# Kara Effector 3.2: Effector Book Vol. I [Tomo 5.1]

# Kara Effector 3.2:

Este **Tomo 5.1** es la expansión del **Tomo V**, ya que hubo un par de funciones de la Librería "**table**" del **Kara Effector** que no quedaron explicadas en él. También debo agregar que desde la fecha de publicación del **Tomo V**, hasta la actualidad (**04 de Junio de 2014**), la Librería se ha venido ampliando con otras funciones que también explicaremos en este **Tomo**.

Para garantizar que todos los ejemplos planteados en este **Tomo** les funcionen tal cual están, es recomendable usar el **Kara Effector 3.2.7** o superior, ya que la mayoría de las funciones nuevas fueron diseñadas a partir de dicha versión. Para las versiones anteriores simplemente les arrojará un error.

## Librería "table" [KE]:

**table.make( Objet, Size, limit_i, limit_f, ... ):** esta función crea una **tabla** de un tamaño determinado "**Size**" y que contiene elementos equidistantes entre sí, del tipo "**Objet**".

El parámetro **Objet** indica el tipo de elementos que tendrá la **tabla** retornada, y tiene cuatro opciones:

- "**number**"
- "**color**"
- "**alpha**"
- **Tags string**

Cada uno de ellos crea elementos distintos en nuestra **tabla** y más adelante veremos ejemplos de cada.

El parámetro **Size** indica el tamaño de la **tabla**, es decir, la cantidad de elementos que contendrá. **Size** debe ser un número entero mayor a cero.

Los parámetros **limit_i** y **limit_f** indican los límites inferior y superior que tendrán los elementos de la **tabla** creada. El primer elemento de la tabla sería **limit_i** y el último vendría a ser **limit_f**, y en medio de ellos todos los elementos equidistantes dependiendo del parámetro Size.

Los tres puntos seguidos ( **…** ) hacen referencia a uno o más tags que queramos añadirle a cada uno de los elementos que harán parte de la **tabla** retornada. Este parámetro es opcional.

- **Ejemplo 1. Objet** = "**number**":

mi_tabla = **table.make( "number", 8, 20, 55 )**

mi_tabla = {
[1] = 20, ← **limit_i**
[2] = 25,
[3] = 30,
[4] = 35,
[5] = 40,
[6] = 45,
[7] = 50,
[8] = 55 ← **limit_f**
}

Del anterior ejemplo vemos que la anterior **tabla** contiene 8 elementos (**#mi_tabla** = 8), y cada uno de ellos es un número equidistante entre 20 (**limit_i**) y 55 (**limit_f**).

- **Ejemplo 2. Añadir un tag**:

mi_tabla = **table.make( "number", 8, 20, 55, "\\fr" )**

mi_tabla = {
[1] = **\fr**20, ← **limit_i**
[2] = **\fr**25,
[3] = **\fr**30,
[4] = **\fr**35,
[5] = **\fr**40,
[6] = **\fr**45,
[7] = **\fr**50,
[8] = **\fr**55 ← **limit_f**
}

Ahora en este ejemplo, como en el quinto parametro colocamos "**\\fr**", entonces la función añade este tag al inicio de cada elemento de la **tabla**.

- **Ejemplo 3. Añadir más de un tag**:
  Para añadir dos o más tags a los elementos hay dos diferentes métodos. **Método 1**:

mi_tabla = **table.make( "number", 8, 20, 55, "\\fr","\\b" )**

mi_tabla = {
[1] = **\fr**20 **\b**20, ← **limit_i**
[2] = **\fr**25 **\b**25,
[3] = **\fr**30 **\b**30,
[4] = **\fr**35 **\b**35,
[5] = **\fr**40 **\b**40,
[6] = **\fr**45 **\b**45,
[7] = **\fr**50 **\b**50,
[8] = **\fr**55 **\b**55 ← **limit_f**
}

Cada elemento de la **tabla** se multiplica para corresponder a cada uno de los tags ingresados en el quinto parámetro de la función ("**\\fr**" y "**\\b**").

