

---

---

# Kara Effector 3.2:

En este **Tomo V** nos adentramos más en el mundo de las Librerías, tanto de **LUA** como las del **Kara Effector**. Algunas funciones solo serán mencionadas y otras más tendrán ejemplos para una mayor comprensión, de todas maneras aquellas que no queden claras, más adelante las veremos con mayor detenimiento.

No es importante memorizarlas todas, pero ayuda el hecho de que tengamos una idea de qué hace cada función y para qué nos podría servir una variable en especial.

## Librería "math" [LUA]:

Es la librería de las funciones matemáticas de **LUA** y es de mucha utilidad. A continuación veremos las funciones y valores más usados, al menos para hacer Efectos. Omití algunas funciones y valores que consideré que no serían relevantes para usar en el **Kara Effector**, pero de todas maneras son fáciles de conseguir en la web.

<b>math.abs(x)</b>	Retorna el Valor Absoluto de x
<b>math.acos(x)</b>	Retorna el Arco Coseno de x en un ángulo medido en radianes
<b>math.asin(x)</b>	Retorna el Arco Seno de x en un ángulo medido en radianes
<b>math.atan(x)</b>	Retorna el Arco Tangente de x en un ángulo medido en radianes
<b>math.atan2(y, x)</b>	Retorna el Arco Tangente de y/x en un ángulo medido en radianes
<b>math.ceil(x)</b>	Retorna el número entero mayor más cercano a x
<b>math.cos(x)</b>	Retorna el Coseno del ángulo x medido en radianes
<b>math.cosh(x)</b>	Retorna el Coseno Hiperbólico de x
<b>math.deg(x)</b>	Retorna el valor de x, convertido de radianes a sexagesimal. Ej: <b>math.deg(math.pi) = 180</b>

---

---

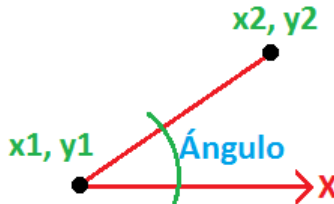
<b>math.exp(x)</b>	Retorna el valor de $e^x$ . $e = 2.718281...$
<b>math.floor(x)</b>	Retorna el número Entero Menor más cercano a x
<b>math.mod(x, y)</b>	Retorna el Residuo de x/y, o sea que retorna el Modulo entre x e y
<b>math.log(x)</b>	Retorna el Logaritmo Natural (base e) de x
<b>math.log10(x)</b>	Retorna el Logaritmo Decimal (base 10) de x
<b>math.max(x, ...)</b>	Retorna el número mayor de entre todos los parámetros
<b>math.min(x, ...)</b>	Retorna el número menor de entre todos los parámetros
<b>math.pi</b>	Retorna el valor de pi. $\pi = 3.14159...$
<b>math.pow(x, y)</b>	Retorna el valor de $x^y$ . o sea x elevado a la y
<b>math.rad(x)</b>	Retorna el valor de x, convertido de sexagesimal a radianes
<b>math.random(m, n)</b>	Retorna un Número Aleatorio. En el caso de haber dos valores (m, n), retorna un número aleatorio entre esos dos valores. En el caso de un valor (m), retorna un número aleatorio entre 1 y ese valor. Y para el caso de no tener ningún valor, retorna un número decimal aleatorio entre 0 y 1
<b>math.sin(x)</b>	Retorna el Seno del ángulo x medido en radianes
<b>math.sinh(x)</b>	Retorna el Seno Hiperbólico de x
<b>math.sqrt(x)</b>	Retorna la Raíz Cuadrada de x. se puede remplazar con $x^{0.5}$
<b>math.tan(x)</b>	Retorna la Tangente del ángulo x medido en radianes
<b>math.tanh(x)</b>	Retorna la Tangente Hiperbólica de x

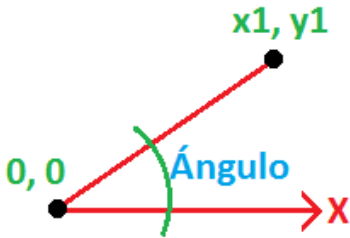
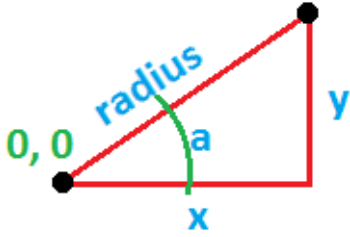
## Librería "math" [KE]:

La anterior Librería es la que ya viene por default el lenguaje **LUA**, pero para hacer alguno de los Efectos parece que ésta se queda corta, es por ello que hubo la

necesidad de "ampliar" la Librería "**math**". Las siguientes funciones y variables hacen parte de la Librería "**math**", pero son solo de uso exclusivo del **Kara Effector**.

