3D plots: pst-3dplot

A PSTricks package for drawing 3d objects, v2.08
## Contents

1. **The Parallel projection**  
   - 2 Options  
   - 3 Coordinates and Axes  
     - 3.1 Ticks, comma and labels  
     - 3.2 Offset  
     - 3.3 Experimental features  
   - 4 Rotation  
   - 5 Plane Grids  

2. **Put**  
   - 6.1 \texttt{\textbackslash{}pstThreeDPut}  
   - 6.2 \texttt{\textbackslash{}pstPlanePut}  

3. **Nodes**  

4. **Dots**  

5. **Lines**  

6. **Triangles**  

7. **Squares**  

8. **Boxes**  

9. **Ellipses and circles**  
   - 13.1 Options  
   - 13.2 Ellipse  
   - 13.3 Circle  

10. **\texttt{\textbackslash{}pstIIIDCylinder}**  

11. **\texttt{\textbackslash{}psCylinder}**  

12. **\texttt{\textbackslash{}pstParaboloid}**  

13. **Spheres**  

14. **Mathematical functions**  
   - 18.1 Function $f(x, y)$  
   - 18.2 Parametric Plots  

15. **Plotting data files**  
   - 19.1 \texttt{\textbackslash{}fileplotThreeD}  
   - 19.2 \texttt{\textbackslash{}dataplotThreeD}  
   - 19.3 \texttt{\textbackslash{}listplotThreeD}
The well known \texttt{pstricks} package offers excellent macros to insert more or less complex graphics into a document. \texttt{pstricks} itself is the base for several other additional packages, which are mostly named \texttt{pst-xxxx}, like \texttt{pst-3dplot}. There exist several packages for plotting three dimensional graphical objects. \texttt{pst-3dplot} is similar to the \texttt{pst-plot} package for two dimensional objects and mathematical functions.

This version uses the extended keyval package \texttt{xkeyval}, so be sure that you have installed this package together with the special one \texttt{pst-xkey} for PSTricks. The \texttt{xkeyval} package is available at \texttt{CTAN:/macros/latex/contrib/xkeyval/}. It is also important that after \texttt{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 \((x_E, y_E)\).

\[
\begin{align*}
\alpha & : \text{horizontal rotating angle} \\
\beta & : \text{vertical rotating angle}
\end{align*}
\]

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.

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

\[
x^* = -x \cdot \cos \alpha + y \cdot \sin \alpha
\]  

(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° horizontally to the left and vertically by \( \beta \) also to the left.

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 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 \( \beta \), so we have them to multiply with \( \sin \beta \) to get the vertical part. We get the following transformation equations:

\[
\begin{align*}
x_E &= -x \cos \alpha + y \sin \alpha \\
y_E &= -(x \sin \alpha + y \cos \alpha) \cdot \sin \beta + z \cos \beta
\end{align*}
\]  

or written in matrix form:

\[
\begin{pmatrix} x_E \\ y_E \end{pmatrix} = \begin{pmatrix} -\cos \alpha & \sin \alpha & 0 \\ -\sin \alpha \sin \beta - \cos \alpha \sin \beta \cos \beta & -\cos \alpha \sin \beta - \cos \alpha \sin \beta \cos \beta \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}
\]

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

## 2 Options

All options which are set with \texttt{\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 \texttt{Alpha} and \texttt{Beta}. Mostly all macro need these values, this is the reason why they should be set with \texttt{\psset} and not part of an optional argument.

## 3 Coordinates and Axes

\texttt{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 \texttt{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 \texttt{!}. For example, assuming \texttt{\def\nA{2}}, \((1,0,2)\) and \((90 \cos, 100 100 \text{sub}, \nA \space 2 \text{div} 1 \text{add})\) specify the same point. (Recall that a \texttt{\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 

