Lab 16: Images in Processing

This lab is graded and is worth 3% of the cs7029 mark.

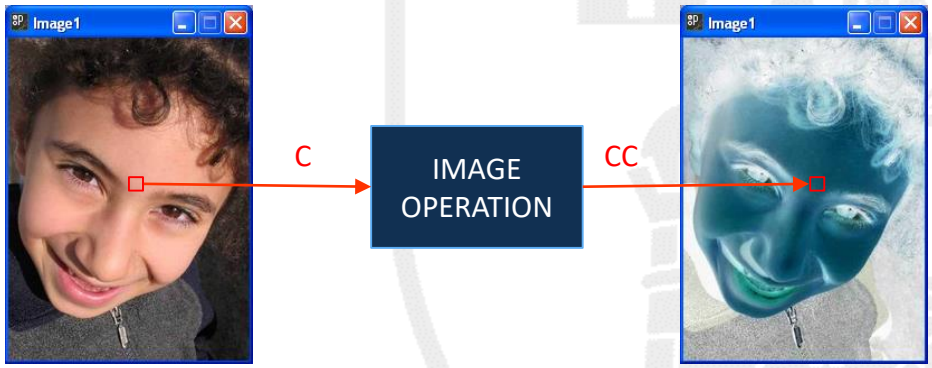
You should submit your solution on mymodule.tcd.ie on or before 11:59pm on 06/03/2017.

The lab is expected to take approximately 2-4 hours to complete (including the lab class).

Please note that late submissions will incur a penalty (mark will be capped at 40%)

In this lab, you will…

- Load an Image from File
- Retrieve pixels from the image
- Draw a modified version of the image on screen (see next page)



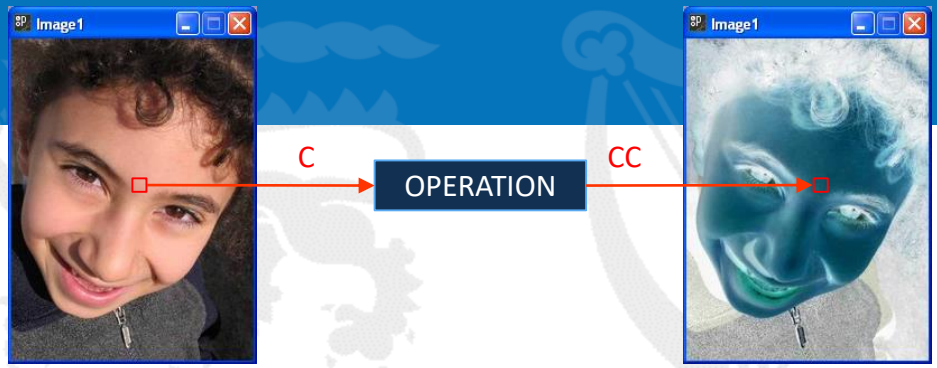
IMPORTANT NOTE: The following slides walk you through most of what you need to do. When you see three questions marks "???", you need to replace this with some value (you have to figure out what works for you).

Specific Objectives:

Implement the following Image operations

1. **Basic operations: 33%**
   - Grayscale the image {solution provided as an example}
   - Threshold the image to different values
   - Invert the colours of image
   - Mirror or flip the image
2. **Stylise the image: 33%**
3. **Upto 33% for any improvements/personalizations beyond the lab notes to either of the above**
   - Personalize stroke size/style
   - Randomize the distribution of points
   - Interactive/animated: e.g. drag mouse to change the threshold value
   - Ensure solution fully works and runs
   - Options are combined into one program. For instance, you can switch between operations with a keypress or mouse click. Or multiple operations applied to a single image.

