CMSI 371
COMPUTER GRAPHICS
Spring 2008

Assignment 0221

Yes, the "for submission" part of this assignment is due on the same day as the midterm — sorry. However, it does reinforce some concepts that will be part of the midterm, so rather than thinking of it as homework, think of it as additional preparation for the midterm.

Not for Submission

Color and how pixels are represented in memory are covered in Section 2.5 in Angel.

For Submission

Extend the Nanoshop sample program by implementing two (2) new color changers for it. Ideas include, but are not restricted to: brightness changer, contrast changer, inverter (i.e., creating a photo negative), and a tinter/blender.

What to Do

Type up the distributed Nanoshop code — if you think about it, you really only need the Nanoshop class and the ColorChanger interface; no need to type up the sample ColorChanger implementations — then add your own color changers. For those of you who have taken CMSI 370, feel free to improve upon the user interface of the core Nanoshop application.

How to Turn it In

Commit your code to */homework/cmsi371/nanoshop*.

For Submission (Extra Credit)

You will receive extra credit if you expand the range of image processing operations that Nanoshop can perform as specified below.

What to Do

Modify the ColorChanger interface's *changeColor()* method so that it accepts an $n \times n$ "square" of colors, where n is an odd number, while still returning a single RGB tuple as a result. This "square" represents the pixels surrounding the "current" pixel in the Nanoshop *processImage()* method, thus providing additional information that can be used when "filtering" the original image.

Implement two (2) color changers that take advantage of this "neighboring pixel" information. For example, you can return the mean of all colors as the result, or perhaps the median (the interpretation of how the mean or median of a set of colors should be computed is left to you). Try to come up with something that you think will produce some kind of recognizable visual effect.

Of course, the Nanoshop *processImage()* will also have to change slightly, so that it builds the square of colors that is sent to the new *changeColor()* method, with the current pixel at the square's center. Make sure to handle the (literal) edge cases — for pixels at the corners or borders of the image, what colors should be placed in their non-existent "neighboring" areas?

How to Turn it In

If you complete this extra credit work, simply commit it as your version of Nanoshop, under the */homework/cmsi371/nanoshop* directory.