

Special Effects

COMP0015 Term 1 Coursework 4 – 15% of the module

This document explains the arrangements for the coursework. You will create an application for modifying images in PPM format according to the specification set out in this document.

Deadline

9.00am 23rd November 2020.

Aim

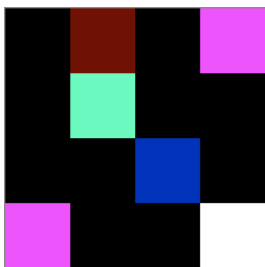
The aim of this coursework is for you to demonstrate that you can use and manipulate 2D arrays by writing the functions specified in this brief. You are given code for the application menu and for reading and writing images. You do not need to change this code.

Overview

Most people are familiar with the fact that a digital image is a grid comprised of thousands of pixels with each pixel containing a single colour. The number of pixels in an image determines the definition of the image. In this coursework you will process an image pixel by pixel and transform it in some way to produce some simple effects.

We will be using a very simple image format called PPM (or Portable Pix Map). PPM images are not really used any more but they're useful to us because a PPM image is encoded as human-readable ASCII text. For those of you who want to know more, the formal image specification can be found [here](#). It's not necessary to read it this for the coursework.

Here is the file tinypix.ppm and the corresponding image. It has two parts:

File content	Explanation
<pre>P3 4 4 255</pre>	This part of the image is the header. The second row is important, it contains the number of rows and columns in the image.
<pre>0 0 0 100 0 0 0 0 0 255 0 255 0 0 0 0 255 175 0 0 0 0 0 0 0 0 0 0 0 0 0 15 175 0 0 0 255 0 255 0 0 0 0 0 0 255 255 255</pre> 	<p>This part represents the pixels. The image has 4 x 4 (16) pixels. Each pixel has a red, green and blue (r, g, b) value.</p> <p>0, 0, 0 means no colour (black) and 255, 255, 255 is the maximum level of a colour (white). Take a look at the RGB calculator to experiment with RGB values here.</p>

Keep in mind, each square is one pixel, so the real thing is much smaller (this image was blown up by 5000%). You will find that you cannot see this image in a viewer.

Viewing PPM files

You will need some software on your computer to:

- View a plain text file
- View an image in PPM format

It's likely you have some choices here. Here's some suggestions:

Software	View plain text	View image	Platforms
Any text editor such as Notepad, atom.	✓	No	All
IDLE	✓	No	Windows, MacOS
Visual Studio	✓	No	Windows, MacOS
IrfanView https://www.irfanview.com/	No	✓	Windows
Gimp https://www.gimp.org/	No	✓	Windows, MacOS UCL Desktop
Website: https://www.kylepaulsen.com/stuff/NetpbmViewer/	You paste in the file contents and the image is rendered on the page.	✓	
Codio	✓	✓	Use the feh command from a terminal (see below)

Viewing the images on Codio

To view any image do this:

1. Open a terminal
2. At the command prompt type `feh python.ppm` (substitute your filename here).
3. Click on the 'Viewer' menu option to see the output.
4. Close the window by clicking on the X in the top right corner of the viewer.

How to use the images

The folder with the starter code contains several images. Images can be very large with thousands of pixels. It's not a good idea to start writing and testing your code with large images because it's difficult to test whether your algorithms are working correctly. For this reason, there are several small images, including `tinypix.ppm` which is only 16 pixels in size. There is also a file `one_pixel.ppm` which only contains 1 pixel. The recommendation is that you should test that your code works with the smaller images by examining the resulting file that your code created in a text editor. You can't really see `tinypix.ppm` in the viewers listed above because it's just too small but this isn't important. When your code is marked, we will just use some very tiny images in our tests for your functions.

Your Assignment

You are provided with some starter code and some sample pictures which you must download and save before starting the assignment. The pictures are all in ppm format.

Functions

The skeleton code contains the empty functions: `negative()`, `rotate_left_90()`, `crop()` and `blur()`; you must complete them according to the instructions here and the specification in the docstrings (comments) at the start of the functions.

(a) Function `negative(image_list)`

This function returns a 2D array containing the new image pixels. A negative image contains pixels in which the red, green and blue values have been subtracted from 255. If a pixel value is `[255, 34, 100]` then the negative pixel will be: `[0, 221, 155]`. For the purposes of this coursework, your function should create a new 2D list which contains the negative image.



