Brushes

abelski

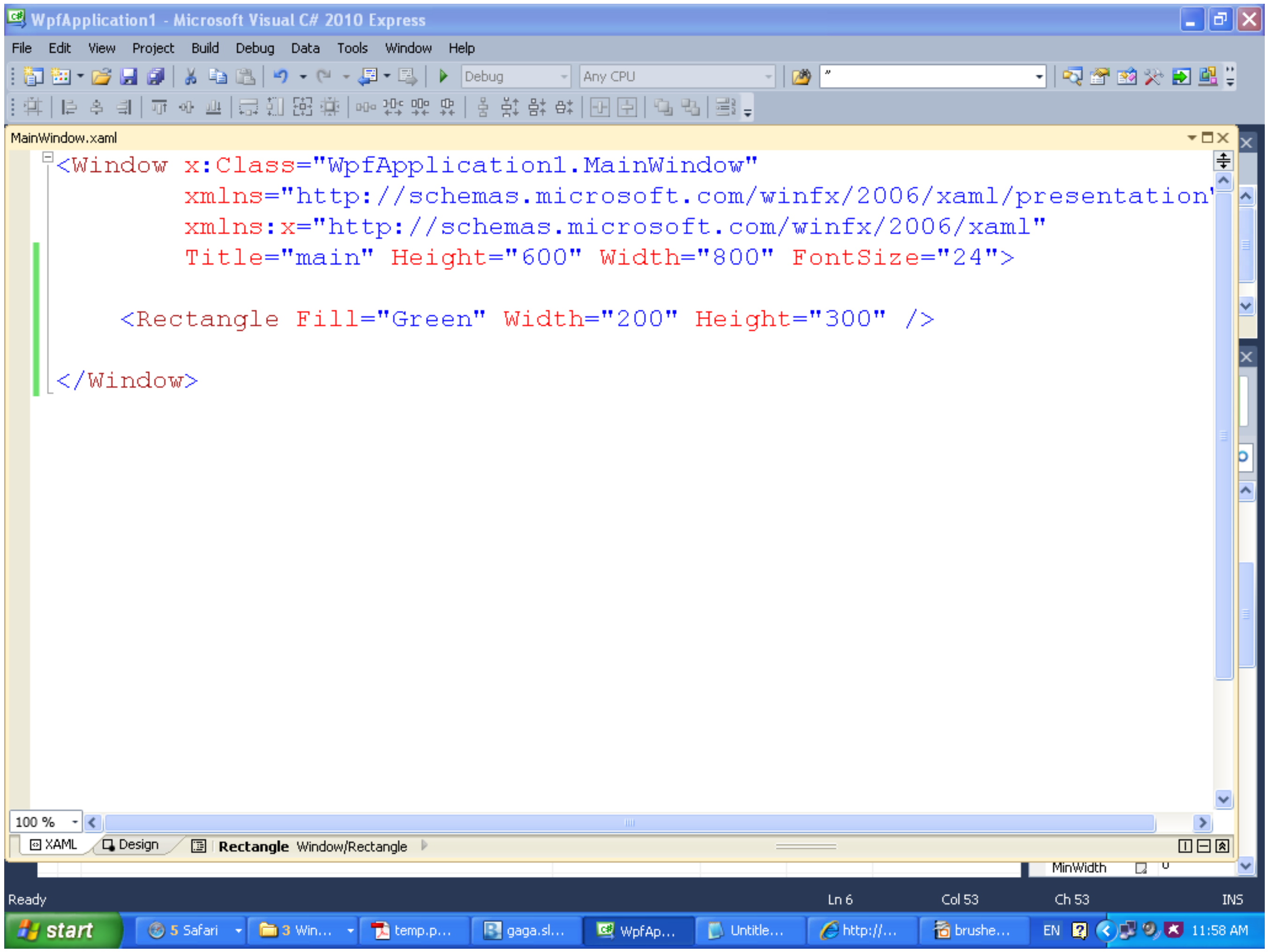
Introduction

- The `brush` determines how to fill-in an area. An area can be either a simple area of a specific share, such as a circle, rectangle or any other shape. An area can also be a complex one, such as the area of a polygon.
- The `brush` also determines how to fill-in the texts graphics. We can fill in the text either with a solid color, a color gradient or even using a picture or a pattern.

Introduction

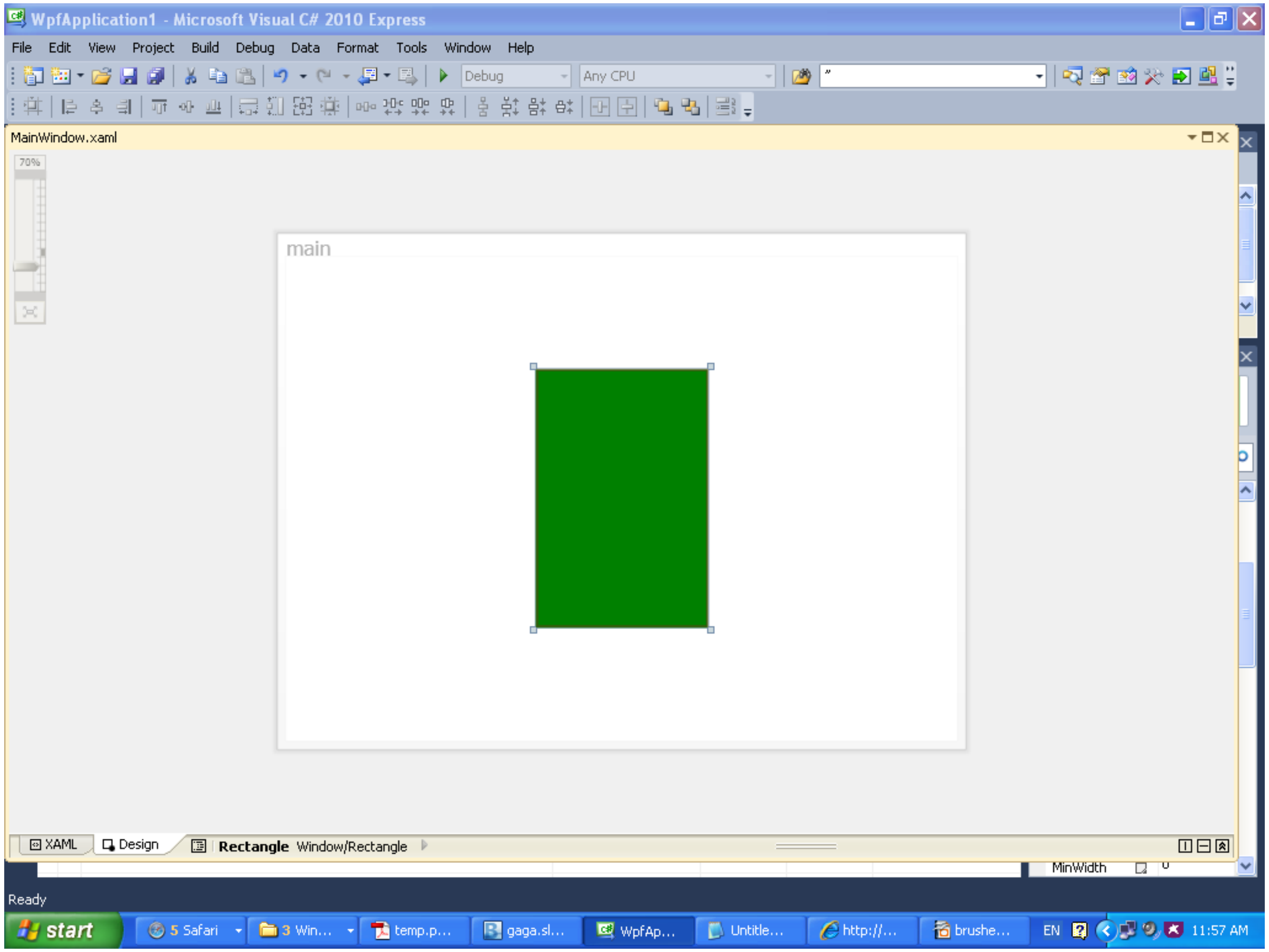
- The simplest way for setting the brush we want to use is by using the `Fill` attribute.

