EECS 22: Advanced C Programming
Week 7

Mihnea Chirila
mchirila@uci.edu

11/10/2017

Outlines

- Introduction to Assignment 4
- Assignment Setup
- Dynamic Memory Allocation
- Modify Existing Functions
- New Image Processing Function - Crop

Assignment 4

- Digital Image Processing Program for varying image size
  - In Assignment 2 & 3, your DIP program could only work with images of fixed size (600 x 400).
  - Now, you need to redesign your program to accommodate varying image sizes.
  - The input image or the result image may differ in size.
  - Based on Assignment 3, you need to use new image structure and pixel mapping functions in your image processing operations.
- Due: Wednesday 11/22/2017 at 6:00pm

Assignment Setup

- Copy the following files to your directory:

```
mkdir hw4
cd hw4
cp ~eecs22/public/FileIO.h .
cp ~eecs22/public/FileIO.c .
cp ~eecs22/public/Test.c .
cp ~eecs22/public/Test.h .
cp ~eecs22/public/Image.h .
cp ~eecs22/public/HSSOE.ppm .
cp ~eecs22/public/watermark_template.ppm .
```

- `Image.h`: the header file for the definition of the new image structure and declarations of the pixel mapping functions
- `FileIO.h` & `FileIO.c`: new header file and source file for File I/Os (for varying image sizes)

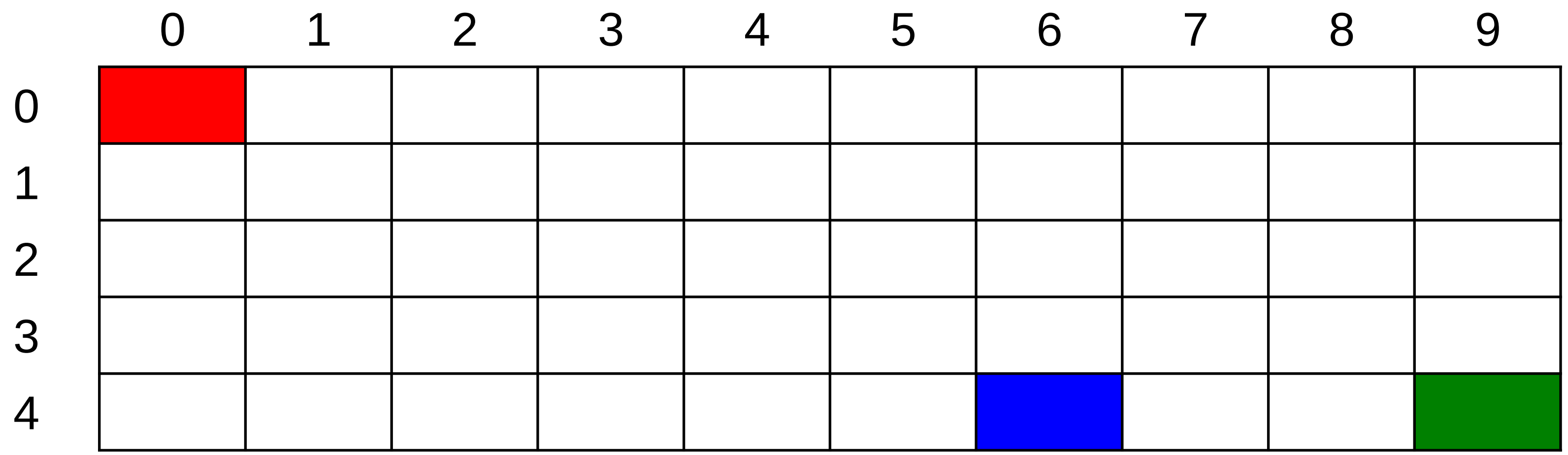
Assignment Setup

- In addition to the previous files, you also need to reuse some of files in Assignment 3. You can copy from your `hw3/` or the shared folder:

```
cp ~eecs22/public/PhotoLab_v3.c .
cp ~eecs22/public/Constants.h .
cp ~eecs22/public/DIPs.h .
cp ~eecs22/public/DIPs.c .
cp ~eecs22/public/Advanced.h .
cp ~eecs22/public/Advanced.c .
cp ~eecs22/public/Makefile .
```

- You need to modify the existing DIP functions by using the new pixel mapping functions (`GetPixelR`, `SetPixelR`, and etc.) and implement four new image processing functions (Crop, Resize, Brightness&Contrast and Watermark).

Pointers in 1-D Memory Space



- In previous assignments, we always use three 2-D unsigned char arrays to store the intensity values of the image.
- In this assignment, we will use three 1-D memory space since the image size is unknown until we run the program.
- 2-D Array Index / 1-D Memory Index

2-D Array Index	1-D Memory Index
(0, 0)	0 + 0 * 10 = 0
(9, 4)	9 + 4 * 10 = 49
(6, 4)	6 + 4 * 10 = 46
(x, y)	x + y * WIDTH

IMAGE struct

```
typedef struct {
    unsigned int W;
    unsigned int H;
    unsigned char *R;
    unsigned char *G;
    unsigned char *B;
} IMAGE;
```

- `Width` & `Height` are the width and height of the image.
- `R`, `G` and `B` are pointers to memory storing R, G and B intensity values.
- Use `malloc()` and `free()` to allocate and deallocate the memory space pointed to by `R`, `G` and `B`.

