Paint App

Skills Used

- C#: Different types of event handling
- XAML: Creating Styles and using an Ink Canvas for stylus and touch events
- WPF

Description

I created this app as part of my SE 4220 GUI Programming class. The requirements were generic which left me with some creative license to make an app that interested me.

Requirements

Write a WPF application with at least three of the following:

- Routed Event
- Keyboard Event
- Mouse Event
- Stylus Event
- Touch Event
- Command

Implementation

C#

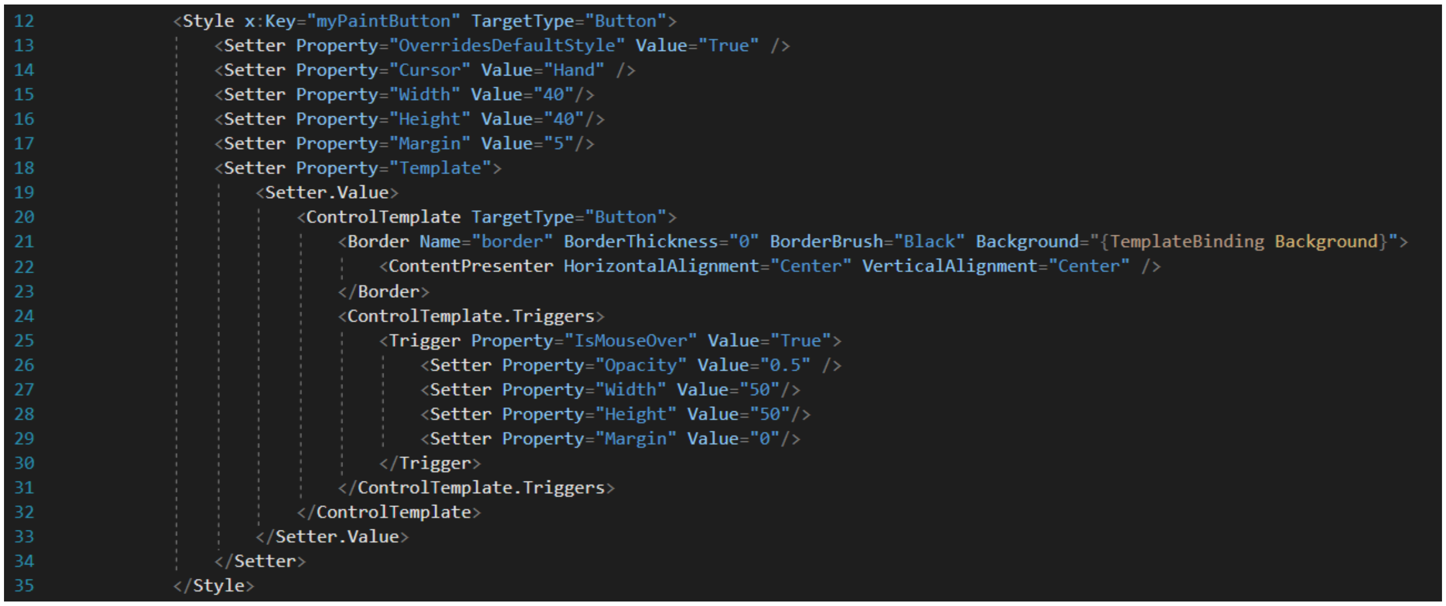
In my final application I included routed events, mouse events, stylus events, and touch events. I wanted the user to be able to use a touch screen to draw along with a stylus and mouse. I handled each of these events in a slightly different way depending on the user input.

XAML

This was a very interesting project to work on and I was able to learn quite a bit even from such a small project. I gained a lot more experience with XAML and learned a few techniques for simplifying it and adding styles to user controls to create a better overall design and experience for the user.

