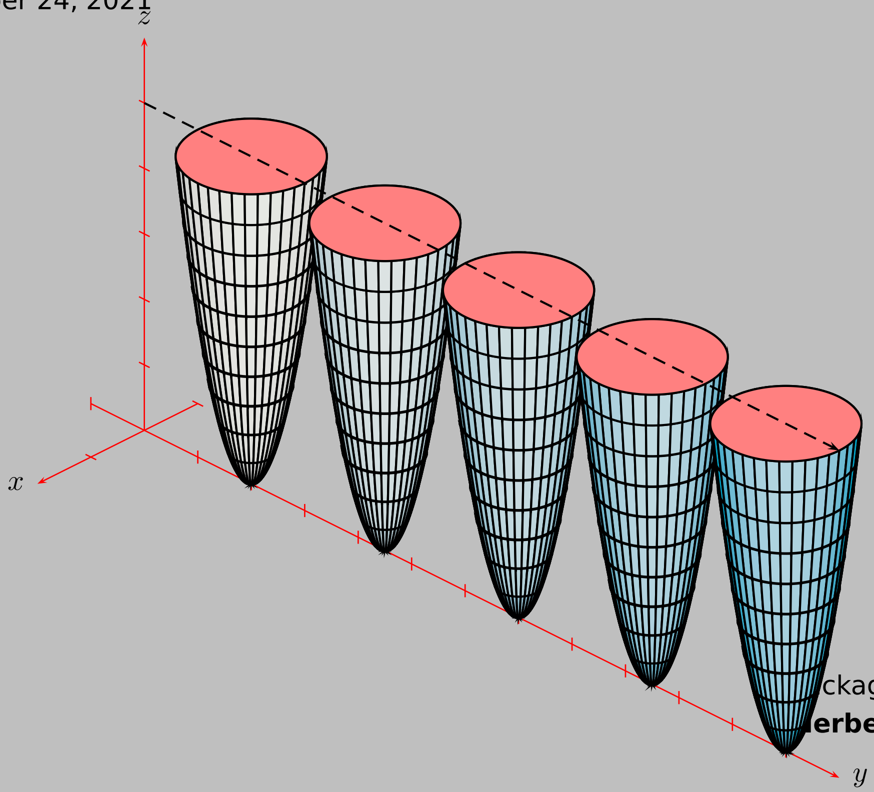
PSTricks

3D plots: pst-3dplot

A PSTricks package for drawing 3d objects, v2.07

September 24, 2021



ckage author(s):
erbert Voß

Contents

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 similiar 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 spcecial one pst-xkey for PSTricks. The xkeyval package is available at 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, Bernhard Elsner, Andreas Fehlner, Christophe Jorssen, Markus Krebs, Chris Kuklewicz, Darrell Lamm, Patrice Mégret, Rolf Niepraschk, Michael Sharpe, Uwe Siart, Thorsten Suhling, Maja Zaloznik

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).

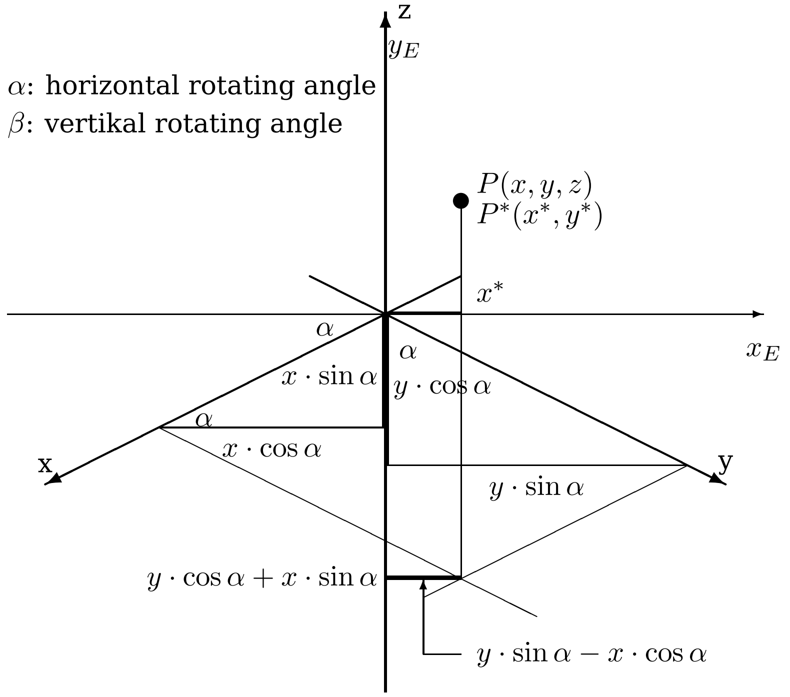


Figure 1: Lengths in a three dimensional System

The angle α is the horizontal rotation with positive values for anti clockwise rotations of the 3D coordinates. The angle β 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 β is greater than 0 degrees.

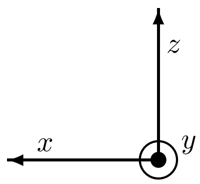


Figure 2: Coordinate System for $\alpha = \beta = 0$ (y-axis comes out of the paper plane)

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):

$$x^* = -x \cdot \cos\alpha + y \cdot \sin\alpha \tag{1}$$

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 β angle is well seen in figure 3 which derives from figure 2, if the coordinate system is rotated by 90° horizontally to the left and vertically by β also to the left.

The value of the perpendicular projected z coordinate is $z^* = z \cdot cos\beta$. With figure 3 we see, that

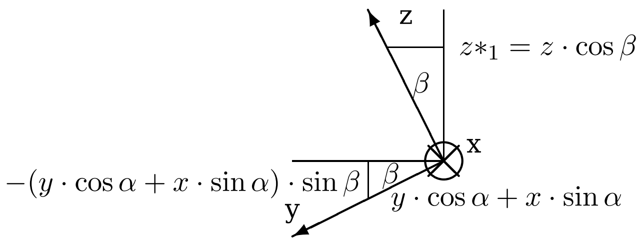


Figure 3: Coordinate System for $\alpha = 0$ and $\beta > 0$ (x-axis goes into the paper plane)

the point $P(x, y, z)$ runs on an elliptical curve when β is constant and α changes continues. 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 β, so we have them to multiply with $\sin\beta$ to get the vertical part. We get the following transformation equations:

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

or written in matrix form:

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

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,ϕ,θ), where r is the radius, ϕ the longitude angle and θ the lattitude angle. For the spherical coordinates set the option `SphericalCoor=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 TEX absorbs the space following a macro.)

The syntax for drawing the coordinate axes is

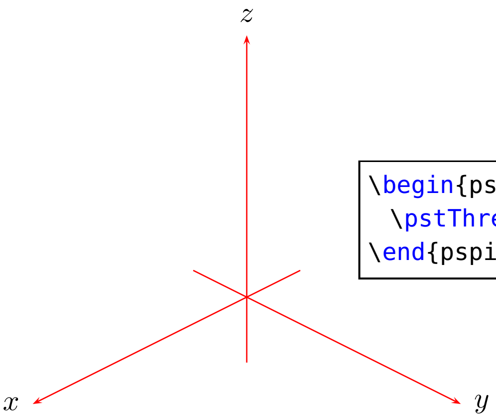
```
\pstThreeDCoor [Options]
```

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-3dplot`

name	type	Default	*page*
Alpha	<angle>	45	7
Beta	<angle>	30	7
xMin	<value>	-1	9
xMax	<value>	4	7
yMin	<value>	-1	9
yMax	<value>	4	7
zMin	<value>	-1	9
zMax	<value>	4	7
nameX	<string>	x	??
spotX	<angle>	180	??
nameY	<string>	y	??
spotY	<angle>	0	??
nameZ	<string>	z	??
spotZ	<angle>	90	??
IIIDticks	false|true	false	9
IIIDlabels	false|true	false	9
Dx	<value>	1	9
Dy	<value>	1	9
Dz	<value>	1	9
IIIDxTicksPlane	xy|xz|yz	xy	9
IIIDyTicksPlane	xy|xz|yz	yz	9
IIIDzTicksPlane	xy|xz|yz	yz	9
IIIDticksize	<value>	0.1	8
IIIDxticksep	<value>	-0.4	9
IIIDyticksep	<value>	-0.2	9
IIIDzticksep	<value>	0.2	9
RotX	<angle>	0	15
RotY	<angle>	0	15
RotZ	<angle>	0	15
RotAngle	<angle>	0	17
xRotVec	<angle>	0	17
yRotVec	<angle>	0	17
zRotVec	<angle>	0	17
RotSequence	xyz|xzy|yxz|yzx|zxy|zyx|quaternion	xyz	15
RotSet	set|concat|keep	set	17
eulerRotation	true|false	false	18
IIIDOffset	{<x,y,z>}	{0,0,0}	11
zlabelFactor	<text>	\relax	11
comma	false|true	false	9

