text.bezier – Librería Text

text.bezier(Shape, mode, Accel, Offset_time) :

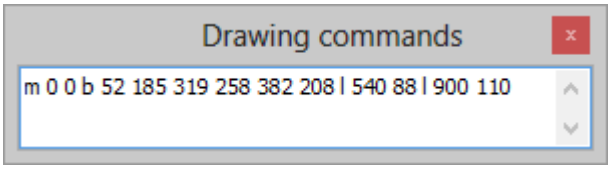
Esta función hace que el texto adopte la forma de una **curva Bézier**, de una **shape** ingresada en el primer parámetro o de un clip dibujado en las líneas del script.

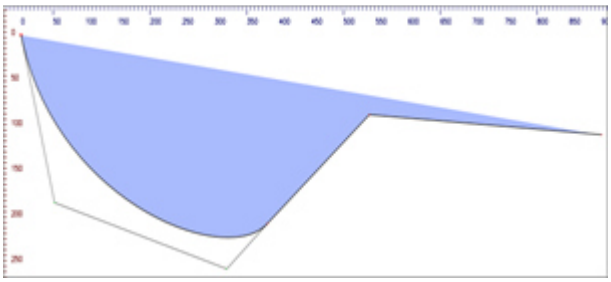
Esta función hace parte de la Librería Text [KE] y está disponible para la versión **3.2.9.4** o superior del **Kara Effector**, debido a su gran importancia se dedicará un artículo completo describiendo sus principales modos de uso en karaokes y en edición de carteles.

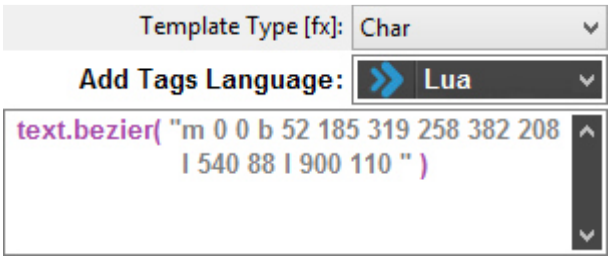
El parámetro **Shape** tiene tres diferentes opciones, y cada una de ella con diversas características que a continuación veremos:

- el código convencional de una **shape**.

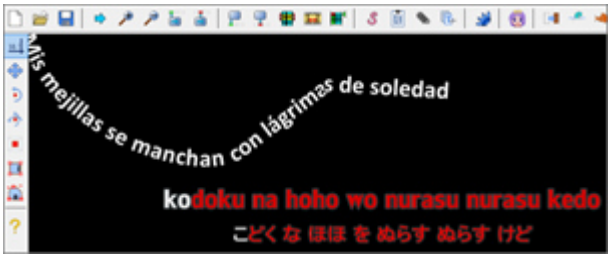
Ejemplo:





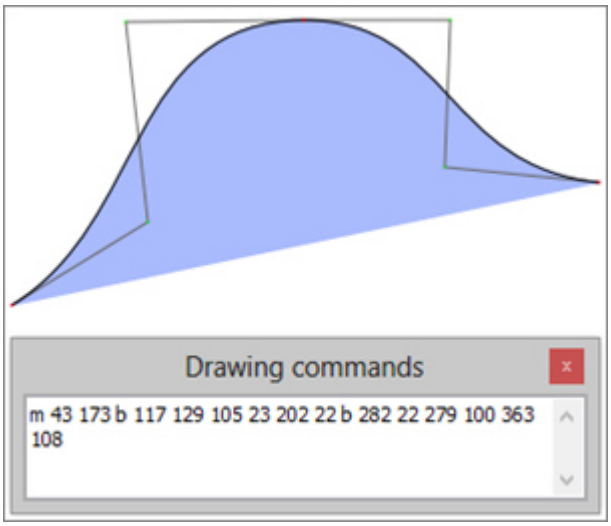


Y al aplicar:

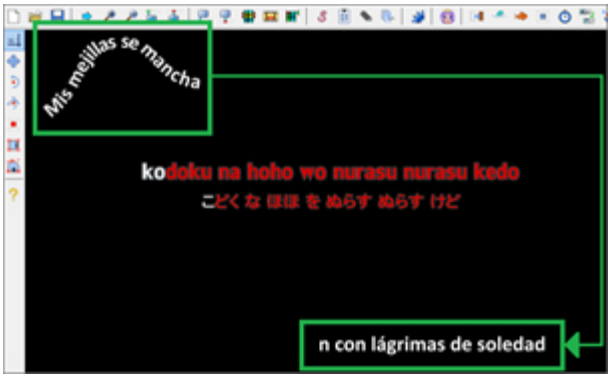


No importa la opción que elijamos en el parámetro **Shape**, debemos cerciorarnos que ésta mida igual o más que la longitud de la línea del script, o sea que su longitud sea mayor o igual que **line.width**.

Ejemplo:

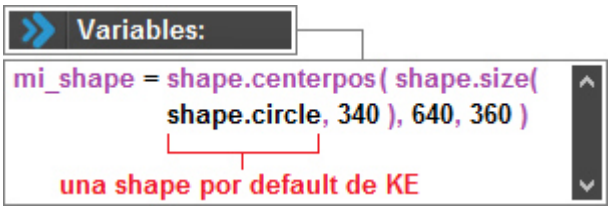


Si ingresamos una shape que mida menos que la longitud de la línea a la que le apliquemos un efecto, el texto quedará en su posición por default como ni no hubiésemos usado la función:

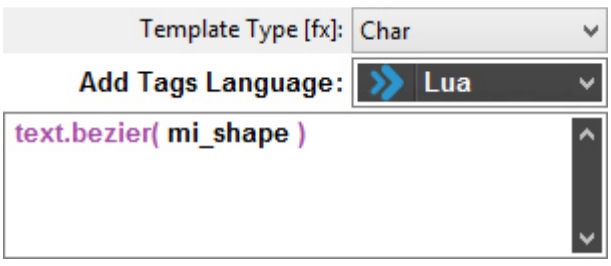


Ejemplo:

Podemos usar alguna de las Shapes que traen por default el **KE** o una modificación de las mismas:



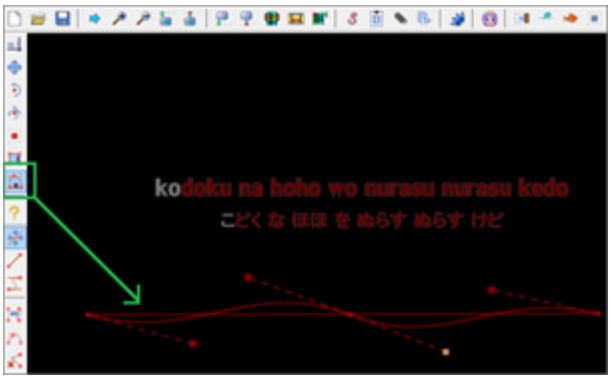
Y en **Add Tags** ponemos:



el parámetro **Shape** puede ser, también, un clip dibujado en una o más líneas del script. Las líneas que no tengan un clip dentro de ellas, no se verán afectadas por la función.

