

Apuntes de L^AT_EX

Capítulo 9: Dibujando con L^AT_EX

(Introducción al paquete PSTricks)

1 Introducción

Ya hemos visto el procedimiento para incluir gráficos postscript generados por un programa externo en nuestro documento, mediante el comando `\includegraphics`. Sin embargo, existe también la posibilidad de *crear* gráficos postscript directamente dentro del documento, utilizando una amplia variedad de paquetes y extras disponibles en L^AT_EX.

El entorno `picture` permite producir gráficos sencillos. Sin embargo, sus capacidades son bastante limitadas, con lo que realizar gráficos complicados es lento y difícil. A pesar de que existen un par de paquetes (`epic` y `eepic`) que introducen algunas mejoras, no los estudiaremos aquí y nos centraremos exclusivamente en la colección de paquetes **PSTricks**, mucho más avanzados. El estudiante curioso puede encontrar documentación extensa de `picture`, `epic` y `eepic` en los archivos `Picture.pdf`, `Epic.pdf` y `Eepic.pdf`.

En líneas generales, los paquetes de PSTricks permiten realizar las siguientes tareas:

- dibujar líneas, polígonos, círculos y curvas
- colocar, escalar y manipular objetos L^AT_EX
- realizar gráficas de funciones ó de listas de datos, incluyendo ejes etiquetados
- colorear líneas, letras y regiones
- realizar diagramas de nodos con conexiones (estructuras de árbol)
- crear ficheros postscript EPS con el compilador L^AT_EX

En definitiva, los comandos L^AT_EX proporcionados por el paquete PSTricks son un conjunto de macros que traducen los comandos propios del lenguaje PostScript.

En realidad, PSTricks es una colección de paquetes, algunos de ellos con propósitos bastante específicos; todos ellos pueden ser descargados, si nuestra distribución de L^AT_EX no nos incluye, de los servidores del CTAN:

(<http://www.ctan.org/tex-archive/graphics/pstricks>)

pst-all: Carga casi todos los paquetes de PSTricks y el paquete **pstcol**

pstricks: Contiene la mayor parte de los comandos de PSTricks.

pstcol: Se distribuye con **graphicx**; hace pstricks compatible con **graphicx** y **color**.

pst-fr3d: Para construir cajas tridimensionales.

pst-3d: Para hacer gráficos 3-D.

pst-gr3d: Para construir cuadrículas 3-D.

pst-char: Para colorear y rellenar caracteres de texto.

pst-circ: Para crear circuitos eléctricos.

pst-coil: Para dibujar líneas y colocar objetos en espirales ó en zig-zag.

pst-eps: Para exportar a ficheros PostScript EPS objetos PSTricks.

pst-fill: Para rellenar y colorear regiones arbitrarias.

pst-grad: Para graduar los colores de relleno.

pst-lens: Para dibujar lentes.

pst-node: Para definir nodos y conexiones entre ellos.

pst-osci: Para dibujar osciloscopios.

pst-plot: Para hacer gráficas 2-D con listas de datos ó gráficas de funciones.

pst-3dplot: Para hacer gráficas con listas de datos 3-D ó gráficas de funciones 3-D.

pst-poly: Para dibujar polígonos.

pst-text: Para escribir textos a lo largo de curvas.

pst-tree: Para realizar estructuras en árbol.

pst-vue3D: Para visualizar objetos 3-D.

La figura 1 muestra algunos ejemplos de los gráficos que pueden llegar a obtenerse con PSTricks. Información completa sobre todos los paquetes, documentación y ejemplos pueden encontrarse en la página: <http://www.tug.org/applications/PSTricks>

2 Nociones básicas

2.1 Entorno pspicture

Aunque los comandos de PSTricks funcionan en cualquier parte del documento, es muchas veces conveniente reservar una caja dentro de la cual se escribirá el dibujo. Esto se hace con el entorno:

```
\begin{pspicture}[Posición] (x0,y0) (x1,y1)
Lista de Objetos
\end{pspicture}
```

