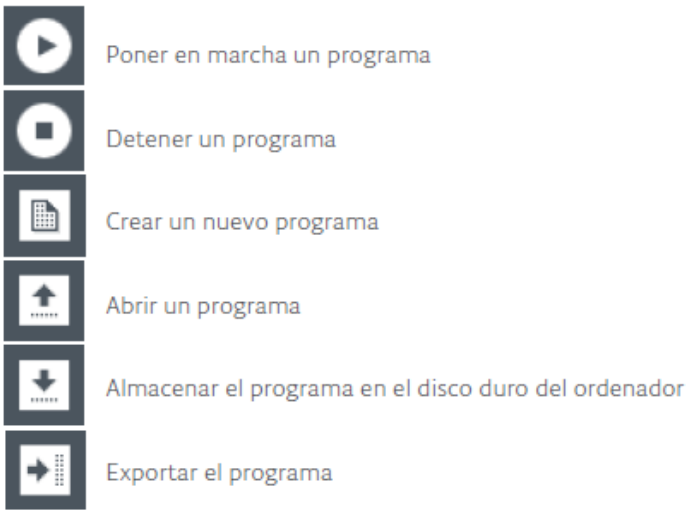
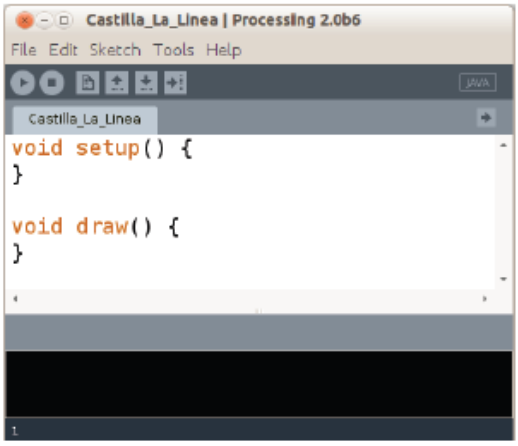
PROCESSING

Processing es un IDE (Integrated Development Environment – Entorno de Desarrollo Integrado), software que se encarga de traducir el lenguaje humano en el lenguaje máquina.

El interfaz del programa es el siguiente:

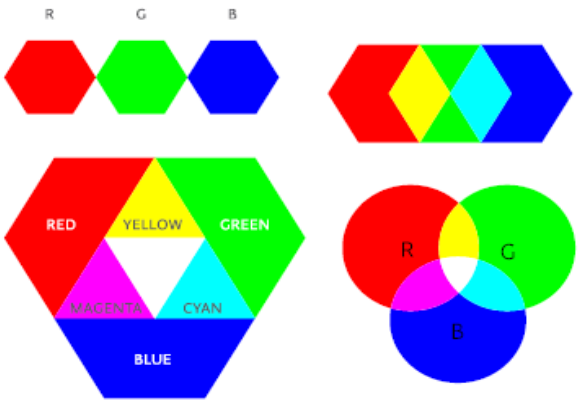




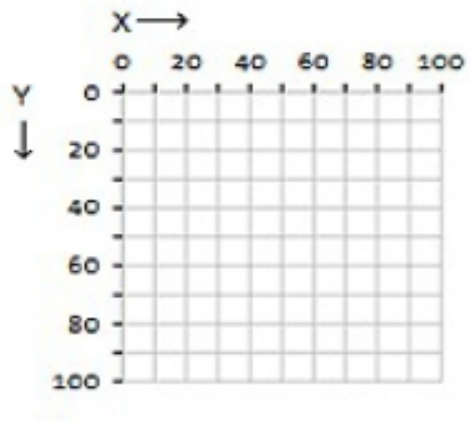
Antes de iniciar su utilización, hay que saber algunos aspectos importantes.

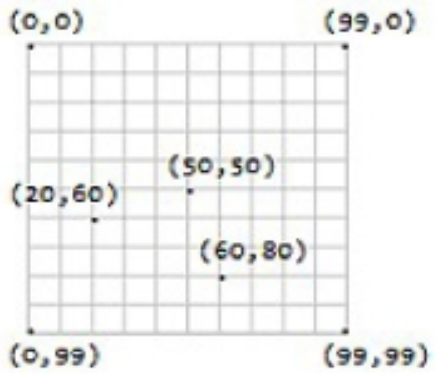
Pantalla y píxeles: Las pantallas están hechas de punto que llamamos píxeles. Por lo general, cuantos más píxeles tenga una pantalla, mejor será. Expresamos la calidad de la pantalla en función de sus píxeles. Esa cantidad es lo que llamamos resolución. Viene en función de su anchura y altura. Por ejemplo: 640x480; 800x600; 1024x768.

