Las **Variables de Punto** son otro paquete de **Funciones Predeterminadas** con las que cuenta **Aegisub 2.1.8** para la **Automatización de los Efectos**. Las Variables de Punto se ejecutan al escribirlas en medio de dos signos de admiración (!____!) Y es por ello que también se les conoce como "**Variables Funciones**". La mayoría de las **Variables de Punto** tienen su Equivalencia con las **Variables Dóla**r, ejemplo:

syl.i == $si
line.duraction == $ldur
syl.start_time == $sdur

Es por ello que la Alternancia Menos // Más se puede expresar como **(-1) ^ syl.i** o **(-1) ^ $si**, dado que **syl.i** y **$si** son exactamente lo mismo.

En la siguiente Tabla están las **Equivalencias** entre las **Variables de Punto** y las **Variables Dólar**:

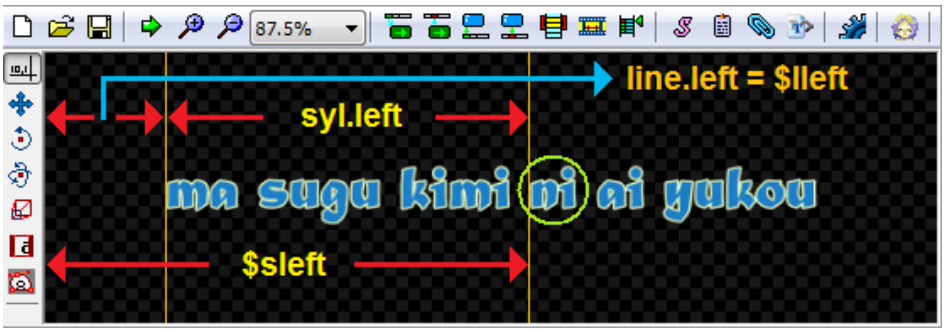
Equivalencias Entre Variables		
Variables de Punto		**Variables Dólar**
line.actor	==	$actor
line.style	==	$style
line.layer	==	$layer
line.start_time	==	$lstart
line.end_time	==	$lend
line.duration	==	$ldur
line.i	==	$li
line.left	==	$lleft
line.center	==	$lcenter
line.right	==	$lright
line.top	==	$ltop
line.middle	==	$lmiddle
line.bottom	==	$lbottom
line.x	==	$lx
line.y	==	$ly
line.height	==	$lheight
line.width	==	$lwidth
line.eff_margin_l	==	$margin_l
line.eff_margin_r	==	$margin_r
line.eff_margin_t	==	$margin_t
line.eff_margin_v	==	$margin_v
line.eff_margin_b	==	$margin_b
syl.start_time	==	$sstart
syl.end_time	==	$send
syl.duration	==	$sduration
syl.kdur	==	$skdur == $kdur
syl.i	==	$si
line.kara.n	==	$syln
syl.left	==	$sleft - $lleft
syl.center	==	$scenter - $lleft
syl.right	==	$sright - $lleft
syl.height	==	$sheight
syl.width	==	$swidth

Todas las **Equivalencias** entre las **Variables de Punto** y las **Variables Dólar** referentes a las distancias no son exactamente iguales, hay a veces una pequeña diferencia de no más de 1 Pixel, que en la mayoría de los casos es despreciable, y depende únicamente del redondeo que se hace en la igualación entre las Variables. Se deben tener en cuenta este detalle para el caso en que se necesite elegir alguna de las Variables referentes a alguna distancia.

Entre las **Equivalencias** presentadas en la Tabla vista anteriormente hay algunas Variables de Punto que son iguales a la Diferencia entre dos Variables Dólar, y en las siguientes imágenes veremos de forma gráfica sus posiciones exactas en el video:

En esta imagen se puede ver la relación que tiene **syl.left** con una sílaba cualquiera de una Línea de Karaoke, que para los próximos tres ejemplos es la Sílaba "**ni**" resaltada en las imágenes.

$sleft es la distancia medida en pixeles que hay desde la izquierda del video hasta la izquierda de la Sílaba y **syl.left** es la distancia medida en pixeles entre el lado izquierdo de la Línea de Karaoke y el lado izquierdo de la Sílaba:



O sea que **syl.left** en la primera Sílaba de una Línea de Karaoke cualquiera siempre equivale a cero, a diferencia de **$sleft**, que equivaldría a **line.left** para esa primera Sílaba.

En un caso similar, **$scenter** equivale a la distancia medida en pixeles entre el lado izquierdo del video y el centro de la Sílaba, y por otro parte, **syl.center** equivale a la distancia medida en pixeles que hay entre el lado izquierdo de la Línea de Karaoke y el centro de la Sílaba.

Y la última de estas Equivalencias es la siguiente: **$right** es la distancia medida en pixeles entre el lado izquierdo del video y el lado derecho de la Sílaba, y **syl.right** es la distancia medida en pixeles que hay entre el lado izquierdo del video y el lado derecho de la Sílaba:

Es importante tener en cuenta estas diferencias entre las **Variables de Punto** y las **Variables Dólar**, más que todo, para aquellas Plantillas que contienen los Tags **\clip** e **\iclip** en sus Script.

Del uso de las **Variables** depende mucho su memorización y su dominio en los **Script de Efectos** y marca la diferencia entre los Efectos de los principiantes.

Trataré que en este Capítulo queden definidas las Variables de Punto más importantes a parte de las que tienen su equivalencia con las Variables Dólar, ya que hay unas que considero de Nivel Superior y las veremos más adelante.

El primer grupo de **Variables de Punto** tienen la **Función** de **Retornar** las características del **Karaoke** asignadas en el **Estilo** que previamente habíamos modificado. Son muy útiles para hacer que los Efectos vuelvan a su estado primario luego de realizar las Transformaciones, como por ejemplo, hacer que los colores cambien al momento de que la Sílaba sea karaokeada y después regresen a los colores predeterminados:

line.styleref.fontname	Retorna el nombre de la fuente **\fn** que se haya seleccionado en el Estilo del Karaoke.
line.styleref.fontsize	Retorna el tamaño de la fuente **\fs** que se haya seleccionado en el Estilo del Karaoke.
line.styleref.color1	Retorna el color primario **\1c** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.color2	Retorna el color secundario **\2c** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.color3	Retorna el color del borde **\3c** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.color4	Retorna el color de la sombra **\4c** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.bold	Negrita
line.styleref.italic	Itálica
line.styleref.underline	Subrayado
line.styleref.strikout	Tachado
line.styleref.outline	Retorna el tamaño del borde **\bord** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.shadow	Retorna el tamaño de la sombra **\shad** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.scale_x	Retorna el porcentaje de la escala con respecto al Eje “x” **\fscx** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.scale_y	Retorna el porcentaje de la escala con respecto al Eje “y” **\fscy** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.spacing	Retorna el valor medido en pixeles del espacio entre las letras **\fsp** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.angle	Retorna el ángulo medido en grados con respecto al Eje “z” **\frz** de la fuente que se haya seleccionado en el Estilo del Karaoke.
line.styleref.bordstyle	Retorna el estilo de las líneas de Subtítulos, 1 para normal y 3 para caja opaca.
line.styleref.align	Retorna el valor de la Alineación **\an** de las Líneas de Subtítulos que se haya asignado.

line.styleref.margin_l	Retorna la medida en pixeles del margen Izquierdo de las Líneas de Subtítulos que se haya asignado.
line.styleref.margin_r	Retorna la medida en pixeles del margen Derecho de las Líneas de Subtítulos que se haya asignado.
line.styleref.margin_t	Retorna la medida en pixeles del margen Superior de las Líneas de Subtítulos que se haya asignado.
line.styleref.margin_v	Retorna la medida en pixeles del margen Vertical de las Líneas de Subtítulos que se haya asignado. Tiene la misma Función que **line.styleref.margin_t**
line.styleref.margin_b	Retorna la medida en pixeles del margen Inferior de las Líneas de Subtítulos que se haya asignado.

De todas las **Variables de Punto** concernientes a las características del **Estilo**, sólo omití tres de ellas: **line.styleref.encoding**, **line. styleref.relative_to** y **line.styleref.vertical**, porque considero que por ahora no son necesarias en este Nivel de desarrollo de Efectos Karaoke.

Las siguientes **Variables de Punto** hacen referencia a las características específicas de las Sílabas de cada Línea, como la posición o el tiempo. Ejemplo:

line.kara[syl.i].left	Retorna la coordenada en “x” de la posición Izquierda de cada Sílaba, que expresada así es equivalente a **syl.left**.
line.kara[syl.i + 1].left	Expresada de esta forma, ahora retorna la coordenada en “x” de la posición Izquierda, pero de la **Siguiente Sílaba**.
line.kara[syl.i - 1].left	Expresada de esta forma, ahora retorna la coordenada en “x” de la posición Izquierda, pero de la **Anterior Sílaba**.
line.kara[$syln].left	Expresada de esta forma, ahora retorna la coordenada en “x” de la posición Izquierda, pero de la **Última Sílaba** de cada Línea.
line.kara[$syln - 1].left	Expresada de esta forma, ahora retorna la coordenada en “x” de la posición Izquierda, pero de la **Penúltima Sílaba** de cada Línea.

Estas **Variables de Punto** dependen de la Silaba que se le indique y de la Variable final, que para el ejemplo anterior era **.left == $sleft == syl.left**. El resto de este tipo de Variables funciona de manera similar:

line.kara[____].right	Retorna la coordenada en “x” de la posición Derecha de la Sílaba Indicada.
line.kara[____].center	Retorna la coordenada en “x” de la posición Central de la Sílaba Indicada.
line.kara[____].height	Retorna la altura de la Sílaba Indicada, que para todas es la misma y equivale al tamaño de la Fuente.
line.kara[____].width	Retorna el ancho de la Sílaba Indicada.
line.kara[____].start_time	Retorna el Tiempo de Inicio en milisegundos de la Sílaba Indicada.

line.kara[____].end_time Retorna el Tiempo Final en milisegundos de la Sílaba Indicada.

line.kara[____].duration Retorna el Tiempo Total de la Duración medida en milisegundos de la Sílaba Indicada.

El siguiente grupo de Variables de Punto que veremos, hacen referencia a las Líneas y Sílabas, antes y después de la Sincronización del Karaoke:

line.text Retorna el Texto de la Línea, incluido cualquier Tag o Efecto que se le haya agregado.

line.text_stripped Retorna el Texto de la Línea, pero sin ningún tipo de modificaciones, como una Línea simple de Subtítulos.

syl.text Retorna el Texto de la Sílaba, incluido cualquier Tag o Efecto que se le haya agregado.

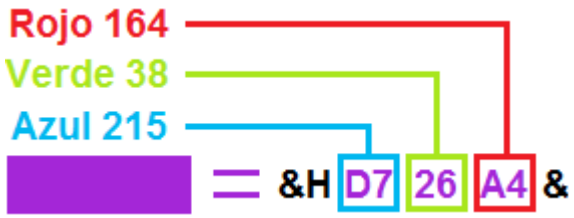
syl.text_stripped Retorna el Texto de la Sílaba, pero sin ningún tipo de modificaciones.

