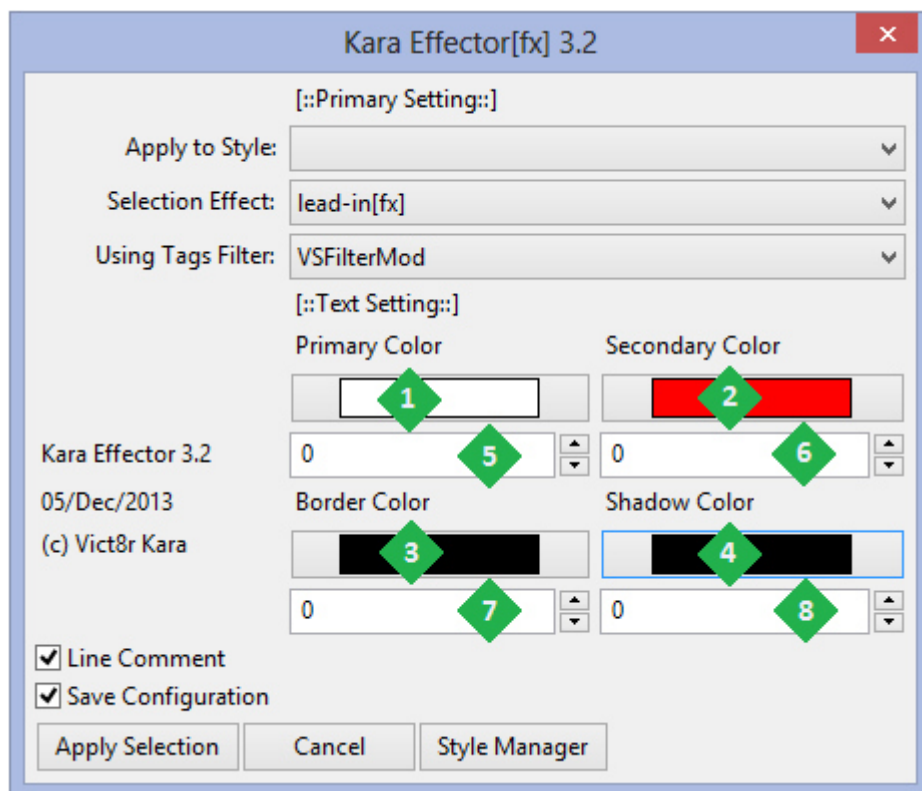


Librería “fx” [KE]

Es la Librería que contiene los valores del Kara Effector.



1. **text.color1**
2. **text.color2**
3. **text.color3**
4. **text.color4**
5. **text.alpha1**
6. **text.alpha2**
7. **text.alpha3**
8. **text.alpha4**

Line Start Time = 1

Line End Time = 2

x(s) = 3

y(s) = 4

s = 5 to 6

Layer = 7

Return [fx]:

loop = 8

Size = 9

Shape Colors [fx]:

Primary Color Shape	Border Color Shape	Shadow Color Shape
<input type="text" value=""/> 10	<input type="text" value=""/> 11	<input type="text" value=""/> 12
<input type="text" value="0"/> 13	<input type="text" value="0"/> 14	<input type="text" value="0"/> 15

Template Type [fx]:

Center in 'X' = 16

Center in 'Y' = 17

Scale in 'X' = 18

Scale in 'Y' = 19

Align [\an] = 20

Pos in 'X' = 21

Pos in 'Y' = 22

Times Move = 23

Add Tags: Add Tags Language:

☐ Print Config [fx] Template Folder [fx]:

New [fx] Name:

1. **fx.start_time**
2. **fx.end_time**
3. **fx.fun_x**
4. **fx.fun_y**
5. **fx.domain_i**
6. **fx.domain_f**
7. **fx.layer**
8. **fx.maxloop_fx**
9. **fx.size_x, fx.size_y**
10. **shape.color1**
11. **shape.color3**
12. **shape.color4**
13. **shape.alpha1**
14. **shape.alpha3**
15. **shape.alpha4**
16. **fx.center_x**
17. **fx.center_y**
18. **fx.scale_x**
19. **fx.scale_y**
20. **fx.align**
21. **fx.move_x1, fx.move_x2, fx.move_x3, fx.move_x4**
22. **fx.move_y1, fx.move_y2, fx.move_y3, fx.move_y4**
23. **fx.movement_i, fx.movement_f**

No parecen ser muchas, pero son suficientes. Ahora veremos una pequeña explicación de cada una de ellas.

Bueno, considero que los valores de la ventana de inicio del **Kara Effector** no necesitan de mucha explicación ya que son los colores y transparencias que tendrán por default cada una de las Líneas de Efecto generadas.

fx.start_time: es el tiempo de inicio de cada una de las Líneas generadas por un Efecto. Puede ser modificado directamente desde su respectiva celda de texto o con la función **retime** desde “**Add Tags**” o alguna función hecha en “**Variables**”.

fx.end_time: es el tiempo final de cada una de las Líneas generadas por un Efecto. Este valor puede ser modificado directamente desde su respectiva celda de texto o con la función **retime** desde “**Add Tags**” o alguna función hecha en “**Variables**”. La variable **fx.start_time** puede ser usada en esta celda de texto, ejemplo:

Line Start Time =	<code>l.start_time + math.random(-2000, 2000)</code>
Line End Time =	<code>fx.start_time + 500</code>

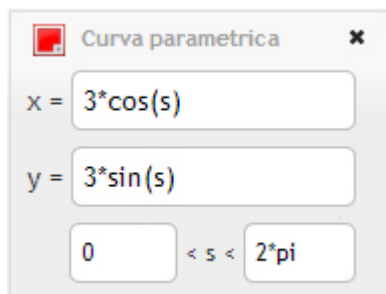
O sea que **fx.start_time** es un valor aleatorio entre -2000 ms y 2000 ms respecto al tiempo de inicio original de cada Línea seleccionada para aplicarle un Efecto (**l.start_time**).

Por otro lado, **fx.end_time** pone que será igual al tiempo en donde inició la Línea de fx (**fx.start_time**), sumado a 500 ms. Es fácil deducir que el **fx.dur** de todas las Líneas que se generen con estos dos tiempos, siempre será de 500 ms. (véase la siguiente variable).

fx.dur: es la duración total de cada Línea fx que sea generada por un Efecto. Es equivalente a la siguiente diferencia:

$$\text{fx.dur} = \text{fx.start_time} - \text{fx.end_time}$$

fx.fun_x: es la ecuación paramétrica de “x” en términos de “s” (donde “s” es el dominio de la función).
Ejemplo:



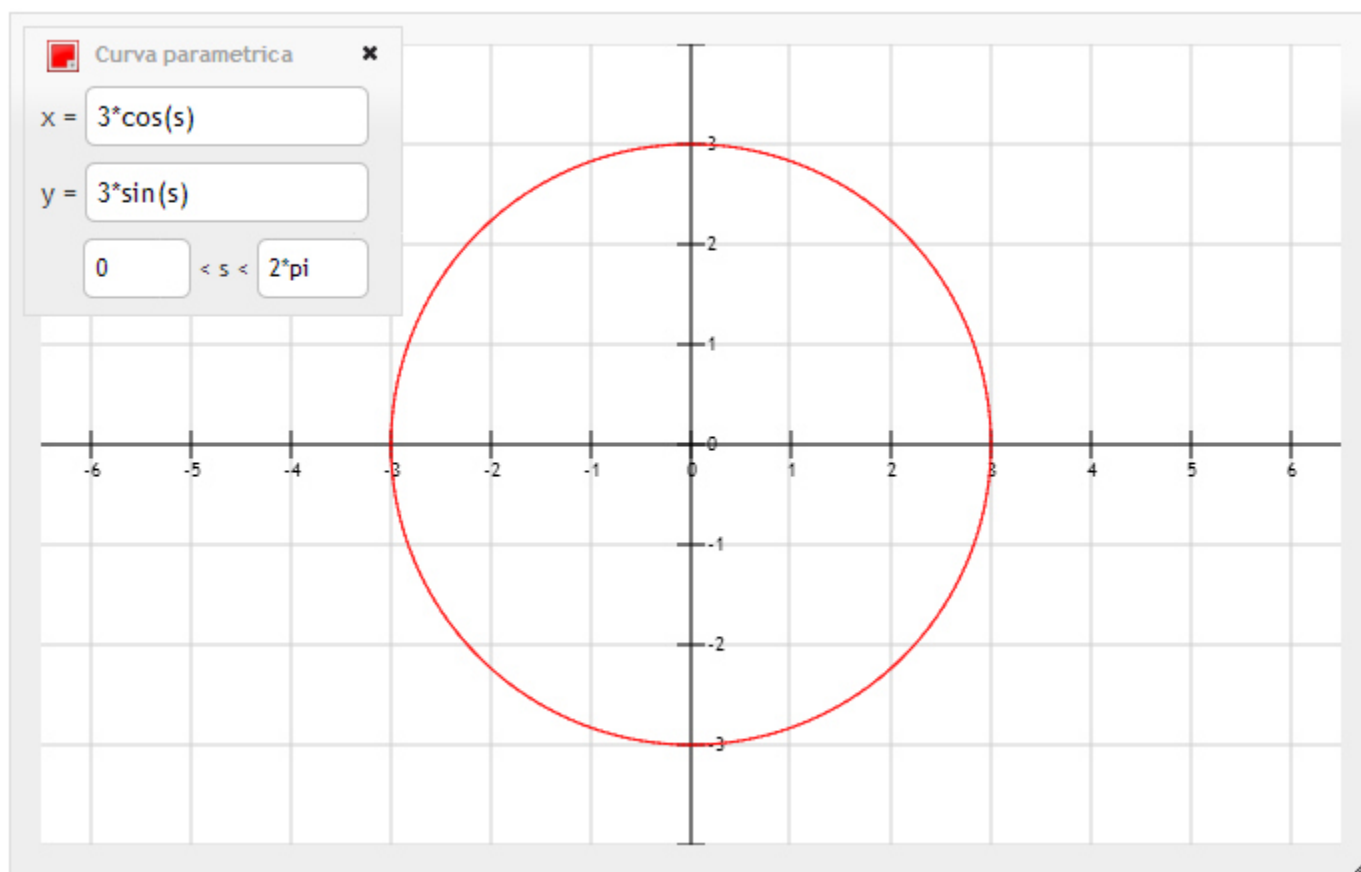
Curva parametrica

x = $3 \cdot \cos(s)$

y = $3 \cdot \sin(s)$

0 < s < 2*pi

En la anterior imagen están las ecuaciones paramétricas de “x” y “y” de un Círculo de Radio 3. También podemos ver el dominio “s” que va desde 0 a 2*pi:



La anterior gráfica fue generada en <http://fooplot.com>

En el Kara Effector pondríamos así:

x(s) =	<input type="text" value="3*cos(s)"/>	^ v	
y(s) =	<input type="text" value="3*sin(s)"/>	^ v	
s =	<input type="text" value="0"/>	^ v	
	to	<input type="text" value="2*pi"/>	^ v

Las escalas en el **Kara Effector**, tanto en el eje “x” como en el “y” son por default 1, es decir que este círculo se verá en el vídeo muy pequeño, ya que el Radio del Círculo es solo de 3 pixeles. Hay dos formas de aumentar su tamaño: o se aumenta el Radio del Círculo directamente en las ecuaciones paramétricas o se aumentan las escalas en las celdas de texto que están al lado derecho de estas, todo depende del gusto y del nivel de habilidad adquirido de cada uno.

