conectar LAB

Taller

# Audio digital reactivo

conectar igualdad
educ.ar SOCIEDAD DEL ESTADO
Ministerio de Educación Argentina

# Sobre la propuesta

Con este taller invitamos a experimentar en el mundo del audio digital reactivo, clave hoy en la industria cultural. Para ello, abordaremos nociones básicas del lenguaje de programación, haciendo foco en el *software* Processing, que permitirá a cada participante elaborar sus propias visuales audiorreactivas.

▶▶▶ Processing es un lenguaje de programación y un entorno de desarrollo integrado de código abierto que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Processing utiliza elementos de programación sencillos, una sintaxis simplificada y un modelo de programación de gráficos. Por lo que, aunque se base en Java, no se requieren conocimientos profundos de este lenguaje.

## Objetivos

- Introducir los conceptos fundamentales del lenguaje de programación.
- Abordar el arte desde la programación.
- Brindar oportunidades para el trabajo colaborativo.
- Programar visuales audiorreactivas.

# Actividades

▸ **Primer momento: producciones visuales**
Se puede empezar con una introducción sobre las nuevas tendencias de las producciones visuales y el audio digital reactivo en los distintos espectáculos, y mostrar algunos ejemplos.

▸▸ **Segundo momento: definiciones de programación**
Breve explicación sobre qué significa programar, qué es un lenguaje de programación, y los tipos de niveles.

▸▸▸ **Tercer momento: ¿qué es Processing?**
Ya dispuestos a trabajar en las *netbooks* o PC, proponemos a los y las participantes que experimenten los comandos básicos de Processing para explorar el funcionamiento y las partes de este lenguaje en particular.

▸▸▸▸ **Cuarto momento: ¡a jugar!**
Siguiendo las instrucciones paso a paso, cada participante va a escribir su propio código para realizar las visuales audiorreactivas. Podrán ayudarse con los ejemplos y puntos de llegada, que fueron creados para este taller: *sketches* del 1 al 5.

Estos *sketches* tienen un código desarrollado para cada instancia de la actividad y deberán ser descargados previamente en las computadoras. (Ver debajo «Descargar *sketches* para esta actividad»).

**Para realizar este taller es indispensable que las personas participantes cuenten con computadoras que tengan instalado Processing 4. Pero, además, presentamos algunas sugerencias que pueden mejorar esta experiencia.**

# Cómo mejorar la experiencia del taller

Para que la actividad logre una dinámica visualmente más atractiva, en caso de contar con un proyector, sugerimos utilizar las paredes para la proyección de los videos audiorreactivos. También pueden resultar útiles instrumentos y micrófonos para grabar sonidos, y ver cómo reaccionan las visuales a la ejecución de los instrumentos.

**Para esto van a necesitar:**

- 1 proyector
- 1 cable HDMI para conectar el proyector a la computadora
- 1 interface de audio para conectar los micrófonos a la computadora
- 1 o 2 micrófonos dinámicos o de contacto para grabar intrumentos o voces
- Cables XLR XLR para conectar los micrófonos con la interface de audio
- 1 cable plug para conectar un instrumento a la computadora mediante la interface de audio
- Instrumentos convencionales; por ejemplo: guitarra, cajón peruano, etc.

▸ **Descargar *sketches* para esta actividad en el siguiente link:**
Copiá y pegá este enlace en tu navegador para descargar los sketches para esta actividad:
https://www.educ.ar/file/%24ebd6787b8aafd2317401fb30abde6332b3f44702851e9ca4ae8228033b84cc21

▸ **Podés descargar Processing 4 desde el siguiente link:**
https://processing.org/download/

# Primer momento

## Producciones visuales

Los shows electrónicos son una experiencia única, una invitación a viajar a otro mundo. Y no solo la música, sino también las luces, los efectos visuales y las pantallas gigantes forman parte de estos. Hoy artistas y equipos visuales ponen mucho esfuerzo en la producción visual, lo que hace que el espectáculo sea aún más impresionante.

- ¿Notaste que muchas veces estas luces, colores, músicas, visuales, etc., están todos sincronizados y responden en grupo entre sí?
- ¿Qué opinas sobre esto?
- ¿Sabés cómo funciona?
- ¿Conocés ejemplos de algo que hayas visto?