Image: python.ppm	Negative image
	

Table 1 Python logo - Wikimedia

(b) Function `rotate_left_90(image_list)`

In this function you will, as the function name suggests, rotate the image `image_list` by 90 degrees, rotating it to the left. The source image may have a different width and height as shown in the example below. Here is an example to show how the transformation works. The pixels are represented by letters to make it clear what is happening.

	0	1	2	3	4	5
0	A	B	C	D	E	F
1	G	H	I	J	K	L
2	M	N	O	P	Q	R
3	S	T	U	V	W	X

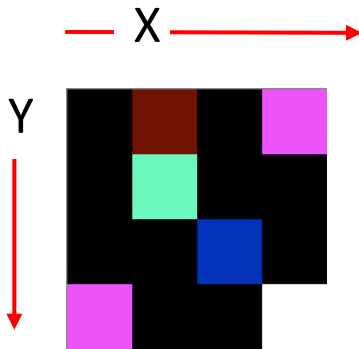
Table 2 - before `rotate_left_90`

	0	1	2	3
0	F	L	R	X
1	E	K	Q	W
2	D	J	P	V
3	C	I	O	U
4	B	H	N	T
5	A	G	M	S

Table 3 - after `rotate_left_90`

(c) Function `crop(image_list, crop_X, crop_Y, crop_width, crop_height)`

This function returns a 2D array containing the image pixels. Cropping an image involves reducing the size of the image so that the top-left pixel of new image is the pixel in the old image with coordinates `crop_X` and `crop_Y`. The `crop_width` and the `crop_height` are the width and height of the new image in pixels.



For example, given this image:

```
[[[0, 0, 0], [100, 0, 0], [0, 0, 0], [255, 0, 255]],
 [[0, 0, 0], [0, 255, 175], [0, 0, 0], [0, 0, 0]],
 [[0, 0, 0], [0, 0, 0], [0, 15, 175], [0, 0, 0]],
 [[255, 0, 255], [0, 0, 0], [0, 0, 0], [255, 255, 255]]]
```

The call `crop(image_list, 1, 2, 3, 2)` will return:

```
[[[0, 0, 0], [0, 15, 175], [0, 0, 0]],
 [[0, 0, 0], [0, 0, 0], [255, 255, 255]]]
```

You may assume that the new image bounds lie entirely within the bounds of the original image.

(d) Function `blur(image_list)`

In this function you will take an image and implement a blur filter. If you want to view the effect, be aware that the function works best on black and white images. The algorithm to do this creates a new image in which a pixel's values are averaged with the values of the immediate neighbours. You can consider that a pixel has eight neighbours:

	0	1	2	3
0	[0, 0, 0]	[100, 0, 0]	[0, 0, 0]	[255, 0, 255]
1	[0, 0, 0]	[0, 255, 175]	[0, 0, 0]	[0, 0, 0]
2	[0, 0, 0]	[0, 0, 0]	[0, 15, 175]	[0, 0, 0]
3	[255, 0, 255]	[0, 0, 0]	[0, 0, 0]	[255, 255, 255]

The pixel at position (1,2) will contain the average of all pixels from (0,0) to (2,2). This includes the values in the pixel itself. The result must be rounded down to the nearest integer:

- The new red value in the pixel value generated will be: $(0 + 100 + 0 + 0 + 0 + 0 + 0 + 0 + 0) / 9 = 11$
- The new green value in the pixel value generated will be: $(0 + 0 + 0 + 0 + 255 + 0 + 0 + 0 + 15) / 9 = 30$
- The new blue value in the pixel value generated will be: $(0 + 0 + 0 + 0 + 175 + 0 + 0 + 0 + 175) / 9 = 39$

Note: Use the `new_image` variable in the function, do not modify the pixel values in place. All pixel values must be integers.

