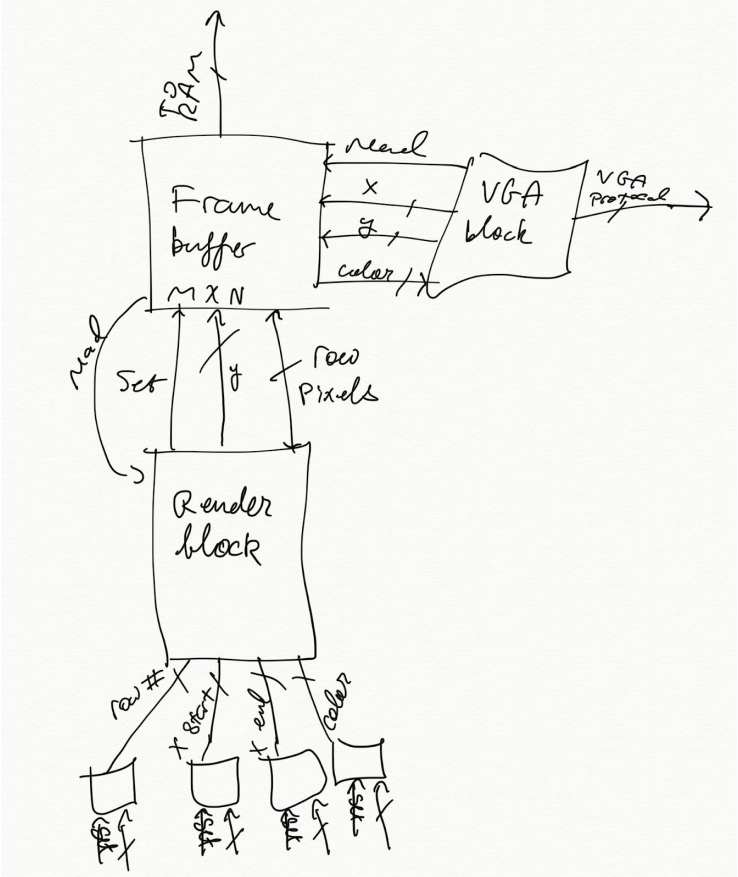
Design Document (WebRender)

Alex Gajewski (apg2162) and Allison Ghuman (ag3910)

The plan is to render one row of the frame buffer at a time, letting software define a start column, an end column, and a color for the line being drawn. The rendering block will take these parameters, read the old state of the row from the frame buffer (initialized to all white at the beginning), and then draw the new color between the two endpoints. The frame buffer just needs to keep a big array in RAM of the values of every pixel in the screen at any given time, and respond to read requests from both the rendering block and the VGA block, and writes from the VGA block. The VGA block will basically just be the same loop we used in Lab 3 to send pixels to the monitor, sending one pixel at a time. If the RAM roundtrip latency ends up being too slow, we may end up needing to read a row at a time from RAM as well, though this would depend on the RAM drivers supporting block requests as well. The frame buffer will need about a megabyte of RAM to keep all of the pixel values in memory. The Linux system will need very little memory; only information about the various objects that it wants to write into memory. Disk usage should also be minimal. As for the hardware/software interface, the plan is to have the four control registers (row number, start column, end column, and color), as well as a status bit (ready to write). The software will poll the device until it is ready to render the next row, perhaps trying to be smart about how long to wait in between polls by computing some statistics about average



rendering times. It should be close to a fixed number of cycles, so the timing should be fairly consistent.