Kara Effector 3.2:

En el **Tomo XVII** seguiremos con los Ejemplos del Modo "**radial**" de la función **shape.array** de la Librería shape, y con el inicio del último Modo de dicha función.

Librería Shape [KE]:

El **tomo XVI** había terminado con el Ejemplo 7 del Modo "**radial**", en donde **mi_shape** es una **tabla** en vez de una simple **shape**:

Variables:

```
mi_shape = {"m 12 0 l 0 15 l 12 30 l 45 15 l 27 7 l 24 11 l
17 11 l 22 5 l 12 0 ", shape.size(shape.circle, 24) }
```

shape.array(shape, loops, A_mode, Dxy):

Modo Radial

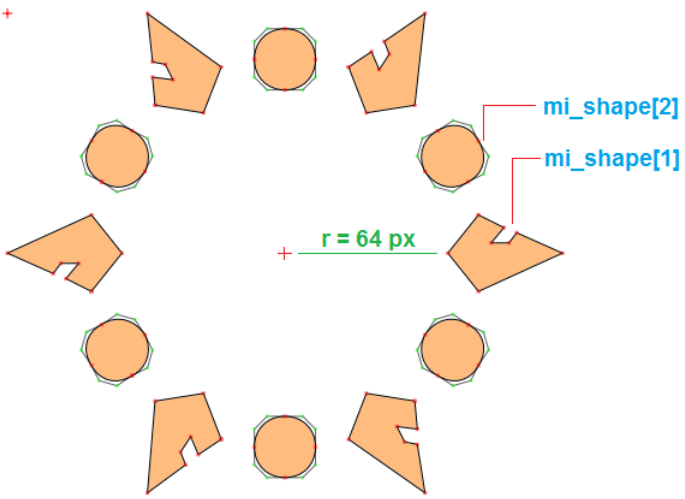
- **Ejemplo 8.** Ingresamos la **tabla mi_shape**, el **loops** vuelve a ser un valor numérico, que para este ejemplo es 6:

Return [fx]:

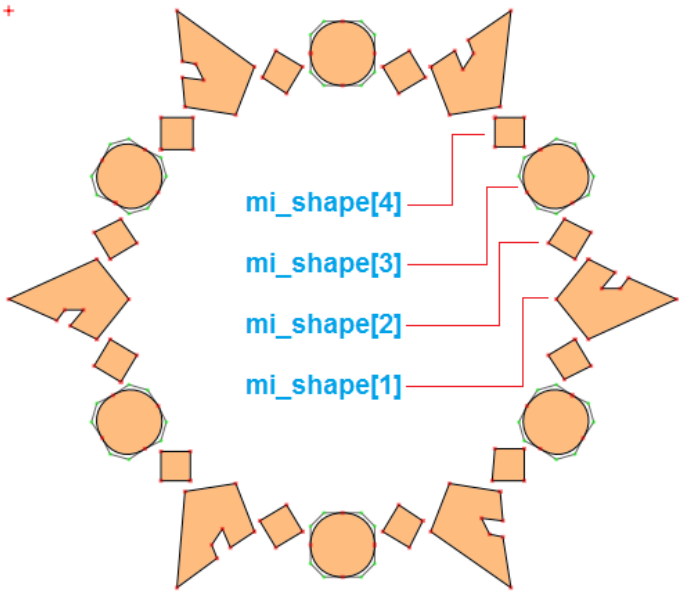
```
shape.array( mi_shape, 6, "radial", 64 )
```

- loops de cada **shape** del Arreglo: 6
- Tamaño de la **tabla**: 2
- loops Total: 6 X 2 = 12
- Modo: "radial"
- Radio Interior del Arreglo: 64 px

Entonces la función hace el **Arreglo Radial** alternando seis veces a cada uno, a los elementos de la **tabla mi_shape**, a un radio de 64 px:



Acá otro ejemplo con una tabla de cuatro Shapes, en la que convenientemente las Shapes 2 y 4 son las mismas:



Modo Matricial

Este modo consiste en hacer los **Arreglos** a modo de una **Matriz** rectangular con una cierta cantidad de repeticiones de la **shape** en ambas direcciones.

