

Librería “random” [KE]

Es una librería exclusiva del **Kara Effector** que consiste en una serie de funciones con resultados aleatorios que podemos aplicar en el desarrollo de los Efectos. La Librería “**random**” contiene las siguientes funciones:

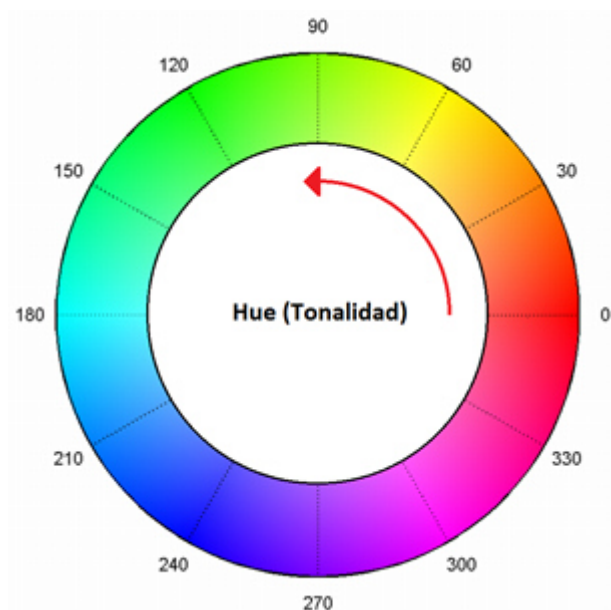
- **random.color**
- **random.colorvc**
- **random.alpha**
- **random.alphava**
- **random.e**
- **random.unique**

Y a continuación veremos en detalle en qué consiste cada una de ellas.

random.color(H, S, V): retorna un color en formato .ass según la teoría HSV. **H**, **S** y **V** son parámetros opcionales y pueden ser valores numéricos o tablas.

Modo 1: sin ningún parámetro

random.color(): retorna un color en formato .ass correspondiente a un tono (**Hue**) al azar entre 0 y 360 de la teoría **HSV**.



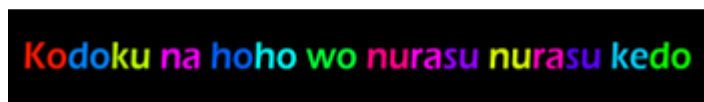
En este modo (Modo 1), la función retorna un tono al azar, pero tanto **S (Saturation)** como **V (Value)** son constantes y el valor de cada uno de ellos es 100, o sea que el color será solo alguno de los de la imagen anterior.

En el **Kara Effector**, en un Efecto tipo “**Syl**“, podemos poner a prueba el Modo 1 de esta función:

Add Tags: Add Tags Language: Automation Auto-4 ▼

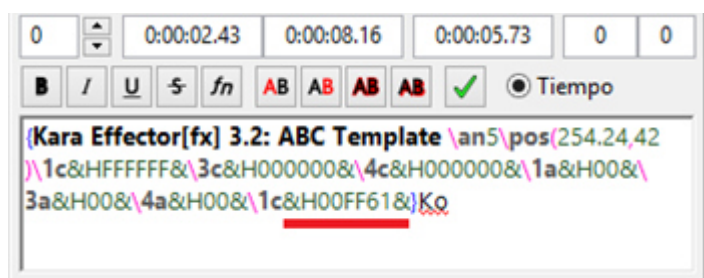
```
\1crandom.color(!
```

Entonces en el vídeo veremos algo como esto:

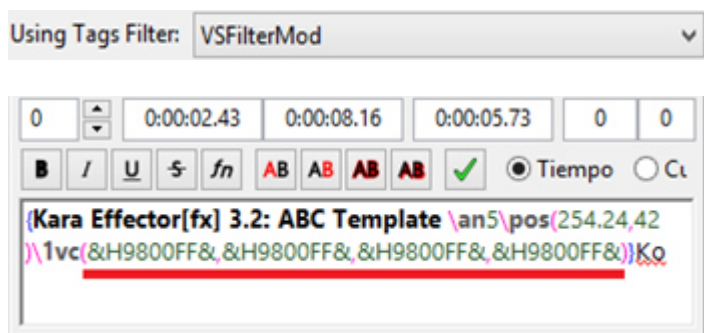


Para cada Sílabla la función generó un color aleatorio con un valor entre 0 y 360 del anterior círculo cromático.

