

# Animar

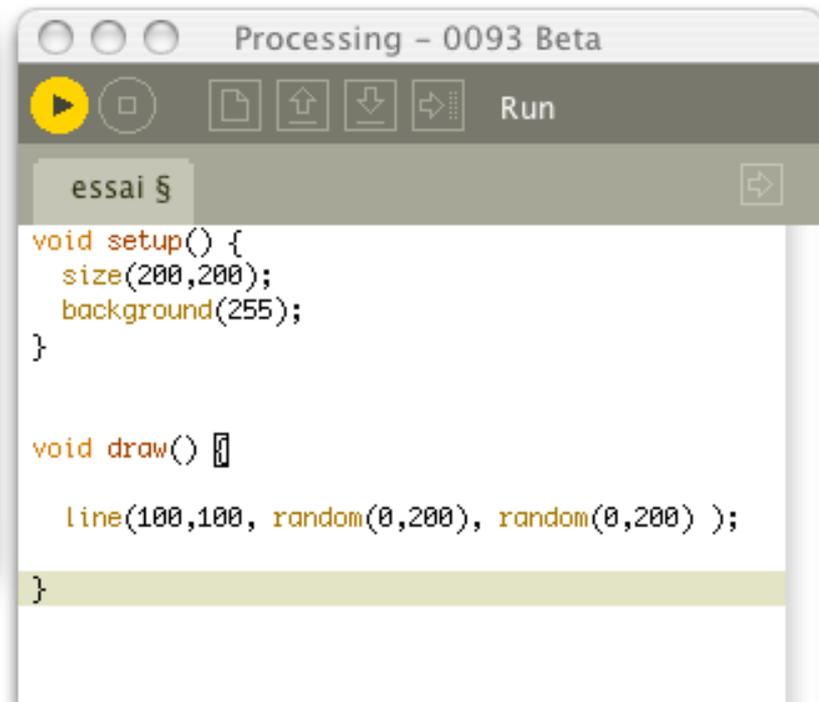
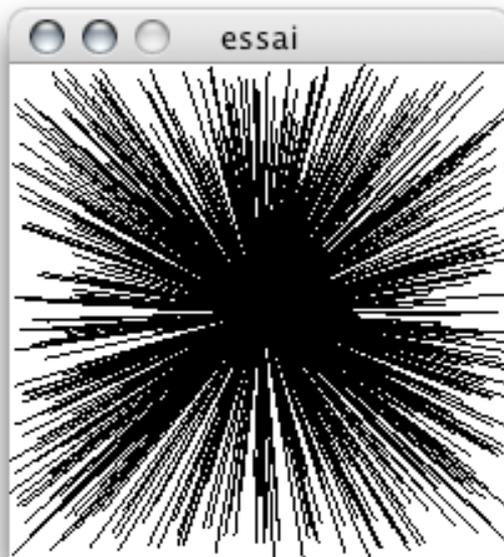
- Escribir el programa siguiente:

```
void setup() {  
  size(200,200);  
  background(255);  
}
```

```
void draw() {  
  line(100,100, random(0,200), random(0,200) );  
}
```

# Animar

- Resultado:
  - Diseño repetitivo y en directo de líneas de manera aleatoria.

A screenshot of the Processing IDE window titled "Processing - 0093 Beta". The window shows the code for the "essai" sketch. The code is as follows:

```
void setup() {  
  size(200,200);  
  background(255);  
}  
  
void draw() {  
  line(100,100, random(0,200), random(0,200) );  
}
```

# Animar

- Interpretación del código:
  - Transición de una programación escrita línea a línea para una programación en ciclo.
  - Creación de 2 nuevas estructuras:
    - `void setup( ) { ... }`  $\Leftrightarrow$  inicio de la animación con sólo 1 ejecución.
    - `void draw( ) { ... }`  $\Leftrightarrow$  movimiento de la animación con ejecución en continúa.

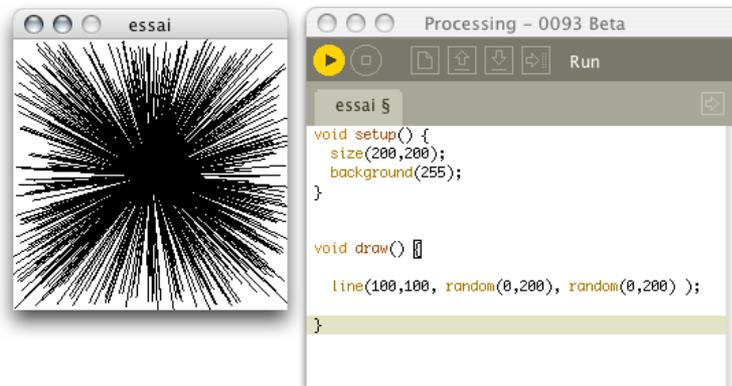
inicio de la animación,  
1 sólo ejecución

```
void setup() {  
  size(200,200);  
  background(255);  
}  
  
void draw() {  
  line(100,100, random(0,200), random(0,200) );  
}
```

ciclo draw(),  
diseño repetitivo

# Animar

- Interpretación de la arquitectura general del programa:
  - En el inicio de la animación (*setup()*):
    - Determinación del tamaño de la ventana y del color del fondo (255 = blanco).
    - En resumen, *setup()* determina lo que tiene que ocurrir cuando presionamos el botón play.
  - En el diseño repetitivo (*draw()*):
    - Diseño de líneas aleatorias desde el centro hasta el resto de la ventana.
    - En resumen, *draw()* determina lo que se pasa una vez iniciada la lectura y hasta que el utilizador pare la animación.

