Introducción al paquete TikZ

José Miguel González Aguilera
jmgaguilera@gmail.com

14 de mayo de 2016

Resumen

Artículo introductorio a TikZ. Incluye una breve explicación sobre la sintaxis básica de funcionamiento, el dibujo de nodos y una visión preliminar de algunas de las librerías y paquetes incluidos o que se basan en TikZ.

1. Introducción

TikZ es un complejo lenguaje gráfico que permite *dibujar* en un documento LaTeX sin ninguna otra herramienta auxiliar.

IMPORTANTE: TikZ tiene un conflicto con el paquete `babel` cuando se trabaja en español, ya que ambos definen como caracteres activos "<" y ">": `babel` para el entrecomillado "multinivel", y `tikz` para las flechas que aparecen al final de una línea. Por ello, es necesario eliminar este conflicto, usando, por ejemplo:

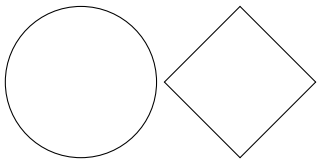
```
\usepackage[spanish,es-noshorthands]{babel}
```

Que suprime el uso "activo" de los caracteres "<" y ">" en babel.

TikZ define un conjunto de comandos y entornos LaTeX elaborar gráficos. Por ejemplo, el código:

```
\tikz \draw (0,0) circle (1cm);
\tikz \draw (-1,0) -- (0,1) -- (1,0) -- (0,-1) -- cycle;
```

Genera el siguiente dibujo:

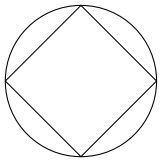


Las unidades de TikZ por defecto, son en centímetros. Por lo que en un caso se dibuja un círculo de un centímetro de diámetro centrado en el punto (0,0), y en el otro se genera un cuadrado de un centímetro de lado.

Se observa que el comando `\tikz` genera dibujos independientes entre sí. Para que ambos dibujos sean uno solo, generando una figura con el cuadrado y el círculo superpuestos, en lugar de usar el comando `\tikz`, es necesario incluir los comandos de dibujo en un entorno `\begin {tikzpicture}`, por ejemplo:

```
\begin{tikzpicture}
\draw (0,0) circle (1cm);
\draw (-1,0) -- (0,1) -- (1,0) -- (0,-1) -- cycle;
\end{tikzpicture}
```

Genera lo siguiente:



Las ventajas de usar `\tikz` en lugar de una herramienta externa son, entre otras:

1. Control completo sobre el diagrama.
2. Se usa el mismo tipo de letra que en el documento.
3. Capacidad de utilizar macros.
4. La alta calidad del resultado.

Como inconvenientes, cabe citar:

1. Elevada curva de aprendizaje.
2. No se “dibuja” sobre el propio “dibujo”.
3. El código puede ser complejo de entender.

Entre las ventajas que presenta TikZ frente a otros paquetes de LaTeX se encuentra su independencia del “driver” de impresión, sea postscript o no. Para ello, se definió un lenguaje independiente de representación de gráficos, *PGF*, Portable Graphics Format, que posteriormente, cada uno de los “drivers” existentes, convierten al formato destino elegido, sea postscript, pdf, ...

2. Estructura capas

TikZ es un lenguaje gráfico que dispone de una capa semántica avanzada que permite diseñar diagramas con relativa facilidad manteniendo en el lenguaje la relación entre los elementos del diagrama. Para conseguirlo, se sustenta en dos capas que pueden utilizarse de forma independiente. En la figura 1 se puede observar la relación entre ellas.

Figura 1: Estructura en capas de TikZ

Capa del sistema. Esta primera capa es la API básica de PGF que abstrae las diferencias entre los diversos "drivers". Cada uno de los elementos de esta API se tiene que implementar de forma diferente para cada "driver", por lo tanto, resulta necesario mantenerla lo más simple posible, a fin de simplificar al máximo el desarrollo y mantenimiento de aquellos.