En estas celdas de texto podemos modificar las Escalas en ambos ejes de una determinada gráfica:

Scale in 'X' =	<input type="text"/>	^ v
Scale in 'Y' =	<input type="text"/>	^ v

Olvidaba mencionar que este tipo de Efecto se hacen con Shapes y con un **loop** lo suficientemente grande para que se generen la gráficas.

Para un ejemplo guiado en el **Kara Effector** tomando como base las ecuaciones paramétricas del Círculo, sería:

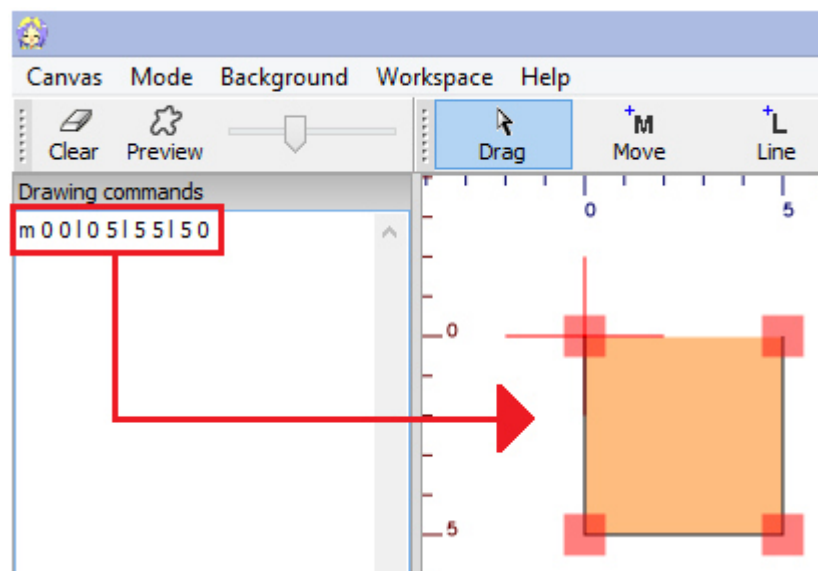
A. Definimos las ecuaciones paramétricas y el dominio de las funciones (“s”):

x(s) =	<input type="text" value="3*cos(s)"/>		
y(s) =	<input type="text" value="3*sin(s)"/>		
s =	<input type="text" value="0"/>	to	<input type="text" value="2*pi"/>

B. Aumentamos las escalas en 10 para que el Radio del Círculo pase de 3 pixeles a $3*10 = 30$ pixeles. (Se puede experimentar con distintos valores en las escalas y ver cómo varían las gráficas):

Scale in 'X' =	<input type="text" value="10"/>
Scale in 'Y' =	<input type="text" value="10"/>

C. Elegimos una **Shape** y la ponemos en **Return [fx]**:



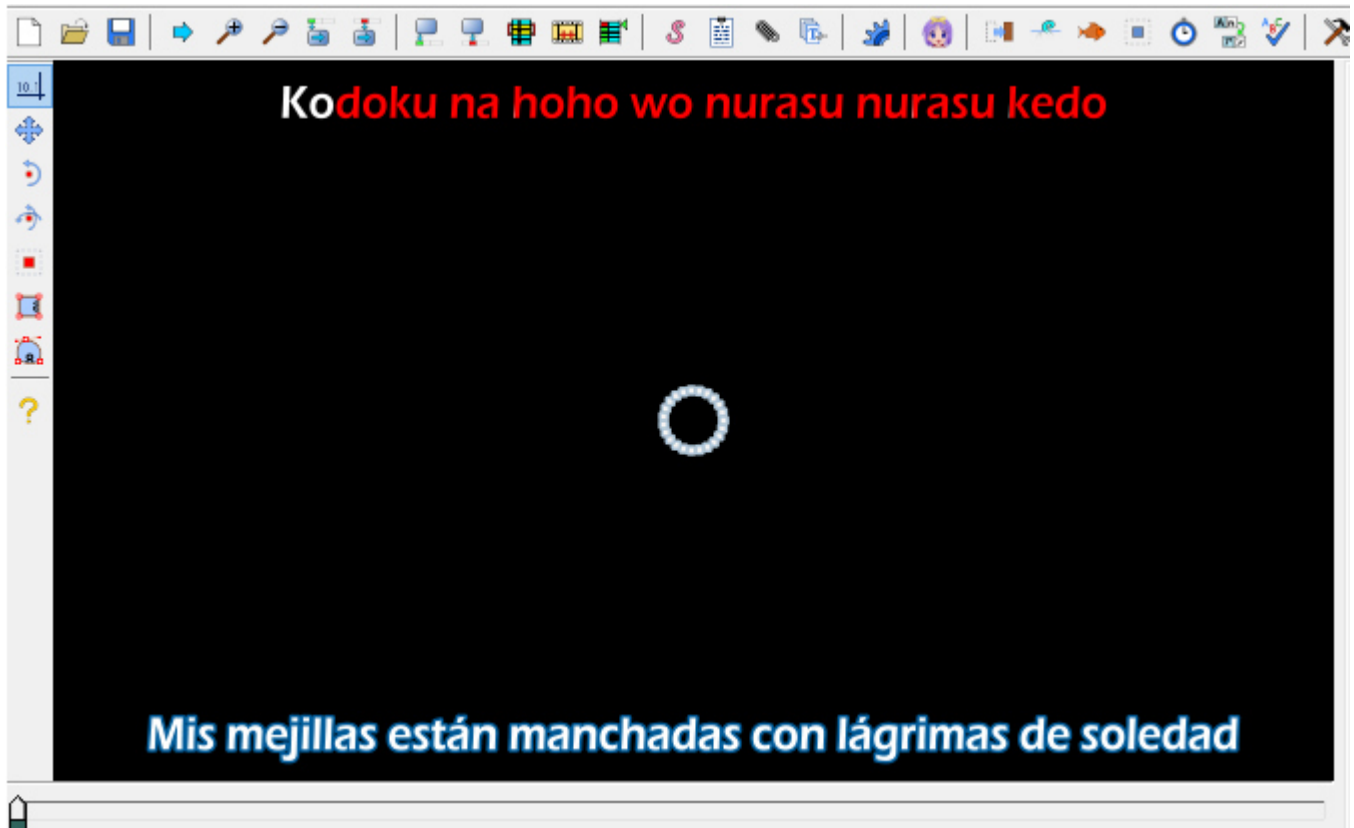
En mi caso, dibujé un Cuadrado en el **AssDraw3**, de 5 pixeles de lado y copié el código de la Shape en **Return [fx]**:

Return [fx]:	<input type="text" value="m 0 0 1 0 5 1 5 5 1 5 0"/>
--------------	--

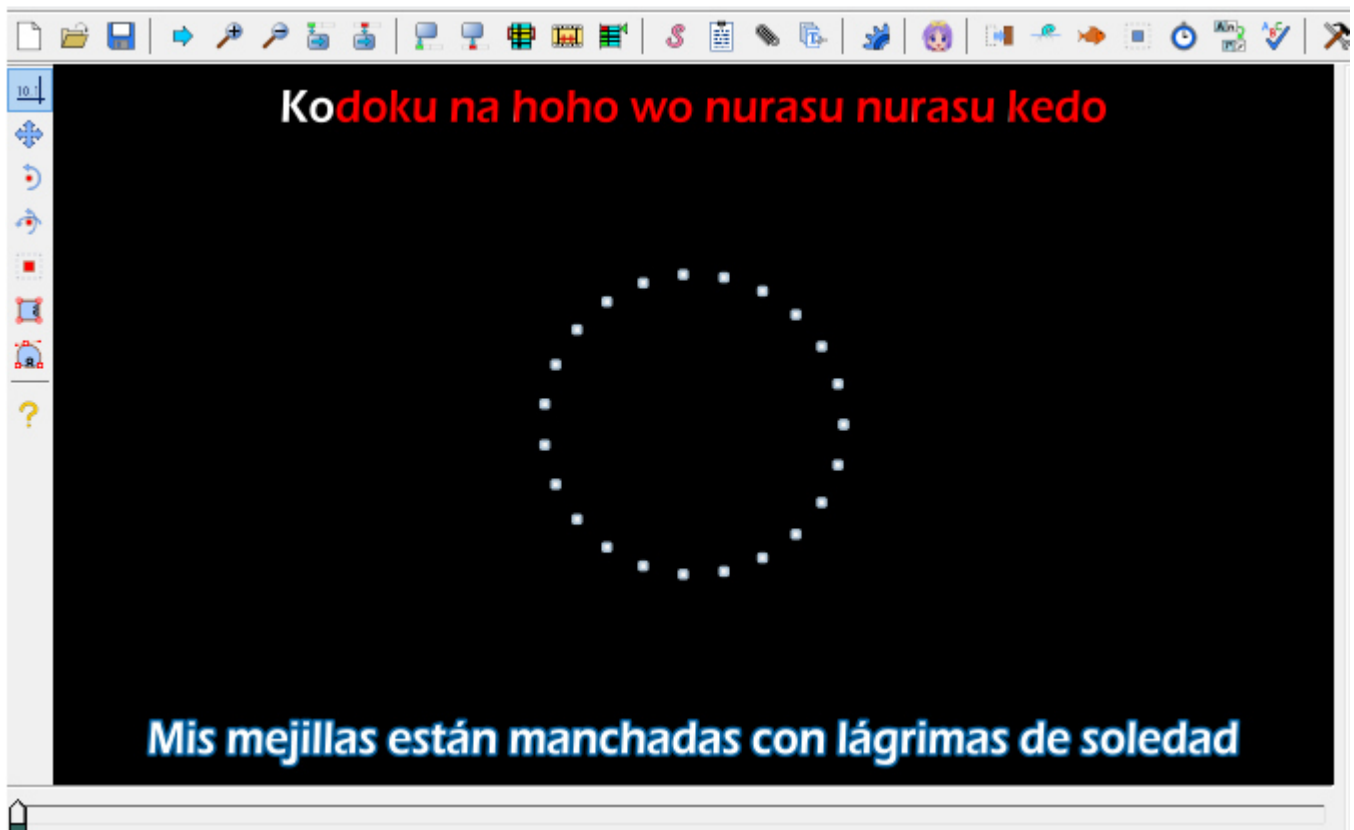
D. Aumentamos el **loop** para que generará la gráfica:

loop =	<input type="text" value="24"/>
--------	---------------------------------

