Kara Effector 3.2:
Effector Book
Vol. II [Tomo XXXIV]

Kara Effector 3.2:

En este **Tomo XXXIV** continuaremos viendo más de los Recursos disponibles en el **Kara Effector**, que espero que con la ayuda de esta documentación, le puedan sacar el máximo provecho a la hora de llevar a cabo sus proyectos, no solo karaokes, sino también para la edición de las líneas de subtítulos.

Recursos [KE]:

En el **Tomo XXXIII** vimos la forma de posicionar el texto de forma estática con la función **text.bezier**, y vimos dos de las tres maneras de agregar una **shape** a la función, al igual que cuatro de los seis valores del parámetro **mode**. Es el momento de seguir en dónde nos habíamos quedado en el **Tomo** anterior.

Recursos [KE]: → Actualización

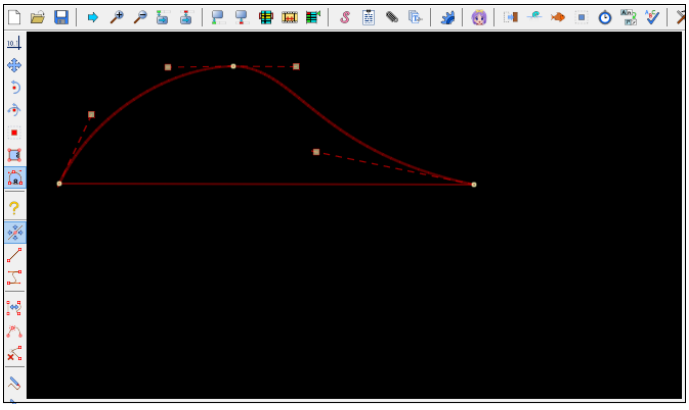
text.bezier(Shape, mode, Accel, Offset_time)

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:

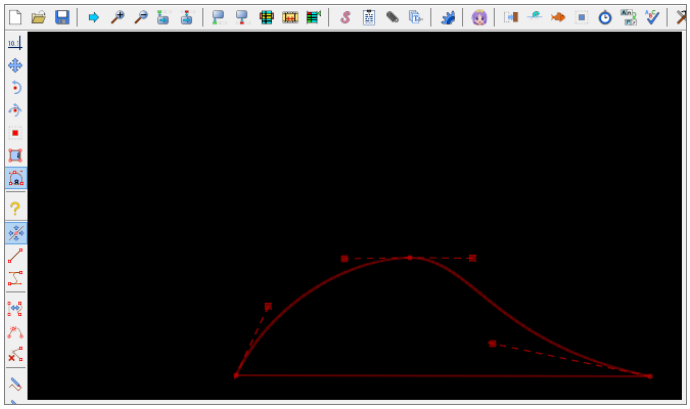


Empezamos creando la primera **shape**, la inicial, y con el código creamos una tabla en la celda "**Variables**":

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:

Variables:

```
mi_shape = {
        [1] = "m 64 298 b 127 163 276 70 404 68 527
        69 566 236 874 300",
        [2] = "m 408 672 b 471 537 620 444 748 442
        871 443 910 610 1218 674"
}
```

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**

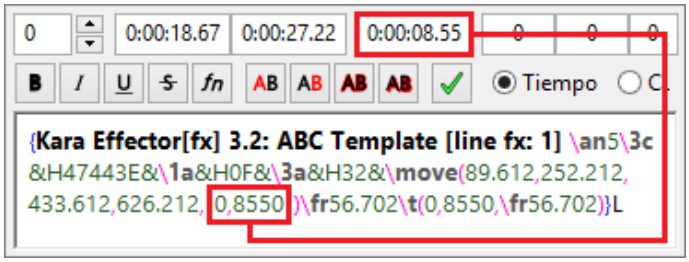
O sea que para más de 2 Shapes es necesario tener en el **Aegisub** al **VSFilterMod**. En este ejemplo solo usaremos 2, lo que nos debe retornar un tag **\move**:

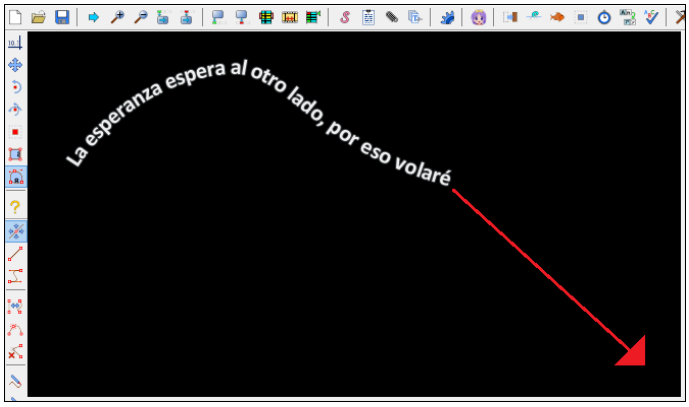
Template Type [fx]: Char

Add Tags Language: Lua

```
text.bezier( mi_shape )
```

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:

Template Type [fx]: Char

Add Tags Language: Lua

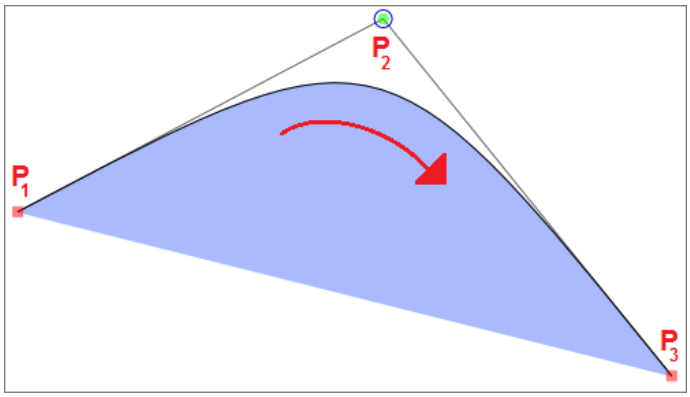
```
text.bezier( mi_shape, 1, 600, fx.dur - 400 )
```

- **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:

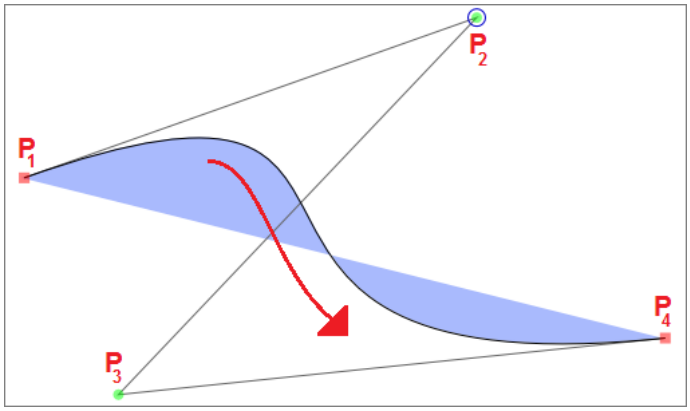
Ejemplo:



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

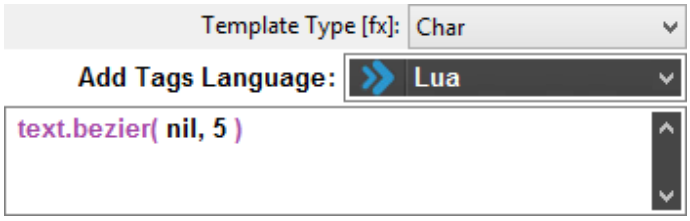
Crea una secuencia del texto y lo mueve respecto avanza el tiempo de la duración de la línea fx, desde su posición en **mode = 2** hasta **mode = 3**, o sea, desde el inicio de la **shape** hasta el final de la misma.

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.

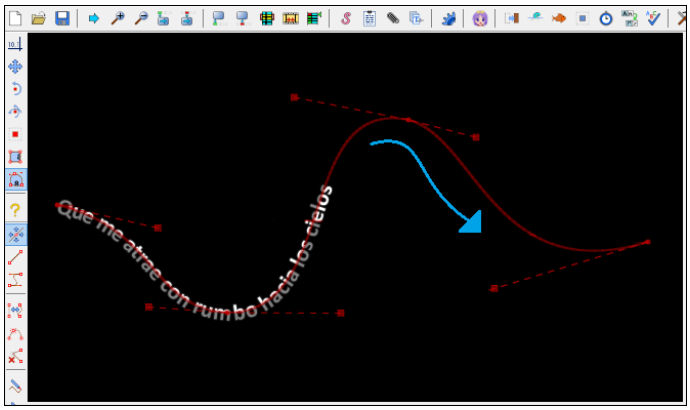
Ejemplo:

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**):

