#include <stdio.h>

#include <stdlib.h>

#include <opencv/cv.h>

#include <opencv/highgui.h>

#define ALTOBLOQUE 32

#define ANCHOBLOQUE 32

void copiarBloque(int x0, int y0, IplImage\* imagenOrigen, int x1, int y1, IplImage\* imagenDestino) {

for (int fila = 0; fila < ALTOBLOQUE; fila++) {

char \*pImg1 = imagenOrigen->imageData + (fila + y0) \* imagenOrigen->widthStep + x0 \* imagenOrigen->nChannels;

char \*pImg2 = imagenDestino->imageData + (fila + y0) \* imagenDestino->widthStep + x0 \* imagenDestino->nChannels;

for (int col = 0; col < ANCHOBLOQUE; col++) {

\*pImg2++ = \*pImg1++;

\*pImg2++ = \*pImg1++;

\*pImg2++ = \*pImg1++;

}

}

}

int main(int argc, char\*\* argv) {

if (argc != 3) {

printf("Usage: %s image\_file\_name\n", argv[0]);

return EXIT\_FAILURE;

}

//CV\_LOAD\_IMAGE\_COLOR = 1 forces the resultant IplImage to be colour.

//CV\_LOAD\_IMAGE\_GRAYSCALE = 0 forces a greyscale IplImage.

//CV\_LOAD\_IMAGE\_UNCHANGED = -1

IplImage\* Img1 = cvLoadImage(argv[1], CV\_LOAD\_IMAGE\_COLOR);

IplImage\* Img2 = cvLoadImage(argv[2], CV\_LOAD\_IMAGE\_COLOR);

// Always check if the program can find a file

if (!Img1) {

printf("Error: fichero %s no leido\n", argv[1]);

return EXIT\_FAILURE;

}

if (!Img2) {

printf("Error: fichero %s no leido\n", argv[2]);

return EXIT\_FAILURE;

}

// a visualization window is created with title 'image'

cvNamedWindow(argv[1], CV\_WINDOW\_AUTOSIZE);

cvNamedWindow(argv[2], CV\_WINDOW\_AUTOSIZE);

// img is shown in 'image' window

cvShowImage(argv[1], Img1);

cvShowImage(argv[2], Img2);

cvWaitKey(0);

int b\_x = Img1->width / ANCHOBLOQUE;

int b\_y = Img1->height / ALTOBLOQUE;

int array[b\_x][b\_y];

memset(array, 0, sizeof(array));

int col, fila;

int i = 0;

while (i < b\_x \* b\_y) {

col = random() % b\_x;

fila = random() % b\_y;

if (!array[col][fila]) {

copiarBloque(col\*ANCHOBLOQUE, fila\*ALTOBLOQUE, Img1, col\*ANCHOBLOQUE, fila\*ALTOBLOQUE, Img2);

//cvShowImage(argv[1], Img1);

cvShowImage(argv[2], Img2);

cvWaitKey(1);

i++;

array[col][fila] = 1;

}

}

cvWaitKey(0);

// memory release for img before exiting the application

cvReleaseImage(&Img1);

cvReleaseImage(&Img2);

// Self-explanatory

cvDestroyWindow(argv[1]);

cvDestroyWindow(argv[2]);

return EXIT\_SUCCESS;

}