The 'pst-gr3d' package
A PSTricks package for three dimensional grids

Denis Girou*

Version 1.34
2005-09-12
Documentation revised September 12, 2005 by Herbert Voss

Abstract

This package allow to draw three dimensional grids using the macro `\PstGridThreeD`. We can also specify how nodes of the grid must look like.

Contents

*CNRS/IDRIS — Centre National de la Recherche Scientifique / Institut du Développement et des Ressources en Informatique Scientifique — Orsay — France — <Denis.Girou@idris.fr>.

1 Introduction

'pst-gr3d' offer a main unique macro with few parameters to interact on it. But we can also use all the relevant PSTricks parameters to change the size, the characteristics of lines, etc.

The syntax is simply: `\PstGridThreeD[optional_parameters](X,Y,Z)`

We can define a macro `\PstGridThreeDHookNode` to specify how the nodes at the interconnections must look like, and there are also some other *hooks* that can be used for special purposes.

The default viewpoint is (`1.2,-0.6,0.8`), but this can of course be changed using the standard way.

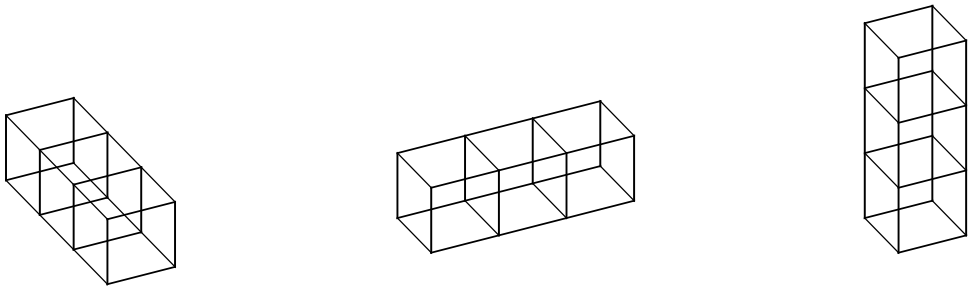
The package try to compute approximatively the size of the object (the `pspicture` parameter, PSTricks speaking), but for three dimensional grids it is an impossible task to found it accurately in the general case. So, if the exact size is needed or if we change the viewpoint for the graphic, the size must be computed *by hand*, using the `\psframebox[framesep=0]{...}` construction to found the correct values by attempts and errors — fortunately, in practice few attempts are often enough...

2 Usage

2.1 Parameters and hooks

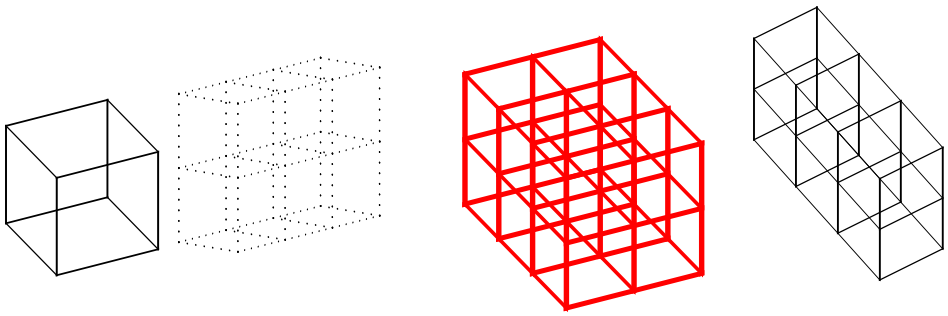
The three required parameters specify the lengths in the X, Y and Z directions, respectively:

```
1 \PstGridThreeD(3,1,1)\hfill
2 \PstGridThreeD(1,3,1)\hfill
3 \PstGridThreeD(1,1,3)
```



