Librería Text [KE]

Esta librería, como su nombre lo indica, está enfocada en el texto de nuestros proyectos. Esta librería contiene una serie de funciones destinadas a la modificación, transformación y mejoramientos de los recursos de las líneas de texto.

text.upper(Text) :

Esta función convierte el texto que le ingresemos en el parámetro **Text**, a mayúsculas.

El parámetro **Text** debe ser un string de texto, y su valor por default dependerá del **Template Type** seleccionado en el efecto que vayamos a aplicar:

Template Type [fx]	Texto por Default
Syl	
Line	**line.text_stripped**
Word	**word.text**
Syl	**syl.text**
Furi	**furi.text**
Char	**char.text**
Convert to Hiragana	**hira.text**
Convert to Katakana	**kata.text**
Convert to Romaji	**roma.text**
Translation Line	**line.text_stripped**
Translation Word	**word.text**
Translation Char	**char.text**
Template Line [Word]	**line.text_stripped**
Template Line [Syl]	**line.text_stripped**
Template Line [Char]	**line.text_stripped**

Ejemplo:

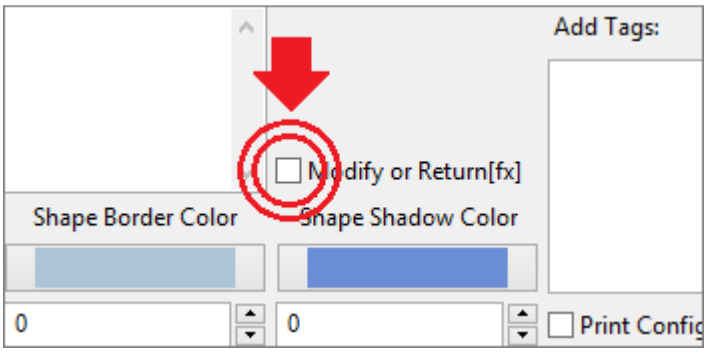
Y acá notamos cómo el texto se cambió todo a mayúsculas:

	Siento los pensamientos que dejaste atrás
	Me aferraré a ti y nunca te soltaré
	Dos corazones que se buscan conforman este sueño
[Fx]	✱MIS MEJILLAS SE MANCHAN CON LÁGRIMAS DE SOLEDAD
[Fx]	✱PERO PUEDO SENTIR LA LLEGADA DEL AMANECER
[Fx]	✱QUE ME ATRAE CON RUMBO HACIA LOS CIELOS
[Fx]	✱LA ESPERANZA ESPERA AL OTRO LADO, POR ESO VOLARÉ
[Fx]	✱ME PIERDO EN EL CAMINO CADA VEZ QUE TE BUSCO
[Fx]	✱SIENTO LOS PENSAMIENTOS QUE DEJASTE ATRÁS

Si ponemos algo distinto en el parámetro **Text**, la función lo transformará automáticamente en mayúsculas. Ejemplo:

- **text.upper(“texto demo”) → “TEXTO DEMO”**
- **text.upper(“Ejemplo n° 2”) → “EJEMPLO N° 2”**
- **text.upper(“Aegisub”) → “AEGISUB”**

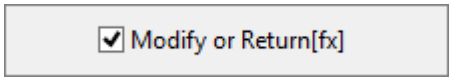
Otra forma de usar esta función es apoyándonos de esta herramienta del **Kara Effector**:



La opción “**Modify or Return[fx]**“, al estar marcada, hace que el efecto en vez de generar nuevas líneas de fx, modifique a las líneas de nuestro script. Como su nombre lo indica, con esta opción decidimos si queremos modificar a nuestras líneas del archivo .ass o queremos generar un efecto a partir de líneas nuevas de fx.

Ejemplo:

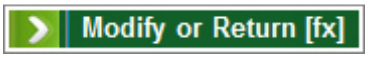
- **Template Type**: Line o Translation Line
- Marcamos la opción “**Modify or Return[fx]**“



- En **Return [fx]** llamamos a nuestra función:

Y lo que sucederá es que no se generarán líneas de fx, sino que las líneas seleccionadas para que se le aplicará el efecto, se pasarán a mayúsculas:

Estilo	Texto
Hiragana	✷す✷れ✷ち✷が✷う ✷い✷し✷き ✷て ✷が ✷ふ✷れ✷た ✷ょ ✷ね
Hiragana	✷つ✷か✷ま✷え✷る ✷ょ ✷し✷っ✷か✷り
Hiragana	✷も✷と✷め ✷あ✷う ✷こ✷こ✷ろ ✷そ✷れ ✷わ ✷ゅ✷め ✷の ✷あ
English	MIS MEJILLAS SE MANCHAN CON LÁGRIMAS DE SOLEDAD
English	PERO PUEDO SENTIR LA LLEGADA DEL AMANECER
English	QUE ME ATRAE CON RUMBO HACIA LOS CIELOS
English	LA ESPERANZA ESPERA AL OTRO LADO, POR ESO VOLARÉ
English	ME PIERDO EN EL CAMINO CADA VEZ QUE TE BUSCO
English	SIENTO LOS PENSAMIENTOS QUE DEJASTE ATRÁS
English	ME AFERRARÉ A TI Y NUNCA TE SOLTARÉ
English	DOS CORAZONES QUE SE BUSCAN CONFORMAN ESTE SUEÑ



Esta opción del **Kara Effector** nos da la posibilidad de decidir si queremos modificar las líneas de nuestro script .ass o si queremos generar nuevas líneas fx.

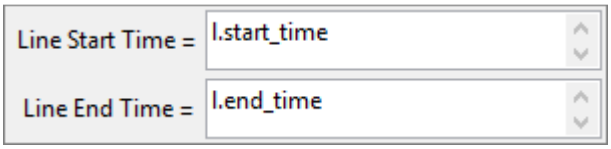
Si marcamos esta opción, podemos modificar a nuestras líneas del script .ass de dos formas diferentes.

1. **Su contenido**:

   El contenido de las líneas se modificará solo con todo aquello que pongamos en la celda de texto **Return [fx].**

2. **Sus tiempos**:

   Podemos modificar los tiempos de inicio y final de las líneas de nuestro script .ass con las celdas de texto del **Kara Effector** destinadas para ello:



   Cualquier tiempo que añadamos o sustraigamos en estas dos celdas, modificarán los tiempos de las líneas del script .ass

Con esta opción marcada, solo funcionan estas tres celdas de texto de la **Ventana de Modificación** del **Kara Effector**, es decir que el resto de las herramientas en esta Ventana quedan inhabilitadas para modificar a la líneas de nuestro script .ass

text.lower(Text) :

Es la función opuesta a **text.upper**, y convierte el string de texto ingresada en ella, a minúsculas.

Ejemplo:

- **text.lower("Texto Demo") → "texto demo"**
- **text.lower("EJEMPLO N° 2") → "ejemplo n° 2"**
- **text.lower("AEGISUB") → "aegisub"**

text.infx(select_fx) :

Esta función aplica un efecto total o parcialmente, únicamente a las Sílabas previamente marcadas en nuestro script .ass

Hay dos formas diferentes con las que podemos marcar las Sílabas del script:

- **-fx**
- **+fx**

Estas marcas deben ir dentro de los tags de las Silabas.

La marca (**-fx**) selecciona únicamente a la sílaba que la contenga en su tag. Ejemplo

{\k19}ma{\k18**-fx**}yo{\k28}i {\k32}na{\k31**-fx**}ga{\k35**-fx**}ra

Sílabas marcadas:

- "yo"
- "ga"
- "ra"

La marca (**+fx**) selecciona a todas las Sílabas desde la marcada hasta el final de la línea karaoke, o hasta la siguiente marca (-fx) que se encuentre a su derecha:

{\k19}ma{\k18**+fx**}yo{\k28}i {\k32}na{\k31}ga{\k35}ra

De este modo, la marca (**+fx**) seleccionará a todas las Sílabas desde "yo" hasta la última, o sea la Sílaba "ra".

{\k19}ma{\k18**+fx**}yo{\k28}i {\k32**-fx**}na{\k31}ga{\k35}ra

Hecho así, quedan seleccionadas las Sílabas desde "yo" hasta "na".

{\k19}ma{\k18**+fx**}yo{\k28}i {\k32**-fx**}na{\k31}ga{\k35**-fx** }ra

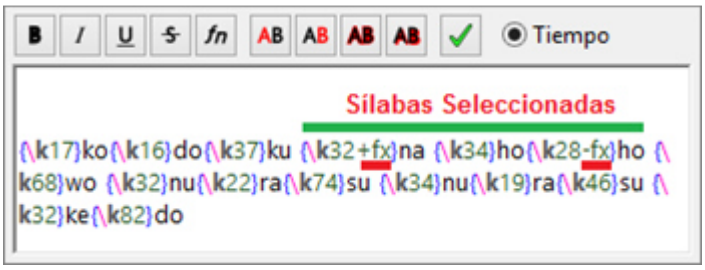
Y así, quedan seleccionadas las Sílabas desde "yo" hasta "na" y la Sílaba "ra".

Una vez entendida la forma en las que podemos marcar las Sílabas de las líneas karaoke, veremos varios ejemplos de cómo usar la función **text.infx**, ya que tiene tres diferentes modos de uso.

