ARTE ELECTRÓNICO

PROCESSING

- **Lenguaje de programación basado en Java.**
- **Código abierto (Open-Source): software distribuido y desarrollado libremente.**
- **Traduce expresiones matemáticas en gráficos.**
- **Desarrollo de gráficos y programación de visuales.**
- **Herramienta de boceto para imágenes, animaciones e interacciones.**
- **Comunicación con la máquina a través de instrucciones: CÓDIGO.**

¿QUÉ ES PROGRAMAR?

El algoritmo y sus características

Programar es escribir algoritmos. Al escribir líneas de código que posteriormente generarán un programa de cómputo, lo que estamos haciendo es traducir un algoritmo. Pero, ¿qué es entonces un algoritmo? Un algoritmo es simplemente un conjunto de pasos ordenados que nos ayudarán a resolver cualquier problema.

Programar es definir instrucciones para ser ejecutadas por una computadora (programa). El objetivo de programar es invariablemente resolver un problema.

¿POR QUÉ PROCESSING?

- PROYECTO iniciado en 2001.

(Ben Fry y Casey Reas en Mit Media Lab.)

- Programación con fines artísticos y diseño

- Código escrito como medio creativo.

- Aprendizaje programación = proyecto complejo

- Trabajamos con una interfaz y sintaxis muy simplificada.

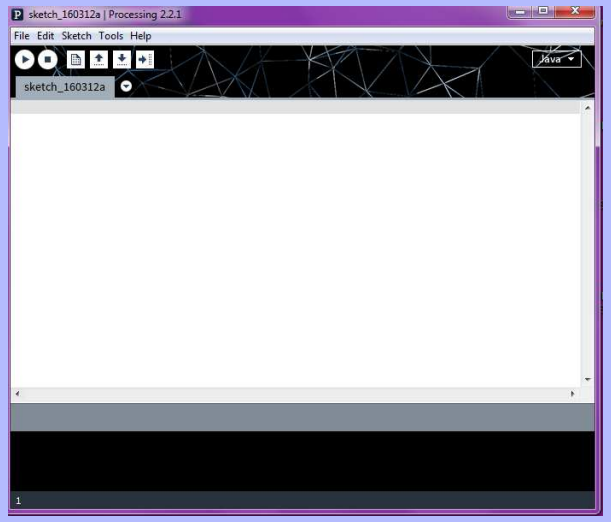
ORDEN

- ¿Qué queremos hacer o con qué parámetros queremos experimentar?
- Sucesión lógica de acontecimientos
- Código con la sintaxis adecuada
- Ejecutar

COMENZANDO…

DESCARGA DEL PROGRAMA:

- https://processing.org/download/



LA INTERFAZ

*** RUN: ejecuta el código**

*** STOP: para el programa**

*** NEW: crea un proyecto nuevo o sketch**

*** OPEN: abre un proyecto o sketch existente**

*** SAVE: guarda un sketch con la extensión .pde.(“save us”)**

*** EXPORT: Sirve para preparar el sketch para ser ejecutado.**

*** ESPACIO DE CÓDIGO Y CONSOLA**

SISTEMA DE COORDENADAS

- La pantalla de visualización que creamos con Processing es un gráfico de píxeles cada uno de ellos registrado con una coordenada (X,Y)

ej/ **point (20,20);**

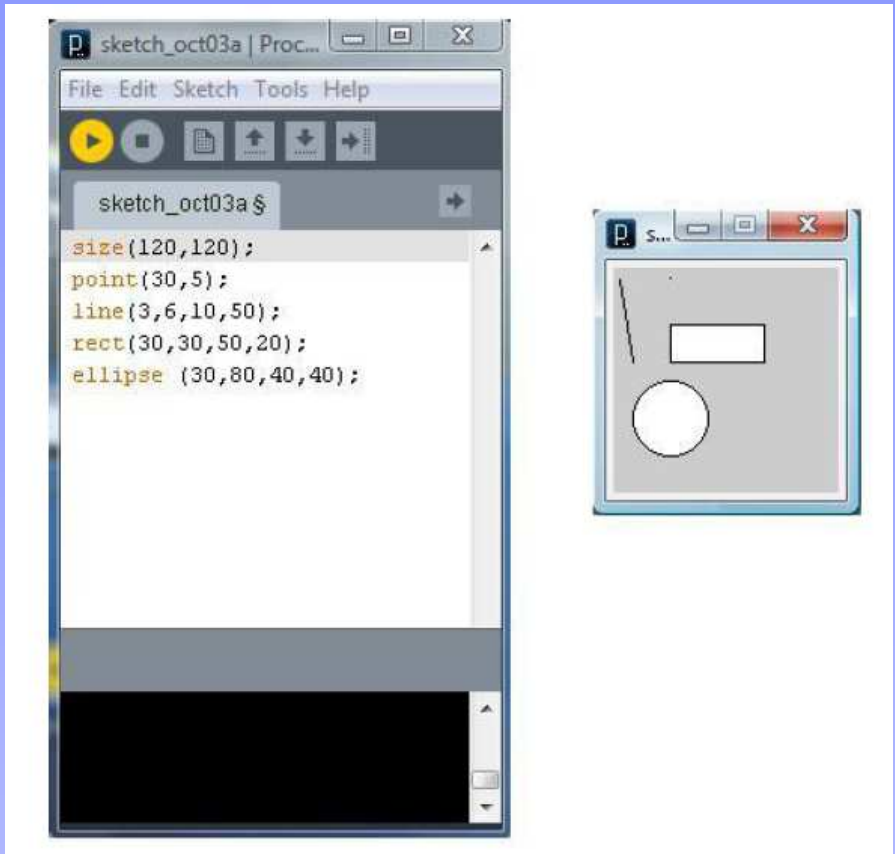
- Centro de coordenadas: esquina superior izquierda del espacio de trabajo.
- Eje Y invertido.
- Unidad en la que trabajamos: pixel

El *pixel*, término muy utilizado en computación, se refiere al elemento más pequeño de los que componen una imagen digital. Empezamos esta iniciación a Processing con esta definición porque es el elemento visual básico que nosotros podemos representar en Processing.