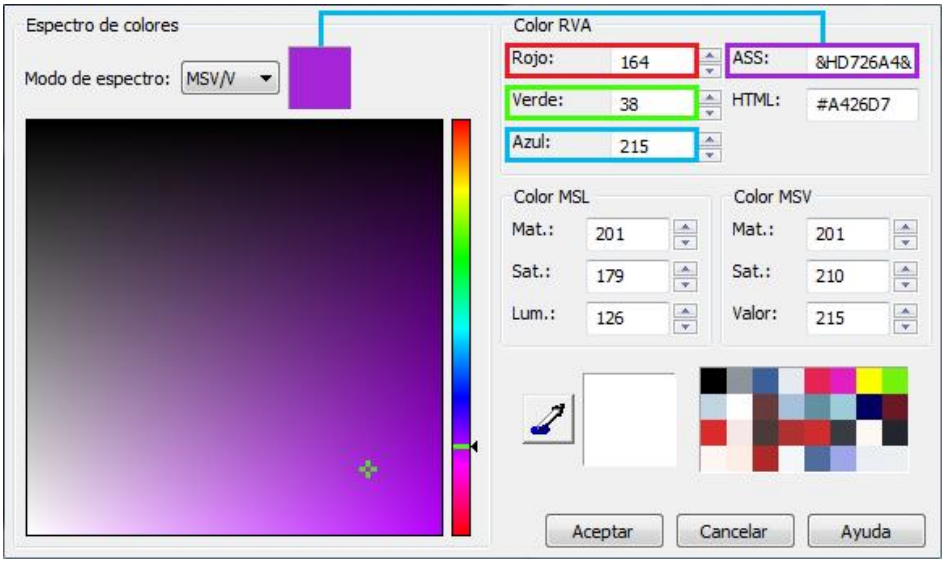
Las cinco siguientes Variables de Punto hacen referencia a las Características Cromáticas en cuanto a Color y Transparencia:

_G.ass_color(R, G, B) Retorna un color en Código Ass al asignarle valores numéricos en base decimal. Los valores asignados a **R**, **G** y **B** (Red, Green, Blue) oscilan entre 0 y 255. Ejemplo:

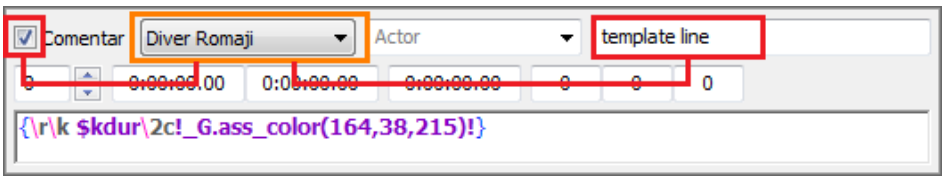
El Código Ass de uno de los Tonos del color "**Violeta**" que está en la imagen es: **&HD726A4&**, y en los rectángulos rojo, verde y azul, se ven los valores en base decimal que conforman el "**Violeta**", y la relación es así:

Tono	Base Decimal (10)	Base Hexadecimal (16)
Rojo	164	A4
Verde	38	26
Azul	215	D7





Aplicaciones:

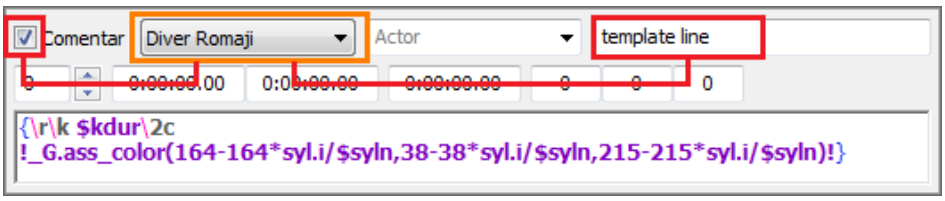


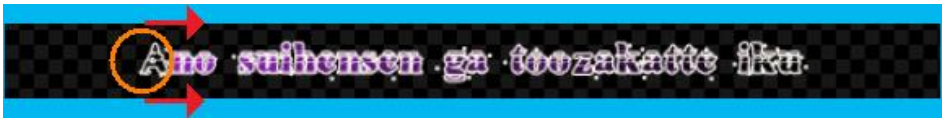
Ahora se ha aplicado el "Violeta" al Color Secundario **\2c** por medio de los valores asignados en Base Decimal; 164, 38 y 215.

Usada de esta forma, la Variable **_G.ass_color** no tendría mucho sentido aplicarla, dado que hay Tags que cumplirían con la misma función, pero cuando se cambian los constantes asignadas por Variables, es cuando cobra su máximo esplendor.

Recordemos que el color negro en Código Ass es: **&H000000&**, es decir, 00 para el rojo, 00 para el verde y 00 para el azul: **_G.ass_color (0, 0, 0)**. Entonces, si queremos hacer una **Degradación** en el Violeta del ejemplo y el negro, lo que debemos hacer el que cada uno de los valores del Violeta (164, 38, 25) se vuelvan Ceros (0, 0, 0):

La expresión **164 – 164 * syl.i /$syln**, toma el valor de 0 cuando **syl.i == $syln**, e igual sucede con **38 – 38 * syl.i /$syln** y **215 – 215 * syl.i /$syln**. Entonces la **Degradación** entre el violeta y el negro sería:

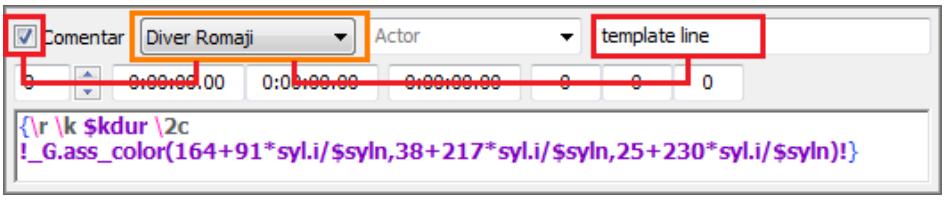




El Código Ass del color blanco es: **&HFFFFFF&**, es decir, FF para el rojo, FF para el verde y FF para el azul: **_G.ass_color (255, 255, 255)**. Entonces, para hacer una **Degradación** entre el Violeta y el Blanco, lo que debemos hacer el que cada uno de los valores del Violeta (164, 38, 25) se vuelvan 255: (255, 255, 255):

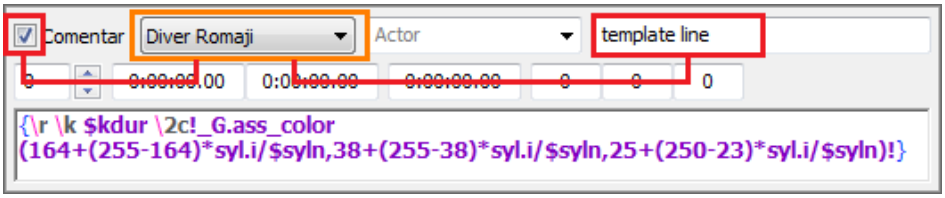
255 - 164 = 91
255 - 38 = 217
255 – 25 = 230

Se haya la diferencia entre 255 y cada uno de los valores del color violeta y ese es el valor que multiplicado por **syl.i/$syln** se debe sumar:

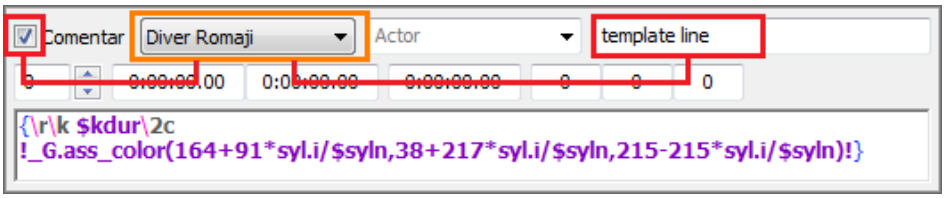


Se crea la ilusión de una **Fuente de Luz** desde la derecha del texto:

En el caso de que se dificulte la operación para hallar la diferencia entre 255 y cada uno de los valores, podemos hacer que el mismo Efecto realice los cálculos:



Ahora vemos la Degradación entre el violeta y el amarillo:



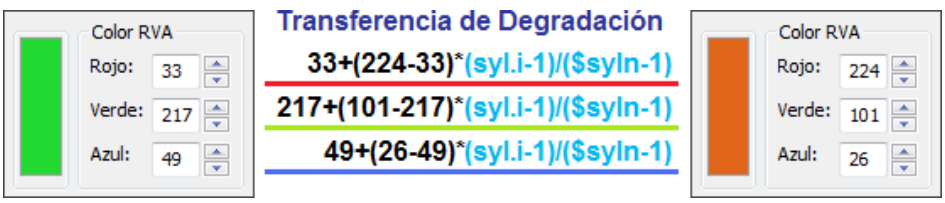
La siguiente tabla muestra los valores en Código Ass y en Base Decimal de los colores notables. Con los valores en Base Decimal podemos hacer la distintas Degradaciones entre ellos:

Valores de los Colores Notables								
Color		Código Ass			Valores Base Decimal			
		Azul	Verde	Rojo	Rojo	Verde	Azul	
	&H000000&	00	00	00	00	00	00	
	&H0000FF&	00	00	FF	255	00	00	
	&H00FF00&	00	FF	00	00	255	00	
	&HFF0000&	FF	00	00	00	00	255	
	&H00FFFF&	00	FF	FF	255	255	00	
	&HFF00FF&	FF	00	FF	255	00	255	
	&HFFFF00&	FF	FF	00	00	255	255	
	&HFFFFFF&	FF	FF	FF	255	255	255	

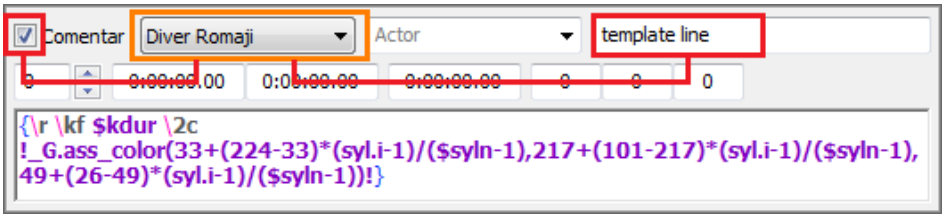
En realidad, se puede hacer la Degradación entre los dos colores de nuestra elección, sabiendo los valores de los colores en Base Decimal y siguiendo los respectivos procedimientos:

Transferencia	Algoritmo
De 0 a 255	255 * (syl.i - 1) / ($syln - 1)
De 255 a 0	255 - 255 * (syl.i - 1) / ($syln - 1)
De A a B	A + (B - A) * (syl.i - 1) / ($syln - 1)

Es un algoritmo relativamente simple, y más si se sabe que con la última expresión es suficiente para hacer la Transferencia entre los valores en Base Decimal de los colores, ejemplo:



Se evidencia cómo se usó la **Transferencia de A a B** para la **Degradación**.



