

Meta_Processing Alpha 1.0

Programación para principiantes

Jose David Cuartas Correa



LOS LIBERTADORES
FUNDACIÓN UNIVERSITARIA



EDICIÓN DE PRUEBA 1.0: MAYO, 2020
© Fundación Universitaria Los Libertadores
© Jose David Cuartas Correa

LOS LIBERTADORES, FUNDACIÓN UNIVERSITARIA
Bogotá D. C., Colombia
Cra 16 No. 63 A - 68 / Tel. 2 54 47 50
www.ulibertadores.edu.co

Juan Manuel Linares Venegas
Presidente del Claustro

Patricia Martínez Barrios
Rectora

Ángela María Merchán Basabe
Vicerrectora Académica

Jose David Cuartas Correa
Diagramación y diseño portada

INTRODUCCIÓN

Con este texto busco introducir al lector en los aspectos básicos de la primera versión de Meta_Processing, un lenguaje de meta-programación que desarrollé para el principiante de la programación. Es una iniciativa personal que es influenciada por mi trabajo como director del Laboratorio Hipermedia¹ (Hitec Lab) de la Fundación Universitaria Los Libertadores (Bogotá, Colombia).

La idea de crear Meta_Processing surge durante el desarrollo de mis estudios de doctorado en Diseño y Creación, el cual finalicé con la tesis: “Programar el mundo en el contexto de las tecnologías libres y las culturas Hacker-Maker. Caso de estudio: Hitec Lab” (cuartas, 2017). Allí destaque la importancia de que los diseñadores y creativos aprendieran a programar, y pude presentar evidencias de la gran variedad de oportunidades creativas que este conocimiento les puede brindar a los estudiantes curiosos e inquietos.

Durante este tiempo también escribí el libro “Digitópolis I: Diseño de Aplicaciones Interactivas para Creativos y Comunicadores” el cual era una guía introductoria al lenguaje de programación Procesing. Con este libro busqué promover el interés por aprender programación en los estudiantes de diseño gráfico, publicidad, comunicación. Sin embargo para aquellos que no tenían buenas bases de inglés, se les dificultaba recordar las palabras claves del lenguaje.

¹Laboratorio Hipermedia <http://hiteclab.libertadores.edu.co/>

Meta_Processing es un entorno de programación diseñado para no permitir que el usuario principiante cometa errores comunes de sintaxis. Es un lenguaje de meta-programación basado en el lenguaje Processing, y todo el código creado con Meta_Processing es exportado como código Processing. Con Meta_Processing se puede escribir y leer el mismo código en diferentes idiomas, como Español, Francés, Hindi, Japonés, Italiano, Chino, Portugués e Inglés.

El concepto de meta-programación hace referencia a un programa que es capaz de escribir o manipular otros programas. El concepto Meta viene de la preposición griega que significa: "después" o "más allá" pero en este caso se usa en la acepción más contemporánea que hace referencia al prefijo "sobre". Un buen ejemplo es cuando se usa en la palabra "meta-cognición" que significaría "cognición sobre la cognición".

Así pues, Meta-Processing lo defino como un lenguaje de meta-programación que funciona sobre Processing. Es un lenguaje de más alto nivel que Processing, pero que se traduce y se ejecuta como código Procesing. En otras palabras así como Processing es una abstracción de Java, Meta_Processing es una abstracción de Processing. Así pues, Meta_Processing continúa con la tradición del Medialab del MIT y de la misma manera como John Maeda se apoyó en los hombros de Java para crear Design by numbers, y de la misma manera que Casey Reas y Ben Fry se inspiraron en Design by numbers para crear Procesing, igualmente Meta_Procesing se apoya en los hombros de Processing para ofrecer una experiencia de programación mucho más amigable con el principiante de programación.

Este lenguaje de meta-programación también surge motivado por la reflexiones hechas por Bret Victor en sus conferencias: Inventing on a Principle (2012) y Stop Drawing Dead Fish (2013), en donde Victor demuestra la urgente necesidad de construir nuevas herramientas que le permita a los creadores, explotar el potencial que tienen por ofrecernos las computadoras. Victor habla de la necesidad de alejarnos del paradigma algebraico y textual que es el más usado a la hora de programar, y propone un paradigma basado en manipulaciones geométricas. Metra_Processing busca hacer otro tipo de aproximación mezclando la metáfora de programación grafica (tipo Scratch) con la metáfora de programación basada en texto (tipo Processing), en una herramienta híbrida que toma las mejores características de ambas, para ofrecer una experiencia amigable que no aleje al usuario del paradigma predominante (que es el textual), pero que le evite algunos momentos de frustración causados por errores insignificantes de sintaxis, que comúnmente le sucede a los principiantes.

Desde hace años existen iniciativas fantásticas de lenguajes de programación para niños como los son Logo y Scratch (desarrollados en Estados Unidos por el MIT) o Pilas Bloques (desarrollado en Argentina por la iniciativa program.ar). Sin embargo Meta_Processing le apunta a otro tipo de usuarios que quieren aprender a programar sin sentir que están usando herramientas diseñadas para niños. También podría llegar a ser usado por niños y personas jóvenes que no quieren usar herramientas con interfaces de estilo infantil.

Jose David Cuartas Correa
Bogotá, Colombia
2020

Agradecimientos:

Al apoyo incondicional recibido por parte de la Fundación Universitaria Los Libertadores quienes han creído en cada proyecto que desarrollamos desde el laboratorio Hitec.

Dedicatoria:

A mi esposa Shahzadi y a mis hijas Helen y Megan, quienes llenan de amor y alegría mi existencia.

CONTENIDO

1. META_PROCESSING PRIMEROS PASOS	12
1.1. ¿Cómo abrir Meta_Procesing?	12
1.2. Ventanas que se abren al iniciar Meta_Processing	15
1.3. Elementos básicos de la interface	16
1.4. ¿Cómo seleccionar Idiomas?	17
1.5. ¿Cómo ejecutar el código?	18
1.6. ¿Cómo agregar una línea de código?	18
1.7. ¿Cómo eliminar una línea de código?	19
1.8. ¿Cómo agregar instrucciones?	19
1.9. ¿Cuál es la estructura de archivos y carpetas en Meta-Processing?	21
1.10. ¿Cómo abrir la carpeta data del proyecto actual?	23
1.11. ¿Cómo crear un nuevo proyecto?	23
1.12. ¿Cómo abrir un proyecto?	24
1.13. ¿Cómo guardar el proyecto actual?	25
1.14. ¿Cómo exportar el proyecto actual como aplicación?	25
1.15. Opciones de configuración	25
1.16. Funciones: principal, teclado y ratón	27
1.16.1. Principal	27
1.16.2. Teclado	28
1.16.3. Ratón	28
2. INSTRUCCIONES BÁSICAS	30
2.1. Documentar el código	31
2.2. Coordenadas de pantalla	32

2.3.	Instrucciones para gráficos en pantalla	34
2.3.1.	fondo.....	34
2.3.2.	línea.....	35
2.3.3.	rectángulo	35
2.3.4.	elipse	36
2.3.5.	triángulo.....	37
2.3.6.	tamtexto.....	37
2.3.7.	texto.....	38
2.3.8.	imagen	39
2.3.9.	colorlinea	39
2.3.10.	relleno	40
2.3.11.	sinrelleno	41
2.3.12.	sin linea	42
2.4.	Instrucciones multimedia	42
2.4.1.	tocanota.....	42
2.4.2.	sonido	43
2.4.3.	video	43
3.	VARIABLES Y CONDICIONES	45
3.1.	Variables	45
3.1.1.	¿Cómo se mira el listado de variables?	46
3.1.2.	¿Cómo se crea una variable?.....	46
3.1.3.	¿Cómo se elimina una variable?.....	47
3.1.4.	¿Cómo se inicializa una variable?	48
3.1.5.	¿Cómo asignarle un nuevo valor a una variable?	49
3.1.6.	¿Cómo se le suma un valor a una variable?	50
3.1.7.	¿Cómo se le resta un valor a una variable?	51

3.1.8. ¿Cómo se le asigna un valor aleatorio a una variable?	52
3.2. Condiciones.....	53
4. EJEMPLOS DE CÓDIGO CON META_PROCESSING	58
4.1. Ejemplo básico con el Teclado	58
4.2. Ejemplo básico con el Ratón.....	59
4.3. Animación	60
4.4. Piano	61
4.4.1. Piano simple.....	62
4.4.2. Piano pantalla de colores	62
4.4.3. Piano colores y notas en pantalla.....	63
4.5. Mini Juego	65
Fuentes de referencia	68

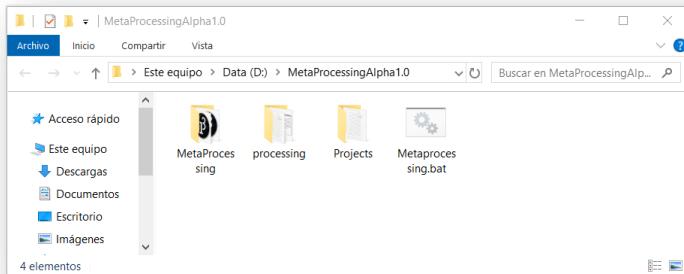
1. META_PROCESSING PRIMEROS PASOS

1.1. ¿Cómo abrir Meta_Procesing?

