


Imagen fotográfica

- Es posible importar imágenes fotográficas para el Processing (a partir de imágenes del web, de una máquina fotográfica numérica...).
- Por eso, existen **2 posibilidades de localizar** el fichero imagen fuente:
 - La imagen situase en un fichero exterior al proyecto (ej., Internet).
 - La imagen situase en la carpeta “data”.

Imagen fotográfica

- Caso de una imagen importada directamente de **internet**:



```
Processing - 0095 Beta

sketch_051122a §

// déclarer une variable qui contiendra toute l'image
PImage photo;

// cadre assez grand
size(300,400);

// aller sur le web, chercher l'image, et mettre ses données dans la variable 'photo'
photo = loadImage("http://upload.wikimedia.org/wikipedia/commons/4/44/MakingAFace.jpg");

// afficher l'image dans la variable à partir de la position 0,0
image(photo,0,0);
```

- Processing importa la imagen directamente en la memoria del software a través del comando: **loadImage ();**
- Una vez importada la imagen, aparece en la pantalla a través del comando: **image();**

Imagen fotográfica

- Dado que las **coordenadas** de `image(photo, 0, 0)` son $x = 0$ e $y = 0$, la foto situase en el ángulo superior-izquierdo de la pantalla.



Imagen fotográfica

- Para integrar la imagen directamente en el programa, es necesario importar en la carpeta “data”, a través del siguiente proceso:
 - 1) Seleccionar el menú: **Sketch** → **Add File**

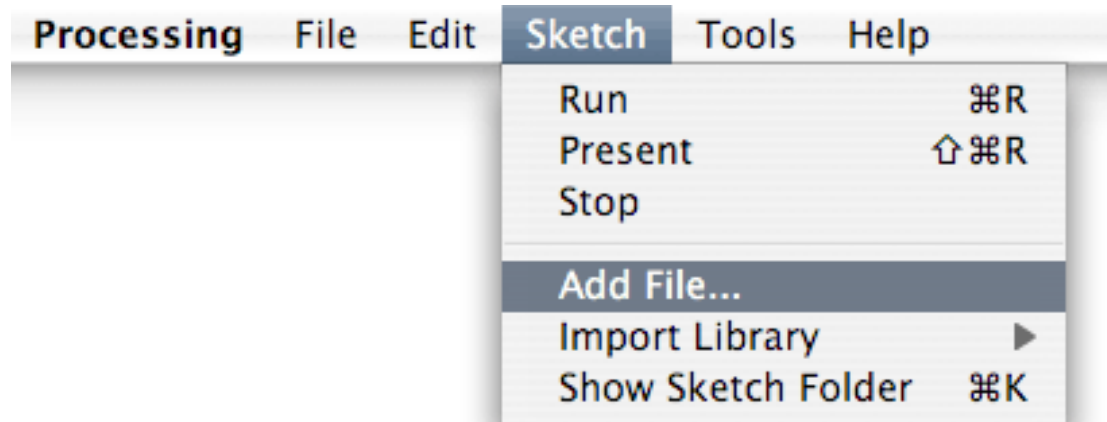


Imagen fotográfica

- 2) **Escoger el fichero** imagen que desea importar.
- Una vez importada la imagen, es posible verificar su localización (en la carpeta “data”), a través del menú: **Sketch → Show Sketch Folder**