Pixels at the edge of the image need to be treated slightly differently. In this example, consider the pixel at (1,0). The average has fewer data points and the new pixel generated for the position (1,0) will be:

- red value: $(0 + 100 + 0 + 0 + 0 + 0) / 6 = 16$
- green value: $(0 + 0 + 0 + 0 + 255 + 0) / 6 = 42$
- blue value: $(0 + 0 + 0 + 0 + 175 + 0) / 6 = 29$
- pixel value is [16, 42, 29]

	0	1	2	3
0	[0, 0, 0]	[100, 0, 0]	[0, 0, 0]	[255, 0, 255]
1	[0, 0, 0]	[0, 255, 175]	[0, 0, 0]	[0, 0, 0]
2	[0, 0, 0]	[0, 0, 0]	[0, 15, 175]	[0, 0, 0]
3	[255, 0, 255]	[0, 0, 0]	[0, 0, 0]	[255, 255, 255]

In this example, consider the pixel at (3, 3). The average has even fewer data points and the new pixel generated for the position (3,3) will be:

- red value: $(0 + 0 + 0 + 255) / 4 = 63$
- green value: $(15 + 0 + 0 + 255) / 4 = 67$
- blue value: $(175 + 0 + 0 + 255) / 4 = 238$
- pixel value is [63, 67, 238]

	0	1	2	3
0	[0, 0, 0]	[100, 0, 0]	[0, 0, 0]	[255, 0, 255]
1	[0, 0, 0]	[0, 255, 175]	[0, 0, 0]	[0, 0, 0]
2	[0, 0, 0]	[0, 0, 0]	[0, 15, 175]	[0, 0, 0]
3	[255, 0, 255]	[0, 0, 0]	[0, 0, 0]	[255, 255, 255]

Here is an example of the blur effect. It's quite a gentle effect:



Image: tijitu.ppm	Blurred image
	

Table 2 Tijitu - Wikimedia

Testing:

You are responsible for testing your program carefully. Make sure that you have thought about all the things that can go wrong and test your program to ensure that you know it works correctly in all cases.

Submitting your assignment

At the submission link on Codio:

1. Make sure your student number (not your name) is included in comments at the top of your program.
2. Upload your program.
3. If you have included any additional functionality not specified in the brief, please submit a short text document on Codio describing what you have done. Keep this short, it is not graded. The purpose of the document is to guide the marking.

Assessment

You are expected to show that you can code competently using the programming concepts covered so far in the course up to the topics lists and strings.

Marking criteria will include:

- Correctness – your code must perform as specified
- Programming style – your variable names should be meaningful and your code as simple and clear as possible. See section 'Style Guide' for more detail.
- Your assignment will be marked using the rubric at the end of this document. This is the standard rubric used in the Department of Computer Science. Marks for your project work will be awarded for the capabilities (i.e. functional requirements) your system achieves, and the quality of the code.

Additional Challenges

- Additional marks may also be gained by taking on extra challenges but you should only attempt an additional challenge if you have satisfied all requirements for the coursework.
- It's up to you what you choose to do, if anything.
- Note: You are strongly encouraged to follow the specification carefully and to use programming techniques as described in the course materials and textbooks. Poor quality code with additional functionality will not improve your marks.

Plagiarism

Plagiarism will not be tolerated. Your code will be checked using a plagiarism detection tool.

Style Guide

You must adhere to the style guidelines in this section.

Formatting Style

1. Use Python style conventions for your variable names (snake case: lowercase letters with words separated by underscores (_) to improve readability).
2. Name constants properly.

3. Choose good names for your variables. For example, `num_bright_spots` is more helpful and readable than `nbs`.
4. Use a tab width of 4 or 8. The best way to make sure your program will be formatted correctly is never to mix spaces and tabs -- use only tabs, or only spaces.
5. Put a blank space before and after every operator. For example, the first line below is good but the second line is not:

```
b = 3 > x and 4 - 5 < 32
```

```
b= 3>x and 4-5<32
```

6. Each line must be less than **80 characters** long *including tabs and spaces*. You should break up long lines using `\`.
7. Functions should be no longer than about 12 lines in length. Longer functions should be decomposed into 2 or more smaller functions.

Docstrings

If you add your own functions you should comment them using docstrings. Take a look at the code you've been given for some examples. Your comments should:

1. Describe precisely *what* the function does.
2. Do not reveal *how* the function does it.
3. Make the purpose of every parameter clear.
4. Refer to every parameter by name.
5. Be clear about whether the function returns a value, and if so, what.
6. Explain any conditions that the function assumes are true. Examples: "n is an int", "n != 0", "the height and width of p are both even."
7. Be concise and grammatically correct.
8. Write the docstring as a command (e.g., "Return the first ...") rather than a statement (e.g., "Returns the first ...")

Acknowledgements

A variation of this coursework was published in [Nifty Assignments](#).