Introduction


WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Tools Window Help
Debug
Any CPU
MainWindow.xaml
100 %
XAML
Design
Rectangle Window/Rectangle
MinWidth
Ready
Ln 6
Col 53
Ch 53
INS
start
11:58 AM

```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="main" Height="600" Width="800" FontSize="24">

    <Rectangle Fill="Green" Width="200" Height="300" />

</Window>
```

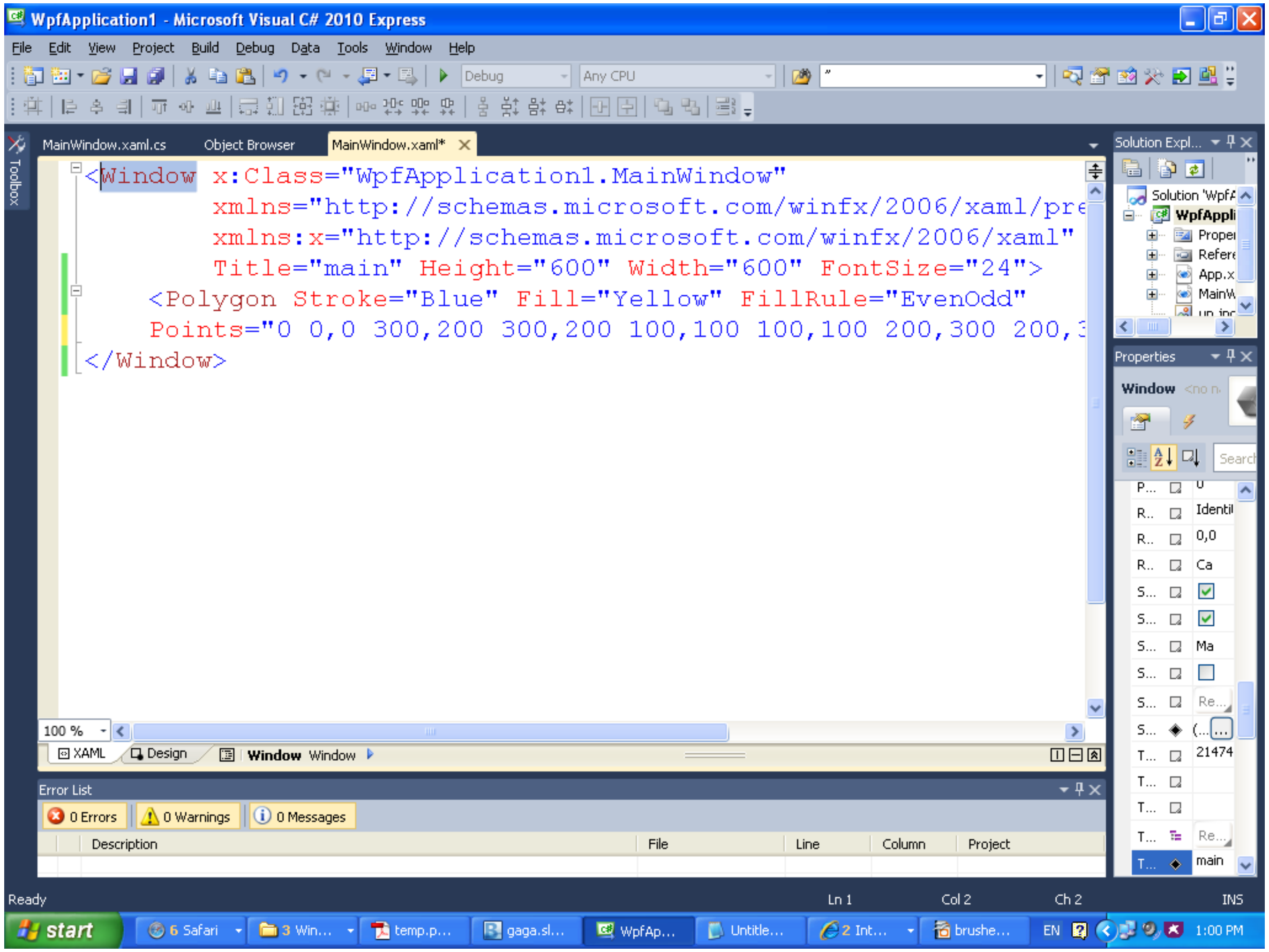
Introduction


WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Format Tools Window Help
Debug
Any CPU
MainWindow.xaml
main
XAML
Design
Rectangle Window/Rectangle
MinWidth
Ready
start

The `FillRule` Attribute

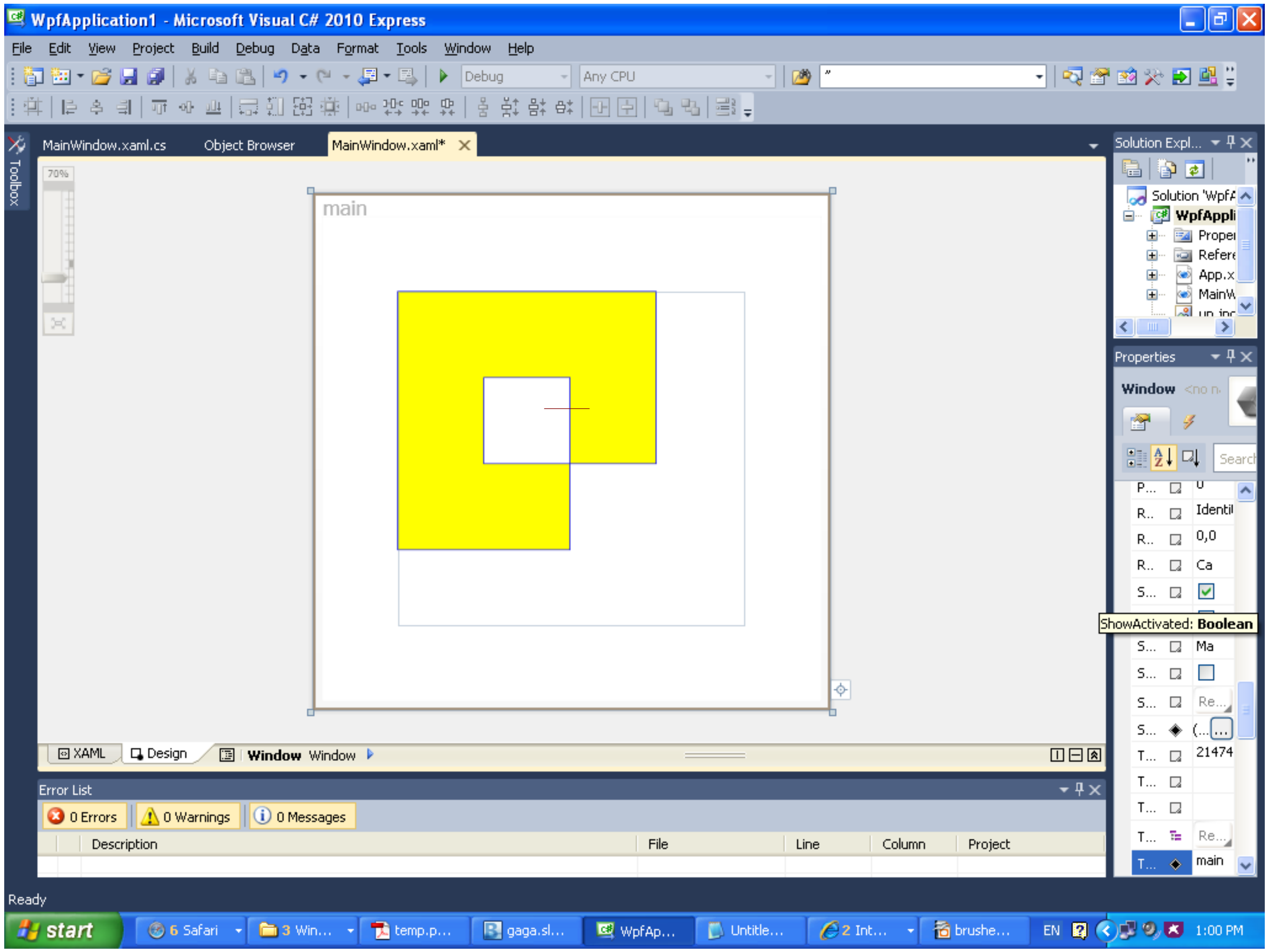
- We use this attribute in order to instruct how to fill in areas created by lines that cross with each other.
- If the value of this attribute is `Nonzero` then each one of the areas will be filled.
- If the value of this attribute is `EvenOdd` then just those areas that are enclosed an odd number of times will be filled-in.

