

PSTricks



Using; plain-T&E X (pT&E X) system running

3 Jun 2010 12:31 p.m.

pTeX で pstricks のコマンドを用いて絵を描く コマンド集です. 若干ですが, 絵を描く為の基本書式を下記参照より抜粋して載せてあります. 原本は L^AT_EX2_ε 使用での解説書ですが, 此処では, T_EX 使用で行っております. 記述詳細は example のソース参照をお願いします. 絵を描くには, pstricks 提供のコマンドをどう使うかのテクニック如何で素晴らしい絵になるかどうか決まります.

色のパレットですが, pstricks(rgb-model) and cmyk-model 提供では 68 色程度なので, Gnu emacs の colors 提供である **tty-colors.el**(約 600 色以上) から T_EX で使えるように tty-colorsdvi.tex を作成して同梱してあります, ご使用エディタが emacs や medaow(emacs-japanes version) 等々なら M-x list-colors-display で 600 色以上を表示させながら選択して描く事が出来ます. この color 書式は例えば `\newrgbcolor{gray85}{0.85097 0.85097 0.85097}` 等々の RGB モデルになっております.

ここで, 色シーケンス例えば, "`\gray85`" 等々は daigram obje(図版) で宣言するには問題ないのですが, text-path obje や text 等々 string へ色付の為に用いるのには "`\csname gray85\endcsname`" の記述で用います, そうでないと, 文字列色付けには "`85`" や "`\red4`" 等々の "`4`" などのシーケンス数字部分が文字列と解釈され, color は gray や red と解釈されます. 文字列の obje に色シーケンスを用いるときには以上を考慮して次の如く, 使う位置で宣言してお使いください. 例えば,

```
{\csname gray85\endcsname Web2C-7.5.3 Kpathsearch 3.5.3}
```

この様にしますと, "Web2C-7.5.3 Kpathsearch 3.5.3" の string を望みの色で染める事が出来ます. もちろん, daigram obje(string less obje) でならストレートに "`\gray85`" と使う事が出来ます.

.....

この解説内容は pTeX(T_EX) での pstricks の解説です. L^AT_EX2_ε ではないのでご注意ください.

.....

文章の中の, "`\`" (エスケープ) は日本語キーボードでは半角 `¥` あるいは `\` で打ち込むことを意味しています.

文書作成: 吉田征夫

参照: L^AT_EX2_ε グラフィックスコンパニオン」 T_EX と PostScript による図解表現テクニック
発行所: 株式会社アスキー ISBN-4-7561-3461-0 2000 年 7 月 1 日 初版発行

PSTricks の基本的な描画コマンド.

コマンドに付けた”*(アクタリスク)”はその object の描線空間内を塗りつぶします. 付けなければ, アウトラインの描線のみです. **ここでは意識的にコマンドへ”*”をつけた記述をしています**

- `\parabola*[settings]{arrows}(x0,y0)(x1,y1)`
(x_0, y_0) を開始点として, 最大または最小の大きさを (x_1, y_1) とする放物線を描く.
- `\psarc*[settings]{arrows}(x,y){radius}{angleA}{angleB}`
 $angleA$ と $angleB$ の間に円のセグメントを描く—反時計回り—.
- `\psarcn*[settings]{arrows}(x,y){radius}{angleA}{angleB}`
psarc と同じだが, 円弧を時計回りの方向で描く.
- `\psbezier*[settings]{arrows}(x0,y0)(x1,y1)(x2,y2)(x3,y3)`
4つの制御ポイントでベジエ(Bézier) 曲線を描く.
- `\psccurve*[settings]{arrows}(x,y).....(xn,yn)`
ポイント間に閉じた曲線を描く.
- `\pscharclip*[settings]{text}.....\endpscharclip`^{¶1}
クリッピングパスを文字の形に設定する.
- `\pscharpath*[settings]{text}`^{¶2}
text を PSTricks の *linestyle* と *fillstyle* のパラメータに従わせる.
- `\pscircle*[settings](x0,y0){radius}`
(x_0, y_0) を中心に円を描く.
- `\pscirclebox*[settings]{text}`
text の周囲に円を描く
- `\pccoil*[settings]{arrows}(x0,y0)(x1,y1)`
(x_0, y_0) から (x_1, y_1) に 3D コイルを描く.
- `\psCoil*[settings]{angle1}{angle2}`
 $angle1$ から $angle2$ に水平にコイルを描く.
- `\pscurve*[settings]{arrows}(x1,y1).....(xn,yn)`
指定したポイントを通る開いた曲線を引く.
- `\psdblframebox*[settings]{text}`
text の周囲に二重ボックスを描く.
- `\psdiabox*[settings]{text}`
text の周囲にダイヤモンドを描く.
- `\psdiamond*[settings](x0,y0)(x1,y1)`
(x_0, y_0) を中心に, 幅を (x_1, y_1) の半分, 高さを (y_1) にしてダイヤモンドを描く.
- `\psdots*[settings](x1,y1)(x2,y2).....(xn,yn)`
それぞれの座標にドットを描く.
- `\psecure*[settings]{arrows}(x0,y0).....(xn,yn)`
開いた曲線を引くが, 最初と最後の点を省略する.
- `\psellipse*[settings](x0,y0)(x1,y1)`
(x_0, y_0) を中心に, 縦横の半径を (x_1) と (y_1) にして楕円を描く.

^{¶1} dvi を PostScript に変換するどのドライバでも動作するとは保証されていないマクロです.

^{¶2} String が PostScript の Type1 フォントを利用している場合にだけ有効.