Para este ejemplo puse en **Template Type** la opción “**Line**” para que se genere un solo Círculo por cada Línea:



Ya se puede ver en el vídeo el Círculo que se generó, pero parece que 30 pixeles para el Radio es poco para poder ver los 24 cuadraditos (loop) de 5 X 5 pixeles que lo conforman, por ello aumentaré las escalas en ambos ejes a 50 y volveré a aplicar el Efecto:



Ahora sí se pueden apreciar los cuadraditos que generan el Círculo. La cantidad de cuadrados, lo mismo que el valor de las escalas, tamaños (**size**) y colores, se pueden modificar para que la gráfica sea más continua, ejemplo:

Es un ejemplo de cómo “graficar” en el **Kara Effector**.

Las configuraciones para generar el anterior Círculo son:

lead-in[fx]: ABC Template

Line Start Time =

Line End Time =

x(s) =

y(s) =

s = to

Layer =

Return [fx]:

Pos in 'X' =

Pos in 'Y' =

Times Move =

loop =

Size =

Shape Colors [fx]: Primary Color Shape Border Color Shape Shadow Color Shape

Template Type [fx]:

Center in 'X' =

Center in 'Y' =

Scale in 'X' =

Scale in 'Y' =

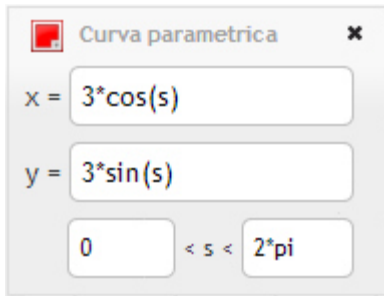
Align [\an] =

Pos in 'X' =

Pos in 'Y' =

En próximos ejemplos veremos cómo hacer un Efecto a partir de la gráfica de las ecuaciones paramétricas.

fx.fun_y: es la ecuación paramétrica de “y” en términos de “s” (donde “s” es el dominio de la función).
Ejemplo:



Curva parametrica

x = $3 \cdot \cos(s)$

y = $3 \cdot \sin(s)$

0 < s < $2 \cdot \pi$

Con el ejemplo anterior del Círculo, esta variable de la Librería “**fx**” no necesita muchas más explicación y por ello iremos directamente a las siguientes variables.

fx.domain_i: es el inicio del dominio “s” de las ecuaciones paramétricas. Si en esta celda de texto no pone nada, el inicio del dominio por default es 0.

fx.domain_f: es el final del dominio “s” de las ecuaciones paramétricas. Si en esta celda de texto no pone nada, el final del dominio por default es 1.

No necesariamente el inicio del dominio “s” debe ser menor que el final, lo convencional es que sí lo sea para que la gráfica se dibuje normalmente de menor a mayor, de lo contrario se dibujará al revés. Este orden cobra importancia cuando hacemos animaciones con las gráficas y las vemos en el vídeo.

fx.layer: es la capa de cada una de las Líneas de fx y decide la prioridad en el vídeo de dos o más Objetos Karaoke que coinciden de forma total o parcial en su posición o trayectoria de movimiento.

fx.maxloop_fx: este valor equivale a la cantidad total de repeticiones de cada una de las Líneas fx. Se puede usar con este nombre o con uno más conocido por aquellos que ya conocen algo de **Automation Auto-4: maxj**

maxj = fx.maxloop_fx

El valor de **maxj** (Máximo valor de “j”) se puede modificar directamente en la celda de texto destinada para ello o con la función **maxloop**.

En el caso de haber un solo valor en esta celda, el valor de **maxj** es ese mismo valor:



loop = 2

En este caso, **maxj** = 2

Para el caso de dos valores:



loop = 2, 5

Para este caso, **maxj** = 2 x 5 = 10

Y para el caso de que haya tres valores:



loop = 2, 5, 4

Y para este último caso, **maxj** = 2 x 5 x 4 = 40

3 es el número máximo de valores que se pueden poner en esta celda de texto. Cuando alguno de ellos o todos, no están, su valor por default es 1.

Más adelante veremos por qué la necesidad de más de un valor en esta celda de texto, pero por ahora les mostraré la variable asignada a cada uno de ellos, ejemplo:

A screenshot of a text input field with a light gray border. To the left of the field is the label "loop =". Inside the field, the text "2, 5, 4" is entered. To the right of the field is a small vertical scrollbar with up and down arrows.

fx.loop_v: (el 2 en la imagen) es el loop vertical.

fx.loop_h: (el 5 en la imagen) es el loop horizontal.

fx.loop_3: (el 4 en la imagen) es un loop de respaldo.

Como lo mencionaba antes, si alguno de estos tres valores no está, por default es 1. Lo que quedaría:

fx.maxloop_fx = **fx.loop_v** * **fx.loop_h** * **fx.loop_3**

maxj = **fx.loop_v** * **fx.loop_h** * **fx.loop_3**

maxj = **fx.maxloop_fx**

fx.size_x: es el porcentaje del tamaño con respecto al eje “x” asociado al tag \fscx, ejemplo:

A screenshot of a text input field with a light gray border. To the left of the field is the label "Size =". Inside the field, the text "80" is entered. To the right of the field is a small vertical scrollbar with up and down arrows.

En este ejemplo, **fx.size_x** valdría 80 y en el Efecto se verán los siguientes tags: \fscx80\fscy80

Es decir que si solo hay un valor en esta celda de texto, el porcentaje en el eje “y” será el mismo que para el eje “x”, o sea 80% para los dos ejes.

fx.size_y: es el porcentaje del tamaño con respecto al eje “y” asociado al tag \fscy, ejemplo:

A screenshot of a text input field with a light gray border. To the left of the field is the label "Size =". Inside the field, the text "80, 125" is entered. To the right of the field is a small vertical scrollbar with up and down arrows.

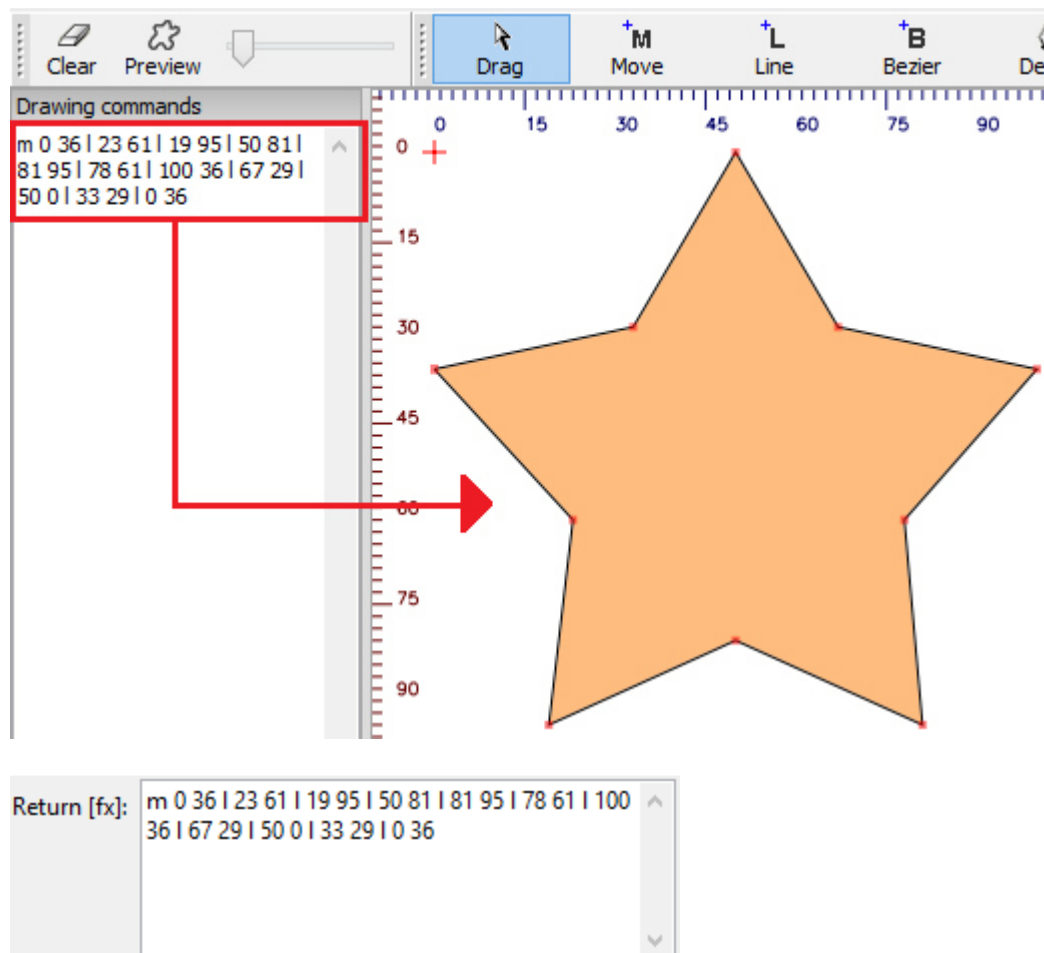
O sea: **fx.size_x** = 80 y **fx.size_y** = 125

