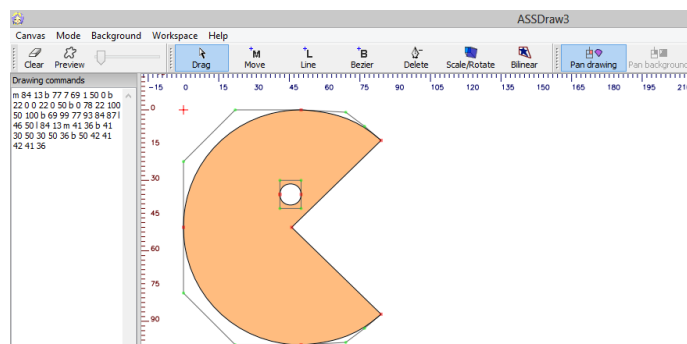
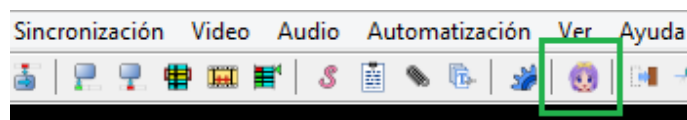

Kara Effector 3.2:

El **Tomo XIV** es el comienzo de una nueva librería y cada vez estamos más cerca de terminarla todas y así ampliar aún más las herramientas a nuestra disposición a la hora de desarrollar un **Efecto Karaoke**, un **Logo**, un **Cartel** y todo aquello que nos dispongamos a hacer para nuestros proyectos en el **Aegisub** y el **Kara Effector**.

Librería Shape [KE]:

Se conoce como **shape** a un dibujo hecho por vectores en el **AssDraw3** que viene por default en el **Aegisub**, y que nos sirven de apoyo como una herramienta más a la hora de desarrollar un Efecto.



La Librería **shape** hace referencia a dichos dibujos y a las funciones relativas a ellos. Aparte de las funciones en esta Librería, el **Kara Effector** consta de un amplio listado de **shapes** (dibujos) prediseñadas para hacer uso de ellas y a continuación veremos el nombre, su tamaño en píxeles y una pre-visualización de las mismas.

Shapes Prediseñadas del Kara Effector

shape.circle

100 x 100 px



shape.triangle

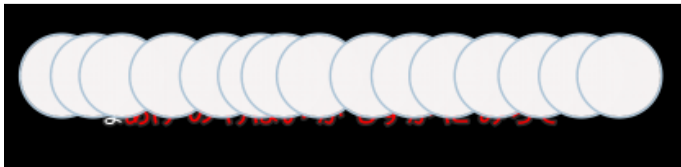
100 x 106 px



Ejemplo:

Return (fx): shape.circle

En dicho caso se retornaría un **círculo** en el lugar en donde antes estaban las sílabas de la línea karaoke:



Shapes Prediseñadas del Kara Effector

shape.rectangle

100 x 100 px



shape.pentagon

100 x 95 px



shape.hexagon

100 x 116 px



shape.octagon

100 x 100 px



shape.heart

100 x 106 px



shape.heart2t

100 x 106 px



Shapes Prediseñadas del Kara Effector

shape.heart_b

100 x 106 px



shape.shine1t

100 x 100 px



shape.shine2t

100 x 100 px



shape.shine3t

100 x 100 px



shape.shine4t

100 x 100 px



shape.trebol

100 x 106 px



shape.feather

100 x 100 px



shape.diamond

100 x 100 px



shape.gear

100 x 100 px



shape.bubble

100 x 100 px



shape.note1t

100 x 56 px















shape.note2t













100 x 100 px















Shapes Prediseñadas del Kara Effector

shape.note3t	shape.note4t
100 x 100 px	100 x 100 px
	
shape.star	shape.star1t
100 x 95 px	100 x 95 px
	
shape.star2t	shape.star3t
100 x 116 px	100 x 116 px
	
shape.star4t	shape.star5t
100 x 116 px	100 x 116 px
	
shape.star6t	shape.star7t
100 x 95 px	100 x 95 px
	
shape.star8t	shape.star9t
100 x 95 px	100 x 116 px
	













Shapes Prediseñadas del Kara Effector

shape.star10t	shape.sakura
100 x 116 px	100 x 130 px
	
shape.sakura1t	shape.sakura2t
100 x 116 px	100 x 116 px
	
shape.sakura3t	shape.sakura4t
100 x 116 px	100 x 106 px
	
shape.sakura5t	shape.sakura6t
100 x 100 px	100 x 116 px
	
shape.sakura7t	shape.snow1t
100 x 116 px	100 x 108 px
	
shape.snow2t	shape.snow3t
100 x 96 px	100 x 94 px
	






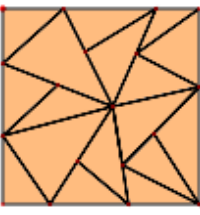
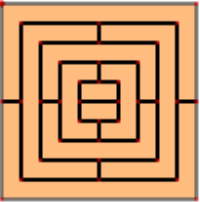


Shapes Prediseñadas del Kara Effector

shape.flower1t	shape.flower2t
100 x 95 px	100 x 95 px
	
shape.flower3t	shape.flower4t
100 x 95 px	100 x 95 px
	
shape.flower5t	shape.flower6t
100 x 95 px	100 x 95 px
	
shape.flower7t	shape.flower8t
100 x 95 px	100 x 95 px
	
shape.flower9t	shape.flower10t
100 x 95 px	100 x 95 px
	
shape.flower11t	shape.flower12t
100 x 116 px	100 x 116 px
	

Shapes Prediseñadas del Kara Effector

shape.flower13t	shape.flower14t
100 x 116 px	100 x 130 px
	
shape.flower15t	shape.flower16t
100 x 116 px	100 x 116 px
	
shape.flower17t	shape.flower18t
100 x 116 px	100 x 116 px
	
shape.flower19t	shape.flower20t
100 x 116 px	100 x 116 px
	
shape.flower21t	shape.flower22t
100 x 116 px	100 x 116 px
	
shape.flower23t	shape.flower24t
100 x 116 px	100 x 116 px
	

Shapes Prediseñadas del Kara Effector

shape.flower25t	shape.flower26t
100 x 116 px	100 x 130 px
	
shape.flower27t	shape.flower28t
100 x 116 px	100 x 116 px
	
shape.flower29t	shape.cristal17
100 x 116 px	100 x 100 px
	
shape.geometric10	shape.diagonal13r
100 x 100 px	100 x 100 px
	
shape.diagonal13l	
100 x 100 px	
	

Como pueden ver, son muchas las opciones de Shapes a escoger entre todas las Shapes que vienen por default en el **Kara Effector**, todo dependerá del efecto a realizar y de los resultados que queremos obtener.

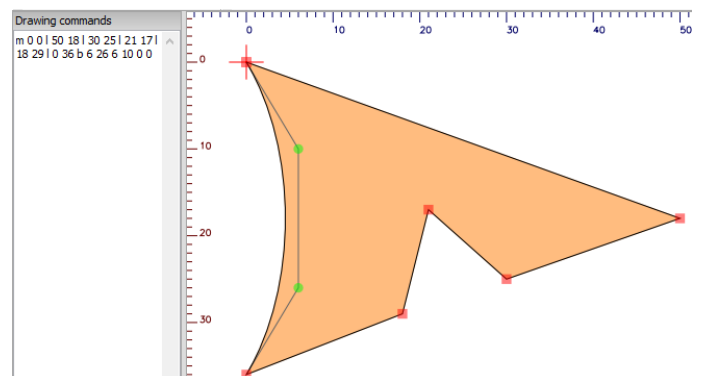
Con el anterior listado de Shapes prediseñadas del **Kara Effector**, comenzamos con la descripción de las funciones de la Librería **shape**, en donde algunas de ellas nos sirven para modificar y crear nuestras propias Shapes o, generar efectos directamente.

shape.rotate(Shape, Angle, x, y): rota a la shape en un ángulo dado respecto al punto **P = (x, y)**.

Los parámetros **x** e **y** son opcionales al tiempo, sus valores por default son las coordenadas del punto **P = (0, 0)**. El parámetro **Angle** hace referencia a un valor numérico de un ángulo entre 0° y 360°.