Este es un ejemplo del color que puede retornar la función en este modo, siempre y cuando hayamos seleccionado la opción “**VSFilter 2.39**” o la opción “**No Tags Color and Alpha**” en **Using Tag Filter:**



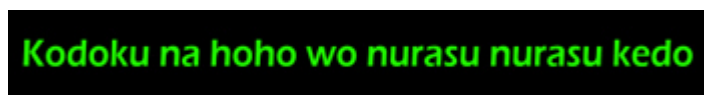
Si por el contrario, elegimos la opción “**VSFilterMod**“, el color se multiplicará cuatro veces y saldrán separados por comas dentro de un paréntesis (en formato **VSFilterMod**):



Modo 2: con un parámetro constante

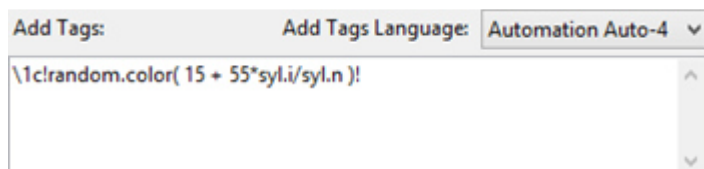
random.color(H): retorna un color en formato .ass correspondiente al tono (**Hue**) entre 0 y 360 que le hayamos asignado a la función. Ejemplo:

random.color(112) = “&H00FF21&”

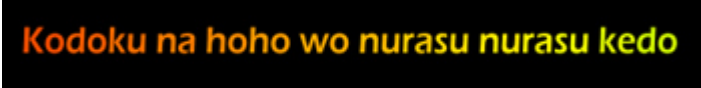


En el círculo cromático vemos que el valor 112 le corresponde a un todo de verde claro. En este modo, la función retorna un color constante correspondiente al **Hue** del valor que le ingresemos.

Usar la función **random.color** para que retorne un color constante es ideal para hacer degradaciones con los colores **HSV**. Ejemplo: (**Template Type:** Syl)



Es un degradado **HSV** con los valores desde 15 para la primera Sílabas, hasta 70 ($15 + 55 = 70$) para la última:



Kodoku na hoho wo nurasu nurasu kedo

Modo 3: con una Tabla como parámetro

random.color({H_i, H_f}): retorna un color en formato .ass correspondiente a un valor aleatorio entre **H_i** y **H_f**. Ejemplo:

random.color({270, 320}) = “&HFF00AA&”



Kodoku na hoho wo nurasu nurasu kedo

Vemos que el tono (**Hue**) que resulta corresponde a un valor aleatorio entre 270 y 320 del círculo cromático **HSV**. O sea que cuando algún parámetro es una Tabla, entonces la función hace un Random entre los dos valores de la Tabla, cada vez que se ejecute la función.

Si combinamos los tres Modos de la función **random.color** (ausencia de uno o más de sus parámetros, uno o más parámetros constantes y por último, que uno o más de los parámetros sean una Tabla), obtenemos todo un mundo nuevo de posibilidades (15 en total):

- **random.color()**
- **random.color(H)**
- **random.color({H_i, H_f})**
- **random.color(H, S)**
- **random.color(H, {S_i, S_f})**
- **random.color({H_i, H_f}, S)**
- **random.color({H_i, H_f}, {S_i, S_f})**
- **random.color(H, S, V)**
- **random.color(H, S, {V_i, V_f})**
- **random.color(H, {S_i, S_f}, V)**
- **random.color({H_i, H_f}, S, V)**
- **random.color(H, {S_i, S_f}, {V_i, V_f})**
- **random.color({H_i, H_f}, S, {V_i, V_f})**
- **random.color({H_i, H_f}, {S_i, S_f}, V)**
- **random.color({H_i, H_f}, {S_i, S_f}, {V_i, V_f})**

En resumen:

- el parámetro **H** es un valor entre 0 y 360 o una tabla con dos elementos numéricos entre 0 y 360. Su valor por default es aleatorio entre 0 y 360.
- **S** y **V** son, o un valor entre 0 y 100 o una tabla con dos elementos numéricos entre 0 y 100. Sus valores por default son siempre de 100 para cada uno.