### Presentamos algunos ejemplos de obras visuales audiorreactivas:

1. Fusión espectral (https://www.youtube.com/watch?v=DepjId-UU5s)
2. Generative art (https://www.youtube.com/watch?v=qtPi0JvmWbs)

Este tipo de obras se conoce como **arte generativo.** Se trata de una forma innovadora de crear obras de arte, que en su totalidad o en parte son generadas por un sistema autónomo, como el código de un programa de computadora.

En este taller vamos a trabajar sobre esta línea para crear visuales audiorreactivas.

# Segundo momento

## Definiciones de programación

Todas las visuales previamente observadas fueron programadas. La **programación informática** es el arte de indicarle a una computadora lo que tiene que hacer mediante un conjunto de instrucciones. Las máquinas en general, y las computadoras en particular, necesitan un lenguaje propio para interpretar las instrucciones que reciben y para que las personas puedan controlar su comportamiento. Ese lenguaje es, justamente, el de programación (C++, Java, JavaScript, etc.).

Asimismo, el lenguaje de programación está conformado por una serie de reglas sintácticas y semánticas que utilizará quien programe para crear un programa o subprograma. Las instrucciones que forman dicho programa son conocidas como **«código fuente».** Lo singular de este lenguaje que utiliza es que le permite hacer las especificaciones en forma precisa. Por eso, todo se interpreta de la misma manera, sea quien fuere el programador o programadora que lo realice. Esto lo diferencia, por ejemplo, del lenguaje humano, donde no siempre las especificaciones se entienden de la misma manera.

Para resumir, programar es tomar una serie de instrucciones definidas y ordenadas para resolver un problema específico (algoritmo) y codificarlas en una notación, un lenguaje que pueda ser ejecutado por una computadora.

Aunque existen muchos lenguajes de programación y varios tipos diferentes de computadoras, el primer paso es la necesidad de tener una solución. Así, sin algoritmo no puede haber programa. Los **algoritmos** describen la solución a un problema en términos de los datos requeridos para representar el caso del problema y el conjunto de pasos necesarios para producir el resultado pretendido. Los lenguajes de programación deben suministrar un modo notacional para representar tanto el proceso como los datos. Para este fin, **los lenguajes suministran estructuras de control y tipos de datos.**

Las **estructuras de control** hacen que los pasos algorítmicos sean representados de una manera conveniente pero sin ambigüedades. Como mínimo, los algoritmos requieren estructuras que lleven a cabo procesamiento secuencial, selección para toma de decisiones e iteraciones para control repetitivo. Siempre y cuando el lenguaje proporcione estas instrucciones básicas, este puede ser usado para la representación del algoritmo. Todos los ítems de datos en la computadora se representan como cadenas de dígitos binarios.

Con el fin de darle significado a estas cadenas, necesitamos tener **tipos de datos.** Estos brindan una interpretación de los datos binarios, para que los consideremos en términos que tengan sentido respecto del problema que está siendo resuelto. Estos tipos de datos incorporados de bajo nivel (a menudo denominados «tipos de datos primitivos») proporcionan los bloques constructivos para el desarrollo de algoritmos.

Las cadenas de dígitos binarios en la memoria de la computadora pueden interpretarse como enteros y se les dan los significados típicos que comúnmente asociamos con estos (por ejemplo, 23, 654 y -19). Además, un tipo de datos también proporciona una descripción de las operaciones en las que los ítems de datos pueden participar. Con enteros, son comunes las operaciones tales como la suma, la resta y la multiplicación. Podemos dar por sentado que los tipos de datos numéricos pueden participar en estas operaciones aritméticas. La mayoría de lenguajes de programación proporciona un tipo de datos para los enteros.

La dificultad más habitual es que los problemas y sus soluciones son muy complejos. Estas estructuras y tipos de datos simples, suministrados por el lenguaje, si bien son ciertamente suficientes para representar soluciones complejas, están típicamente en desventaja a medida que trabajamos en el proceso de solución de problemas. Requerimos maneras de controlar esta complejidad y contribuir con la creación de soluciones.