N.B. You should not to use the filter() function for this lab



Sample Code

- **The following sample code can be used for all of the basic operations.**
- What we essentially want to do in image processing is calculate a new colour (let's call it **`newcolor`**)* for each pixel.
- This is based on the original colour (call it **`oldcolor`**) and some pixel operations which we will define.

```
PImage im;
void setup()
{
 im=loadImage(" ??? ");
 size(??? , ???);
 noLoop();
}

void draw()
{
    color oldcolor, newcolor;
     for(int x=0; x<im.width; x++)
     for(int y=0; y<im.height; y++)
    {
      oldcolor = im.get(x,y);
      //nc = ? Provide operation here
      set (x, y, newcolor);
    }
}
```

REPLACE THIS LINE :: Provide additional code into this sample program to get the desired effect. (see lecture)

*NOTE: **`oldcolor`** and **`newcolor`** are just variable names. You can call them anything you want

0. Setting up

- Start a new sketch with at least a **`setup`** and **`draw`** function

  OPTIONAL: Use **`noLoop()`** *to prevent constantly updating the screen (in this lab we're interested in just drawing once)*

- Find an image file (ideally not bigger than the resolution of your screen), save it to your sketch folder
- Create a window of approximately the same resolution as your image
- Load the image

  OPTIONAL: Check that you've loaded it correctly by drawing it using the **`image(…)`** *function. Then delete this before further steps (as we want to draw it differently)*

```
PImage im;
void setup()
{
 im=loadImage("download.jpg");
 size(800, 800);
 noLoop();
}

void draw()
{
   image(im, 0, 0);




}
```

1. Iterate over all the pixels

- **You need two nested* for loops to iterate across the image**
  - Basically this says go across every column of pixels one by one, and for each of these go across each row

    Here, the current column is stored as **x** and the current row is stored as **y**

    (x and y are just variable names we made up, you can rename them to anything you want)

    NOTE that the outer loop continues while x is less than the width of the image. While the inner loop continues

```
PImage im;
void setup()
{
 im=loadImage("download.jpg");
 size(800, 800);
 noLoop();
}

void draw()
{
  for(int x=0; x<im.width; x=x+1)
  for(int y=0; y<im.height; y=y+1)
   {




   }
}
```

*: a nested loop is essential a loop within a loop

2. Retrieve colors from original image

- **Retrieve each pixel using the get(…) function.**
  - To get the pixel from an image prefix it with the variable name of the image (im).
- **Store the retrieved color in a `color` object**

  Here we use a variable that we call `oldcolor` (just a variable name, you can rename this to anything you wish)

```
PImage im;
void setup()
{
 im=loadImage("download.jpg");
 size(800, 800);
 noLoop();
}

void draw()
{
  for(int x=0; x<im.width; x=x+1)
  for(int y=0; y<im.height; y=y+1)
   {
     color oldcolor;
     oldcolor = im.get(x,y);



   }
}
```

3. Drawing a retrieved color to the screen

- **In order to change the color returned by get(..), we need to look at its components i.e. its red, green and blue values**
- **These can be extracted using the functions red(...) green (…) and blue (…)**
- **Extract these and store them into variables so that you can change them later**
  - here we use r, g, and b (but these are just variable names we chose)
- **Use set to change the colour of a pixel on the screen at a certain location**
  - Note that set is not prefixed with im

```
PImage im;
void setup()
{
 im=loadImage("download.jpg");
 size(800, 800);
 noLoop();
}

void draw()
{
  for(int x=0; x<im.width; x=x+1)
  for(int y=0; y<im.height; y=y+1)
   {
     color oldcolor;
     oldcolor = im.get(x,y);

     float r = red(oldcolor);
     float g = green(oldcolor);
     float b = blue(oldcolor);
     color newcolor = color (r, g, b);
     set(x,y, newcolor);
   }
}
```

4. Modifying the pixels

- **In this example the image has been grayscaled**
- **You need to change the program to do the following (the first two are discussed in lectures)**
  - Threshold the image
  - Invert the colours
  - Mirror or Flip the image

