INTRODUCCION A PROCESSING

Miguel Angel Ruiz Gálvez
Visita: miguelo.me
Material en: http://goo.gl/j05JbT

SOMEFI

MIGUELRUIZ
MECHATRONIC ENGINEER

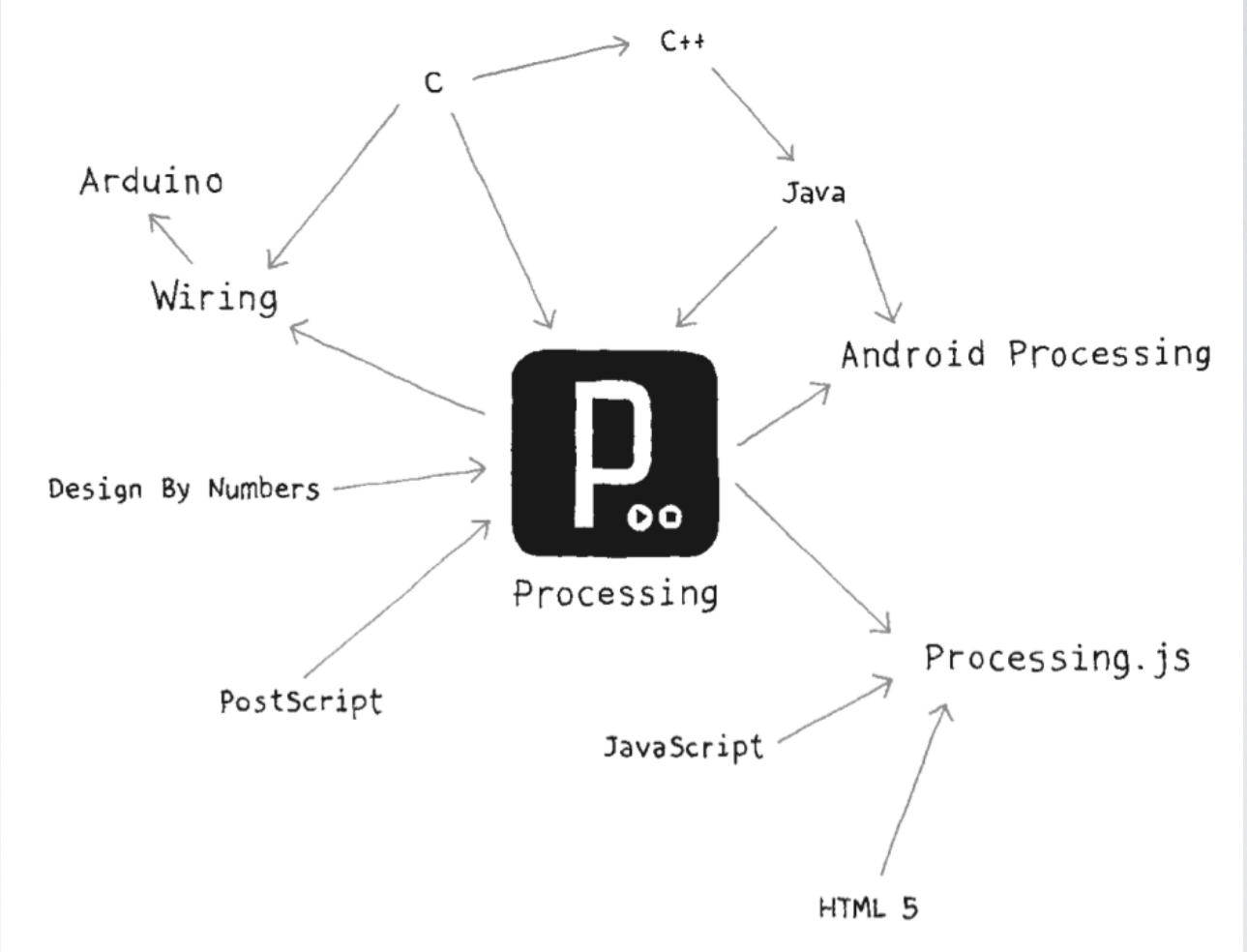
CC BY NC SA

¿QUE ES PROCESSING?

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización, y que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Fue iniciado por Ben Fry y Casey Reas en el Aesthetics and Computation Group del Media Lab en el MIT.

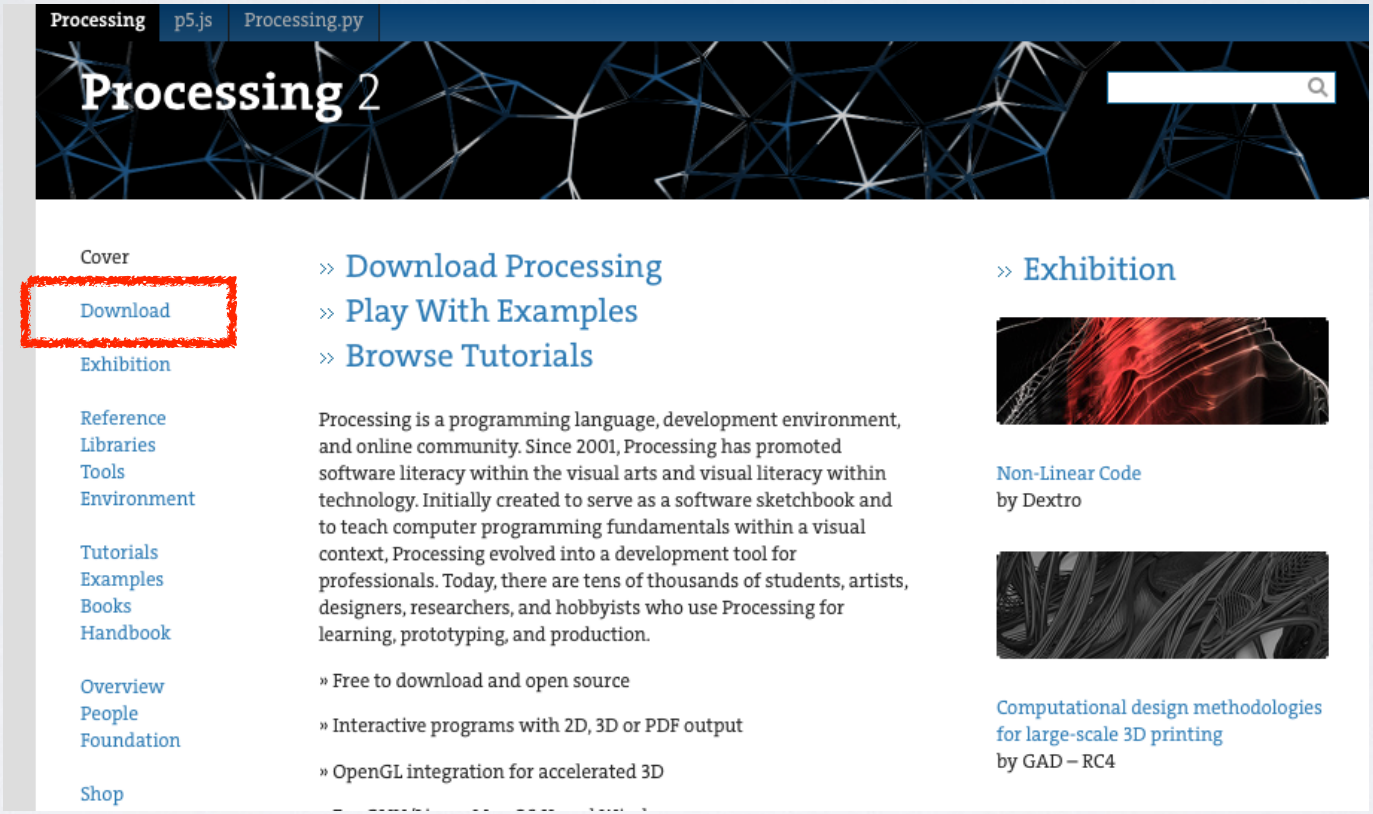
MIGUELRUIZ
MECHATRONIC ENGINEER

- Código abierto
- Licencia libre
- Multiplataforma
- Lenguaje e IDE



DESCARGA

- Pagina oficial: https://processing.org



Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below.