Posición es un parámetro optativo entre 0 y 1, que establece la posición relativa del dibujo y de la línea base (0 equivaldría a la línea base pasando por la parte de abajo, con 1 tendríamos la línea base pasando por encima del dibujo, y con 0.5 la línea base pasaría por el centro del dibujo). Los pares de coordenadas (x0,y0) y (x1,y1) establecen el sistema de coordenadas para la caja, con (x0,y0) representando la esquina inferior izquierda y (x1,y1) la esquina superior derecha. Por defecto la unidad de distancia es 1cm, pero esto puede cambiarse con:

`\psset{unit=NuevaUnidadDeDistancia}`¹

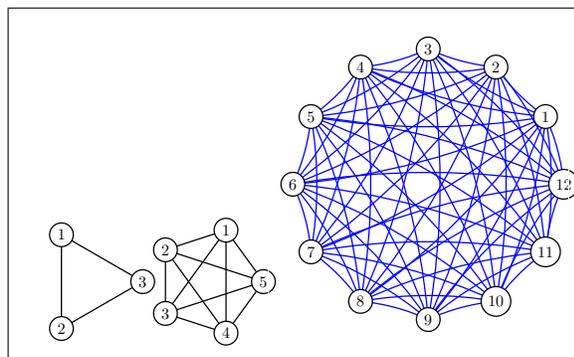
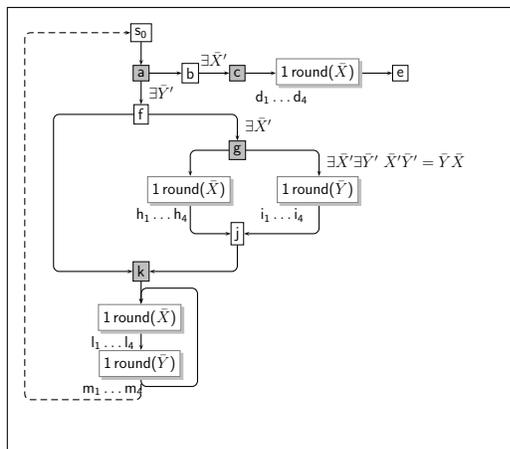
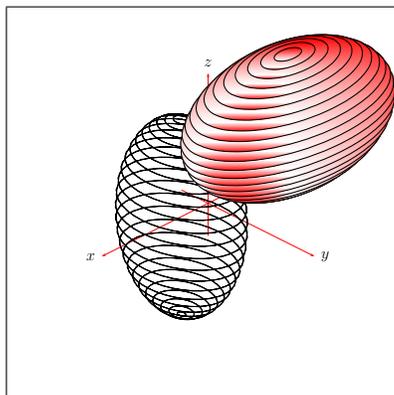
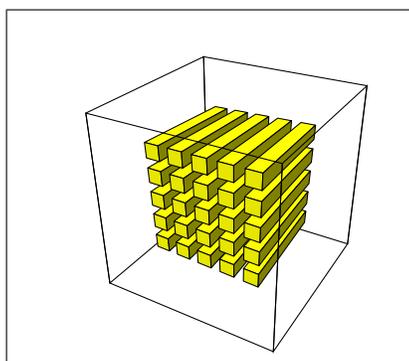


Figure 1: Ejemplos de figuras realizadas con PSTricks

La instrucción anterior es un ejemplo de funcionamiento del comando:

`\psset{Param1=valor1,Param2=valor2,...}`

que se utiliza para dar valores a los distintos parámetros de un gráfico (color, grosor de líneas, etc...)

¹Además del parámetro `unit`, podemos declarar por separado los parámetros `xunit` e `yunit`, cambiando por separado la escala para cada eje.

2.2 Coordenadas polares

La declaración `\SpecialCoor` nos permite utilizar un sistema de coordenadas polares, donde las posiciones se definen dando valores al par (r, θ) , siendo r la distancia al origen y θ el ángulo del vector de posición con el eje horizontal.

La unidad de medida de ángulos, grados por defecto, puede cambiarse a radianes con `\radians`

2.3 Colores

Utilizando el paquete `pst-all` disponemos de todos los colores del paquete `color`. Además, podemos definir nuevos colores con el comando:

```
\definecolor{Nombre}{Modelo}{Especificación}
```

donde los modelos son los mismos que los del paquete `color`.