- **Ejemplo 4. Añadir más de un tag**:
  **Método 2**:

Tags = {"**\\frx**", "**\\fry**", "**\\frz**", "**\\fscy**"}
mi_tabla = **table.make( "number", 8, 20, 55, Tags )**

mi_tabla = {
[1] = **\frx**20 **\fry**20 **\frz**20 **\fscy**20,
[2] = **\frx**25 **\fry**25 **\frz**25 **\fscy**25,
[3] = **\frx**30 **\fry**30 **\frz**30 **\fscy**30,
[4] = **\frx**35 **\fry**35 **\frz**35 **\fscy**35,
[5] = **\frx**40 **\fry**40 **\frz**40 **\fscy**40,
[6] = **\frx**45 **\fry**45 **\frz**45 **\fscy**45,
[7] = **\frx**50 **\fry**50 **\frz**50 **\fscy**50,
[8] = **\frx**55 **\fry**55 **\frz**55 **\fscy**55
}

En resumen, cuando **Objet** = "**number**", los parámetros **limit_i** y **limit_f** deben ser números también, de modo que la función cree a cada uno de sus elementos de forma equidistante entre esos dos valores.

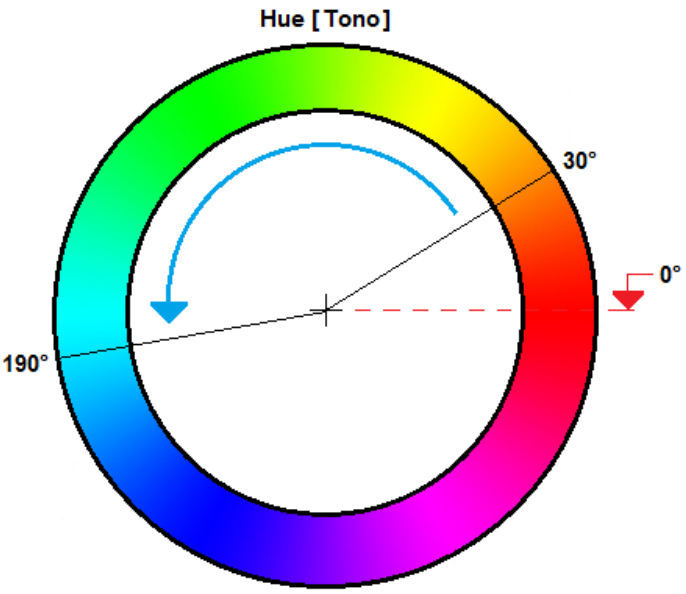
Para los siguientes ejemplos, usaremos la siguiente opción del parámetro **Objet**: "**color**"

- **Ejemplo 5. Objet** = "**color**":
  **Límites Numéricos:**

mi_tabla = **table.make( "color", 15, 30, 190 )**

- **Objet** = "**color**"
- **Size** = 15 (tamaño de la **tabla**)
- **limit_i** = 30 (límite inferior)
- **limit_f** = 190 (límite superior)

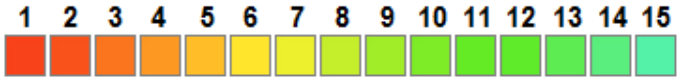
Los valores 30 y 190 hacen referencia a un ángulo entre 0° y 360° del Círculo Cromático de la **Teoría del Color HSV** (**Hue**, **Saturation**, **Value**):




Entonces la función creará una **tabla** de 15 colores entre los 30° y los 190°, equidistantes entre sí:

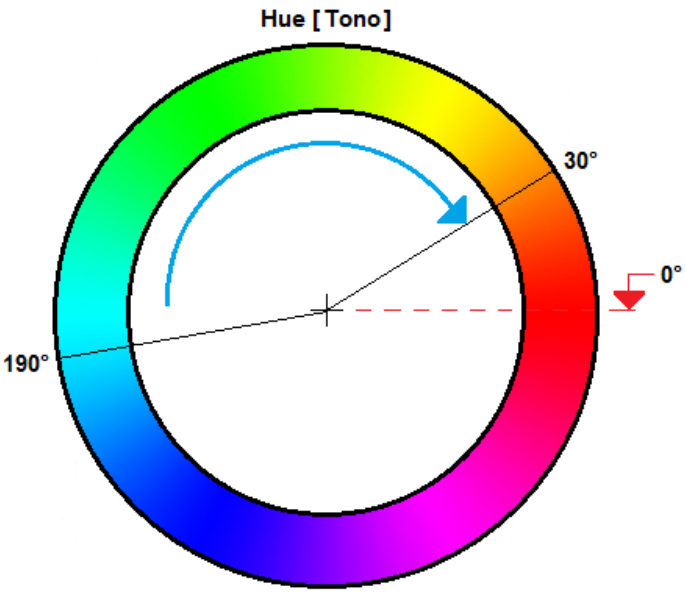
```
mi_tabla = {
                [1]  = "&H007FFF&",
                [2]  = "&H00B0FF&",
                [3]  = "&H00E0FF&",
                [4]  = "&H00FFEC&",
                [5]  = "&H00FFBC&",
                [6]  = "&H00FF8B&",
                [7]  = "&H00FF5B&",
                [8]  = "&H00FF2A&",
                [9]  = "&H06FF00&",
                [10] = "&H36FF00&",
                [11] = "&H67FF00&",
                [12] = "&H97FF00&",
                [13] = "&HC8FF00&",
                [14] = "&HF8FF00&",
                [15] = "&HFFD400&"
        }
```

Se entiende un poco más si se ven los tonos de los 15 colores de la **tabla** generada. Noten que equivalen a los colores entre 30° y 190°:




No necesariamente el valor de **limit_i** debe ser menor que el de **limit_f**. Ejemplo:

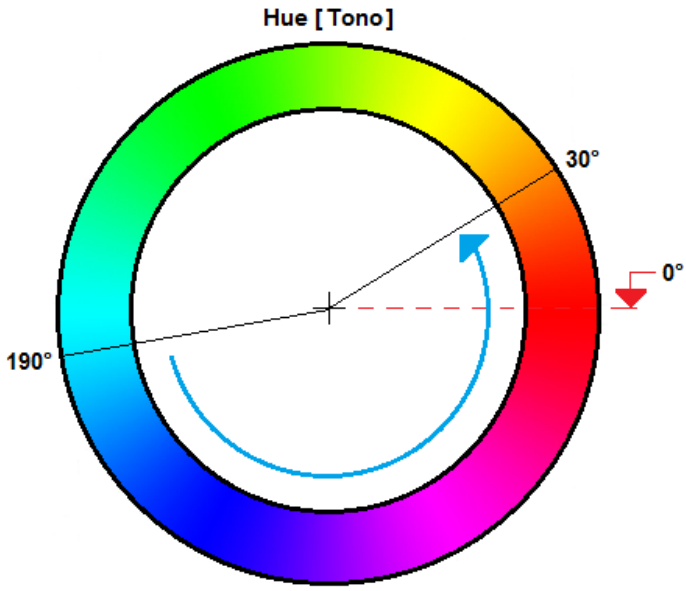
mi_tabla = **table.make( “color”, 20, 190, 30 )**




Entonces, lo que hace la función es crear una **tabla** de 20 colores desde los 190°, hacia atrás, hasta los 30°. La función hace el recorrido de las tonalidades en el Círculo Cromático en sentido horario.

Si alguno de los límites de la función excede a los 360°, dicho valor dará la vuelta en el Círculo, literalmente. Por cada giro se restan 360°. Ejemplo:

mi_tabla = **table.make( “color”, 10, 190, 390 )**




La función tomará a 10 colores desde los 190° hasta los 30° en sentido anti horario: 30° = 390° - 360°