- `\psframe*[settings](x0,y0)(x1,y1)`
(x_0, y_0) と (x_1, y_1) を隅にした長方形の枠を描く.
- `\psframebox*[settings]{text}`
text の周囲にボックスを描く.
- `\psgrid(x0,y0)(x1,y1)(x2,y2)`
(x_0, y_0) から始まって軸にラベルをつけ, (x_1, y_1) と (x_2, y_2) を隅としたグリッドを重ねて描画する. グリッドのデフォルトの大きさは *pspicture* で囲われる.
- `\psline*[settings]{arrows}(x0,y0)(x1,y1)(xn,yn)(x1,y1)`
指定した座標を通るように線を引き.
- `\psovalbox*[settings]{text}`
text の周囲に長方形を描く
- `\pspolygon*[settings]{arrows}(x1,y1)(x2,y2)(xn,yn)`
指定した座標を通るように線を引き. そのオブジェクトを塗りつぶせるようにパスを閉じる.
- `\psshadowbox*[settings]{text}`
text の周囲に影付きのボックスを描く.
- `\pstextpath[pos](x,y){graphics object}{text}`^{¶3}
graphicsobject で定義した線に沿って *text* を描く. *pos* には, パスに関連してテキストをどのように描くかを指定する. デフォルト (*l*) ではパスの先頭から開始される. (*c*) ではパスに沿ってテキストを中央揃えにし, (*r*) ではパスの最後でテキストが終わるようにする. また, (x, y) には, パスに関するオフセットを指定する. デフォルトで, テキストは線の上に $0.7ex$ だけオフセットされる.
- `\pstriangle*[settings](x0,y0)(x1,y1)`
(x_0, y_0) を底辺の中心として, 幅 (x_1, y_1), 高さ (y_1) の二等辺三角形を描く.
- `\pstribox*[settings]{text}`
text の周囲に三角形を描く.
- `\pswedge*[settings]{radius}{angle1}{angle2}`
angle1 と *angle2* の間にくさびのセグメントを引く (反時計回り).
- `\pszigzag*[settings]{arrows}(x0,y0)(x1,y1)`
(x_0, y_0) から (x_1, y_1) にジグザグを引く.
- `\qdisk(x,y){radius}`
オプション引数を指定しない *pscircle* の高速版.
- `\qline(x1,y1)(x2,y2)`
オプション引数を指定しない *psline* の高速版で. 単一の線のセグメントを引く.

^{¶3} dvi を PostScript に変換するどのドライバでも動作するとは保証されていないマクロです.

PSTricks の基本的なグラフィック用パラメータ.

パラメータ	デフォルト	説明
一 般		
<i>unit = dim</i>	1cm	基本的な PSTricks 単位の大きさ.
<i>xunit = dim</i>	1cm	<i>x</i> 軸の PSTricks 単位の大きさ.
<i>yunit = dim</i>	1cm	<i>y</i> 軸の PSTricks 単位の大きさ.
<i>runit = dim</i>	1cm	その他の PSTricks 単位の大きさ.
<i>showpoint = true/false</i>	<i>false</i>	<i>true</i> を設定すると、ページ曲線をいろいろ活用したい場合のために、ほとんどのオブジェクトの制御ポイントが表示される.
<i>origin = {coor}</i>	0pt, 0pt	後続するグラフィックオブジェクト用の座標系の原点.
<i>dimen = where</i>	<i>outer</i>	オブジェクトの何処から距離を測るか、値として <i>outer</i> , <i>inner</i> , <i>middle</i> が指定可能.
<i>swapaxes = true</i>	<i>false</i>	<i>x</i> 軸と <i>y</i> 軸を反転する
<i>linewidth = dim</i>	0.8pt	線の幅.
<i>linecolor = color</i>	<i>black</i>	線の色.
<i>fillcolor = color</i>	<i>white</i>	オブジェクトの塗りつぶしカラー.
<i>fillstyle = style</i>	<i>none</i>	その他のスタイルとしては、 <i>solid</i> , <i>vlines</i> , <i>vlines*</i> , <i>hlines</i> , <i>hlines*</i> , <i>crosshatch</i> , <i>crosshatch*</i> がある. *付きのバージョンは背景を塗りつぶす. <i>pst-grad</i> パッケージは、グラデーションでの塗りつぶし用に <i>gradient</i> という補足スタイルを追加する. 以下、次行の <i>grad</i> 関連キーが指定できる.
<i>gradbegin = color</i>		グラデーションによる塗りつぶしの開始と終了カラー.
<i>gradend = color</i>		中間地点のカラー.
<i>gradlines = int</i>	300	グラデーションでの塗りつぶしの線数. 線数が大きいほどグラデーションも細くなるが、印刷速度も遅くなる.
<i>gradmidpoint = num</i>	0.9	上端から下端までの距離の分数として中間点の位置. (<i>num</i> は 0 から 1 の間)
<i>gradangle = angle</i>	0	<i>angle</i> でグラデーションを回転.
<i>hatchwidth = dim</i>	0.8pt	塗りつぶし線の幅.
<i>hatchsep = dim</i>	4pt	塗りつぶし線どうしの空き.
<i>hatchcolor = color</i>	<i>black</i>	ハッチング線のカラー.
<i>hatchangle = angle</i>	45	ハッチング線の角度.
<i>arrows = style</i>	<i>none</i>	線のどちらかの端に任意の記号が出力できる、まったく出力しなくてもかまわない. 指定可能なスタイルは、 see- p. 13 - 「PSTricks の行末処理, 値とスタイル(グラフィック用パラメータ: <i>arrows</i> の値)」 に表付けしてあります.
線・曲線・ボックス		
<i>linestyle = style</i>	<i>solid</i>	他にも <i>dashed</i> , <i>dotted</i> , <i>none</i> が指定可能.
<i>dash = dim1 dim2</i>	5pt, 3pt	破線のモノクロのダッシュパターン.
<i>dotsep = dim</i>	3pt	ドット間の空き.
<i>border = dim</i>	0pt	線または曲線のどちらか側に、幅 (<i>dim</i>), カラー (<i>bordercolor</i>) の枠を描画. - 次行のパラメータ -
<i>bordercolor = color</i>	<i>white</i>	枠線のカラー.
<i>doubleline = true/false</i>	<i>false</i>	二重線として線を描画, 線どうしを <i>doublesep</i> で分け, 線の間には <i>bordercolor</i> のカラーをつける.
<i>doublesep = dim</i>		デフォルトは、1.25×現在の <i>linewidth</i> .
<i>doublecolor = color</i>	<i>white</i>	<i>doubleline</i> が <i>true</i> の場合に描画する線のカラー.
<i>shadow = true/false</i>	<i>false</i>	角度 (<i>shadowangle</i>), 深さ (<i>shadowsize</i>), カラー (<i>shadowcolor</i>) で影を描画.
<i>shadowsize = dim</i>	3pt	ズラして描画する影の深さ.
<i>shadowangle = angle</i>	-45	影を描画する角度.
<i>shadowcolor = color</i>	<i>darkgray</i>	影のカラー.
<i>linearc = dim</i>	0pt	ボックスまたは一連の線分の角に描かれる円弧の半径.