Meta_Processing fue desarrollado para que funcionara en los sistemas operativos: Windows, Mac y GNU/Linux. Los pasos para abrir Meta_Procesing varían un poco según el sistema operativo que se use, a continuación se describen los paso para cada sistema:

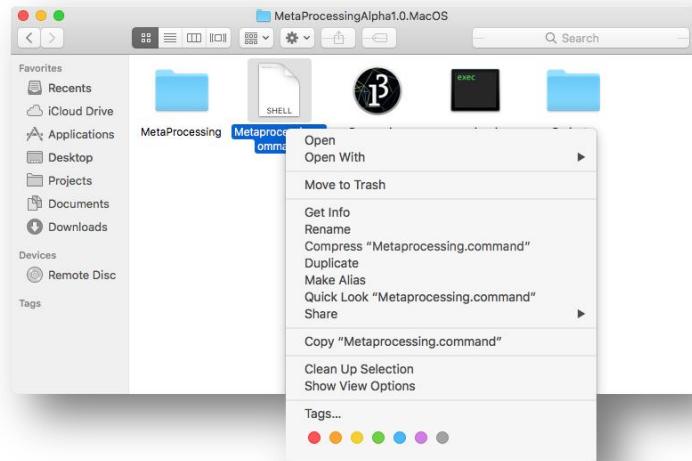
Windows

En Microsoft Windows se debe hacer doble clic en el archivo **Metaprocessing.bat**



Mac OS

En Mac Os se debe hacer clic con el botón derecho del ratón sobre el archivo **Metaprocesing.command** y seleccionar la opción: **Open o Abrir**

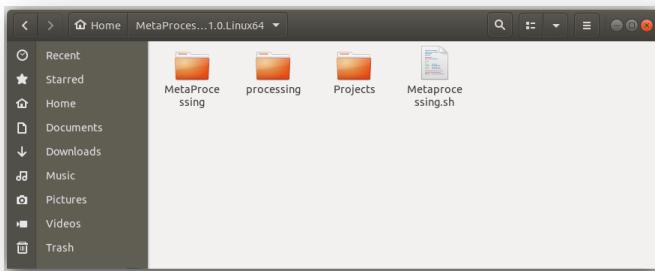


En la ventana que se abre se debe seleccionar la opción: **Open o Abrir**

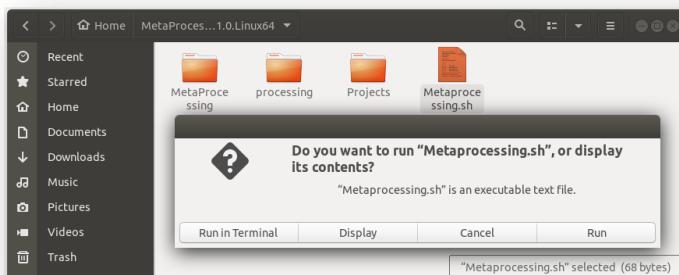


GNU/Linux

En GNU/Linux se debe hacer doble clic en el archivo **Metaprocessing.sh**



En la ventana que se abre se debe seleccionar la opción: **Run o Ejecutar**

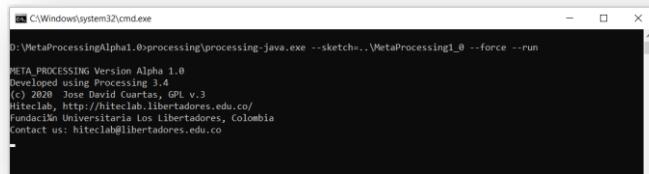


Si el script no abre al hacer doble clic sobre este, se puede ejecutar el siguiente comando en el terminal de Linux, para activar el anterior cuadro de dialogo.

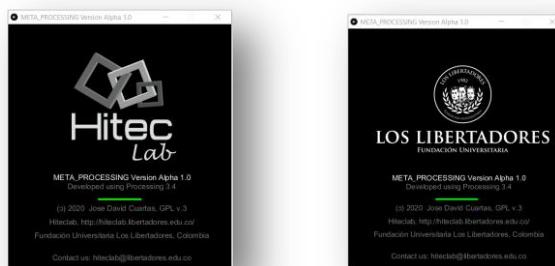
```
gsettings set org.gnome.nautilus.preferences executable-text-activation ask
```

1.2. Ventanas que se abren al iniciar Meta_Prosesing

Una vez se hace ejecuta el archivo Meta_Prosesing sin importar el sistema operativo en el que se esté trabajando primero se abre la ventana terminal donde se pueden ver mensajes que provienen de la ventana principal de Meta_Processing.

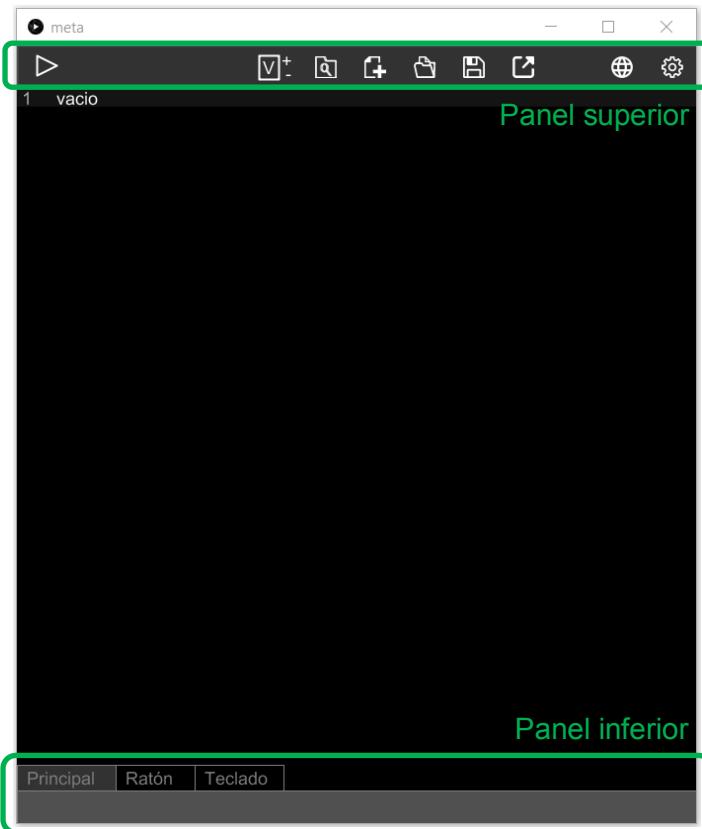


Luego se abre una segunda ventana con el splash animado de bienvenida a Meta_Processing. Al hacer clic sobre esta ventana o al cerrarla se abre la ventana principal de Meta_Processing.



1.3. Elementos básicos de la interface

En el panel superior están los botones: Ejecutar, Variables, Data, Nuevo, Abrir, Guardar, Exportar, Idiomas y Configuración.



En el panel inferior se encuentran las pestañas: Principal, Ratón y Teclado. Y se encuentra la barra

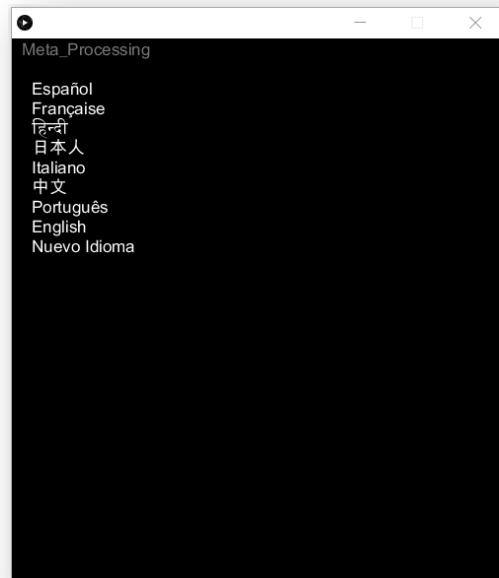
de descripción: en donde se muestra los nombres de los botones y el prototipo de las instrucciones.

1.4. ¿Cómo seleccionar Idiomas?

Para cambiar el idioma de **Meta_Processing** se debe hacer clic en el botón idiomas en la barra superior.



Luego en la ventana que se abre se debe hacer clic en el idioma deseado.



1.5. ¿Cómo ejecutar el código?

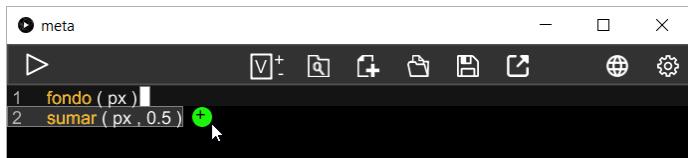
Para ejecutar el código se debe hacer clic en el botón ejecutar en la barra superior.



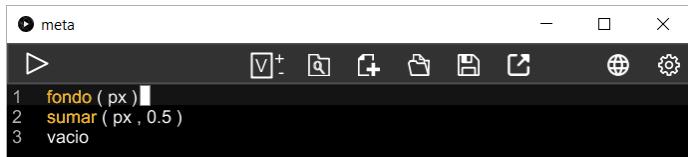
Se espera uno segundos y debe aparecer una nueva ventana en la que se ejecutará el código creado.

1.6. ¿Cómo agregar una línea de código?

Para agregar una línea de código se debe mover el cursor del ratón hasta que aparezca un círculo verde con el carácter más (+) en su interior.



