Kara Effector 3.2:

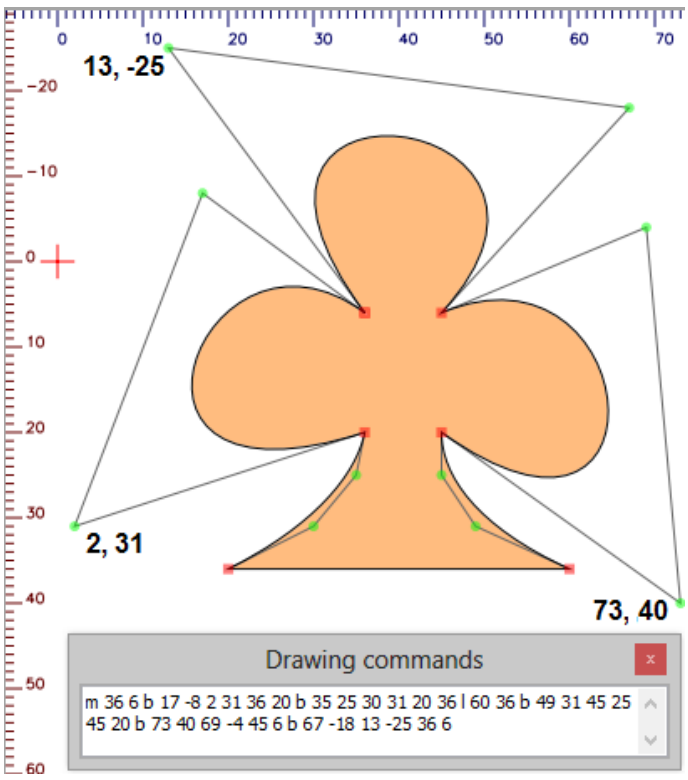
En el **Tomo XVIII** seguimos viendo el resto de las funciones de la Librería **shape** que hemos venido viendo desde hace ya varios Tomos. Habrán notado el gran tamaño de esta Librería y la importancia de la misma, ya que es una parte muy importante de todo lo que se debe saber para hacer efectos de alta calidad.

Librería Shape [KE]:

shape.config(Shape, Return, Ratio): esta función es similar a **shape.info**, pero con la ventaja que retorna mucha más información de la **shape** ingresada.

El parámetro **Ratio** es opcional y hace referencia a un valor por el cual se multiplicarán todos los puntos de la **shape** ingresada.

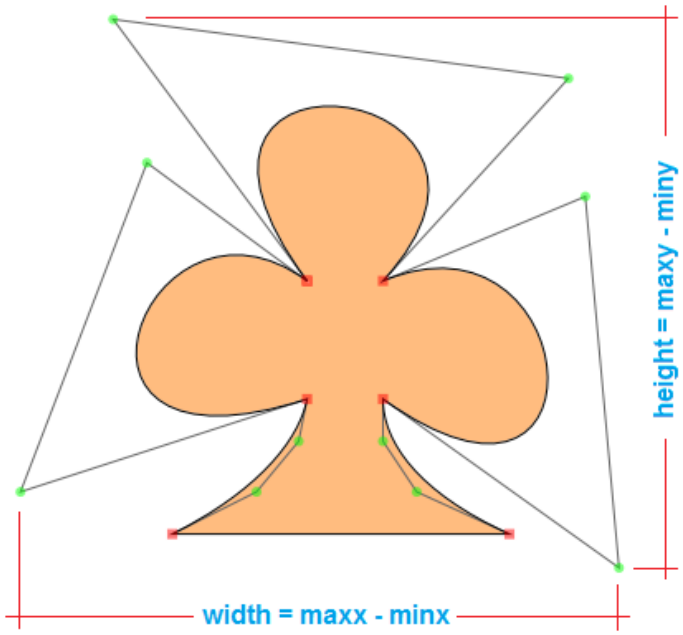
El parámetro **Return** es el que decide lo que retornará la función y tiene múltiples opciones:



Usaremos la anterior **shape** para los siguientes ejemplos, y como siempre (aunque no es obligatorio), la declararé a modo de variable en la celda de texto "**Variables**" para que sea un poco más simple su uso:

```
Variables:
mi_shape = "m 36 6 b 17 -8 2 31 36 20 b 35 25 30 31
            20 36 l 60 36 b 49 31 45 25 45 20 b 73 40
            69 -4 45 6 b 67 -18 13 -25 36 6 "
```