Capa básica. Añade semántica que facilita el diseño de gráficos. Esta capa se puede extender con paquetes adicionales.

TikZ. Se trata de una API avanzada, que hace de "frontal" sobre la capa básica simplificando y extendiendo su sintaxis. Su sintaxis se basa en la de METAFONT y PSTRICKS. Sobre este último presenta la ventaja de ser "portable" entre "drivers". Lo que, actualmente, se traduce en la generación directa de documentos PDF.

Además de lo visto, PGF dispone de algunos paquetes de utilidades como: `pgfpages`, para "jugar" con las páginas de un documento componiendo varias de ellas en una única física; `pgfkeys`, para la gestión de parejas de parámetros, clave-valor; `pgfcalendar`, para la generación de calendarios con LaTeX, o `pgffor`, que define el comando `foreach`.

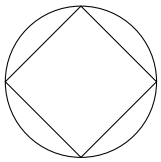
3. Dibujo básico

Normalmente, un gráfico TikZ se compone utilizando el entorno `tikzpicture`, como se vio en el ejemplo inicial[1].

Un elemento fundamental en TikZ es el *camino*[2]. Existen diversos comandos, como `\draw`, `\filldraw` o `\path` que se basan en la idea de encadenar posiciones y unirlas mediante otros elementos.

En el ejemplo anterior:

[1] El círculo con el cuadrado inscrito.
[2] *Path*, en inglés.



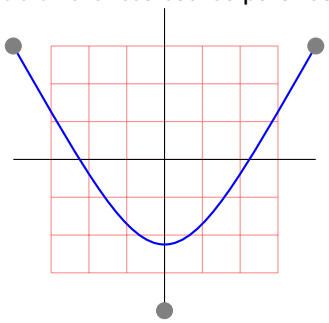
```
\begin{tikzpicture}
\draw (0,0) circle (1cm);
\draw (-1,0) -- (0,1) -- (1,0) -- (0,-1) -- cycle;
\end{tikzpicture}
```

- `\draw` indica que se debe "dibujar" el camino a realizar. Cada comando `\draw` inicia un camino completo nuevo.
- El camino se expresa por una sucesión de puntos y "operaciones" que expresan lo que hay que hacer entre cada punto.
- "(-1,0)" indica que se debe posicionar en dicho punto. En centímetros por defecto.
- Con el comando `draw`, la operación "--" indica que se debe dibujar una línea recta entre el punto anterior y el siguiente.
- Con el comando `draw`, la operación "circle" indica que se debe dibujar un círculo con el radio indicado en el siguiente parámetro, en este caso "(1cm)".
- "cycle" indica que se vaya al punto de inicio del camino actual.

Existen diversas operaciones para recorrer un camino: `rectangle`, `grid`, `ellipse`, son algunas de las factibles. Cada una de ellas necesita unos parámetros determinados para su correcto dibujo.

4. Estilos

Los diferentes comandos tienen parámetros opcionales que facilitan el diseño de diferentes estilos para los gráficos. Por ejemplo:



```
\draw[step=0.5,color=red!50!white, very thin] (-1.5,-1.5)
     grid (1.5,1.5);
\draw (-2,0) -- (2,0);
\draw (0,-2) -- (0,2);
\draw[thick, color=blue] (-2,1.5) .. controls (0,-2) ..
↪  (2,1.5);
```

```
\filldraw[color=gray] (-2,1.5) circle (3pt); % inicio de la
↪ curva
\filldraw[color=gray] (2,1.5) circle (3pt); % fin de la curva
\filldraw[color=gray] (0,-2) circle (3pt); % punto de control
```

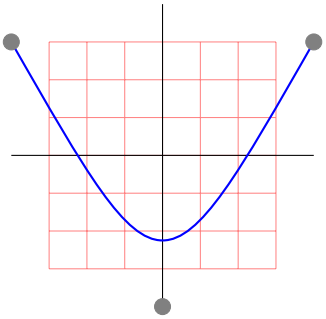
Se observa que:

- El primer comando tiene unos parámetros envueltos entre corchetes. Estos parámetros permiten matizar el funcionamiento de las operaciones a realizar en el comando. En concreto, están indicando:
  - `step`, que al dibujar el `grid` la malla tenga un grosor de medio centímetro.
  - `color`, expresa, en formato `xcolor`, que debe dibujar en color rojo claro.
  - `very thin`, expresa el grosor de las líneas que se dibujen en este comando, en este caso concreto, la malla a pintar.
- Los dos siguientes comandos `\draw` dibujan los ejes horizontales con el estilo por defecto (líneas negras de grosor `thin`).
- El siguiente comando `\draw` dibuja una línea desde el punto `(-2,1.5)` al punto `(2,1.5)`, pero en lugar de ser recta, la operación `.. controls (0,-2) ..`, establece que se dibuje la línea utilizando este punto de control como elemento de referencia para trazarla curva. Lo que hace `tikz` es "salir" del punto inicial de forma paralela a la recta imaginaria que llegaría al punto de control, para después girar de tal modo que llegue al punto de destino de forma paralela a la recta imaginaria que va del punto de destino al punto de control.

  La sintaxis de esta operación permite utilizar dos puntos de control diferentes para el origen y el destino, así tanto el ángulo como la distancia a dichos puntos definen la forma de la curva a trazar.

  En este comando también se ha especificado un estilo entre corchetes, línea gruesa y de color azul.
- Los tres últimos comandos `\filldraw` dibujan con relleno: se posicionan en un punto; después, la operación a ejecutar es `circle`, con un radio de 3pt, que en este caso se rellenan del mismo color que se ha indicado, `gray`, gris.

Con estas herramientas se pueden realizar dibujos muy complejos. Para facilitar al máximo su desarrollo, `tikz` permite, además, darle nombre a los estilos y reutilizarlos tantas veces como se necesite. Por ejemplo, el dibujo anterior se puede reescribir de la siguiente forma:



```
\begin{tikzpicture}[
migrid/.style ={step=0.5,color=red!50!white, very thin},
micurva/.style ={thick, color=blue},
mipunto/.style ={color=gray}]
\draw[migrid] (-1.5,-1.5) grid (1.5,1.5);
\draw (-2,0) -- (2,0);
\draw (0,-2) -- (0,2);
\draw[micurva] (-2,1.5) .. controls (0,-2) .. (2,1.5);
\filldraw[mipunto] (-2,1.5) circle (3pt); % inicio de la curva
\filldraw[mipunto] (2,1.5) circle (3pt); % fin de la curva
\filldraw[mipunto] (0,-2) circle (3pt); % punto de control
\end{tikzpicture}
```

- `migrid/.style` define un estilo denominado `migrid` que se puede utilizar entre corchetes en los diferentes comandos para aplicar los parámetros que definen el estilo.
- Después del signo =, entre llaves, se describen los parámetros que definen el estilo.
- Esto permite la reutilización como se observa en los tres comandos `\filldraw`.

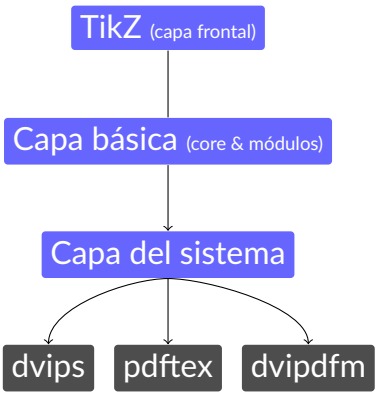
5. Nodos

Los nodos facilitan la generación de diagramas que contengan textos o que requieran mostrar relaciones entre elementos, entre otras cosas.

A continuación se muestra un ejemplo de uso de los nodos que utiliza algunas librerías adicionales para facilitar su diseño.

- `fit`, para calcular el tamaño de los nodos.
- `positioning`, para colocar los nodos en posiciones relativas a otros.

Para utilizar estas librerías auxiliares de TikZ se utiliza el comando: `\usetikzlibrary {fit,positioning}`.