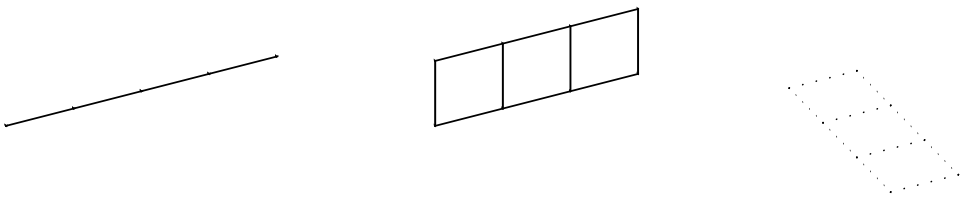
Of course, we can use all the relevant generic PSTricks parameters, specially those applying to grids:

```
\PstGridThreeD[unit=1.5](1,1,1)\hfill
\PstGridThreeD[viewpoint=1.2 -1.5 0.4,griddots=7](1,3,2)\hfill
\PstGridThreeD[gridwidth=0.08,gridcolor=red](3,2,2)\hfill
\begin{pspicture}(-1.7,0)(0.8,3.6)
  \PstGridThreeD[viewpoint=-0.4 -0.6 0.8,PstPicture=false](1,3,2)
\end{pspicture}
```



We can draw one and two dimensional grids, using degenerated cases:

```
\PstGridThreeD(0,4,0)\hfill
\PstGridThreeD[linewidth=0.05](0,3,1)\hfill
\PstGridThreeD[griddots=5](3,1,0)
```



To change the way the grids are drawn, we can also use **nine** specific parameters and **five** specific *hooks*:

PstDebug (integer) : to obtain some internal debugging informations — here, a framed box around the boundix box used (the `pspicture` environment) could be drawn. It can take the values 0 (no debug) or 1. (*Default: 0* — no debugging informations).

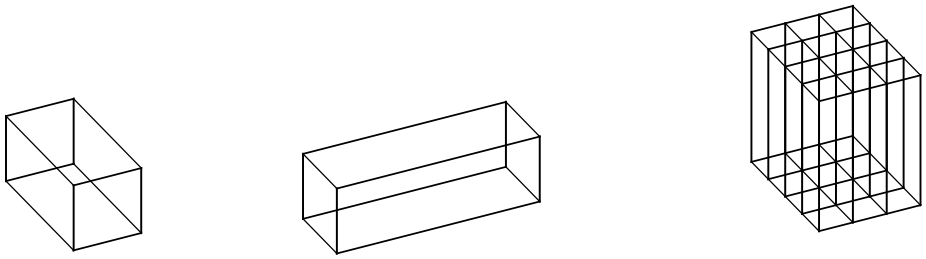
PstPicture (boolean) : to define or not a `pspicture` environment for the grid. We have to define this parameter to *false* mainly if we choose a viewpoint different than the default one — see examples later (*Default: true* — which is not the case for basic PSTricks objects).

GridThreeDXUnit (integer) : unit coefficient in the X direction (*Default: 1* — it must be an integer, not a real).

GridThreeDYUnit (integer) : unit coefficient in the Y direction (*Default: 1* — it must be an integer, not a real).

GridThreeDZUnit (integer) : unit coefficient in the Z direction (*Default: 1* — it must be an integer, not a real).

```
\PstGridThreeD[GridThreeDXUnit=2](1,1,1)\hfill
\PstGridThreeD[GridThreeDYUnit=3](1,1,1)\hfill
\PstGridThreeD[unit=0.5,GridThreeDZUnit=4](4,3,1)
```



GridThreeDXPos (integer) : position of the origin in the X direction (*Default: 0* — it must be an integer, not a real).

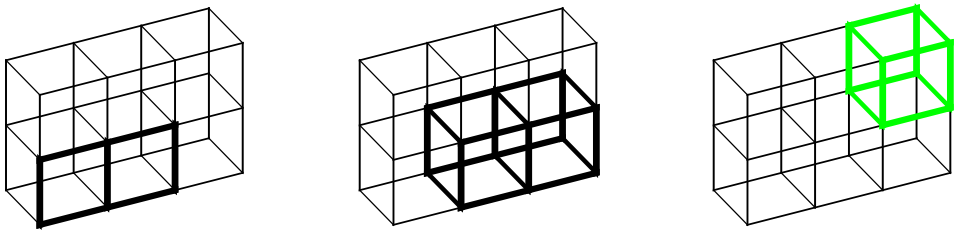
GridThreeDYPos (integer) : position of the origin in the Y direction (*Default: 0* — it must be an integer, not a real).

GridThreeDZPos (integer) : position of the origin in the Z direction (*Default: 0* — it must be an integer, not a real).