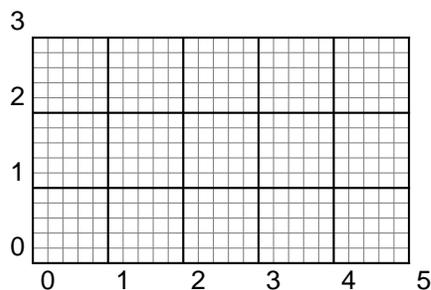
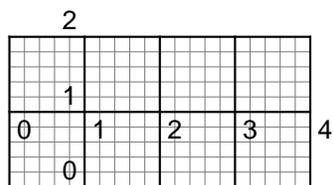
2.4 Cuadrículas

A la hora de escribir código para dibujar un gráfico, es muy útil incluir una cuadrícula para tener referencia visual de las coordenadas. Ello se consigue con el comando:

```
\psgrid[Parámetros](OrigenEjeX,OrigenEjeY)(x0,y0)(x1,y1)
```

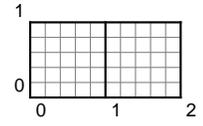
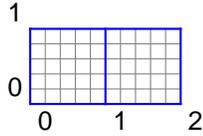
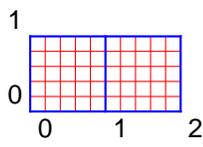
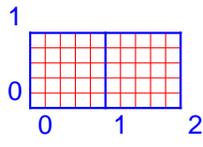
Donde el único argumento obligatorio son las coordenadas del vértice superior derecho $(x1,y1)$. El argumento $(x0,y0)$ representa las coordenadas del vértice inferior izquierdo, y $(OrigenEjeX,OrigenEjeY)$ indica el origen para los ejes etiquetados. Por ejemplo:

```
\begin{pspicture}(0,0)(5,3)
\psgrid(1,1)(0,0)(4,2)
\end{pspicture}
\hspace{3cm}
\begin{pspicture}(0,0)(5,3)
\psgrid
\end{pspicture}
```



Los `Parámetros` permiten modificar la apariencia de la cuadrícula, como se ve en los siguientes ejemplos:

Parámetro	Significado	Defecto	Ejemplo	
subgriddiv	Número de subdivisiones de la cuadrícula principal	5	<code>\psgrid[subgriddiv=1]%(0,0)(2,1)</code>	
gridwidth	Anchura de línea de la cuadrícula principal	0.8 pt	<code>\psgrid[gridwidth=2pt]%(0,0)(2,1)</code>	
subgridwidth	Anchura de las líneas de la subcuadrícula	0.4 pt	<code>\psgrid[gridwidth=2pt,%subgridwidth=1pt]%(0,0)(2,1)</code>	
griddots	Si el número es positivo, las líneas de la cuadrícula principal son punteadas, con "griddots" puntos por división	0	<code>\psgrid[griddots=10,%subgriddiv=1]%(0,0)(2,1)</code>	
subgriddots	Análogo al anterior, pero para la subcuadrícula	0	<code>\psgrid[subgriddots=5]%(0,0)(2,1)</code>	

Parámetro	Significado	Defecto	Ejemplo
gridlabels	Tamaño de fuente de las etiquetas de los ejes	10 pt	<code>\psgrid[gridlabels=7pt]%(0,0)(2,1)</code> 
gridcolor	Color de las líneas de la cuadrícula	Black	<code>\psgrid[gridcolor=blue]%(0,0)(2,1)</code> 
subgridcolor	Color de las líneas de la subcuadrícula	Black	<code>\psgrid[gridcolor=blue,%subgridcolor=red]%(0,0)(2,1)</code> 
gridlabelcolor	Color de las etiquetas de los ejes	Black	<code>\psgrid[gridcolor=blue,%subgridcolor=red,%gridlabelcolor=blue]%(0,0)(2,1)</code> 