\begin{verbatim}
\pstThreeDCoor [Options]
\end{verbatim}

The only special option is \texttt{drawing=true|false}, which enables the drawing of the coordinate axes. The default is true. In nearly all cases the \texttt{\pstThreeDCoor} macro must be part of any drawing to initialize the 3d-system. If \texttt{drawing} is set to \texttt{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>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th>Default</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha</td>
<td>&lt;angle&gt;</td>
<td>45</td>
<td>8</td>
</tr>
<tr>
<td>Beta</td>
<td>&lt;angle&gt;</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>xMin</td>
<td>&lt;value&gt;</td>
<td>-1</td>
<td>10</td>
</tr>
<tr>
<td>xMax</td>
<td>&lt;value&gt;</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>yMin</td>
<td>&lt;value&gt;</td>
<td>-1</td>
<td>10</td>
</tr>
<tr>
<td>yMax</td>
<td>&lt;value&gt;</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>zMin</td>
<td>&lt;value&gt;</td>
<td>-1</td>
<td>10</td>
</tr>
<tr>
<td>zMax</td>
<td>&lt;value&gt;</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>nameX</td>
<td>&lt;string&gt;</td>
<td>$x$</td>
<td>70</td>
</tr>
<tr>
<td>spotX</td>
<td>&lt;angle&gt;</td>
<td>180</td>
<td>70</td>
</tr>
<tr>
<td>nameY</td>
<td>&lt;string&gt;</td>
<td>$y$</td>
<td>70</td>
</tr>
<tr>
<td>spotY</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>70</td>
</tr>
<tr>
<td>nameZ</td>
<td>&lt;string&gt;</td>
<td>$z$</td>
<td>70</td>
</tr>
<tr>
<td>spotZ</td>
<td>&lt;angle&gt;</td>
<td>90</td>
<td>70</td>
</tr>
<tr>
<td>IIIIDticks</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>IIIIDlabels</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>Dx</td>
<td>&lt;value&gt;</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Dy</td>
<td>&lt;value&gt;</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>Dz</td>
<td>&lt;value&gt;</td>
<td>1</td>
<td>11</td>
</tr>
<tr>
<td>IIIIDxTicksPlane</td>
<td>xy</td>
<td>xz</td>
<td>yz</td>
</tr>
<tr>
<td>IIIIDyTicksPlane</td>
<td>xy</td>
<td>xz</td>
<td>yz</td>
</tr>
<tr>
<td>IIIIDzTicksPlane</td>
<td>xy</td>
<td>xz</td>
<td>yz</td>
</tr>
<tr>
<td>IIIIDticks</td>
<td>&lt;value&gt;</td>
<td>0.1</td>
<td>10</td>
</tr>
<tr>
<td>IIIIDxticksep</td>
<td>&lt;value&gt;</td>
<td>-0.4</td>
<td>11</td>
</tr>
<tr>
<td>IIIIDyticksep</td>
<td>&lt;value&gt;</td>
<td>-0.2</td>
<td>11</td>
</tr>
<tr>
<td>IIIIDzticksep</td>
<td>&lt;value&gt;</td>
<td>0.2</td>
<td>11</td>
</tr>
<tr>
<td>RotX</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>RotY</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>RotZ</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>RotAngle</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>xRotVec</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>yRotVec</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>zRotVec</td>
<td>&lt;angle&gt;</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>RotSequence</td>
<td>xyz</td>
<td>xzy</td>
<td>yxz</td>
</tr>
<tr>
<td>RotSet</td>
<td>set</td>
<td>concat</td>
<td>keep</td>
</tr>
<tr>
<td>eulerRotation</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>IIIIDOffset</td>
<td>{x,y,z}</td>
<td>{0,0,0}</td>
<td>13</td>
</tr>
<tr>
<td>zlabelFactor</td>
<td>&lt;text&gt;</td>
<td>\relax</td>
<td>13</td>
</tr>
<tr>
<td>comma</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>
There are no restrictions for the angles and the max and min values for the axes; all \texttt{pstricks} options are possible as well. The following example changes the color and the width of the axes.

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

\begin{verbatim}
\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}
\end{verbatim}

\begin{verbatim}
\begin{pspicture}(-2,-2)(2,2)
\pstThreeDCoor[xMax=2, yMax=2, zMax=2]
\end{pspicture}
\end{verbatim}
\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}
3.1 Ticks, comma and labels

With the option \texttt{IIIDticks} the axes get ticks and with \texttt{IIIDlabels} labels. Without ticks also labels are not possible. The optional argument \texttt{comma}, which is defined in the package \texttt{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

\begin{verbatim}
def\psxyzlabel#1{\bgroup\footnotesize\textsf{#1}\egroup}
\end{verbatim}

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

\begin{verbatim}
\begin{pspicture}(-3,-2.5)(3,4)
pstThreeDCoor[IIIDticks,IIIDticksize=0.05]%
pstThreeDPut(3,0,3){\Huge default}
\end{pspicture}
\end{verbatim}
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}
For the z axis it is possible to define a factor for the values, e.g.

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

3.2 Offset

The optional argument IIIIDOffset 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:
3.3 Experimental features

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

With \texttt{coorType}=1 the y–z-axes are orthogonal and the angle between x- and y-axis is Alpha. The angle Beta is not valid.

With \texttt{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.

With $\text{coorType}=3$ 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.

$\text{coorType}=4$ 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.
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.

For `coorType=6` the $x$-axis is shortened by 0.559.
For `coorType=6` the $x$-axis is shortened by 0.5.

\begin{verbatim}
\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,lincolor=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}
\end{verbatim}

For `coorType=7` the $x$-axis is shortened 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}}
\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=zyx, RotX=90,RotY=90, RotZ=90,]
  linecolor=blue\(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, RotX=90, RotY=90, RotZ=90,]
  linecolor=red\(0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\end{pspicture}

\begin{pspicture}{-2,-1.5}(2,2.5)
\pstThreeDCoor[xMin=0,xMax=2,yMin=0,yMax=2,zMin=0,zMax=2]
\pstThreeDBox(0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\pstThreeDBox[RotX=90, linecolor=red\(0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\pstThreeDBox[RotX=90, RotY=90, linecolor=green\(0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\pstThreeDBox[RotX=90, RotY=90, RotZ=90, linecolor=blue\(0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
\end{pspicture}
It is sometimes more convenient to rotate the coordinate system by specifying a single angle of rotation $\text{RotAngle}$ (in degrees) about a vector whose coordinates are $\text{xRotVec}$, $\text{yRotVec}$, and $\text{zRotVec}$ using the quaternion option for $\text{RotSequence}$.

\begin{verbatim}
\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)
\put(0){(-2.28,1.2){$\vec{R}_\Phi$}}
\end{pspicture}
\end{verbatim}

Rotations of the coordinate system may be “accumulated” by applying successive rotation sequences using the RotSet variable, which is set either as a \texttt{pst-3dplot} object’s optional argument, or with a $\backslash$psset\{\texttt{pst-3dplot}\}{$\backslash$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 \texttt{pst-3dplot} object, or if not previously set within the object’s scope by a $\backslash$psset\{\texttt{pst-3dplot}\}{$\backslash$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 $\texttt{pstThreeDCoor}$ calls are made, with or without actual plotting of the axes, to accumulate rotations. A previous value of RotSet=\texttt{set} must have been made!