## ¿Qué es un lenguaje de programación?

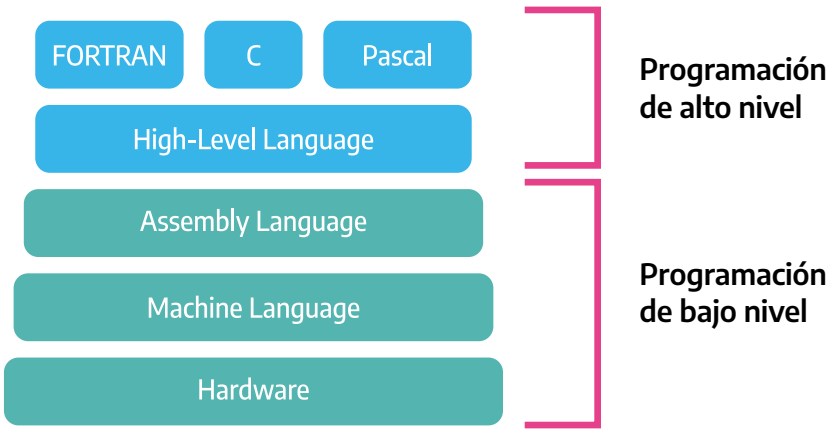
Se trata de un lenguaje formal de reglas diseñado para que el programador o la programadora **escriba las instrucciones y la computadora las interprete,** para realizar procesos que llevan a cabo máquinas como las computadoras.

Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión o como modo de comunicación humana.

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Se llama «programación» **el proceso por el cual se escribe, prueba, depura, compila (de ser necesario) y mantiene el código fuente de un programa informático.**

Los lenguajes de programación pueden clasificarse de diversas maneras, como, por ejemplo, según su nivel de abstracción: **lenguaje de bajo nivel** (es el código fuente de la máquina, es decir, el que esta puede interpretar), **lenguaje de nivel medio** (un término entre el lenguaje de la máquina y el lenguaje natural) y **lenguaje de alto nivel** (los que están compuestos por elementos del lenguaje natural -es decir, el humano-, especialmente el inglés).

También puede haber lenguajes según la forma de ejecución. Aquí nos encontramos con el compilador (que permite traducir un programa del lenguaje natural al lenguaje de bajo nivel) y lenguajes interpretados (que solo traducen los datos que se van a utilizar en ese momento y no los guarda para un uso posterior). Hay tres tipos de lenguaje de programación: lenguaje de máquina (bajo nivel), lenguaje ensamblador (bajo nivel) y lenguaje de alto nivel. **Processing se encuentra en un lenguaje de alto nivel.**



# Tercer momento

## ¿Qué es Processing?

Processing es un **lenguaje de programación** y **entorno de desarrollo integrado de código abierto** basado en Java, de fácil utilización, y sirve como medio para la enseñanza y producción de proyectos **multimedia** e interactivos de diseño digital.

Para comenzar, veamos diferentes ejemplos de lo que se puede desarrollar con esta herramienta. En el documento Cheat de Processing, se explican las partes principales y sus funcionamientos.

Podés descargar el documento Processing Cheat sheet copiando y pegando el siguiente enlace en tu navegador:
http://www.educ.ar/file/%24e54c21c5fda0ce81f80580993d5d7b60fff1a0f6ff8f27afbc1473f20479f061

Sugerimos ver el Documental Hello World desde el siguiente enlace:
https://vimeo.com/61191770

# Cuarto momento

## A codear

Sugerimos ir trabajando cada paso de la actividad como se detalla debajo.

Si surgen dudas, dificultades o algún participante se atrasa en la actividad, podrán usar los *sketches* de apoyo creados para este taller (*sketches* del 1 al 5). Estos *sketches* tienen el mismo código que iremos desarrollando en cada instancia del paso a paso.

Si hacen las modificaciones correspondientes en los distintos *sketches*, podrán desarrollar sus propias visuales audiorreactivas. Por ejemplo, en el *sketch* 01, el código indica un "size(600,600)", y, si cada participante modifica alguna variable dentro de los paréntesis, podrá personalizar su programa.

### Instrucciones paso a paso

Vamos a escribir el código: lo que se encuentra en letras verdes es el código en sí mismo y lo que está en letras grises son comentarios para explicar qué realiza esa parte del código, pero no es necesario escribirlos para que el programa funcione.