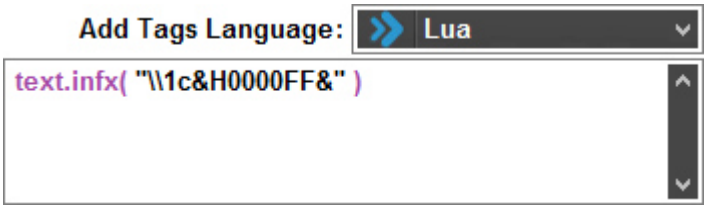
- **Modo 1**: aplicar un efecto parcial. Un **string**:

Ejemplo:

Marcamos algunas sílabas, no importa de cuál o de cuántas líneas karaokes:



Y en **Add Tags** agregamos algún **string** en la función:

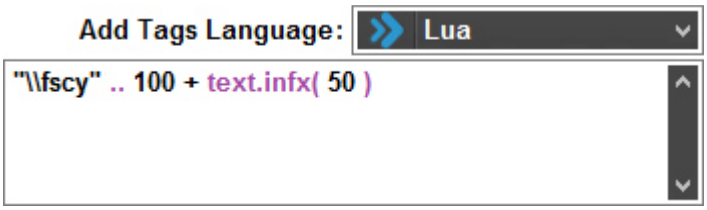


Entonces al aplicar el efecto, la función solo agregará el string ingresado únicamente a las Sílabas seleccionadas previamente:

Notamos cómo el string (en este caso el color primario rojo) fue agregado solo a las Sílabas marcadas, a las demás no le agregó absolutamente nada.

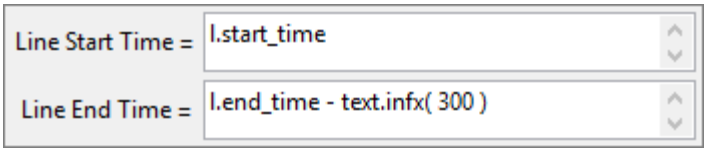
- **Modo 2**: aplicar un efecto parcial. Un **número**.

Ejemplo:



Entonces la función sumará los 50 a los 100 que están antes de ella, solo a las Sílabas marcadas, a las restantes le sumará 0. O sea que la función, a las Sílabas que no están marcadas les agregará **“\\fscy100”** y a las que sí lo están, les agregará **“\\fscy150”**:

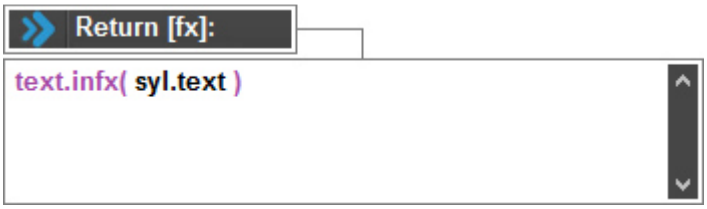
Con este modo de adicionar o sustraer un número con la función **text.infx**, también lo podemos hacer en las celdas de texto de tiempo. Ejemplo:



- **Modo 3**: aplicar un efecto total. Únicamente a las Sílabas marcadas:

Ejemplo:

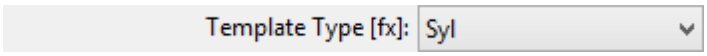
Para que un efecto se aplique únicamente a las Sílabas que previamente hemos marcado, lo que debemos hacer es usar la función en la calda de texto **Return** [fx], así:



Entonces el efecto que hayamos elegido solo se aplicará a la Sílabas marcadas, el resto no se tendrán en cuenta, y por ende no saldrán en pantalla:

En resumen, la función usada en cualquier celda de texto que no sea **Return [fx]** aplica el efecto de modo parcial a las sílabas marcadas, al resto solo se le aplicará el efecto general y no el que esté dentro de la función. Y al usar la función en **Return [fx]** como en el último ejemplo, el efecto general solo se aplicará a las Sílabas marcas y las demás no se tendrán en cuenta.

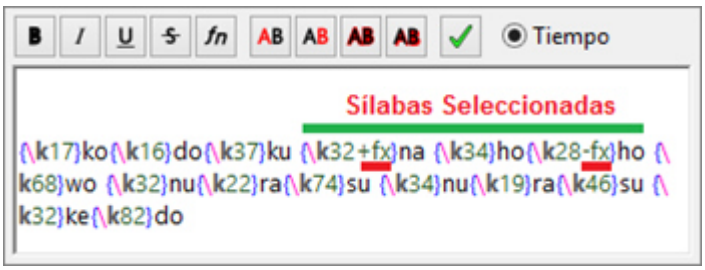
Solo resta decir que esta función, dado que marcamos las Sílabas, está diseñada únicamente para ser usada en el modo **Template Type: Syl**

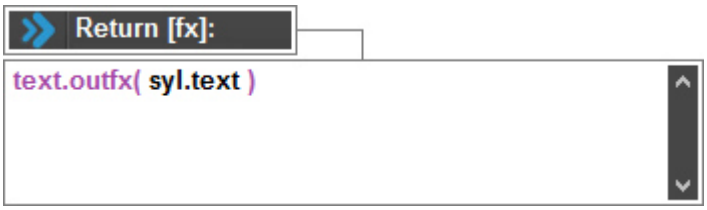


text.outfx(select_fx) :

Esta es la función opuesta a **text.infx**, es decir que nunca toma en cuenta a las Sílabas marcadas con (**+fx**) o con (**-fx**).

Ejemplo:

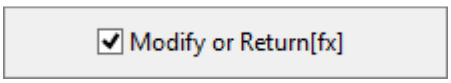


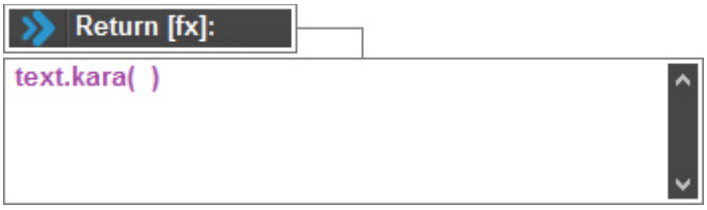


Y vemos el resultado opuesto de la función **text.infx**, es decir que no toma en cuenta a las Sílabas marcadas:

text.kara( ) :

Esta función está diseñada para ser usada con la opción "**Modify or Return [fx]**" marcada, y dentro de la celda de texto **Return [fx]**.





Y lo que hace esta función es convertir las líneas normales de nuestro script .ass a líneas karaoke:

Ejemplo:

Convierte estas líneas:

English	Mis mejillas se manchan con lágrimas de soledad
English	Pero puedo sentir la llegada del amanecer
English	Que me atrae con rumbo hacia los cielos
English	La esperanza espera al otro lado, por eso volaré
English	Me pierdo en el camino cada vez que te busco
English	Siento los pensamientos que dejaste atrás
English	Me aferraré a ti y nunca te soltaré
English	Dos corazones que se buscan conforman este sueño

A líneas karaoke:

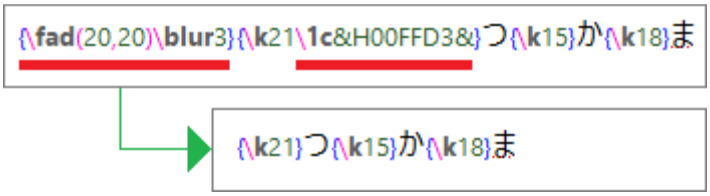
English	*Mis *mejillas *se *manchan *con *lágrimas *de *soledad
English	*Pero *puedo *sentir *la *llegada *del *amanecer
English	*Que *me *atrae *con *rumbo *hacia *los *cielos
English	*La *esperanza *espera *al *otro *lado, *por *eso *volaré
English	*Me *pierdo *en *el *camino *cada *vez *que *te *busco
English	*Siento *los *pensamientos *que *dejaste *atrás
English	*Me *aferraré *a *ti *y *nunca *te *soltaré
English	*Dos *corazones *que *se *buscan *conforman *este *sueño

{\k36}Me {\k145}aferraré {\k18}a {\k36}ti {\k18}y {\k91}nunca {\k36}te {\k127}soltaré

La función asigna un tiempo aproximado a cada palabra dependiendo de la cantidad de caracteres (letras) que tenga dicha palabra.

Y la otra función que cumple **text.kara( )** es remover todos los tags que no sean karaoke, de las líneas de karaoke:

Ejemplo:



text.tag(…) :

Esta función agrega los tags que asignemos en sus parámetros, a cada uno de los caracteres (letras) del texto por default de un efecto.

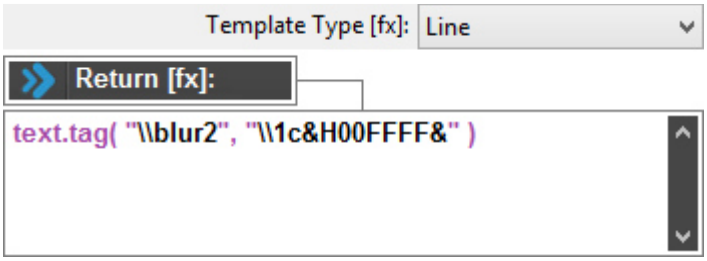
Recordemos que el texto por default en un efecto depende del **Template Type**:

Template Type [fx]	Texto por Default
Syl	
Line	**line.text_stripped**
Word	**word.text**
Syl	**syl.text**
Furi	**furi.text**
Char	**char.text**
Convert to Hiragana	**hira.text**
Convert to Katakana	**kata.text**
Convert to Romaji	**roma.text**
Translation Line	**line.text_stripped**
Translation Word	**word.text**
Translation Char	**char.text**
Template Line [Word]	**line.text_stripped**
Template Line [Syl]	**line.text_stripped**
Template Line [Char]	**line.text_stripped**

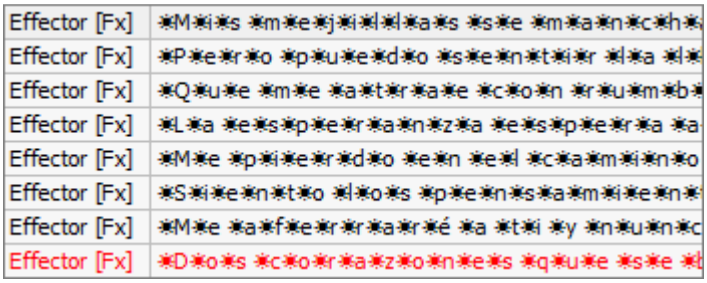
Yo, por lo general uso esta función en los modos **Line** y **Translation Line**. Ya que esta función agrega tags a cada letra, entonces no tiene mucho sentido usarla en los modos **Char** y **Translation Char**.

