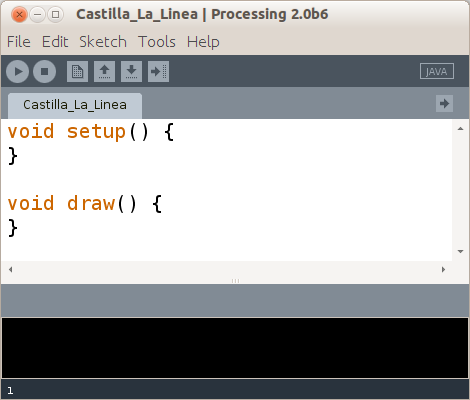
Processing

Los códigos de programación son comandos comprensibles para los humanos. El código se transforma en los programas que los computadores entienden. Los computadores sólo entienden en realidad ceros y unos, por lo que se necesita una manera de traducir nuestro código humano al código de los computadores. Este es el motivo por el que escribimos el código en un editor de texto con propiedades especiales. Este editor es el software de desarrollo llamado Entorno de Desarrollo Integrado, abreviado por sus siglas del inglés IDE (Integrated Development Environment), el cual es capaz de traducir el código al lenguaje máquina. Cuando el IDE transforma nuestro código al lenguaje máquina, es lo que se llama compilar.



Cada ordenador, smartphone, servidor de internet, ascensor, horno – actualmente cualquier dispositivo con inteligencia digital – tiene diferentes capacidades y por lo tanto requiere de un lenguaje máquina algo diferente al resto. Existe un lenguaje de programación llamado Java que, una vez compilado, es capaz de utilizarse fácilmente en diferentes dispositivos. Hay muchas maneras diferentes de programar en Java, pero nosotros nos centraremos en utilizar Processing.

Processing es código abierto y software libre, esto significa que se puede descargar de Internet, instalarlo sin coste alguno, y modificar su código. Processing funciona en cualquier tipo de ordenador de sobremesa o portátil en los sistemas operativos: Windows, Mac OSX y Linux.

El IDE de Processing es muy sencillo, hay una barra de herramientas que puedes utilizar para:  
https://farm3.staticflickr.com/2806/12161744004_3f3bd04c60_o.png Poner en marcha un programa  
https://farm8.staticflickr.com/7393/12161989676_2aa8218e6d_o.png Detener un programa  
https://farm6.staticflickr.com/5492/12161744854_d3e4d22937_o.png Crear un nuevo programa  
https://farm8.staticflickr.com/7418/12161745074_d9bfa45da8_o.png Abrir un programa  
https://farm3.staticflickr.com/2859/12161745454_a8406e89dc_o.png Almacenar el programa en el disco duro del ordenador  
https://farm4.staticflickr.com/3675/12161570073_a21fa743a5_o.png Exportar el programa

Processing crea programas que se ejecutan directamente dentro de tu ordenador. También puedes exportarlos y enviarlos a los ordenadores de otras personas. Los programas, una vez exportados en código máquina, se llaman aplicaciones. Puedes exportar el mismo programa para que se ejecute en un ordenador, en un teléfono, o en un sitio web.

Para hacer tus propios programas, lo único que necesitas saber es cómo utilizar un teclado y cómo funciona la pantalla del ordenador. Esto te será útil cuando escribas tus propios programas interactivos.

Pantallas y pixels

Las pantallas se utilizan para representar imágenes. Esas imágenes pueden ser dinámicas, como en las películas, o estáticas, como en las fotografías. Se puede incluir texto, como en los documentos, o solo gráficos. La mayoría de las veces, las imágenes en la pantalla son una mezcla de muchas cosas; incluyen animaciones, textos, estadísticas, etc.

Pixels – Resolución

Las pantallas están hechas de puntos que llamamos pixels. Por lo general, cuantos más pixels tenga una pantalla, mejor será. Expresamos la calidad de la pantalla en función de su cantidad de pixels, esa cantidad es lo que llamamos resolución. Normalmente, hablamos de una pantalla en función de su anchura y altura. Por ejemplo, muchos proyectores tienen una resolución de 1024×768 pixels. Esto significa que tiene 1024 pixels de ancho y 768 pixels de alto.

