

Introduzione a PSTricks

Massimo Caschili *

Sommario

PSTricks è un potente sistema grafico costituito da un ampio numero di estensioni; offre molti strumenti per produrre grafici, diagrammi e in generale figure con effetti ed una resa tipografica d'alta qualità.

Abstract

PSTricks is a powerful graphic system established by a large number of extensions; it offers many tools to produce pictures, graphical representations and figures with high-quality effects and an high-quality typographical performance.

1 Introduzione

Gli utenti di \LaTeX hanno spesso bisogno d'inserire nei loro documenti disegni, schemi e figure di varia complessità; l'ambiente standard `picture` ha molti limiti, ma nell'universo di \LaTeX ci sono tanti pacchetti che offrono ambienti adatti a produrre ottima grafica. In questo articolo presenterò uno dei sistemi più completi e potenti.

PSTricks è una collezione di macro istruzioni basate su PostScript ¹. Il pacchetto di base `pstricks` è correlato da un insieme di altri pacchetti che ne estendono le potenzialità. Non entrerà nei dettagli del codice di PSTricks, ma è importante sapere che il sistema si basa, fra i tanti, su due file principali: `pstricks.sty` e `pstricks.tex`, pertanto l'utente non dovrà chiamare i propri file con questi nomi. In queste pagine si farà una presentazione del pacchetto principale e una veloce carrellata sui tanti pacchetti che estendono il sistema, una sorta d'indice commentato. Ogni pacchetto è rilasciato con la relativa documentazione che sarà il naturale sicuro riferimento dell'utente.

2 Il primo passo

Nella figura 1 è riportato il codice minimale per creare un semplice documento, la seconda riga richiama il pacchetto principale, nella quarta riga il comando `\red` rende rosso tutto quanto è all'interno delle parentesi graffe. I pacchetti d'estensione di PSTricks sono un buon numero, e ciascuno

*Desidero ringraziare: Enrico Bini per aver ben presentato questo lavoro in mia vece, il revisore per la sua pazienza e i suoi preziosi consigli.

1. PostScript è un linguaggio di descrizione di pagina interpretato, sviluppato da Adobe Systems. Non è necessario conoscere il PostScript per usare PSTricks.

```
1 \documentclass{report}
2 \usepackage{pstricks}
3 \begin{document}
4   {\red Il primo esempio}
5 \end{document}
```

Il primo esempio

FIGURA 1: Il codice essenziale per il primo esempio.

deve essere, all'occorrenza, richiamato inserendo nel preambolo (cioè prima della riga 3) il comando `\usepackage{nomepacchetto}`. Attenzione: il pacchetto `pstricks-add` deve essere caricato per ultimo. Il codice deve essere compilato con il programma `latex`. Non può essere usato il programma `pdflatex` per produrre direttamente file in formato PDF; è comunque facile ottenere un file PDF seguendo lo schema di figura 2. Il programma `latex` elabora il file di testo `file.tex` e crea un file DVI, `dvips` converte il file DVI in formato PS e il programma `ps2pdf` converte il file PS in formato PDF.

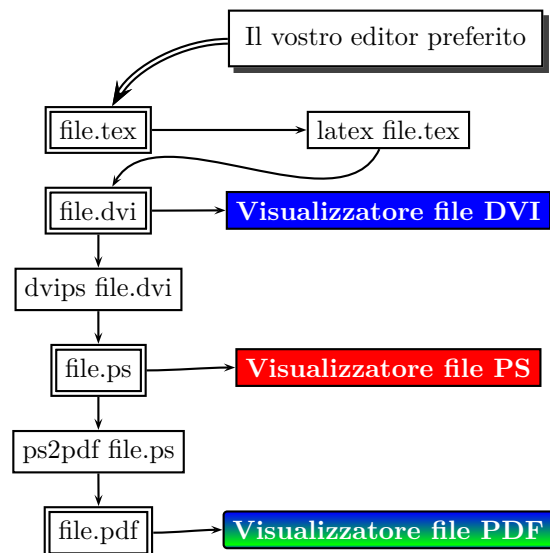


FIGURA 2: Questa figura è essa stessa un esempio delle possibilità offerte da PSTricks, il codice è nella sezione dedicata a `pst-node`.

I visualizzatori del formato DVI possono dare qualche problema con PSTricks, non si tratta di errori rilevati in fase di compilazione, ma di visualizzazione errata. Convertendo il file `*.dvi` in formato PostScript il risultato è perfetto. L'ulteriore conversione in formato PDF è ugualmente perfetta.

Chi usa la linea di comando, deve dare uno die-

tro l'altro i seguenti comandi oppure inserirli in un batch:

```
latex file.tex
dvips -D800 file.dvi
ps2pdf -sPAPERSIZE#a4 file.ps file.pdf
```

3 Le basi di pstricks

In questa sezione illustrerò le principali caratteristiche del pacchetto, i comandi di base, i parametri. I comandi di PSTricks hanno il loro argomento fra parentesi graffe, gli argomenti opzionali fra parentesi quadre. Il segno di uguale associa un valore ad un parametro valido per quel comando. Le parentesi tonde racchiudono i valori per le coordinate. PSTricks fornisce un ambiente analogo a `picture` che si chiama `pspicture`. La figura 3 illustra la sintassi ed evidenzia l'area delimitata dall'ambiente, l'uso delle coordinate permette d'inserire gli oggetti ovunque nell'area e anche fuori di essa. Gli oggetti con coordinate esterne all'area *protetta* potranno andare a sovrapporsi a qualunque elemento della pagina.² La variante con asterisco `pspicture*` evita che un oggetto oltrepassi i limiti dell'area dell'ambiente, in pratica gli oggetti saranno tagliati in corrispondenza del perimetro dell'area delimitata dall'ambiente.

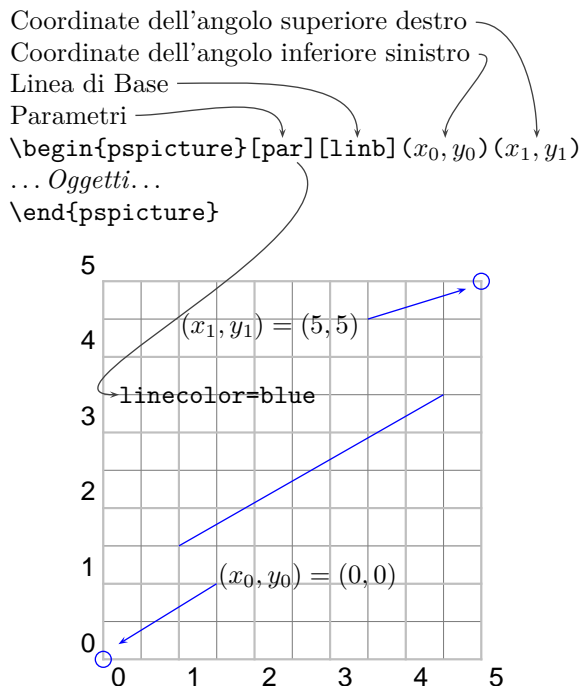


FIGURA 3: Sintassi ed esempio dell'ambiente `pspicture`. Si noti che l'indicazione del parametro `linecolor=blue` ha effetto per tutti gli oggetti che fanno uso di quel parametro. I parametri inseriti nella dichiarazione dell'ambiente hanno valore solo all'interno dell'ambiente.

La variante con asterisco, della sintassi illustrata nella figura 3, è:

2. Questi concetti di base sono comuni all'ambiente standard `picture`. Vedi bibliografia.

```
\begin{pspicture*}[par][linb](x_0, y_0)(x_1, y_1)
... Oggetti...
\end{pspicture*}
```

Nella figura 4 è illustrata la differenza di comportamento fra l'ambiente `pspicture` con e senza l'asterisco.

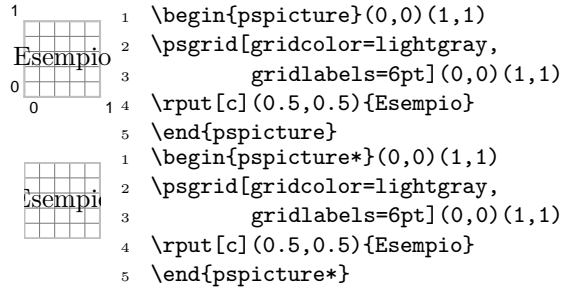


FIGURA 4: L'ambiente `pspicture` e `pspicture*`.

Per fissare la posizione degli oggetti ci sono diversi comandi, che possono essere usati anche fuori dall'ambiente `pspicture`. Ci sono due varianti, una con l'asterisco e l'altra senza. I comandi d'inserimento sono i seguenti:

```
\rput*[rif]{angolo}(x_0, y_0){ogg.}
```

Qualora un oggetto, inserito con la variante asterisco, si sovrapponga ad un altro oggetto, questo non si vedrà più in trasparenza, ma sarà completamente coperto, si veda l'esempio 5.

```
\uput*{dist}[rif]{angolo}(x_0, y_0){ogg.}
```

Questo comando permette d'inserire un oggetto, alle coordinate indicate, in modo particolare: l'oggetto è inserito, rispetto al suo punto di riferimento `rif`, a una distanza `dist` dalle coordinate (x_0, y_0) , in direzione indicata da `angolo`.

```
\cput*[par]{angolo}(x_0, y_0){ogg.}
```

Questo comando crea un cerchio, nello stile indicato in `par`, attorno all'oggetto in (x_0, y_0) , che può essere ruotato di un certo angolo.

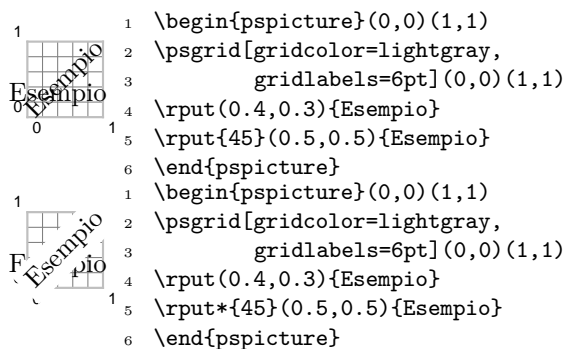


FIGURA 5: Si noti che la variante con asterisco si sovrappone, coprendo, l'oggetto sottostante.

Un qualunque oggetto può essere considerato come se fosse inscritto in un rettangolo. Il punto di

riferimento predefinito è l'incrocio delle diagonali, detto impropriamente centro dell'oggetto. Quando s'inserisce un oggetto con i comandi di posizione, il punto di riferimento coincide con le coordinate indicate con il comando. E' possibile modificare il punto di riferimento dando al parametro `rif` i valori indicati in figura 6.

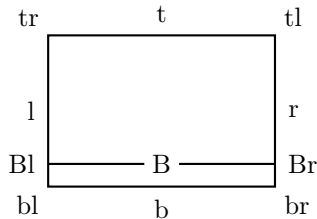


FIGURA 6: I valori del parametro `rif` sono riportati in prossimità del punto di riferimento che individuano, si noti la linea di base.

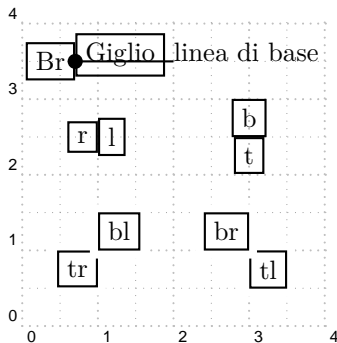


FIGURA 7: Applicazione dei parametri di posizione. In ogni rettangolo è indicato il punto di riferimento usato per l'inserimento dell'oggetto. L'esempio in alto evidenzia la linea di base e il punto di riferimento della parola nel suo rettangolo.