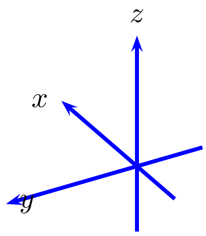



```
\begin{pspicture}(-3,-2.5)(3,4.25)
  \pstThreeDCoor
\end{pspicture}
```

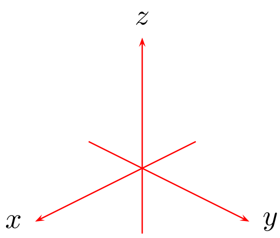
There are no restrictions for the angles and the max and min values for the axes; all pstricks options are possible as well. The following example changes the color and the width of the axes.

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.

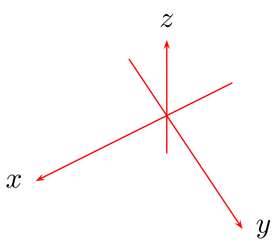
Alpha ist the horizontal and Beta the vertical rotation angle of the Cartesian coordinate system.



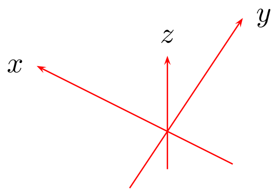

```
\begin{pspicture}(-2,-1.25)(1,2.25)
  \pstThreeDCoor[linewidth=1.5pt,linecolor=blue,
    xMax=2,yMax=2,zMax=2,
    Alpha=-60,Beta=30]
\end{pspicture}
```



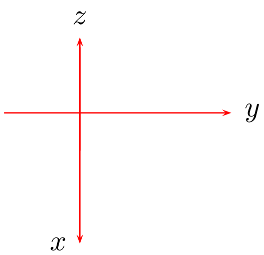

```
\begin{pspicture}(-2,-2)(2,2)
  \pstThreeDCoor[xMax=2,yMax=2,zMax=2]
\end{pspicture}
```



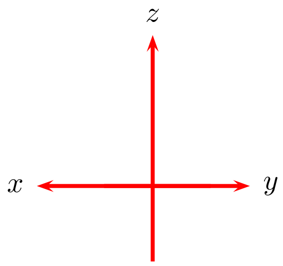

```
\begin{pspicture}(-2,-2)(2,2)
  \pstThreeDCoor[xMax=2,yMax=2,zMax=2,
     Alpha=30,Beta=60]
\end{pspicture}
```




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




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

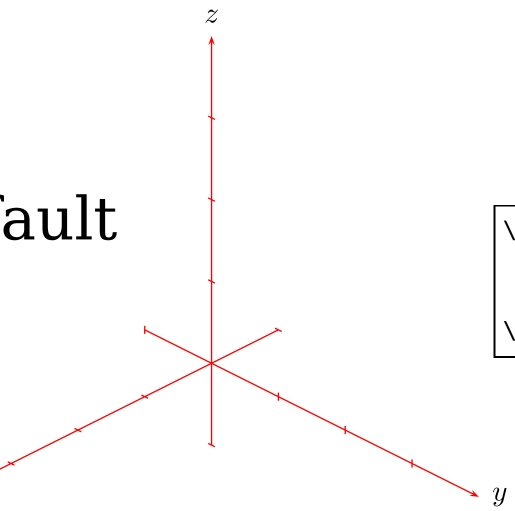



```
\begin{pspicture}(-2,-2)(2,2)
 \pstThreeDCoor[linewidth=1.5pt,
    xMax=2,yMax=2,zMax=2,
    Alpha=40,Beta=0]
\end{pspicture}
```

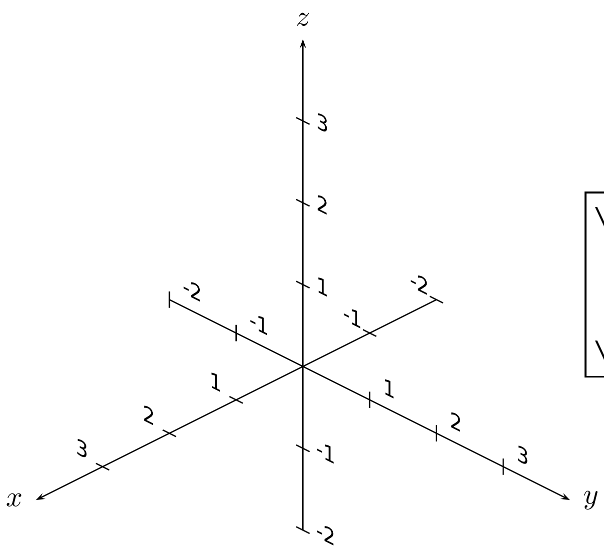
3.1 Ticks, comma and labels

With the option `IIIDticks` the axes get ticks and with `IIIDlabels` labels. Without ticks also labels are not possible. The optional argument `comma`, which is defined in the package `pst-plot` allows to use a comma instead of a dot for real values. 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

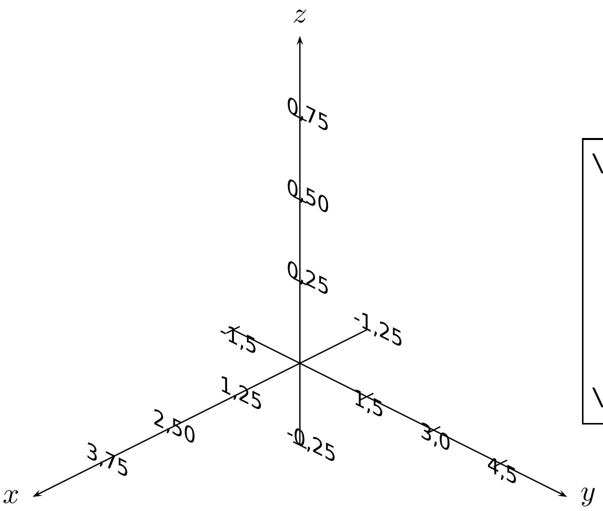
```
\def\psxyzlabel#1{\bgroup\footnotesize\textsf{#1}\egroup}
```




```
\begin{pspicture}(-3,-2.5)(3,4)
  \pstThreeDCoor[IIIDticks,IIIDticksize=0.05]%
  \pstThreeDPut(3,0,3){\Huge default}
\end{pspicture}
```

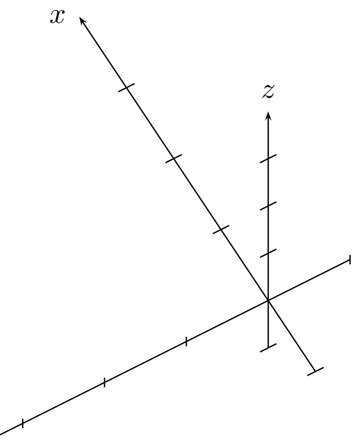



```
\begin{pspicture}(-3,-2.5)(3,4)
  \pstThreeDCoor[linecolor=black,
     IIIDticks,IIIDlabels,
     xMin=-2,yMin=-2,zMin=-2]
\end{pspicture}
```

