Los ciclos *(loops)*

- Definición:
  - Un ciclo es una lista de instrucciones que debe ser ejecutada de forma repetitiva.
  - => Entonces, los ciclos son ejecutados en ciclo, en ciclo, en ciclo… hasta que una condición sea verificada.

Los ciclos *(loops)*

- Imaginemos que queremos diseñar un rectángulo que va a cambiar de color, de manera progresiva, desde el blanco hasta el negro.
  - Podemos escribir eso:

```
size(300,300);

background(255);

stroke(0);
line(10,10,290,10);

stroke(1);
line(10,11,290,11);

stroke(2);
line(10,12,290,12);

stroke(3);
line(10,13,290,13);

// continuar así durante una hora…

stroke(254);
line(10,264,290,264);

stroke(255);
line(10,265,290,265);
```

Es un poco fastidioso e ingrato... <=

Los ciclos *(loops)*

- Para transformar su computador en amigo pode pedir que ejecuta para nosotros este trabajo repetitivo, a través del uso de 2 cosas:
  - 1. Un contador (<=> una variable)
  - 2. Una lista de tareas que el computador debe realizar en ciclo

Los ciclos *(loops)*

- El ciclo while( ) {…}

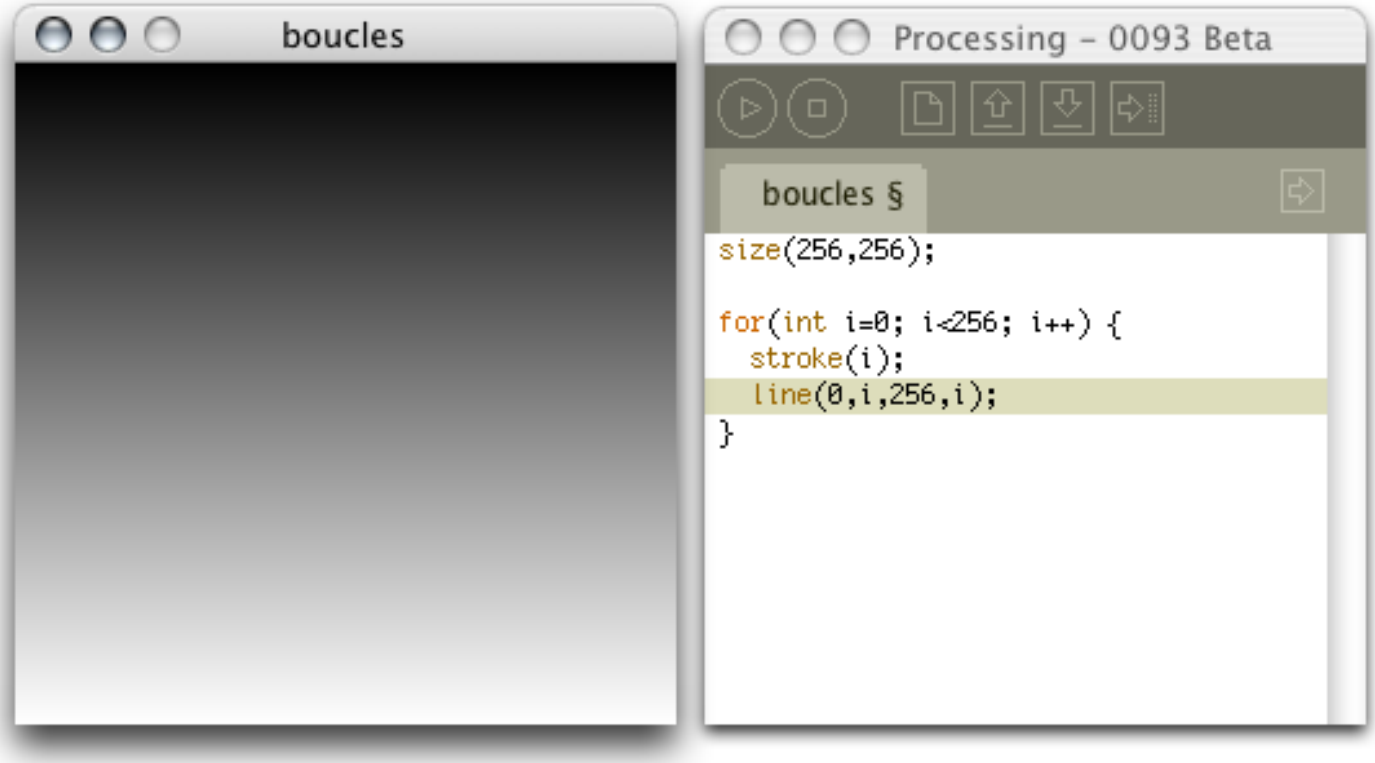
