PROCESSING

INTRODUCCION

- ¿Qué es Processing?

  Sistema que integra un software, un lenguaje de programación y una metodología de enseñanza.

  Busca introducir fundamentos de programación dentro de un contexto visual.

  Herramienta de boceto para imágenes, animaciones e interacciones.

  Herramienta de producción.

- Objetivos

  Proporcionar una mejor forma de escribir código que sirva de apoyo e inspiración para el proceso creativo.

  Desarrollar una herramienta accesible, económica y de fuente abierta (open source).

  Distribución libre.Proyecto

- Proyecto

  Iniciado en 2001 por Ben Fry y Casey Reas en MIT Media Lab.

  Es usado en diversas áreas como: producción de comerciales de TV, videos musicales, visualización de ecosistemas, instalaciones, etc.

INTRODUCCION

- Sobre Processing

  Lenguaje de programación.

- “Plástica Digital”

  Código como materia flexible y modificable a voluntad del creador.

  Código como medio creativo.

  “To sketch electronic media, it’s important to work with electronic materials”

- Formato del taller

EJEMPLOS

BIG SCREENS - NYU's Interactive Telecommunications Program (Daniel Shiffman)

"White Sun"

by Mooshir Vahanvati

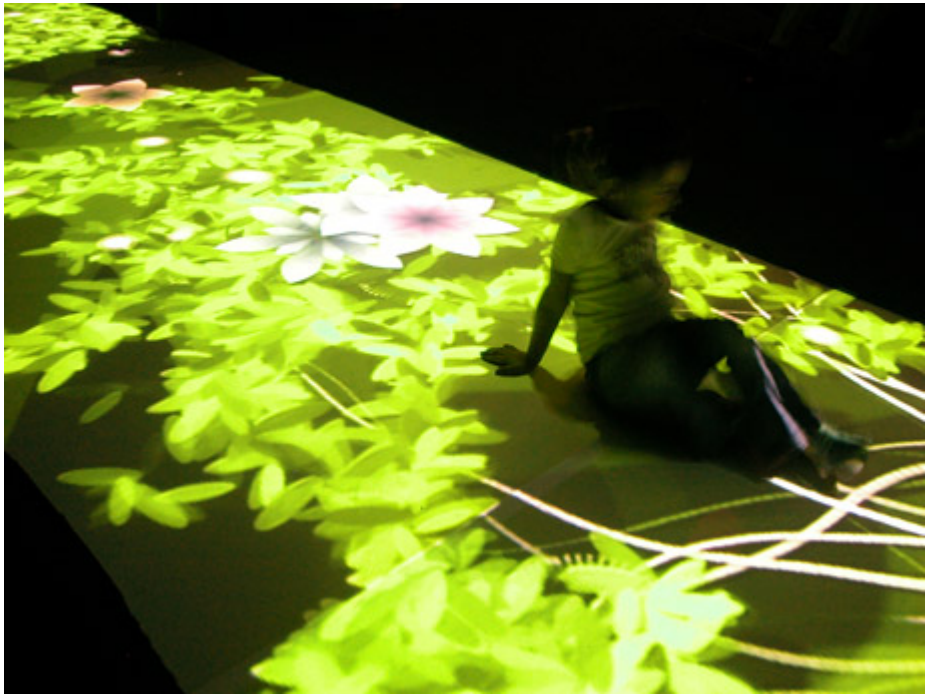
"Caves of Wonder"

by Matt Parker

http://gizmodo.com/5110633/120-feet-of-video-art-final-exams-at-nyus-big-screens-class

EJEMPLOS

"Vattenfall media facade" (ART+COM)



http://www.artcom.de/index.php?option=com_acprojects&page=6&id=30&Itemid=144&details=&lang=en

EJEMPLOS

"Reconfigurable House" (Adam Somlai-Fischer, Et al.)

http://house.propositions.org.uk/

TALLER

- Ambiente de programación de Processing
  - Fundamentos
- Tipos de datos
  - Variables
- Instrucciones repetitivas
  - for() loop
- Propiedades de la forma
  - fill
  - stroke
  - background
- Movimiento
  - Métodos setup() y draw()
- Instrucciones condicionales
  - if ()
- Respuesta y Estímulo
  - Input (mouse)
- Dibujo
  - Herramientas de dibujo

Processing Environment

- Instrucciones

  Son los elementos estructurales del programa. Todas las instrucciones deben finalizar con "**;**"

  Ejemplo:

  ```
  point (100,100);
  ```

- Comentarios

  Son usados para hacer notas (apuntes) entre las líneas de código que por lo general facilitan la comprensión del programa.

  Para hacer comentarios de una línea se debe iniciar el comentario con "**//**"

  Ejemplo:

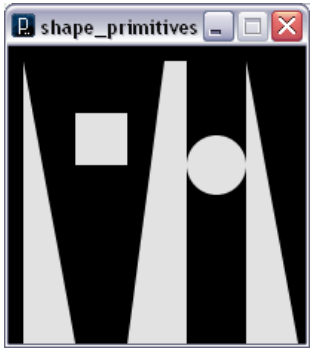
  ```
  //int v - velocidad de desplazamiento.
  ```

  Para hacer comentarios que requieren más de una línea, se debe iniciar el comentario con "**/***" y finalizarlo con "***/**"

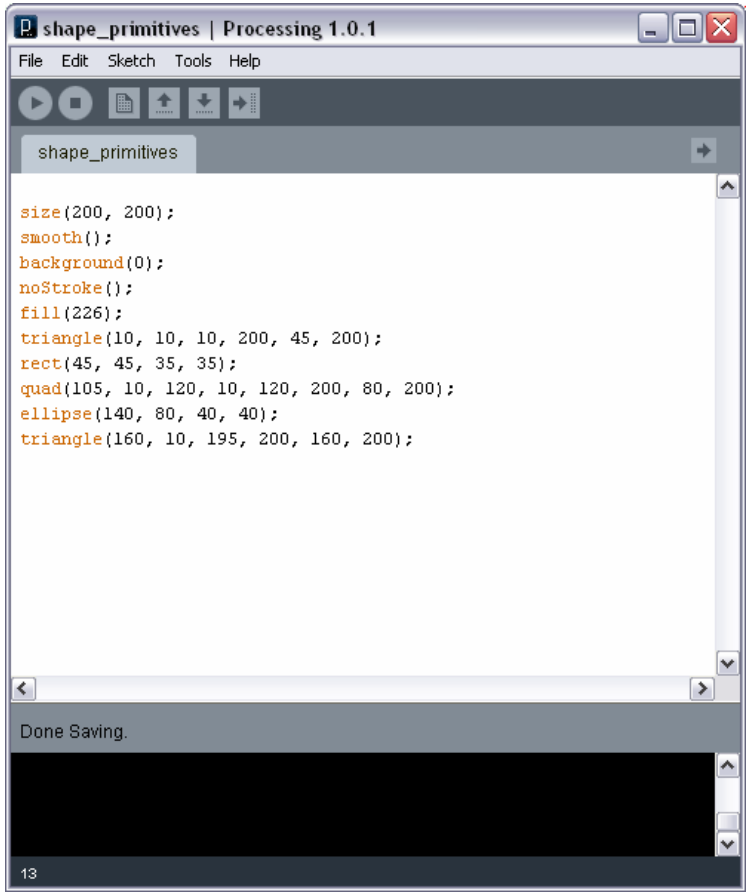
  Ejemplo:

  ```
  /* Esta instrucción dibuja 20 líneas y modifica
  el color según con la posición en el eje y */
  ```

Processing Environment



"Lienzo"



Menú

Barra de Herramientas

Pestañas

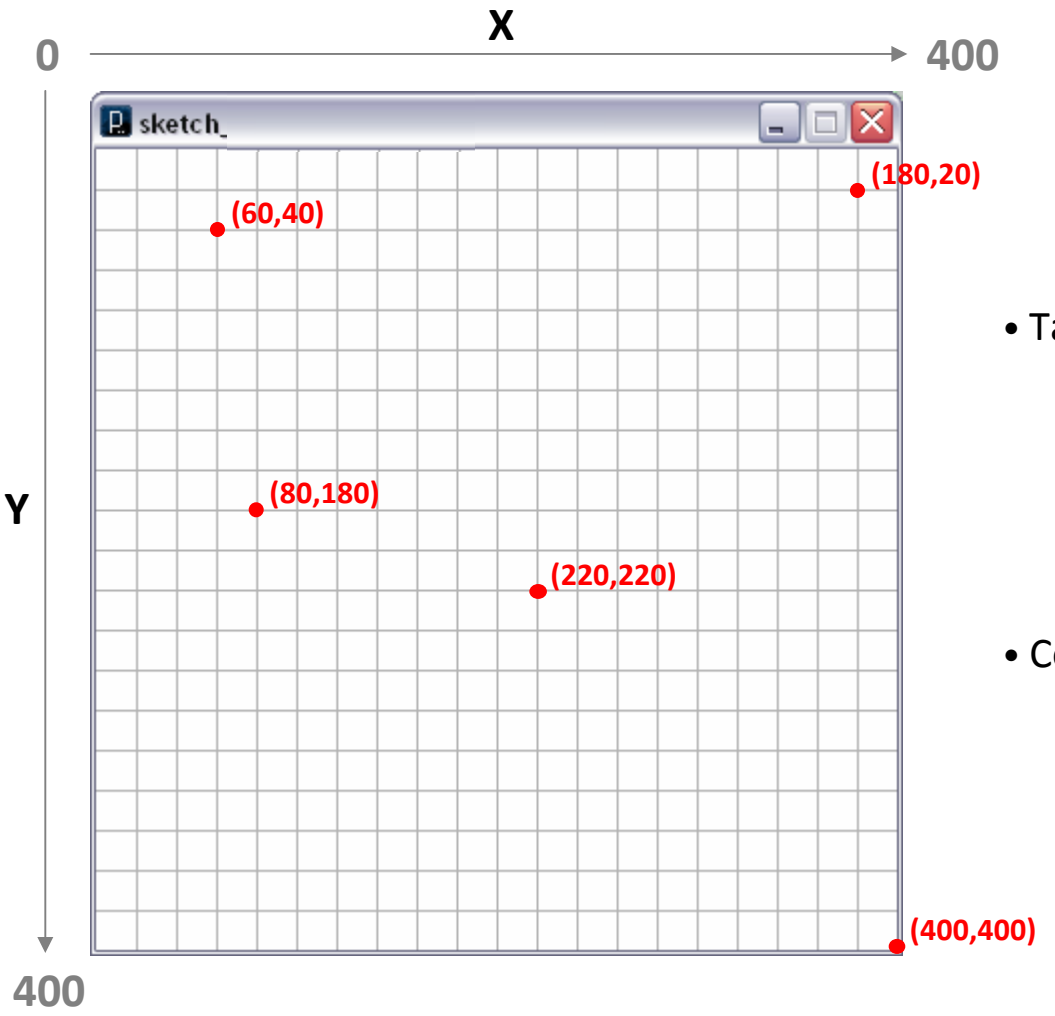
Editor de Texto

Mensajes

Consola

No. De Línea

Processing Environment



- Tamaño del lienzo o superficie de dibujo

  Para establecer el tamaño del lienzo es necesario definir el ancho (eje x) y el alto (eje y) de la superficie.

  size(ancho,alto);

  Ej. size(400,400);

- Coordenada

  Posición (punto) en el espacio.

