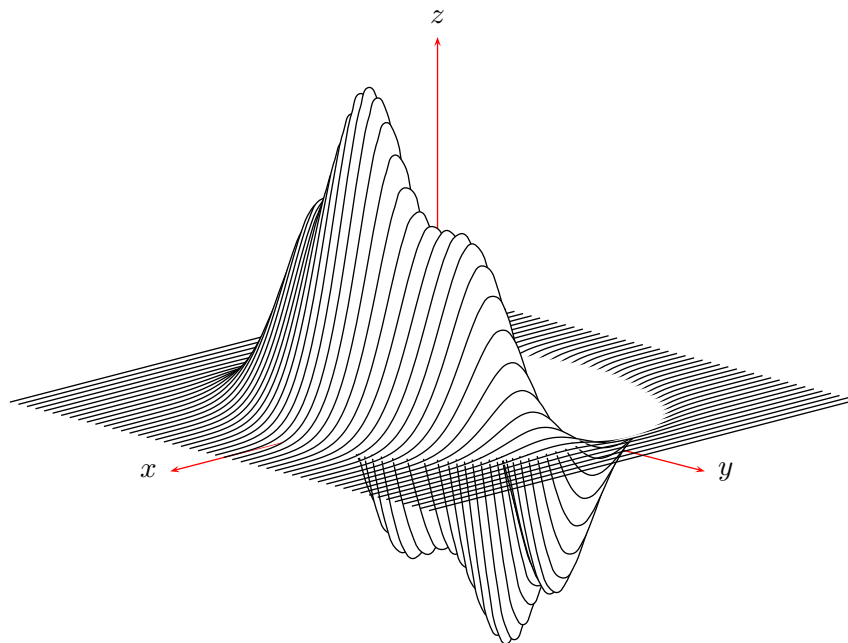
$$\begin{aligned}
 x &= f(t, u) \\
 y &= f(t, u) \\
 z &= f(t, u)
 \end{aligned}
 \tag{8}$$

To draw a spiral we have the parametric functions:

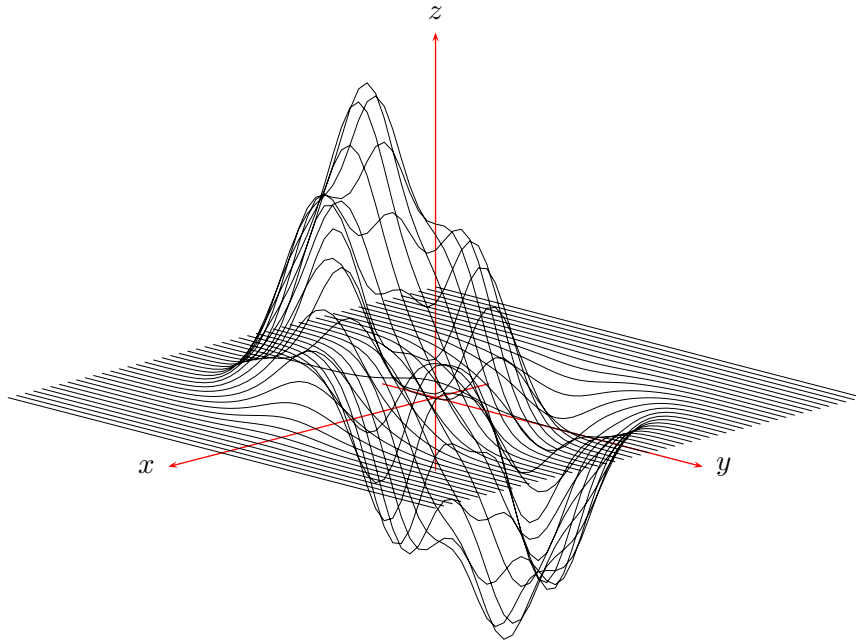
$$\begin{aligned}
 x &= r \cos t \\
 y &= r \sin t \\
 z &= t/600
 \end{aligned}
 \tag{9}$$