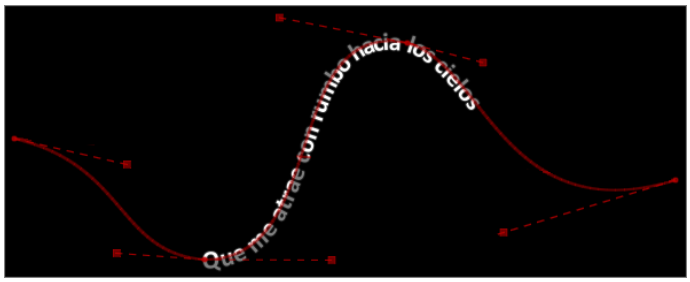
Hecho esto, ponemos:



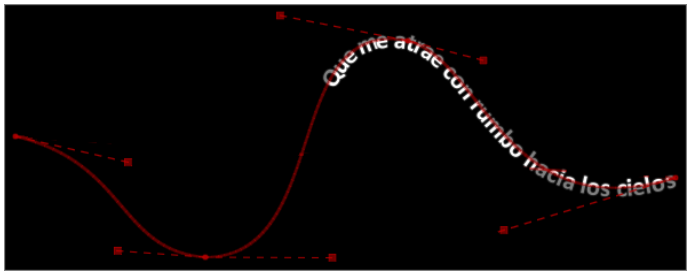
Y al aplicar veremos algo como esto (sin las curvas):



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:

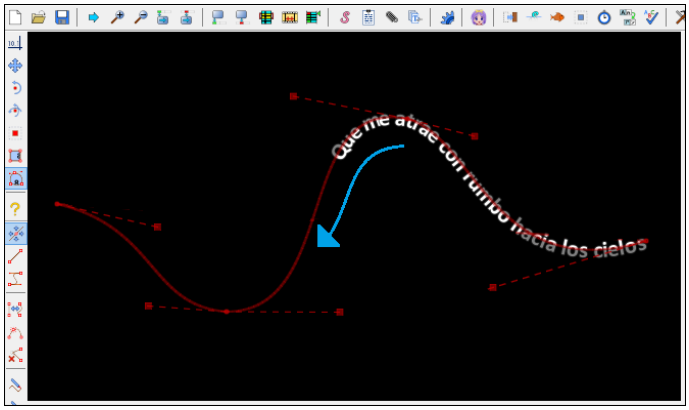
Ejemplo:

Template Type [fx]: Char

Add Tags Language: Lua

```
text.bezier( nil, 6 )
```

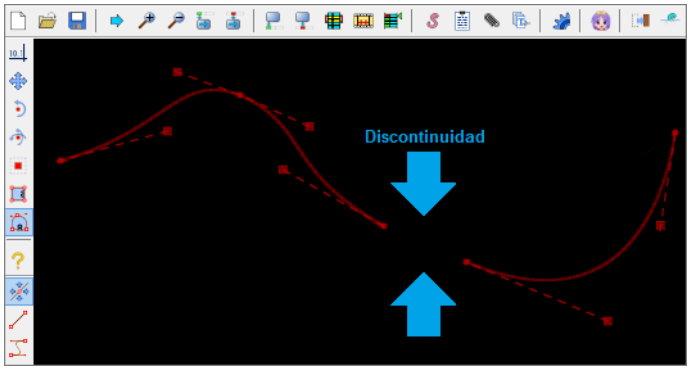
Lo que hará que el texto inicie en la parte final del clip y se vaya desplazando en función del tiempo, hacia el inicio del mismo, de forma inversa que en **mode = 5**:



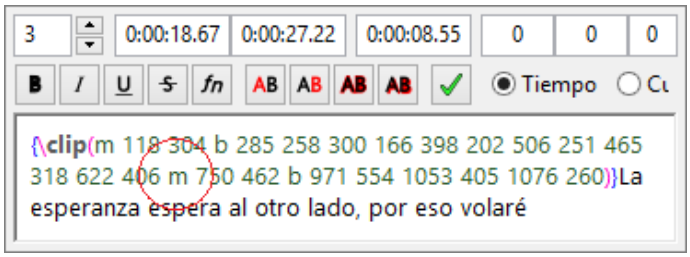
Hasta este punto, todo los ejemplos que hemos visto, han sido basado en Shapes continuas, de un solo trazo, pero la función también puede aplicarse con Shapes que no lo son.