Y en tags: \fscx80\fscy125

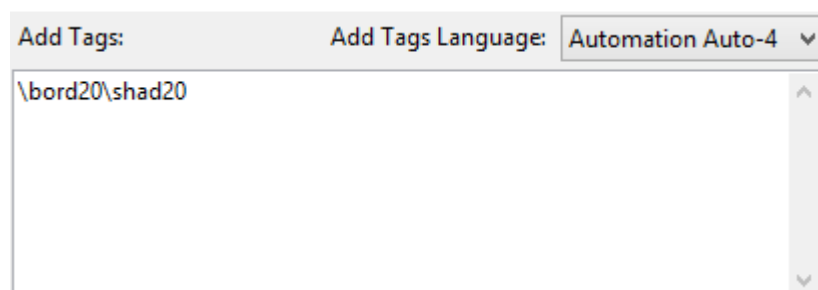
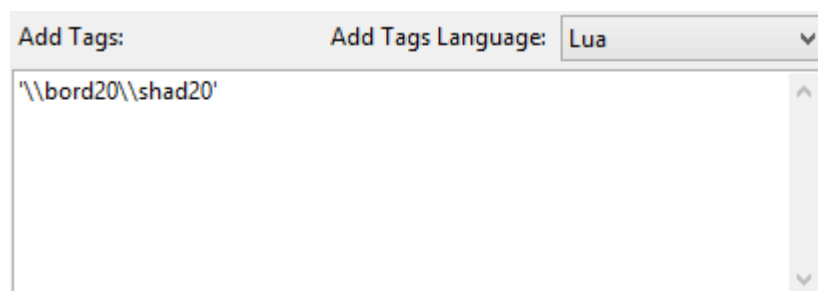
Si esta celda de texto está vacía los valores por default de estas dos variables son:

fx.size_x = l.scale_x y **fx.size_y = l.scale_y**

Para las variables de la Librería “fx” concernientes a las figuras hechas en el **AssDraw3** usaré la siguiente Shape:


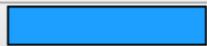



Y le agregaré los siguientes tags:

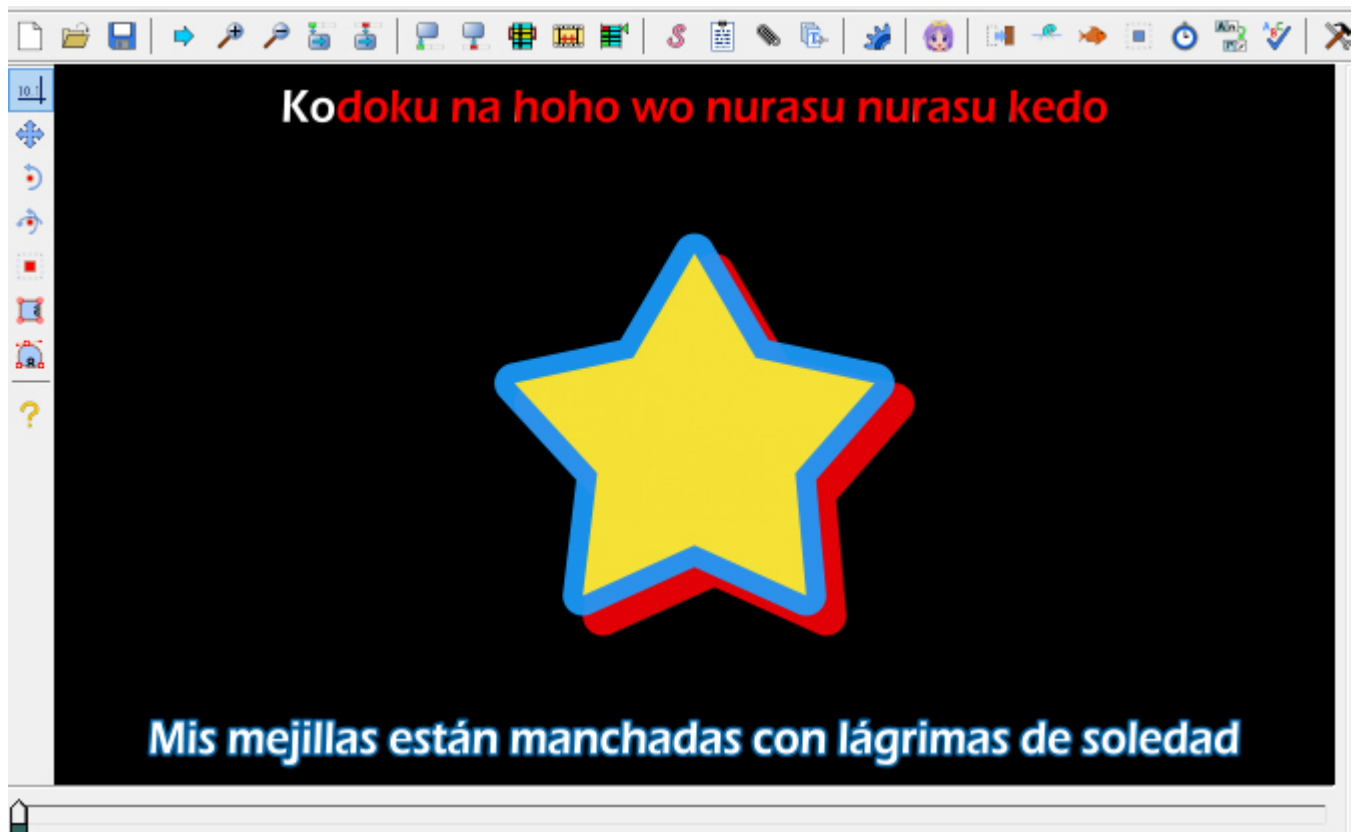


O sea, 20 pixeles para el tamaño del borde y otros 20 para el tamaño de la sombra.

Además de esto, las siguientes configuraciones:

loop =	1	
Size =	360	
Primary Color Shape	Border Color Shape	Shadow Color Shape
		
10	20	30

Y al aplicar el Efecto, se verá en pantalla la Shape con las configuraciones que le di:



shape.color1: es el color Primario de la Shape (amarillo).

shape.color3: es el color del Borde de la Shape (azul).

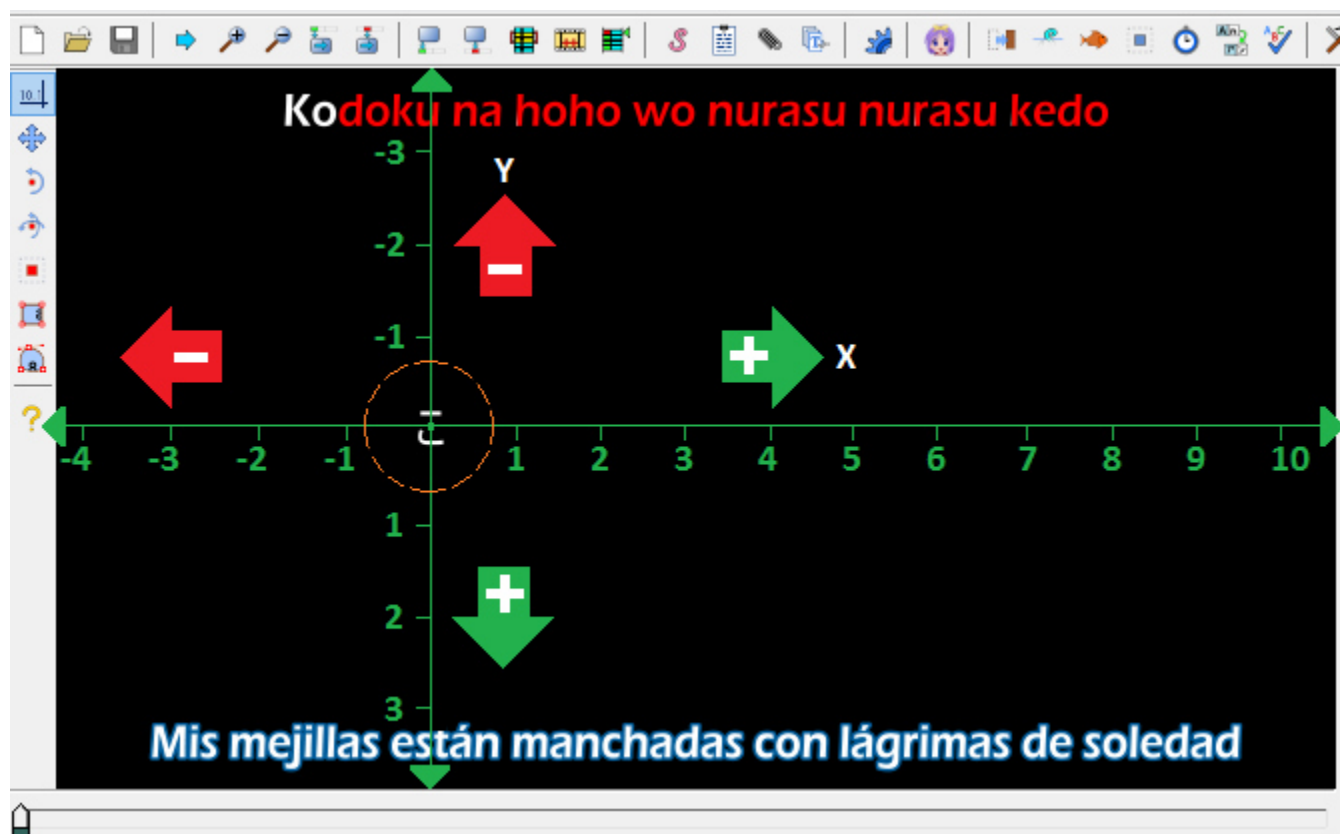
shape.color4: es el color de la Sombra de la Shape (rojo).

shape.alpha1: es la transparencia del color Primario de la Shape (10 para este ejemplo).

shape.alpha3: es la transparencia del color del Borde de la Shape (20).

shape.alpha4: es la transparencia del color de la Sombra de la Shape (30).

fx.center_x: es el Centro medido en pixeles con respecto al eje “x” y hace las veces de la Abscisa al Origen de un sistema de coordenadas cartesianas para cada Línea de fx.



En la imagen anterior dibujé las coordenadas cartesianas con el origen en el centro del Hiragana “ko”:

こ

La coordenada en “x” de ese origen sería **syl.center** que como su nombre lo indica, es el centro de la Sílabla. Nótese que en los archivos .ass, con respecto al eje “y”, hacia arriba es negativo y hacia abajo es positivo, es decir que este eje está invertido.

