Computer Graphics

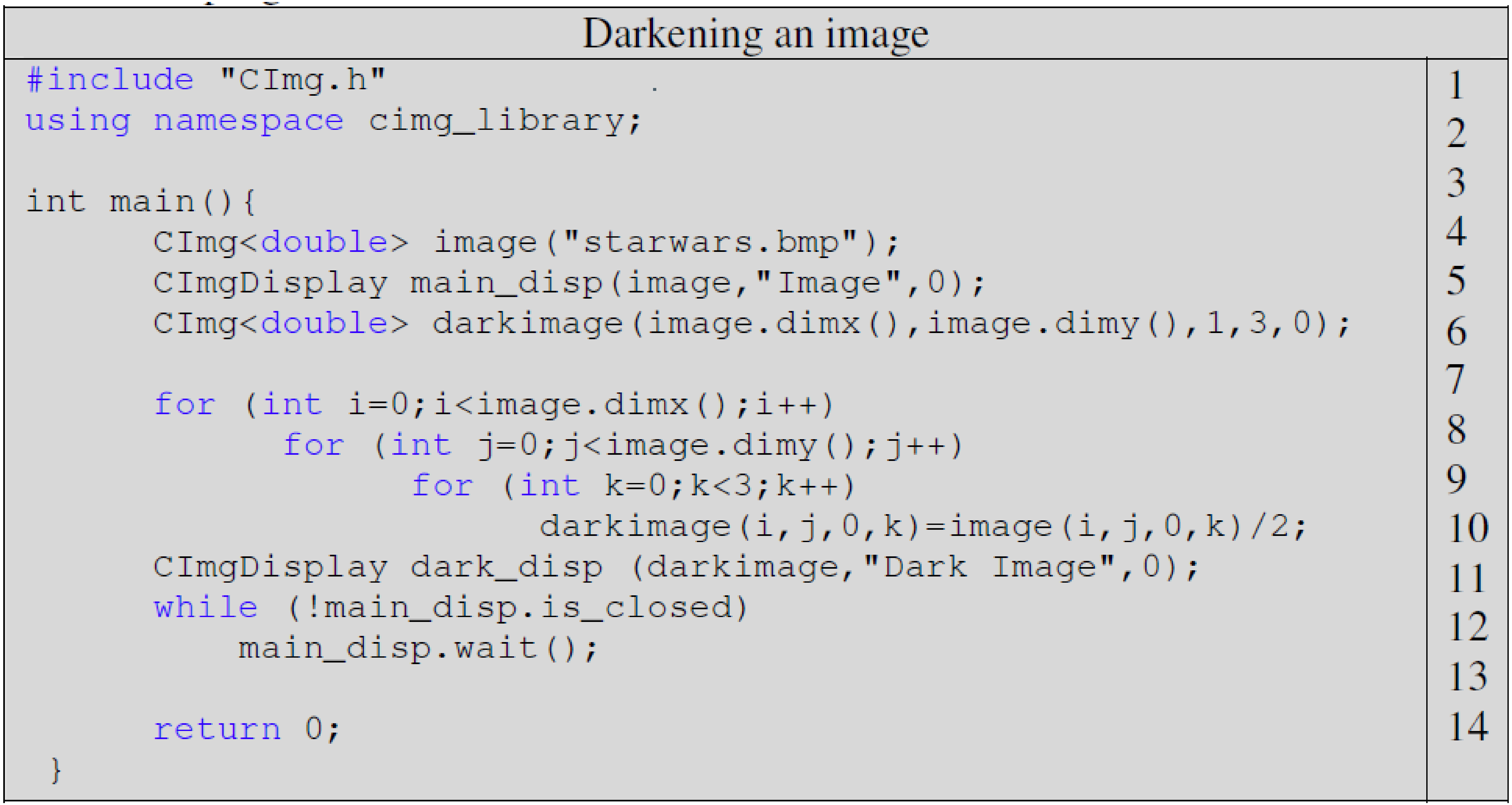
- Basic Image Programming via CImg

Junjie Cao @ DLUT

Spring 2017

http://jjcao.github.io/ComputerGraphics/

First example



Darkening an image

```
#include "CImg.h"
using namespace cimg_library;

int main(){
      CImg<double> image("starwars.bmp");
      CImgDisplay main_disp(image,"Image",0);
      CImg<double> darkimage(image.dimx(),image.dimy(),1,3,0);

      for (int i=0;i<image.dimx();i++)
            for (int j=0;j<image.dimy();j++)
                  for (int k=0;k<3;k++)
                        darkimage(i,j,0,k)=image(i,j,0,k)/2;
      CImgDisplay dark_disp (darkimage,"Dark Image",0);
      while (!main_disp.is_closed)
          main_disp.wait();

      return 0;
 }
```