- Canvas o lienzo: size (100,100)

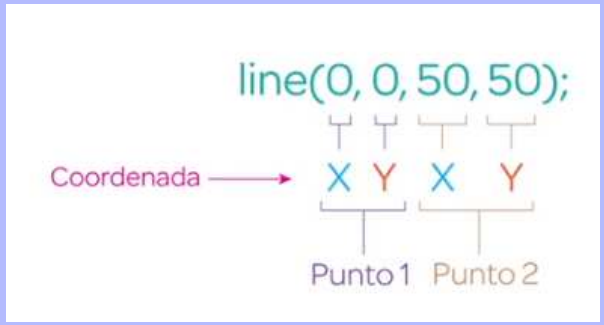
EL SIZE

Otra función muy importante es `size()`, que permite definir el tamaño de la ventana donde vamos a desplegar todos los elementos visuales generados por el programa. Esta función generalmente se ubica en la parte superior de los programas. Los parámetros utilizados por la función son el número de pixeles correspondientes al ancho y el alto de la ventana.

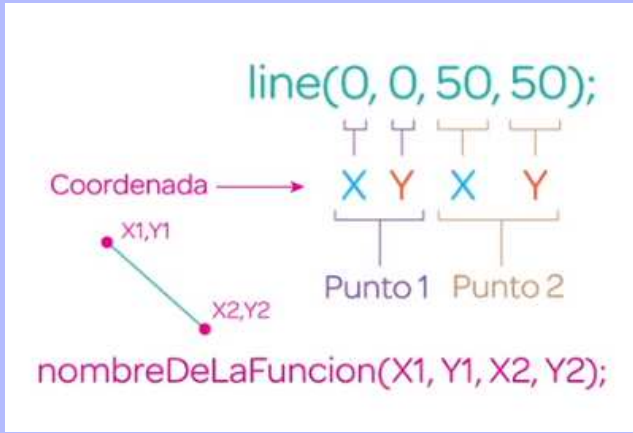


GRÁFICAS BÁSICAS O PRIMITIVAS

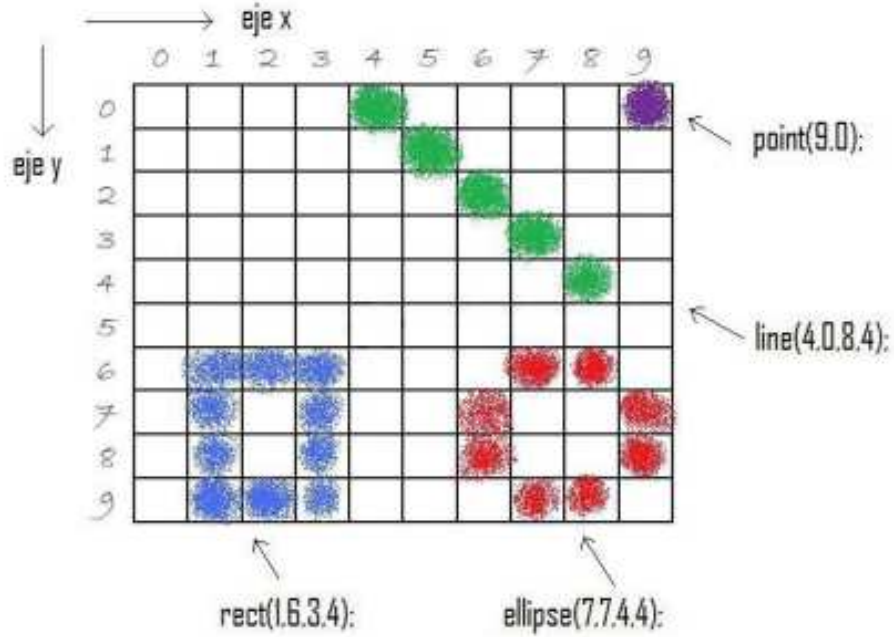
line (0,0,50,50) - line (x, y, x2, y2)



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



Práctica: línea horizontal, vertical, diagonal

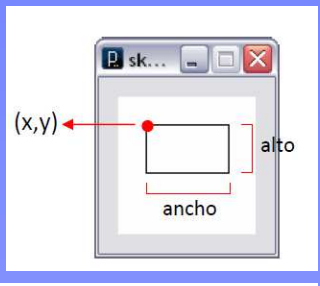


Al analizar cómo se pintaron las formas básicas de manera automática, podemos entonces identificar dos elementos que conforman la instrucción:

```
line(4,0,8,4);
```

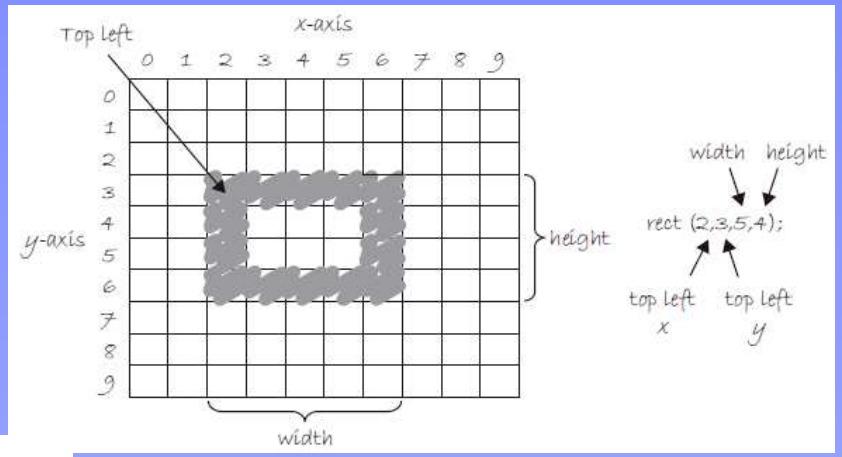
nombre de la función — parámetros

RECT (rectángulo)

