

## Tipografía: Visualización

### Elementos que se introducen en esta Unidad:

PFont, loadFont(), textFont(), text()  
 textSize(), textLeading(), textAlign(), textWidth()

Las letras en una pantalla son creadas por establecer el color de los píxeles en una determinada ubicación. Por lo tanto, la calidad de la tipografía se ve limitada por la resolución de la pantalla. Como las pantallas tienen una resolución baja en comparación con el papel, se han desarrollado técnicas para mejorar la apariencia del texto en la misma. Las fuentes de las primeras computadoras de Apple Macintosh estaban compuestas por pequeñas imágenes de mapa de bits, creadas específicamente en tamaños como de 10, 12, y 24 puntos. Utilizando esta tecnología, una variación de cada fuente fue diseñada para cada tamaño de un tipo de letra en particular. Por ejemplo, el carácter de un tipo de letra en el San Francisco, utiliza una imagen diferente para mostrar el carácter en un tamaño de 12 y 18. Por otra parte, cuando la impresora LaserWriter se introdujo en 1985, la tecnología Postscript definió fuentes con una descripción matemática del esquema de cada carácter. Este tipo permitió, en la pantalla, una escala de tamaños grandes. Apple y Microsoft desarrollaron, más tarde, el TrueType, otro formato de fuente. Recientemente, estas tecnologías se fusionaron en el formato OpenType. Mientras tanto, el método para suavizar texto de pantalla en blanco-y-negro se introdujo. Estas técnicas del tipo *anti-aliasing* utilizan píxeles grises en el borde de los caracteres para compensar la baja resolución de pantalla.

### -Cargando Fuentes y Dibujando Texto

Antes que las letras sean mostradas en la ventana de Processing, es necesario cargar la fuente y para esto debemos convertir el archivo en uno de formato VLW. Para esto, Processing nos ofrece una sencilla herramienta. En el menú "Tools" (herramientas) seleccionar la opción "Create Font" (Crear Fuente). Un ventana se abrirá y le mostrará las fuentes que tiene actualmente instaladas en su equipo y que pueden ser convertidas al formato específico. Seleccione la fuente deseada de la lista y, a continuación, haga click en "OK". La fuente se copiará y convertirá al formato VLW. Luego, automáticamente, será movida a la carpeta "Data" de su Sketch (de forma similar a como se carga una imagen).

Como en las primeras computadoras de Apple Macintosh, el formato VLW permite cargar el alfabeto de una fuente como una serie de simples imágenes. En la ventana del "Create Font" también se encuentra una opción que permite seleccionar el tamaño al exportar al fuente en un formato VLW. Esto es útil, ya que fuentes muy grandes requieren mayor cantidad de memoria. Si se va a utilizar una fuente no mas grande que 12, exportarla en tamaño 96 será totalmente innecesario.

Después que la fuente es creada, antes de que el programa la pueda utilizar es necesario cargarla en el mismo. Processing posee un tipo de dato llamado PFont, en el que deben cargarse las fuentes. Se crea una nueva variable tipo PFont y se utiliza la función loadFont() para cargarla. La función de textFont() se utiliza para seleccionar la fuente que se quiere utilizar. Por último, el texto es escrito con la función text().

```
text(datos, x, y)
text(stringdatos, x, y, ancho, alto)
```

El parámetro *datos* puede ser un String, char, int o float. El parámetro *stringdatos* es un parámetro que solo acepta datos tipo String. Los parámetros *x* e *y* cambian la posición horizontal y vertical del texto (como en cualquier figura o imagen). Existen los parámetros opcionales *ancho* y *alto*, los cuales modifican el tamaño del texto. La función fill() controla el color del texto y la transparencia, así como también afecta a figuras como ellipse() o rect(). El texto no es afectado por stroke().