Para este y los próximos ejemplos usaremos un **Template Type: Line**, con el fin de generar una única línea por cada línea karaoke y así, sea más sencillo visualizar el resultado.

La **shape** que usaré en estos ejemplos será una muy simple, pero con un diseño en particular que permita ver la rotación de la misma:



Ejemplo:

Por lo general, siempre suelo declarar a las Shapes en la celda de texto "**Variable**", con el fin de hacer un poco más cómodo el uso de la misma, pero también se pueden usar directamente, entre comillas, dentro de las funciones que requieran una **shape** como alguno de los parámetros a usar dentro de ella:

Variables:

```
mi_shape = "m 0 0 | 50 18 | 30 25 | 21 17 | 18 29 | 0 36  
b 6 26 6 10 0 0 "
```

Las dos formas de usar la **shape** son válidos. Con el fin de que se visualice en pantalla, usaremos la función en la celda **Return [fx]**.

Opción 1:

```
Return [fx]:  
shape.rotate( mi_shape, 45 )
```

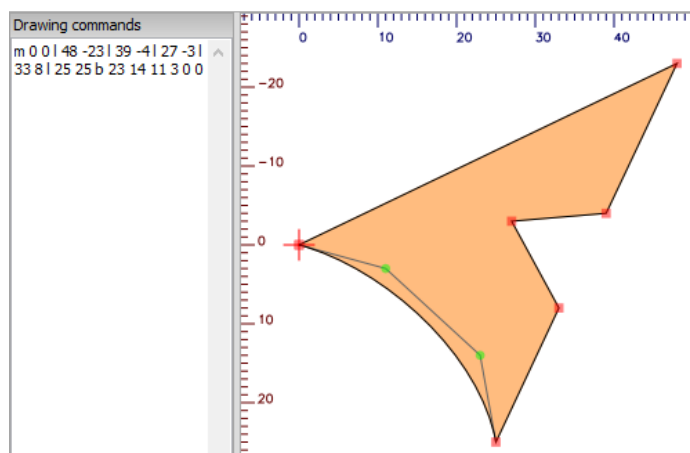
Opción 2:

```
Return [fx]:  
shape.rotate( "m 0 0 | 50 18 | 30 25 | 21 17 | 18 29 | 0 36  
b 6 26 6 10 0 0", 45 )
```

En vídeo:



En el **AssDraw3**:



La **shape** ha rotado 45° respecto al punto **P = (0, 0)**, ya que en el ejemplo anterior se omitieron los parámetros **x** e **y**.

Hecho este ejemplo, la pregunta pareciera caer por su propio peso, ¿por qué no simplemente usar el tag **\frz** para rotar la **shape** en vez de esta función?

Cuando la **shape** rotada se va a usar como una simple **shape**, entonces sí es lo mismo usar el tag **\frz** para rotarla y la función **shape.rotate**, pero cuando se va a usar una **shape** para generar un **\clip** o un **\iclip**, entonces ningún tag puede afectar a las **Shapes** que se encuentren dentro de ellos. Ejemplo:

\clip(m 0 0 | 0 50 | 0 0)

Como la **shape** "m 0 0 | 0 50 | 0 0" está dentro del **\clip**, no hay forma de modificar las características de la misma, a menos que usemos las funciones de la Librería **shape**. Si quisiéramos rotar la **shape** dentro del **\clip**, lo haríamos de la siguiente forma. Ejemplo:

```
Add Tags: Add Tags Language: Automation Auto-4  
\clip( !shape.rotate( "m 0 0 | 0 50 | 0 0", 30 ) ! )
```

Entonces se usará la **shape** rotada 30° dentro del **\clip**.

