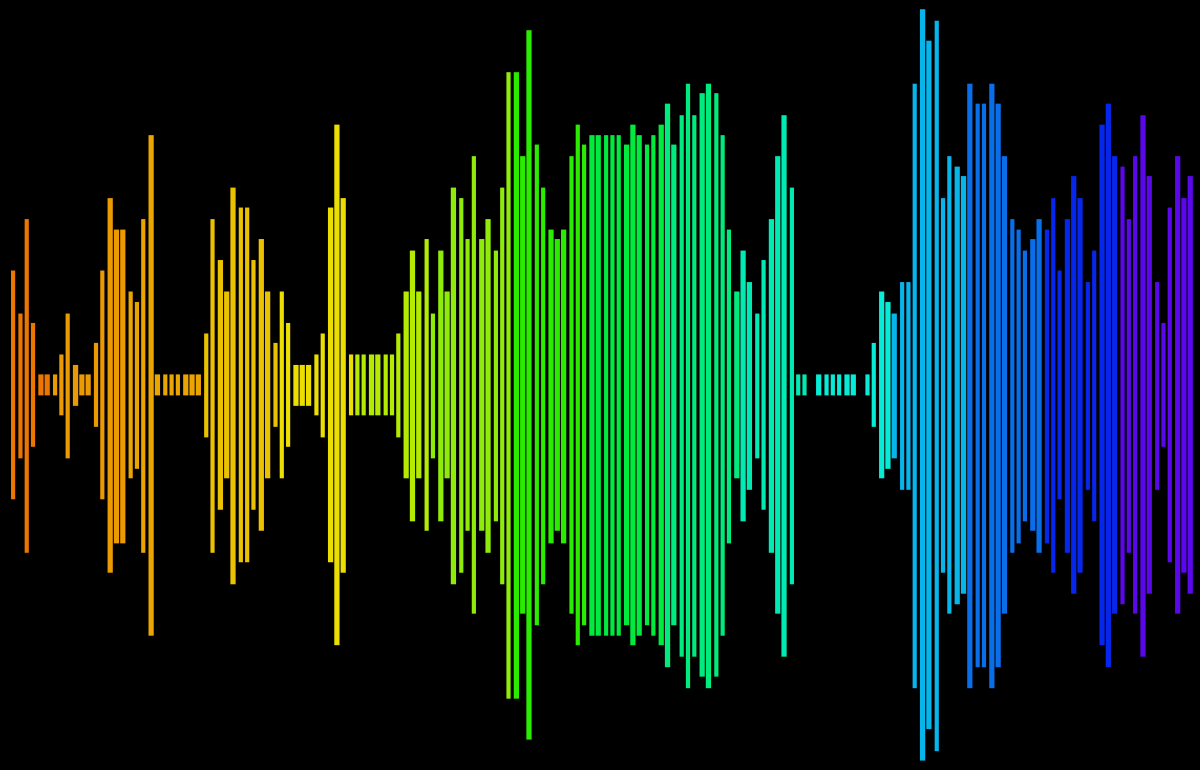


VuMeterGG

Manual de uso del widget VuMeterGG

David García González

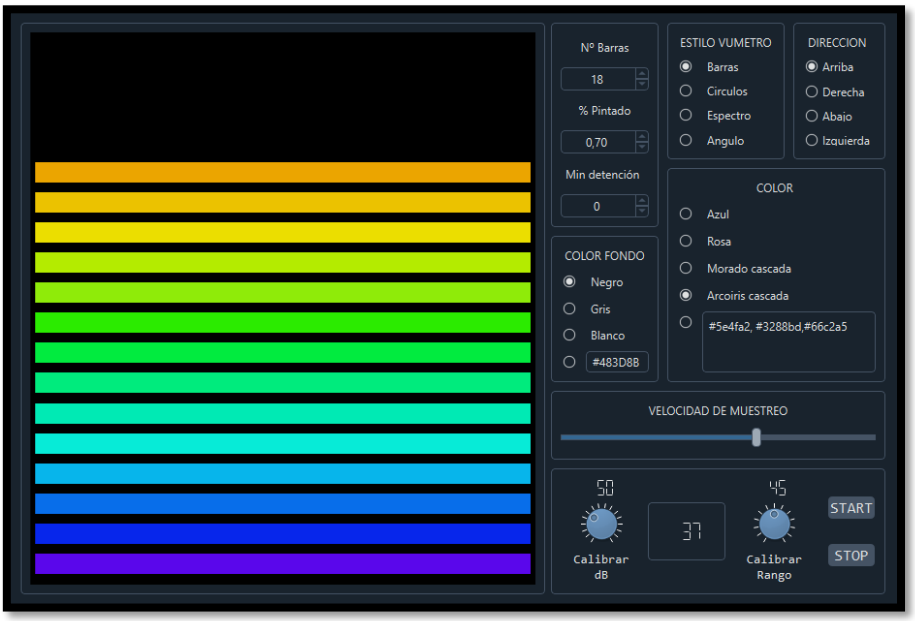
Contenido

INTRODUCCION

VuMeterGG es un widget desarrollado con PySide6. El widget capta el audio recogido de un micrófono y hace una representación gráfica del sonido captado. Cuenta con variedad de ajustes de tratamiento de audio para ajustarlo correctamente y adaptarse al caso de uso, así como múltiples opciones para configurar el estilo de la representación gráfica del sonido.

En el presente manual se abordará el funcionamiento del widget, todas las opciones que permite y cómo sacarle el mayor provecho.