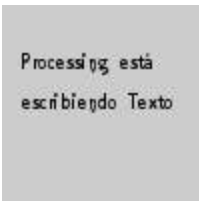
Los siguientes ejemplos utilizan una fuente llamada Aressence, para que pueda ver su texto asegúrese de cargar su propia fuente y cargarla correctamente (con la extensión) en la función loadFont().



```
PFont font; // Declare the variable
font = loadFont("Aressence-32.vlw"); //Carga la fuente
textFont(font); //Selecciona la fuente actual
fill(0);
text("LAX", 0, 40); //Escribe "LAX" en la posición (0,40)
text("AMS", 0, 70); //Escribe "AMS" en la posición (0,70)
text("FRA", 0, 100); //Escribe "FRA" en la posición (0,100)
```



```
PFont font;
font = loadFont("Aressence-32.vlw");
textFont(font);
fill(0);
text(19, 0, 36); //Escribe 19 en la posición (0,36)
text(72, 0, 70); //Escribe 72 en la posición (0,70)
text('R', 62, 70); //Escribe 'R' en la posición (62,70)
```



```
PFont font;
font = loadFont("Aressence-12.vlw");
textFont(font);
fill(0);
String s = "Processing está escribiendo Texto";
text(s, 10, 20, 80, 50);
```



```
PFont font;
font = loadFont("Aressence-32.vlw");
textFont(font);
fill(255); //Blanco
text("DAY", 0, 40);
fill(0); //Negro
text("CVG", 0, 70);
fill(102); //Gris
text("ATL", 0, 100);
```



```
PFont font;
font = loadFont("Aressence-72.vlw");
textFont(font);
fill(0, 160); //Negro con baja Opacidad
text("1", 0, 80);
text("2", 15, 80);
text("3", 30, 80);
text("4", 45, 80);
text("5", 60, 80);
```

También pueden utilizarse dos fuentes distintas en el mismo programa, pero deberán crearse dos variables tipo PFont que las almacenen:



```
PFont font1, font2;
font1 = loadFont("Aressence-32.vlw");
font2 = loadFont("Arhermann-32.vlw");
fill(0);
textFont(font1);
text("GNU", 6, 45);
textFont(font2);
text("GNU", 6, 80);
```

### -Atributos de Texto


Processing incluye funciones para controlar la forma en la que el texto se muestra (por ejemplo, alterando su tamaño o alineación). Processing, además, puede calcular el ancho de algún carácter o grupo de caracteres y

ponerlo en función de alguna figura o forma.


Las fuentes en Processing son tomadas como imágenes, y no como vectores. Por eso si la fuente que cargamos está en 12 de tamaño, y luego la agrandamos a 96, se verá desenfocada. La función correcta para cambiar el tamaño de la fuente es la de `textSize()`.

`textSize(tamaño)`

El parámetro *tamaño* define la dimensión que ocuparán las letras en unidades de píxeles.



```
//Reducción del tamaño de la letra
PFont font;
font = loadFont("Aressence-32.vlw");
textFont(font);
fill(0);
text("LNZ", 0, 40);    //Grande
textSize(18);
text("STN", 0, 75);    //Mediano
textSize(12);
text("BOS", 0, 100);   //Pequeño
```



```
//Agrandar tamaño de letra
PFont font;
font = loadFont("Aressence-12.vlw");
textFont(font);
textSize(32);
fill(0);
text("LNZ", 0, 40);    //Grande
textSize(18);
text("STN", 0, 75);    //Mediano
textSize(12);
text("BOS", 0, 100);   //Pequeño
```

La función `textLeading()` permite alterar el espacio entre líneas:

`textLeading(dist)`

La propiedad *dist* define la distancia del espacio en unidad de píxeles.

```
PFont font;
font = loadFont("Aressence-12.vlw");
textFont(font);
String lines = "L1 L2 L3";
textLeading(10);
fill(0);
text(lines, 5, 15, 30, 100);
textLeading(20);
text(lines, 36, 15, 30, 100);
textLeading(30);
text(lines, 68, 15, 30, 100);
```

Además, el texto puede ser marginado al centro, a la izquierda o a la derecha utilizando la función `textAlign()`:

`textAlign(MODO)`

En este caso, el modo es una constante que puede ser `CENTER` (centro), `RIGHT` (derecha) y `LEFT`

(izquierda), y se refiere simplemente a la alineación que tendrá el texto mostrado en pantalla.



```
PFont font;  
font = loadFont("Aressence-12.vlw");  
textFont(font);  
line(50, 0, 50, 100);  
fill(0);  
textAlign(LEFT);  
text("Izquierda", 50, 20);  
textAlign(RIGHT);  
text("Derecha", 50, 40);  
textAlign(CENTER);  
text("Centro", 50, 80);
```

La función `textwidth()` calcula y regresa el ancho de pixel de un carácter o texto. El calculo se realiza dependiendo de las propiedades de `textFont()` y `textSize()`. Ya que cada carácter posee un ancho diferente entre sí, es imposible saber la medida exacta de una cadena de caracteres. Para esos casos donde necesitamos una medida específica, o conocer el tamaño de nuestra cadena, se utiliza esta función.



```
PFont font;  
font = loadFont("Aressence-32.vlw");  
textFont(font);  
fill(0);  
char c = 'U';  
float cw = textWidth(c);  
text(c, 22, 40);  
rect(22, 42, cw, 5);  
String s = "UC";  
float sw = textWidth(s);  
text(s, 22, 76);  
rect(22, 78, sw, 5);
```