UCL Computer Science: Marking Criteria and Grade Descriptors



	Fail		Pass (2:2)		Merit (2:1)		Distinction (1 st)	
	Inadequate	Weak	Satisfactory		Good		Excellent	Exceptional
	Below 40: BSc: Fail MEng: Fail	40-49: BSc: 3rd MEng: Fail	50-54: Low pass	55-59: High pass	60-64: Low merit	65-69: High merit	70-79	80-89 90+
1 Quality of the response to the task set: answer, structure and conclusions	Either no argument or argument presented is inappropriate and irrelevant. Conclusions absent or irrelevant.	An indirect response to the task set, towards a relevant argument and conclusions.	A reasonable response with a limited sense of argument and partial conclusions.		A sound response with a reasonable argument and straightforward conclusions.		A distinctive response that develops a clear argument and sensible conclusions, with evidence of nuance.	Exceptional response with a convincing, sophisticated argument with precise conclusions.
2 Understanding of relevant issues	Misunderstanding of the issues under discussion.	Rudimentary, intermittent grasp of issues with confusions.	Reasonable grasp of the issues and their broader implications.		Sound understanding of issues, with insights into broader implications.		Thorough grasp of issues; some sophisticated insights.	Exceptional grasp of complexities and significance of issues.
3 Engagement with related work, literature and earlier solutions	Very limited or irrelevant reading.	Significant omissions in reading with weak understanding of literature consulted.	Evidence of relevant reading and some understanding of literature consulted.		Evidence of plentiful relevant reading and sound understanding of literature consulted.		Extensive reading and thorough understanding of literature consulted. Excellent critical analysis of literature.	Expert-level review and innovative synthesis (to a standard of academic publications).
4 Analysis: reflection, discussion, limitations	Erroneous analysis. Misunderstanding of the basic core of the taught materials. No conceptual material.	Analysis relying on the partial reproduction of ideas from taught materials. Some concepts absent or wrongly used.	Reasonable reproduction of ideas from taught materials. Rudimentary definition and use of concepts.		Evidence of student's own analysis. Concepts defined and used systematically/effectively.		Evidence of innovative analysis. Concepts deftly defined and used with some sense of theoretical context.	Exceptional thought and awareness of relevant issues. Sophisticated sense of conceptual framework in context.
5 Algorithms and/or technical solution	No solution to the given problem, completely incorrect code for the given task.	Rudimentary algorithmic/technical solution, but mostly incomplete.	Reasonable solution, using basic required concepts, several flaws in implementation.		Good solution, skilled use of concepts, mostly correct and only minor faults.		Excellent algorithmic solution, novel and creative approach.	Exceptional solution and advanced algorithm/technical design.
6 Testing of solution (e.g., correctness, performance, evaluation)	No testing or evaluation done.	Few test cases and/or evaluation, but weak execution.	Basic testing done, but important test cases or parts of evaluation missing or incomplete.		Solid testing or evaluation of solution, well done evaluation with good summary of findings.		Very well done test cases, excellent evaluation and very high quality summary of findings.	Exceptionally comprehensive testing, extremely thorough approach to testing and/or evaluation.
7 Oral presentation or demonstration of solution	Poorly done presentation or demonstration, very low quality.	Ineffective oral presentation or demo of the solution.	Able to communicate, present and/or demonstrate solution and summarise work in appropriate format.		Overall good presentation or demo, persuasive and compelling.		Very high quality of delivery. Use of presentation medium with professional style.	Flawless and polished presentation, exceptional quality of demonstration.
8 Writing, communication and documentation	Style and word choice seriously interfere with comprehension.	Style and word choice seriously detract from conveying of ideas.	Style and word choice sometimes detract from conveying of ideas.		Style and word choice work well to convey most important ideas. Well documented.		Style and word choice show fluency with ideas and excellent communication skills.	Reads as if professionally copy edited. Exceptional high quality of writing.
9 Formatting aspects, visuals, clarity, references	Poorly formatted, inappropriate visuals, and incorrect reference formatting.	Formatting, visuals and referencing seriously distract from argument.	Formatting, visuals and referencing sometimes distract from argument.		Formatting well-done and consistent, good visuals and consistent referencing.		Formatting, visuals and referencing are impeccable.	Exceptional presentation, impeccable formatting of the document and references.

1-19: Misunderstanding of assignment or similar
20-29: 5 inadequate
30-39: 4 inadequate
34-39: 3 inadequate