Aparecerá una nueva línea de código una vez se haga clic en el círculo verde.



1.7. ¿Cómo eliminar una línea de código?

Para eliminar una línea de código se debe mover el cursor del ratón hasta que aparezca un círculo rojo con el carácter menos (-) en su interior, y se vea una línea gris que tacha toda la instrucción.

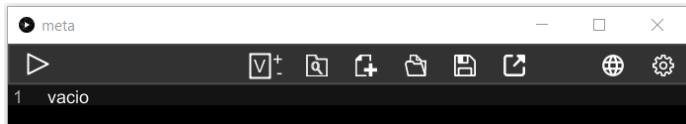


La línea desaparece una vez se hace clic.



1.8. ¿Cómo agregar instrucciones?

Para agregar una instrucción se debe hacer clic en la palabra que dice **vacio**.



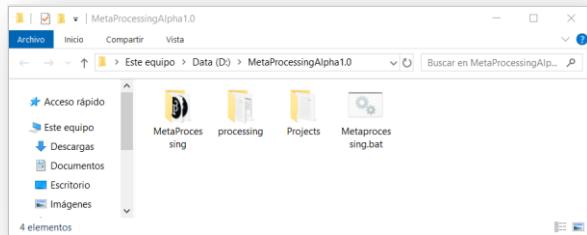
Una vez se hace esto, se abrirá una nueva ventana en donde aparecerán todas las instrucciones disponibles en Meta_Processing organizadas por categorías así:



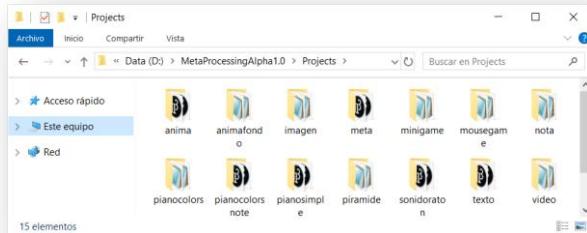
Cuando se ubica el cursor del ratón sobre alguna de estas instrucciones se resalta la instrucción, es este ejemplo se puede ver cómo se resalta la instrucción **triangulo**. Al hacer clic se abrirá una nueva ventana en la que se podrán ingresar las propiedades de cada instrucción. La descripción de cada una de estas instrucciones se hará en el capítulo 2.

1.9. ¿Cuál es la estructura de archivos y carpetas en Meta-Processing?

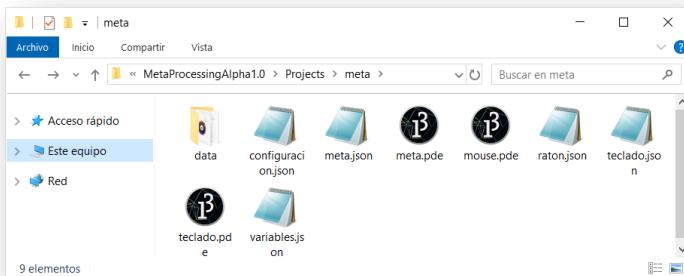
Dentro de la carpeta de Meta_Processing se encuentra el archivo que ejecuta Meta_Processing y tres sub-carpetas. La carpeta MetaProcessing contiene los archivos que le permiten funcionar. En la carpeta processing hay una distribución de este lenguaje que se usa para compilar y ejecutar cada uno de los proyectos que sean creados por el usuario.



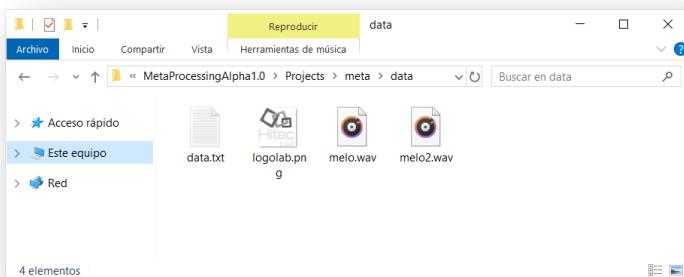
En la carpeta **Projects** contiene las carpetas de cada uno de los proyectos de programas escritos usando el entorno de programación Meta_Processing.



En la carpeta de cada proyecto se encuentran archivos .json y archivos .pde. Los .json contienen las instrucciones en lenguaje Meta_Processing y los archivos .pde contienen el código processing, y son generados cada vez se hace clic en el botón ejecutar.



Y en la carpeta **data** de cada proyecto se guardan los archivos que se usarán en la ejecución del programa como pueden ser imágenes, sonidos y videos.

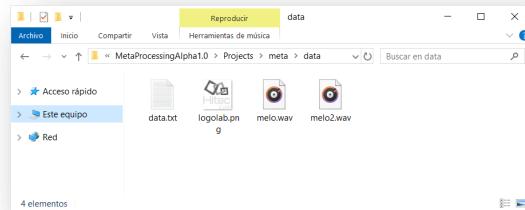


1.10. ¿Cómo abrir la carpeta data del proyecto actual?

Para abrir la carpeta data del proyecto actual, se debe hacer clic en el botón **data** en la barra superior.



Una vez se hace clic se abre en otra ventana la carpeta data del proyecto actual

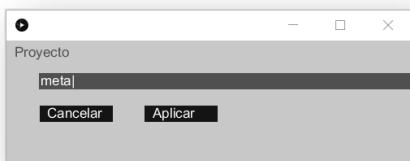


1.11. ¿Cómo crear un nuevo proyecto?

Para crear un nuevo proyecto se debe hacer clic en el botón **nuevo** en la barra superior.



En la ventana que se abre se debe escribir el nombre que se le quiere dar al nuevo proyecto y hacer clic en el botón aplicar.

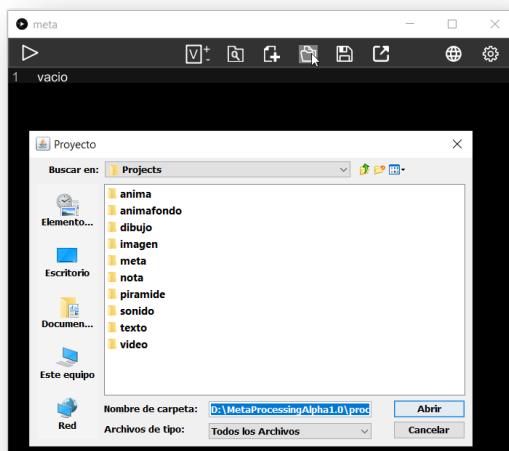


1.12. ¿Cómo abrir un proyecto?

Para abrir un proyecto se debe hacer clic en el botón **abrir** en la barra superior.



A continuación se abre una nueva ventana en la que se puede seleccionar la carpeta del proyecto que se quiere abrir y se hace clic en botón abrir.



1.13. ¿Cómo guardar el proyecto actual?

Para guardar el proyecto actual se debe hacer clic en el botón **guardar** en la barra superior.



1.14. ¿Cómo exportar el proyecto actual como aplicación?

Para exportar el proyecto actual como aplicación se debe hacer clic en el botón **exportar** en la barra superior.



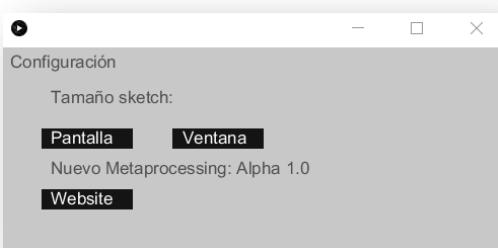
La aplicación se guarda en la subcarpeta llamada **app** dentro de la carpeta del proyecto actual.

1.15. Opciones de configuración

Para cambiar las opciones de configuración se debe hacer clic en el botón confirmación en la barra superior.



A continuación se abre una nueva ventana en la que se ofrecen dos opciones: cambiar el tamaño del sketch y revisar cuál es última versión de Meta_Processing.



La opción **Tamaño del sketch** tiene dos botones, **Pantalla** para hacer que la aplicación se ejecute en pantalla completa o el botón **Ventana** para que la aplicación se abra en una ventana, a la cual se le puede cambiar el tamaño manualmente.

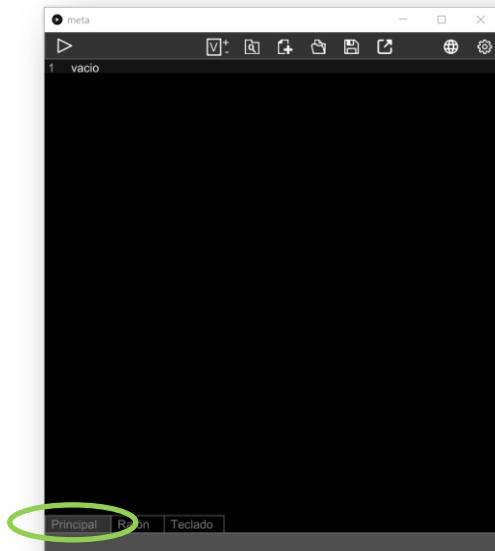
La opción **Nuevo Metaprocessing** permite revisar última versión de Meta_Processing publicada en internet. La información que aparece después de los dos puntos es la versión disponible para descargar. Si se desea descargar esa nueva versión se puede hacer clic en el botón Website que apunta a al sitio web oficial de descarga de Meta_Processing.

