# Repo Link:

# <https://github.com/meBielawski/Paint_Editor.git>

# Summary

**What changed:**

There was a lot that I tried to tackle in this project. I initially really wanted to be able to have a nicely put together data model that could save all the details in Json and then use that info to convert to SVG. Instead, I decided to use a stream and XML writer in the program to write the data of the bitmap pixels and then used an SVG namespace to make it displayable. I also really wanted to make an editable property grid that could allow a user to change the details of a line or shape and have the edits be displayed. I was able to get the nodes into the viewer and have the details of a line be shown, but they are not editable. I was not able to get shape details to be shown. Snap to grid and drawing a grid/ruler was also a feature that I was excited to implement, but I did not end up getting around to it. After seeing the example, you showed, I tried to reformat some of what I had to be similar in nature. I was able to make a mode and selection change controller which helped.

**Easier/Harder than expected:**

It took me a long time to research exactly how to use a streamwriter and XMLwriter. Longer than I would have liked, and I could have used that time to research how to implement Json instead. The select feature also took me a long time to figure out. It was hard to figure out how to get it to delete items in the rectangle that I used to select. I also had a tough time with shapes. For whatever reason, I could draw a shape, but it would break other features like free draw and the eraser. It would mess with my start and end points in a way that would just draw a random line across the screen. Turns out I was not handling my start and end points correctly, which Antoine was able to help me figure out after a lot of debate and headache. Undo and Redo were also a bit of a challenge. I wasn’t sure if I wanted to deal with individual elements or the whole bitmap. I ended up going with an implementation that saves the entire bitmap instead of just the elements, which in hindsight is probably a bad practice for memory usage. I also implemented it using stacks instead of a list, which wasn’t much of a learning curve but also had its own challenges. Saving and Opening weren’t extremely difficult but did cause some headaches during my testing since I couldn’t overwrite an existing file.

What was relatively easy was the free draw feature. I was able to figure that out quickly using Microsoft Learn and then realized that the eraser is basically the same thing but in the same colour as the background. I also had some experience from the previous assignment figuring out buttons and UI design for color dialog interaction, which made taking care of those things relatively easy and quick.

**What I learned:**

I think my main takeaway from this assignment is that I need to practice segmenting my code into useful sections and other files more. I found it hard to implement at first, then I tried some things that worked such as the Mode selection and Shape Selection. I tried to implement one with shade details specifically kind of like the SVG editor that you created but hit a wall at one point and decided to scrap it. I wish I wasn’t so quick to do so, as I think it would have made things easier to document for later use, such as implementing the nodes in the property view and saving shape information into Json. I also should have maybe focused on the data a little more. I really dived into the UI first and then worried about the data dealing with later, and in retrospect I should have really hunkered down and focused on the data handling and modeling.   
  
I think I will try to redo this project for personal use in the future, as I truly had a great experience. I was proud of the product at the end, despite the few bugs I was able to find.