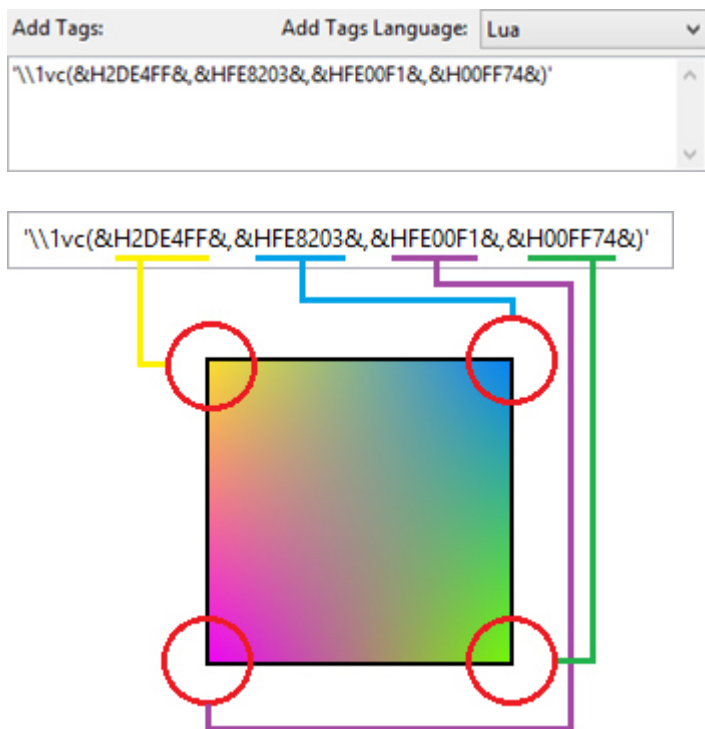
Ejemplos:

- `random.color(115)`
- `random.color({30, 220})`
- `random.color(304, 70)`
- `random.color(216, {60, 80})`
- `random.color({320, 340}, 90)`
- `random.color({125, 150}, {50, 100})`
- `random.color(0, 55, 92)`
- `random.color(86, 10, {0, 72})`
- `random.color(328, {60, 80}, 64)`
- `random.color({80, 120}, 77, 81)`
- `random.color(143, {0, 100}, {50, 80})`
- `random.color({45, 77}, 44, {66, 100})`
- `random.color({0, 105}, {70, 90}, 16)`
- `random.color({170, 204}, {55, 76}, {47, 88})`

Recomiendo poner a prueba en el **Kara Effector**, a cada uno de los anteriores ejemplos con el fin de practicar con esta función hasta que se familiaricen con cada uno de sus modos y opciones de uso. Esta es la primera función de la Librería “**random**” y restan cinco más por ver.

random.colorvc(H, S, V): es similar a **random.color** con la única diferencia que retorna un color en formato del filtro **VSFiltermod**.

Un color en formato **VSFilterMod** está compuesto por cuatro colores para cada uno de sus cuadrantes. Ejemplo:



O sea que todo color en formato **VSFilterMod** tiene la siguiente estructura:

(color **SI**, color **SD**, color **II**, color **ID**)

- **color SI:** Superior Izquierdo
- **color SD:** Superior Derecho
- **color II:** Inferior Izquierdo
- **color ID:** Inferior Derecho

Y lo que hace la función **random.colorvc** es usar la función **random.color** para generar a cada uno de esos cuatro anteriores colores. Los modos y todas las combinaciones posibles de la función **random.colorvc** son exactamente los mismos que en **random.color**, vista anteriormente:

- **random.colorvc()**
- **random.colorvc(H)**
- **random.colorvc({H_i, H_f})**
- **random.colorvc(H, S)**
- **random.colorvc(H, {S_i, S_f})**
- **random.colorvc({H_i, H_f}, S)**
- **random.colorvc({H_i, H_f}, {S_i, S_f})**
- **random.colorvc(H, S, V)**
- **random.colorvc(H, S, {V_i, V_f})**
- **random.colorvc(H, {S_i, S_f}, V)**
- **random.colorvc({H_i, H_f}, S, V)**
- **random.colorvc(H, {S_i, S_f}, {V_i, V_f})**
- **random.colorvc({H_i, H_f}, S, {V_i, V_f})**
- **random.colorvc({H_i, H_f}, {S_i, S_f}, V)**
- **random.colorvc({H_i, H_f}, {S_i, S_f}, {V_i, V_f})**

No sobra recordarles que para usar colores en formato **VSFilterMod** debemos seleccionar esa opción en **Using Tags Filter**, de otro modo el **Kara Effector** convertirá el color a formato **VSFilter 2.39**.

random.alpha(A1, A2): retorna una transparencia (**alpha**) aleatoria en formato .ass (**VSFilter 2.39**) desde el valor de **A1** hasta **A2**.

Los parámetros **A1** y **A2** pueden ser números reales entre 0 y 255, o valores **alpha** entre “&H00&” y “&HFF&”.