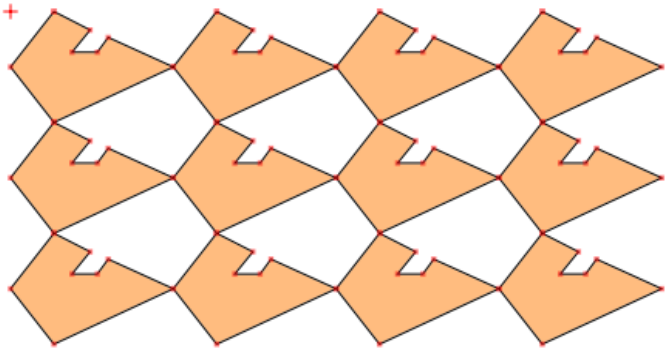
Ejemplo 1. Nuevamente iniciamos con **mi_shape** como una **shape**, declarada en forma de variable. El parámetro **loops** siempre debe ser una **tabla** con dos valores numéricos en donde se dan la cantidad de repeticiones, el primer número indica las repeticiones de la **shape** respecto al eje "**x**" (a lo ancho) y el segundo número indica la cantidad de repeticiones respeto al eje "**y**" (a lo alto). Por último, en **A_mode** ponemos la palabra "**array**":

```
Variables:
mi_shape = "m 12 0 l 0 15 l 12 30 l 45 15 l 27 7 l 24 11 l
            17 11 l 22 5 l 12 0 "
```

Y en **Return [fx]** ponemos así:

```
Return [fx]:
shape.array( mi_shape, {4, 3}, "array" )
```

Se generará el siguiente arreglo:



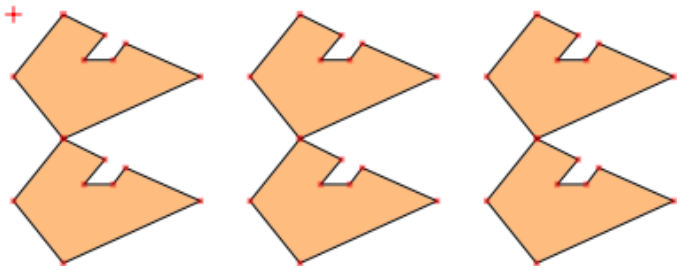
- loop Horizontal: 4
- loop Vertical: 3
- loops Total: 4 X 3 = 12
- Modo: "array"
- Distancia Separadora: 0 px (por default)

Ejemplo 2. Usamos las mismas configuraciones del ejemplo anterior, pero ahora incluimos a **Dxy** en forma de valor numérico:

```
Return [fx]:
shape.array( mi_shape, {3, 2}, "array", 12 )
```

- loop Horizontal: 3
- loop Vertical: 2

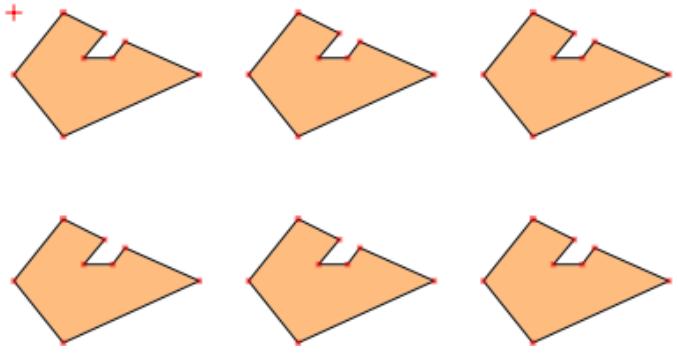
- loops Total: 3 X 2 = 6
- Modo: “array”
- Distancia Separadora Horizontal: 12 px
- Distancia Separadora Vertical: 0 px (default)



Ejemplo 3. Modificamos a **Dxy** a una **tabla** con dos valores numéricos, el primero indicará la distancia separadora horizontal y el segundo la vertical:

```
Return [fx]:
shape.array( mi_shape, {3, 2}, "array", {12, 20} )
```