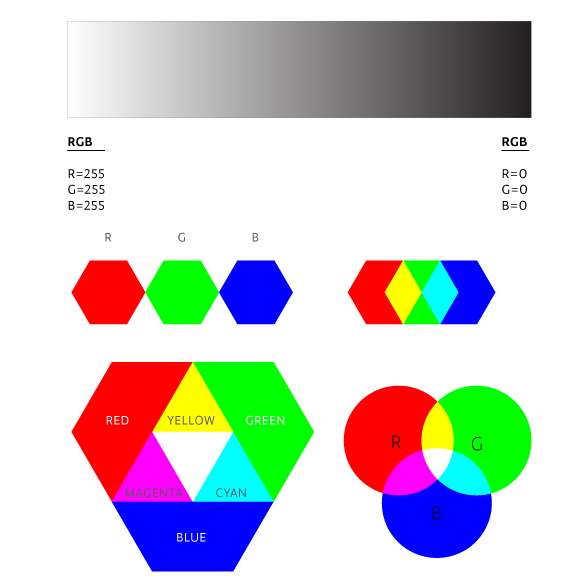
Colores

Cada píxel se puede iluminar de muchos colores diferentes. Una vez más, de cuantos más colores se pueda pintar el píxel, mejor será la pantalla. Los colores en la pantalla del ordenador se expresan como una combinación de tres componentes de color diferentes: rojo (R), verde (G) y azul (B). Cualquier color que representes en una pantalla, se puede hacer mezclando los componentes RGB en diferentes cantidades.

La cantidad de color para cada componente se expresa con un número entre 0 y 255. 0 representa el valor mínimo de un color, por ejemplo, si tiene 0 del componente rojo, significa que ese color no contiene rojo, sin embargo, si tiene 255 del componente rojo, significa que está saturado de color rojo.

Si las coordenadas son 255,255,255 – los píxels producirán blanco. Si los valores R,G,B son 0,0,0 – los píxels mostrarán negro. Si gradualmente cambias los valores R,G,B en la misma proporción desde 255,255,255 a 0,0,0, verás que el color cambia de blanco a negro.

La siguiente imagen muestra cómo el color de un píxel se funde de blanco a negro y cómo cambian los valores de R, G, y B:



Coordenadas

Puedes escribir programas que cambien los pixels de la pantalla. Puedes incluso cambiar un solo pixel. La acción de acceso a un solo pixel en la pantalla es lo que llamamos direccionar un pixel. La ubicación de un pixel se determina mediante coordenadas.

Cada pixel tiene 2 coordenadas:

* La coordenada horizontal es la que llamamos coordenada X (anchura, o en inglés *width*).
* La coordenada vertical, es la que llamamos coordenada Y (altura, o en inglés *height*).

Si recuerdas lo que aprendiste en clase de matemáticas y física, puedes dibujar gráficos, puntos, líneas y curvas usando sus coordenadas. Esta misma idea se aplica a la pantalla del ordenador, la única diferencia, es que el llamado *origen de coordenadas* de la pantalla del ordenador es la esquina superior izquierda.



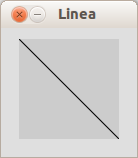
# ﻿ Línea

La mejor manera de aprender acerca de la programación es escribiendo tus propios programas. Vamos a empezar por hacer el programa lo más corto posible usando Processing. Será un programa para dibujar una línea. Esto te va a ayudar a entender cómo funciona.

Abre el IDE y haz click en el icono https://farm6.staticflickr.com/5492/12161744854_d3e4d22937_o.png para crear un nuevo programa, esto te mostrará un nuevo programa vacío. Escribe el siguiente código:

|  |  |
| --- | --- |
| 1 | line(0, 0, 100, 100); |

Ejecuta el programa apretando el botón *ejecutar*: https://farm3.staticflickr.com/2806/12161744004_3f3bd04c60_o.png, que es el primero en la barra de herramientas. Esto es lo que deberías ver ahora por pantalla:



Sólo utilizamos un único comando aquí:

* line(x1, y1, x2, y2): Dibuja una línea desde la coordenada x1,y1 hasta la x2,y2. En este ejemplo, desde 0,0 – esquina superior izquierda, hasta 100,100 – esquina inferior derecha. Esto es debido a que la ventana es 100 por 100 píxels.