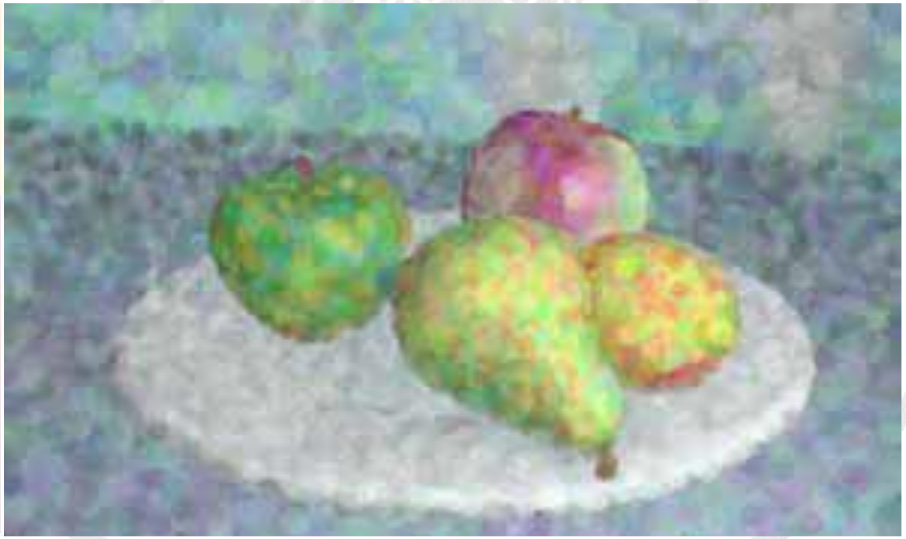
```
PImage im;
void setup()
{
 im=loadImage("download.jpg");
 size(800, 800);
 noLoop();
}

void draw()
{
  for(int x=0; x<im.width; x=x+1)
  for(int y=0; y<im.height; y=y+1)
   {
     color oldcolor;
     oldcolor = im.get(x,y);
     float r = red(oldcolor);
     float g = green(oldcolor);
     float b = blue(oldcolor);
      float ave = (r + g + b) / 3;
     color newcolor = color (ave, ave, ave);
     set(x,y, newcolor);
   }
}
```

- **Your task is to change what goes on here.**
- **Essentially some operations To change the value of `newcolor`.**

B. Image Stylization

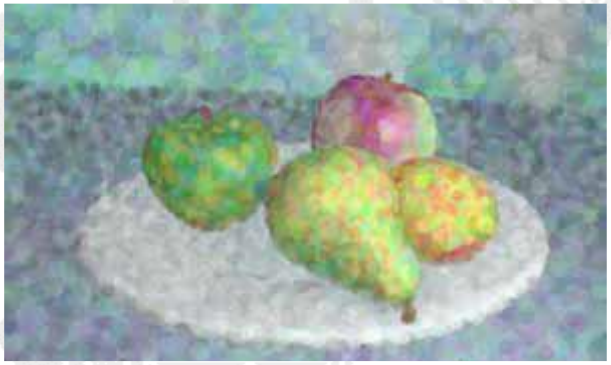
- **Painterly rendering, essentially using static particles to simulate brushstrokes (instead of pixels), is a widely used approach in computer graphics**

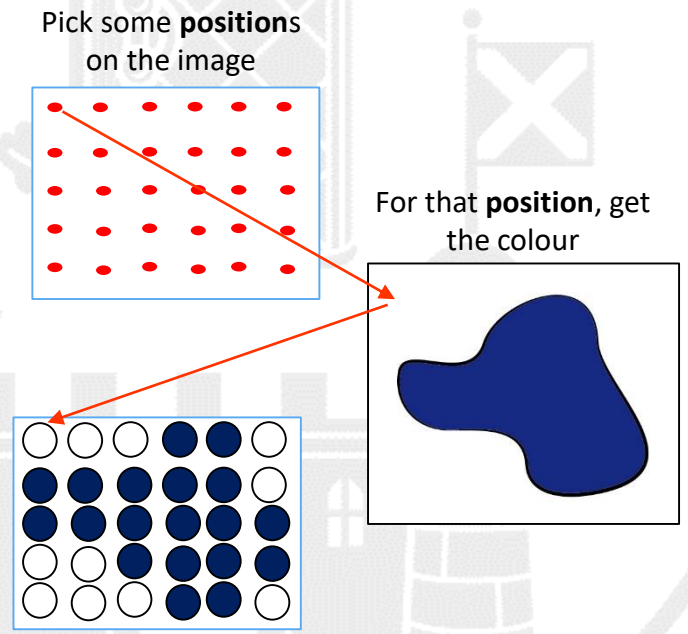
  Images below are from a famous paper by Barbara Meier from Walt Disney Feature Animation in 1996 [SLIDES] [PAPER]



B. Image Stylization

- **Painterly stylization replaces (a subset of) pixels in the image with brush like elements (e.g. strokes, stipples or points)**
- **The basic approach**
  a) Iterate over "random" points on the image
  b) Retrieve the color of the image at that point
  c) Draw a "stroke" at this point with the color you retrieved





Fruit image from the paper "Painterly Rendering for Animation". Barbara Meier. SIGGRAPH 1996.