Con la **Transferencia de A a B** es posible realizar un **Degradación** entre dos colores cualesquiera seleccionados al gusto, o se puede usar la **Función Variable** _G.ass_color con valores asignados aleatoriamente, sabiendo que los valores deben estar en el rango de 0 a 255, ejemplo:

(math.random(0,255), math.random(0,255), math.random(0,255))

Aunque para el Nivel de Habilidad que ya deben tener, no recomiendo este tipo de asignación, dado que es demasiado básica y hay demasiada incertidumbre en los resultados.

Si se deben usar valores aleatorios, recomiendo los siguientes algoritmos:

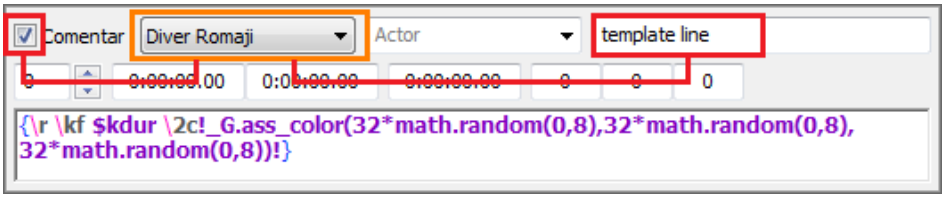
(16*math.random(0,16), 16*math.random(0,16), 16*math.random(0,16))

(32*math.random(0,8), 32*math.random(0,8), 32*math.random(0,8))

(64*math.random(0,4), 64*math.random(0,4), 64*math.random(0,4))

O pueden combinarlas para que no quede tan monótono el Efecto. De todas formas, más adelante veremos algunos usos interesantes de asignar los valores de forma aleatoria.

Esta es una pequeña muestra de los resultados del color secundario asignando valores aleatorios:



Como pueden observar, el resultado es muy básico y demasiado colorido cuando se usa asignando valores aleatorios a los colores primario y secundario, es por ello que la **Degradación** es una mejor opción para los colores de la fuente, aunque para algunos le parezca demasiado complicado hacer la Transferencia de los Valores.

La siguiente **Variable de Punto** que veremos es la Herramienta perfecta para hacer la **Degradación** del Texto de las Líneas o de las Sílabas; y al igual que la Variable _G.ass_color, veremos maneras simples y complejas de usarla, y aunque tengo claro que esto no es un manual de Matemáticas o Cálculo, es muy importante que entiendan cómo funcionan cada uno de los Efectos planteados para que puedan crear los suyos propios.

_G.interpolate_color (n1, “Color 1”, “Color 2”)

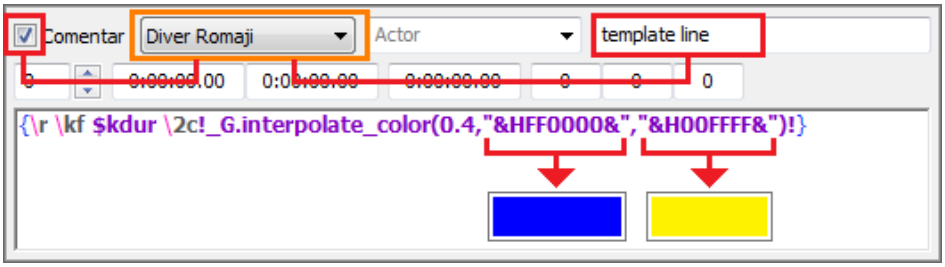
La Función Variable **_G.interpolate_color** entre dos colores, retorna un tercer color que depende del valor **n1** que le asignemos:

El valor de **n1** (Nivel de Interpolación), puede ser un número real cualquiera entre 0 y 1: **0 <= n1 <= 1**

Cuando **n1** = 0, retorna el **Color 1**

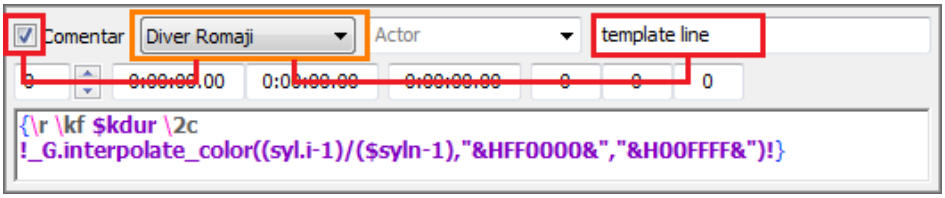
Cuando **n1** = 1, retorna el **Color 2**

Y para los demás valores de **n1** entre 0 y 1, retorna la Interpolación de los dos colores dependiendo qué tan cerca se encuentre de 0 o de 1, ejemplo:



El resultado se asemeja más al azul que al amarillo dado que **n1** está más cerca de 0 que de uno.

Siempre que **n1** sea un valor **Constante**, el resultado será un único color; el Efecto de **Degradación** se ve cuando **n1 varía** entre 0 y 1. Recordemos que la expresión **(syl.i - 1) / ($syln - 1)** equivale a 0 cuando **syl.i = 1**, y a 1, cuando **syl.i = $syln**

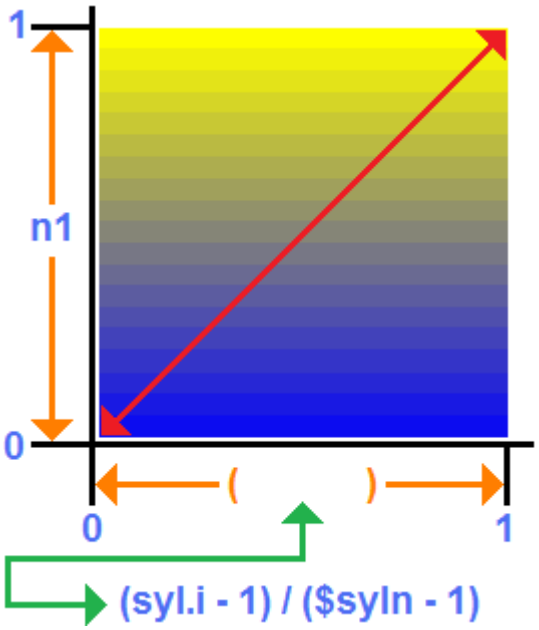


La expresión **(syl.i - 1) / ($syln - 1)**, es muy usada a lo largo de estos últimos Capítulos y es por el Rango que tiene entre 0 y 1, que nos es de tanta utilidad en el desarrollo de los Efectos y la veremos con mucha regularidad de ahora en adelante.

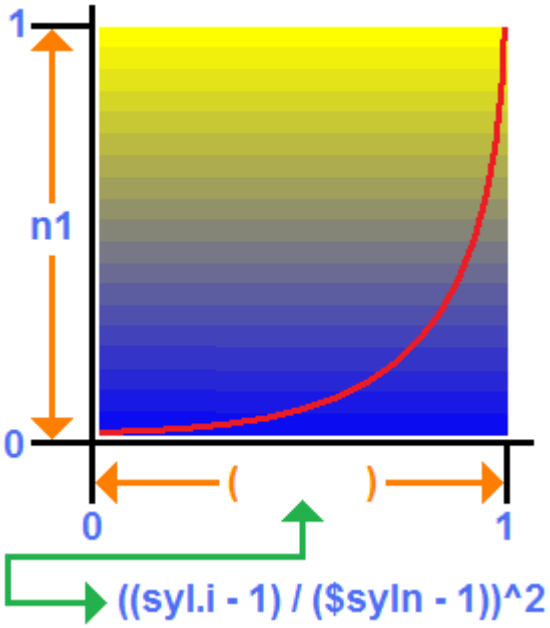
Y ahora ya es posible visualizar la **Degradación** entre el Azul (**&HFF0000&**) y el Amarillo (**&H00FFFF&**), y el resultado es exactamente igual al que hacíamos con la Transferencia de valores con la Variable **_G.ass_color**, pero mucho más simple y fácil de aplicar. Es aquí en donde se acaban las coincidencias entre las Variables **_G.ass_color** y **_G.interpolate_color**, y de ahora en adelante veremos las múltiples posibilidades que tenemos con la Interpolación de los colores y las herramientas con las que contamos para obtener el máximo rendimiento de estas Variables y sus aplicaciones.

Y como para variar, se viene otra necesaria clase de **Matemáticas** para poder entender mejor el origen de los procedimientos aplicados.

Si en la Función Variable **_G.interpolate_color** se usa el algoritmo **(syl.i - 1) / ($syln - 1)** para **n1**, la Interpolación entre los dos colores es "**Lineal**", como se puede ver en la gráfica:

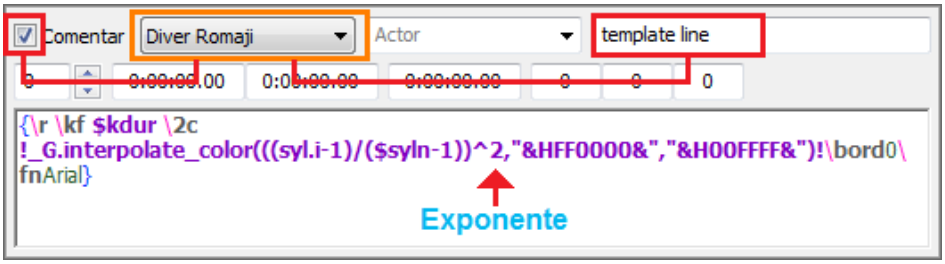


La "**Interpolación Lineal**" hace que la **Degradación** sea **Simétrica** para todos los casos, pero otras formas de plantear la Interpolación para evitar la Simetría en los Colores. Si el camino que toma la Interpolación fuera una "**Curva**" en vez de una **Línea Recta**, los resultados variarían:



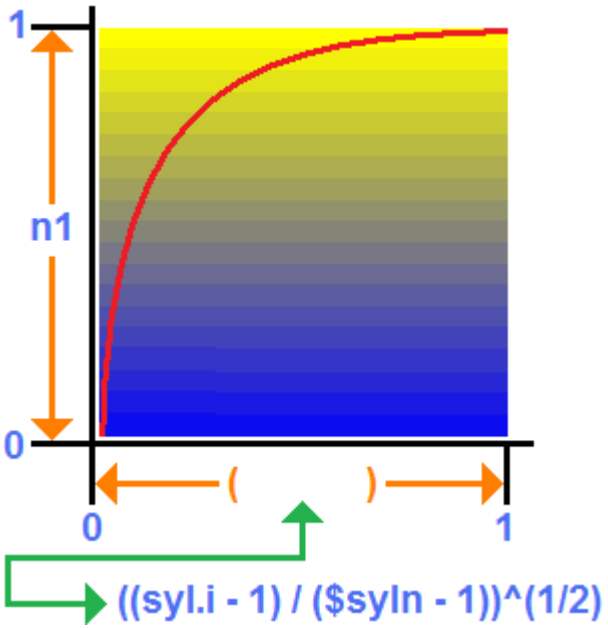
La "**Interpolación de Curva**" hace que la **Degradación** tienda más al primer color que al segundo, y entre mayor sea el **Exponente** de la expresión, más amplia será esa tendencia.

