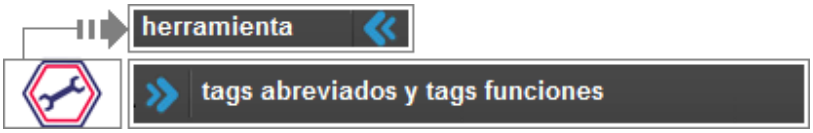
Kara Effector 3.3

En este **Tomo 39** continuaremos viendo más de los tags abreviados, modificados y añadidos en el **Kara Effector**, con el fin de saber cómo aprovecharlos y aplicarlos a nuestros proyectos. Todos estos tags nos harán el trabajo un poco más simple e incluso nos ayudarán a descubrir nuevas combinaciones y efectos, dada las múltiples posibles combinaciones que se pueden realizar entre ellos.

Recursos [KE]:

Comenzaremos viendo una tabla con el resumen de todos los tags abreviados vistos hasta este momento, con el fin de tenerlos presentes y de continuar viendo los que aún nos hacen falta:

	tags abreviados	
	tag	**equivalencia**
1	\\fscxr	\fscx
2	\\fscyr	\fscy
3	\\fscxyr	\fscx + \fscy
4	\\fscxyi	\fscx + \fscy
5	\\fscxyir	\fscx + \fscy
6	\\fscxy	\fscx + \fscy
7	\\frxy	\frx + \fry
8	\\frxz	\frx + \frz
9	\\fryz	\fry + \frz
10	\\frxyz	\frx + \fry + \frx
11	\\faxy	\fax + \fay
12	\\xybord	\xbord + \ybord
13	\\xyshad	\xshad + \yshad



Los siguientes tags que veremos incluyen a los tags de rotación y al tag \org, y son:

- \\frx**o**
- \\fry**o**
- \\frz**o** o \\fr**o**
- \\frxy**o**
- \\frxz**o**
- \\fryz**o**
- \\frxyz**o**

Y precisamente, lo que hace la letra "o" en los anteriores tags, es agregar al tag \org con los parámetros **fx.pos_x** y **fx.pos_y**, o sea, las coordenadas de la posición por default del objeto karaoke.

Ejemplo:

- \\frxz**o**Rs(360) → \org(fx.pos_x, fx.pos_y)\frx-24\frz238
- \\t(0,300,\\fry**o**90) → \org(fx.pos_x, fx.pos_y)\t(0,300,\fry90)
- \\frx**o**R(60, 120)\\t(\\fr**o**360) → \org(fx.pos_x, fx.pos_y)\frx72\t(\frz360)

Cuando las coordenadas del tag \org son las de las posición por default del objeto karaoke, cualquier rotación que se haga, se hará respecto a dicho origen, como si esa fuera la aguja de un compás con el cual trazamos un círculo.

Siempre que efectuemos una rotación, si no está el tag \org, ésta se hará respecto al tag \pos o a las dos primeras coordenadas del tag \move, pero como ya se habrán dado cuenta, el **KE** tiene algunas funciones que generan un tag \pos, como la función **shape.Rmove**, que casi siempre genera un \pos(0,0), por lo que si no especificamos el \org, las rotaciones se harán respecto al punto P = (0, 0) o a cualquier otro del tag \pos.

Con estas últimas siete abreviaciones ya completamos un total de veinte, pero no son las únicas modificaciones que podemos hacerle a los tags para generar nuevas combinaciones. Las siguientes abreviaciones incluyen a los tags "animados" ya conocidos e incluso los anteriores veinte que acabamos de aprender, y aparte de esto le sumamos el tag \t (transformación). Se dice que un tag es "animado" si se puede ingresar dentro de un tag \t y es afectado por éste:

	tags animados							
	xy-vsfilter			vsfiltermod			abreviaciones	
1	c	xbord	fax	1vc	3img	rndz	fscxr	faxy
2	1c	ybord	fay	2vc	4img	distort	fscyr	xybord
3	2c	bord	fr	3vc	fsc		fscxyr	xyshad
4	3c	xshad	frx	4vc	frs		fscxyi	frxo
5	4c	yshad	fry	1va	fsvp		fscxyir	fryo
6	alpha	shad	frz	2va	jitter		fscxy	frzo - fro
7	1a	blur	fs	3va	z		frxy	frxyo
8	2a	be	fsp	4va	rnd		frxz	frxzo
9	3a	fscx	clip	1img	rndx		fryz	fryzo
10	4a	fscy	iclip	2img	rndy		frxyz	frxyzo

Y si a cualquiera de los anteriores 72 tags animados le agregamos la letra "t", el tag se ingresará inmediatamente dentro de un tag \t:

Ejemplo:

- \\frz**t**45 = \t(\frz45)
- \\blur**t**R(2, 5) = \\t(\\blur**R**(2, 5)) → por ejemplo: \t(\blur**3**)
- \\frxzo**t**Rs(120) = \\org(fx.pos_x, fx.pos_y)\\t(\\frx**Rs**(120)\\frz**Rs**(120))
- \\3c**t**&H0000FF& = \t(\3c&H0000FF&)

Como podrán notar en los ejemplos anteriores, los tags son ingresados en un tag \t sin tiempos, es decir que la transformación se lleva a cabo a lo largo de la duración total de la línea fx (**fx.dur**):

\\t(\\tags) = \\t(0,fx.dur,\\tags)

La anterior tabla de tags animados la debemos tener siempre presente, ya que son todos aquellos tags que podemos "transformar" conforme el tiempo transcurre. Y para continuar con la misma temática de las transformaciones, veremos las siguientes abreviaciones también aplicables a los anteriores 72 tags animados:

- \\tag-
- \\tag~

Al agregar el signo menos (-) al final de un tag, el **KE** le añade una transformación por default que contiene el mismo tag, pero con el valor inverso:

Ejemplo:

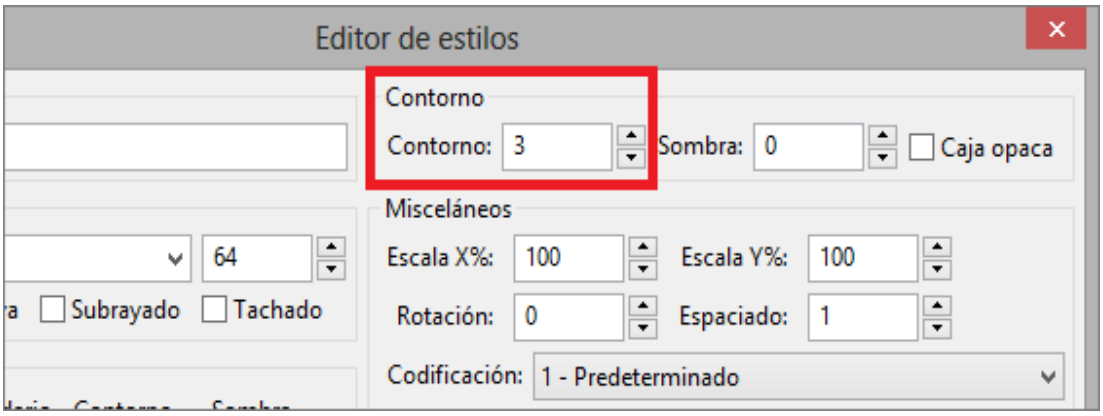
- \\frx86- → \fr86\t(\fr-86)
- \\yshad-4.2- → \yshad-4\t(\yshad4)
- \\frxzR(20, 40)- →\frx36\frz24\t(\frx-36\frz-24)
- \\fryoRs(15)- →\org(fx.pos_x, fx.pos_y)\fry-11\t(\fry11)

Y como ya se habrán imaginado, no todos los tags tienen un valor inverso, para todos aquellos con esta característica el **KE** le asigna su valor por default o uno nulo (0).

Al agregar el signo (~) al final de un tag, el **KE** le añade una transformación por default que contiene el mismo tag, pero con el valor por default o nulo:

Ejemplo:

Si para un Estilo de Línea le ponemos un contorno de 3, ese 3 será su valor por default:



Entonces para los siguientes ejemplos tenemos:

- \\bord4~ →\bord4\t(\bord3)
- \\bordRd(5, 8)~ →\bord6.7\t(\bord3)
- \\blur3~\\bord6~ →\blur3\bord6\t(blur0\bord3)

La siguiente tabla nos ayudará a recordar cuáles son los valores por default e inversos que el **KE** le asigna a cada uno de los tags animados, al agregarles un signo (~) o un signo (-) al final de los mismos:

	tags animados					
	xy-vsfilter			vsfiltermod		
	tag	valor default (\\tag~)	valor inverso (\\tag-)	tag	valor default (\\tag~)	valor inverso (\\tag-)
1	c	text.color1	text.color1	1vc	text.color1	text.color1
2	1c	text.color1	text.color1	2vc	text.color2	text.color2
3	2c	text.color2	text.color2	3vc	text.color3	text.color3
4	3c	text.color3	text.color3	4vc	text.color4	text.color4
5	4c	text.color4	text.color4	1va	text.alpha1	text.alpha1
6	alpha	&H00&	&H00&	2va	text.alpha2	text.alpha2
7	1a	text.alpha1	text.alpha1	3va	text.alpha3	text.alpha3
8	2a	text.alpha2	text.alpha2	4va	text.alpha4	text.alpha4
9	3a	text.alpha3	text.alpha3	1img	---	---
10	4a	text.alpha4	text.alpha4	2img	---	---
11	xbord	l.outline	-1 * xbord	3img	---	---
12	ybord	l.outline	-1 * ybord	4img	---	---
13	bord	l.outline	l.outline	fsc	l.scale_x	l.scale_x
14	xshad	l.shadow	-1 * xshad	frs	0	-1 * frs
15	yshad	l.shadow	-1 * yshad	fsvp	0	-1 * fsvp
16	shad	l.shadow	l.shadow	jitter	---	---
17	blur	0	0	z	0	-1 * z
18	be	0	0	rnd	0	-1 * rnd
19	fscx	l.scale_x	l.scale_x	rndx	0	-1 * rndx
20	fscy	l.scale_y	l.scale_y	rndy	0	-1 * rndy
21	fax	0	-1 * fax	rndz	0	-1 * rndz
22	fay	0	-1 * fay	distort	---	---
23	fr	l.angle	-1 * fr			
24	frx	0	-1 * frx			
25	fry	0	-1 * fry			
26	frz	l.angle	-1 * frz			
27	fs	l.fontsize	l.fontsize			
28	fsp	l.spacing	-1 * fsp			
29	clip	---	---			
30	iclip	---	---			

Como acabamos de ver, la transformación generada siempre es una por default, o sea sin tiempos dentro del tag \t. Para controlar el tiempo en que la transformación retorna el tag a su valor por default o a su valor inverso, lo único que debemos hacer es colocar el tiempo en milisegundos (ms) luego del signo (~) o del signo (-), o si queremos realizar una operación para obtener dicho tiempo, entonces colocamos la operación o función dentro de paréntesis después de los signos:

Ejemplo:

- \\fryzo90-300 →\org(fx.pos_x, fx.pos_y)\fry90\frz90\t(0,300,\fry-90\frz-90)
- \\blur3~(fx.dur/3) →\blur3\t(0,fx.dur/3,\blur0)
- \\xyshadRs(10)-200 →\xshad-7\yshad9\t(0,200,\xshad7\yshad-9)

Con esta adaptación al final de los signos (~) y (-), podemos controlar el tiempo exacto en que la transformación se realizará, lo que amplía aún más nuestras posibilidades a la hora de hacer un efecto. Estas abreviaciones tienen muchas aplicaciones en los efectos **lead-in** y **hi-light** sobre todo, pero el tipo de efecto en las que las podemos usar solo depende de cada uno de nosotros. Intenten hacer sus propias combinaciones y de a poco se irán familiarizando a ellas.

Ya con los conceptos de valores inversos y por default de los tags animados, es hora de ver unas abreviaciones un poco más complejas e igual de prácticas como las anteriores. Para las siguientes abreviaciones debemos colocar unas letras especiales entre el nombre del tag y su valor, dichas letras son:

- **mr** → \tag + \t(0,fx.dur/2,\tag-) + \t(fx.dur/2,fx.dur,\tag)
- **md** → \tag + \t(0,fx.dur/2,\tag~) + \t(fx.dur/2,fx.dur,\tag)
- **mrd** → \tag + \t(0,fx.dur/2,\tag-) + \t(fx.dur/2,fx.dur,\tag~)
- **k** → \t(0,fx.dur/4,\tag)+\t(fx.dur/4,3*fx.dur/4,\tag-) + \t(3*fx.dur/4,fx.dur,\tag~)

Ejemplo:

- \\frxmr-45 → \frx-45\t(0,fx.dur/2,\frx45)\t(fx.dur/2,fx.dur,\frx-45)
- \\blurmd5 → \blur5\t(0,fx.dur/2,\blur0)\t(fx.dur/2,fx.dur,\blur5)
- \\frmrdR(20) → \fr17\t(0,fx.dur/2,\fr-17)\t(fx.dur/2,fx.dur,\fr0)
- \\xyshadkRs(8)
- \\frxyzomdRds(120, 240)
- \\fscxyrkRc(0.75, 1.55)

En la siguiente tabla vemos las modificaciones vistas hasta ahora, que podemos hacerle a la mayoría de los tags animados (ya que hay algunas excepciones que pronto abordaremos):

	modificaciones de un tag animado			
	añadido	**valor normal del tag**	**valor aleatorio de la función R ***	**operación o función**
tag		\tag<valor>	\tagR( )	\tag(operación)
	t	\tagt<valor>	\tagtR( )	\tagt(operación)
	~	\tag<valor>~	\tagR( )~	\tag(operación)~
	-	\tag<valor>-	\tagR( )-	\tag(operación)-
	mr	\tagmr<valor>	\tagmrR( )	\tagmr(operación)
	md	\tagmd<valor>	\tagmdR( )	\tagmd(operación)
	mrd	\tagmrd<valor>	\tagmrdR( )	\tagmrd(operación)
	k	\tagk<valor>	\tagkR( )	\tagk(operación)

* La modificación de un valor aleatorio con la función **R** no aplica para todos los tags.