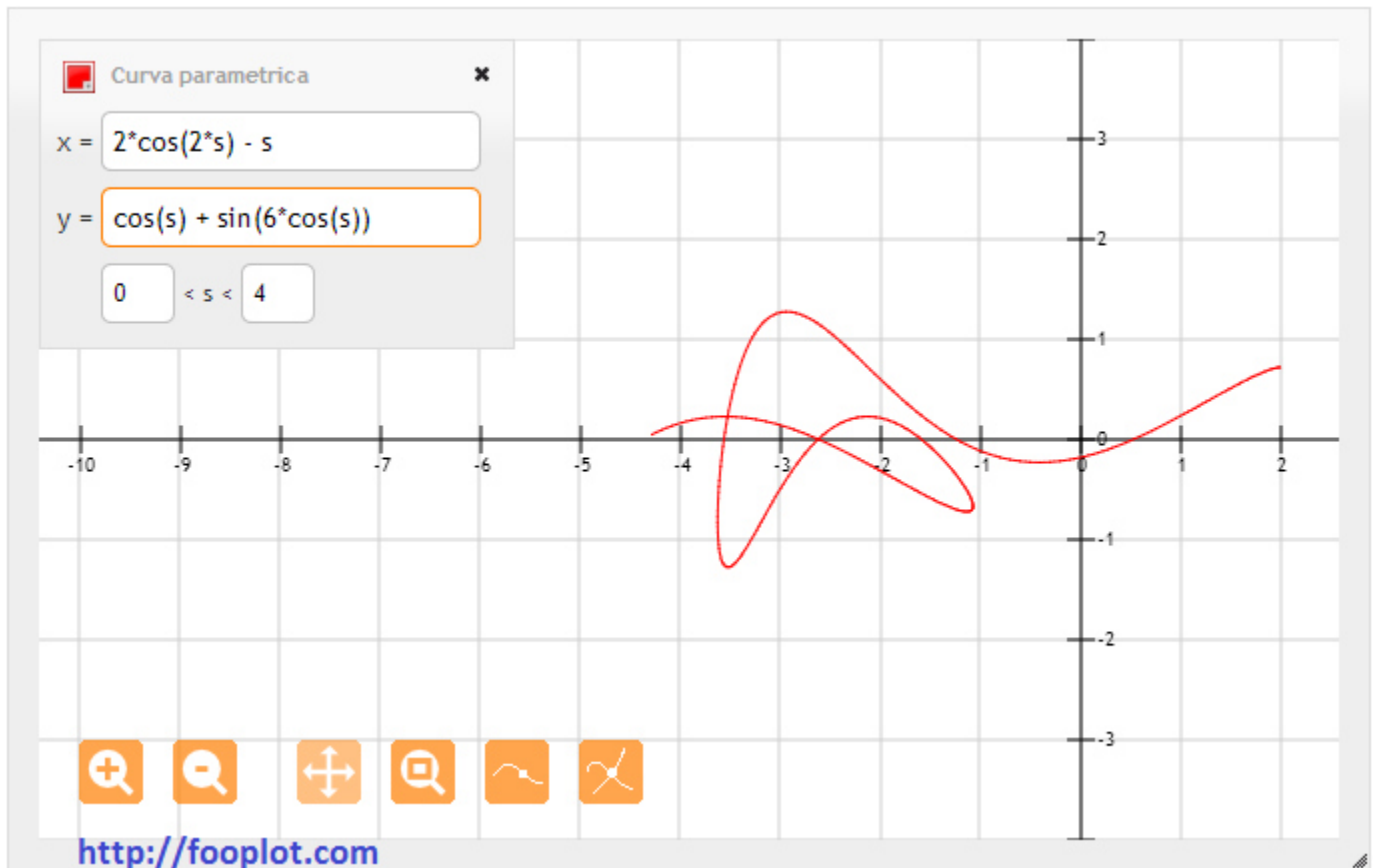
fx.center_y: es el Centro medido en pixeles con respecto al eje “y” y hace las veces de la Ordenada al Origen de un sistema de coordenadas cartesianas para cada Línea de fx.

Como podemos ver en el ejemplo del Hiragana “ko”, sería **syl.middle**, que es el centro de la Sílabla, pero medido verticalmente.

syl.center y **syl.middle** son variables de la Librería “syl” que originalmente ya viene en el Aegisub por default, pero que también explicaré en los próximos tomos, ya que he agregado algunas variables más a esta Librería y es necesario que sepamos en qué consisten dichas variables.

fx.scale_x: junto a **fx.scale_y**, es la Escala con respecto al eje “x” de las gráficas que se generen con ecuaciones paramétricas.

fx.scale_y: es la Escala con respecto al eje “y” de las gráficas que se generen con ecuaciones paramétricas.



Acá hay otro ejemplo de una Gráfica generada por ecuaciones paramétricas, y son este tipo de gráficas las que serán afectadas por las anteriores Escalas. El valor por default de ambas Escalas es 1.

Una vez conocidas las variables **fx.center_x**, **fx.center_y**, **fx.fun_x**, **fx.fun_y**, **fx.scale_x** y **fx.scale_y**; es importante que veamos las siguientes variables:

- **fx.pos_x:** es la coordenada en “x” de la posición final de la Línea fx luego de haber asignado valores a las variables **fx.center_x**, **fx.center_y**, **fx.fun_x**, **fx.fun_y**, **fx.scale_x** y **fx.scale_y**.

$$\text{fx.pos_x} = \text{fx.center_x} + \text{fx.fun_x} * \text{fx.scale_x}$$

- **fx.pos_y:** es la coordenada en “y” de la posición final de la Línea fx luego de haber asignado valores a las variables **fx.center_x**, **fx.center_y**, **fx.fun_x**, **fx.fun_y**, **fx.scale_x** y **fx.scale_y**.

$$\text{fx.pos_y} = \text{fx.center_y} + \text{fx.fun_y} * \text{fx.scale_y}$$

Veamos un ejemplo para tener claro lo de la Posición final:

	Center in 'X' =	syl.center
	Center in 'Y' =	syl.middle
x(s) =	cos(s)	Scale in 'X' = 2
y(s) =	sin(s)	Scale in 'Y' = 4
Pos in 'X' =	fx.pos_x	
Pos in 'Y' =	fx.pos_y	

Para este ejemplo obtendríamos los siguientes valores:

$$\mathbf{fx.pos_x} = \mathbf{syl.center} + \cos(s) * 2$$

$$\mathbf{fx.pos_y} = \mathbf{syl.middle} + \sin(s) * 4$$

El valor por default de **fx.fun_x** y **fx.fun_y** es 0, teniendo en cuenta esto, para el siguiente ejemplo tenemos:

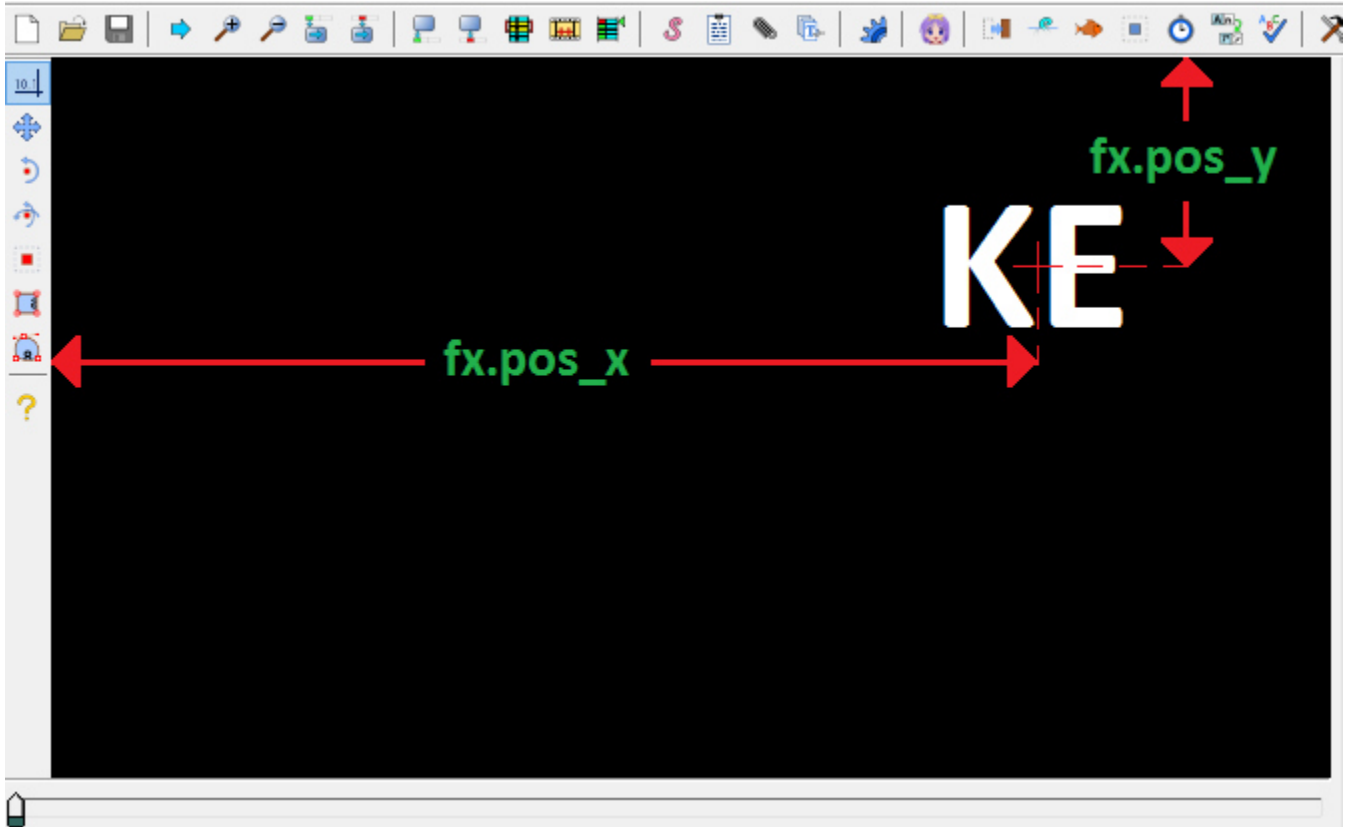
	Center in 'X' =	syl.center
	Center in 'Y' =	syl.middle
x(s) =		Scale in 'X' =
y(s) =		Scale in 'Y' =
Pos in 'X' =	fx.pos_x	
Pos in 'Y' =	fx.pos_y	

$$\mathbf{fx.pos_x} = \mathbf{syl.center} + 0 * 1 = \mathbf{syl.center}$$

$$\mathbf{fx.pos_y} = \mathbf{syl.middle} + 0 * 1 = \mathbf{syl.middle}$$

Los ceros de las anteriores ecuaciones corresponden a los valores por default de **fx.fun_x** y **fx.fun_y**, ya que en la imagen ambas celdas están vacías. Los unos son los valores por default de **fx.scale_x** y **fx.scale_y**, dado que también están vacías sus respectivas celdas de texto.

