

Research Report
Week 7

Student: *Craig Harrison*
Advisor: *Dr. Richard Eason*

Progress:

This week has seen major progress in the development of a new graphical user interface. Last week I believed the drawing area was nearly complete, but when I went to add drawing functionality such as `draw_pixel()` or `draw_line()` I discovered that my drawn windows was not persistent. After getting covered, either by other windows or my application's menus, whatever was previously underneath would be gone. This has now been dealt with and all the necessary `draw_()` functions have been implemented on top of the Cairo graphics library.

Approach:

Unfortunately, multiple online searches turned up no recent examples of persistent drawing surfaces. One example could be found, but it was an old C example using many GTK functions that no longer exist in `gtkmm 2.x`. However, it did point me in the direction of the `Gtk::Pixmap` object. In looking at the doxygen (documentation auto-generator) pages for `Gtk::Pixmap` I was able to figure out a method by which I could use a pixmap object as my drawing buffer and just redraw the window from that, as every time a section of the window was uncovered an `ExposeEvent` was generated. Installing my own callback function for the window's `ExposeEvent` allowed me to discover what area of the window needed redrawing and efficiently redraw just that section from the `Pixmap` buffer.

However, as I was no longer drawing directly onto the window, my drawing functions would not show up on the screen until some other `ExposeEvent` caused that portion of the window to be redrawn. To this end I created a `refresh()` function which generates an `ExposeEvent` for any given section of the drawing area. Each of my drawing functions intelligently calls `refresh()` with the smallest possible area when finished drawing to the buffer.

Another problem created by using an intermediate `Pixmap` object was that this buffer is of a set size. Whenever the window was resized the drawing area would follow but the buffer I was drawing onto would not, limiting the usefulness of having just resized the window. To allow the buffer to behave more logically I implemented an `on_configure_event()` callback function. This function gets called both when the window is first created and every time it is resized. Here the function check if a buffer already exists, if not one is made and cleared, if one already exists a new one is made and then the old buffer is draw onto the new buffer. This keeps the same image draw to the screen even over resizes.

Other Work:

I have also spent time this week looking into LaTeX as a layout engine for the final research report. Reading through documentation and trying my hand at some simple demos I have decided that the power and simplicity of the LaTeX engine would make for a great improvement on both the efficiency and looks of the report.