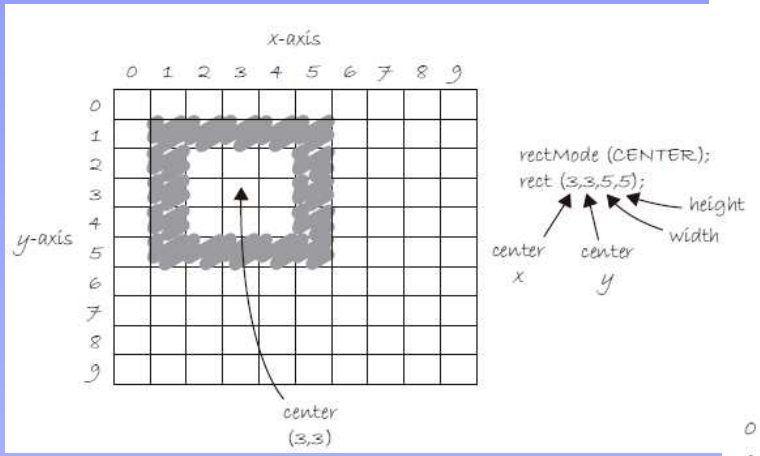


Rect (20,20,60,35) = rect (x,y,ancho,alto)

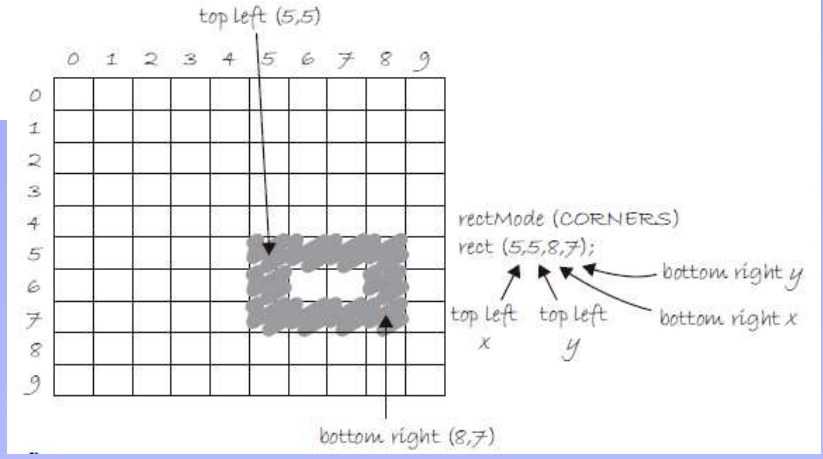
x- posición inicial en el eje X
y- posición inicial en el eje Y
Ancho (width) del rectángulo
Alto (height) del rectángulo



rectMode(CENTER);



rectMode(CORNERS)



RECT (rectángulo)

Si quisiéramos pintar un rectángulo en la coordenada (30, 30) con un ancho de 50 y una altura de 20, la sintaxis sería la siguiente:

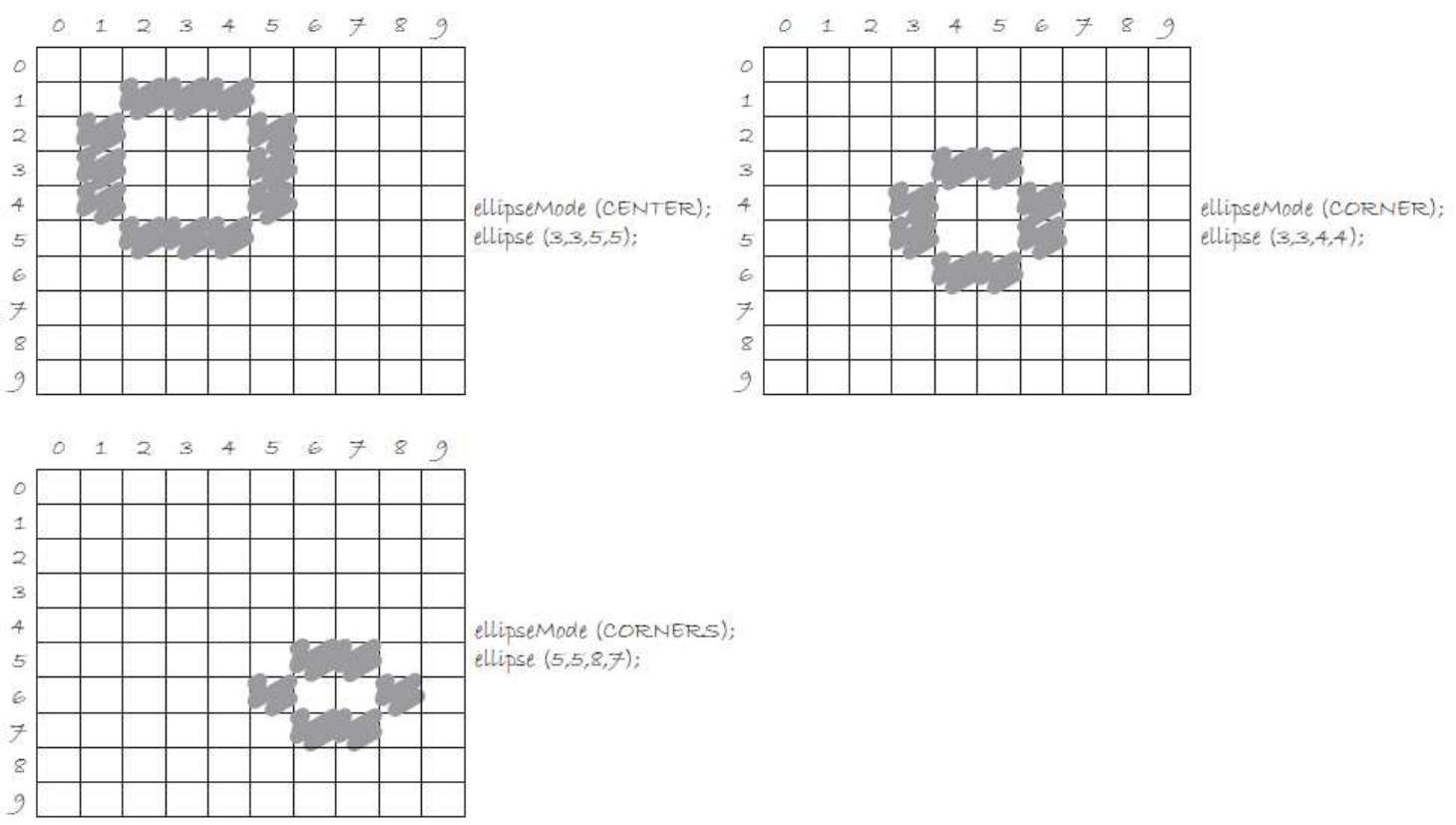
rect(30,30,50,20);
x- posición inicial en el eje X
y- posición inicial en el eje Y
Ancho (width) del rectángulo
Alto (height) del rectángulo

rectMode(CENTER); Cuando dibujamos el rectángulo utilizando las coordenadas (x, y) como punto central y definimos el ancho y el alto, estamos utilizando el modo CENTER.