También he omitido algunas funciones de la ampliación de la Librería "**math**" del **Kara Effector**, pero es porque no son para usar en los Efectos, sino que son para ayudar a otras funciones a hacer su trabajo.

<b>math.R(m, n)</b>	Cumple prácticamente la misma función que <b>math.random</b> , pero con la diferencia que genera un número aleatorio por cada vez que se aplica el Efecto. La forma abreviada es: <b>R(m, n)</b>
<b>math.Rfake(m, n, i)</b>	Parecida a <b>math.random</b> , pero el random se genera una única vez para todos los Efectos que se use, además consta de un tercer parámetro opcional, que hace que el random haga más operaciones antes de retornar un resultado. La forma abreviada es: <b>Rf(m,n,i)</b>
<b>math.round(n, dec)</b>	Redondea un número n según las cantidad de cifras decimales que indiquemos (dec). Si no está el parámetro "dec" entonces retorna el entero más cercano
<b>math.distance(x1, y1, x2, y2)</b>	Retorna la Distancia entre dos puntos $P1=(x1, y1)$ y $P2=(x2, y2)$ . También se puede usar así: <b>math.distance(x1, y1)</b> En este caso retorna la Distancia entre $P1=(0, 0)$ y $P2(x1, y1)$
<b>math.angle(x1, y1, x2, y2)</b>	Retorna el Ángulo que forma el segmento formado por los dos puntos, y el tramo positivo del eje "x":  También se puede usar así: <b>math.angle(x1, y1)</b> En este caso retorna el Ángulo que forma el segmento formado

	<p>por los puntos <math>P1 = (0, 0)</math> y <math>P=(x1, y1)</math>, y el tramo positivo del eje "x":</p> 
<b>math.polar</b> (a, radius, Return)	<p>Retorna las coordenadas Polares que forma el ángulo (a) y el radio (radius). El parámetro <b>Return</b> puede ser "x" o "y" según la coordenada que queremos que retorne. Si <b>Return</b> no está, retorna ambas coordenadas.</p> 
<b>math.point</b> (n, x, y, xi, yi, xf, yf)	<p>Retorna un conjunto de Puntos.  <b>n</b>: es el tamaño del conjunto  <b>x</b>: es el rango horizontal  <b>y</b>: es el rango vertical  <b>xi, yi</b>: es el primer punto  <b>xf, yf</b>: es el último punto</p>
<b>math.factorial(n)</b>	<p>Retorna <b>n!</b>, es decir que retorna el factorial de <b>n</b></p>
<b>math.bezier</b> (Return, ...)	<p>Retorna una <b>Curva Bezier</b> a partir de una serie de puntos o el trazado de una Shape. Es una función exclusiva para Efectos con Shapes.</p>

Para ver la siguiente Librería es importante que antes tengamos claro el concepto de "tabla". Una **tabla** es un conjunto de elementos, también se le conoce como "arreglo".

El nombre de una **tabla** es asignado por uno mismo y sus elementos están dentro de llaves ("{" }). Veamos un corto ejemplo:

```
Letras = {"A", "B", "C", "D"}
```

En este ejemplo, la **tabla** se llama "Letras" y tiene cuatro elementos. Los elementos de una **tabla** están separados por coma (,) o por punto y coma (;), no importa cuál de las dos se use.

Para usar los elementos de la **tabla** "Letras", se hace con el nombre de la **tabla** seguido del número de la posición del elemento entre corchetes:

```
Letras = {"A", "B", "C", "D"}
```

```
Letras[1] = "A"
```

```
Letras[2] = "B"
```

```
Letras[3] = "C"
```

```
Letras[4] = "D"
```

Otra forma de definir la tabla de este ejemplo y aun así tener el mismo resultado sería:

```
Letras = {[1] = "A", [2] = "B", [3] = "C", [4] = "D"}
```

Y por si fuera poco, todavía hay una tercera forma de definir la **tabla** "Letras". Se define vacía o con algunos de sus elementos y luego se le "insertan" el resto:

```
Letras = { }
```

```
Letras[1] = "A"
```

```
Letras[2] = "B"
```

```
Letras[3] = "C"
```

```
Letras[4] = "D"
```

```
Letras = {"A", "B", "C", "D"}
```