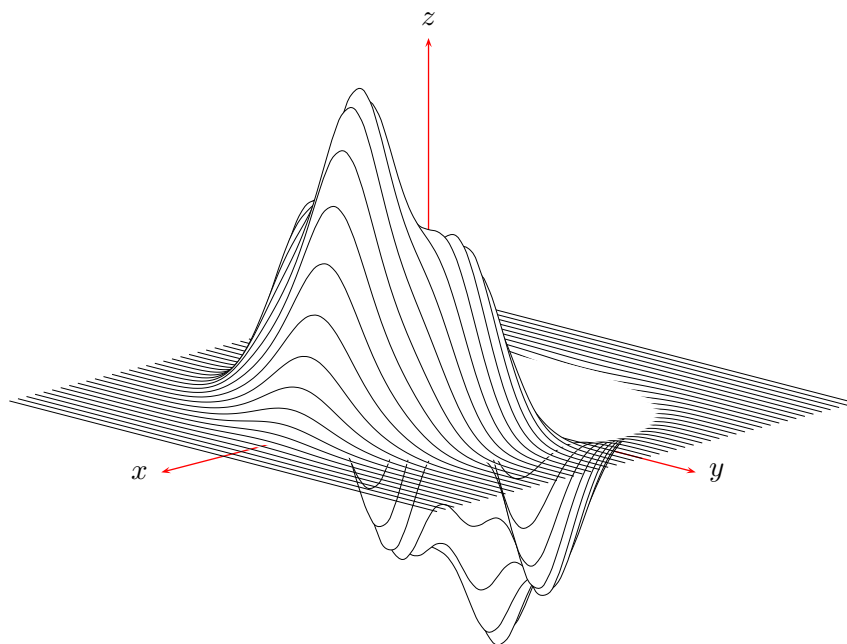
**Figure 6:** Plot of the equation 7



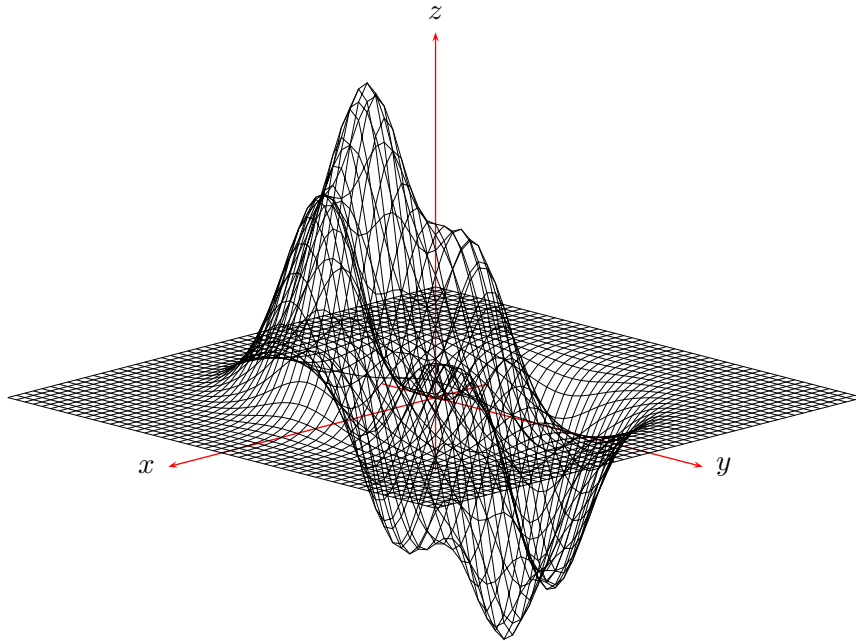
**Figure 7:** Plot of the equation 7 with the hiddenLine=true option



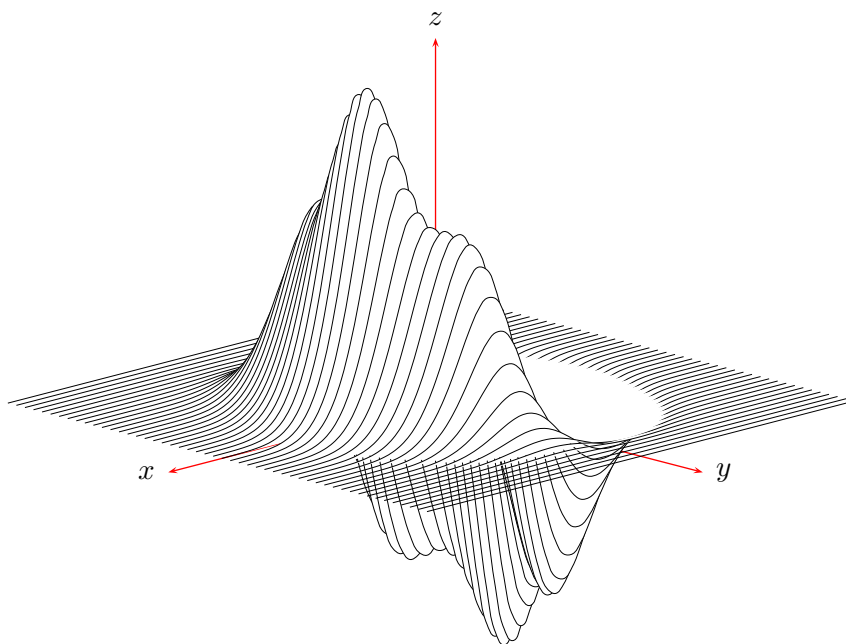
**Figure 8:** Plot of the equation 7 with the `drawStyle=yLines` option



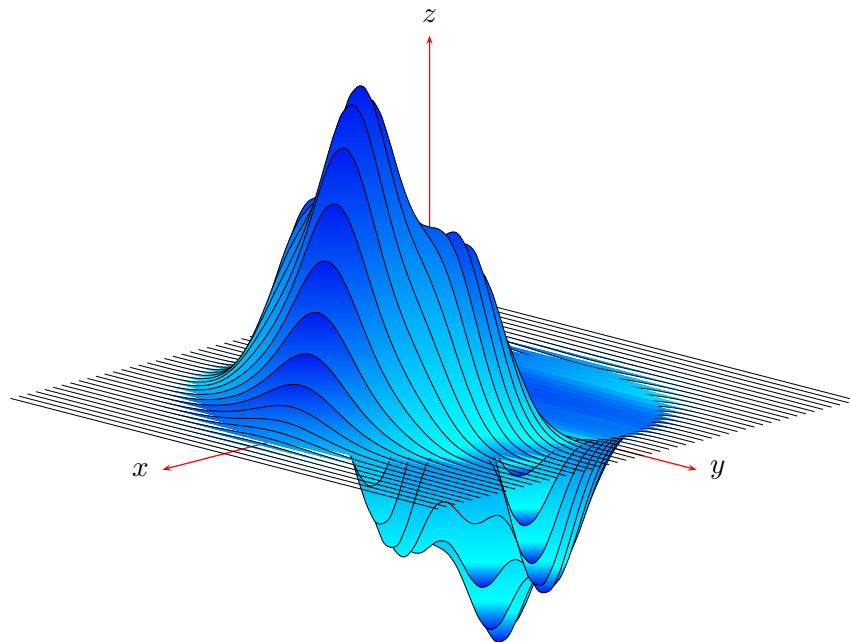
**Figure 9:** Plot of the equation 7 with the `drawStyle=yLines` and `hiddenLine=true` option