Para agregar tags a los colores de la tabla, hacemos como en los primeros ejemplos:

mi_tabla = **table.make( “color”, 7, 60, 150, “\\1c” )**

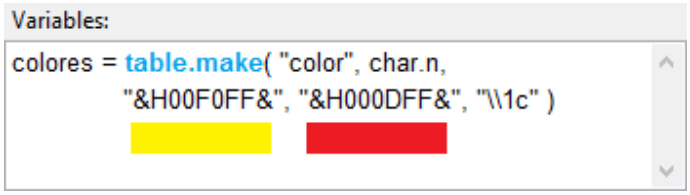
```
mi_tabla = {
                [1] = \1c&H00FFFF&,
                [2] = \1c&H00FFBF&,
                [3] = \1c&H00FF7F&,
                [4] = \1c&H00FF3F&,
                [5] = \1c&H00FF00&,
                [6] = \1c&H3FFF00&,
                [7] = \1c&H7FFF00&
        }
```

- **Ejemplo 6.** Cuando **limit_i** y **limit_f** son dos **strings** de colores:

tbl = **table.make(“color”,6, “&HFFFFFF&”, “&H000000&”)**

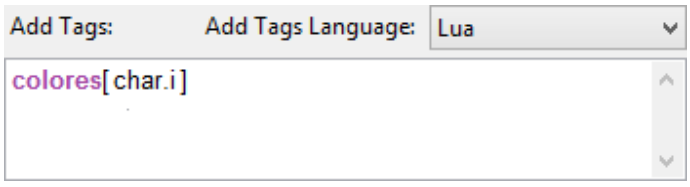
```
tbl = {
                [1] = "&HFFFFFF&",
                [2] = "&HCCCCCC&",
                [3] = "&H999999&",
                [4] = "&H666666&",
                [5] = "&H333333&",
                [6] = "&H000000&"
        }
```

Hagamos un ejemplo práctico en el **Kara Effector**, para el cuál usaremos un **Template Type: Char** y la plantilla **ABC Template leadin**:



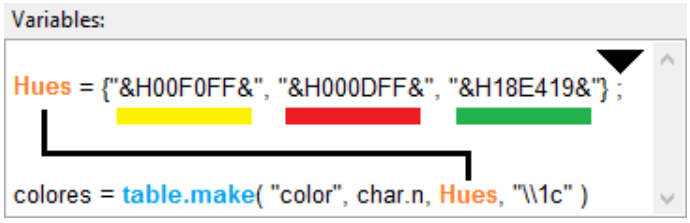

Como podemos ver, el tamaño de la **tabla** es **char.n**, que es el número total de caracteres (letras, números, signos) de la línea karaoke.

Para asignar los colores a cada carácter, hacemos:



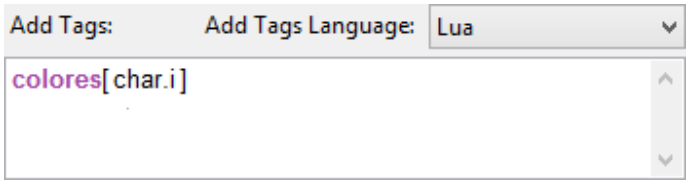

De esta manera, asignamos al primer carácter, el primer color; al segundo le asignamos el segundo color y así con todos los caracteres de la línea karaoke:


- **Ejemplo 7.** Cuando **limit_i** y **limit_f** se “fusionan” es un solo parámetro para ingresar más de dos colores. Este nuevo parámetro será ingresado en forma de **tabla**:

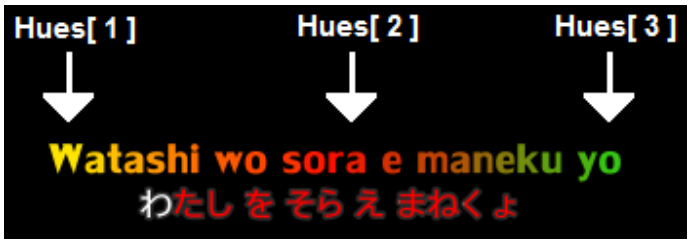



En “**Variables**” declaramos una **tabla** (**Hues**, en la imagen anterior) con las tonalidades de referencia (amarillo, rojo y verde, en este caso) para que la función **table.make** genere los colores equidistantes entre ellas. Nótese que resalto el punto y coma (;) con el cual se deben separa las variables.