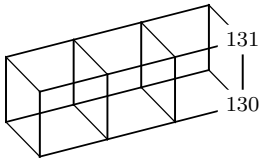
These parameters are in fact mainly useful if we want to superpose grids, which can be done easily using the `\PstGridThreeDHookEnd` macro (see description below):

```
% First grid
\def\PstGridThreeDHookEnd{%
  \PstGridThreeD[PstPicture=false,gridwidth=0.1,
                 GridThreeDXPos=1](0,2,1)}
\PstGridThreeD(1,3,2)\hfill
% Second grid
\def\PstGridThreeDHookEnd{%
  \PstGridThreeD[PstPicture=false,gridwidth=0.1,
                 GridThreeDYPos=1](1,2,1)}
\PstGridThreeD(1,3,2)\hfill
% Third grid
\def\PstGridThreeDHookEnd{%
  \PstGridThreeD[PstPicture=false,gridwidth=0.1,
```

```
                    gridcolor=green,
                    GridThreeDYPos=2,
                    GridThreeDZPos=1](1,1,1)}
\PstGridThreeD(1,3,2)
```



GridThreeDNodes (boolean) : to define or not the nodes at interconnection points of the grid. The nodes are named `Gr3dNodeXYZ`. We can use the `Rx` and `Ry` parameters to position the relevant material relatively to the nodes, specifying the distance in cartesian coordinates. The parameter `angle` used with `Rx` allow to use polar ones. (*Default: false* — no nodes defined).

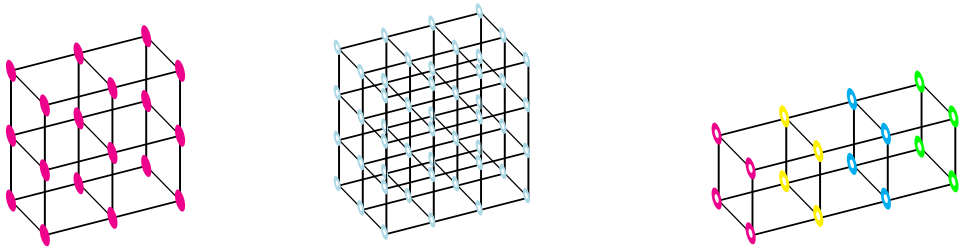



```
\PstGridThreeD[GridThreeDNodes=true](1,3,1)
\SpecialCoor
\rput*(Gr3dNode130){\footnotesize 130}
\rput*(Gr3dNode131){\footnotesize 131}
```

\PstGridThreeDHookNode (macro) : this hook allow to define the form of the nodes. A predefined `\PstGridThreeDNodeProcessor` macro exist, which define a circle with a little white circle in it. We can also use the `\iy` counter to differentiate the nodes according to the Y faces — but note that we can't do the same thing for the X or Z faces (*Default: empty*).

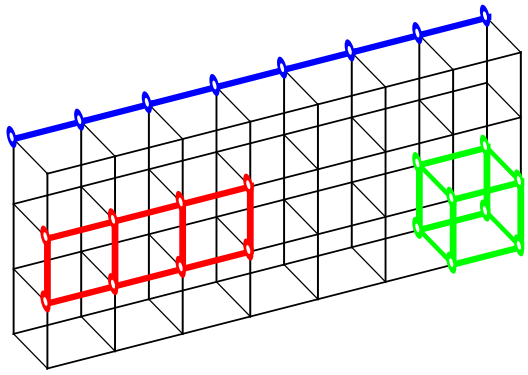
```
% First grid
\def\PstGridThreeDHookNode{%
  \begin{pspicture}(-0.15,-0.15)(0.15,0.15)
    \pscircle*[linecolor=magenta]{0.15}
  \end{pspicture}}
\PstGridThreeD(1,2,2)\hfill
% Second grid
\definecolor{LightBlue}{rgb}{0.68,0.85,0.9}
\def\PstGridThreeDHookNode{%
  \PstGridThreeDNodeProcessor{LightBlue}}
\PstGridThreeD[unit=0.7](2,3,3)\hfill
```

```
% Third grid
\def\PstGridThreeDHookNode{%
  \ifcase\iy
       \PstGridThreeDNodeProcessor{magenta}%
    \or\PstGridThreeDNodeProcessor{yellow}%
    \or\PstGridThreeDNodeProcessor{cyan}%
    \else\PstGridThreeDNodeProcessor{green}%
  \fi}
\PstGridThreeD(1,3,1)
```