**Figure 10:** Plot of the equation 7 with the `drawStyle=xyLines` option



**Figure 11:** Plot of the equation 7 with the `drawStyle=xLines` and `hiddenLine=true` option

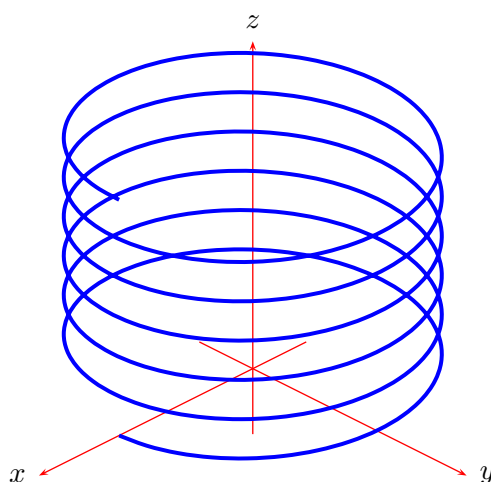


**Figure 12:** Plot of the equation 7 with the drawStyle=yLines and hiddenLine=true option

In the example the  $t$  value is divided by 600 for the  $z$  coordinate, because we have the values for  $t$  in degrees, here with a range of  $0^\circ \dots 2160^\circ$ . Drawing a curve in a three dimensional coordinate system does only require one parameter, which has to be by default  $t$ . In this case we do not need all parameters, so that one can write

`\parametricplotThreeD [Options] (t1,t2) {three parametric functions x y z}`

which is the same as  $(0,0)$  for the parameter  $u$ .

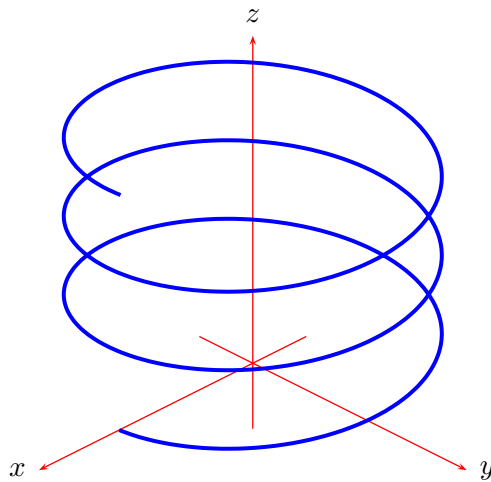


```

1 \begin{pspicture}(-3.25,-2.25)(3.25,5.25)
2   \pstThreeDCoor[zMax=5]
3   \parametricplotThreeD[xPlotpoints=200,
4     linecolor=blue,%
5     linewidth=1.5pt,plotstyle=curve](0,2160){%
6     2.5 t cos mul 2.5 t sin mul t 600 div}%degrees
7 \end{pspicture}

```

And the same with the algebraic option:



```
\begin{pspicture}(-3.25,-2.25)(3.25,5.25)
2 \pstThreeDCoor[zMax=5]
3 \parametricplotThreeD[xPlotpoints=200,
4   linecolor=blue,%
5   linewidth=1.5pt,plotstyle=curve,
6   algebraic](0,18.86){% radiant
7     2.5*cos(t) | 2.5*sin(t) | t/5.24}
8 \end{pspicture}
```

Instead of using the `\pstThreeDSphere` macro (see section 17) it is also possible to use parametric functions for a sphere. The macro plots continuous lines only for the  $t$  parameter, so a sphere plotted with the longitudes needs the parameter equations as

$$\begin{aligned}x &= \cos t \cdot \sin u \\y &= \cos t \cdot \cos u \\z &= \sin t\end{aligned}\tag{10}$$

The same is possible for a sphere drawn with the latitudes:

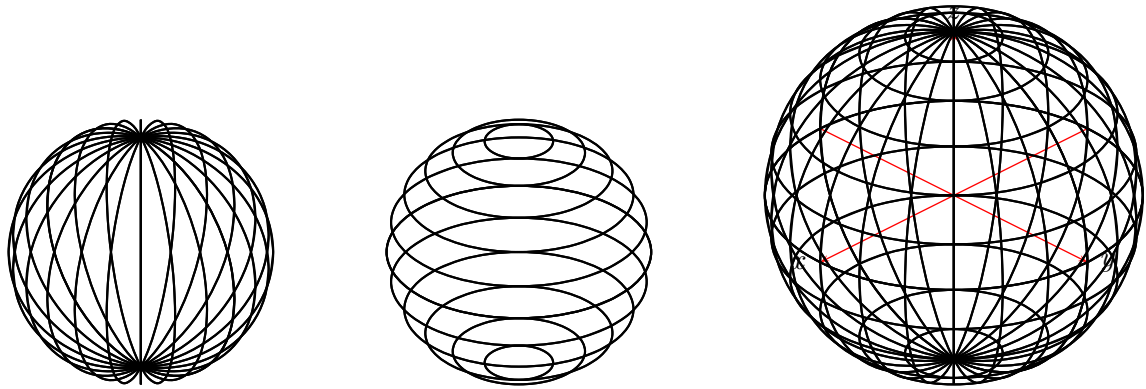
$$\begin{aligned}x &= \cos u \cdot \sin t \\y &= \cos u \cdot \cos t \\z &= \sin u\end{aligned}\tag{11}$$

and at last both together is also not a problem when having these parametric functions together in one `pspicture` environment (see figure 13).

```
1 \begin{pspicture}(-1,-1)(1,1)
2 \parametricplotThreeD[plotstyle=curve,yPlotpoints=40](0,360)(0,360){%
3   t cos u sin mul t cos u cos mul t sin
4 }
5 \parametricplotThreeD[plotstyle=curve,yPlotpoints=40](0,360)(0,360){%
6   u cos t sin mul u cos t cos mul u sin
7 }
8 \end{pspicture}
```

