Dock

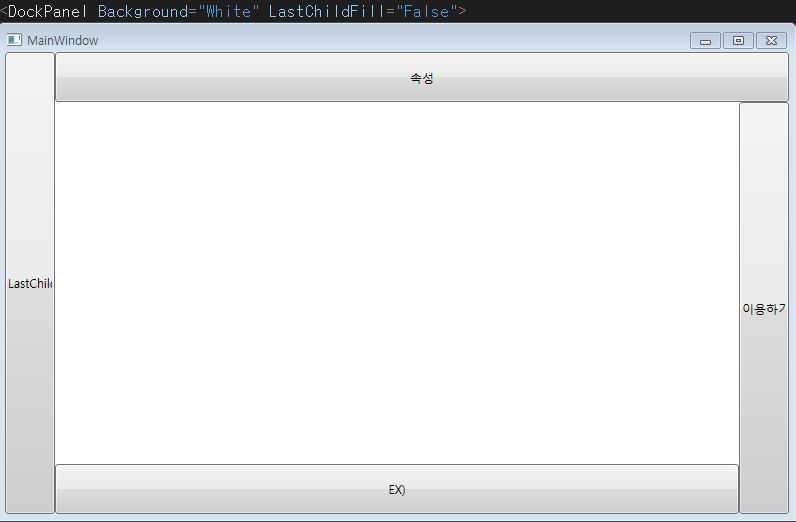
Dock이란 전체 레이아웃을 표현할 때 주로 사용하며 패널에서 지정한 방향으로 항목을 배치시킨다. 자식 요소를 지정한 방향으로 배치시키고 마지막 자식 요소는 남은 공간을 채우도록 배치하는 것이 기본 동작이다.

Dock은 Canvas와 같이 결합 속성을 사용하며 자식 요소들이 지정한 방향에 배치된다.

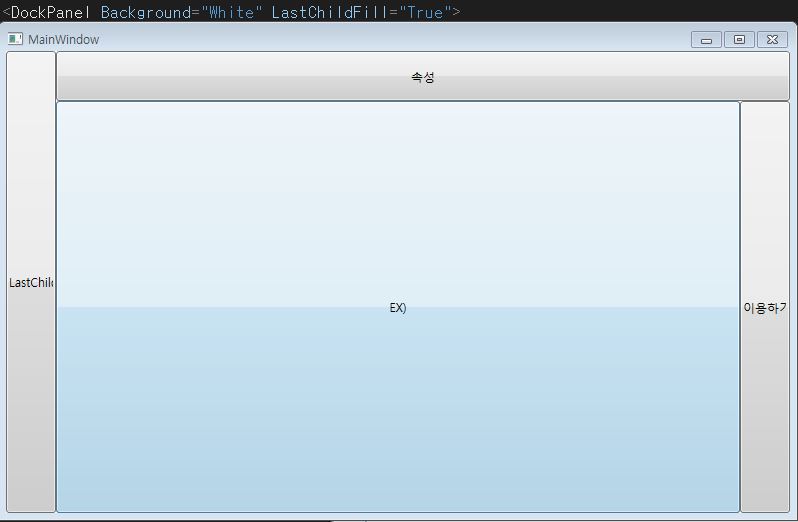
Dock에는 LastChildFill이란 속성이 있다. 이 속성을 False로 설정하면 Dock패널의 남은 공간을 채우지 않고 남긴다. ( 하지만 True로 설정 할 경우 남은 공간을 채운다. )