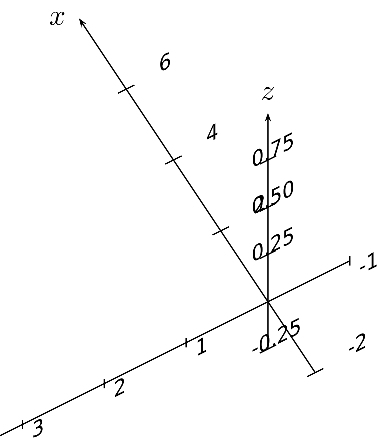



```
\begin{pspicture}(-3,-2.5)(3,4)
 \pstThreeDCoor[linecolor=black,
  IIIDticks,IIIDzTicksPlane=yz,
  IIIDzticksep=-0.2,IIIDlabels,
  IIIDxTicksPlane=yz,,IIIDxticksep=-0.2,
  IIIDyTicksPlane=xy,,IIIDyticksep=0.2,
  comma,Dx=1.25,Dy=1.5,Dz=0.25]
\end{pspicture}
```

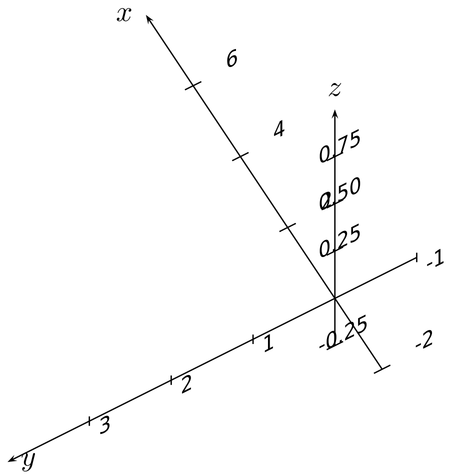
The following example shows a wrong placing of the labels, the planes should be changed.




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

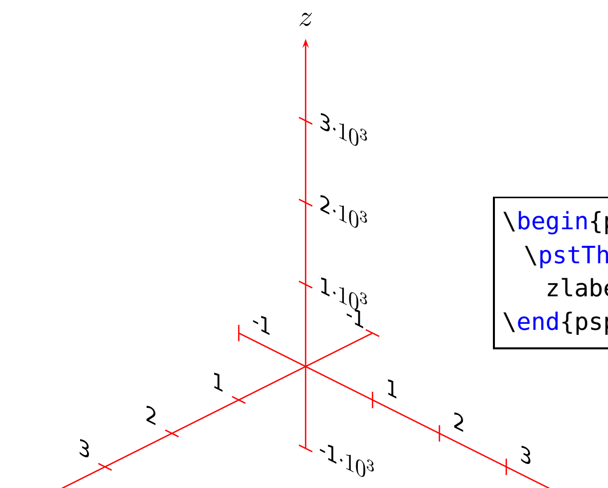



```
\psset{Alpha=-60,Beta=60}
\begin{pspicture}(-4,-2.25)(1,3)
 \pstThreeDCoor[linecolor=black,%
    IIIDticks,IIIDlabels,
    planecorr=normal,
    Dx=2,Dy=1,Dz=0.25]%
\end{pspicture}
```




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

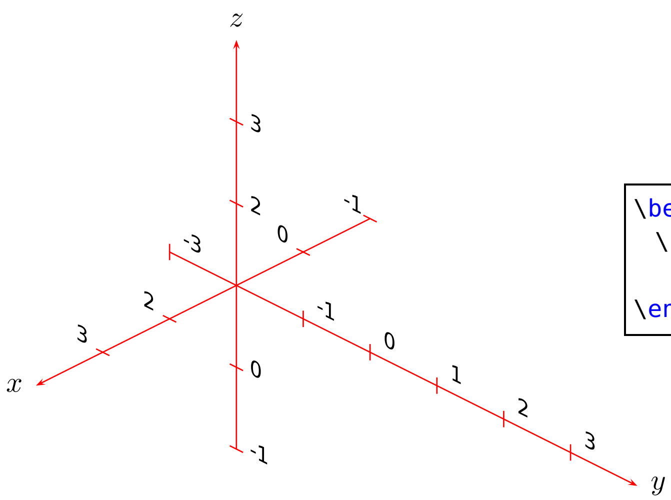
For the z axis it is possible to define a factor for the values, e.g.




```
\begin{pspicture}(-4,-2.25)(1,4)
 \pstThreeDCoor[IIIDticks,IIIDlabels,
   zlabelFactor=$\cdot10^3$]
\end{pspicture}
```

3.2 Offset

The optional argument `IIIDOffset` allows to set the intermediate point of all axes to another point as the default of `(0,0,0)`. The values have to be put into braces:

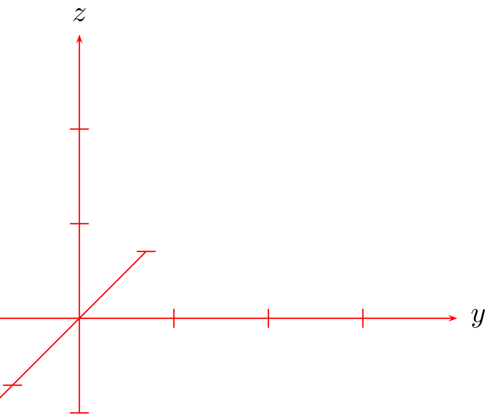



```
\begin{pspicture}(-4,-1.25)(1,4)
 \pstThreeDCoor[IIIDticks,IIIDlabels,
   yMin=-3,IIIDOffset={(1,-2,1)}]
\end{pspicture}
```

3.3 Experimental features

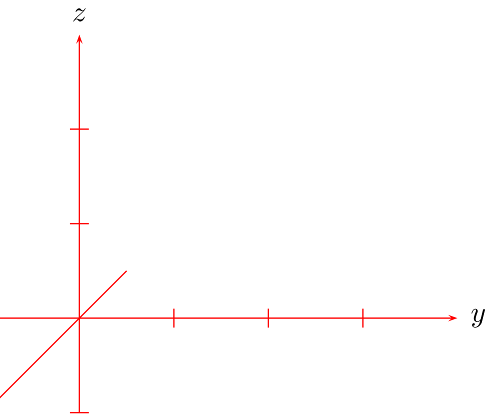
All features are as long as they are not really tested called experimental. With the optional argument coorType, which is by default 0, one can change the the viewing of the axes and all other three dimensional objects.

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.



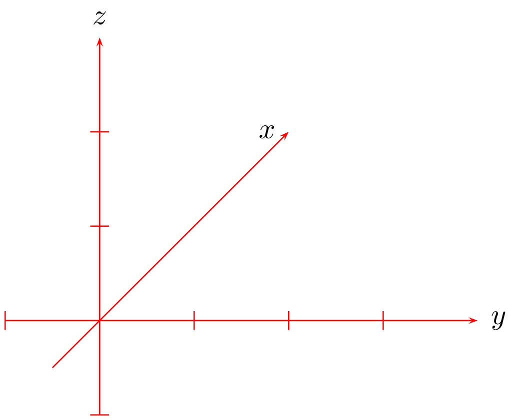

```
\psset{coorType=1,Alpha=135}
\begin{pspicture}(-2,-3)(3,3.5)
\pstThreeDCoor[IIIDticks,zMax=3]%
\end{pspicture}
```

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.



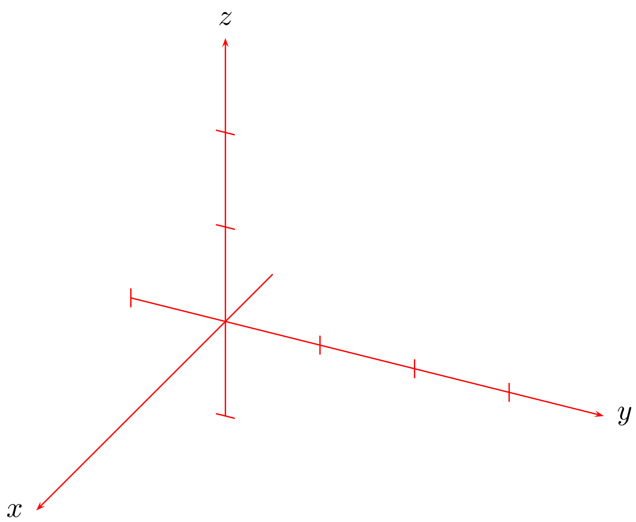

```
\psset{coorType=2,Alpha=90,
   IIIDxTicksPlane=yz}
\begin{pspicture}(-2,-2)(3,3.5)
\pstThreeDCoor[IIIDticks,zMax=3]%
\end{pspicture}
```

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.