1.16. Funciones: principal, teclado y ratón

Para escribir el código en Meta_Processing, se pueden usar tres funciones: Principal, Ratón y Teclado. Cada una de estas funciones se selecciona haciendo clic en su pestaña correspondiente.

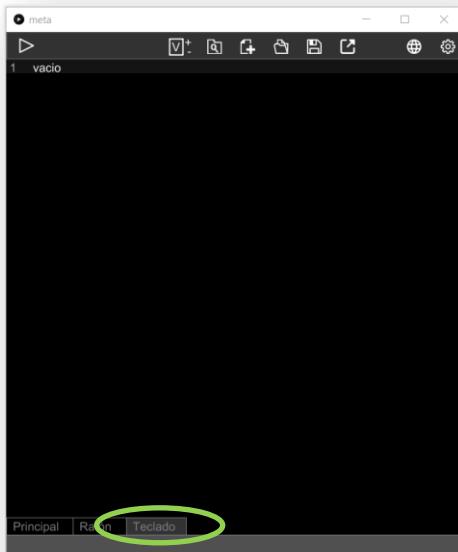
1.16.1. Principal

El código que se escribe en la pestaña **Principal** se ejecuta en un ciclo infinito, hasta que se cierre la ventana de la aplicación.



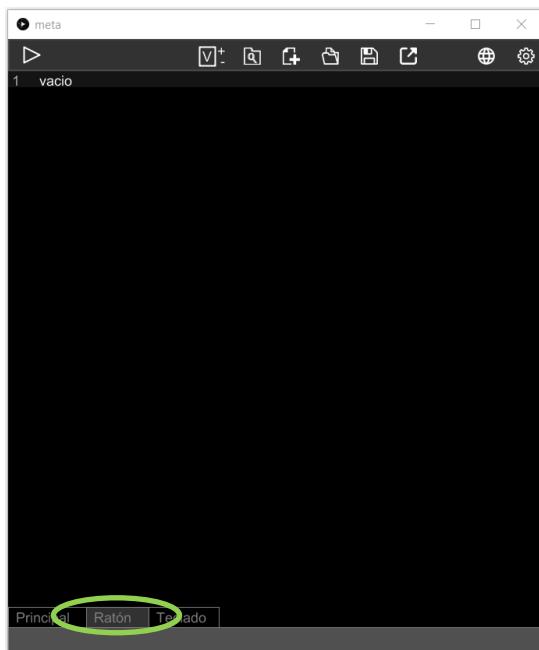
1.16.2. Teclado

El código que se escribe en la pestaña **Teclado** se ejecuta en el momento que se presiona cualquier tecla.



1.16.3. Ratón

El código que se escribe en la pestaña **Ratón** se ejecuta en el momento que se presiona cualquier botón del ratón.



2. INSTRUCCIONES BÁSICAS

En esta sección se abordarán las instrucciones básicas para programar con Meta_Processing. Como se explicó en el punto 1.8. para agregar una instrucción se debe hacer clic en la palabra que dice **vacío**.



Luego, se abrirá una nueva ventana en donde se muestran todas las instrucciones disponibles en Meta_Processing organizadas por categorías así:



2.1. Documentar el código

Documentar el código de programación es una de las primeras cosas que debe aprender cualquier persona que quiera aprender a programar. Para este propósito todos los lenguajes de programación permiten agregar líneas de **comentario**. La principal característica de esta línea de código es que no se ejecuta, está allí solo para darle información al programador sobre cómo funciona esa parte del código. Para agregar un comentario se debe hacer clic en la opción **comentario** dentro de la categoría de **Documentar**.



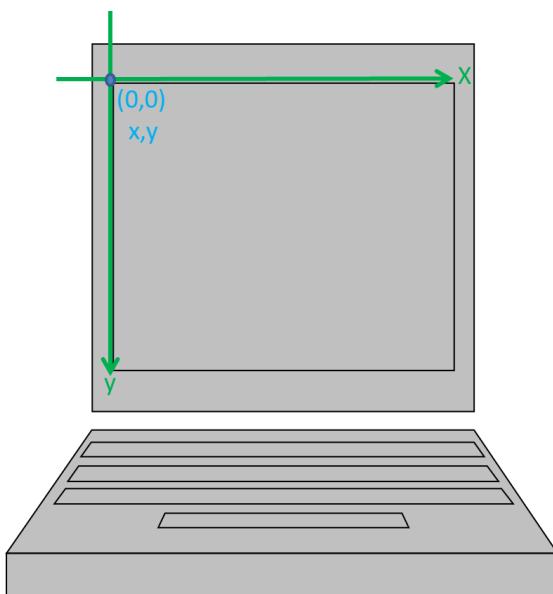
Entonces se abrirá una nueva ventana en la cual se escribe el comentario y se hace clic en **aplicar**.



2.2. Coordenadas de pantalla

Para poder hacer gráficos en pantalla es necesario primero conocer cómo funcionan las coordenadas de pantalla en Meta_Processing. La posiciones en pantalla se miden en pixels y cada pantalla tiene un cierto número de pixels en el eje X y en el eje Y.

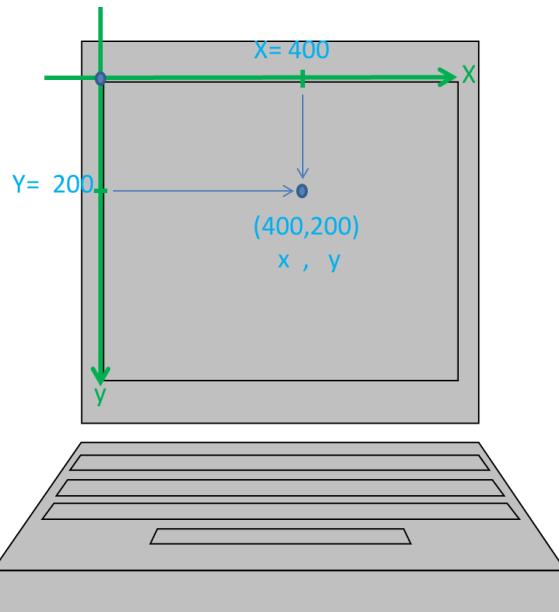
Como se puede observar en la gráfica 1, la esquina superior izquierda de la pantalla es el punto de origen del sistema de coordenadas. Este punto es la posición 0 en el eje X y la posición 0 en el eje Y. Las coordenadas siempre se organizan primero el valor en el eje X, luego una coma y después el valor en el eje Y, así pues este punto es (0,0).



Gráfica 1 Punto de origen en el sistema de coordenadas

Las posiciones en el eje X aumentan de izquierda a derecha y las posiciones en el eje Y aumentan de arriba hacia abajo. En la gráfica 2 se muestra un ejemplo para ilustrar un poco mejor este concepto.

Si se quiera ubicar el punto (400,200) en pantalla, lo que se hace es contar 400 pixels hacia la derecha desde el punto (0,0) de la pantalla y contar 200 pixels de hacia abajo desde el punto (0,0) de la pantalla. Así se ubica el punto (400,200).



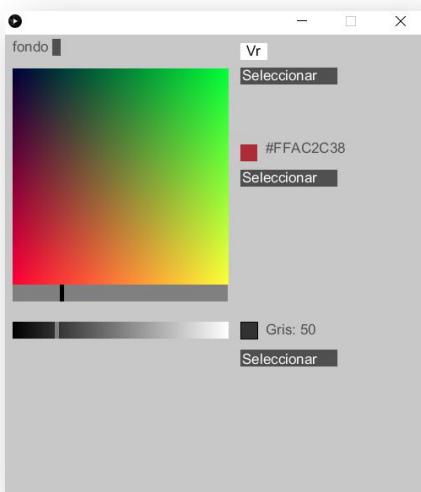
Gráfica 2 Punto en la posición 400 en X y 200 en Y de la pantalla

2.3. Instrucciones para gráficos en pantalla

Entre algunas de las instrucciones para mostrar en pantalla se encuentra: línea, triángulo, rectángulo, elipse, texto, imagen. A continuación se explicará cómo usar cada una de ellas.

2.3.1. fondo

La instrucción **fondo** sirve para definir el color de fondo de toda la venta de la aplicación. Esta instrucción borra todo que se esté mostrando en pantalla y deja toda la pantalla del color seleccionado.



2.3.2. Línea

La instrucción **Línea** sirve para dibujar una línea en pantalla. Para traza una línea se requiere definir la posición x,y del punto donde inicia la línea y la posición x,y del punto donde termina la línea.



2.3.3. rectángulo

La instrucción **rectángulo** sirve para dibujar un rectángulo en pantalla. Para usar esta instrucción se debe definir en las dos primeras casillas la posición x,y de la esquina superior izquierda desde donde se dibujará el cuadrado, y en las dos casillas siguientes definir su ancho y alto.



2.3.4. elipse

La instrucción **elipse** sirve para dibujar una elipse en la pantalla. Para usar esta instrucción se debe definir en las dos primeras casillas el punto central x,y desde donde se dibujará la elipse y en las dos casillas siguientes definir su ancho y alto.