## 19 Plotting data files

There are the same conventions for data files which holds 3D coordinates, than for the 2D one. For example:



**Figure 13:** Different Views of the same Parametric Functions

```

0.0000  1.0000  0.0000
-0.4207  0.9972  0.0191
....

0.0000, 1.0000, 0.0000
-0.4207, 0.9972, 0.0191
....

(0.0000,1.0000,0.0000)
(-0.4207,0.9972,0.0191)
....

{0.0000,1.0000,0.0000}
{-0.4207,0.9972,0.0191}
....

```

There are the same three plot functions:

```

\fileplotThreeD [Options] {<datafile>}
\dataplotThreeD [Options] {data object}
\listplotThreeD [Options] {data object}

```

The in the following examples used data file has 446 entries like

6.26093349..., 2.55876582..., 8.131984...

This may take some time on slow machines when using the `\listplotThreeD` macro. The possible options for the lines are the ones from table 4.

## 19.1 \fileplotThreeD

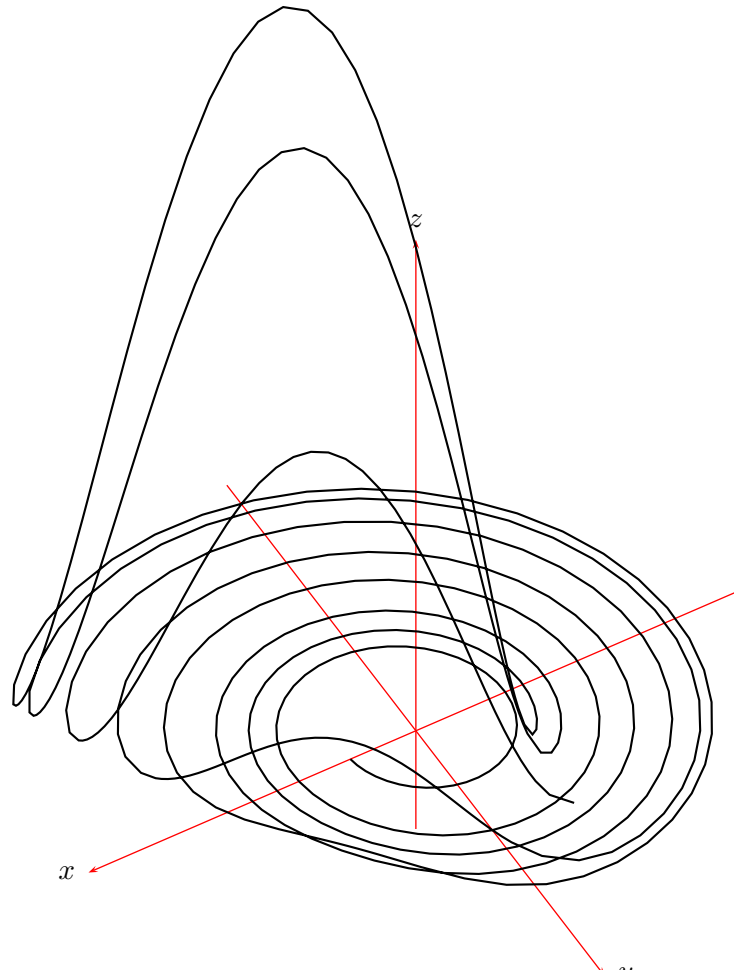
The syntax is very easy

```

\fileplotThreeD [Options] {datafile}

```

If the data file is not in the same directory than the document, insert the file name with the full path. Figure 15 shows a file plot with the option `linestyle=line`.



```

1 \begin{pspicture}(-6,-3)(6,10)
2   \psset{xunit=0.5cm,yunit=0.75cm,Alpha=30,Beta=30}% the global parameters
3   \pstThreeDCor[xMin=-10,xMax=10,yMin=-10,yMax=10,zMin=-2,zMax=10]
4   \fileplotThreeD[plotstyle=line]{data3D.Roessler}
5 \end{pspicture}%