Con il comando `\multirput` è possibile inserire oggetti un numero di volte prefissato. La sintassi di questo comando è:

```
\multirput*[rif]{\alpha}(x_0,y_0)(\Delta x,\Delta y){N}{0}
```

Questo comando è una variante di `\rput`, il parametro `rif` ha il significato già visto. Gli oggetti 0 sono inseriti a partire dalle coordinate cartesiane (x_0, y_0) un numero di volte pari al numero intero N . Gli oggetti sono posti, distanziati, orizzontalmente e verticalmente dalle distanze $(\Delta x, \Delta y)$, possono essere anche ruotati di un angolo α . Esiste un altro comando più efficiente per oggetti grafici più complessi:

```
\multips{\alpha}(x_0,y_0)(\Delta x,\Delta y){N}{0Graf}
```

3.1 Parametri

Gli elementi grafici in PSTricks hanno, ciascuno, proprie caratteristiche modificabili attraverso l'uso di parametri. Ogni comando può essere seguito da un gruppo, delimitato da parentesi quadre, che

```
1 \begin{pspicture}(0,0)(1,1)
2   \psgrid[gridcolor=lightgray,
3     gridlabels=6pt](0,0)(1,1)
4   \multirput*{45}(0,0)(.3,.3){3}{A}
5 \end{pspicture}
1 \begin{pspicture}(0,0)(1,1)
2   \psgrid[gridcolor=lightgray,
3     gridlabels=6pt](0,0)(1,1)
4   \multips{45}(.2,.2)(.3,.3){3}{%
5     \psellipse(0,0)(.4,.2)}
6 \end{pspicture}
```

FIGURA 8: Applicazione dei comandi `\multips` e `\multirput*`.

contiene i parametri, con relativo valore, separati da una virgola. Per cambiare i valori dei parametri predefiniti, globalmente, si usa il seguente comando con la sintassi:

```
\psset{par1=val1, ..., parN=valN}
```

Se il comando `\psset` è all'interno di un ambiente, i parametri assumono i valori indicati, per ogni oggetto grafico inserito nel medesimo ambiente; al di fuori dell'ambiente i valori attivi sono quelli predefiniti. Se il comando è posto fuori da qualunque ambiente, il valore dei parametri influenza tutti gli oggetti che seguono, a parte quelli protetti da un particolare ambiente.

In sintesi, con la sintassi:

```
\comando[par1=val1, ..., parN=valN](...){...}
```

I parametri hanno effetto solo per il comando specificato.

Con la sintassi:

```
\begin{ambiente}[par1=val1, ..., parN=valN]
```

...

```
\end{ambiente}
```

I parametri hanno effetto solo localmente, l'esempio seguente è equivalente al precedente.

```
\begin{ambiente}
  \psset{par1=val1, ..., parN=valN}
```

...

```
\end{ambiente}
```

Il parametri sono numerosi, alcuni agiscono su più oggetti, altri hanno parametri peculiari. Di seguito è riportata una lista di alcuni dei parametri usati in questo articolo. La lettura della documentazione ufficiale di ciascun pacchetto è in questo caso consigliabile. Poiché è possibile indicare più parametri, separati da una virgola, il loro effetto è cumulativo. Come s'intuisce le combinazioni sono innumerevoli. Il valore dei parametri è assegnato con il segno di uguale.

```

1 \begin{pspicture}(0,0)(1.8,2)
2 \psline[linestyle=dashed,
3 dash=5pt 1pt,doubleline=true,
4 doublecolor=red,linearc=.2,
5 doublesep=2pt,shadow=true,
6 shadowsize=5pt,shadowangle=-45,
7 shadowcolor=green]
8 {->}(0,2)(1,0)(1.5,2)
9 \end{pspicture}

```

FIGURA 9: Esempio dell'uso cumulativo dei parametri.

linewidth Controlla lo spessore del tratto del disegno. Per una linea sarà lo spessore della linea, per un cerchio o un poligono sarà lo spessore del tratto del perimetro. Valore predefinito 0.8pt.

linecolor Controlla il colore delle linee e curve il cui tratto ha lo spessore indicato da **linewidth**. Valore predefinito **black**.

fillstyle Stile di riempimento delle figure. Se **fillstyle=solid** il colore uniforme di riempimento sarà dato da **fillcolor**. Valore predefinito **none**.

fillcolor Colore di riempimento.

showpoints Mostra i punti di riferimento.

linearc Controlla l'arrotondamento degli spigoli in una spezzata o un poligono. Il valore indicato è il raggio di curvatura dell'arco. Valore predefinito 0pt.

framearc Come il precedente ma per **\psframe**. I valori possibili sono fra 0 e 1. Valore predefinito 0.

linestyle Determina come una linea, o curva, debba essere tracciata. I valori possibili sono **none**, **solid**, **dashed**, **dotted**. Valore predefinito **solid**.

dash Il tratteggio è definito dalla lunghezza dei segmenti d1 del tratteggio e dalla loro spaziatura d2. Valore predefinito 5pt 3pt.

3.2 Dimensioni

L'unità di misura predefinita è il centimetro, è comunque possibile modificare l'unità assegnando alla grandezza direttamente il valore. Ad esempio l'oggetto linea ha come caratteristica il proprio spessore modificabile con il parametro **linewidth**.

PSTricks fa uso di registri per fissare i valori dei parametri dimensionali. Questi valori possono essere modificati con i comandi:

```

\pssetlength{reg}{dim}
\psaddtolength{reg}{dim}

```

```

1 \begin{pspicture}(1.7,2)
2 \rput(0,1.7){%
3 \psline[linewidth=1pt]{->}(1.5,0)}
4 \rput(0,1){%
5 \psline[linewidth=.5]{->}(1.5,0)}
6 \psset{linewidth=2pt}
7 \rput(0,0.3){\psline{->}(1.5,0)}
8 \end{pspicture}

```

FIGURA 10: Nel primo esempio lo spessore del segmento è esplicitato con l'unità di misura; nel secondo esempio manca l'unità: è attiva quella predefinita (il centimetro). Di seguito il comando **\psset** fissa una nuova unità che influenza il seguente oggetto.

Registro	Parametro
\psunit	unit
\psxunit	xunit
\psyunit	yunit
\psrunit	runit

TABELLA 1: Registri e parametri: il valore di ciascun parametro è associato al registro corrispondente, il primo è usato per lunghezze generiche, il secondo e terzo per le dimensioni in un sistema di coordinate cartesiane, l'ultimo per coordinate polari. I valori predefiniti sono per tutti uguali ad 1 cm.

L'argomento **reg** è uno dei registri riportati nella tabella 1. L'uso di questi comandi è analogo a quelli standard di LATEX **\setlength** e **\addtolength**.

Le coordinate possono essere regolate con opportuni cambiamenti di unità, nella figura 11 è illustrato l'uso dei parametri **xunit** e **yunit**.

```

1
2 \begin{pspicture}(0,0)(2,3)
3 \psgrid[xunit=0.5cm,
4 yunit=.5in,
5 gridlabels=6pt]
6 (0,0)(2,2)
7 \end{pspicture}
8
9 \begin{pspicture}(0,0)(2,2)
10 \psgrid[xunit=1cm,
11 yunit=0.25in,
12 gridlabels=6pt]
13 (0,0)(2,2)
14 \end{pspicture}

```

FIGURA 11: Cambiamento di unità per le coordinate: i valori numerici sono gli stessi, ma con unità di misura differenti.

Con il comando **\degrees[N]** si può cambiare l'unità di misura per gli angoli. Il parametro **N** indica il numero di unità per ciclo (360 gradi), senza argomenti equivale a **\degrees[360]**. Il comando **\radians** equivale a **\degrees[6.28319]**

3.3 Linee e Poligoni

Una linea può essere generata con l'uso del comando `\psline`, ecco la sintassi generale:

```
\psline*[par]{term}(x_0,y_0) \dots (x_n,y_n)
```

Le coordinate indicano gli estremi del segmento che si vuole tracciare. Indicando più di due punti, il comando disegna una spezzata. Con il parametro `linearc` si possono arrotondare gli angoli della spezzata, il valore indicato è quello del raggio dell'arco che raccorda i segmenti.

Il comando `\qline(x_1,y_1)(x_2,y_2)` senza parametri traccia un semplice segmento, il comportamento è del tutto analogo a `\psline` passante per due punti.

L'argomento `term` fra parentesi graffe specifica il tipo di terminatori che possono essere usati, agli estremi della linea, nella figura 16 sono indicati gli stili e il comando per generarli.

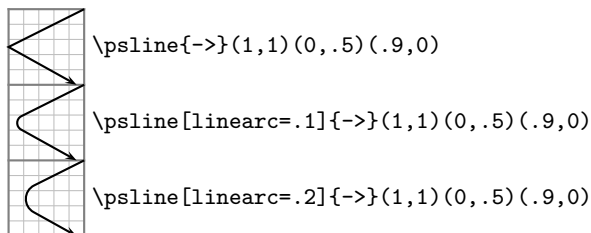


FIGURA 12: Questi tre esempi sono identici a parte il valore del parametro `linearc`. Si noti che al crescere del parametro, l'arco di raggio maggiore allontana la linea dal punto (0, .5). Vedi anche la figura 15.

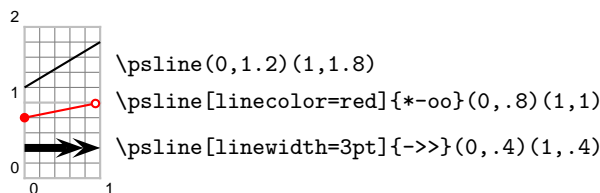


FIGURA 13: Esempi d'uso del comando `\psline`. Si noti l'uso dei parametri fra parentesi quadre.

Per tracciare poligoni si usa il comando:

```
\pspolygon*[par](x_0,y_0)(x_1,y_1) \dots (x_n,y_n)
```

Questo comando genera una spezzata chiusa fra le ultime coordinate indicate e le prime. Se si omette l'asterisco sarà tracciato il perimetro, con l'asterisco tutta l'area del poligono sarà riempita. Gli angoli possono essere arrotondati specificando il parametro `linearc`.

Per disegnare cornici sono disponibili diversi comandi con particolari caratteristiche. Una semplice cornice si ottiene con il comando:

```
\psframe*[par](x_0,y_0)(x_1,y_1)
```

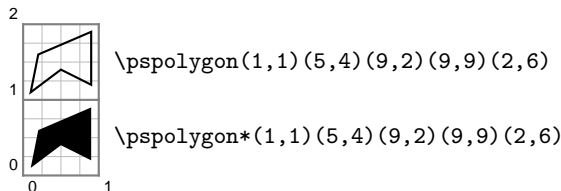


FIGURA 14: Poligoni con il comando `\pspolygon` e `\pspolygon*`.

```
\begin{pspicture}[xunit=1cm,yunit=1cm]
  [baseline=0.1](0,0)(7,2)
  \psset{gridcolor=gray,gridlabels=6pt}
  \psgrid(0,0)(0,0)(7,2)
  \psset{linecolor=red,linewidth=.5pt,
  linestyle=dashed}
  \psline(0,0)(7,2)
  \psline(1,1.5)(7,2)
  \psline(0,1.5)(.5,.5)
  \psline(1,1.5)(.5,.5)
  \psset{linecolor=blue,linestyle=solid,
  linewidth=.8pt}
  \psline[linearc=3mm]{->}
  (0,1.5)(.5,.5)(1,1.5)(7,2)(0,0)
\end{pspicture}
```

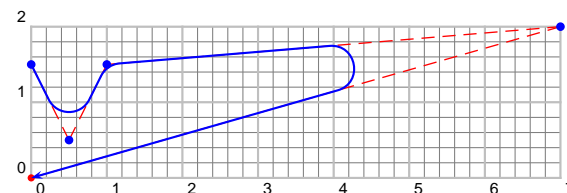


FIGURA 15: In questo esempio si vede come funziona il parametro `linearc`. Si notino le tangenti alla curva, ed i punti di riferimento.

