Kara Effector 3.2:
Effector Book
Vol. II [Tomo XXXV]

Kara Effector 3.2:

En este **Tomo XXXV** continuaremos viendo más de los Recursos disponibles en el **Kara Effector**, que espero que con la ayuda de esta documentación, le puedan sacar el máximo provecho a la hora de llevar a cabo sus proyectos, no solo karaokes, sino también para la edición de las líneas de subtítulos.

Recursos [KE]:

Recursos [KE]: → función

tag.module1(...)

Esta función es una entre una serie de cuatro de la librería tag enfocadas a aplicar tags respecto a los módulos del **KE**. Los módulos del **Kara Effector** son:

1. module
2. module1
3. module2
4. moduler

tag.module1 agrega una secuencia de tags a los efectos basado en el **module1** según el **Template Type**, que es una interpolación de 0 a 1 respecto al objeto karaoke. El **module1** solo es aplicable a los modos inferiores a **Line**:

Template Type	module1
Word	(word.i - 1) / (word.n - 1)
Syl	(syl.i - 1) / (syl.n - 1)
Furi	(furi.i - 1) / (furi.n - 1)
Char	(char.i - 1) / (char.n - 1)
Convert to Hiragana	(hira.i - 1) / (hira.n - 1)
Convert to Katakana	(kata.i - 1) / (kata.n - 1)
Convert to Romaji	(roma.i - 1) / (roma.n - 1)
Translation Word	(word.i - 1) / (word.n - 1)
Translation Char	(char.i - 1) / (char.n - 1)

Cada uno de los parámetros de esta función debe ser una **tabla** con tres elementos, en el siguiente orden:

1. **tag**
2. **valor inicial**
3. **valor final**

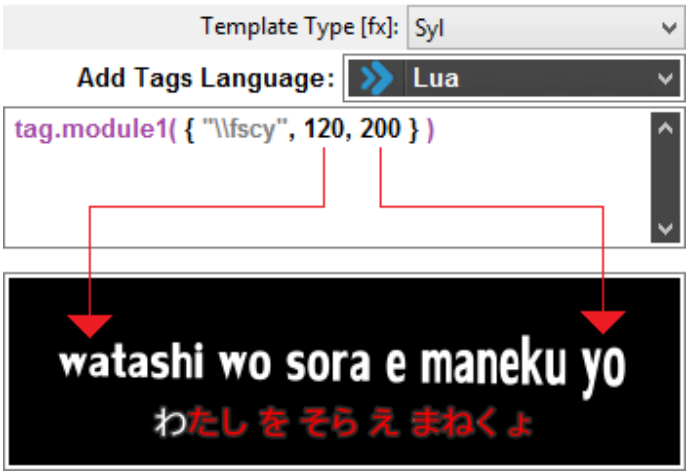
Ejemplo:

- **{ "\\1c", "&HFFFFFF&", "&H000000&" }**
- **{ "\\blur", 2, 6 }**
- **{ "\\fscx", 120, 250 }**

- **{ "\\alpha", "&HFF&", "&HAF&" }**
- **{ "\\frx", 45, 135 }**
- **{ "\\fsp", -5, 8 }**
- **{ "\\fry", -90, 90 }**

Ejemplo:

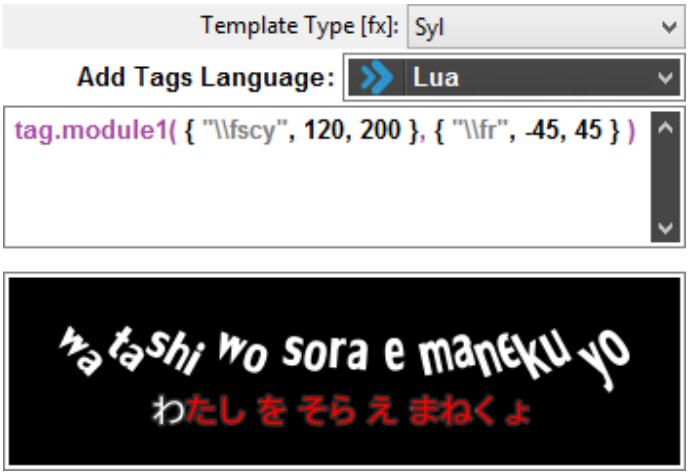
Notemos cómo el tamaño en el eje "**y**" (**\\fscy**) de las sílabas aumenta progresivamente desde el 120% de su tamaño por default hasta 200%; desde **syl.i = 1** hasta **syl.n**:



Para el siguiente ejemple veremos cómo agregar dos o más parámetros en la función:

Ejemplo:

Ahora no solo crece progresivamente respecto al eje "y", sino que también rota progresivamente respecto al eje "z" desde los -45° hasta los 45°:



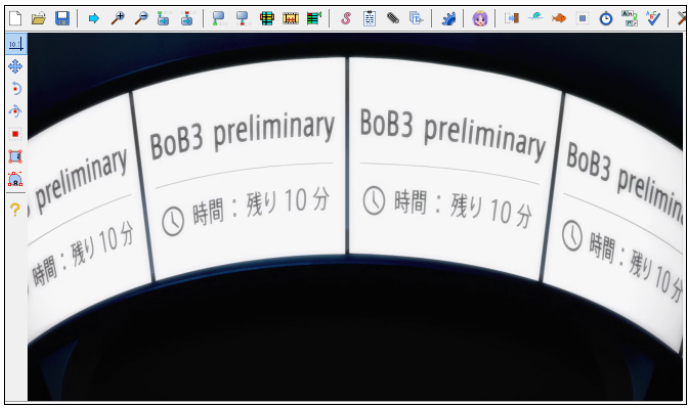
Esta función, al igual que las otras tres similares:

- **tag.module**
- **tag.module2**
- **tag.module3**

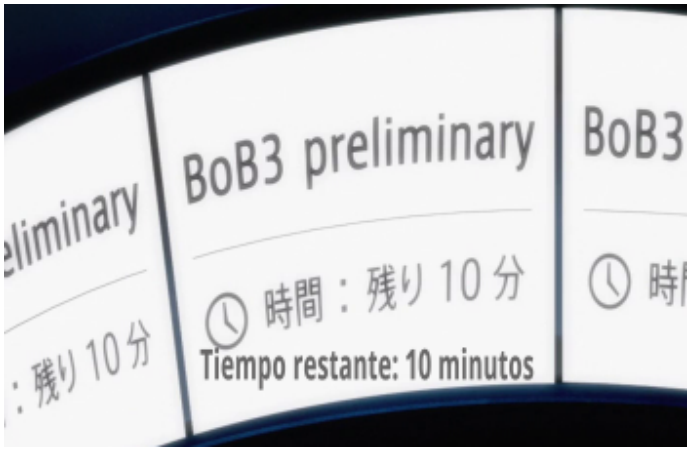
tiene la ventaja de poder ingresar la cantidad de parámetros que necesitemos, y como veremos en el próximo ejemplo, podemos complementar otras funciones con ella.

Ejemplo:

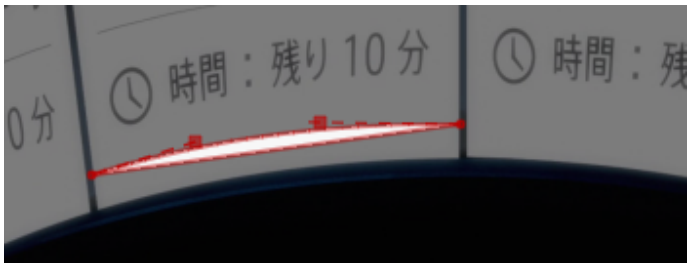
Supongamos que queremos hacer un cartel curvo como el de la siguiente imagen:



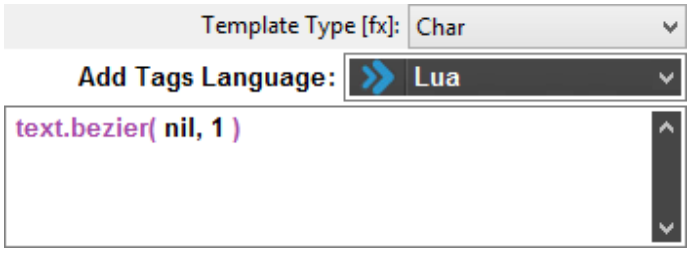
Empezamos poniendo el texto cerca, modificando el estilo del mismo hasta que quedemos satisfechos con el tamaño y color con los mismos:



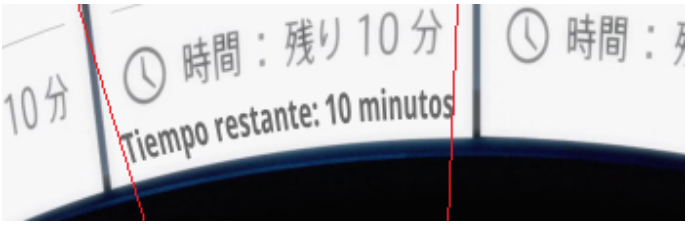
Luego trazamos el clip vectorial para que al usar la función **text.bezier**, el texto adopte su forma:



En el primer parámetro de la función ponemos "**nil**" para que se tome como shape a clip vectorial recientemente trazado:



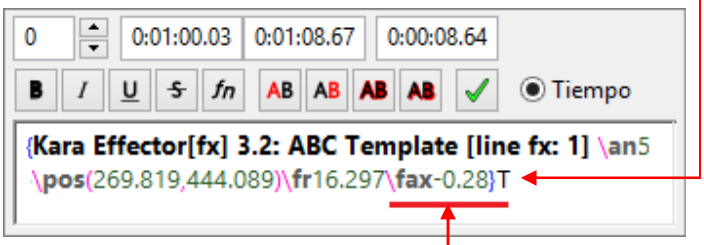
Al aplicar, notamos que efectivamente el texto adoptó la forma curva que esperábamos, pero si nos fijamos en las líneas rojas de la siguiente imagen, podemos ver que a pesar de que el texto tiene la inclinación correcta, no tiene la perspectiva misma que tiene el cartel en el vídeo:

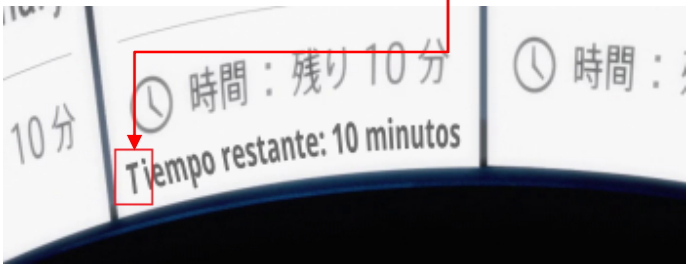


Entonces, lo que debemos hacer con las líneas generadas es dar las perspectivas correctas a la primera y última letra de dichas líneas fx. En este caso usaremos el tag **\fax**:

Cartel			Tiempo restante: 10 minutos
Cartel	lead-in	Effector [Fx]	T
Cartel	lead-in	Effector [Fx]	i
Cartel	lead-in	Effector [Fx]	e
Cartel	lead-in	Effector [Fx]	m

Ubicamos la primera línea fx generada y manualmente le vamos dando valores al tag **\fax** hasta que la letra en esta línea tenga la perspectiva correcta. Este tag hace que el objeto karaoke se incline hacia la derecha para palores negativos, y hacia la izquierda para los valores positivos. Casi nunca estos valores se salen del rango [-1, 1]:





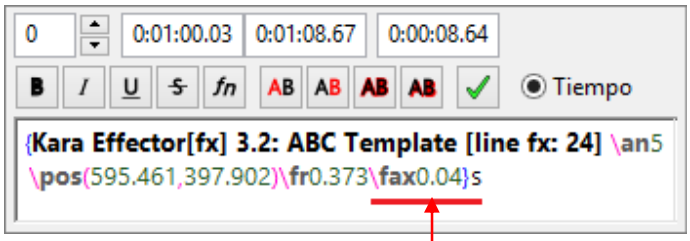
Una vez hayamos encontrado el valor correcto del \fax de la primera letra, lo usaremos más adelante como el valor inicial en la función **tag.module1**:

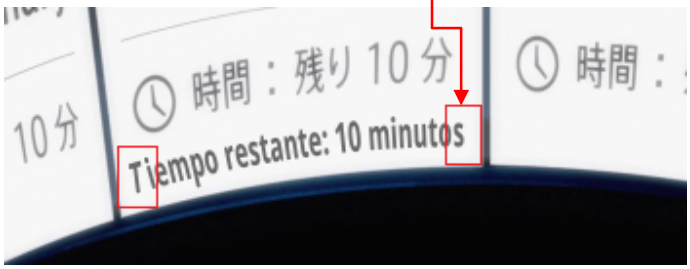
tag.module1({ "\\fax", -0.28, valor_final })

Ahora nos ponemos en la búsqueda del valor final y para ello seleccionamos a la última línea de fx generada:

Cartel	lead-in	Effector [Fx]	t
Cartel	lead-in	Effector [Fx]	o
Cartel	lead-in	Effector [Fx]	s