El número de la posición que ocupa un elemento dentro de una **tabla**, se le llama **índice**:

**Letras[2] = "B"**  


  
**tabla**      **índice**      **elemento**

El tamaño de una **tabla** es la cantidad de elementos que tenga la misma. Para obtener el tamaño de una **tabla** se escribe el signo número seguido del nombre de la **tabla**:

---

---

**#Letras = 4**

Veamos un ejemplo en el **Kara Effector** con este tipo de **tabla**:

**Paso 1.** Definimos una **tabla** en la celda de texto Variables:

Variables: `colores = {"&H00FF20&", "&HFD440E&", "&H9C03FE&"}`

Le di por nombre "colores" y contiene tres colores: verde, azul y lila. De esta **tabla** se infiere que su tamaño es:

**#colores = 3**

**Paso 2.** Usar los elementos de la **tabla** en Add Tags:

Add Tags: `\1c!colores[math.random(#colores)]!` Add Tags Language: Automation Auto-4

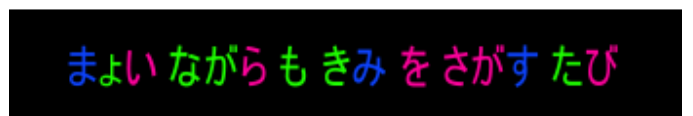
Para este ejemplo, usé los elementos de la **tabla** con un random. Una pequeña explicación de la expresión usada sería:

- `\1c!colores[math.random(#colores)]!`
- `#colores = 3`
- `\1c!colores[math.random(3)]!`
- `\1c!colores[math.random(1,3)]!`

Como es un random, los posibles resultados son:

- `\1c!colores[1]!` ← verde
- `\1c!colores[2]!` ← azul
- `\1c!colores[3]!` ← lila

Y en el vídeo veríamos algo más o menos como esto:



Este es solo un pequeño ejemplo de cómo usar los tres elementos de esta **tabla**, pero hay muchas formas más y en futuros ejemplos veremos esas formas de hacerlo. Si al momento de hacer este, no se ve en el vídeo como se ve en la imagen anterior, les recomiendo volver a descargar el **Kara Effector** en el **Blog Oficial**.

---

---

Los elementos de la **tabla** "colores" del ejemplo anterior también pueden ser declarados por medio de un nombre, lo que me da pie para mostrarles otra forma de declarar una **tabla**:

```
colores = {  
  
    verde = "&H00FF20&",  
  
    azul = "&HFD440D&",  
  
    lila = "&H9C03FE&"  
  
}
```

Y la forma de "llamar" a los elementos declarados de esta forma es:

- `colores.verde`
- `colores.azul`
- `colores.lila`

O sea, el nombre de la **tabla** seguido de un punto (.) y luego el nombre que le hayamos dado a cada elemento.

El nombre que se le da a cada elemento de una **tabla**, en este caso la **tabla** "colores", también es asignado a gusto propio. Lo pude haber hecho de esta otra forma y sería lo mismo:

```
colores = {  
  
    v = "&H00FF20&",  
  
    a = "&HFD440D&",  
  
    l = "&H9C03FE&"  
  
}
```

Veamos un ejemplo de cómo usar los valores de esta **tabla**: (en lenguaje **Automation Auto-4**)

`\3c!colores.v!`

O sea que el color del **Borde** (\3c) sería verde. Este mismo ejemplo pero en lenguaje **LUA** sería: (recordemos que hay dos formas de hacerlo)

- `string.format("\\3c%s", colores.v)`
  - `"\\3c"..colores.v`
- 
-

Lo anterior es solo una pequeña introducción al mundo de las **tablas**, pero el concepto es mucho más amplio de lo que hasta ahora hemos visto y de apoco les mostraré lo que aun reste por ver.