3 Algunos objetos gráficos

3.1 Puntos

Para dibujar una lista de puntos (x_i, y_i) , utilizamos

```
\psdots[Parámetros](x_0,y_0)(x_1,y_1)(x_2,y_2)...
```

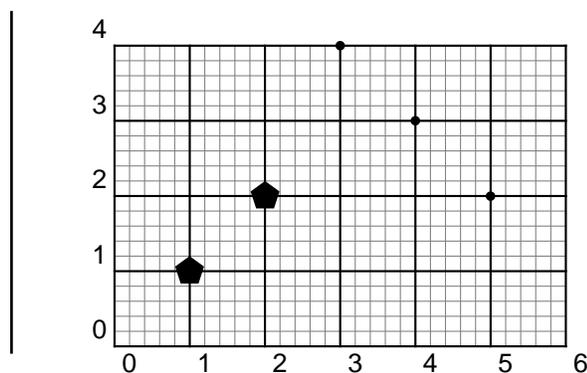
En el argumento `Parámetros` podemos asignar su tamaño (`dotsize`) y estilo (`dotsyle`); podemos también ajustar la orientación del punto con `dotangle`. Existen la siguientes opciones para `dotstyle`:

Estilo	Punto	Estilo	Punto	Estilo	Punto	Estilo	Punto
*	●			o	○	+	+
x	×	asterisk	*	diamond	◇	diamond*	◆
square	□	square*	■	oplus	⊕	otimes	⊗
pentagon	◊	pentagon*	◆	triangle	△	triangle*	▲

Nota: Para algunos estilos, se observa cómo la versión con asterisco (*) produce una figura rellena. Otros muchos comandos (todos los que delimitan curvas, por ejemplo) rellenan el área limitada por la curva (con el mismo color del contorno) cuando se emplea la versión con asterisco.

Ejemplo:

```
\begin{pspicture}(6,4)
\psgrid \psdots[dotstyle=pentagon*,%
dotsize=10pt](1,1)(2,2)
\psdots(3,4)(4,3)(5,2)
\end{pspicture}
```



3.2 Líneas rectas

Para dibujar líneas, se emplea el comando:

```
\psline[Parámetros]{TipoDeFlecha}(x0, y0)(x1, y1)...(xn, yn)
```

donde $(x_0, y_0)(x_1, y_1) \dots (x_n, y_n)$ es la lista de puntos que definen la poligonal. Para el argumento optativo TipoDeFlecha podemos elegir entre:

Estilo	Resultado	Estilo	Resultado	Estilo	Resultado
-	————	->	————>	<->	<————>
>-<	>————<	<<->>	<<————>>	[-]	[————]
- *	————	<->	<————>	(-)	(————)
-	————	<*-> *	<————>	c-c	————
-	●————●	o-o	○————○	cc-cc	————
-	●————●	oo-oo	○————○	C-C	————

En cuanto a los parámetros, podemos modificar los siguientes valores, que también son válidos para modificar propiedades de líneas curvas (al final, entre paréntesis, se dan los valores por defecto):

linewidth Grosor de curvas y líneas (0.8 pt)

linecolor Color de curvas y líneas (black)

linestyle Estilo de las líneas (**solid**)

solid → línea continua **none** → ausencia de línea

dashed → línea discontinua de segmentos de longitud *Long1* situados a una distancia *Long2*; éstas longitudes se cambian declarando el parámetro **dash=Long1 Long2** con el comando **\psset{...}**

dotted → línea de puntos, situados a una distancia **dotsep=Long** (también modificable con **\psset{...}**)

linearc Valor del radio de la circunferencia utilizada para suavizar ángulos de líneas poligonales (0)