```

**Figure 14:** Demonstration of \fileplotThreeD with Alpha=30 and Beta=15



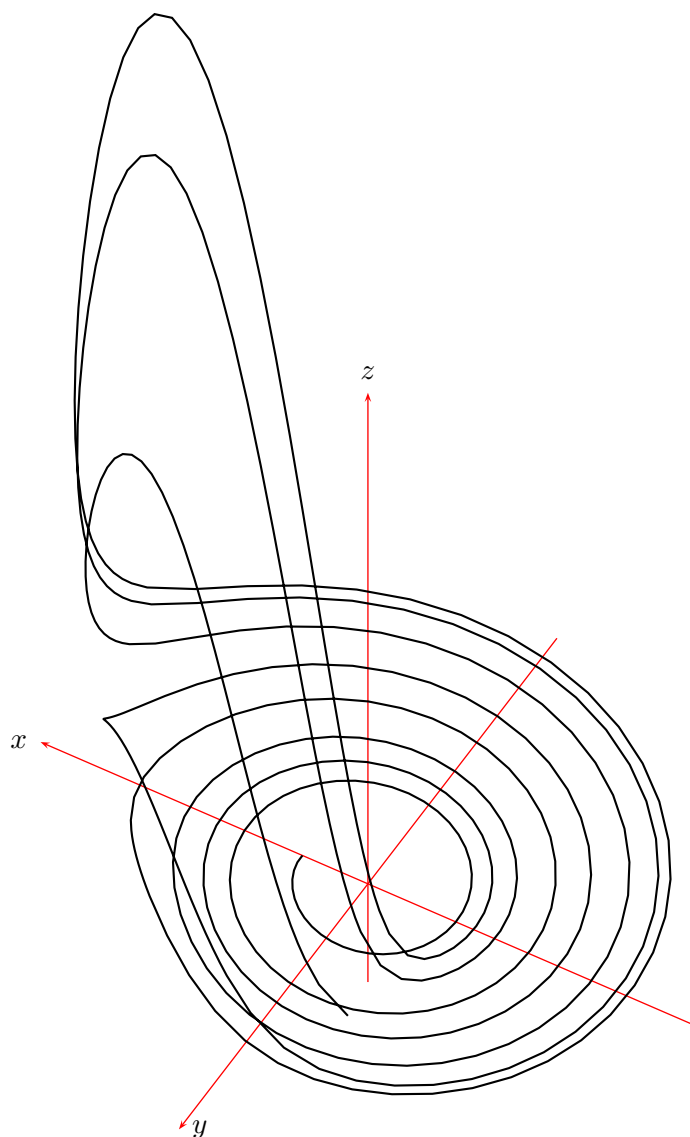
## 19.2 \dataplotThreeD

The syntax is

```
\dataplotThreeD [Options] {<data object>}
```

In difference to the macro \fileplotThreeD the \dataplotThreeD cannot plot any external data without reading this with the macro \readdata which reads external data and save it in a macro, f.ex.: \dataThreeD.[2]

```
\readdata{data object}{datafile}
```



```
\begin{pspicture}(-4.5,-3.5)(4,11)
\psset{xunit=0.5cm,yunit=0.75cm,Alpha=-30}
\pstThreeDCoor[xMin=-10,xMax=10,yMin=-10,yMax=10,zMin=-2,zMax=10]
\dataplotThreeD[plotstyle=line]{\dataThreeD}
\end{pspicture}%
```

**Figure 15:** Demonstration of \dataplotThreeD with Alpha=-30 and Beta=30

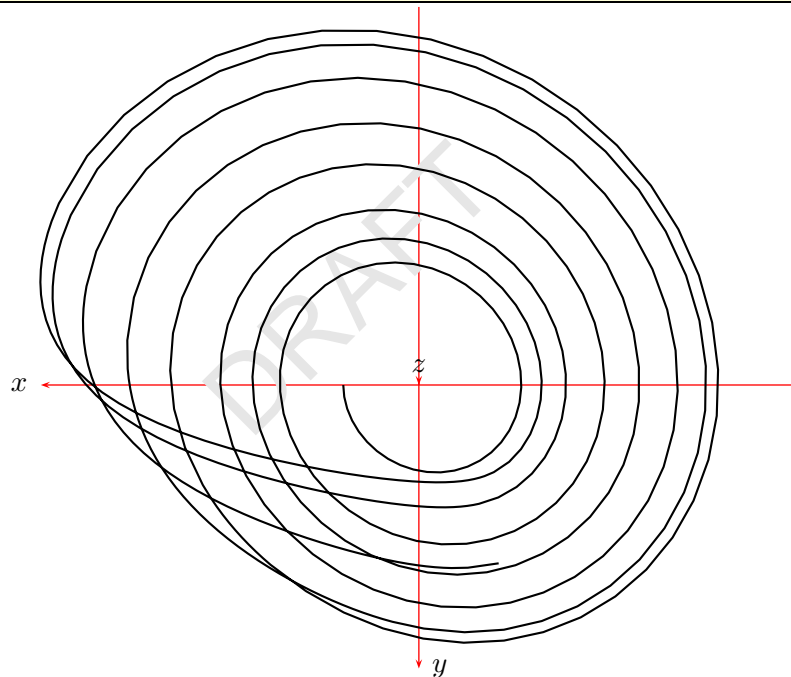
## 19.3 \listplotThreeD

The syntax is

```
\listplotThreeD [Options] {data object}
```

\listplotThreeD is similar to \dataplotThreeD, so it cannot plot any external data in a direct way, too. But \readdata reads external data and saves it in a macro, f.ex.: \dataThreeD.[2] \listplot can handle some additional PostScript code, which can be appended to the data object, f.ex.:

```
1 \dataread{\data}{data3D.Roessler}
2 \newcommand{\dataThreeDDraft}{%
3   \data\space
4   gsave      % save graphic status
5   /Helvetica findfont 40 scalefont setfont
6   45 rotate  % rotate 45 degrees
7   0.9 setgray % 1 list white
8   -60 30 moveto (DRAFT) show
9   grestore
10 }
```



```
1 \begin{pspicture}(-5,-4)(5,4)
2   \psset{xunit=0.5cm,yunit=0.5cm,Alpha=0,Beta=90}
3   \pstThreeDCoor[xMin=-10,xMax=10,yMin=-10,yMax=7.5,zMin=-2,zMax=10]
4   \listplotThreeD[plotstyle=line]{\dataThreeDDraft}
5 \end{pspicture}%
```

**Figure 16:** Demonstration of \listplotThreeD with a view from above (Alpha=0 and Beta=90) and some additional PostScript code

Figure 16 shows what happens with this code. For another example see [5], where the macro `\ScalePoints` is modified. This macro is in `pst-3dplot` called `\ScalePointsThreeD`.