Lo que también necesitas saber es:

* Debes terminar cada línea de código con un puto y coma, ‘;‘. Si te olvidas, el código no podrá ser compilado ni ejecutado.
* Por defecto, las ventanas de Processing son de 100×100 píxels con un fondo gris. El color por defecto para dibujar es el negro. Pero por supuesto, todo esto se puede cambiar.

Vamos a escribir un programa que dibuje dos líneas de diferentes colores sobre un fondo blanco:

|  |  |
| --- | --- |
| 1  2  3  4 | background(255);  line(0, 0, 100, 100);  stroke(0, 0, 255);  line(0, 100, 100, 0); |

Los nuevos comandos utilizados aquí son:

* background(gray): Establece el color de fondo desde 0 – negro, hasta 255 – blanco. También puedes utilizar background(red, green, blue)para establecer el color que tú quieras.
* stroke(red, green, blue): Establece el color de la línea. Cada valor de color puede ser desde 0 hasta 255. En este caso, la línea es de color azul puesto que red=0, green=0 y blue=255.

Cuando ejecutes el programa, lo que deberías ver por pantalla es:



Tus programas se ejecutarán siempre de arriba a abajo línea por línea. Este programa empieza estableciendo el color de fondo en blanco. Luego dibuja la línea negra igual que en el ejemplo anterior, desde la coordenada 0,0 hasta la 100,100. En tercer lugar, establece el color de línea a azul, justo antes de dibujar esta segunda línea desde la coordenada 0,100 – esquina inferior izquierda, hasta la 100,0 – esquina superior derecha.

https://farm3.staticflickr.com/2859/12161745454_a8406e89dc_o.png No olvides guardar tu programa en el disco duro o en una memoria USB si quieres llevártelo a casa.

Variables

Las variables son algo que usas todo el tiempo en programación. Son como un contenedor para diferentes tipos de datos. Para cada variable, necesitas especificar qué tipo de datos contendrán, cuál es el nombre de la variable y qué valor se le asigna.

Piensa en ellas como en botes. Digamos que tienes dos botes, uno para galletas y otro para palabras, estos son los tipos de datos. Ahora hay que darles un nombre a cada bote; cookieJar (*bote de galletas*) y jarOfWord (*bote de palabras*). Ahora tú decides que poner en cada bote. En el cookieJar pones una galleta de doble cocholate y en el jarOfWord decides poner “Arduino”. Ahora cada bote tiene un valor.



Puedes cambiar el contenido de los botes, su valor, en cualquier momento, pero siempre y cuando sea del mismo tipo. Por ejemplo, puedes cambiar la galleta de doble chocolate por una oreo y “Arduino” por “spaceinvader”.

Para hacerlo más claro, vamos a escribir un programa utilizando variables. Escribiremos un programa que dibuje, de nuevo, dos líneas, pero esta vez utilizaremos estas variables:

|  |  |
| --- | --- |
| 1  2  3  4 | int value1 = 0;  int value2 = 100;  line(value1, value1, value2, value2);  line(value1, value2, value2, value1); |

Nuevos comandos:

* int variableName = value: Crea una variable del tipo integer, un número entero. Puedes llamarla con cualquier nombre de tu elección, pero para hacerlo más sencillo para tu futuro yo (quien probablemente leerá el código en el futuro y querrá entender lo que hizo en el pasado) asegúrate que el nombre va acorde con el contexto en el que se usa la variable. value es el valor que decides darle al contenido de tu variable.

Lo que hace el programa es dibujar una linea desde la coordenada (0,0) hast la (100,100) y luego dibuja una segunda linea desde la coordenada (0,100) hasta la (100,0). Puesto que value1 contiene el valor 0, en cualuier lugar del código donde escribamos value1 significará 0. Esto significa que si quieres cambiar la cruz que hemos dibujado, basta con cambiar el valor de la variable en lugar de cambiar el valor en los ocho lugares diferentes dentro de line(). Pruébalo tu mismo y verás.