\PstGridThreeDHookEnd (macro) : this hook allow to execute a macro at the end of the grid drawing, before the pspicture environment closing. This is specially interesting for instance to superpose grids, if we take care to define the PstPicture parameter to false for them (*Default: empty*).

```
\def\PstGridThreeDHookEnd{{%
  \psset{PstPicture=false,gridwidth=0.1}
  {\def\PstGridThreeDHookNode{%
     \PstGridThreeDNodeProcessor{blue}}%
   \PstGridThreeD[gridcolor=blue,
                  GridThreeDZPos=3](0,7,0)}%
  {\def\PstGridThreeDHookNode{%
     \PstGridThreeDNodeProcessor{red}}%
   \PstGridThreeD[gridcolor=red,
                  GridThreeDXPos=1,
                  GridThreeDZPos=1](0,3,1)}%
  {\def\PstGridThreeDHookNode{%
     \PstGridThreeDNodeProcessor{green}}%
   \PstGridThreeD[gridcolor=green,
                  GridThreeDYPos=6](1,1,1)}}}
\PstGridThreeD(1,7,3)
```



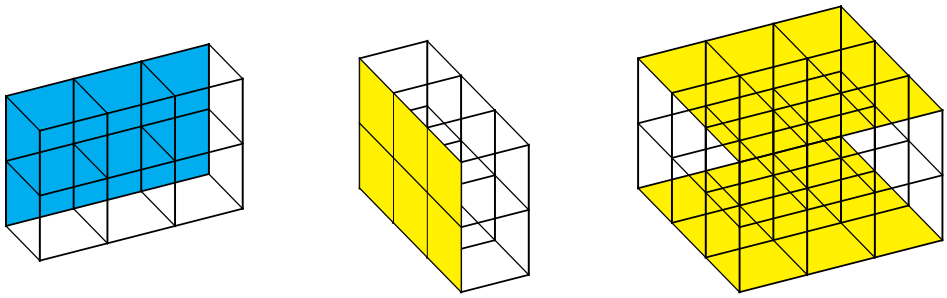
`\PstGridThreeDHookXFace` (macro) : this hook allow to execute a macro before to draw the X faces (*Default: empty*).

`\PstGridThreeDHookYFace` (macro) : this hook allow to execute a macro before to draw the Y faces (*Default: empty*).

`\PstGridThreeDHookZFace` (macro) : this hook allow to execute a macro before to draw the Z faces (*Default: empty*).

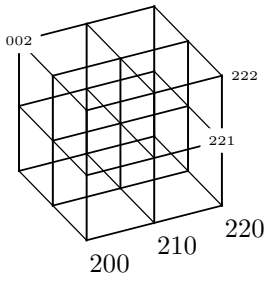
In fact, these hooks are not very powerful, because we can't control the order of the faces drawing as we can dream... For instance, we can't use this technic to draw objects with only *true* visible lines. Take care also that for the Y faces, the direction is negative in the horizontal direction, so the coordinates must take this fact in account.

```
{\def\PstGridThreeDHookXFace{%
  \ifnum\multidocount=1\psframe*[linecolor=cyan](3,2)\fi}%
 \PstGridThreeD(1,3,2)}\hfill
{\def\PstGridThreeDHookYFace{%
  \ifnum\multidocount=2\psframe*[linecolor=yellow](-3,0)(0,2)\fi}%
 \PstGridThreeD(3,1,2)}\hfill
{\def\PstGridThreeDHookZFace{%
  \ifnum\multidocount=2
  \else
    \psframe*[linecolor=yellow](3,3)
  \fi}%
 \PstGridThreeD(3,3,2)}
```

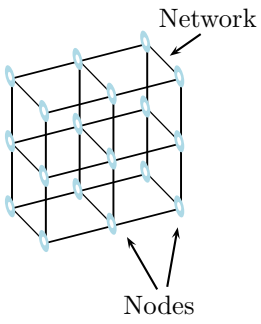


3 Examples

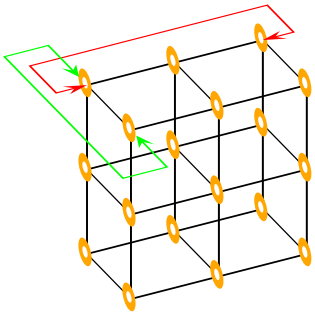
We give here more advanced examples, most of them from technical drawings describing the architecture of a multiprocessors supercomputer.




```
\PstGridThreeD[GridThreeDNodes=true](2,2,2)
\SpecialCoor
\rput*(Gr3dNode002){\tiny 002}
\rput*(Gr3dNode221){\tiny 221}
\rput([Rx=0.3]Gr3dNode222){\tiny 222}
\multido{\i=0+1}{3}{%
  \rput([Rx=0.3,Ry=-0.3]Gr3dNode2\i0){2\i0}}
```