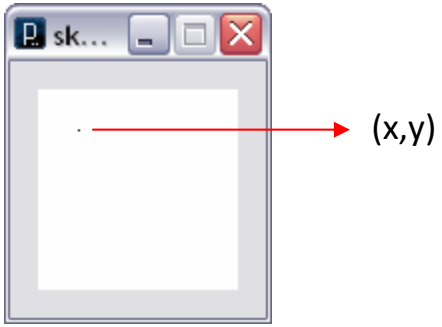
Processing Environment

FORMAS BÁSICAS

- point

Ejemplo:

point (20,20);
point (x,y);
en donde
x = posición en el eje x
y = posición en el eje y



- line

Ejemplo:

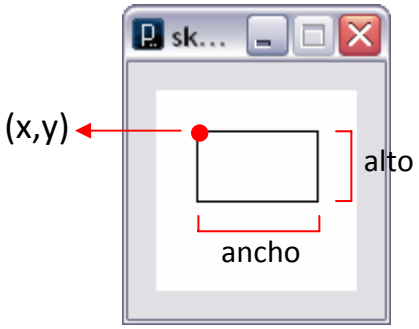
line (**20**,20,**20**,60); line (20,**20**,60,**20**);
line (x,y,x2,y2);
en donde
x = posición en el eje x donde inicia la línea
y = posición en el eje y donde inicia la línea
x2 = posición en el eje x donde termina la línea
y2 = posición en el eje y donde termina la línea

Processing Environment

- rect

Ejemplo:

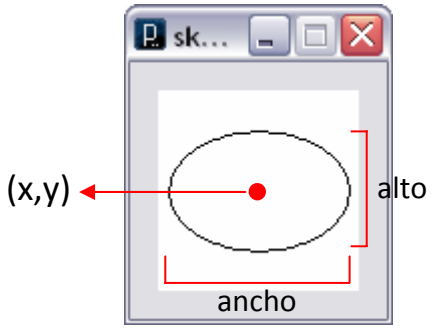
rect (20,20,60,35);
rect (x,y,ancho,alto);
en donde
x = posición inicial en el eje x
y = posición inicial en el eje y
ancho = ancho del rectángulo
alto = alto del rectángulo



- ellipse

Ejemplo:

ellipse (50,50,90,60);
ellipse (x,y,ancho,alto);
en donde
x = posición en el eje x donde inicia la elipse
y = posición en el eje y donde inicia la elipse
ancho = ancho de la elipse
alto = alto de la elipse



EJERCICIOS

EJERCICIO 01: Dibujar tres puntos.

EJERCICIO 02: Dibujar tres líneas: una vertical, una horizontal y una diagonal.

EJERCICIO 03: Dibujar un rectángulo.

EJERCICIO 04: Dibujar cinco círculos.

VARIABLES

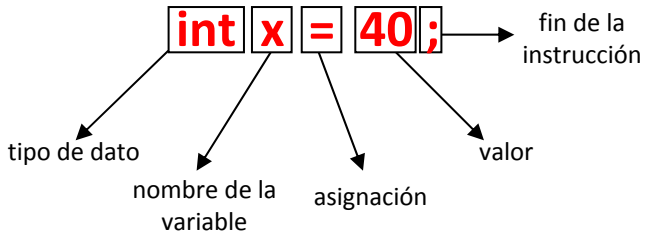
TIPOS DE DATOS

- Enteros (int)

  Permiten modelar características cuyos valores posibles son los valores numéricos de tipo entero.

  Ejemplo:

  ```
  int x = 40;
  int k = -90;
  ```

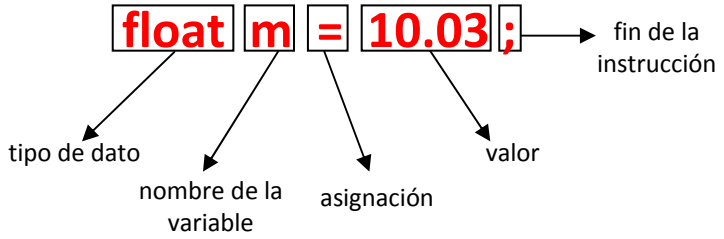


- Reales (float)

  Permiten representar valores de tipo real.

  Ejemplo:

  ```
  float m = 10.03;
  float h = 0.75;
  ```

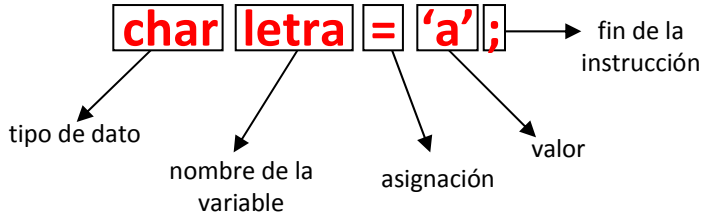


- Caracteres (char)

  Permiten representar caracteres.

  Ejemplo:

  ```
  char letra = 'a';
  char uno = 'z';
  ```



VARIABLES

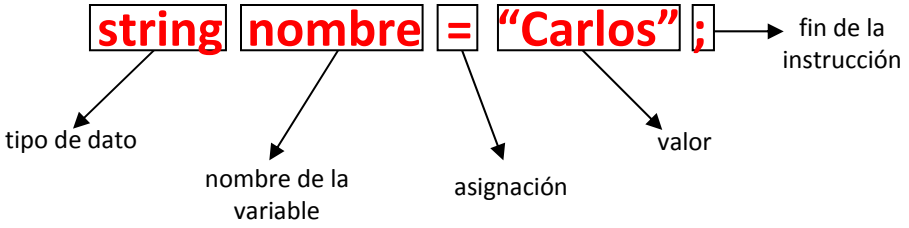
TIPOS DE DATOS

- Cadenas de caracteres (string)

  Permiten representar cadenas de caracteres.

  Ejemplo:

  string nombre = “Carlos”;
  string apellido = “Arango”;

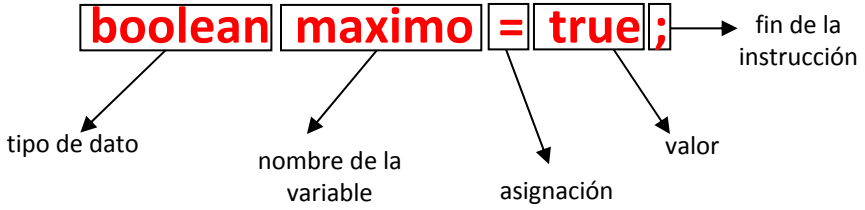


- Valores lógicos (boolean)

  Permiten representar valores lógicos (true/false).

  Ejemplo:

  boolean maximo = true;
  boolean lleno = false;



EJERCICIOS

EJERCICIO 05: Dibujar el rectángulo del ejercicio 03, pero usando variables.

EJERCICIO 06: Dibujar los círculos del ejercicio 05, pero usando variables.

ITERACIONES

INSTRUCCIONES REPETITIVAS

- for ( ) loop

  Esta tipo de instrucción es utilizada para optimizar el código en caso de tener varias líneas de código que ejecutan una instrucción similar. Esto hace que el programa se a mas eficiente y sea mas corto.

  Ejemplo:

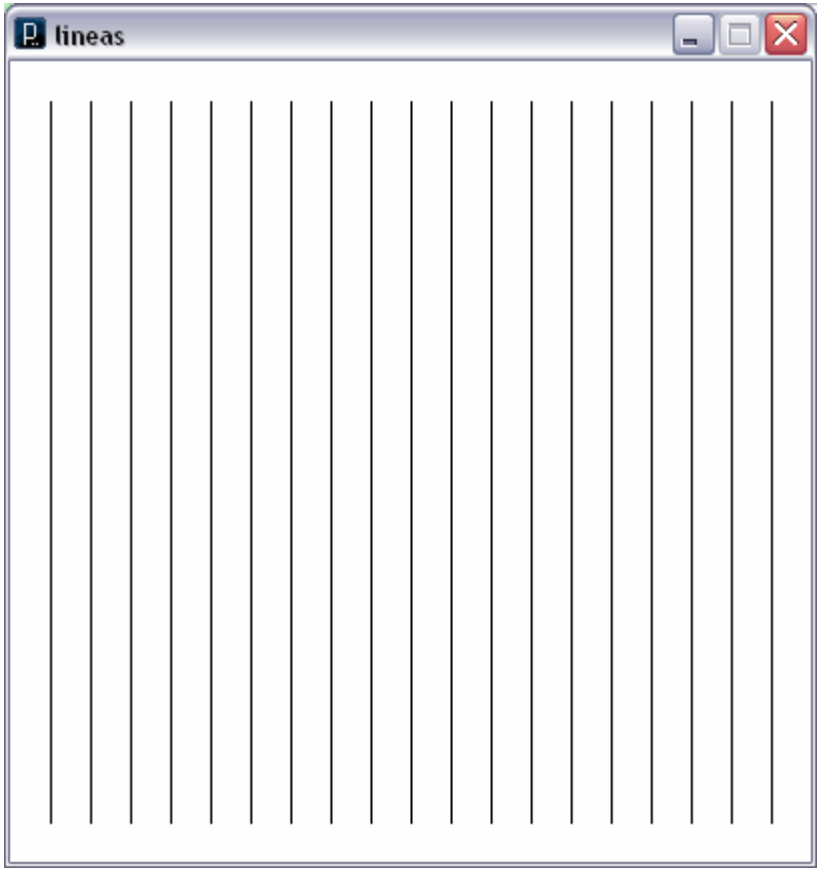
Indice para avanzar en el ciclo.

En donde inicia.

Ejecuta la instrucción hasta que x = 380

Cada vez que se ejecuta la instrucción x se incrementa en 20.

```
for (int x = 20; x <= 380; x = x+20) {
        line (x, 20, x, 380);
}
```



ITERACIONES

Indice para avanzar en el ciclo.

En donde inicia.