Ejemplo:

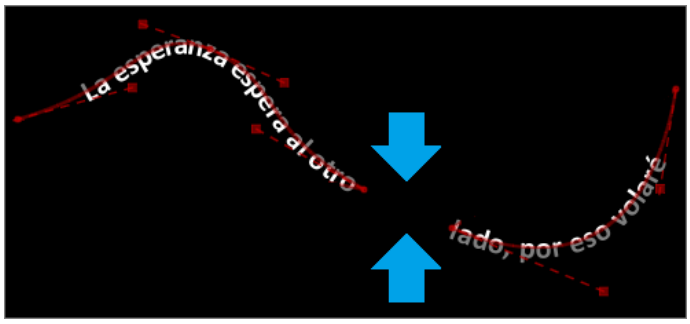
Dibujamos una **shape** o un clip vectorial que contenga una o más discontinuidades, que sea algo notoria, como en la siguiente imagen:



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:



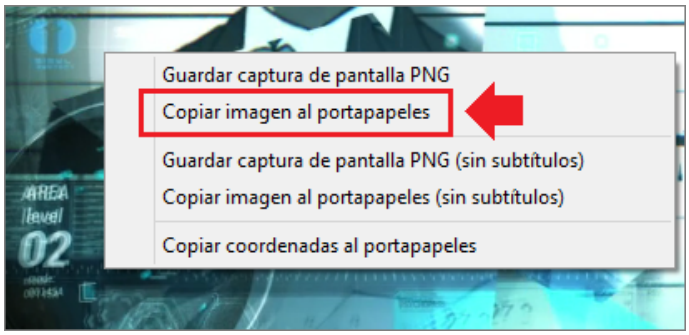
Visto sin el clip vectorial en pantalla:

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:

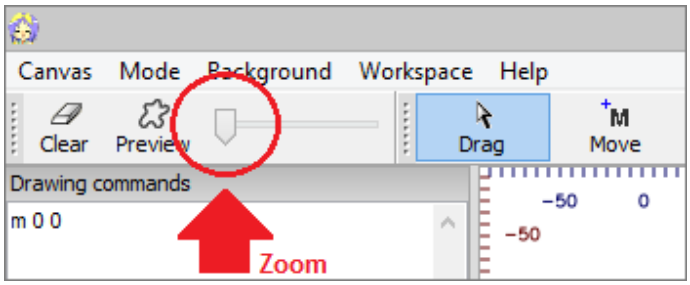
Ejemplo:

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**:

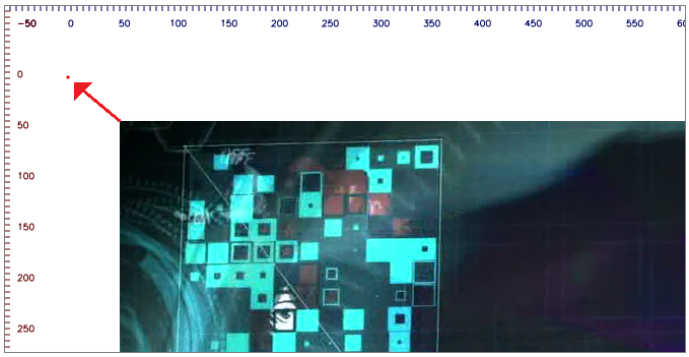
Y elegimos la opción de "**Copiar imagen al portapapeles**", lo que creará una captura instantánea del vídeo con sus dimensiones reales:



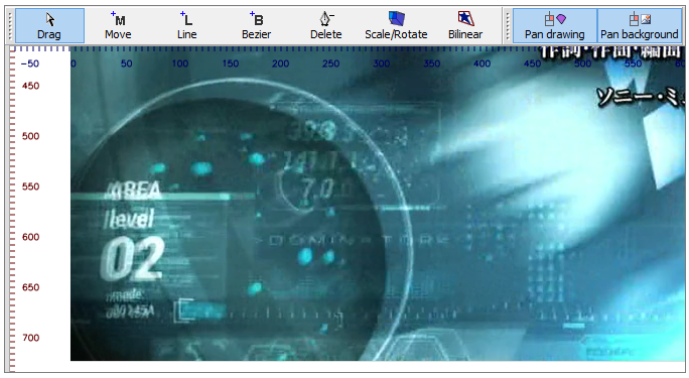
Luego abrimos el **ASSDraw3** y ponemos el zoom al mínimo, antes de pegar la captura del vídeo:



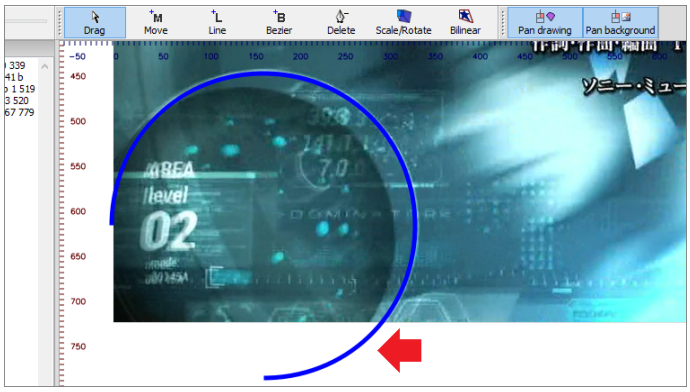
Pegamos la imagen que habíamos capturado del vídeo, que posteriormente tendremos que desplazar, de modo que la esquina superior izquierda de la captura quede justo en las coordenadas (0, 0):



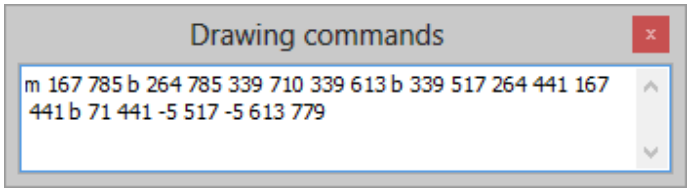
Hecho esto, desplazamos el lienzo de trabajo hasta que nos quede cómo para trazar la curva que deseamos extender por fuera de los límites del vídeo:



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

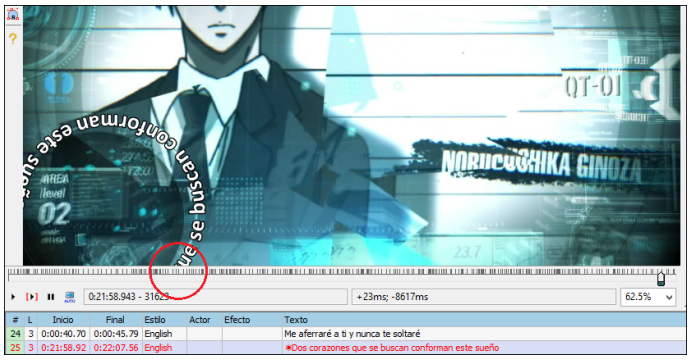


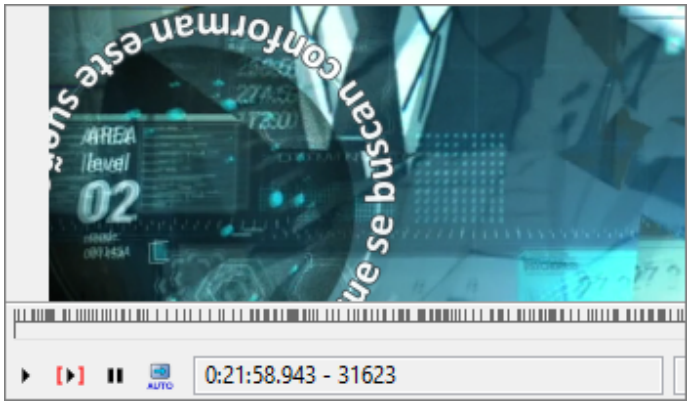
Ya podemos tomar el código de la **shape** y usarlo entre comillas, ya sean simples o dobles, para definir una variable en la celda de texto "**Variable**" o usarlo directamente como el primer parámetro de la función:



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:

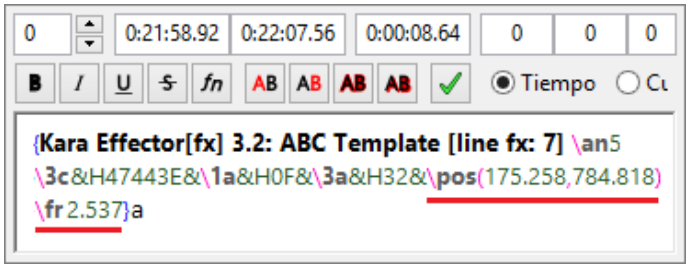
Y al aplicar, notamos cómo el texto está ligeramente por fuera de los límites del vídeo gracias a la extensión que le hicimos al trazar la shape. Así, al darle movimiento creamos el efecto de que el texto va apareciendo:





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.

Es todo por ahora para el **Tomo XXXIV**. Intenten poner en práctica todos los ejemplos vistos 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. Pueden visitarnos y dejar su comentario en nuestra página de **Facebook**:

- www.karaeffector.blogspot.com
- www.facebook.com/karaeffector
- www.youtube.com/user/victor8607
- www.youtube.com/user/NatsuoKE
- www.youtube.com/user/karalaura2012