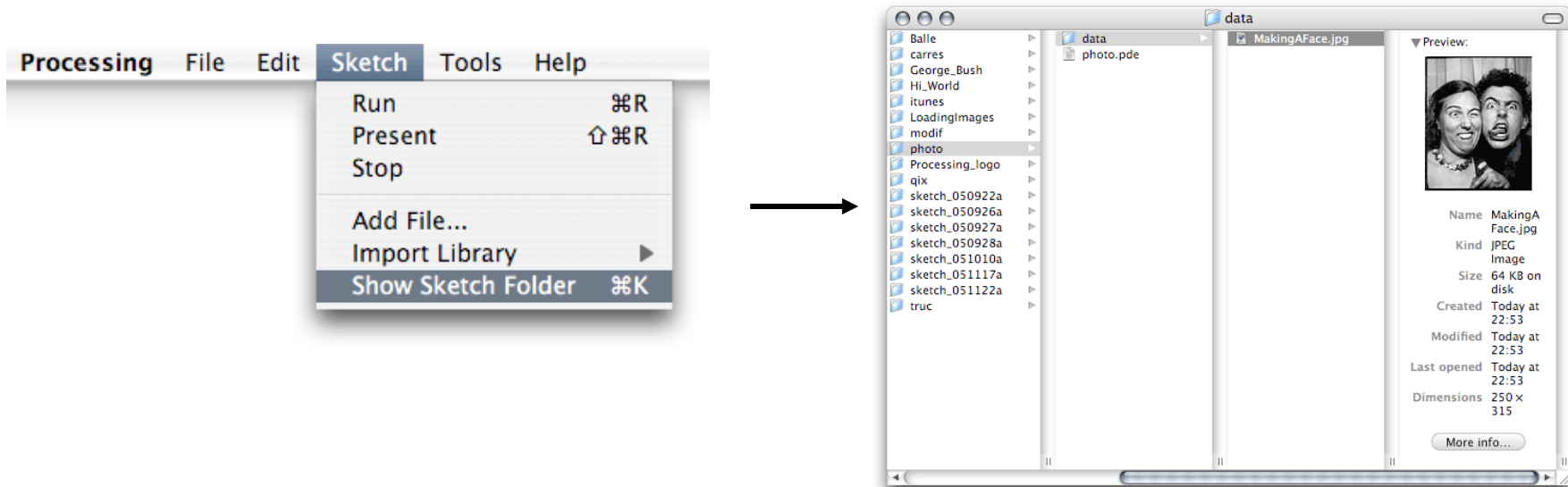


Imagen fotográfica

- En este 2º caso, tiene que modificar, ligeramente, la **manera de escribir el endereço** de la imagen a importar en el programa:



```
Processing - 0095 Beta

photo §

// déclarer une variable qui contiendra toute l'image
PImage photo;

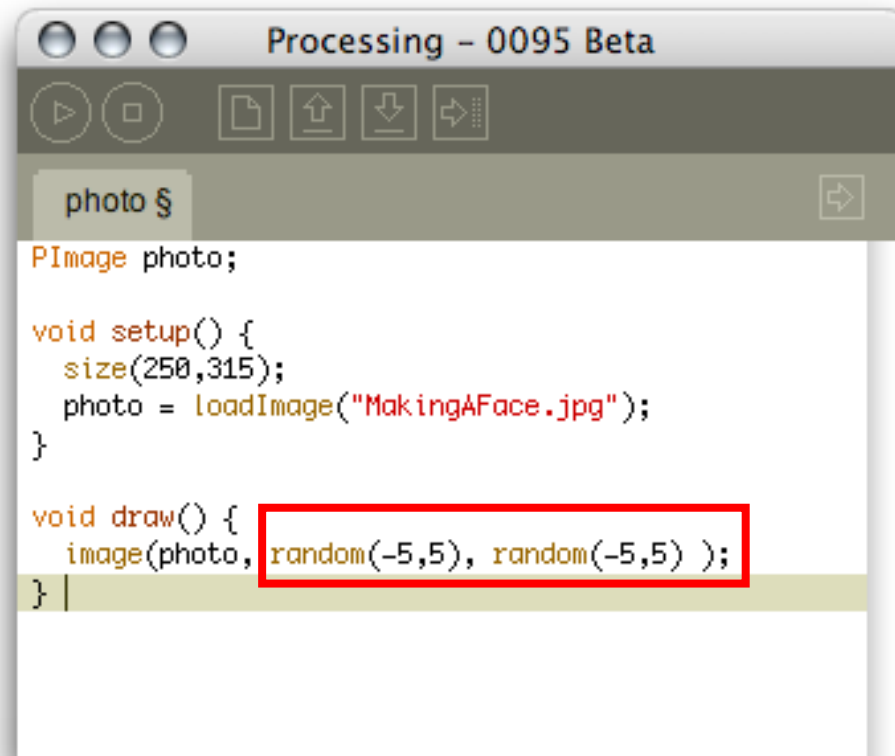
// cadre assez grand
size(300,400);

// aller dans le dossier "data" et transférer l'image dans la variable 'photo'
photo = loadImage("MakingAFace.jpg");

// afficher l'image dans la variable à partir de la position 0,0
image(photo,0,0);
```

Imagen fotográfica

- Poner la imagen en movimiento:

A screenshot of the Processing IDE window titled "Processing - 0095 Beta". The window shows a code editor with the following code:

```
photo §  
PImage photo;  
  
void setup() {  
  size(250,315);  
  photo = loadImage("MakingAFace.jpg");  
}  
  
void draw() {  
  image(photo, random(-5,5), random(-5,5) );  
}
```

The code is color-coded: keywords are orange, strings are red, and comments are grey. The line `image(photo, random(-5,5), random(-5,5));` is highlighted with a red rectangular box. The IDE interface includes a toolbar with icons for play, stop, save, upload, download, and help, and a tab labeled "photo §".