Ejemplo:

Dibujamos un clip en una o más de las líneas del script:

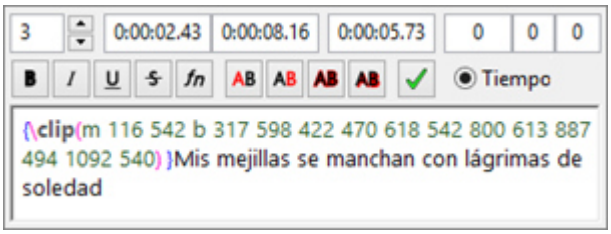


Y en **Add Tags** ponemos a alguna de estas dos opciones:

- **text.bezier(nil)**
- **text.bezier( )**

O sea que: **text.bezier(nil) = text.bezier( )**

En la línea debemos ver algo más o menos así:



Y al aplicar:

Y aquellas líneas en las que no dibujamos un clip, no se verá afectadas por la función.

La tercera opción del parámetro **Shape** es cuando es una **tabla** con mínimo 2 Shapes y máximo 4, y la veremos más adelante con el fin de ver primero el resto de los parámetros de la función.

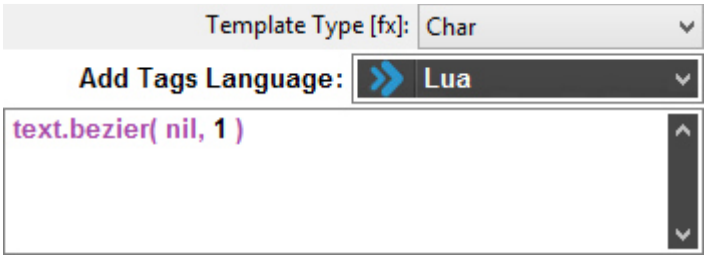
El parámetro **mode** es un número entero entre 1 y 6, lo que nos da seis opciones diferentes de usar la función. Cada uno de ellos es independiente del parámetro **Shape**, es decir que no importa la manera en que le ingresemos la **shape** a la función:

mode = 1

Justifica el texto en el centro de la longitud de la shape:

Ejemplo:

Usamos como ejemplo el clip dibujado del ejemplo anterior, pero con cualquier **shape** ingresada, el **mode =1** hará que el texto quede justificado en el centro de la longitud de la misma:

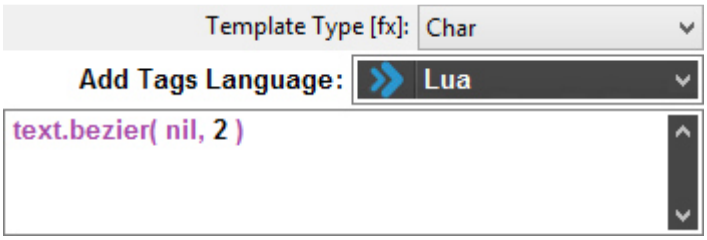


Al aplicar, el texto (en este caso, los caracteres de la línea) adopta la forma del clip dibujado previamente y como lo había mencionado antes, éste queda justificado respecto al centro de la longitud total de la **shape**:

mode = 2

Es el modo por default de la función, y justifica el texto a la izquierda de la **shape**, o dicho de otra manera, coloca al texto desde el inicio de la **shape**:

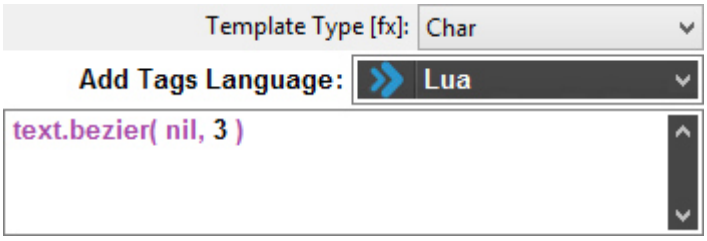
Ejemplo:



mode = 3

Justifica el texto a la derecha de la **shape**, o dicho de otra manera, coloca al texto desde el final de la **shape**:

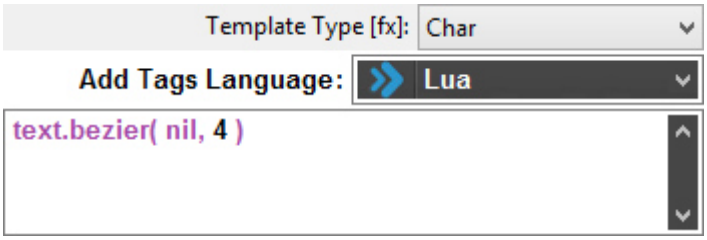
Ejemplo:



mode = 4

Hace que el texto se extienda a lo largo de la longitud total de la **shape**:

Ejemplo:

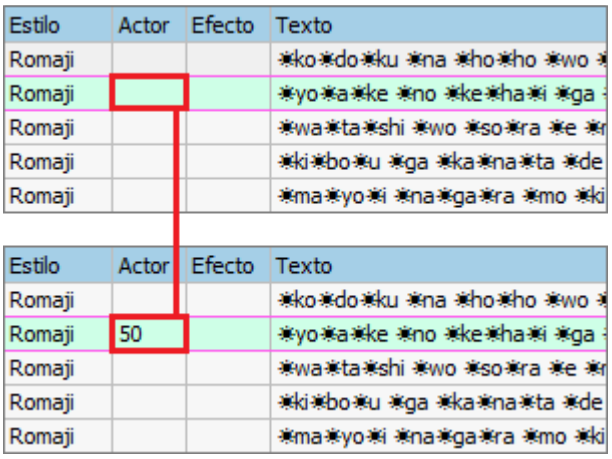


Los tres primeros valores del parámetro **mode** (1, 2 y 3) tienen una ventaja más que podemos usarla con la celda "**Actor**" del **Aegisub**, es un número que indica la cantidad de pixeles que necesitemos desplazar el texto una vez que éste adopta la forma de la **shape**.

No importa que en la celda "**Actor**" de una o más líneas del script ya hayamos escrito algo, lo único que debemos hacer es poner el valor que necesitemos al lado de lo que ahí ponga:

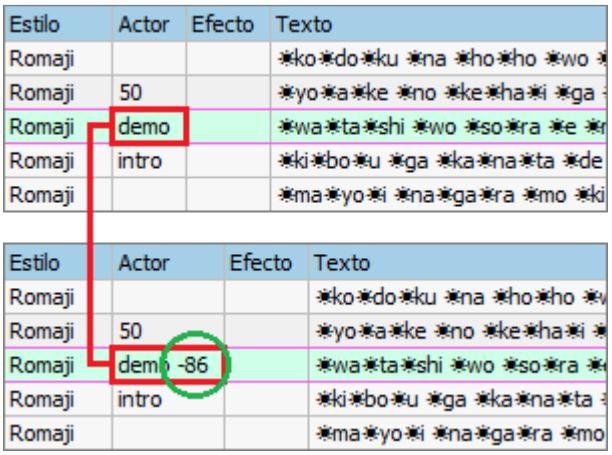
- Si la celda "**Actor**" está vacía solo ponemos el valor que necesitemos en la línea:

Ejemplo:



Si en la celda "**Actor**" ya pone algo, solo debemos agregar un espacio seguido del valor que vamos a desplazar el texto sobre el perímetro de la shape:

Ejemplo:

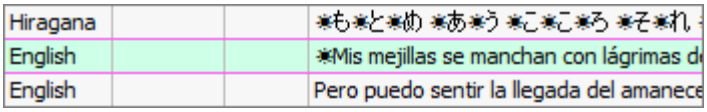


De cualquiera de las dos formas, la función reconoce el valor numérico que haya en la celda "**Actor**" y lo tomará como la distancia medida en pixeles para desplazar el texto según queramos.