—	-				
←	<-	→	->	↔	<->
↘	>-	↙	-<	↘	><
←	<<-	→	->>	↔	<<->>
↘	>>-	↙	-<<	↘	>>-<<
	-	—	-		-
	*-	—	- *		*- *
[[-	—]	-]	[[-]
((-	—)	-)	((-)
o	o-	—o	-o	o	o-o
•	*-	—•	-*	•	•-•
o	oo-	—o	-oo	o	oo-oo
•	**-	—•	-**	•	**-**
—	c-	—	-c	—	c-c
—	cc-	—	-cc	—	cc-cc
—	C-	—	-C	—	C-C

FIGURA 16: In questi tre gruppi di colonne sono riportati gli stili dei terminatori di linea utilizzabili. Il trattino indica la linea (segmento o curva). I simboli che creano i terminatori possono essere anche solo su un estremo o combinati fra loro.

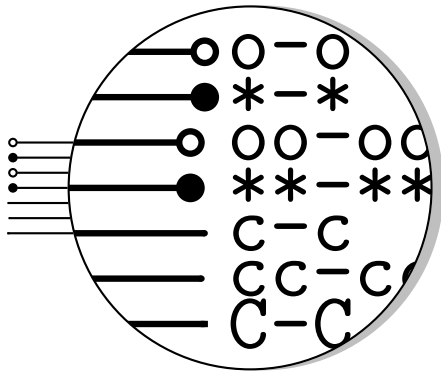


FIGURA 17: Alcuni estremi *sembrano* uguali, occhio ai dettagli.

Le coordinate da indicare sono quelle dell'angolo inferiore sinistro (non obbligatorio) e quelle dell'angolo superiore destro. Senza l'asterisco, il comando disegna una cornice rettangolare, con l'asterisco riempie l'area della figura.

I seguenti comandi generano varianti di cornici:

```
\psframebox*[par]{Ogg}
\psdblframebox*[par]{Ogg}
\psshadowbox*[par]{Ogg}
\pscirclebox*[par]{Ogg}
\psovalbox*[par]{Ogg}
```

Nella figura 18 sono riportati alcuni esempi senza parametri.

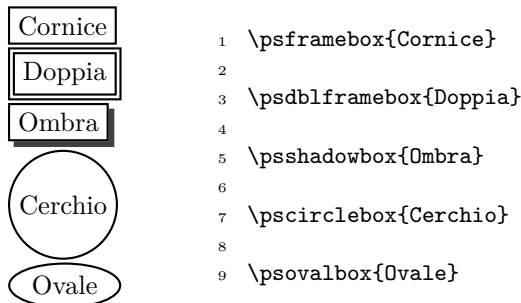
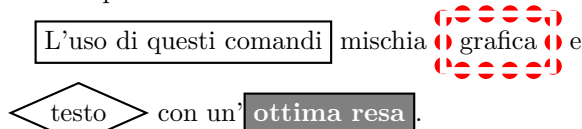


FIGURA 18: Esempi di cornici.

Per illustrare la varietà degli effetti dei parametri nella figura 19 gli esempi sono stati costruiti sul comando `\psframebox`.

Per costruire cornici con ombreggiatura si usa il comando `\psshadowbox`, in figura 20 è illustrato un esempio.



Tutto ciò è ottenuto con il codice seguente:

```
\psframebox[L'uso di questi comandi] mischia
\psdblframebox[linecolor=red,linestyle=dotted,
linewidth=2pt]{grafica} e \psdiabox{testo} con
un'\psframebox[fillstyle=solid,fillcolor=gray]
{\bfseries\color{white}ottima resa}.
```

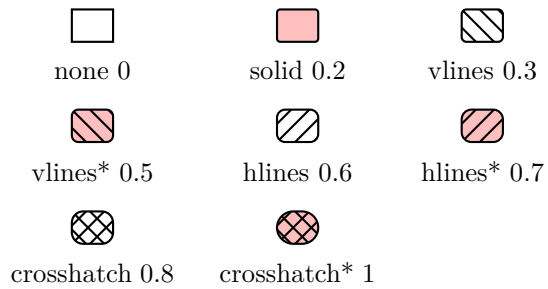


FIGURA 19: Ciascun elemento è ottenuto con il comando `\psframebox`: è indicato il valore di `fillstyle` e il valore di `framearc`, gli altri parametri rimangono gli stessi; `fillcolor=pink`. Si noti il comportamento del parametro con asterisco.

```
1 \psshadowbox[fillstyle=solid,
2 fillcolor=yellow]
3 {\color{red}
4 \begin{tabular}{c}
5 Una scatola\\colore ed ombra
6 \end{tabular}}
```

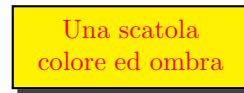


FIGURA 20: Il colore `yellow` è fra quelli predefiniti, nella sezione 3.8 è spiegato come ottenere altri colori personalizzati.

Per ottenere effetti di riempimento graduale si può attivare il parametro `fillstyle=gradient`. In questo caso è necessario caricare il pacchetto `pst-grad`. Si veda la sezione dedicata.

Per ruotare degli oggetti si usa il comando

```
\rotateleft{Oggetto}
\rotateright{Oggetto}
\rotatedown{Oggetto}
```

Nella figura 21 sono visibili tre semplici esempi.

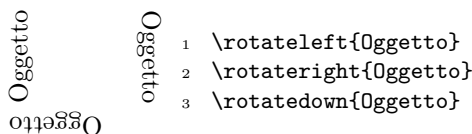


FIGURA 21: Ribaltamenti.

3.4 Archi, cerchi ed ellissi

Per disegnare un cerchio (o disco) si può usare il comando senza argomenti:

```
\qdisk(x_c, y_c){raggio}
```

Le coordinate da indicare sono quelle del centro, l'argomento è la misura del raggio. Il comando che offre completa flessibilità è:

```
\pscircle*[par](x_c, y_c){raggio}
```

Anche qui le coordinate sono quelle del centro, con l'asterisco si ottiene un cerchio, senza asterisco una circonferenza, con i parametri è possibile modificare caratteristiche quali il colore. Se si vuole disegnare solo un settore circolare il comando è:

```
\pswedge*[par](x_c,y_c){raggio}{ang1}{ang2}
```

Qui oltre alle coordinate del centro e alla misura del raggio, occorre indicare l'angolo iniziale del settore e quello finale.

Per disegnare un semplice arco si usa il comando:

```
\psarc*[par]{term}(x_c,y_c){raggio}{ang1}{ang2}
```

La variante senza asterisco traccia un semplice arco, con l'asterisco si genera una figura *piena* fra l'arco e la corda della circonferenza fra gli estremi dell'arco. Per il centro, il raggio e gli angoli vale quanto detto per il comando `\pswedge`, per l'argomento `ter` vale quanto detto per il comando `\psline`, i valori possibili sono riportati in figura 16. Per disegnare ellissi il comando è:

```
\psellipse*[par](x_c,y_c)(AsseOr,AsseVert)
```

Anche con questo comando c'è la variante senza asterisco e con asterisco (figura piena). Occorre indicare le coordinate del centro, la misura del semiasse maggiore (orizzontale) e quella del semiasse minore (verticale).

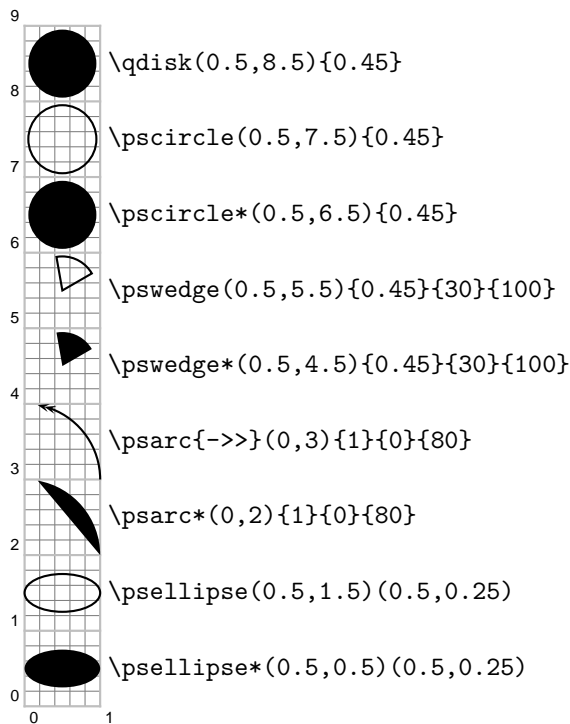


FIGURA 22: Cerchi, ellissi, settori e archi.

3.5 Curve

Per costruire e tracciare semplici curve è disponibile il comando `\psbezier`. Il comando è seguito dalle coordinate di quattro punti. I primi due (x_i, y_i) e (x_{t1}, y_{t1}) individuano la tangente di un estremo della curva (x_i, y_i) . Gli altri due punti (x_{t2}, y_{t2}) e (x_f, y_f) , individuano la tangente dell'altro estremo (x_f, y_f) .

```
\psbezier*[par]{term}
(x_i,y_i)(x_{t1},y_{t1})(x_{t2},y_{t2})(x_f,y_f)
```

I parametri `par` sono gli stessi validi per `\psline`, lo stesso vale per i terminatori `term`.

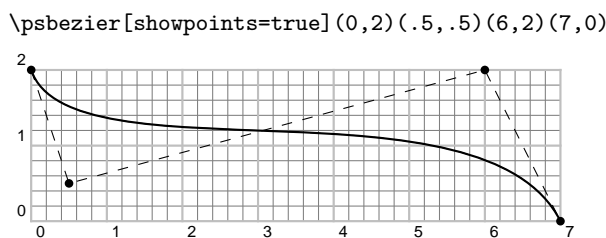


FIGURA 23: Esempio di curva `\psbezier`.

Per tracciare una parabola esiste un comando specifico:

```
\parabola*[par]{term}(x_i,y_i)(x_{Mm},y_{Mm})
```

La parabola è disegnata a partire dalla coordinata (x_i, y_i) fino al suo punto di massimo (o minimo) indicato con (x_{Mm}, y_{Mm}) . Il comando completa la parabola per simmetria.

La variante con asterisco *riempie* l'area concava dalla parabola fino alla congiungente degli estremi dell'arco. I valori dell'argomento `term` sono indicati in figura 16.

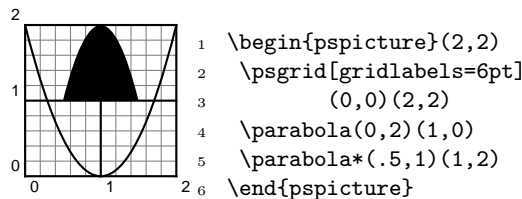


FIGURA 24: Esempio di parabola.

Ci sono altri tre comandi per tracciare curve (vedi le figure 25-29) la cui sintassi è la seguente:

```
\pscurve*[par]{sti}(x_1,y_1)...(x_n,y_n)
\psccurve*[par]{sti}(x_1,y_1)...(x_n,y_n)
\psecurve*[par]{sti}(x_1,y_1)...(x_n,y_n)
```

Questi tre comandi disegnano la curva per interpolazione fra i punti indicati. L'argomento `sti` individua lo stile degli estremi della curva in modo analogo con quanto visto per il comando `\psline` secondo la figura 16.