Imagen fotográfica

- Redimensionar la imagen, a través de la adición de 2 variables a seguir a las posiciones x e y:
 - Para recubrir toda la anchura y altura de la pantalla: **width** y **height**.
 - Para un redimensionamiento dinámico: **mouseX** y **mouseY**.



```
Processing - 0095 Beta

photo §

PImage photo;

void setup() {
  size(250,315);
  photo = loadImage("MakingAFace.jpg");
}

void draw() {
  background(0);
  image(photo, 0, 0, mouseX, mouseY );
}
```


Imagen fotográfica

- Pintar la imagen:



```
Processing - 0095 Beta  
Run  
photo §  
PImage photo;  
  
void setup() {  
  size(250,315);  
  photo = loadImage("MakingAFace.jpg");  
}  
  
void draw() {  
  background(0);  
  tint(mouseX,mouseY,mouseY-mouseX,255);  
  image(photo, 0, 0);  
}
```

R G B alpha

Ejercicios

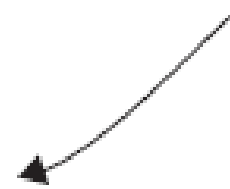
- Encontrar una imagen en Google, recuperar su *url* (o sea, el endereço `http://...`) y hacer un programa que anime esta imagen.
- Aplicar vuestros conocimientos a la imagen en términos de diseños y programación en Processing...

Tratamiento de imágenes

- Pixeles

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

How the pixels look



How the pixels are stored.



0	1	2	3	4	5	6	7	8	9	.	.	.			
---	---	---	---	---	---	---	---	---	---	---	---	---	--	--	--

Tratamiento de imágenes

- Ejemplo para tener a acceso a cada pixel de la ventana:

```
size(200,200);  
loadPixels(); // carga todos los pixeles de la ventana  
for(int i = 0; i < pixels.length; i++) {  
    float rand = random(0,255);  
    color c = color(rand);  
    pixels[i] = c;  
}  
updatePixels(); // actualiza las modificaciones hechas a los pixeles
```

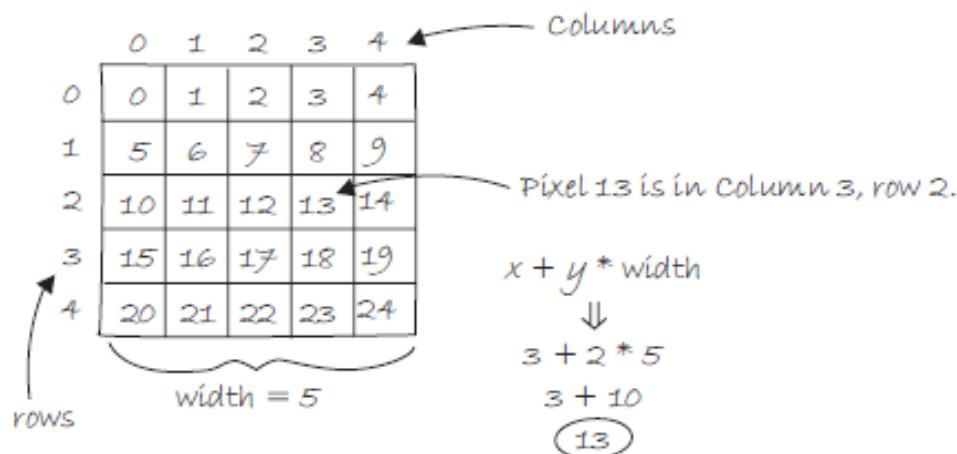


- Ejercicio – Alterar el código de arriba para difundir colores R,G,B aleatorias para cada pixel.