$framearc = dim$	0pt	$cornersize$ が” <i>relative</i> ”の場合、フレームボックスの角丸の半径を $num \times$ フレームの幅または高さに設定する。 num を 1 以上にはできない。 $cornersize$ が” <i>absolute</i> ”の場合には、 num を角丸の円弧の半径に設定する。
$cornersize = type$	<i>relative</i>	” <i>relative</i> ”と” <i>absolute</i> ”が指定可能(上記の $framearc$ と併用)
円 弧		
$arcsepA = dim$	0pt	これと次の 2 つのパラメータで、 $\backslash psarc$ の角度を調整して、その角度の方向に中心から伸びた dim 角度の線に触れるようにする。
$arcsepB = dim$	0pt	(上記の説明を参照)
$arcsep = dim$	0pt	(上記の説明を参照)
曲 線		
$curvature = abc$	1.10	a の値を小さくすると曲線がきつくなる。3 つのポイントの角度が 45° 以上の場合、 b の値を小さくすると曲線がきつくなる。それ以外の場合にはゆるくなる。 c の値で各ポイントの傾斜を決める。
テキストのフレーム		
$framesep = dim$	3pt	テキストとテキストを囲っているフレームとの空き。
$boxsep = true/false$	<i>true</i>	\TeX で生成したものにフレームの大きさそのものを含めるかどうか。
ドット		
$dotstyle = style$	*	指定可能なスタイルは、see- p.13 - 「PSTricks の dot スタイル. スタイルと効果」に表付けしてあります。
$dotsize = dim num$	2pt 2	円またはディスクの直径を、 dim に num を加えた値に現在の $linewidth$ を掛け合わせた値に設定。
$dotscale = num1 num2$	1	ドットを横方向に、 $num1$ 、縦方向に、 $num2$ だけ拡大/縮小。
$dotangle = angle$	0	ドットの回転角度。
矢 印		
$arrowsize = dim num$	1.5pt 2	矢印の頭の幅を、 dim に num を加えた値に現在の $linewidth$ を掛け合わせた値に設定。
$arrowlength = num$	4	矢印の頭の長さを幅の $arrowlength$ 倍に設定。
$arrowinset = num$	0.4	矢印の挿入位置を矢印の長さの分数 $arrowinset$ に設定。
$tbar size = dim num$	2pt 5	T バー、大括弧、小括弧の幅を、 $num \times linewidth$ に dim を加えた値に設定。
$arrowscale = num1 num2$	0.1	矢印の幅を $num1$ 、長さを $num2$ に拡大/縮小する。 $num2$ を省略した場合、矢印は両方向とも同じ大きさに拡大/縮小する。
ダイヤモンドと三角形		
$gangle = angle$	0	ダイヤモンドと三角形の回転。
グリッド		
$gridwidth = dim$	0.8pt	グリッド線の幅。
$gridcolor = color$	<i>black</i>	グリッド線の色。
$griddots = num$	0	num が正の場合、境界ごとのドット数を num にしてグリッド線を点線にする。
$gridlabels = dim$	10pt	グリッドに印を付けるのに用いる値の大きさ。
$gridlabelcolor = color$	<i>black</i>	ラベル値のカラー。
$subgriddvi = int$	5	グリッドの部分境界の数。
$subgridwidth = dim$	0.4pt	サブグリッドの線幅。
$subgridcolor = color$	<i>gray</i>	サブグリッドの線のカラー。
$subgriddots = num$	0	サブグリッドの境界ごとのドット数。
コイルとジグザグ		
$coilwidth = dim$	1cm	コイルまたはジグザグの直径。
$coilheight = num$	1	コイルまたはジグザグの長さは $coilwidth \times coilheight$ 。

<i>coilarm = dim</i>	0.5cm	コイルまたはジグザグの終わりの直線部分の長さ. 以下の2つのパラメータで個別に設定可能.
<i>coilarmA = dim</i>	0.5cm	(上の説明を参照)
<i>coilarmB = dim</i>	0.5cm	(上の説明を参照)
<i>coilaspect = angle</i>	45	3D オブジェクトの表示角度.
<i>coilinc = angle</i>	10	コイル曲線の滑らか度.

PS Tricks のノード描画コマンド.