**예제 0> LastChildFill 속성 이용해보기**

**-False**

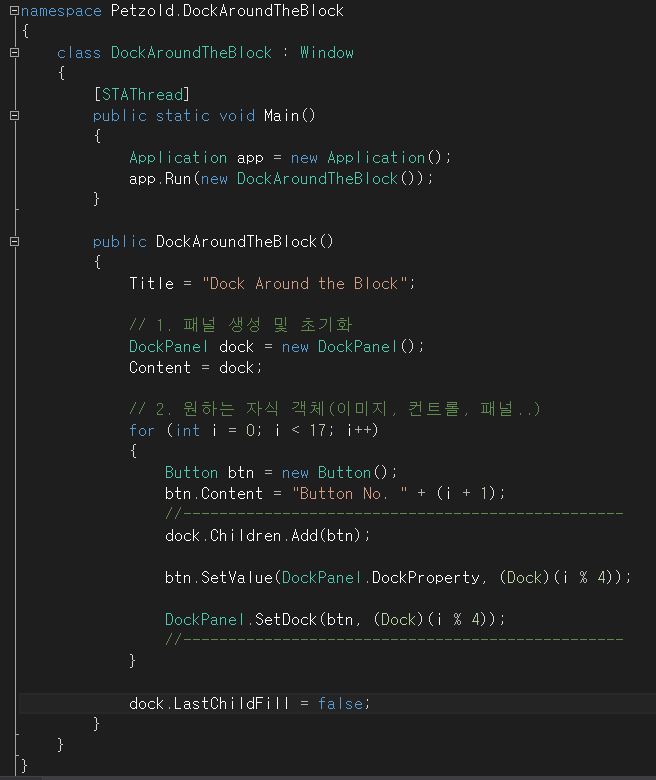
****

**-True**

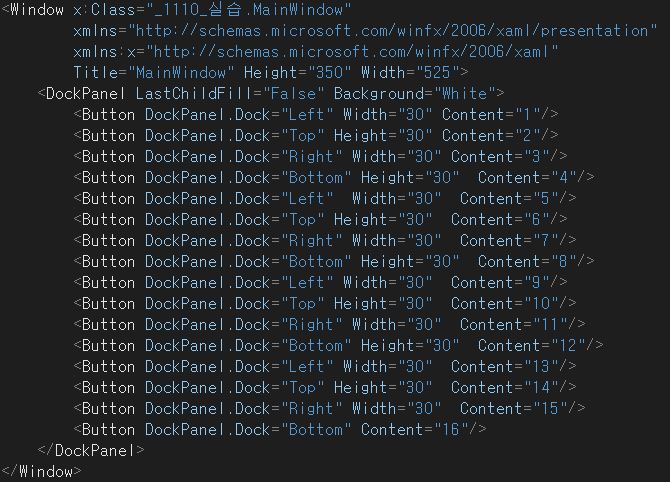
****

**예제 1> DockPanel의 자식 요소**

이것은 C#으로 된 DockPanel의 자식요소이다.



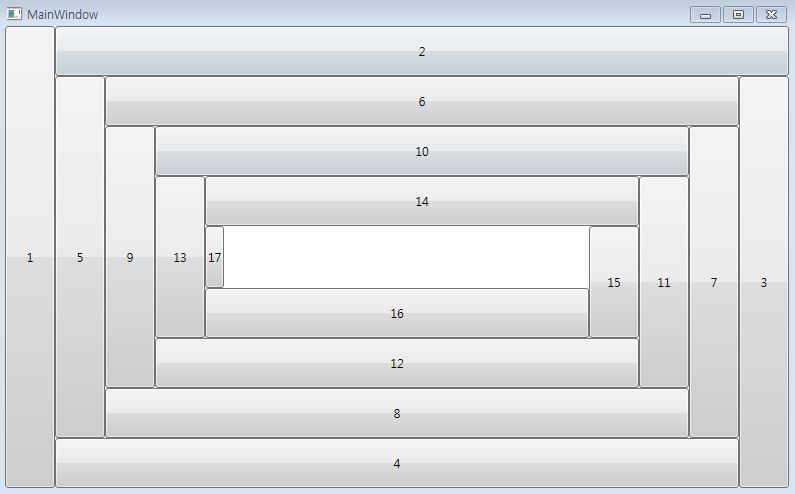
밑의 사진은 C#의 코드를 XAML로 바꾼 코드이다.



C#코드에서 DockAroundTheBlock()생성자에서 dockpanel을 만들고 그 안에 for문으로 버튼을 만들어서 넣었다.

그것을 XAML로 바꾸는 방법은 버튼을 각각 만들어 주고 DockPanel로 지정한 방향으로 버튼을 배치한다.