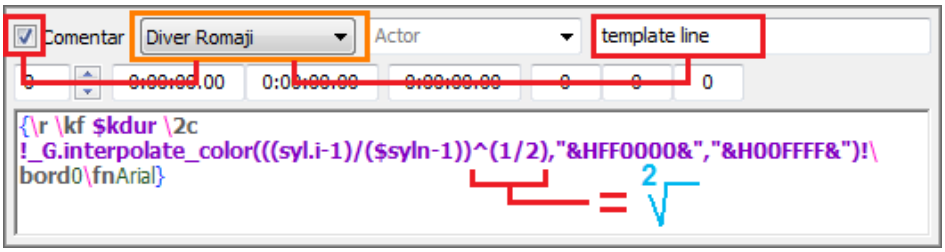
Las "Interpolaciones de Curva" son una recursiva herramienta para modificar los resultados de la Degradación, que nos puede ser muy útil para aquellos fondos en el video que tiene un contraste de colores muy marcado y definido.



Los cambios son evidentes, la Degradación ya no es Simétrica y la mayor parte del texto tiende al Azul y una mínima al Amarillo.

Si se invierte el rumbo de la curva, la tendencia de la Degradación es hacia el segundo color:

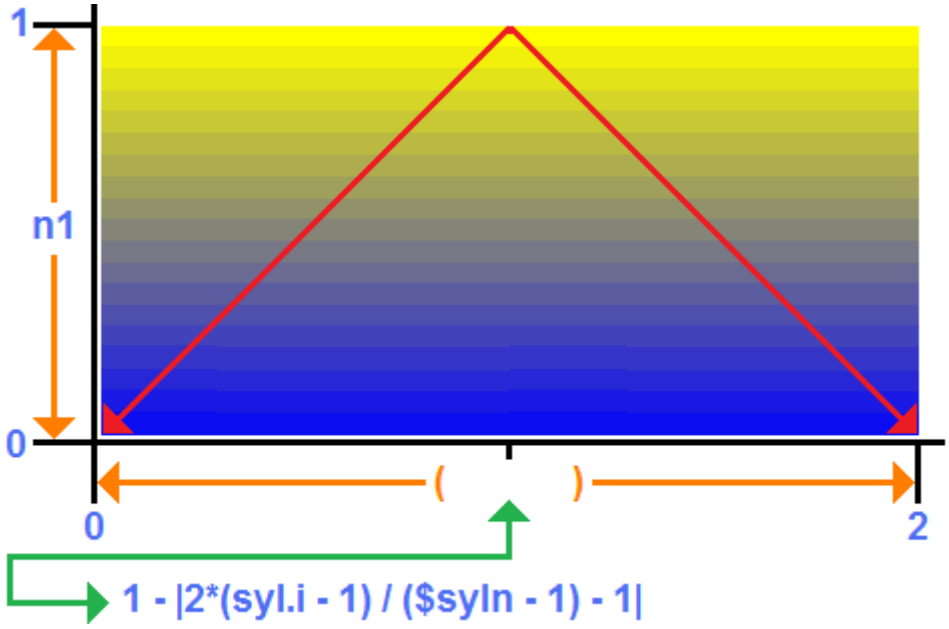


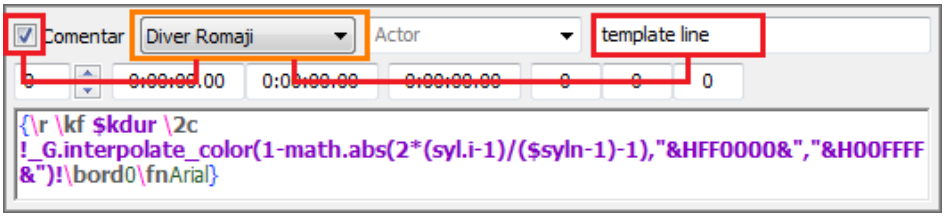


En este caso, el **Exponente** es (1/2 = 0.5) que equivale a la **raíz cuadrada**, y al igual que la primera **Interpolación de Curva**, si se aumenta el **Índice Radical** (raíz cúbica, raíz cuarta o raíz quinta) la tendencia hacia el segundo color será cada vez mayor.

Hasta ahora, para colores estáticos (que no están en un Tag **\t**), sólo hemos visto Degradaciones de dos colores y de un sólo sentido, es decir, pasar del primer color al segundo o viceversa. Pero las Interpolaciones que hemos visto no nos servirían si quisiéramos una Degradación de más de dos colores, o de dos colores, pero que luego de pasar al segundo color, regrese nuevamente al primero, o sea, que termine donde empezó.

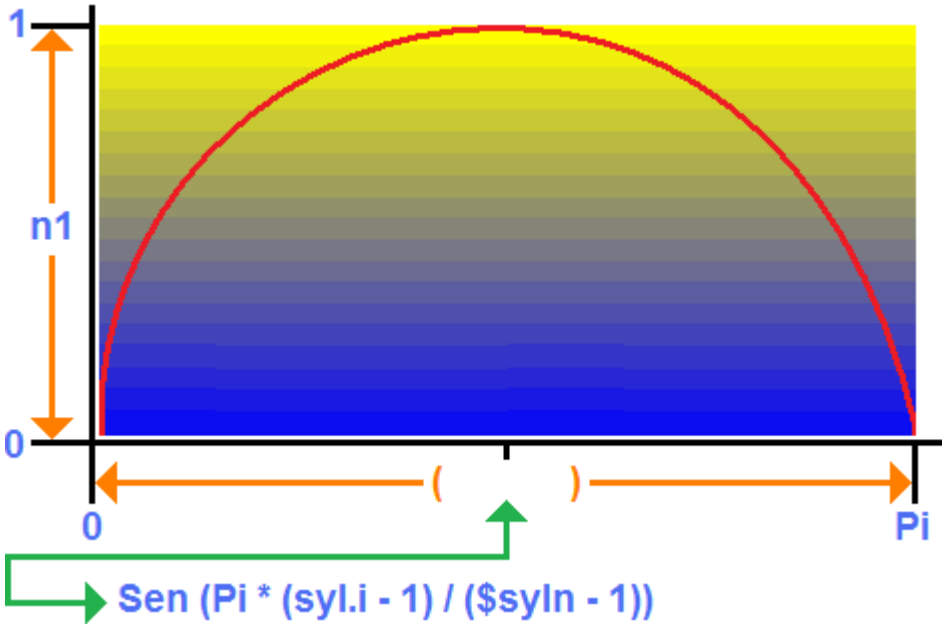
La “**Interpolación Lineal Cíclica**” soluciona el anterior problema, ya que nos brinda la posibilidad de hacer una **Degradación entre dos colores**, pero regresando al primero de ellos:

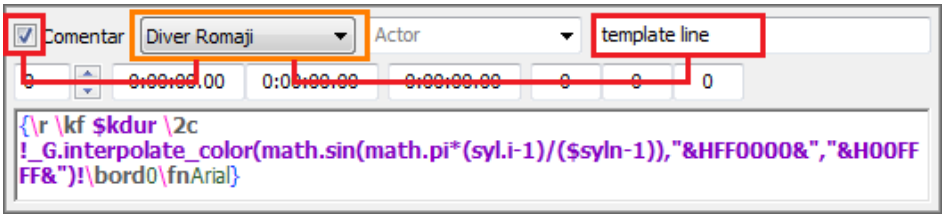




(Recordemos que la Función **math.abs** retorna el valor absoluto de una cantidad que le asignemos). La **Interpolación Lineal Cíclica** puede ser útil para crear el Efecto de Brillo central, si se coloca el de segundo el color blanco.

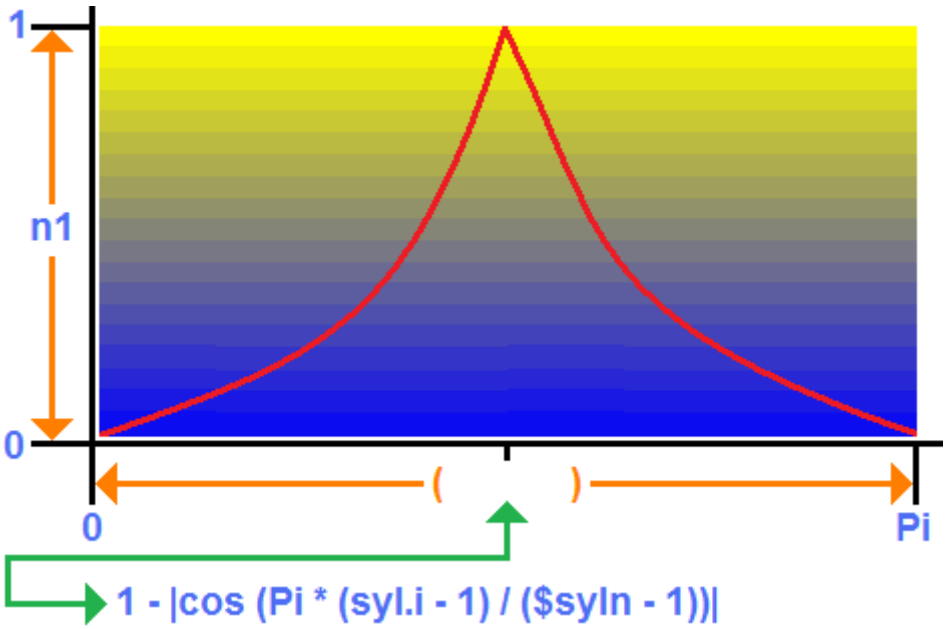
Otra manera de hacer la Interpolación Cíclica es por medio de la Funciones Trigonométricas y a las propiedades que éstas poseen. En la gráfica se puede ver la trayectoria de la Interpolación para regresar nuevamente al primer color:

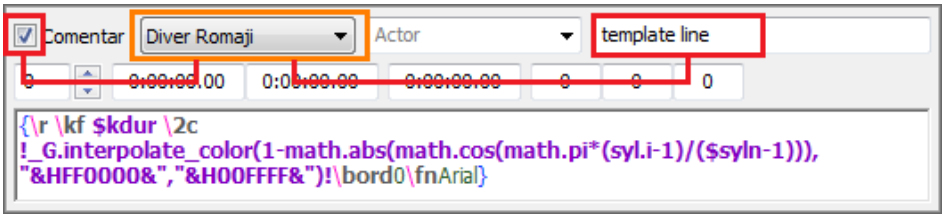




Los resultados son parecidos a los de la **Interpolación Lineal Cíclica**, pero ahora el segundo color (Amarillo), abarca más espacio en la Degradación.

Sacando el máximo potencial de todas las **Funciones Matemáticas** con las que contamos, es posible aprovechar las gráficas de las mismas para darle a la **Interpolación** las más curiosas trayectorias:



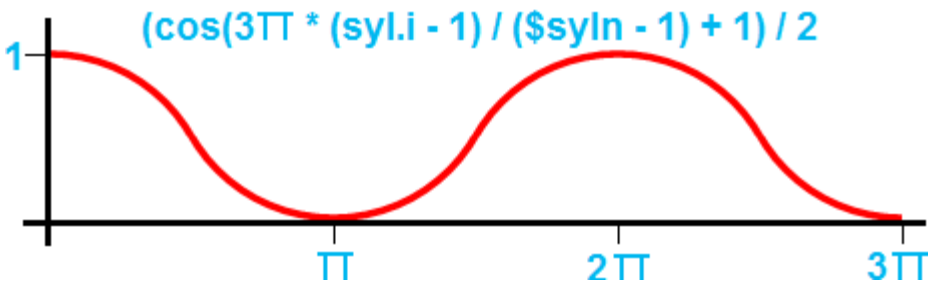


En este tipo de Interpolación, la Degradación sigue siendo de dos colores, pero como podemos ver, el texto vuelve al color inicial (Azul). Evidentemente, la selección de los tonos de los colores a Degradar debe hacerse basándose en el video y no simplemente copiando el **Script** de los ejemplos hasta ahora vistos.

