Molly Junck (mjunck)
October 12, 2011

Filter README

All of my Filter classes extend from Filter.cpp in the filter folder.

Invert and Grayscale filters were done pretty much as expected, nothing fancy.

For the blur filter, I implemented an O(n^2) algorithm of a box filter, where n is the width * height of the image. I used the ACM Siggraph paper elynxsdk.free.fr/ext-docs/Blur/Fast_box_blur.pdf as a resource for how one might do this. The blur is done in 2 passes, first horizontally, then vertically. Since the weight of each pixel is constant, one can sum the accumulation for the diameter of the blur and then add/remove the next/last pixel's value. For edge cases, I essentially extended the edges and took the beginning/final value repeatedly.

For scale, I created separate methods for scaling up and scaling down over x and y. It is not great object-oriented form, but not having to parse through a bunch of if statements really saves time.

Instead of i, j, I used x and y because I think better in coordinates. I tried to always iterate for all y and then for all x for faster memory look-up.

My extra filter is a scale filter and then a filter I call Lichtenstein based on dot images of Roy Lichtenstein. The Lichtenstein filter makes an empty image array filled with black, then goes through and samples from that location in the original image and paints dots on the blank image. The dot size and the distance between dots are constants declared at the top of the paint() method.

This is very late and I'm sorry :0(. I am hoping to use all my late days on this one.