rectMode(CORNERS); Cuando dibujamos el rectángulo utilizando dos puntos; las coordenadas (x, y) del primer punto y las coordenadas (x, y) del segundo punto, estamos utilizando el modo CORNERS.

ELLIPSE (elipse)

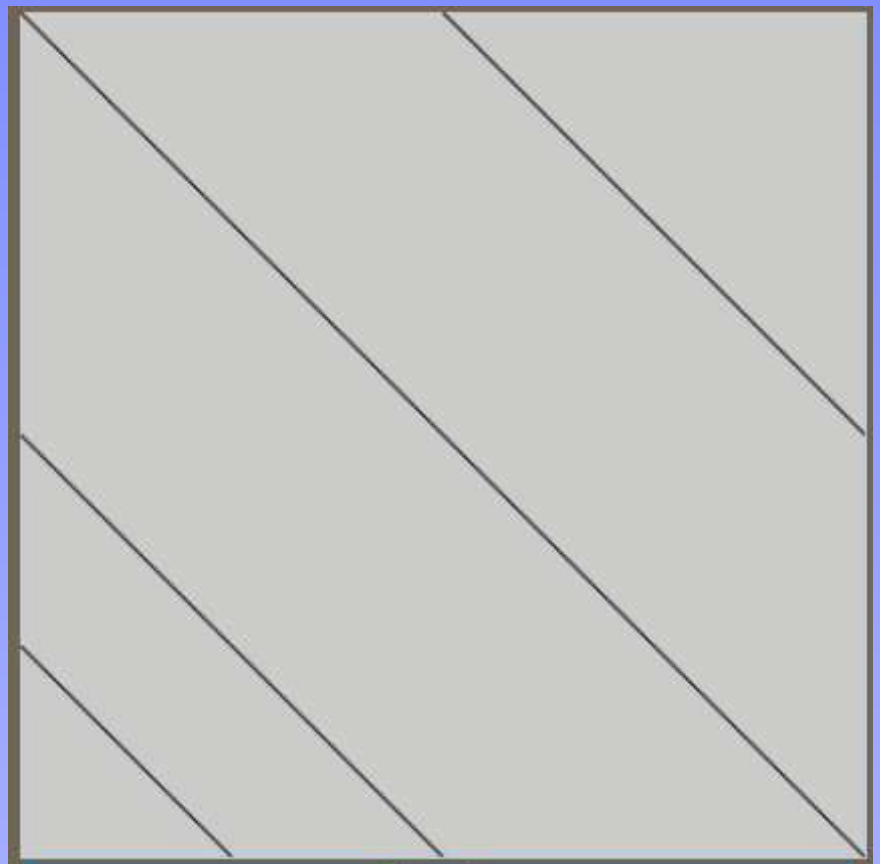
El funcionamiento de la instrucción elipse (`ellipse`) es muy similar a la del rectángulo (`rect`) antes vista. Para dibujar cualquier elipse, se sigue la técnica del Rectángulo Mínimo de Delimitación (MBR). Esta técnica se refiere a que cualquier elipse puede ser enmarcada siempre por un rectángulo mínimo. Teniendo esto en cuenta, debemos entonces verificar las coordenadas de este rectángulo. A la función `ellipse` podemos asociar cuatro parámetros, donde los dos primeros parámetros corresponden a las coordenadas (x, y) del punto central de la elipse y los dos parámetros restantes corresponden al ancho y al alto.



http://processing.org/reference/

Processing - Prácticas de dibujo

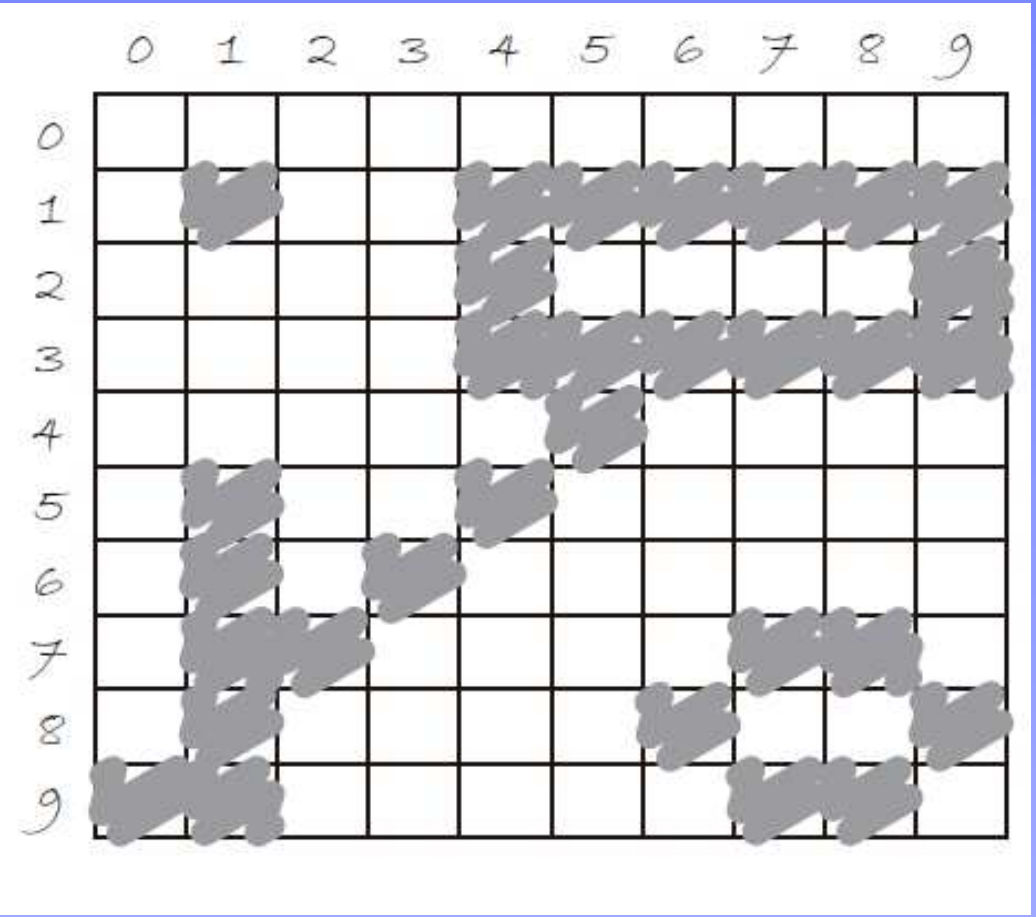
- Todos los canvas tienen un tamaño de 400 pixeles de ancho y 400 pixeles de alto.
- Traten siempre de ingresar los parámetros de las funciones de manera relativa, y no números absolutos.
  O sea, no ingresen "200", sino "width/2"