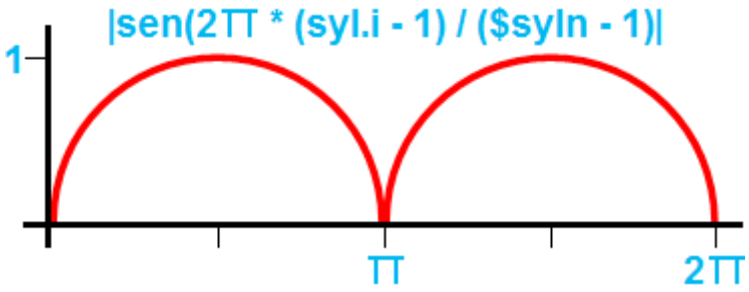
Hay muchos tipos de Interpolaciones, y escoger alguna depende del resultado que queramos obtener, y de la distribución de los colores a degradar.

A continuación mostraré múltiples opciones de Interpolaciones que son un poco más complejas y de ustedes mismo depende qué aplicaciones darle y en qué Efectos usarlos.

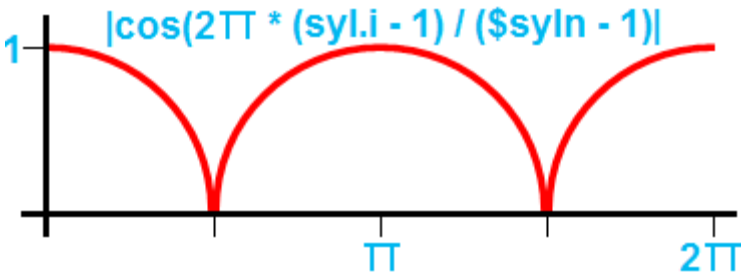
- **Interpolación Múltiple I**



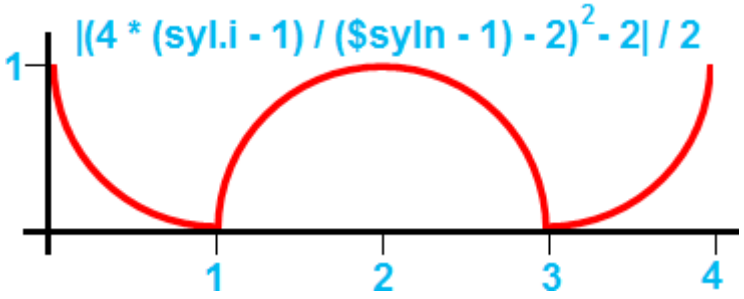
- **Interpolación Múltiple II**



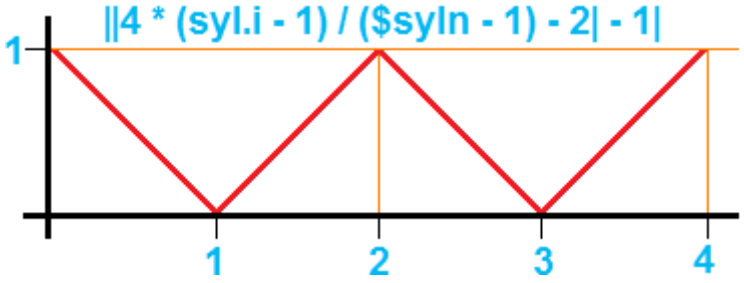
- **Interpolación Múltiple III**



- **Interpolación Múltiple IV**

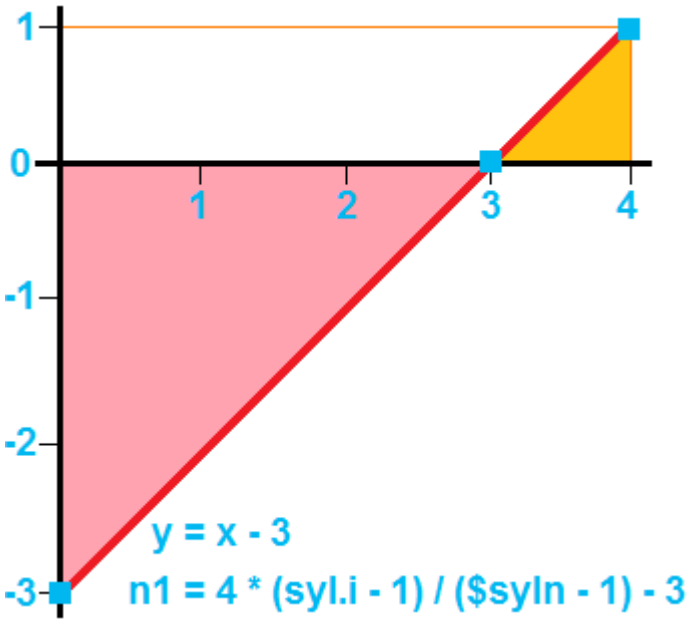


- **Interpolación Múltiple V**

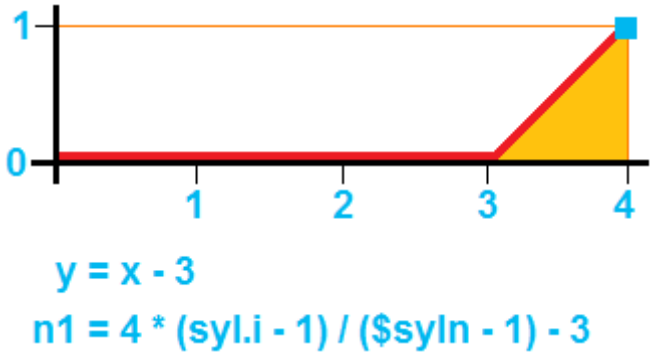


Estas **Interpolaciones Múltiples** se pueden usar tal y como están expresadas en las gráficas, también se pueden extender o recortar, dependiendo del gusto de cuada uno. Extender o Recortar las Interpolaciones depende de modificar el Dominio de las gráficas, Ejemplo: cambiando el 4 por un 2 en la Interpolación Múltiple V, la gráfica se reduce a la mitad. Todo es cuestión de práctica.

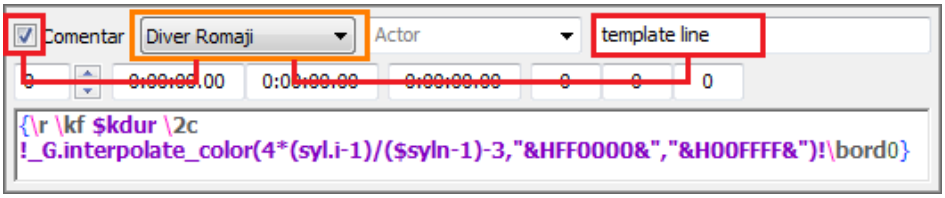
Unas de las formas más creativas de aprovechar las características de **n1** en las Interpolaciones es la siguiente: los valores de **n1** deben estar entre 0 y 1, cualquier valor inferior a 0, la Interpolación lo iguala a 0; y cualquier valor que exceda a 1, la Interpolación lo iguala a 1, ejemplo:



En la Gráfica, los valores de **n1** son negativos entre 0 y 3, y al ser menores que 0, la Interpolación los toma como 0:



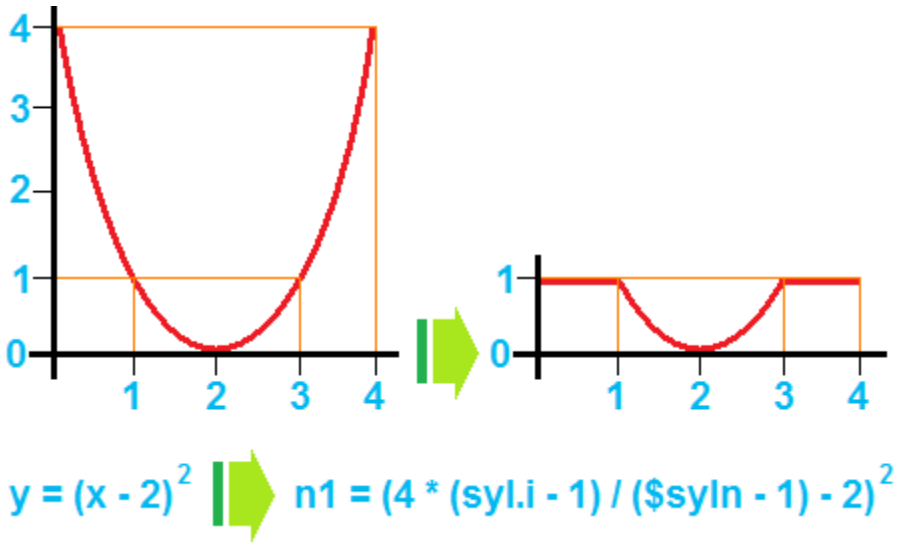
Es decir, que entre 0 y 3, **n1** siempre será 0, o sea, que las tres cuartas partes del texto tendrán el primer color, y en el último cuarto empieza la Degradación hasta el segundo color:



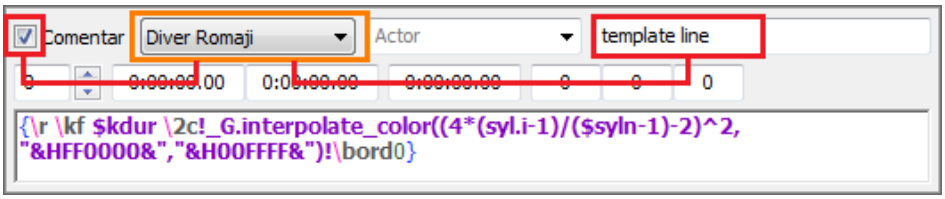
Exceder los Límites de **n1** es una forma simple de controlar de forma precisa el inicio de la Degradación.

Con una “desempolvada” a nuestro viejo libro de Álgebra, podemos hacer maravillas a la hora de las Degradaciones, y eso no es todo, todas estas ecuaciones que hemos visto en el transcurso de este Capítulo no serán de gran utilidad para muchos de los Efectos a desarrollar.

En la siguiente gráfica, se ve claramente la relación entre la función cuadrática y el comportamiento de n1 en la gráfica de dicha función:



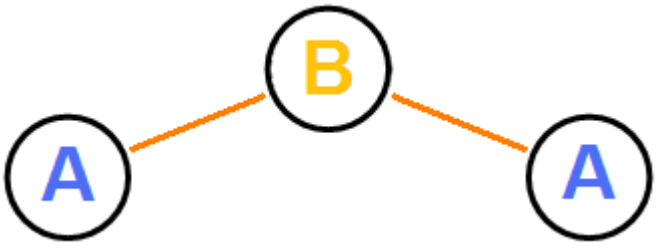
Ahora veremos el fragmento del **Script** y una visualización de cómo se vería la Degradación de dicha Interpolación:



El texto empieza ahora por amarillo, recordemos que cuando **n1** es igual a 1, la Interpolación retorna el segundo color.