Modo 1: sin parámetros

random.alpha(): retorna una transparencia al azar entre totalmente visible (0 en decimal o “&H00” en formato .ass), hasta totalmente invisible (255 en decimal o “&HFF&” en formato .ass).

Modo 2: parámetros numéricos

random.alpha(A1, A2): retorna una transparencia aleatoria entre los valores numéricos de **A1** y **A2**. Ejemplos:

- **random.alpha(55, 142)**
- **random.alpha(0, 200)**
- **random.alpha(180, 255)**

Modo 3: parámetros **alpha** en formato .ass

random.alpha(A1, A2): retorna una transparencia aleatoria entre los valores **alpha** (.ass) de **A1** y **A2**. Ejemplos:

- **random.alpha("&HA3&", "&HED&")**
- **random.alpha("&H0C&", "&H7F&")**

Una especie de Modo 4 sería combinar los modos 2 y 3. Ejemplos:

- **random.alpha(120, "&HED&")**
- **random.alpha("&H0C&", 215)**

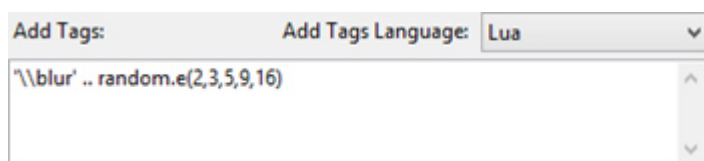
random.alphava(A1, A2): es muy similar a la anterior función y retorna una transparencia (**alpha**) aleatoria en formato **VSFilterMod** desde el valor de **A1** hasta **A2**. Los Modos y formas de uso son los mismos que usamos en **random.alpha**

Para cada una de las cuatro transparencias (alpha), que conforman a una transparencia en formato **VSFilterMod**, se ejecuta el random que hace que sea prácticamente imposible que las cuatro sean iguales. Ejemplos:

- **random.alphava(90, 180)**
- **random.alphava(0, 100)**
- **random.alphava("&HC4&", "&HFF&")**
- **random.alphava("&H00&", "&H86&")**
- **random.alphava(12, "&HAA&")**
- **random.alphava("&HDF&", 125)**

random.e(...): retorna un parámetro al azar de entre todos los que se le hayan ingresado o un elemento al azar, entre los elementos de una tabla que se le haya ingresado como parámetro.

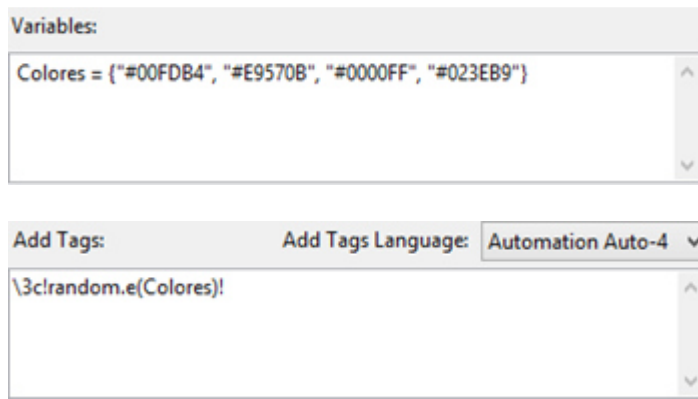
Ejemplo 1:



La función retornaría alguno de esos cinco valores que se le ingresaron, seleccionado de forma aleatoria, o sea que los posibles resultados serán:

- \\blur2
- \\blur3
- \\blur5
- \\blur9
- \\blur16

Ejemplo 2:



Declaré una tabla llamada “Colores”, entonces la función **random.e** selecciona un elemento al azar de entre los cuatro que tiene la tabla.

random.unique(T, i): primero desordena la posición de los elementos de la tabla **T** para posteriormente retornar al elemento **T[i]**. **T** puede ser una tabla propiamente dicha o un número entero positivo. En el caso de que **T** sea un entero, entonces se crea una tabla con los números desde 1 hasta **T**.