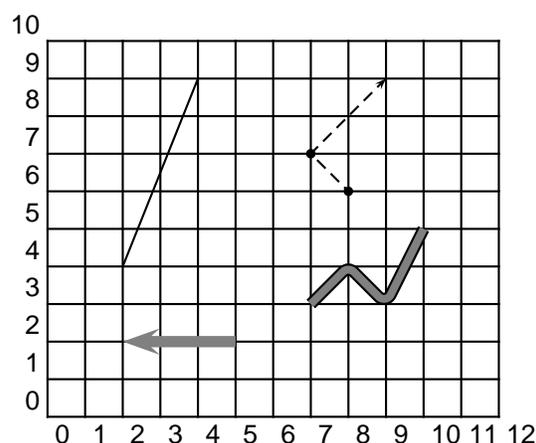
arrows Describe los extremos de líneas abiertas. Sirve como alternativa al argumento optativo **TipoDeFlecha** (-)

doubleline Si toma el valor “true” se dibujan líneas dobles, separadas por la distancia **doublesep**, y con color para la zona intermedia **doublecolor** (ambos parámetros modificables con **\psset**) (false)

showpoints Si toma el valor “true” se dibujan puntos en los ángulos intermedios de la línea (false)

Ejemplo:

```
\psset{unit=5mm,doublecolor=gray,%  
doublesep=3pt}  
\begin{pspicture}(12,10)  
\psgrid[subgriddiv=1]  
\psline(2,4)(4,9)  
\psline[linewidth=0.3,%  
linecolor=gray]{<-}(2,2)(5,2)  
\psline[linestyle=dashed,arrows=->]%  
(8,6)(7,7)(9,9)  
\psline[linearc=0.2,doubleline=true]%  
(7,3)(8,4)(9,3)(10,5)  
\end{pspicture}
```



3.3 Curvas

Podemos trazar diversos tipos de curvas con PSTricks; en ellas podemos utilizar los parámetros ya explicados para rectas. Para curvas cerradas, además se pueden especificar:

fillcolor Color de relleno (white)

fillstyle Estilo de relleno; los posibles estilos son `none`, `solid`, `vlines`, `vlines*`, `hlines`, `hlines*`, `crosshatch`, `crosshatch` y `gradient` (disponible en el paquete `pst-grad`, y para el cual hay que especificar los colores inicial (`gradbegin`) y final (`gradend`)). Para los estilos `vlines`, `hlines` y `crosshatch` se hace un relleno con líneas, cuyo ángulo, separación y color son modificables con `hatchangle`, `hatchsep` y `hatchcolor`, respectivamente. Las versiones con asterisco superponen el relleno de líneas con el relleno tipo “solid”. (`none`)

shadow Con el valor “true” añade una sombra al objeto, de tamaño `shadowsize`, a un ángulo `shadowangle` y en color `shadowcolor`. (`false`)

- **Curvas bezier:** Obtenidas por interpolación cúbica de cuatro puntos, mediante el comando: `\psbezier[Parámetros](x0, y0)(x1, y1)(x2, y2)(x3, y3)` Para este comando el parámetro `showpoints` muestra los puntos utilizados para calcular la curva

- **Curvas con “splines”:** Existen 3 variantes:

```
\pscurve[Parámetros](x0, y0)(x1, y1)...(xn, yn)
\psecurve[Parámetros](x0, y0)(x1, y1)...(xn, yn)
\psccurve[Parámetros](x0, y0)(x1, y1)...(xn, yn)
```

En la variante `\psecurve` sólo se traza la curva entre (x_1, y_1) y (x_{n-1}, y_{n-1}) , aunque se usan el primer y último puntos para calcularla. La variante `\psccurve` traza una curva cerrada enlazando el último punto con el primero

- **Representación gráfica de una serie de datos:** Disponemos de dos comandos para representar gráficamente una función a partir de datos externos:

```
\savedata{\VariableDatos}[Datos]
\readdata{\VariableDatos}{FicheroDatos}
```

Los dos crean una comando `\VariableDatos` donde se almacenan los datos. El primero de ellos lee los `Datos` directamente de una lista, y con el segundo los leemos de un archivo `FicheroDatos` que los contenga. Una vez cargados, se puede representar los datos con:

```
\dataplot[Parámetros]{\VariableDatos}
```

El estilo de la curva se especifica con el parámetro `plotstyle`, que puede tomar los

valores `line` (defecto), `dots`, `curve`, `ecurve` y `ccurve` (con el mismo significado que en el comando `pscurve`). También se puede declarar el origen de referencia de los datos con el parámetro `origin={ $x_0.y_0$ }`