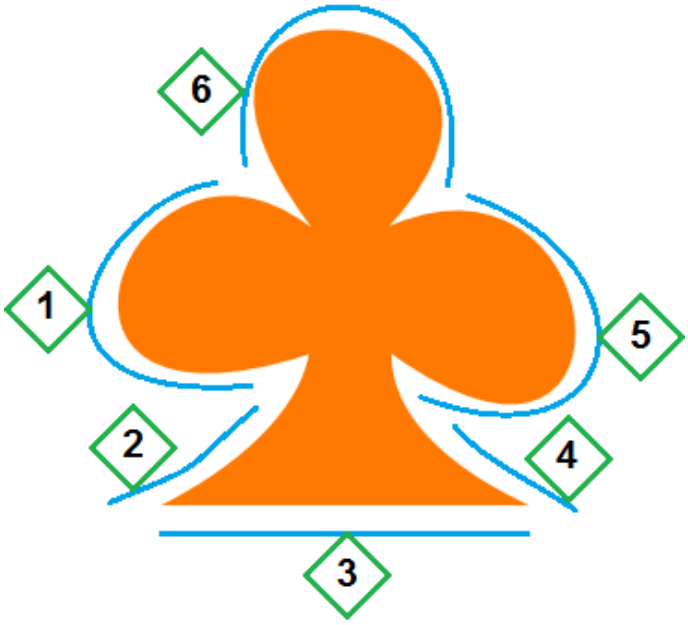
- **shape.config(mi_shape, "minx"):** este modo retorna el mínimo valor de las coordenadas "x". como se puede ver en la imagen, este valor es 2.
- **shape.config(mi_shape, "maxx"):** este modo retorna el máximo valor de las coordenadas "x", que para esta **shape** es 73.
- **shape.config(mi_shape, "miny"):** este modo retorna el mínimo valor de las coordenadas "y", que para esta **shape** es -25.
- **shape.config(mi_shape, "maxy"):** este modo retorna el máximo valor de las coordenadas "y", que para esta **shape** es 40.
- **shape.config(mi_shape, "width"):** este modo retorna el ancho de la **shape** calculado como la diferencia entre **maxx** y **minx**: 73 – 2 = 71 px



- **shape.config(mi_shape, "height"):** este modo retorna el alto de la **shape** calculado como la diferencia entre **maxy** y **miny**: 40 – (– 25) = 65 px

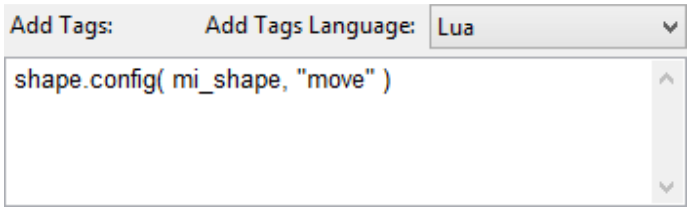
- **shape.config(mi_shape, "length"):** este modo retorna la medida de la longitud de la shape, es decir la medida de su perímetro:

- **shape.config(mi_shape, "segments"):** este modo retorna una **tabla** que contiene las coordenadas de cada uno de los segmentos de la **shape**. Dichas coordenadas están a su vez dentro de una **tabla**:

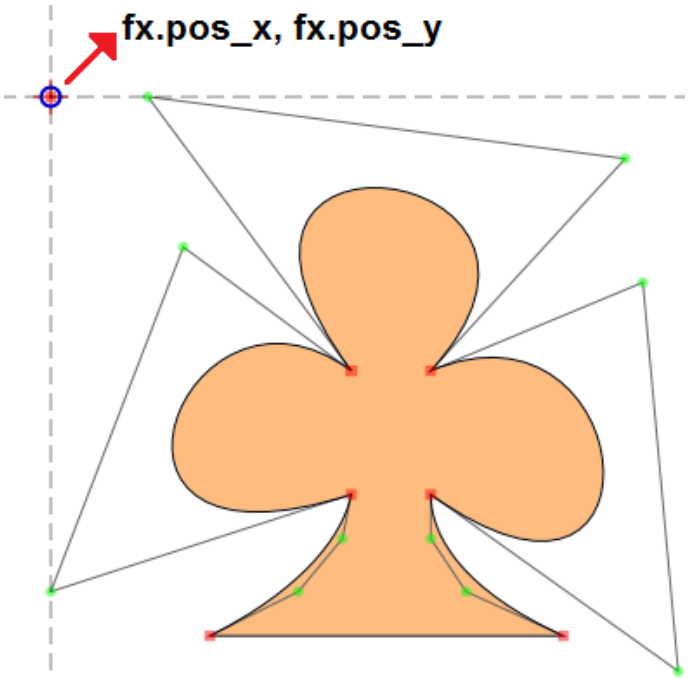



```
Segmentos = {
  [1] = {36, 6,  17, -8,  2,   31, 36, 20},
  [2] = {36, 20, 35,  25, 30,  31, 20, 36},
  [3] = {20, 36, 60,  36, },
  [4] = {60, 36, 49,  31, 45,  25, 45, 20},
  [5] = {45, 20, 73,  40, 69, -4,  45, 6 },
  [6] = {45, 6,  67, -18, 13, -25, 36, 6 }
}
```

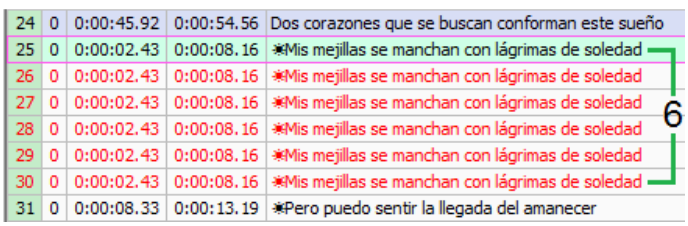
- **shape.config(mi_shape, “move”):** este modo usa toda la información concerniente a la **shape** para generar una secuencia de movimientos a modo de efecto, que hacen que el objeto karaoke se mueva siguiendo la trayectoria de la **shape** ingresada.
  Por ejemplo, en un **Template Type: Line** usamos la función y hacemos algo como esto:



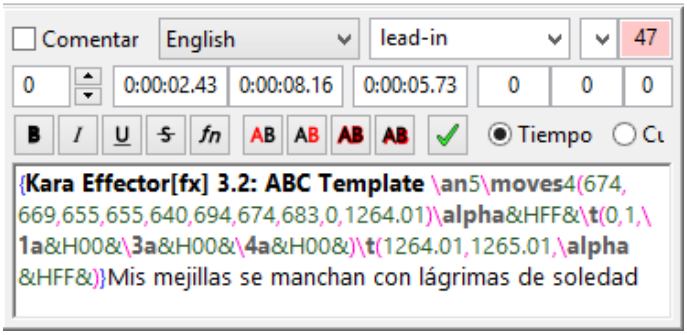
Lo que internamente hace la función es desplazar a la shape a un origen relativo, que ya no es el punto (0, 0) sino el centro en el vídeo del objeto karaoke:



Luego la función creará tantos **loops** de una misma línea, como segmentos tenga la **shape** ingresada en la función, que para este ejemplo son 6, como ya lo habíamos visto en el modo “**segments**”:



De la anterior imagen se pueden apreciar los seis **loops** por cada línea de fx generada, y que todas ellas tienen los mismos tiempos de inicio y final, pero que gracias a una serie de transformaciones, éstas generan el efecto de movimiento continuo armónico a través del perímetro de la **shape**:



El tag **\alpha&HFF&** genera la invisibilidad, y luego los tags **\1a**, **\3a** y **\4a** dan la transparencia por default del objeto karaoke. Esta característica imposibilita que usemos estos mismos tags para cualquier otra cosa dentro del mismo efecto, porque pueden afectar las transformaciones.

Por cada segmento de la **shape** que esté conformado por una **curva Bezier**, la función retornará un tag **\moves4** para poder moverse a través de ella. Si el segmento de la **shape** es una recta, entonces la función retorna un tag **\move** para moverse de un punto a otro.