Ejemplo 1:

random.unique(syl.n, syl.i): primero crea una tabla con los números consecutivos desde 1 hasta syl.n:

$T = \{1, 2, 3, 4, 5, 6, 7, 8, \dots, \text{syl.n}\}$

Luego desordena la posición de los elementos de la tabla de manera aleatoria:

$T = \{5, 8, 4, \text{syl.n}, 2, 6, 3, \dots, 1, 7\}$

Por último, retorna **T[syl.i]**:

- $T[1] = 5$
- $T[2] = 8$
- $T[3] = 4$
- $T[4] = \text{syl.n}$
- $T[\text{syl.n} - 1] = 1$
- $T[\text{syl.n}] = 7$

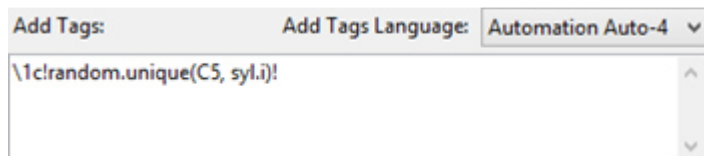
O sea que la función **random.unique** nunca repite un resultado, a menos que sea usada una cantidad mayor de veces que el tamaño de la tabla.

Ejemplo 2:

Declaramos una tabla con cinco colores dispuestos de la siguiente forma:



Si intentamos usar un color para cada Sílabas, y si una Línea de Texto tiene más de cinco Sílabas, entonces esto hace imposible que la función **random.unique** le asigne un único color de esta tabla a cada una de las Sílabas. Lo que hace la función es agotar las opciones y luego repite el mismo patrón una y otra vez:



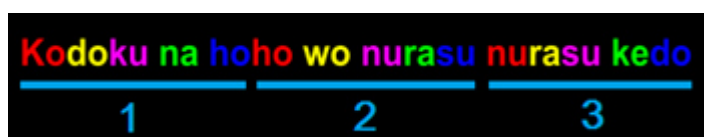
Como la tabla “C5” de nuestro ejemplo solo tiene cinco elementos, el resultado C5[6] o cualquier otro mayor no existiría, entonces la función repite los resultados:



Para esta línea de texto la función reorganizó los colores de la tabla en forma aleatoria así:

1. Rojo
2. Amarillo
3. Lila
4. Verde
5. Azul

Y vemos que el patrón, al menos en esta línea, se repitió tres veces:



Esta se podría considerar como la función más compleja de la Librería “**random**“, pero con un poco de práctica se irán acostumbrando y encontrarán la forma de darle una aplicación en algunos de sus Efectos y proyectos.

Nuevos Modos de la Función R [KE]

Los nuevos modos acá documentados de la función **R** están disponibles a partir de la versión **3.2.9.5** del **KE**, y tienen la finalidad de ampliar las posibilidades a la hora de generar valores numéricos aleatorios.

La primera actualización consiste en un tercer parámetro de la siguiente forma:

R(Val1, Val2, Step)

El parámetro **Step** es un número entero no mayor a la diferencia entre **Val2** y **Val1**, y lo que hace es marcar la distancia entre los valores que retornará la función.

Ejemplo:

R(40, 90, 10)

En este caso, lo que hace el **Step = 10** es que la función retorne un valor aleatorio entre 40 y 90, pero de 10 en 10, o sea que los posibles valores que podrían ser retornados por la función **R** son:

- 40
- 50
- 60
- 70
- 80
- 90

Ejemplo:

R(-21, 35, 7)

Ahora el **Step** tiene un valor de 7 unidades, entonces el valor retornado es un valor aleatorio entre -21 y 35, pero de 7 en 7.

El valor por default del parámetro **Step** es 1, y en el caso de usar este parámetro en la función, debe ser mayor que cero.

Usemos o no el parámetro **Step** en la función, ésta siempre retornará números enteros, y basado en esta particularidad, vienen las siguientes actualizaciones de la función:

- **Rd**
- **Rc**
- **Rm**

Rd hace que la función **R** retorne valores redondeados en décimas.

Ejemplo:

Rd(2, 10)

La función retorna un número aleatorio entre 2 y 10 con una precisión de hasta una décima:

- 2.6
- 5.4
- 9
- 3.1
- 10

Rc hace que la función **R** retorne valores redondeados en centésimas.

Ejemplo:

Rc(-3, 5)

La función retorna un número aleatorio entre -3 y 5 con una precisión de hasta una centésima:

- -1.63
- 1.48
- 0
- 0.31
- 4.92

Rm hace que la función **R** retorne valores redondeados en milésimas.

Ejemplo:

Rm(1, 2.43)

La función retorna un número aleatorio entre 1 y 2.43 con una precisión de hasta una milésima:

- 2.326
- 1
- 1.934
- 2

También podemos usar el parámetro **Step** en la función, en los tres anteriores modos documentados, lo que hará que las posibilidades aumenten aún más.

Todas estas actualizaciones de la función **R** nos servirán de apoyo para muchas de las funciones ya documentadas y para algunas más que aún no hemos visto, y que de a poco aprenderemos a sacarle el máximo provecho.