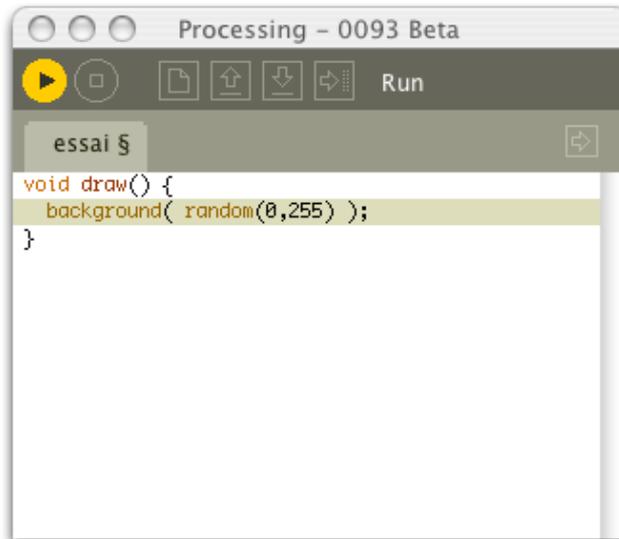


# Animar

- Interpretación de la expresión *random( )*:
  - = selección de un valor aleatorio (aquí entre 0 y 200) y colocación de este valor en la 3ª e 4ª posición de la función *line( )*.
  - => cada vez que el programa ejecuta la función *draw( )*, 2 nuevos valores son seleccionados y colocados dentro de la función *line( )*.
- Generalización de la sintaxis de la expresión *random( )*:
  - *random({valor mínima}, {valor máxima});*

# Animar

- Ejemplo de aplicación de la expresión *random()* para cambiar aleatoriamente el color de fondo de la pantalla:



// inicio del programa

```
void draw() {  
  background( random(0,255) );  
}
```

// fin del programa

- O programa escoge aleatoriamente ( $\approx 30$  veces por segundo) un nuevo color de fondo entre varios niveles de gris.

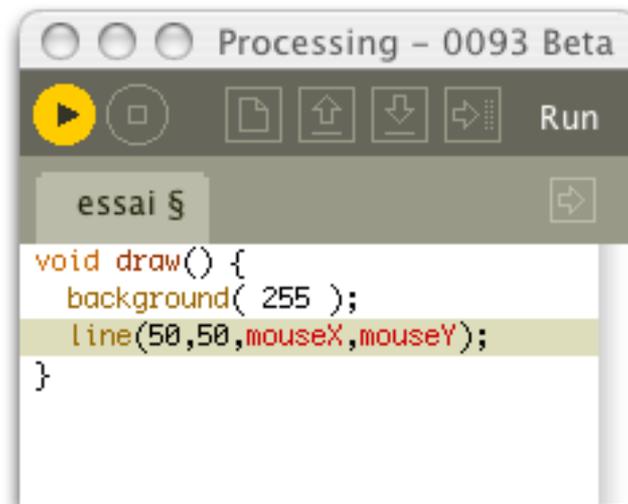
# Animar e interactuar

- Escribir el programa siguiente:

```
void draw() {  
    background( 255 );  
    line(50,50,mouseX,mouseY);  
}
```

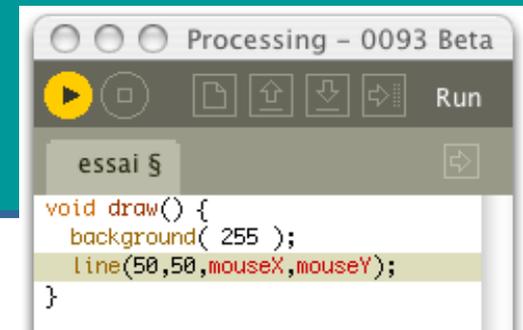
# Animar e interactuar

- Resultado:
  - Diseño de una línea que sigue la posición del puntero del ratón dentro de la ventana.

A screenshot of the Processing IDE interface. The window title is "Processing - 0093 Beta". The interface includes a toolbar with icons for play, stop, save, copy, paste, and run. Below the toolbar is a text area with the code:

```
essai §  
void draw() {  
  background( 255 );  
  line(50,50,mouseX,mouseY);  
}
```

# Animar e interactuar



- Interpretación:
  - Diseño repetitivo de una línea mediante la colocación de la función *line( )* dentro del ciclo de animación *void draw( ) { ... }*.
  - Sustitución de la expresión *random( )* por las nuevas expresiones *mouseX* y *mouseY*:
    - Durante la lectura de la animación esas expresiones son substituidas por el valor numérico instantáneo de las posiciones horizontales y verticales del ratón.
  - Resumen cronológico de los comandos que envía el programa:
    - La animación llena la ventana de 100x100 pixeles blancos (≈ 30 veces por segundo).
    - Después, diseño de una línea desde las coordenadas 50,50 hasta las posiciones *mouseX,mouseY* del ratón.
    - Sucesión continua de esas operaciones = **limpiar el antiguo diseño antes de diseñar el nuevo.**

# Animar e interactuar

- Ejercicios:
  - Hacer un programa que diseña en continua una forma cualquier.
  - Hacer un diseño influenciado por las posiciones del ratón (desplazamiento y cambio de color y/o cambio de forma).

A screenshot of the Processing IDE window titled "interaction\_souris2 | Processing 2.0.3". The window shows the code editor with the following code:

```
void setup(){
  size(400,400);
  background(255);
}
void draw() {
  fill(random(0,255), random(0,255), random(0,255));
  noStroke();
  ellipse(mouseX,mouseY, (width/2)-abs(mouseX-(width/2)),
    (height/2)-abs(mouseY-(height/2)));
}
```

The IDE interface includes a menu bar (File, Edit, Sketch, Tools, Help), a toolbar with icons for play, stop, and other functions, and a dropdown menu for the current sketch name "interaction\_souris2".