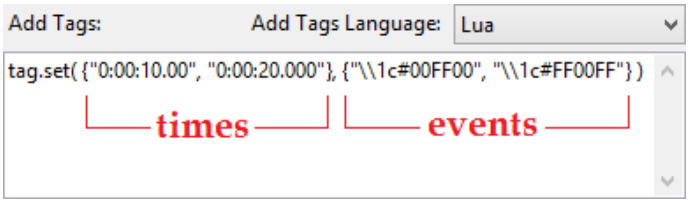
Kara Effector 3.2:

Este **Tomo XI** es la continuación de las funciones de la librería "**tag**". No hay mucho más que se pueda decir acerca de esta librería, sino que como todas las vistas en los Tomos del **Kara Effector**, es muy importante conocerla y saber qué tipo de ayuda nos puede ofrecer cada una de las funciones que contiene.

Librería "tag" [KE]

tag.set(times, events): esta función asume que todas las líneas habilitadas para aplicarle un Efecto son una sola y luego aplica una transformación de un 1 ms de duración de cada uno de los elementos de la **tabla** "**events**", según los tiempos de la **tabla** "**times**".

Las tablas "**times**" y "**events**" pueden ser ingresadas directamente en la función o declararlas a modo de variables en la celda de texto "**Variables**". Veamos un ejemplo de los dos casos:



No está de más decir que ambas tablas deben tener la misma cantidad de elementos. En este ejemplo las tablas fueron ingresadas directamente.

Los tiempos de la tabla "**times**" son copiados de la parte inferior del vídeo, en el Aegisub y posteriormente pegados entre comillas, no importando si son sencillas o dobles.

Los tags en la **tabla** "**events**" también se deben escribir entre comillas, así como se puede apreciar en la anterior imagen. Como la función **tag.set** retorna transformaciones de los tags que están en la **tabla** "**events**", entonces éstos deben ser tags que puedan ser "**animados**" por el tag "\t", por ejemplo: \bord, \blur, \3c, \alpha, \1a, \jitter, \fsvp, \clip, \iclip, \fscx, \fsc, \fscy, \fsp, \shad, entre otros.

Ahora veamos algunos ejemplos de tags que no pueden ser "**animados**" por el tag "\t": \pos, \move, \org, \moves3, \moves4, \movevc, \mover, \fad, \t, entre otros.

Lo que hará la función en este ejemplo será que a los 10 s ("**0:00:10.000**") contados desde el inicio del vídeo (cero absoluto), convertirá al color primerio ("**\\1c**") de su color por default a Verde ("**#00FF00**" está en formato **HTML**, pero no es obligación que esté en este formato, también se podría hacer en formato **.ass**, o sea "**&H00FF00&**"):

#	L	Start	End	Style	Text
1	1	0:00:02.43	0:00:08.16	Romaji	✱Ko✱do✱ku ✱na ✱ho✱ho ✱wo ✱
2	1	0:00:08.33	0:00:13.19	Romaji	✱Yo✱a✱ke ✱no ✱ke✱ha✱i ✱ga ✱
3	1	0:00:13.54	0:00:18.39	Romaji	✱Wa✱ta✱shi ✱wo ✱so✱ra ✱e ✱
4	1	0:00:18.67	0:00:27.22	Romaji	✱Ki✱bo✱u ✱ga ✱ka✱na✱ta ✱de
5	1	0:00:28.52	0:00:34.30	Romaji	✱Ma✱yo✱i ✱na✱ga✱ra ✱mo ✱ki

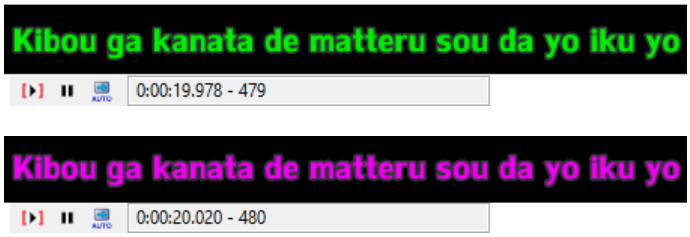
En la imagen, como la línea 1 va desde los 2.43 s hasta los 8.16 s, entonces no se verá afectada por la función y el color primario de ésta será el que ya tiene por default, que en este ejemplo es Blanco:

Pero la línea 2, como va desde los 8.33 s hasta los 13.19 s, sí se verá afectada, a partir de los 10 s. O sea que el color primario de la línea 2 será Blanco desde que inicia (8.33 s) hasta los 10 s:

Y justo cuando el vídeo llegue a los 10 s, el color primario cambiará de forma automática a Verde:

Como la línea 3 inicia en 13.54 s y finaliza en 18.39 s, su color primario será siempre el Verde:

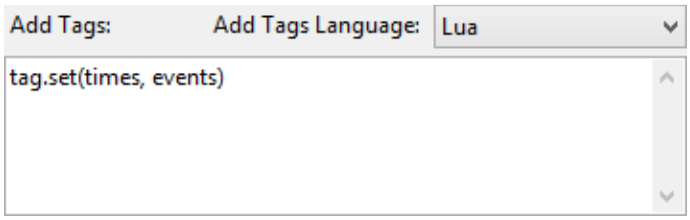
En el caso de la línea 4 pasará algo similar a la línea 2, ya que su tiempo de inicio está en 18.67 s, entonces su color primario iniciará siendo Verde, pero al llegar a los 20 s ("**0:00:20.000**") cambiará a Lila ("**#FF00FF**" en **HTML**), ya que su tiempo final está en 27.22 s:



Y desde la línea 4 en adelante, el color primario será siempre Lila, ya que eso es lo que dicta la función.

Si queremos declarar las variables de las dos tablas de la función ("**times**" y "**events**"), haríamos algo como:

Y dentro de la función, en **Add Tags**:



Y sin importar cuál de los dos métodos usemos, los resultados serán los mismos. Es decir, que los tags que coloquemos en la **tabla** "**events**" sucederán de forma inmediata, según los tiempos que hayamos registrado en la **tabla** "**times**". Esta función es ideal para hacer que nuestros efectos hagan cosas puntuales a medida que en el vídeo al que le hacemos un karaoke, sucedan cambios que nos llamen la atención, como un cambio de color o de escena, cambios de estilos por un personaje y demás.

El **lead-in** de la línea que se ve en la imagen, incluye una **shape** de Plumas para imitar el estilo de las plumas que salen en el vídeo:

La función **tag.set** nos serviría para hacer que las plumas de nuestro efecto desaparezcan exactamente en el mismo momento que lo hacen las del vídeo, haciendo algo como:

tag.set({acá el tiempo exacto}, {"\\alpha&HFF&"})

Entonces en ese preciso momento las plumas quedaran invisibles gracias al tag \alpha. Como se podrán imaginar, las posibilidades son practicamente infinitas y a gusto de cada uno de nosotros.

Un tag en la **tabla** "**events**" no necesariamente debe ser uno solo, pueden ser varios:

events = {"\\fscxy125\\3c&H000000&\\blur4", "\\shad2"}

Si queremos que una o más de las transformaciones no suceda de forma inmediata (ya que por default cada una de la transformaciones en esta función tarda solo 1 ms), el **Kara Effector** nos da dos opciones para ello:

- **Opción 1**: duración de la transformación en ms. Ej:

times = { "0:00:10.000", **{"0:00:20.000", 460}** }

Esto hará que la segunda transformación ya no dure 1 ms, sino que ahora tardará 460 ms.

- **Opción 2**: tiempo final de la transformación. Ej:

times = { "0:00:10.000", **{"0:00:20.000", "0:00:21.810"}** }

Esto hará que la segunda transformación ya no dure 1 ms, sino que ahora tardará 1810 ms, ya que:

0:00:21.810 - 0:00:20.000 = 1810 ms

De lo anterior es fácil deducir que como la duración por default de cada una de las transformaciones de la función **tag.set** es 1 ms, se vería de la siguiente forma, ejemplo:

times = { **{"0:00:10.000", 1}**, **{"0:00:20.000", 1}** }

Con este método haremos que una transformación tarde exactamente el tiempo que necesitemos y no siempre 1 ms como lo hace por default. Recuerden que la cantidad de tiempos en la **tabla** "**times**" es ilimitada, y que por cada uno de esos tiempos debe haber un tag o serie de tags que le correspondan, para que las transformaciones sean posibles y la función no nos arroje un error.

---

tag.glitter(time, add_i, add_f): esta función hace tranformaciones al azar en un tiempo determinado "**time**" que consisten en cambios bruscos de las dimensiones del objeto karaoke. Los parámetros "**add_i**" y "**add_f**" son opcionales.