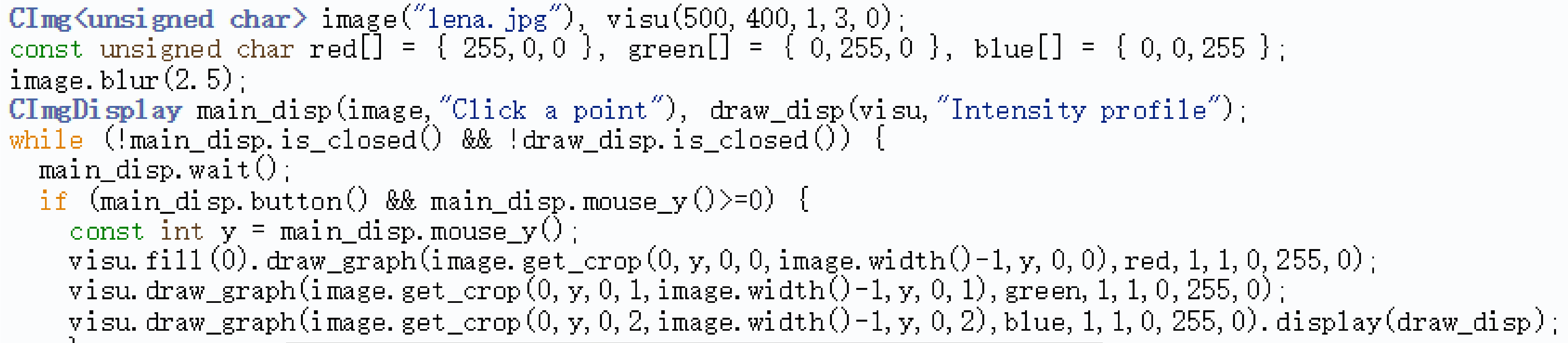
Declaration of Images

- #include "CImg.h"
- using namespace cimg_library;
- CImg<float> myEmptyFloatImg /* pure declaration of image (no size) */
- CImg<float> myFloatImg (300,200); /* 300x200 float pixel image */
- CImg<float> myFloatImg1 ("demo.png"); /* directly loading a file */
- CImg<float> myFloatImg2 (myFloatImg); /* copying a previous image */

