COMS4115 Proposal: YOLOP (Your Octothorpean Language for Optical Processing)

Sasha McIntosh, Jonathan Liu & Lisa Li
sam2270, jl3516 and ll2768

Motivation

Programming and attempting to manipulate images can often prove difficult and tiresome, while simple tasks such as cropping, changing images to black and white, and importing photos prove to be complicated. To ease this process, we aim to develop a language, YOLOP (Your Octothorpean Language for Optical Processing), that simplifies these common image processing jobs and provides a user-friendly programming to transform original photos into a finished product. By simplifying the data types used and providing key methods in our standard library, we have focused the scope of the language into one that can easily manipulate and process images. Our language will include techniques to import/export images, manipulate colors, and trim sizes. As such, YOLOP is a useful tool not only for photos and pictures, but charts and graphs as well. Potential uses for programming in YOLOP thus include developing typical image processing programs to apply a standardized color mapping onto each part of the image (i.e. like a popular application's use of filters), sharply saturating colors to identify and separate specific areas of an image, manipulating and comparing multiple charts and graphs together, and more!

Language Specification

Data Types:

- int: the same integer you know and love. The operators +, ++, --, * and / function as expected with divide taking the ceiling of decimal numbers. The following boolean operators are included and also function as expected: <, <=, >, >=, !=, ==, && and ||
- string: similar to what you have seen in the past, with some limitations. String concatenation can be performed with the + operator. Boolean operators are not supported, and string manipulation is not heavily supported either. String cannot be operated on with integers. In our language, strings are used mainly for naming files and very little else.
- img: the img type is used for image multiplication and stores the int values for each pixel in a 3D (RGB) array. The img allows for easier manipulation of the image and contains several public attributes such as:
    - img a.width returns an integer denoting the number of pixels in a row of a
    - a.length returns an integer denoting the number of pixels in a column of a
    - a.red, a.green, and a.blue allow a programmer to access those color arrays within the program

Commenting

- #* Comments are written like this *# where only the text inside the symbols are commented out. There is no definition for single-line comments.
- Commenting can be nested:

```
#* Commenting out the function

        int doNothing() {
                return 0;       #* This returns 0 *#
        }

*#
```

Sample Code

```
img src = “cat.jpg”;

int makeRedder () {
   int x = src.length;          #* This is the length of the image *#
   int y = src.width;

   for (int i = 0; i < x; i++)
      src.red(i,0)++;           #* Increase the “red” of the RGB *#

   src.save(“newimage.jpg”, “/home/media/images/”);
   return 0;
}
```

Standard Library Functions

The following functions may appear within the standard library to provide some methods that the user may find useful and use often.

“Image Concatenation” or appending one image to another and making up the size difference with white space.

Import/Export of images. To be more specific, the importing of images and conversion to the “img” data type and the saving of “img” as an image file.

Hex Code colors may be built in to allow for easy manipulation. One implementation could be to assign a hex code to a color.

e.g. PURPLE = #800080;

Color Functions may be built in. For example a gradient function that takes two points as parameters and blends the first color into the second in a “line”.