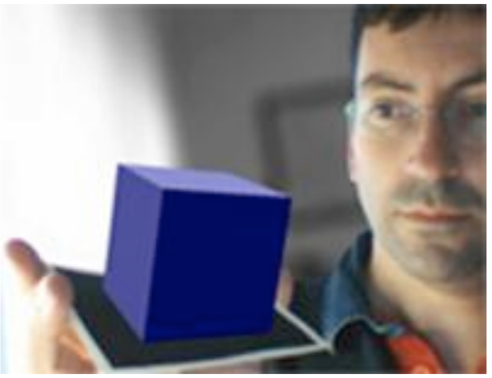
Computer Graphics

Images and text



Jordi Linares i Pellicer
Escola Politècnica Superior d'Alcoi
Dep. de Sistemes Informàtics i Computació
jlinares@dsic.upv.es
http://www.dsic.upv.es/~jlinares

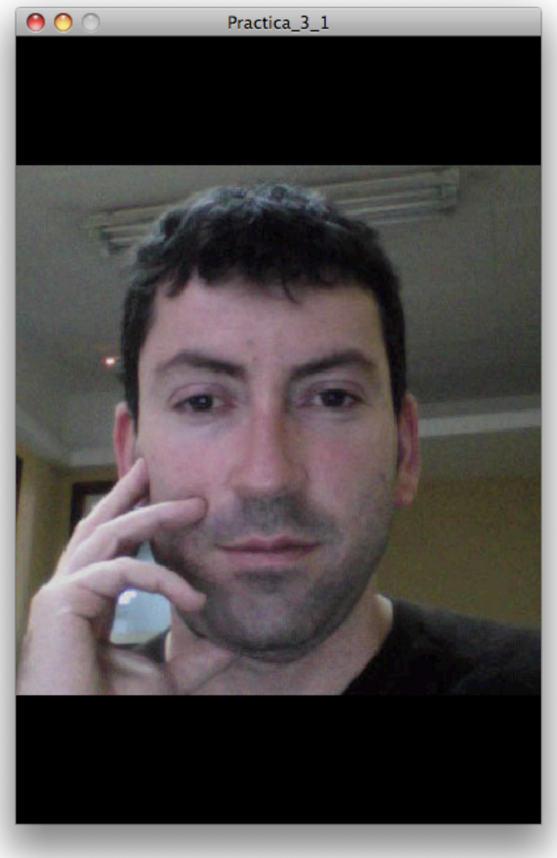
Images

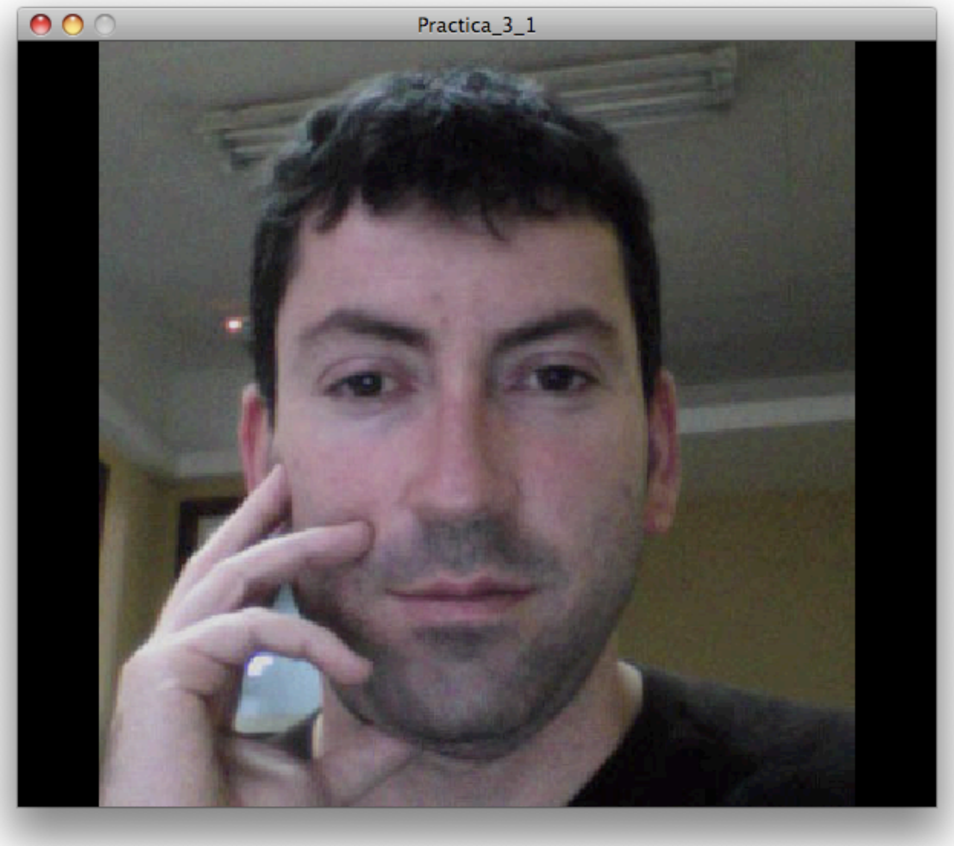
- *processing* offers the `PImage` class, with which objects of the type image can be created
- Function `loadImage()` allows to open an image in any of the following formats: GIF, JPG, TGA and PNG
- Files of images to load have to be in the application `data` directory (*processing* PDE has the command '*Add File*' that allows to select images and insert them automatically is such a directory)
- In order to visualise the image `image()` function can be used:
  - `image(image, x, y)` -> Show the image at coordinates (x, y), without changing its original size
  - `image(image, x, y, width, height)` -> Show the image at coordinates (x, y) and scale it to the specified width and height
- Examples:

```
PImage foto = loadImage("foto.jpg");
image(foto, 0, 0);
```

Practice 3-1

- Create a new application. Add an image to the data directory.
- Show the image covering all the window area, keeping its original proportions and centered (isotropic and centered transformation)
- It is possible to consult the image size with the attributes `width` and `height`. If `im` is of the type `PImage`, its size is `im.width` and `im.height`





Images

`createImage(width, height, color mode)`

- Allows to create a new image
- The color mode can be `RGB`, `ARGB` or `ALPHA`
  - `RGB` allows 24 bits images (8 per channel, 'true color')
  - `ARGB` adds an additional channel for transparency
  - `ALPHA` for images of just one channel (transparency effects)

Images

```
get()
get(x, y)
get(x, y, width, height)
```

- Methods `get()` allow either to get a pixel of an image, `get(x, y)`, or a new image as a fragment of the original one `get(x, y, width, height)`. Method `get()` without parameters returns an image copy of the image over which the method is applied.

Images

```
set(x, y, color)
set(x, y, image)
```

- Methods `set()` allow either to change the value of a pixel of the image over which the method is applied, `set(x, y, color)`, or to map an image at coordinates (x, y) over the image the method `set()` is applied.