- keep: Keeps the current rotation matrix, ignoring the rotation parameters. Mostly used internally to eliminate redundant calculations.
By default, the rotations defined by $\text{RotX}$, $\text{RotY}$, and $\text{RotZ}$ 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 \texttt{pst-3dplot} variable option \texttt{eulerRotation} can be set to \texttt{true} to activate Euler angle definitions; i.e., \texttt{eulerRotation=true}. The default is \texttt{eulerRotation=false}. 

\begin{pspicture}(-3,-3)(3.6,3)
\pstThreeDCoor[linecolor=blue, RotSequence=quaternion, RotAngle=0, RotSet=set, xRotVec=0, yRotVec=0, zRotVec=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, yRotVec=0, 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=1, 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}
\begin{pspicture}(-4,-5)(6,5)
\pstThreeDCoor[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}
\begin{pspicture}(3,-3)(7,7)
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]
\end{pspicture}
\begin{pspicture}(3,-3)(7,7)
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)
\psset{linewidth=2pt}
\pstThreeDGrid[xMin=0,yMin=0,zMin=0,linewidth=0.1pt,linecolor=lightgray]
\end{pspicture}

\begin{pspicture}(-3,-3.5)(5,4)
\psset{coorType=2}% set it globally!
\pstThreeDGrid[xMin=0,yMin=0,zMin=0,linewidth=2pt]
\end{pspicture}

\(^1\) This option is also defined in the package `pstricks-add`, so it is necessary to set this option locally or with the family option of `pst-xkey`, e.g. `\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(7,7)(7,0){ 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(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}
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:

\pstart\|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:

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

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

The syntax of the \pstPlanePut is

```latex
\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.

---

2 Thanks to Torsten Suhling
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.

\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\textbf{xy plane}}
\pstPlanePut[plane=xy](0,0,0){fbox\Huge\red\textbf{xy plane}}
\pstPlanePut[plane=xy](0,0,3){fbox\Huge\red\textbf{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 plane}}
\pstPlanePut[plane=xz](0,0,0){fbox\Huge\green\textbf{xz plane}}
\pstPlanePut[plane=xz](0,3,0){fbox\Huge\green\textbf{xz plane}}
\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)
\psset{planecorr=normal}
\pstPlanePut[plane=xy](1,-1,2)
\psset{linecolor=red}
\fbox{Huge\textbf{XY}}
\pstThreeDDot[drawCoor=true,linecolor=green](1,3,1)
\psset{planecorr=normal}
\pstPlanePut[plane=xz](1,3,1)
\psset{linecolor=green}
\fbox{Huge\textbf{XZ}}
\pstThreeDDot[drawCoor=true,linecolor=blue](-1.5,0.5,3)
\psset{planecorr=normal}
\pstPlanePut[plane=yz](-1.5,0.5,3)
\psset{linecolor=blue}
\fbox{Huge\textbf{YZ}}
\end{pspicture}

But, why we have a third value \texttt{xyrot} of \texttt{planecorr}? If there isn’t an symmetrical view, – just like in this example – it could be usefull to rotate the label for \textit{xy}-plane, so that body line of letters is parallel to the \textit{y} axis. It’s done by setting \texttt{planecorr=xyrot}:
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 \texttt{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 \texttt{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 \texttt{drawCoor} is set to \texttt{true}. It is also possible to draw an unseen dot with the option \texttt{dotstyle=none}. In this case the macro draws only the coordinates when the \texttt{drawCoor} option is set to \texttt{true}. 

\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)
\psPlanePut[plane=xy,planecorr=xyrot](1,-1,2)
\psPlanePut[plane=xz,planecorr=xyrot](1,3.5,1)
\psPlanePut[plane=yz,planecorr=xyrot](-2,1,3)
\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}(-2,-2)(2,2)
\pstThreeDCoor[xMin=-2,xMax=2,yMin=-2,yMax=2,zMin=-2,zMax=2]
\psset{dotstyle=*,dotscale=2,linestyle=red,dotstyle=*,dotsize=3}
\pstThreeDDot(-1,1,1)
\pstThreeDDot(1.5,-1,-1)
\end{pspicture}

9 Lines

The syntax for a three dimensional line is just like the same from \texttt{\textbackslash psl ine}
\begin{verbatim}
\psline[Options] [\textbackslash arrow] (x1,y1,z1)(...)(xn,yn zn)
\end{verbatim}

The option and arrow part are both optional and the number of points is only limited to the memory. All options for lines from \texttt{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 "", and the second of "".

There is no special polygon macro, because you can get nearly the same with \texttt{\textbackslash p sThreeDLine}. 

\begin{pspicture}(-3,-3.25)(2,3.25)
\psset{Alpha=30,Beta=60,dotstyle=square*,dotsize=3pt,}
\psset{linecolor=blue,dotstyle=square*,dotsize=3pt,}
\psset{\textbackslash Alpha=30,\textbackslash Beta=60,\textbackslash dotstyle=square*,\textbackslash dotsize=3pt,}
\psset{\textbackslash linecolor=blue,\textbackslash dotstyle=square*,\textbackslash dotsize=3pt,}
\pstThreeDCoor[xMin=-3,xMax=3,yMin=-3,yMax=3,zMin=-3,zMax=3]
\multido{\textbackslash n=-2+1}{5}{\psThreeDDot(\textbackslash n,\textbackslash n,\textbackslash n)}
\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(-1,1,0.5)
\pstThreeDDot(1.5,-1,-1)
\pstThreeDLine[linewidth=3pt,linecolor=blue,arrows=->](1.5,-1,-1)
\end{pspicture}
\begin{pspicture}(-2,-2.25)(2,2.25)
\psset{Alpha=30,Beta=60,dotstyle=pentagon*,dots=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[SphericalCoor=true, 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}
  \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,linestyle=black,\%
linewidth=2pt](3,1,-2)(1,4,-1)(-2,2,0)
\pstThreeDTriangle[fillcolor=yellow,fillstyle=solid,\%
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.

\begin{figure}
\centering
\special{ps: /Title {The meaning of the option linejoin=0|1|2 for drawing lines}}
\pspicture[linewidth=2pt](10,5)
\psline(1,0)(9,0)
\psline(1,5)(9,5)
\psline(5,9)(1,9)
\psline(5,1)(9,1)
\psline(5,3)(9,3)
\psline(5,7)(9,7)
\end{pspicture}
\caption{The meaning of the option linejoin=0|1|2 for drawing lines}
\end{figure}

11 Squares

The syntax for a 3D square is:
\pstThreeDSquare[Options](vector o)(vec u)(vec v)
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.

A box is a special case of a square and has the syntax
\[ \text{\texttt{\textbackslash{pspicture}}} (-3,-2)(4,3) \]
\[ \text{\texttt{\textbackslash{pstThreeDCoor}}(xMin=-3, xMax=3, yMin=-1, yMax=4, zMin=-1, zMax=3)} \]
\[ \text{\texttt{\textbackslash{psset}}} \{\text{fillcolor=blue, fillstyle=solid, drawCoor=true, dotstyle=*}\} \]
\[ \text{\texttt{\textbackslash{pstThreeDSquare}}} (-2,2,3)(4,0,0)(0,1,0) \]
\[ \text{\texttt{\textbackslash{end{pspicture}}}} \]

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.
Hidden lines are only possible if you view the object from the front and not from behind.
\texttt{\texttt{psBox}[Options](vector \ o\{width\}{depth\}{height})}

The origin vector $\vec{o}$ determines the left corner of the box.
\begin{pspicture}(-3,-4)(3,2)
\psset{Beta=40}
pstThreeDcoor[zMax=3]
\psBox[RotY=20,showInside=false](0,0,0){2}{5}{3}
\end{pspicture}

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

\psset{Beta=10,xyzLight=-7 3 4}
\begin{pspicture}{-3,-2}(3,3)
\psset{Alpha=200}
\pstThreeDCoor[zMax=3]
\psBox(0,0,0){2}{2}{3}
\end{pspicture}
13 Ellipses and circles

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

\[
ed : \frac{(x - x_M)^2}{a^2} + \frac{(y - y_M)^2}{b^2} = 1
\]  

(4)

(x_m; y_m) is the center, a and b the semi major and semi minor axes respectively and e the eccentricity. 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 parameter form is

\[
x = a \cdot \cos \alpha
\]
\[
y = b \cdot \sin \alpha
\]

(5)

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

\[
ed : \vec{x} = \vec{m} + \cos \alpha \cdot \vec{u} + \sin \alpha \cdot \vec{v} \quad 0 \leq \alpha \leq 360
\]

(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):

\begin{align*}
\text{beginAngle} & = 0 \\
\text{endAngle} & = 360
\end{align*}

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:

\begin{verbatim}
\pstThreeDEllipse[Options] (cx,cy,cz)(ux,uy,uz)(vx,vy,vz)
\end{verbatim}

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.