Ejecuta la instrucción hasta que x = 380

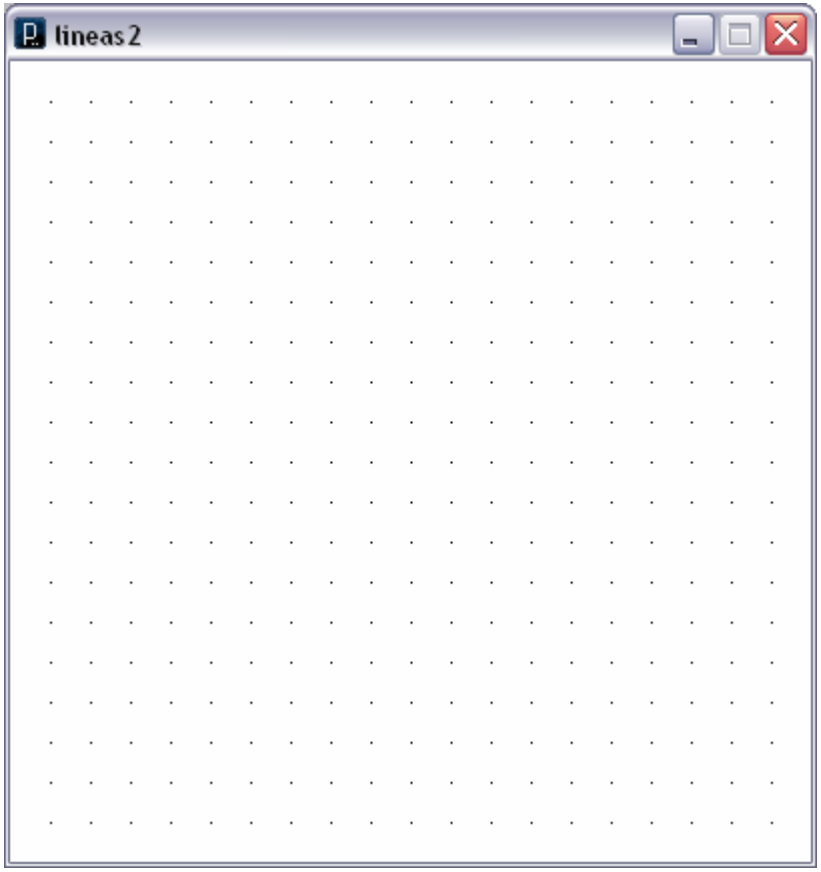
Cada vez que se ejecuta la instrucción x se incrementa en 20.

```
for (int x = 20; x <= 380; x = x+20) {
   for (int y = 20; y <= 380; y = y+20) {
         point (x, y);
   }
}
```

La variable “y” se inicia en 20.

Ejecuta la instrucción hasta que y = 380

Cada vez que se ejecuta la instrucción y se incrementa en 20.



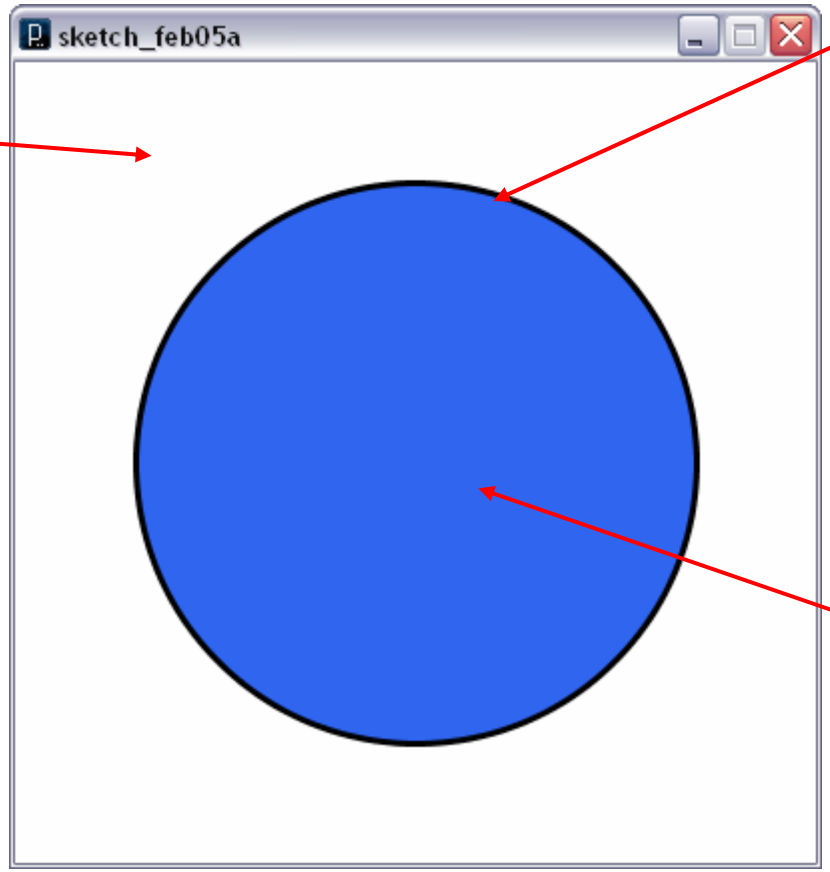
EJERCICIOS

EJERCICIO 07: Crear un patrón usando la instrucción for( ).

EJERCICIO 08: Crear un patrón diferente al del ejercicio 07 usando la instrucción for ( ).

PROPIEDADES DE LA FORMA

FONDO
Color de fondo
background (r,g,b);
background (h,s,b);
background (#000000);



BORDE
Color del borde
stroke (r,g,b);
stroke (h,s,b);
stroke (#000000);

Grosor de línea
strokeWeight (x);

RELLENO
Color de relleno
fill (r,g,b);
fill (h,s,b);
fill (#000000);

PROPIEDADES DE LA FORMA

COLOR

Color RGB (0 – 255)

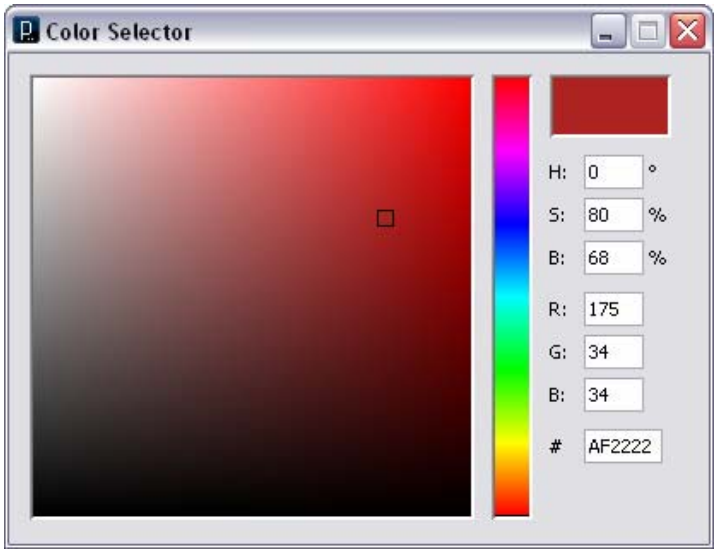
background (valor rojo, valor verde, valor azul, transparencia);
stroke (valor rojo, valor verde, valor azul);
fill (valor rojo, valor verde, valor azul);

Color HSB (1% - 100%)

background (porcentaje tono, porcentaje saturación, porcentaje brillo);
stroke (porcentaje tono, porcentaje saturación, porcentaje brillo, transparencia);
fill (porcentaje tono, porcentaje saturación, porcentaje brillo);

Color #000000 (notación hexagesimal)

background (valor rojo, valor verde, valor azul);
stroke (valor rojo, valor verde, valor azul);
fill (valor rojo, valor verde, valor azul, transparencia);



EJERCICIOS

EJERCICIO 09: Modifique el ejercicio 06 cambiando el color del fondo, el color de relleno y el color del borde.

EJERCICIO 10: Modifique el ejercicio 08 cambiando el color del fondo, el color de relleno y el color del borde.

EJERCICIO 11: Usando el modelo de color RGB, cambie el color del fondo por su color favorito.

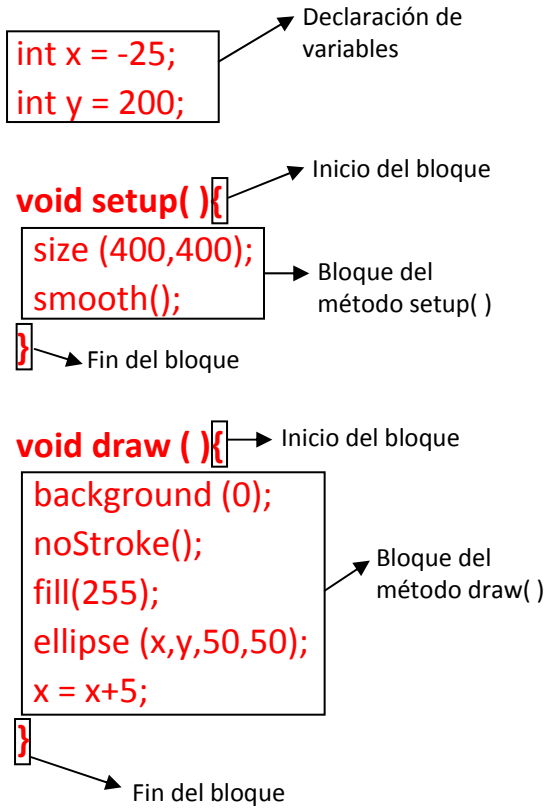
ESTRUCTURA

- void setup()

  Las instrucciones que están dentro de este método solo se ejecutan una vez cuando el programa se inicia.

- void draw()

  Las instrucciones que están dentro de este método se ejecutan indefinidamente (hacen loop) hasta que el programa es detenido. Cada vez las instrucciones contenidas en el método draw() se ejecutan, el ciclo vuelve a empezar ejecutando las instrucciones desde la primera línea hasta la ultima.



NOTA: Las variables cuyo valor cambia con cada loop del método draw() deben ser declaradas y asignadas por fuera de los métodos setup() y draw().

MOVIMIENTO

MOVIMIENTO RECTILINEO (horizontal (eje x), vertical (eje y), diagonal (eje x + eje y))

Poner una figura en movimiento significa que hay un cambio de posición de la misma. Para generar esto, se debe introducir una variable que varíe la posición de la figura en cualquiera de sus ejes.