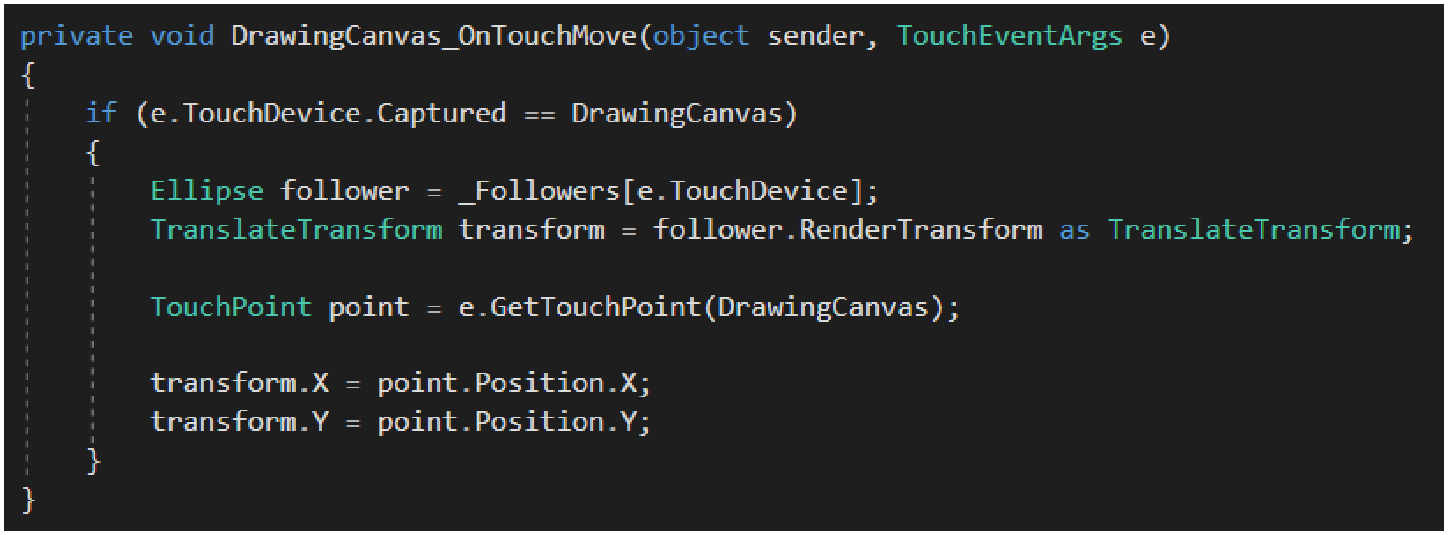
CODE SNIPPETS

To simplify my XAML I created the style below for the paint colors in the color pallete. Each color is a button and when the mouse scrolls over a button I wanted it to change opacity and size which is done with a trigger dependent on the IsMouseOver propery.



```
        <Style x:Key="myPaintButton" TargetType="Button">
            <Setter Property="OverridesDefaultStyle" Value="True" />
            <Setter Property="Cursor" Value="Hand" />
            <Setter Property="Width" Value="40"/>
            <Setter Property="Height" Value="40"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="Template">
                <Setter.Value>
                    <ControlTemplate TargetType="Button">
                        <Border Name="border" BorderThickness="0" BorderBrush="Black" Background="{TemplateBinding Background}">
                            <ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center" />
                        </Border>
                        <ControlTemplate.Triggers>
                            <Trigger Property="IsMouseOver" Value="True">
                                <Setter Property="Opacity" Value="0.5" />
                                <Setter Property="Width" Value="50"/>
                                <Setter Property="Height" Value="50"/>
                                <Setter Property="Margin" Value="0"/>
                            </Trigger>
                        </ControlTemplate.Triggers>
                    </ControlTemplate>
                </Setter.Value>
            </Setter>
        </Style>
```

The following method handles the touch input and to visualize the touch input I attached a follower, so the user could visualize each touch input they have on the screen, this is just a large dot that follows the user finger around the screen to help them see all the touch input. I only added this follower to touch input, so it would not show up if the user is drawing with a stylus. You can see an example of what touch input would look like in the screenshots below.