Tipos de datos

Los tipos de datos más comunes que utilizarás son los siguientes:

* 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 “&%!@¤”.

Processing incluye algunas variables de sistema para hacerlas más accesibles dentro de tus programas. Un ejemplo es width y height. Estas variables devuelven la anchura (*width*) y altura (*height*) de la ventana de tu programa. Éste es un pequeño ejemplo:

|  |  |
| --- | --- |
| 1  2 | size(400, 200);  ellipse(width/2, height/2, width, height); |

Comandos nuevos:

* size(width, height): Establece el tamaño de la ventana en píxels.
* ellipse(x, y, x diameter, y diameter): dibuja un elipse centrado en las coordenadas x e y. El tamaño se establece con x diameter e y diameter. Cuando estos dos parámetros son iguales, el resultado es un círculo.
* width: la anchura de la ventana de programa.
* height: la altura de la ventana de programa.



Este programa empieza estableciendo el tamaño de ventana. Luego dibuja la elipse en el medio de la ventana. No importa el tamaño de la venta, el elipse la llenará siempre por completo. Intenta cambiar el tamaño de la ventana.

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().

Vamos a escribir de nuevo un programa que dibuje una línea, pero esta vez utilizando las nuevas funciones:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void setup() {  size(300, 300);  }    void draw() {  line(0 ,0, width, height);  } |

Nuevos comandos:

* setup(): El código dentro de las llaves se ejecuta una única vez cuando el programa empieza. Sólo necesitamos establecer el tamaño de la ventana una vez, al principio del programa.
* draw(): El código entre llaves se ejecuta una y otra vez. Se ejecuta línea por línea, de arriba a abajo hasta la última línea, donde vuelve a empezar desde el principio.

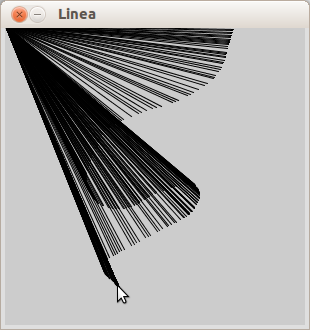
Este programa establece el tamaño de ventana a 300 x 300 pixels y luego dibuja una línea sobre toda la ventana una y otra vez. En este momento, todavía no se puede apreciar que esto esta pasando una y otra vez, así que vamos a escribir otro programa para hacerlo más visible.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void setup() {  size(300,300);  }    void draw() {  line(0, 0, mouseX, mouseY);  } |

Nuevos comandos:

* mouseX: La coordenada X del puntero del ratón.
* mouseY: La coordenada Y del puntero del ratón.

Este programa te permite interactuar con la pantalla mientras mueves el ratón, como muestra la siguiente imagen. Esto es porque cada vez que el código dentro de draw() se ejecuta, una nueva línea se dibuja. El segundo punto de la línea cambia dependiendo de la posición del puntero del ratón.



Como puedes ver, se deja un rastro de líneas por donde vas moviendo el ratón. Esto es porque cada línea dibujada no es eliminada ni se dibuja nada más por encima. ¿Qué crees que ocurrirá cuando ejecutes el siguiente programa?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | void setup() {  size(300,300);  }    void draw() {  background(255);  line(0, 0, mouseX, mouseY);  } |

Hemos añadido una línea de código al principio de draw(), esto sirve para establecer el color de fondo a blanco. Como puedes ver, este programa solo muestra una línea todo el tiempo. Esto se debe a que, como hemos explicado antes, el código dentro de draw() se ejecuta una y otra vez, línea por línea de arriba a abajo. Por tanto, la primera cosa que pasa cada vez que draw() se ejecuta es que se crea el fondo blanco, sobre escribiendo todo lo anterior. Después una nueva línea es dibujada, y luego el fondo se vuelve a poner en blanco… Eso se repetirá una y otra vez por siempre.

﻿