**1 Imitando a Mondrian: desarrollamos la función void setup():**

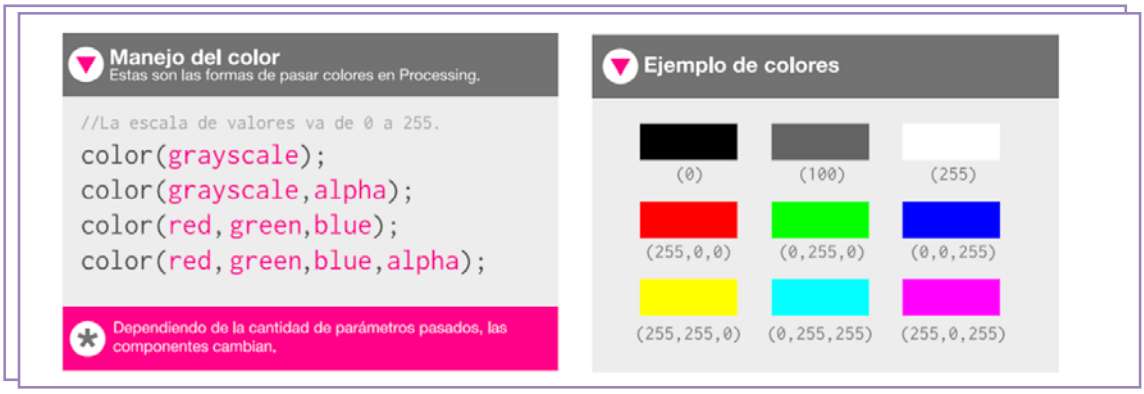
```
void setup() {  // La función setup() se ejecuta una vez al comienzo del programa. Aquí se inicializa la ventana y se configuran las variables estables. "{" es necesario para abrir la función y la parte final de esta función hay que cerrarla "}".

size(600,600); // establece el tamaño de la ventana en píxeles, el primer número eje X y el segundo, eje Y.

background(0); // establece el color de fondo.

}
```

- Invitamos a las y los participantes a personalizar esta primera parte de su código eligiendo un color propio. Los colores funcionan dentro de Processing de la siguiente forma:




- Quienes no pudieron terminar de escribir bien esta parte del código pueden abrir directamente el sketch_01_mondrian_voidsetup y así continuar con el siguiente paso.

### 2 Imitando a Mondrian: desarrollamos la función void draw():

**void draw(){** // La función draw() se ejecuta continuamente mientras se esté ejecutando el programa. Aquí se dibujan las formas y se procesa el audio.

**Como ejemplo dibujamos un rectángulo agregando la siguiente línea:**

**rect(10,10,200,300);** // Dibuja un rectángulo: los dos primeros números indican la posición de los ejes x/y de la arista superior izquierda 10,10; los últimos dos números indican el ancho (200) y el alto (300) del rectángulo.

- Los y las participantes pueden personalizar el tamaño del rectángulo modificando los dos últimos números.
- Quienes no pudieron terminar de escribir bien esta parte del código pueden abrir directamente el sketch_02_mondrian_voiddraw y así continuar con el siguiente paso.

## 3 Imitando a Mondrian: colores, rellenos y bordes

Para modificar información sobre el rectángulo previamente dibujado, las nuevas líneas del código se deben escribir de forma previa a cualquier figura geométrica y no de forma posterior.

Para el color de relleno del rectángulo, el color de borde de la línea del rectángulo y el ancho de la línea, vamos a utilizar las siguientes funciones:

```
fill(color);
```
Configura el color de relleno de lo que se dibuje posteriormente.

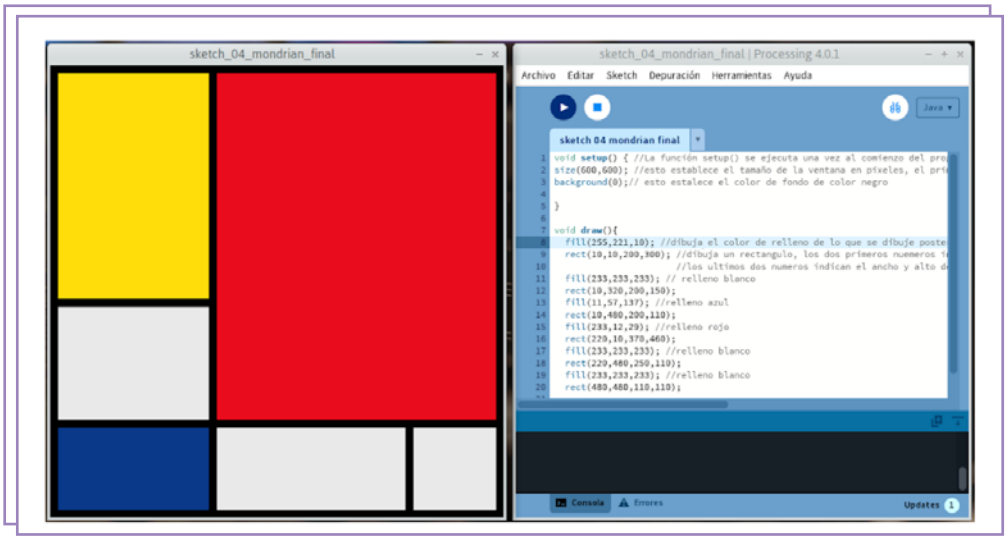
```
stroke(color);
```
Configura el color de borde de lo que se dibuje posteriormente.