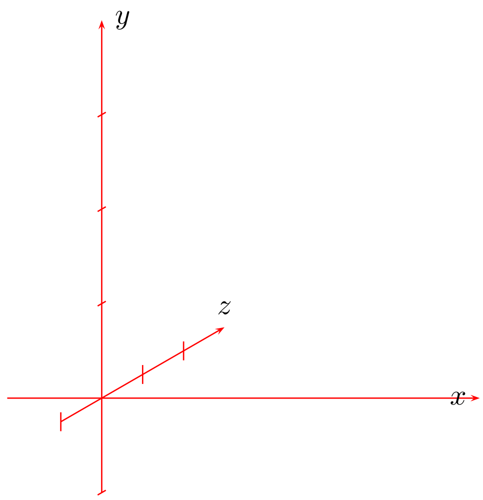

```
\psset{coorType=3,Alpha=90,
  IIIDxTicksPlane=yz}
\begin{pspicture}(-2,-2)(3,3.5)
\pstThreeDCoor[IIIDticks,zMax=3]%
\end{pspicture}
```

coorType=4 is also called the trimetrie-view. One angle of the axis is 5 and the other 15 degrees. The angles Alpha and Beta are not valid.



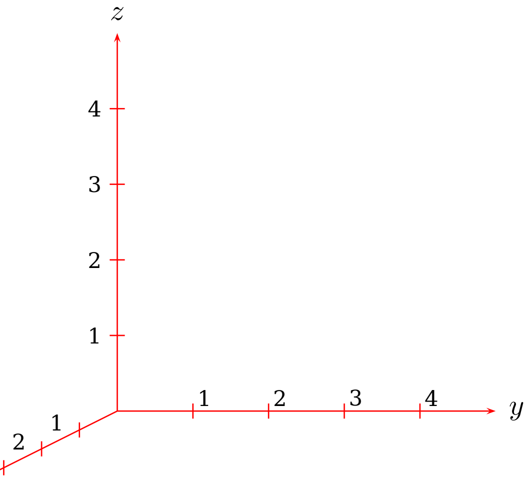

```
\psset{coorType=4,IIIDxTicksPlane=yz}
\begin{pspicture}(-2,-2)(3,3.5)
\pstThreeDCoor[IIIDticks,zMax=3]%
\end{pspicture}
```

With coorType=5 the y-z-axes are orthogonal and the angle between x- and y-axis is variable but should be 30 or 45 degrees and the x-axis is shortened by a factor of 0.5. The angle Beta is not valid.



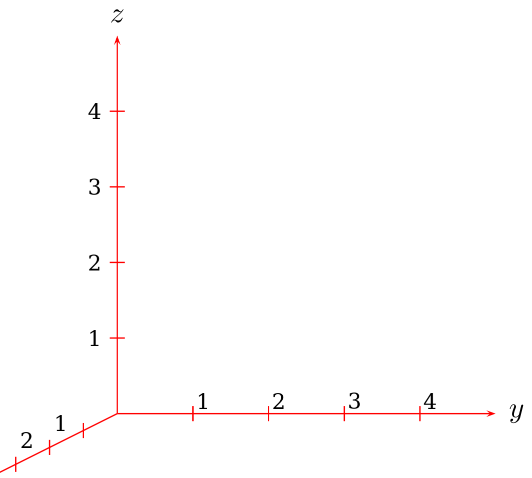

```
\psset{coorType=5,Alpha=30,
  IIIDxTicksPlane=yz}
\begin{pspicture}(-2,-2)(3,4)
\pstThreeDCoor[IIIDticks,zMax=3]%
\end{pspicture}
```

For coorType=6 the x-axis us shortend by 0.559.




```
\psset{coorType=6}
\begin{pspicture}(-3,-2)(6,6)
\psset{IIIDxTicksPlane=xz,IIIDyTicksPlane=yz}
\pstThreeDCoor[xMin=0,xMax=5,yMin=0,yMax=5,
 zMin=0,zMax=5,IIIDticks,spotX=180,
 IIIDlabels=false,linecolor=red]%
\multido{\iA=1+1}{4}{\footnotesize%
 \pstThreeDPut(\iA,-0.3,0.1){\iA}%
 \pstThreeDPut(-0.3,\iA,0.1){\iA}%
 \pstThreeDPut(0,-0.3,\iA){\iA}}
\end{pspicture}
```

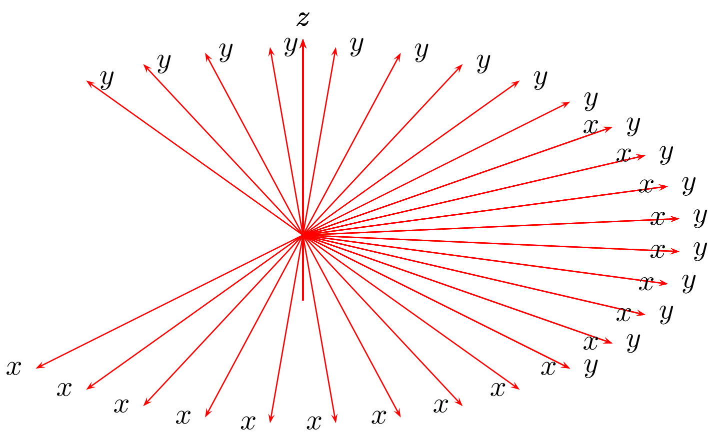
For coorType=7 the x-axis us shortend by 0.5.



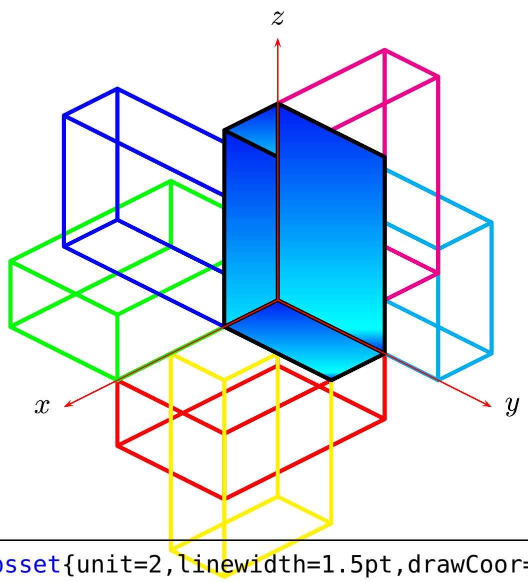

```
\psset{coorType=7}
\begin{pspicture}(-3,-2)(6,6)
\psset{IIIDxTicksPlane=xz,IIIDyTicksPlane=yz}
\pstThreeDCoor[xMin=0,xMax=5,yMin=0,yMax=5,
 zMin=0,zMax=5,IIIDticks,spotX=180,
 IIIDlabels=false,linecolor=red]%
\multido{\iA=1+1}{4}{\footnotesize%
 \pstThreeDPut(\iA,-0.3,0.1){\iA}%
 \pstThreeDPut(-0.3,\iA,0.1){\iA}%
 \pstThreeDPut(0,-0.3,\iA){\iA}}
\end{pspicture}
```

4 Rotation

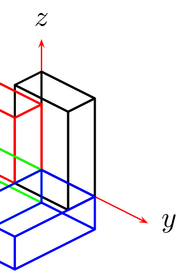
The coordinate system can be rotated independent from the given Alpha and Beta values. This makes it possible to place the axes in any direction and any order. There are the three options RotX, RotY, RotZ and an additional one for the rotating sequence (rotSequence), which can be any combination of the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