The `FillRule` Attribute (`EvenOdd`)

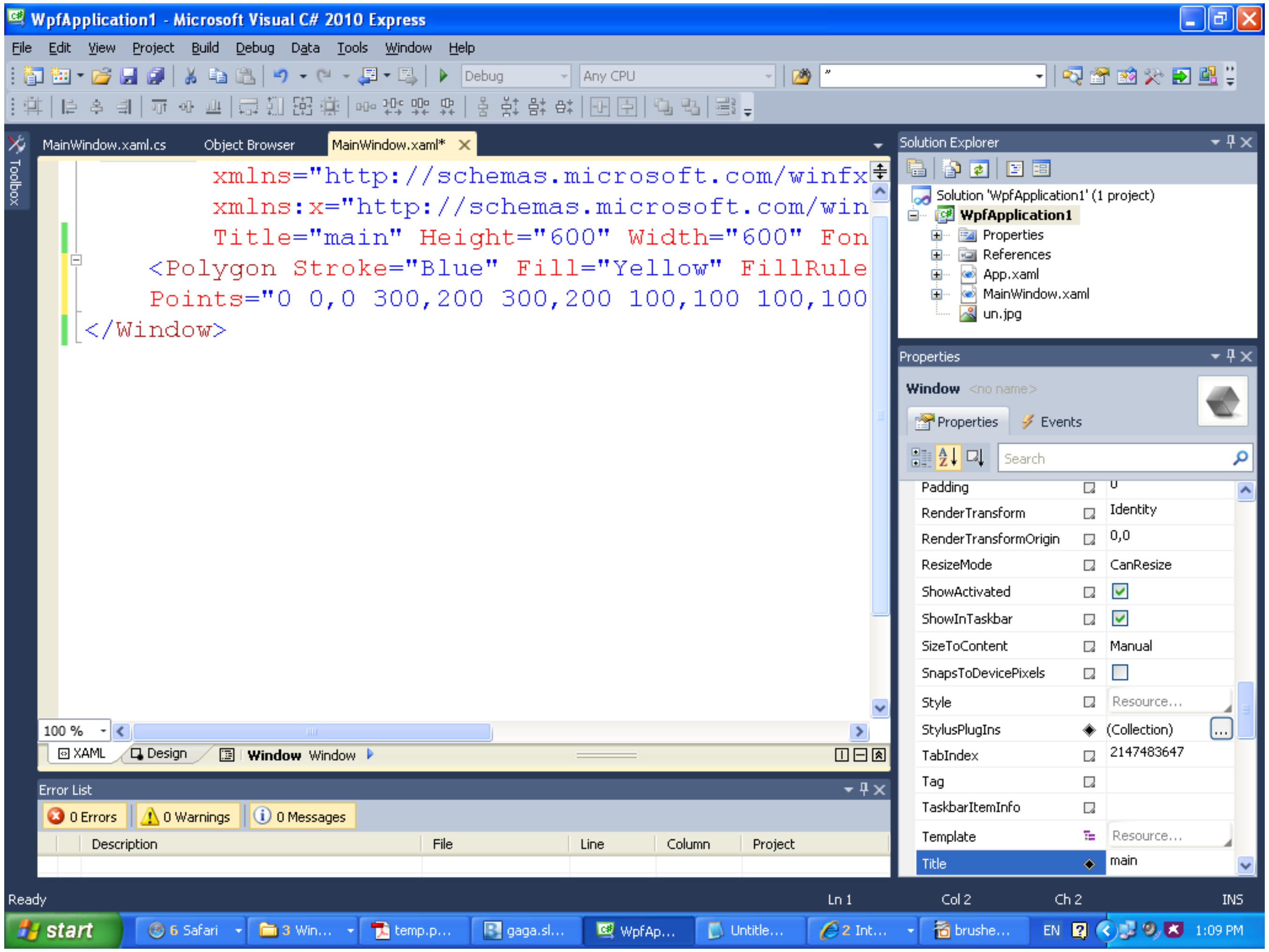


```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pre
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="main" Height="600" Width="600" FontSize="24">
    <Polygon Stroke="Blue" Fill="Yellow" FillRule="EvenOdd"
    Points="0 0,0 300,200 300,200 100,100 100,100 200,300 200,3
</Window>
```

The `FillRule` Attribute (`EvenOdd`)



The `FillRule` Attribute (`EvenOdd`)



```
        xmlns="http://schemas.microsoft.com/winfx
        xmlns:x="http://schemas.microsoft.com/win
        Title="main" Height="600" Width="600" Fon
    <Polygon Stroke="Blue" Fill="Yellow" FillRule
    Points="0 0,0 300,200 300,200 100,100 100,100
</Window>
```

WpfApplication1 - Microsoft Visual C# 2010 Express

Solution 'WpfApplication1' (1 project)
- WpfApplication1
  - Properties
  - References
  - App.xaml
  - MainWindow.xaml
  - un.jpg