2.3.5. triángulo

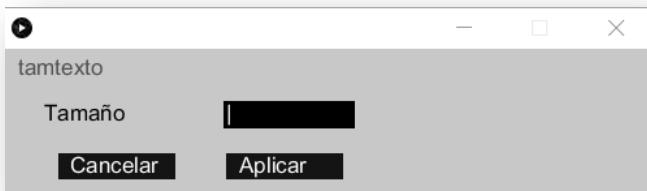
La instrucción **triángulo** sirve para dibujar un triángulo en pantalla. Para dibujar cualquier triángulo se requiere definir los tres puntos correspondientes a cada una de sus esquinas. Para usar esta instrucción se debe definir en las dos primeras casillas la posición del primer punto, en las dos siguientes casillas definir la posición del segundo punto y en las últimas dos casillas de finir la posición del tercer punto.



2.3.6. tamtexto

La instrucción **tamtexto** sirve para definir el tamaño de la fuente a la hora de mostrar un texto en pantalla. Para usar esta instrucción se debe llenar la casilla de tamaño con el número correspondiente al tamaño de fuente que se desea aplicar.

Para que esta instrucción tenga efecto se debe agregar antes de la instrucción **texto**.



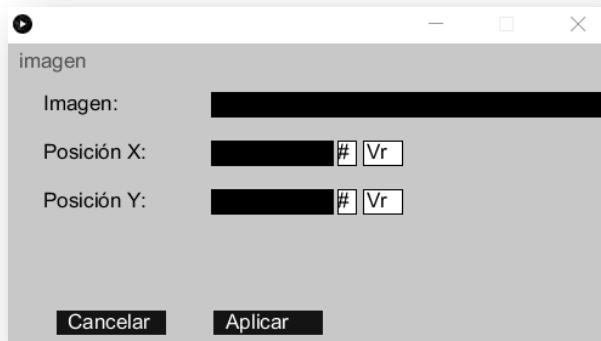
2.3.7. texto

La instrucción texto sirve para mostrar texto en pantalla. Para usar esta instrucción se debe escribir en la primera casilla el texto que se desea mostrar, y definir en las dos siguientes casillas la posición x,y de la esquina inferior izquierda desde donde se comenzará a mostrar el texto en pantalla.



2.3.8. imagen

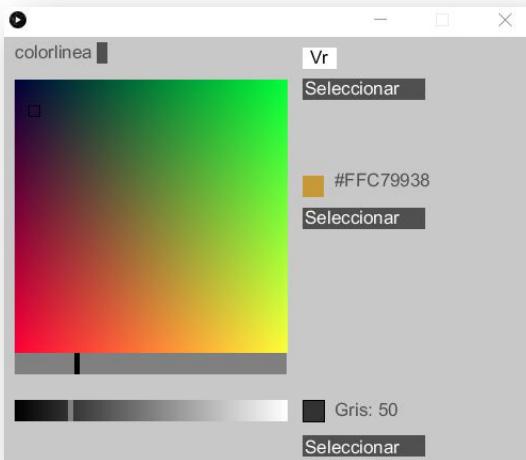
La instrucción **imagen** sirve para mostrar una imagen en pantalla. Para usar esta instrucción primero se debe guardar la imagen que se quiere usar dentro de la carpeta data del proyecto, luego se debe seleccionar la imagen que fue guardada en la carpeta data y por último definir en las dos siguientes casillas la posición x,y de la esquina superior izquierda desde donde se comenzará a mostrar la imagen en pantalla.



2.3.9. colorlinea

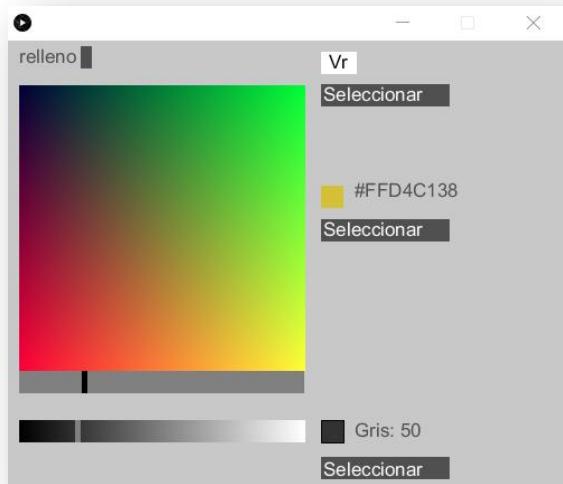
La instrucción **colorlinea** sirve para definir el color de las líneas y color del borde de las figururas rectángulo, elipse y triángulo. Para asignar el color de línea usando esta instrucción se puede, utilizar el selector de color, o el selector de escala de grises y hacer clic en el botón aplicar. También se puede

usar una variable para cambiar de forma dinámica el color. Para que esta instrucción tenga efecto se debe agregar antes de las instrucciones con las que se dibuja líneas, o figuras en pantalla.



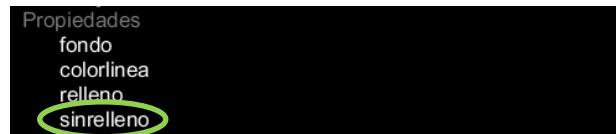
2.3.10. relleno

La instrucción **relleno** sirve para definir el color de relleno de las figuraras rectángulo, elipse y triángulo. Para asignar el color de relleno usando esta instrucción se puede, utilizar el selector de color, o el selector de escala de grises y hacer clic en el botón aplicar. También se puede usar una variable para cambiar de forma dinámica el color. Para que esta instrucción tenga efecto se debe agregar antes de las instrucciones con las que se dibuja figuras en pantalla.



2.3.11. sinrelleno

La instrucción **sinrelleno** sirve para quitar el relleno de las figuras rectángulo, elipse y triángulo. Esta se verán transparentes y solo se verá su borde. Para usar esta instrucción solo se necesita que en la ventana de agregar instrucción, se haga clic en sobre la instrucción **sinrelleno** dentro de la categoría Propiedades y automáticamente se agregará esta línea al código del proyecto.



2.3.12. sinlinea

La instrucción **sinlinea** sirve para quitar el borde de las figuras rectángulo, elipse y triángulo. Si se agrega esta instrucción antes de la instrucción línea entonces no se mostrará en pantalla la línea. Para usar esta instrucción solo se necesita que en la ventana de agregar instrucción, se haga clic en sobre la instrucción **sinlinea** dentro de la categoría Propiedades y automáticamente se agregará esta línea al código del proyecto.

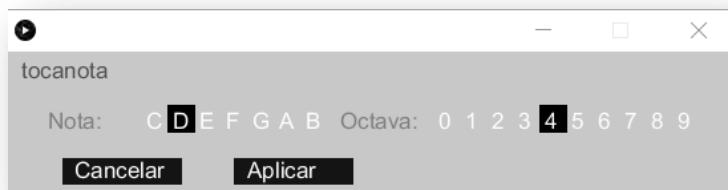


2.4. Instrucciones multimedia

Entre algunas de las instrucciones multimedia que se puede usar con Meta_Processing se encuentra: tocanota, sonido y video. A continuación se explicará cómo usar cada una de ellas.

2.4.1. tocanota

La instrucción **tocanota** sirve para reproducir el sonido de una nota de la escala musical. Para usar esta instrucción se debe seleccionar la nota que se quiere tocar y luego seleccionar la octava que se quiere que suene la nota. Por último se debe hacer clic en el botón **Aplicar**.



2.4.2. sonido

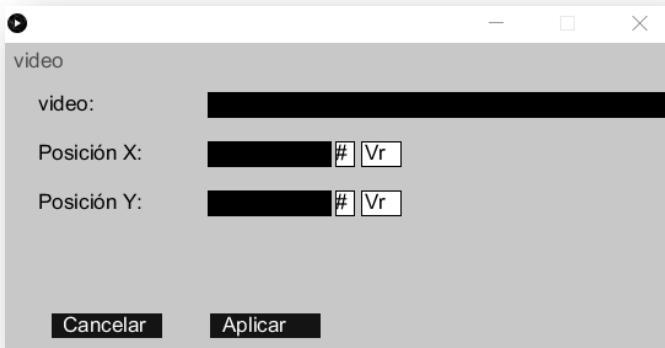
La instrucción **sonido** sirve para reproducir un archivo de sonido en formato wav o mp3. Para usar esta instrucción primero se debe guardar dentro de la carpeta data del proyecto, el archivo de sonido que se quiere usar, luego se debe hacer clic en el botón **Seleccionar** para elegir el archivo que fue guardado previamente en la carpeta data y por último se debe hacer clic en el botón **Aplicar**.



2.4.3. video

La instrucción **video** sirve para reproducir en pantalla un archivo de video en formato mov, avi o mpg. Para usar esta instrucción primero se debe guardar dentro de la carpeta data del proyecto, el

archivo de video que se quiere usar, luego se debe seleccionar el archivo que fue guardado en la carpeta data y por último hacer clic en el botón **Aplicar**.



3. VARIABLES Y CONDICIONES

En esta sección se abordara los conceptos de variables y condiciones, los cuales son fundamentales a la hora de aprender a programar.

3.1. Variables

Una variable es un espacio de memoria reservado para almacenar un valor que va cambiando durante el trascurso de la ejecución del programa. En Meta_Processing se hay dos tipos de variables: Las variables del sistema y las variables creadas por el usuario.