Entonces concluimos que **fx.pos_x** y **fx.pos_y** son los centros de la posición final de cada una de las Líneas fx:

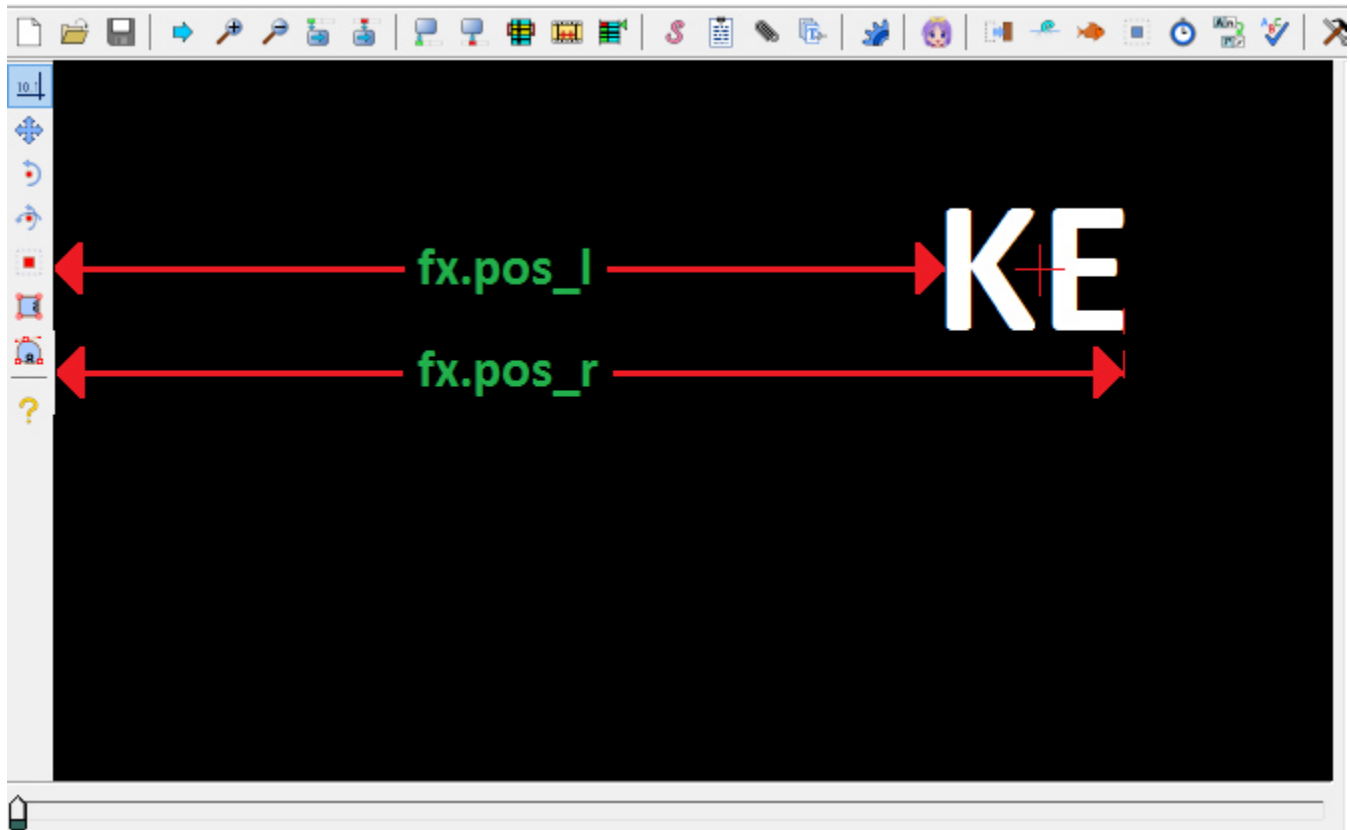


Una vez aclarados los valores de las anteriores dos variables, será más simple entender las cuatro siguientes, ya que dependen directamente de ellas.

fx.pos_l y **fx.pos_r**: (left y right) son la parte izquierda y derecha de **fx.pos_x** respectivamente.

La izquierda y la derecha de **fx.pos_x** dependen del “**Template Type**”, es decir que depende del tipo de plantilla del Efecto. Ejemplo:

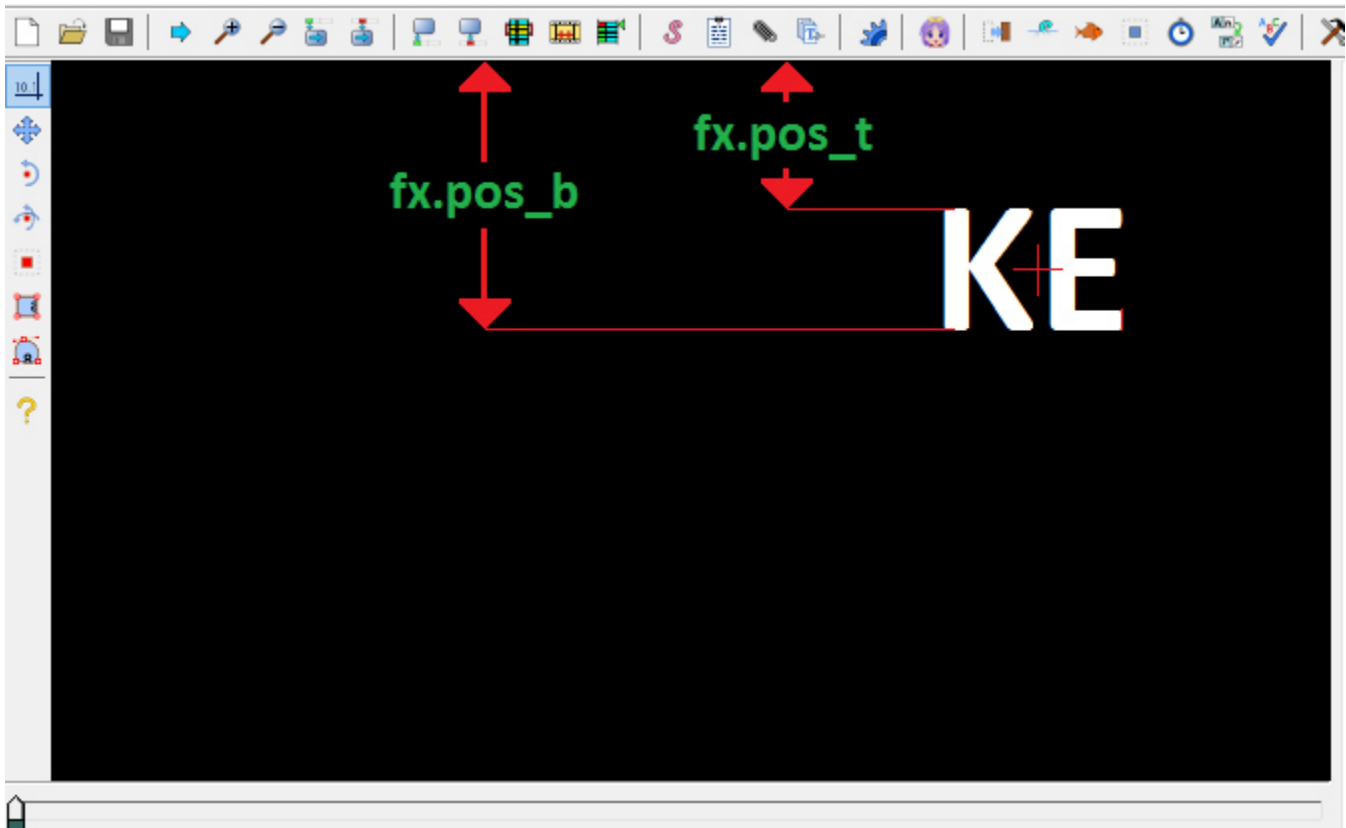
Si la plantilla es tipo “**Syl**” entonces **fx.pos_l** y **fx.pos_r** serán la izquierda y la derecha de la Sílabas, sin importar en dónde esté esa Sílabas:



Para **Template Type** “**Line**”, entonces **fx.pos_l** y **fx.pos_r** serán la izquierda y la derecha de la Línea de Texto. Lo mismo pasaría para los demás tipos de Plantillas: “**Word**”, “**Furi**”, “**Char**” y las demás.

fx.pos_t y **fx.pos_b**: (top y bottom) son la parte superior e inferior de **fx.pos_y** respectivamente.

Como en el caso de las dos variables anteriores, la parte superior e inferior depende únicamente de la posición final:



fx.align: es la alineación de la Línea fx. Debe ser un valor entero entre 1 y 9, inclusive. El valor por default de esta variable es 5. 5 es la alineación recomendada para aquellos que aún no tienen mucha experiencia a la hora de hacer Efectos Karaoke, pero todas son importantes.

fx.move_x1, **fx.move_x2**, **fx.move_x3**, **fx.move_x4**: son la cuatro posibles coordenadas con respecto al eje “x” de las posición o de la trayectoria del movimiento de la Línea fx. Son las coordenadas en “x” de los tags: \pos, \move, \moves3, \moves4 y \mover.

fx.move_y1, **fx.move_y2**, **fx.move_y3**, **fx.move_y4**: son la cuatro posibles coordenadas con respecto al eje “y” de las posición o de la trayectoria del movimiento de la Línea fx. Son las coordenadas en “y” de los tags: \pos, \move, \moves3, \moves4 y \mover.

Veamos algunos ejemplos de la anteriores ocho variables y de a poco las iremos aclarando:



fx.move_x1 = **fx.pos_x** - 45

fx.move_y1 = **fx.pos_y**

Los valores por default de **fx.move_x1** y **fx.move_y1** son, **fx.pos_x** y **fx.pos_y**, respectivamente. Del ejemplo de la anterior imagen, su resultado en la Línea fx sería el tag \pos, ya que solo hay una coordenada para ambos ejes.