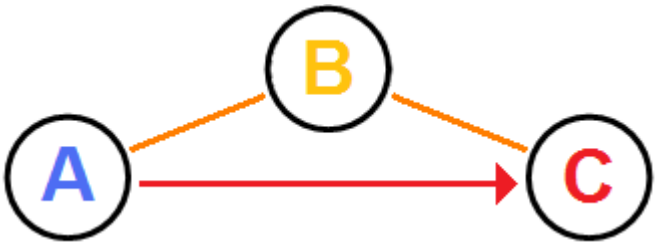
DEGRADACIÓN ENTRE 3 COLORES

A esta altura, no me sorprendería que ya hayan llegado a la conclusión de qué se debe hacer para poder lograr una **Degradación entre 3 Colores**; y para los que aún no saben cómo, empezaremos recordando la **Interpolación Lineal Cíclica**:

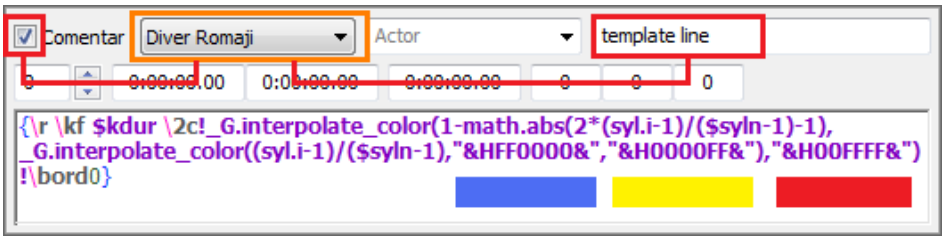


La **Interpolación Lineal Cíclica** consiste en pasar de un color **A** a un color **B**, para finalmente regresar al color **A.**

Entonces, lo que se debe lograr es que luego que la **Interpolación** pasó de **A** a **B** pase a un tercer color **C**, y así habremos logrado una Degradación entre 3 Colores. Para ello aprovecharemos la habilidad que tenemos de usar las Variables unas dentro de las otras y de combinarlas libremente para el que los Efectos que desarrollamos logren los propósitos que planeamos.



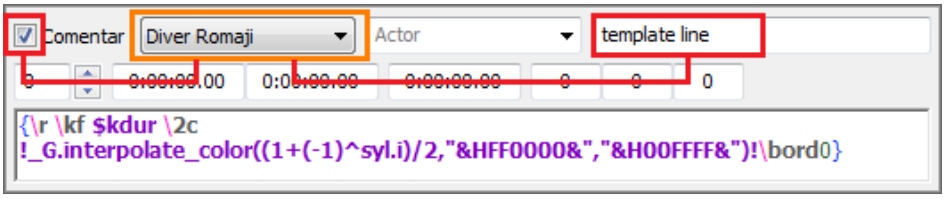
Lo gráfica indica que cuando la Interpolación haga que el color **A** pase al color **B**, al mismo tiempo **A** se transforme en **C**, para que cuando el color **B** intente regresar a **A** en realidad pasa a **C**. En el **Script** se vería así:



Lo que se hizo fue que, el primer color de la **Interpolación Cíclica** se remplazó por otra Interpolación, es este caso, Lineal. Entonces mientras la **Interpolación Lineal** hace el **Azul** pase al **Amarillo**, la **Interpolación Cíclica** hace que el **Amarillo** pase al **Rojo**:

Con las combinaciones de todas la Interpolaciones vistas hasta ahora, es posible lograr Efectos nuevos distintos de los vistos en el **KaraBook**. Esta forma de hacer las Degradaciones de 3 colores es la única que hasta ahora he visto que es compatible con la aplicación **Apply Karaoke Template** y por ende se puede Automatizar.

Y ya para concluir todo lo concerniente al tema de la **Variable de Punto** **_G.ass_color**, y haciendo uso de la alternancia **Off // On** vista en el Capítulo anterior, se puede lograr este tipo de resultados, que consiste precisamente en alternar los colores de la Variable por medio de la Interpolación:



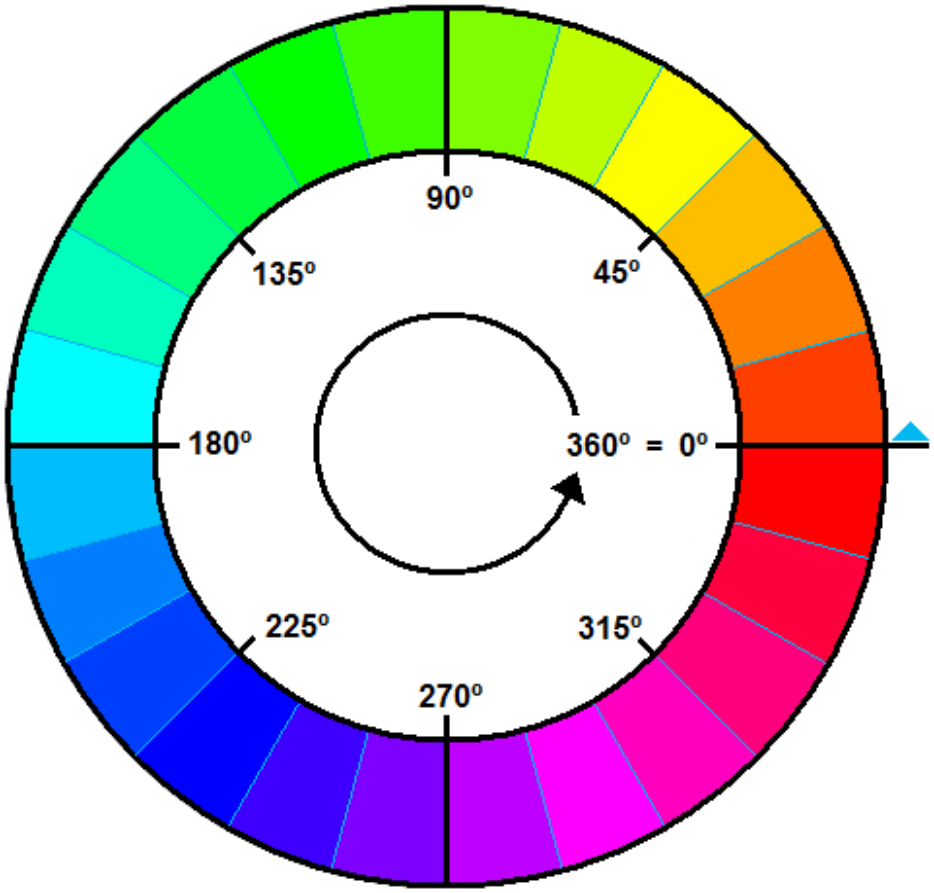
Sobra recordar que la elección de los colores en este y todos los ejemplos que se vienen planteando a lo largo del Capítulo, depende única y exclusivamente del gusto de cada uno de ustedes. La recomendación es que se dejen llevar por la canción y las imágenes del video, que ellos mismos dictan la forma que debe ir adquiriendo el Karaoke, dado que una de sus partes más importantes es la selección de los **Colores**, de la mano con una apropiada **Fuente** y unos novedosos **Efectos**.

No mostraré degradaciones de más de tres colores para dejarles la curiosidad de cómo hacerlo y que puedan ir ensayando cada uno de los ejemplos aquí planteados y experimentando con ellos. Además son muy poco recomendables las degradaciones de más de tres colores, a no ser que el video del Karaoke así lo requiera.

_G.ass_color (_G.HSV_to_RGB (h, s, v))

Es una **Variable de Punto** que al asignarle los valores **h**, **s** y **v**, nos retorna un color en formato Ass. En donde **h** es la inicial de "**Hue**" o **Tonalidad** en español, **s** es la inicial de "**Saturation**" o **Saturación** y **v** es la inicial de "**Value**" o **Valor** en español.

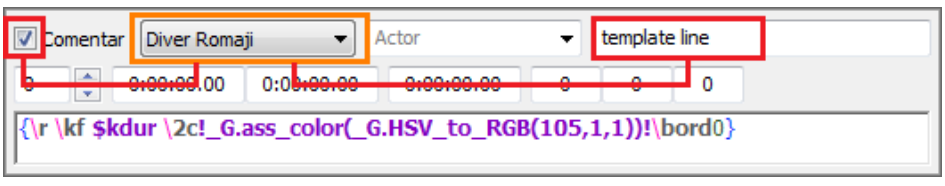
A **h** (Hue) se le debe asignar un valor numérico entre 0 y 360, que corresponde a las tonalidades del **Círculo Cromático de los Colores Primarios**, como se ve en la siguiente imagen:



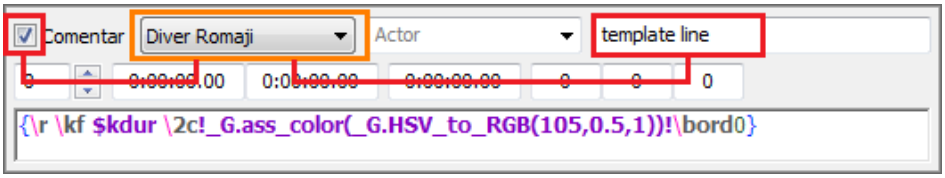
Este **Círculo Cromático** sólo incluye los tres colores primarios y los tres secundarios: **Amarillo**, **Azul**, **Rojo**, **Verde**, **Naranja** y **Violeta**; empezando por el Rojo, que pasa por el Naranja para llegar al Amarillo, este pasa por el Verde para llegar al Azul, quien a su vez pasa por el Violeta para nuevamente volver al Rojo. En la escala está la división cada 15º grados de las Tonalidades del Círculo.

La **Saturación** **s**, se mide en una escala de 0 a 1, en donde 1 es la Tonalidad (Hue) predeterminada que se haya escogido y 0 es totalmente blanco, entonces los valores entre 0 y 1 definen la **cantidad de blanco** de la Tonalidad.