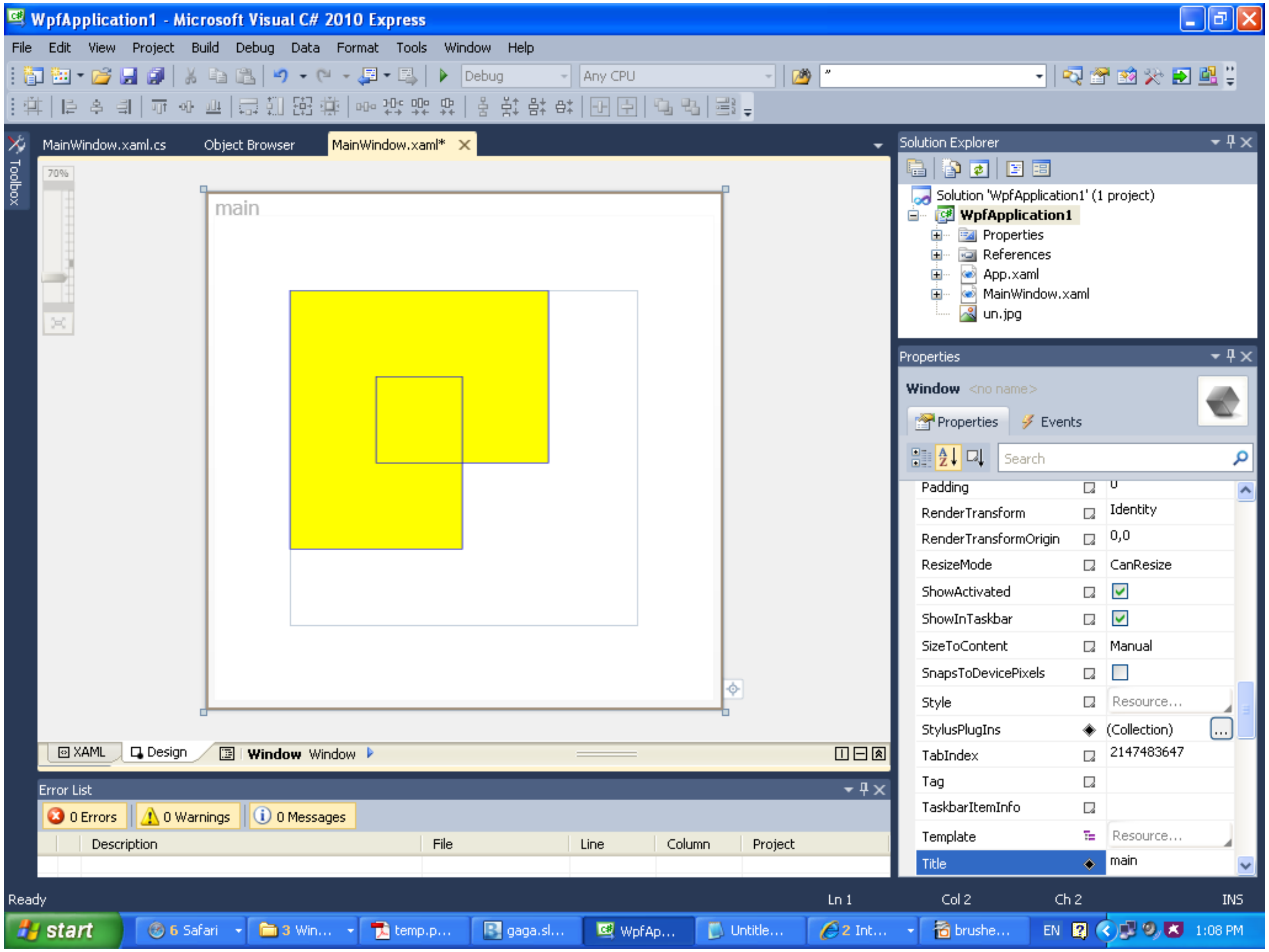
Properties

Window <no name>

Padding	0
RenderTransform	Identity
RenderTransformOrigin	0,0
ResizeMode	CanResize
ShowActivated	✓
ShowInTaskbar	✓
SizeToContent	Manual
SnapsToDevicePixels	☐
Style	Resource...
StylusPlugIns	(Collection)
TabIndex	2147483647
Tag	
TaskbarItemInfo	
Template	Resource...
Title	main

Error List: 0 Errors, 0 Warnings, 0 Messages

The `FillRule` Attribute (`EvenOdd`)



Padding	0
RenderTransform	Identity
RenderTransformOrigin	0,0
ResizeMode	CanResize
ShowActivated	✓
ShowInTaskbar	✓
SizeToContent	Manual
SnapsToDevicePixels	☐
Style	Resource...
StylusPlugIns	(Collection)
TabIndex	2147483647
Tag	
TaskbarItemInfo	
Template	Resource...
Title	main

0 Errors 0 Warnings 0 Messages

The `SpreadMethod` Attribute

- This attribute determines how to fill a drawn area when the brush isn't big enough to cover the entire area.
- This attribute can take the following values: `Pad`, `Reflect` and `Repeat`.
- Using the `Pad` value the remaining of the area will be filled in with the final color.

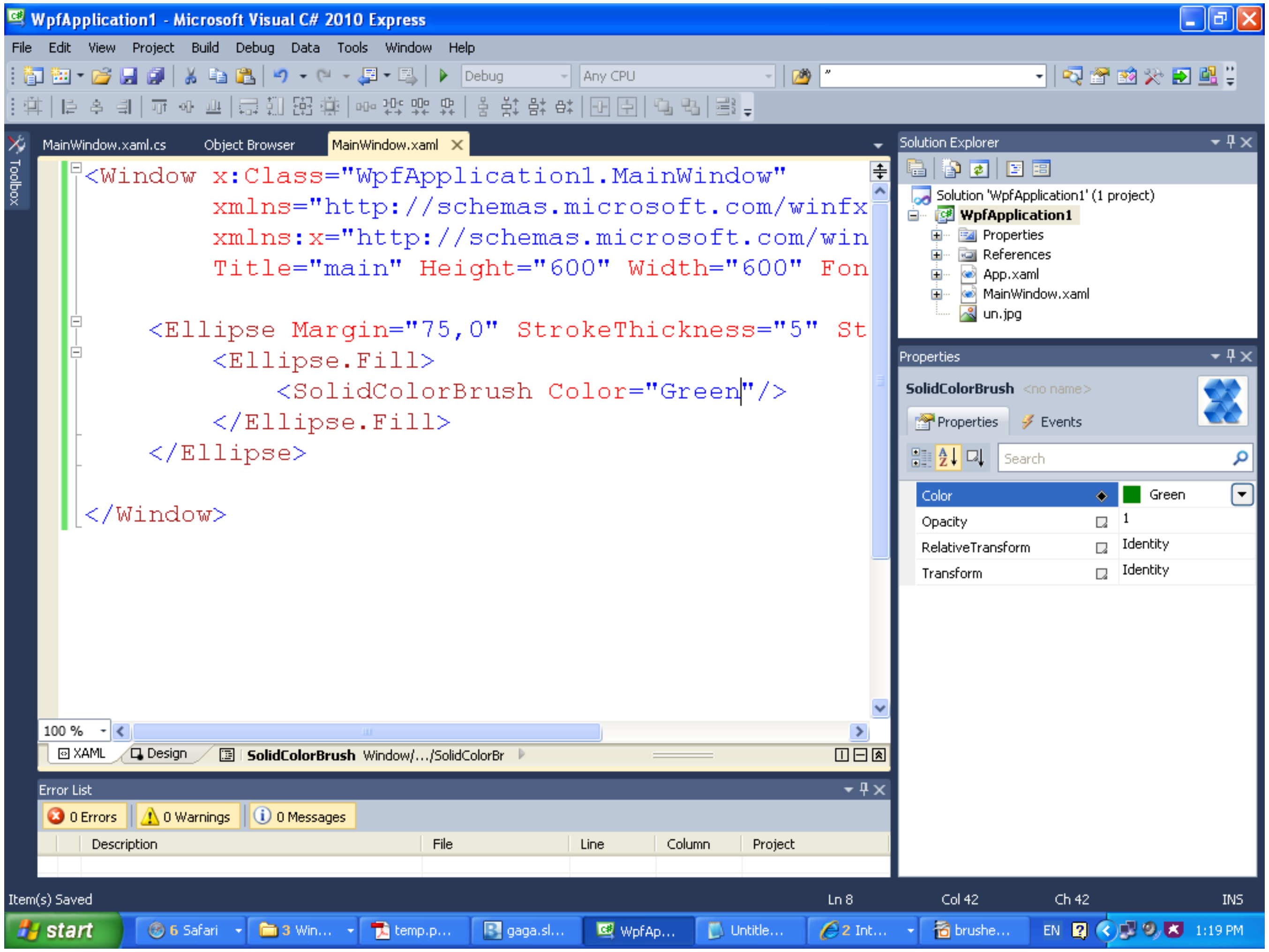
The `SpreadMethod` Attribute

- Using the `Reflect` value the brush will reverse it self and continue to fill in the area. This pattern will repeat until the entire area is filled.
- Using the `Repeat` value the brush will start over and repeat itself again and again till the entire area is filled.