Ejemplo

```
size(400, 400);
line(0, 0, width, height);                  — Linea del Medio
line(width / 2, 0, width, height /2);       — Linea Superior
line(0, height / 2, width / 2, height);     — Linea Inferior
line(0, height - (height / 4), width / 4, height);  — Linea más Corta
```

Traten de pensar de esta manera:
*Para llegar a tal número.. ¿A qué número le tengo que restar o sumar cuáles otros números?



¿Cúal sería el código escrito en processing?

El punto (*point*), la línea *(line)*, la elipse *(ellipse)* y el rectángulo *(rect)* no son las únicas formas utilizadas por Processing, también existen las funciones de triángulo (triangle), cuadrilátero *(quad)* el arco *(arc)*. A continuación, se muestra su funcionamiento.

EL TRIÁNGULO: conecta tres puntos.

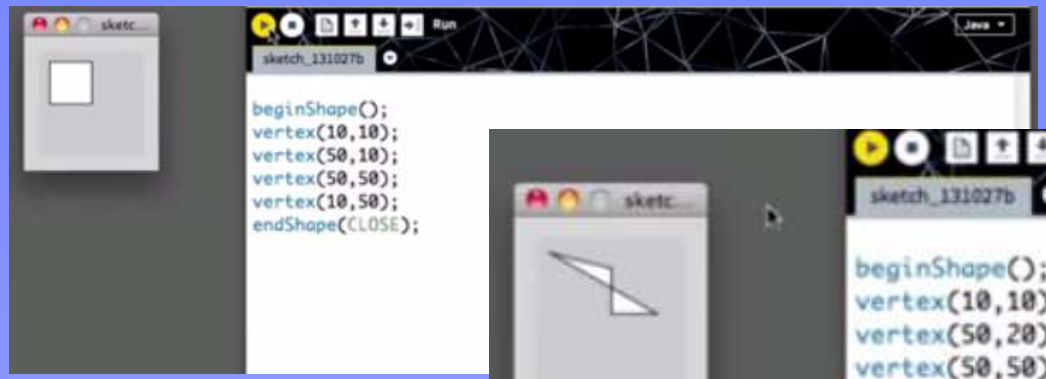
```
triangle (0,90,45,45,70,70);
```

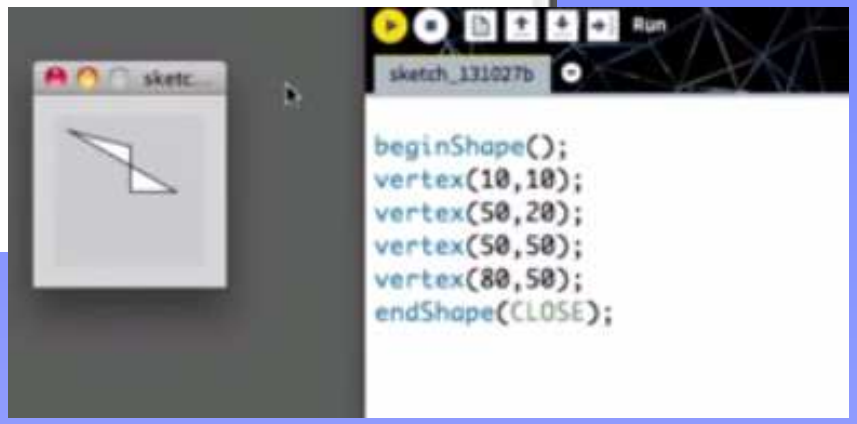
El cuadrilátero

El cuadrilátero es similar a la función *rect()*, solo que ahora debemos de dar un total de cuatro coordenadas diferentes, lo que provocará que los ángulos no necesariamente sean de 45 grados, esto quiere decir que el número de parámetros será de 8.

```
quad(90,60,130,45,180,90,150,85);
```

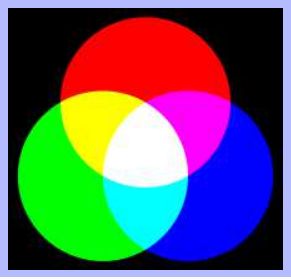
OTRAS FORMAS COMPLEJAS





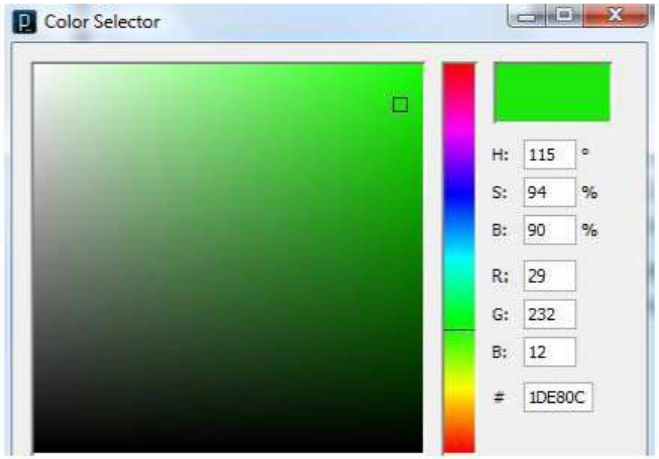
¿qué pasa si eliminamos el CLOSE?