Esta función no surge ningún efecto en los tres modos de **Template Line** (**[Word], [Syl] y [Char]**), y hay dos formas diferentes de agregar los tags a las letras:

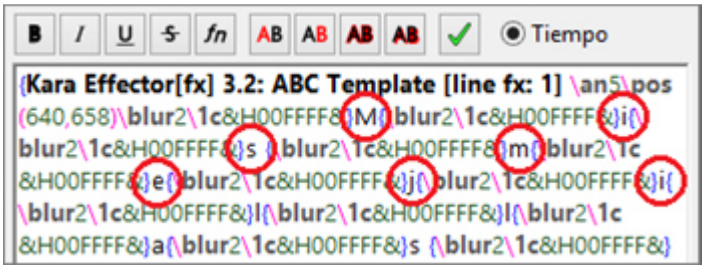
Ejemplo:



Como el ejemplo está hecho en modo **Line**, entonces en cada línea fx va la línea completa, y al aplicar el efecto con la función, entonces se le agregarán los tags ingresados a cada una de las letras de la línea de texto, como se nota en la siguiente imagen:

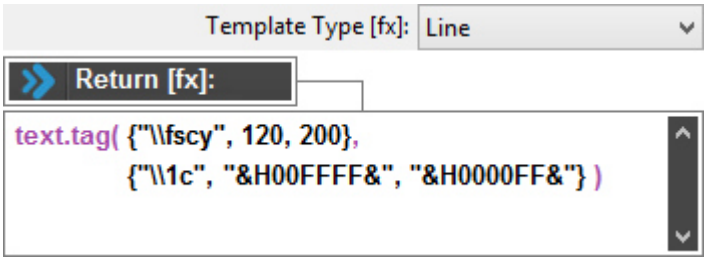


Acá vemos cómo los tags ingresados están antes de cada letra de la línea de texto:



La segunda forma de agregarle los tags a las letras, con esta función, es la más interesante:

Ejemplo:



Lo que hacemos es que, si un parámetro de la función es una tabla, debemos poner en el primer elemento de dicha tabla, al tag que queremos usar seguido de los dos valores a interpolar, el inicial y el final:

Vemos cómo el tag \fscy va creciendo desde 120 hasta 200, de letra a letra, y cómo el color primario \1c va cambiando de amarillo a rojo.

También se pueden combinar las dos formas de agregar los tags, y la cantidad de tags que se pueden añadir a la función es ilimitada.

text.to_shape(Text, Scale, Tags) :

Esta función, como su nombre lo indica, convierte un string de texto a **shape**.

El parámetro **Text** es el string de texto que queremos convertir a **shape** y su valor por default es el texto por default dependiendo el **Template Type**, como ya lo hemos visto en las anteriores funciones.

El parámetro **Scale** es un número entero positivo que hace referencia a las escala de la **shape** en la que se convertirá el texto. Las opciones recomendadas son:

- 1
- 2
- 4
- 8
- 16

De hecho, la escala puede ser cualquier otra potencia de 2, como 32, 64, 128 o superiores, pero eso hará que el texto se convierta en una **shape** demasiado grande, lo que generará un código muy extenso por línea, lo que hará que el script quedé muy pesado. Su valor por default es 1.

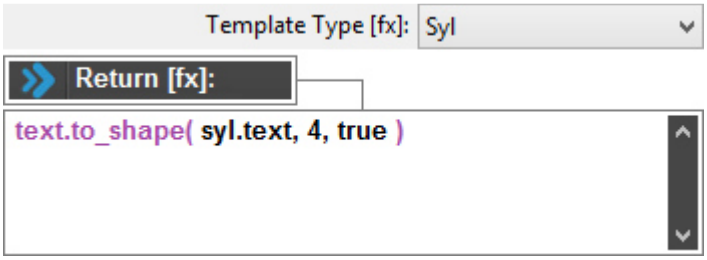
El parámetro **Tags** que añade 3 tags a la **shape** generada:

- **\an** ← Alineación respecto a la línea
- **\pos** ← Posición en la línea
- **\p** ← Proporción de la **shape**

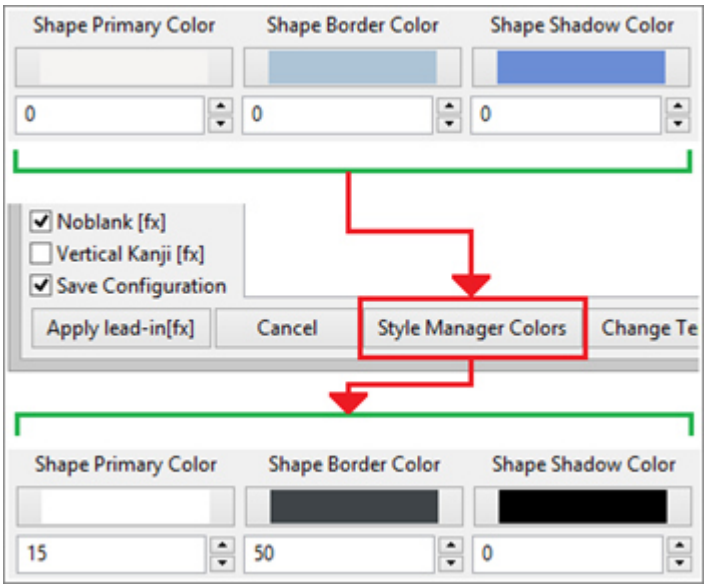
Estos tags se agregan a la **shape** con el fin ver en pantalla al texto convertido en **shape**, en su posición y alineación correcta sin necesidad de calcular otra cosa más en la **Ventana de Modificación** del **KE**. Este parámetro es un **boleano**, que de no estar en la función, se asumirá cono **nil** (nulo), o sea que no le añadirá ningún tag a la **shape** generada, es decir que **nil** es su valor por default. Y para añadir los tags mencionados, en el parámetro **Tags** debemos poner la palabra **true** (verdadero).

Ejemplo:

Llamamos a la función en **Return [fx]:**



Si aplicamos hasta este punto el ejemplo, el texto saldrá convertido en **shape** y por ende saldrá con los colores de las Shapes y no con los colores de los textos. Recordemos el procedimiento para convertir los colores a los asignados en el estilo:

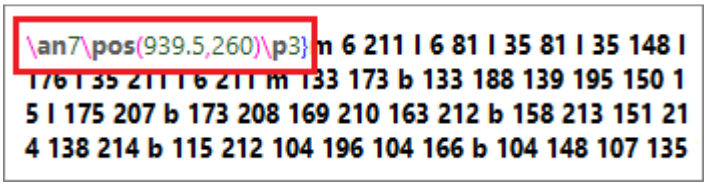


Ahora al aplicar, podremos ver el texto convertido a **shape**, pero con los colores del texto. En apariencia no se notan las diferencias entre el texto original y la shape a la que se transformó:

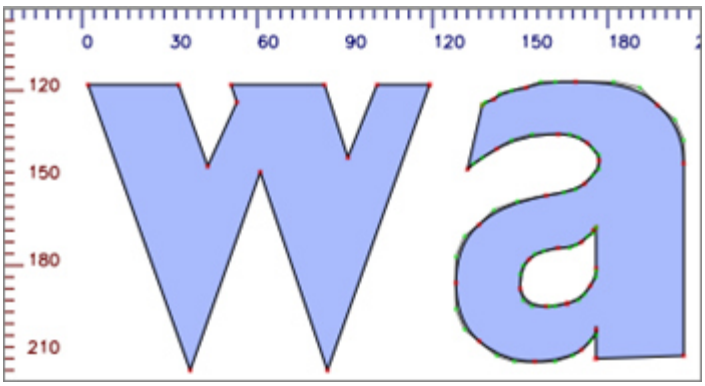
Al ver las líneas generadas, notamos que en lugar de cada Sílaba, está la **shape** de la misma:

	Dos corazones que se buscan conforman este sueño
Effector [Fx]	m 6 211 l 6 81 l 35 81 l 35 148 l 65 118 l 93 118 l 57 154
Effector [Fx]	m 85 81 l 85 211 l 56 211 l 56 206 b 54 207 52 209 49 2
Effector [Fx]	m 6 211 l 6 81 l 35 81 l 35 148 l 65 118 l 93 118 l 57 154
Effector [Fx]	m 84 144 l 84 211 l 62 211 l 62 150 b 62 146 61 143 58
Effector [Fx]	m 84 144 l 84 212 l 62 211 l 62 150 b 62 146 61 143 58
Effector [Fx]	m 84 144 l 84 212 l 62 211 l 62 150 b 62 146 61 143 58
Effector [Fx]	m 119 118 l 84 216 l 61 148 l 37 216 l 2 118 l 33 118 l 4

Y como pusimos "**true**" en el parámetro Tags, entonces la **shape** generada viene con los 3 tags que la posicionan:



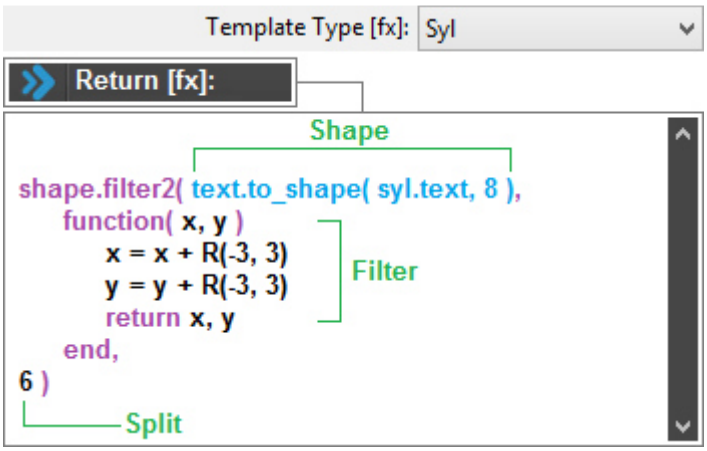
Al copiar el código de una de las Shapes generadas, y pegarlo en el **ASSDraw3**, veremos algo como esto:



La ventaja de poder convertir el texto en una **shape**, es que hecho una vez esto, le podemos aplicar cualquier función de la **librería shape**:

Ejemplo:

Para usar el texto como una shape convencional, debemos omitir al parámetro **Tags**, para que la función no retorne a los 3 tags junto con la **shape**:



shape.filter2:

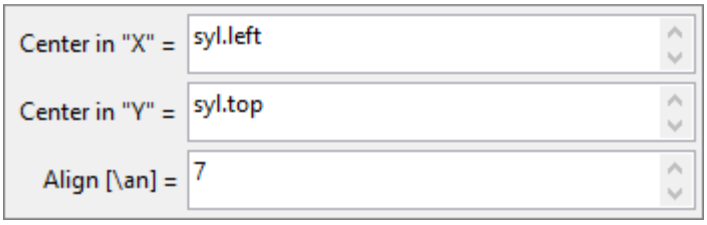
- **Shape**: el texto aumentado en un factor de 8.
- **Filter**: una función que hace que cada coordenada, tanto "x" como "y", queden desplazados de su posición original, por un valor aleatorio entre -3 y 3.
- **Split**: es el valor de la longitud de cada segmento recto en la que se dividirá la shape, en este caso al texto convertido en shape. Tiene en este ejemplo un valor de 6 px.

Y al aplicar sucederá lo siguiente:

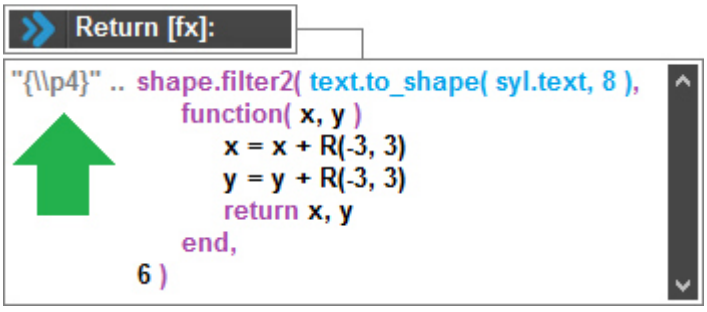
El texto está convertido en una **shape**, que a su vez fue deformado por el filtro de la función **shape.filter2**, pero está 8 veces más grande de lo que debería estar y no tiene ni la alineación ni la posición correcta. Y para solucionar esto debemos hacer lo siguiente:

Ejemplo:

Modificamos los puntos de referencia a las posiciones **Left**, **Top** (Izquierda, Superior). Esta modificación depende del **Template Type**. Y cambiamos la alineación del texto a 7:



Ahora nos resta añadir el tag **\p** que es el que nos dará la proporción correcta de la shape para que tenga el tamaño real del texto, y lo debemos hacer así:



Y al aplicar veremos el texto deformado y en su posición y alineación correcta:

Y con un poco más de pixeles en la distorsión en el filtro:

En el anterior ejemplo vimos cómo el tag de proporción \p4 es el que le corresponde a una escala de 8 para el texto. La siguiente tabla nos mostrará la proporción correspondiente a cada una de las diferentes escalas:

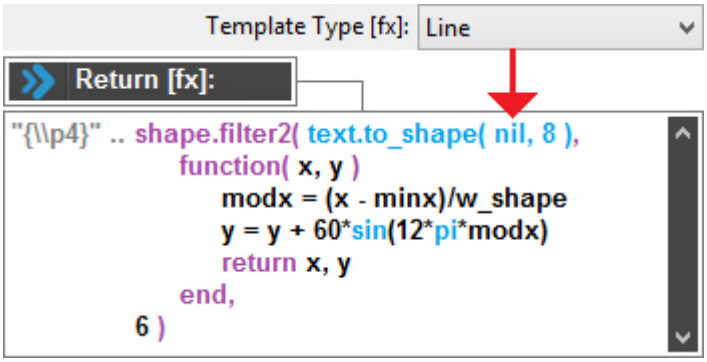
Escala del Texto	Proporción \\p
1	1
2	2
4	3
8	4
16	5

Como las Shapes se modifican por medio de una función, de un filtro, las posibilidades son infinitas y todo dependerá de la creatividad de cada uno.

Para el siguiente ejemplo, ya no convertiremos a cada Sílaba una por una sino a la línea de texto completa, y para ello ya saber qué hacer, modificar el **Template Type**:

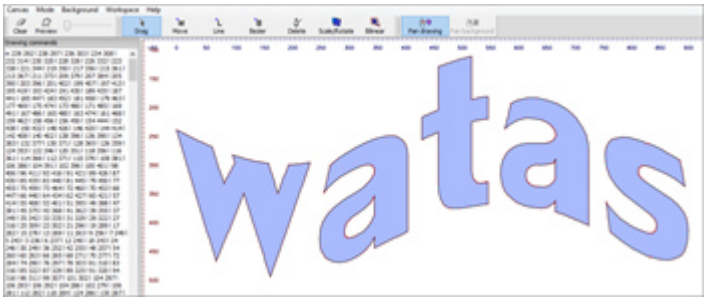
Ejemplo:

Recordemos que **minx** es el mínimo valor de "x" y que **w_shape** es el ancho medido en pixeles de la **shape**:



Al poner **nil** en el parámetro **Text** de **text.to_shape**, ésta lo asumirá por su valor por default, que en modo **Line** es: **line.text_stripped**

Y vemos cómo la línea de texto completa (en **shape**) se distorsiona gracias a la función del filtro:



text.do_shape(Text, Shape, Scale, Mode, Tags) :

Esta función es algo similar a la anterior, y de hecho el principio es el mismo, ya que convierte el texto ingresado en **shape**. Esta función contiene dentro de sí a un filtro que hace que el texto, luego de haber sido convertido a **shape**, adopte la forma de otra **shape** ingresada en el parámetro **Shape**.

El parámetro **Text** es el texto para convertir a **shape** y su valor por default es el texto por default dependiendo el **Template Type**, como ya lo hemos visto en las anteriores funciones.

El parámetro **Shape** es la nueva figura que adoptará el texto una vez convertido en **shape**. Su valor por default es una **shape** recta equivalente a no aplicarle ningún filtro al texto, lo que en principio no tiene mucho sentido, ya que si usamos esta función es precisamente para convertir el texto en una **shape** nueva.

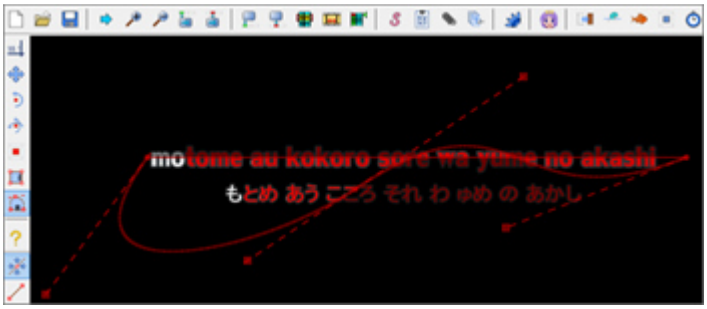
Los parámetros **Scale** y **Tags** cumplen exactamente la misma tarea que en la función anterior, que es la de darle proporción texto convertido en **shape** y añadirle los tags de posicionamiento al mismo.

El parámetro **Mode** es un número entero entre 1 y 4, para que la función haga la transformación en uno de sus cuatro modos habilitados. Su valor por default es 1.

Para los siguientes ejemplos, usaremos la función con un **Template Type: Line**, aunque con cualquiera es posible usar la función, ya que no importa el tipo de texto ingresado, ésta lo convertirá en shape.

Una forma sencilla de dibujar Shapes para esta función es usando la herramienta para recortar subtítulos en un área vectorial (**\clip**):

Y con solo dos bezier dibujé la siguiente curva usando dicha herramienta:



Este es el código de la anterior curva clip, el cual usaremos como **shape** para los siguientes ejemplos:

```
{\clip(m 204 298 b 31 532 375 474 558 352 845 160 815
418 1122 298)}
```

Y para mayor comodidad, definimos esta **shape** como una variable, que luego usaremos en la función:

```
Variables:
mi_shape = "m 204 298 b 31 532 375 474 558 352
            845 160 815 418 1122 298"
```

Ejemplo:

Usamos la función con **Mode** = 1

```
Template Type [fx]: Line
Return [fx]:
text.do_shape(
        line.text_stripped,   <-- Texto
        mi_shape,             <-- Shape
        8,                    <-- Escala
        1,                    <-- Modo
        true                  <-- Tags
)
```

Entonces la función convierte el texto a **shape** y luego lo obliga a adoptar la forma de la **shape** ingresada en el parámetro **Shape**:

Con **Mode** = 1, la función justifica al texto en el centro de la **shape**, de modo que queda repartido equitativamente, desde el centro hacia los extremos. El clip que dibujamos no se verá en la imagen, solo lo dejé para referencia.