Ya con el concepto de **tabla** un poco más claro, veremos la siguiente Librería:

## Librería "table" [LUA]:

Es la librería de las funciones de LUA que hacen referencia a las **tablas**. Esta Librería también tiene una ampliación en el **Kara Effector** y por ahora solo explicaré las funciones que son útiles para hacer Efectos:

<b>table.concat</b> (t, separador)	Retorna los elementos de una tabla, pero concatenados. Si el parámetro " <b>separador</b> " no está, simplemente concatena a los elementos. Ejemplo: T = {"a", 7, "FF"} <b>table.concat</b> (T) = "a7FF" Ejemplo con " <b>separador</b> ": T = {"a", 7, "FF"} <b>table.concat</b> (T, " +-+ ") = "a +-+ 7 +-+ FF"
<b>table.insert</b> (t, i, elemento)	Inserta un elemento asignado a una tabla. Si el parámetro " <b>i</b> " no está, inserta al elemento al final de la tabla. Ejemplo: A = {2, 4, 6, 8, 10} <b>table.insert</b> (A, "&HFF&") Entonces: A = {2, 4, 6, 8, 10, "&HFF&"} Si para este mismo ejemplo ponemos el parámetro " <b>i</b> ", sería: <b>table.insert</b> (A, 3, "&HFF&") Entonces: A = {2, 4, "&HFF&", 6, 8, 10} Nótese cómo el elemento se insertó en la posición 3.
<b>table.maxn</b> (t)	Cumple la misma función que <b>#t</b> , o sea que retorna el tamaño de la tabla <b>t</b> .
<b>table.remove</b> (t, i)	Remueve el elemento de la posición " <b>i</b> " de la tabla " <b>t</b> ". Ej: B = {1, 3, 5, 7} <b>table.remove</b> (B, 2) Entonces: B = {1, 5, 7}

<b>table.sort</b> (t)	Organiza de menor a mayor los elementos de una tabla, sí y solo sí todos los elementos son números. Ejemplo: D = {5, 4, 7, 9, 2, 1} <b>table.sort</b> (D) Entonces: D = {1, 2, 4, 5, 7, 8} Los elementos de la tabla D ahora están organizados de menor a mayor.
-----------------------	---

Es momento de mencionar que las funciones de la Librería "**table**" se usan para crear nuevas funciones. Crear una nueva función es una herramienta para hacer Efectos que el **Aegisub**, el lenguaje **LUA** y **Kara Effector** nos ofrecen. Hacer funciones será la parte final de este curso, ya que cuando cada uno pueda hacer sus propias funciones, le resta muy poco por aprender acerca de Efectos Karaoke, por eso es importante ir viendo todos estos conceptos, para que cuando llegue el momento de usarlos a pleno, no quedaremos tan perdidos.

## Librería "table" [KE]:

Es la ampliación de la Librería "**table**" con que cuenta el **Kara Effector** y algunas de sus funciones nos sirven para hacer Efectos y las restantes para crear nuevas funciones.

<b>table.inside</b> (t, e)	Retorna <b>true</b> (verdadero) en el caso que el elemento " <b>e</b> " esté en la tabla " <b>t</b> ". En caso contrario retorna <b>false</b> (falso). Ejemplo: P = {"a", "b", "c"} <b>table.inside</b> (P, "c") = <b>true</b> <b>table.inside</b> (P, "e") = <b>false</b>
<b>table.index</b> (t, e)	Retorna el <b>índice</b> del elemento " <b>e</b> " en el caso de que dicho elemento pertenezca a la tabla " <b>t</b> ". En caso contrario retorna de nuevo al elemento. Ejemplo: R = {13, 42, 63, 34, 25} <b>table.index</b> (R, 42) = 2 Retorna 2, porque R[2] = 42 <b>table.index</b> (R, "AA") = "AA" Retorna "AA", porque: <b>table.inside</b> (R, "AA") = <b>false</b> O sea que "AA" no está en R

<b>table.compare(t1, t2)</b>	Retorna <b>true</b> (verdadero) en el caso de que la tabla <b>t1</b> sea igual que la tabla <b>t2</b> , de otro modo retorna <b>false</b> (falso). Ejemplo: A = {1, 2, 3} B = {1, 2, 3, 4} <b>table.compare(A, B) = false</b> C = {7, "a"} D = {7, "a"} <b>table.compare(C, D) = true</b>
<b>table.disorder(t)</b>	<b>t</b> puede ser una tabla o un número entero positivo mayor o igual que 2. Para el caso en que <b>t</b> sea una tabla, retorna a la misma tabla con sus elementos en desorden. Los elementos se desordenan de forma aleatoria (random). Ej: H = {"a", "b", "c", 3, 9} G = <b>table.disorder(H)</b> Un posible resultado sería: G = {"b", 9, "a", 3, "c"} Si <b>t</b> es un entero positivo, entonces la función crea una tabla de números consecutivos desde 1 hasta <b>t</b> , para luego desordenar esos números. Ej: G = <b>table.disorder(6)</b> Un posible resultado sería: G = {3, 5, 6, 2, 1, 4} El número total de resultados es <b>#t!</b> , o sea el factorial del tamaño de la tabla. Para este último ejemplo sería: 6! = 720 posibilidades