Processing – 0093 Beta

essai

```
// définir le cadre
size(200,300);
// remplir l'image de blanc
background(255);

// créer notre compteur (un "variable") et donner la valeur de 0
int i=0;

// boucler des operations pendant que compteur est plus petit que 256
while( i < 256) {

  // profiter de la valeur du compteur pour dessiner avec cette valeur
  stroke(i);

  // profiter de la valeur du compteur pour définir la position y de la ligne
  line(10, i, 190, i);

  // incrementer notre compteur
  i++;
}
```

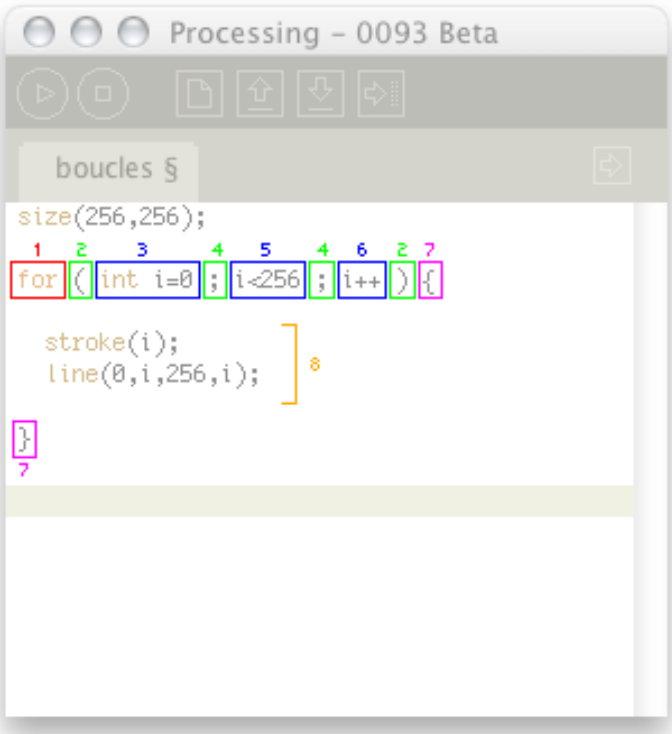
Los ciclos *(loops)*

- El ciclo for( ) {…}



Los ciclos *(loops)*

- Descripción del ciclo for( ):



1. La expresión *for.*

2. Los paréntesis.

3. La declaración (si necesario) de la variable-contador y su valor de inicio.

4. Puntos y comas para separar las 3 informaciones (nombre del contador, limite y valor de incrementación) de un ciclo *for.*

5. Condición para quedarse en el ciclo (aquí, i < 256).

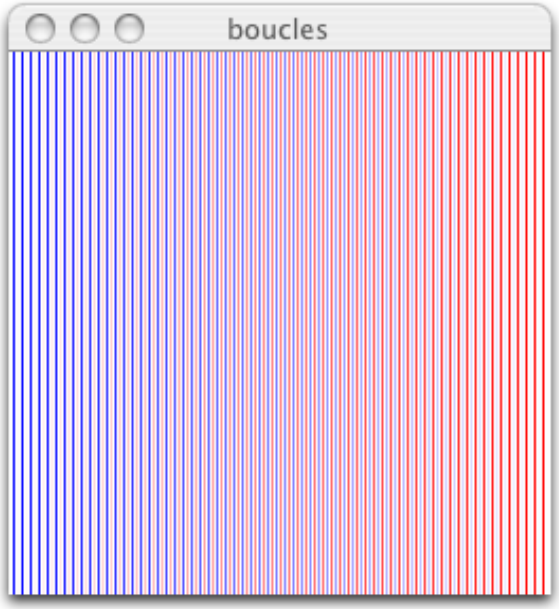
6. Valor de incrementación (aquí, i = i + 1).