Sin ese clip de referencia se debe ver así:

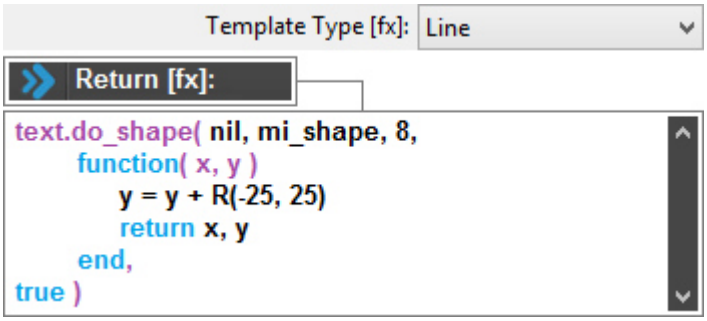
Con **Mode = 2**, el texto queda justificado a la derecha de la **shape** ingresada, según la longitud del mismo, que para este ejemplo sería: **line.width**

Con **Mode = 3**, el texto queda justificado a la izquierda de la **shape** ingresada:

Y con **Mode = 4**, el texto se deforma hasta alcanzar la longitud total de la **shape** ingresada:

Una quinta opción que tiene el parámetro **Mode** es la de ser una función filtro:

Ejemplo:



Entonces la función asumirá la justificación de **Mode = 4**, y adicional a ello, aplicará el filtro a la **shape**:

Esta es otra función ideal para hacer algunos carteles pocos convencionales, como aquellos que tienen forma de arco:

O también para hacer logos que contengan texto que no esté en línea recta. Ejemplo:

text.bord_to_shape(Text, Scale, Tags, Bord) :

Esta función convierte el borde del texto ingresado y lo convierte en una **shape**.

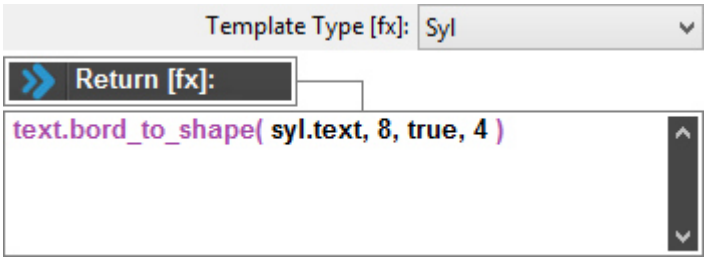
El parámetro **Text** es el string de texto al que la función le convertirá su borde en una **shape**, y su valor por default es el texto por default según el **Template Type**.

Los parámetros **Scale** y **Tags** cumplen con la misma tarea que en las dos funciones anteriores, darle proporción y agregar tags de posicionamiento. Sus valores por default son los mismos, es decir:

- **Scale** = 1
- **Tags** = nil

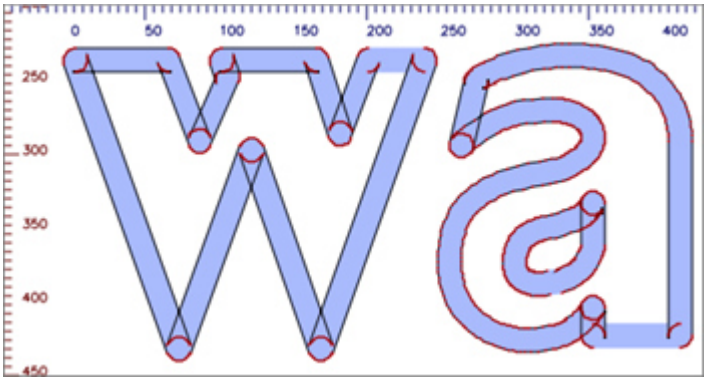
El parámetro **Bord** es un número que indica el grosor del borde hecho shape, su valor por default es 2.

Ejemplo:



Al aplicar veremos algo como esto:

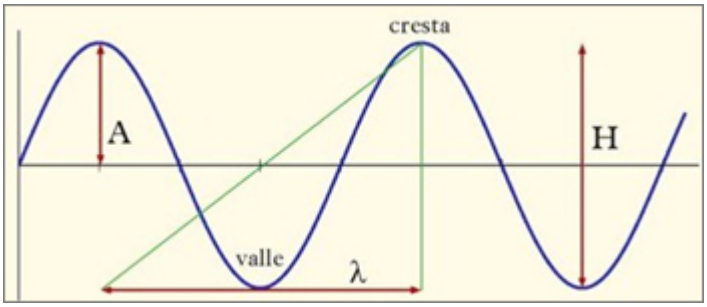
Una de las Sílabas vistas en el **ASSDraw3**:



Y el borde hecho **shape** ya queda apto para aplicarle la función de la librería **shape** que queramos.

text.deformed(Text, Deformed, Pixel, Axis) :

Esta función convierte el texto ingresado en una **shape** y posteriormente lo deforma adaptando la forma de onda senoidal:



El parámetro **Deformed** es un número mayor que cero, que indica la cantidad de crestas y valles que tendrá la onda en la que se convertirá el texto. Su valor por default es 2.

El parámetro **Pixel** es la altura en pixeles de cada una de las crestas y valles que tendrá la onda, lo que en la imagen anterior corresponde a la letra "**A**". Su valor por default es **line.height**

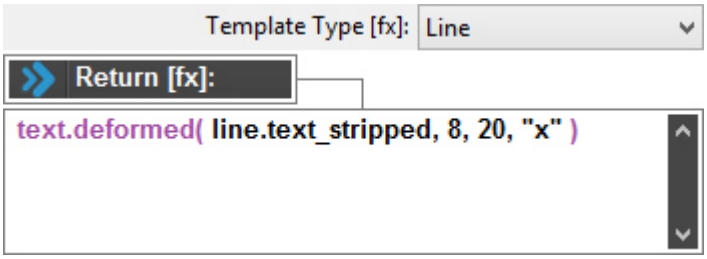
El parámetro **Axis** tiene tres opciones:

- "**x**": dibuja la curva sobre el eje "x"
- "**y**": dibuja la curva sobre el eje "y"
- { **Deforme2**, **Pixel2** }: asume a **Deformed** y **Pixel** como valores en la deformación respecto al eje "x" y a **Deformed2** y **Pixel2** como valores para la deformación respecto al eje "y". O sea que de este modo se deforma el texto respecto a ambos ejes.

El valor por default del **Axis** es “x”.

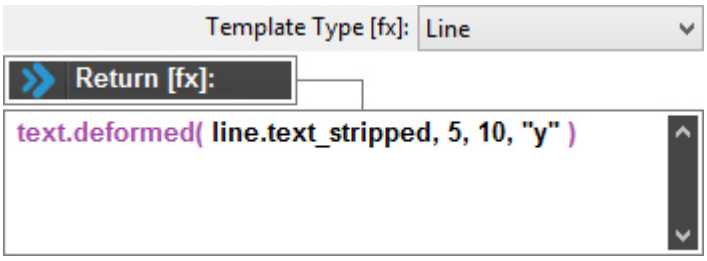
Para los siguiente ejemplos usaremos un **Template Type**: **Line**, aunque se puede usar esta función con cualquiera de los demás modos. Es solo que para fines ilustrativos es más visible ver la deformación sobre la línea de texto completa que en una letra o sílaba.

Ejemplo:



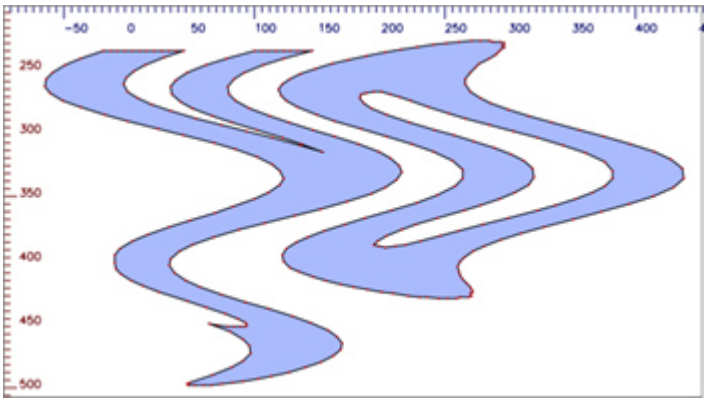
Y al aplicar. Vemos cómo hay 8, entre crestas y valles, y la altura de cada una de ellas es de 20 px:

Ejemplo:

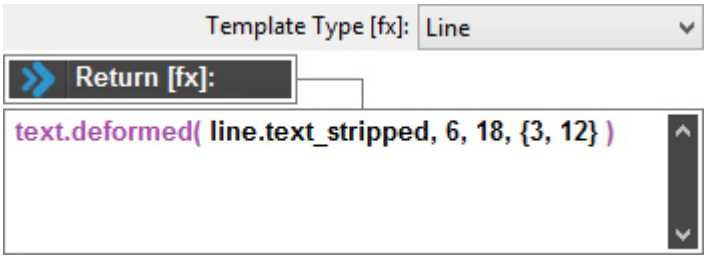


Ahora el texto se deforma respecto al eje “y” con cinco, entre crestas y valles, y 10 px como su altura:

El poder controlar los valores en que se deformará el texto nos da muchas opciones y resultados. En la siguiente imagen vemos en el **ASSDraw3** la Sílaba “yo” deformada:

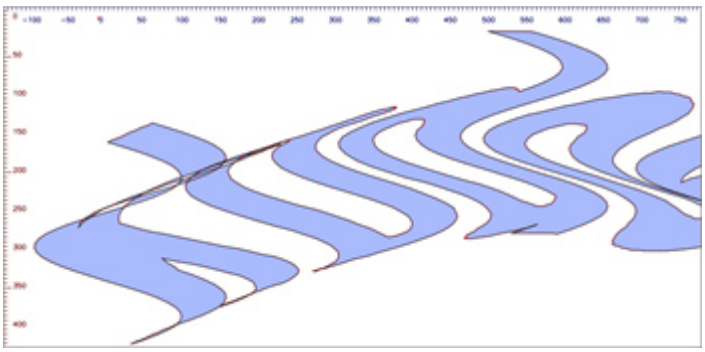


Ejemplo:



De este modo, se deforma el texto respecto a ambos ejes:

Visto en **ASSDraw3**, vemos un ejemplo de la deformación, en este caso pone: “kodo”



text.deformed2(Text, Mode) :

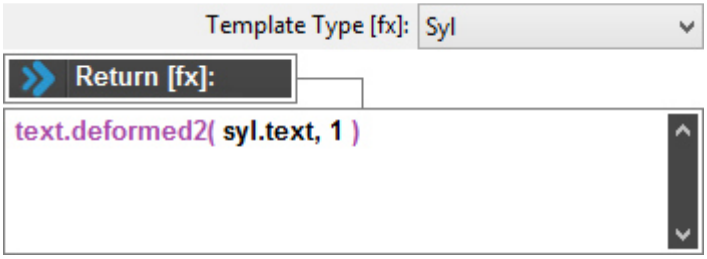
Esta función retorna tres ejemplos más de cómo deformar un texto convertido a **shape**.