He decidido hacer un pequeño paréntesis acá para aclarar algunos conceptos que nos ayudarán a entender mucho de lo que se viene. El lenguaje **Automation Auto-4** es una modificación del lenguaje **LUA** para que reconozca las variables **Dólar (\$)** y las operaciones hechas dentro de los signos de admiración (!!). En pocas palabras, el lenguaje **Automation Auto-4** está basado en el lenguaje **LUA**.

Otro de los conceptos que quería mencionar es uno que en ocasiones nos encontramos en los parámetros que requiere una función, y son tres puntos seguidos (...). Los tres puntos seguidos pueden ser una variable, una tabla, un elemento o un listado de elementos y/o una combinación de éstos. Estos tres puntos se colocan para indicar que podemos poner cuantas cosas queramos.

<b>table.concat1(t, ...)</b>	Retorna la tabla <b>t</b> con todos sus Elementos concatenados a (...). Ejemplo 1: (tabla y un elemento) A = {"a", "b", "c"} B = <b>table.concat1(A, 9)</b> B = {"9a", "9b", "9c"} Ejemplo 2: (tabla y más de un elemento) F = {1, 2, 3} G = <b>table.concat1(F, a, b)</b> G = {a1, b1, a2, b2, c1, c2} Ejemplo 3: (tabla con tabla) M = {4, 6, 8} N = {f, g} O = <b>table.concat1(M, N)</b> O = {f4, g4, f6, g6, f8, g8}  En estos ejemplos vemos cómo los tres puntos pueden ser un solo elemento o varios, también pueden ser una tabla
<b>table.concat2(t, ...)</b>	Es similar a <b>table.concat1</b> y la diferencia se notará en los ejemplos. Ejemplo 1: (tabla y un elemento) A = {"a", "b", "c"} B = <b>table.concat1(A, 9)</b> B = {"9a", "9b", "9c"} Ejemplo 2: (tabla y más de un elemento) F = {1, 2, 3} G = <b>table.concat1(F, a, b)</b> G = {a1b1, a2b2, c1c2} Ejemplo 3: (tabla con tabla) M = {4, 6, 8} N = {f, g, h} O = <b>table.concat1(M, N)</b> O = {f4g4h4, f6g6h6, f8g8h8}
<b>table.replay(n, ...)</b>	Retorna una tabla con <b>n</b> veces repetidas a (...). Ejemplo 1: A = <b>table.replay(4, "a")</b> A = {"a", "a", "a", "a"} Ejemplo 2: B = <b>table.replay(3, f, g, h)</b> B = {f, g, h, f, g, h, f, g, h} Ejemplo 3: C = {7, 8, 9} D = <b>table.replay(2, C)</b> D = {7, 8, 9, 7, 8, 9} Se nota cómo se repiten <b>n</b> veces el o los elementos ingresados

<b>table.count(t, e)</b>	Retorna el número de veces en el que el elemento <b>e</b> aparece en la tabla <b>t</b> . en el caso en que el elemento <b>e</b> no esté en <b>t</b> , retorna 0. Ejemplo: A = {a, b, a, 7, a, 8, 9, a} <b>table.count(A, a) = 4</b> <b>table.count(A, c) = 0</b> <b>table.count(A, b) = 1</b>
<b>table.pos(t, e)</b>	Retorna una tabla con el o los <b>índices</b> del elemento <b>e</b> en la tabla <b>t</b> . En el caso de que el elemento <b>e</b> no esté en <b>t</b> , retorna una tabla vacía. Ejemplo: A = {a, b, a, 7, a, 8, 9, a} B = <b>table.count(A, a)</b> B = {1, 3, 5, 8} C = <b>table.count(A, c)</b> C = { } D = <b>table.count(A, b)</b> D = {2}
<b>table.retire(t, ...)</b>	Retorna la tabla <b>t</b> , pero retira a (...) de la tabla en el caso en que están en ella. Ejemplo 1: A = {a, b, a, 7, a, 8, 9, a} B = <b>table.retire(A, a)</b> B = {b, 7, 8, 9} Ejemplo 2: C = <b>table.retire(A, a, 7, 8)</b> C = {b, 9} Ejemplo 3: D = {7, 8, 9} E = <b>table.retire(A, D)</b> E = {a, b, a, a, a}
<b>table.inserttable(t1, t2, i)</b>	Inserta los elementos de la tabla <b>t2</b> en la tabla <b>t1</b> a partir del <b>índice i</b> , o en el caso de no estar el parámetro <b>i</b> , se insertan al final de la tabla <b>t1</b> . Ejemplo 1: A = {6, 7, 8} B = {a, b, c, d} C = <b>table.inserttable(A, B, 3)</b> C = {6, 7, a, b, c, d, 9} Los elementos de la tabla B se insertaron en la tabla A, a partir del <b>índice 3</b> (la tercera posición). Ejemplo 2: D = {5, 4, 1}      E = {f, g} F = <b>table.inserttable(D, E)</b> F = {5, 4, 1, f, g}