```latex
\begin{pspicture}(-3,-2)(3,3)
\pstThreeDCoor[IIIDticks]
\psset{arrowscale=2,arrows=>}
\pstThreeDLine(0,0,0)(3,0,0)\pstThreeDLine(0,0,0)(0,3,0)\pstThreeDLine(0,0,0)(0,0,3)
\psset{linestyle=dashed,linestyle=->}
\pstThreeDEllipse(0,0,0)(3,0,0)(0,3,0) \pstThreeDEllipse(0,0,0)(0,3,0)(3,0,0) \pstThreeDEllipse(0,0,0)(0,0,3)(0,3,0)
\end{pspicture}
```

```latex
\begin{pspicture}(-3,-2)(3,3)
\pstThreeDCoor[IIIDticks]
\psset{arrowscale=2,arrows=>}
\pstThreeDLine(0,0,0)(3,0,0)\pstThreeDLine(0,0,0)(0,3,0)\pstThreeDLine(0,0,0)(0,0,3)
\psset{linestyle=dashed,linestyle=->}
\pstThreeDEllipse(0,0,0)(3,0,0)(0,3,0) \pstThreeDEllipse(0,0,0)(0,3,0)(3,0,0) \pstThreeDEllipse(0,0,0)(0,0,3)(0,3,0)
\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.