El parámetro **Text** es el string de texto a convertir a **shape** para deformar y su valor por default es el texto por default dependiendo del **Template**
Type.

El parámetro **Mode** es un entero entre 1 y 3 que hará que la función retorne uno de los tres ejemplos de deformación. Su valor por default es 1.

Ejemplo:

- **Mode** = 1



Los puntos de la Sílaba convertida en **shape** se desplazan hasta el perímetro de un círculo de 80 px de diámetro:

- **Mode** = 2

Los puntos de desplazan al perímetro de 3 círculos con diferentes diámetros:

- **Mode** = 3

Los puntos de la **shape** se desplazan al perímetro de dos hexágonos concéntricos:

Las tres anteriores deformaciones son logradas gracias a las distintas funciones filtros que contienen cada una de ellos, es por eso que anteriormente les mencionaba que las posibilidades son infinitas a la hora de deformar una **shape** por medio de un filtro como función.

El **Mode** = 3 está basado en el siguiente filtro:

```
mi_filtro = function(x, y)
   local center_dx = minx + w_shape/2
   local center_dy = miny + h_shape/2
   local def_angle = math.angle(center_dx, center_dy, x, y)
   local def_angRE = (def_angle - 1)%60 + 1
   local def_ang3A = 180 - 60 - def_angRE
   local des_radiu = 200
   local des_dista = des_radiu*sin(rad(60))/sin(rad(def_ang3A))
   local des_radDE = (math.distance(center_dx,center_dy,x,y)<=80)
                      and des_dista/2 or des_dista
   x = center_dx + math.polar(def_angle, des_radDE, "x")
   y = center_dy + math.polar(def_angle, des_radDE, "y")
   return x, y
end
```

Este filtro nos ayudará a definir a una de las Shapes del siguiente ejemplo, también como variable. Y usaremos la función **shape.morphism** para ver a cada una de las Shapes, desde que el texto está normal hasta que se convierte en uno de los hexágonos del ejemplo anterior:

Ejemplo:

shape1 = shape.filter2(text.to_shape(“DE”, 8), nil, 6)

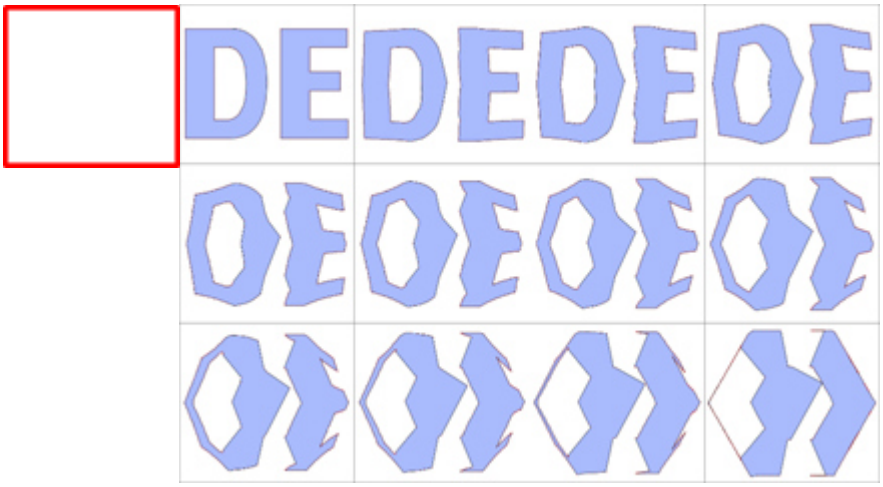
shape2 = shape.filter2(text.to_shape(“DE”, 8), mi_filtro, 6)

En la **shape1** el texto “DE” está normal, ya que en el filtro de la función pusimos “**nil**“. Y la **shape2**, como le pusimos el filtro del ejemplo anterior, entonces el texto “DE” ya está convertido en unos de los hexágonos vistos.

Ahora creamos la **tabla** con las interpolaciones entre una shape y la otra:

mi_tabla = **shape.morphism**(12, shape1, shape2)

Entonces veremos 12 Shapes de interpolación, de cómo el texto “DE” se transforma en uno de los hexágonos del ejemplo anterior:

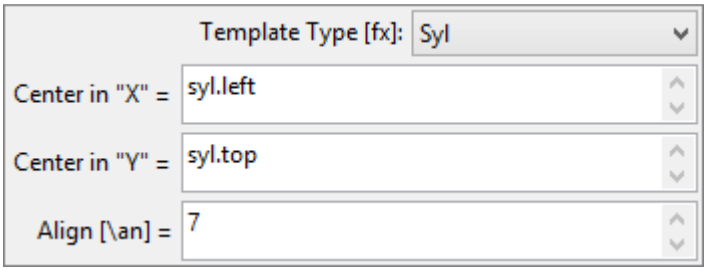


Aplicando el anterior ejemplo, en un **Template Type: Syl**, cambiamos el texto “DE” por **syl.text**, así:

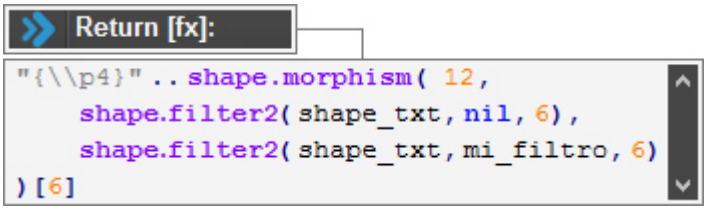
```
mi_filtro = function(x, y)
  local center_dx = minx + w_shape/2
  local center_dy = miny + h_shape/2
  local def_angle = math.angle(center_dx, center_dy, x, y)
  local def_angRE = (def_angle - 1)%60 + 1
  local def_ang3A = 180 - 60 - def_angRE
  local des_radiu = 200
  local des_dista = des_radiu*sin(rad(60))/sin(rad(def_ang3A))
  local des_radDE = (math.distance(center_dx,center_dy,x,y)<=80)
                    and des_dista/2 or des_dista
  x = center_dx + math.polar(def_angle, des_radDE, "x")
  y = center_dy + math.polar(def_angle, des_radDE, "y")
  return x, y
end;
shape_txt = text.to_shape( syl.text, 8 )
```

Y como vimos en las 12 Shapes de la transformación del texto “DE”, éste es legible como hasta la sexta o séptima **shape**. Sabido esto, adicionalmente hacemos lo siguiente:

Ejemplo:



Finalmente, ponemos esto en **Return [fx]:**



Y al aplicar veremos algo como esto:

Lo que hará que cada una de las Sílabas quede deformada de forma similar a la sexta **shape** del ejemplo anterior por haber puesto: **mi_tabla[6]**

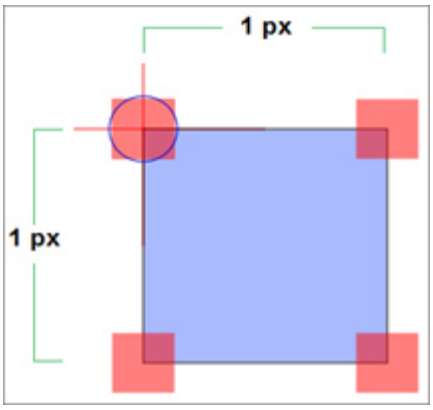
text.to_pixels(Text, Mode, Shape) :

Esta función convierte al texto ingresado en pixeles. Esta función genera un loop equivalente a la cantidad de pixeles que tenga el texto ingresado, según los valores de escala y tamaño que tenga en el estilo del mismo.

El parámetro **Text** es el texto a convertir en pixeles y su valor por default es el texto por default según el **Template Type**.

El parámetro **Mode** es un número entero entre 1 y 5 que hace referencia a las distintas posiciones y movimientos de cada uno de los pixeles que conforman el texto. Su valor por default es 1.