2.2.1 (19 May 2014)

Windows 64-bit | Linux 64-bit | Mac OS X

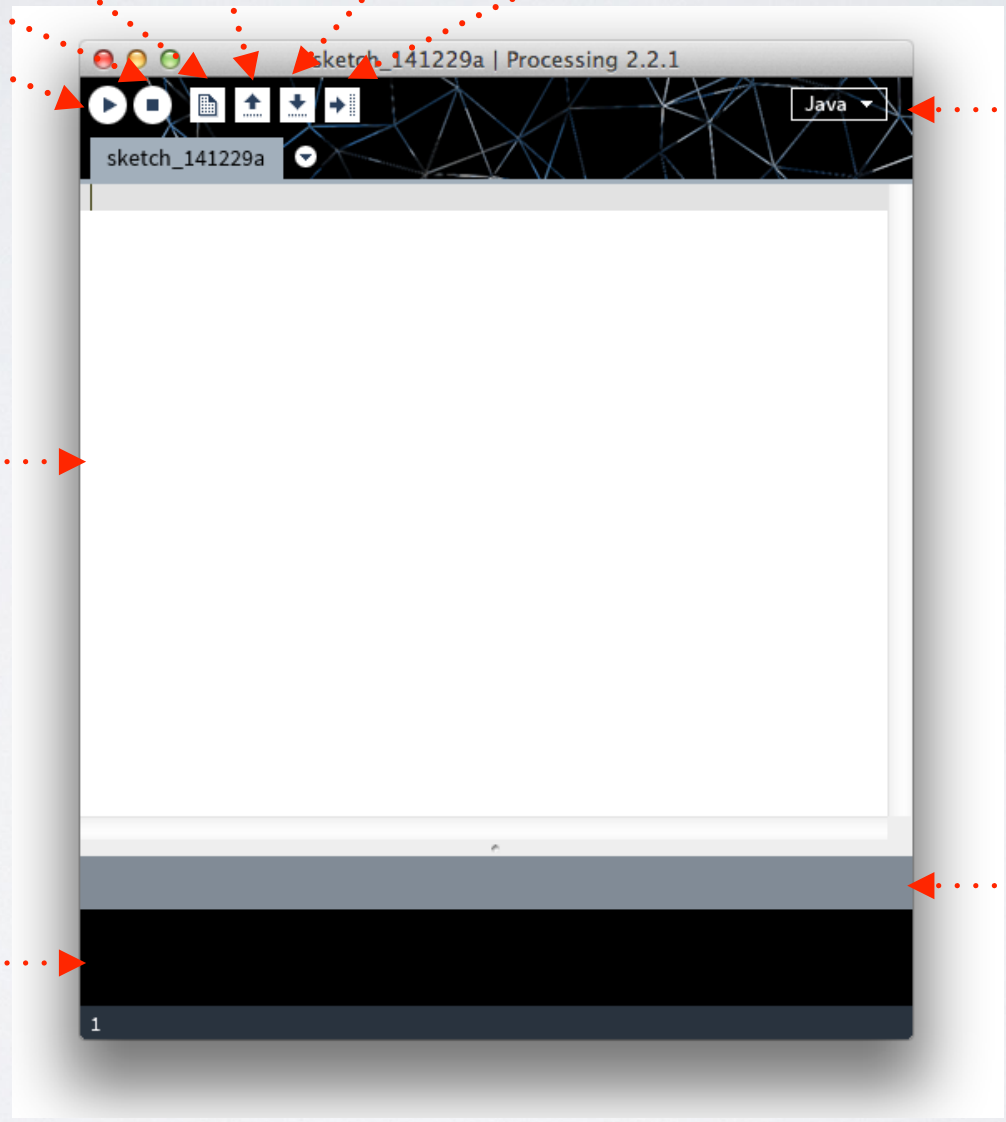
Windows 32-bit | Linux 32-bit

» Github
» Report Bugs
» Wiki
» Supported Platforms

The list of revisions covers the differences between releases in detail. Please read the changes if you're new to the 2.0 series.