El código del vúmetro viene acompañado de un programa con una interfaz que permite probar sus posibilidades gráficamente.



El vumetro se adapta y ajusta automaticamente a su contenedor por lo que es facilmente adaptable al espacio.

ESTILO DEL VUMETRO

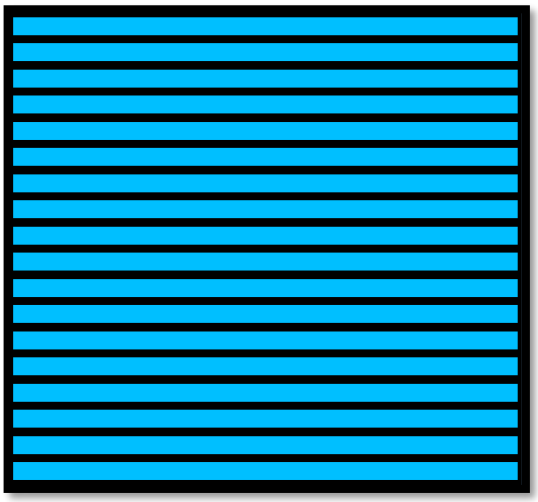
Cambia el estilo del vúmetro, mostrando gráficamente la detección de audio de diferentes formas.

```
def changeStyleVumetro(self, estilo: EstiloVumetro):
```

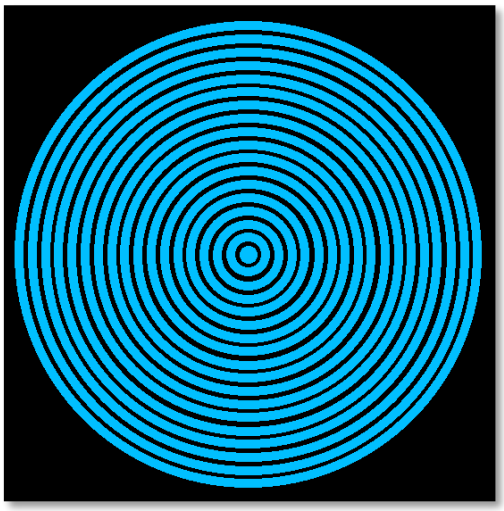
Recibe *estilo*, que es un *enum* de constantes de *EstiloVumetro* que, como posibles valores, tiene:

- BARRAS
- CIRCULOS
- ESPECTRO
- ANGULO

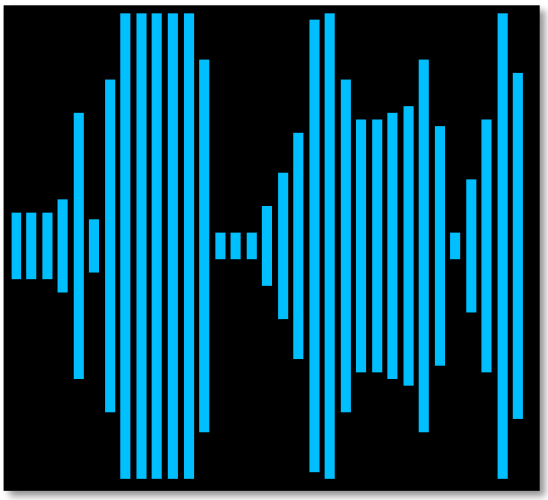
Ejemplo BARRAS



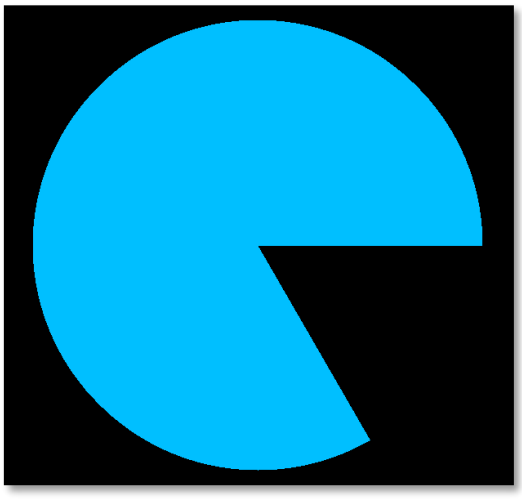
Ejemplo CIRCULOS



ESTILO ESPECTRO



ESTILO ANGULO



REPRESENTACION NUMERICA DEL VOLUMEN

VuMeterGG cuenta con una instancia del widget *QLCDNumber* con nombre *lcd_volumen* conectada al volumen. El motivo de esto es que, si se accedía desde fuera al volumen, se obtenía el volumen que captaba el vúmetro en ese momento, pero no se actualizaba con el cambio de su valor. Por lo que para obtener la representación numérica del volumen en tiempo real este elemento tenía que pertenecer a la clase *VuMeterGG*.

Este elemento no está en la representación gráfica del vúmetro, se crea, pero no se asigna a ningún contenedor por lo que por defecto no es visible. Esto está hecho para que se pueda añadir en cualquier lado de la interfaz que se desee dando total libertad.

Para hacer uso de *lcd_volumen* solo se tiene que agregar al contenedor que se desee de la siguiente manera:

```
self.nombre_del_contenedor.addwidget(self.nombre_instancia_DVumetro.lcd_volumen)
```

Donde *nombre_del_contendor* es dentro de donde se quiere meter *lcd_volumen* y *nomre_instancia_VuMeterGG* es el nombre que le hayas dado a la instancia de *VuMeterGG*.

En este ejemplo lo he colocado dentro de un layout llamado *layout_ldc*, pasando como argumento la propia instancia de *lcd_volumen*.

```
self.layout_lcd.addWidget(self.vumetro.lcd_volumen)
```

INICIAR Y PARAR EL VUMETRO

El vúmetro cuenta con funciones para activarlo y desactivarlo.

Con esta función se vuelve a activar el vúmetro:

```
def activar_vumetro(self):
```

Y con esta otra, se detiene el vúmetro, por lo que deja de actualizar el volumen y la representación gráfica del audio se queda parada. Pudiendo así, observar estos dos elementos parados en un momento concreto.

```
def desactivar_vumetro(self):
```

CALIBRARCION DE LOS DECIBELIOS

Permite corregir y ajustar a los decibelios reales del audio que capta el micrófono.

```
def set_lcd_calibrarDB(self, db: int):
```

Recibe como argumento *db*, si *db* es menor de 1 se le asigna 1. *db* es el valor de escala con el que se trata el audio captado. Por defecto es 50.

CALIBRACION DEL RANGO DETECION

Ajusta el rango de detección de audio que se quiere mostrar, de esta manera el lienzo de visualización se reparte para el rango asignado y no representa el volumen que supere ese rango.

```
def set_lcd_calibrarRango(self, rango: int):
```

Recibe como argumento *rango*, si *rango* es menor de 1 se le asigna 1. *rango* es el volumen máximo que se quiere representar. Por defecto es 45.