Colores: Cada píxel se puede iluminar de muchos colores diferentes. Los colores se expresan como una combinación de tres colores diferentes: Rojo (R), Verde (G), Azul (B). La cantidad de color para cada componente se expresa con un número entre 0 y 255.



Coordenadas: La pantalla es un gráfico de píxeles, cada uno registrado para una única coordenada (x,y). El origen (0,0) está en la esquina izquierda arriba de la pantalla. Hacia abajo sería +y y a la derecha +x. La asignación de puntos es parecido al juego “combate naval”.





1.- PRÁCTICAS

1.- Abre el programa y escribe el siguiente código:

line(0, 0, 100,100);

Pulsa “Ejecutar”.

Recuerda:
No debes de olvidar, que cada línea de instrucción, debe finalizar con un punto y coma (;). Además, indicarte que las ventanas de dibujo, por defecto, si no se indica lo contrario al programa es 100x100 píxeles.

2.- Modifica el programa con el siguiente texto:

```
background(255);
line(0, 0, 100, 100);
stroke(0, 0, 255);
line(0, 100, 100, 0);
```

Pulsa “Ejecutar”. ¿Qué aparece representado?

3.- Modifica el programa:

```
background(0, 0, 0);
stroke(255, 255, 255);
line(0, 0, 60, 40);
stroke(255, 255, 0);
line(30, 50, 100, 100);
```

Pulsa “Ejecutar”.

4.- Utilizando las coordenadas, y el comando line, dibuja un cuadrado de 50 píxeles de lado, de diferentes colores (rojo, azul, verde y blanco). Primero tienes que determinar las coordenadas de los vértices del cuadrado. Una vez que sepas las coordenadas de los vértices, comienza a escribir el código utilizando los comandos “line” y “stroke”. ¡Ánimo!

2.- COMANDOS:

line(x1, y1, x2, y2)	Dibuja una línea desde la coordenada x1,y1 hasta la x2,y2.
background(gray)	Establece el color de fondo desde 0 – negro hasta 255 – blanco.
background(0, 0, 0)	Permite cambiar el fondo en base los colores R, G, B.
size(width, height)	Define el tamaño de la pantalla en número de píxeles.
stroke(0, 0, 0)	Establece el color de la línea en base a los colores R, G, B.
fill(0, 0, 0)	Rellena la forma del color definido por RGB.

3.- REPRESENTAR FORMAS.

El siguiente paso en nuestro itinerario para aprender a utilizar Processing es realizar formas. Para ello, vamos a realizar una serie de prácticas:

5.- Escribe el siguiente código en Processing:

```
size(150, 100);
quad(61, 60, 94, 60, 99, 83, 81, 90);
rect(10, 10, 60, 60);
triangle(12, 50, 120, 15, 125,60);
```

¿Qué aparece en pantalla?

6.- En el programa anterior, introduce el código que aparece en negrita:

```
size(150, 100);
fill(255, 0, 0);
stroke(0, 0, 255);
quad(61, 60, 94, 60, 99, 83, 81, 90);
rect(10, 10, 60, 60);
triangle(12, 50, 120, 15, 125,60);
```

¿Qué cambio se ha producido?

7.- Realiza los dos programas siguientes y escribe qué sucede.

```
rect(15, 15, 50, 50);
ellipse(60, 60, 55, 55);
```

```
ellipse(60, 60, 55, 55);
rect(15, 15, 50, 50);
```

4.- COMANDOS:

triangle(x1, y1, x2, y2, x3, y3);	Dibujará un polígono de tres puntas (triángulo)
quad(x1, y1, x2, y2, x3, y3, x4, y4);	Dibuja un polígono de cuatro puntas (cuadrado)
rect(x, y, width, height);	Dibuja un rectángulo. (x,y) indican la posición.
ellipse(x, y, width, height);	Dibuja una elipse. (x,y) es el centro.