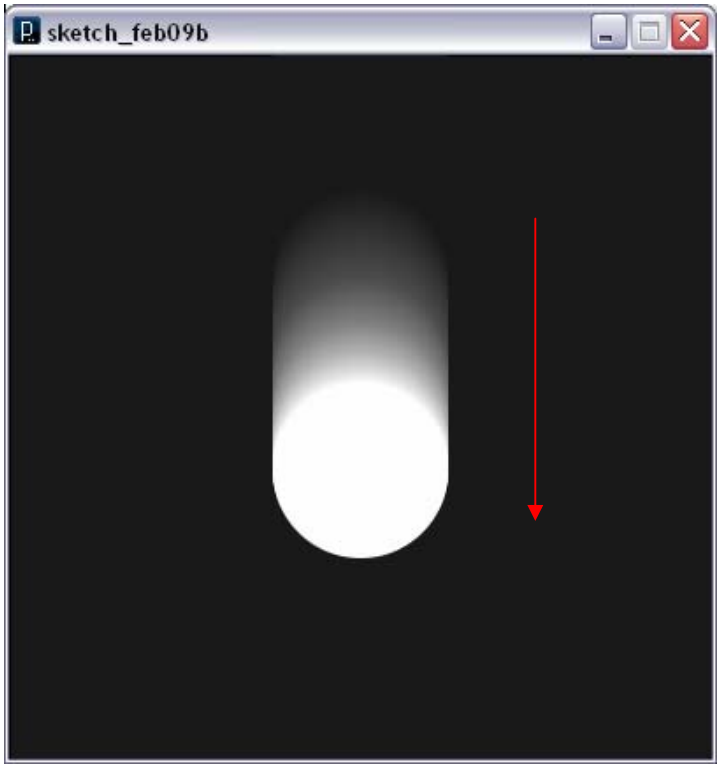
```
int pos_x = 200;
int pos_y = 0;
int diam = 100;
float vel = 0;

void setup( ){
  size (400,400);
  smooth();
}

void draw ( ){
  background (0);
  noStroke();
  fill(255);
  ellipse (pos_x, pos_y+vel, diam, diam);
vel++;
}
```

Declaración de variable que "controla" la velocidad

El cambio de posición se va a dar en el eje y

Indica el incremento de la variable "vel" en cada loop

vel++ equivale a vel = vel+1



MOVIMIENTO

MOVIMIENTO RECTILINEO (horizontal (eje x), vertical (eje y), diagonal (eje x + eje y))

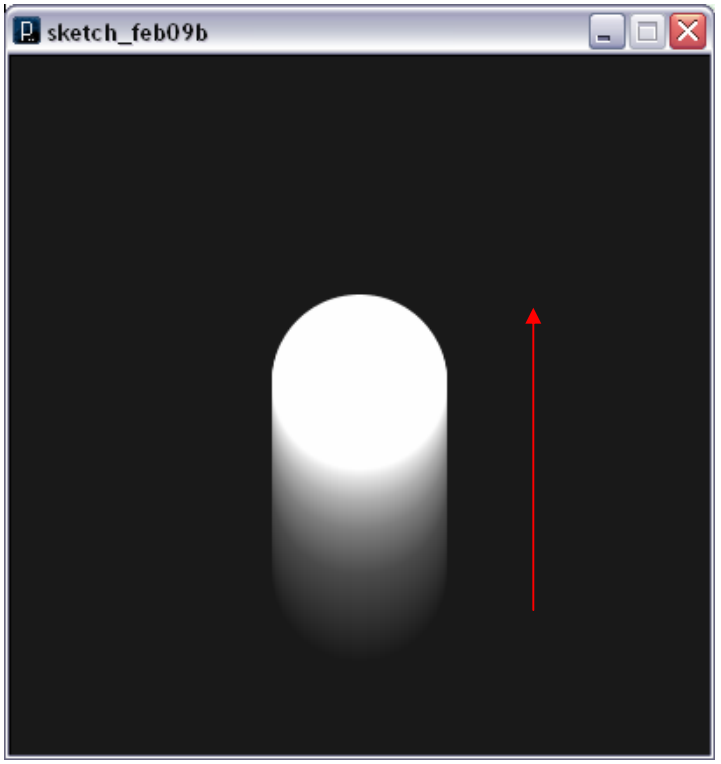
```
int pos_x = 200;
int pos_y = 400;
int diam = 100;
float vel = 0;

void setup( ){
  size (400,400);
  smooth();
}

void draw ( ){
  background (0);
  noStroke();
  fill(255);
  ellipse (pos_x, pos_y+ vel, diam, diam);
  vel--;
}
```

Declaración de variable que "controla" la velocidad

El cambio de posición se va a dar en el eje y

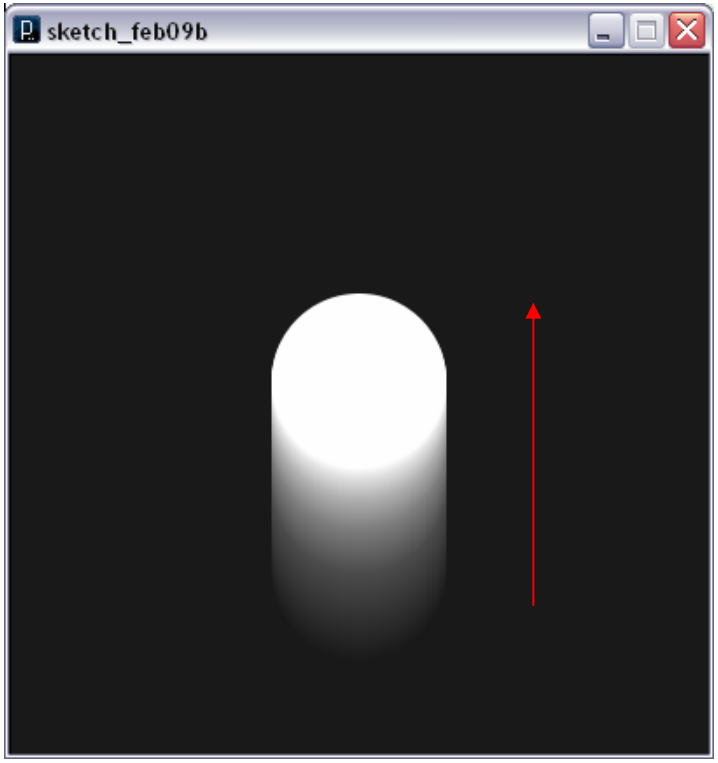
Indica el cambio de la variable "vel" en cada loop

vel-- equivale a vel = vel - 1



MOVIMIENTO

MOVIMIENTO RECTILINEO (horizontal (eje x), vertical (eje y), diagonal (eje x + eje y))



Loop #	Pos_y	Vel	Pos_y + vel
1	100	0	100
2	100	-1	99
3	100	-2	98
4	100	-3	97
5	100	-4	96
N	100	…	…

ellipse (pos_x, **pos_y+ vel**, diam, diam);
vel = vel--;

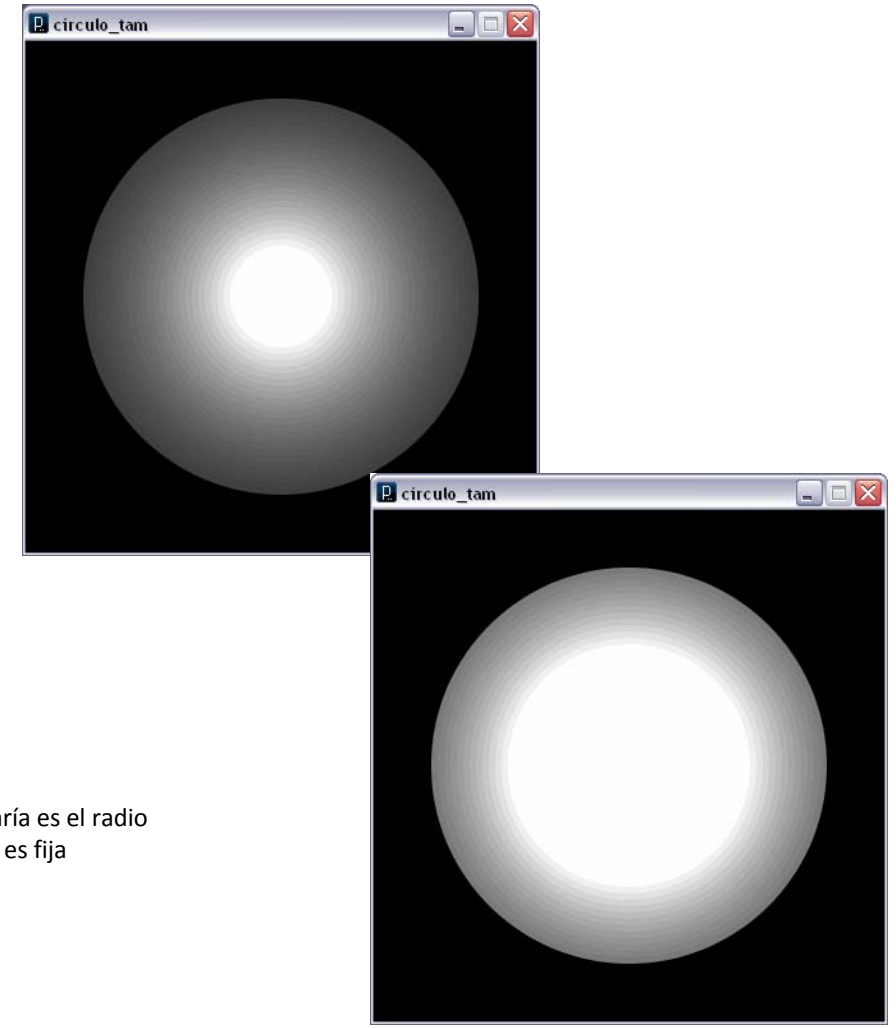
MOVIMIENTO

MOVIMIENTO POR CAMBIO DE PARAMETROS DE FORMA (ancho y alto)

Incrementar gradualmente el diámetro de un círculo.

```
float radio=10;

void setup (){
  size (400,400);
  background(0);
  smooth();
}

void draw (){

  noStroke();
  fill (255);
  ellipse(200,200,radio,radio);
  radio = radio + 0.5;
}
```