Tratamiento de imágenes

- Formula para determinar la posición de un pixel (coordenadas bidimensionales X, Y) dentro de un array unidimensional:

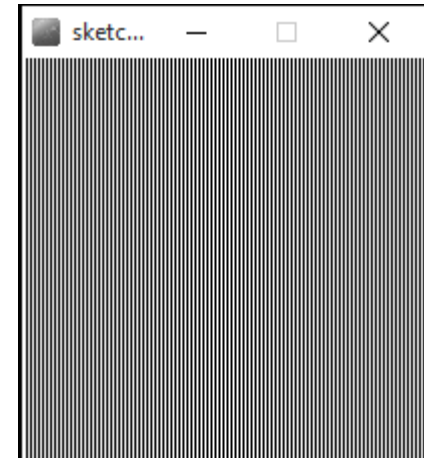
$$\text{LOCATION} = X + Y * \text{WIDTH}$$



Tratamiento de imágenes

- Ejemplo para dar un color diferente a las columnas de pixeles, de forma a que las **columnas pares aparecen blancas** y las **columnas impares aparecen negras**:

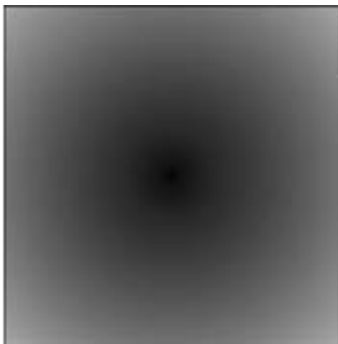
```
size(200,200);  
loadPixels();  
for(int x = 0; x < width; x++) {  
  for(int y = 0; y < height; y++) {  
    int loc = x + y * width;  
    if(x % 2 == 0) {  
      pixels[loc] = color(255);  
    } else {  
      pixels[loc] = color(0);  
    }  
  }  
}  
updatePixels();
```



- Ejercicio – Altera el código para alternar los colores de las líneas en vez de las columnas.

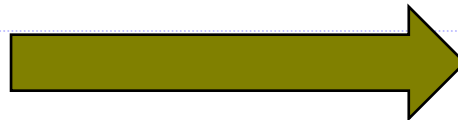
Tratamiento de imágenes

- Ejercicio – Completar el siguiente código para obtener la imagen a la izquierda:



```
size(255,255);  
_____  
for (int x = 0; x < width; x++) {  
  for (int y = 0; y < height; y++) {  
    int loc = _____;  
    float distance = _____);  
    pixels[loc] = _____;  
  }  
}  
_____;
```

- Solución



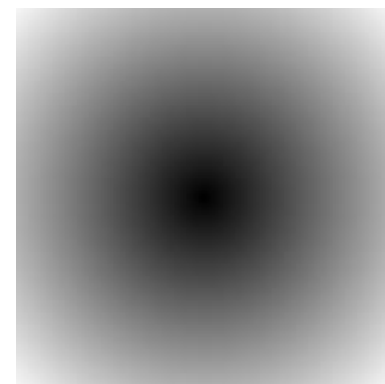
Tratamiento de imágenes

■ Solución 1:

```
size(255,255);
loadPixels();
for(int x = 0; x < width; x++) {
  for(int y = 0; y < height; y++) {
    int loc = x + y * width;
    float distance = sqrt((width/2-x)*(width/2-x)+(height/2-y)*(height/2-y));
    pixels[loc] = color(distance);
  }
}
updatePixels();
```

■ Solución 2 (para mapear la totalidad del espectro de color con la dimensión de la ventana):

```
size(255,255);
loadPixels();
for(int x = 0; x < width; x++) {
  for(int y = 0; y < height; y++) {
    int loc = x + y * width;
    float distance = sqrt((width/2-x)*(width/2-x)+(height/2-y)*(height/2-y));
    float dist = map(distance, sqrt((width/2)*(width/2)+(height/2)*(height/2)), 0, 255, 0);
    pixels[loc] = color(dist);
  }
}
updatePixels();
```



Tratamiento de imágenes

- Difundir los pixeles de una imagen

```
PImage img;

void setup() {
  size(768,768);
  img = loadImage("sunflower.jpg");
}

void draw() {
  loadPixels();
  img.loadPixels(); // para ler los pixeles de la imagen
  for(int y = 0; y < height; y++) {
    for(int x = 0; x < width; x++) {
      int loc = x + y * width;
      // para extraer los 3 componentes de color de cada pixel
      float r = red(img.pixels[loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);

      pixels[loc] = color(r,g,b);
    }
  }
  updatePixels();
}
```



Tratamiento de imágenes

- Ejercicio – Transformar el código anterior para difundir la imagen en una escala de niveles de gris