Obtenemos los colores de la **tabla** “**colores**” de la misma forma que en el ejemplo anterior:




Y vemos los resultados:




Con el método usado en el anterior ejemplo, se pueden hacer **Gradientes** (degradaciones) de tres o más colores. Todo depende de los resultados deseados y del efecto que queremos hacer. Les recomiendo que practiquen con otros tres colores distintos y luego usando más colores.

- **Ejemplo 8. Objet** = "**alpha**" y límites numéricos. Los límites numéricos ya no están entre 0 y 360, como en el caso de **Objet** = "**color**"; en este caso los valores de los límites van desde **0** a **255**:

mi_tabla = **table.make( "alpha", 10, 45, 86 )**

```
mi_tabla = {
            [1]  = "&H2D&",  ← 45 en Hexadecimal
            [2]  = "&H31&",
            [3]  = "&H36&",
            [4]  = "&H3A&",
            [5]  = "&H3F&",
            [6]  = "&H43&",
            [7]  = "&H48&",
            [8]  = "&H4C&",
            [9]  = "&H51&",
            [10] = "&H56&"  ← 86 en Hexadecimal
    }
```

Para asignarle los tags, hacemos lo mismo que en los ejemplos de colores. Ejemplo:

mi_tabla = **table.make("alpha",8, 50, 200, "\\1a", "\\3a")**

```
mi_tabla = {
            [1] = "\\1a&H32& \\3a&H32&",
            [2] = "\\1a&H47& \\3a&H47&",
            [3] = "\\1a&H5C& \\3a&H5C&",
            [4] = "\\1a&H72& \\3a&H72&",
            [5] = "\\1a&H87& \\3a&H87&",
            [6] = "\\1a&H9D& \\3a&H9D&",
            [7] = "\\1a&HB2& \\3a&HB2&",
            [8] = "\\1a&HC8& \\3a&HC8&"
    }
```

- **Ejemplo 9**. Usando los límites como **strings alpha**:

mi_tabla = **table.make( "alpha", 6, "&H4E&", "&HAD&")**

```
mi_tabla = {
            [1] = "&H4E&",
            [2] = "&H61&",
            [3] = "&H74&",
            [4] = "&H87&",
            [5] = "&H9A&",
            [6] = "&HAD&"
    }
```

- **Ejemplo 10**. "fusionar" los parámetros **limit_i** y **limit_f** para poder ingresar una tabla con tres o más **strings alpha**:

**Alphas** = {"&HFF", "&HA2&", "&H16&", "&HDE&"};

mi_tabla = **table.make( "alpha", 12, Alphas )**

- **Ejemplo 11. Objet** = **Tags strings:**

mi_tabla = **table.make( "\\blur", 5 )**

```
mi_tabla = {
            [1] = "\\blur5",
            [2] = "\\blur5",
            [3] = "\\blur5",
            [4] = "\\blur5",
            [5] = "\\blur5"
    }
```

Es decir que el valor de Size (5 para el anterior ejemplo), no solo indica el tamaño de la **tabla**, sino que también se concatena (se une) con el tag ingresado ("**\\blur**").

- **Ejemplo 12:**

mi_tabla = **table.make( "\\fscx", 6, 80, 120 )**

```
mi_tabla = {
            [1] = "\\fscx80",
            [2] = "\\fscx88",
            [3] = "\\fscx96",
            [4] = "\\fscx104",
            [5] = "\\fscx112",
            [6] = "\\fscx120"
    }
```

Los valores numéricos equidistantes entre **limit_i** y **limit_f** (80 y 120) se concatenan al tag "**\\fscx**".

---

**table.rmake( Objet, Size, limit_i, limit_f, ... ):** esta función es similar a **table.make**, pero con la diferencia que los elementos de la **tabla** ya no están ni organizados ni equidistantes entre sí, sino que crea los elementos de forma totalmente aleatoria, teniendo en cuenta los limites inferior y superior de los parámetros **limit_i** y **limit_f**.