Como ya lo había mencionado anteriormente, la función genera el **loop** de forma automática dependiendo de los segmentos que contenga la **shape**, así que para generar un **loop** independiente de ése, debemos hacerlo de la siguiente manera:

Sabemos que el 1 en este caso se pasa por alto, ya que la función generó un **loop** 6, y el 3 hace referencia ahora a la cantidad de repeticiones de esos 6 **loops**. O sea que el **loop** total será de 6 X 3 = 18, pero en pantalla, modificando un poco los tiempos de inicio y final, solo veremos los 3 que hemos asignado previamente en la celda de texto “**loop**”:

Es algo complicado intentar explicar con palabras, incluso con imágenes, lo que esta función hace, ya que el efecto se basa en movimientos y en la sincronía de los mismos. Lo que recomiendo es que pongan la función en práctica con varias Shapes para poder notar las diferencias entre los resultados. Ejemplos

- **shape.config(shape.rectangle, "move")**
- **shape.config(shape.triangle, "move")**
- **shape.config(shape.circle, "move")**

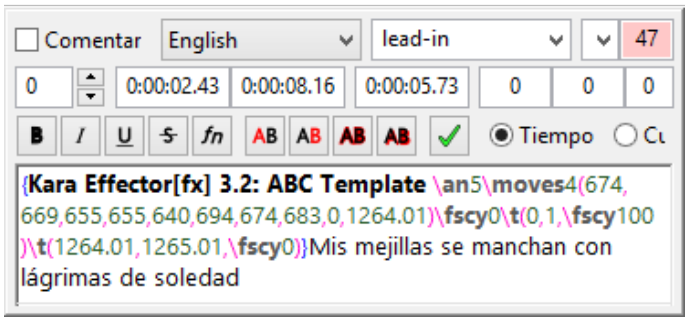
Este último ejemplo haría que el objeto karaoke se mueva siguiendo como trayectoria al perímetro de **shape.circle**, que tiene 100 px de diámetro, que es la medida de esta **shape predeterminada** del **Kara Effector**. Si quisiéramos que un objeto karaoke se moviera siguiendo la trayectoria de un círculo más grande o más pequeño, tenemos las dos siguientes opciones. Ejemplos:

1. **shape.config(shape.circle, "move" , 1.5):** este modo hace que el **Ratio** (1.5) aumente el tamaño de la **shape** ingresada en un 150%, entonces el objeto karaoke se moverá siguiendo la trayectoria de un círculo de 150 px de diámetro.
2. **shape.config(shape.size(shape.circle, 72), "move"):** la función **shape.size** redefine el tamaño del círculo a 72 px, que será el diámetro del círculo por el cual se moverá el objeto karaoke.

Entonces, podemos usar el tercer parámetro de la función **shape.config** (**Ratio**) o usar la función **shape.size**, para modificar las dimensiones de la **shape** ingresada y así redefinir las trayectorias de los desplazamientos.

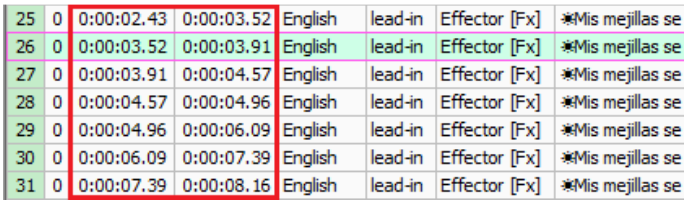
Como les mencioné antes, el modo "**move**" usa los tags de transparencia para generar el efecto de fluidez en el movimiento del objeto karaoke, lo que imposibilitaba el volver usar dichos tags nuevamente en el mismo efecto. Si inevitablemente tuviéramos que usar estos tags, tenemos un modo más que lo hace posible:

- **shape.config(mi_shape, "move2"):** este modo es similar al modo "**move**", ya que hacen lo mismo, pero no usa los tags de transparencia en sus transformaciones. El tag que el modo "**move2**" usa para generar el efecto de movimiento fluido es el tag **\fscy**:



Entonces en el modo "**move2**" el tag que no se puede usar en el resto del efecto es **\fscy**, ya que es el que hace que cada uno de los loops desaparezca en el momento justo y dé la sensación de fluidez en el movimiento.