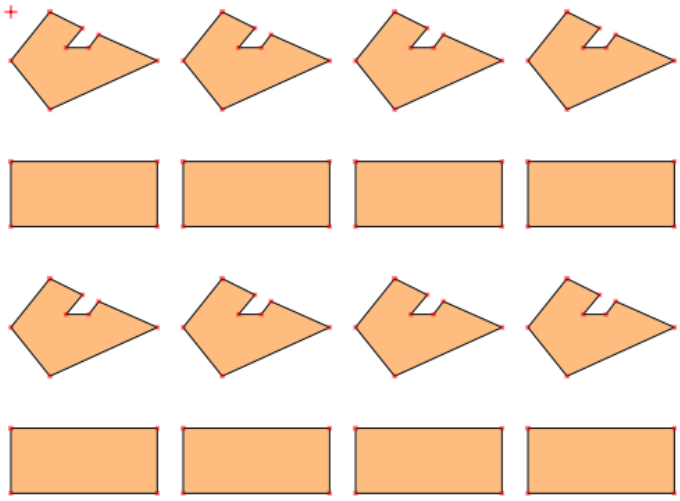
- loop Horizontal: 3
- loop Vertical: 2
- loops Total: 3 X 2 = 6
- Modo: “array”
- Distancia Separadora Horizontal: 12 px
- Distancia Separadora Vertical: 20 px



Ejemplo 4. mi_shape la usamos ahora como una **tabla** de Shapes, la cantidad de elementos de dicha **tabla** es decidida por cada quien:

```
Return [fx]:
shape.array( mi_shape, {4, 4}, "array", {8, 16} )
```

Entonces las Shapes de la **tabla mi_shape** se alternarán en el **Arreglo**, similar como pasaba en el **Arreglo Radial**:



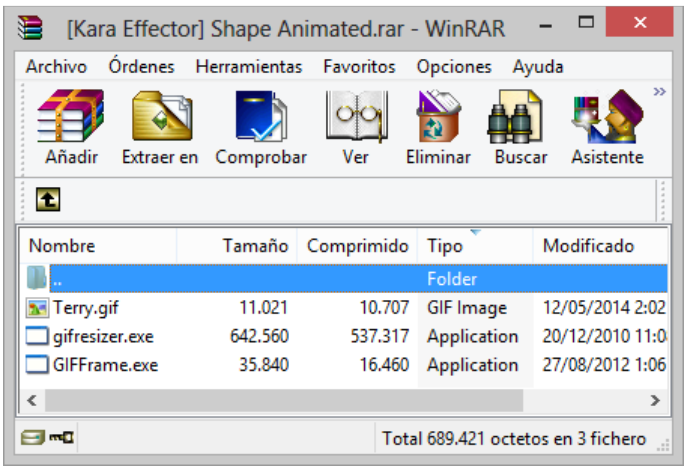
---

shape.animated(dur, frame_dur, frames, Sx, Sy): retorna Shapes rectangulares que contienen imágenes en formato **PNG** para hacer efectos de animaciones.

Esta función es de uso exclusivo para el filtro **VSFilterMod**, ya que está basado en el tag **\1img**, que es el que hace posible el poder insertar imágenes en formato **PNG** en un archivo .ass y por ello no es posible poder visualizar los resultados si solo usamos el **VSFilter 2.39**.

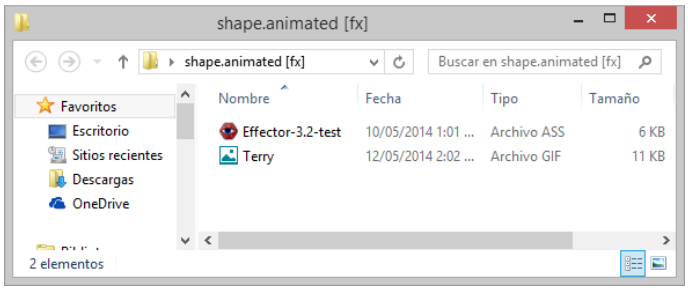
Para el siguiente ejemplo he subido un **.rar** que contiene tres archivos, un archivo **GIF** y dos aplicaciones que no requieren de instalación, que nos ayudarán a manipular y modificar los archivos **GIF**:

[Kara Effector] Shape Animated



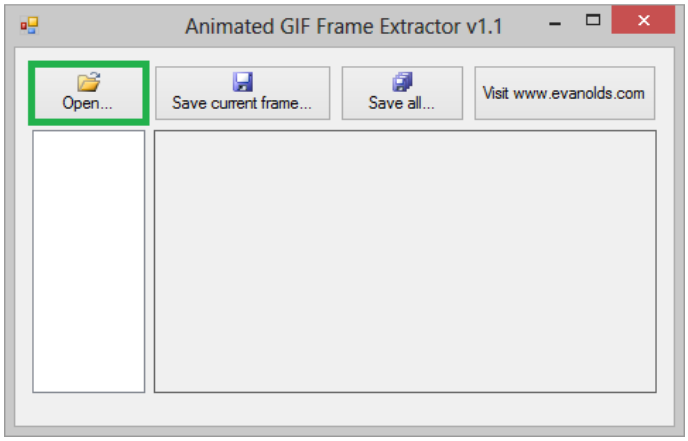
1. **Terry.gif:** es un ejemplo de imágenes animadas que pueden usar para un efecto en esta función.
2. **Gifresizer:** es una aplicación que nos permite dar las dimensiones a las imágenes que necesitemos, para que la animación se ajuste a las proporciones de las líneas karaoke y del vídeo usado.
3. **GIFFrame:** es una aplicación que extrae cada uno de los **frames** de un archivo **GIF**, en formato **PNG**, para poder insertarlas en nuestros karaokes.

El primer paso para usar la función **shape.animated** es tener el archivo **.ass** y el archivo **GIF** en la misma carpeta. No importa el nombre ni la ubicación de la carpeta, lo que importa es que ambos archivos estén en la misma:

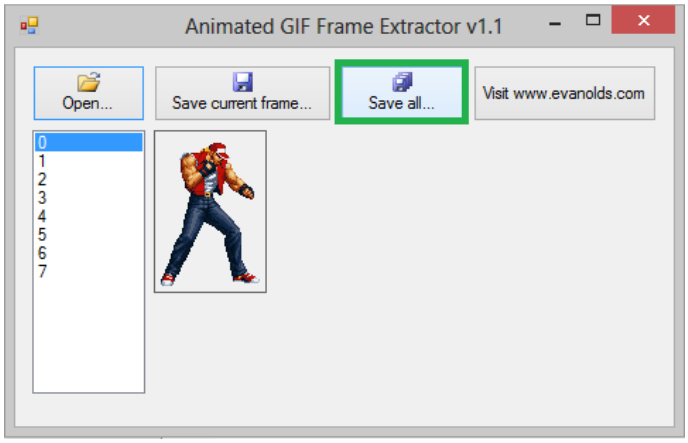


En el caso que nuestra **GIF** tenga un tamaño superior o inferior al que necesitamos (aunque no es recomendable ampliar un **GIF**, ya que la imagen pixela y empieza a perder calidad), abrimos la aplicación **GIFResizer** y en la parte inferior le damos las dimensiones en pixeles que se ajuste a nuestras necesidades. Esta aplicación crea un nuevo archivo, en tal caso se debe guardar en la misma carpeta del archivo **.ass** y del **GIF** original:

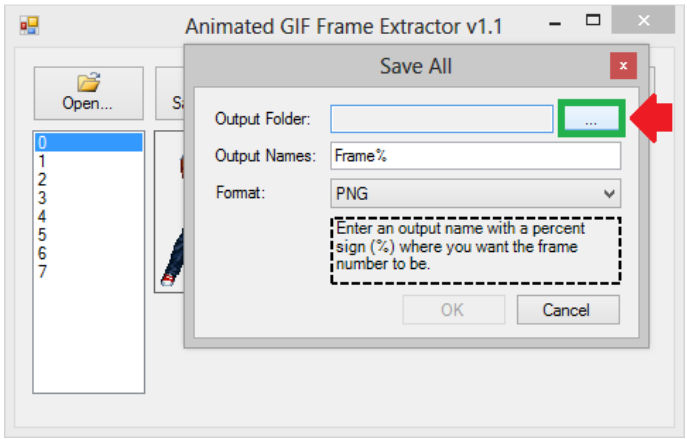
Para este ejemplo, no hubo la necesidad de cambiar el **GIF** de tamaño, pero en cualquiera de los dos casos, el tercer paso es abrir la aplicación **GIFFrame** y le damos al botón que pone "**Open…**":