## 20 Utility macros

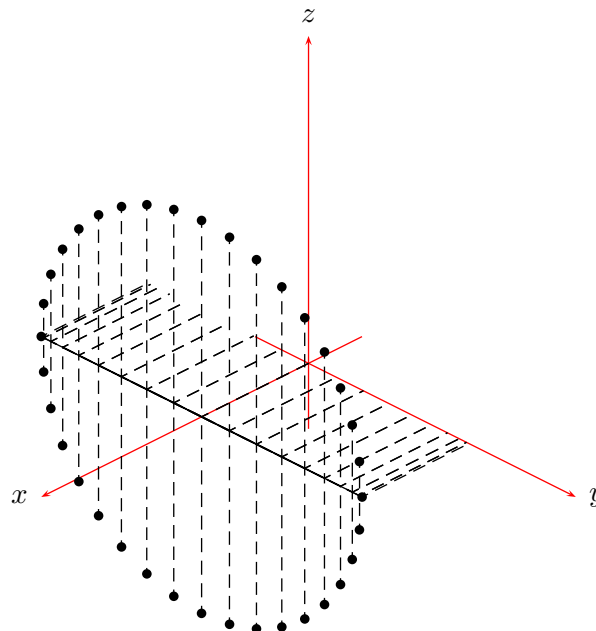
### 20.1 Rotation of three dimensional coordinates

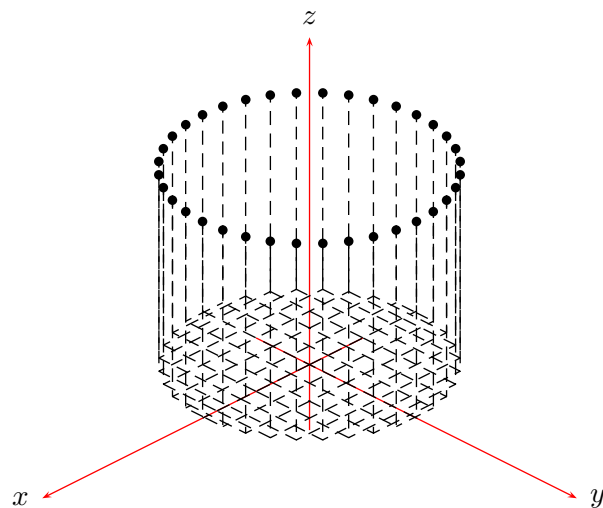
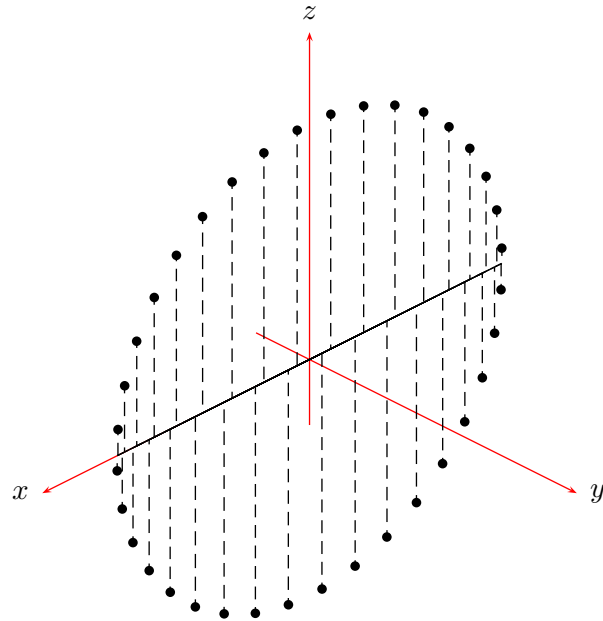
With the three optional arguments `RotX`, `RotY` and `RotZ` one can rotate a three dimensional point. This makes only sense when one wants to save the coordinates. In general it is more powerful to use directly the optional parameters `RotX`, `RotY`, `RotZ` for the plot macros. However, the macro syntax is

```
\pstRotP0intIIID [RotX=...,RotY=...,RotZ=...] (x,y,z)\xVal\yVal\zVal
```

the `\xVal` `\yVal` `\zVal` hold the new rotated coordinates and must be defined by the user like `\def\xVal{}`, where the name of the macro is not important.

The rotation angles are all predefined to 0 degrees.





```

1 \def\xVal{}\def\yVal{}\def\zVal{}
2 \begin{pspicture}(-6,-4)(6,5)
3   \pstThreeDCoor[xMin=-1,xMax=5,yMin=-1,yMax=5,zMin=-1,zMax=5]
4   \multido{\iA=0+10}{36}{\pstRotPointIIID[RotX=\iA](2,0,3){\xVal}{\yVal}{\zVal}
5     \pstThreeDDot[drawCoor=true](\xVal,\yVal,\zVal)
6   }
7 \end{pspicture}
8

```

```

9 \begin{pspicture}(-6,-4)(6,5)
10 \pstThreeDCoor[xMin=-1,xMax=5,yMin=-1,yMax=5,zMin=-1,zMax=5]
11 \multido{\iA=0+10}{36}{\pstRotPointIIID[RotY=\iA](2,0,3){\xVal}{\yVal}{\zVal}
12 \pstThreeDDot[drawCoor=true](\xVal,\yVal,\zVal)
13 }
14 \end{pspicture}
15
16 \begin{pspicture}(-6,-4)(6,5)
17 \pstThreeDCoor[xMin=-1,xMax=5,yMin=-1,yMax=5,zMin=-1,zMax=5]
18 \multido{\iA=0+10}{36}{\pstRotPointIIID[RotZ=\iA](2,0,3){\xVal}{\yVal}{\zVal}
19 \pstThreeDDot[drawCoor=true](\xVal,\yVal,\zVal)
20 }
21 \end{pspicture}

```

## 20.2 Transformation of coordinates

To run the macros with more than 9 parameters `pst-3dplot` uses the syntax (#1) for a collection of three coordinates (#1,#2,#3). To handle these triple in PostScript the following macro is used, which converts the parameter #1 into a sequence of the three coordinates, divided by a space. The syntax is:

```
\getThreeDCoor(<vector>)<\macro>
```

`\macro` holds the sequence of the three coordinates `x y z`, divided by a space.