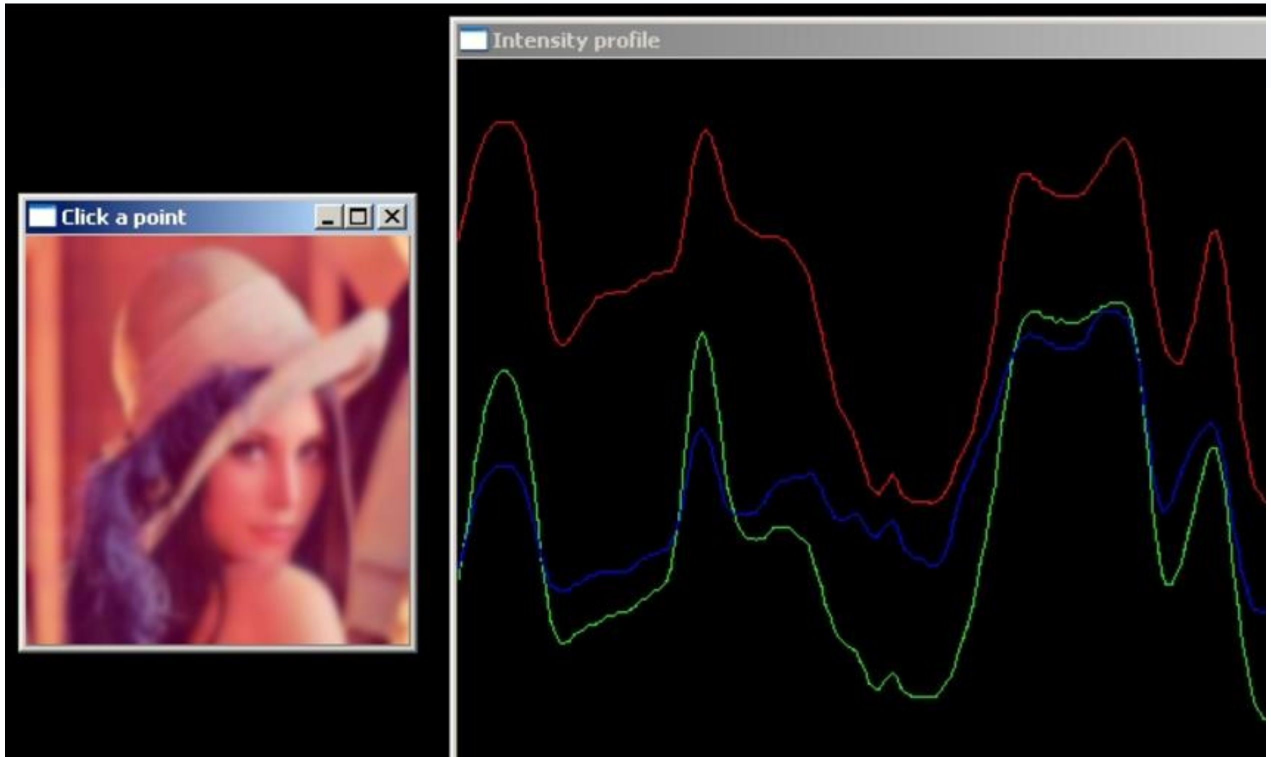
Loading of Images

- string testFilename="testImage.png";
- myFloatImg.load (testFilename.c_str()); /* loading from file */
- myFloatImg.save ("outputImage.png"); /* saving to file */

Image inspection



```
CImg<unsigned char> image("lena.jpg"), visu(500,400,1,3,0);
const unsigned char red[] = { 255,0,0 }, green[] = { 0,255,0 }, blue[] = { 0,0,255 };
image.blur(2.5);
CImgDisplay main_disp(image,"Click a point"), draw_disp(visu,"Intensity profile");
while (!main_disp.is_closed() && !draw_disp.is_closed()) {
  main_disp.wait();
  if (main_disp.button() && main_disp.mouse_y()>=0) {
    const int y = main_disp.mouse_y();
    visu.fill(0).draw_graph(image.get_crop(0,y,0,0,image.width()-1,y,0,0),red,1,1,0,255,0);
    visu.draw_graph(image.get_crop(0,y,0,1,image.width()-1,y,0,1),green,1,1,0,255,0);
    visu.draw_graph(image.get_crop(0,y,0,2,image.width()-1,y,0,2),blue,1,1,0,255,0).display(draw_disp);
    }
  }
```



Processing the image

- unsigned width = myFloatImg.dimx(); // access image width
- unsigned height = myFloatImg.dimy(); // access image height

- for (unsigned y=0; y < height; y++)
  - for (unsigned x=0; x < width; x++)
    - myFloatImg(x,y) = x*y;

How pixel data are stored with CImg

CImg<T>: 4th-dimensional array (width,height,depth,dim)

- stored linearly in a single memory buffer of general size (width*height*depth*dim)
- The address of this memory buffer: T *ptr = img.data()
- the ordering of the pixel values in this buffer follows these rules :
  - The values are *not* interleaved, and are ordered first along the X,Y,Z and V axis respectively (corresponding to the width,height,depth,dim dimensions)
  - starting from the upper-left pixel to the bottom-right pixel of the image

ordering of pixel values in buffer

CImg<T>: 4th-dimensional array (width,height,depth,dim)

A color image with dim=3 and depth=1

- R1R2R3R4R5R6......G1G2G3G4G5G6.......B1B2B3B4B5B6....

- *not* as R1G1B1R2G2B2R3G3B3... (interleaved channels)

- R1 = img(0,0,0,0) is the first upper-left pixel of the red component of the image,
- R2 is img(1,0,0,0)
- G1 = img(0,0,0,1), G2 = img(1,0,0,1)
- B1 = img(0,0,0,2)

ordering of pixel values in buffer

- CImg<T>: 4th-dimensional array (width,height,depth,dim)

- a (1x5x1x1) CImg<T> (column vector A) will be stored as :
- A1A2A3A4A5
- where A1 = img(0,0), A2 = img(0,1), ... , A5 = img(0,4)

ordering of pixel values in buffer

- CImg<T>: 4th-dimensional array (width,height,depth,dim)
- R1R2R3R4R5R6......G1G2G3G4G5G6.......B1B2B3B4B5B6....

- a 2D color image is stored in memory exactly as a 3D scalar image having a depth=3
- you can write 'img(x,y,k)' instead of 'img(x,y,0,k)' to access the kth channel of the (x,y) pixel.

Suggestions