0. Setting up (As before)

- Start a new sketch with at least a **`setup`** and **`draw`** function

  OPTIONAL: Use **`noLoop()`** *to prevent constantly updating the screen (in this lab we're interested in just drawing once)*

- Find an image file (ideally not bigger than the resolution of your screen), save it to your sketch folder
- Create a window of approximately the same resolution as your image
- Load the image

  OPTIONAL: Check that you've loaded it correctly by drawing it using the **`image(…)`** *function. Then delete this before further steps (as we want to draw it differently)*

```
PImage im;
void setup()
{
 im=loadImage(" ??? ");
 size(??, ??);
 noLoop();
}

void draw()
{



}
```

1. Iterate across points on the screen

- **As before, you need two *nested* for loops**
- **<u>Unlike before</u> you don't go to every pixel. Instead you go to a subset of positions.**
  - the **update** part of the for loop should skip a few positions in x and y directions

```
PImage im;
void setup()
{
 im=loadImage(" ??? ");
 size(??, ??);
 noLoop();
}

void draw()
{
   color imcolor;
    for(int x=0; x<im.width; x=x+??)
    for(int y=0; y<im.height; y=y+??)
    {


    }
}
```

NOTE: You must replace the **??** With some value (try 10 for now and see what this does… then come back here after completing the rest of the steps).

2. Retrieve colors

- **For each point retrieve the color of that point from the image you loaded**

```
PImage im;
void setup()
{
 im=loadImage(" ??? ");
  size(??, ??);
  noLoop();
}

void draw()
{
   color imcolor;
    for(int x=0; x<im.width; x=x+??)
    for(int y=0; y<im.height; y=y+??)
    {
      imcolor = im.get(x,y);

    }
}
```

3. Draw strokes

- **Now using the color that you retrieved, draw a "stroke"**
  - This can be any geometrical shape. But for now use an ellipse
  - Generally we make the strokes bigger than the size of a pixel, creating a stippled or brush like effect

```
PImage im;
void setup()
{
 im=loadImage("Image8.jpg");
 size(??, ??);
 noLoop();
}

void draw()
{
    color imcolor;
     for(int x=0; x<im.width; x=x+??)
     for(int y=0; y<im.height; y=y+??)
     {
       imcolor = im.get(x,y);
      fill(imcolor);
      ellipse(x, y,  ???, ???);
     }
}
```

OPTIONAL: Randomize

- **ADVANCED STEP (only try this when all else is done successfully):**
  - You can make the image look more interesting by **randomizing** the x and y positions slightly e.g.

```
float rx = x + random( ??? )
float ry = x + random( ??? )
imcolor = im.get(rx, ry);
…
ellipse(rx, ry, ???, ???);
```

4. Customize

- **Try to understand the lines of code provided in the sample (we may use them again later)**
- **Then change a few parameters to customize your result (try some of the following – N.B. it is not compulsory to do ALL of the following):**
  - Try using the command `noStroke()` to draw ellipses (or other shapes) without the outline.
  - Try different sizes of stroke
  - Use a different input image
  - Skip a different number of pixels

ADVANCED OPTIONS

- Instead of using the for loops, draw brushstroke (ellipses) with the mouse that `get` their colour from the loaded image

- Slightly modify the colour you get from the image (e.g. randomly make it brighter or darker; invert colours; greyscale colours) before you draw the stroke

- Add transparency to the strokes (this might make them blend better). This would go something like this:

```
color oldcolor = im.get( x, y);
color newcolor = color (red(oldcolor), green(oldcolor), blue(oldcolor), 150);
fill (newcolor);
ellipse(x, y, ???, ???);
```

Transparency value between 0 – 255.