CONOCIENDO EL IDE

Play Stop New Open Save Export



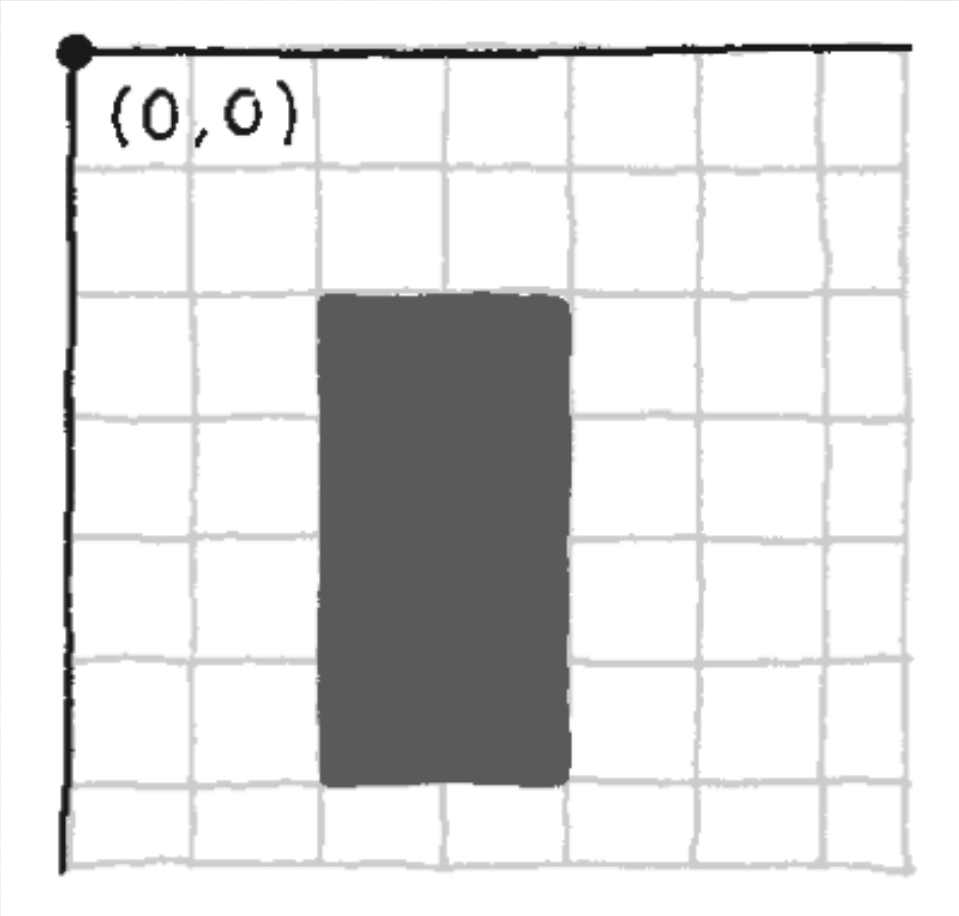
Modo

Editor de texto

Mensajes

Consola

PROPIEDADES DE LA VENTANA



MIGUELRUIZ
MECHATRONIC ENGINEER

`size(ancho, alto);`

- Define la dimensión de la ventana de visualización en **píxeles**.
- La función `size()` debe ser la primera línea de código, o el primer código dentro de `steup()`.
- Si no se utiliza la, a la ventana se le dará un tamaño predeterminado de 100x100 píxeles.
- La función `size()` sólo se puede utilizar una vez, y no se puede utilizar para cambiar el tamaño.
- No utilice variables en los parámetros de `size()`. Pero es posible utilizar valores numéricos y luego utilizar las variables de `ancho` y `alto` cuando se necesitan las dimensiones de la ventana.
- El `ancho` y `alto` máximo está limitada por el sistema operativo, y por lo general el ancho y alto de la pantalla real. Se recomienda utilizar las palabras reservadas `width` y `height`.

background(color);

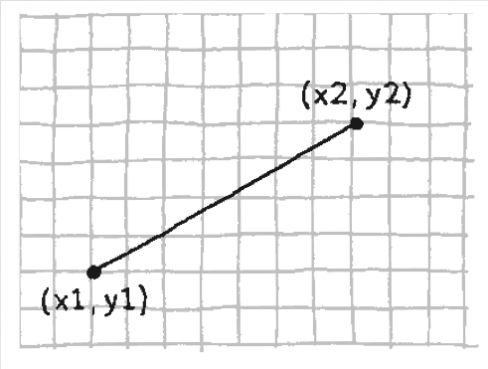
- La función de `background(color);` configura el fondo de la ventana. El fondo predeterminado es gris claro.
- Una imagen también se puede utilizar como fondo, aunque el ancho y la altura de la imagen deben coincidir con la de la ventana de dibujo.
- Es común llamar a `background(color);` al principio de `draw()` para borrar el contenido de la ventana.

VARIABLES Y TIPOS DE DATO

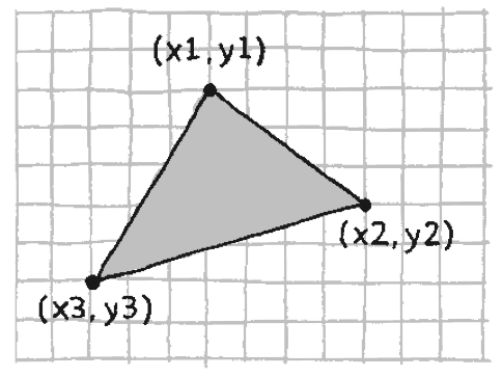
Tipo	Valores	Uso
boolean	True/False	`boolean a = false;`
byte	127 to -128	`byte b = -12;`
char	Cualquier caracter Unicode	`char c = 'A';`
color	Notación RGB o web	`color c1 = color(204, 153, 0);` `color c2 = #FFCC00;`
double	1.7E+308 a 1.7E-308 y tiene una precisión de 15 dígitos.	`double d = 3.14156493.`
float	3.4E+38 a -3.4E+38. y tiene una precisión de 7 dígitos.	`float e = -2.984;`
int	2,147,483,647 a -2,147,483,648	`int f = 256;`
long	9,223,372,036,854,775,808 a -9,223,372,036,854,775,807	`long g = -25632412313;`
Constantes	**HALF_PI, PI, QUARTER_PI, TAU, TWO_PI**	

MIGUELRUIZ
MECHATRONIC ENGINEER

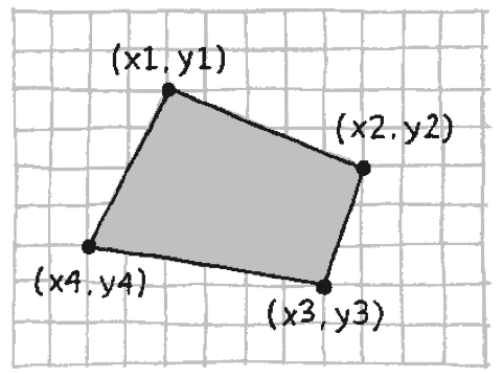
FORMAS



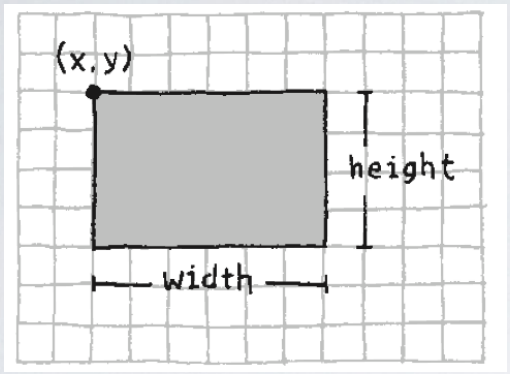

```
point(x1, y1);

line(x1, y1, x2, y2);
```



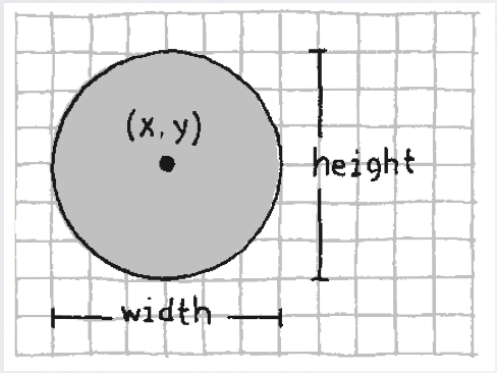

```
triangle(x1, y1, x2, y2, x3, y3);
```



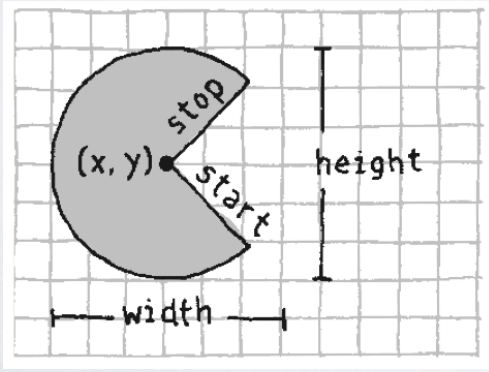

```
quad(x1, y1, x2, y2, x3, y3 , x4, y4);
```



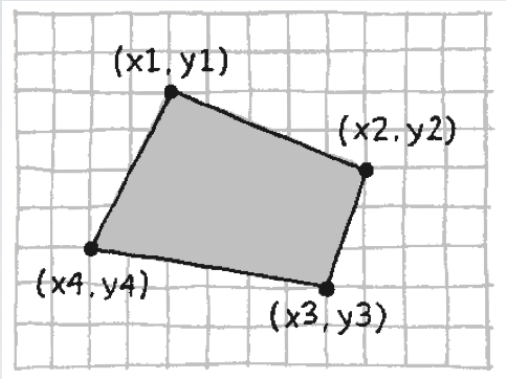

```
rect(x, y, width, height);
rect(x, y, width, height, r);
rect(x, y, width, height, tl, tr, br, bl);
rectMode(mode);
mode = CENTER, RADIUS,CORNER y
 CORNERS
```




```
ellipse(x, y, width, height);
ellipseMode(mode);
mode = CENTER, RADIUS,CORNER y
 CORNERS
```



