Lab 3 - Part 2

Monday, September 30, 2019 1:14 AM

Part 1 Recap

- Use handouts for ppm.
- Finish support.cpp & .h first. Test with ppm_copy on Piazza.

★ Class: Pixel
- Stores x, y pixel pair

★ Class: Matrix
- Stores 2D array
- There are 3 versions on the handout on Canvas. Use what you want (I used version 3).

★ Class: RGB
- Stores red, green & blue values for a single pixel.

★ Class: PPM
- Stores a matrix<RGB> to hold pixels.
- Read in pixels w/ ppm::read.
- Write PPM to binary w/ ppm::write.

- Has helpers like get_Ncols and get_Nrows which call functions in the matrix class of the same name.

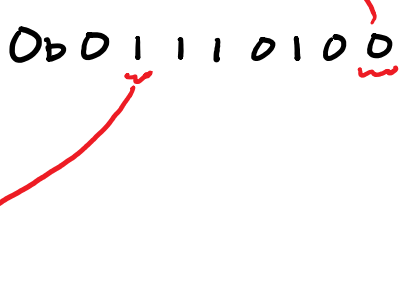
★ Function: set_pixel_list
- Make a vector<pixel>.
- Store coordinates of the order of pixels to modify.
  - Part 1: Every even pixel
    - (0,0), (0,2), (0,4), etc.
  - Part 2: Every single pixel
    - (0,0), (0,1), (0,2), etc.
- Either return vector (copy, slow), or pass-by-reference (fast). I don't care...

★ Function: Encode
- Iterate through vector<pixel> given by set_pixel_list and encode bits into LSB of a colour of a pixel.
  - Alternate what colour is used "per pixel"... Starting w/ RED for (0,0).

- Yes. GREEN for (0,2). and BLUE for (0,4). Then start at RED again for (0,6). etc.

- Only one colour's LSB per pixel is modified. So:
  (0,0).R.LSB = BIT 1
  (0,2).G.LSB = BIT 2
  (0,4).B.LSB = BIT 3

- Only encode the first 7 bits in the byte... skipping the 8TH. We will assume that it's always 0.
  - "t" is 0b01110100 in hex...
    (0,0).R.LSB = 0
    (0,2).G.LSB = 0
    (0,4).B.LSB = 1
    (0,6).R.LSB = 0
    (0,8).G.LSB = 1
    (0,10).B.LSB = 1
    (0,12).R.LSB = 1



See the pattern?

★ Function: Decode
- Opposite of encode...