En el **mode = 1**, dado que el texto queda justificado en el centro de la longitud total del perímetro de la shape, si el valor puesto en la celda "**Actor**" es positivo, lo desplazará hacia la derecha, en caso contrario, hacia la izquierda.

Ejemplo:

Si no hay un valor numérico en la celda "**Actor**", el valor del desplazamiento se asume como cero (0).



Es decir, que para **mode = 1**, el texto no se desplazará ni a la izquierda ni a la derecha.

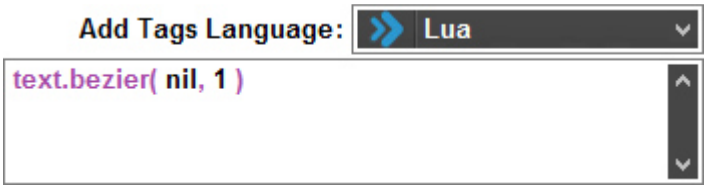
Si ponemos un valor positivo, se desplazará a la derecha:

Ejemplo:

Hiragana			✷も✷と✷め ✷あ✷う ✷こ✷こ✷ろ ✷
English	80		✷Mis mejillas se manchan con lágr
English			Pero puedo sentir la llegada del ar

Y para valores negativos, se desplazará a la izquierda:

Ejemplo:



Hiragana			✷も✷と✷め ✷あ✷う ✷こ✷こ✷ろ ✷
English	-50		✷Mis mejillas se manchan con lágr
English			Pero puedo sentir la llegada del ar

Entonces, con **mode = 1**, sí se tiene en cuenta el signo de la cantidad numérica que pongamos en la celda "**Actor**", todo depende de hacia dónde queramos desplazar al texto.

Para **mode = 2** y **mode = 3**, el valor numérico en la celda "**Actor**" debe positivo.

Ejemplo:

En **mode = 2**, el texto se desplazará tomando como punto de partida el inicio de la **shape**:

Hiragana			✱も✱と✱め ✱あ✱う ✱こ✱こ✱ろ ✱
English	100		✱Mis mejillas se manchan con lágr
English			Pero puedo sentir la llegada del ar

Ejemplo:

En **mode = 3**, el texto se desplazará tomando como punto de partida el final de la **shape**:

Hiragana			✱も✱と✱め ✱あ✱う ✱こ✱こ✱ろ ✱
English	125		✱Mis mejillas se manchan con lágr
English			Pero puedo sentir la llegada del ar

En los anteriores ejemplos se comentaba que en **mode = 1**, el valor numérico en la celda "**Actor**" determinada hacia dónde se desplazaba el texto; que si este valor era positivo, se desplazaría hacia la derecha y en caso contrario, hacia la izquierda. Una explicación más precisa de ese fenómeno es la siguiente:

mode = 1

- Si el valor numérico de la celda "**Actor**" es positivo, el texto se desplazará hacia el inicio de la **shape**, hacia su primer punto.
- Si el valor numérico de la celda "**Actor**" es negativo, el texto se desplazará hacia el final de la **shape**, hacia su último punto.

mode = 2

El valor numérico de la celda "**Actor**" debe ser un número positivo, entonces el texto se desplazará desde el inicio de la **shape**, hacia su último punto.

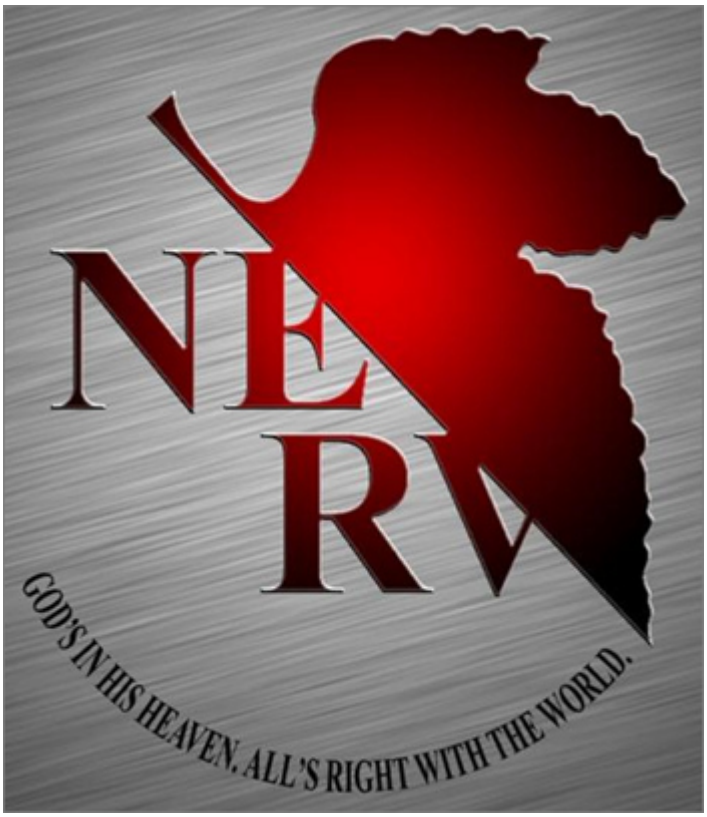
mode = 3

El valor numérico de la celda "**Actor**" debe ser un número positivo, entonces el texto se desplazará desde el final de la **shape**, hacia su primer punto.

Y para **mode = 4**, que es hasta donde vimos en los ejemplos anteriores, el valor numérico de la celda "**Actor**", no surte efecto alguno, no hace que el texto se desplace.

Para la siguiente habilidad de la función **text.bezier**
usaré un logo en donde aparece texto en forma circular:

Ejemplo:



Primero modificamos el estilo del texto hasta que creamos que ya está acorde con el del vídeo:

Seleccionamos la herramienta para recortar subtítulos en un área vectorial y seleccionamos la herramienta para hacer rectas con ella.

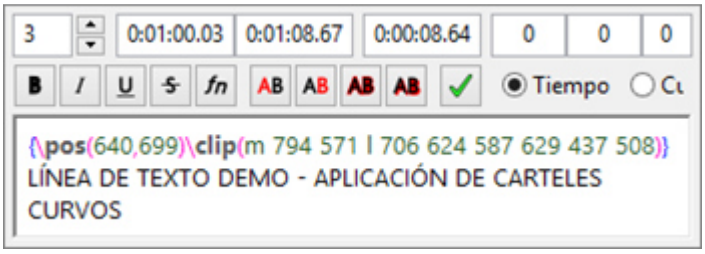
Debemos poner cuatro puntos de forma que cada uno de ellos quede en la misma circunferencia imaginaria que describe el texto original:

No importa en donde pongamos a cada uno de los puntos, pero si podemos conservar una distancia prudente entre ellos es más conveniente:

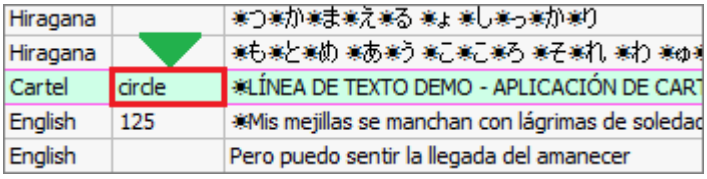
No es obligatorio que los puntos cumplan una secuencia en específico, pero sí debemos tratar que todos queden en la circunferencia del círculo:

El único punto es opcional, pero de estar, debe cumplir una simple condición, lo debemos poner en donde necesitemos que nuestro texto empiece:

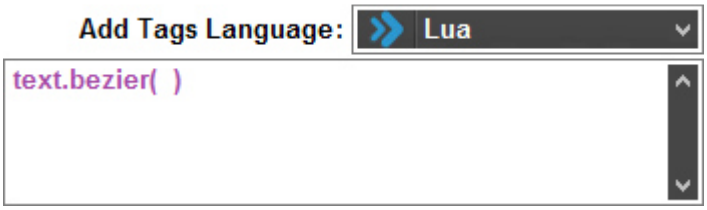
Hecho esto, nuestra línea en el script debe verse más o menos como en la siguiente imagen:



Ahora debemos hacer es escribir la palabra “circle” en la celda “**Actor**“, y si ya pone algo en esta celda, simplemente ponemos un espacio al final y la escribimos:



Y llamamos a nuestra función en **Add Tags**, sin ponerle la **shape**, para que tome los puntos creados en el clip:



Al aplicar, veremos cómo el texto adopta la forma espera:

Lo malo es que se ve encima del texto original del vídeo, pero cumple con los detalles que le ingresamos:

- La circunferencia imaginaria que describe el texto pasa justo pos los tres primeros puntos trazados en el clip.
- El texto inicia justo en donde pusimos el cuarto punto del clip.

Para remediar el hecho de que el texto generado se ve justo encima del texto original, debemos modificar el radio del círculo que lo genera, para ello debemos escribir en la celda “Efecto” del Aegisub la cantidad en pixeles que necesitemos se aumente o se reduzca el radio. Si la cantidad es positiva se aumenta, en caso contrario, se disminuye:

Ejemplo:

English			Me aferraré a ti y nunca te soltaré
English			Dos corazones que se buscan con
Cartel	circle	38	✱LÍNEA DE TEXTO DEMO - APLIC/

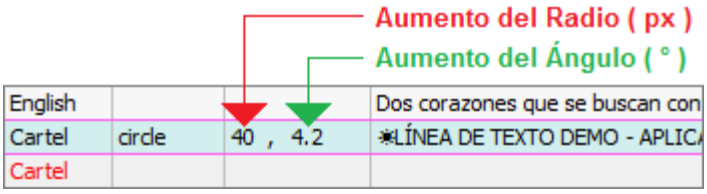
Y al aplicar:

Quizá 30 px sea muy poco, pero eso ya depende del gusto de cada uno.

Al poner el cuarto punto en el clip dibujado, éste forma un ángulo con el centro del círculo:

Este ángulo que se forma con el centro del círculo y el cuarto punto del clip que señala el punto de inicio del texto, lo podemos modificar añadiendo o restando un valor en grados, con un segundo valor en la celda “Efecto”.

Ejemplo:



En la anterior imagen, los dos números están separados por una coma, por no es obligatorio ponerla, con que estén separados por un espacio es suficiente.

Ahora aplicamos y se verán las diferencias:

Hicimos que el texto se desplazara 4.2° positivamente, es decir en forma inversa al movimiento de las manecillas del reloj.

Si ajustamos el radio y le restamos los ángulos necesarios, podemos hacer otra variante del mismo cartel.

Ejemplo:

- **Radio + 10 px**
- **Ángulo – 120°**

English			Me aferraré a ti y nunca te soltaré
English			Dos corazones que se buscan conforr
Cartel	circle	10 -120	☀LÍNEA DE TEXTO DEMO - APLICACI(
Cartel			

Estos dos valores salieron luego de un par de intentos a ensayo y error, y obviamente los valores serán diferentes dependiendo del cartel que estemos haciendo:

Siempre que pongamos en "**Actor**" la palabra "circle", el texto se posicionará en la circunferencia en sentido inverso a la de las manecillas del reloj, como lo indica la flecha de la anterior imagen.

Si queremos que el texto se escriba en sentido horario, debemos escribir en “**Actor**” la palabra “icircle”:

Ejemplo:

English				Me aferraré a ti y nunca te soltaré
English				Dos corazones que se buscan conforma
Cartel	icircle	10	-3	✱LÍNEA DE TEXTO DEMO - APLICACIÓN
Cartel				

Ahora el texto está en el mismo sentido del movimiento de las manecillas del reloj:

Con “circle” e “icircle” en la celda de texto “**Actor**” y los dos valores que podemos usar en la celda “**Efecto**“, podemos lograr cualquier tipo de cartel que tenga forma circular. De aquí en adelante todo dependerá de la práctica:

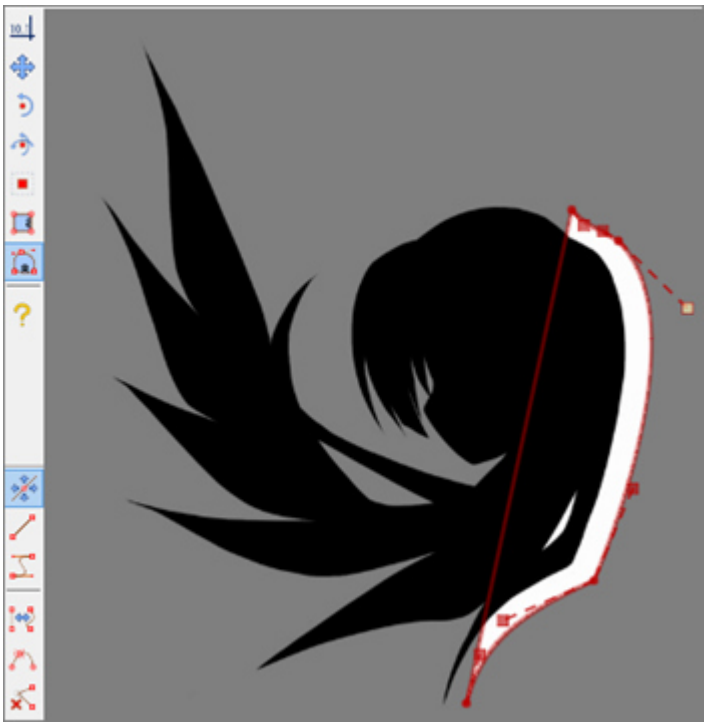
Ejemplo:

Otro ejemplo, este a dos colores:

Ejemplo:

Para los carteles que no sean perfectamente circulares o que tengan otra forma libre, entonces dibujamos el contorno deseado con el clip:

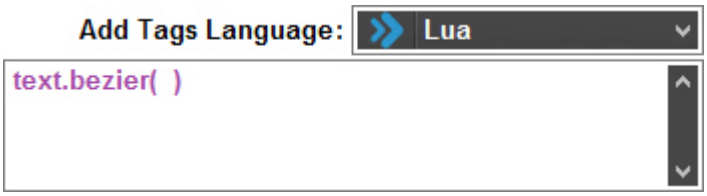
Ejemplo:



Y ya no es necesario escribir ni “circle” ni “icircle” en la celda “**Actor**“, solo necesitamos el clip:

English		Me aferraré a ti y nunca te soltaré
English		Dos corazones que se buscan conforman este su
Cartel		☀LÍNEA DE TEXTO DEMO - APLICACIÓN CARTEL

Con un **Template Type: Char**, ponemos en **Add Tags**:



Y los resultados no podrían ser mejores:

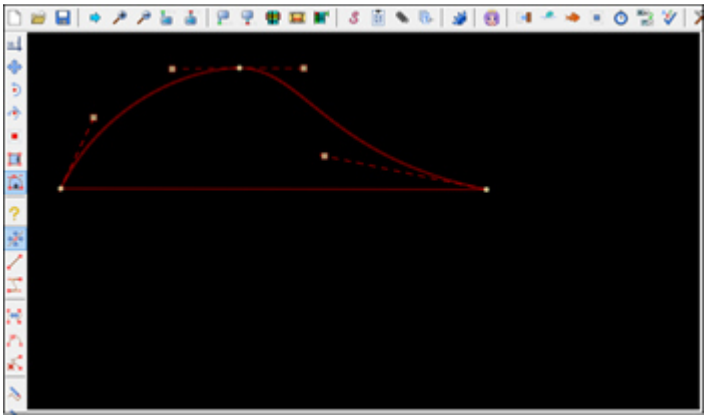
Cuando de hacer carteles y logos se trata, las posibilidades son muchas, pero de a poco vamos develando los secretos de las diferentes posiciones y rotaciones que el texto puede tener en ellos, pero hasta este punto solo hemos visto la forma de posicionar el texto de forma estática, no con un movimiento o desplazamiento respecto al tiempo. A continuación veremos carteles con dichas características.

Una vez dominado el arte de posicionar el texto en cualquier forma no lineal, ahora veremos varios métodos de cómo darle movimiento a dicho texto y así poder ampliar nuestras posibilidades cuando desarrollamos un efecto.

Ya hemos visto varios ejemplos de cuando el parámetro **Shape** es el código de una **shape** normal o una de las Shapes por default del **KE**, también vimos ejemplos con clip’s como remplazo de la **shape** o como marcador de puntos para la ubicación del texto en la circunferencia de un círculo.

Ahora veremos cuando el parámetro **Shape** es una **tabla** de Shapes.

Ejemplo:

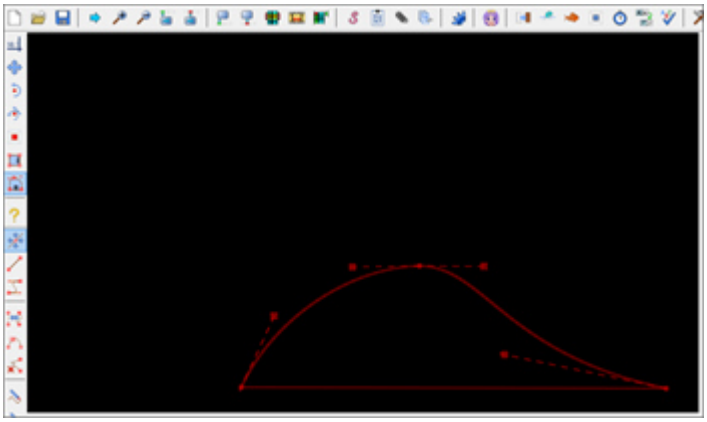


Empezamos creando la primera **shape**, la inicial, y con el código creamos una tabla en la celda "**Variables**":

Variables:

```
mi_shape = {
        [ 1 ] = "m 64 298 b 127 163 276 70 404 68 527
        69 566 236 874 300"
}
```

Luego ubicamos la segunda **shape**, que para este ejemplo será la **shape** de destino:



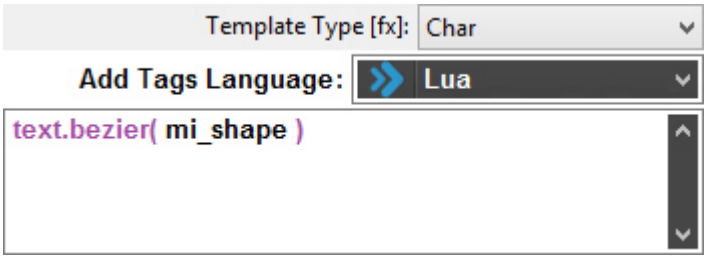
Entonces la **tabla** de Shapes se verá de la siguiente forma:

La idea es poner una **shape** que indique la posición inicial del texto y otra para la posición final.

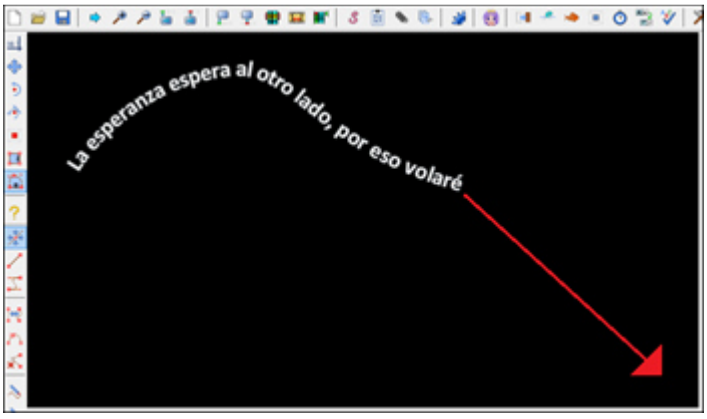
Cuando la **tabla** de Shapes tiene 2 elementos, la función retorna un tag **\move**. Como ya lo había mencionado antes, la **tabla** debe tener entre 2 y 4 Shapes, lo que hace que la función retorne los siguientes tags:

- **2 shapes: \move**
- **3 shapes: \moves3**
- **4 shapes: \moves4**

O sea que para más de 2 Shapes es necesario tener en el **Aegisub** al **VSFilterMod**. En este ejemplo solo usaremos 2, lo que nos debe retornar un tag **\move**:



Al aplicar, el texto inicia en la posición de la primera **shape** y a medida que trancurre el tiempo de la duración total de la linea fx, se va moviendo hacia la posición de la segunda **shape**:



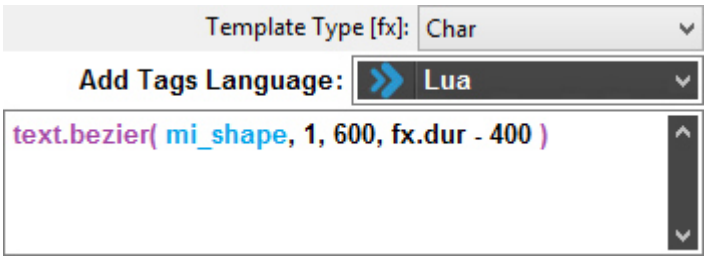
Para modificar los tiempos de inicio y final del movimiento del texto generado, debemos entender la estructura que toma la función **text.bezier**
cuando el parámetro **Shape** es una **tabla**:

mi_shape = { shape1, shape2, … }

text.bezier(mi_shape, mode, time_i, time, f)

El parámetro **mode** es un entero entre 1 y 4, **time_i** es el tiempo de inicio del movimiento y su valor por default es 0, y **time_fx** es el tiempo final del movimiento, donde su valor por default es **fx.dur**

Ejemplo:

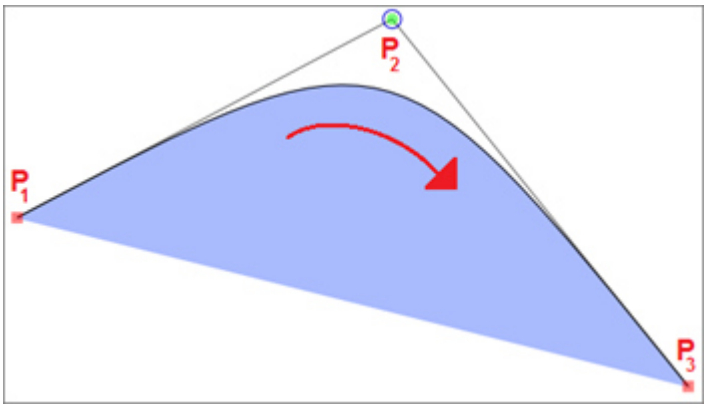


- **mi_shape = tabla** de Shapes
- **mode = 1**: el texto queda alineado en el centro
- **time_i = 600**: el movimiento del texto empezará 600 ms luego de haber iniciado la línea fx.
- **time_f = fx.dur – 400**: el movimiento terminará 400 ms antes de que termine la línea fx.