El parámetro que se varía es el radio del círculo. La posición es fija



MOVIMIENTO

MOVIMIENTO CIRCULAR

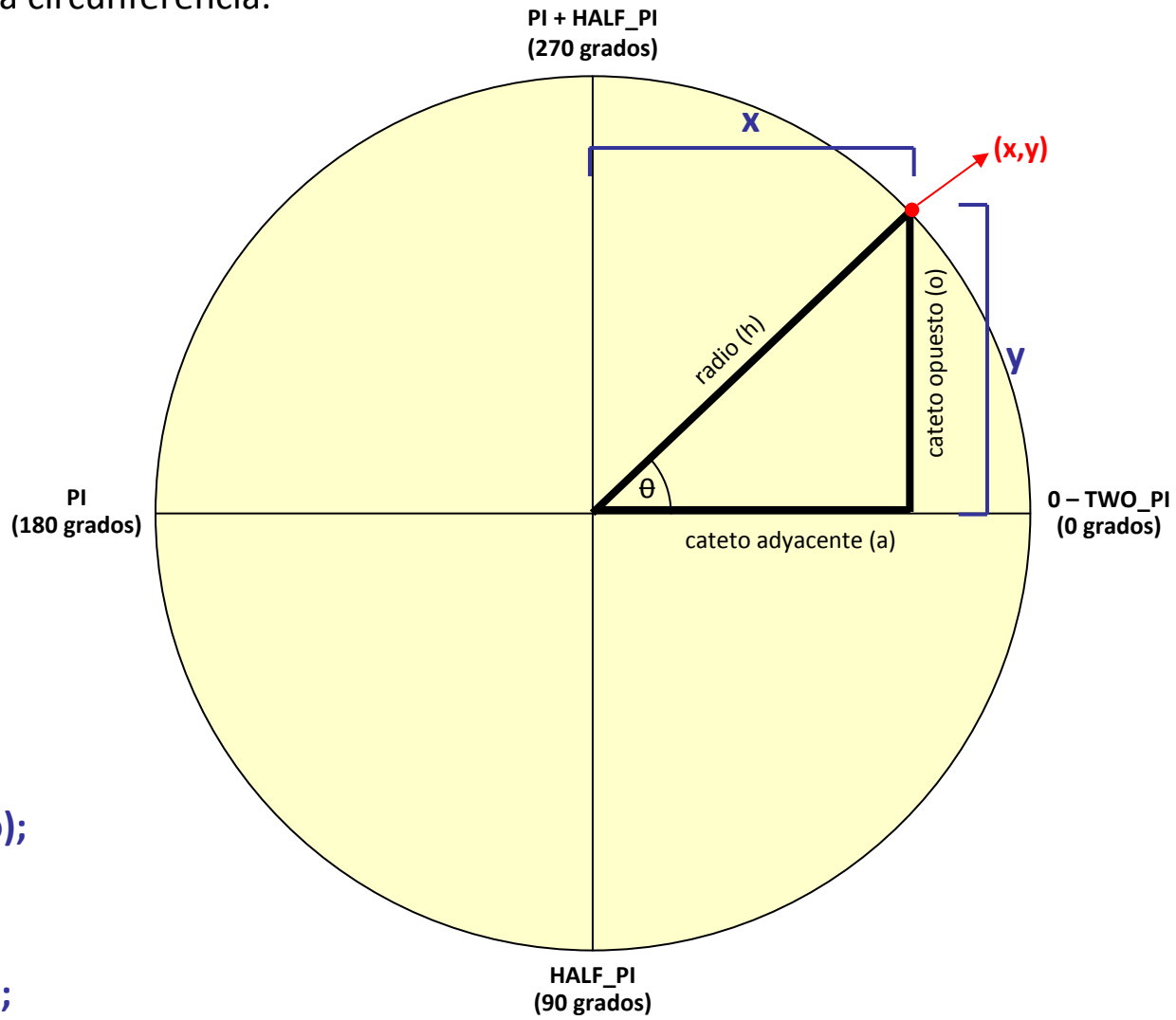
Mover una figura alrededor de una circunferencia.

$$\sin\theta = \frac{o}{h}$$

$$\cos\theta = \frac{a}{h}$$

$$\tan\theta = \frac{o}{a}$$

$$\tan\theta = \frac{\sin\theta}{\cos\theta}$$



a = h*cos (angulo)

```
x = pos_x + radio* cos(angulo);
```

o = h*sin (angulo)

```
y = pos_y +radio* sin(angulo);
```

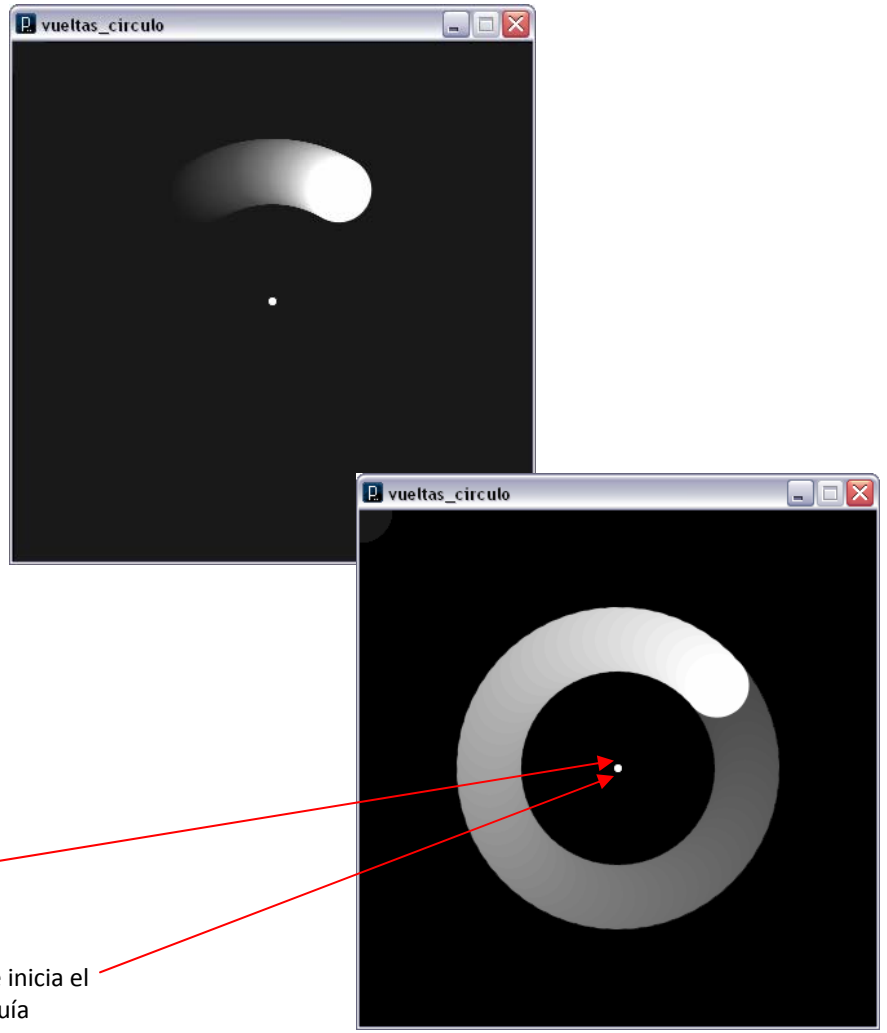
MOVIMIENTO

MOVIMIENTO CIRCULAR

```
float x=0;
float y=0;
float angulo = 0;
float R=100;

void setup (){
  size (400,400);
  background(0);
  smooth();
}

void draw (){
  fill(0,5);
  rect (0,0,width,height);

  noStroke();
  fill (255);
  ellipse(200,200,5,5);

  ellipse (x,y,50,50);
  x = width/2 + R*cos(angulo) ;
  y = height/2 + R*sin(angulo);
  angulo = angulo +0.01;
}
```

Variable que corresponde al valor inicial del ángulo

Radio de la circunferencia guía



Posición en el eje x donde se inicia el dibujo de la circunferencia guía

Posición en el eje y donde se inicia el dibujo de la circunferencia guía

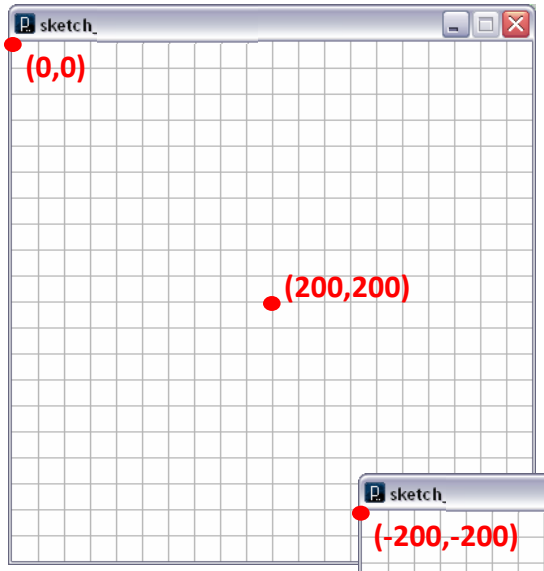
La variación en el ángulo determina la velocidad del movimiento

MOVIMIENTO

ROTACION

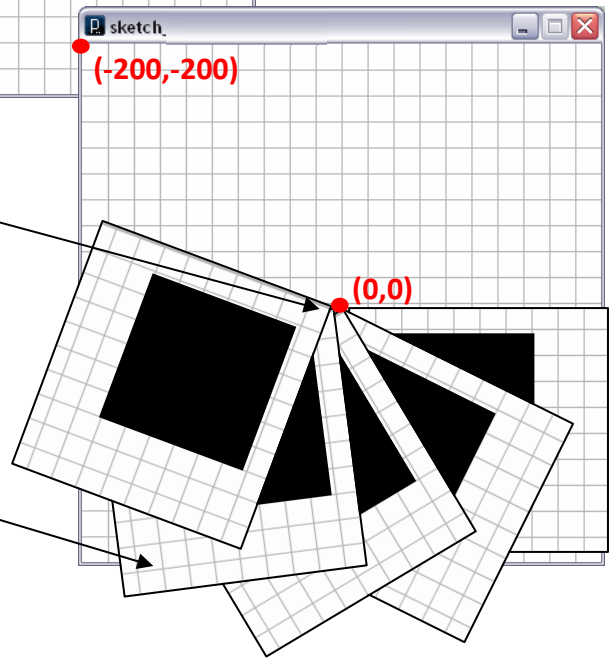
```
float angulo = 0;

void setup() {
  size(400, 400);
  smooth();

}

void draw() {
  fill(0, 12);
  rect (0, 0, width, height);

  fill(255);
  noStroke();
  ellipse(200,200,5,5);

  angulo = angulo + 0.02;
  translate(200, 200);
  rotate (angulo);
  rect(10, 10, 100, 100);
}
```