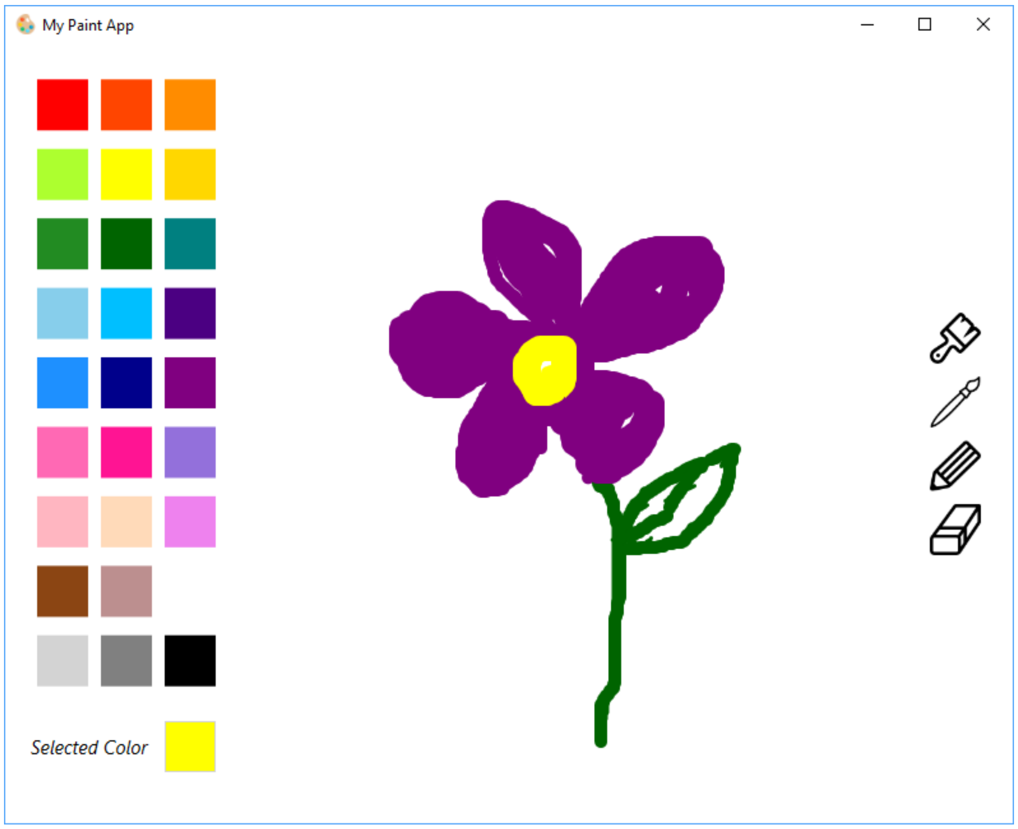


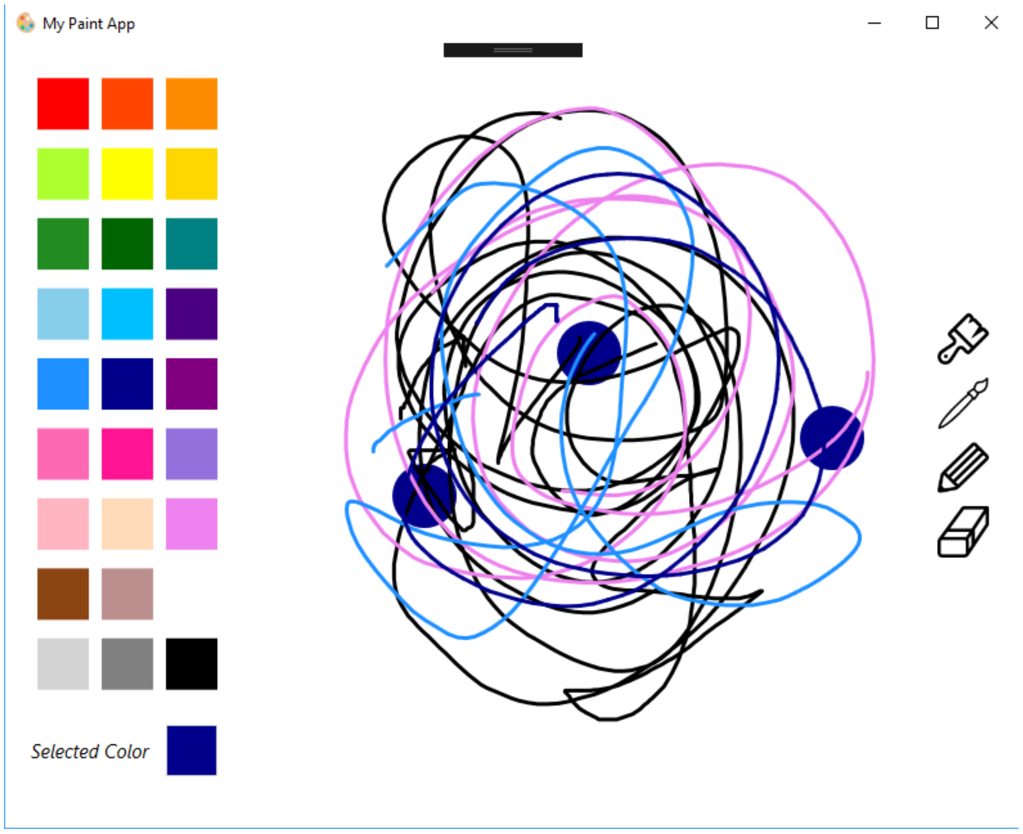
```
private void DrawingCanvas_OnTouchMove(object sender, TouchEventArgs e)
{
    if (e.TouchDevice.Captured == DrawingCanvas)
    {
        Ellipse follower = _Followers[e.TouchDevice];
        TranslateTransform transform = follower.RenderTransform as TranslateTransform;

        TouchPoint point = e.GetTouchPoint(DrawingCanvas);

        transform.X = point.Position.X;
        transform.Y = point.Position.Y;
    }
}
```

Screen Shots





(The large blue dots are feedback from touch input and are not added to the canvas)