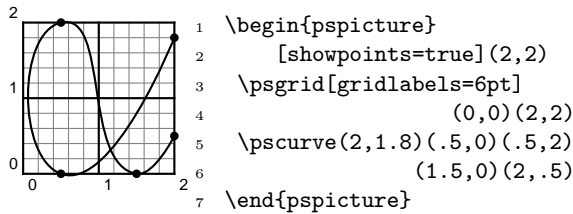


FIGURA 25: In questo esempio è stato valorizzato il parametro opzionale `showpoints` per evidenziare i punti di riferimento.

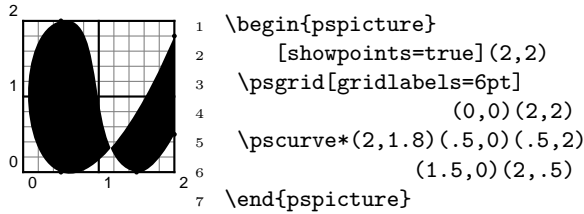


FIGURA 26: Come per la figura 25 ma con la variante asterisco.

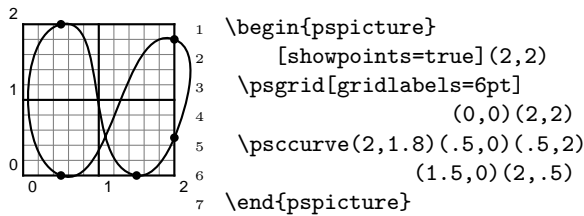


FIGURA 27: In questo esempio il comando `\psccurve` chiude la curva tra il primo e l'ultimo punto indicato.

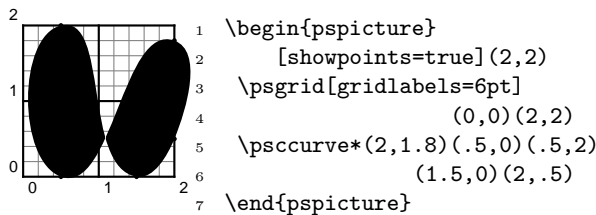


FIGURA 28: Variante con asterisco dell'esempio di figura 27.

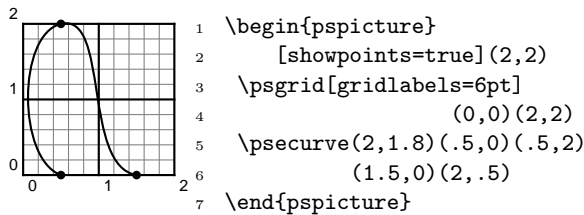


FIGURA 29: L'andamento della curva tiene conto di tutti i punti, ma omette di tracciare le parti che collegano il primo e l'ultimo punto.

3.6 Punti

PSTricks offre una varietà di stili di *punti* il cui uso è limitato solo dalla fantasia e dal gusto. Nella tabella 2 sono presentati gli stili disponibili e nella figura 30 alcuni esempi.

La sintassi dell'oggetto *punto* è la seguente:

```
\psdots*[par](x1,y1)(x2,y2)... (xn,yn)
```

I parametri possibili sono:

dotstyle=stile Il valore di *stile* è indicato nella tabella 2.

dotscale=num1 num2 Deforma il punto di tipo `dotstyle=stile` con scala `num1` orizzontalmente e con scala `num2` verticalmente.

dotangle=angolo Ruota il simbolo scelto dell'angolo indicato.

Stile	Esempio
Predefinito	• • • •
o	o o o o
+	+ + + +
x	x x x x
asterisk	* * * *
diamond	◊ ◊ ◊ ◊
diamond*	◆ ◆ ◆ ◆
oplus	⊕ ⊕ ⊕ ⊕
otimes	⊗ ⊗ ⊗ ⊗
triangle	△ △ △ △
triangle*	▲ ▲ ▲ ▲
square	□ □ □ □
square*	■ ■ ■ ■
pentagon	◊ ◊ ◊ ◊
pentagon*	◆ ◆ ◆ ◆

TABELLA 2: Varietà di stili per i punti

```

1 \begin{pspicture}(1,2)
2   \psgrid[gridlabels=6pt,
3     subgridcolor=lightgray](0,0)(1,2)
4   \psdots*[dotstyle=triangle*](0.4,.5)(0.7,.5)
5   \psdots*[dotstyle=triangle*,dotscale=2 3,
6     dotangle=45](0.3,1.5)(0.6,1.5)
7 \end{pspicture}

```

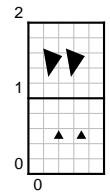


FIGURA 30: Alcuni esempi dell'uso dell'oggetto `\psdots`.

3.7 Griglie

Per chi avesse bisogno di tracciare una griglia di riferimento, PSTricks offre il comando:

```
\psgrid(x0,y0)(xis,yis)(xsd,ysd)
```

Il comando usa le coordinate di tre punti: (x_0, y_0) per indicare l'origine del sistema di coordinate, (x_{is}, y_{is}) indica la posizione dell'angolo inferiore sinistro e l'angolo superiore destro è

dato da (x_{sd}, y_{sd}) . Se viene omessa l'origine, questa coinciderà con l'angolo inferiore sinistro. Usando il comando all'interno dell'ambiente `pspicture` si possono omettere le coordinate di tutti e tre i punti, in tal modo la griglia è tracciata secondo le dimensioni dell'ambiente. Il comando è correlato da parametri opzionali riepilogati nella tabella 3.

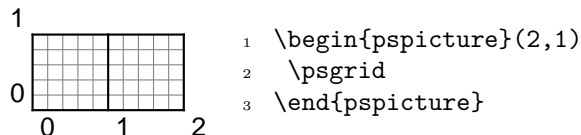


FIGURA 31: Esempio d'uso di `\psgrid` senza parametri e coordinate all'interno dell'ambiente `pspicture`

La struttura della griglia è costruita per multipli interi dell'unità `xunit` e `yunit`, è possibile cambiare l'unità di misura direttamente come mostrato in figura 32 e figura 11.

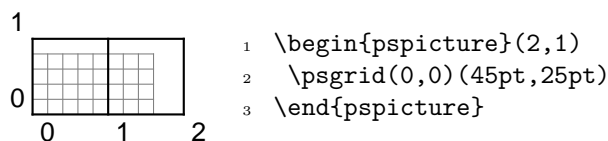


FIGURA 32: In questo caso sono state indicate le unità di misura. La griglia principale è stata disegnata con le dimensioni dell'ambiente `pspicture`.

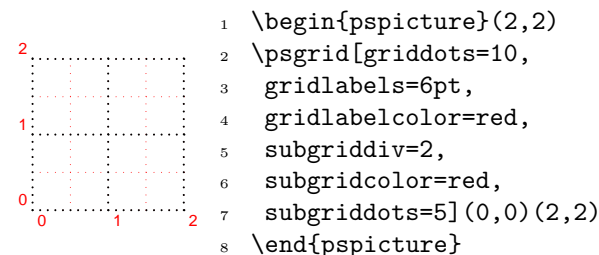


FIGURA 33: In questo caso sono stati valorizzati diversi parametri, si noti la dimensione delle etichette e la diversa divisione della griglia con l'uso dei punti in luogo della linea continua.

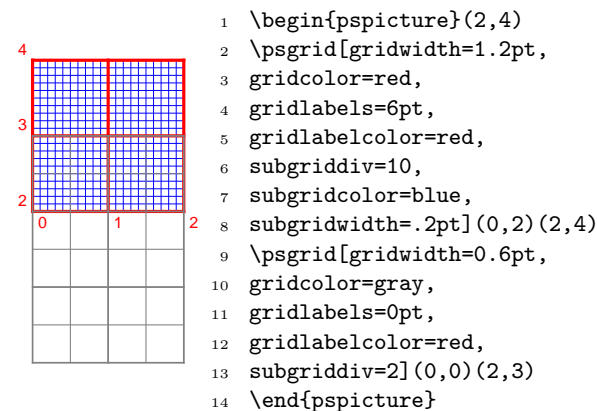


FIGURA 34: Due griglie sovrapposte. Sono state modificate le dimensioni dello spessore delle linee della griglia e della sottogriglia. Si noti che per eliminare le etichette della seconda griglia si è posto `gridlabels=0pt`. Altri parametri sono illustrati nella tabella 3.

3.8 I colori

PSTricks offre una serie di colori e una scala di grigi predefiniti, ma con opportuni comandi, l'utente può definire e generare tutti i colori che desidera. Nella figura 35 sono indicati i comandi per ciascun colore predefinito. L'attivazione del colore ha effetto sul testo e sugli oggetti grafici che seguono il comando che attiva il colore specifico. Tale effetto può essere confinato all'interno di un gruppo delimitato dalle parentesi graffe oppure quando è contenuto in ambienti delimitati da

`\begin{...}\end{...}`

L'influenza di un comando può essere interrotta da un altro comando che attiva un diverso colore, o schermata dagli ambienti o dalle parentesi graffe.

Questo è un esempio ottenuto con il seguente codice:

`{\blue Questo è un \red esempio}`

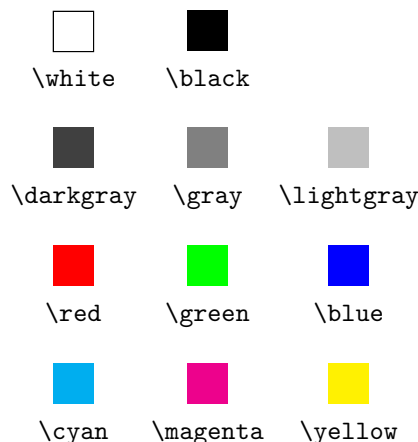


FIGURA 35: Colori predefiniti e comandi relativi

Per generare sfumature di grigio personalizzate si usa il comando:

`\newgray{nomecolore}{num}`

L'argomento `num` deve avere un valore compreso tra 0 (nero) e 1 (bianco).

Per richiamare il colore personalizzato si usa il comando:

`\color{nomecolore}`

Per generare i colori che si desiderano, occorre definirli con uno dei seguenti comandi:

`\newrgbcolor{nomecolore}{n1 n2 n3}`
`\newhsbcolor{colore}{n1 n2 n3}`
`\newcmykcolor{colore}{n1 n2 n3 n4}`

Ciascuno dei precedenti comandi usa una codifica numerica. I valori numerici devono avere valori $0 \leq nx \leq 1$.

Nome	Significato	Predefinito
gridwidth	Spessore delle linee della griglia	.8pt
gridcolor	Colore delle linee della griglia	black
griddots	Punti, nel numero indicato, per ogni divisione data da subgriddiv	0
gridlabels	Dimensione dei numeri d'etichetta	10pt
gridlabelcolor	Colore per gridlabels	false
subgriddiv	Numero di linee divisorie per unità di misura	5
subgridwidth	Spessore delle linee divisorie	4pt
subgridcolor	Colore delle linee divisorie	gray
subgriddots	Come per griddots, ma per le sotto divisioni della griglia	0

TABELLA 3: Elenco dei parametri per il comando `\psgrid`

Esempio di grigio
con altro grigio

```
\newgray{grigiotre}{.3}
\newgray{grigiosette}{.7}
Esempio di \color{grigiotre}\ grigio\
con \color{grigiosette}\ altro grigio
```

FIGURA 36: Tonalità personalizzate di grigio. Definizione e uso.