Antes de la traslación del origen del lienzo



Después de la traslación del origen del lienzo

Ejecuta la rotación del plano a partir del origen del lienzo



MOVIMIENTO

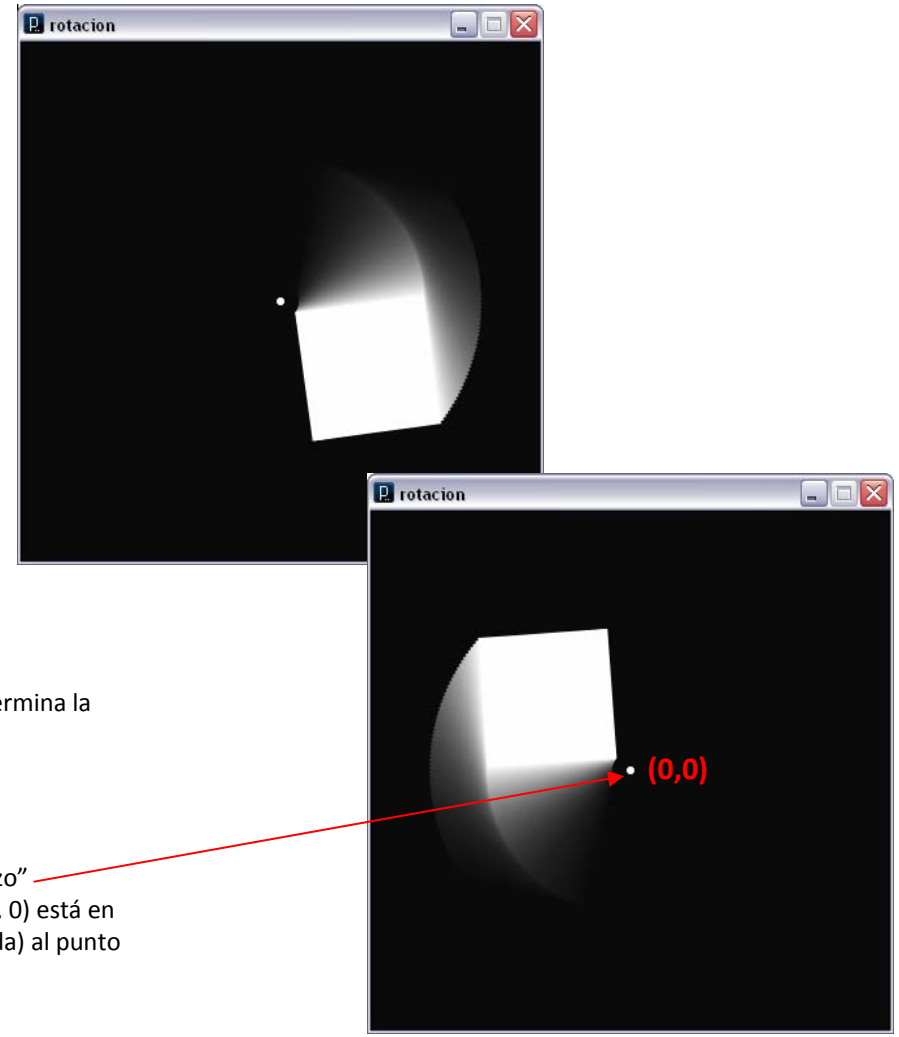
ROTACION

```
float angulo = 0;

void setup() {
  size(400, 400);
  smooth();

}

void draw() {
  fill(0, 12);
  rect (0, 0, width, height);

  fill(255);
  noStroke();
  ellipse(200,200,5,5);

  angulo = angulo + 0.02;
  translate(200, 200);
  rotate(angulo);
  rect(10, 10, 100, 100);
}
```

La variación en el ángulo determina la velocidad del movimiento

Traslada el origen del "lienzo" (originalmente, el punto (0, 0) está en la esquina superior izquierda) al punto señalado (200,200)

Ejecuta la rotación de todo el lienzo tomando como eje el origen del mismo



EJERCICIOS

EJERCICIO 12: Mueva una figura rápidamente a través de la pantalla.

EJERCICIO 13: Mueva tres figuras, cada una de una forma diferente.

INSTRUCCIONES CONDICIONALES

Los operadores relacionales y lógicos son expresiones que permiten describir y evaluar situaciones asociadas a un valor. El resultado de aplicar este tipo de operaciones es un dato de tipo lógico, es decir, verdadero o falso (true or false).

- OPERADORES RELACIONALES

  Se utilizan para hacer comparaciones entre valores.

>	**mayor que**	**a > b**
<	**menor que**	**a < b**
>=	**mayor o igual que**	**a >= b**
<=	**menor o igual que**	**a >= b**
==	**igual que**	**a == b**
!=	**diferente de**	**a != b**

- OPERADORES LOGICOS

  Se utilizan para describir situaciones mas complejas a partir de la composición de varias expresiones relacionales.

&&	**y**	**(a > b) && (c >= a)**
||	**o**	**(a > b) || (c >= a)**
!	**no**	**! (a > b)**

INSTRUCCIONES CONDICIONALES

- OPERADORES RELACIONALES

y
&&

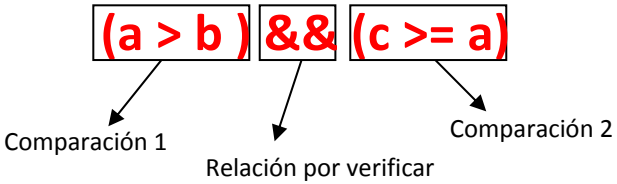
Comparación 1	Comparación 2	Resultado
true	true	true
true	false	false
false	true	false
false	false	false

o
||

Comparación 1	Comparación 2	Resultado
true	true	true
true	false	true
false	true	true
false	false	false

no
!

Comparación	Resultado
true	false
false	true



Caso 1: true
a = 10
b = 5
c = 15

Caso 2: false
a = 10
b = 5
c = 4

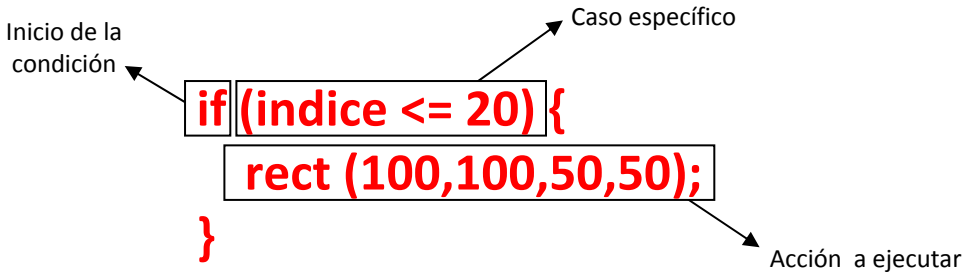
Caso 3: false
a = 10
b = 15
c = 20

Caso 4: false
a = 10
b = 5
c = 8

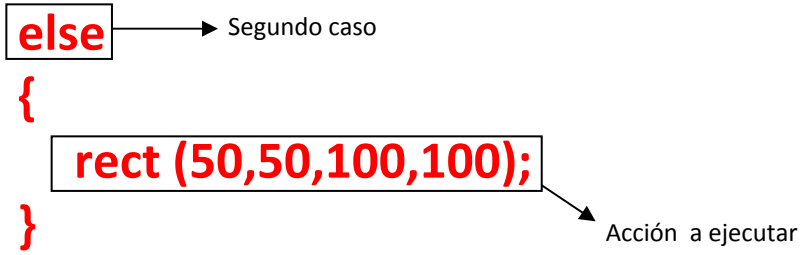
INSTRUCCIONES CONDICIONALES

- if ()

Las instrucciones condicionales permiten plantear acciones y ejecutar acciones en casos específicos. Para crear una instrucción condicional es necesario establecer cuales son los casos específicos y cuales son las acciones a ejecutar para cada caso.



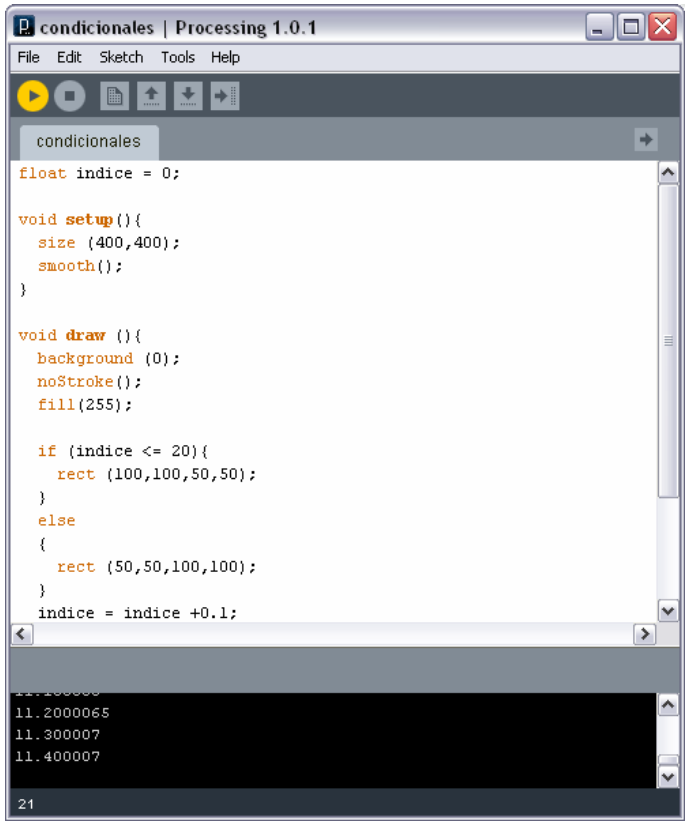
Si “indice” es menor o igual a 20 entonces dibuje un cuadrado en la posición (100,100), con 50 de lado….

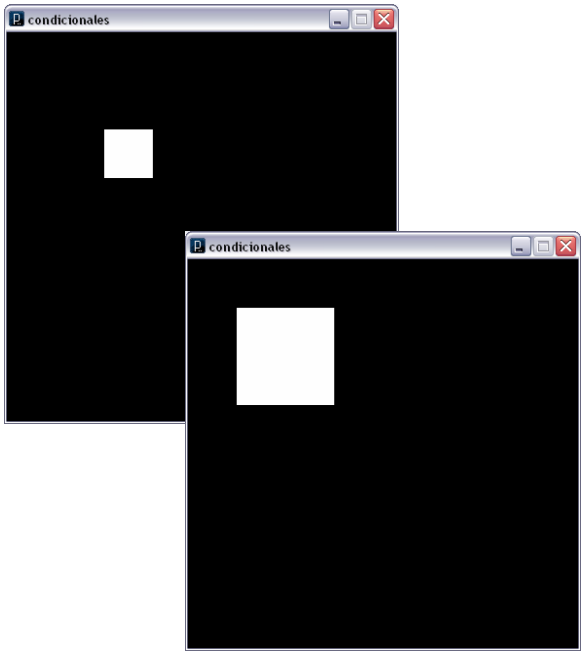


…. Sino (si “indice” es diferente y mayor que 20) dibuje un cuadrado en la posición (50,50), con 100 de lado.