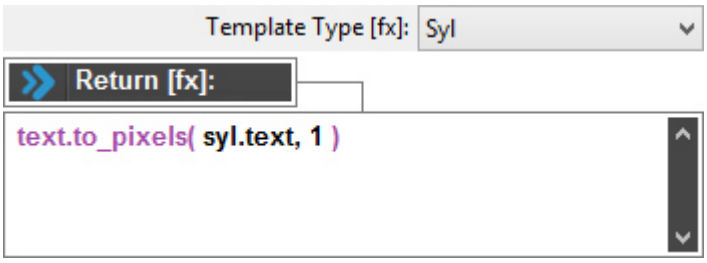
El parámetro **Shape** nos da la opción de que el texto no se convierta en pixeles (**shape.pixel**), sino en la **shape** que asignemos en este parámetro de la función. Su valor por default es **shape.pixel**:



Lo que hace la función es, que por medio de Shapes de 1 x 1 px, remplazar el texto en su tamaño, forma y posición, por todos los pixeles que lo componen. Ejemplo:

Ejemplo:

- **Mode = 1**



En **Modo = 1**, todos los pixeles que componen al texto salen en su posición exacta de tal manera que no se notará la diferencia entre el texto normal y el texto conformado por lo pixeles. Para notar a los pixeles que conforma al texto, añadí cierta cantidad de variación en la posición de los mismos, así:

Y obtendremos algo similar a esto:

Ejemplo:

- **Mode = 2**

Los pixeles se empiezan a desplazar desde el centro del texto, en todas direcciones, y conforman un círculo:

El tiempo total de este desplazamiento está determinado por **fx.dur**

Ejemplo:

- **Mode = 3**

Los pixeles se desplazan de forma muy similar a la anterior, pero ya no tienden a dibujar un círculo, sino una elipse vertical:

Ejemplo:

- **Mode = 4**

Los pixeles ahora se desplazan desde el centro del texto, describiendo tres elipses horizontales alineadas una arriba de la otra:

Ejemplo:

- **Mode = 5**

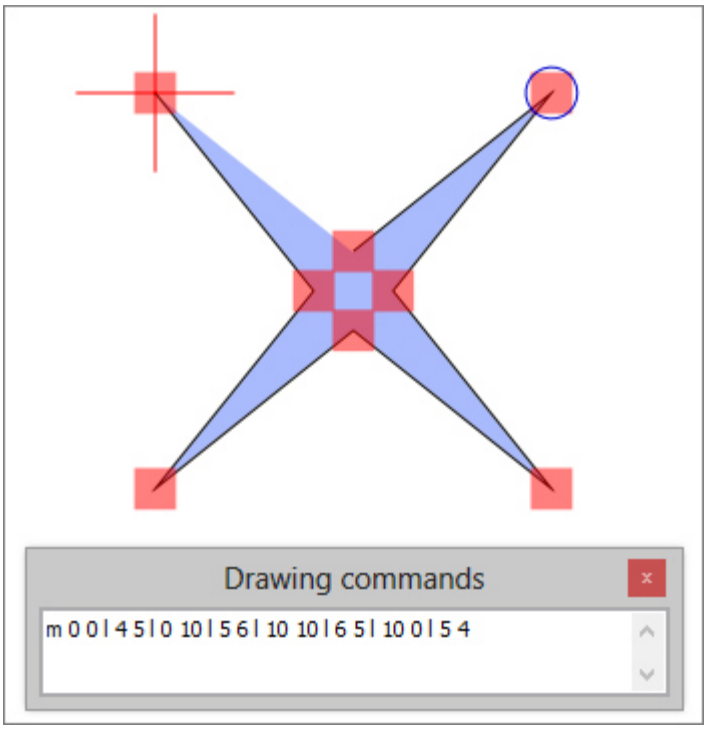
Los pixeles de desplazan describiendo una curva bezier seleccionada al azar:

A partir del modo 1, y con algo de imaginación, se pueden hacer muchas cosas con los pixeles de un texto que genera esta función. Recordemos que con solo variar el tiempo de inicio o el final de una línea fx, se puede modificar el **fx.dur**, lo que le dará una duración diferente a cada uno de los pixeles en el desplazamiento. Es cuestión de experimentar con todo aquello que hasta acá han aprendido.

Hasta este punto, el parámetro **Shape** siempre se usó por default, lo que hacía que la función siempre "pixelará" el texto con la shape cuadrada de 1 x 1 (**shape.pixel**):

Romaji	lead-in	Effector [Fx]	m 0 0 l 0 1 l 1 1 l 1 0 l 0 0
Romaji	lead-in	Effector [Fx]	m 0 0 l 0 1 l 1 1 l 1 0 l 0 0
Romaji	lead-in	Effector [Fx]	m 0 0 l 0 1 l 1 1 l 1 0 l 0 0
Romaji	lead-in	Effector [Fx]	m 0 0 l 0 1 l 1 1 l 1 0 l 0 0
Romaji	lead-in	Effector [Fx]	m 0 0 l 0 1 l 1 1 l 1 0 l 0 0
Romaji	lead-in	Effector [Fx]	m 0 0 l 0 1 l 1 1 l 1 0 l 0 0
Romaji	lead-in	Effector [Fx]	m 0 0 l 0 1 l 1 1 l 1 0 l 0 0

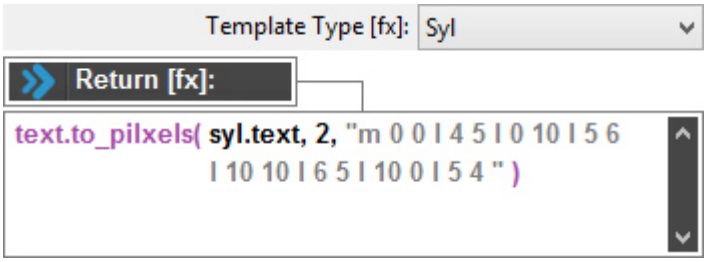
Para el siguiente ejemplo, usaremos esta **shape** en el tercer parámetro de la función, con el fin que el texto ingresado en ella se descomponga por medio de esta **shape**:



Ejemplo:

Y lo seguiremos haciendo en **Template Type: Syl**, aunque esta función puede aplicarse en cualquiera de los modos de fx a aplicar.

Elegí la función en **Mode = 2**, pero ustedes pueden practicar con cualquiera de los ya vistos hasta acá, y pegamos el código de la shape en el tercer parámetro de la función:



Y al aplicar, el texto se descompuso en esta nueva **shape**, lo que da un efecto similar a un brillo:

lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4
lead-in	Effector [Fx]	m 0 0 l 4 5 l 0 10 l 5 6 l 10 10 l 6 5 l 10 0 l 5 4

Este hecho de poder elegir a nuestro antojo a la **shape** en la que "pixelará" el texto ingresado en la función, aumenta en forma exponencial nuestras posibilidades de nuevos e ingeniosos efectos. Tengan en cuenta que esta función genera un loop aproximado entre 7000 y 10000 líneas de fx por cada línea de karaoke o de traducción, a la que se le aplique estos efectos.

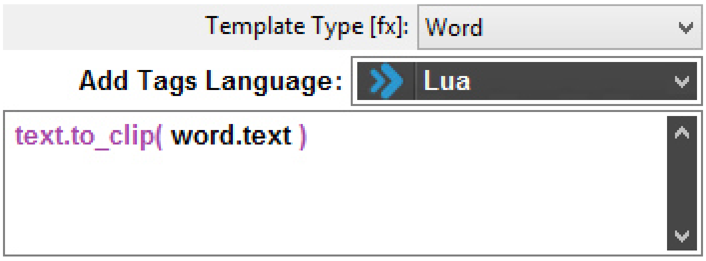
text.to_clip(Text, relative_pos, iclip) :

Esta función retorna un clip con la forma exacta del texto ingresado. El parámetro **Text** es el string de texto a ser convertido en clip y su valor por default es el texto por default según el **Template Type** del efecto aplicado.

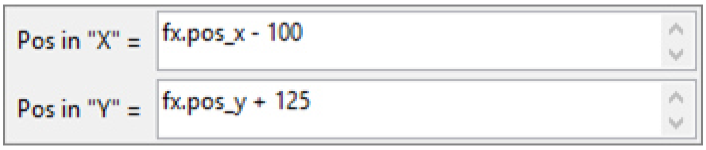
Los parámetros **relative_pos** e **iclip** son dos boleanos, el primero hace que el clip generado tenga la misma posición del texto original sin importar cuál sea su posición inicial y su valor por default es **nil**. El parámetro **iclip** decide si el texto ingresado se convertirá en un clip o en un iclip. Su valor por default es **nil**.

Ejemplo:

Para este ejemplo usamos un **Template Type: Word**



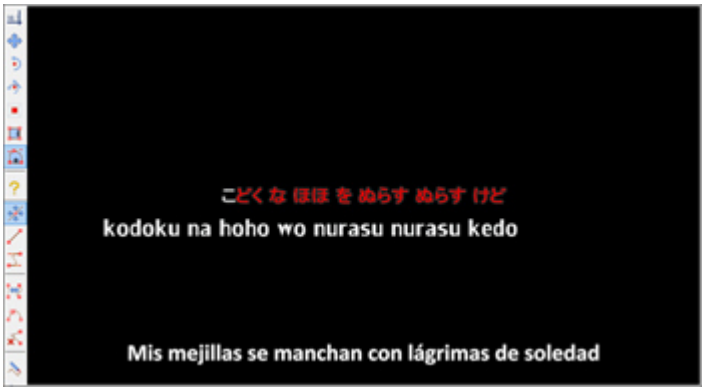
Y modificamos un poco las posiciones por default de cada Palabra generada por el efecto, por ejemplo, así:



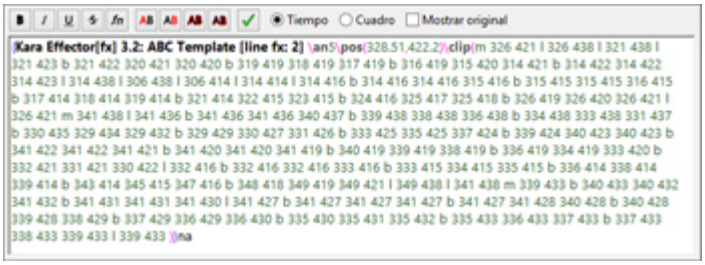
Es decir, 100 pixeles a la izquierda de su centro por default y 125 pixeles hacía abajo.

A aplicar, en apariencia solo veremos el texto en la posición que debería tener dada las modificaciones que previamente le indicamos.