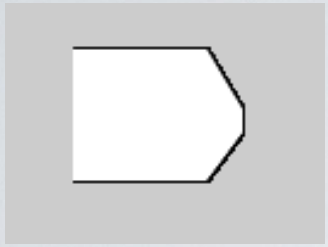

```
arc(x, y, width, height, start, stop);
arc(x, y, width, height, start, stop,mode);
mode = PIE, OPEN, and CHORD
```



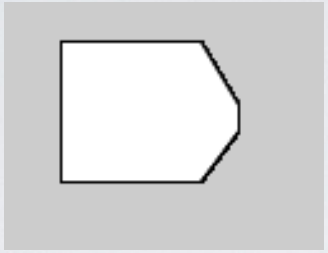

```
beginShape(mode1);
vertex(x1, y1);
. . . . . .
. . . . . .
. . . . . .
vertex(xn, yn);
endShape(mode2);
```

mode1 = POINTS, LINES, TRIANGLES, TRIANGLE_FAN, TRIANGLE_STRIP, QUADS y QUAD_STRIP.
mode2 = CLOSE.

```
//Ejemplo
beginShape(mode1);
vertex(30, 20);
vertex(85, 20);
vertex(100, 20);
vertex(100, 75);
vertex(85, 75);
vertex(30, 75);
endShape(mode2);
```

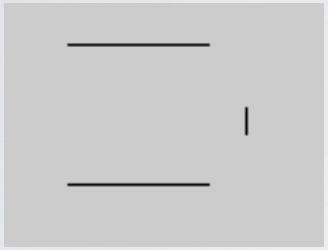


mode1 = nada
mode2 = nada

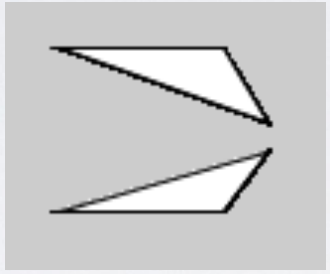


mode1 = nada
mode2 = CLOSE

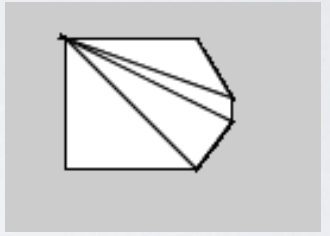
mode1 = POINTS
mode2 = nada



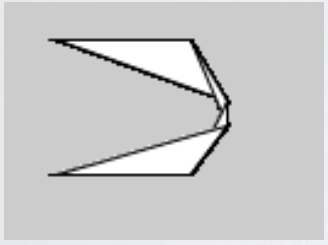
mode1 = LINES
mode2 = nada



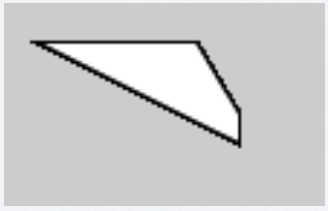
mode1 = TRIANGLES
mode2 = nada



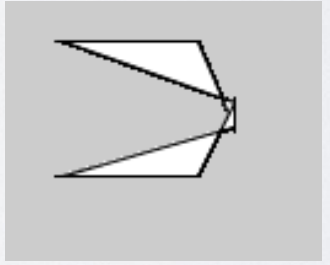
mode1 = TRIANGLE_FAN
mode2 = nada



mode1 = TRIANGLE_STRIP
mode2 = nada



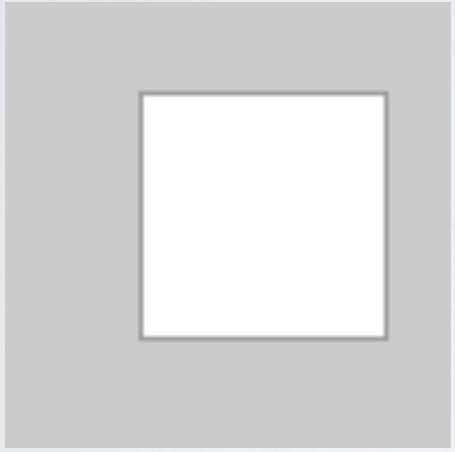
mode1 = QUADS
mode2 = nada



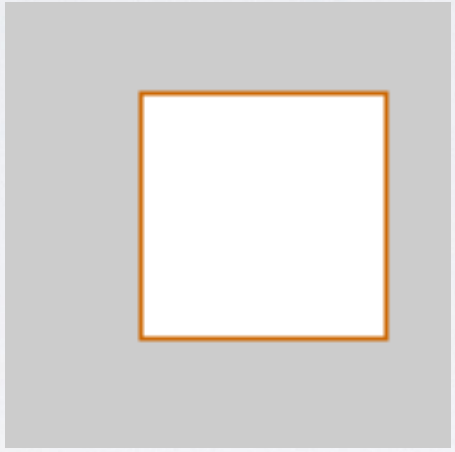
mode1 = QUAD_STRIP
mode2 = nada

CONTORNO

stroke(color);



```
stroke(153);
rect(30,20,55,55);
```

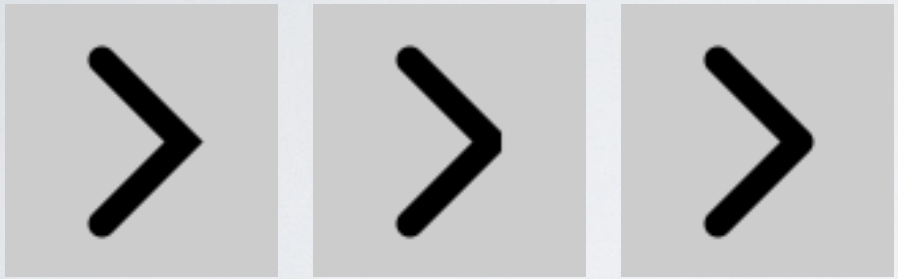


```
stroke(204, 102, 0);
rect(30,20,55,55);
```

noStroke();

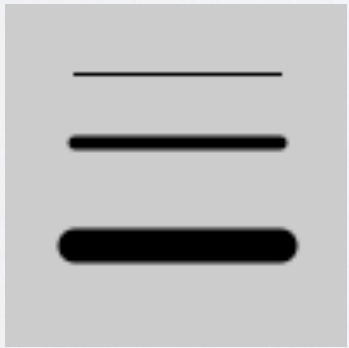

```
noStroke();
rect(30,20,55,55);
```

```
strokeJoin(mode);
```
mode= MITER, BEVEL y ROUND



```
strokeCap(mode);
```
mode= ROUND, SQUARE y PROJECT

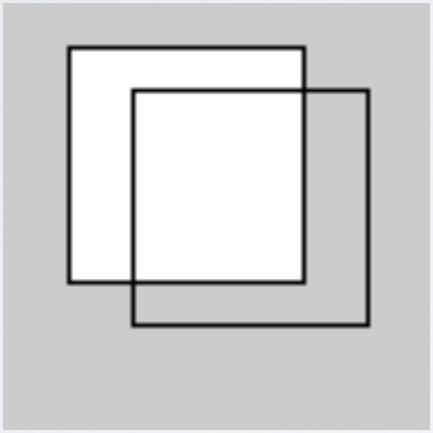
```
strokeWeight(ancho);
```



```
strokeWeight(1);
strokeWeight(4);
strokeWeight(10);
```