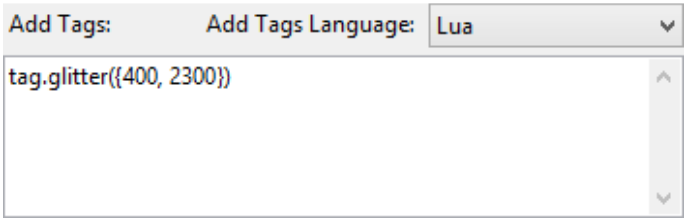
El parámetro "**time**" puede ser un tiempo en ms, lo que hará que las transformaciones al azar se hagan en el lapso de tiempo entre 0 y dicho valor. Ejemplo:

tag.glitter(1200)

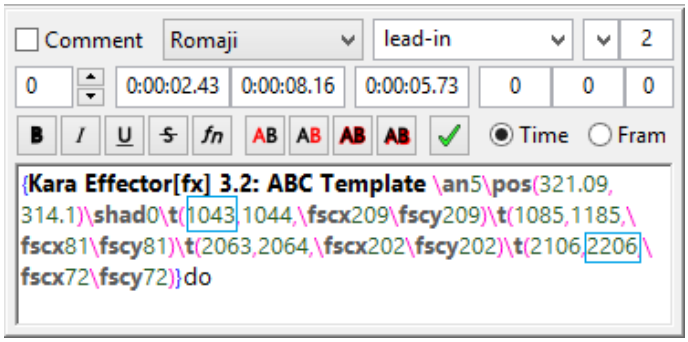
La otra forma que puede tener el parámetro "**time**" es en forma de **tabla**, en donde el primer elemento de la misma sea el tiempo en ms del inicio y el segundo elemento sea el tiempo final. Ejemplo:

tag.glitter({400, 2300})

O sea que las transformaciones al azar sucederán entre los 400 ms y los 2300 ms. Pongamos a prueba este mismo ejemplo:



Y verificamos si las transformaciones se realizaron en el rango de 400 a 2300 ms:

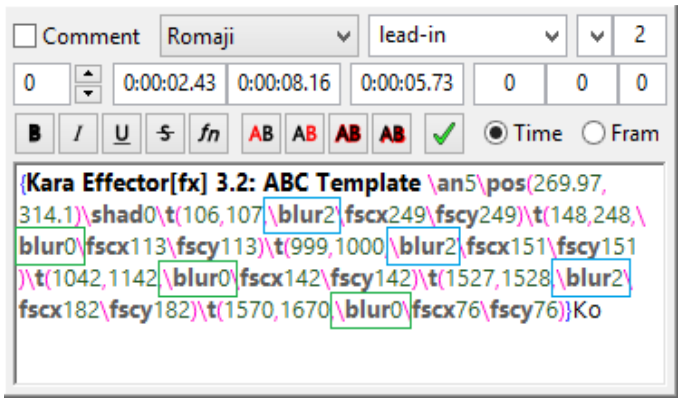


El valor por default de "**time**" es **fx.dur**, o sea, la duración total de cada una de las líneas fx:

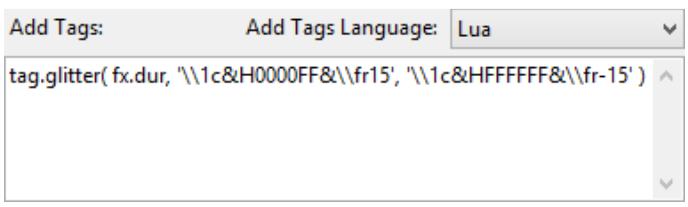
tag.glitter( ) = tag.glitter(fx.dur)

Las transformaciones que retorna la función **tag.glitter** vienen en cantidades pares. El parámetro "**add_i**" es un tag o una serie de ellos que a la postre se agregarán a las transformaciones impares retornadas. De forma similar, el parámetro "**add_f**" se agregará a las transformaciones pares retornadas. Ejemplo:

tag.glitter({0,1700}, "\\blur2", "\\blur0")



El truco radica en que los tags de "**add_i**" sean opuestos a los de "**add_f**" para que una transformación "contradiga" a la otra. Si por ejemplo en "**add_i**" colocamos un tag \1c, entonces debemos usar en "**add_f**" otro tag \1c, pero con un color distinto, ejemplo:



Del anterior ejemplo, notamos cómo los dos colores son distintos (Rojo y Blanco), y cómo los ángulos son opuestos entre sí (\fr15 y \fr-15).