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

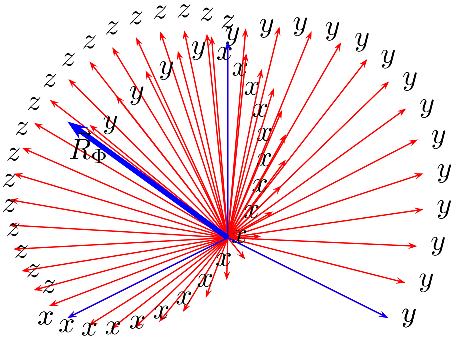



```
\psset{unit=2,linewidth=1.5pt,drawCoor=false}
\begin{pspicture}(-2,-1.5)(2,2.5)%
  \pstThreeDCoor[xMin=0,xMax=2,yMin=0,yMax=2,zMin=0,zMax=2]%
  \pstThreeDBox[RotX=90,RotY=90,RotZ=90,%
    linecolor=red](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
  \pstThreeDBox[RotSequence=xzy,RotX=90,RotY=90,RotZ=90,%
    linecolor=yellow](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
  \pstThreeDBox[RotSequence=zyx,RotX=90,RotY=90,RotZ=90,%
    linecolor=green](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
  \pstThreeDBox[RotSequence=zxy,RotX=90,RotY=90,RotZ=90,%
    linecolor=blue](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
  \pstThreeDBox[RotSequence=yxz,RotX=90,RotY=90,RotZ=90,%
    linecolor=cyan](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
  \pstThreeDBox[RotSequence=yzx,RotX=90,RotY=90,RotZ=90,%
    linecolor=magenta](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
  \pstThreeDBox[fillstyle=gradient,RotX=0](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
  \pstThreeDCoor[xMin=0,xMax=2,yMin=0,yMax=2,zMin=0,zMax=2]%
\end{pspicture}%
```




```
pspicture}(-2,-1.5)(2,2.5)%
reeDCoor[xMin=0,xMax=2,yMin=0,yMax=2,zMin=0,zMax=2]%
reeDBox(0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
reeDBox[RotX=90,linecolor=red](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
reeDBox[RotX=90,RotY=90,linecolor=green](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
reeDBox[RotX=90,RotY=90,RotZ=90,linecolor=blue](0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
picture}%
```

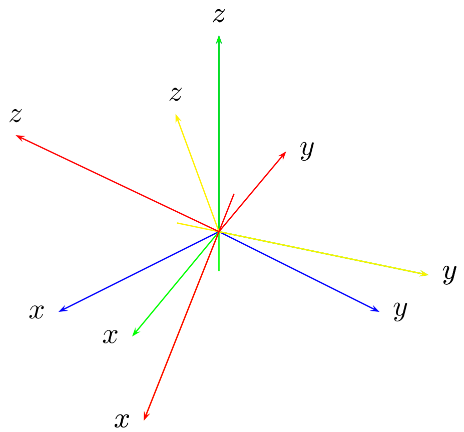
It is sometimes more convenient to rotate the coordinate system by specifying a *single* angle of rotation `RotAngle` (in degrees) about a vector whose coordinates are `xRotVec`, `yRotVec`, and `zRotVec` using the `quaternion` option for `RotSequence`.



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

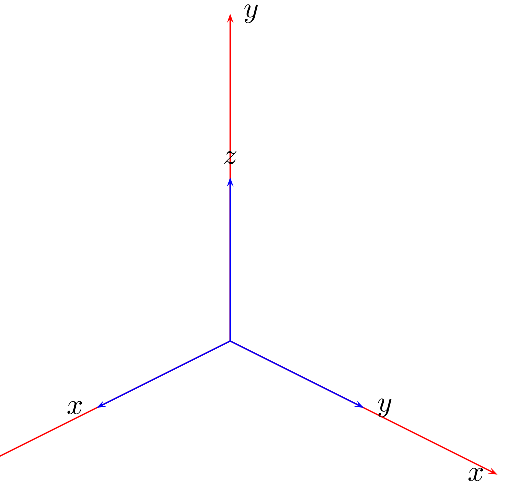
Rotations of the coordinate system may be “accumulated” by applying successive rotation sequences using the `RotSet` variable, which is set either as a `pst-3dplot` object’s optional argument, or with a `\psset[pst-3dplot]{RotSet=value}` command. The usual TEX scoping rules for the value of `RotSet` 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.

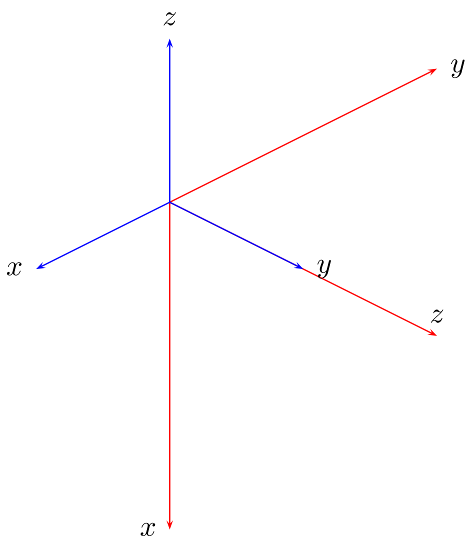



```
\begin{pspicture}(-3,-3)(3.6,3)
\pstThreeDCoor[linecolor=blue, RotSequence=quaternion, RotAngle=0, RotSet=set, xRotVec=0,yRotVec=0,zRot
   =1,
 xMin=0,xMax=3, yMin=0,yMax=3, zMin=0,zMax=3]
 \pstThreeDCoor[linecolor=green, RotSequence=quaternion, RotSet=concat, RotAngle=22.5, xRotVec=0,yRotVe
    zRotVec=1,
  xMin=0,xMax=3, yMin=0,yMax=3, zMin=-0.6,zMax=3]
 \pstThreeDCoor[linecolor=yellow, RotSequence=quaternion, RotSet=concat, RotAngle=30, xRotVec=0,yRotVec
    zRotVec=0,
  xMin=0,xMax=3,yMin=-0.6,yMax=3, zMin=0,zMax=3]
 \pstThreeDCoor[linecolor=red, RotSequence=quaternion, RotSet=concat, RotAngle=60, xRotVec=1,yRotVec=0,
    zRotVec=0,
  xMin=-0.6,xMax=3, yMin=0,yMax=3, zMin=0,zMax=3]%
\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`.




```
n{pspicture}(-4,-5)(6,5)
ThreeDCoor[linecolor=red, RotSequence=zyx, RotZ=90,RotY=90,RotX=0,
n=0,xMax=5, yMin=0,yMax=5, zMin=0,zMax=5]
ThreeDCoor[linecolor=blue, RotSequence=zyx, RotZ=0,RotY=0,RotX=0,
n=0,xMax=2.5, yMin=0,yMax=2.5, zMin=0,zMax=2.5]
pspicture}
```