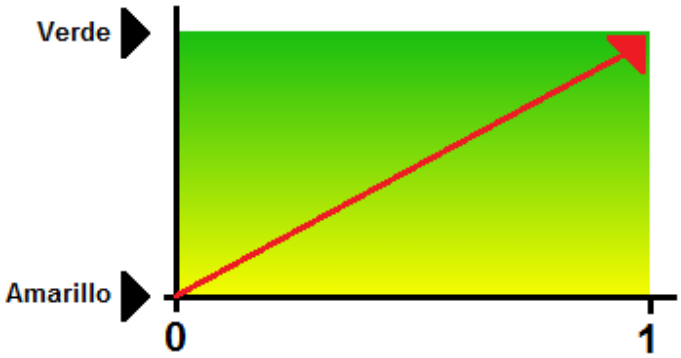
**table.gradient( color1, color2, algorithm ):** crea una tabla de colores (o de alphas), correspondientes a un Gradiente entre los parámetros **color1** y **color2**. El tamaño de la **tabla** generada dependerá del **Template Type**, de tal manera que a cada objeto karaoke le corresponda un único elemento. Si por ejemplo el **Template Type** es **Translation Word**, entonces el tamaño de la **tabla** que se generará será **word.n**, de modo que por cada palabra de la línea karaoke, haya un elemento en la **tabla** que le corresponde.

El parámetro **algorithm** es opcional y determina el modo de transición desde **color1** hasta **color2**.

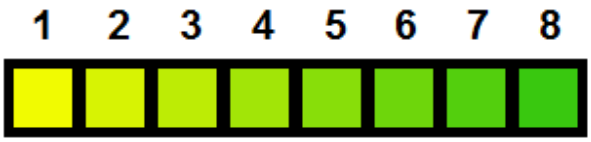
- **Ejemplo 1. Template Type: Word**
  **word.n** = 8 (suponiendo que una línea tiene 8 palabras. Este número es solo para saber cómo se determina el tamaño de la **tabla**)

mi_tabla = **table.gradient( "&H00FFFF&", "&H12BE12&" )**

Como hemos omitido al tercer parámetro de la función (**algorithm**), entonces la transición entre el amarillo y el verde se hará de forma lineal:




Los 8 elementos de la tabla generada serán:

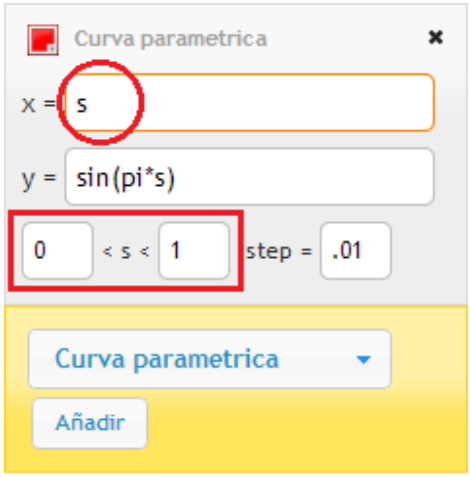



De este modo, la degradación se hace normalmente, ya que omitimos un algoritmo que modifique la transición entre los dos colores ingresados.

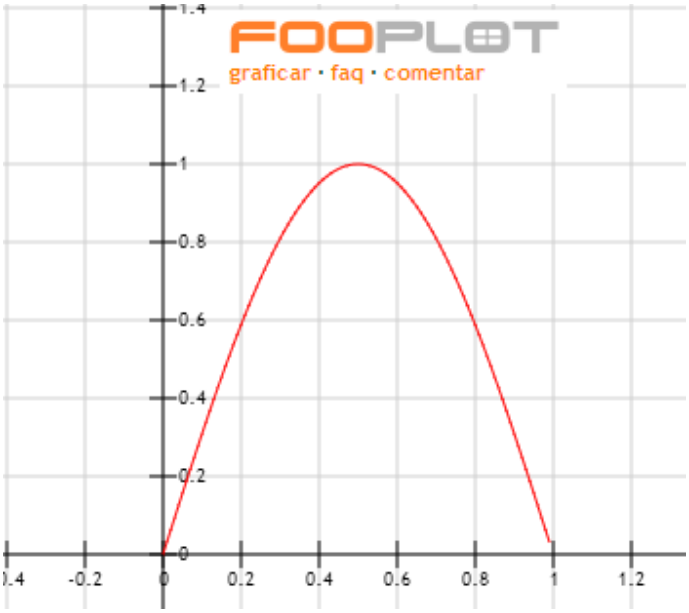
- **Ejemplo 2. Template Type: Syl**
  **syl.n** = 12