La función **tag.glitter** es muy útil para asemejar efectos de brillo en **shapes** pequeñas, ya que asemeja un efecto de "titileo" como el de las estrellas en el firmamento. Así que la pueden poner a prueba con la **shape** de una estrella con un tamaño en pixeles entre 5 y 10.

---

tag.oscill(time, delay, …): esta función genera transformaciondes de duración "**delay**" en un lapso de tiempo "**time**".

El parámetro "**time**" puede ser un valor en ms, lo que hará que las transformaciones se ejecuten desde cero hasta dicho valor, ejemplo:

- **time** = 1000
- **delay** = 200

Estos dos valores hacen que la función **tag.oscill** retorne cinco transformaciones:

1. de **0** a 200 ← Duración = 200
2. de 200 a 400 ← Duración = 200
3. de 400 a 600 ← Duración = 200
4. de 600 a 800 ← Duración = 200
5. de 800 a **1000** ← Duración = 200

El parámetro "**time**" también puede ser una **tabla**, en donde el primer elemento es el tiempo inicial y el segundo elemento es el tiempo final, ejemplo:

- **time** = {400, 1120}
- **delay** = 180

Estos dos valores hacen que la función retorne cuatro transformaciones:

1. de **400** a 580 ← Duración = 180
2. de 580 a 760 ← Duración = 180
3. de 760 a 940 ← Duración = 180
4. de 940 a **1120** ← Duración = 180

Entonces, el parámetro "**time**" puede ser tanto un **valor** numérico, como una **tabla** con dos valores que limiten el rango de tiempo en el cual se ejecutará la función.

El parámetro "**delay**" tiene las dos mismas cualidades del parámetro "**time**", de poder ser tanto un **valor** numérico como una **tabla** de valores.

Para un "**delay**" con valor numérico, nos sirve un ejemplo similar a los dos anteriores:

- **time** = 300
- **delay** = 100

Lo que generará tres transformaciones:

1. de 0 a 100 ← Duración = 100
2. de 100 a 200 ← Duración = 100
3. de 200 a 300 ← Duración = 100

cuando "**delay**" es una **tabla**, ampliamos las posibilidades de manipular más a la función **tag.oscill** y a los resultados que nos ofrece. Recordemos que una transformación se basa en el tag \t, y uno de los modos de uso del tag \t es:

\t(t1, t2, **accel**, \…

En donde el parámetro "**accel**" es la aceleración de la transformación, cuyo valor por default es 1. Si queremos modificar la aceleración en las transformaciones de la función **tag.oscill**, lo que debemos hacer es especificarla en el segundo elemento de la tabla "**delay**", ejemplo:

- **time** = 450
- **delay** = {150, 0.8}

Lo que generará tres transformaciones, pero con la aceleración de 0.8:

1. \t(0, 150, 0.8, \... ←Duración = 150
2. \t(150, 300, 0.8, \... ←Duración = 150
3. \t(300, 450, 0.8, \... ←Duración = 150

Entonces decimos que el primer elemento de la **tabla** "**delay**" será la duración de cada transformación, y el segundo elemento equivale a la aceleración "**accel**" de las transformaciones.

El "**delay**" como **tabla** nos da otra tercera posibilidad, que es el "dilatar" la duración de cada transformación que se genere. El valor de dicha dilatación es el tercer elemento de la **tabla** "**delay**", y este valor puede ser positivo, lo que hará que cada transformación dure más tiempo que la inmediatamente anterior; o si el valor es negativo, hará que cada transformación dure cada vez menos tiempo que la transformación inmediatamente anterior. Ejemplo:

- **time** = {500, 1110}
- **delay** = {100, 1.2, 10}

o sea que:

- **Duración** = 100
- **Aceleración** = 1.2
- **Dilatación** = 10

Este ejemplo generará cinco transformaciones con las siguientes características:

1. \t(500, 600, 1.2, \... ←Duración = 100
2. \t(600, 710, 1.2, \... ←Duración = 110
3. \t(710, 830, 1.2, \... ←Duración = 120
4. \t(830, 960, 1.2, \... ←Duración = 130
5. \t(960, 1100, 1.2, \... ←Duración = 140

Es notorio cómo cada transformación dura 10 ms más que la transformación inmediatamente anterior, ya que ese fue el valor asignado como "dilatación".