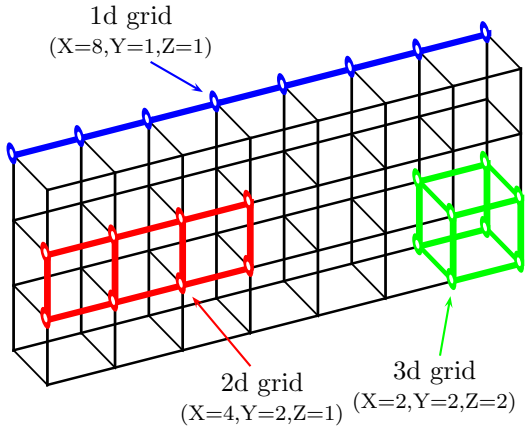

```
\definecolor{LightBlue}{rgb}{0.68,0.85,0.9}
\def\PstGridThreeDHookNode{%
  \PstGridThreeDNodeProcessor{LightBlue}}
\PstGridThreeD[GridThreeDNodes=true](1,2,2)
\SpecialCoor
\rput([Rx=-0.15,Ry=0.3]Gr3dNode122){%
  \psline{<-}(0.5;35)}
\rput([Rx=0.35,Ry=0.8]Gr3dNode122){Network}
\rput([Rx=0.15,angle=-40]Gr3dNode110){%
  \psline{<-}(0.8;-60)}
\rput([Rx=0.25,angle=-100]Gr3dNode120){%
  \psline{<-}(0.8;-100)}
\rput([Rx=1.5,angle=-55]Gr3dNode010){Nodes}
```



```
\definecolor{Orange}{rgb}{1.,0.65,0.}
\def\PstGridThreeDHookNode{%
  \PstGridThreeDNodeProcessor{Orange}}
\psset{unit=1.3}
\PstGridThreeD[GridThreeDNodes=true](1,2,2)
\SpecialCoor
\psset{arrows=<->,arrowscale=2}
\ThreeDput[normal=0 0 -1](0,0,0){%
  \ncloop[linecolor=red,arm=0.35,
          loopsize=0.6,angleA=-90,angleB=90]
         {Gr3dNode022}{Gr3dNode002}
  \ncloop[linecolor=green,arm=0.7,
          nodesepA=0.18,nodesepB=0.12,
          loopsize=-0.5,angleA=180]
         {Gr3dNode002}{Gr3dNode102}}
```

```
\def\PstGridThreeDHookEnd{{%
  \psset{PstPicture=false,gridwidth=0.1}
  {\def\PstGridThreeDHookNode{%
     \PstGridThreeDNodeProcessor{blue}}%
   \PstGridThreeD[gridcolor=blue,
                  GridThreeDZPos=3](0,7,0)}%
  {\def\PstGridThreeDHookNode{%
     \PstGridThreeDNodeProcessor{red}}%
   \PstGridThreeD[gridcolor=red,
                  GridThreeDXPos=1,
                  GridThreeDZPos=1](0,3,1)}%
  {\def\PstGridThreeDHookNode{%
     \PstGridThreeDNodeProcessor{green}}%
   \PstGridThreeD[gridcolor=green,
                  GridThreeDYPos=6](1,1,1)}}}
\PstGridThreeD[gridwidth=0.04,
               GridThreeDNodes=true](1,7,3)
\SpecialCoor
\rput([Rx=0.15,angle=140]Gr3dNode033){%
  \psline[linecolor=blue]{<-}(0.8;150)}
\rput([Rx=0.95,angle=140]Gr3dNode033){%
  \shortstack{1d grid\\\footnotesize (X=8,Y=1,Z=1)}}
\rput([Rx=0.15,angle=-50]Gr3dNode121){%
  \psline[linecolor=red]{<-}(1.2;-50)}
\rput([Rx=1.5,angle=-55]Gr3dNode121){%
  \shortstack{2d grid\\\footnotesize (X=4,Y=2,Z=1)}}
\rput([Rx=0.2,angle=-100]Gr3dNode160){%
  \psline[linecolor=green]{<-}(0.8;-100)}
\rput([Rx=1.4,angle=-100]Gr3dNode160){%
```

```
\shortstack{3d grid\\\footnotesize (X=2,Y=2,Z=2)}}
```