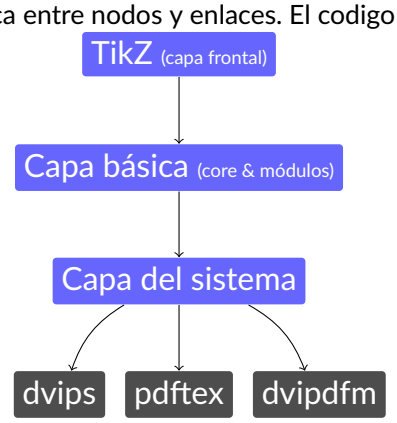



```
\begin{tikzpicture}[node distance= 1.5cm, auto,
% Estilos
every node/.style={text=white, rounded corners=0.05cm},
grande/.style={rectangle,  fill=blue!60!white, font=\large},
peque/.style={rectangle, fill=white!30!black, font=\large}]

% nodos
\node[grande](sistema){Capa del sistema};
\node[grande, above of=sistema](basico){Capa básica \scriptsize
↪  (core \& módulos)};
\node[grande, above of=basico](tikz){TikZ \scriptsize (capa
↪  frontal)};
\node[peque, below of=sistema](pdftex){pdftex};
\node[peque, left=0.25cm of pdftex](dvips){dvips};
\node[peque, right=0.25cm of pdftex](dvipdfm){dvipdfm};
%flechas entre nodos
\draw[->] (tikz.south) -- (basico.north) (basico.south) --
↪  (sistema.north);
\draw[->] (sistema.south) .. controls +(left:0.5cm) and
↪  +(up:0.5cm) .. (dvips.north);
\draw[->] (sistema.south) .. controls +(right:0.5cm) and
↪  +(up:0.5cm) .. (dvipdfm.north);
\draw[->] (sistema.south) -- (pdftex.north);

\end{tikzpicture}
```

- Las primeras líneas establecen los estilos a utilizar: El texto será de color blanco para todos los nodos y las esquinas redondeadas; los nodos con el estilo “grande” utilizan un color de relleno diferente a los pequeños, el tipo de letra será grande y deberán tener forma de “rectangle”.
- En cada comando `\node` se observa, entre paréntesis, el “nombre” que tendrá el nodo, seguido de su contenido entre llaves. El “nombre” se utiliza para referenciar al nodo desde otro lugar. En este caso concreto, para establecer la posición relativa entre ellos.
- Entre los parámetros de cada nodo se incluye el estilo y la posición relativa a otro nodo: “above of”, “below of”, “left” y “right”. Se observa que se puede incluir la distancia relativa de separación entre ellos.

- Para dibujar las líneas que conectan los nodos, se utiliza `draw`. El parámetro especifica el tipo de línea a dibujar, en este caso con una flecha al final (pero no al comienzo) del camino.
- Si se desease una línea con flecha tanto al comienzo como al final del camino se debería utilizar "<->". Para expresar una flecha solo al comienzo, se utilizar "<-".
- Para indicar los lugares de origen y de destino de la línea, se indican los nombres de los respectivos nodos, seguidos de un punto y del modificador que expresa el punto del que debe salir: "north", desde el borde superior; "south", desde el inferior; "east", desde el borde derecho, y "west", desde el borde izquierdo. Si no se indican los puntos de inicio y fin de la línea, TikZ busca los más próximos entre sí.
- Para expresar una línea curva se utiliza `controls` como se vio anteriormente. En este caso se indican dos puntos de control. Ambos, precedidos del símbolo "+", que sirve para expresar coordenadas relativas, a los puntos de salida y llegada respectivamente.
- Cada punto de control, además de ser relativo, indica posición, no mediante un punto (x,y) sino mediante un desplazamiento a través de un eje. Este se expresa con el texto "left" o "right", en este caso.