```
\begin{pspicture}(-3,-5)(7,5)
 \pstThreeDCoor[eulerRotation=true, linecolor=red, RotSequence=zyx, RotZ=90, RotY=90, RotX=0,
  xMin=0,xMax=5, yMin=0,yMax=5, zMin=0,zMax=5]
 \pstThreeDCoor[linecolor=blue, RotSequence=zyx, RotZ=0,RotY=0,RotX=0,
  xMin=0,xMax=2.5, yMin=0,yMax=2.5, zMin=0,zMax=2.5]
\end{pspicture}
```

5 Plane Grids

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

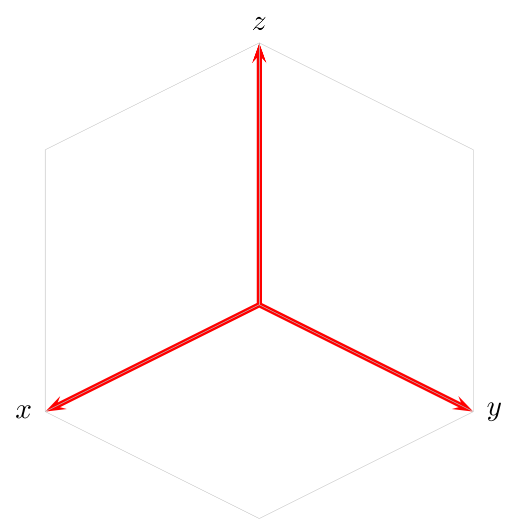
There are three additional options

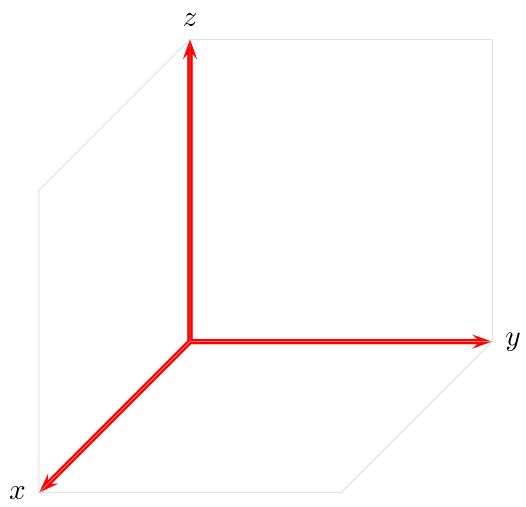
planeGrid can be one of the following values: xy, xz, yz. Default is xy.

subticks Number of ticks. Default is 10.[1]

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.

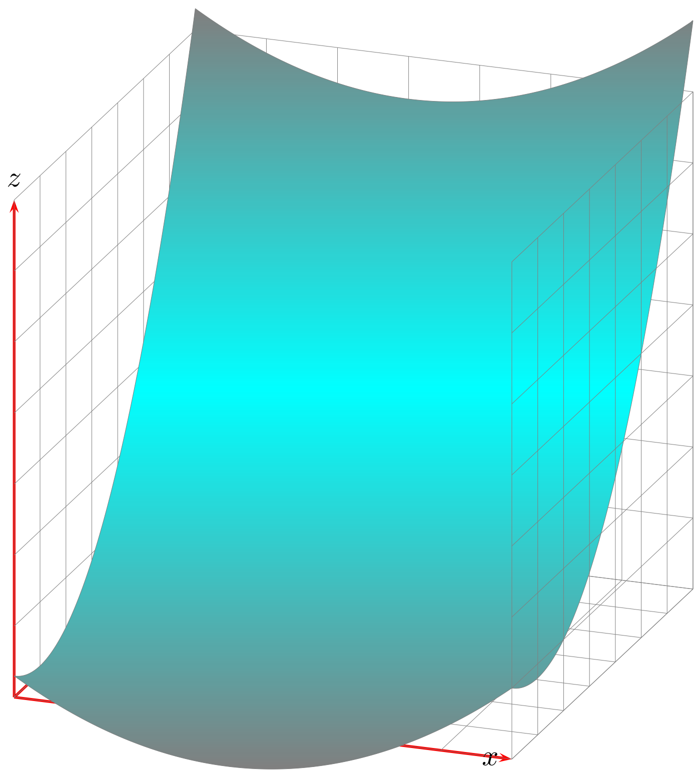




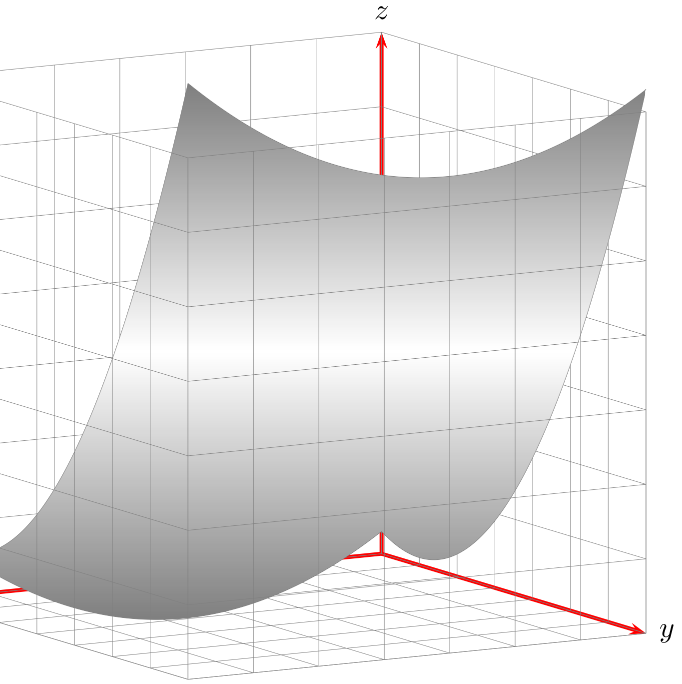

```
\begin{pspicture}(-4,-3.5)(5,4)
  \pstThreeDCoor[xMin=0,yMin=0,zMin=0,
      linewidth=2pt]
  \psset{linewidth=0.1pt,linecolor=lightgray}
  \pstThreeDPlaneGrid(0,0)(4,4)
  \pstThreeDPlaneGrid[planeGrid=xz](0,0)(4,4)
  \pstThreeDPlaneGrid[planeGrid=yz](0,0)(4,4)
\end{pspicture}
```

```
\begin{pspicture}(-3,-3.5)(5,4)
  \psset{coorType=2}% set it globally!
  \pstThreeDCoor[xMin=0,yMin=0,zMin=0,
      linewidth=2pt]
  \psset{linewidth=0.1pt,linecolor=lightgray}
  \pstThreeDPlaneGrid(0,0)(4,4)
  \pstThreeDPlaneGrid[planeGrid=xz](0,0)(4,4)
  \pstThreeDPlaneGrid[planeGrid=yz](0,0)(4,4)
\end{pspicture}
```

1 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=...}




```
\begin{pspicture}(-1,-2)(10,10)
  \psset{Beta=20,Alpha=160,subticks=7}
  \pstThreeDCoor[xMin=0,yMin=0,zMin=0,xMax=7,yMax=7,zMax=7,linewidth=1pt]
  \psset{linewidth=0.1pt,linecolor=gray}
  \pstThreeDPlaneGrid(0,0)(7,7)
  \pstThreeDPlaneGrid[planeGrid=xz,planeGridOffset=7](0,0)(7,7)
  \pstThreeDPlaneGrid[planeGrid=yz](0,0)(7,7)
  \pscustom[linewidth=0.1pt,fillstyle=gradient,gradbegin=gray,gradmidpoint=0.5,plotstyle=curve]{%
    \psset{xPlotpoints=200,yPlotpoints=1}
    \psplotThreeD(0,7)(0,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }
    \psset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
    \psplotThreeD(7,7)(0,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }
    \psset{xPlotpoints=200,yPlotpoints=1,drawStyle=xLines}
    \psplotThreeD(7,0)(7,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }
    \psset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
    \psplotThreeD(0,0)(7,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }}
  \pstThreeDPlaneGrid[planeGrid=yz,planeGridOffset=7](0,0)(7,7)
\end{pspicture}
```




```
n{pspicture}(-6,-2)(4,7)
set{Beta=10,Alpha=30,subticks=7}
tThreeDCoor[xMin=0,yMin=0,zMin=0,xMax=7,yMax=7,zMax=7,linewidth=1.5pt]
set{linewidth=0.1pt,linecolor=gray}
tThreeDPlaneGrid(0,0)(7,7)
tThreeDPlaneGrid[planeGrid=xz](0,0)(7,7)
tThreeDPlaneGrid[planeGrid=yz](0,0)(7,7)
custom[linewidth=0.1pt,fillstyle=gradient,gradbegin=gray,gradend=white,gradmidpoint=0.5,
otstyle=curve]{%
sset{xPlotpoints=200,yPlotpoints=1}
splotThreeD(0,7)(0,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }
sset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
splotThreeD(7,7)(0,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }
sset{xPlotpoints=200,yPlotpoints=1,drawStyle=xLines}
splotThreeD(7,0)(7,7){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }
sset{xPlotpoints=1,yPlotpoints=200,drawStyle=yLines}
splotThreeD(0,0)(7,0){ x dup mul y dup mul 2 mul add x 6 mul sub y 4 mul sub 3 add 10 div }}
tThreeDPlaneGrid[planeGrid=xz,planeGridOffset=7](0,0)(7,7)
tThreeDPlaneGrid[planeGrid=yz,planeGridOffset=7](0,0)(7,7)
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:

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

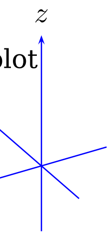
for the placing of the text or other objects.