AJUSTE DE LA TASA DE MUESTREO

Configura la velocidad a la que se actualiza la información captada, cambiando más rápido o lento el volumen y su representación gráfica.

```
def set_vumeter_speed(self, speed: int):
```

Recibe como argumento *velocidad*, que si es menor de 1 se le asigna 1 siendo la mayor velocidad, y si es mayor de 25 se le asigna 25 siendo la menor velocidad. Por defecto es 10.

DETECION MINIMA

Permite modificar el rango inicial de volumen que ignora el vúmetro a la hora de representar visualmente el sonido. Esto es útil cuando quieres que no capte ruidos débiles de fondo.

```
def setMinDetection(self, minimal: int):
```

Recibe de argumento *minimal*, si es menor de 0 se le asigna 0. Por defecto es 0.

COLOR DEL LIENZO

Modifica el color del lienzo sobre el que se dibuja la representación del sonido captado.

```
def setBackgroundColor(self, color: str):
```

Recibe de argumento *color*, el *String* pasado tiene que ser la representación de un color hexadecimal (# seguida de 3 o 6 números). En caso de que no sea un color valido no realiza ningún cambio.

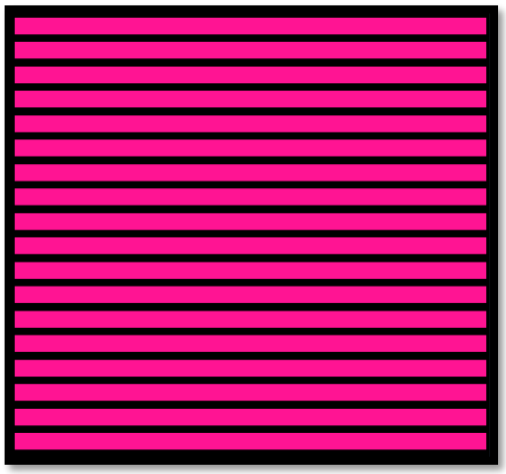
COLOR DE LA REPRESENTACION DEL SONIDO

Permite cambiar el color de la representación gráfica del sonido. Se le tiene que pasar un *String* con el color, o colores separados por comas, en formato hexadecimal (# seguida de 3 o 6 números).

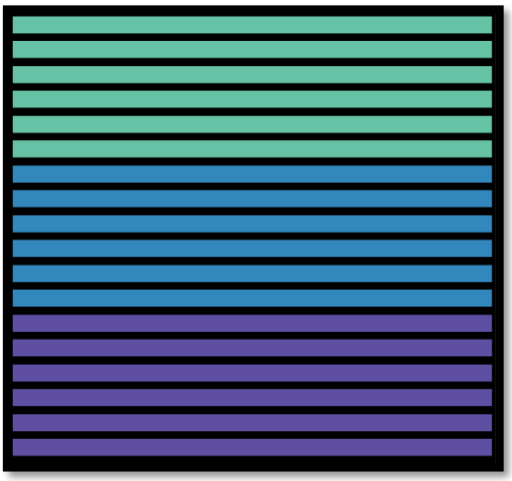
```
def setBarColor(self, color: str):
```

Recibe *color*, si el formato no es correcto no cambia de color. Por defecto '#00BFFF'

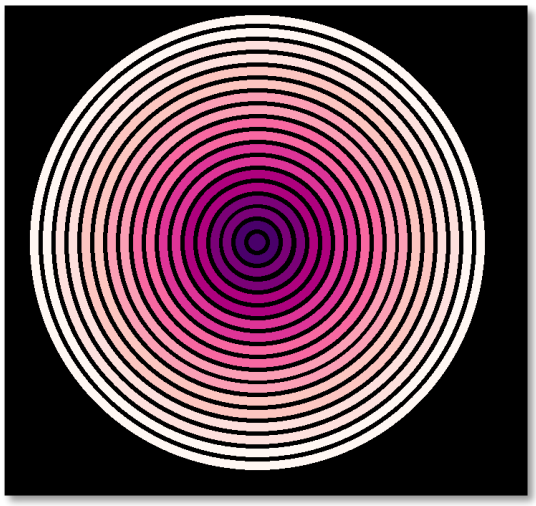
Ejemplo BARRAS con *setBarColor*('#FF1493'):