4 Driver file

The next bit of code contains the documentation driver file for TEX, i.e., the file that will produce the documentation you are currently reading. It will be extracted from this file by the `docstrip` program.

```
⟨*driver⟩
\documentclass{ltxdoc}
\GetFileInfo{pst-gr3d.dtx}
\usepackage{fancyvrb}
\usepackage{pstricks}
\usepackage[colorlinks,linktocpage]{hyperref}
\usepackage{pst-gr3d}
%
\AtBeginDocument{
%  \OnlyDescription % comment out for implementation details
  \EnableCrossrefs
  \CodelineIndex
  \RecordChanges}
\AtEndDocument{
  \PrintIndex
  \setcounter{IndexColumns}{1}
  \PrintChanges}
\hbadness=7000            % Over and under full box warnings
\hfuzz=3pt
\begin{document}
  \DocInput{pst-gr3d.dtx}
\end{document}
⟨/driver⟩
```

5 ‘pst-gr3d’ LaTeX wrapper

⟨∗latex − wrapper⟩
```
\RequirePackage{pstricks}
\ProvidesPackage{pst-gr3d}[2005/01/13 package wrapper for
  pst-gr3d.tex (hv)]
\input{pst-gr3d.tex}
\ProvidesFile{pst-gr3d.tex}
  [\filedate\space v\fileversion\space ‘PST-gr3d’ (dg)]
```
⟨/latex − wrapper⟩

6 ‘pst-gr3d’ code

⟨∗pst − gr3d⟩

6.1 Preambule

Who we are.

```
\def\fileversion{1.34}
\def\filedate{2005/09/12}
\message{‘PST-Grid3d’ v\fileversion, \filedate\space (Denis Girou)}
\csname PSTGridThreeDLoaded\endcsname
\let\PSTGridThreeDLoaded\endinput
```

Require the PSTricks, ‘pst-node’, ‘pst-3d’ and ‘multido’ packages.

```
\ifx\PSTricksLoaded\endinput\else\input pstricks.tex\fi
\ifx\PSTnodesLoaded\endinput\else\input pst-node.tex\fi
\ifx\PSTthreeDLoaded\endinput\else\input pst-3d.tex\fi
\ifx\MultidoLoaded\endinput\else\input multido.tex\fi
```

interface to the extended ‘keyval’ package.

```
\ifx\PSTXKeyLoaded\endinput\else\input pst-xkey\fi
%%
```

Catcodes changes and defining the family name for xkeyvalue.

```
\edef\PstAtCode{\the\catcode‘\@}\catcode‘\@=11\relax
\pst@addfams{pst-gr3d}
%%
```

6.2 Definition of the parameters

`PstDebug` is for internal debugging purposes — here, a framed box around the grid is shown (to debug, set `PstDebug=1`).

```
%% change Pst@Debug to prevent a clash with pst-fill
%% which has the same option. Now pstricks defines Pst@Debug
%%\define@key[psset]{pst-gr3d}{PstDebug}{\pst@getint{#1}{\Pst@Debug}}
%% end hv 2004-06-22
```

`PstPicture` allow to define a “`pspicture`” environment.

```
\newif\ifPst@PstPicture
\define@key[psset]{pst-gr3d}{PstPicture}[true]{\@nameuse{Pst@PstPicture#1}}
```

`GridThreeDNodes` allow to define nodes on the grid.

```
\newif\ifPstGridThreeD@Nodes
\define@key[psset]{pst-gr3d}{GridThreeDNodes}[true]{%
   \@nameuse{PstGridThreeD@Nodes#1}}
```

`GridThreeDXUnit`, `GridThreeDYUnit` and `GridThreeDZUnit` define the X, Y and Z units (must be integers).

```
\define@key[psset]{pst-gr3d}{GridThreeDXUnit}{%
  \pst@getint{#1}{\PstGridThreeD@XUnit}}
\define@key[psset]{pst-gr3d}{GridThreeDYUnit}{%
  \pst@getint{#1}{\PstGridThreeD@YUnit}}
\define@key[psset]{pst-gr3d}{GridThreeDZUnit}{%
  \pst@getint{#1}{\PstGridThreeD@ZUnit}}
```