EL COLOR



El color, en el mundo digital, es generalmente construido al mezclar tres colores primarios (rojo, verde, azul), en inglés RGB (Red, Green, Blue). Processing no es la excepción. Para generar algún color, tenemos que formar una combinación de estas tonalidades.

Processing cuenta con una herramienta llamada ***color selector***, la cual puede ser desplegada a partir del menú *Tools*, como se muestra a continuación:

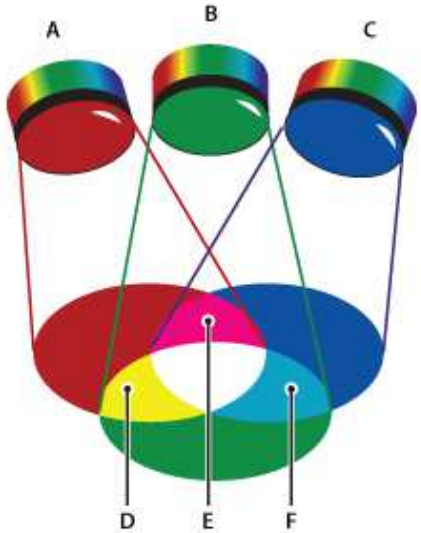


RGB, HSB (Hue, Saturation y Brightness), CODIFICACIÓN HEXADESIMAL

Modelo RGB

Un amplio porcentaje del espectro visible se puede representar combinando luz roja, verde y azul (RGB) en distintas proporciones e intensidades. Estos tres colores se conocen como *primarios aditivos*. Al unir las luces roja, verde y azul, se obtiene una luz blanca. Cuando se solapan dos colores, se crean el cian, el magenta y el amarillo.

Los colores primarios aditivos se usan para iluminación, vídeo y monitores. El monitor, por ejemplo, crea color mediante la emisión de luz a través de fósforos de color rojo, verde y azul.



Colores aditivos (RGB)

A. *Rojo* ***B.*** *Verde* ***C.*** *Azul* ***D.*** *Amarillo* ***E.*** *Magenta* ***F.*** *Cian*

Modelo HSB

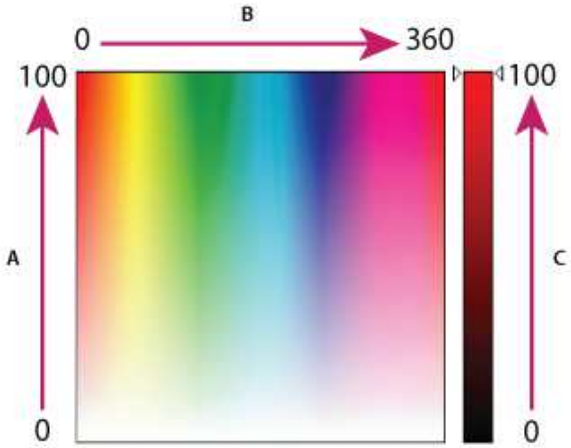
El modelo HSB se basa en la percepción humana del color y describe tres características fundamentales del color:

Tono Color reflejado o transmitido a través de un objeto. Se mide como una posición en la rueda de colores estándar y se expresa en grados, entre 0° y 360°. Normalmente, el tono se identifica por el nombre del color, como rojo, naranja o verde.

Saturación Fuerza o pureza del color. La saturación, que a veces se denomina *cromatismo*, representa la cantidad de gris que existe en proporción al tono y se mide como porcentaje comprendido entre 0 (gris) y 100 (saturación completa). En la rueda de colores estándar, la saturación aumenta a medida que nos aproximamos al borde de la misma y disminuye a medida que nos acercamos al centro.

Brillo Luminosidad u oscuridad relativa del color. Se suele medir como un porcentaje comprendido entre 0 (negro) y 100 (blanco).

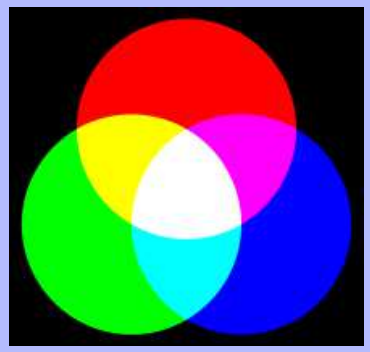
Aunque puede usar el modelo HSB en Photoshop Elements para definir un color en el cuadro de diálogo Selector de color, no puede usar el modo HSB para crear ni editar imágenes.



Vista HSB del Selector de color de Adobe
***A.** Saturación* ***B.** Tono* ***C.** Brillo*

EL COLOR

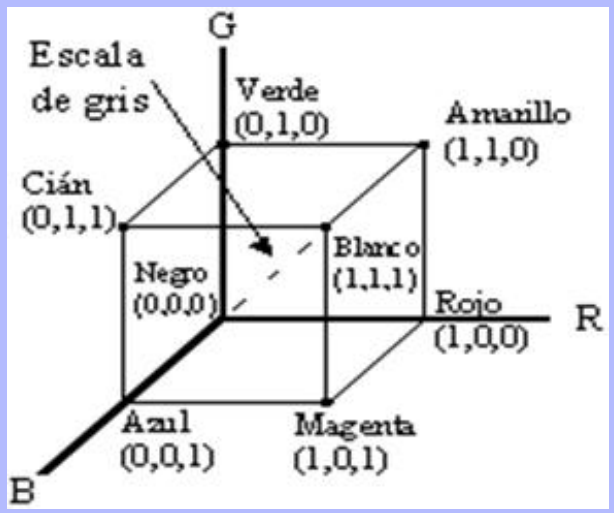
- Se representa por 3 canales = RGB (rojo, verde, azul).
- Escala que va de 0 a 255.