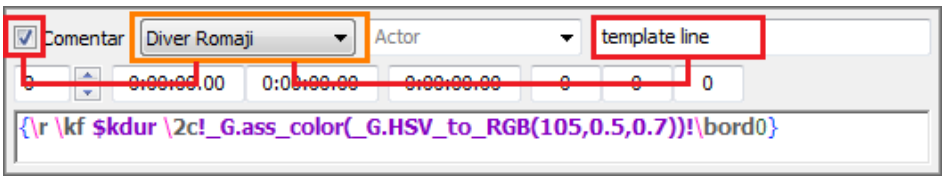
El **Valor** **v**, también se mide en una escala de 0 a 1, en donde 1 es la Tonalidad (Hue) predeterminada que se haya escogido y 0 es totalmente negro, es decir, que los valores entre 0 y 1 definen la **cantidad de negro** de la Tonalidad. Ejemplo:



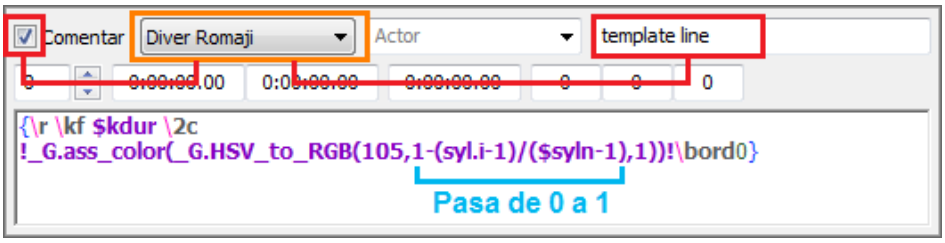
En el ejemplo anterior, la tonalidad es la 105 y los otros dos valores están en predeterminados (1, 1). Para el siguiente ejemplo, la Saturación es de 0.5, es por eso que el color ahora se ve un poco más pálido que en el ejemplo anterior:



Para este ejemplo, se han modificado todos los valores, pero todos en cantidades constantes:

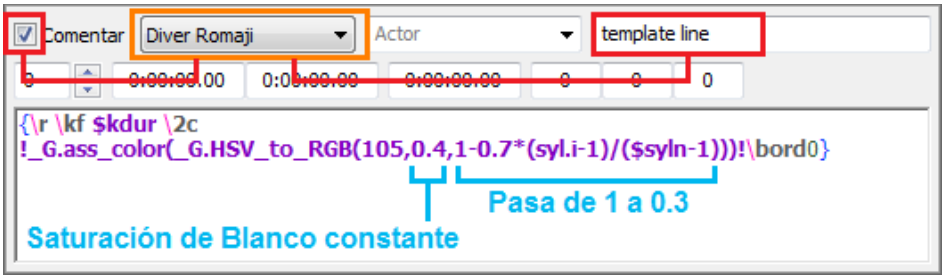


Si se modifica la Saturación de cantidad constante a cantidad variable, entonces se crea una Degradación entre la Tonalidad asignada y el Blanco:

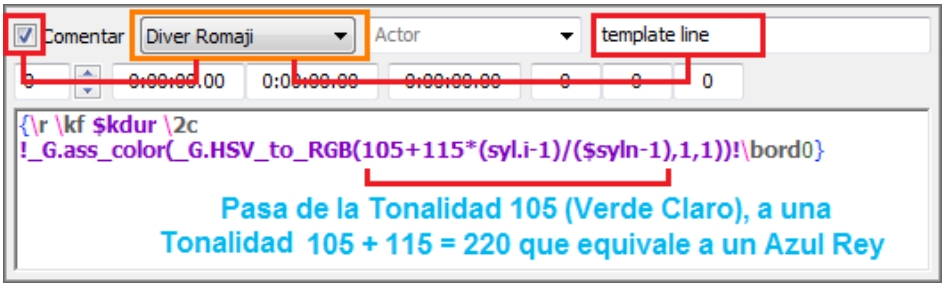


Es una forma más de hacer Degradaciones, aunque con esta Variable, si la Tonalidad es constante, entonces la Degradación es entre la Tonalidad y el Blanco o el Negro, según lo programemos.

Ahora, la cantidad variable es la que determina la cantidad de Negro (Value), y la Tonalidad y Saturación son constantes. Esto hace que la Tonalidad empiece con una Saturación de 0.4 y que se vaya oscureciendo por la cantidad variable:

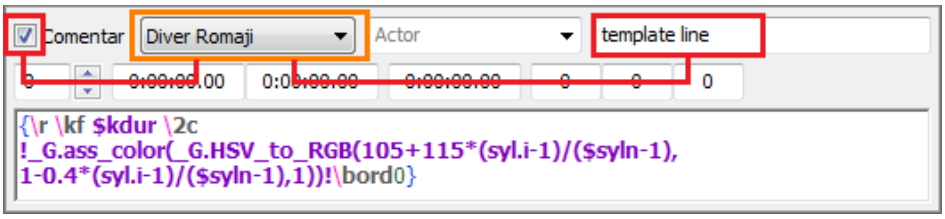


Otra forma de hacer Degradaciones es haciendo variable la Tonalidad, y dejar constantes a **s** y **v**:



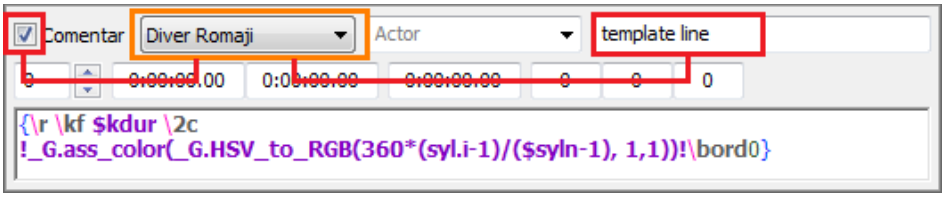
La Tonalidad variable empieza en 105, y para cada una de las Sílabas, va aumentando hasta llegar a una Tonalidad de 105 + 115 = 200, aprovechando que **(syl.i - 1) / ($syln - 1)** siempre empieza en 0 y termina en 1.

Para este ejemplo, la Tonalidad y la Saturación son variables, y crea una Degradación parecida a la del ejemplo anterior, pero con las Tonalidades un poco aclaradas:



Aunque por ahora, lo más recomendable en esta Variable es dejar siempre a **s** y **v** en predeterminados, o sea en 1. Esta recomendación es sólo para que las Degradaciones no se les hagan tan complicadas y mejor se coordinen con las anteriores Variables de Color que hemos visto y que ya han practicado.

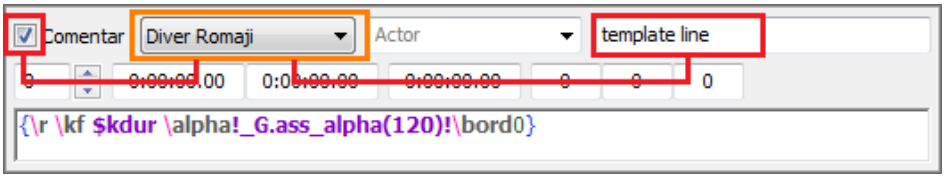
Y la forma más colorida de usar la Variable **_G.ass_color (_G.HSV_to_RGB)** es haciendo variable a la Tonalidad en todo su dominio, es decir, desde 0 hasta 360. De ahí que el Círculo Cromático de los Colores Primarios que vimos al comienzo de esta Variable, es muy importante para poder definir de dónde a dónde queremos que se ejecute la Degradación:



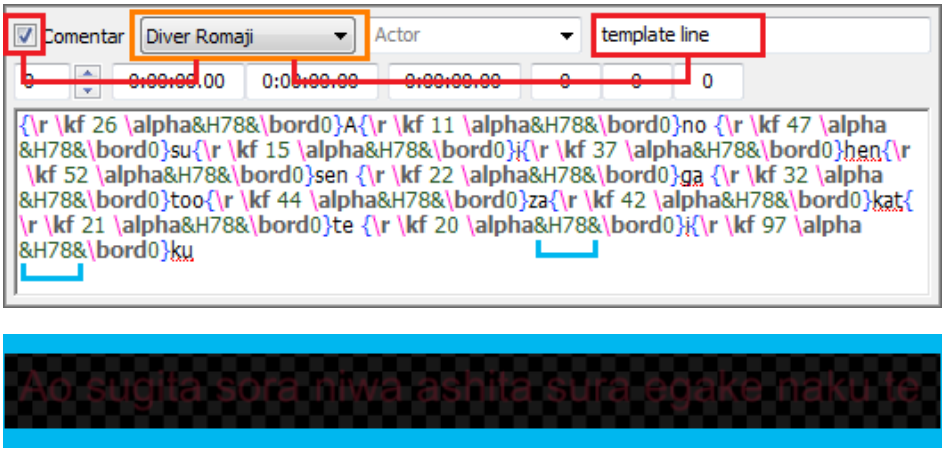
La Variable **_G.ass_color (_G.HSV_to_RGB)** es una de las Variables de Punto más simples de usar y ofrece resultados muy buenos en la medida que le asignemos conscientemente los valores a **h**, **s** y **v**. Esta Variable de Punto la usaremos con mucha regularidad cuando lleguemos al tema de las **Figuras**, ya que es una manera sencilla de darles color sin tener que complicarnos tanto por elegirlos de manera individual.

_G.ass_alpha (N)

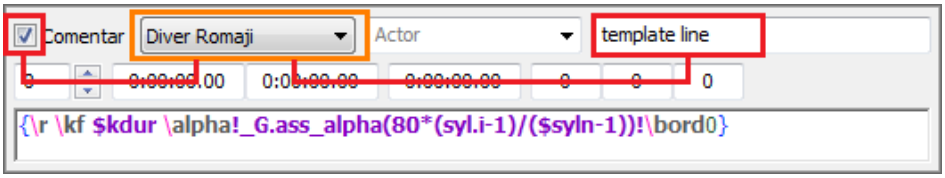
Es la Variable de punto que nos permite asignarle valores a **N** de 0 a 255 en Base Decimal, a la Transparencia (alpha), y los retorna en Base Hexadecimal.



El valor 120 de Tag **\alpha** que está dado en Base Decimal equivale a 78 en Base Hexadecimal.



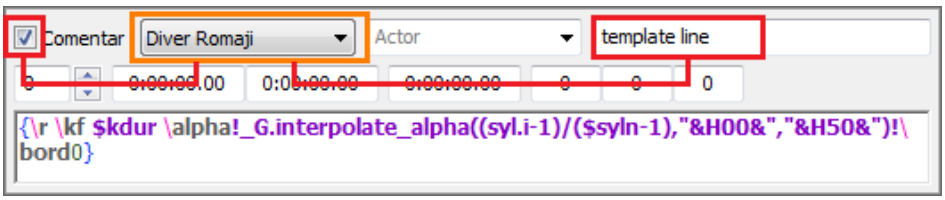
Una de las ventajas de esta Variable, es poder asignar valores variable a la Transparencia, como en el siguiente ejemplo:



_G.interpolate_alpha (n1, “a1”, “a2”)

Es la última Variable de Color que veremos en este Capítulo, y su forma de uso es muy similar a la Interpolación de Colores. Los valores asignados a **n1** definen la tendencia de la Transparencia; **a1** y **a2** son las dos Transparencias que asignamos.

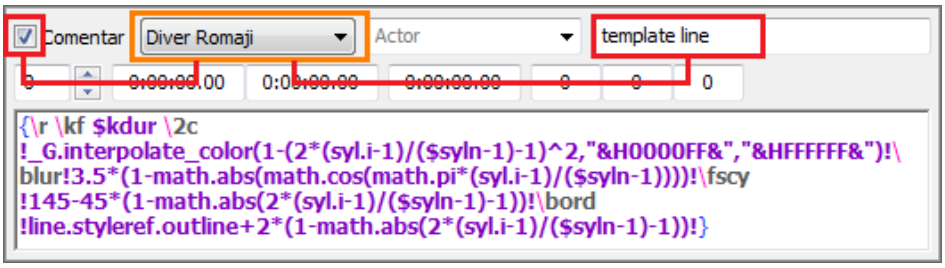
En otras palabras, esta Variable nos ayuda a hacer una Degradación, pero no una de colores, sino una de Transparencias:



Esta variable nos será de gran utilidad para desarrollar Efectos de Entrada y de Salida de nuestros Karaokes, además también las podemos usar en las **Figuras**, ya que las Degradaciones de Transparencias son útiles para causar la sensación de profundidad y distancia.