`GridThreeDXPos`, `GridThreeDYPos` and `GridThreeDZPos` define the X, Y and Z positions.

```
\define@key[psset]{pst-gr3d}{GridThreeDXPos}{%
  \pst@getint{#1}{\PstGridThreeD@XPos}}
\define@key[psset]{pst-gr3d}{GridThreeDYPos}{%
  \pst@getint{#1}{\PstGridThreeD@YPos}}
\define@key[psset]{pst-gr3d}{GridThreeDZPos}{%
  \pst@getint{#1}{\PstGridThreeD@ZPos}}
```

`Rx` and `Ry` are aliases for relative moves from nodes.

```
\define@key[psset]{pst-gr3d}{Rx}{\psset{XnodesepA=#1}}
\define@key[psset]{pst-gr3d}{Ry}{\psset{offsetA=#1}}
```

Default values (including `viewpoint` first).

```
\psset{viewpoint=1.2 -0.6 0.8}
\psset{%
PstDebug=0,PstPicture=true,GridThreeDNodes=false,
GridThreeDXPos=0,GridThreeDYPos=0,GridThreeDZPos=0,
GridThreeDXUnit=1,GridThreeDYUnit=1,GridThreeDZUnit=1}
```

6.3 Main macro

The general `\PstGridThreeD` macro to draw three dimensional grids.

`\PstGridThreeD`

```
\def\PstGridThreeD{\@ifnextchar[{\PstGridThreeD@i}{\PstGridThreeD@i[]}}
```

`\PstGridThreeD@i`

```
\def\PstGridThreeD@i[#1](#2,#3,#4){{%
\psset{dimen=middle}%
\psset{#1}%
```

First, we try to compute rather accurate values for the "`pspicture`" environment (it is not possible to found them in the general case, so we must define our own ones, setting `PstPicture` to false before, if the ones automatically computed here are really wrong in our case...).

Ymin pspicture value.

```
\pst@cnth=#2
\multiply\pst@cnth\PstGridThreeD@XUnit
\divide\pst@cnth\tw@
\ifodd\pst@cnth
  \edef\PstGridThreeD@PictureYmin{-\the\pst@cnth}%
\else
  \edef\PstGridThreeD@PictureYmin{-\the\pst@cnth.5}%
\fi
```

Xmax pspicture value (stored in `\pst@cntg`).

```
\pst@cntg=#3
\multiply\pst@cntg\PstGridThreeD@YUnit
\pst@cnth=#2
\divide\pst@cnth\tw@
\multiply\pst@cnth\PstGridThreeD@XUnit
\advance\pst@cntg\pst@cnth
```

Ymax pspicture value.

```
\pst@cnth=#3
\advance\pst@cnth\m@ne
\multiply\pst@cnth\PstGridThreeD@YUnit
\divide\pst@cnth\tw@
\pst@cntd=#4
\multiply\pst@cntd\PstGridThreeD@ZUnit
\advance\pst@cntd\pst@cnth
\ifnum\pst@cnth=\z@
  \edef\PstGridThreeD@PictureYmax{\the\pst@cntd.5}%
\else
  \edef\PstGridThreeD@PictureYmax{\the\pst@cntd}%
\fi
```

If required, the pspicture environment.

```
\ifPst@PstPicture
```

If `PstDebug=1`, we draw a framed box around the grid.

```
  \ifnum\Pst@Debug=\@ne
    \psframebox[framesep=0]{%
  \fi
  \pspicture(0,\PstGridThreeD@PictureYmin)
            (\the\pst@cntg,\PstGridThreeD@PictureYmax)
\fi
\pst@cntd=\PstGridThreeD@XPos
\multiply\pst@cntd\PstGridThreeD@XUnit
\pst@cntg=\PstGridThreeD@YPos
\multiply\pst@cntg\PstGridThreeD@YUnit
\pst@cnth=\PstGridThreeD@ZPos
\multiply\pst@cnth\PstGridThreeD@ZUnit
```