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 similiar to the `\rput` macro:

`\pstThreeDPut[Options](`*x,y,z*`){`*any stuff*`}`




```
\begin{pspicture}(-2,-1.25)(1,2.25)
 \psset{Alpha=-60,Beta=30}
 \pstThreeDCoor[linecolor=blue,%
   xMin=-1,xMax=2,yMin=-1,yMax=2,zMin=-1,zMax=2]
 \pstThreeDPut(1,0.5,1.25){pst-3dplot}
 \pstThreeDDot[drawCoor=true](1,0.5,1.25)
\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 od the coordinate system, the 3D dot must not be seen as the center of the printed stuff.

6.2 pstPlanePut[2]

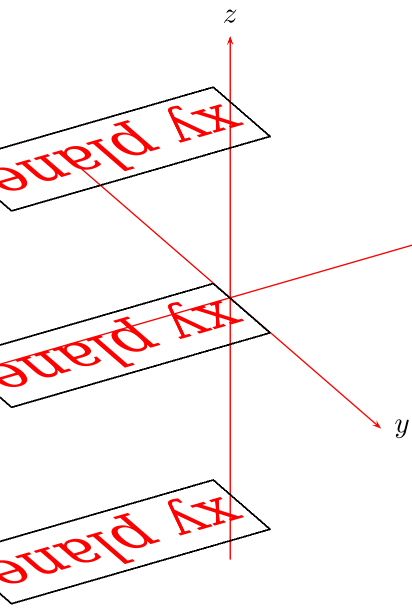
The syntax of the `\pstPlanePut` is

`\pstPlanePut[Options](`*x,y,z*`){`*Object*`}`

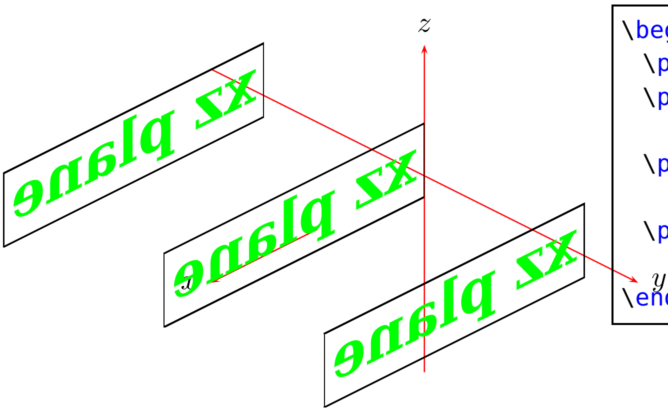
We have two special parameters, `plane` and `planecorr`; both are optional. Let's start with the first parameter, `plane`. Possible values for the two dimensional plane are `xy`, `xz`, and `yz`. If this parameter is missing then `plane=xy` is set. The first letter marks the positive direction for the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

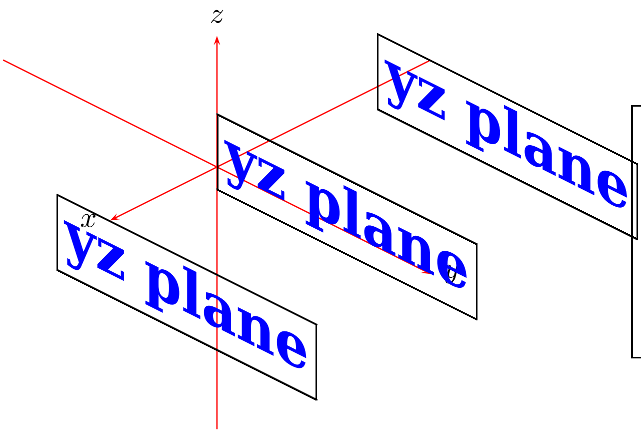
2 Thanks to Torsten Suhling




```
\begin{pspicture}(-4,-4)(3,4)
  \psset{Alpha=30}
  \pstThreeDCoor[xMin=-4,yMin=-4,zMin=-4]
  \pstPlanePut[plane=xy](0,0,-3){\fbox{\Huge\red xy plane}}
  \pstPlanePut[plane=xy](0,0,0){\fbox{\Huge\red xy plane}}
  \pstPlanePut[plane=xy](0,0,3){\fbox{\Huge\red xy plane}}
\end{pspicture}
```




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




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

The following examples use the `pOrigin` 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.

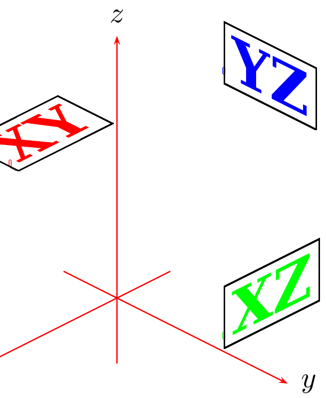
`normal` Default correction, planes will be rotated to be readable.

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

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

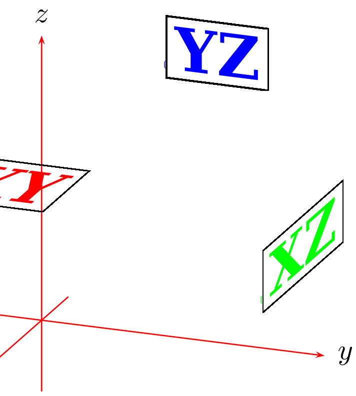
`\pstPlanePut` puts the labels on the plane of it's value. That means, `plane=xy` puts the label on the xy plane, so that the x marks the positive direction for the width, the y for the height and the label XY plane on the top side of plane. If you see the label mirrored, you just look from the bottom 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{pOrigin=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 usefull to rotate the label for xy-plane, so that body line of letters is parallel to the y axis. It's done by setting `planecorr=xyrot` :




```
\begin{pspicture}(-2,-2)(4,4)
 \psset{pOrigin=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}`

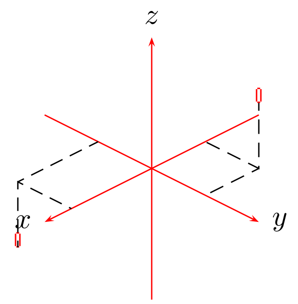
This node is internally a two dimensional node, so it cannot be used as a replacement for the parameters `(x,y,z)` of a 3D dot, which is possible with the `\psline` macro from `pst-plot`: `\psline{A}{B}`, where `A` and `B` are two nodes. It is still on the to do list, that it may also be possible with `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

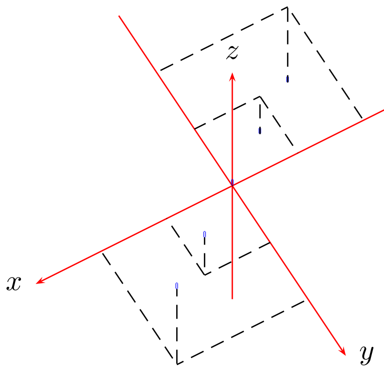
`\pstThreeDDot[Options](x,y,z)`

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)
  \pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
  \psset{dotstyle=*,dotscale=2,linecolor=red,drawCoor=true}
  \pstThreeDDot(-1,1,1)
  \pstThreeDDot(1.5,-1,-1)
\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)
  \psset{Alpha=30,Beta=60,dotstyle=square*,dotsize=3pt,%
    linecolor=blue,drawCoor=true}
  \pstThreeDCoor[xMin=-3,xMax=3,yMin=-3,yMax=3,zMin=-3,zMax=3]
  \multido{\n=-2+1}{5}{\pstThreeDDot(\n,\n,\n)}
\end{pspicture}
```

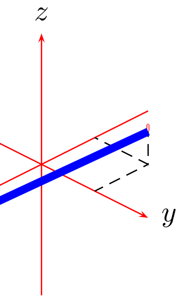
9 Lines

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

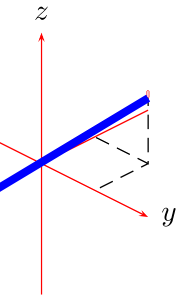
`\pstThreeDLine[Options][<arrow>](x1,y1,z1)(...)(xn,yn,zn)`

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 "'->"'.

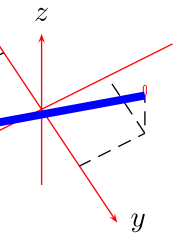
There is no special polygon macro, because you can get nearly the same with `\pstThreeDLine`.



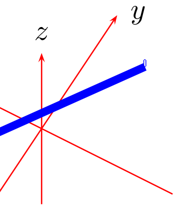