Nos resta ver ejemplos de cómo usar la función con los parámetros **x** e **y** incluidos. Ejemplos:

- **shape.rotate(mi_shape, 100, 0, 20)**
- **shape.rotate(mi_shape, 150, -30, 0)**
- **shape.rotate(mi_shape, 30, 45, -20)**
- **shape.rotate(mi_shape, 210, -15, -50)**

shape.reflect(Shape, Axis): refleja a la **shape** respecto al eje asignado en el parámetro **Axis**. Ejemplo:

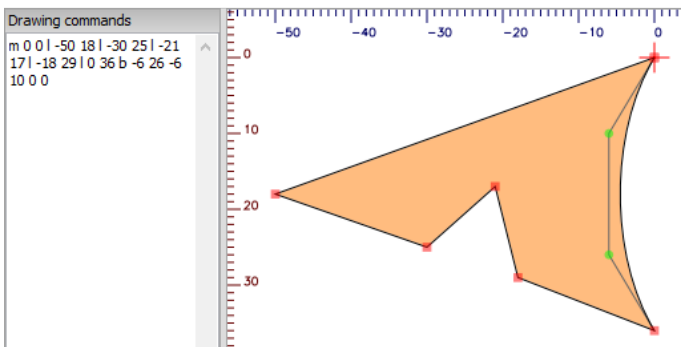
```
Return [fx]:  
shape.reflect( mi_shape, 'y' )
```

El parámetro **Axis** puede ser "x" para hacer referencia al eje "x", o "y" para hacer referencia al eje "y", como se acaba de usar en el ejemplo inmediatamente anterior.

Lo que resultaría así:



Y en el **AssDraw3** se verá así:



Entonces decimos que si **Axis** es "x", la **shape** se reflejará respecto al eje "x"; si es "y" se reflejará en dicho eje, pero hay una tercera opción que hace que la **shape** se refleje respecto a ambos ejes al mismo tiempo, así:

```
shape.reflect( mi_shape, "xy" )
```

Cuando **Axis** es "xy" la **shape** se refleja respecto a los ejes "x" e "y" de manera simultánea. Se obtiene el mismo resultado si solo no ponemos el parámetro **Axis**:

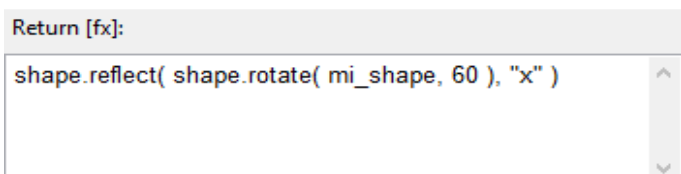
```
shape.reflect( mi_shape )
```

En ambos casos la **shape** será reflejada respecto a los dos ejes del plano.

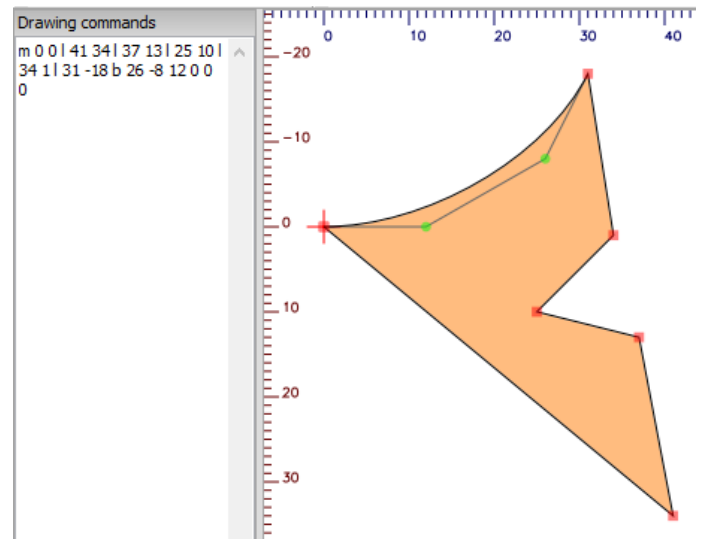
Veamos un ejemplo combinando las dos funciones hasta acá vistas de la Librería **shape**:

```
shape.reflect( shape.rotate( mi_shape, 60 ), "x" )
```

O sea que la **shape** primero se rota 60° y luego se refleja respecto al eje "x":



Y en el **AssDraw3** podemos ver el suceso anteriormente descrito:



Son muchas las posibilidades al combinar tan solo estas dos funciones. Espero que puedan inventar sus propios ejemplos y que se puedan sorprender con los diferentes resultados.

Este **Tomo XIV** es solo la introducción de la Librería **shape**, ya que aún nos resta por ver muchas más funciones de la misma y todas las posibilidades que ellas nos ofrecen. 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