INSTRUCCIONES CONDICIONALES

EJEMPLO

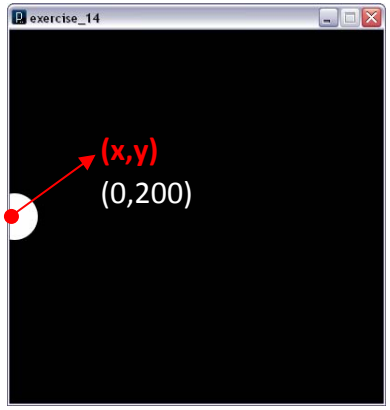




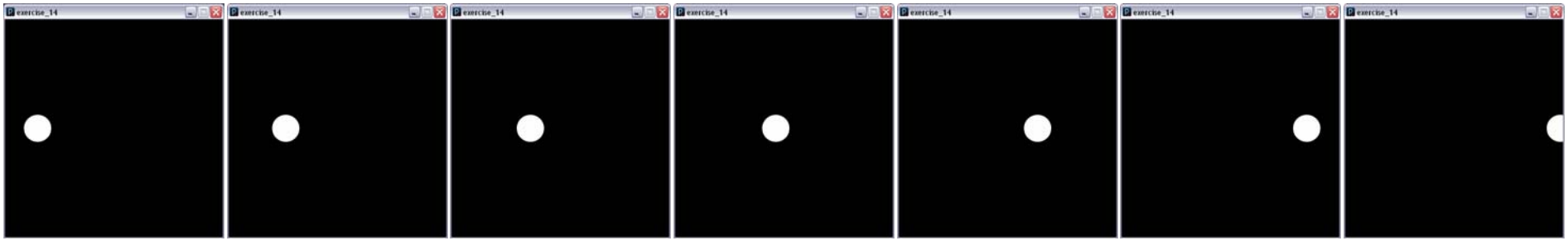

```
float indice = 0;

void setup(){
  size (400,400);
  smooth();
}

void draw (){
  background (0);
  noStroke();
  fill(255);

  if (indice <= 20){
    rect (100,100,50,50);
  }
  else
  {
    rect (50,50,100,100);
  }

  indice = indice + 0.1;
  println (indice);
}
```

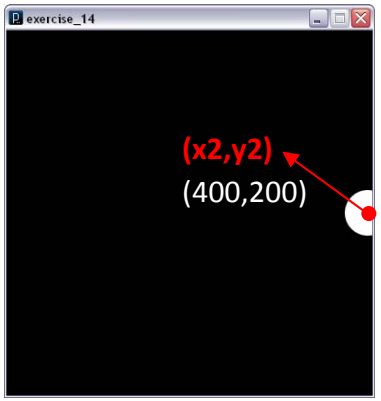
Para ver el valor de "indice" en la consola

INSTRUCCIONES CONDICIONALES




```
x = 0;
y = 200;
ellipse (x,y,50,50);
```





INSTRUCCIONES CONDICIONALES

ZONA 1

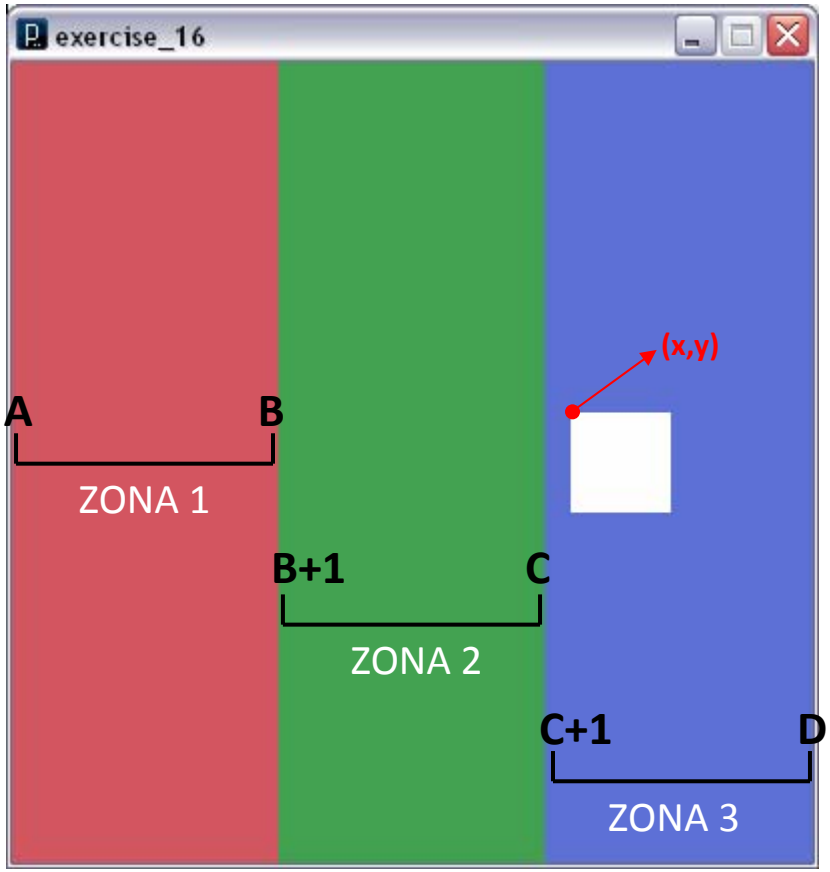
Si “x” está entre A y B, entonces x=x+1

ZONA 2

Si “x” está entre B+1 y C, entonces x=x+3

ZONA3

Si “x” está entre C+1 y D, entonces x=x+0.1



EJERCICIOS

EJERCICIO 14: Mueva una figura a través de la pantalla. Cuando ésta pase el borde debe aparecer nuevamente por el lado opuesto de la pantalla.

EJERCICIO 15: Mueva una figura a través de la pantalla. Cuando ésta toque el borde debe cambiar de devolverse.

EJERCICIO 16: Mueva una figura a través de la pantalla que cambie de movimiento a medida que ésta pasa por diferentes zonas de la pantalla.

INTERACCION

MODELO ESTIMULO – RESPUESTA

Permite establecer una relación entre lo que se reconoce como el estímulo (acción, entrada, input) con la respuesta esperada (reacción, salida, output)

Acción – Reacción
Entrada – Salida
Input - Output

Estímulo	relación	Respuesta
x = 60	1:1	rect (x,100,50,50);
x = 60	1:2	rect (x/2,100,50,50);
x = 60	2:1	rect (x*2,100,50,50);

INTERACCION

MOUSE como estímulo

Es posible hacer uso de los datos recogidos por el mouse para controlar , la posición, tamaño, color, y otras propiedades de las formas.

Los datos proporcionados por el mouse, son interpretados como una posición en la pantalla, es decir, indica un valor específico para el ***eje x***, y otro para el ***eje y*** los cuales pueden ser modificados de acuerdo a la manipulación del mouse.

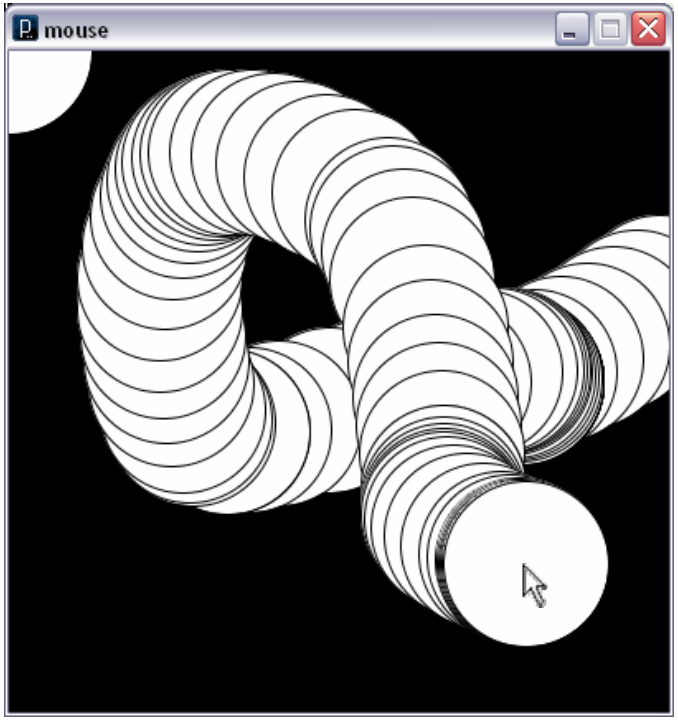
- FUNCIONES

mouseX (posición en el eje x del mouse)

mouseY (posición en el eje y del mouse)

Ejemplo:

```
void setup(){
  size (400,400);
  background(0);
  smooth();
}

void draw (){
  ellipse (mouseX, mouseY, 100,100);
}
```



INTERACCION

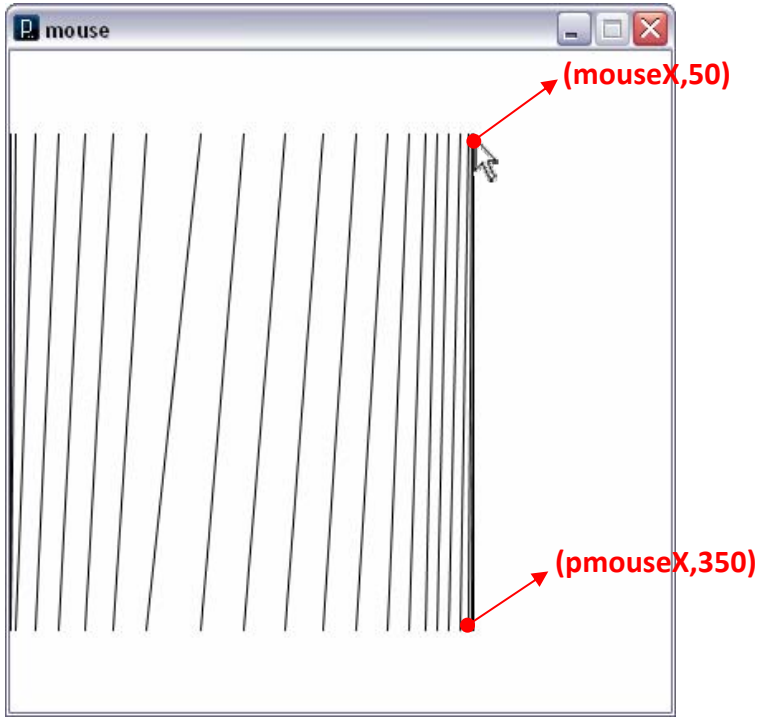
- FUNCIONES

pmouseX (guarda la posición en x del mouse en el cuadro (frame) anterior al cuadro (frame) actual.)

pmouseY (guarda la posición en y del mouse en el cuadro (frame) anterior al cuadro (frame) actual.)

Ejemplo:

```
void setup(){
  size (400,400);
  background(255);
  smooth();
}

void draw (){
  stroke(0);
  line (mouseX, 50,pmouseX,350);
}
```



EJERCICIOS

EJERCICIO 17: Dibuje una figura que siga el cursor.

EJERCICIO 18: Dibuje una figura que se mueva de una forma diferente al cursor.