Recordemos que para una **tabla** de dos Shapes, retorna un tag **\move**. Para más Shapes tenemos lo siguiente:

Para una **tabla** de tres Shapes, obtenemos un **\moves3**, y el movimiento tendría la siguiente particularidad:

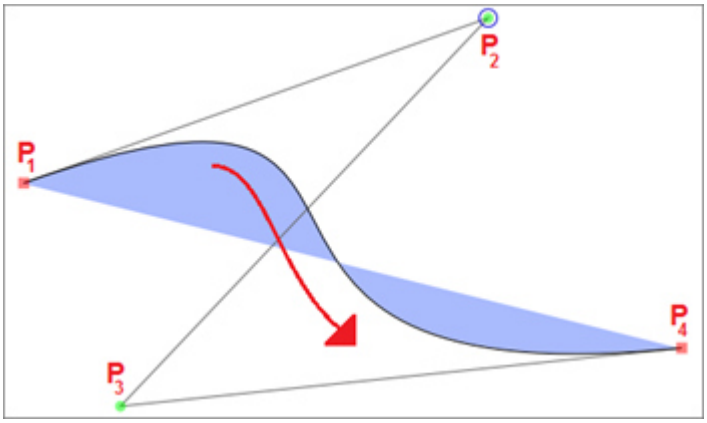
Ejemplo:



Un **\moves3** no hace un movimiento lineal entres los tres puntos ingresados, sino que hace una trayectoria **Bézier** entre ellos, como en la imagen anterior que se hace una curva entre los tres puntos.

Para una **tabla** de cuatro Shapes, obtenemos un **\moves4**, y el movimiento tendría la siguiente particularidad:

Ejemplo:



En el **\moves4** tampoco el movimiento es lineal entre sus puntos, sino que traza una curva **Bézier** entre sus cuatro puntos.

Sea cual sea la cantidad de Shapes que haya en nuestra **tabla**, no se nos debe olvidar tener en cuenta que cada una de ellas debe tener una longitud igual o mayor que el ancho de la línea a la que le apliquemos un efecto: **line.width**

Visto estos recientes ejemplos, solo nos resta por ver a los dos últimos valores del parámetro **mode** que son:

- **mode = 5**
- **mode = 6**

Estos dos valores de **mode** son válidos para cuando el parámetro **Shape** no es una **tabla**, o sea que es el código de una shape convencional, una shape por default del **KE** o un clip dibujado en la línea del script.

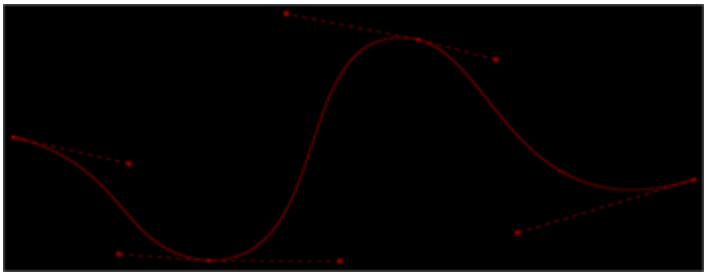
mode = 5

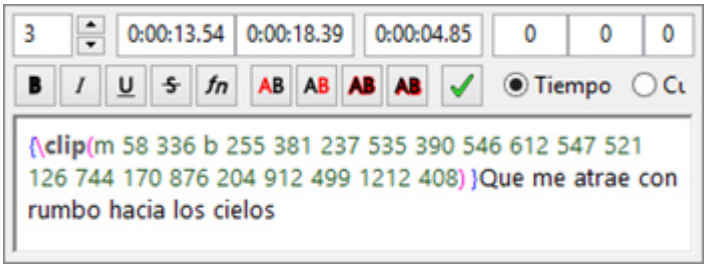
Crea una secuencia del texto y lo mueve respecto avanza el tiempo de la duración de la línea fx, desde su posición en **mode = 2** hasta **mode = 3**, o sea, desde el inicio de la **shape** hasta el final de la misma.

Lo que la función hace tanto en **mode = 5** y **mode = 6**, es una animación cuadro a cuadro de las diferentes posiciones del texto para dar la impresión de que éste se mueve a lo largo de la longitud total de la **shape** ingresada.

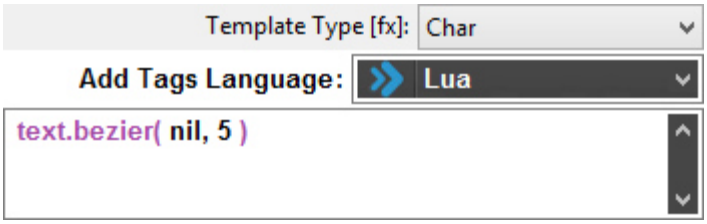
Ejemplo:

Dibujamos un clip vectorial en una de las líneas del script, asegurándonos de que éste sea más largo que la longitud del ancho de la línea. (**line.width**):

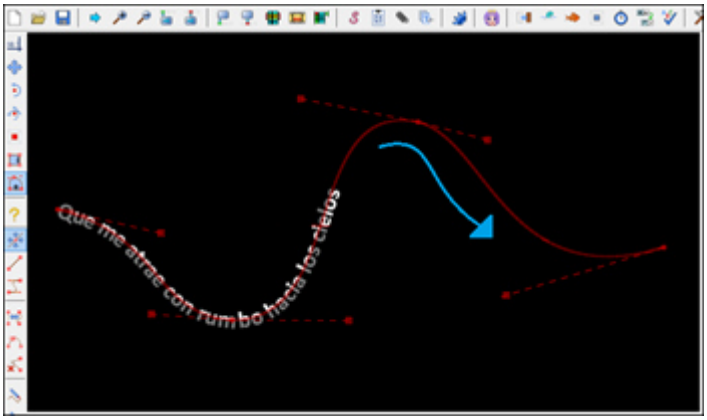




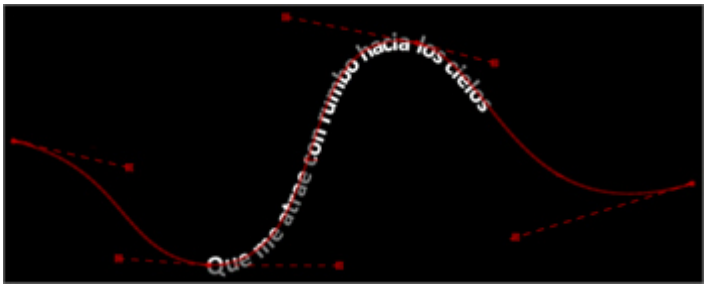
Hecho esto, ponemos:



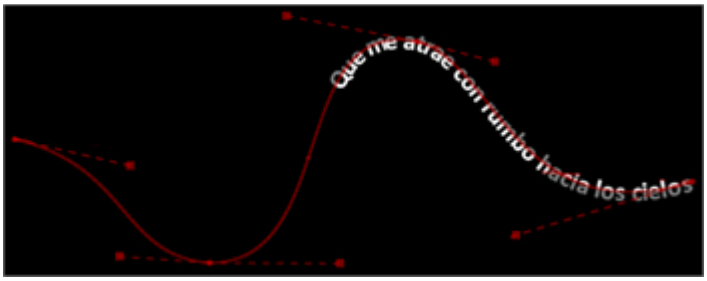
Y al aplicar veremos algo como esto (sin las curvas):



La primera posición del texto es en el inicio de la **shape**, y conforme avanza el tiempo, se irá desplazando a lo largo de todo el clip vectorial:



Así, hasta llegar al final del clip vectorial, que equivale al punto final de la **shape**:



La función crea el efecto cuadro por cuadro, y por default cada uno de ellos tiene la misma duración:

frame_dur = 41.708 ms

0:00:40.70	0:00:45.79	English			Me aferra
0:00:45.92	0:00:54.56	English			Dos coraz
0:00:13.54	0:00:13.58	English	lead-in	Effector [Fx]	*Q
0:00:13.58	0:00:13.62	English	lead-in	Effector [Fx]	*Q
0:00:13.62	0:00:13.66	English	lead-in	Effector [Fx]	*Q
0:00:13.66	0:00:13.70	English	lead-in	Effector [Fx]	*Q
0:00:13.70	0:00:13.74	English	lead-in	Effector [Fx]	*Q

fx.dur = frame_dur

Al usar la función en **mode = 5** o **mode = 6**, podemos usar los otros dos parámetros restantes, así:

text.bezier(Shape, mode, Accel, Offset_time)

El parámetro **Accel** hace referencia a la aceleración del texto al moverse, al cambio de velocidad, y su valor por default es 1. Para valores mayores a 1, el texto acelera positivamente, en caso contrario, desacelera.

Para valores menores que 1, recomiendo uno entre 0.2 y 0.9; y para valores mayores que 1, entre 1.1 y 2.6

La elección de algunos de estos valores para la aceleración del texto dependerá de lo que estemos necesitando y es más algo de prueba y error que de una ciencia exacta.

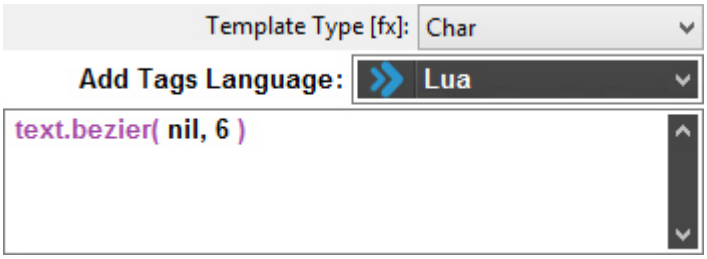
El parámetro **Offset_time** hace referencia a un tiempo medido en milisegundos (**ms**) que se añadirá o restará de la duración por default (**41.708 ms**) de cada uno de los cuadros generados por la función. Su valor por default es 0.

mode = 6

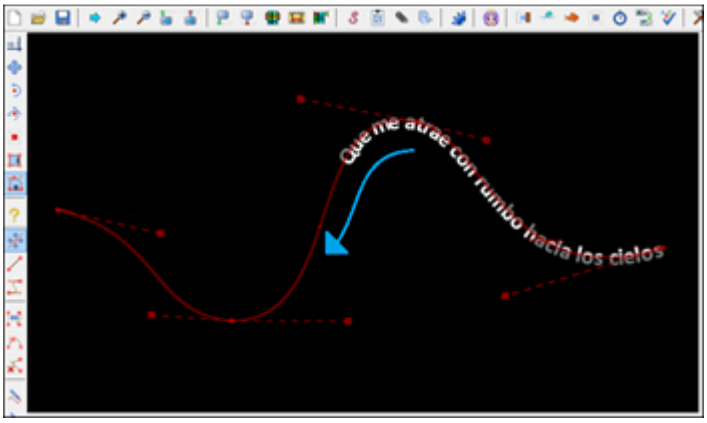
Crea una secuencia del texto y lo mueve respecto avanza el tiempo de la duración de la línea fx, desde su posición en **mode = 3** hasta **mode = 2**, o sea, desde el final de la **shape** hasta el inicio de la misma.

En este modo, la función hace que el texto se desplace de forma inversa a como lo hace con el modo anterior:

Ejemplo:



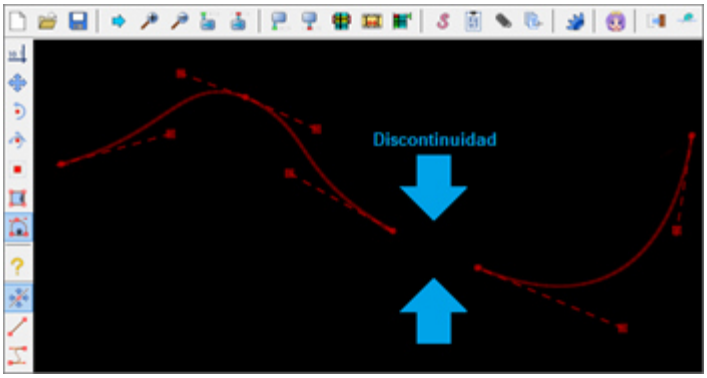
Lo que hará que el texto inicie en la parte final del clip y se vaya desplazando en función del tiempo, hacia el inicio del mismo, de forma inversa que en **mode = 5**:



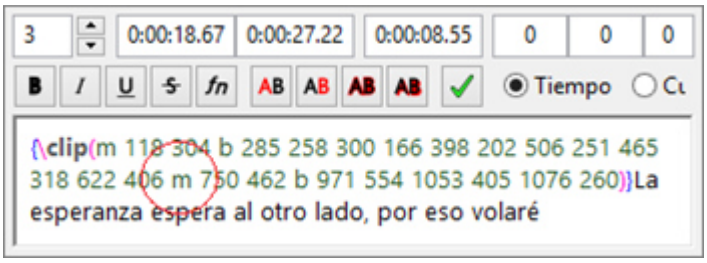
Hasta este punto, todo los ejemplos que hemos visto, han sido basado en Shapes continuas, de un solo trazo, pero la función también puede aplicarse con Shapes que no lo son.