Este ejemplo también retornaría un tag \pos:

Pos in 'X' =	<input type="text"/>	^ v
Pos in 'Y' =	<input type="text" value="fx.pos_y"/>	^ v

fx.move_x1 = fx.pos_x ← por Default

fx.move_y1 = fx.pos_y

Para el caso de que ambas celdas de texto estén vacías, no retornaría ningún tag de posición ni de movimiento:

Pos in 'X' =	<input type="text"/>	^ v
Pos in 'Y' =	<input type="text"/>	^ v

El valor por default de **fx.move_x2**, **fx.move_y2** y todas las superiores, es siempre la variable inmediatamente a cada una de ellas:

Pos in 'X' =	<input type="text"/>	^ v
Pos in 'Y' =	<input type="text" value="fx.pos_y, fx.pos_y + 20"/>	^ v

fx.move_x1 = fx.pos_x ← por Default

fx.move_x2 = fx.move_x1 ← por Default

fx.move_y1 = fx.pos_y

fx.move_y2 = fx.pos_y + 20

El ejemplo anterior daría como resultado un tag \move, ya que habría dos coordenadas para cada eje.

Para el próximo ejemplo, veremos el caso cuando ambos tienen tres coordenadas o al menos una celda de texto tiene tres coordenadas:

Pos in 'X' =	<input type="text" value="fx.pos_x, fx.pos_x - math.random(-30, 70)"/>	^ v
Pos in 'Y' =	<input type="text" value="fx.pos_y, fx.pos_y + 25, fx.pos_y - 25"/>	^ v

fx.move_x1 = fx.pos_x

fx.move_x2 = fx.pos_x - math.random(-30,70)

fx.move_x3 = fx.move_x2 ← por Default

fx.move_y1 = fx.pos_y

fx.move_y2 = fx.pos_y + 25

fx.move_y3 = fx.pos_y - 25

El tag que resultaría de este ejemplo sería un \moves3. Para el caso en el que al menos una de las dos celdas de texto tenga cuatro coordenadas, veamos este ejemplo:

Pos in 'X' =	50, 120, -80, syl.center	^ v
Pos in 'Y' =	200, 400, 680, syl.middle	^ v

fx.move_x1 = 50

fx.move_x2 = 120

fx.move_x3 = -80

fx.move_x4 = syl.center

fx.move_y1 = 200

fx.move_y2 = 400

fx.move_y3 = 680

fx.move_y4 = syl.middle

Para el caso del ejemplo anterior obtendríamos como tag un \moves4, lo que quiere decir que el tag que resulta de la combinación de **Pos in 'X'** y **Pos in 'Y'** depende de cuál de estas dos celdas de texto tenga más coordenadas.

Y para el caso del tag \mover hacemos lo siguiente:

Pos in 'X' =	x1, x2, Angle1, Angle2, Radius1, Radius2	^ v
Pos in 'Y' =	y1, y2	^ v

fx.movet_i, **fx.movet_f**: son los tiempo de inicio y final para los tags de movimiento: \move, \moves3, \moves4 y \mover:

Times Move =	t1, t2	^ v
--------------	--------	--------

“**t1**” representa el tiempo de inicio del movimiento y “**t2**” el tiempo final. Sus valores por default son 0 y **fx.dur**, respectivamente.

Veamos algunos ejemplos:

Pos in 'X' =	x1, x2, x3	^ v
Pos in 'Y' =	y1, y2, y3	^ v
Times Move =	t1, t2	^ v

→ **\moves3(x1, y1, x2, y2, x3, y4, t1, t2)**

El anterior es un ejemplo sencillo de cómo obtener un `\moves3` junto con los parámetros de tiempo incluidos.

Pos in 'X' =	x1
Pos in 'Y' =	y1, y2, y3
Times Move =	

→ **`\moves3(x1, y1, x1, y2, x1, y4)`**

Este ejemplo ilustra cómo `fx.move_x2` y `fx.move_x3` son `fx.move_x1` por default, y al no haber parámetros de tiempo, entonces no salen en el tag.

Pos in 'X' =	x1, x2
Pos in 'Y' =	y1, y2
Times Move =	t1

→ **`\move(x1, y1, x2, y2, t1, fx.dur)`**

Y en este otro ejemplo vemos que `fx.movement_f` equivale a `fx.dur` (duración de la Línea fx) por default.

Todo tag de posición o de movimiento que hagamos utilizando estas tres celdas de texto:

Pos in 'X' =	fx.pos_x
Pos in 'Y' =	fx.pos_y
Times Move =	

Se verá reflejado en las Líneas fx:

B <i>I</i> <u>U</u> S <i>fn</i> AB AB AB AB <input checked="" type="checkbox"/> <input type="radio"/> Tiempo <input type="radio"/> Cuadro
<code>{Kara Effector[fx] 3.2: ABC Template \an5\pos(308.34,42)}do</code>

Pero no es la única forma de usar los tags de posición y movimientos, ya que desde **Add Tag** también lo podemos hacer, de hecho, se si hace desde **Add Tags**, entonces cualquier otro tag de posición o de movimiento es anulado:

Pos in 'X' =	<input type="text" value="fx.pos_x"/>	⬆ ⬇ ⬆
Pos in 'Y' =	<input type="text" value="fx.pos_y"/>	⬆ ⬇ ⬆
Times Move =	<input type="text"/>	⬆ ⬇ ⬆
Add Tags:		Add Tags Language: <input type="text" value="Lua"/>
<input type="text" value="'\\move(10,20,30,40)'"/>		⬆ ⬇ ⬆

El **Kara Effector** detecta de forma automática que hay un tag de movimiento (\move) en **Add Tags**, entonces anula el tag \pos que se iba a generar por las configuraciones de **Pos in 'X'** y **Pos in 'Y'**:

Si por algún motivo se coloca más de un tag de posición o de movimiento en **Add Tags**, el **Kara Effector** solo tomará en cuenta el último de ellos:

Add Tags: Add Tags Language: Automation Auto-4

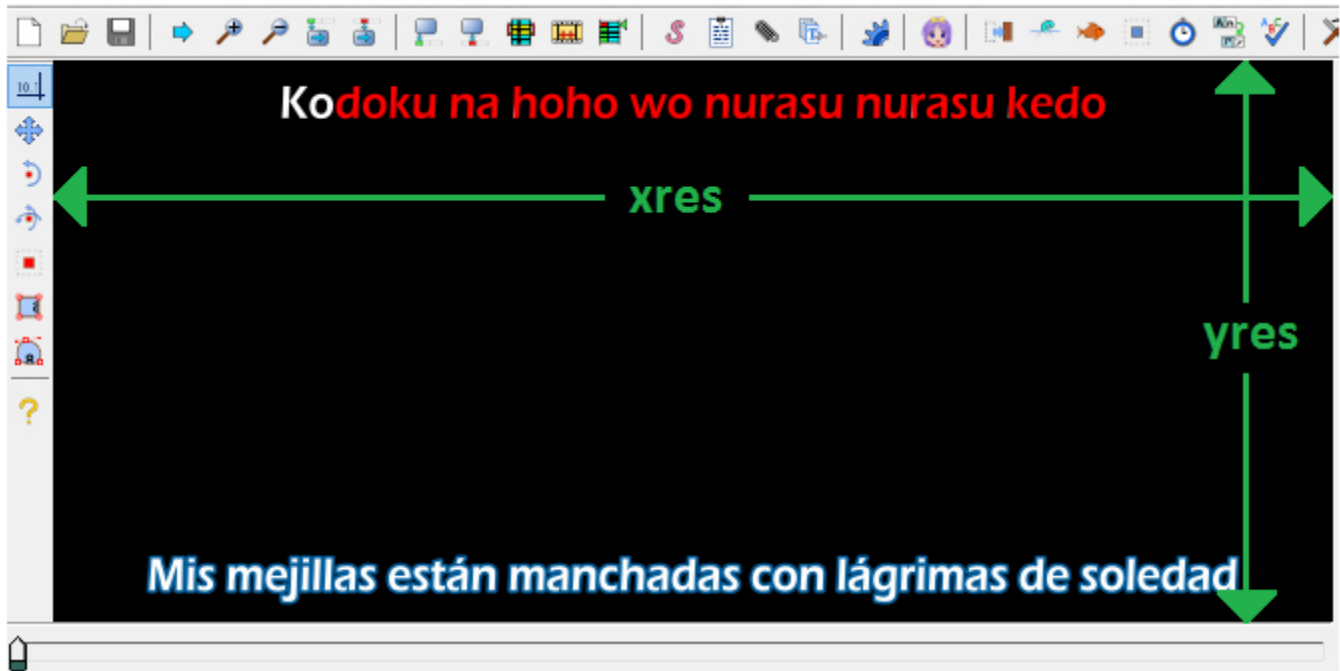
```
\move(100,200,120,300,0,1000)\pos($center,$middle)
```

En este caso solo se toma en cuenta el tag \pos ya que es el último y el tag \move no saldrá en la Línea de fx.

Con los anteriores ejemplos concluye la explicación de la Librería “fx” pero hay otra serie de variables y valores propios del **Kara Effector** que veremos a continuación:

xres y **yres**: son las dimensiones medidas en pixeles del vídeo que se esté usando al momento de aplicar un Efecto y sus valores por default son 1280 y 720 p.

xres es el ancho del vídeo y **yres**, el alto:



ratio: es la Proporción que usa el **Kara Effector** para que un Efecto funcione exactamente igual sin depender de las dimensiones del vídeo. Al aplicar un Efecto con un vídeo abierto, el valor del **ratio** es: $xres/1280$, para el caso contrario su valor por default es 1.

frame_dur: es la duración medida en milisegundos de cada uno de los cuadros (**frames**) del vídeo que se esté usando al momento de aplicar un Efecto. Su valor por default es de **41.708** ms.

line.i: es el Contador numérico de cada una de las Líneas seleccionadas a las que le aplicaremos un Efecto. Es similar a la variable **syl.i**, salvo que esta última es el contador numérico de las Sílabas que contiene una Línea.

line.n: es la cantidad total de Líneas seleccionadas para aplicar un Efecto. Es similar a la variable **syl.n**, salvo que esta última es la cantidad total de Sílabas que contiene una Línea.