Dynamic Memory Allocation

```
#include <stdlib.h>
void *malloc(size_t size);
```

- Allocate `size` bytes of memory space on the heap
  - Allocated memory space is uninitialized.
- Returns a pointer to the memory (address of first byte)
  - Return type is `void*`, meaning “pointer to unknown type”
  - Return value is `NULL` if requested size could not be allocated

```
void free(void *p);
```

- Deallocates the memory at address `p`
  - Argument `p` must be a pointer to space allocated by `malloc()`
  - Does nothing if `p` is `NULL`

- Advise:
  - Always check the return value of `malloc()`!
  - Always use `malloc()` and `free()` in pairs!

ImageFunctions

```
unsigned char GetPixelR(IMAGE *image, unsigned int x,
                        unsigned int y);
unsigned char GetPixelG(IMAGE *image, unsigned int x,
                        unsigned int y);
unsigned char GetPixelB(IMAGE *image, unsigned int x,
                        unsigned int y);

void SetPixelR(IMAGE *image, unsigned int x, unsigned int y,
               unsigned char r);
void SetPixelG(IMAGE *image, unsigned int x, unsigned int y,
               unsigned char g);
void SetPixelB(IMAGE *image, unsigned int x, unsigned int y,
               unsigned char b);
```

- Implement these functions to get and set the intensity values of each pixel in the image in `Image.c`

ImageFunctions

```
/* Return the image's width in pixels */
unsigned int ImageWidth(IMAGE *image);

/* Return the image's height in pixels */
unsigned int ImageHeight(IMAGE *image);
```

- Implement these functions to get the Width and Height values of the image in `Image.c`.
- Use assertions to make sure the input is valid.
- Extend `Makefile` to generate `Image.o` and add `Image.o` when generating `PhotoLab` and `PhotoLabTest`.

File I/Os

```
IMAGE *LoadImage(const char *fname);
int SaveImage(const char *fname,
                  IMAGE *image);
```

- `LoadImage` reads the file `fname.ppm`, creates the memory space of the image (`R`, `G` and `B`), stores the color intensities in the memory space, and returns the image pointer (or `NULL` if error happens).
- `SaveImage` saves the color intensities to the file `fname.ppm` and deallocate the memory space of the image.
- The above two functions depends on the following two functions to handle memory allocation and deallocation, which you need to implement in `Image.c`:
  - `IMAGE *CreateImage(unsigned int Width, unsigned int Height);`
  - `void DeleteImage(IMAGE *image);`

Modify Existing Function - BlackNWhite

- **void** BlackNWhite(**unsigned char** R[WIDTH][HEIGHT], **unsigned char** G[WIDTH][HEIGHT], **unsigned char** B[WIDTH][HEIGHT])

```
for (y = 0; y < HEIGHT; y++)
{
        for (x = 0; x < WIDTH; x++)
      {
         tmp = (R[x][y] + G[x][y] + B[x][y]) / 3;
         R[x][y] = G[x][y] = B[x][y] = tmp;
      }
}
```

- **IMAGE** *BlackNWhite(**IMAGE** *image)

```
     for (y = 0; y < ImageHeight(image); y++)
     {
          for (x = 0; x < ImageWidth(image); x++)
          {
               tmp = (GetPixelR(image, x, y) + GetPixelG(image, x, y) +
GetPixelB(image, x, y)) / 3;
               SetPixelR(image, x, y, tmp);
                SetPixelG(image, x, y, tmp);
                SetPixelB(image, x, y, tmp);
          }
     }
```

Modify Existing Functions

- `IMAGE *BlackNWhite(IMAGE *image);`
- `IMAGE *Negative(IMAGE *image);`
- `IMAGE *ColorFilter(IMAGE *image, int target_r, int target_g, int target_b, int threshold, int replace_r, int replace_b, int replace_b);`
- `IMAGE *Edge(IMAGE *image);`
- `IMAGE *Shuffle(IMAGE *image);`
- `IMAGE *VFlip(IMAGE *image);`
- `IMAGE *VMirror(IMAGE *image);`
- `IMAGE *AddBorder(IMAGE *image, char color[SLEN], int border_width);`
- `IMAGE *AddNoise(int n, IMAGE *image);`
- `IMAGE *Sharpen(IMAGE *image);`
- `IMAGE *Posterize(IMAGE *image, unsigned int rbits, unsigned int gbits, unsigned int bbits);`
- `IMAGE *MotionBlur(int bluramount, IMAGE *image);`
- `void AutoTest(IMAGE *image);`

Main Menu

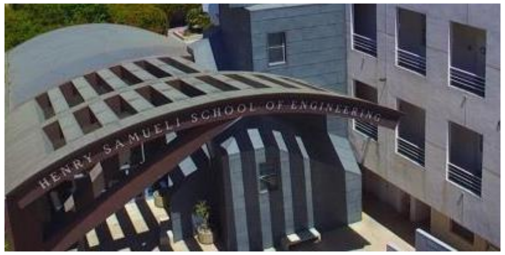
```
--------------------------------
1: Load a PPM image
2: Save an image in PPM and JPEG format
3: Change a color image to Black and White
4: Make a negative of an image
5: Color filter an image
6: Sketch the edge of an image
7: Shuffle an image
8: Flip an image vertically
9: Mirror an image vertically
10:Add border to an image
11:Add noise to an image
12:Sharpen an image
13:Posterize an image
14:Blur an image
15:Crop
16:Resize
17:Brightness and Contrast
18:Watermark
19:Test all functions
20:Exit
Please make your choice:
```

Assignment 2

Assignment 3

Assignment 4

Crop



- `IMAGE *Crop(IMAGE *image, unsigned int x, unsigned int y, unsigned int W, unsigned int H);`
- Crop an image starting from `(x, y)` and the crop width and height are `W` and `H` respectively.
- Only crop up to the maximum length of the original image.