The `SolidColorBrush` Attribute

- This attribute represents a single solid color. We can specify the brush's color either by name (e.g. `Yellow`) or by hexadecimal value (e.g. `#FFAA0033`).

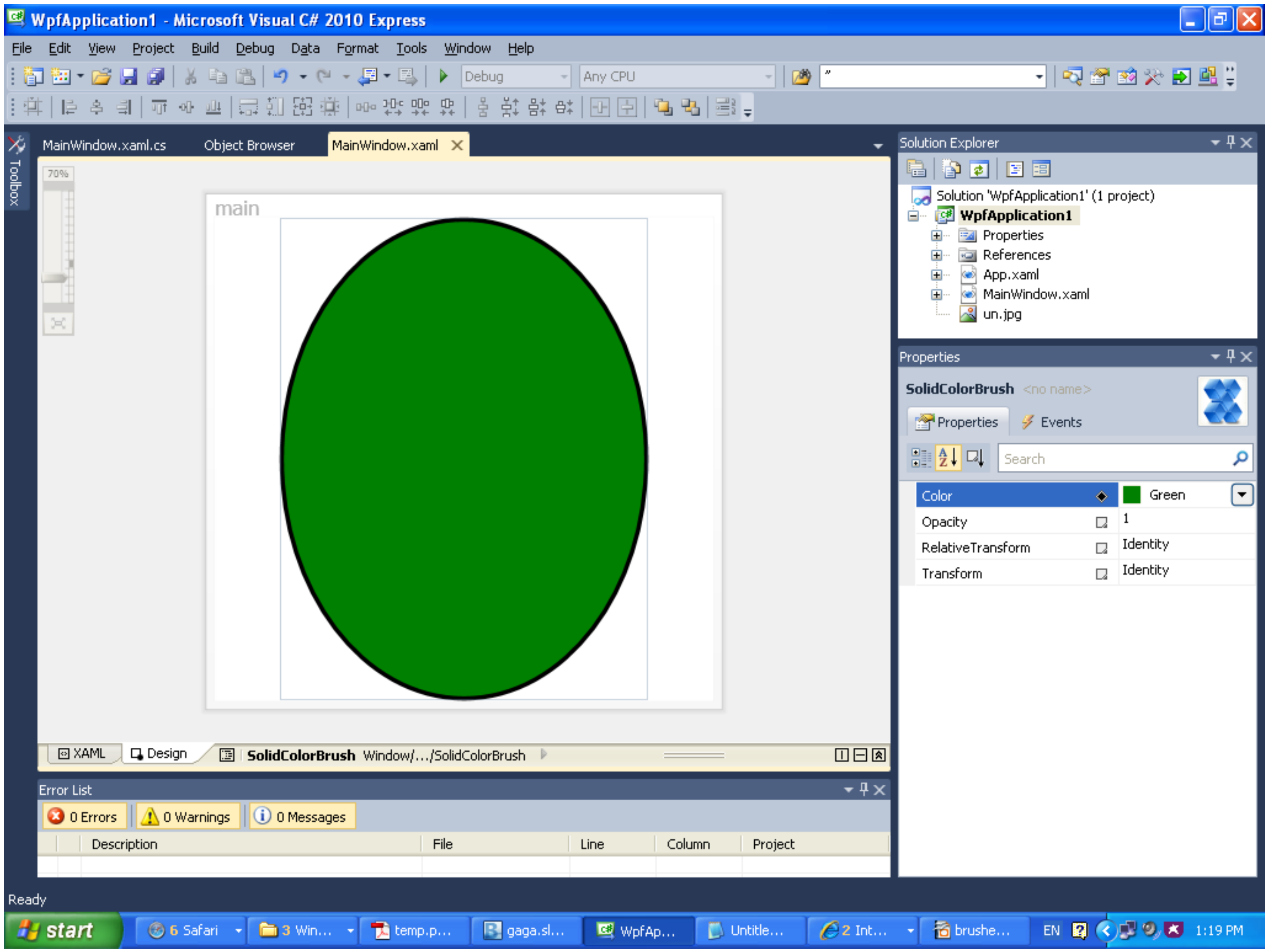
The `SolidColorBrush` Attribute



```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx
        xmlns:x="http://schemas.microsoft.com/win
        Title="main" Height="600" Width="600" Fon

    <Ellipse Margin="75,0" StrokeThickness="5" St
        <Ellipse.Fill>
            <SolidColorBrush Color="Green"/>
        </Ellipse.Fill>
    </Ellipse>

</Window>
```

Color	🟩 Green
Opacity	1
RelativeTransform	Identity
Transform	Identity

The `SolidColorBrush` Attribute


WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Format Tools Window Help
MainWindow.xaml.cs
Object Browser
MainWindow.xaml
main
Solution Explorer
Solution 'WpfApplication1' (1 project)
WpfApplication1
Properties
References
App.xaml
MainWindow.xaml
un.jpg
XAML
Design
SolidColorBrush Window/…/SolidColorBrush
Error List
0 Errors
0 Warnings
0 Messages
Ready