コマンドに付けた”*(アクタリスク)”はその object の描線空間内を塗りつぶします. 付けなければ, アウトラインの描線のみです. **ここでは意識的にコマンドへ”*”をつけた記述をしています**

ノードの作成.

- `\rnode [refpoint] {name} {text}`
text で構成された *name* というノードを作成する. 結合子は *refpoint* を指す.
- `\Rnode (x,y) {name} {text}`
`\rnode` と同じだが, 参照点がボックスのベースラインの中央に (x,y) を加えた位置となる.
- `\pnode (x,y) {name}`
 (x,y) にノードを作成するが, 領域を取らない.
- `\dotnode* [settings] (x,y) {name} {text}`
`\psdots` と同じだが, ノードを作成する.
- `\fnode* [settings] (x,y) {name} {text}`
`\psframe` と同じだが, ノードを作成する.
- `\cnode* [settings] (x,y) {radius} {name}`
radius の円で構成されたノードを作成する.
- `\Cnode* [settings] (x,y) {name}`^{¶1}
グラフィックパラメータ *radius* で設定した半径を使って, 円で構成されたノードを作成する.
- `\cnodeput* [settings] {angle} (x,y) {name} {text}`
`\cput` と同じように, 円の中に *text* を配置したノードを作成する.
- `\circlenode* [settings] {name} {text}`
`\pscirclebox` と同じだが, ノードを作成する.
- `\ovalnode* [settings] {name} {text}`
`\psovalbox` と同じだが, ノードを作成する.
- `\dianode* [settings] {name} {text}`
`\psdiabox` と同じだが, ノードを作成する.
- `\trinode* [settings] {name} {text}`
`\pstribox` と同じだが, ノードを作成する.

ノードの結合.

- `\ncline* [settings] {arrows} {firstnode} {secondnode}`
ノード間に直線を描画する.
- `\ncLine* [settings] {arrows} {firstnode} {secondnode}`
ノード間に直線を描画するが, ノードの中央に線が来るようにラベルを配置する.
- `\ncarc* [settings] {arrows} {firstnode} {secondnode}`
ノード間に円弧を描画する.
- `\ncdiag* [settings] {arrows} {firstnode} {secondnode}`
arm と *angle* のパラメータを利用して, 各ノードから”アーム”を伸ばして線と結ぶ, コーナーの形はパラメータ *linearc* で制御する.
- `\ncdiagg* [settings] {arrows} {firstnode} {secondnode}`
`\ncnode` と同じだが, 2 番目のアームは描画しない.

^{¶1} 数多くの円の半径を設定するのに役立つ.

- `\ncbar*[settings]{arrows}{firstnode}{secondnode}`
ノードから角度 $angleA$ でアームを伸ばした線を描画する。アームの長さは必要に応じて調整される。
- `\ncangle*[settings]{arrows}{firstnode}{secondnode}`
`\ncdiag` のように結合を描くが、アーム A と結合線との角度は強制的に直角に設定される。
- `\ncangles*[settings]{arrows}{firstnode}{secondnode}`
`\ncangle` と同じだが、アーム A とアーム B を直角で結合する 2 つの線で結ぶ。
- `\ncloop*[settings]{arrows}{firstnode}{secondnode}`
`\ncangles` と同じだが、5 つの線分を利用する。2 番目と 4 番目の線分が、`loopsize` の長さとなる。
- `\nccurve*[settings]{arrows}{firstnode}{secondnode}`
パラメータ `ncurve` を使って制御ポイントの位置を決め、ノード間にベジェ曲線を描画する。
- `\nccircle*[settings]{arrows}{node}{radius}`
ノード自体に戻って結合する半径 $radius$ の円、または円の一部を描画する。

コイルとジグザグのノード結合子。これらの補足ノード結合を使うには、`pst-coil` パッケージを読み込んでおく。

- `\nccoil*[settings]{arrows}{firstnode}{secondnode}`
2 つのノードをコイルで結ぶ。
- `\nczigzag*[settings]{arrows}{firstnode}{secondnode}`
2 つのノードをジグザグで結ぶ。

PSTricks のノード結合へのラベル付けコマンド。

結合子の長さを基にしたラベル付け。

- `\ncput*[settings]{something}`
結合線上に `something` を配置する。
- `\naput*[settings]{something}`
結合線の上側に `something` を配置する。
- `\nbput*[settings]{something}`
結合線の下側に `something` を配置する。

ノード間の空きを基にしたラベルつけ。

- `\tvput*[settings]{something}`
ノード間の縦方向の空きに対して機能し、線の中央に `something` を配置する。
- `\tlput*[settings]{something}`
ノード間の縦方向の空きに対して機能し、線の左側に `something` を配置する。
- `\trput*[settings]{something}`
ノード間の縦方向の空きに対して機能し、線の右側に `something` を配置する。
- `\thput*[settings]{something}`
ノード間の横方向の空きに対して機能し、線の中央に `something` を配置する。
- `\taput*[settings]{something}`
ノード間の横方向の空きに対して機能し、線の上側に `something` を配置する。
- `\tbput*[settings]{something}`
ノード間の横方向の空きに対して機能し、線の下側に `something` を配置する。

ノードのラベルつけ。

- `\nput[settings]{angle}{name}{something}`
ノード `name` の横に `nodesep` だけ間を空けて、さらにノードの中央から $angle$ の方向に `something` を配置する。

PSTricks のノード結合用のグラフィックパラメータ.