Per questioni di resa e compatibilità si consiglia d'usare la codifica RGB implementata dal primo comando. Le cifre della tripletta di valori definiscono il colore *mischiando* tre colori base: rosso, verde e blu.

```
1 \newrgbcolor{rosso}{1 0 0}
2 \newrgbcolor{verde}{0 1 0}
3 \newrgbcolor{blu}{0 0 1}
4 \newrgbcolor{primo}{.1 .6 0}
5 \newrgbcolor{secondo}{0 1 1}
6 \newrgbcolor{terzo}{.5 .2 .7}
7 {\color{rosso}\rule{15pt}{15pt}} \
8 {\color{verde}\rule{15pt}{15pt}} \
9 {\color{blu}\rule{15pt}{15pt}} \
10 {\color{primo}\rule{15pt}{15pt}} \
11 {\color{secondo}\rule{15pt}{15pt}} \
12 {\color{terzo}\rule{15pt}{15pt}} \
```

FIGURA 37: Colori: definizioni ed uso.

4 Le estensioni di PSTricks

Le estensioni di PSTricks sono in continua evoluzione. Il loro numero aumenta in continuazione. La seguente lista non può che essere manchevole, ma un'accurata ricerca su internet farà scoprire pacchetti per le più disparate applicazioni ³.

3. Alcuni degli esempi di questa sezione sono tratti dai manuali dei rispettivi pacchetti ai quali si riferiscono. Altri notevoli esempi si possono trovare nel sito <http://www.tug.org/PSTricks/main.cgi?file=examples>

pstricks

È il pacchetto di base del mondo PSTricks. La sua estensione diretta è il pacchetto `pstricks-add`. Per compatibilità con gli altri pacchetti correlati è importante caricare `pstricks-add` in coda agli altri pacchetti `pst-xxxx`.

pst-3d

Pacchetto di base per generare elementari effetti in 3D.

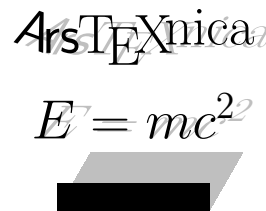


FIGURA 38: Esempio per il pacchetto `pst-3d`

Nella figura 38 sono mostrati gli effetti di base possibili con il pacchetto `pst-3d`.

_____ Codice per la figura 38 _____

```
1 \newgray{gray75}{.75}
2 \psset{Tshadowcolor=gray75}
3 \psshadow{\huge \Ars}\[10pt]
4 \psshadow{\huge $E=mc^2$}\[15pt]
5 \psshadow[Tshadowsize=2.5]{%
6 \rule{2cm}{10pt}}
```

pst-3dplot

Permette di disegnare grafici e figure in 3D.

_____ Codice per la figura 39 _____

```
1 \begin{pspicture}(-1,-1)(2,2.5)%
2 \pstThreeDCoor[xMin=-1,xMax=2,yMin=-1,yMax=2,
3 zMin=-1,zMax=2]%
4 \pstThreeDBox(-1,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
5 \pstThreeDBox[RotX=90,linecolor=red]
6 (0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
7 \pstThreeDBox[RotX=90,RotY=90,linecolor=green]
8 (0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
```

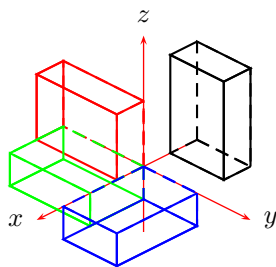


FIGURA 39: Oggetti in 3D con il pacchetto `pst-3dplot`

```

9 \pstThreeDBox[RotX=90,RotY=90,RotZ=90,
10                 linecolor=blue]
11                 (0,0,0)(.5,0,0)(0,1,0)(0,0,1.5)
12 \end{pspicture}%

```

pstricks-add

Pacchetto addizionale che estende il pacchetto principale `pstricks` e altri due pacchetti standard: `pst-node` e `pst-plot`. È importante che `pstricks-add` sia caricato per ultimo, dopo tutti i pacchetti correlati con `pstricks`. Automaticamente è caricato `xkeyval`, `pstricks-add` dipende da questo pacchetto poiché si poggia su `pst-xkey` che è parte del pacchetto `xkeyval`.

Il pacchetto `pstricks-add` incrementa di molto la varietà di stili per gli assi cartesiani e le scale dei sistemi di riferimento. Di seguito le figure riportano alcuni esempi di scale logaritmiche e una varietà di assi riferimento.

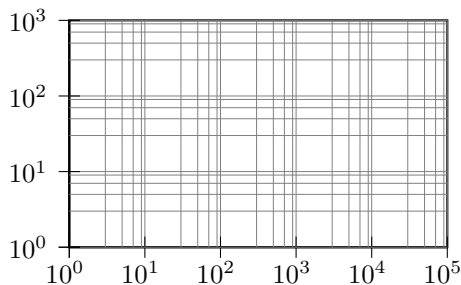


FIGURA 40: Mappe logaritmiche

————— Codice per la figura 40 —————

```

1 \psaxes[subticks=5,axesstyle=frame,
2         xylogBase=10,logLines=all](5,3)

```

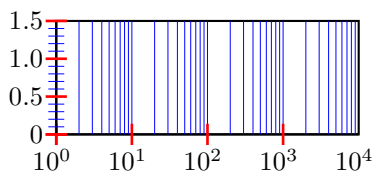


FIGURA 41: Mappe semilogaritmiche

————— Codice per la figura 41 —————

```

1 \psaxes[axesstyle=frame,logLines=x,
2         xlogBase=10,Dy=0.5,tickcolor=red,

```

```

3     subtickcolor=blue,tickwidth=1pt,
4     ysubticks=5,xsubticks=10](4,1.5)

```

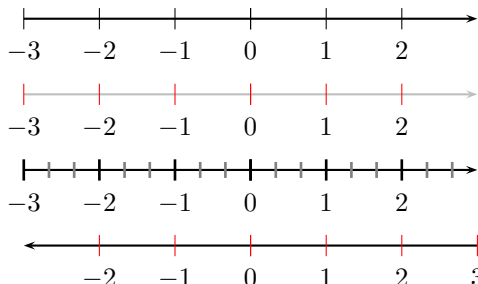


FIGURA 42: Assi

————— Codice per la figura 42 —————

```

1 \psset{arrowscale=1}
2 \rput(0,3){\psaxes[yAxis=false,
3             subticks=0]{->}(0,0)(-3,-3)(3,3)}
4 \rput(0,2){\psaxes[yAxis=false,
5             subticks=0,tickcolor=red,
6             linecolor=blue]{->}(0,0)(-3,-3)(3,3)}
7 \rput(0,1){\psaxes[yAxis=false,
8             subticks=3,tickwidth=1pt,
9             subtickwidth=1pt]{->}(0,0)(-3,-3)(3,3)}
10 \rput(0,0){\psaxes[yAxis=false,subticks=0,
11                  tickcolor=red]{>}(0,0)(-3,-3)(3,3)}

```

pst-abspos

Permette d’inserire oggetti tramite coordinate assolute. L’oggetto può essere inserito in qualunque punto con precisione. Per questo pacchetto l’origine delle coordinate coincide con l’angolo superiore sinistro del foglio della pagina.

pst-asr

Linguistica: permette di creare svariati diagrammi utili per questa disciplina.

pst-bar

Pacchetto per la creazione di grafici ad istogrammi. Per disegnare gli assi di riferimento occorre richiamare anche il pacchetto `pst-plot`.

pst-barcode

Permette la creazione e la stampa di codici a barre.

pst-blur

Questo pacchetto permette di ottenere ombre sfumate, meno nette di quelle prodotte con `PSTricks` standard. Vedi la figura 44.

pst-calendar

Genera calendari in forma tabellare e in 3D a forma di dodecaedro.

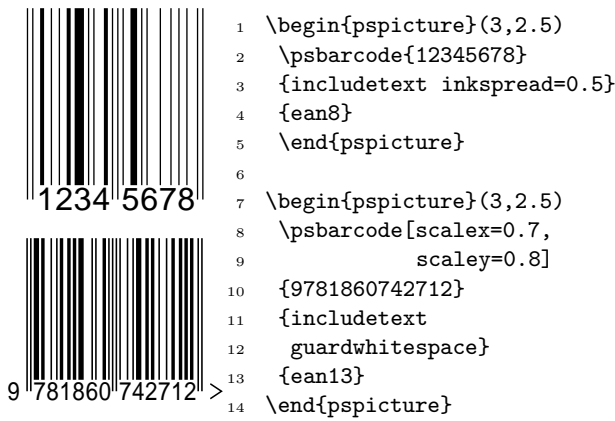


FIGURA 43: Esempio per il pacchetto pst-barcode.

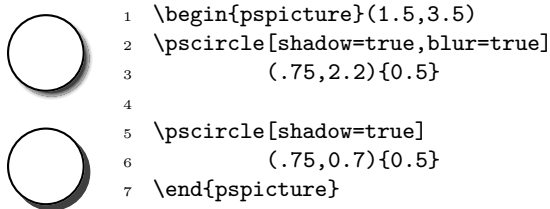


FIGURA 44: Si noti la differenza: l'ombra del secondo cerchio è quella standard di PSTricks.

pst-circ

Con questo pacchetto è possibile disegnare circuiti elettronici con una buona resa tipografica. Il pacchetto offre una buona varietà di comandi per generare i componenti ideali quali bipoli, tripoli e quadripoli. Resistenze, condensatori, diodi, transistor (per citarne solo alcuni) hanno ciascuno un proprio comando. Una nutrita schiera di parametri, consente di aggiungere al circuito informazioni che completano il disegno in ogni sua parte. La sintassi è semplice ed intuitiva.

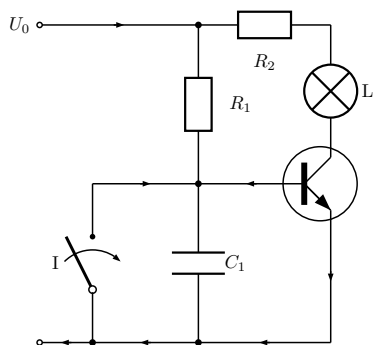


FIGURA 45: Circuiti con pst-circ

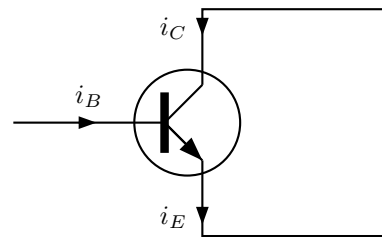
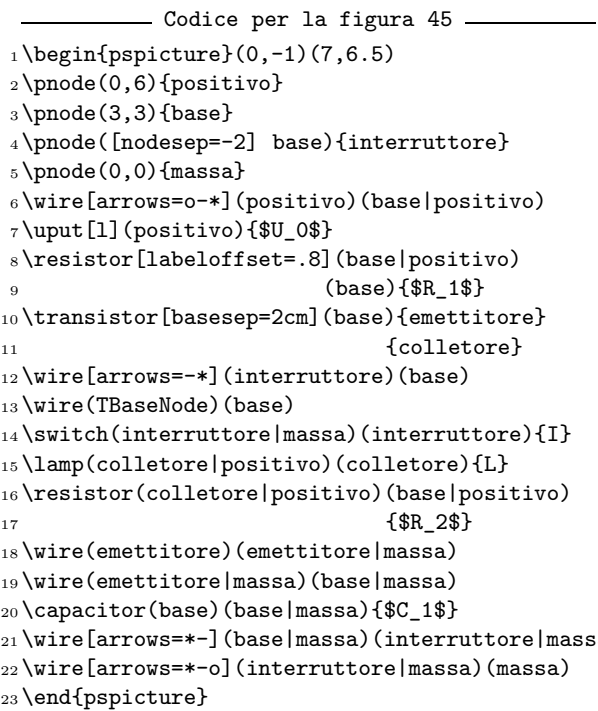
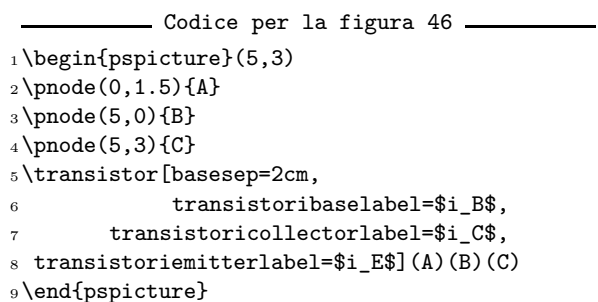


FIGURA 46: Le connessioni sono ottenute con la stessa tecnica usata per i nodi dei grafi del pacchetto pst-node.



pst-coil

Permette di disegnare molle, eliche e spire. Il codice per ottenere questi oggetti è:

```

\pscoil(0,4)(4,4)
\psset{coilwidth=0.75}
\pscoil[coilaspect=0](0,3)(4,3)
\pscoil[coilaspect=30,coilheight=0.3]
(0,2)(4,2)

```

Le coordinate indicano gli estremi delle molle.