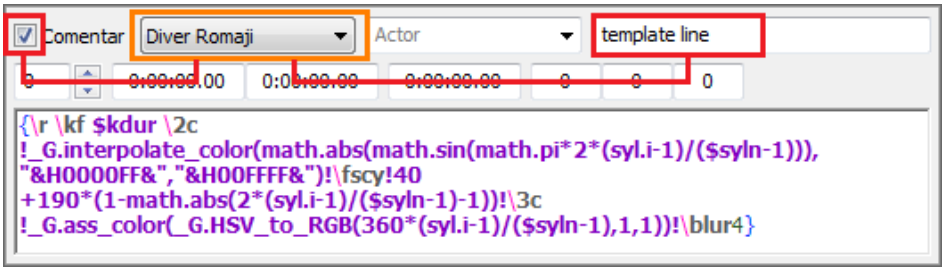
Ahora veremos unos cuantos ejemplos de los usos de las Variables vistas hasta el momento aplicadas en varios de los Tags. También es el momento para ver algo de la segunda etapa de la “template line” y nuevas Plantillas para poner en práctica:

Plantilla 015



Es una de las Plantillas que considero “**Plantilla Estática**”, por el hecho de no poseer ninguna Transformación en el **Script**. Esta Plantilla tiene múltiples Variantes que dependen de lo aprendido hasta aquí por cada uno de ustedes, dado a su condición Estática, es muy fácil agregarle alguna de las Transformaciones que han aprendido de la primera etapa de template line, como alargar la fuente, brillo o estelas:

Plantilla 016



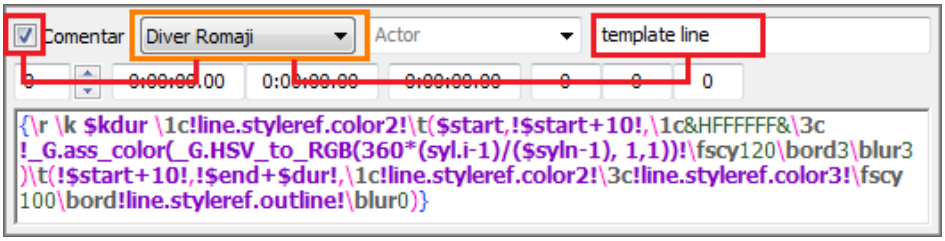
Esta es otra Plantilla Estática y la propongo para que puedan ver la variación en el tamaño de la fuente producida por el Tag \fscy. El borde es multicolor y el color secundario tiene una Degradación Cíclica entre el Rojo y el Amarillo:

Sobra mencionar que el uso de los colores se debe restringir a los necesarios en el video, ya que si usan el Script de esta Plantilla tal cual, sólo lograrán obtener el mismo resultado, que en una opinión personal, es demasiado colorido.

La ventaja de estas “**Plantillas Estáticas**” es que nos dan un punto de partida para el desarrollo de un Script más completo y unos Efectos más laboriosos. Otra de sus ventajas es la facilidad que tienen para fusionarse con las primeras Plantillas aprendidas y la mayoría de sus Transformaciones.

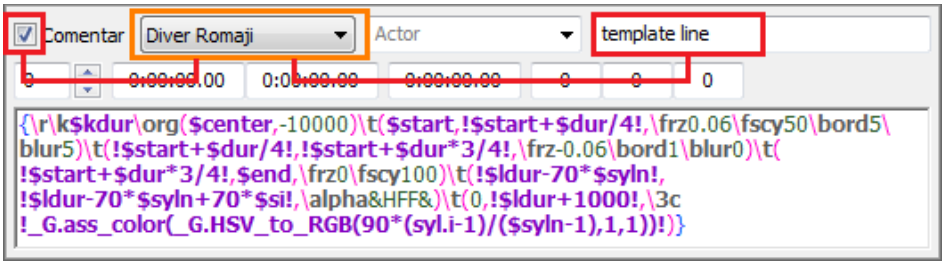
Son las dos primeras Plantillas Estáticas que he propuesto, pero a lo largo del resto del KaraBook, plantearé mucho más de ellas para que se apoyen en ellas y aprendan de sus partes y componentes; es por eso que es de vital importancia tener a mano los Algoritmos vistos y las Variables usadas.

Plantilla 017



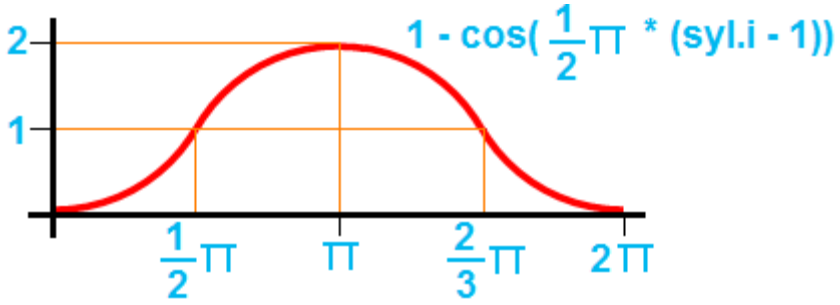
Esta Plantilla ya posee Transformaciones e incluye otra de las Variables vistas en este Capítulo, line.styleref.color y line.styleref.outline, que nos ayuda a que el Efecto retorne a su estado primario:

Plantilla 018



Esta Plantilla es, en principio, parecida a unas de las Plantillas vistas en la primera etapa de la template line, pero ahora las Sílabas no se mueven de arriba a abajo, sino de derecha a izquierda. También incluye un achicamiento de la Sílaba karaokeada durante la rotación, la ya conocida Estela de Salida y una Degradación en el color del Borde:

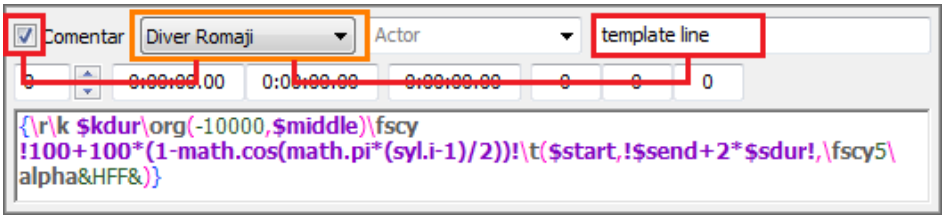
Para las próximas dos Plantillas haremos uso de un Algoritmo Trigonométrico muy útil en muchos Tags por los resultados que retorna:



Este Algoritmo siempre retorna los valores: 0, 1, 2, 1 y regresa nuevamente a 0. Sólo depende de syl.i, y al ser una función trigonométrica, su comportamiento en cíclico.

Existen muchos Algoritmos de este tipo, y trataré de plantear la mayor cantidad de ellos con sus respectivas aplicaciones y formas de uso, además de las gráficas y una breve explicación de cómo funciona.

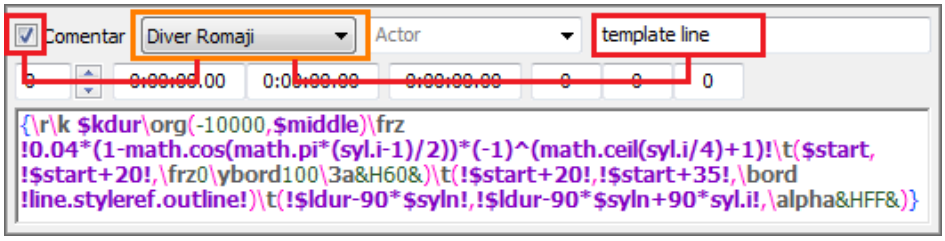
Plantilla 019



El anterior Algoritmo aplicado al Tag \fscy de la Plantilla 019, hace que el tamaño de la fuente sea 100, 200, 300, 200 y nuevamente 100. Estos valores se van alternando y crean un Efecto se Sierra Dentada en el texto.

Aunque la Plantilla 019 posee una Transformación, se podría considerar también como Plantilla Estática o de Base, y las variantes de ella las dejo a su imaginación y creatividad:

Plantilla 020



Aquí el Algoritmo es aplicado a las rotaciones del Tag \frz, el Algoritmo está multiplicado por otro que a su vez retorna cuatro 1 o cuatro -1 de manera cíclica y crea el efecto de ondulación en el texto.

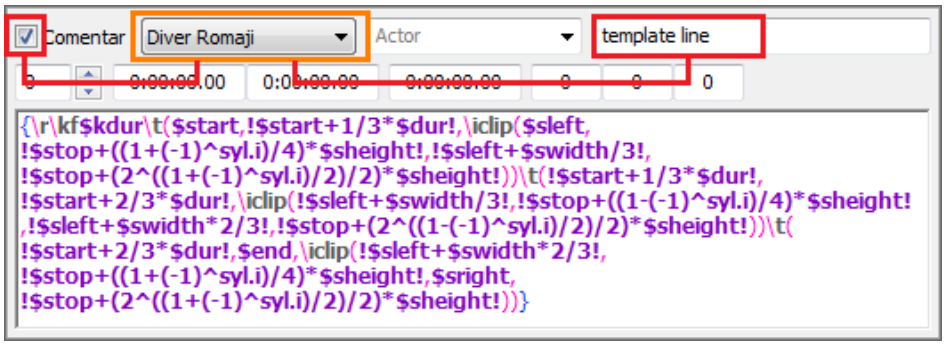
El efecto creado por el Tag \ybord, está a medio terminar para dejar espacio al toque personal:

Esta segunda etapa de la template line presume un nivel más elevado de dificultad, aunque considero que aún estamos en la creación de efectos básicos, pero aprovechando la mayor cantidad de herramientas posibles: los Tags, las Variables Dólar, las Variables de Punto y de la mano con los Algoritmos Matemáticos que nos ayudan a hacer creativos y novedosos Script de Efectos.

Ya estamos a portas de entrar en un nuevo nivel de Efectos y en la práctica de todas las Plantillas radica la habilidad de usar las herramientas explicadas y la memorización de los Algoritmos con sus respectivas aplicaciones y es por ello que una de las pretensiones del KaraBook es convertirse en la mano derecha de todos los que nos gusta crear los Script de nuestros propios Karaokes.

Y para despedir este Capítulo, la Plantilla 021 es un homenaje a los Algoritmos Matemáticos que hemos venido practicando y por ello se hace merecedora de ser la primera que lleva un nombre propio:

Plantilla 021 Amazing Movement



"**Amazing Movement**" es una de las Plantillas de Script más complejas creada de una única template line y tiene unos novedosos resultados. Sus variantes son principalmente en los colores, la fuente y el tamaño:

El Capítulo VII nos amplió las posibilidades a la hora de desarrollar los Efectos al mostrarnos más de las herramientas con las que podemos contar como las Variables Dólar y las Variables de Punto, sin dejas de lado la indispensables Matemáticas.

KaraBook Aegisub Libro Tutorial