<b>table.reverse(t)</b>	Retorna a la tabla <b>t</b> , pero con el <b>índice</b> invertido. Ejemplo 1: A = {3, 4, 5, 6, 7} B = <b>table.reverse(A)</b> B = {7, 6, 5, 4, 3} Ejemplo 2: C = {f, 5, 3, a, m, 2, 9} D = <b>table.reverse(C)</b> D = {9, 2, m, a, 3, 5, f}
<b>table.cyclic(t)</b>	Genera un " <b>ciclo</b> " con todos los elementos de la tabla <b>t</b> . Ejemplo 1: A = {2, 4, 6, 8} B = <b>table.cyclic(A)</b> B = {2, 4, 6, 8, 6, 4, 2} Ejemplo 2: C = {a, 4, 2, d, 5, b} D = <b>table.cyclic(C)</b> D = {a, 4, 2, d, 5, b, 5, d, 2, 4, a}
<b>table.op(t, mode)</b>	Genera operaciones con los elementos de la tabla <b>t</b> . Dichas operaciones dependen del modo " <b>mode</b> ". Esta función es exclusiva para las tablas con elementos numéricos. Ejemplo 1: <b>mode</b> = "sum" Suma los elementos de la tabla A = {9, 1, 16, 4} <b>table.op(A, "sum")</b> = 9 + 1 + 16 + 4 = 30 Ejemplo 2: <b>mode</b> = "concat" Une a los elementos de la tabla <b>table.op(A, "concat")</b> = 14916 Ejemplo 3: <b>mode</b> = "average" Obtiene un promedio de la tabla <b>table.op(A, "average")</b> = (9 + 1 + 16 + 4) / #A = 30 / 4 = 7.5 Ejemplo 4: <b>mode</b> = "min" Da al menor de los elementos <b>table.op(A, "min")</b> = 1 Ejemplo 5: <b>mode</b> = "max" Da al mayor de los elementos <b>table.op(A, "max")</b> = 16 Ejemplo 6: <b>mode</b> = "add" Adiciona un tercer parámetro a cada uno de los elementos B = <b>table.op(A, "add" -2)</b> B = {9 - 2, 1 - 2, 16 - 2, 4 - 2} B = {7, -1, 14, 2}



---

---

Omití algunas funciones de la ampliación de la Librería “**table**” por un par de motivos: para ver algunos conceptos previos y para poder darle más espacio y mayor número de ejemplos para total comprensión.

Este ha sido un **Tomo** cargado con mucha teoría, teoría que no necesariamente deben memorizar ni dominar toda al 100%, pero es mejor tener a la mano el medio para consultar alguna duda, que necesitar un concepto y no saber en dónde buscar.

Lo que con certeza les puedo asegurar es que el tener claro cuáles son y para qué sirve cada una de las variables y funciones de las Librerías vistas en esta series de tomos, les dará las herramientas necesarias para crear sus propias funciones, en donde las posibilidades son infinitas y los Efectos que se pueden lograr con cada una de ellas no tienen comparación.

En el **Kara Effector** la teoría es tan importante como la práctica, ambas deben ir de la mano, ya que sin una de ellas la otra no sería suficiente. De a poco iremos haciendo un equilibrio entre ellas y por eso en los próximos tomos irá aumentando el número de ejemplos para ponerlos en práctica y aumentar nuestras destrezas.

---

Espero que en este tramo del camino ya hayan podido ver algunos de los Efectos que por default vienen en el **Kara Effector** y hayan podido entender un poco mejor en qué consiste cada uno de ellos, y eso gracias a los conceptos, variables y funciones vistas. No olviden 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.

---

---