5.- VARIABLES.-

Una variable es un contenedor para guardar datos. Las variables permiten que cada dato sea reutilizado muchas veces en un programa. Cada variable tiene dos partes: un nombre y un valor. Por ejemplo, si definimos la variable "edad" y le otorgamos un valor "14", cuando se ejecute el programa, la palabra "edad" de cualquier parte del programa se cambia por el valor 14.

Una variable debe ser, siempre, declarada antes de ser usada. Además, debemos declarar el tipo de datos que aceptará esa variable.

En un determinado momento del programa, podemos cambiar el contenido de una variable.

Tipos de Datos:

`int`: Número entero, p.ej., 2, 99 o 532.

`float`: Número decimal, p.ej., 2.76, 8.211 o 900.3.

`boolean`: Puede ser verdadero o falso.

`char`: Un caracter, p.ej. 'r', '2' o '%'.

`String`: Una secuencia de carateres, p.ej. "hola", "¡Me encanta programar!" o "&%!@¤".

6.- PRÁCTICAS:

8.- Vamos a realizar las dos líneas, que hicimos al principio usando variables. Copia el código:

```
int valor1 = 0;
int valor2 = 100;
line(valor1, valor1, valor2, valor2);
line(valor1, valor2, valor2, valor1);
```

Podemos modificar los valores de las variables, y veremos que se modifica la línea. En lugar de cambiar 8 valores, tan sólo tenemos que cambiar dos valores para que haya cambios en el programa.

7.- COMANDOS:

int variableName = value Crea una variable del tipo entero.

8.- SETUP Y DRAW.

Los tipos de programas que hemos hecho hasta ahora son los llamados programas estáticos. Esto significa que nunca cambian. Se ejecutan una única vez y cuando llegan a la última línea de código, se paran. Si queremos que un programa sea interactivo, tenemos que habilitar la entrada de datos continuamente mientras el programa se ejecuta. Esto sólo es posible si la ejecución es continua.

Con Processing, puedes crear programas que se ejecuten continuamente utilizando la función **draw()**. Esta función repetirá el bloque de código una y otra vez hasta que el programa se pare.

Sin embargo, no todo el código escrito necesita ser repetido continuamente. Para el código que sólo necesita ser ejecutado una única vez, debes usar la función llamada **setup()**.

Ejemplo:

```
void setup() {
        size(300, 300);
}
void draw() {
        line(0, 0, width, height);
}
```

9.- COMANDOS

setup() El código dentro de las llaves se ejecuta una única vez cuando inicia el programa.

draw() El código entre llaves se ejecuta una y otra vez, línea por linea, de arriba a abajo.

mouseX La coordenada X del puntero del ratón.

mouseY La coordenada Y del puntero del ratón.

10.- PRACTICA.

9.- Escribe el siguiente programa en Processing e indica qué ocurre:

```
void setup() {
        size(300, 300);
}
void draw() {
        line(0, 0, mouseX, mouseY);
}
```

11.- RATÓN Y TECLADO.

En Processing, la función **mousePressed()** se llama cada vez que el ratón es presionado y el método **mouseReleased()** es llamado cada vez que el ratón es liberado o dejado de presionar.

La utilización del teclado se realiza empleando el evento **void keyPressed().**

Para aclarar estas funciones y referencia realiza las siguientes prácticas.

12.- PRÁCTICAS.

10.- Realiza la siguiente práctica. Copia en Processing el código y explica qué ocurre en la pantalla al mover el ratón y pulsarlo.

```
void draw() {
        background(190);
        rect(mouseX-5, mouseY-5, 10, 10);
}
void mousePressed() {
        fill(0);
}
void mouseReleassed() {
        fill(255);
}
```

11.- En este caso, copia el código en Processing y explica su funcionamiento:

```
void draw() {
        if(keyPressed) {
                fill(102, 0, 0);
}       else {
                fill(204, 102, 0);
}
        rect(30, 20, 55, 55);
}
```

12.- Con el siguiente programa podrás controlar con el teclado el movimiento de un cuadrado, lo que nos puede servir para diversas aplicaciones. Observa, atentamente, las funciones y eventos que utiliza.

```
int x= 50;
int y = 50;

void draw() {
        background(190);
        rect(x, y, 10, 10);
}

void keyPressed() {
        if(key=='w' || key=='W') {
                y--;
}       else if{key=='s' || key=='S') {
                y++;
}       else if(key=='a' || key=='A') {
                x--;
}       else if(key=='d' || key=='D') {
                x++;
}
}
```