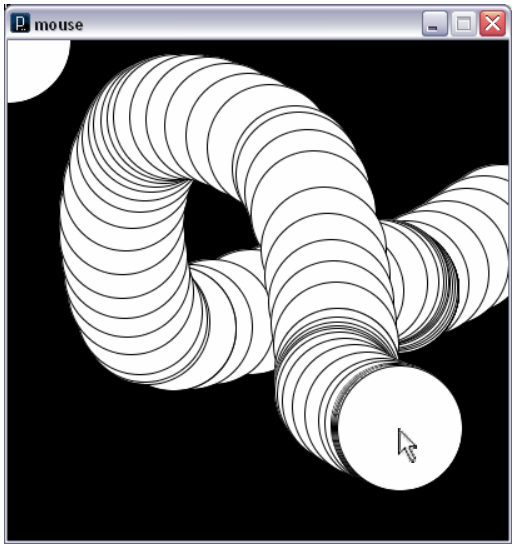
EJERCICIO 19: Dibuje tres figuras que se muevan de formas diferentes respecto al cursor.

DIBUJO

BACKGROUND

- background dentro del método setup( )

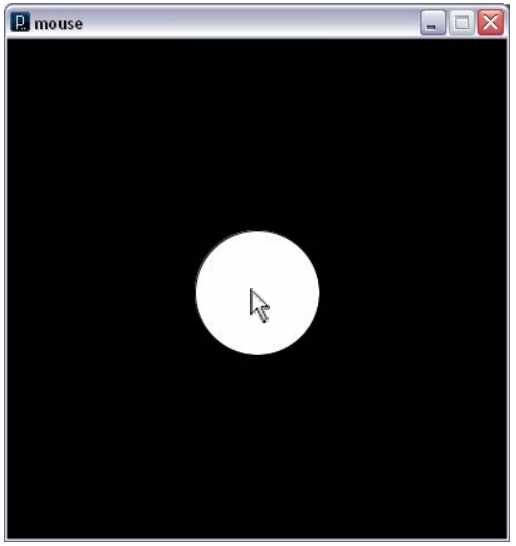
  Cuando las características de la función “background( )” están descritas en el método setup( ), esto significa que el fondo solo va a ser pintado 1 vez al ejecutar el programa.




```
void setup(){
  size (400,400);
  background(0);
  smooth();
}

void draw (){
  ellipse (mouseX, mouseY, 100,100);
}
```

- background dentro del método draw( )

  Cuando las características de la función “background( )” están descritas en el método draw( ), el fondo se pinta cada vez que se repite el draw( ) lo que genera el efecto de refrescar la pantalla

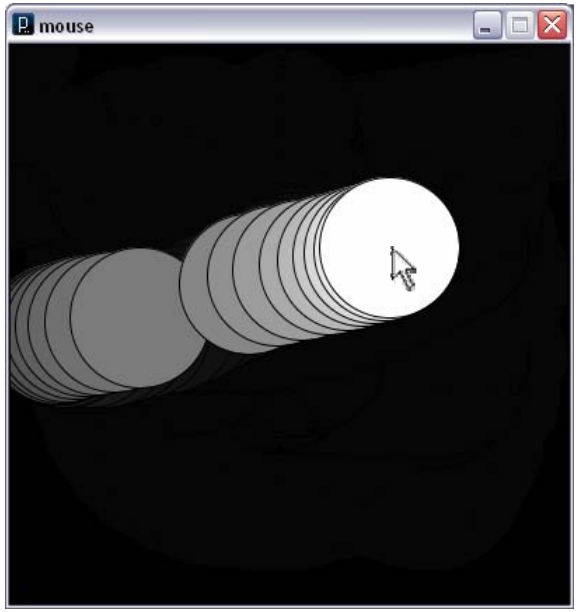



```
void setup(){
  size (400,400);
  smooth();
}

void draw (){

  background(0);
  ellipse (mouseX, mouseY, 100,100);
}
```

BACKGROUND

- transparencia

Para conseguir un efecto de “refrescar pantalla” gradualmente, es necesario aplicar un fondo falso (un rectángulo cuyo relleno tenga una transparencia) que se dibuja cada vez que el metodo draw( ) hace un loop.




```
void setup(){
 size (400,400);
 background(0);
 smooth();
}

void draw (){
 fill(0,0,0,20);
 rect(0,0,width,height);

 fill (255);
 ellipse (mouseX, mouseY, 100,100);
}
```

Color base para el fondo

Color de relleno para el fondo falso

Rectángulo que dibuja el fondo falso

DIBUJO

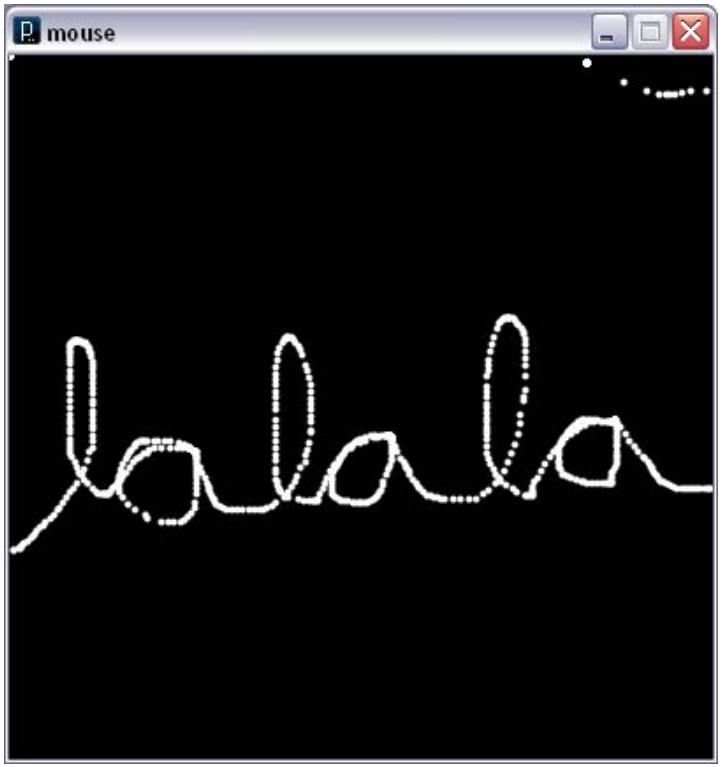
HERRAMIETA DE DIBUJO

Es una forma interactiva de dibujar. El usuario de alguna forma puede “controlar” el comportamiento de lo que se dibuja en el lienzo.

Ejemplo:

```
void setup(){
  size (400,400);
  background(0);
  smooth();
}

void draw (){
  strokeWeight(4);
  stroke (255);
  point (mouseX, mouseY);
}
```

Patrón de dibujo



HERRAMIETA DE DIBUJO CONFIGURABLE

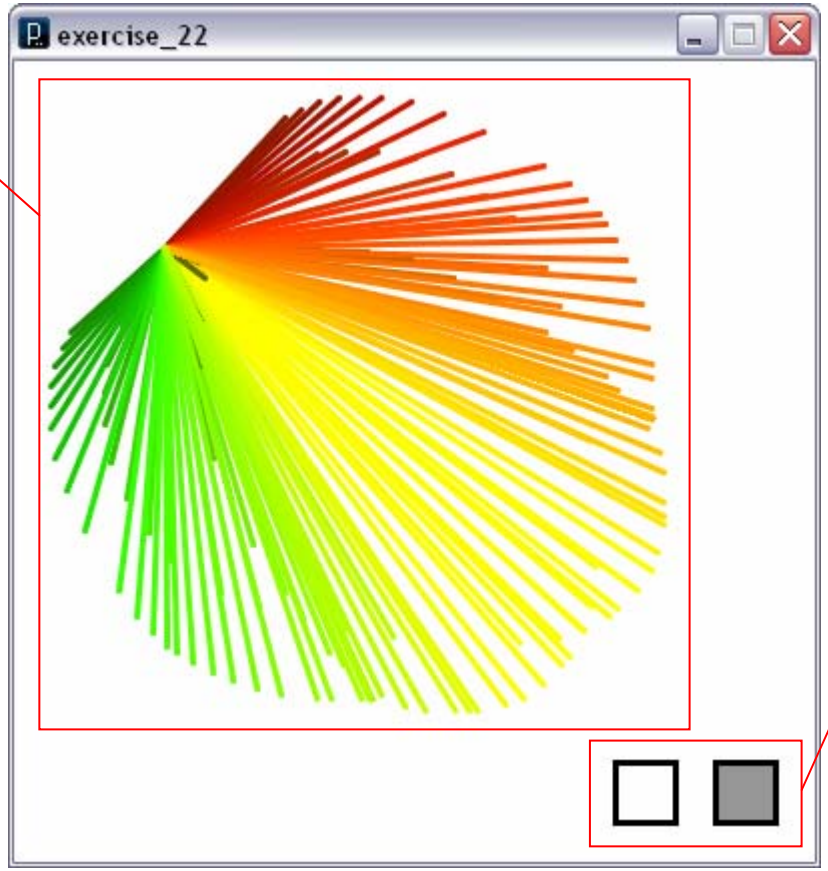
El color de las líneas depende de la posición en el eje x del mouse y del botón del mouse que es presionado

Si el botón izquierdo está presionado

stroke (mouseX, mouseY,0);

Si el botón derecho está presionado

stroke (mouseX);



Para cambiar el color del fondo y borrar el lienzo

Si alguno de los botones del mouse está presionado sobre la zona del rectángulo blanco

background (255);

EVENTOS DE MOUSE

- mousePressed

  Funciona como una variable de tipo boolean que permite identificar cuando alguno de los botones del mouse es presionado.

  Ejemplo

```
void setup(){
  size (400,400);
  smooth();
}

void draw (){
  background(0);
  if (mousePressed){
     ellipse (mouseX, mouseY, 100,100);
  }
}
```

Si alguno de los botones del mouse está presionado, dibuje el círculo descrito.

- mouseButton

  Funciona como una variable de tipo boolean que permite identificar que botón del mouse fue presionado (RIGHT o LEFT).

  Ejemplo

```
void setup(){
  size (400,400);
  smooth();
}

void draw (){
  background(0);
  if (mousePressed){
    if (mouseButton == LEFT){
      ellipse (mouseX, mouseY,
100,100);
    }
  }
}
```

Si el botón izquierdo del mouse está presionado, dibuje el círculo descrito.

EJERCICIOS

EJERCICIO 20: Dibuje un rastro de puntos que muestre la trayectoria del cursor en el espacio.

EJERCICIO 21: Cree una herramienta de dibujo personalizable.

EJERCICIO 22: Cree una herramienta de dibujo que cambie cuando es presionado el botón derecho/izquierdo del mouse.