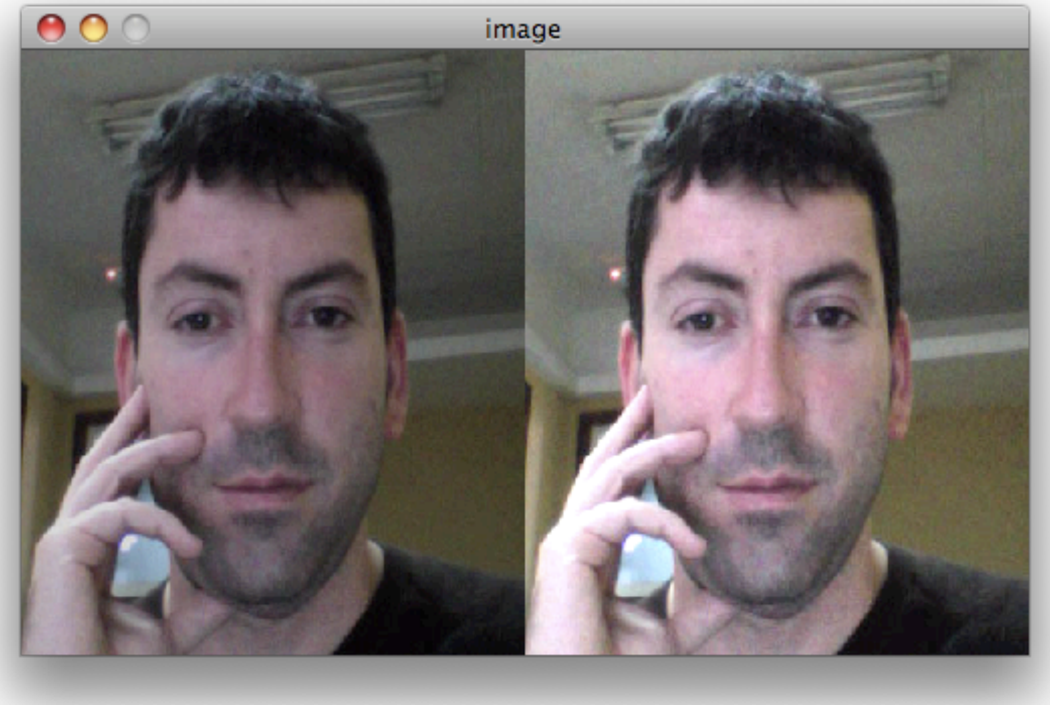
Images

- To facilitate the work with colors, it is possible to use the `color` type (just an integer really), the function `color(r, g, b)` that allows to create a new color from the RGB corresponding values, and functions `red(color)`, `green(color)` and `blue(color)` that returns the appropriate channel from the RGB color.

Images

- Example:

```
// foto.jpg must be in 'data' application directory
PImage foto_original = loadImage("foto.jpg");

PImage foto_modified = createImage(foto_original.width,
                                   foto_original.height, RGB);

// 50% more intensity
for (int i = 0; i < foto_original.width; i++)
  for (int j = 0; j < foto_original.height; j++) {
    color c_o = foto_original.get(i, j);
    color c_d = color(min(255, red(c_o) * 1.5),
                      min(255, green(c_o) * 1.5),
                      min(255, blue(c_o) * 1.5));
    foto_modified.set(i, j, c_d);
  }

// We show both images
size(500, 300);
image(foto_original, 0, 0, width/2, height);
image(foto_modified, width/2, 0, width/2, height);
```