\[
\text{\texttt{\textbackslash pstThreeDCircle[Options]} (cx,cy,cz)(ux,uy,uz)(vx,vy,vz)}
\]
\def\radius{4} \def\PhiI{20} \def\PhiII{50} \\
\def\RadIs{\radius \PhiI \sin \mul} \\
\def\RadIc{\radius \PhiI \cos \mul} \\
\def\RadIIc{\radius \PhiII \sin \mul} \\
\def\RadIIs{\radius \PhiII \cos \mul} \\
\begin{pspicture}(-4,-4)(4,5) \\
\psset{\Alpha=45,\Beta=30,\linestyle=dashed} \\
\pstThreeDCoor[\linestyle=solid,\xMin=-5,\xMax=5,\yMax=5,\zMax=5,IIIIDticks] \\
\pstThreeDEllipse[\linecolor=red](0,0,0)(0,\rad\PhiI,0)(0,0,\rad) \\
\pstThreeDEllipse(\RadIs,0,0)(0,\RadIc,0)(0,0,\RadIc) \\
\pstThreeDEllipse[\linestyle=dotted,\SphericalCoor,\beginAngle=-90,\endAngle=90](0,0,0)(\rad,90,\PhiI)(\rad,0,0) \\
\pstThreeDEllipse[\SphericalCoor,\beginAngle=90,\endAngle=90](0,0,0)(\rad,90,\PhiII)(\rad,0,0) \\
\begin{pspicture}(-4,-4)(4,5) \\
\psset{\linecolor=blue,\arrows=->,\arrowscale=2,\linewidth=1.5pt,\linestyle=solid} \\
\pstThreeDEllipse[\SphericalCoor,\beginAngle=\PhiII,\endAngle=\PhiII]\(\rad,0,0)\(0,\rad,0) \\
\pstThreeDEllipse[\beginAngle=\PhiII,\endAngle=\PhiII]\(\RadIIs,0,0)\(0,0,\RadIIc) \\
\pstThreeDEllipse[\SphericalCoor,\beginAngle=\PhiII,\endAngle=\PhiII]\(\RadIc,0,0)\(0,0,\rad) \\
\pstThreeDEllipse[\beginAngle=\PhiII,\endAngle=\PhiII]\(\rad,0,0)\(0,\rad,0) \\
\end{pspicture} \\
\begin{pspicture}(-4,-4)(4,5) \\
[ ... ]
\pstThreeDEllipse[linestyle=dotted,SphericalCoor](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
\pstThreeDEllipse[SphericalCoor,
beginAngle=-90,endAngle=90](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
\pstThreeDEllipse[linestyle=dotted,SphericalCoor](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
\pstThreeDEllipse[SphericalCoor,
beginAngle=-90,endAngle=90](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
% \pscustom[fillstyle=solid,fillcolor=blue]{
  \pstThreeDEllipse[SphericalCoor,beginAngle=\PhiII,endAngle=\PhiII](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
  \pstThreeDEllipse[beginAngle=\PhiII,endAngle=\PhiII](\RadIIIs,0,0)(0,\RadIic,0)(0,0,\RadIic)
  \pstThreeDEllipse[SphericalCoor,beginAngle=\PhiII,endAngle=\PhiII](0,0,0)(\radius,90,\PhiII)(\radius,0,0)
  \pstThreeDEllipse[beginAngle=\PhiII,endAngle=\PhiII](\RadIIis,0,0)(0,\RadIic,0)(0,0,\RadIic)
}\end{pspicture}

14 \pstIIIIDCylinder

The syntax is
\pstIIIIDCylinder[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.
\psframebox{
\begin{pspicture}(-3.5,-2)(3,6.75)
\pstThreeDCoor[zMax=7]
\pstIIDCylinder[RotY=30,fillstyle=solid,
    fillcolor=red!20, linecolor=black!60](0,0,0){2}{5}
\end{pspicture}
}
\psframebox{%
\begin{pspicture}(-3.2,-1.75)(3,6.25)
\pstThreeDCoor[zMax=7]
\pstIIIDCylinder[linecolor=black!20, increment=0.4, fillstyle=solid]{2}{5}
\psset[linecolor=red]
\pstThreeDLine{->}(0,0,5)(0,0,7)
\end{pspicture}
}
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. With increment for the angle step and $H$increment for the height step, the number of segments can be defined. They are preset to 10 and 0.5.
\begin{pspicture}(-3,-2)(3,2)
\psset{Beta=10}
\pstThreeDcoor[zMax=7]
\psCylinder[increment=5]{2}{0.5}
\end{pspicture}
\begin{pspicture}(-3,-2)(3,6) \psset{Beta=60} \pstThreeDCoor[zMax=9] \psCylinder[RotX=10, increment=5]{3}{5} \pstThreeDLine[linecolor=red](0,0,0)(0,0,8.5) \end{pspicture}
\begin{pspicture}(-3,-2)(3,6)
\psset{Beta=60}
\pstThreeDCoor[zMax=9]
\psCylinder[RotY=-45](0,1,0){2}{5}
\end{pspicture}
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>
<thead>
<tr>
<th>Option name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SegmentColor</td>
<td>cmyk color for the segments (0.2,0.6,1,0)</td>
</tr>
<tr>
<td>showInside</td>
<td>show inside (true)</td>
</tr>
<tr>
<td>increment</td>
<td>number for the segments (10)</td>
</tr>
</tbody>
</table>

\begin{pspicture}(-2,-1)(2,5)
\pstThreeDCoor[xMax=2,yMax=2,zMin=0,zMax=6,IIIDticks]%
\pstParaboloid{5}{1}% Hoehe 5 und Radius 1
\end{pspicture}
\begin{pspicture}(0,-3)(7,5)
\pstThreeDCoor[xMax=2,yMax=13,zMin=0,zMax=6,IIIIDticks]
\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>
<thead>
<tr>
<th>Option name</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SegmentColor</td>
<td>cmyk color for the segments $(0.2,0.6,1,0)$</td>
</tr>
<tr>
<td>increment</td>
<td>number for the segments $(10)$</td>
</tr>
</tbody>
</table>
18 Mathematical functions

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

18.1 Function $f(x, y)$

The macro for plotting functions does not have the same syntax as the one from \texttt{pst-plot} \cite{6}, but it is used in the same way:

\begin{verbatim}
\psplotThreeD[\textbf{Options}](xMin,xMax)(yMin,yMax){the function}
\end{verbatim}

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 \texttt{psplot}, except the \texttt{plotpoints}-option which is split into one for $x$ and one for $y$ (table \ref{table:plotstyles}).