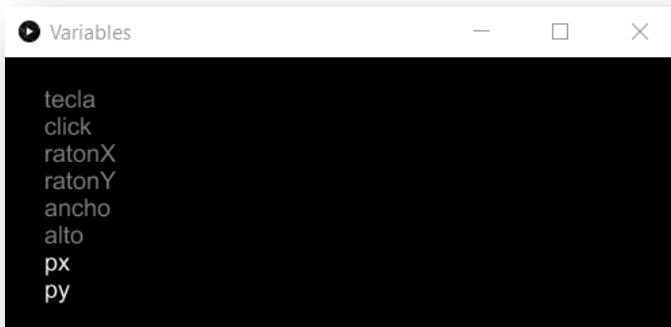
Las variables del sistema son: **tecla**, **click**, **ratonX**, **ratonY**, **ancho** y **alto**. La variable **tecla** almacena el valor de la última tecla presionada en el teclado. La variable **click** amacena el valor del último botón presionado en el ratón. La variable **ratonX** almacena la posición del ratón en el eje X. La variable **ratonY** almacena la posición del ratón en el eje Y. La variable **ancho** almacena el valor del ancho de la pantalla en la que ese está ejecutando el código. Y la variable **alto** almacena el valor del alto de la pantalla en la que ese está ejecutando el código.

3.1.1. ¿Cómo se mira el listado de variables?

Para mirar el listado de variables del proyecto se debe hacer clic en el botón variable.



En la ventana que se abre se verán las variables que se están usando. Las que se muestran en color gris son las variables del sistema y las que se muestran de color blanco son las variables creadas por el usuario.



3.1.2. ¿Cómo se crea una variable?

Para crear una variable se debe hacer clic en el ícono mas (+) que está a un lado del botón variable.



En la ventana que se abre se debe escribir el nombre que se le quiere dar de la variable que se va a crear, en este ejemplo se le da el nombre **px**.

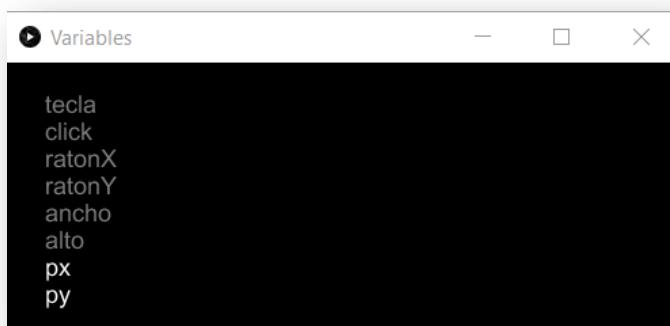


3.1.3. ¿Cómo se elimina una variable?

Para eliminar una variable se debe hacer clic en el ícono menos (-) que está a un lado del botón variable.



En la ventana que se abre se debe hacer clic sobre el nombre de la variable que se quiere eliminar. Solo se pueden eliminar las variables que han sido creadas por el usurario, es decir, solo le pueden eliminar las variables que su texto es color blanco. Las variables en color gris son las variables del sistema y no se pueden eliminar.

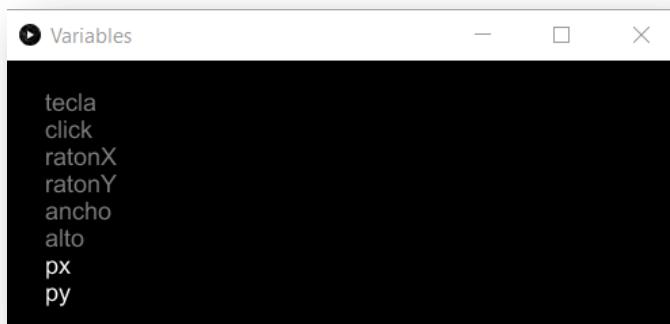


3.1.4. ¿Cómo se inicializa una variable?

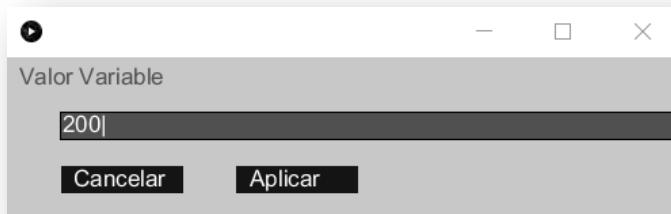
Para inicializar una variable creada por el usuario se debe hacer clic en el botón variable.



En la ventana que se abre se hace clic sobre la variable que se quiere inicializar.



Hecho esto aparecerá otra ventana en la que se debe escribir el valor con el que se quiere inicializar la variable. En este ejemplo se inicializa la variable con el valor 200.



3.1.5. ¿Cómo asignarle un nuevo valor a una variable?

Dentro del código se le puede asignar un nuevo valor a una variable, para ello se debe agregar la instrucción **asignar** que se encuentra dentro de la categoría Matemática en la ventana de agregar instrucción.

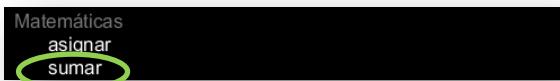


Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere asignar un nuevo valor y en la segunda casilla se escribe el valor que se le va a asignar.

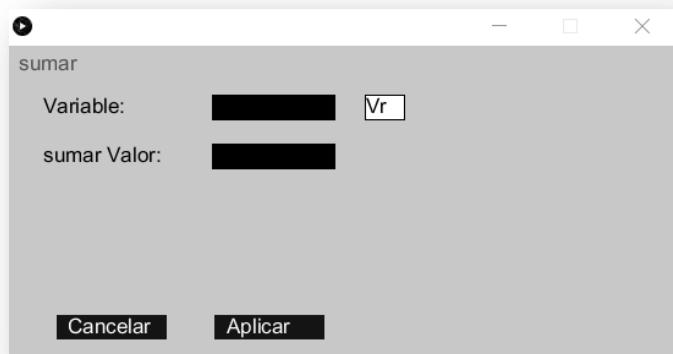


3.1.6. ¿Cómo se le suma un valor a una variable?

Dentro del código se le puede sumar un valor a una variable, para ello se debe agregar la instrucción **sumar** que se encuentra dentro de la categoría Matemática en la ventana de agregar instrucción.



Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere sumar el valor y en la segunda casilla se escribe el valor que se le va a sumar.

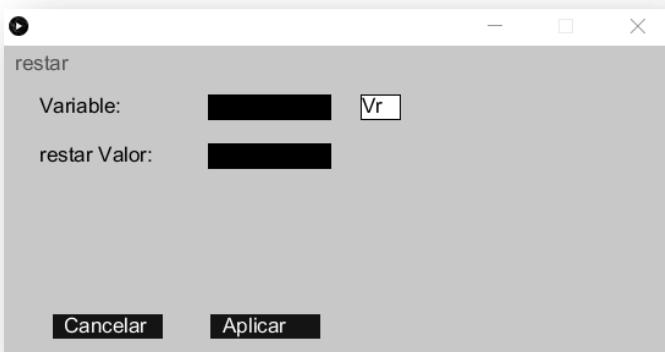


3.1.7. ¿Cómo se le resta un valor a una variable?

Dentro del código se le puede restar un valor a una variable, para ello se debe agregar la instrucción **restar** que se encuentra dentro de la categoría Matemática en la ventana de agregar instrucción.



Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere restar el valor y en la segunda casilla se escribe el valor que se le va a restar.



3.1.8. ¿Cómo se le asigna un valor aleatorio a una variable?

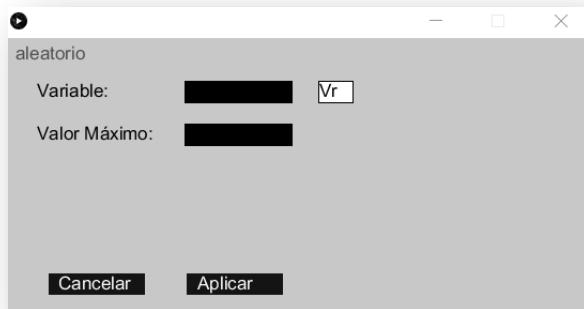
Para asignarle un valor aleatorio a una variable....

Dentro del código se le puede asignar un valor aleatorio a una variable, para ello se debe agregar la instrucción **aleatorio** que se encuentra dentro de la categoría Matemática en la ventana de agregar instrucción.



Luego en la ventana que se abre, en la primera casilla se selecciona la variable a la que se le quiere asignar el valor aleatorio y en la segunda casilla se

escribe el valor máximo que se podría generar aleatoriamente. De manera que se generaría un valor aleatorio entre 0 y el valor máximo que se le asigne a la instrucción.



3.2. Condiciones

Las condiciones son un tipo de estructuras algorítmicas que le permiten al programa tomar decisiones según se cumpla una condición.

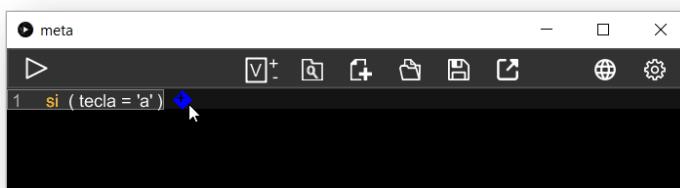
Para agregar una condición se debe agregar la instrucción **si** que se encuentra dentro de la categoría Estructuras en la ventana de agregar instrucción.