Los tags de colores son aquellos para los que la modificación de un valor aleatorio con la función **R** no aplica, o no de manera convencional, pero el **KE** tiene también la solución para ello, y es la siguiente convención:

- \\1c**R**( ) → \\1c(**random.color( )**)

Ejemplo:

- \\3c**R**({ 60, 120 }) = \\3c(**random.color({ 60, 120 })**)
- \\4c**R**(nil, 80) = \\4c(**random.color(nil, 80)**)
- \\1c**R**(40, nil, { 60, 90 }) = \\1c(**random.color(40, nil, { 60, 90 })**)
- \\3c**R**( ) = \\3c(**random.color( )**)

A parte de la anterior convención, tenemos las siguientes para "llamar" a los colores y alphas propios de las ventanas del **KE**:

convenciones de colores y alphas del KE			
convención	**equivalencia**	**convención**	**equivalencia**
TC1	text.color1	**SC1**	shape.color1
TC2	text.color2		
TC3	text.color3	**SC3**	shape.color3
TC4	text.color4	**SC4**	shape.color4
TA1	text.alpha1	**SA1**	shape.alpha1
TA2	text.alpha2		
TA3	text.alpha3	**SA3**	shape.alpha3
TA4	text.alpha4	**SA4**	shape.alpha4
\\txt.c	text.color	**\\shp.c**	shape.color
\\txt.a	text.alpha	**\\shp.a**	shape.alpha
\\txt.s	text.style	**\\shp.s**	shape.style

Ejemplo:

- \\3c**SC1** = \\3c(**shape.color1**) = **format**("\\3c%s", **shape.color1**)
- \\1c**TC4** = \\1c(**text.color4**) = **format**("\\1c%s", **text.color1**)
- \\3c**R**( )\\3c**t**SC3 = **format**("\\3c%s\\t(\\3c%s)", **random.color( )**, **shape.color3**)

También, para mayor practicidad, ahora el **KE** puede darle valores decimales entre 0 y 255 a los tags y a las funciones alpha, para que no nos compliquemos tanto al tener que calcular los mismos valores en base hexadecimal (16).

Ejemplo:

- \\1a**86** = **format**("\\1a%s", **ass_alpha(86)**)
- \\3a**255~** = **format**("\\3a%s\\t(\\3a%s)", **ass_alpha(255)**, **text.alpha3**)
- \\4a**92~**300 = **format**("\\4a%s\\t(0,300,\\4a%s)", **ass_alpha(92)**, **text.alpha4**)

En los anteriores ejemplos, el **KE** convierte automáticamente esos valores en base decimal (10), a base hexadecimal (16), que es el formato .ass de los valores alphas.

Y como lo mencionaba anteriormente, también podemos ingresar valores en base decimal (10) dentro de las funciones alpha, que de forma normal las usaríamos así:

Ejemplo:

- **alpha.module**(“&HAA&”, “&HFF&”)

Esta misma función, aplicada con valores decimales sería:

- **alpha.module**(170, 255)

Lo que evidentemente es más simple de hacer y de aplicar. También nos da la opción dentro de las funciones de combinar ambos casos: **alpha.module**(170, “&HFF&”)

Esta misma habilidad de poner valores decimales a los tags y funciones alpha también es aplicable a los tags del **VSfiltermod**.

Ejemplo:

- \\1va(0,0,255,255)
- \\3va255 = \\3va(255,255,255,255) = \\3va(&HFF&,&HFF&,&HFF&,&HFF&)

Hay una adaptación más hecha especialmente para aplicarla a los tags alpha del **xy-vsfilter**, y es la de poder darle el mismo valor a más de un tag:

Ejemplo:

- **13**a&A4& → \1a&HA4&\3a&HA4&
- **142**a200 → \1a200\4a200\2a200 → \1a&HC8&\4a&HC8&\2a&HC8&
- **31**a255 → \3a255\1a255 → \3a&HFF&\1a&HFF&

Es todo por ahora para el **Tomo XIL**. Intenten poner en práctica todos los ejemplos vistos y no olviden descargar la última actualización disponible del **Kara Effector 3.3** y visitarnos en la **Web Oficial**, lo mismo que en los canales de **YouTube** para descargar los nuevos Efectos o dejar algún comentario. Pueden visitarnos y dejar su comentario en nuestra página de **Facebook**:

- www.karaeffector.com
- www.facebook.com/karaeffector
- www.youtube.com/user/victor8607
- www.youtube.com/user/NatsuoKE
- www.youtube.com/user/karalaura2012