7. Las llaves para delimitar el ciclo.

8. Las instrucciones contenidas dentro del ciclo.

Los ciclos *(loops)*

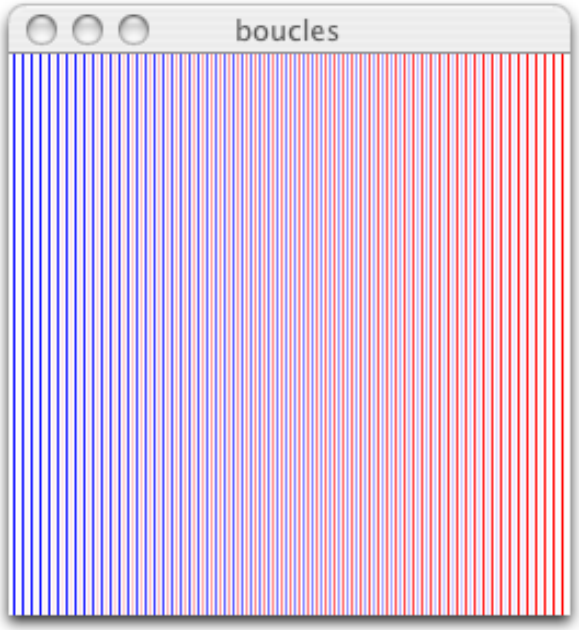
- Otro ejemplo de ciclo *for( )*, con las siguientes características:
  - Valor de incrementación = 4
  - Desalineamiento de 2 pixeles entre el ciclo de leñas azules y el ciclo de leñas rojas
  - Gradiente de opacidad (valor *alpha*) de dirección opuesta entre el ciclo azul y el ciclo rojo

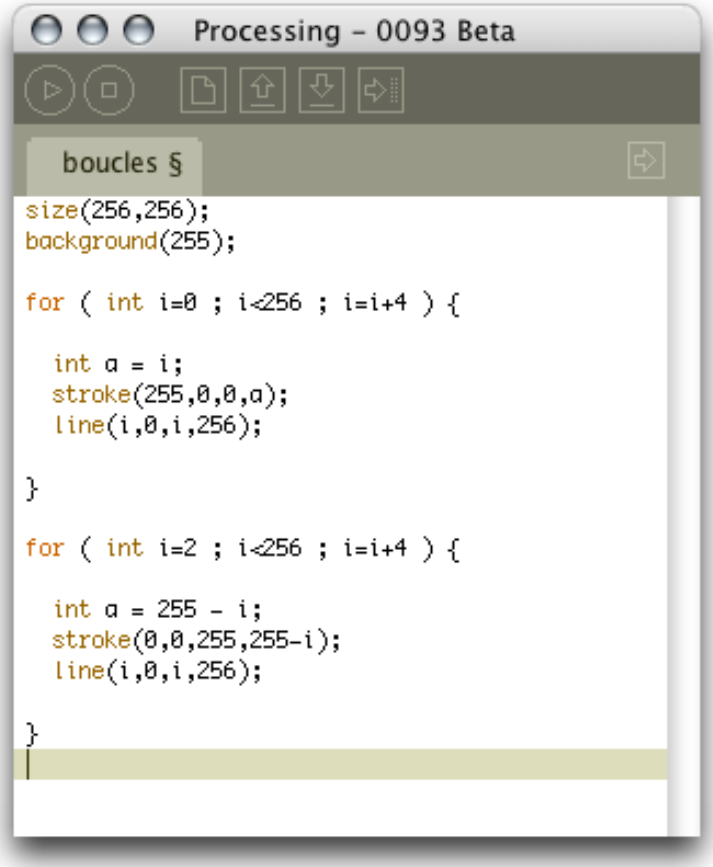


Intenta reencontrar el código correspondiente a este diseño…

Los ciclos *(loops)*

- Solución:



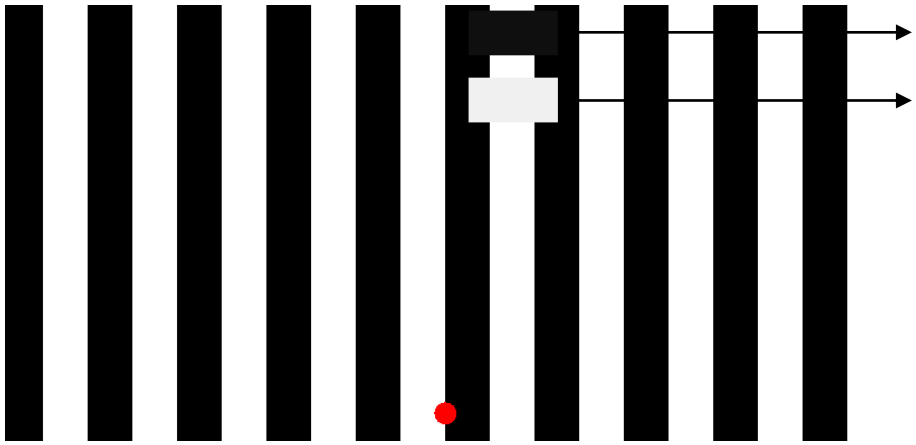


Los ciclos *(loops)*

- Resumo sobre los ciclos (3º concepto fundamental de la programación):
  - Un ciclo permite hacer cualquier cosa de forma repetitiva, a través de la incrementación automática de una variable.

Los ciclos *(loops)*

- Ejercicios:
  - Modificar antiguos programas, a través del aditamento de algunos ciclos *while( )* o *for( ).*
  - Por ejemplo, en vez de escribir varios métodos *rect( )* para definir el fondo del “footsteps illusion”, criar un ciclo que va a hacer eso automáticamente.



Los ciclos *(loops)*

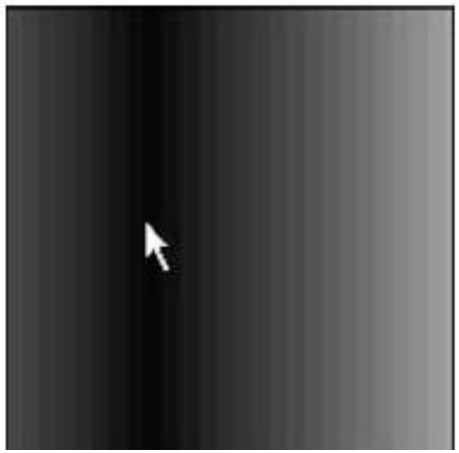
- Atención!!!
  - **Siempre confirmar** que ha una **manera de salir** del ciclo, a fin de evitar los *dead lock* (ciclos infinitos) como por ejemplo:

```
int x = 0;
while (x < 10){
  println(x);
  x = x - 1;
}
```

Decrementing *x* results in an infinite loop here because the value of *x* will never be 10 or greater. Be careful!

Los ciclos *(loops)*

- Ejercicios:
  - Diseñar una serie de rectángulos verticales (10 pixeles de anchura), cada un coloridos con un brillo que depende de la distancia al ratón:
    - La distancia entre un determinado rectángulo y el ratón es igual al valor absoluto (*abs()*) de la distancia entre la variable *i* (coordenada x de un rectángulo) y la variable *mouseX.*
    - Esta distancia es usada para llenar el color de un rectángulo con una localización horizontal *i.*
  - Reescribir el programa de arriba con un tipo de ciclo diferente (*for()* o *while()*).



Los ciclos *(loops)*

- Solución:

```
void setup() {
  size(250,250);
}

void draw() {
  background(0);
  int i = 0;
  while(i < width) {
    noStroke();
    float distance = abs(mouseX - i);
    fill(distance);
    rect(i,0,10,height);
    i += 10;
  }
}
```