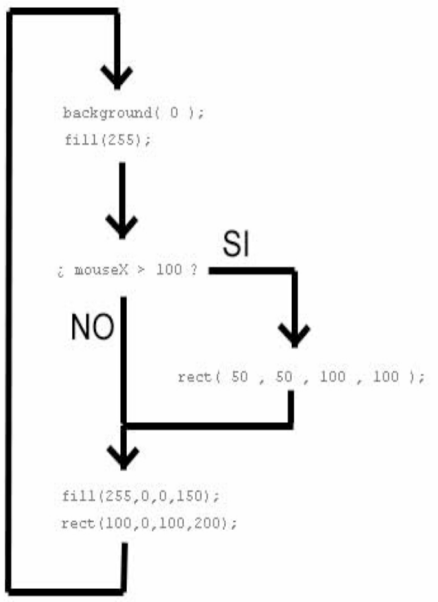
13.- ESTRUCTURAS DE CONTROL.

La mayoría de los ejercicios realizados anteriormente (excepto los de la página 6) siguen una línea única de ejecución, es decir, no tiene posibilidad de bifurcar su línea de procesamiento, ni de repetir ninguna porción de su código. Para controlar el flujo de ejecución, existen dos tipos de estructuras: las condicionales y las iterativas (repetitivas).

14.- PRACTICA.

13.- Copia el siguiente código, ejecútalo e indica que ocurre cuando movemos el ratón por la pantalla.

```
Void setup() {
        size(200, 200);
}
void draw() {
        background(0);
        fill(255);
        if (mouseX>100) {
                rect(50,50,100,100);
}
        fill(255,0,0,150);
        rect/100,0,100,200);
}
```



13.1.- Estructuras *if-then-else* **y** *if-then-else if-else***.**

La estructura *if-then*, permite ejecutar u omitir una porción de código en función de una condición. Su aplicación es:

```
if (condicion) {
se ejecuta si la condición es verdadera
} else {
se ejecuta si la condición es falsa
}
```

La estructura *if-then-else if-else*, permite evaluar varias condiciones para tomar diferentes caminos en función de cada una y por último tomar un camino si no se ha cumplido ninguna de estas:

```
if( condición 1 ){
se ejecuta si la condición 1 es verdadera
}else if( condición 2 ){
se ejecuta si la condición 1 es falsa y la 2 es verdadera
}else if( condición 3 ){
se ejecuta si la condición 1 y 2 son falsas y la 3 es verdadera
...
}else{
se ejecuta si todas las condiciones anteriores son falsas
}
```

15.- PRÁCTICAS.

14.- Copia el siguiente ejemplo en Processing e indica qué ocurre:

```
void setup(){
size(200,200);
}
void draw(){
background( 0 );
fill(255);
if( mouseX > 100 ){
rect( 50 , 50 , 100 , 100 );
}else{
ellipse( 100 , 100 , 100 , 100 );
}
fill(255,0,0,150);
rect(100,0,100,200);
}
```

15.- Comprueba que sucede en este caso, tras copiar el código en Procesing:

```
void setup(){
size(200,200);
}
void draw(){
background( 0 );
fill(255);
if( mouseX > 100 ){
rect( 50 , 50 , 100 , 100 );
}else if( mouseX > 50 ){
ellipse( 100 , 100 , 100 , 100 );
}else{
triangle(50,150,100,50,150,150);
}
fill(255,0,0,70);
rect(100,0,100,200);
rect(50,0,150,200);
}
```

13.2.- Estructuras de control iterativas (repetitivas).

La estructura iterativa más utilizada es la que se conoce como ciclo for-next. Esta estructura permite repetir, una cantidad determinada de veces, las instrucciones que se encuentra en su interior. La estructura es la siguiente:

```
for( inicializacion ; condición ; incremento ){
cuerpo de la estructura
}
```

16.- PRÁCTICAS.

16.- Copia las siguientes instrucciones y observa que la instrucción se repite tres veces.

```
size(200,200);
int a = 10;
for(int i=0 ; i<3 ; i++){
rect( a , 75 , 50 , 50 );
a += 60;
}
```

17.- Realiza los siguientes ejercicios y observa lo que ocurre. Descríbelo.

```
for(int i=1 ; i<10 ; i+=2){
println(i);
}

for(int i=10 ; i>0 ; i--){
println(i);
}

for(int i=1 ; i<=256 ; i*=2){
println(i);
}
```