## 20.3 Adding two vectors

The syntax is

```
\pstaddThreeDVec(<vector A>)(<vector B>)\tempa\tempb\tempc
```

`\tempa\tempb\tempc` must be user or system defined macros, which holds the three coordinates of the vector  $\vec{C} = \vec{A} + \vec{B}$ .

## 20.4 Subtract two vectors

The syntax is

```
\pstsubThreeDVec(<vector A>)(<vector B>)\tempa\tempb\tempc
```

`\tempa\tempb\tempc` must be user or system defined macros, which holds the three coordinates of the vector  $\vec{C} = \vec{A} - \vec{B}$ .

## 21 PDF output

`pst-3dplot` is based on the popular `pstricks` package and writes pure PostScript code[3], so it is not possible to run  $\text{\TeX}$  files with `pdf $\text{\LaTeX}$`  when there are `pstricks`

macros in the document. If you still need a PDF output use one of the following possibilities:

- package pdftricks[6]
- the for Linux free available program VTeX/Lnx<sup>3</sup>
- build the PDF with ps2pdf (dvi→ps→pdf)
- use the pst-pdf package.<sup>4</sup>

If you need package graphicx load it before any pstricks package. You do not need to load pstricks.sty, it will be done by pst-3dplot by default.

## 22 FAQ

- The labels for the axis are not right placed in the preview.  
Be sure that you view your output with a dvi viewer which can show PostScript code, like kdvi but not xdvi. It is better to run dvips and then view the ps-file with gv.

- The three axes have a wrong intersection point.

Be sure that you have the "newest" pst-node package

```
\def\fileversion{97 patch 11}
\def\filedate{2000/11/09}
```

and the "newest" pst-plot

```
\def\fileversion{97 patch 2}
\def\filedate{1999/12/12}
```

- Using amsmath and \hat or other accents as label for the axes gives an error. In this case save prevent expanding with e.g.: \psset{nameX=\$\noexpand\hat{x}\$}.

## References

- [1] Michel Goossens, Frank Mittelbach, Sebastian Rahtz, Denis Roegel, and Herbert Voß. *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 2007.
- [2] Laura E. Jackson and Herbert Voß. Die Plot-Funktionen von pst-plot. *Die T<sub>E</sub>Xnische Komödie*, 2/02:27–34, June 2002.
- [3] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.

<sup>3</sup> <http://www.micropress-inc.com/linux/>

<sup>4</sup> <http://www.perce.de/LaTeX/ps4pdf/>

- [4] Manuel Luque. *Vue en 3D*. <http://members.aol.com/Mluque5130/vue3d16112002.zip>, 2002.
- [5] Herbert Voß. Die mathematischen Funktionen von Postscript. *Die T<sub>E</sub>Xnische Komödie*, 1/02:40–47, March 2002.
- [6] Herbert Voss. *PSTricks Support for pdf*. <http://PSTricks.de/pdf/pdfoutput.phtml>, 2002.
- [7] Herbert Voß. *L<sup>A</sup>T<sub>E</sub>X Referenz*. DANTE – Lehmanns, Heidelberg/Hamburg, 1. edition, 2007.
- [8] Herbert Voß. *PSTricks – Grafik für T<sub>E</sub>X und L<sup>A</sup>T<sub>E</sub>X*. DANTE – Lehmanns, Heidelberg/Hamburg, 5. edition, 2008.
- [9] Michael Wiedmann and Peter Karp. *References for T<sub>E</sub>X and Friends*. <http://www.miwie.org/tex-refs/>, 2003.
- [10] Timothy Van Zandt. *PSTricks - PostScript macros for Generic TeX*. <http://www.tug.org/application/PSTricks>, 1993.

## Index

### A

Alpha, 6–8, 12, 13, 26  
amsmath, 62

### B

Beta, 6–8, 12, 13, 26

### C

c, 24  
concat, 18  
coordinates, 27  
coorType, 12, 13

### D

\dataplotThreeD, 55, 57, 58  
\dataThreeD, 57, 58  
dotstyle, 27  
drawCoor, 27  
drawing, 7  
Dx, 7  
Dy, 7  
Dz, 7

### E

Environment  
– pspicture, 54  
eulerRotation, 8

### F

\fileplotThreeD, 55–57

### G

\getThreeDCoor, 61  
graphicx, 62

### H

\hat, 62

### I

IIIDlabels, 7, 10  
IIIDticks, 7, 10  
IIIDticksize, 7  
IIIDxticksep, 7  
IIIDxTicksPlane, 7  
IIIDyticksep, 7  
IIIDyTicksPlane, 7