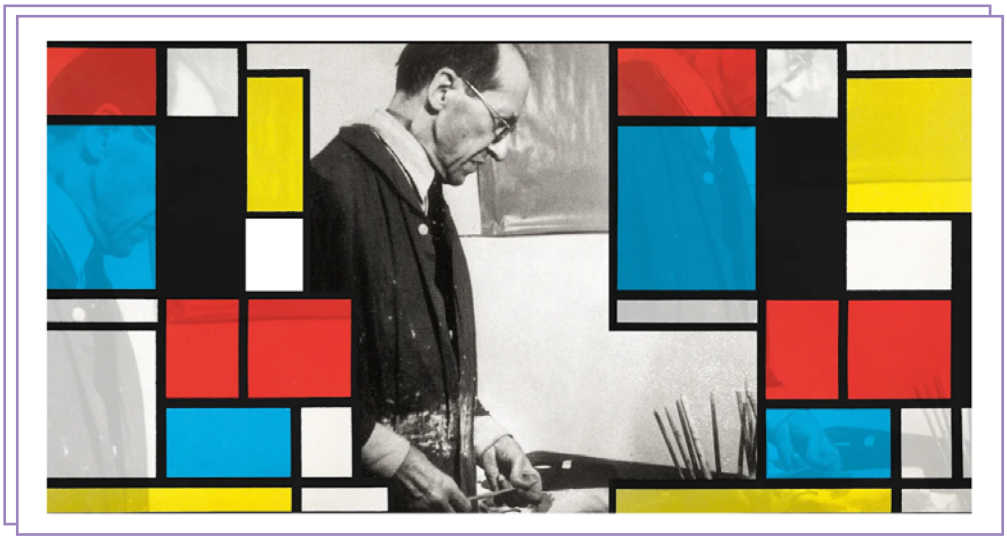
```
fill(255);          fill(255);          noFill();
noStroke();         stroke(0);          stroke(0);
```

- Proponemos que dibujen distintas figuras personalizadas con colores, tamaños, etc.
- Quienes no pudieron terminar de escribir bien esta parte del código pueden abrir directamente el sketch_03_mondrian_colores y así continuar con el siguiente paso.

## 4 Imitando a Mondrian: ¿llegamos al final?

Abrimos y mostramos nuestro código finalizado, que presentará una obra del artista Piet Mondrian, un pintor vanguardista neerlandés. Nuestro código no es necesariamente igual al producido por los y las participantes. Por último, sugerimos mostrar las distintas imágenes de las obras del autor para explicar que no solo están dibujando cuadros de colores sino reproduciendo una obra famosa.



- **Mondrian en movimiento:** se ejecuta “sketch 05” y “sketch04” al mismo tiempo para comparar y graficar que el nuevo objetivo es lograr que el dibujo previamente creado sea reactivo al sonido.