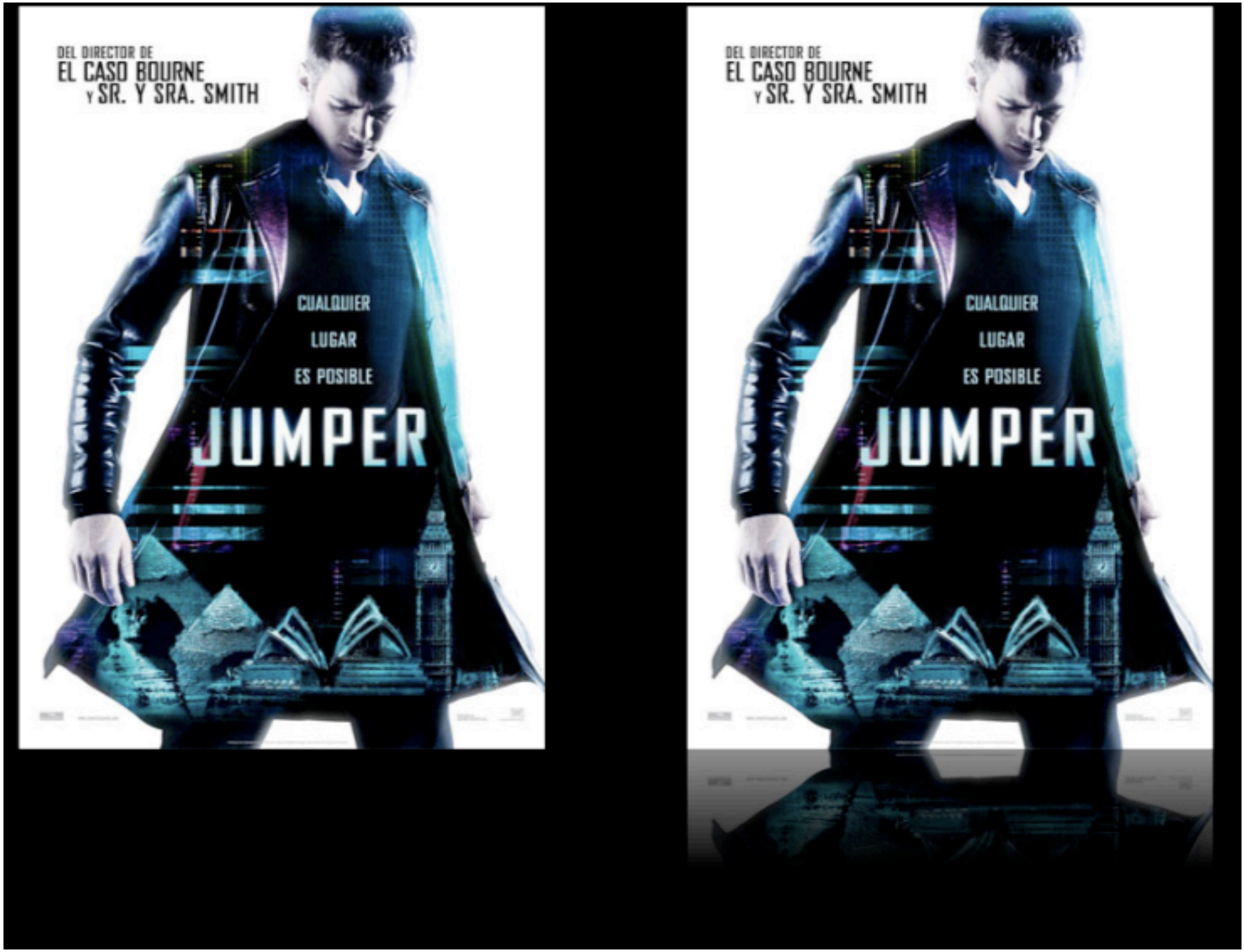


Images

- Example:

```
// A reflected effect
PImage foto_original = loadImage("foto.jpg");
PImage foto_modified = createImage(foto_original.width,
                                   foto_original.height / 3,
                                   ARGB);

// We take 1/3 of the original image,
// make a mirror, and we apply a gradient of
// transparencies
int h23 = foto_original.height * 2 / 3;
int h13 = foto_original.height / 3;
for (int j = h23; j < foto_original.height; j++) {
  int alpha = int((j - h23) * (255.0 / h13)) - 128;
  for (int i = 0; i < foto_original.width; i++) {
    color c_o = foto_original.get(i, j);
    color c_d = color(red(c_o), green(c_o), blue(c_o), alpha);
    foto_modified.set(i, h13 - (j - h23), c_d);
  }
}

size(1000, 700);
background(0);

// Image
image(foto_original, 50, 10);

// Original image with reflected effect
image(foto_original, 500, 10);
image(foto_modified, 500, foto_original.height + 10);
```

Images


DEL DIRECTOR DE
EL CASO BOURNE
Y SR. Y SRA. SMITH
CUALQUIER
LUGAR
ES POSIBLE
JUMPER

Images

```
save(nombre fichero)
```

- Saves the image in the graphic file format specified in the extension.
- Allowed formats are TIFF, TARGA, JPEG and PNG

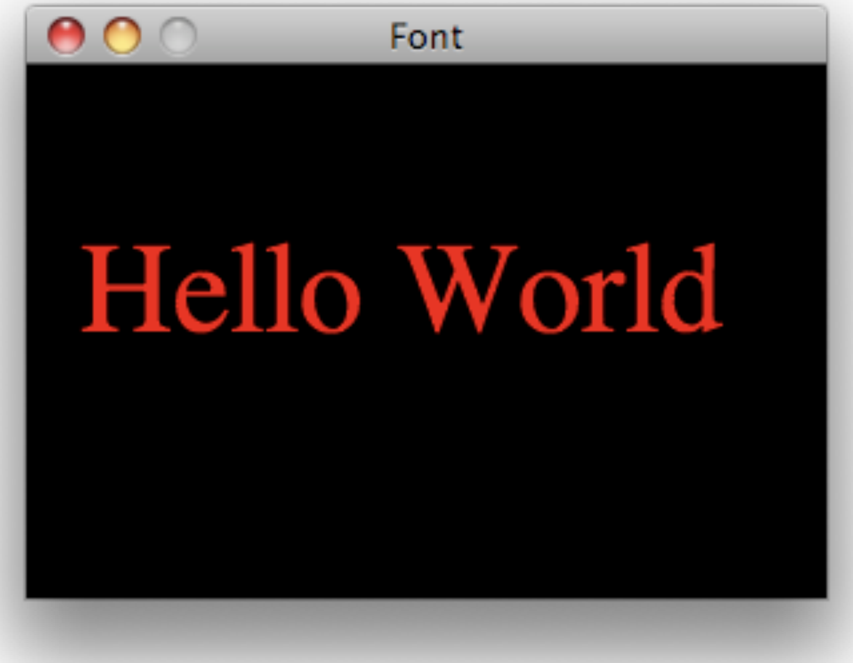
Typography

- *processing* offers the `PFont` class to create fonts
- Fonts must be created and save in the `data` directory of the applicaction. In order to do that, *processing* PDE has the 'Create Font' command in the tools menu. Fonts will be saved with .vlw extension
- Font has to be loaded with `loadFont()` and selected with `loadFont()` before being able to be used with `text()`
- There are a lot of functions and possibilities: `textAlign()`,`textSize()`,`textMode()`, `textLeading()` etc.

Typography

- Example:

```
size(300, 200);
background(0);

// We load the font (previously it is
// necessary to create it with the
// appropriate processing command)
PFont font = loadFont("Serif-48.vlw");

// We select the font
textFont(font);

// Color
fill(255, 0, 0);

// At x=10, y = 50 (bottom to top)
text("Hello World", 20, 100);
```