Ejemplo BARRAS con setBarColor('#5e4fa2, #3288bd,#66c2a5'):



Ejemplo CIRCULOS con setBarColor('#49006a, #7a0177, #ae017e, #dd3497, #f768a1, #fa9fb5, #fcc5c0, #fde0dd, #fff7f3'):



Ejemplo ANGULO con setBarColor("#5A07EB, #0726EB, #086EEA, #07B5EB, #07EBD7, #00EAB4, #00EB7D, #01EB3F, #2BEB00, #8FEB09, #B4EB00, #EBDE00, #EBC200, #EBA500, #EB9A01, #EB7600, #EB4D00, #EB2400"

DIRECCION DE LA REPRESENTACION DEL SONIDO

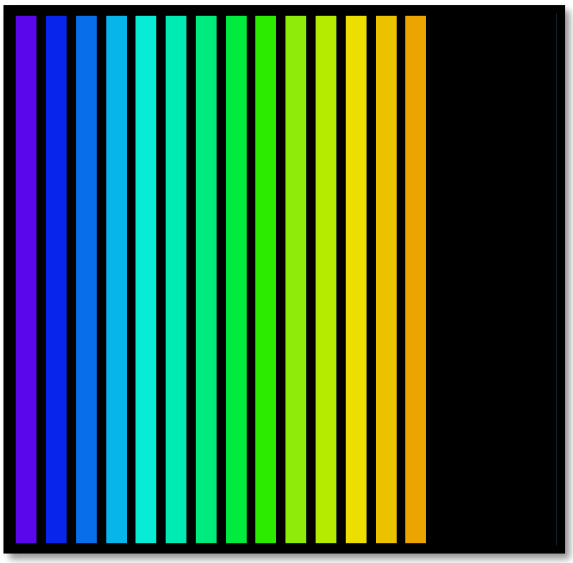
Permite modificar la dirección desde la que empieza y termina la representación gráfica del sonido. Esta función solo modifica el comportamiento del vúmetro con estilo de BARRAS y ESPECTRO.

```
def setDirectionVumetroBarras(self, direction: DirectionVumetroBarras):
```

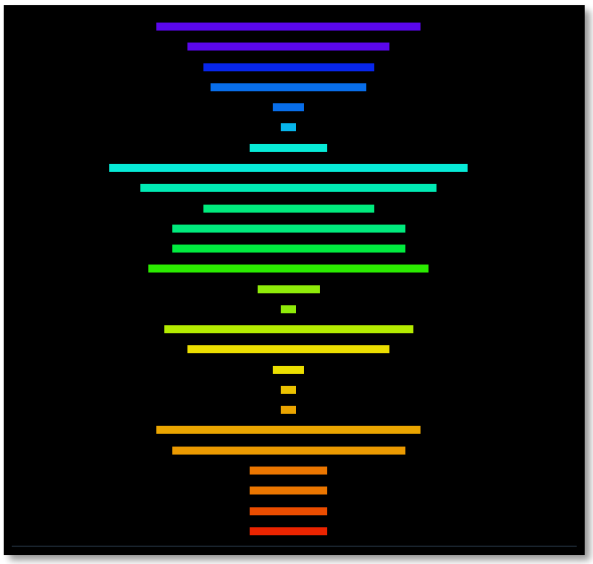
Recibe *direction*, que es un *enum* de constantes de *DirectionVumetroBarras* que, como posibles valores tiene:

- ARRIBA
- DERECHA
- ABAJO
- IZQUIERDA

Ejemplo BARRAS con setDirectionVumtroBarras(DirectionVumetroBarras.DERECHA):



Ejemplo ESPECTRO con setDirectionVumtroBarras(DirectionVumetroBarras.ARRIBA):