IIIDzticksep, 7

IIIDzTicksPlane, 7

### K

keep, 18

Keyword

- Alpha, 6–8, 12, 13, 26
- Beta, 6–8, 12, 13, 26
- coorType, 12, 13
- dotstyle, 27
- drawCoor, 27
- drawing, 7
- Dx, 7
- Dy, 7
- Dz, 7
- eulerRotation, 8
- IIIDlabels, 7, 10
- IIIDticks, 7, 10
- IIIDticksize, 7
- IIIDxticksep, 7
- IIIDxTicksPlane, 7
- IIIDyticksep, 7
- IIIDyTicksPlane, 7
- IIIDzticksep, 7
- IIIDzTicksPlane, 7
- linestyle, 55
- nameX, 7
- nameY, 7
- nameZ, 7
- plane, 24, 26
- planecorr, 24–26
- RotAngle, 8, 17
- RotSequence, 8, 17
- RotSet, 8, 17, 18
- RotX, 7, 15, 59
- RotY, 7, 15, 59
- RotZ, 7, 15, 59
- SphericalCoor, 6
- spotX, 7
- spotY, 7
- spotZ, 7
- subticks, 21
- xMax, 7
- xMin, 7



- xRotVec, 8, 17
- xsubticks, 21
- yMax, 7
- yMin, 7
- yRotVec, 8, 17
- ysubticks, 21
- zMax, 7
- zMin, 7
- zRotVec, 8, 17

## L

- latitude angle, 6
- lB, 24
- line, 55
- linestyle, 55
- \listplot, 58
- \listplotThreeD, 55, 58
- longitude angle, 6

## M

### Macro

- \dataplotThreeD, 55, 57, 58
- \dataThreeD, 57, 58
- \fileplotThreeD, 55–57
- \getThreeDCoor, 61
- \hat, 62
- \listplot, 58
- \listplotThreeD, 55, 58
- \parametricplotThreeD, 49, 53
- \psBox, 34
- \psCylinder, 42
- \psline, 27
- \psplotThreeD, 48
- \psset, 6, 21
- \pstaddThreeDVec, 61
- \pstIIIDCylinder, 40
- \pstParaboloid, 45
- \pstPlanePut, 24, 26
- \pstRotPointIIID, 59
- \pstsubThreeDVec, 61
- \pstThreeDBox, 32
- \pstThreeDCircle, 38
- \pstThreeDCoor, 7, 18
- \pstThreeDDot, 27
- \pstThreeDEllipse, 37
- \pstThreeDLine, 28
- \pstThreeDNode, 27

- \pstThreeDPlaneGrid, 21
- \pstThreeDPut, 24
- \pstThreeDput, 45
- \pstThreeDSphere, 47, 54
- \pstThreeDSquare, 31
- \readdata, 57, 58
- \rput, 24
- \ScalePoints, 59
- \ScalePointsThreeD, 59
- \space, 7

## N

- nameX, 7
- nameY, 7
- nameZ, 7
- none, 27
- normal, 26

## O

- off, 26

## P

### Package

- amsmath, 62
- graphicx, 62
- pdftricks, 62
- pst-3dplot, 4, 6, 17, 27, 59, 61, 62
- pst-node, 62
- pst-pdf, 62
- pst-plot, 4, 27, 62
- pst-xkey, 4
- pstricks, 4, 24
- xcolor, 45
- xkeyval, 4
- \parametricplotThreeD, 49, 53
- pdftricks, 62
- plane, 24, 26
- planecorr, 24–26
- \psBox, 34
- \psCylinder, 42
- \psline, 27
- pspicture, 54
- \psplotThreeD, 48
- \psset, 6, 21
- pst-3dplot, 4, 6, 17, 27, 59, 61, 62
- pst-node, 62
- pst-pdf, 62
- pst-plot, 4, 27, 62

pst-xkey, 4  
\pstaddThreeDVec, 61  
\pstIIIDCylinder, 40  
\pstParaboloid, 45  
\pstPlanePut, 24, 26  
pstricks, 4, 24  
\pstRotPointIIID, 59  
\pstsubThreeDVec, 61  
\pstThreeDBox, 32  
\pstThreeDCircle, 38  
\pstThreeDCoor, 7, 18  
\pstThreeDDot, 27  
\pstThreeDEllipse, 37  
\pstThreeDLine, 28  
\pstThreeDNode, 27  
\pstThreeDPlaneGrid, 21  
\pstThreeDPut, 24  
\pstThreeDput, 45  
\pstThreeDSphere, 47, 54  
\pstThreeDSquare, 31

## Q

quaternion, 17

## R

\readdata, 57, 58  
RotAngle, 8, 17  
RotSequence, 8, 17  
RotSet, 8, 17, 18  
RotX, 7, 15, 59  
RotY, 7, 15, 59  
RotZ, 7, 15, 59  
\rput, 24

## S

\ScalePoints, 59  
\ScalePointsThreeD, 59  
set, 18  
\space, 7  
SphericalCoor, 6  
spotX, 7  
spotY, 7  
spotZ, 7  
subticks, 21

## T

true, 6

## V

Value

- c, 24
- concat, 18
- keep, 18
- lB, 24
- line, 55
- none, 27
- normal, 26
- off, 26
- quaternion, 17
- RotSet, 18
- set, 18
- true, 6
- xLines, 49
- xy, 21, 24, 26
- xyLines, 49
- xyrot, 26
- xz, 21
- yLines, 49
- yxLines, 49
- yz, 21

## X

xcolor, 45  
xkeyval, 4  
xLines, 49  
xMax, 7  
xMin, 7  
xRotVec, 8, 17  
xsubticks, 21  
xy, 21, 24, 26  
xyLines, 49  
xyrot, 26  
xz, 21

## Y

yLines, 49  
yMax, 7  
yMin, 7  
yRotVec, 8, 17  
ysubticks, 21  
yxLines, 49  
yz, 21

## Z

zMax, 7  
zMin, 7

zRotVec, 8, 17