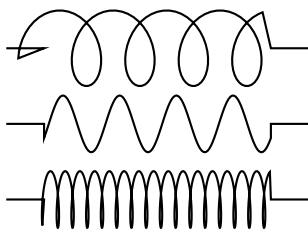


FIGURA 47: Eliche, spire e molle.

pst-dbicons

Permette di disegnare la struttura di una base dati, tabelle e loro relazioni.

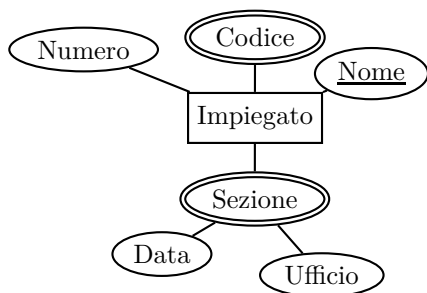


FIGURA 48: I collegamenti sono ottenuti tramite l'uso di etichette e nodi, il funzionamento ricorda il pacchetto pst-node.

pst-eps

Pacchetto che permette di salvare grafici da ambienti PSTricks in file in formato eps.

pst-eucl

Consente di disegnare le figure della geometria euclidea.

pst-fill

Permette di generare riempimenti con tratteggio di vario tipo, un effetto analogo a quello prodotto dai vecchi retini usati nel disegno tecnico.

pst-fr3d

Consente la costruzione di *cornici* 3D

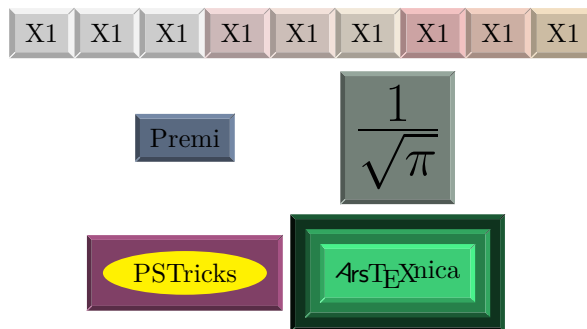


FIGURA 49: Pulsanti in 3D costruiti con pst-fr3d

```

14 \Huge $\frac{1}{\sqrt{\pi}}$ \
15 \PstFrameBoxThreeD[FrameBoxThreeDColorHSB
16 =0.9 0.5 0.5]{%
17 \psovalbox[fillstyle=solid,fillcolor=yellow]
18 {PSTricks}} &
19 \PstFrameBoxThreeD[FrameBoxThreeDColorHSB
20 =0.4 0.7 0.2]{%
21 \PstFrameBoxThreeD[FrameBoxThreeDColorHSB
22 =0.4 0.7 0.5]{%
23 \PstFrameBoxThreeD[FrameBoxThreeDColorHSB
24 =0.4 0.7 0.8]{\Ars}}
25 \end{psmatrix}
    
```

pst-func

Per disegnare il grafico di speciali funzioni matematiche.

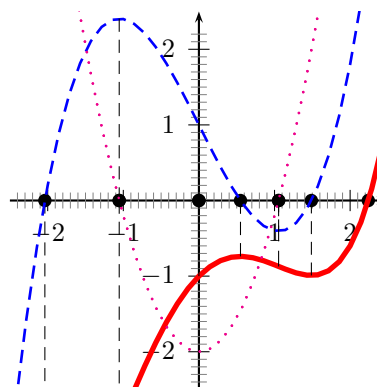


FIGURA 50: Grafico generato con pst-func.

————— Codice per la figura 49 —————

```

1
2 \begin{pspicture}[doublesep=0.05](0,0)(6,1)
3 \multido{\nButtonA=0+0.1}{3}{%
4 \multido{\nButtonB=0+0.05}{3}{%
5 \PstFrameBoxThreeD[FrameBoxThreeDColorHSB=%
6 \nButtonB\space \nButtonA\space 0.8]{X1}}\
7 \end{pspicture}
8
9 \begin{psmatrix}[rowsep=1mm,colsep=1mm]
10 \PstFrameBoxThreeD[FrameBoxThreeDColorHSB
11 =0.6 0.3 0.5]{Premi} &
12 \PstFrameBoxThreeD[FrameBoxThreeDColorHSB
13 =0.4 0.1 0.5]{%
    
```

————— Codice per la figura 50 —————

```

1 \begin{pspicture*}(-2.5,-2.5)(2.5,2.5)
2 \psaxes{->}(0,0)(-2.5,-2.5)(2.5,2.5)%
3 \psset{dotscale=2}
4 \psPolynomial[markZeros,linecolor=red,
5 linewidth=2pt,coeff=-1 1 -1 0 0.15]{-4}{3}
6 \psPolynomial[markZeros,linecolor=blue,
7 linewidth=1pt,linestyle=dashed,%
8 coeff=-1 1 -1 0 0.15,Derivation=1,
9 zeroLineTo=0]{-4}{3}%
10 \psPolynomial[markZeros,linecolor=magenta,
11 linewidth=1pt,linestyle=dotted,%
    
```

```

12 coeff=-1 1 -1 0 0.15,Derivation=2,
13 zeroLineTo=0}{-4}{3}%
14 \psPolynomial[markZeros,linecolor=magenta,
15 linewidth=1pt,linestyle=dotted,%
16 coeff=-1 1 -1 0 0.15,Derivation=2,
17 zeroLineTo=1}{-4}{3}%
18 \end{pspicture*}

```

pst-geo

Consente di disegnare mappe di stati e continenti.

pst-gr3d

Questo pacchetto consente di disegnare griglie tridimensionali.

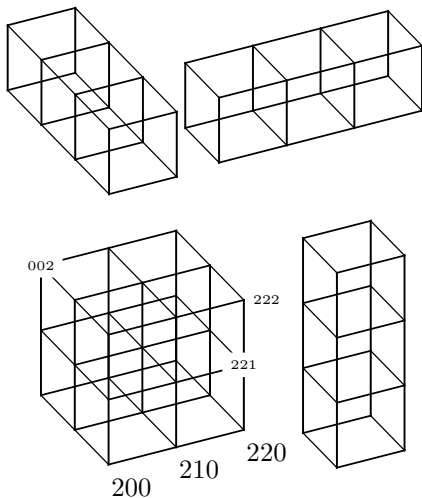


FIGURA 51: Figure 3D con pst-gr3d

————— Codice per la figura 51 —————

```

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

```

pst-grad

Permette di generare sfumature di colore con più colori.

————— Codice per la figura 54 —————

```

1 \newcommand{\Fig}[1][[]]{%
2 \begin{pspicture}(2,2)
3 \psframe[#1](2,2)
4 \end{pspicture}}
5 \newhsbcolor{ColorA}{0 0 0.7}
6 \newhsbcolor{ColorB}{0 1 0.7}
7 \newhsbcolor{ColorC}{.5 0.8 0}
8 \newhsbcolor{ColorD}{.5 0.8 1}

```

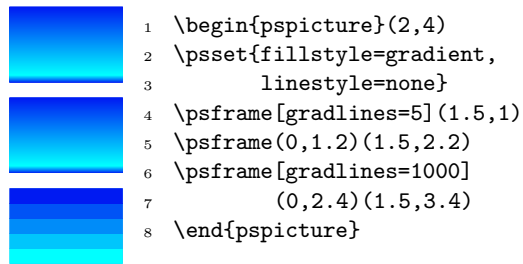


FIGURA 52: Si noti la qualità della sfumatura in relazione al parametro gradlines.

```

1 \begin{pspicture}(2,4)
2 \psset{fillstyle=gradient,
3 linestyle=none,
4 gradmidpoint=0.5}
5 \psframe[gradangle=0](1.5,1)
6 \psframe[gradangle=45](0,1.2)
7 \psframe[gradangle=90](0,2.4)
8 \psframe[gradangle=1000]
9 (0,2.4)(1.5,3.4)
10 \end{pspicture}

```

FIGURA 53: Sfumature con diversa angolazione al variare del valore del parametro gradangle.

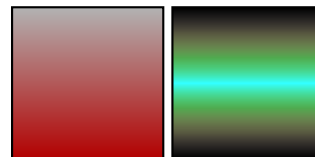


FIGURA 54: Sfumature con più colori.

```

9 \psset{fillstyle=gradient,
10 gradientHSB=true}
11 \psframe[gradmidpoint=1,
12 gradbegin=ColorA,gradend=ColorB](2,2)
13 \psframe[gradmidpoint=0.5,
14 gradbegin=ColorC,gradend=ColorD](2,2)

```

pst-infixplot

Permette di disegnare grafici di funzioni a partire dalla notazione algebrica. Semplifica molto l'uso di pst-plot dal quale dipende e che carica automaticamente, insieme al pacchetto complementare infix-RPN.

————— Codice per la figura 55 —————

```

1 \psaxes{->}(0,0)(0,-2)(5.5,3)
2 \infixtoRPN{sqrt(x)}
3 \psplot[linecolor=green]{0}{5}{\RPN}
4 \infixtoRPN{x^0.2}
5 \psplot[linecolor=red]{0}{5}{\RPN}
6 \infixtoRPN{sin(-x*180/3.1415)}
7 \psplot[linecolor=blue]{0}{5}{\RPN}

```

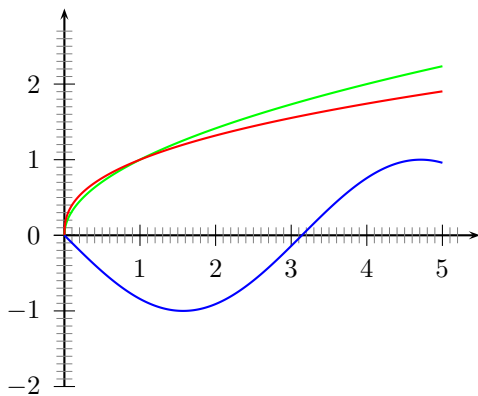


FIGURA 55: Grafici con `infix-RPN`.

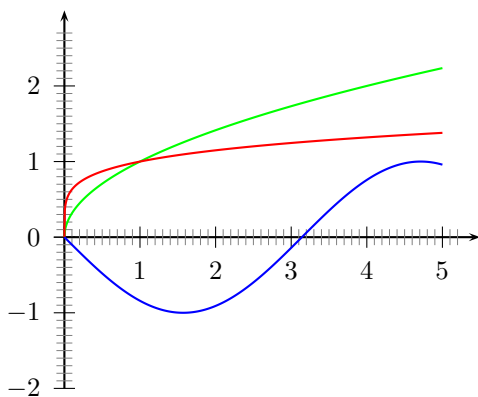


FIGURA 56: Grafici con `pst-infixplot`.