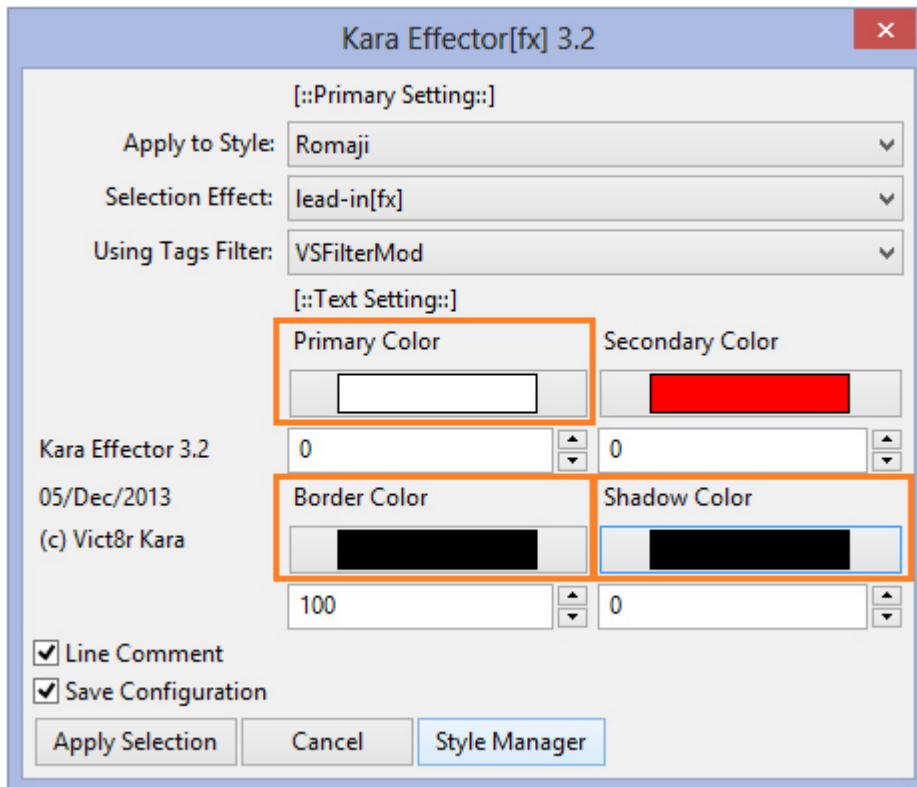
line.index: es el contador de todas las Líneas de Dialogo de un archivo .ass y también puede ser llamado como: **ii**

text.color: son los tres tag de color de la ventana de inicio del Kara Effector. En Lenguaje **Automation Auto-4** sería:

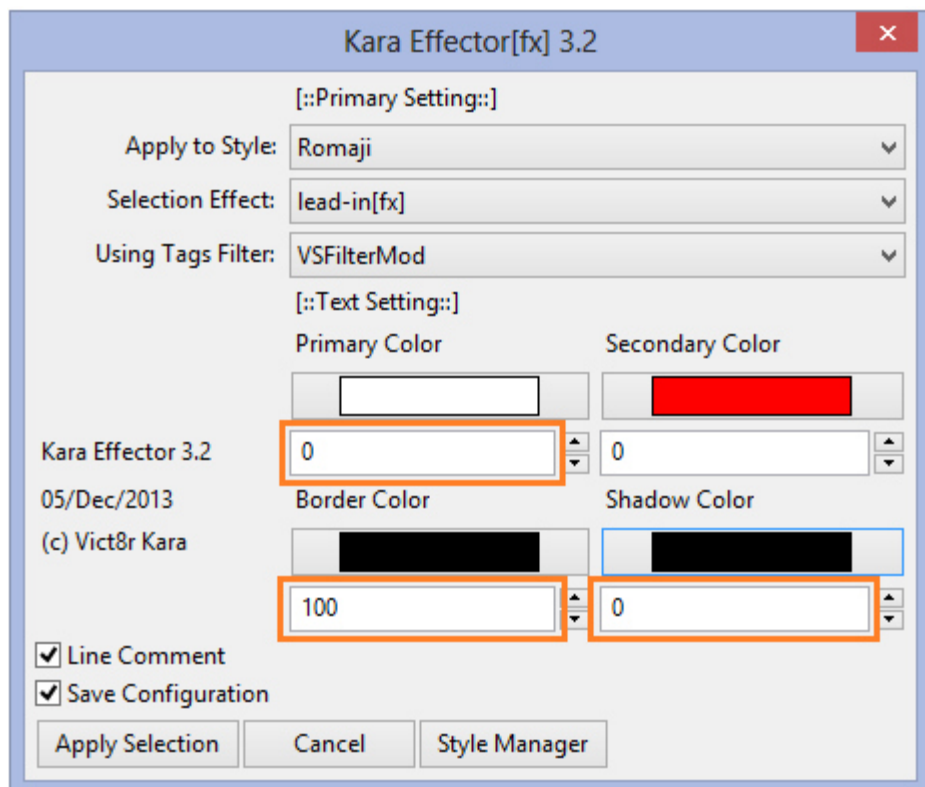
`\1c!text.color1!\3c!text.color3!\4c!text.color4!`

El formato en que se ven depende del filtro seleccionado.

Es decir que **text.color** equivale a estos tres colores:



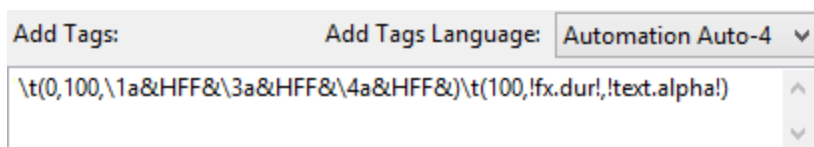
text.alpha: es similar a **text.color**, pero hace referencia a las transparencias de los tres colores anteriormente mencionados:



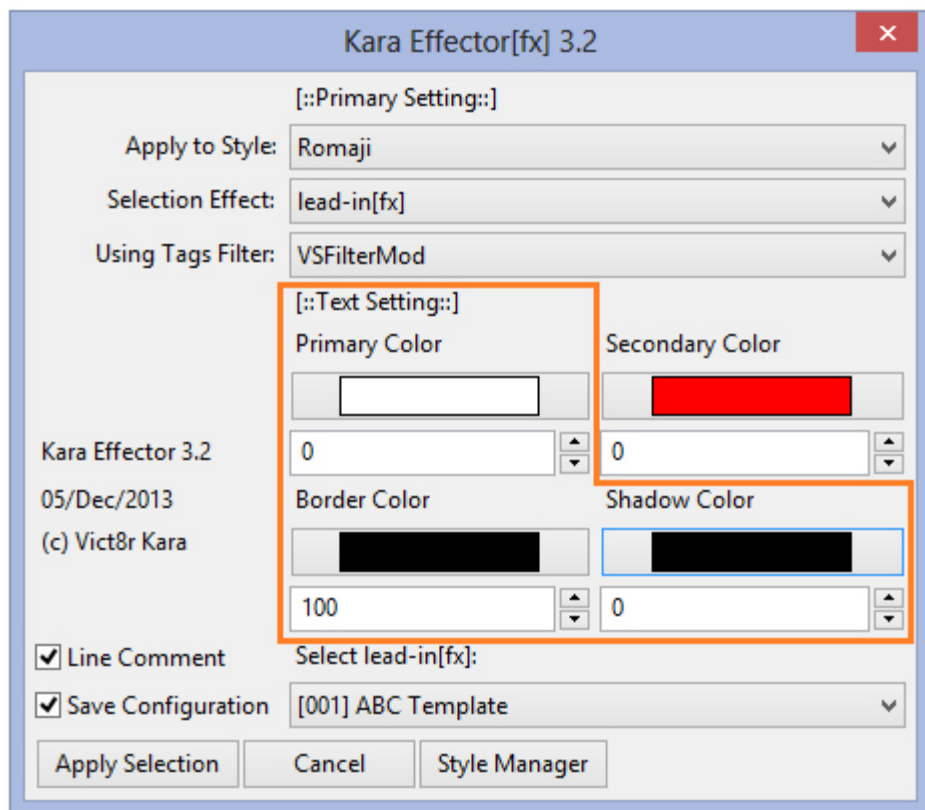
En Lenguaje **Automation Auto-4** sería:

\1a!text.alpha1!\3a!text.alpha3!\4a!text.alpha4!

Tanto **text.color** y **text.alpha** pueden servir para volver a las configuraciones de la ventana de inicio luego de alguna transformación de colores y/o de transparencias, ejemplo:

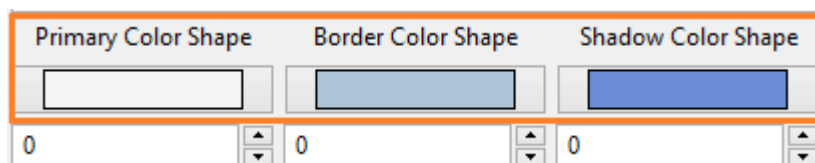


text.style: es la unión de **text.color** y **text.alpha**, o sea que hace referencia a estos seis valores de la Ventana de Inicio del **Kara Effector**:



text.alpha0: equivale a \alpha&HFF&.

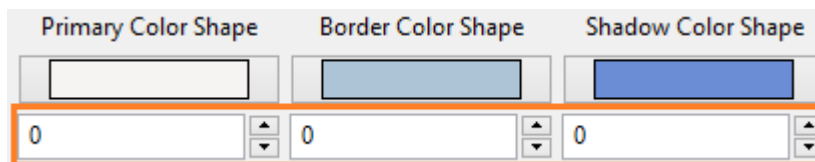
shape.color: son los tres tag de color de las Shapes:



En Lenguaje **Automation Auto-4** sería:

\1c!shape.color1!\3c!shape.color3!\4c!shape.color4!

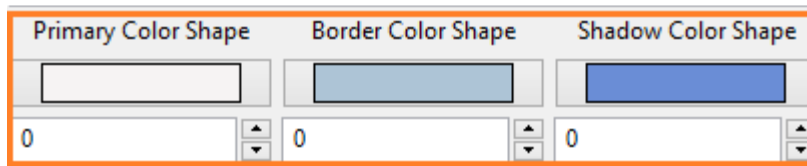
shape.alpha: son los tres tag de transparencia de las Shapes:



En Lenguaje **Automation Auto-4** sería:

\1a!shape.alpha1!\3a!shape.alpha3!\4a!shape.alpha4!

shape.style: es la unión de **shape.text** y **shape.alpha:**



shape.alpha0: equivale a **\alpha&HFF&**.

module: es la interpolación equidistante de los valores numéricos entre 0 y 1 con respecto al loop de un Efecto.

$$\mathbf{module} = (\mathbf{j} - 1) / (\mathbf{maxj} - 1)$$

module1: es la interpolación equidistante de los valores numéricos entre 0 y 1 con respecto a la cantidad de sílabas de las Líneas seleccionadas para un Efecto.

$$\mathbf{module1} = (\mathbf{syl.i} + \mathbf{module} - 1) / \mathbf{syl.n}$$

module2: es la interpolación equidistante de los valores numéricos entre 0 y 1 con respecto a la cantidad de Líneas seleccionadas para un Efecto.

$$\mathbf{module2} = (\mathbf{line.i} + \mathbf{module1} - 1) / \mathbf{line.n}$$