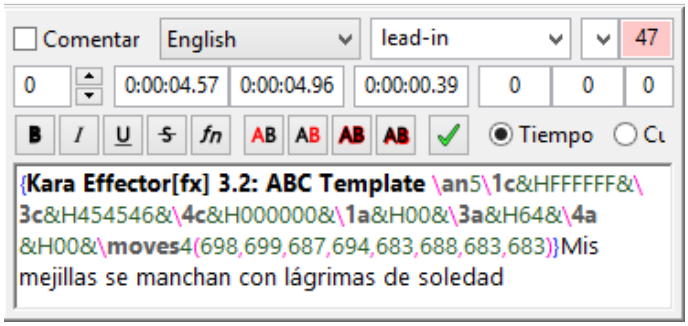
- **shape.config(mi_shape, "move3"):** este modo es similar a los modos "**move**" y "**move2**", genera el mismo efecto de movimiento, pero sin usar los tags de transparencias ni tampoco el tag **\fscy** en sus transformaciones. El modo "**move3**" genera líneas independientes respecto al tiempo, o sea que no necesita de un tag para hacer desaparecer al objeto karaoke que se mueve3 en determinado segmento.



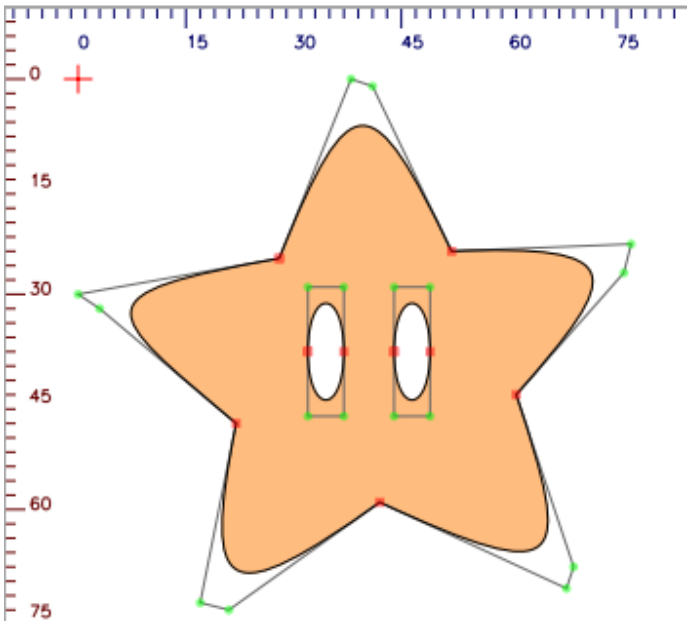
25	0	0:00:02.43	0:00:03.52	English	lead-in	Effector [Fx]	Mis mejillas se
26	0	0:00:03.52	0:00:03.91	English	lead-in	Effector [Fx]	Mis mejillas se
27	0	0:00:03.91	0:00:04.57	English	lead-in	Effector [Fx]	Mis mejillas se
28	0	0:00:04.57	0:00:04.96	English	lead-in	Effector [Fx]	Mis mejillas se
29	0	0:00:04.96	0:00:06.09	English	lead-in	Effector [Fx]	Mis mejillas se
30	0	0:00:06.09	0:00:07.39	English	lead-in	Effector [Fx]	Mis mejillas se
31	0	0:00:07.39	0:00:08.16	English	lead-in	Effector [Fx]	Mis mejillas se

El modo "**move3**" estará disponible a partir de la **Versión 3.2.7** del **Kara Effector**, para versiones anteriores solo es posible usar los modos "**move**" y "**move2**".

Al ver en detalle una de las líneas de fx generadas en este modo, notamos que el único tag que retorna es el de movimiento, **\move** para las restas de la **shape** y **\moves4** para las curvas **Bezier**, como en este ejemplo:



shape.incenter(Shape): esta función desplaza a la **shape** ingresada en el centro de las coordenadas del **AssDraw3**, con referencia al punto P = (0, 0). Ejemplo:

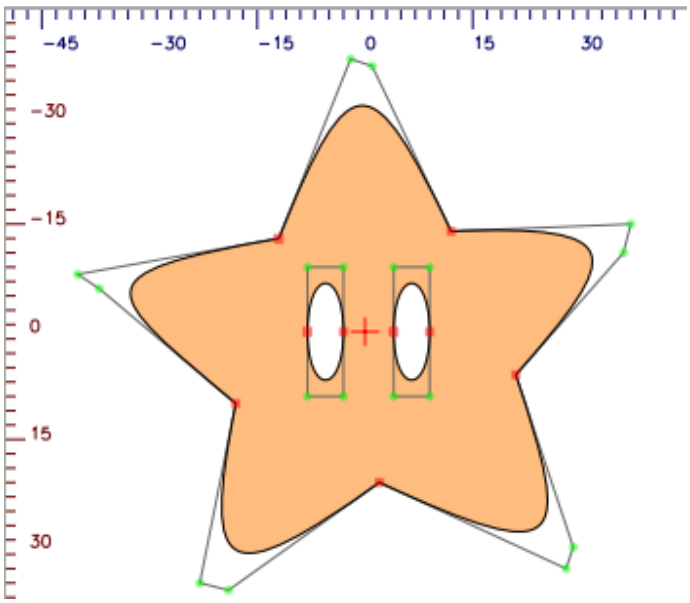


La anterior **shape** está ubicada en el **Cuadrante IV** del **AssDraw3.** Usamos la función con esta **shape**, por ejemplo en la celda de texto **Return [fx]:**

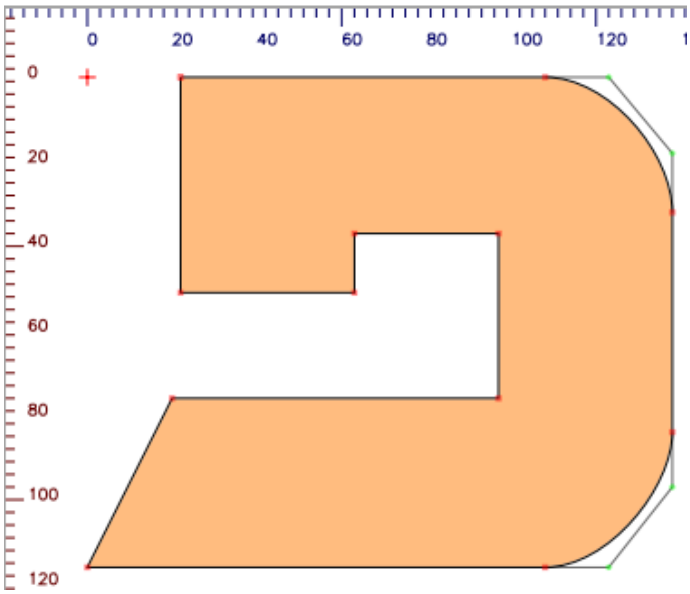
Return [fx]:

```
shape.incenter( mi_shape )
```

Y al copiarla la **shape** generada y pegarla en el **AssDraw3**:



shape.centerpos(Shape, Dx, Dy): desplaza a la **shape** ingresada con referencia a su centro, al punto con coordenadas P = (**Dx**, **Dy**). Ejemplo:



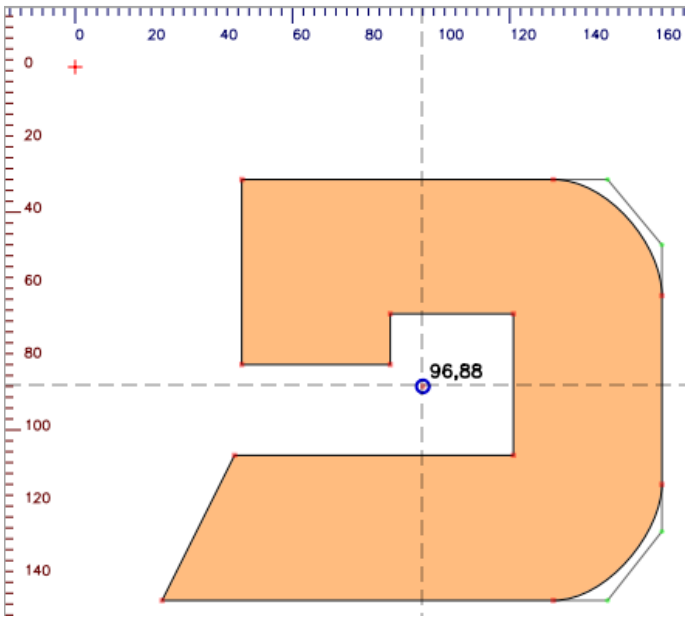
Ingresamos los dos valores que necesitemos desplazar la **shape** respecto a su centro, por ejemplo:

Return [fx]:

```
shape.centerpos( mi_shape, 96, 88 )
```

Dx Dy

Ahora la nueva ubicación del centro de la **shape** es el que se haya especificado en la función (96, 88):



shape.trajectory(loop, D_nim, D_max): crea una **shape** con una definida cantidad de segmentos (**loop**), que distan entre sí dependiendo de los parámetros **D_min** y **D_max** (Distancia mínima y Distancia máxima).

Cada uno de los segmentos es creado de forma aleatoria y dibujados con una **Curva Bezier** de tal manera que, entre todos los segmentos formen una sola trayectoria fluida con cada una de las curvas.

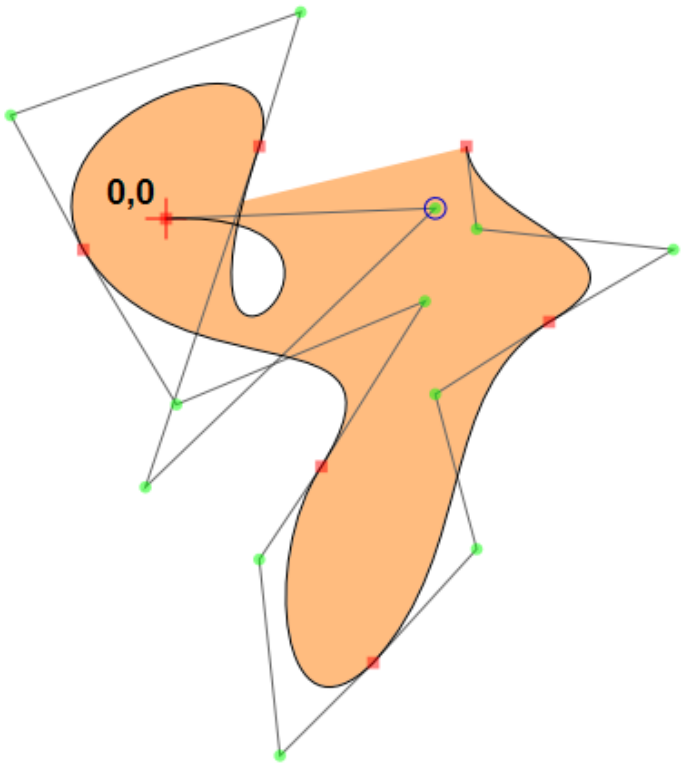
Los parámetros **D_min** y **D_max** son opcionales. En el caso de no ponerlos en la función, éstos tienen sus respectivos valores por default:

- **D_min** = 10*ratio
- **D_max** = 20*ratio

Ejemplo:

```
Return [fx]:
shape.trajectory( 5 )
```

Se genera una curva fluida con 5 segmentos de **Curvas Bezier**, y cada una de ellas separada de la otra a una distancia aleatoria entre 10 y 20 px:



Las curvas fluidas que genera esta función se pueden usar como una trayectoria de movimiento dentro de la función **shape.config** en los modos "**move**", "**move2**" y "**move3**". Ejemplos:

```
Add Tags:  Add Tags Language: Lua
shape.config( shape.trajetory( 6, 25, 50 ), "move" )
```

```
Add Tags:  Add Tags Language: Lua
shape.config( shape.trajetory( R(4,8) ), "move2" )
```

```
Add Tags:  Add Tags Language: Lua
shape.config( shape.trajetory( 5 ), "move3" )
```

Y para cada uno de los ejemplos anteriores, la función **shape.config** hará que el objeto karaoke de la línea de fx se mueva siguiendo como trayectoria a la curva generada por la función **shape.trajectory**.

Las curvas fluidas que genera la función **shape.trajectory** tienen muchas más aplicaciones y opciones de uso. En los próximos tomos, de a poco iremos viendo cómo sacarle el máximo provecho a esta y otras funciones.

Es todo por ahora. En el **Tomo XIX** continuaremos viendo más de las funciones de la librería **shape**. 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