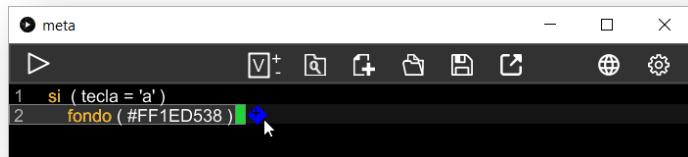
Luego en la ventana que se abre en la primera casilla se puede escribir un valor o seleccionar una variable. Se debe seleccionar un operador, el cual puede ser igual (=), menor que (<), mayor que (>) o diferente (!=). En la segunda casilla se puede escribir un valor, seleccionar una variable, elegir una tecla o elegir uno de los botones del ratón.



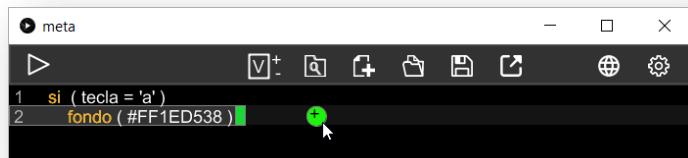
Una vez se hace clic en el botón Aplicar se podrá ver la condición en la ventana de código de Meta_Procesing. Para agregar una instrucción dentro de la condición, se debe mover el cursor del ratón hasta que aparezca un rombo azul con el carácter más (+) en su interior.



Una vez se hace clic aparecerá la nueva línea de código vacía y se hace clic para asignarle la instrucción deseada. Si se quiere crea una instrucción más dentro de la condición entonces , se debe mover el cursor del ratón hasta que aparezca un rombo azul con el carácter más (+) en su interior.

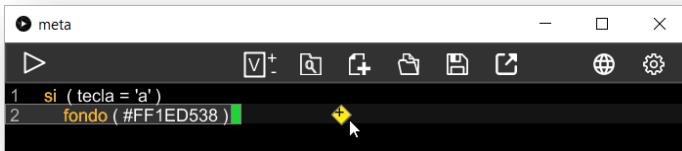


Si una vez se termina de agregar las instrucciones dentro de la condición, lo que se quiere es agregar otra línea de código pero por fuera de la condición, entonces se debe mover el cursor del ratón hasta que aparezca un circulo verde con el carácter más (+) en su interior.

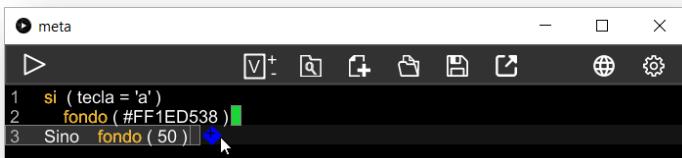


Si por el contrario lo que se quiere es agregar una línea para el caso cuando no se cumpla dicha condición entonces se podrán agregar líneas dentro del sino. Para esto se debe mover el cursor del ratón

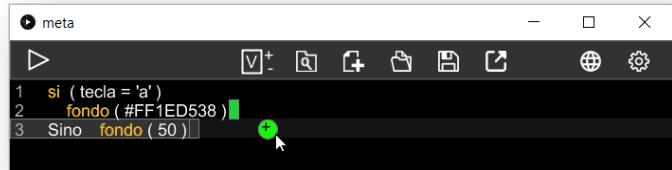
hasta que aparezca un rombo amarillo con el carácter más (+) en su interior.



Una vez se seleccione la instrucción que se quiere usar se vería que adelante de la instrucción aparece la palabra sino. Para cuando se quiera agregar más instrucciones dentro del sino se debe mover el cursor del ratón hasta que aparezca un rombo azul con el carácter más (+) en su interior.

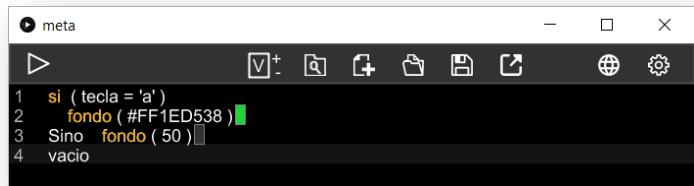


Si por el contrario se quiere es agregar otra línea de código pero por fuera del sino, entonces se debe mover el cursor del ratón hasta que aparezca un circulo verde con el carácter más (+) en su interior.



```
meta
▶ [V+] [?] [+] [!] [!] [!] [!] [!]
1 si ( tecla = 'a' )
2 fondo ( #FF1ED538 )
3 Sino fondo ( 50 )
```

Al hace esto entonces aparecerá una nueva línea vacía por fuera de las instrucciones de la condición.



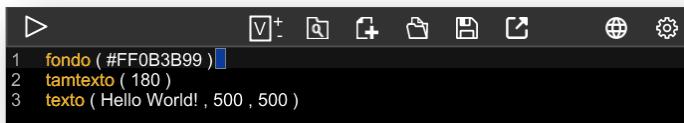
```
meta
▶ [V+] [?] [+] [!] [!] [!] [!] [!]
1 si ( tecla = 'a' )
2 fondo ( #FF1ED538 )
3 Sino fondo ( 50 )
4 vacio
```

4. EJEMPLOS DE CÓDIGO CON META_PROCESSING

A continuación se presentan una serie de ejemplos de cómo usar Meta_Processing. El primero de ellos nos permite experimentar con la función teclado, el segundo nos permite dibujar círculos en pantalla al presionar el ratón. El tercero es un ejemplo de cómo se puede crear una animación. El cuarto nos muestra tres formas de programar un piano. Y el último nos muestra como crea un sencillo minijuego.

4.1. Ejemplo básico con el Teclado

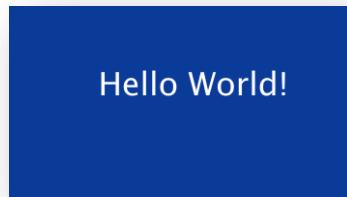
El siguiente código se debe escribir en la pestaña **Teclado** de manera que se ejecute en el momento que se presione cualquier tecla.



The screenshot shows the Processing IDE interface. At the top, there is a toolbar with various icons: play, stop, clear, save, and settings. Below the toolbar, the code is displayed in a text editor:

```
1 fondo (#FF0B3B99)
2 tamtexto ( 180 )
3 texto ( Hello World! , 500 , 500 )
```

La idea es que al presionar cualquier tecla, se pone la pantalla azul y aparece en la pantalla un texto en color blanco.



4.2. Ejemplo básico con el Ratón

El siguiente código se debe escribir en la pestaña **Ratón** de manera que se ejecute en el momento que se presione cualquiera de los botones del ratón.

```
▶ [ ] + [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]  
1 sinlínea  
2 relleno ( #FFFDD738 )  
3 ellipse ( ratonX , ratonY , 80 , 80 )
```

The screenshot shows the Processing IDE interface. At the top, there's a toolbar with various icons. Below it is a code editor window containing three lines of code. The first line is "sinlínea", the second is "relleno (#FFFDD738)", and the third is "ellipse (ratonX , ratonY , 80 , 80)". The "relleno" command has a color swatch next to it, and the "ellipse" command has parameters for width and height.

La idea es que al presionar cualquiera de los botones del ratón se dibujen círculos amarillos en pantalla.



4.3. Animación

A continuación se muestra un ejemplo de cómo se puede hacer una animación en Meta_Processing. Primero se debe crear dos variables una llamada **px** y otra llamada **py**. La variable **px** con el valor 10.



Mientras que la variable **py** se debe inicializar con el valor 950.



Luego se debe agregar el siguiente código:

A screenshot of the Meta_Procesing code editor. The title bar says "anima". The code area contains the following pseudocode:

```
1 fondo ( px )■
2 sumar ( px , 0.5 )
3 restar ( py , 0.5 )
4 relleno ( #FFF2E138 )■
5 ellipse ( 200 , py , 200 , 200 )
6 relleno ( #FF0B259A )■
7 rectangulo ( 0 , 800 , ancho , 400 )
```

The code uses color-coded keywords: "fondo", "sumar", "restar", "relleno", "ellipse", and "rectangulo". The numbers "200", "800", and "400" are also highlighted in yellow.

Al hacer clic en el botón ejecutar se vería es que la animación empezaría recreando una escena

nocturna en el mar, y lentamente irá amaneciendo hasta tener el firmamento completamente iluminado.

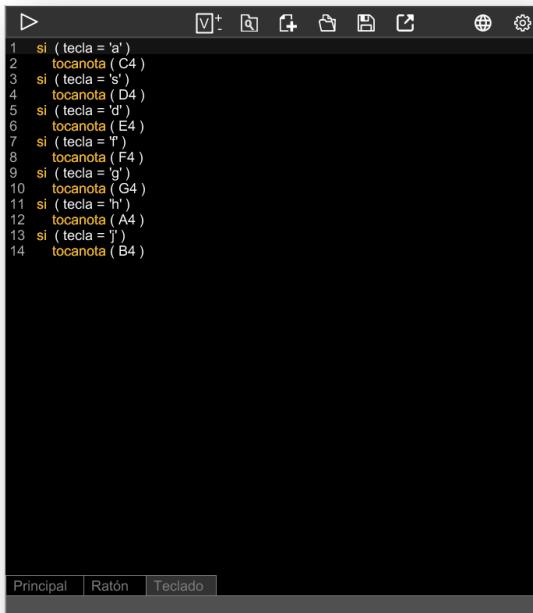


4.4. Piano

A continuación se muestra tres ejemplos de cómo hacer un piano en Meta_Processing. El primero es un piano simple, el segundo es un piano que además cambia el color de la pantalla y el tercero es un piano que cambia el color de la pantalla y muestra un texto con la nota que suena.