Ejemplo:

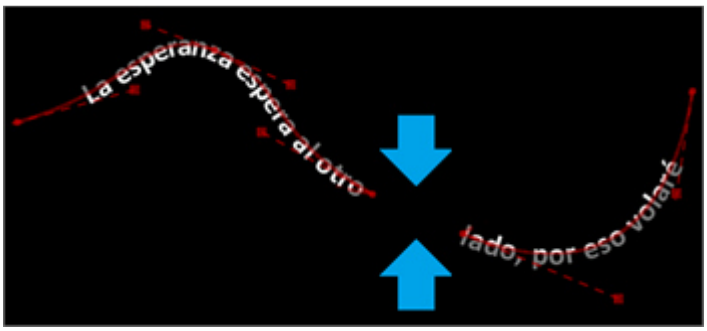
Dibujamos una **shape** o un clip vectorial que contenga una o más discontinuidades, que sea algo notoria, como en la siguiente imagen:



En la línea del script veremos algo parecido a esto, ya que el clip vectorial está conformado por dos Shapes adheridas, una separada de la otra:



Al aplicar, el texto que no alcance a quedar en un tramo de la **shape**, saldrá en la parte restante:



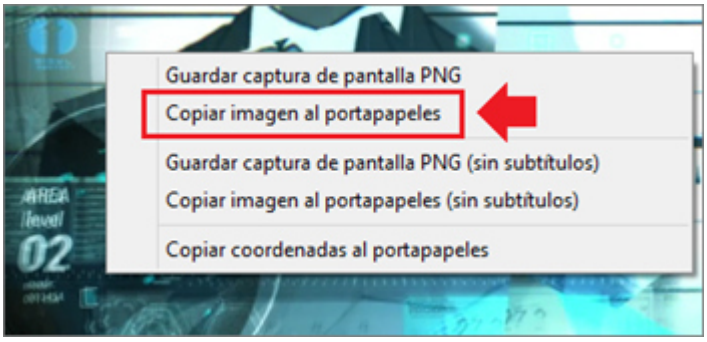
Visto sin el clip vectorial en pantalla:

Habrá ocasiones que el clip vectorial que usaremos como **shape** para la función, se extienda mucho más allá de los límites del vídeo, como la curva resaltada a continuación:

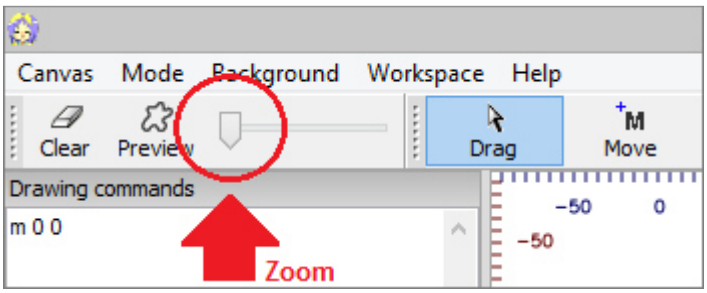
Ejemplo:

Entonces, uno de los trucos que podemos usar para poder dibujar esa parte de la **shape** que no se ve, es darle clic derecho sobre el vídeo, en el **Aegisub**:

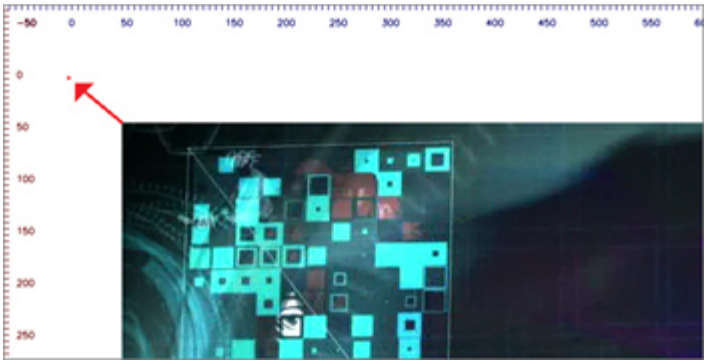
Y elegimos la opción de "**Copiar imagen al portapapeles**", lo que creará una captura instantánea del vídeo con sus dimensiones reales:



Luego abrimos el **ASSDraw3** y ponemos el zoom al mínimo, antes de pegar la captura del vídeo:



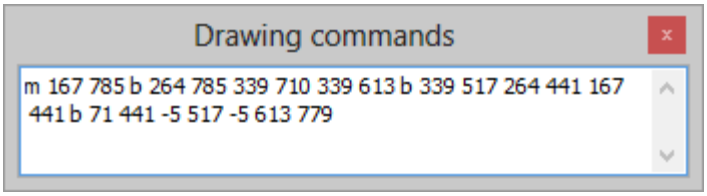
Pegamos la imagen que habíamos capturado del vídeo, que posteriormente tendremos que desplazar, de modo que la esquina superior izquierda de la captura quede justo en las coordenadas (0, 0):



Hecho esto, desplazamos el lienzo de trabajo hasta que nos quede cómo para trazar la curva que deseamos extender por fuera de los límites del vídeo:

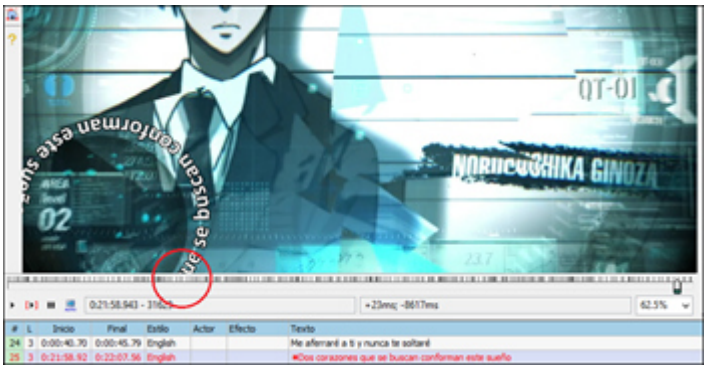
Ahora ya podemos trazar libremente la **shape** y superar los límites del vídeo, hasta dos necesitemos:

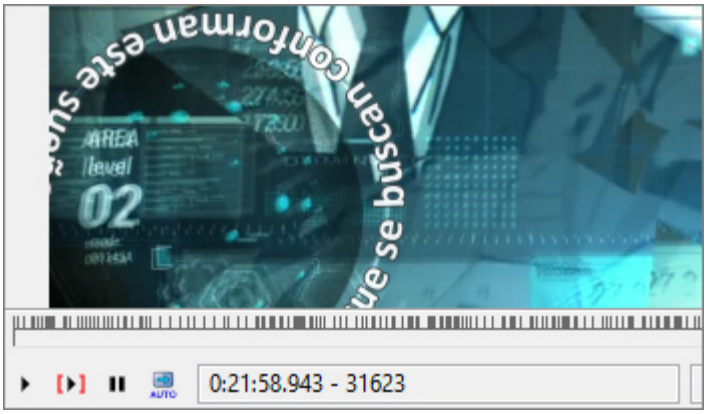
Ya podemos tomar el código de la **shape** y usarlo entre comillas, ya sean simples o dobles, para definir una variable en la celda de texto "**Variable**" o usarlo directamente como el primer parámetro de la función:



O si queremos, usamos el código de la **shape** recién creada y lo pegamos dentro de un clip vectorial, para que la función solo afecte a la línea o líneas que contengan a dicho clip:

Y al aplicar, notamos cómo el texto está ligeramente por fuera de los límites del vídeo gracias a la extensión que le hicimos al trazar la shape. Así, al darle movimiento creamos el efecto de que el texto va apareciendo:





La función **text.bezier** retorna dos tags, uno de posición y otro de rotación:

Ejemplo:

El tag de posición retornado depende del modo en que usemos la función, y las cuatro opciones son: **\pos, \move, \moves3** y **\moves4**. Y el tag de rotación siempre será el **\fr** que es el mismo **\frz** (**font rotation in axis "z"**).

Con este ejemplo damos por terminada la documentación de la función **text.bezier**, que como ya habrán notado, tiene muchas aplicaciones que espero hayan sido de su agrado. No descarto más adelante agregarle más modos y opciones para ampliar aún más esas posibilidades.

¡Muchas gracias por seguirnos!