————— Codice per la figura 56 —————

```

1 \psaxes{->}(0,0)(0,-2)(5.5,3)
2 \infixtoRPN{sqrt(x)}
3 \psPlot[linecolor=green]{0}{5}{sqrt(x)}
4 \psPlot[linecolor=red]{0}{5}{x^0.2}
5 \psPlot[linecolor=blue]{0}{5}
6   {sin(-x*180/3.1415)}

```

pst-jtree

Consente di disegnare le strutture ad albero più usate nel campo della linguistica.

pst-labo

Consente di riprodurre le attrezzature di base di un laboratorio fisico-chimico. Di seguito sono proposte alcune immagini accompagnate dal codice che genera ciascun oggetto. Da notare l'uso combinato dei parametri.

————— Codice per la figura 57 —————

```

1 \psset{bouchon=true}
2 \pstTubeEssais[glassType=tube]
3 \pstTubeEssais[glassType=ballon]
4 \pstTubeEssais[glassType=erlen]
5 \pstTubeEssais[glassType=flacon]
6 \pstTubeEssais[tubeDroit=true,
7   tubePenche=10]

```

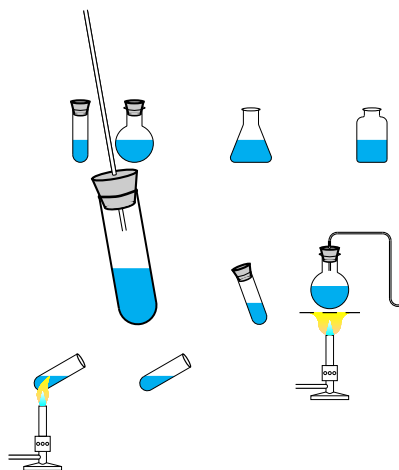


FIGURA 57: Teoria di strumenti da laboratorio.

```

8 \pstTubeEssais[tubePenche=20,bouchon]
9 \pstChauffageBallon[tubeRecourbeCourt]
10 \psset{tubeSeul=true}
11 \pstChauffageTube
12 \pstChauffageTube[becBunsen=false]

```

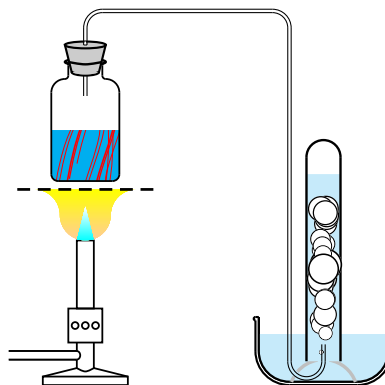


FIGURA 58: Ogni oggetto che compone la figura è *regolato* dall'uso dei parametri.

————— Codice per la figura 58 —————

```

1 \psset{glassType=flacon,recuperationGaz,
2   substance=\pstFilaments{red}}
3 \pstChauffageBallon[tubeRecourbe]

```

pst-lens

Implementa lenti per ingrandire o rimpicciolire figure, testo o altri oggetti di una pagina. Il codice seguente è quello della figura 17.

————— Codice per la figura 17 —————

```

1 \def\Estremi{{ \rput[b1](0,0){%
2 \begin{pspicture}(0,0)(7,3)
3 \rput(3,1.7){\psline{o-oo}(-3,0)}
4 \rput[l](3.1,1.7){\texttt{o-o}}

```

```

5 \rput(3,1.5){\psline{*-*}(-3,0)}
6 \rput[1](3.1,1.5){\texttt{*-*}}
7 \rput(3,1.3){\psline{oo-oo}(-3,0)}
8 \rput[1](3.1,1.3){\texttt{oo-oo}}
9 \rput(3,1.1){\psline{**-*}(-3,0)}
10 \rput[1](3.1,1.1){\texttt{**-*}}
11 \rput(3,.9){\psline{c-c}(-3,0)}
12 \rput[1](3.1,.9){\texttt{c-c}}
13 \rput(3,.7){\psline{cc-cc}(-3,0)}
14 \rput[1](3.1,.7){\texttt{cc-cc}}
15 \rput(3,.5){\psline{C-C}(-3,0)}
16 \rput[1](3.1,.5){\texttt{C-C}}
17 \end{pspicture}
18 }}}
19 \begin{pspicture}(0,-1)(7,3.5)
20 \Estremi
21
22 \PstLens[LensHandle=false,
23         LensSize=2.4,
24         LensMagnification=3]
25         (3.2,1.1){\Estremi}
26 \end{pspicture}

```

La lente ingrandisce quanto indicato nel secondo argomento del comando `\PstLens` con riferimento alle coordinate indicate nel primo argomento (3.2,1.1). Risulta comodo definire una macro che definisce l'oggetto da sottoporre alla lente; in questo caso si è definita la macro `\Estremi`. I parametri usati indicano che non si vuole far disegnare il manico della lente: `LensHandle=false`, il raggio della lente è pari a 2.4 cm, l'ingrandimento è pari a tre volte l'originale. La lente può avere svariate forme personalizzabili.

pst-light3d

Implementa effetti di luce 3D



FIGURA 59: Nel codice riportato di seguito, si noti l'uso del parametro `LightThreeDAngle` che consente di variare l'angolo d'incidenza della luce.

————— Codice per la figura 59 —————

```

1 \DeclareFixedFont{\Bf}{T1}{ptm}{b}
2         {n}{1.5cm}
3 \PstLightThreeDText[fillstyle=solid,
4         fillcolor=white]{\Bf PSTricks}\
5 \DeclareFixedFont{\Bf}{T1}{ptm}{b}

```

```

6         {n}{1.5cm}
7 \PstLightThreeDText[linestyle=none,
8         fillstyle=solid,
9         fillcolor=darkgray]
10        {\Bf Luce}\
11 \psset{linestyle=none,fillstyle=solid,
12        fillcolor=green}
13 \PstLightThreeDText[LightThreeDAngle=0]
14        {\Bf Zero}\
15 \PstLightThreeDText[LightThreeDAngle=90]
16        {\Bf Novanta}\

```

pst-math

Estende, migliorandoli, gli operatori matematici standard. Il manuale del pacchetto è semplice e schematico, si possono ottenere grafici complessi con relativamente poco codice.

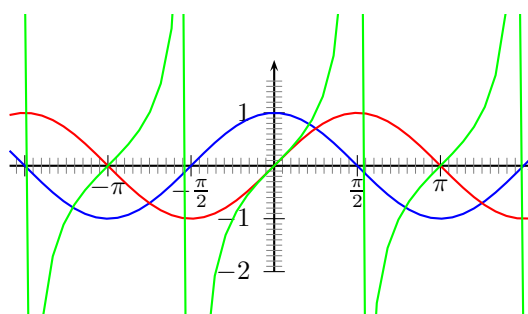


FIGURA 60: Esempio di grafico con il pacchetto `pst-math`.

————— Codice per la figura 60 —————

```

1 \psset{unit=0.5cm,xunit=0.5cm,yunit=0.5cm}
2 \SpecialCoor
3 \psaxes[labels=y,Dx=\pstPI2]{->}%
4 (0,0)(-5,-2)(5,2)
5 \uput[-90](!PI 0){$\pi$}
6 \uput[-90](!PI neg 0){$-\pi$}
7 \uput[-90](!PI 2 div 0){$\frac{\pi}{2}$}
8 \uput[-90](!PI 2 div neg 0){
9  $-\frac{\pi}{2}$}
10 \psplot[linecolor=blue]{-5}{5}{x COS}
11 \psplot[linecolor=red]{-5}{5}{x SIN}
12 \psplot[linecolor=green]{-5}{5}{x TAN}

```

pst-node

Permette la costruzione di grafi aperti e chiusi con un completo controllo delle curve di collegamento con i nodi. L'idea di base consiste nell'assegnare un'etichetta ai nodi. Le etichette saranno gli estremi delle curve che collegano i nodi. Di seguito è riportato il codice della figura 2

————— Codice per la figura 2 —————

```

1 \begin{psmatrix}[rowsep=.4cm,colsep=0.5cm]
2 & \rnode{Ed}{\psshadowbox[fillstyle=solid,
3         framesep=0.2]
4         {Il vostro editor preferito}}\
5 \rnode{ftex}{\psdblframebox{file.tex}}&

```

```

6 \rnode{ptex}{\psframebox{latex file.tex}}\
7 \rnode{fdvi}{\psdblframebox{file.dvi}}&
8 \rnode{vdvi}{\rnode{vdvi}}{&
9 \psframebox[fillcolor=blue,fillstyle=solid]
10 {\bfseries %
11 \color{white}Visualizzatore file DVI}}\
12 \rnode{pdvi}{\psframebox{dvips file.dvi}}&\
13 \rnode{fps}{\psdblframebox{file.ps}}&
14 \rnode{vps}{\psframebox[fillcolor=red,
15 fillstyle=solid]
16 {\bfseries %
17 \color{white}Visualizzatore file PS}}\
18 \rnode{ppspdf}{\psframebox{ps2pdf file.ps}}&\
19 \rnode{fpdf}{\psdblframebox[doubleline=true]
20 {file.pdf}}&
21 \rnode{vpdf}{\psframebox[framearc=.2,
22 fillstyle=gradient,%
23 gradangle=0,
24 gradbegin=blue,
25 gradend=green]%
26 {\bfseries
27 \color{white}Visualizzatore file PDF}}
28 \ncurve[angleA=180,angleB=60,
29 doubleline=true]{->}{Ed}{ftex}
30 \ncurve[angleA=0,angleB=180]
31 {->}{ftex}{ptex}
32 \ncurve[angleA=245,angleB=60]
33 {->}{ptex}{fdvi}
34 \ncurve[angleA=0,angleB=180]
35 {->}{fdvi}{vdvi}
36 \ncurve[angleA=-90,angleB=90]
37 {->}{fdvi}{pdvi}
38 \ncurve[angleA=-90,angleB=90]
39 {->}{pdvi}{fps}
40 \ncurve[angleA=-90,angleB=90]
41 {->}{pps}{fps}
42 \ncurve[angleA=0,angleB=180]
43 {->}{fps}{vps}
44 \ncurve[angleA=-90,angleB=90]
45 {->}{fps}{ppspdf}
46 \ncurve[angleA=-90,angleB=90]
47 {->}{ppspdf}{fpdf}
48 \ncurve[angleA=0,angleB=180]
49 {->}{fpdf}{vpdf}
50 \end{psmatrix}

```

La sintassi è semplice, consideriamo la riga di codice:

```
\rnode{ftex}{\psdblframebox{file.tex}}
```

Il comando `\rnode` ha due argomenti, il primo è il nome del nodo `ftex`, il secondo è `\psdblframebox{file.tex}`; cioè un qualunque oggetto che vogliamo fare diventare il nodo del nostro grafo. A questo punto, individuati tutti i nodi di cui c'è bisogno, si creano le curve di collegamento:

```
\ncurve[angleA=0,angleB=180]
{->}{ftex}{ptex}
```

In sequenza: il nome del comando che genera la curva, i parametri che indicano l'angolo di partenza della curva e di arrivo sull'altro nodo. Il tipo di terminatore della curva, i nomi dei nodi di partenza e arrivo.