パラメータ	デフォルト	説明
<i>offset = dim</i>	0pt	ノードへの線幅ポイントのオフセット.
<i>nodesep = dim</i>	0pt	結合線が引かれるノードの周囲の枠.
<i>nodesepA = dim</i>	0pt	最初のノードの周囲の枠.
<i>nodesepB = dim</i>	0pt	2 番目のノードの周囲の枠.
<i>arcangle = angle</i>	8	\ncarc の場合, ノード間に描画した直線と円弧の間の角度.
<i>angle = angle</i>	0	結合子をノードとつなげる角度.
<i>angleA = angle</i>	0	結合子を最初のノードとつなげる角度.
<i>angleB = angle</i>	0	結合子を 2 番目のノードとつなげる角度.
<i>arm = dim</i>	10pt	結合子とノードを結ぶ線分の長さ.
<i>armA = dim</i>	10pt	結合子と最初のノードを結ぶ線分の長さ.
<i>armB = dim</i>	10pt	結合子と 2 番目のノードを結ぶ線分の長さ.
<i>loopsize = dim</i>	1cm	\ncloop 用の線分の長さ.
<i>nccurv = num</i>	0.67	\nccurv でのベジェ曲線の制御ポイント間の空き. 値を小さくするとそれだけ曲線がきつくなる. ノードから最初の制御ポイントまでの距離は, <i>nccurv</i> の半分の値 × 2 つの端点の間の距離となる.
<i>nccurvA = num</i>	0.67	<i>nccurv</i> と同じだが, 最初のノード専用.
<i>nccurvB = num</i>	0.67	<i>nccurv</i> と同じだが, 2 番目のノード専用.
<i>boxsize = dim</i>	0.4cm	\ncbox と \ncarcbox のボックスの幅を半分にする.
ノードラベル用のパラメータ.		
<i>ref = ref</i>	<i>c</i>	ラベルの参照点を設定.
<i>nrot = rot</i>	0	ラベルのテキストを回転. 角度の前に : を記述すると, 結合線から計った値となる. 省略文字については, see- p. 13 - 「一般的な角度の PSTricks での省略文字.」を参照. :U は, 主にテキストを結合線上に揃えるのに利用する.
<i>npos = num</i>		結合線の長さに沿ってラベルを配置する位置. それぞれの結合線は 1 つか複数のセグメントを持ち, <i>npos</i> + 1 の値でラベルを設定するセグメントが決まる. デフォルト値については PSTricks のマニュアルを参照.
<i>shortput = type</i>	<i>none</i>	ラベル付け結合子用の省略名. 値として, <i>none</i> , <i>nab</i> , <i>tblr</i> , <i>tab</i> が指定省略可能. see- p. 13 - 「ノード結合用グラフィックパラメータで」を参照.
<i>tpos = num</i>	0.5	ラベルを結合子上に配置するノード間の距離の比率.
<i>mnode = type</i>	<i>none</i>	デフォルトのノードの種類 (マトリックス用). <i>R</i> (\Rnode), <i>r</i> (\rnode), <i>C</i> (\Cnode), <i>f</i> (\fnode), <i>p</i> (\pnode), <i>cirde</i> (\circlenode), <i>oval</i> (\ovalnode), <i>dis</i> (\dianode), <i>tri</i> (\trinode), <i>dot</i> (\dotode), <i>none</i> が指定可能.
<i>emnode = type</i>	<i>none</i>	マトリックス中で空のセル用に作成するノードの種類 (マトリックス用).
<i>name = name</i>		ノードの名前 (マトリックス用). これと類似のパラメータでセル中の大括弧に設定できる.
<i>nodealign = true/false</i>	<i>false</i>	ノードのベースラインをノードの中央から超えさせるかどうか (マトリックス用).
<i>mcol = l/r/c</i>	<i>c</i>	マトリックスのセル内でのノードの揃え (マトリックス用).
<i>mnodesize = dim</i>	-1pt	正の値を指定すると, ノードを強制的に指定したサイズにする (マトリックス用).
<i>rowsep = dim</i>	1.5cm	行間の空き (マトリックス用).
<i>colsep = dim</i>	1.5cm	カラム間の空き (マトリックス用).

PSTricks のノード結合子に相当する描画コマンド.

コマンドに付けた”*(アクタリスク)”はその object の描線空間内を塗りつぶします. 付けなければ, アウトラインの描線のみです. **ここでは意識的にコマンドへ”*”をつけた記述をしています.**

- `\pcline*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pccurve*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pcarc*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pcbar*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pcdiag*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pcangle*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pcloop*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pczigzag*[settings]{arrows}(x1,y1)(x2,y2)`
- `\pccoil*[settings]{arrows}(x1,y1)(x2,y2)`

PSTricks のツリー描画コマンド.

必要パッケージは, `pst-tree.sty`. コマンドに”*”を付けるとアウトライン描画でなく, その空間内をオブジェクト描画として塗りつぶされます.

- `\pstree{mode}{subtrees}`
結合するノードとサブツリーを描画.
- `\psTree\rootnode\subtrees \endpsTree`
`\pstree` の「環境」形式.
- `\Tn`
ヌルのツリーノード.
- `\Tspace{dim}`
次のレベルの直前を `dim` だけ空ける.
- `\TC*[settings]`
`\Cnode` ノードのようなツリーノード.
- `\TR*[settings]{something}`
`\Rnode` ノードのようなツリーノード.
- `\Tcircle*[settings]{something}`
`\circlenode` ノードのようなツリーノード.
- `\Tc*[settings]{dim}`
`\cnode` ノードのようなツリーノード.
- `\Tdia*[settings]{something}`
`\dianode` ノードのようなツリーノード.
- `\Tf*[settings]`
`\fnode` ノードのようなツリーノード.
- `\Tfan*[settings]`
祖先ノードの上隅に三角形を描画.
- `\Toval*[settings]{something}`
`\ovalnode` ノードのようなツリーノード.

