# CS 211 Lab #3

This lab is intended to get you started on PA #2. To this end, Lab 3 introduces you to the portable pixel map (PPM) image format and basic image manipulation.

## Prerequisites

Before beginning this lab, you will need some sort of imaging software that can open PPM-based images. I would recommend [Ifranview](http://www.irfanview.com/) for Windows as it's free and lightweight.

## PPM Image Format

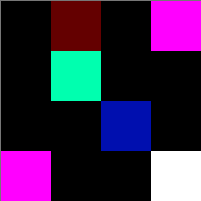
The PPM image format is encoded in human-readable ASCII text. It might be helpful to read over the [formal PPM specification document](http://netpbm.sourceforge.net/doc/ppm.html). A PPM document has two pieces: the header and the body. The header is always the top three uncommented lines in the file:

|  |
| --- |
| P3  4 4  255 |

The first line specifies the type of image that is contained within the file. We will always use the "P3" specification. The second line specifies the number of columns and rows present in the image. In this example, we have a 4x4 image. The final number indicates the maximum value for each red, green, and blue element in the picture. Having a max value of 255 is quite common and is the value that we will always use. Below the header is the body of the PPM file. Each pixel has a red, green, and blue value. For example, the content of our 4x4 image might be:

|  |
| --- |
| 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 |

With these values, the pixel in the first column and row has a RGB value of (0,0,0) and the last column in the first row has an RGB value of (255,0,255). A blown-up image containing these values would look like:

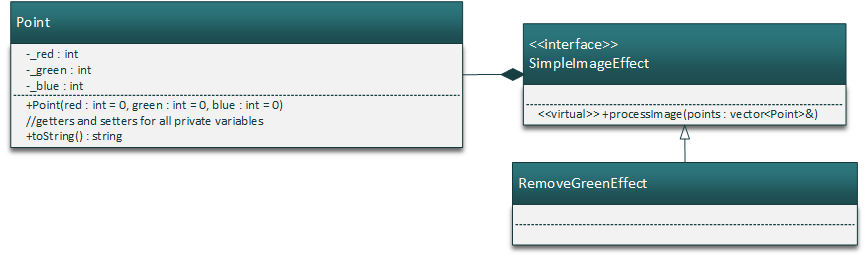


## Task

Your task for this lab is to write an image effect that removes all green from a given image. Note that this can be accomplished by simply setting the green value in each RGP triplet to zero.

## UML Diagram

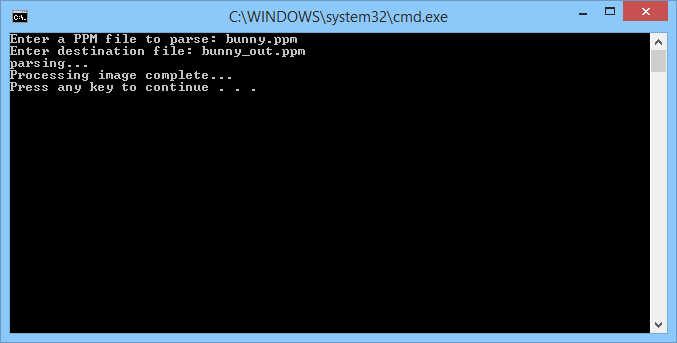
Below is a UML diagram for the lab. While it is most certainly overkill for this lab task, it sets you up nicely to complete the homework assignment.



Note that we use the Point class to represent a given RGB triplet in the PPM file. We create the SimpleImageEffect interface to represent a given image effect (the HW has many more) with RemoveGreenEffect being a concrete implementation of a given image effect. Note that the processImage function should process the PPM one line at a time.

## Sample Output

Below is a screenshot of my lab along with the input and output files:

Note that the "parsing" stage may take a while (several seconds).

### Bunny.ppm

This is the original bunny file:



### Bunny\_out.ppm

And the image with green removed:

