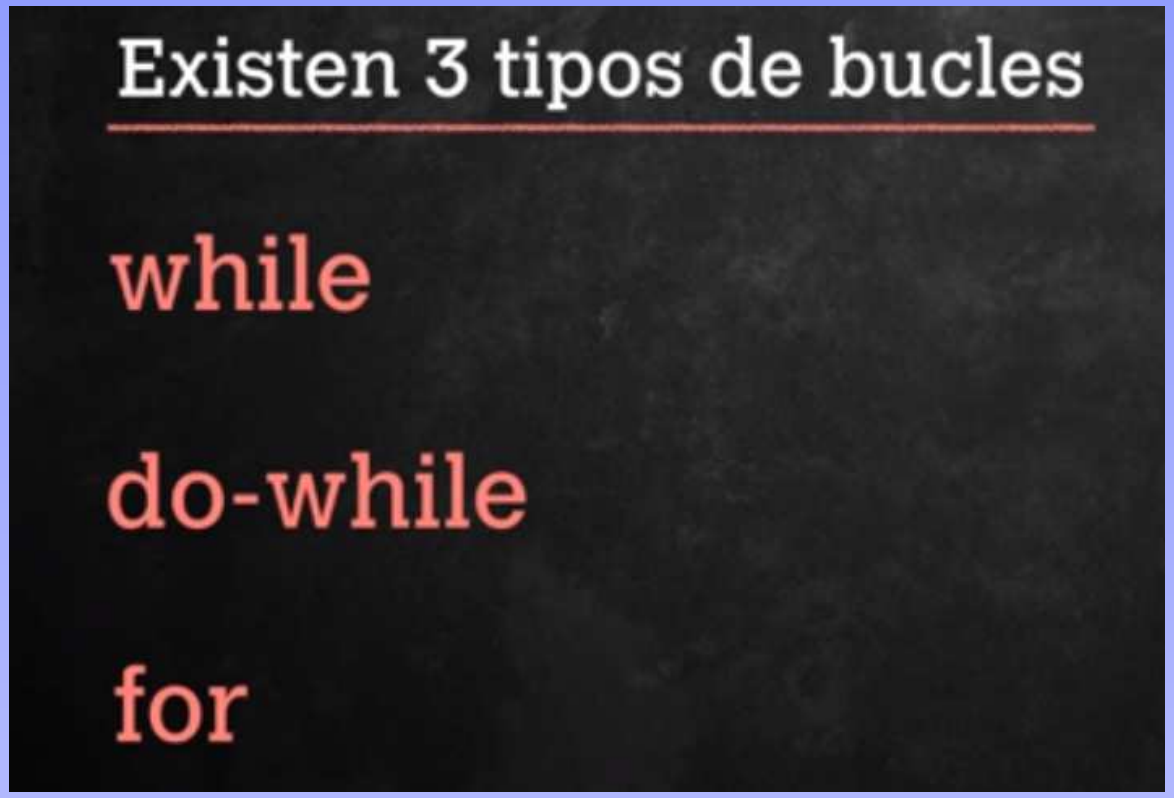
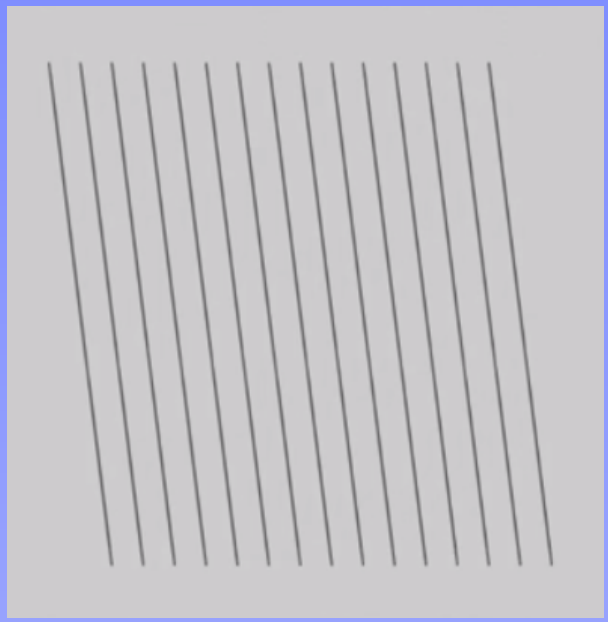
BUCLES

Realizan de forma rápida cálculos repetitivos dentro de código.

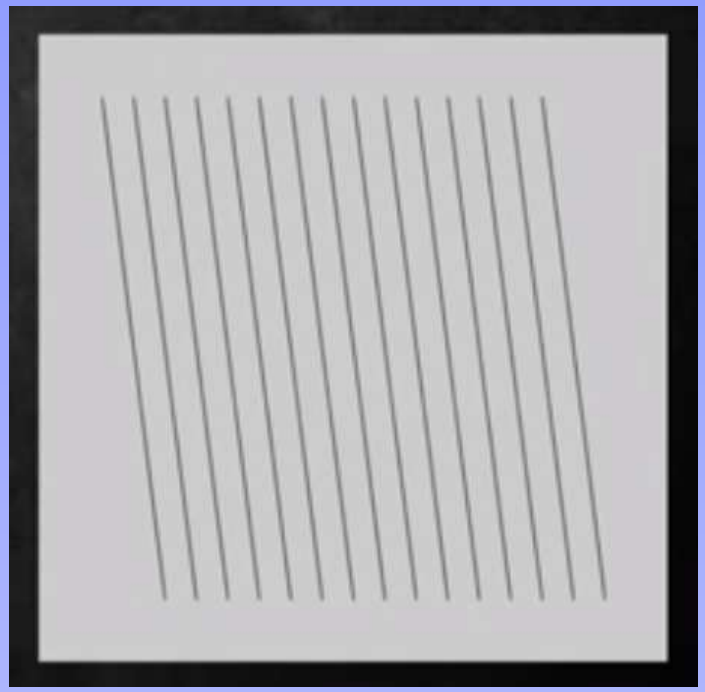
*Las estructuras repetitivas son las encargadas de producir iteraciones,o estructuras iterativas que sirven para compactar grandes líneas de código.



```
size(500, 500);

line(50, 50, 100, 450);
line(75, 50, 125, 450);
line(100, 50, 150, 450);
line(125, 50, 175, 450);
line(150, 50, 200, 450);
line(175, 50, 225, 450);
line(200, 50, 250, 450);
line(225, 50, 275, 450);
line(250, 50, 300, 450);
line(275, 50, 325, 450);
line(300, 50, 350, 450);
line(325, 50, 375, 450);
line(350, 50, 400, 450);
line(375, 50, 425, 450);
line(400, 50, 450, 450);
```



```
size(500, 500);

int x = 50;
while(x <= 400){
  line(x, 50, x+50, 450);
  x = x + 25;
}
```

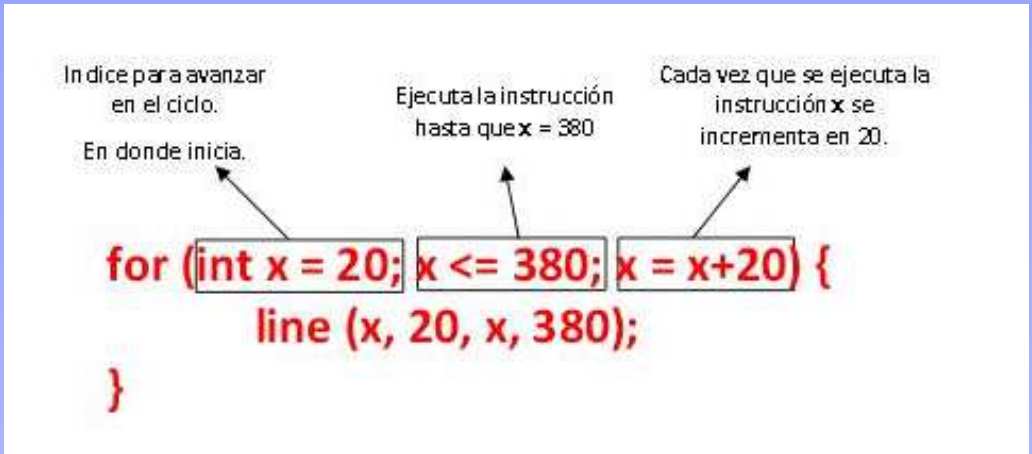


Código Original

```
size(200, 200);
line(20, 20, 20, 180);
line(30, 20, 30, 180);
line(40, 20, 40, 180);
line(50, 20, 50, 180);
line(60, 20, 60, 180);
line(70, 20, 70, 180);
line(80, 20, 80, 180);
line(90, 20, 90, 180);
line(100, 20, 100, 180);
line(110, 20, 110, 180);
line(120, 20, 120, 180);
line(130, 20, 130, 180);
line(140, 20, 140, 180);
```

Código utilizando un FOR

```
size(200, 200);
for (int i = 20; i < 150; i += 10) {
      line(i, 20, i, 180);
}
```

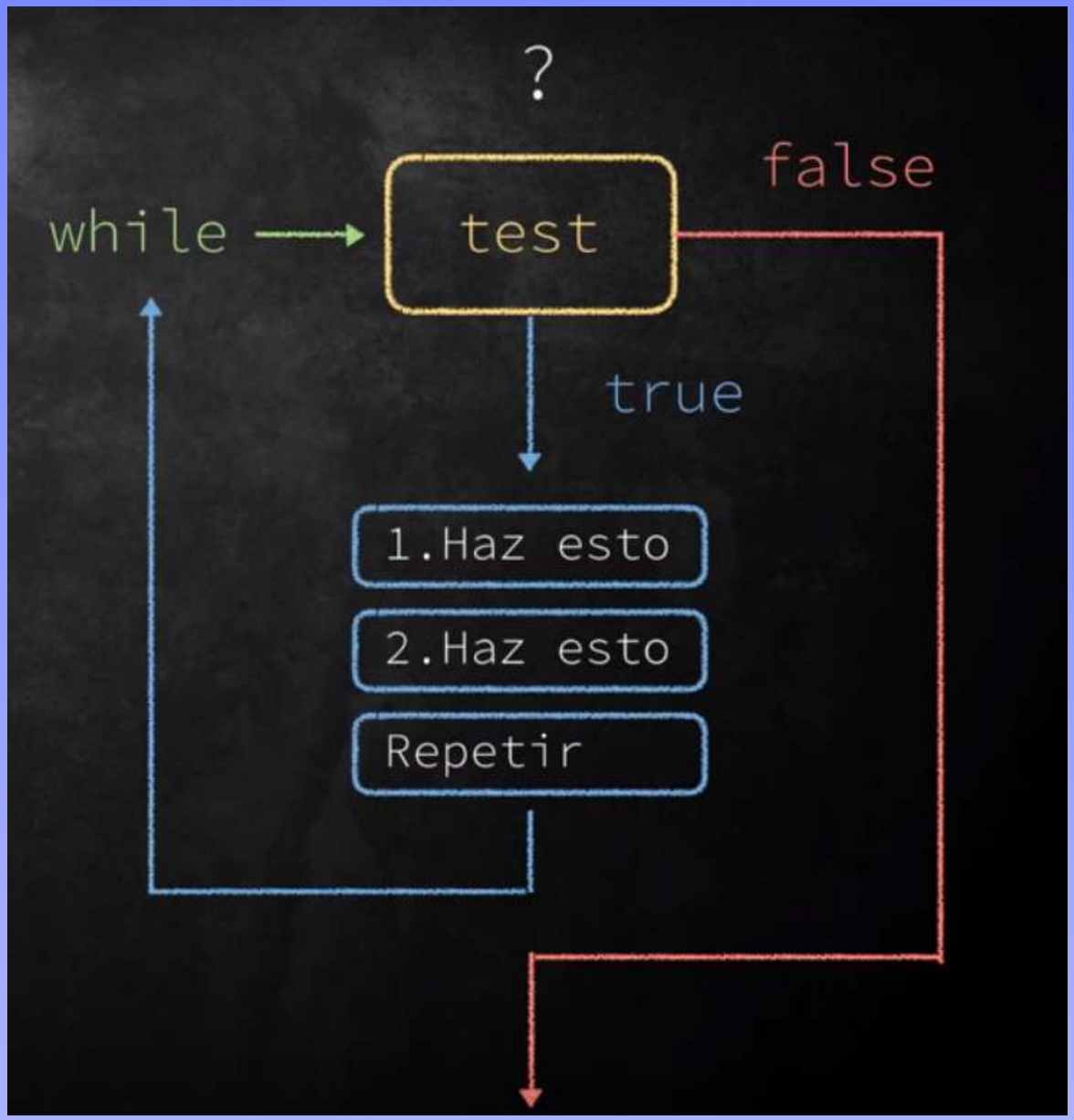


```
while (condición de ejecución){
    // Bloque de código a ejecutar
}

for (inicialización; condición de ejecución; incremento){
    // Bloque de código a ejecutar
}
```

Inicialización- establece un valor inicial para aquellas variables que participan en la condición.

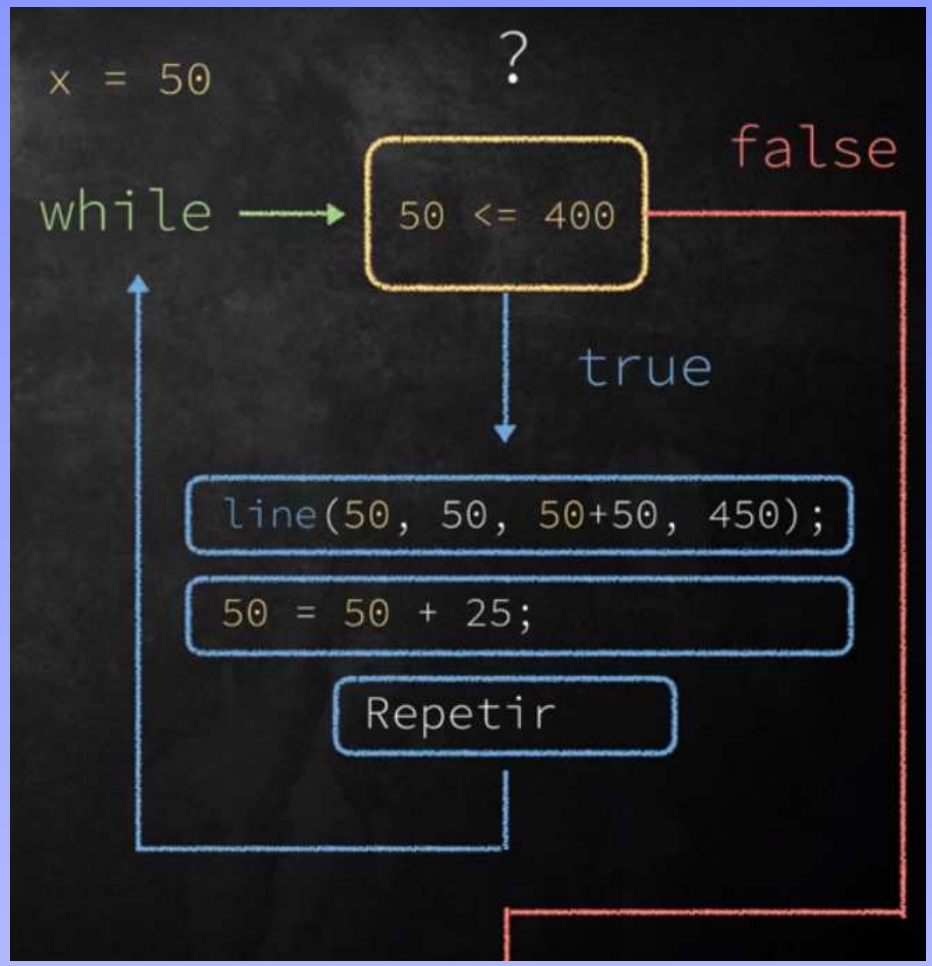
Condición- la expresión con la que se evalúa como verdadero o falso, y la que decide si el cuerpo de la estructura se repite o no.

Incremento o actualización- es la instrucción que hace cambiar el valor de las variables que forman parte de la condición.


?
while
test
false
true
1.Haz esto
2.Haz esto
Repetir

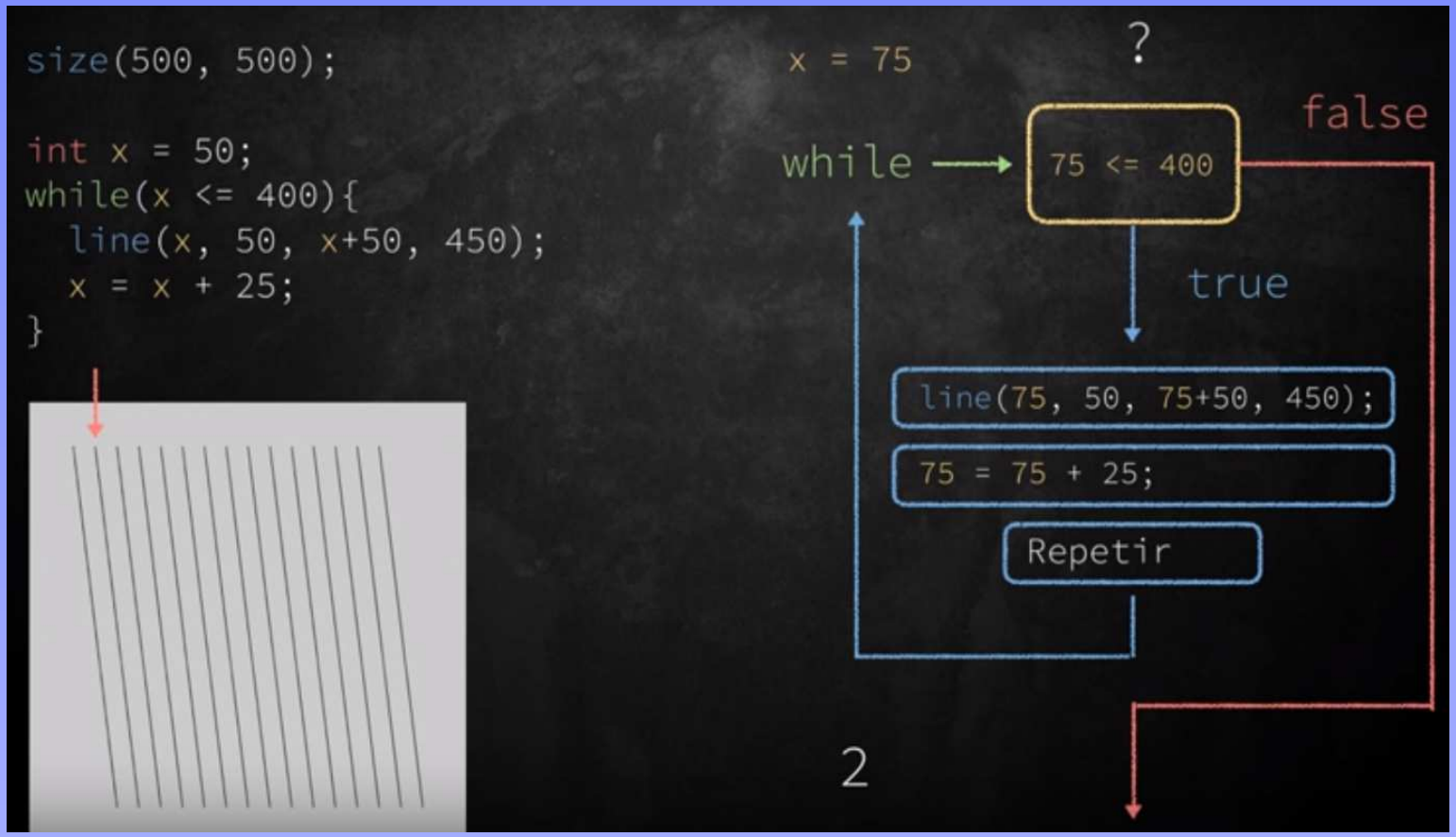
```
size(500, 500);

int x = 50;
while(x <= 400){
  line(x, 50, x+50, 450);
  x = x + 25;
}
```

Al llegar al bucle pregunta: ¿Es 50 menor que 400?



Añade 25 a nuestra variable X, y vuelve al bucle while.

Cuando repite la lectura del bloque while…



sigue siendo verdadero lo que expresa en el bucle de while 75<=400

1. ¿Cuál es la condición inicial para el bucle?

El primer rectángulo se encuentra en la localización "y" 5.

```
int y = 5;
```

2. ¿Cuándo debe parar el bucle?

Dado que queremos visualizar rectángulos hasta la parte inferior de la ventana, el bucle debe parar cuando "y" sea mayor que la altura (height).

```
while(y < height){
  //loop!!
}
```

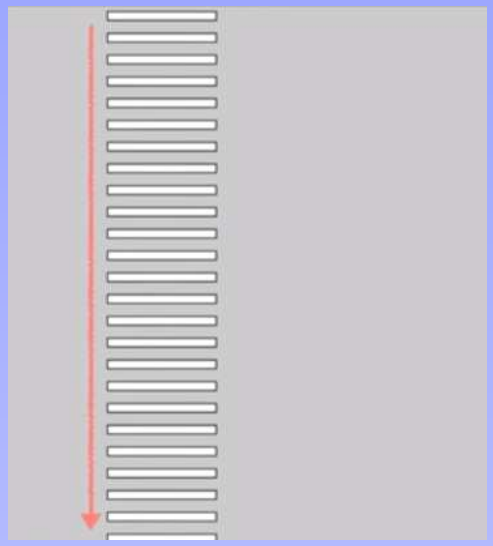
3. ¿Cuál será el funcionamiento bucle?

Cada paso por el bucle queremos dibujar un rectángulo nuevo debajo del anterior. Hacemos esto con la función rect() e incrementando y.

```
rect(100, y, 100, 8);
y = y + 20;
```

Ejemplo:

```
size(500, 500);

int y = 5;
while(y < height){
  rect(100, y, 100, 8);
  y = y + 20;
}
```



COMPARATIVA BUCLES:

Bucles while vs do while

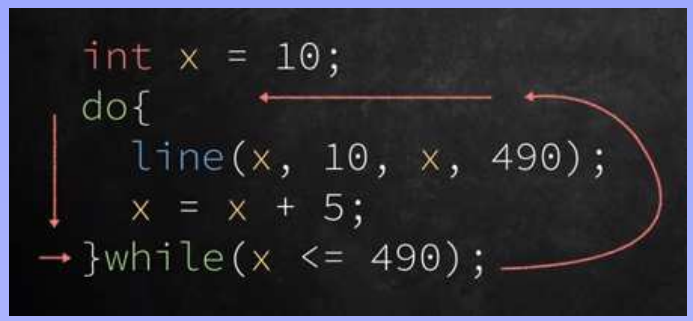
do while

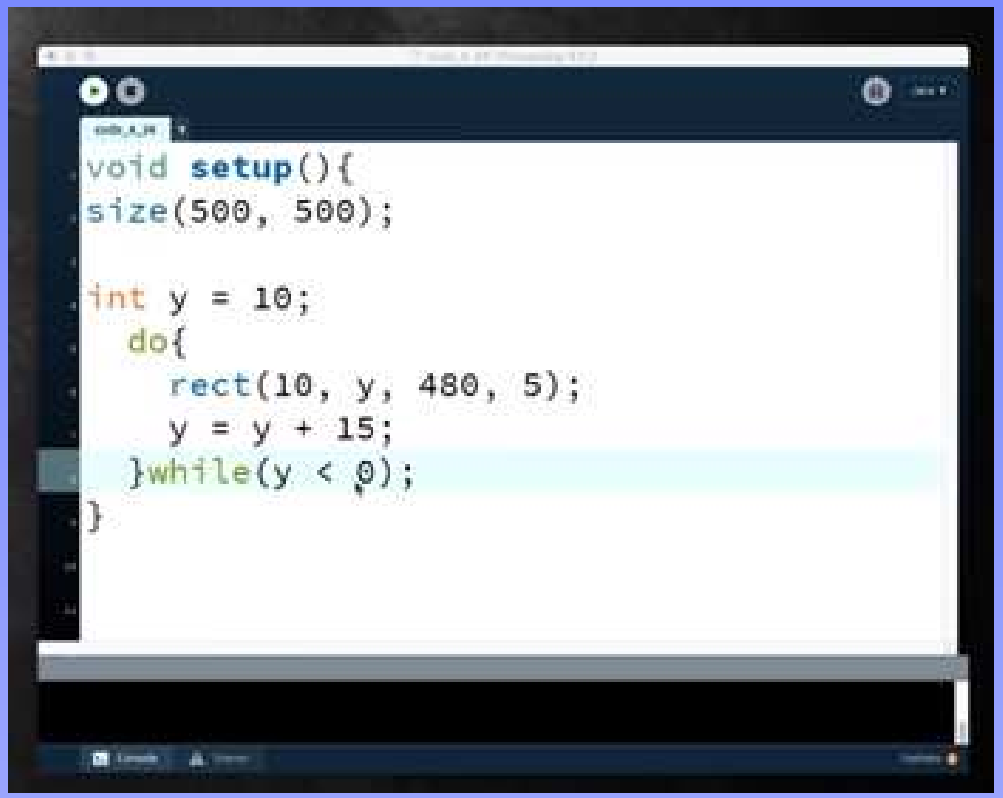
```
size(500, 500);

int x = 10;
do{
  line(x, 10, x, 490);
  x = x + 5;
}while(x <= 490);
```

while

```
size(500, 500);

int x = 10;
while(x <= 490){
  line(x, 10, x, 490);
  x = x + 5;
}
```

-Variante del bucle while
-Garantiza que al menos una vez.
-Contrucción invertida





```
void setup(){
size(500, 500);

int y = 10;
  do{
    rect(10, y, 480, 5);
    y = y + 15;
  }while(y < 0);
}
```

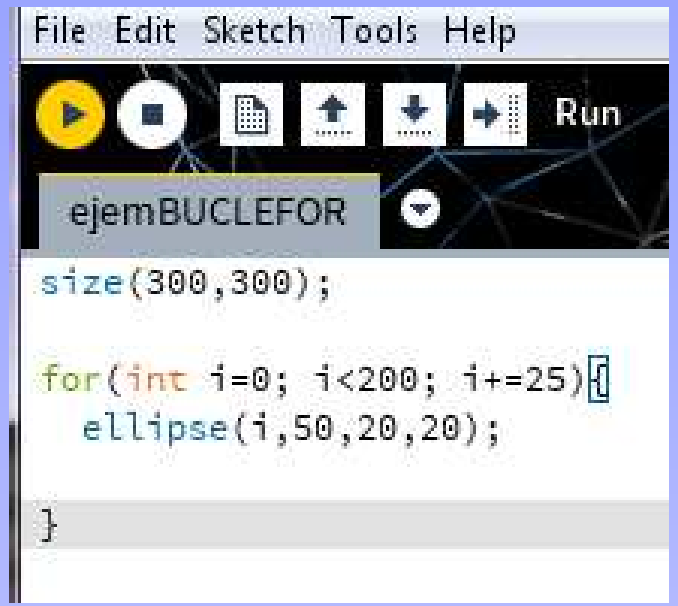
El valor “y”, nunca va a ser menor que 0, Pero se dibuja un rectángulo por la posición de la figura antes de la condición while.

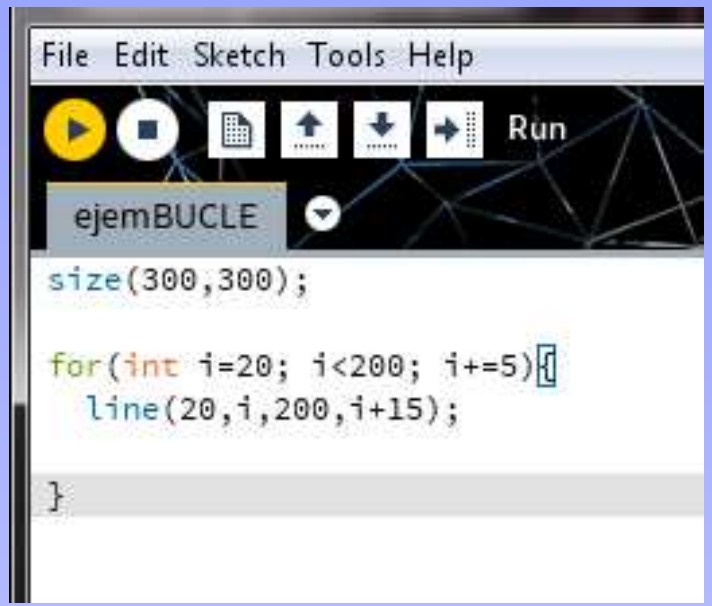
Bucles for

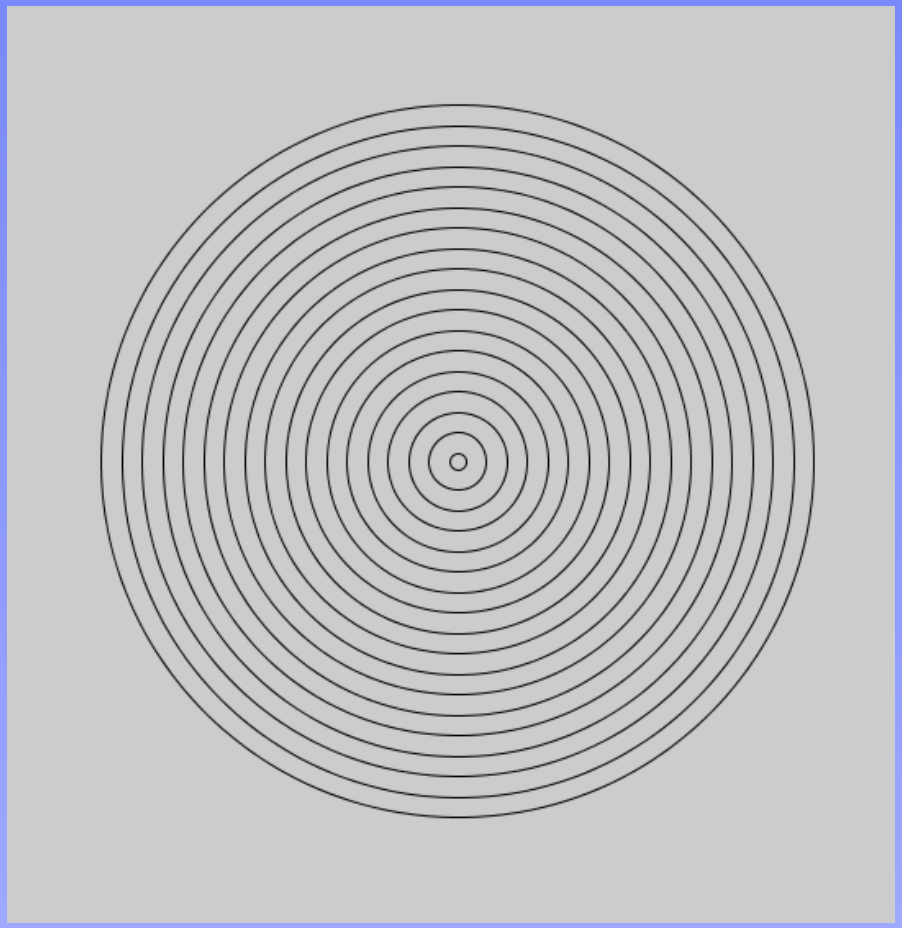
```
for (inicialización, condición, incremento) {
// dentro de las llaves es el conjunto de instrucciones que se ejecutan //
}
```

En el paréntesis asociado con la estructura se encierran tres declaraciones: inicio, prueba y actualización.

Las declaraciones dentro del bloque se ejecutan continuamente mientras la prueba es evaluada como verdadera.







¿Cuál sería el código con ambas estructuras de bucle, FOR y WHILE?

File Edit Sketch Tools Help

sketchBUCLE_CUADRICULA

```
size(100,100);

int posX;


for(posX=0; posX<width; posX+=10){

  line(posX,0,posX,height);
  posX=posX+=2;
}
```

¿Cuál sería el otro bucle “for” para crear una cuadrícula en mi lienzo?

BUCLES ANIDADOS

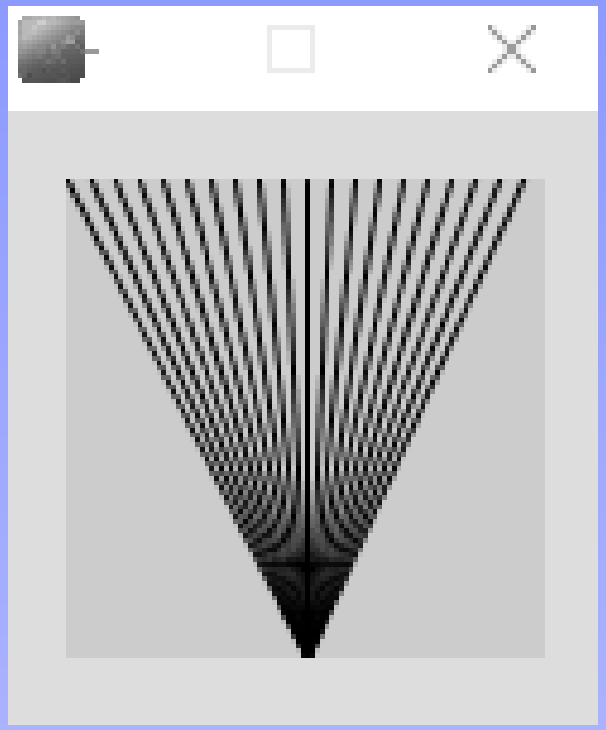
sketchBUCLE_ANIDADOS

```
size(100,100);
strokeWeight(5);


for(int posX=10; posX<width; posX+=7){
  for(int posY=10; posY<height; posY+=7){
  point(posX,posY);

}
}
```

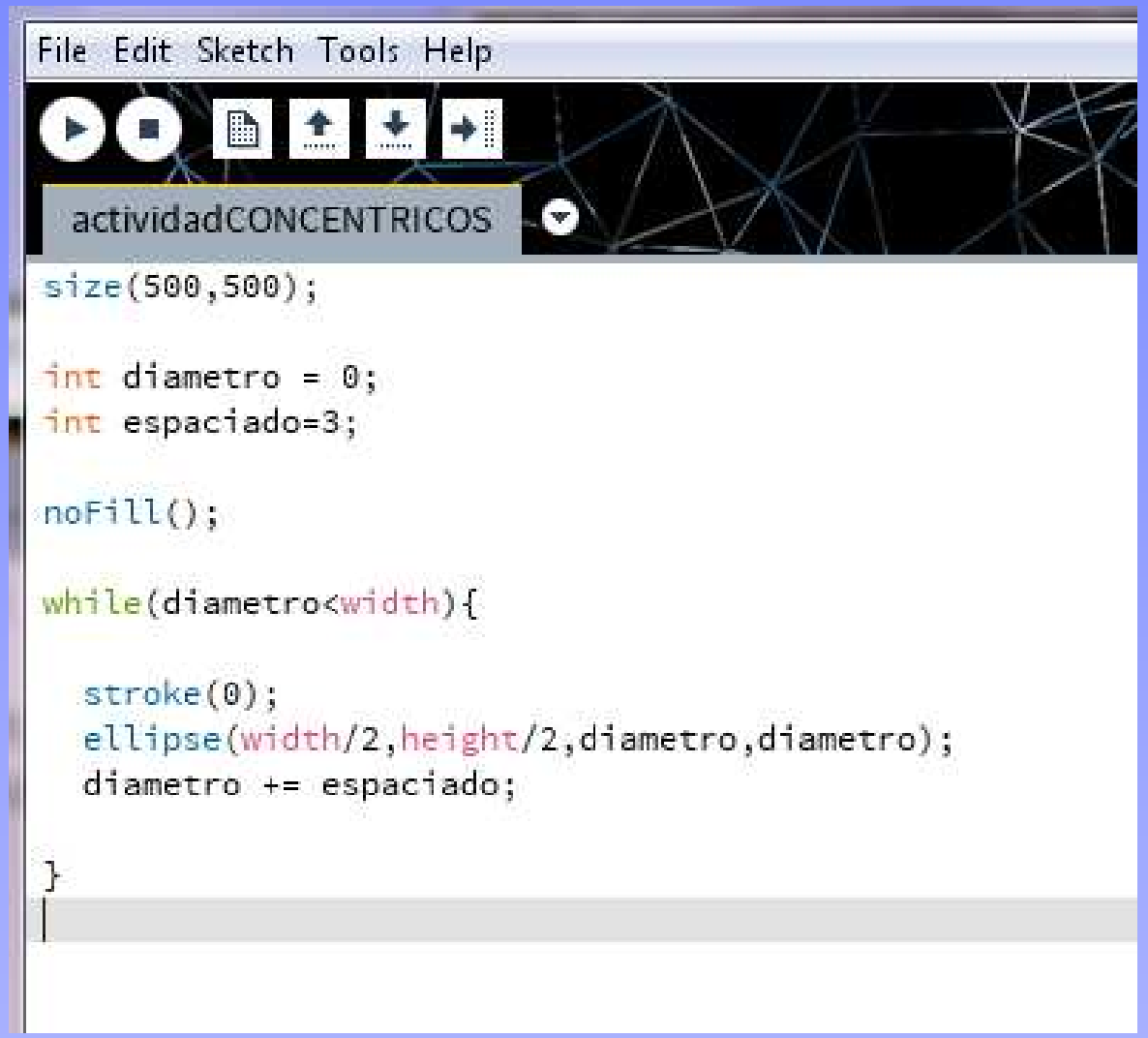
ACTIVIDAD:

Iteración de líneas que describen un abanico partiendo de la mitad del lienzo.

¿Cúal sería el código?



Ejemplo elipse:



**¿Si quiero un color aleatorio en los contornos de las elipses?

ACTIVIDAD:

```
for (inicialización, condición, actualización) {
// Este es el conjunto de instrucciones que se ejecutan   //

}
```

- Repetición de rectángulos con el tipo de bucle FOR

LAS FUNCIONES rotate() y translate()

- Estas funciones no mueven objetos o elementos, sino todo el espacio de dibujo. No movemos o giramos el objeto en sí, sino todo el plano donde se encuentran esos objetos, incluidos sus ejes de coordenadas.

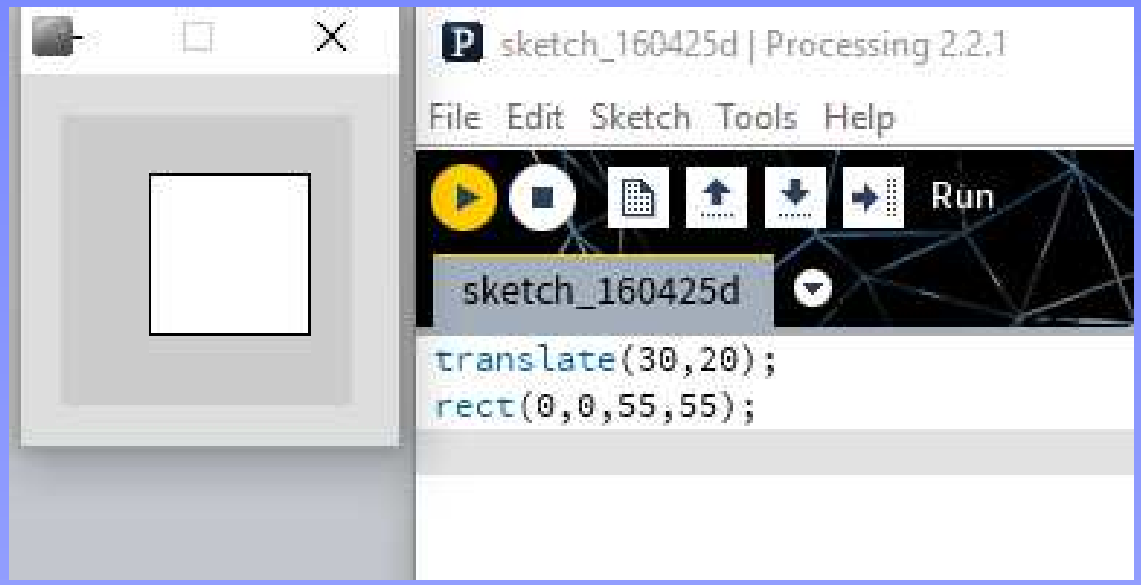
- rotate() gira alrededor del origen de coordenadas. Rota respecto al punto que se dibuja.

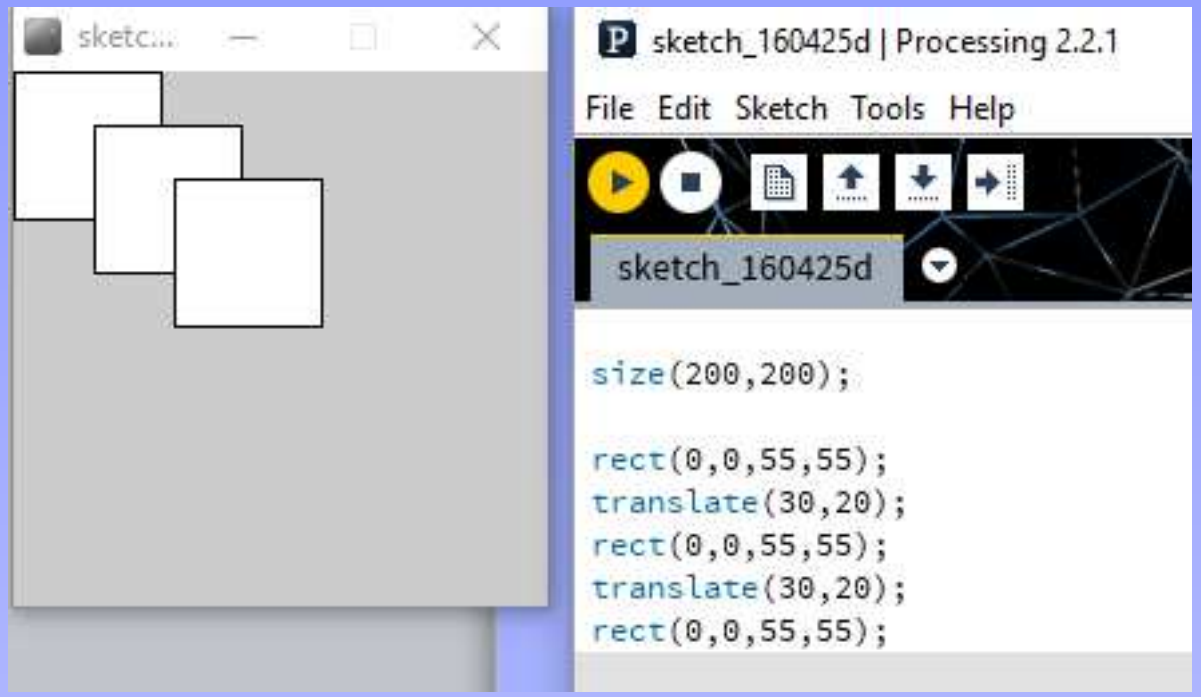
- translate() mueve cada elemento una distancia dada en una dirección también dada. Traslada a partir del centro de la figura en X e Y la cantidad que se indica en el interior de sus paréntesis.
  Esta función toma dos parámetros, x e y.
  Mueve el origen, el vértice superior izquierdo.

```
translate (x, y);
```

- Si quieres aplicar diferentes movimientos a diferentes elementos de una misma aplicación, tienes que resetear el cambio anterior mediante resetMatrix().

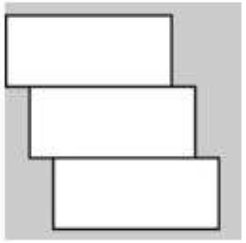
TRANSLATE:





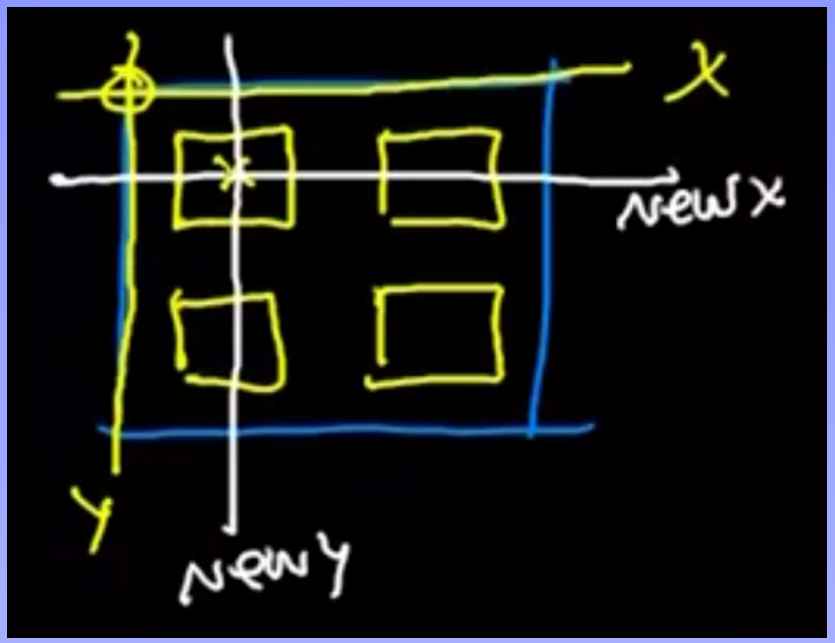
La translación es aditiva.

Hay que tener en cuenta que la función `translate()` es aditiva. Si `translate(10, 30)` se ejecuta dos veces, entonces la posición de base será `(20, 60)`.



```
rect(0, 5, 70, 30);
translate(10, 30);//Mueve 10 pixeles a la derecha y 30 abajo
rect(0, 5, 70, 30)
translate(10, 30);//Repite traslación, por lo tanto
rect(0, 5, 70, 30);//mueve 20 pixeles a la derecha y 60 abajo
```

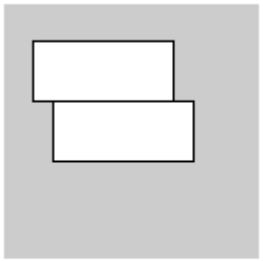
Se modifica el origen de nuestro eje de coordenadas.




```
translate(x, y)
```

Los valores de *x* y de *y* que se ingresan afectan a cualquier figura o forma que se pretenda dibujar después. Si 10 es el parámetro de *x*, y 30 es el parámetro de *y*, un punto dibujado en la posición (0, 5) se mostrará en la posición (10, 35).

En los siguientes ejemplos se puede ver como funciona, el mismo rectángulo dibujado antes y después de la translación:.



```
rect(0, 5, 70, 30);
translate(10, 30); // Transladamos 10 px a la derecha y 30 hacia abajo
rect(0, 5, 70, 30);
```

Trasladamos en el eje X y en el eje Y. ➡ translate (x,y);

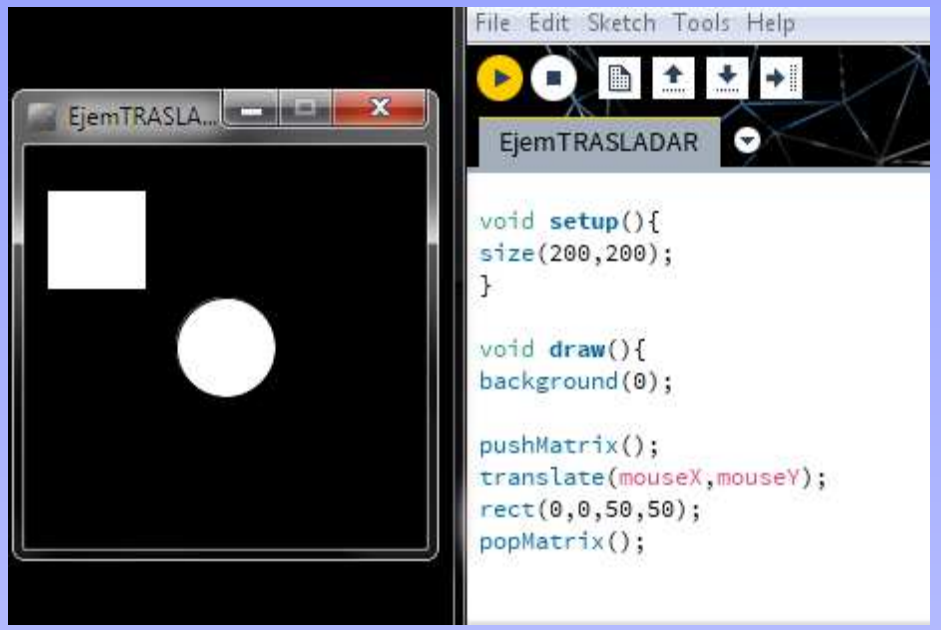
FUNCIONES pushMatrix(); y popMatrix();

Limita las transformaciones a las figuras que se encuentran dentro de ellas

Ejemplo pushMatrix y popMatrix en clase

Bloques setup y draw

Translate con mouseX y mouseY

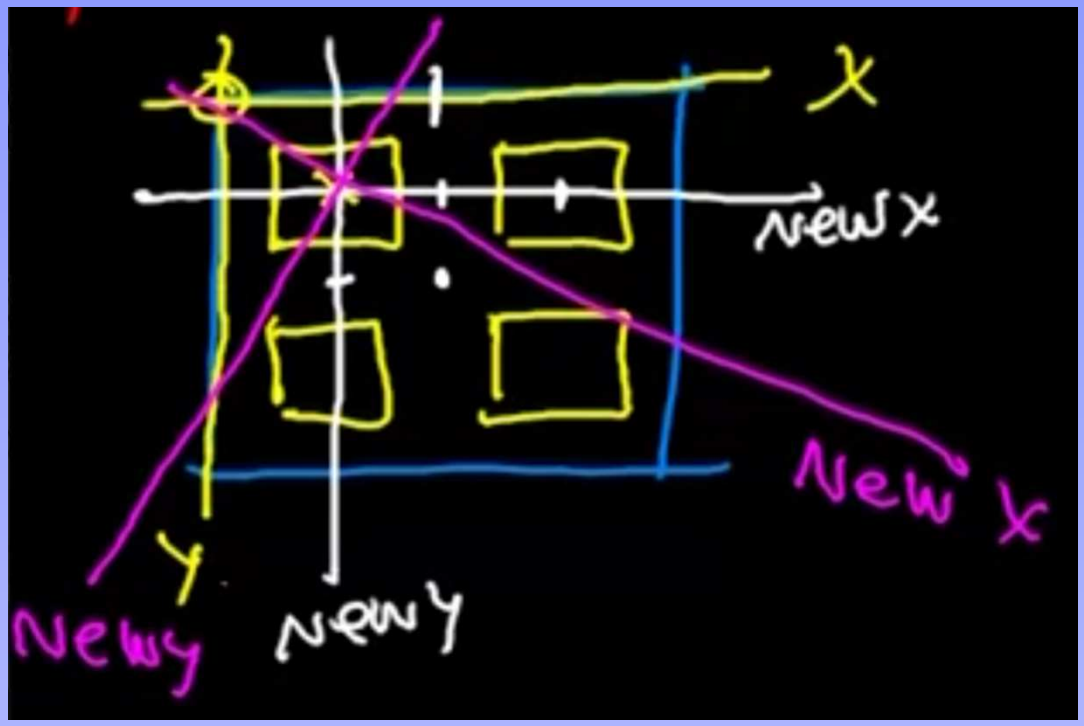


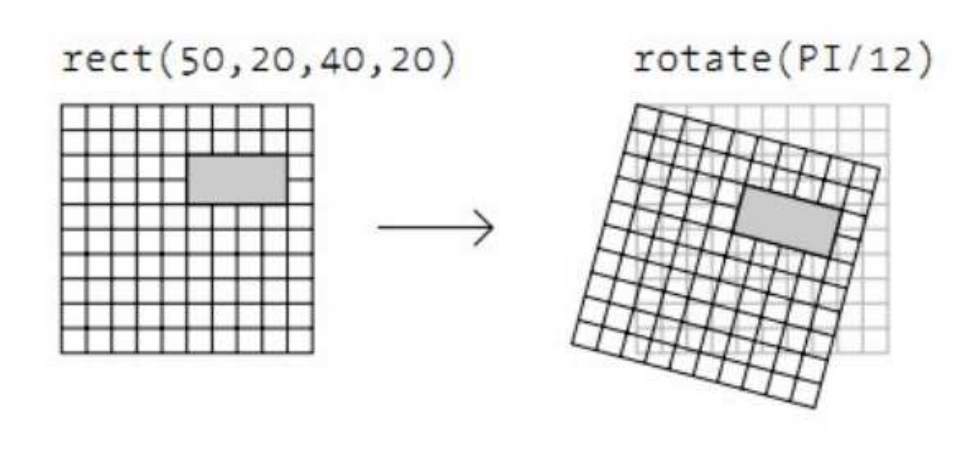
ROTATE: mueve todo el espacio del dibujo

Desplazamiento del eje de coordenadas en la rotación. Recibe sólo un parámetro el cual se modifica con un ángulo.

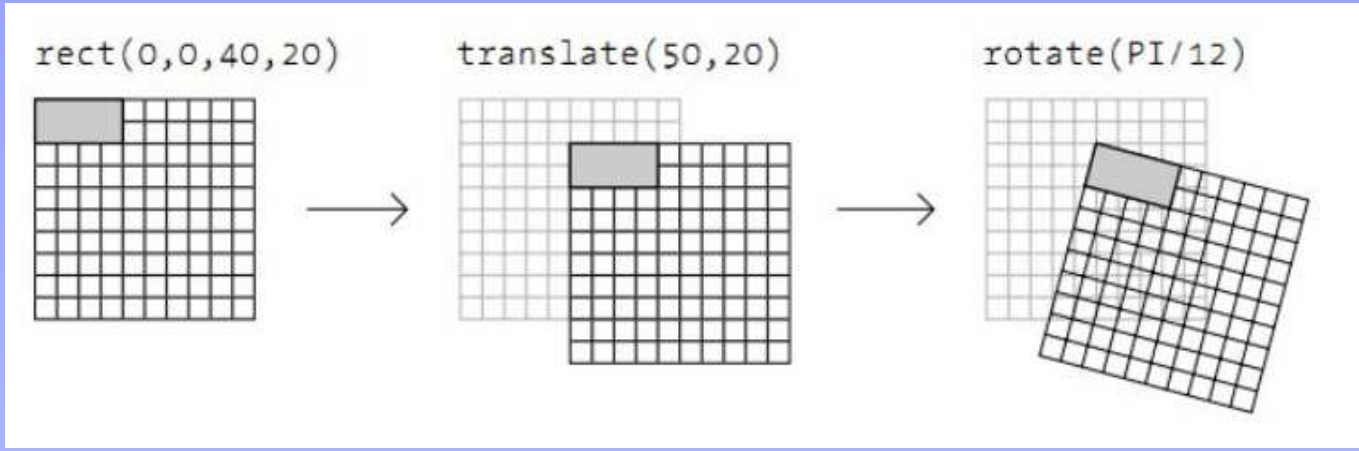
- El efecto rotación también es acumulativo.

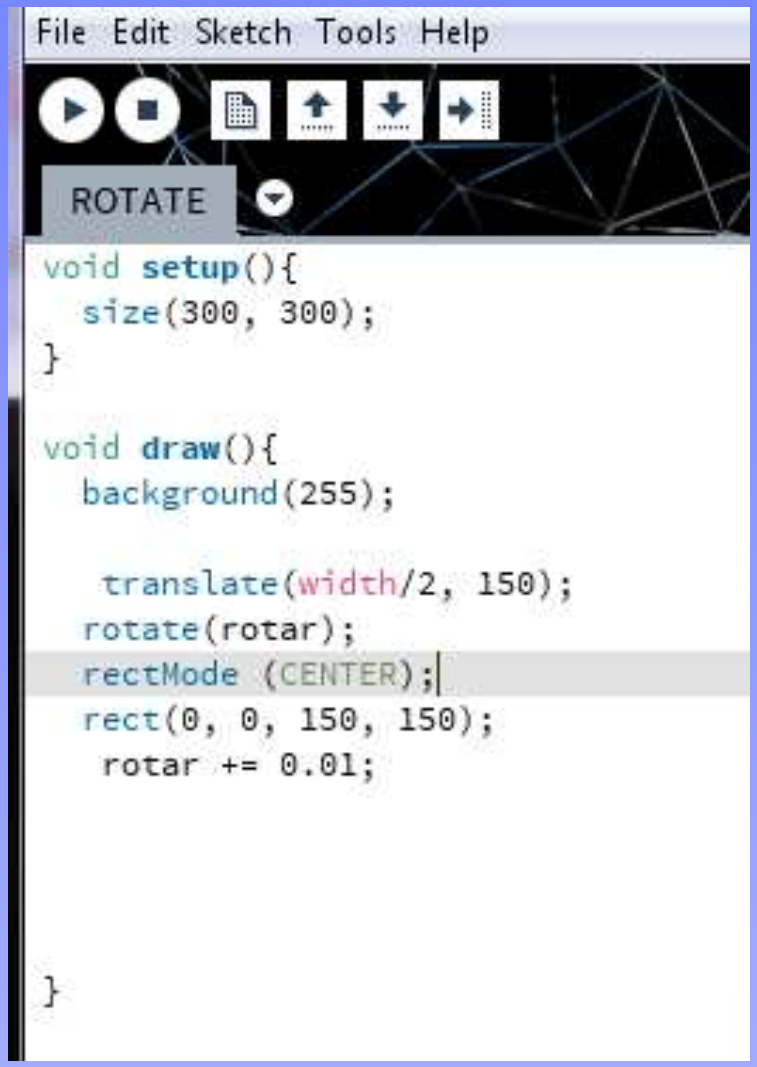
```
rotate(angulo)
```





Para hacer que la figura rote sobre su propia esquina, deberemos primero trasladar la coordenada a la posición 0,0