- `\Tp*[settings]`
\pnode ノードのようなツリーノード.
- `\Tr*[settings]{something}`
\rnode ノードのようなツリーノード.
- `\Tri*[settings]`
\trinode ノードのようなツリーノード.
- `\skipllevel*[settings]nodes` または `subtrees`
指定したサブツリー全体のレベルを省略.
- `\skiplelevels*[settings]{n}node` または `subtrees`
n レベルをスキップ.

PSTricks のツリー用グラフィックパラメータ.

パラメータ	デフォルト	説 明
<i>bbd</i> = <i>dim</i>		下側バウンディングボックスを <i>dim</i> に設定.
<i>bbh</i> = <i>dim</i>		上側バウンディングボックスを <i>dim</i> に設定.
<i>bbl</i> = <i>dim</i>		左側バウンディングボックスを <i>dim</i> に設定.
<i>bbr</i> = <i>dim</i>		右側バウンディングボックスを <i>dim</i> に設定.
<i>edge</i> = <i>command</i>		ツリーノードを結ぶのに利用するノード結合.
<i>fansize</i> = <i>dim</i>	1cm	\Tfan ツリーノード用のベースサイズ.
<i>levelsep</i> = <i>dim</i>	2cm	ツリーの中で連続するレベル間の空き.
<i>showbbox</i> = <i>true/false</i>	<i>false</i>	点線のフレームでツリーを囲っている矩形を表示.
<i>thislevelsep</i> = <i>dim</i>		<i>levelsep</i> と同じだが, 現在のツリーにだけ適用.
<i>thistreesep</i> = <i>tight/loose</i>		<i>treefit</i> と同じだが, 現在のツリーにだけ適用.
<i>thistreenodesize</i> = <i>dim</i>		<i>treenodesize</i> と同じだが, 現在のツリーにだけ適用.
<i>thistreesep</i> = <i>dim</i>		<i>treesep</i> と同じだが, 現在のツリーにだけ適用.
<i>tndepth</i> = <i>dim</i>		ツリーノードのラベルの最小限の深さ.
<i>tnheight</i> = <i>dim</i>		ツリーノードのラベルの最小限の高さ.
<i>tnpos</i> = <i>l/r/a/b</i>	<i>b</i>	ノードからの相対的なツリーノードのラベル位置 (左/右/上/下).
<i>tnsep</i> = <i>dim</i>		ツリーノードのラベルとノードの間の空き (デフォルトは <i>labelsep</i> と同じ).
<i>treefit</i> = <i>tight/loose</i>	<i>tight</i>	" <i>tight</i> " の場合, <i>treesep</i> がすべてのレベルでノード間の最小限の空きとなる. " <i>loose</i> " の場合, <i>treesep</i> がサブツリーを囲っているバウンディングボックスどうしの空きとなる.
<i>treeflip</i> = <i>true/false</i>	<i>false</i>	ノードを反転させて鏡面画像を作成する.
<i>treemode</i> = <i>L/R/U/D</i>	<i>D</i>	ツリーを広げていく方向 (左/右/上/下).
<i>treenodes</i> = <i>dim</i>	0.75cm	ツリーの中で連続するノード間の空き.
<i>xbbd</i> = <i>dim</i>		下側バウンディングボックスを <i>dim</i> だけ増やす.
<i>xbbh</i> = <i>dim</i>		上側バウンディングボックスを <i>dim</i> だけ増やす.
<i>xbbl</i> = <i>dim</i>		左側バウンディングボックスを <i>dim</i> だけ増やす.
<i>xbbr</i> = <i>dim</i>		右側バウンディングボックスを <i>dim</i> だけ増やす.

PSTricks の 3D 用コマンド.

コマンドに付けた”*(アクタリスク)”はその object の描線空間内を塗りつぶします. 付けなければ, アウトラインの描線のみです. **ここでは意識的にコマンドへ”*”をつけた記述をしています.**

- `\psshadow*[settings]{text}`
text に影をつける.
- `\pstilt{degrees}{text}`
text をタイルして配置する.
- `\ThreeDput(x0,y0,z0){object}`
 (x_0) , (y_0) , (z_0) の座標に *object* を配置し, 現在の視点に従って出力する.

PSTricks の 3D グラフィック風パラメータ.

パラメータ	デフォルト	説明
<code>Tshadowsize = size</code>	1	影の長さ.
<code>Tshadowcolor = color</code>	<code>lightgray</code>	影の色.
<code>Tshadowangle = angle</code>	60	影の角度.
<code>viewpoint = xyz</code>	1-11	オブジェクト原点を見る人の位置.
<code>normal = xyz</code>	001	2D オブジェクトの平面上での直交するベクトル. 3D 空間での位置を指定.
<code>embedangle = angle</code>	0	オブジェクトの参照点を基準に位置ベクトルの方向に軸の周りに回転させる.

一般的な角度の **PSTricks** での省略文字.
ノード結合用グラフィックパラメータの値.

文字	度	文字	度
U	上 0	N	北 *0
L	左 90	W	西 *90
D	下 180	S	南 *180
R	右 270	E	東 *270
r	右 0	ur	右上 45
u	上 90	ul	左上 135
l	左 180	dl	左下 235
d	下 270	dr	右下 315

ノード結合用グラフィックパラメータで [shortput=nab] に設定すると,
`\naput` の代わりに `^` を, `\nbput` の代わりに `_` が利用できる.

ノード結合用グラフィックパラメータで [shortput=tab] に設定すると,
`^` は `\taput` を, `_` は `\tbput` を, `<` は `\tlput` を, `>` は `\trput` を表す.