Z faces (only if \PstGridThreeDHookZFace is defined).

```
\ifx\PstGridThreeDHookZFace\empty
\else
  \pst@cntc=#4
  \advance\pst@cntc\@ne
  \multido{\iz=#4+-\PstGridThreeD@ZUnit}{\pst@cntc}{% Z face hook
    \ThreeDput[normal=0 0 1](\pst@cntd,\pst@cntg,\iz){\PstGridThreeDHookZFace}}
\fi
```

X faces.

```
\pst@cntc=#2
\advance\pst@cntc\@ne
\multido{\ix=\pst@cntd+\PstGridThreeD@XUnit}{\pst@cntc}{%
  \ThreeDput[normal=1 0 0](\ix,\pst@cntg,\pst@cnth){%
```

with an X face hook.

```
    \PstGridThreeDHookXFace
    \psgrid[xunit=\PstGridThreeD@YUnit,yunit=\PstGridThreeD@ZUnit,
            subgriddiv=0,gridlabels=0](#3,#4)}}
```

Y faces.

```
\pst@cnta=#3
\multiply\pst@cnta\PstGridThreeD@YUnit
\advance\pst@cnta\pst@cntg
\pst@cntc=#3
\advance\pst@cntc\@ne
\multido{\iy=\pst@cnta+-\PstGridThreeD@YUnit}{\pst@cntc}{%
  \ThreeDput[normal=0 1 0](\pst@cntd,\iy,\pst@cnth){%
    \PstGridThreeDYFace{#2}{#4}{\iy}}}
```

Hook at the end, if defined.

```
\PstGridThreeD@HookEnd
```

Then we close the "pspicture" environment, if defined.

```
\ifPst@PstPicture
  \endpspicture
```

If PstDebug=1, we close of the framed box.

```
  \ifnum\Pst@Debug=\@ne
}
  \fi
```

And we close the \PstGridThreeD@ macro.

```
\fi}}
```

One face of the three dimensional grid.

\PstGridThreeDYFace

```
\def\PstGridThreeDYFace#1#2#3{%
```

Vertical faces.
First, Y face hook.

```
\PstGridThreeDHookYFace%
\psgrid[xunit=\PstGridThreeD@XUnit,yunit=\PstGridThreeD@ZUnit,
        subgriddiv=0,gridlabels=0](-#1,#2)
\pst@cnta=#1
\advance\pst@cnta\@ne
\pst@cntb=#2
\advance\pst@cntb\@ne
\pst@cntg=#3
\multido{\ia=0+-\PstGridThreeD@XUnit}{\pst@cnta}{%
  \pst@cntc=\multidocount
  \advance\pst@cntc\m@ne
  \multido{\ib=0+\PstGridThreeD@ZUnit}{\pst@cntb}{%
```

Nodes definition.

```
    \ifPstGridThreeD@Nodes
      \pst@cntd=\multidocount
      \advance\pst@cntd\m@ne
      \pnode(\ia,\ib){Gr3dNode\the\pst@cntc\the\pst@cntg\the\pst@cntd}
    \fi
    \ifx\PstGridThreeDHookNode\empty
    \else
```

Nodes drawing.

```
      \rput(\ia,\ib){\PstGridThreeDHookNode}
    \fi}}
```

6.4 Default hooks (empty)

\PstGridThreeDHookNode

```
\def\PstGridThreeDHookNode{}
```

\PstGridThreeDHookXFace

```
\def\PstGridThreeDHookXFace{}
```

\PstGridThreeDHookYFace

```
\def\PstGridThreeDHookYFace{}
```

\PstGridThreeDHookZFace

```
\def\PstGridThreeDHookZFace{}
```

For the end hook, we must avoid infinite recursion if the hook contain a \PstGridThreeD macro!

\PstGridThreeDHookEnd

```
\def\PstGridThreeDHookEnd{}
```

\PstGridThreeD@HookEnd

```
\def\PstGridThreeD@HookEnd{%
\def\PstGridThreeD@HookEnd{}%
\PstGridThreeDHookEnd}
```

Default definition of a processor node.

\PstGridThreeDNodeProcessor

```
\def\PstGridThreeDNodeProcessor#1{{%
\psset{unit=0.3}
\pspicture(-0.5,-0.5)(0.5,0.5)
  \pscircle*[linecolor=#1]{0.5}
  \pscircle*[linecolor=white]{0.2}
\endpspicture}}
```

6.5 Closing

Catcodes restoration.

```
\catcode`\@=\PstAtCode\relax
```

⟨/pst − gr3d⟩

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

M

P

S

T