CS 580 Assignment 1

During the course of the assignments, you will build a small but useful graphics library. We'll call it the Gz library, and all functions, vars, etc, will be named accordingly. Certain standards will apply to make the code interfaces consistent. Consistency will be established by the use of prepared include files and an Application Program Interface (API).

The include files you need for this assignment are Gz.h and disp.h. Both of these are found in the zip file hw1.zip.

There are several other files there that may be useful.

- **disp.cpp**
  - Your task for this assignment is flesh out the functions in disp.cpp.

- Application.cpp
  - Pointer for frame buffer and display is defined.
  - Just user it

- Application1.cpp
  - This application is complete and calls the display routines you will write.
  - Just use it.

- rects
  - A data file used by app1. You should be able to figure it out.

- output1
  - A sample ppm-format result image which can be viewed by "i_view".
  - It is created by running your program.

- Other files are used to make program frame work including user interface and windows display. Just use them

All displays are addressed by pixel coordinates, and accept or return pixel values. Upper left pixel is (x=0, y=0). x increases to the right, and y increases downward (raster order).

A flush operation writes the accumulated pixels to a disk file and display. Disk files will be in the "ppm" file format. ppm files are color image files that have an ascii header with xs and ys image-dimensions and a binary 3-byte pixel format:

P6 xs yx 255\r
rgbrgbrgbrgb...

A sample ppm header might be: P6 512 480 255\r. This would be correct for an image with 512 pixels horizontally and 480 pixels vertically.

Also, you should assign frame buffer and put the accumulated pixels of a display to the buffer. Then, *program draw* your result to the window. Pointer of the frame buffer is assigned in the “Application” class and you should use that pointer to connect it to the window display routine. (NOTE: Pixels for *program draw* should be stored “bgrbgrbgrbgr …” instead of rgb…)

Display objects hide the organization of pixel memory and its allocation from the application and renderer. Only applications create, free, and flush Displays. Display class and size are determined by the application. **(The size should be same or less than the frame buffer size.)** Pixels are written by the Renderer using the Put call. Defining the API interfaces makes the application and Renderer library display-independent. See the disp.cpp file for a complete description of the API.