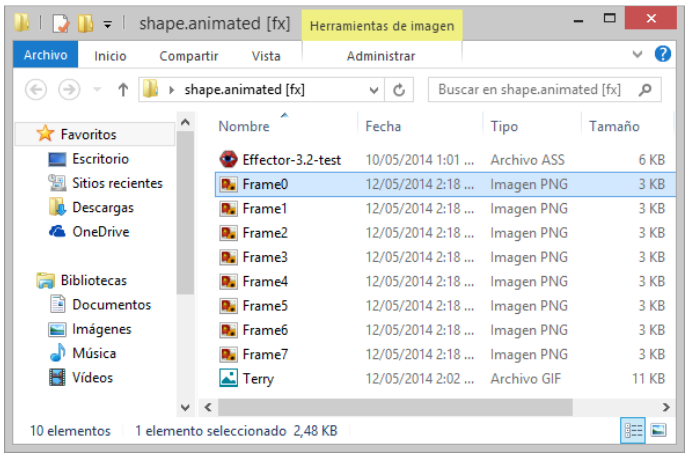
Ubicamos el **GIF** que vamos a utilizar, y pulsamos el botón "**Save All…**":



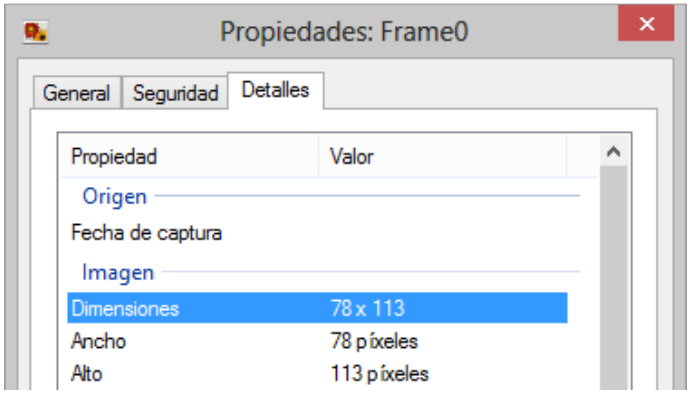
Y en donde pone "**Output Folder**" ubicamos la carpeta en donde están el archivo **.ass** que usaremos y el **GIF**, y en donde pone "**Format**" siempre debe poner "**PNG**":



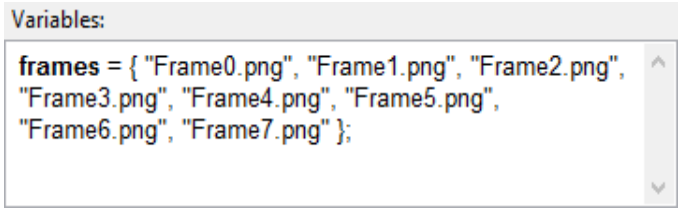
Entonces la aplicación **GIFFrame** extrae las imágenes **PNG** que componen al **GIF**, y éstas quedan en la carpeta en donde estaban el archivo **.ass** y el **GIF**:



Para confirmar las dimensiones de los archivos **PNG**, las podemos ver en sus propiedades (78 X 113 px):



Hasta este punto, podemos decir que hemos hechos todos los pasos preliminares para usar la función y generar un efecto de animación. Los pasos siguientes ya los debemos realizar desde el **Kara Effector** directamente, y uno de ellos es crear una tabla en la celda de texto "**Variables**" que contenga, entre comillas, el nombre de cada una de las imágenes y su extensión (.**png**):



Para este ejemplo, tan solo son ocho imágenes, desde la 0 hasta la 7, y en algunos casos hay archivos **GIF** que están compuestos de mucho más.

Y el segundo paso en la ventana de modificación del **Kara Effector** es llamar a la función **shape.animated** en la celda de texto **Return [fx]:**

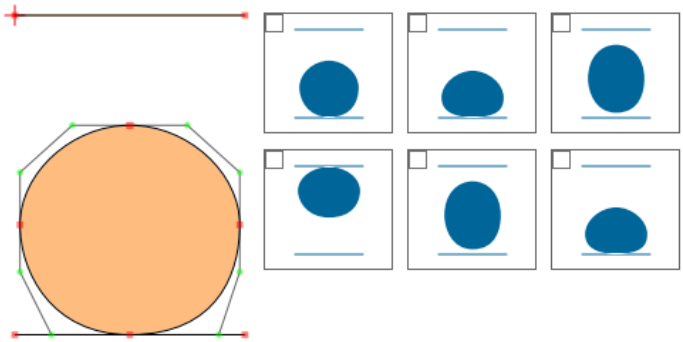
1. En el primer parámetro de la función ponemos el tiempo total de duración de la animación, para este ejemplo he usado la variable **fx.dur**, que ya sabemos que es el tiempo de cada una de las líneas de efecto generadas.
2. En el segundo parámetro debemos poner la duración en milisegundos que tendrá cada uno de los **frames** de la animación de nuestro efecto. Para este ejemplo, **100** ms. Son recomendables valores entre 40 y 300 ms.
3. Para el tercer parámetro ponemos el nombre de la **tabla** declarada en "**Variables**", que para este ejemplo se llama **frames**.
4. Y los parámetros cuatro y cinco son el ancho y el alto en pixeles de cada una de las imágenes **PNG**.

