Images in Processing

These notes introduce the techniques to include and use images in sketches.

The first step is that of saving the sketch so that a folder is created for the sketch in secondary storage, exactly like the example of Text and typography. This allows us to copy images into this folder to facilitate access from our sketch.

For this example, it is recommended that the image used in this example is resized to 600x600px. This will allow us to create examples that make use of the full real-estate of the sketch to others that allow us to make use of smaller images.

Loading Images

When using images in Processing, we will take the same approach that was taken with the usage of fonts. Before the `setup()` method we need to create an object that would act as a place holder for the image within the program. The image object in Processing is `PImage` and it is impetrative that the case of letters is followed. This object needs to be declared outside of the `setup()` and `draw()` blocks so that it would be accessible by all methods in the sketch.

The function to load the image is `loadImage(ImageFileName)` and this loads the image in question into the placeholder. In this example, the image "*image.jpg*" will be used it is assumed that this image is placed in the same folder where the sketch was saved.

```
PImage img;

void setup(){
     size(600,600);
     background(255);
```

```
      img = loadImage("image.jpg");
}
```

Drawing Images

Up till this point, the image is loaded into the placeholder through the instance `img` and can be accessed at any time from now on in the program.

Once the image is loaded, it can be then drawn on the display window by calling the function `image(img, x, y)`. This function draws the image stored in `img` on the display window with the top left corner of the image placed at coordinates x and y. The draw method below shows an example where "image.jpg" will be drawn on the display window. Since the size of the image is in this case equal to the size of the sketch, this will result in the program drawing the image as 'background' of the sketch. Any other shape drawn after the placement of the image, would be drawn on top of it, in the same way as other the ordering experienced in the previous examples.

```
void draw(){
      image(img, 0,0);
}
```

The function `image(img, x, y)` can also take 2 additional parameters, width and height, hence: `image(img, x, y, width, height)`. These two parameters allow for resizing of the image. All the parameters except `img` are numbers. Therefore, one may replace these parameters with the techniques demonstrated in the notes set named 'Generating Numbers'.

Advanced Image Techniques

The above example shows a very basic way of using images, displaying them in various sizes or locations on the screen. One has to underline that the placement of images happened as a second stage, after the image was loaded onto the

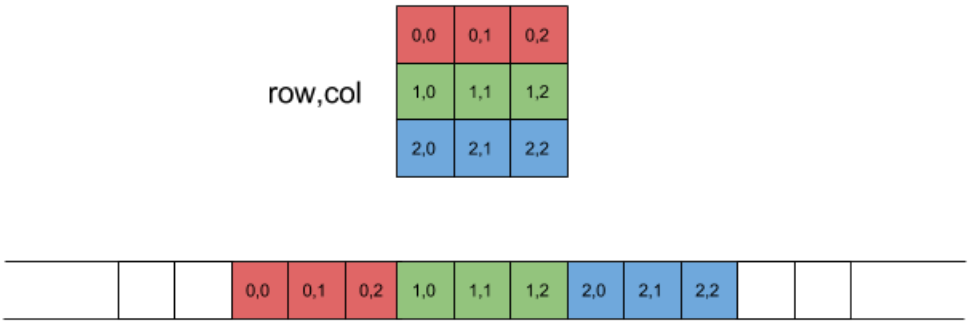
program's memory in the object `img`. In this example, we will be making use of the image data without explicitly displaying it.

This section is a walkthrough of an example that accesses random pixel coordinates (x, y) of the original image and reads the pixel values of these pixels. Pixel values are Red, Blue and Green values that result in the colour of the pixel in question. Full program code will be provided at the end of the example.

The first step is accessing a random x-value where the lowest value is 0 and the highest value is the width of the image. The y-value follows a similar logic, ranging from 0 to the height of the image. As discussed in the notes dealing with random numbers, the function `random(max)` returns a number with a decimal point (`float`) but we need a whole number (`int`) for both x and y. Float values are converted to `int` values by simply writing `(int)` before the `random` function as shown below. Note that these are to be written at the start of the draw() method so that new values are generated every time the program loops.

```
int x = (int)random(img.width);
int y = (int)random(img.height);
```

The next step is that of accessing the pixel values of img placed at location x and y and extract the values. The first step is that of make Processing convert the image into a format that we can easily access. The function to be called is `loadPixels()`. This function converts the 2D image into a single dimension (1D) structure as shown in the figure below:



The challenge is that in the 1D array of values, each element has an address that is not in the coordinate format x, y. This single-number address can be obtained by the following calculation:

```
int loc = x + y*img.width;
```

The value of `loc` would therefore refer to the corresponding value of x and y that are used to generate it.

Once that the pixel values are stored in the 1D array and we have a way of accessing specific elements of this array, we now need to store the colour values of the pixel being accessed. In order to achieve this, we will create 3 variables r, g and b to store Red, Green and Blue values respectively. For each colour, a corresponding colour extraction function will be called in order to extract the colour value from the combined pixel value as shown below:

```
float r = red(img.pixels[loc]);
float g = green(img.pixels[loc]);
float b = blue(img.pixels[loc]);
```

With these 3 values stored, we can now make use of the other techniques to draw any shape on the screen in the colour corresponding to the original pixel value of the image in question. In this example, we will be drawing a circle on the blank canvas for each random coordinate generated.

```
stroke(r,g,b);
fill(r,g,b);
ellipse(x, y, 5, 5);
```

The circle will have the colour of the pixel value of the original image found in the same coordinate. However, since the circle will be bigger than the pixel, the end result of the sketch would be a pointillised version of the original image.

Full Program Code – Pointillism

```
PImage img;

void setup(){
  size(600,600);
  img = loadImage("image.jpg");
  background(255);
}

void draw(){
  //Generate random x,y coordinates
  int x = (int)random(img.width);
  int y = (int)random(img.height);

  //Looking up the colour
  loadPixels();
  int loc = x + y*img.width;

  float r = red(img.pixels[loc]);
  float g = green(img.pixels[loc]);
  float b = blue(img.pixels[loc]);

  //drawing on canvas
  noStroke();
  fill(r,g,b,100);
  ellipse(x,y,5,5);
}
```