The equation \ref{eq:7} is plotted with the following parameters and seen in figure \ref{fig:6}.

\begin{equation}
  z = 10 \left( x^3 + xy^4 - \frac{x}{9} \right) e^{-(x^2+y^2)} + e^{-((x-1.225)^2+y^2)}
\end{equation}

The function is calculated within two loops:

\begin{verbatim}
for (float y=yMin; y<yMax; y+=dy)
  for (float x=xMin; x<xMax; x+=dx)
    z=f(x,y);
\end{verbatim}

It depends on the inner loop in which direction the curves are drawn. There are four possible values for the option \texttt{drawStyle}:

- \texttt{xLines} (default) Curves are drawn in x direction
- \texttt{yLines} Curves are drawn in y direction
### 18.2 Parametric Plots

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

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

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
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
18.2 Parametric Plots

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

curve and not an area.

\[
\begin{align*}
  x &= f(t, u) \\
  y &= f(t, u) \\
  z &= f(t, u)
\end{align*}
\] (8)

To draw a spiral we have the parametric functions:

\[
\begin{align*}
  x &= r \cos t \\
  y &= r \sin t \\
  z &= t/600
\end{align*}
\] (9)

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 \(0^\circ \ldots 2160^\circ\). 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

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

which is the same as \((0,0)\) for the parameter \(u\).
And the same with the algebraic option:

Instead of using the \texttt{\texttt{pstThreeDSphere}} macro (see section \texttt{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 need the parameter equations as

$$\begin{align*}
x &= \cos t \cdot \sin u \\
y &= \cos t \cdot \cos u \\
z &= \sin t
\end{align*}$$

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

$$\begin{align*}
x &= \cos u \cdot \sin t \\
y &= \cos u \cdot \cos t \\
z &= \sin u
\end{align*}$$

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

\begin{verbatim}
\begin{pspicture}(-1,-1)(1,1)
\parametricplotThreeD[plotstyle=curve,yPlotpoints=40](0,360)(0,360){
  t cos u sin mul t cos u cos mul t sin
}
\parametricplotThreeD[plotstyle=curve,yPlotpoints=40](0,360)(0,360){
  u cos t sin mul u cos t cos mul u sin
}
\end{pspicture}
\end{verbatim}

\textbf{Figure 13:} Different Views of the same Parametric Functions

\section*{19 Plotting data files}

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

\begin{verbatim}
0.000  1.000  0.000
-0.4207  0.9972  0.0191
....
0.000,  1.000,  0.000
-0.4207,  0.9972,  0.0191
....
\end{verbatim}
There are the same three plot functions:

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

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.

\subsection{19.1 \fileplotThreeD}

The syntax is very easy

\begin{verbatim}
\fileplotThreeD[Options] {datafile}
\end{verbatim}

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.

\subsection{19.2 \dataplotThreeD}

The syntax is

\begin{verbatim}
\dataplotThreeD[Options] {data object}
\end{verbatim}

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[4]

\begin{verbatim}
\readdata{data object}{datafile}
\end{verbatim}

\subsection{19.3 \listplotThreeD}

The syntax is

\begin{verbatim}
\listplotThreeD[Options] {data object}
\end{verbatim}

\listplotThreeD ist similiar 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[4] \listplot can handle some additional PostScript code, which can be appended to the data object, f.ex.:
\begin{pspicture}(-6,-3)(6,10)
\psset{xunit=0.5cm,yunit=0.75cm,Alpha=30,Beta=30}\% the global parameters
\pstThreeDCoor[xMin=-10,xMax=10,yMin=-10,yMax=10,zMin=-2,zMax=10]
\fileplotThreeD{plotstyle=line}{data3D.Roessler}
\end{pspicture}\%

**Figure 14:** Demonstration of \fileplotThreeD with Alpha=30 and Beta=15
\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 \texttt{\dataplotThreeD} with $\text{Alpha}=-30$ and $\text{Beta}=30$
Figure 16: Demonstration of `\listplotThreeD` with a view from above (\(\text{Alpha}=0\) and \(\text{Beta}=90\)) and some additional PostScript code

Figure 16 shows what happens with this code. For another example see [6], where the macro `\ScalePoints` is modified. This macro is in `\opt{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
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.
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:
\texttt{\textbackslash getThreeDCoor(vector)} \texttt{\textbackslash macro} \mbox{\textbackslash macro} \textbackslash macro \textbackslash holds \textbackslash the \textbackslash sequence \textbackslash of \textbackslash the \textbackslash three \textbackslash coordinates \textbackslash x \ y \ z, \textbackslash divided \textbackslash by \textbackslash a \textbackslash space.}

\section*{20.3 Adding two vectors}

The syntax is \texttt{\textbackslash pstaddThreeDVec(vector A)(vector B)} \texttt{\tempa\tempb\tempc} \texttt{\tempa\tempb\tempc} \texttt{\tempa\tempb\tempc} \texttt{\tempa\tempb\tempc} \texttt{must \textbackslash be \textbackslash user \textbackslash or \textbackslash system \textbackslash defined \textbackslash macros, \textbackslash which \textbackslash holds \textbackslash the \textbackslash three \textbackslash coordinates \textbackslash of \textbackslash the \textbackslash vector \textbackslash \textbf{\vec{C}} = \textbf{\vec{A}} + \textbf{\vec{B}}.}

\section*{20.4 Substract two vectors}

The syntax is \texttt{\textbackslash pstsubThreeDVec(vector A)(vector B)} \texttt{\tempa\tempb\tempc} \texttt{\tempa\tempb\tempc} \texttt{\tempa\tempb\tempc} \texttt{\tempa\tempb\tempc} \texttt{must \textbackslash be \textbackslash user \textbackslash or \textbackslash system \textbackslash defined \textbackslash macros, \textbackslash which \textbackslash holds \textbackslash the \textbackslash three \textbackslash coordinates \textbackslash of \textbackslash the \textbackslash vector \textbackslash \textbf{\vec{C}} = \textbf{\vec{A}} - \textbf{\vec{B}}.}
## List of all optional arguments for pst-3dplot