Properties

SolidColorBrush <no name>

Color	🟩 Green
Opacity	1
RelativeTransform	Identity
Transform	Identity

© 2008 Haim Michael

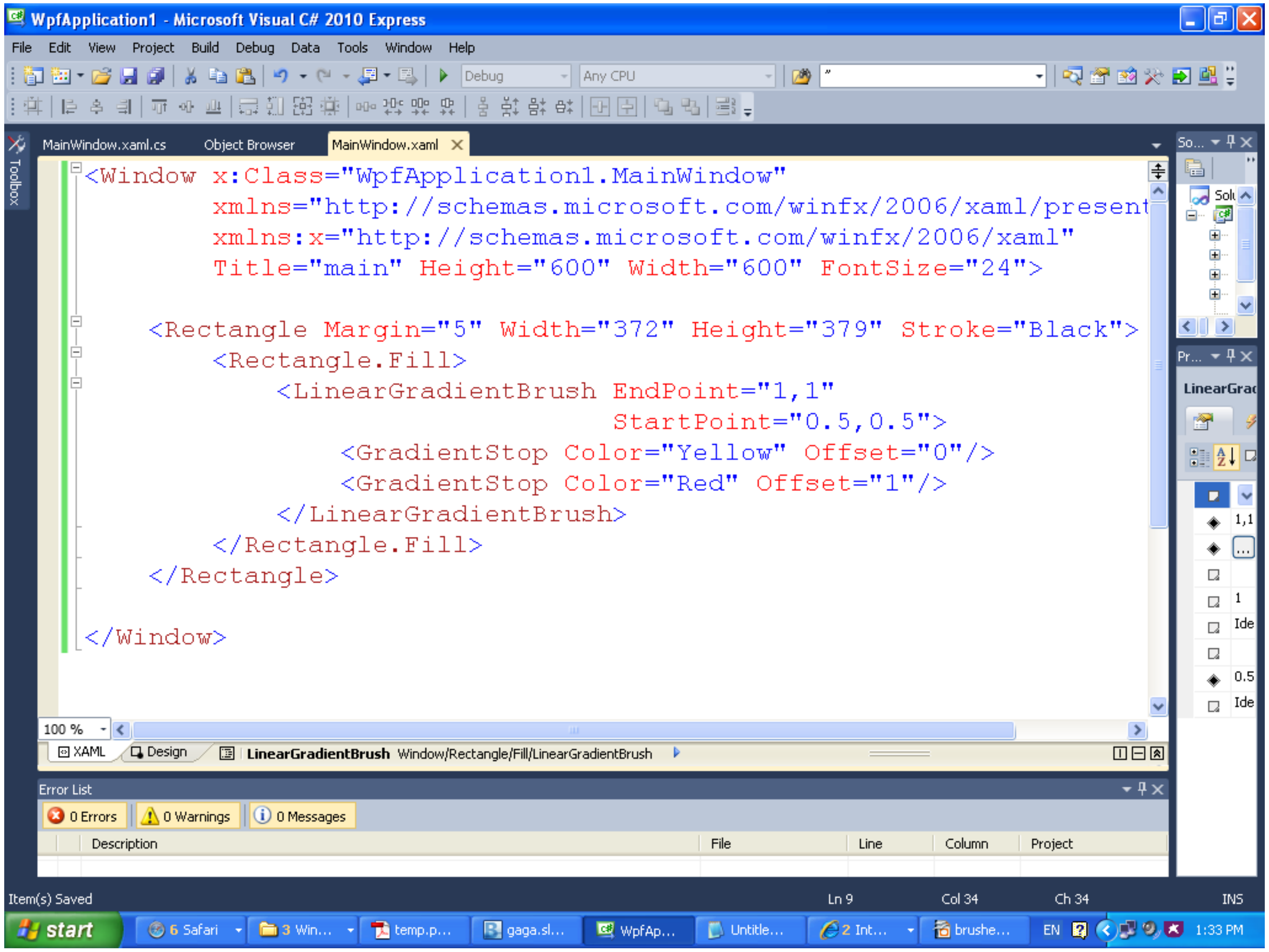
The `LinearGradientBrush` Attribute

- This attribute fills an area with a sequence of colors that smoothly linearly blend with each from one color to the others.
- The `LinearGradientBrush`'s `StartPoint` and `EndPoint` attributes determine where the gradient starts and when it ends.
- The coordinates of these points use a scale in which (0, 0) is the upper left corner and (1,1) is the lower right one.

The `LinearGradientBrush` Attribute

- Collection of `GradientStop` objects describe the exact way the area is filled.
- Each `GradientStop` object describes the exact color the brush should use in a specific point.
- The `Offset` attribute determines on a 0 to 1 scale how far through the brush the color should be positioned.

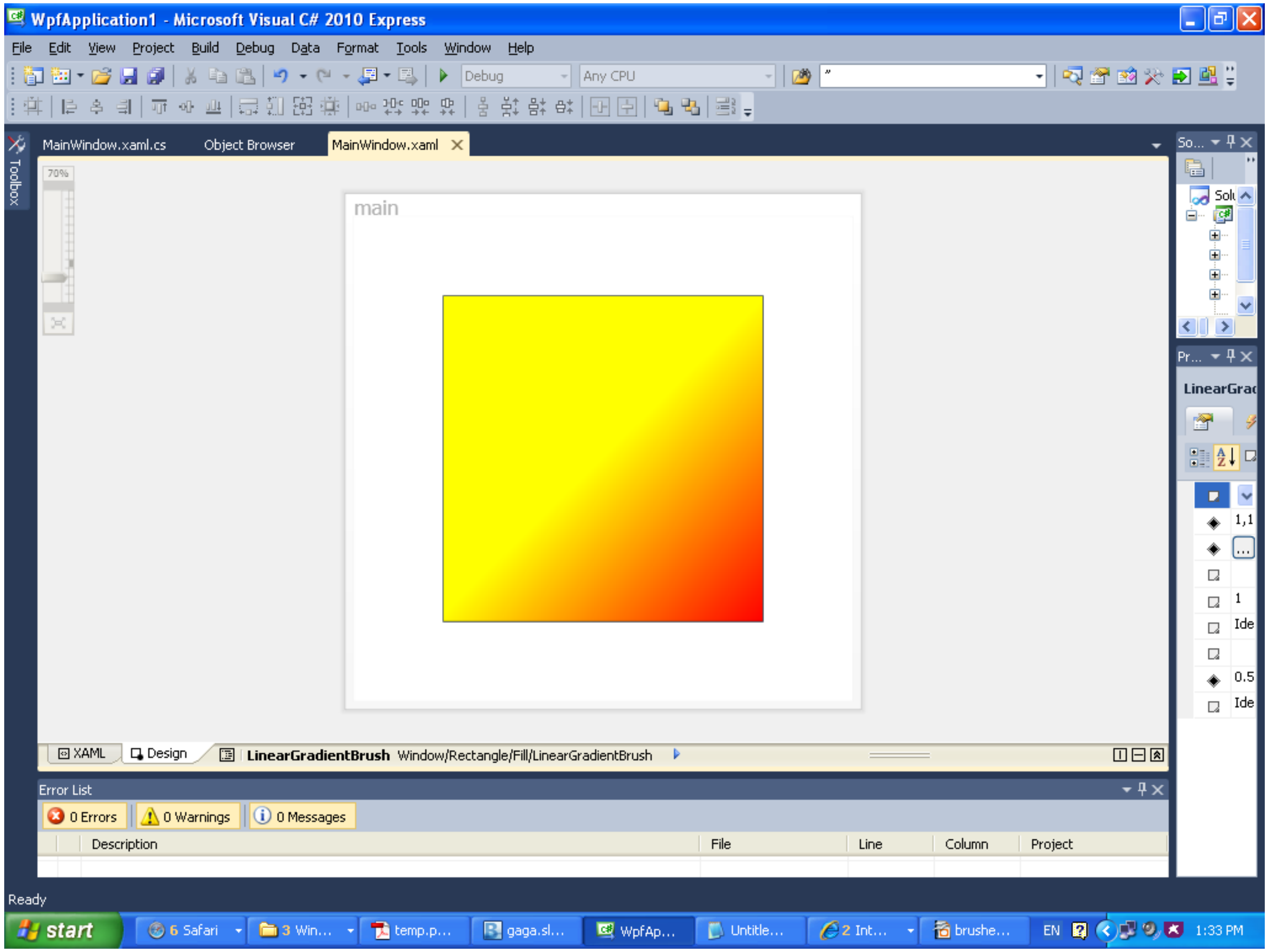
The `LinearGradientBrush` Attribute



```
<Window x:Class="WpfApplication1.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/present
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="main" Height="600" Width="600" FontSize="24">

    <Rectangle Margin="5" Width="372" Height="379" Stroke="Black">
        <Rectangle.Fill>
            <LinearGradientBrush EndPoint="1,1"
                                 StartPoint="0.5,0.5">
                <GradientStop Color="Yellow" Offset="0"/>
                <GradientStop Color="Red" Offset="1"/>
            </LinearGradientBrush>
        </Rectangle.Fill>
    </Rectangle>

</Window>
```

WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Tools Window Help
MainWindow.xaml.cs
Object Browser
MainWindow.xaml
100 %
XAML
Design
LinearGradientBrush Window/Rectangle/Fill/LinearGradientBrush
Error List
0 Errors
0 Warnings
0 Messages
Description File Line Column Project
Item(s) Saved Ln 9 Col 34 Ch 34 INS
start
1:33 PM

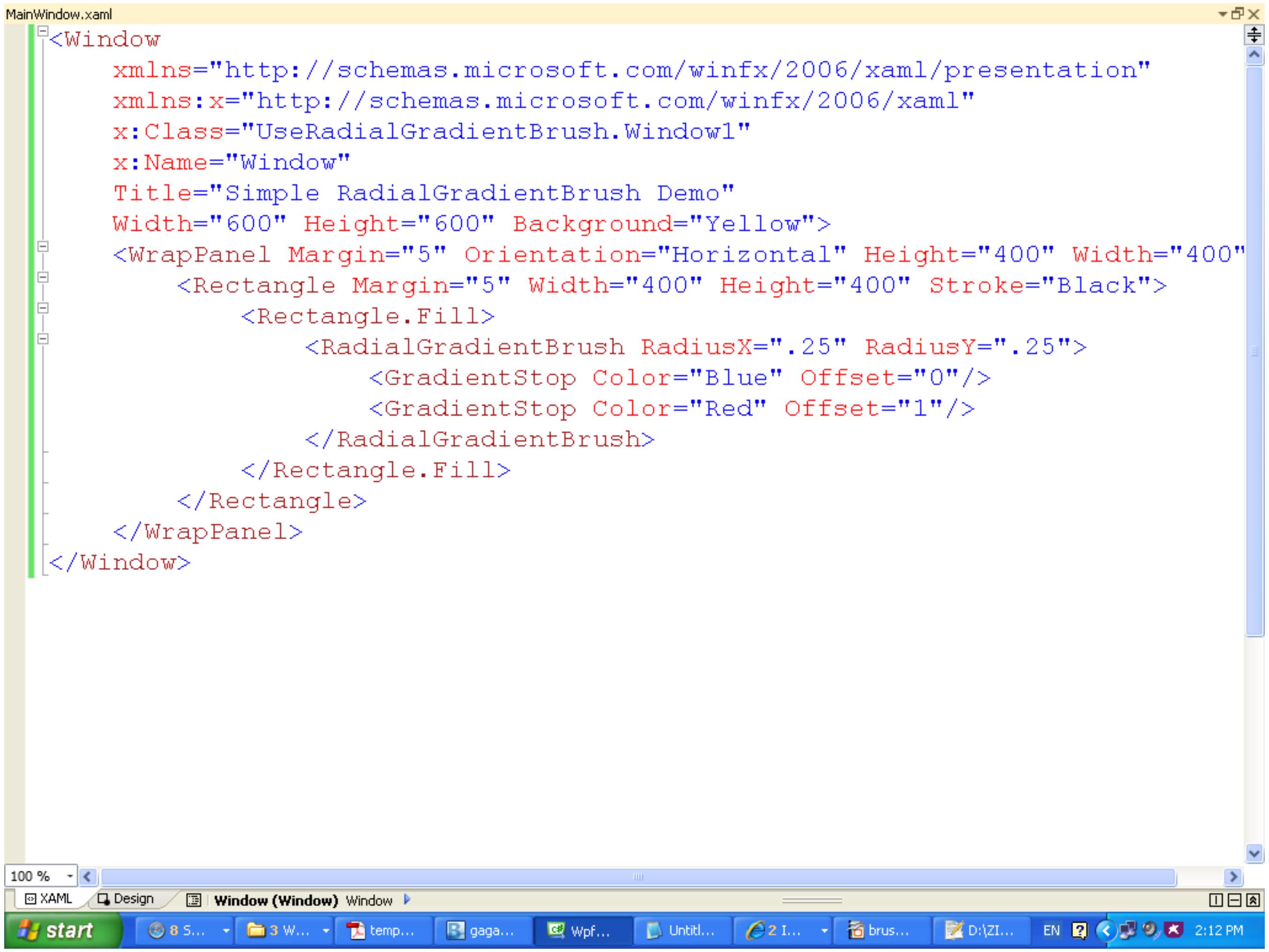
The `LinearGradientBrush` Attribute


WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Format Tools Window Help
Debug
Any CPU
MainWindow.xaml.cs
Object Browser
MainWindow.xaml
main
XAML
Design
LinearGradientBrush Window/Rectangle/Fill/LinearGradientBrush
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Ready
start
1:33 PM

The `RadialGradientBrush` Attribute

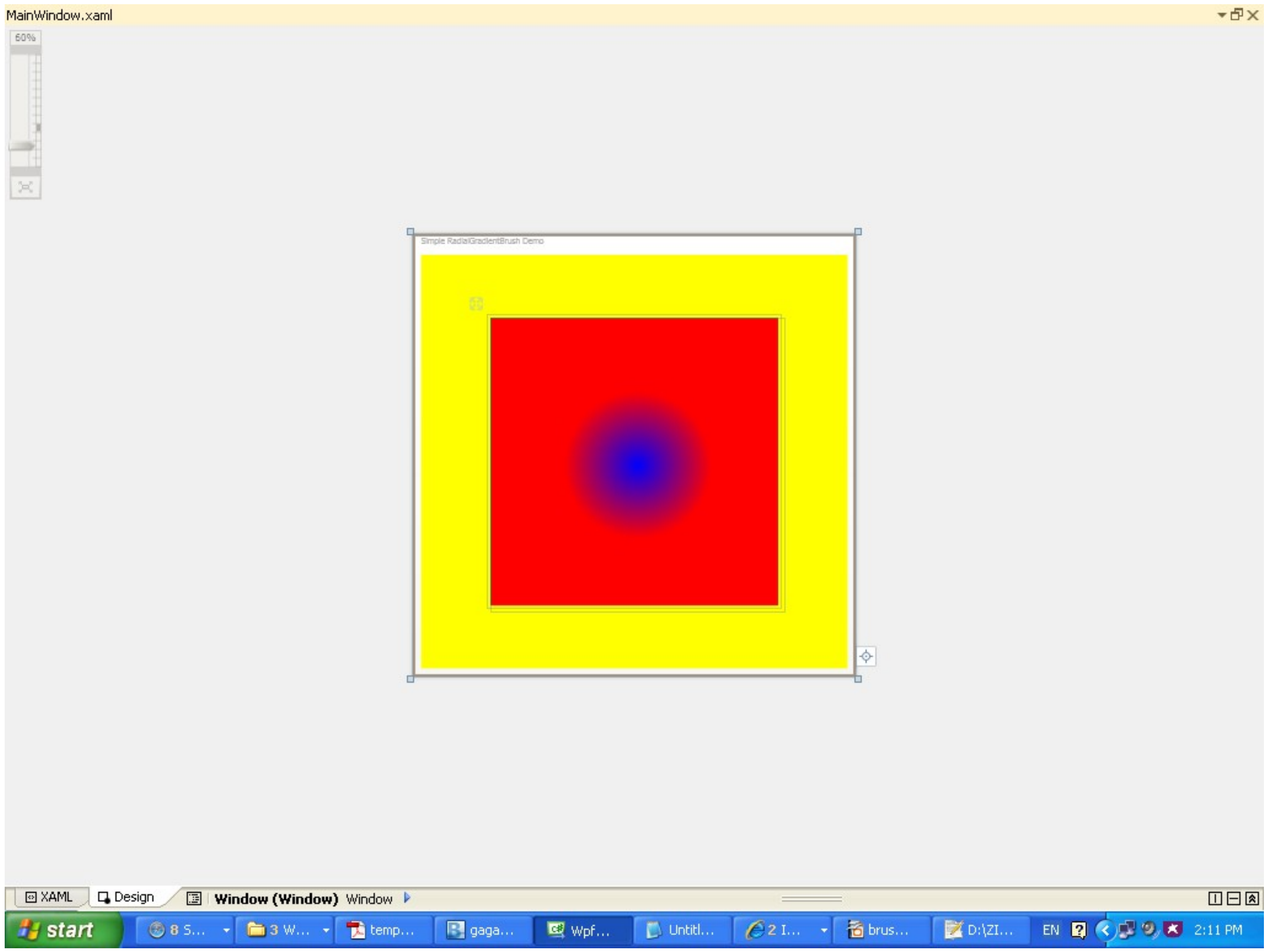
- The `RadialGradientBrush` smoothly blends the colors radiating away from the central point.
- The `GradientOrigin` determines the point from which the colors radiate.
- The `RadiusX` and `RadiusY` attributes determine how far the brush extends horizontally and vertically from the center. These two attributes use the (0,0) to (1,1) coordinate system.