4.4.1. Piano simple

Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:



```
1 si ( tecla = 'a' )
2   tocanota ( C4 )
3 si ( tecla = 's' )
4   tocanota ( D4 )
5 si ( tecla = 'd' )
6   tocanota ( E4 )
7 si ( tecla = 'f' )
8   tocanota ( F4 )
9 si ( tecla = 'g' )
10  tocanota ( G4 )
11 si ( tecla = 'h' )
12  tocanota ( A4 )
13 si ( tecla = 'j' )
14  tocanota ( B4 )
```

Con este código se puede tocar las 7 notas musicales usando las teclas a, s, d, f, g, h, j. Para que funcione bien el teclado no puede estar en modo mayúsculas.

4.4.2. Piano pantalla de colores

Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:



```
1 si ( tecla = 'a' )
2   tocanota ( C4 )
3   fondo ( #FF2AD538 ) █
4 si ( tecla = 's' )
5   tocanota ( D4 )
6   fondo ( #FFC72538 ) █
7 si ( tecla = 'd' )
8   tocanota ( E4 )
9   fondo ( #FF3348AE ) █
10 si ( tecla = 'f' )
11   tocanota ( F4 )
12   fondo ( #FFFCD138 ) █
13 si ( tecla = 'g' )
14   tocanota ( G4 )
15   fondo ( #FFF48C0E ) █
16 si ( tecla = 'h' )
17   tocanota ( A4 )
18   fondo ( #FFEF1EC8 ) █
19 si ( tecla = 'j' )
20   tocanota ( B4 )
21   fondo ( #FF57938D ) █
```

Con este código se puede tocar la 7 notas musicales usando las teclas a, s, d, f, g, h, j. Para que funcione bien el teclado no puede estar en modo mayúsculas.

4.4.3. Piano colores y notas en pantalla

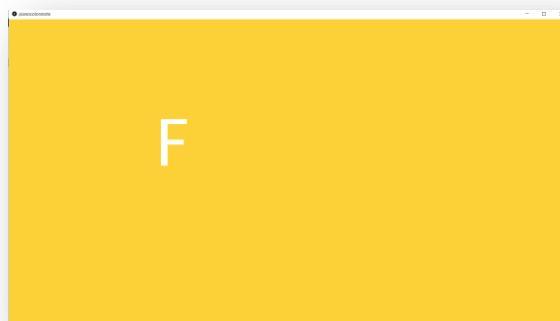
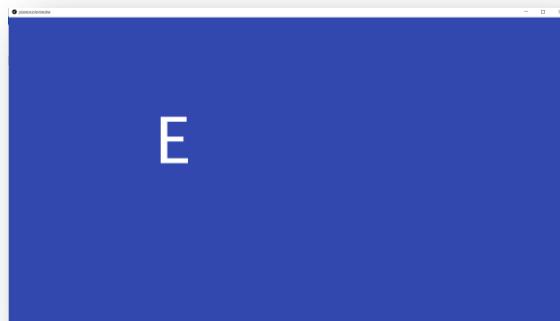
Para programar este piano se debe agregar en la pestaña **Teclado** el siguiente código:

The screenshot shows the Processing IDE interface. The top bar contains icons for play, stop, and various file operations. The main area displays a script with numbered lines from 1 to 29. The script uses the `tocanota` and `texto` functions to play notes and display text based on key presses ('a' through 'j'). The background of the code editor has a grid pattern where certain words like `tocanota` and `fondo` are highlighted in different colors (green, red, blue, yellow, orange, purple, teal). The bottom of the window has tabs for "Principal", "Ratón", and "Teclado".

```
1 tamtexto ( 220 )
2 si ( tecla = 'a' )
3 tocanota ( C4 )
4 fondo ( #FF2AD538 ) █
5 texto ( C , 500 , 500 )
6 si ( tecla = 's' )
7 tocanota ( D4 )
8 fondo ( #FFC72538 ) █
9 texto ( D , 500 , 500 )
10 si ( tecla = 'd' )
11 tocanota ( E4 )
12 fondo ( #FF3348AE ) █
13 texto ( E , 500 , 500 )
14 si ( tecla = 'f' )
15 tocanota ( F4 )
16 fondo ( #FFFCDD138 ) █
17 texto ( F , 500 , 500 )
18 si ( tecla = 'g' )
19 tocanota ( G4 )
20 fondo ( #FF48C0E ) █
21 texto ( G , 500 , 500 )
22 si ( tecla = 'h' )
23 tocanota ( A4 )
24 fondo ( #FFEF1EC8 ) █
25 texto ( A , 500 , 500 )
26 si ( tecla = 'j' )
27 tocanota ( B4 )
28 fondo ( #FF68B0A0 ) █
29 texto ( B , 500 , 500 )
```

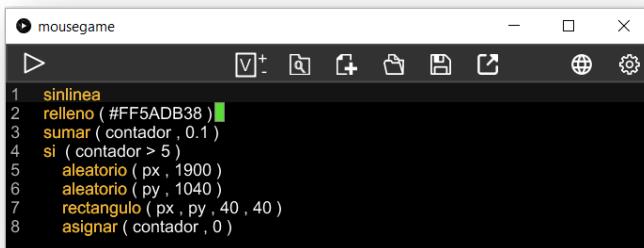
Con este código se puede tocar las 7 notas musicales usando las teclas a, s, d, f, g, h, j. Para que funcione bien el teclado no puede estar en modo mayúsculas.

El piano al ejecutarse se vería así:



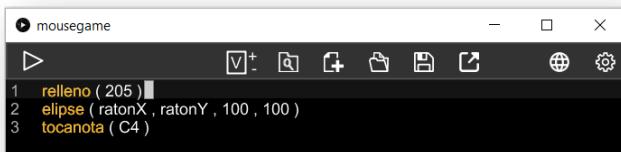
4.5. Mini Juego

A continuación se muestra un ejemplo de cómo se puede hacer un mini juego en Meta_Processing. En la pestaña **Principal** se debe agregar el siguiente código:



```
mousegame
V+ A S F S D G E
1 sinlinea
2 relleno ( #FF5ADB38 )
3 sumar ( contador , 0.1 )
4 si ( contador > 5 )
5 aleatorio ( px , 1900 )
6 aleatorio ( py , 1040 )
7 rectangulo ( px , py , 40 , 40 )
8 asignar ( contador , 0 )
```

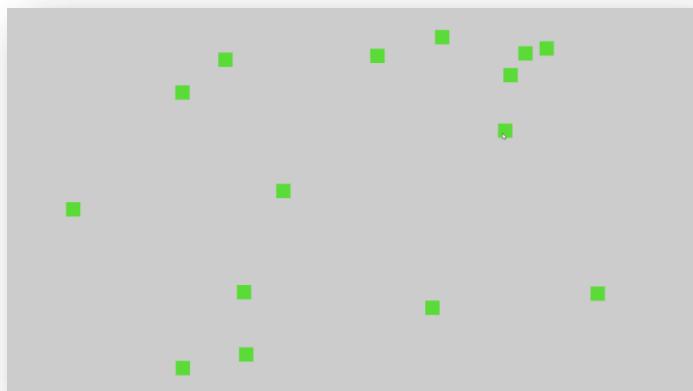
Y en la pestaña **Ratón** se debe agregar el siguiente código:



```
mousegame
V+ A S F S D G E
1 relleno ( 205 )
2 ellipse ( ratonX , ratonY , 100 , 100 )
3 tocanota ( C4 )
```

El juego consiste en una serie de cuadrados verdes que van apareciendo aleatoriamente en la pantalla. El propósito del juego es tratar hacer desaparecer todos los cuadros verdes haciendo clic sobre ellos.

El juego al ejecutarse se ve algo así:



Fuentes de referencia

Cuartas, J.D. (2014). Digitópolis I: Diseño de Aplicaciones Interactivas para Creativos y Comunicadores. Bogotá: Universidad los libertadores.

Cuartas, J.D. (2017). Programar el mundo en el contexto de las tecnologías libres y las culturas Hacker-Maker. Caso de estudio: Hitec Lab. (Tesis doctoral). Doctorado en Diseño y Creación. Universidad de Caldas, Manizales.

Hitec Lab (2020). Hitec Lab Homepage. Recuperado de <http://hiteclab.libertadores.edu.co/>

Hitec Lab (2020). Meta_Processing. Recuperado de https://github.com/hiteclab/Meta_Processing

Libertadores, L. (2019). Institución Universitaria Los Libertadores. Recuperado de <http://www.ulibertadores.edu.co/>

- Maeda, J. (1999). Design by numbers. Cambridge, Mass: MIT Press.
- MIT Medialab (20203). Design by numbers. Recuperado de <https://dbn.media.mit.edu/>
- MIT Medialab (2020). Scratch - Imagine, Program, Share. Recuperado de <https://scratch.mit.edu/>
- Processing. (2019). Processing. Recuperado de <http://www.processing.org/>
- Program.ar (2020). Pilas Bloques. Recuperado de <http://program.ar/pilas-bloques-secundaria/>
- Victor, B. (2012). Bret Victor - Inventing on a Principle [Archivo de video]. Recuperado de <https://www.youtube.com/watch?v=a-OyoVcbwWE&feature=youtu.be>
- Victor, B. (2013). Bret Victor - Stop Drawing Dead Fish [Archivo de video]. Recuperado de <https://www.youtube.com/watch?v=ZfytHvgHybA>