Ejemplos:

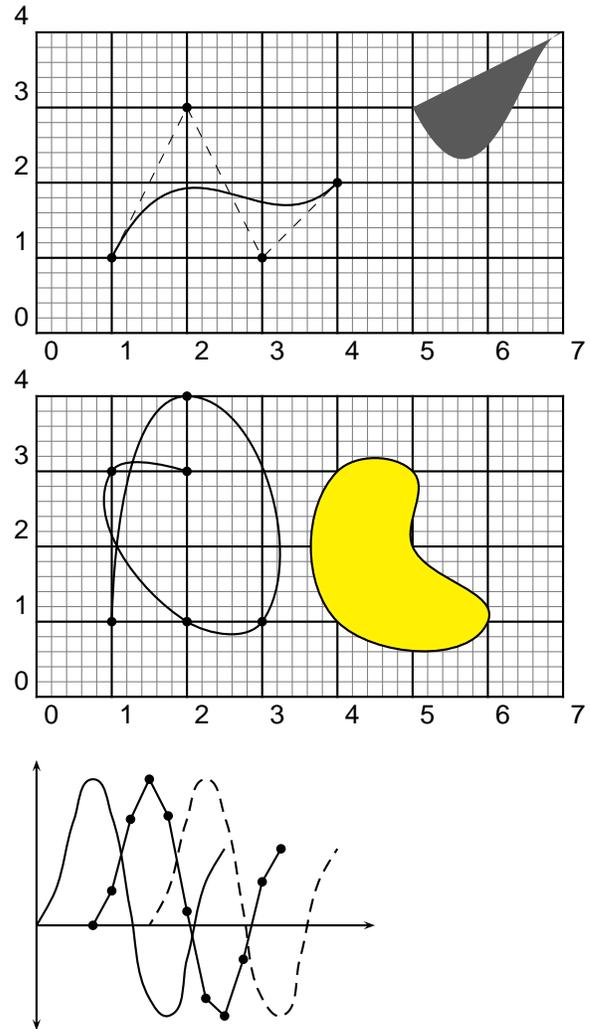
```

\begin{pspicture}(7,4)
\psgrid
\definecolor{migris}{gray}{0.35}
\psbezier[showpoints=true]%
(1,1)(2,3)(3,1)(4,2)
\psbezier*[linecolor=migris]%
(5,3)(6,1)(6.5,4)(7,4)
\end{pspicture}

\begin{pspicture}(7,4)
\psgrid \pscurve[showpoints=true]%
(1,1)(2,4)(3,1)(2,1)(1,3)(2,3)
\psccurve[fillstyle=solid,%
fillcolor=yellow]%
(4,1)(6,1)(5,2)(5,3)(4,3)
\end{pspicture}

\readdata{\misdatos}{bessel.txt}
\psset{xunit=0.25cm,yunit=4cm}
\begin{pspicture}(0,-0.35)(18,0.55)
\dataplot[plotstyle=curve]{\misdatos}
\psline{<->}(0,-0.35)(0,0.55)
\psline{->}(0,0)(18,0)
\dataplot[origin={-3,0},%
showpoints=true]{\misdatos}
\dataplot[origin={-6,0},%
plotstyle=curve,%
linestyle=dashed]{\misdatos}
\end{pspicture}

```



3.4 Polígonos, rectángulos, elipses, etc...

El comando `\pspoligon` simplifica la tarea de construir polígonos uniendo el primero y último de los vértices descritos. Su sintaxis es:

```
\pspoligon[Parámetros]( $x_0, y_0$ )( $x_1, y_1$ )...( $x_n, y_n$ )
```

Para rectángulos, se utiliza:

`\psframe[Parámetros](x0,y0)(x1,y1)` → (vértices inferior derecho y superior izquierdo)

Para elipses:

`\psellipse[Parámetros](x0,y0)(A,B)` → (centro y semiejes mayor y menor)

Para círculos, arcos de circunferencia y sectores circulares tenemos:

`\pscircle[Parámetros](x0,y0){Radio}`

`\psarc[Parámetros](x0,y0){Radio}{AnguloIni}{AnguloFin}`

`\pswedge[Parámetros](x0,y0){Radio}{AnguloIni}{AnguloFin}`

4 Colocando objetos

Podemos colocar cualquier objeto \LaTeX dentro de un entorno `pspicture` con los comandos:

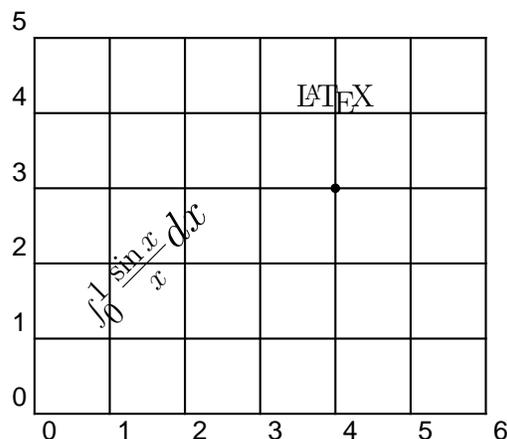
`\rput[Posición]{Rotación}(x,y){Objeto}`

`\uput{Separación}[Ángulo]{Rotación}(x,y){Objeto}`

que colocan `Objeto` en las coordenadas (x,y) . Con `\rput` podemos indicar también la posición con respecto al punto (x,y) , con una combinación de l,r,t,b (left, right, top, bottom), y rotarlo con un ángulo `Rotación`. El comando `\uput` permite indicar la distancia `Separación` entre el objeto y el punto (x,y) , en la dirección dada por `Ángulo`.

Ejemplo:

```
\begin{pspicture}(6,5)
\psgrid[subgriddiv=1]
\rput[b1]{45}(1,1){\Large%
 $\int_0^1 \frac{\sin x}{x} dx$ }
\psdots(4,3)
\uput{1}[90](4,3){\LaTeX}
\end{pspicture}
```



Cualquier objeto puede enmarcarse y adornarse con los siguientes comandos:

`\psframebox[Parámetros]{Objeto}`

`\psshadowbox[Parámetros]{Objeto}`

`\psdblframebox[Parámetros]{Objeto}`

`\psdiabox[Parámetros]{Objeto}`

`\pstribox[Parámetros]{Objeto}` `\pscirclebox[Parámetros]{Objeto}`
`\psovalbox[Parámetros]{Objeto}`

Para incluir varias copias de un mismo objeto tenemos la siguiente variante de `\rput`:

`\multirput[Posición]{Rotación}(x_0, y_0)(\Delta x, \Delta y){NumVeces}{Objeto}`

que repite `Objeto` el número de veces `NumVeces`, siendo $(\Delta x, \Delta y)$ el vector desplazamiento. `[Posición]` y `{Rotación}` (optativos) tienen el mismo significado que para `\rput`

Si el objeto a repetir está formado sólo por dibujos PSTricks, se puede utilizar el comando:

`\multips{Rotación}(x_0, y_0)(\Delta x, \Delta y){NumVeces}{Objeto}`

análogo a `\multirput`, que puede utilizar también coordenadas polares.

Una variante más sofisticada es cargar el paquete `multido`, que proporciona el comando:

`\multido{Variables}{NumVeces}{Acción}`

que permite repetir acciones en función de parámetros variables. Las variables pueden ser `\d` para una longitud, `\n` para un número (entero ó decimal), `\i` para un entero, y `\r` para un número decimal. Podemos declarar expresiones como `\n=Nini+Delta` (ojo! si delta es negativo, hay que escribir de todos modos: `\n=Nini+-0.1`, por ejemplo)

Ejemplo:

```
\psset{unit=2cm}
\begin{pspicture}(-1,-1)(1,1)
\psgrid
\multido{\r=1+-0.1666,%
\d=1.2+-0.4}{6}{%
\definecolor{gris}{gray}{\r}
\pscircle[fillstyle=solid,%
fillcolor=gris](0,0){\r}
\rput(1.8,\d){r=\r}
\end{pspicture}
```