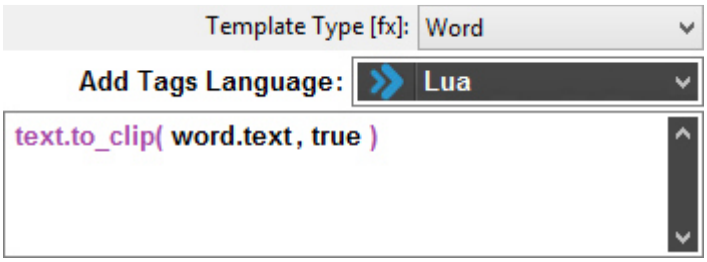
Y vemos que los clip's son generados en la posición en la que se encuentra el texto sin importar que éste no esté en su posición por default: (**word.text, word.middle**)



Para poder apreciar el clip que genera esta función, es necesario que veamos en detalle una de las líneas fx generadas:



Si queremos que el clip generado tenga la posición por default del texto sin importar las modificaciones en las posiciones, ponemos true en el segundo parámetro de la función:



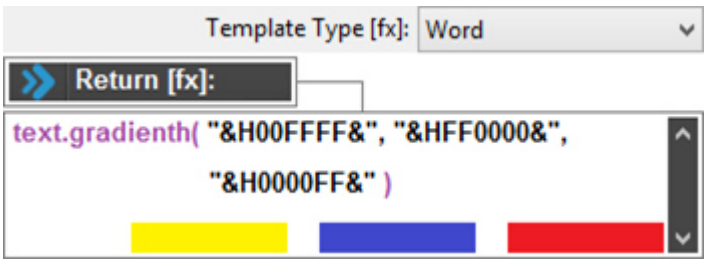
text.gradienth(…) :

Crea un gradiente (degradado) horizontal entre todos los colores ingresados, sin la necesidad de usar el **VSFilterMod** y solo generando una única línea de fx.

La función convierte el texto por default según el **Template Type** del efecto, en un clip y luego genera dentro de él una serie de Shapes consecutivas de manera cada una tenga un color diferente y entre todas juntas conformen en gradiente horizontal.

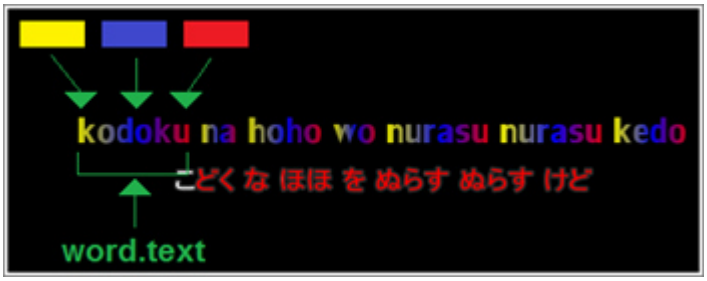
Ejemplo:

Para este ejemplo usaremos un **Template Type: Word**, y pondremos tres colores en la función, pero la cantidad total no tiene límites:



Recordemos que para crear un gradiente horizontal o vertical, la mínima cantidad de colores a ingresar en la función es de dos.

Al aplicar el ejemplo, notamos cómo cada palabra (Word) contiene un gradiente entre los tres colores ingresados:



Si ampliamos un poco la imagen veremos cómo la función crea el gradiente entre los tres colores, en cada palabra de la línea karaoke:

Una de las ventajas de esta función es que crea el gradiente horizontal entre todos los colores ingresados, generando una única línea de fx, en este ejemplo, por cada palabra de cada línea a la que se le aplica el efecto:

25	3	0:00:45.92	0:00:54.56	English			Dos corazones que se buscan cor
26		0:00:02.43	0:00:08.16	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
27		0:00:02.43	0:00:08.16	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
28		0:00:02.43	0:00:08.16	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
29		0:00:02.43	0:00:08.16	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
30		0:00:02.43	0:00:08.16	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
31		0:00:02.43	0:00:08.16	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
32		0:00:02.43	0:00:08.16	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
33		0:00:08.33	0:00:13.19	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m
34		0:00:08.33	0:00:13.19	Romaji	lead-in	Effector [Fx]	*m 0 0 1 0 74 1 2 74 1 2 0 1 0 0 *m

Esta ventaja hace que el **script** sea más liviano y sea leído fácilmente, en el caso de los subtítulos en modo "**Softsub**".

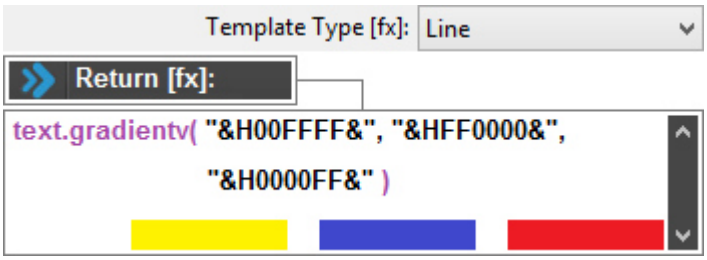
text.gradientv(…) :

Crea un gradiente (degradado) vertical entre todos los colores ingresados, de manera similar a la función anterior, sin la necesidad de usar el **VSFilterMod** y solo generando una única línea de fx.

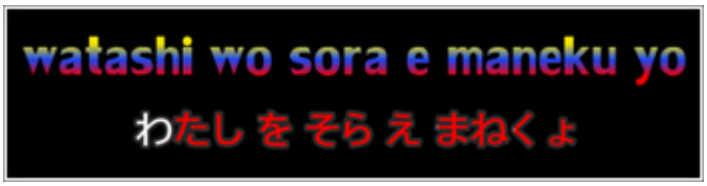
La función convierte el texto por default según el **Template Type** del efecto, en un clip y luego genera dentro de él una serie de Shapes consecutivas de manera cada una tenga un color diferente y entre todas juntas conformen en gradiente vertical.

Ejemplo:

Con un **Template Type: Line**, y en **Return [fx]** escribimos lo siguiente:



Y al aplicar:



text.inside(inside, Text) :

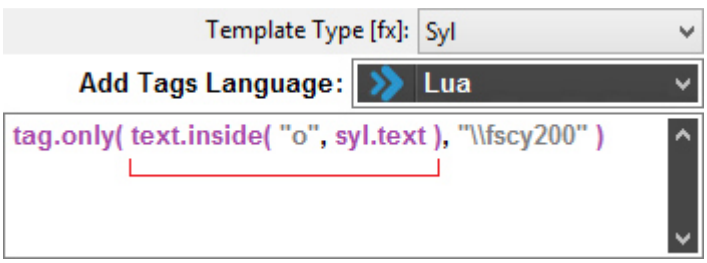
Esta función retorna **true** o **false** (verdadero o falso) al verificar si un **string** de texto del parámetro **inside**, hace parte parcial o total de otro **string** ingresado en el parámetro **Text**.

El parámetro **inside** es el string que la función buscará para determinar si el resultado es verdadero o falso.

El parámetro **Text** es el texto y/o **string** en el que la función buscará al parámetro **inside** para determinar si éste hace parte o no de él. Su valor por default es el texto por default dependiendo del **Template Type**.

La particularidad que tiene esta función de retornar solo **true** o **false** le impide que sea llamada directamente en un efecto ya que ambos son valores booleanos, pero sí la podemos usar como una condición de verdad dentro de otras funciones.

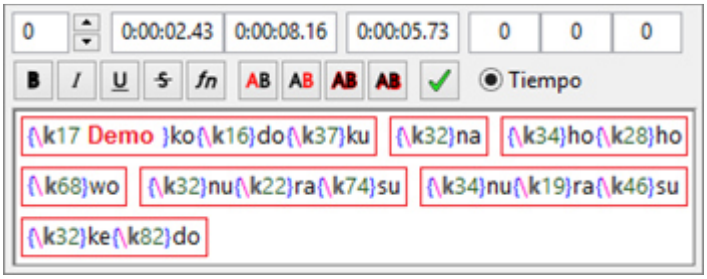
Ejemplo:



La función **text.inside**
verifica si la letra “o” hace parte de cada una de las sílabas, en caso de ser verdadero, aplica el tag de la función **tag.only**:

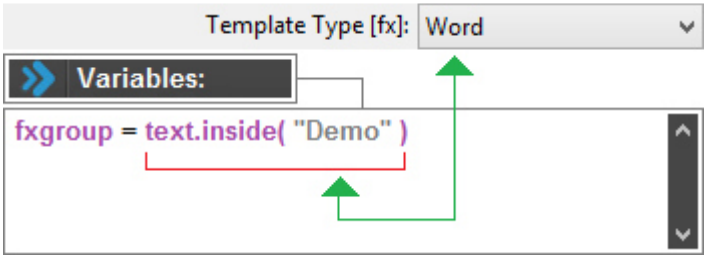
Ejemplo:

Ponemos una marca en una de los tags karaoke de una o más sílabas:



En la imagen anterior se resaltan las palabras de una línea karaoke, y en el tag karaoke de la sílaba “**ko**” puse una palabra “**Demo**“.

Esa marca la usaremos para aplicar un efecto únicamente a la palabra que contenga dicho **string**:



Y al aplicar veremos cómo se generó una única línea fx correspondiente a la palabra “**kodoku**” que era la única a la que le incluí el **string** “**Demo**“:

Estilo	Actor	Efecto	Texto
English			Me pierdo en el camino cada vez que
English			Siento los pensamientos que dejaste
English			Me aferraré a ti y nunca te soltaré
English			Dos corazones que se buscan confor
Romaji	lead-in	Effector [Fx]	✱kodoku

La ventaja de esta función es que puede verificar si un string del parámetro **inside** está, no solo en el texto de un objeto karaoke de la línea, sino que también dentro de sus tags. Esta ventaja la podemos usar para poner marcas en el script y luego aplicar uno o más efectos especiales solo en donde las pusimos. La cantidad de marcas y la diversidad de ellas, depende solo de la necesidad de cada uno.