The `RadialGradientBrush` Attribute



```
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="UseRadialGradientBrush.Window1"
    x:Name="Window"
    Title="Simple RadialGradientBrush Demo"
    Width="600" Height="600" Background="Yellow">
    <WrapPanel Margin="5" Orientation="Horizontal" Height="400" Width="400"
        <Rectangle Margin="5" Width="400" Height="400" Stroke="Black">
            <Rectangle.Fill>
                <RadialGradientBrush RadiusX=".25" RadiusY=".25">
                    <GradientStop Color="Blue" Offset="0"/>
                    <GradientStop Color="Red" Offset="1"/>
                </RadialGradientBrush>
            </Rectangle.Fill>
        </Rectangle>
    </WrapPanel>
</Window>
```

The `RadialGradientBrush` Attribute



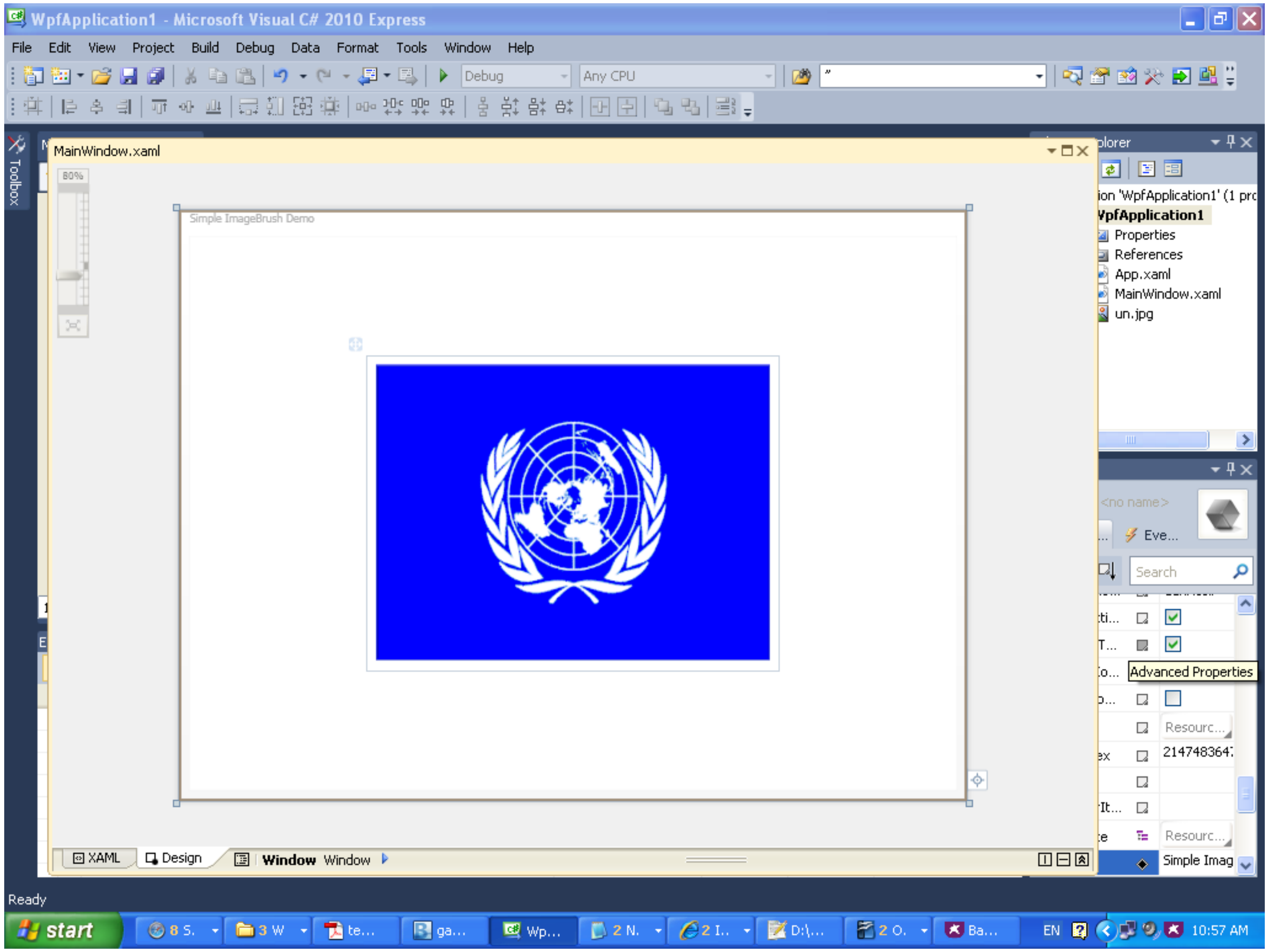
The `ImageBrush` Class

- This class fills in the area with an image. We can import any image into our project and refer it using the `ImageSource` attribute.

The `ImageBrush` Class

```
<Window  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
         Title="Simple ImageBrush Demo"
         Width="800" Height="600" Background="White"
         Loaded="Window_Loaded">

    <WrapPanel Height="322" Width="420">

        <Rectangle  Width="400" Height="300"
                    Stroke="Blue" StrokeThickness="2"
                    HorizontalAlignment="Left"
                    VerticalAlignment="Top"
                    Margin="10,10,0,0">
            <Rectangle.Fill>
                <ImageBrush ImageSource="un.jpg"/>
            </Rectangle.Fill>
        </Rectangle>

    </WrapPanel>

</Window>
```

The `ImageBrush` Class


WpfApplication1 - Microsoft Visual C# 2010 Express
File Edit View Project Build Debug Data Format Tools Window Help
Debug
Any CPU
MainWindow.xaml
Simple ImageBrush Demo
XAML
Design
Window Window
Properties
References
App.xaml
MainWindow.xaml
un.jpg
Advanced Properties
Search
Ready
start
10:57 AM