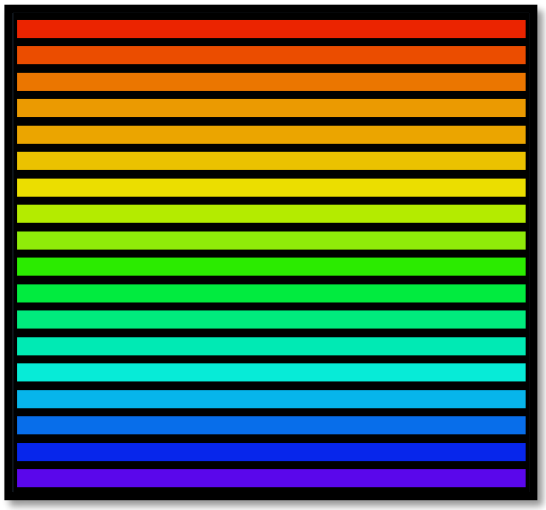
NUMERO DE ELEMENTOS

Modifica el número de elementos, o divisiones, en los que se representa el sonido.

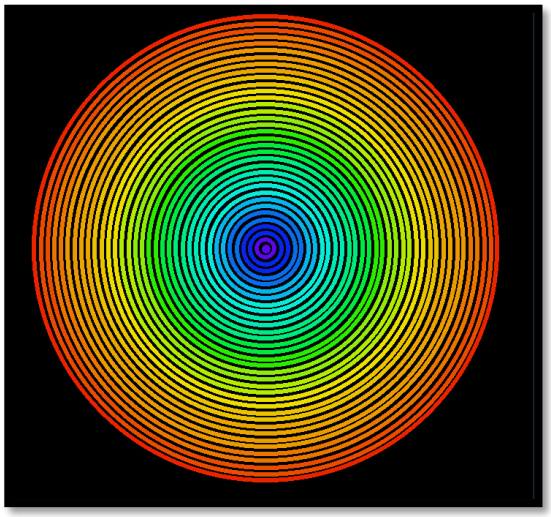
```
def setStep(self, step: int):
```

Recibe *step* que es el número de elementos por el que se quiere dividir la representación gráfica del sonido. Si se le pasa un valor menor de 1 se le asigna 1. Por defecto 18.

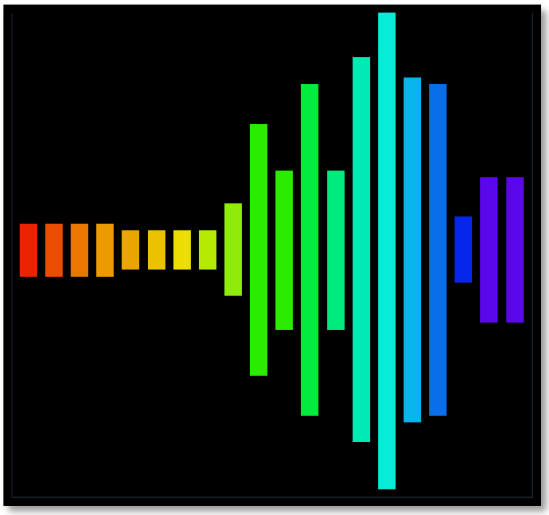
Ejemplo BARRAS con setStep(18)



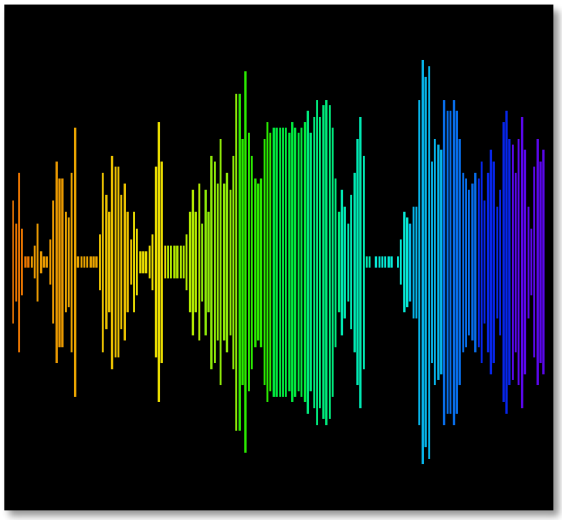
Ejemplo CIRCULOS con setStep(35)