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>alternative</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>drawing</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>drawCoor</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>hiddenLine</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>SphericalCoor</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>IIIDshowgrid</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>CoorCheck</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>CylindricalCoor</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>leftHanded</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>eulerRotation</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>coorType</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>SphericalCoorType</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>xMin</td>
<td>ordinary</td>
<td>-1</td>
</tr>
<tr>
<td>xMax</td>
<td>ordinary</td>
<td>4</td>
</tr>
<tr>
<td>yMin</td>
<td>ordinary</td>
<td>-1</td>
</tr>
<tr>
<td>yMax</td>
<td>ordinary</td>
<td>4</td>
</tr>
<tr>
<td>zMin</td>
<td>ordinary</td>
<td>-1</td>
</tr>
<tr>
<td>zMax</td>
<td>ordinary</td>
<td>4</td>
</tr>
<tr>
<td>xThreeDunit</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>yThreeDunit</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>zThreeDunit</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>xRotVec</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>yRotVec</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>zRotVec</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>deltax</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>deltay</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>deltaz</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>DeltaX</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>DeltaY</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>DeltaZ</td>
<td>ordinary</td>
<td>1.0</td>
</tr>
<tr>
<td>Alpha</td>
<td>ordinary</td>
<td>45</td>
</tr>
<tr>
<td>Beta</td>
<td>ordinary</td>
<td>30</td>
</tr>
<tr>
<td>RotX</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>RotY</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>RotZ</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>RotAngle</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>RotSequence</td>
<td>ordinary</td>
<td>xyz</td>
</tr>
<tr>
<td>RotSet</td>
<td>ordinary</td>
<td>set</td>
</tr>
<tr>
<td>PlaneSequence</td>
<td>ordinary</td>
<td></td>
</tr>
<tr>
<td>zCoor</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>drawStyle</td>
<td>ordinary</td>
<td>xLines</td>
</tr>
<tr>
<td>xPlotpoints</td>
<td>ordinary</td>
<td>25</td>
</tr>
<tr>
<td>yPlotpoints</td>
<td>ordinary</td>
<td>25</td>
</tr>
</tbody>
</table>

Continued on next page
Continued from previous page

<table>
<thead>
<tr>
<th>Key</th>
<th>Type</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>beginAngle</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>endAngle</td>
<td>ordinary</td>
<td>360</td>
</tr>
<tr>
<td>plane</td>
<td>ordinary</td>
<td>xy</td>
</tr>
<tr>
<td>pOrigin</td>
<td>ordinary</td>
<td>c</td>
</tr>
<tr>
<td>IIIDdAlpha</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>visibleLineStyle</td>
<td>ordinary</td>
<td>solid</td>
</tr>
<tr>
<td>invisibleLineStyle</td>
<td>ordinary</td>
<td>dashed</td>
</tr>
<tr>
<td>IIIDticks</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>IIIDlabels</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>Dz</td>
<td>ordinary</td>
<td>1</td>
</tr>
<tr>
<td>IIIDxTicksPlane</td>
<td>ordinary</td>
<td>xy</td>
</tr>
<tr>
<td>IIIDyTicksPlane</td>
<td>ordinary</td>
<td>yz</td>
</tr>
<tr>
<td>IIIDzTicksPlane</td>
<td>ordinary</td>
<td>yz</td>
</tr>
<tr>
<td>IIIDticksize</td>
<td>ordinary</td>
<td>0.1</td>
</tr>
<tr>
<td>IIIDxticksep</td>
<td>ordinary</td>
<td>-0.2</td>
</tr>
<tr>
<td>IIIDyticksep</td>
<td>ordinary</td>
<td>-0.2</td>
</tr>
<tr>
<td>IIIDzticksep</td>
<td>ordinary</td>
<td>0.2</td>
</tr>
<tr>
<td>nameX</td>
<td>ordinary</td>
<td>$x$</td>
</tr>
<tr>
<td>spotX</td>
<td>ordinary</td>
<td>180</td>
</tr>
<tr>
<td>nameY</td>
<td>ordinary</td>
<td>$y$</td>
</tr>
<tr>
<td>spotY</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>nameZ</td>
<td>ordinary</td>
<td>$z$</td>
</tr>
<tr>
<td>spotZ</td>
<td>ordinary</td>
<td>90</td>
</tr>
<tr>
<td>planecorr</td>
<td>ordinary</td>
<td>none</td>
</tr>
<tr>
<td>planeGrid</td>
<td>ordinary</td>
<td>xy</td>
</tr>
<tr>
<td>planeGridOffset</td>
<td>ordinary</td>
<td>0</td>
</tr>
<tr>
<td>showInside</td>
<td>boolean</td>
<td>true</td>
</tr>
<tr>
<td>SegmentColor</td>
<td>ordinary</td>
<td>[none]</td>
</tr>
<tr>
<td>increment</td>
<td>ordinary</td>
<td>10</td>
</tr>
<tr>
<td>Hincrement</td>
<td>ordinary</td>
<td>0.5</td>
</tr>
<tr>
<td>xyzLight</td>
<td>ordinary</td>
<td>1 1 2</td>
</tr>
<tr>
<td>IIIDOffset</td>
<td>ordinary</td>
<td>[none]</td>
</tr>
<tr>
<td>zlabelFactor</td>
<td>ordinary</td>
<td>\relax</td>
</tr>
<tr>
<td>height</td>
<td>ordinary</td>
<td>5</td>
</tr>
<tr>
<td>move</td>
<td>ordinary</td>
<td>0 0</td>
</tr>
<tr>
<td>stepFactor</td>
<td>ordinary</td>
<td>0.67</td>
</tr>
</tbody>
</table>

References


Index

A
Alpha, 6–8, 14, 15, 30

B
Beta, 6–8, 14–16, 30

C
c, 28
comma, 7, 10
concat, 21
coordinates, 32
coorType, 14–17

D\dataplotThreeD, 66
\dataThreeD, 66
dots, 59
dotstyle, 32
drawCoor, 32
drawing, 6
drawStyle, 58
Dx, 7
Dy, 7
Dz, 7