TikZ es tan potente que permite expresar el mismo dibujo de diversas maneras. En este caso se puede hacer uso de la operación `edge` que facilita la unión entre nodos, sin necesidad de indicar `draw`. De este modo se captura la semántica entre nodos y enlaces. El codigo queda así:




```
\begin{tikzpicture}[node distance= 1.5cm, auto,
% Estilos
every node/.style={text=white, rounded corners=0.05cm},
grande/.style={rectangle,  fill=blue!60!white, font=\large},
peque/.style={rectangle, fill=white!30!black, font=\large}]

% nodos
\node[grande](sistema){Capa del sistema}
    edge[->, bend right=20] (dvips)
    edge[->] (pdftex)
    edge[->, bend left=20] (dvipdfm);
```

```
\node[grande, above of=sistema](basico){Capa básica \scriptsize
↪ (core \& módulos)}
    edge[->] (sistema);
\node[grande, above of=basico](tikz){TikZ \scriptsize (capa
↪ frontal)}
    edge[->] (basico);
\node[peque, below of=sistema](pdftex){pdftex};
\node[peque, left=0.25cm of pdftex](dvips){dvips};
\node[peque, right=0.25cm of pdftex](dvipdfm){dvipdfm};
\end{tikzpicture}
```

Como se observa, la operación `edge`, en el comando `\node`, facilita el dibujo del enlace entre el nodo que se está definiendo y otro que se indica por su nombre. Además, es posible expresar el "grado de curvatura" del enlace y TikZ se encargará de posicionar la curva en el mejor lugar posible.

6. Otras librerías

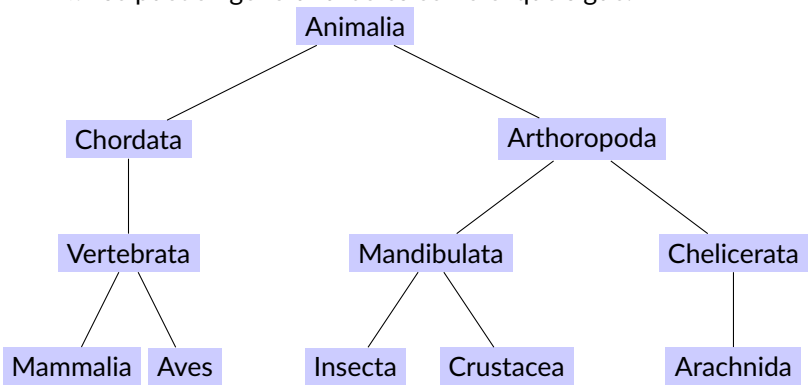
Como se ve, TikZ es un paquete muy potente. Con lo visto anteriormente se ha mostrado de forma superficial parte de esta potencia. Sin embargo, existen muchos parámetros, operaciones y comandos que no se han mostrado.

Además, existen muchas librerías que proporcionan la semántica apropiada para dibujar determinados tipos de diagramas. Por ejemplo, grafos, redes de petri, capas, mapas mentales, calendario, gráficos de funciones, etc.

A continuación se muestran algunos ejemplos en los que se observará cómo la semántica de TikZ describe perfectamente la estructura del diagrama correspondiente:

6.1. Árboles

En TikZ se pueden generar árboles como el que sigue:



El código que lo genera es el siguiente:

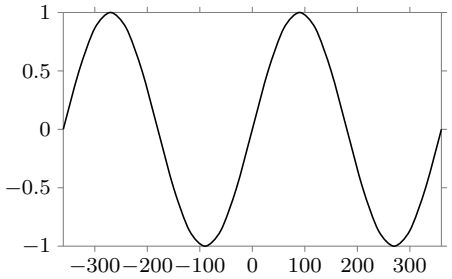
```
\begin{tikzpicture}[every node/.style={rectangle,
↪ fill=blue!20!white}]
\node {Animalia} [sibling distance=6cm]
child {node {Chordata}
    child {node {Vertebrata} [sibling distance=1.5cm]
```

```
        child {node {Mammalia}}
        child {node {Aves}}
    }
}
child {node {Arthoropoda} [sibling distance=4cm]
    child {node {Mandibulata}[sibling distance=2cm]
        child {node {Insecta}}
        child {node {Crustacea}}
    }
    child {node {Chelicerata}
        child {node {Arachnida}}
    }
};
\end{tikzpicture}
```

Los árboles se pueden dibujar también en horizontal. Existe, además, una librería, `mindmap`, de Ti*k*Z que facilita el diseño de árboles con un diseño más aproximado para los mapas mentales.

