WORKED EXAMPLE 4.2

Manipulating the Pixels in an Image

A digital image is made up of *pixels*. Each pixel is a tiny square of a given color. In this Worked Example, we will use a `Picture` type that has member functions for loading an image and accessing its pixels.

Cay Horstmann.

Problem Statement Your task is to convert an image into its negative, turning white to black, cyan to red, and so on. The result is a negative image of the kind that old-fashioned film cameras used to produce.

Step 1 We provide you with a `Picture` type for manipulating images in the `.png` format. In order to use this type, you need to copy the files `picture.cpp`, `picture.h`, `lodepng.cpp`, and `lodepng.h` into the same folder as your source file, and you need to include them in the compilation. You can find these files in the `ch04/worked_example_2` folder of the companion code.

You also need to include the `picture.h` file into your source file, with the statement

```
#include "picture.h"
```

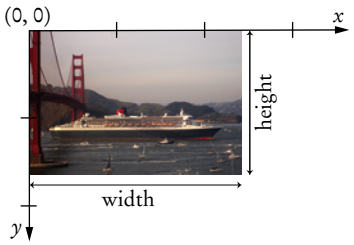
Note that the header file name, `picture.h`, is surrounded by quotation marks, not angle brackets, to indicate that you include a local file and not a system file.

To obtain a picture from a `.png` file, supply the file name like this:

```
Picture pic("queen-mary.png");
```

This statement defines a variable `pic` of type `Picture` that holds the pixels from the provided file.

Each pixel in the picture has an x- and y-coordinate, with $0 \le x <$ width and $0 \le y <$ height. The pixel with coordinates (0, 0) is in the top-left corner, and the y-axis points downward.



Each pixel has an RGB color value that is composed of the three primary colors: red, green, and blue. Each primary color amount is given as an integer between 0 (primary color not present) and 255 (maximum amount present). For example, a color with red value 255, green value 0, and blue value 255 is a bright purple color called magenta.

Step 2 Once you have a picture, you can call member functions to find out about the picture, or to modify it.

Here is what you can do:

- You can obtain the width and height of the picture by calling `pic.width()` and `pic.height()`.
- You can get the red, green, and blue values of a pixel at a particular location by calling `pic.red(x, y)`, `pic.green(x, y)`, and `pic.blue(x, y)`.
- You can set a pixel to any RGB color value by calling `pic.set(x, y, red, green, blue)`.
- You can save the changes to a file: `pic.save("result.png")`.
- You can add another picture to this picture—see Section 4.9.

Instead of loading a picture from an image file, you can also start with a monochromatic picture:

```
Picture all_magenta(300, 200, 255, 0, 255); // Specify width, height, and RGB color
```

Step 3 Now consider the task of converting an image into its negative.

Cay Horstmann.

A pixel is turned into its negative like this:

```
int red = pic.red(x, y);
int green = pic.green(x, y);
int blue = pic.blue(x, y);
pic.set(x, y, 255 - red, 255 - green, 255 - blue);
```

We want to apply this operation to each pixel in the image. To process all pixels, we can use one of the following two strategies:

For each row
  For each pixel in the row
    Process the pixel.

or

For each column
  For each pixel in the column
    Process the pixel.

Because our pixel class uses x/y coordinates to access a pixel, it turns out to be more natural to use the second strategy. (In Chapter 6, you will encounter two-dimensional arrays that are accessed with row/column coordinates. In that situation, use the first form.)

To traverse each column, the x-coordinate starts at `0`. Because there are `pic.width()` columns, we use the loop

```
for (int x = 0; x < pic.width(); x++)
```

Once a column has been fixed, we need to traverse all y-coordinates in that column, starting from `0`. There are `pic.height()` rows, so our nested loops are

```
for (int x = 0; x < pic.width(); x++)
{
   for (int y = 0; y < pic.height(); y++)
   {
      . . .
   }
}
```

The following program solves our image manipulation problem:

worked_example_2/negative.cpp

```
#include "picture.h"

int main()
{
   Picture pic("queen-mary.png");

   for (int x = 0; x < pic.width(); x++)
   {
      for (int y = 0; y < pic.height(); y++)
      {
         int red = pic.red(x, y);
         int green = pic.green(x, y);
         int blue = pic.blue(x, y);
         pic.set(x, y, 255 - red, 255 - green, 255 - blue);
      }
   }
   pic.save("out.png");
   return 0;
}
```