E
Environment
- pspicture, 65
eulerRotation, 7, 22

F\fileplotThreeD, 66, 67
fillcolor, 37
fillstyle, 37

G\getThreeDCoor, 72

H
hiddenLine, 59
Hincrement, 50

I
IIIDlabels, 7, 10
IIIDOffset, 7, 13
IIIDticks, 7, 10
IIIDticksize, 7
IIIDxticksep, 7
IIIDxTicksPlane, 7
IIIDyTicksep, 7
IIIDyTicksPlane, 7
IIIDzTicksep, 7
IIIDzTicksPlane, 7
increment, 50

K
keep, 21
Keyvalue
- c, 28
- concat, 21
- dots, 59
- increment, 50
- keep, 21
- lB, 28
- line, 59
- none, 37
- normal, 30
- off, 30
- quaternion, 21
- RotSet, 21
- set, 21
- xLines, 58
- xy, 25, 28
- xyLines, 59
- xyrot, 12, 30, 31
- xz, 25, 28
- yLines, 58
- yxLines, 59
- yz, 15, 16, 25, 28
Keyword
- Alpha, 6–8, 14, 15, 30
- Beta, 6–8, 14–16, 30
- comma, 7, 10
- coorType, 14–17
- dotstyle, 32
- drawCoor, 32
- drawing, 6
- drawStyle, 58
- Dx, 7
- Dy, 7
- Dz, 7
- eulerRotation, 7, 22
- fillcolor, 37
- fillstyle, 37

76
- \space, 6

N
nameX, 7
nameY, 7
nameZ, 7
none, 32, 37
normal, 30

O
off, 30

P
Package
- \texttt{\textbackslash pst-3dplot}, 4, 6, 21, 32, 69, 71
- \texttt{\textbackslash pst-plot}, 4, 10, 32, 58
- \texttt{\textbackslash pst-xkey}, 4
- \texttt{\textbackslash pstricks}, 4, 28
- \texttt{xcolor}, 54, 57
- \texttt{xkeyval}, 4
\texttt{\textbackslash parametricplotThreeD}, 59, 63
plane, 28, 30
plane\texttt{corr}, 28, 30, 31
plotpoints, 58
p\texttt{Origin}, 28, 30
\texttt{\textbackslash psBox}, 40
\texttt{\textbackslash psCylinder}, 50
\texttt{\textbackslash psline}, 32
pspicture, 65
\texttt{\textbackslash psplot}, 58
\texttt{\textbackslash psplotThreeD}, 58, 59
\texttt{\textbackslash psset}, 6, 25
\texttt{\textbackslash pst-3dplot}, 4, 6, 21, 32, 69, 71
\texttt{\textbackslash pst-plot}, 4, 10, 32, 58
\texttt{\textbackslash pst-xkey}, 4
\texttt{\textbackslash pstaddThreeDVec}, 72
\texttt{\textbackslash pstIIDDcylinder}, 47
\texttt{\textbackslash pstParaboloid}, 54
\texttt{\textbackslash pstPlanePut}, 28, 30
p\texttt{stricks}, 4, 28
\texttt{\textbackslash pstRotPointIIDD}, 70
\texttt{\textbackslash p\texttt{ts}subThreeDVec}, 72
\texttt{\textbackslash pstThreeDBox}, 38
\texttt{\textbackslash pstThreeDCircle}, 45
\texttt{\textbackslash pstThreeDCoor}, 6, 7, 21
\texttt{\textbackslash pstThreeDDot}, 32
\texttt{\textbackslash pstThreeDEllipse}, 44
\texttt{\textbackslash pstThreeDLine}, 33
\texttt{\textbackslash pstThreeDNode}, 32
\texttt{\textbackslash pstThreeDPlaneGrid}, 25
\texttt{\textbackslash pstThreeDPut}, 10, 28
\texttt{\textbackslash pstThreeDput}, 54
\texttt{\textbackslash pstThreeDSphere}, 57, 64
\texttt{\textbackslash pstThreeDSquare}, 37
\texttt{\textbackslash pstThreeDTriangle}, 37

Q
quaternion, 21

R
\texttt{\textbackslash readdata}, 66
RotAngle, 7, 21
RotSequence, 7, 21
rotSequence, 19
RotSet, 7, 21
RotX, 7, 19, 69
RotY, 7, 19, 69
RotZ, 7, 19, 69
\texttt{\textbackslash rput}, 28

S
\texttt{\textbackslash ScalePoints}, 69
\texttt{\textbackslash ScalePointsThreeD}, 69
\texttt{\textbackslash SegmentColor}, 57
\texttt{\textbackslash set}, 21
\texttt{\space}, 6
Spherical\texttt{Coor}, 6
\texttt{\textbackslash spotX}, 7
\texttt{\textbackslash spotY}, 7
\texttt{\textbackslash spotZ}, 7
\texttt{\textbackslash square}, 38
\texttt{\textbackslash subticks}, 25

V
Value
- \texttt{\textbackslash line}, 66
- \texttt{\textbackslash none}, 32
- \texttt{\textbackslash set}, 21
- \texttt{\textbackslash xy}, 28, 30
- \texttt{\textbackslash xyrot}, 31
\texttt{\textbackslash vector}, 38

X
\texttt{\textbackslash xcolor}, 54, 57
\texttt{\textbackslash xkeyval}, 4
\texttt{\textbackslash xLines}, 58
\texttt{\textbackslash xMax}, 7
\texttt{\textbackslash xMin}, 7
\texttt{\textbackslash xPlotpoints}, 59
xRotVec, 7, 21
xsupticks, 25
xy, 25, 28, 30
xyLines, 59
xyrot, 12, 30, 31
xz, 25, 28

Y
yLines, 58
yMax, 7
yMin, 7
yPlotpoints, 59
yRotVec, 7, 21
ysupticks, 25
yxLines, 59
yz, 15, 16, 25, 28

Z
zlabelFactor, 7, 13
zMax, 7
zMin, 7
zRotVec, 7, 21