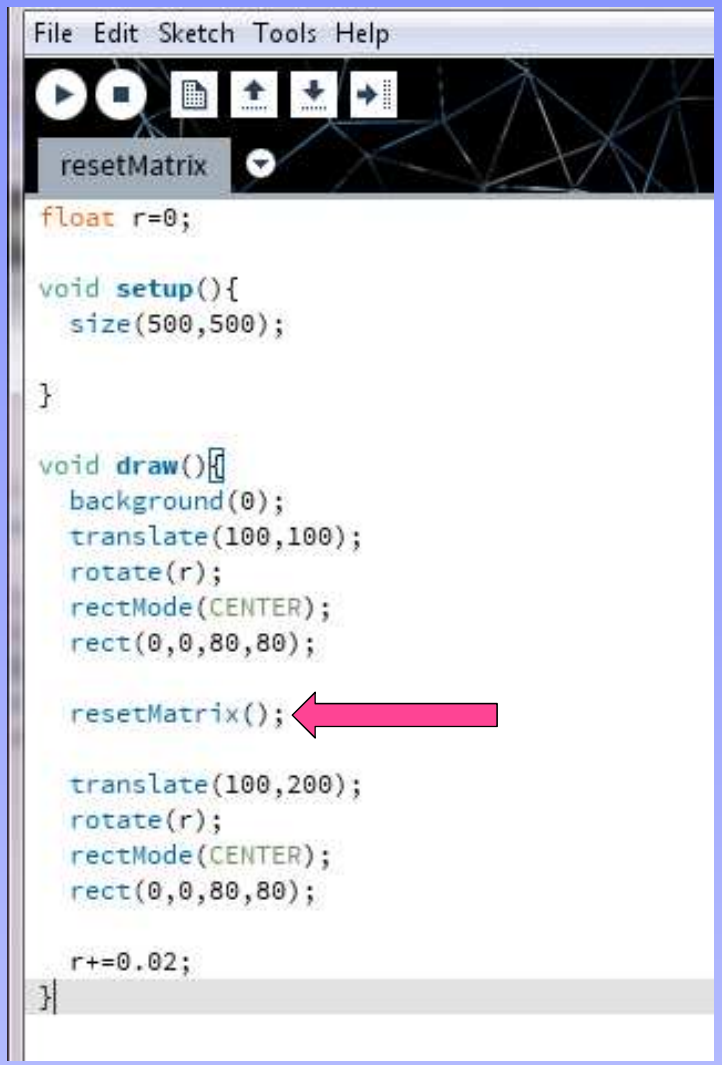
Atención al modo en que se dibuja “rect”

**** Si queremos hacer girar un cuadrado alrededor de su centro, necesitamos hacer coincidir el centro del cuadrado con el origen de coordenadas y trasladarlo.**

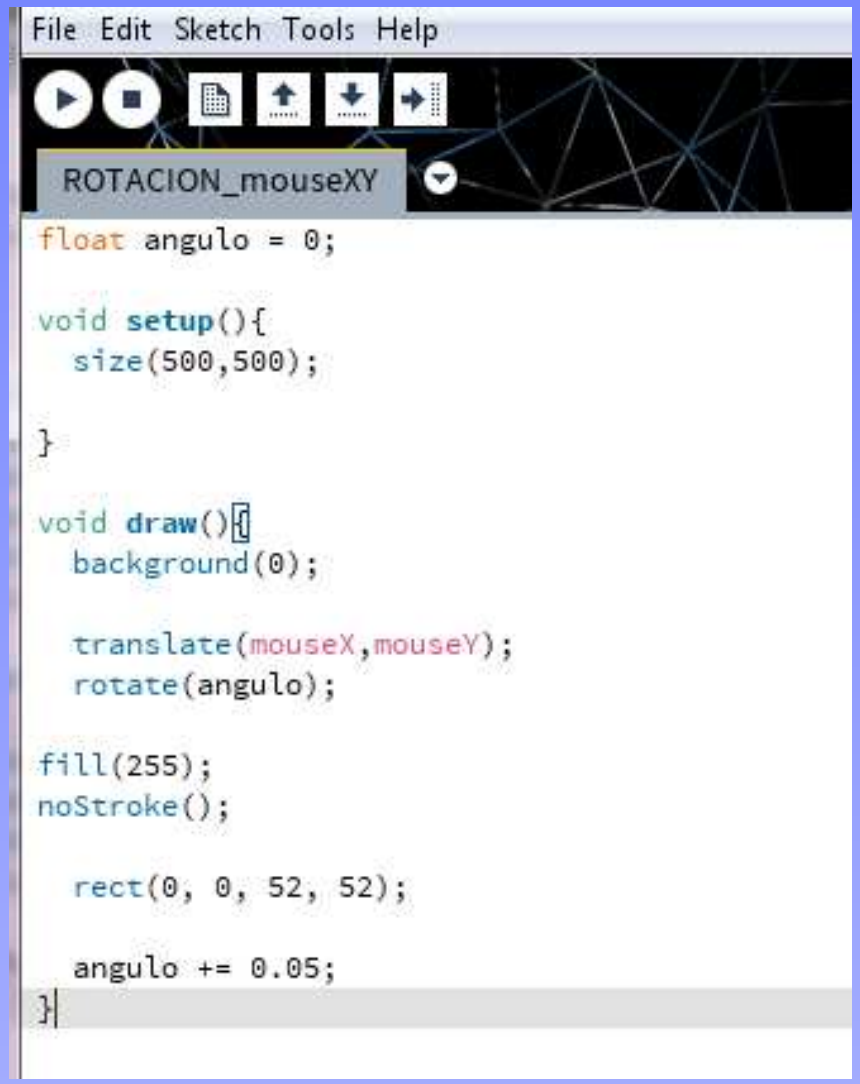
muestra traslación-rotación

resetMatrix();

Cuando quieres aplicar diferentes movimientos y traslaciones a diferentes figuras dentro del mismo sketch de processing, podemos hacer uso de resetMatrix. Esta función devuelve el origen de coordenadas a su posición inicial, por lo que las transformaciones siguientes no se sumarán a las anteriores.



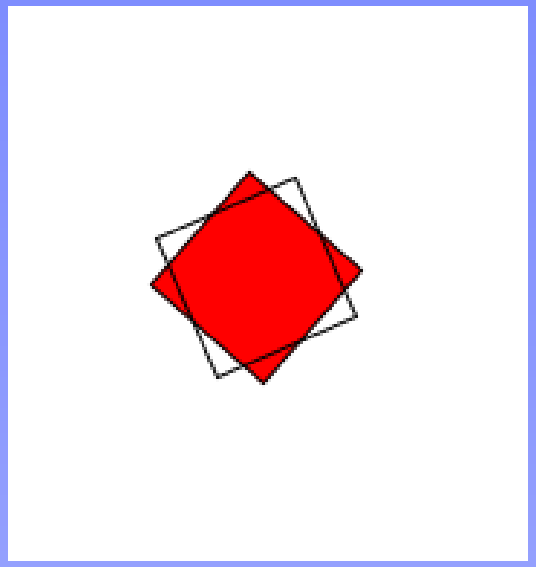
Ejemplo práctico en clase



ACTIVIDAD de clase

Partiendo del siguiente código, hacer que el rectángulo gire sobre su propio eje central y que describa una ráfaga en su movimiento rotatorio.

ACTIVIDAD ROTACION Y TRASLACIÓN:



Rotación de dos figuras, sobre el mismo eje y rotación con distinta angulación.

ACTIVIDADES repaso condicionales

** Pelota que rebota en la parte superior e inferior de la pantalla.

** Pelota rebota en las cuatro paredes.