6.2. Gráficos

En Ti*k*Z se pueden generar gráficos como el que sigue:



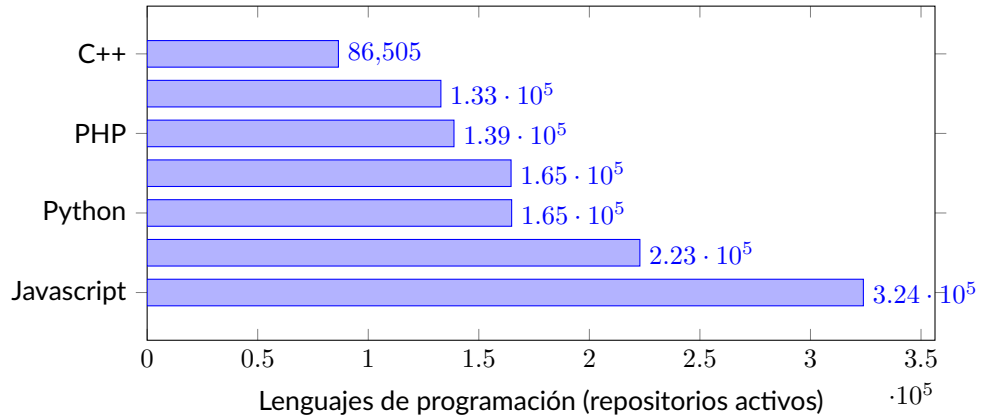
El código es el que sigue:

```
\begin{tikzpicture}
\datavisualization [scientific axes, visualize as smooth line]
data [format=function] {
    var x : interval [-360:360];
    func y = sin(\value x);
};
\end{tikzpicture}
```

Para diagramas más avanzados, como por ejemplo, un diagrama de barras, se puede utilizar el paquete `pgfplots` que se basa en `pgf/tikz` y simplifica su construcción.

Para ello es necesario incluir en el preámbulo: `\usepackage {pgfplots}`, que añade la librería `pgfplots`.

Por ejemplo:



Se obtiene con este código:

```
\begin{tikzpicture}
\begin{axis}[
xbar, xmin=0,
width=12cm, height=6cm, enlarge y limits=0.2,
xlabel={Lenguajes de programación (repositorios activos)},
symbolic y coords={Javascript, Java, Python, CSS, PHP, Ruby,
↪ C++},
nodes near coords, nodes near coords align={horizontal},
]
\addplot coordinates {(323928,Javascript)
    (222852,Java)
    (164852,Python)
    (164585,CSS)
    (138771,PHP)
    (132848,Ruby)
    (86505,C++)};
\end{axis}
\end{tikzpicture}
```

6.3. Calendarios

Existe un paquete basado en TikZ que permite gestionar días, días de la semana, meses y años, y dibujarlos de forma sencilla. Para utilizarlo, se incluye `\usetikzlibrary {calendar}`. Por ejemplo:

Abril

			1	2	3	
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Mayo

						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Se genera con el siguiente código:

```
\begin{tikzpicture} [column sep=1cm]
\matrix {
\calendar[dates=2016-04-01 to 2016-04-last,week list, month
↪ label above centered];
&
\calendar[dates=2016-05-01 to 2016-05-last,week list, month
↪ label above centered]; \\
};
\end{tikzpicture}
```

Para que el texto de los meses salga en español es necesario incluir en el preámbulo, lo siguiente: `\usepackage [spanish]{translator}`. Justo después de cargar el paquete babel.

7. Referencias

1. Página del proyecto TikZ en CTAN.
2. Introducción simple a TikZ: A very minimal introduction to TikZ.
3. Página del paquete pgfplots.