RELLENO Y SUAVIDAD

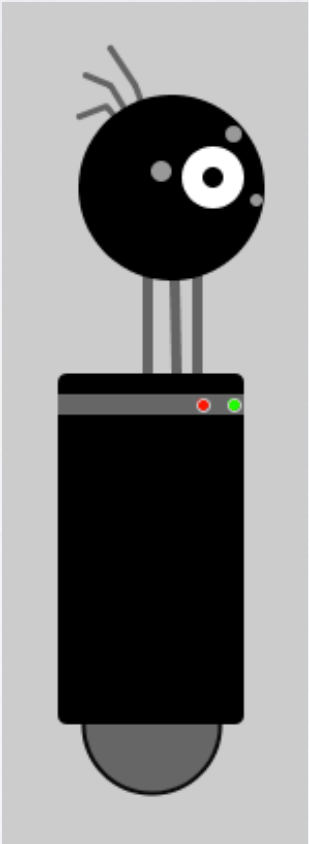
```
fill(color);
noFill();
```



```
smooth();
noSmooth();
```

MIGUELRUIZ
MECHATRONIC ENGINEER

¡DIBUJANDO UN ROBOT!



MIGUELRUIZ
MECHATRONIC ENGINEER

ESTRUCTURA BASICA

```
void setup(){

   //Tu código

}
```

- La función `setup()` se llama una vez al iniciar el programa.
- Se utiliza para definir las propiedades iniciales y para cargar el material.
- Sólo puede haber una función `setup()` para cada programa y no debería ser llamado de nuevo después de su ejecución inicial.

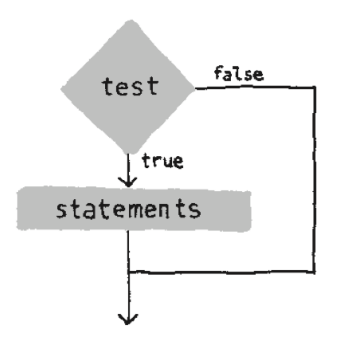
Nota: Las variables declaradas dentro de `setup()` no son accesibles dentro de otras funciones, incluyendo `draw()`.

```
void draw(){

//Tu código

}
```

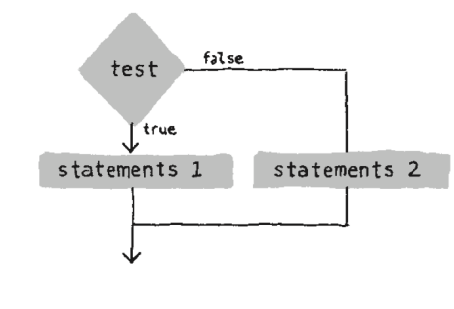
- La función `draw()` ejecuta continuamente las líneas de código contenido hasta que el programa se detiene con `noLoop();` o se termina la ejecución con `exit();`.
- `draw()` es llamado automáticamente y nunca debería llamarse de forma explícita.
- El número de veces `draw()` ejecuta en cada segundo puede ser controlada con la función `frameRate(velocidad);`. Se tiene una velocidad predeterminada de 60 fotogramas por segundo.
- Sólo puede haber una función `draw()` para cada programa y no debería ser llamada por otra función, para forzara que `draw()` se actualice `redraw();` sin embargo no se recomienda llamar dentro de `draw()`.

```
void setup() {
   size(width, height);
   background(color);
   //tu código
}
void draw(){
   //tu código
}
```

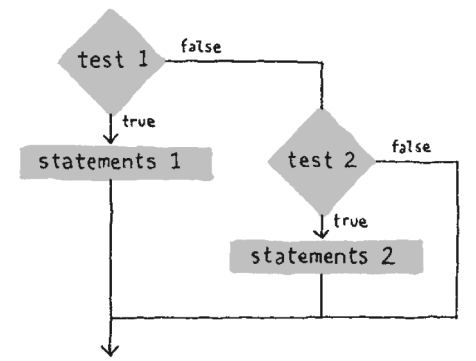
CONDICIONALES




```
if (test) {
  result = expression1;
}

if (test) result = expression1;
```



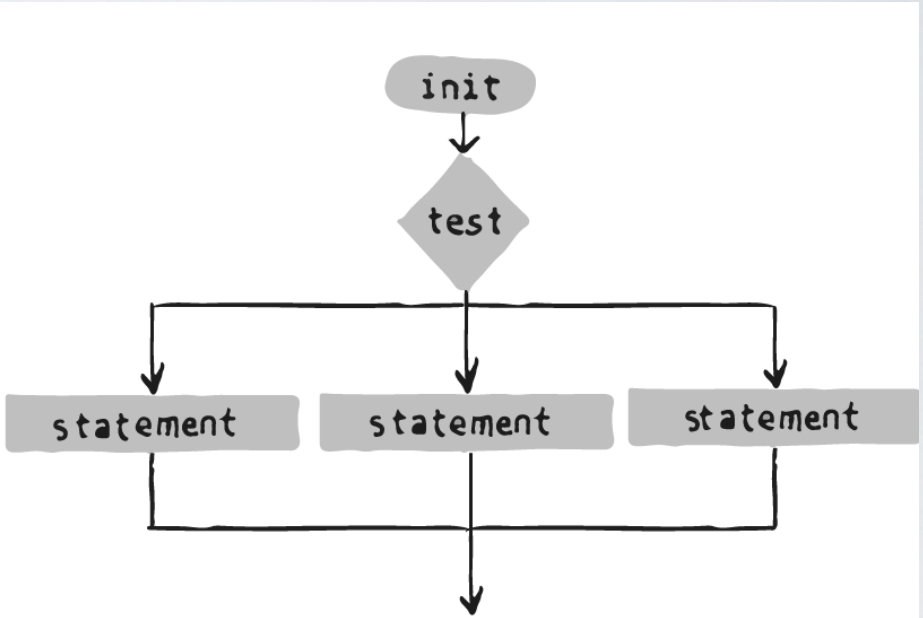

```
if (test) {
  result = expression1;
} else {
  result = expression2;
}

result = test ? expression1 : expression2;
```




```
if (test) {
  result = expression1;
} else { if (test) {
  result = expression2;
    }
}
```

MIGUELRUIZ
MECHATRONIC ENGINEER

```
switch(val)
{
  case label:
      //Tu código
  break;
 case label:
      //Tu código
  break;
  default:
     //Tu código
   break;
}
```




```
val = byte, char o int
```

OPERADORES DE COMPARACIÓN Y LÓGICOS

Nombre	Acción	Parametros	Ejemplo
!=	Diferente	int, float, char, byte, boolean	if (a != b)
<	Menor que	int, float, char, or byte	if (a < b)
<=	Menor que igual	int, float, char, or byte	if (a <= c)
>	Mayor que	int, float, char, or byte	if (a > c)
>=	Mayor que igual	int, float, char, or byte	if (a >= b)
==	Igual	int, float, char, byte, boolean	if (a == b)
!	Not	!expresión	!(a >= b)
&&	AND	expresión1 && expresión 2	(a >= b) && (a < c)
||	OR	expresión1 || expresión 2	(a >= c) || (b < c)

MOUSE

Nombre	Valor
mouseX	Posición del cursor en x
mouseY	Posición del cursor en y
pmouseX	Posición del cursor en x en el fotograba anterior.
pmouseY	Posición del cursor en y en el fotograba anterior.
mousePressed	True/Falce
mouseButton	LEFT, RIGHT y CENTER