```
\begin{pspicture}(-2,-2.25)(2,2.25)
  \pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
  \psset{dotstyle=*,linecolor=red,drawCoor=true}
  \pstThreeDDot(-1,1,0.5)
  \pstThreeDDot(1.5,-1,-1)
  \pstThreeDLine[linewidth=3pt,linecolor=blue,arrows=->]%
    (-1,1,0.5)(1.5,-1,-1)
\end{pspicture}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



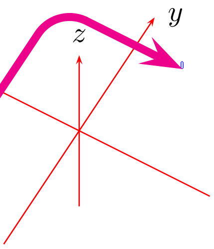

```
\begin{pspicture}(-2,-2.25)(2,2.25)
  \pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
  \psset{dotstyle=*,linecolor=red,drawCoor=true}
  \pstThreeDDot(-1,1,1)
  \pstThreeDDot(1.5,-1,-1)
  \pstThreeDLine[linewidth=3pt,linecolor=blue](-1,1,1)(1.5,-1,-1)
\end{pspicture}
```



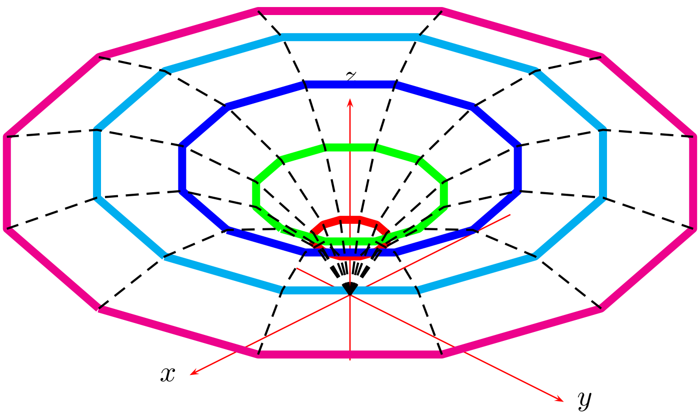

```
\begin{pspicture}(-2,-2.25)(2,2.25)
  \psset{Alpha=30,Beta=60,dotstyle=pentagon*,dotsize=5pt,%
    linecolor=red,drawCoor=true}
  \pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
  \pstThreeDDot(-1,1,1)
  \pstThreeDDot(1.5,-1,-1)
  \pstThreeDLine[linewidth=3pt,linecolor=blue](-1,1,1)(1.5,-1,-1)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




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




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



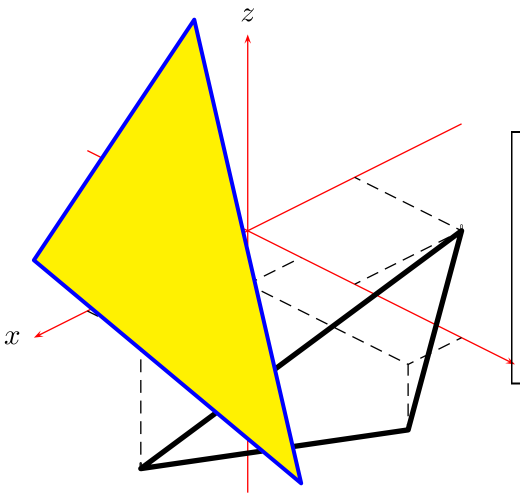

```
\begin{pspicture}(-3,-2)(4,5)\label{lines}
  \pstThreeDCoor[xMin=-3,xMax=3,yMin=-1,yMax=4,zMin=-1,zMax=3]
  \multido{\iA=1+1,\iB=60+-10}{5}{%
    \ifcase\iA\or\psset{linecolor=red}\or\psset{linecolor=green}
      \or\psset{linecolor=blue}\or\psset{linecolor=cyan}
      \or\psset{linecolor=magenta}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

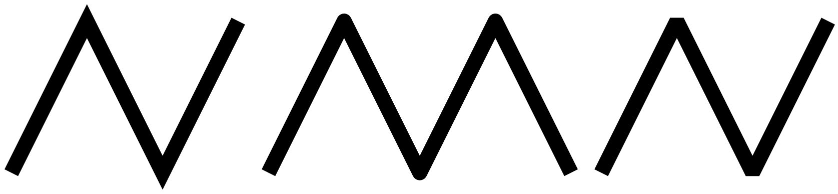
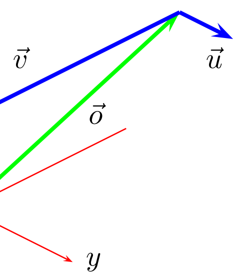


Figure 4: The meaning of the option `linejoin=0|1|2` for drawing lines

11 Squares

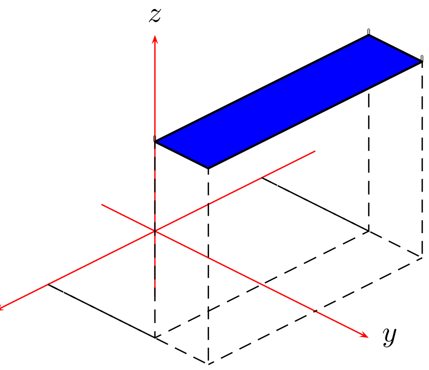
The syntax for a 3D square is:

`\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.



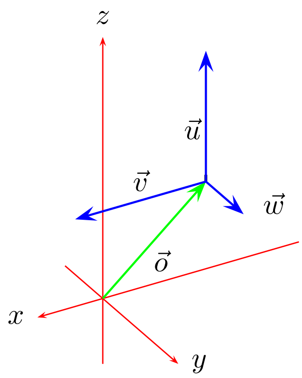

```
\begin{pspicture}(-3,-2)(4,3)
  \pstThreeDCoor[xMin=-3,xMax=3,yMin=-1,yMax=4,zMin=-1,zMax=3]
  {\psset{fillcolor=blue,fillstyle=solid,drawCoor=true,dotstyle=*}
   \pstThreeDSquare(-2,2,3)(4,0,0)(0,1,0)}
\end{pspicture}
```

12 Boxes

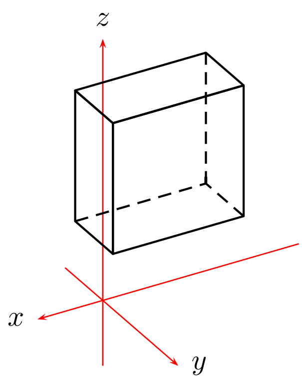
A box is a special case of a square and has the syntax

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

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




```
\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]
  \pstThreeDDot[drawCoor=true](-1,1,2)
  \psset{arrows=->,arrowsize=0.2}
  \pstThreeDLine[linecolor=green](0,0,0)(-1,1,2)
  \uput[0](0.5,0.5){$\vec{o}$}
  \uput[0](0.9,2.25){$\vec{u}$}
  \uput[90](0.5,1.25){$\vec{v}$}
  \uput[45](2,1.){$\vec{w}$}
  \pstThreeDLine[linecolor=blue](-1,1,2)(-1,1,4)
  \pstThreeDLine[linecolor=blue](-1,1,2)(1,1,2)
  \pstThreeDLine[linecolor=blue](-1,1,2)(-1,2,2)
\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)(2,0,0)(0,1,0)(0,0,2)
  \pstThreeDDot[drawCoor=true](-1,1,2)
\end{pspicture}
```

Hidden lines are only possible if you view the object from the front and not from behind.

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

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


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