Hasta este punto, en la función **tag.oscill**, el "**delay**" no solo decide la duración de cada transformación, sino también su frecuencia. Si por ejemplo tenemos un "**delay**" de 200 ms, lo que significaría que la duración de las tranformaciones generadas será de 200 ms, y que cada transformación empezará 200 ms después de haber iniciado la transformación inmediatamente anterior.

Para modificar la frecuencia de las transformaciones generadas, el "**delay**" en modo de **tabla** también nos da esa posibilidad. Ejemplo:

- **time** = 500
- **delay** = {{100, 25}, 1}

Notamos que el primer elemento de la **tabla** "**delay**" es otra **tabla**, en donde el primer elemento de esta otra **tabla** (100) marca la **frecuencia**, y el segundo (25), marca la **duración** de las transformaciones:

1. \t(0, 25, 1, \... ←Duración = 25
2. \t(100, 125, 1, \... ←Duración = 25
3. \t(200, 225, 1, \... ←Duración = 25
4. \t(300, 325, 1, \... ←Duración = 25

5. \t(400, 425, 1, \... ←Duración = 25

Y en las cinco transformaciones vemos que cada una de ellas empieza a 100 ms después de haber iniciado la transformación inmediatamente anterior (dado que 100 ms es la frecuencia asignada) y, la duración total de cada transformación es de 25 ms.

En resumen, el parámetro "**delay**" en la función **tag.oscill** tiene los siguientes cuatro modos:

1. **delay** = dur
2. **delay** = { dur, accel }
3. **delay** = { dur, accel, dilatation }
4. **delay** = { { frequency, dur }, accel, dilatation }

Los valores por default de las anteriores variables son de la siguiente forma:

Modo	frequency	accel	dilatation
1	**dur**	**1**	**0**
2	**dur**		**0**
3	**dur**		**0**
4		**1**	**0**

Lo que en resumen sería:

delay = dur = { { dur, dur }, 1, 0 }

El tercer parámetro de la función **tag.oscill** pone (**…**), ya sabemos que los tres puntos seguidos hacen referencia a una cantidad indeterminada de parámetros, que para este caso son los tags que necesitamos que se **alternen** en las transformaciones.

Como el objetivo de la función **tag.oscill** es alternar tags, se debería usar por lo menos con dos de ellos. De ahí en adelante podemos usar la cantidad de tags que deseemos. Ejemplo:

tag.oscill(1000, 200, "\\blur1", "\\blur3", "\\blur5")

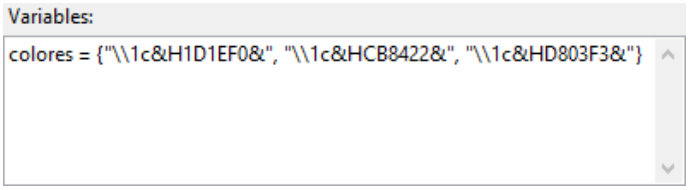
Obtendríamos las siguientes transformaciones:

1. \t(0, 200, \blur1) ←Duración = 200
2. \t(200, 400, \blur3) ←Duración = 200
3. \t(400, 600, \blur5) ←Duración = 200
4. \t(600, 800, \blur1) ←Duración = 200
5. \t(800, 1000, \blur3) ←Duración = 200

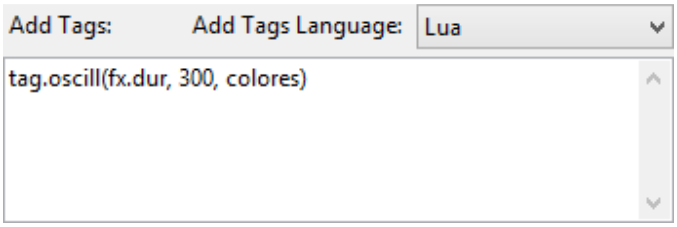
Los tres tags ingresados en la función se alternaron en las transformaciones, a manera de ciclo. Este procedimiento será el mismo sin importar la cantidad de tags, lo que hace que esta función tenga muchas utilidades, como alternar cambios de tamaños, de colores, de blur, de ángulos o lo que nos podamos imaginar.

Los tags a alternar se pueden ingresar en la función como lo hecho en el anterior ejemplo, pero también hay otra forma de hacerlo, y es en forma de **tabla**. Ejemplo:

Definimos nuestra **tabla** con los tags a alternar, que para este ejemplo, es una tabla de tres colores primarios:



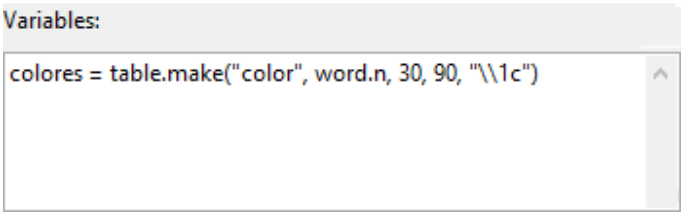
Y en **Add Tags** ponemos:



Lo que haría que esos tres colores primarios se alternen cada 300 ms durante la duración total de cada línea de fx.

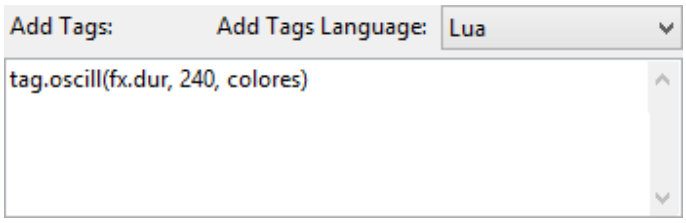
La última opción que nos da la función **tag.oscill** es poder decidir cuál será el primer elemento de la **tabla** de tags, con el que empezará la primera transformación entre todas las retornadas. Ejemplo:

Primero declaramos la siguiente **tabla**, con un Template Type: **Translation Word**



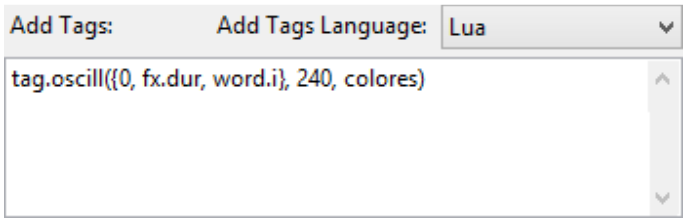
Recordemos que la función **table.make** crea una **tabla**, en este caso de colores, de word.n de tamaño, con colores equidistantes entre los ángulos 30° y 90° con un tag \1c.

Ahora usamos la **tabla** creada en la función **tag.oscill** de la siguiente manera:



Así la primera transformación de todas las generadas en cada palabra (**word**) empezarán con el mismo color:

Y la opción que ya había mencionado, de decidir con cuál tag empezarán las transformaciones es:



- **time = { 0, fx.dur, word.i }**

  El tercer elemento de la **tabla** "**time**" es el que nos ayuda a decidir el primer tag con el que empezará la primera transformación. Para este ejemplo es **word.i**

Hecho esto, el tag de la primera transformación para la primera palabra, será el pimer color de la **tabla** "**colores**" ya que **word.i** = 1; para la segunda palabra será el color 2 de la **tabla**, porque **word.i** = 2; y así sucesivamente con las demás palabras.

---

tag.ipol(valor_i, valor_f, index_ipol): esta función interpola los parámetros "**valor_i**" y "**valor_f**" teniendo como referencia al parámetro interpolador "**index_ipol**". Retorna el valor que deseemos entre todos los que existan entre "**valor_i**" y "**valor_f**", inclusive ellos mismos.

Los parámetros "**valor_i**" y "**valor_f**" pueden ser colores, transparencias (alpha) o números reales y ambos deben ser del mismo tipo. Y el parámetro "**index_ipol**" es un número real entre 0 y 1, ya que si es menor que 0 la función lo tomará como 0, si es mayor que 1, la función lo tomará como 1, y su valor por default es 0.5 para que retorne el valor promedio entre "**valor_i**" y "**valor_f**".

Ejemplos:

- **tag.ipol**("&HFFFFFF&", "&H000000&", 0.7)
- **tag.ipol**("&HFF&", "&HAA&", **j/maxj**)
- **tag.ipol**(200, 100, **char.i/char.n**)
- **tag.ipol**(**text.color3**, "&H00FFFF&")

Y con la función **tag.ipol** se da por terminada la librería "**tag**" y seguimos avanzando en el estudio de recursos del **Kara Effector** y profundizando cada vez más en el mundo de los Karaokes.

Intenten poner en práctica todos los ejemplos vistos en este **Tomo** 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, exponer alguna duda o hacer alguna sugerencia. También pueden visitarnos y dejar su comentario en nuestra página de **Facebook**: www.facebook.com/karaeffector