

This week, we completed the edits needed to run Tight VNC viewer with the region options. The screen was having severe update issues because we changed the option for the incremental update to be a full update, thus causing the client to send much more information to the server at times than it should have, causing the client to hang up waiting for server response. Once this was fixed, we went through to clean up the code and made sure final changes were made as needed, as well as cleaning up the code in various places in preparation for submittal to the developers.

The final changes were made in the files we expected to make them in, though several debugging statements were thrown in to other files when figuring out how other functions worked and when specifically they were called. We edited the main file `vncviewer.c` to interpret the argument `-region` and parse its value `WxH+X+Y` so the correct values were read in to the correct variables. We wrote a `ParseRegion` function and placed it in the arguments code file `argsresources.c` to take care of this, scanning through the C string until a number was reached and putting the value into the proper variable. The variables were named `regionW` and `regionH` for the region's dimensions, and `offsetX` and `offsetY` for the offset and globally declared in the main header file `vncviewer.h` and initialized to zero in the main file. The arguments were then removed from the list of arguments to be handled by X.

The RFO protocol file `rfbproto.c` was where a majority of the changes were then made. Error checking was done in the main file for negative values, and in `rfbproto` checking was done to make sure the offset and dimension combination did not exceed the frame buffer dimension of the screen being accessed. For example, if the native desktop you were accessing had dimensions 1024x768 and you wanted a 1000x1000 region at point (50, 50), it would throw an error and exit because the region goes outside of the frame buffer. If the user initialized the width or height to zero, the program catches this by setting the dimension to the matching frame buffer dimension minus the offset. Once the values were checked, the frame buffer width and height were set to the new width and height to create the new dimensions. All other changes made were to correct for the offset. We wanted the screen to behave as if the upper-left-hand corner was the point (0, 0), regardless of the offset given. Once we found the correct places to make these changes, as well as some negative number checking, the program worked exactly as desired.

Once the final changes were made, we did some simple performance tests on the wall at the Innovation Center. By using `glxgears` and a stopwatch program, we were able to determine the frame rate of when running one large session versus two Tight VNC viewer sessions, one on each node, versus running one session at the correct offset per monitor. In the end, there was not that much of a difference when running one session or sixteen sessions (all ran at about 2-3 frames per second), but there was a slight increase of speed when only two sessions were running. Still, we did not get a chance to work on server to client communication, so it is still a mystery as to why we cannot get the frame rate to match that which the server is sending, which is a frame rate of around 6-7 frames per second.

After this was working correctly, we began to hack the Java code and match up where changes needed to be made in the `.class` files versus our changes made in the `.c` files. Currently, the region implementation works and we are working on a site that will implement the applet.