- Canal alfa o transparencia.

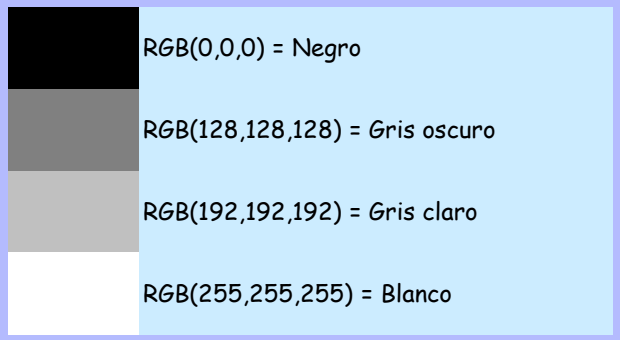
COLORES PRIMARIOS

RBG(255,0,0) = Rojo claro (o rojo brillante, o rojo intenso)
RBG(0,255,0) = Verde claro (o verde brillante, o verde intenso)
RBG(0,0,255) = Azul claro (o azul brillante, o azul intenso)

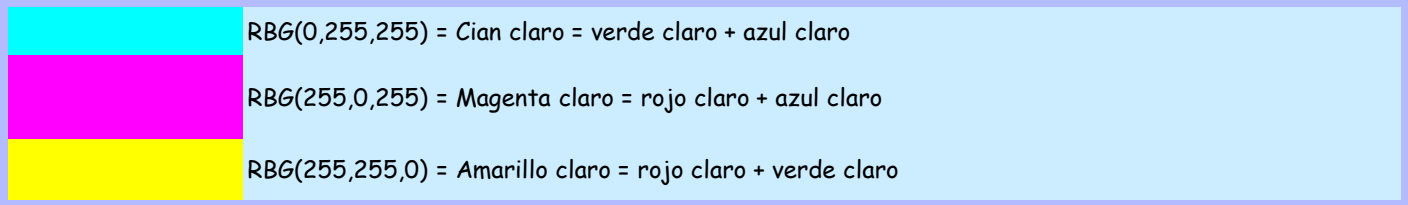


EL COLOR

TONOS GRISES: se representan con tres valores iguales para sus 3 componentes rojo, verde y azul.



COLORES SECUNDARIOS:

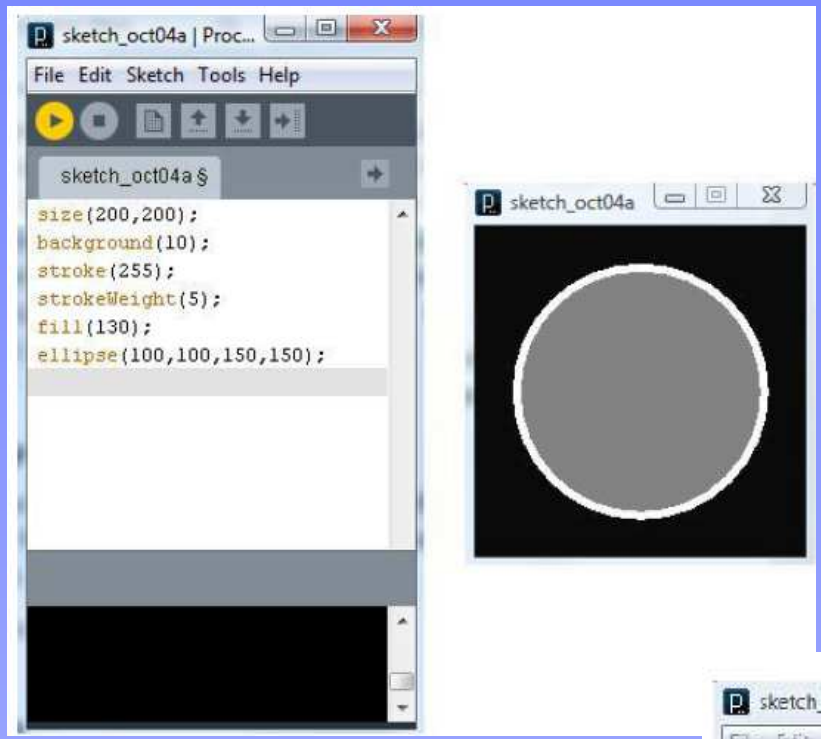


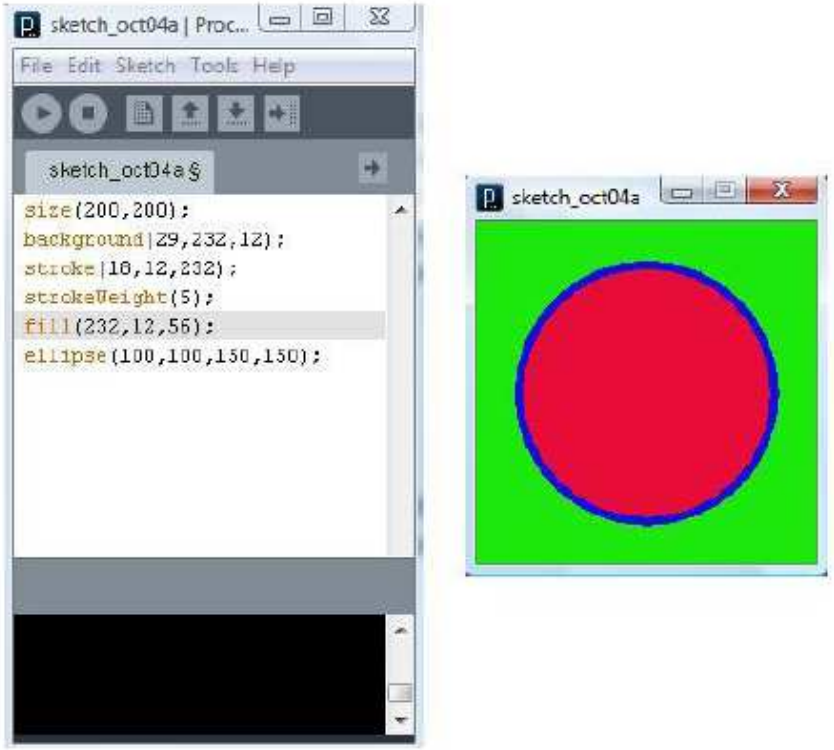
TRES FUNCIONES DEL COLOR

* fill(); relleno de formas - noFill();
* background(); fondo
* stroke(); línea de contorno. - noStroke();
* strokeWeight(); grosor de línea