## Mondrian en movimiento: personalización

Regresamos al “sketch 04” para agregar las nuevas funciones y que el *sketch* reaccione:

**1.** Previo a todo lo escrito, importamos la librería* Minim y agregamos las siguientes líneas:

```
import ddf.minim.*;
Minim minim;
AudioPlayer player;
```

**2.** Dentro de la función void setup, después de la variable “background” escribimos las siguientes líneas que deben quedar arriba de “}”:

```
minim = new Minim(this); // Crear una instancia de Minim
player = minim.loadFile(“song.mp3”); // Cargar el archivo de audio
player.play(); // Reproducir el archivo de audio
```

**3.** Dentro de la función void draw, agregar las siguientes líneas antes del “fill” de nuestra primera figura geométrica:

```
float = player.mix.level();  // Obtiene el nivel de amplitud del audio
```

* Una librería es un conjunto de código preescrito y empaquetado que se puede utilizar para ampliar las capacidades de Processing. Las librerías contienen clases, funciones y recursos que se pueden utilizar en un programa de Processing para agregar nuevas funcionalidades, como procesamiento de audio, reconocimiento de gestos, gráficos 3D, entre otros.

**4.** Por último, en cada figura geométrica creada vamos a elegir uno de los valores de los tamaños y agregar lo siguiente: + level * “X” (“x” es el tamaño que queremos que se agrande nuestra imagen de acuerdo con el audio). Por ejemplo, dentro del sketch 04 el primer rectángulo de color amarillo está descripto de la siguiente forma:

```
rect(10, 10,200,300);
```

Para agregar la nueva modificación sobre el eje “y” del tamaño del rectángulo, hacemos lo siguiente:

```
rect(10, 10,200,300 + level * 200);
```

**5.** Para finalizar, proponemos experimentar modificando el nivel de amplitud para cada figura geométrica que se haya creado. También es posible cambiar los colores, etc., y así seguir personalizando los proyectos. Una vez realizado esto, se tendrá el cuadro de Mondrian audiorreactivo terminado.

Teniendo en cuenta que cada participante hará su propia versión del cuadro de Mondrian audiorreactivo, compartimos, solo a modo de ejemplo, un video de un proyecto terminado: https://www.youtube.com/watch?v=vfqmeYo72CE

## Para seguir profundizando, compartimos el código audiorreactivo comentado

***import ddf.minim.*;***
*Esto importa la biblioteca Minim, que es una biblioteca de audio para Processing.*

***Minim minim;***
*Esto declara una variable minim de tipo Minim, que se utilizará para inicializar la biblioteca Minim.*

***AudioInput in;***
*Esto declara una variable in de tipo AudioInput, que se utilizará para capturar el audio de entrada.*

***void setup() {***
*La función setup() se ejecuta una vez al comienzo del programa. Aquí se inicializa la ventana y se configuran las variables.*

***size(400, 300);***
*Esto establece el tamaño de la ventana en píxeles.*

***minim = new Minim(this);***
*Esto inicializa la biblioteca Minim utilizando la ventana actual como argumento.*

***in = minim.getLineIn(Minim.STEREO, 512);***
*Esto inicializa la variable in para capturar audio de entrada. El primer argumento indica que queremos capturar audio estéreo, mientras que el segundo argumento indica el tamaño del búfer utilizado para capturar el audio.*

***void draw() {***
*La función draw() se ejecuta continuamente mientras se esté ejecutando el programa. Aquí se dibujan las formas y se procesa el audio.*

***background(255);***
*Esto establece el color de fondo en blanco.*

***float left = in.left.level() * width / 2;***
*Esto calcula el nivel del audio en el canal izquierdo y lo almacena en la variable left. El nivel se multiplica por la mitad del ancho de la ventana para escalarlo.*

***float right = in.right.level() * width / 2;***
*Esto calcula el nivel del audio en el canal derecho y lo almacena en la variable right. El nivel se multiplica por la mitad del ancho de la ventana para escalarlo.*

***fill(255, 0, 0);***
*Esto establece el color de relleno en rojo.*

***ellipse(width / 4, height / 2, left, left);***
*Esto dibuja una elipse en la mitad izquierda de la ventana, utilizando la variable left para determinar su tamaño.*

***fill(0, 255, 0);***
*Esto establece el color de relleno en verde.*

***ellipse(width / 4 * 3, height / 2, right, right);***
*Esto dibuja una elipse en la mitad derecha de la ventana, utilizando la variable right para determinar su tamaño.*

## Algunos tutoriales para seguir trabajando con Processing:

https://processing.org/tutorials/

## Algunos enlaces para seguir conociendo a Piet Mondrian:

https://www.piet-mondrian.org/

https://es.wikipedia.org/wiki/Piet_Mondrian

https://www.youtube.com/watch?v=JLNhxf8fmDM&t=214s

**Autor: Martín Jerez**