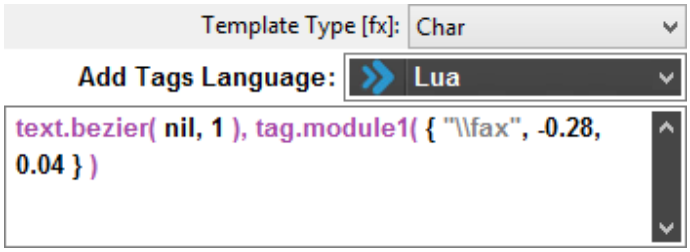
Ponemos el tag **\fax** y le ponemos valores hasta dar con el correcto (no son más de tres intentos):





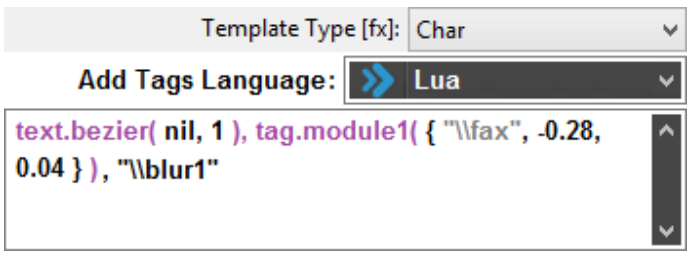
Entonces tenemos: **tag.module1({ "\\fax", -0.28, 0.04 })**

Ahora agregamos la función **tag.module1** en **Add Tags**, justo después de la función **text.bezier** que ya habíamos puesto:



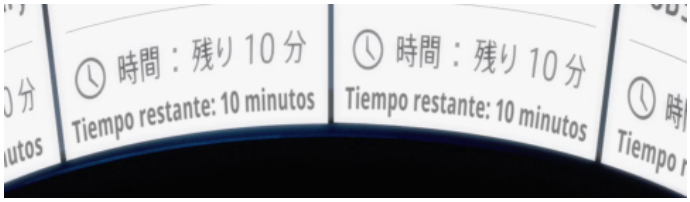
Ahora al aplicar, vemos que la función **tag.module1** sirvió como complemento de **text.bezier** para que el texto se adaptara mejor a las condiciones del cartel:

Una vez la perspectiva del texto nos deje satisfechos, ya solo es cuestión de cambiar lo que sea necesario para que el estilo sea lo más parecido al texto del cartel, como por ejemplo la opacidad. Ejemplo:



Y este sería el resultado final:

Aplicado a las cuatro secciones del cartel:



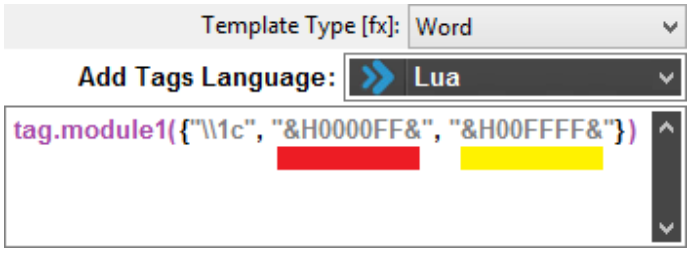
La función **tag.module1** interpola los valores de inicio y final ingresados en cada uno de sus parámetros desde el inicio hasta el final de la línea, sin importar el **loop**:

Ejemplo:

A un **Template Type: Word** le ponemos un cuadrado en la parte superior con un **loop** 3, con diferentes posiciones para para a cada uno de ellos:

En la anterior imagen se ve cómo cada palabra (**Word**) tiene encima de ella a los tres cuadrados (**loop 3**).

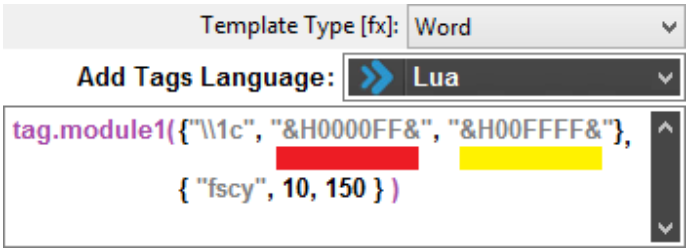
Aplicamos la función de tal manera que afectemos el color primario (**\\1c**) de los cuadrados:



Entonces la función interpola los colores desde el rojo hasta el amarillo, desde la primera palabra hasta la última y a través de cada uno de los tres **loops** de los cuadrados de 10 x 10 px, como se ve en la siguiente imagen:

Podemos agregar otro parámetro a la función para intentar dejar un poco más claro cómo se interpolan los valores de palabra en palabra y de **loop** en **loop**:

Ejemplo:

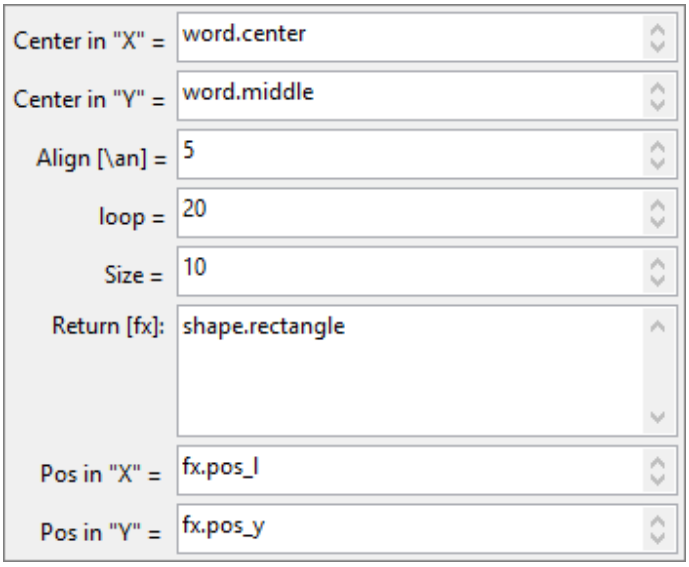


tag.module(...)

Esta función interpola los dos valores de uno o más tags de cada uno sus parámetros, en referencia a la variable **module** del **KE**.

A diferencia de la función anterior que interpolaba los dos valores de cada tag ingresado apoyada en la variable **module1**, es decir que lo hacía a lo largo de los elementos de la línea según el **Template Type**; la función **tag.module** interpola los dos valores en referencia a los **loops** de cada uno de los elementos de la línea.

Ejemplo:



Y como es de suponer, elegimos un **Template Type: Word** y en **Add Tags** ponemos lo siguiente:

Template Type [fx]: Word

Add Tags Language: Lua

```
format("\\org(%s,%s)", fx.pos_x, fx.pos_y),
tag.module( { "\\frz", 18, 360 } )
```

Al aplicar, los 20 cuadrados se dispondrán desde el centro de cada palabra, y partiendo desde la parte izquierda de las mismas, en un círculo de manera inversa al movimiento de las manecillas del reloj:

Podemos añadir otro tag, para ver cómo la función interpola sus valores respecto al **loop**, a diferencia de la función anterior que lo hacía a lo largo de la línea karaoke:

Template Type [fx]: Word

Add Tags Language: Lua

```
format("\\org(%s,%s)", fx.pos_x, fx.pos_y),
tag.module( { "\\frz", 18, 360 },
            { "\\1c", shape.color1, shape.color3} )
```

tag.module2(...)

Esta función interpola los valores de inicio y final de un tag ingresado en uno o más de sus parámetros, basado en la variable **module2** del **KE**, es decir, desde **line.i = 1**, hasta **line.n**

Ejemplo:

Template Type [fx]: Char

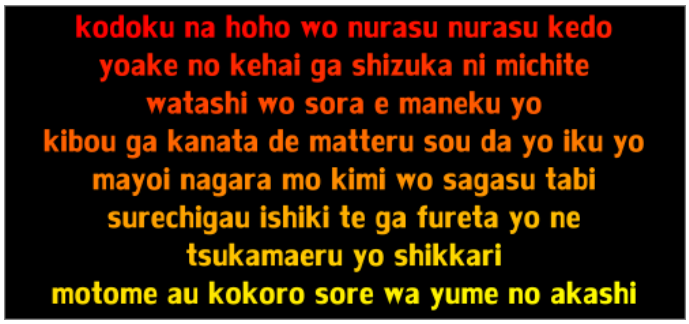
Add Tags Language: Lua

```
tag.module2({"\\1c", "&H0000FF&", "&H00FFFF&"})
```

Seleccioné todas las líneas de estilo "**Romaji**", que en este caso son 8:

Estilo	Texto
Romaji	✱m 0 0 l 0 100 100 100 100 0
Romaji	✱ko✱do✱ku ✱na ✱ho✱ho ✱wo ✱nu✱ra✱su ✱nu✱ra✱su ✱ke✱do
Romaji	✱yo✱a✱ke ✱no ✱ke✱ha✱i ✱ga ✱shi✱zu✱ka ✱ni ✱mi✱chi✱te
Romaji	✱wa✱ta✱shi ✱wo ✱so✱ra ✱e ✱ma✱ne✱ku ✱yo
Romaji	✱ki✱bo✱u ✱ga ✱ka✱na✱ta ✱de ✱ma✱t✱te✱ru ✱so✱u ✱da ✱yo ✱i✱ku ✱yo
Romaji	✱ma✱yo✱i ✱na✱ga✱ra ✱mo ✱ki✱mi ✱wo ✱sa✱ga✱su ✱ta✱bi
Romaji	✱su✱re✱chi✱ga✱u ✱i✱shi✱ki ✱te ✱ga ✱fu✱re✱ta ✱yo ✱ne
Romaji	✱tsu✱ka✱ma✱e✱ru ✱yo ✱shi✱k✱ka✱ri
Romaji	✱mo✱to✱me ✱a✱u ✱ko✱ko✱ro ✱so✱re ✱wa ✱yu✱me ✱no ✱a✱ka✱shi
Hiragana	✱こ✱ど✱く ✱な ✱ほ✱ほ ✱を ✱ぬ✱ら✱す ✱ぬ✱ら✱す ✱け✱ど
Hiragana	✱よ✱あ✱け ✱の ✱け✱は✱い ✱が ✱し✱ず✱か ✱に ✱み✱ち✱て

Al aplicar el efecto, notamos que cada letra de cada una de las líneas va cambiando progresivamente, desde el rojo hasta el amarillo, desde **line.i = 1** hasta **line.n**:



tag.moduler(...)

Cuarta y última de las funciones de la **librería tag** enfocadas en aplicar gradualmente los valores de inicio y final de uno o más tags ingresados como alguno de sus parámetros.

Esta función interpola dichos valores respecto a la variable **moduler**, que es una variable con los valores equidistantes entre 0 y 1 según la cantidad de repeticiones hechas con la función **replay** del **KE**.

Ejemplo:

Seleccionamos el efecto **[001] ABC Template Hilight Syl** de la librería de efectos **hi-light [fx]** y comenzamos a hacer las siguientes modificaciones:

```
loop = 20
Size = 20
Return [fx]: shape.circle
```

Creamos dos nuevas variables:

```
Variables:
mi_angle = R(360);
mi_radius = R(60,80)
```

Modificamos las posiciones:

```
Pos in "X" = fx.pos_x + math.polar(var.syl.mi_angle,
var.syl.mi_radius*(1 - module), "x")
Pos in "Y" = fx.pos_y + math.polar(var.syl.mi_angle,
var.syl.mi_radius*(1 - module), "y")
```

Modificamos los tiempos del fx con el fin de crear un efecto tipo **pre-hilight**:

```
Line Start Time = l.start_time + syl.start_time - 400*(1 - module)
Line End Time = fx.start_time + 380
```

Y por último, añadimos unos cuantos tags para los estilos de la **shape** y llamamos a la función **tag.moduler**, así:

```
Add Tags:  Add Tags Language: Lua
"\\bord0\\blur2\\t(\\fscx5\\fscy5)\\fad(50,120)",
tag.moduler( {"\\1c", "&H00FFFF&", "&H0000FF&" } )
```

Entonces, a cada sílaba le llega una especie de rayo justo antes de ser karaokeada, y como no repetimos el fx ni una sola vez, solo se verá de color amarillo para todas las sílabas:

Ahora repetimos el fx con la función **replay**, de manera que a cada sílaba ahora le lleguen tres rayos en vez de uno:

```
Add Tags:  Add Tags Language: Lua
replay( 3 ), "\\bord0\\blur2\\t(\\fscx5\\fscy5)\\fad(50,120)",
tag.moduler( {"\\1c", "&H00FFFF&", "&H0000FF&" } )
```

Hecho de esta manera, ya podemos ver la interpolación de los colores entre el amarillo y el rojo, y los tres rayos que le llegan a cada una de las sílabas:

Se nota claramente la interpolación de los colores hecha por la función, un color diferente para cada rayo.

Es todo por ahora para el **Tomo XXXV**. Intenten poner en práctica todos los ejemplos vistos y no olviden descargar la última actualización disponible del **Kara Effector 3.2** y visitarnos en el **Blog 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.blogspot.com
- www.facebook.com/karaeffector
- www.youtube.com/user/victor8607
- www.youtube.com/user/NatsuoKE
- www.youtube.com/user/karalaura2012