Ejemplo ESPECTRO con setStep(20)



Ejemplo ESPECTRO con setStep(200)



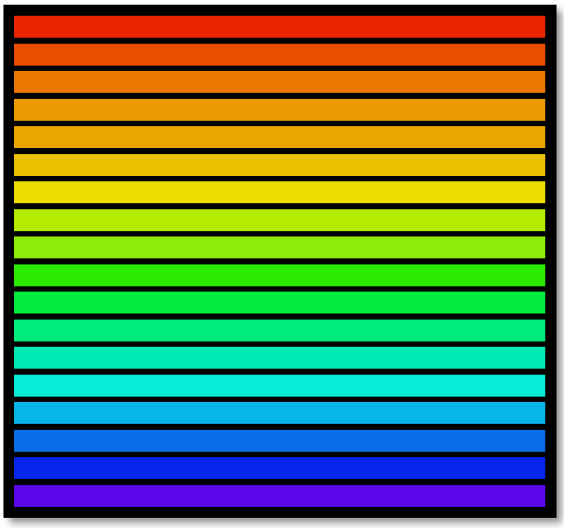
PORCENTAJE DE RELLENO DE LAS BARRAS

El lienzo se reparte entre el número de elementos seleccionados para representar el sonido. Esta función indica cuanto se pinta de ese espacio que le corresponde a cada elemento. Solo aplicable a estilo BARRAS, CIRCULOS y ESPECTRO.

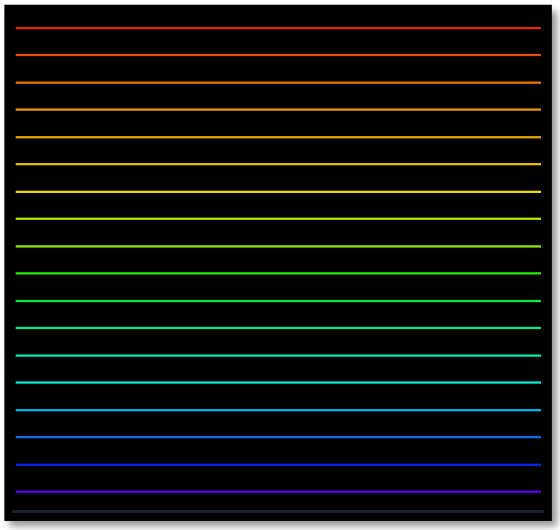
```
def setSolidPercent(self, percent: float):
```

Recibe *percent*, si el número es inferior a 0.1 se le asigna 0.1 que representa el menor porcentaje pintado y si es superior a 0.9 se le asigna 0.9 que representa el mayor porcentaje pintado. Por defecto es 0.7.

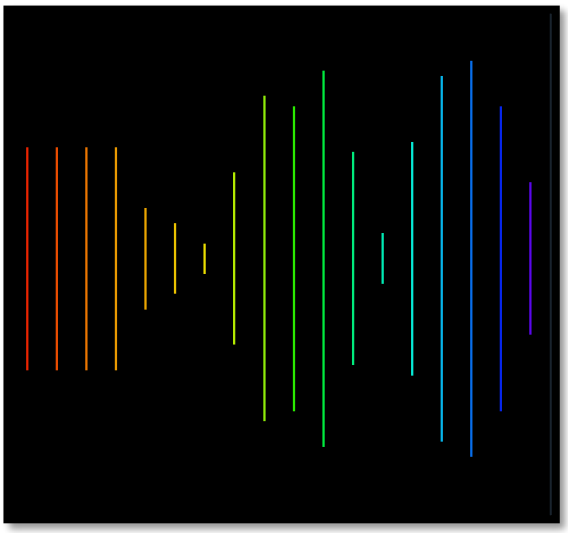
Ejemplo BARRAS con setSolidPercent (0.8)



Ejemplo BARRAS con setSolidPercent(0.1)



Ejemplo ESPECTRO con setSolidPercent(0.1)



Ejemplo CIRCULOS con setSolidPercent(0.3)