Nombre	Acción
mouseClicked()	Se activa cuando se hace click en algún punto de la pantalla
mouseMoved()	Se activa cuando el cursor se mueve
mouseDragged()	Se activa cuando el cursor se mueve y se mantiene presionado el algún botón (izquierdo, derecho o centro)
mousePressed()	Se activa cuando se presiona algún botón (izquierdo, derecho o centro)
mouseReleased()	Se activa cuando se deja de presionar algún botón (izquierdo, derecho o centro)
mouseWheel()	Se activa cuando se gira el scroll. Valores > 0 arriba, < 0 abajo.
(MouseEvent event)	getButton(),getCount(),getX(),getY()

```
void mouseWheel(MouseEvent event) {
  float e = event.getCount();
   //Tu código
}
```

```
void mouseDragged()
{
  //Tu código
}
```

TECLADO

Nombre	Valor
key	Valores ASCII, BACKSPACE, TAB, ENTER, RETURN, ESC, DELETE y CODED
keyCode	UP, DOWN, LEFT, RIGHT, ALT, CONTROL, SHIFT, PAGE_UP, PAGE_DOWN, HOME y END
keyPressed	True/False

Nombre	Acción
keyPressed()	Se activa cuando se presiona cualquier tecla.
keyReleased()	Se activa cuando se libera cualquier tecla.
keyTyped()	Se activa cuando se presiona un tecla, se ignoran teclas de funciones.

```
void keyPressed() {
  //Tu código
}
```

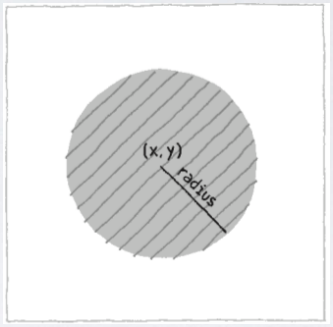
¡LITTLE PAINT!

MIGUELRUIZ
MECHATRONIC ENGINEER

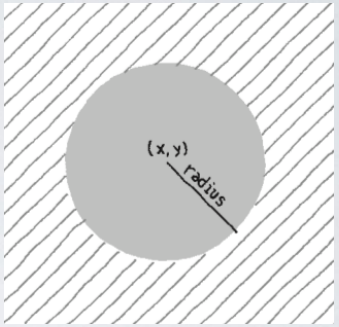
FUNCIONES UTILES

```
float valor = dist(x1, y1, x2, y2);
```

valor < radio

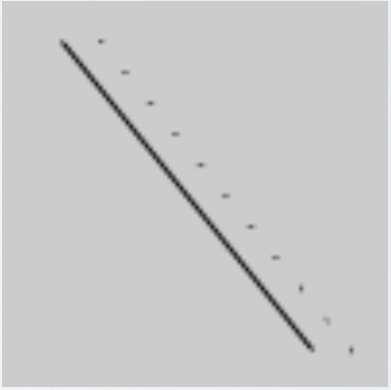


valor < radio




```
float valor = lerp(val, star, stop);
```

```
int x1 = 15;
int y1 = 10;
int x2 = 80;
int y2 = 90;
line(x1, y1, x2, y2);
for (int i = 0; i <= 10; i++) {
  float x = lerp(x1, x2, i/10.0) + 10;
  float y = lerp(y1, y2, i/10.0);
  point(x, y);
}
```



Nombre	Acción
float valor = random(min, max);	Retorna un valor entre min y max son tocar a max.
int/float x = map(int/float, min, max, limMin, limMax);	Realiza una interpolación linea, min y max es el rango de valores de la variable de entrada mientras LimMin y LimMax son los limites de la variable deseada.
int/float x = constrain(int/float, min, max);	Limita los valores de la variable de entrada a un rango.

float b = pow(val,exp);
float a = log(float)
float a = sq(float);
float b = sqrt(float);
float v1 = exp(float);
	pow: Potencia.
	log: Logaritmo natural
	sq: Cuadrado.
	sqrt: raiz cuadrada.
	exp: exponente.

int x = round(float);
int a = ceil(float);
int a = floor(float);
int/float b = abs(int/float);
	round:redondeo convencional.
	ceil: mayor inmediato.
	floor: menor inmediato
	abs: valor abosoluto

IMAGENES

PImage foto;

```
foto = loadImage("data/foto.png");

String url = "https://processing.org/img/processing-web.png";
foto = loadImage(url, "png");

image.resize(ancho, alto);

image(PImage, x, y);
image(PImage, x, y, ancho, alto);

imageMode(mode);
mode = CORNER, CORNERS y CENTER;
```

- Processing puede mostrar .gif, .jpg, .tga, y .png.
- Antes de que se utiliza una imagen, debe ser cargado con la función `loadImage(string);`.
- Se utiliza `image(PImage, x, y);` para posicionar y desplegar la imagen en la ventana.

ARREGLOS

```
datatype[] var = new datatype[cantidad];
var[element] = value;
```

```
int[] var = new int[10];
var[0] = 5;
int tamaño = var.length();
```

```
float[] var ={12.5, 13.6,145};

String[] var ={ “data/foto1.png”,
“data/foto1.png”,“data/foto3.png”};
```

- Una matriz es un conjunto de datos. Es posible tener una matriz de cualquier tipo de datos.
- Cada dato se identifica por un número de índice que representa su posición en el arreglo. El primer elemento de la matriz es [0], el segundo elemento es [1], y así sucesivamente.
- Se utiliza `.length();` para obtener el tamaño de una cadena.

CLASES

¿Qué es un objeto?

Los objetos son un grupo de datos que nos permiten modelar los objetos del mundo real.

¿Qué es una clase?

Una clase es un modelo o prototipo que dicta las características y comportamientos de los objeto.

¿Qué es la herencia?

Nos permite generar nuevas clases obteniendo los atributos de una clase superior.

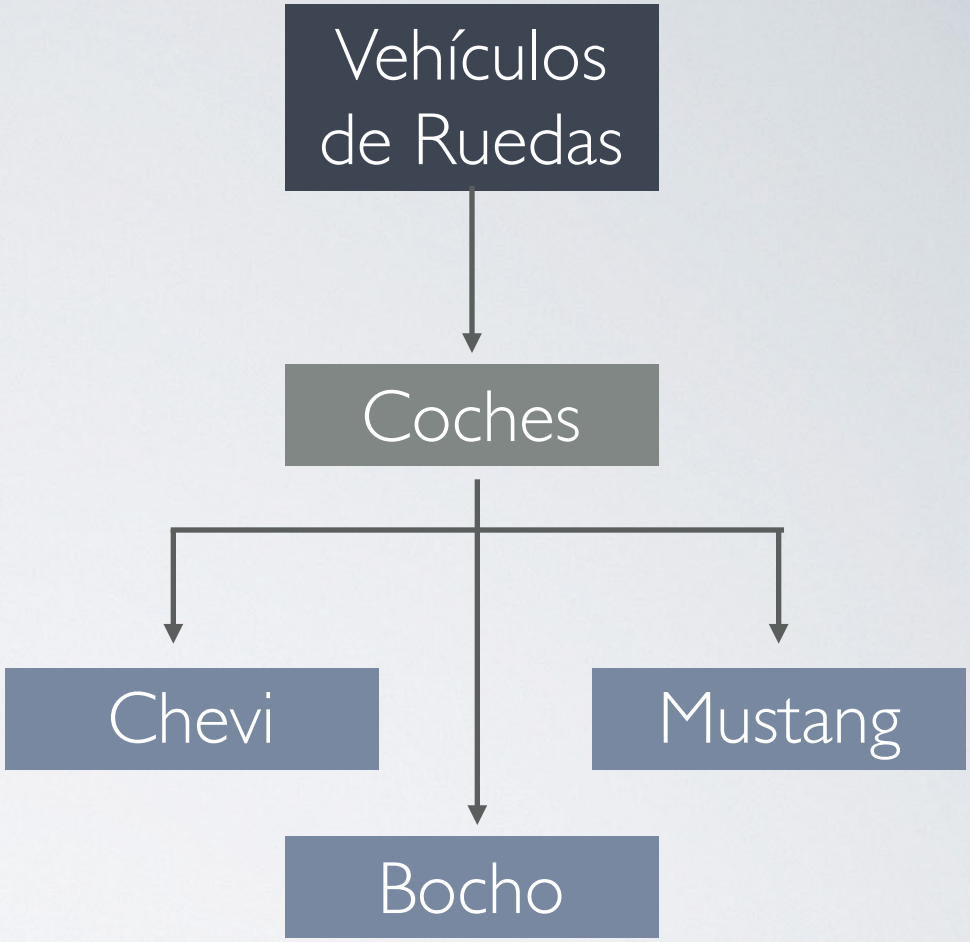
¿Qué es una interacción?