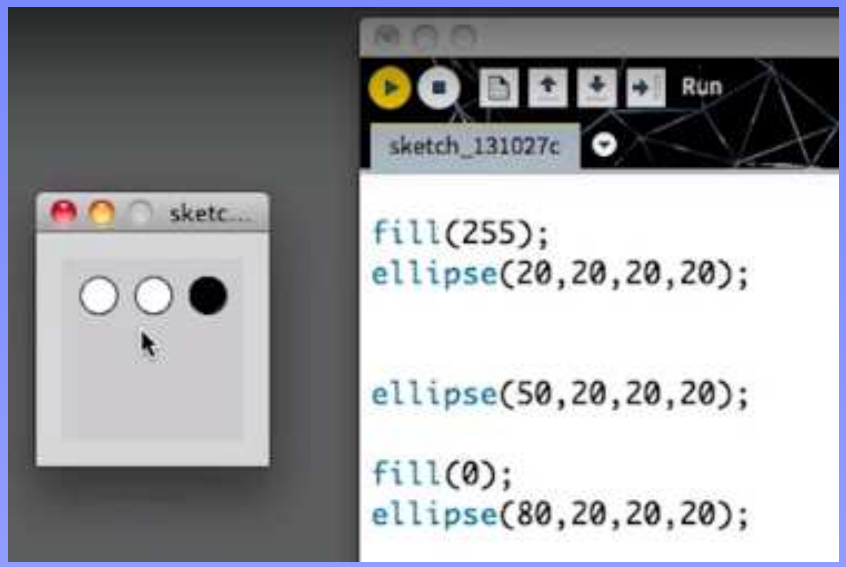
**Se indica antes del dibujo a realizar

Color y grises

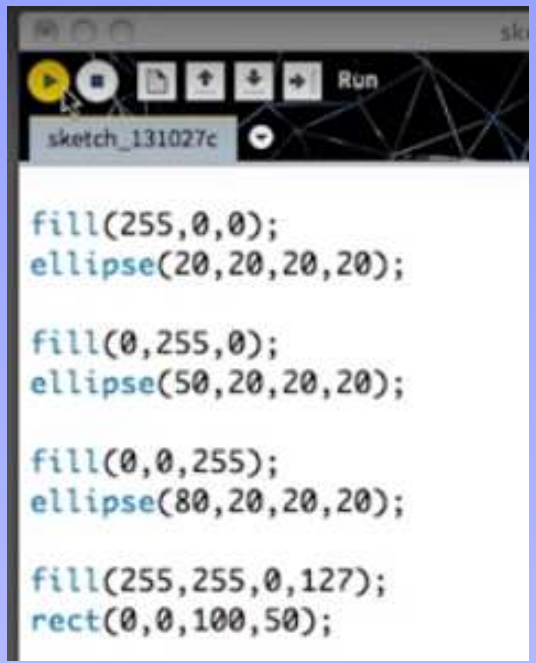

sketch_oct04a | Proc...
File Edit Sketch Tools Help
sketch_oct04a §
size(200,200);
background(10);
stroke(255);
strokeWeight(5);
fill(130);
ellipse(100,100,150,150);
sketch_oct04a


sketch_oct04a | Proc...
File Edit Sketch Tools Help
sketch_oct04a §
size(200,200);
background(29,232,12);
stroke(16,12,232);
strokeWeight(5);
fill(232,12,56);
ellipse(100,100,150,150);
sketch_oct04a

- ORDEN DEL COLOR



- 4º parámetro de TRANSPARENCIA:



* últimos parámetros

La transparencia

Además de definir cualquier combinación de color, Processing permite definir el nivel de transparencia de cada uno de los colores. Este nivel va del 0 (color completamente sólido) al 255 (color completamente transparente). Este nivel de transparencia, muchas veces llamado nivel *alpha, se define como un cuarto parámetro dentro de las funciones* `background()`, `fill()`, y `stroke()`.

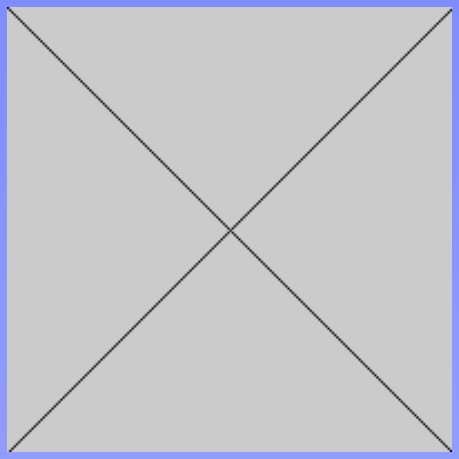
PRÁCTICAS

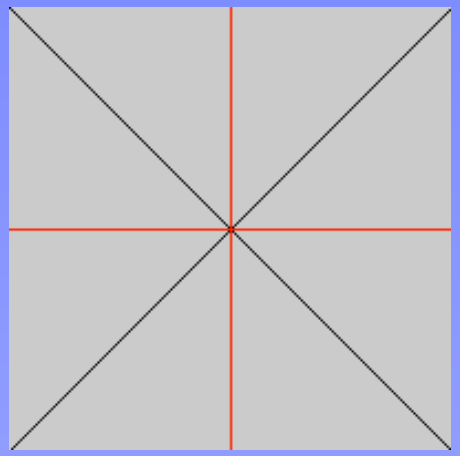
```
size(300,300);
background(255);
fill(127,255,0);
stroke(255,0,0);
rect(50,50,200,200);
```

```
size(300,300);
background(255);
fill(0,0,255);
stroke(255,0,0);
ellipse(200,100,66,99);
```

* Superponer y transparencia

PRÁCTICAS





Actividad Integradora

Realizar un dibujo libre utilizando las primitivas básicas y el color.