Tratamiento de imágenes

- Solución:

```
PImage img;

void setup() {
  size(768,768);
  img = loadImage("sunflower.jpg");
}

void draw() {
  loadPixels();
  img.loadPixels(); // para ler los pixeles de la imagen
  for(int y = 0; y < height; y++) {
    for(int x = 0; x < width; x++) {
      int loc = x + y * width;
      // para extraer los 3 componentes de color de cada pixel
      float r = red(img.pixels[loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);

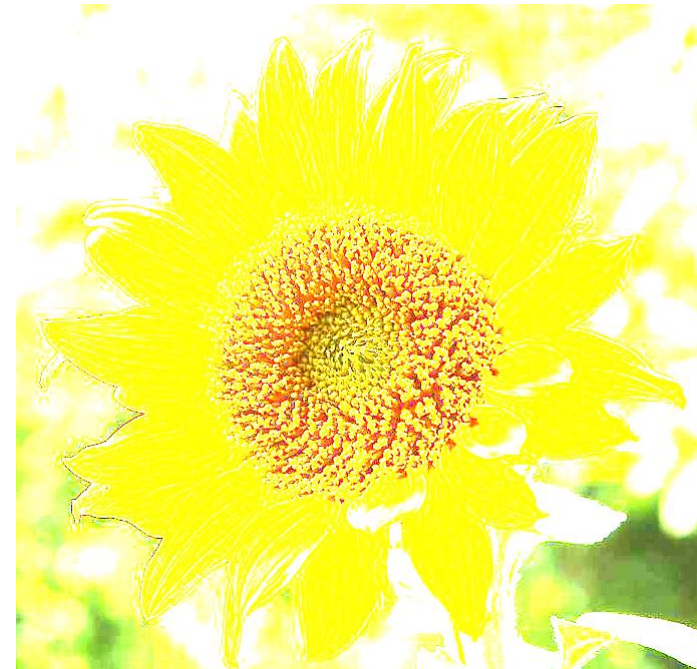
      // para transformar la imagen inicial en una escala de gris
      float grayscale = (r+g+b)/3;

      pixels[loc] = color(grayscale);
    }
  }
  updatePixels();
}
```

Tratamiento de imágenes

- Ajustar el brillo de la imagen en función de la posición del ratón

```
for(int y = 0; y < height; y++) {  
    for(int x = 0; x < width; x++) {  
        int loc = x + y * width;  
        float r = red(img.pixels[loc]);  
        float g = green(img.pixels[loc]);  
        float b = blue(img.pixels[loc]);  
        // cambia el brillo segundo la posición del ratón  
        float adjustBrightness = ((float)mouseX / width) * 8.0;  
        r *= adjustBrightness;  
        g *= adjustBrightness;  
        b *= adjustBrightness;  
        // constreñan los valores RGB entre 0 y 255  
        r = constrain(r,0,255);  
        g = constrain(g,0,255);  
        b = constrain(b,0,255);  
        // crea el nuevo color  
        color c = color(r,g,b);  
        pixels[loc] = color(c);  
    }  
}
```



Tratamiento de imágenes

- Transformar una imagen en negro y blanco en función de un determinado umbral (el resultado será presentado en una otra imagen)

```
PImage source; // Imagen fuente  
PImage destination; // Imagen tratada
```

```
void setup() {  
  size(768,768);  
  source = loadImage("sunflower.jpg");  
  destination = createImage(source.width, source.height, RGB);  
}
```

```
void draw() {  
  float threshold = 127;  
  source.loadPixels();  
  destination.loadPixels();  
  
  for(int x = 0; x < source.width; x++) {  
    for(int y = 0; y < source.height; y++) {  
      int loc = x + y*source.width;  
      if(brightness(source.pixels[loc]) > threshold) {  
        destination.pixels[loc] = color(255);  
      } else {  
        destination.pixels[loc] = color(0);  
      }  
    }  
  }  
}
```

```
destination.updatePixels();  
image(destination,0,0);  
}
```



Tratamiento de imágenes

- Detección de bordos a partir de la diferencia de brillo entre de 2 pixeles vecinos

```
PIimage source;
PIimage destination;

void setup() {
  size(768,768);
  source = loadImage("sunflower.jpg");
  destination = createImage(source.width, source.height, RGB);
}

void draw() {
  source.loadPixels();
  destination.loadPixels();
  for(int x = 1; x < source.width; x++) {
    for(int y = 0; y < source.height; y++) {
      // Lectura del pixel actual
      int loc = x + y * source.width;
      color pix = source.pixels[loc];
      // Lectura del pixel a la izquierda
      int leftLoc = (x - 1) + y * source.width;
      color leftPix = source.pixels[leftLoc];
      // El nuevo color es la diferencia entre el pixel actual y el pixel izquierdo
      float diff = abs(brightness(pix) - brightness(leftPix));
      destination.pixels[loc] = color(diff);
    }
  }
  destination.updatePixels();
  image(destination,0,0);
}
```