El resto de las configuraciones dependerá del efecto que queremos hacer, configuraciones como el **Template Type**, las posiciones y los tiempos. Para este ejemplo usé un **Template Type: Line**, con posición izquierda de la línea y con los tiempos por default de la misma:

Para el caso en que a la derecha o en la parte superior de las imágenes de la animación se vea alguna parte de otra imagen, debemos reducir las medidas ingresadas, si por ejemplo se ve a la derecha, reducimos un poco al ancho.

shape.animated2(dur, frame_dur, shapes): es similar a **shape.animated**, pero con la diferencia que no retorna imágenes **PNG** sino Shapes. Otra de las diferencias con su función homónima, es que solo son necesarios tres parámetros para que haga la animación: la duración total, la duración de cada **frame** y la tabla de Shapes.

El primer paso es dibujar la secuencia de Shapes que harán el efecto de animación. Para este ejemplo hice seis Shapes que simulan una pelota rebotando:



Las dos líneas horizontales paralelas extras se dibujan para delimitar la animación, es decir que el ancho de las líneas hacen referencia al ancho total y están a una distancia una de la otra, de tal manera que estas dos líneas delimitan el alto de la animación. En este ejemplo las dos líneas ayudan a dar la referencia del suelo, para la línea inferior, y del techo para la línea superior.

El segundo paso es copiar el código de las Shapes y hacer una **tabla** en la celda de texto "**Variables**" con ellas. Recordemos que las Shapes la debemos poner entre comillas, ya sean dobles o sencillas, y el nombre de la **tabla** es a gusto de cada quien:

Variables:

```
shapes = {"m 0 0 l 44 0 m 0 61 l 44 61 m 22 21 b 11 21 1
30 1 40 b 1 49 7 61 22 61 b 39 61 43 49 43 40 b 43 30 33
21 22 21 ", "m 0 0 l 44 0 m 0 61 l 44 61 m 22 28 b 11 28 0
37 0 47 b 0 56 7 61 22 61 b 39 61 44 56 44 47 b 44 37 33
28 22 28 ", "m 0 0 l 44 0 m 0 61 l 44 61 m 22 10 b 11 10 2
19 2 34 b 2 43 7 58 22 58 b 39 58 42 43 42 34 b 42 19 33
10 22 10 ", "m 0 0 l 44 0 m 0 61 l 44 61 m 22 0 b 11 0 0 7
0 17 b 0 25 7 36 22 36 b 39 36 44 25 44 17 b 44 7 33 0 22
0 ", "m 0 0 l 44 0 m 0 61 l 44 61 m 22 10 b 11 10 2 19 2
34 b 2 43 7 58 22 58 b 39 58 42 43 42 34 b 42 19 33 10
22 10 ", "m 0 0 l 44 0 m 0 61 l 44 61 m 22 28 b 11 28 0 37
0 47 b 0 56 7 61 22 61 b 39 61 44 56 44 47 b 44 37 33 28
22 28 " }
```

Llamamos en "**Return [fx]**" a la función y como tercer parámetro ponemos a la tabla declarada en "**Variables**" que contiene a todas las Shapes de la animación:

Return [fx]:

```
shape.animated2(fx.dur, 120, shapes)
```

Y la función generará la animación:

La ventaja de **shape.animation2** es poder hacer modificar fácilmente el tamaño de la misma ya que está hecha con Shapes, y éstas se modifican con los tags **\fscx** y **\fscy**.

Ya hemos avanzado mucho en la Librería **shape** y cada vez resta menos para terminar de ver todas sus funciones. En el **Tomo XVIII** continuaremos con el resto de las funciones hasta que las hayamos visto todas y consideremos que ya dominamos un poco más el mundo de los efectos hechos con Shapes. 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.facebook.com/karaeffector