Il diagramma è stato costruito entro l'ambiente `psmatrix`. Si tratta di un ambiente semplice e flessibile, consente di creare semplici matrici di oggetti. Gli oggetti su ogni riga sono separati dal simbolo `&`. La fine riga è segnata dalle `\`.

pst-ob3d

Implementa semplici oggetti grafici tridimensionali

pst-pdf

Utile strumento che consente di usare figure costruite con codice PSTricks in documenti PDF.

pst-optic

Per disegnare figure utili nell'illustrazione dell'ottica fisica e tecnica

pst-osci

Riproduce i grafici tipici generati dagli oscilloscopi.

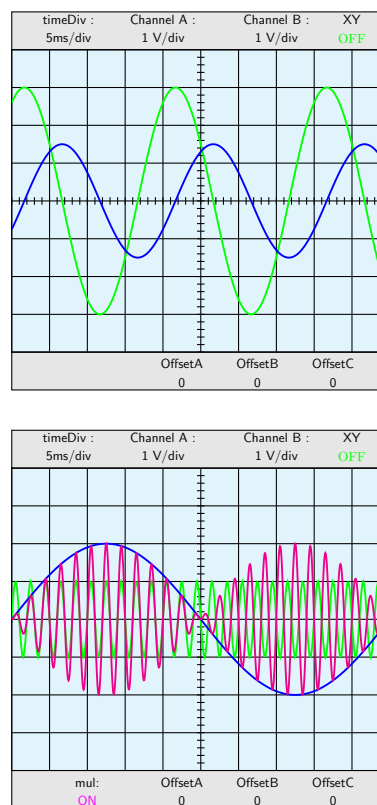


FIGURA 61: Oscilloscopio: `pst-osci`

Codice per la figura 61

```

1 \psscalebox{0.5}{\Oscillo[amplitude1=3,
2 amplitude2=1.5, phase1=60,
3 phase2=-30]}
4
5 \psscalebox{0.5}{\Oscillo[amplitude1=1,
6 amplitude2=2,period2=50,
7 period1=2, combine= true, operation= mul]}

```

pst-pdgr

Consente di elaborare alberi genealogici utili in ambito medico.

pst-plot

Questo potente pacchetto definisce alcune macro che consentono di tracciare grafici a partire da un insieme di punti.

Le coordinate dei punti possono essere passate alla macro in due modi.

Primo metodo: consiste nell’inserire le coordinate dei punti in un file. A questo punto occorre indicare il file come argomento al comando che tratterà la curva.

`\fileplot*[par]{filedati}`

Il comando `\fileplot` consente di tracciare una curva, costruita per interpolazione, a partire dai punti elencati nel file `filedati` in termini di coordinate cartesiane. Il file dati dovrà essere composto con le seguenti regole:

- Tutti i dati devono essere contenuti fra parentesi quadre [...]
- I valori delle coordinate possono essere delimitati dalle parentesi graffe, dalle parentesi tonde, dalla virgola o da spazi.
- Non devono essere indicate unità di misura o altro.
- Sono ammessi i commenti identificati con il carattere %

```
{0, 0}, {1., 0.946083},
{2., 1.60541}, {3., 1.84865},
{4., 1.7582}, {5., 1.54993}]
```

FIGURA 62: Esempio di file dati.

Secondo metodo: Con il `\dataplot` i dati possono essere inseriti come argomento del comando o passati dalla lettura di un file come nel caso precedente.

La sintassi è la seguente:

`\dataplot*[par]{miocom}`

I dati sono inseriti fra le parentesi quadre del comando:

`\savedata*{miocom}[dati]`

oppure letti da file con:

`\readdata*{miocom}{nomefile}`

Con `miocom` s’intende un nome generico scelto dall’utente che diviene a tutti gli effetti il comando `\miocom` col quale si richiamano i dati inseriti fra le parentesi quadre.

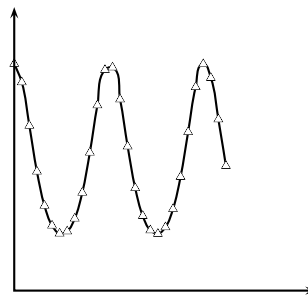


FIGURA 63: Esempio di grafico con `pst-plot`

————— Codice per la figura 63 —————

```
1 \begin{pspicture}(-1.5,0)(6,5)
2 \psset{xunit=.2cm,yunit=1.5cm}
3 \savedata{\ImieiDati}{%
4 {0,2},{0.5,1.837294076},
5 {1,1.454277218},{1.5,1.050253215},
6 {2,0.749423521},{2.5,0.573894075},
7 {3,0.503480394},{3.5,0.522514621},
8 {4,0.635672851},{4.5,0.864060479},
9 {5,1.217280953},{5.5,1.634296532},
10 {6,1.945539521},{6.5,1.967805494},
11 {7,1.686347963},{7.5,1.271591535},
12 {8,0.904065977},{8.5,0.65883454},
13 {9,0.531768321},{9.5,0.500981017},
14 {10,0.559003209},{10.5,0.719199076},
15 {11,1.00307237},{11.5,1.397942246},
16 {12,1.794838404},{12.5,1.996950095},
17 {13,1.875722987},{13.5,1.510389524},
18 {14,1.099415876}]
19 \dataplot[plotstyle=curve,showpoints=true,
20 dotstyle=triangle]{\ImieiDati}
21 \psline{<->}(0,2.5)(0,0)(20,0)
22 \end{pspicture}
```

Esiste un altro comando per disegnare un grafico a partire dalle coordinate:

`\listplot*[par]{lista}`

la sostanziale differenza con gli altri, è che in questo caso, la lista di coordinate deve essere separata solo degli spazi.

Per tracciare il grafico di una generica funzione $y = f(x)$ esiste il comando:

`\psplot*[par]{xmin}{xmax}{f(x)}`

La funzione $f(x)$ deve essere espressa in codice PostScript.

————— Codice per la figura 64 —————

```
1 \begin{pspicture}(0,-2)(5,2)
2 \psset{xunit=0.2pt}
3 \psplot[linecolor=red,plotstyle=curve]%
4 {0}{720}{x cos }
5 \psplot[plotpoints=100]{0}{720}{x sin }
6 \psline{<->}(0,-1.5)(0,1.5)
```

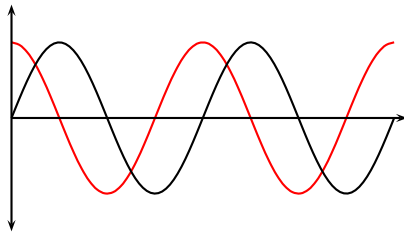


FIGURA 64: Esempio di grafico con `psplot`.

```
7 \psline{->}(745,0)
8 \end{pspicture}
```

È inoltre possibile disegnare i grafici di funzioni parametriche con il comando:

```
\parametricplot*[par]{x_min}{x_max}{x(t) y(t)}
```

Anche in questo caso la funzione parametrica deve essere definita tramite codice PostScript.

pst-poly

Consente di disegnare poligoni regolari e non.

pst-slpe

Questo pacchetto permette di generare gradienti di colore in modo efficiente superando le limitazioni del pacchetto `pst-grad`.

pst-stru

Consente di disegnare gli schemi tipici delle strutture semplificate usate nell'ambito della scienza delle costruzioni.

pst-text

Consente di manipolare le parole di un testo per ottenere svariati effetti grafici. Le parole seguono la curva che è costruita per interpolazione attraverso i punti indicati in coordinate cartesiane oppure tracciata con un comando standard. Nel primo esempio di figura 65 la curva è un cerchio creato con il comando standard `\pscircle`. Il parametro `linecolor=white` assegna il colore alla curva *guida*. Nel secondo esempio `linecolor=gray`.

pst-tree

Consente di generare strutture ad albero con diversi stili.

pst-uml

Consente di disegnare diagrammi in notazione UML.

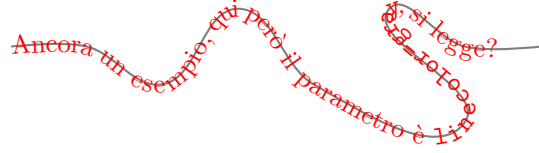
pst-vue3d

Consente di rappresentare e visualizzare oggetti in 3D.

bardiag

Consente di disegnare istogrammi.

```
1 \begin{pspicture}(0,0)(3,3)
2 \psset{linecolor=white}
3 \pstextpath
4 {\pscircle(1.5,1.5){1.3}}
5 {\color{blue}
6 Si parte! Vediamo cosa
7 si può fare con
8 PSTricks \ldots}
9 \end{pspicture}
```



```
1 \begin{pspicture}(0,-1)(7,.7)
2 \psset{linecolor=gray}
3 \pstextpath
4 {\pscurve(0,0)(1,0)(2,-.5)(3,.5)(4,-.5)(6,-1)
5 (5,.5)(6,0)(7,0)}
6 {\color{red}
7 Ancora un esempio, qui però il parametro è
8 \texttt{linecolor=gray}, si legge?}
9 \end{pspicture}
```

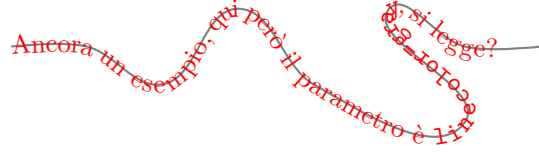


FIGURA 65: Esempio per `ps-text`.

dbicons

Permette di generare la struttura di data base (ER-diagrams).

euklides

Consente di generare figure di geometria piana.

gastex

Collezione di macro per generare disegni, diagrammi e figure di diverso tipo.

JasTeX

Interfaccia sviluppata in Java per GasTeX.

makeplot

Consente di disegnare grafici a partire da dati generati con Matlab.

multido

Macro utile per generare operazioni ripetitive.

psgo

Consente di disegnare diagrammi inerenti il gioco GO.

RRGtree

Alberi sintattici basati su RRG (Role and Reference Grammar).

spectrum

Consente di generare e gestire colori per produrre figure e testo con sfumature complesse.

uml

Un altro pacchetto per generare diagrammi UML.

vaucanson

Pacchetto per la generazione di grafi.

Sketch

Sistema per generare figure in 3D.

psMath

Sistema per generare figure bi e tridimensionali.

Riferimenti bibliografici

Internet è senz'altro la fonte principale dove reperire le informazioni su PSTricks. Nessun sito contiene tutto. Non tutti i pacchetti sono inseriti nelle distribuzioni ufficiali di T_EX e L^AT_EX.

URL <http://tug.org/PSTricks/main.cgi/>.

Questo sito contiene tutta la documentazione e gli aggiornamenti del pacchetto base e della maggior parte dei pacchetti correlati. Vi si trovano molti altri collegamenti ad altre risorse. Di seguito altri siti utili. <ftp://ftp.ctan.org/tex-archive/graphics/pstricks/>

<http://melusine.eu.org/syracuse/>
<http://members.aol.com/mluque5130/>
<http://latexdraw.sourceforge.net/index.html>.

CASCHILI, M. (2006). «Semplici figure con l'ambiente `picture`». *ArsT_EXnica*, (1), pp. 20–28. URL <http://www.guit.sssup.it/arstexnica.php>.

GOOSSENS, M., MITTELBACH, F., RAHTZ, S., ROEGEL, D. B. e VOSS, H. (2007). *The L^AT_EX Graphics Companion*. Tools and Techniques for Computer Typesetting. Addison-Wesley Professional, Reading, Massachusetts, 2^a edizione.

VOSS, H. (2007). *PSTricks: Grafik für T_EX und L^AT_EX*. Pubblicato nel febbraio del 2007. A breve è prevista l'uscita della traduzione in lingua inglese.

▷ Massimo Caschili
massimo.caschili@tiscali.it