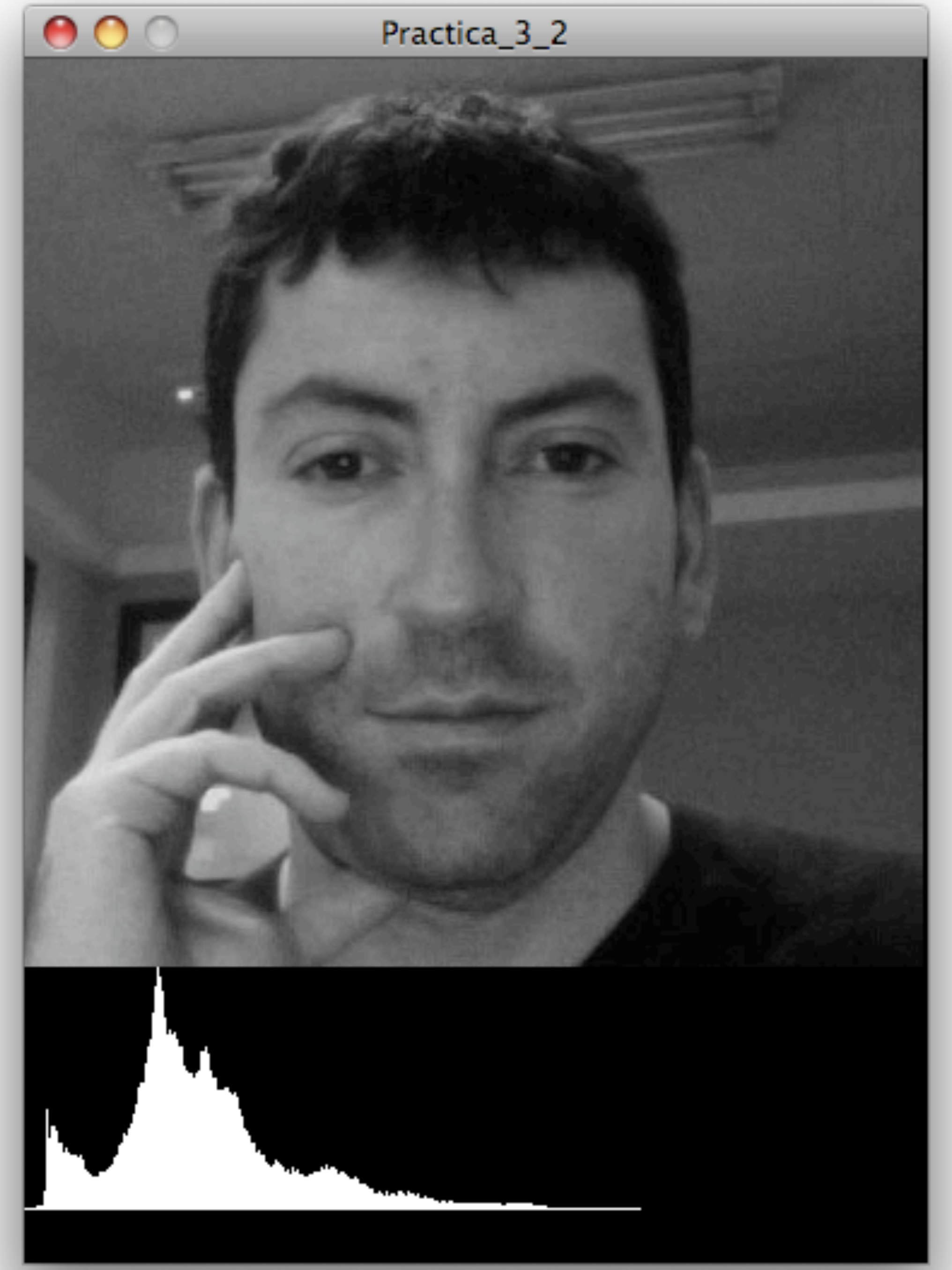


Practice 3-2

- Calculate and visualise the histogram of an image
- Histogram represents the number of times an intensity value is present on an image
- It can be calculated either for each of the three channels separately or in a integrated way for the 3 channels (RGB histogram)
- RGB histogram can be calculated as follows:
  - Open the image
  - Convert it to grayscale. You can use `filter()` method: `im.filter(GRAY)`, that modifies the image (`im`) transforming it to grayscale (1 channel). This only channel will now be accessible at the RED one: `red(im.get(x, y))` returns the gray value of the pixel after the transformation carried out by `filter()`
  - Create an array histogram: `int[] histogram = new int[256]`
  - Traverse all image pixels, RED channel, increasing its counter in the histogram: `histogram[red(im.get(x, y))]++`. With that, we count the number of times each value is present in the image. It is also convenient to calculate the maximum value of all saved in the histogram.

Practice 3-2

- Represent graphically this histogram, so its width will be 256 and its height will be 100, and each value will be drawn as a line
- Example:


Practica_3_2