PSTricks の行末処理, 値とスタイル (グラフィック用パラメータ : *arrows* の値)

<code>-</code>		<code>-></code>		<code><-</code>		なし
<code><-></code>		<code>-<</code>		<code>>-</code>		矢印
<code>>-<</code>		<code>->></code>		<code>>>-</code>		頭が逆向きの矢印
<code><<->></code>		<code>->>></code>		<code><<-</code>		二重矢印
<code>>>-<<</code>		<code>-<<<</code>		<code>>>-</code>		頭が逆向きの二重矢印
<code> _</code>		<code>- </code>		<code> _</code>		Tバー, 端点に寄せて出力
<code> * *</code>		<code>- *</code>		<code> *-</code>		Tバー, 端点为中心
<code>[_]</code>		<code>-]</code>		<code>[_</code>		大括弧
<code>(-)</code>		<code>-)</code>		<code>(-</code>		小括弧
<code>*-*</code>		<code>-*</code>		<code>*-</code>		ディスク, 端点为中心
<code>**-* **</code>		<code>-**</code>		<code>**-</code>		ディスク, 端点に寄せて出力
<code>c-c</code>		<code>-c</code>		<code>c-</code>		伸張, 端丸
<code>cc-cc</code>		<code>-cc</code>		<code>cc-</code>		端寄せ, 端丸
<code>C-C</code>		<code>-C</code>		<code>C-</code>		伸張, 端角
<code> <-> </code>		<code>-> </code>		<code> <-</code>		Tバーと矢印
<code> <*-> *</code>		<code>-> *</code>		<code> <*-</code>		Tバーと矢印, 端点に寄せて出力

○-○等々, ○○-○○等々がありますが, ○は日本語キーボードに割り当てがないので省略.

PSTricks の dot スタイル. スタイルと効果 (グラフィック用パラメータ : *dotstyle* の値)

<code>*</code>	●	<code>+</code>	+	<code> </code>		<code>x</code>	×
<code>asterisk</code>	*	<code>diamond*</code>	◆	<code>diamond</code>	◇	<code>oplus</code>	⊕
<code>otimes</code>	⊗	<code>pentagon*</code>	◆	<code>pentagon</code>	◇	<code>square*</code>	■
<code>square</code>	□	<code>triangle*</code>	▲	<code>triangle</code>	△		

○は日本語キーボードに割り当てがないので省略.

PSTricks の簡単な説明.

PSTricks のパッケージ, 及び T_EX 出力の dvi ファイル用デバイスである dvips 等々は, 角藤版配布の T_EX システムならすべて含まれて降ります. 新たに get するパッケージは皆無です.

ただ, DVI ファイルに埋め込まれた PostScript 言語を理解できて画面表示や印刷できるインタプリタが必要です. 通常それらは Ghostscript とその上で動作する GSwin32, GSview32(windows OS なら) という GUI です. 各 T_EX サイトの URL なら, どちらも get できる用にリンクされていますし, 日本語環境用にも対応されています. もちろん, T_EX 関連の市販本にはこれらのファイルを CD-ROM に収録されているのがあります.

また, 原書訳本としてアスキーより発刊されました「L^AT_EX2 ϵ グラフィックスコンパニオン」に載っているサンプルのソース集が T_EX の CTAN にリソースされています, 例えば, ftp.u-aizu.ac.jp の"/pub/tex/CTAN/info/lgc" です. PSTricks 部分に使用した描画のソース記述は 4-2-1.inl~ 4-10-11.ltx までの files です.

T_EX(pT_EX) で PSTricks の描画を描く場合の書式は, 一般的に次の宣言で始めます.^{¶1}

```
\input pstricks.tex
\input pst-grad.tex
\input pst-3d.tex
\input pst-node.tex
\input pst-tree.tex
\input pst-text.tex
\input pst-char.tex
\input pst-coil.tex
\pspicture(x0,y0)(x1,y1)
```

此処 (から) へ描画コマンドで記述始める.

```
\endpspicture
\vfill
\bye
```

ただ, この書式のみですと”colors”提供が *black, darkgray, gray, lightgray, white, red, green, blue, cyan, magenta, yellow* のみですので, 別途ファイルで対応します. 一般的には L^AT_EX2 ϵ 提供の CMYK モデルの 68 色ですが. この文書取得アーカイブファイルには RGB モデル約 600 色の”tty-colorsdvi.tex”を同梱してあります. T_EX システムの所定場所へ格納すれば”\input tty-colorsdvi.tex”追加記述で使用可能です.

Gnu Emacs か Meadow(日本語版 Win32 Emacs)のエディター使用ですと, この 600 色以上をディスプレイに表示しながら編集できます (emacs の tty-colors.el のクローンです).

\pspicture(x0,y0)(x1,y1) の宣言で, この場所に描画領域 (x0,y0)(x1,y1) で指定された長方形を確保する Postscript の空間 (通常これは T_EX 組版領域ではありません) です. 標準の寸法単位は 1cm です (任意な寸法に設定可能). ここで, \pspicture*(x0,y0)(x1,y1) と宣言すると, この長方形領域をはみ出す部分はクリッピング (カット) されます. 描画を (x0,y0)(x1,y1) 内にきっちりと収めることができます. また, \pspicture(x0,y0)(x1,y1)\psgrid とすると, (x0,y0)(x1,y1) の領域を方眼紙と x 軸 y 軸の寸法単位を T_EX は組版します, この寸法を見ながら描画すべき座標点を推測し絵を描いて (記述して) いけます. 描き終えたら \psgrid のコマンドだけを外せば良いわけです. 例えば, \pspicture(0,0)(10,16) と宣言すれば, その場所に横 10cm:縦 16cm の長方形描画空間を確保し, 以下その領域内へ描画を実施できます. \endpspicture でこの領域を終了します. もちろん, 終了宣言以前に描画コマンドは全て終了していなければなりません.