Uno de los métodos simples para crear un **algoritmo** para esta función, es usar un graficador de funciones online; el que generalmente usamos es "**fooplot**" en la opción de **Curva Paramétrica**. Los pasos son:

- En [ **x =** ] ponemos la letra (s)
- El dominio se pone desde 0 a 1



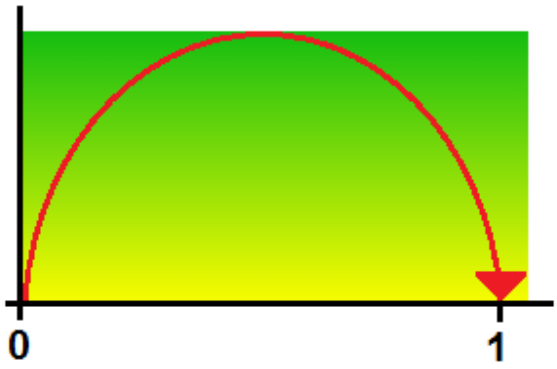

- En [ **y =** ] declaramos el algoritmo en función de la letra (s), como en la imagen anterior, y en la gráfica veremos esto:




Hecho esto, copiamos el algoritmo hecho en [ y ] y lo pegamos en el tercer parámetro de la función, entre comillas simples o dobles, y añadiendo el signo de porcentaje (%) antes de cada letra (s) que haya en el algoritmo:

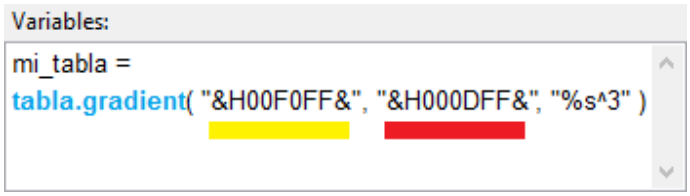
**sin( pi*s ) => "sin( pi*%s )"**

mi_tabla = **table.gradient( “&H00FFFF&”, “&H12BE12&”, “sin( pi*%s )” )**

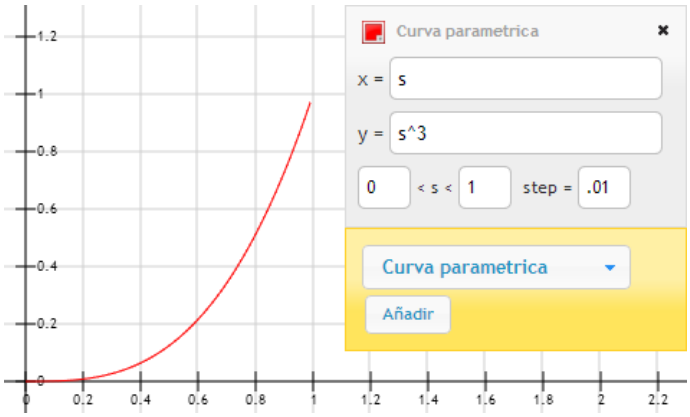



Entonces se hará un Gradiente desde el amarillo hasta el verde y luego regresará al amarillo:

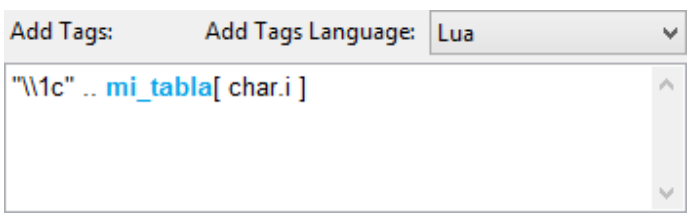
- **Ejemplo 3:**




Vemos la gráfica de ( s^3 ) desde 0 a 1:



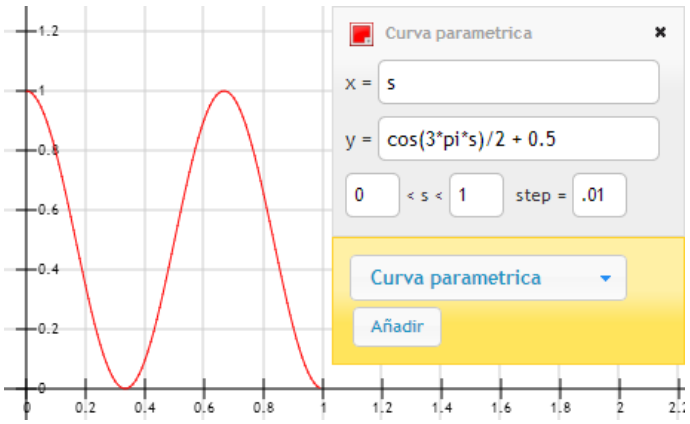

Con un **Template Type: Char**, llamamos a los colores que la **tabla** creó de la siguiente manera:




El Gradiente creado por el algoritmo “**%s^3**” se verá en el vídeo más o menos como en la siguiente gráfica:


Notamos que el rojo empieza casi al final de la línea y ya no es simétrico como en el **Gradiente lineal**.

- **Ejemplo 3:**
  Algoritmo = “**cos( 3*pi*%s )/2 + 0.5**”





Del algoritmo dependerá la transición entre el color 1 y el color2, así que las posibilidades son infinitas, hay tantas transiciones como algoritmos puedan crear.

Para hacer Gradientes de alphas con esta función, solo se deben colocar los **strings alphas** en ambos parámetros de la misma, ejemplo:

mi_tabla = **table.gradient( “&HFF&”, “&HD8&” )**

---

**table.gradient2( ... ):** esta función crea una **tabla** con los colores (o alphas) de un **Gradiente Lineal** entre todos los elementos ingresados (dos o más) en la función.

Al igual que la anterior función, el tamaño de la **tabla** es dependiente del **Template Type**.

- **Ejemplo 1:**

mi_tabla = **table.gradient2(“&H00FFFF&”, “&H12BE12&”)**

Entonces la función crea un **Gradiente Lineal** entre los dos colores ingresados.

- **Ejemplo 2:**

alphas = { “&H00&”, “&HAA”, “&H5D&”, “&HFF&” };

mi_tabla = **table.gradient2( alphas )**

En resumen, los tres puntos seguidos (…) en la función, hacen referencia a los colores o alphas a ingresar, o a una **tabla** de colores o de alphas, de la cual necesitemos crear una **tabla** del Gradiente Lineal generados por ellos.

Si le ingresamos 3, 4, 7, 10 o la cantidad de colores que deseemos, la función hará una **tabla**, en donde el tamaño dependerá del **Template Type**, con el **Gradiente Lineal** de todos los colores ingresados. Aplica de la misma manera para los alphas.

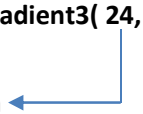
---

**table.gradient3( Size_table, ... ):** esta función hace exactamente los mismo que la función anterior, pero el tamaño de la **tabla** generada ya no dependerá del tipo de plantilla (**Template Type**), sino que dependerá del parámetro **Size_table**.

- **Ejemplo 1:**

colores = {“&H00FF00&”, “&HFFAA00&”, “&H00DDFF&”};

mi_tabla = **table.gradient3( 24, colores )**



Tamaño de la **tabla**

Recordemos que este **Tomo 5.1** es una extensión del **Tomo V**, en el que vienen explicadas cinco funciones de la librería **table [KE]**. 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. 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/NatsuoDCE
- www.youtube.com/user/karalaura2012