Los objetos pueden ser modificados o modificar parámetros de otros objetos. Se define como la relación entre el objeto y el ecosistema.

¿Qué es un paquete?

Un paquete es un nombre para la organización de las clases y las interfaces de una manera lógica. La colocación de código en paquetes hace grandes proyectos de software más fácil de manejar.

```
class Coche{

  int nCilindros, nPuertas;
  color c;
  float velocidad;
  boollean encendido = false;


   // Constructor
  Coche(cilindros, puertas, color) {
    nCilindros = cilindros;
    nPuertas = puertas;
  }

  void encendido() {
    encendido = !encendido;
   }

  boollean getEncendido(){
       return encendido;
   }

}
```




```
Coche Bocho = new Coche(4,2,Amarillo);
Bocho.encendido();

if( Bocho.getEncendido() == true){
    println("Esta prendido el Bocho");
}
```

¡PIEDRA, PAPEL, TIJERAS, LAGARTO, SPOCK!

Piedra, Papel, Tijeras, Lagarto, Spock

Tijeras cortan papel
Papel tapa a piedra
Piedra aplasta a lagarto
Lagarto envenena a Spock
Spock rompe tijeras
Tijeras decapitan lagarto
Lagarto devora papel
Papel desautoriza Spock
Spock vaporiza piedra
Piedra aplasta a tijeras

MIGUELRUIZ
MECHATRONIC ENGINEER

TIPOGRAFIAS

```
PFont letra;
```

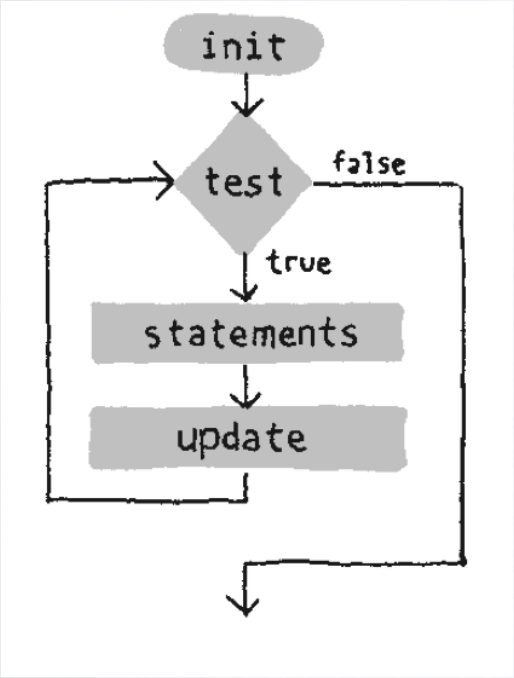
```
letra = loadFont("LetterGothicStd-32.vlw");
textFont(font, 32);
```

- El datatype `PFont` solo puede trabajar con formatos .vlw.
- Crear una fuente .vlw seleccionando "Crear Fuente ..." en el menú Herramientas.
- Debido a que las letras se definen como texturas el tamaño en el que las fuentes se crean deben considerarse en relación con el tamaño en el que se dibujan.
- La función `loadFont()` se utiliza para cargar contenido en un `PFont`.

CICLOS

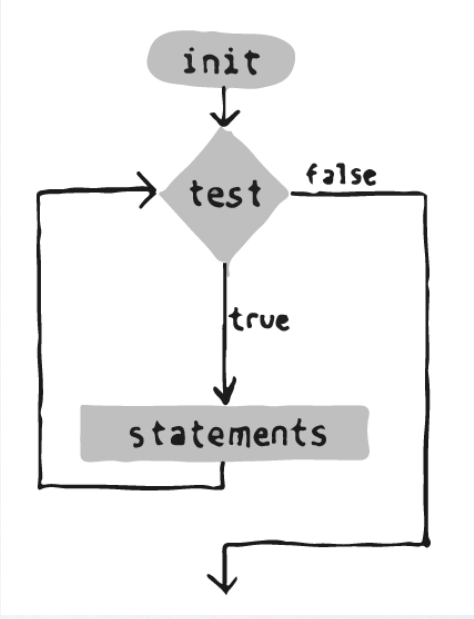
```
for (init; test; update) {
  //Tu código
}
```

```
for (int i = 0; i < 10; i++) {

}
```




```
while (expression) {
//Tu código
}
```

```
int i = 0;
while (i < 10) {
 i += 10;
}
```



¡CHAFA BIRD!

MIGUELRUIZ
MECHATRONIC ENGINEER

STRINGS

Nombre	Acción
`String s2 = trim(s1);`	Elimina espacios entre cadenas.
`String[] lista = split(s1, 'char');`	Separa la cadena cada vez que encuentre el carácter de separación.
`String[] lista = match(s1, "String");`	Busca dentro de la cadena un elemento.
`String s1 = join(lista, separador);`	Junta un arreglo en un solo String, se puede utilizar un separador.