PSTricks は本質的に L^AT_EX2 ϵ のパッケージである事を忘れないでください.

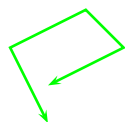
^{¶1} この他に, pst-plot.tex(データのプロット), pst-fill.tex(補足の psboxfill マクロ) 等々があります. この解説書では取り上げていません, 詳細は「L^AT_EX2 ϵ グラフィックスコンパニオン」の参照をお願いします.

描画コマンドの一般的なコマンドシンタックスは例えば以下の如くです。

```
\command[setting]{arrows/parameters}(coordinates)
```

例えば 1 :

```
\pspicture(0,0)(2,2)
\psline[linewidth=1pt,linecolor=green]{<->}%
(0.5,0)(0,1)(1,1.5)(1.5,1)(.5,.5)
\endpspicture
```



[setting] 内容は: key と value を対で記述します. 複数の key 記述はカンマで区切ります. いかなる場合でも, 明示がなければ半角空白の挿入は絶対に禁止です (T_EX の領域ではありません).

{arrows/parameters} 内容は: キーボードの記号キーで表現できる obje です.

(coordinates) 内容は: 複数個 (原則として) の座標がなければならない. 各々の座標点は (1,1.5) の如く (x 軸座標点,y 軸座標点) とカンマで軸を区別する.

例えば 2 :

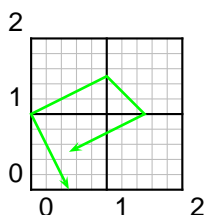
描画コマンドへ”*(アスタリスク)”をつけると, その描画空間を塗りつぶします.



```
\pspicture(0,0)(2,2)
\psline*[linewidth=1pt,linecolor=green]{<->}%
(0.5,0)(0,1)(1,1.5)(1.5,1)(.5,.5)
\endpspicture
```

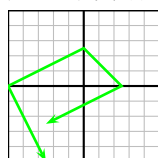
例えば 3 :

描画空間確保宣言に \psgrid をつけると. 方眼紙と単位を組版します.



```
\pspicture(0,0)(2,2)\psgrid
\psline[linewidth=1pt,linecolor=green]{<->}%
(0.5,0)(0,1)(1,1.5)(1.5,1)(.5,.5)
\endpspicture
```

描画空間確保宣言に”*”をつけると, 領域をはみ出す部分はクリッピング (カット) されます.

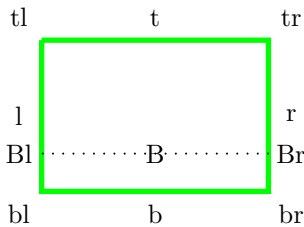


```
\pspicture*(0,0)(2,2)\psgrid
\psline[linewidth=1pt,linecolor=green]{<->}%
(0.5,0)(0,1)(1,1.5)(1.5,1)(.5,.5)
\endpspicture
```

PSTricks で何かの位置決め (必要なら回転) 行う為のコマンド.

`\rput*[settings][refpoint]{angle}(x0,y0){stuff}`

アスタリスク付きは *stuff* の周囲に `\psframe` を出力する. *angle* を指定すると, *stuff* を回転させる, *angle* の前に * を付けると, 外側での `\rput` の呼び出しの回転を全て取り消す. 引数の *refpoint* は *stuff* の参照点を指定するデフォルトではボックスの中央に揃えられる.



refpoint の値

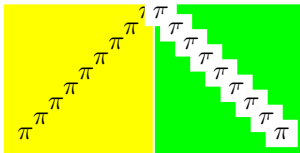
横		縦	
l	左	t	上
r	右	b	下
		B	ベースライン

PSTricks でオブジェクトを一定の間隔 (繰り返し) で配置するコマンド.

`\multirput*[refpoint]{angle}(x0,y0)(x1,y1){int}{stuff}`

stuff を (x0,y0) から開始して毎回 x1,y1 ずつ進んで *int* 個のコピーを作成する. ただし, このコマンドはデカルト座標系でしか動作しません.

例



```
\pspicture(0,0)(4,2)
\psframe[linestyle=none,fillstyle=solid,%
fillcolor=yellow](0,0)(2,2)
\multirput[lb](.2,.2)(.2,.2){9}{\pi}
\psframe[linestyle=none,fillstyle=solid,%
fillcolor=green](2,0)(4,2)
\multirput*[rb](2.2,1.8)(.2,-.2){9}{\pi}
\endpspicture
```

表示図は原本を変えてあります.

注意: (文字) ボックスを copy しています.

`\multips{angle}(x0,y0)(x1,y1){int}{graphics}`

これは, *graphics-object* に対してのみ対応です.

書式例: `\multips(1,1)(.4,.4){5}{\pscircle{.35}}` 等々.

PSTricks のカスタマイズとプログラミング.

自分専用のデフォルトパラメータを持ったオブジェクトを作成しておく事ができます.

`\newsobject{name}{object}{parl=value1,.....}`

提供されている *object* を基にさまざまなパラメータを前もって設定して, `\name` という新しいオブジェクトを作成できる.

`\newstyle{name}{parl=value1,.....}`

style キーを付けて, 何度も使うグラフィックパラメータを *name* へ一式関連付けておく.

例 *style* キーの設定記述

```
\newsstyle{bomb}{fillcolor=gray,fillstyle=solid,%
linestyle=dashed,linewidth=2pt}
```

使用は: コマンドの引数としいの大括弧の中に記述

```
\pswedge[style=bomb](4,.5)..... 等々
```

これ以上の詳細は「[L^AT_EX2_ε グラフィックスコンパニオン](#)」を参照をお願いします.

End.