COMUNICACIÓN SERIAL

```
import processing.serial.*;
```

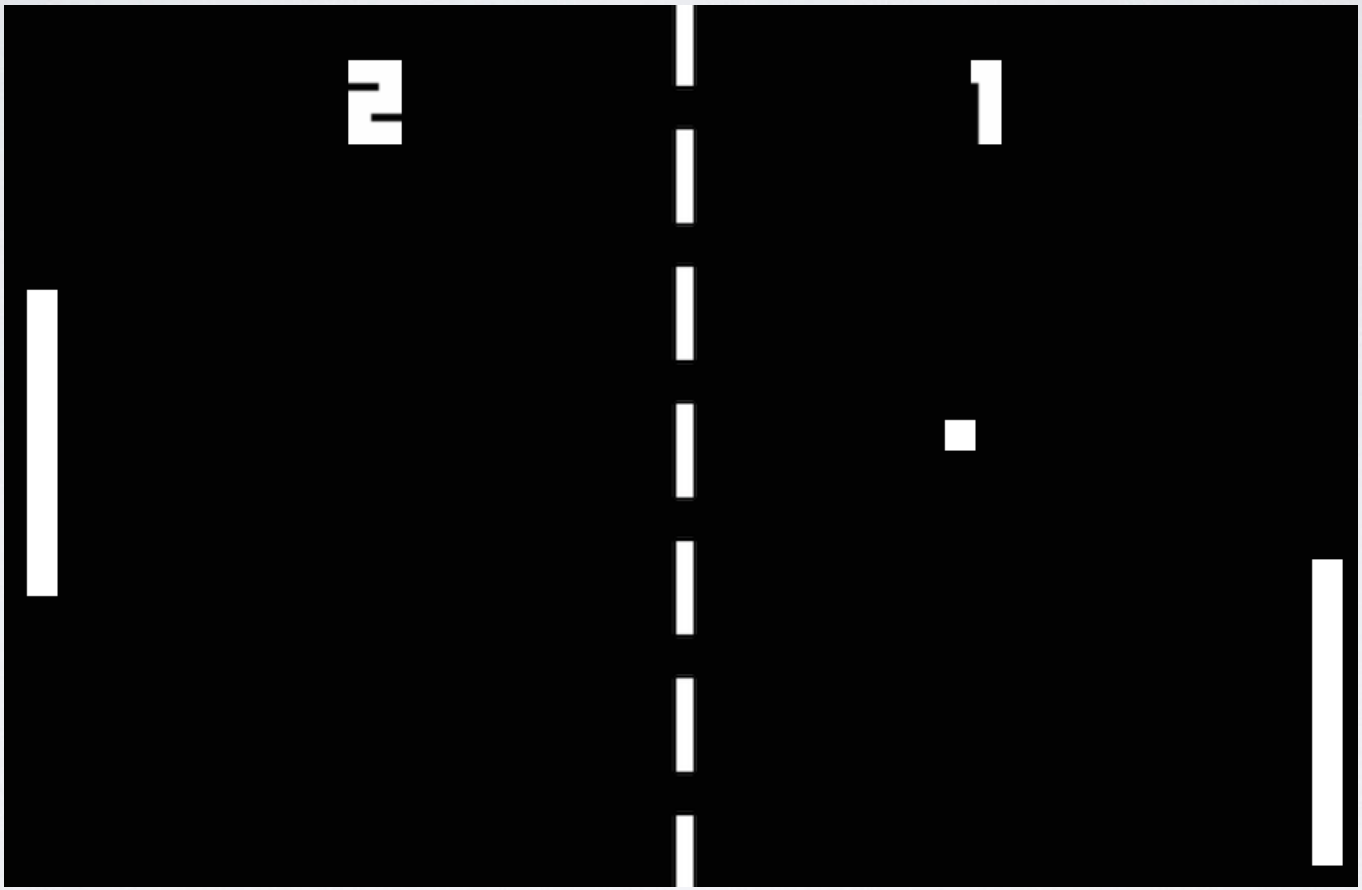
```
Serial myPort;

println(Serial.list());

myPort = new Serial(this, Serial.list()[0], 9600);
```

```
void serialEvent(Serial p) {
  inString = p.readString();
}
```

- La librería de Serial lee y escribe datos un byte desde y hacia dispositivos externos.
- Esta librería permite la comunicación con dispositivos externos como microcontroladores y utilizarlos como entradas o salidas.
- El puerto serial es un puerto de nueve pines I / O que existe en muchos PCs y puede ser emulado a través de USB.

Nombre	Acción
int val = myPort.available()	Regresa el numero de bytes que esperan en el buffer.
int val = myPort.read();	Regresa el primer byte en la cola del biffer.
String cadena = myPort.readString();	Regresa todo el contenido del buffer en un string.
println(Serial.list());	Regresa una lista de todos los puertos disponibles.
myPort.clear();	Elimina todo el contenido del buffer.
myPort.write(int/char/byte);	Envia un byte.
myPort.bufferUntil(valor);	Establece un byte específico al almacenar en el buffer hasta llamar al serialEvent ().
string myString = myPort.readStringUntil(valor);	Le una cadena hasta encontrar un byte específico.
serialEvent()	Funcion que se activa cada vez que llega un nuevo dato al buffer.

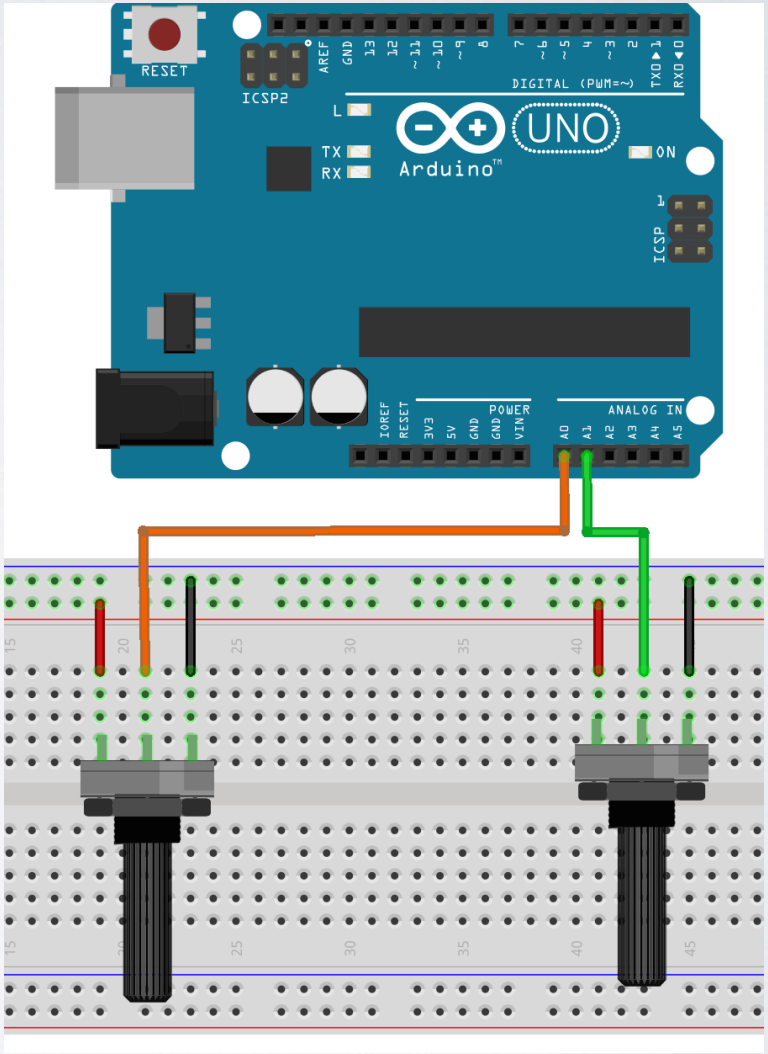
¡PING PONG!



MIGUELRUIZ
MECHATRONIC ENGINEER

ARDUINO PING PONG

```
void setup(){
    Serial.begin(9600);
 }

void loop(){
    Serial.print(analogRead(A0));
    Serial.print(",");
    Serial.print(analogRead(A1));
         Serial.print(" ");
         delay(100);
}
```



MIGUELRUIZ
MECHATRONIC ENGINEER

Creado por: Miguel Angel Ruiz Gálvez
Contacto: miguelo.me

Agradecimientos a:

- Ben Fry
- Casey Reas
- Douglas Adams
- Ellison Leão.

MIGUELRUIZ
MECHATRONIC ENGINEER

creative commons