**실행결과 >**

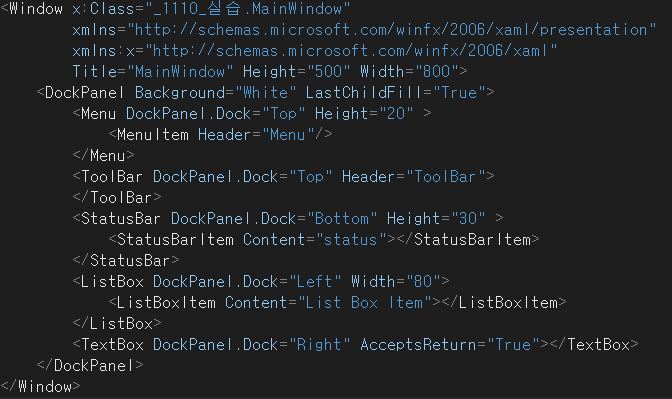
****

**예제 2> DockPanel 만들기**

C#으로 DockPanel 만들기

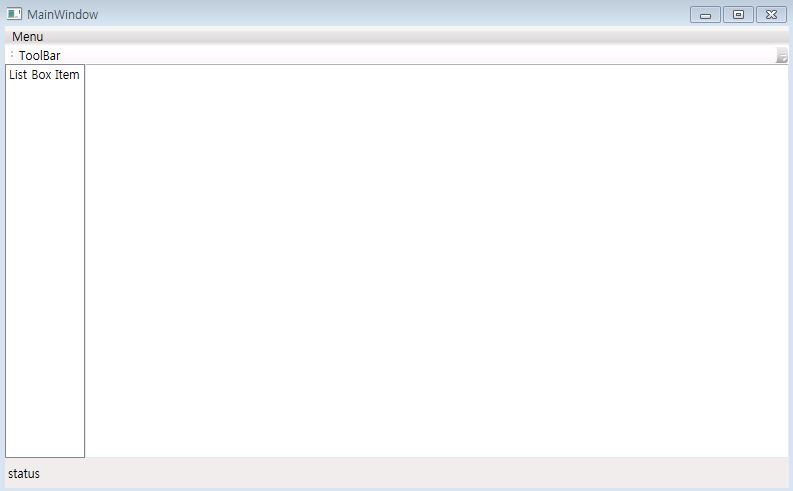
****

XAML로 바꾼 코드



C#에서 보면 또한 생성자에서 Menu, ToolBar, StatusBar, ListBox를 만들었다. 이것을 XAML코드로 바꾸었다.

**실행결과 >**

****

Grid

Grid는 패널의 한 종류이다. Grid패널은 패널 중 가장 강한 패널이다. 자식 요소들 간의 간격과 배율을 유지시킬 수 있다. Grid는 각 요소들을 배치 할 때 행과 열로 이루어진 셀 안에 각행, 열의 위치에 요소를 배치하거나 여러 셀에 자식 요소를 배치할 수 있다.

다음은 Grid의 속성에 대한 표이다.

|  |  |
| --- | --- |
| 속성 명 | 설명 |
| Grid.RowDefinitions | 행의 개수를 결정하는 속성이다.  속성 안의 RowDefinition의 개수만큼 행이 추가된다. |
| Grid.ColumnDefinitions | 열의 개수를 결정하는 속성이다.  속성 안의 ColumnDefintion의 개수만큼 행이 추가된다. |
| Grid.RowSpan | 입력한 행 개수만큼을 하나의 셀로 사용할 수 있다. |
| Grid.ColumnSpan | 입력한 열 개수만큼을 하나의 셀로 사용할 수 있다. |
| GridSplitter | 열의 크기를 동적으로 변경할 수 있다. |
| Width,Height | ColumnDefintion안에 사용할 수 있으며 열의 크기를 조정할 수 있다.  Auto(자동),\*(배율) 등을 넣을 수 있다. |
| Grid.IsSharedSizeScope | 서로 다른 grid의 열을 하나의 grid열처럼 다루게 해주는  크기 공유 그룹이다. |

다음은 Grid를 이용하여 작성한 시간 표출 예제이다.

<Window x:Class="CalculateYourLife.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Calculate Your Life" Width="275" Height="170" >

<Grid ShowGridLines="True">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Label Content="Begin Date: " HorizontalAlignment="Left" VerticalAlignment="Top" Margin="5,10,0,0" Grid.ColumnSpan="2"/>

<Label Content="End Date: " HorizontalAlignment="Left" VerticalAlignment="Top" Margin="5,7,0,0" Grid.Row="1"/>

<Label Content="Life Years: " HorizontalAlignment="Left" VerticalAlignment="Top" Margin="5,6,0,0" Grid.Row="2"/>

<TextBox Name ="Text1" HorizontalAlignment="Left" Height="23" Margin="10,9,-130,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="172" Grid.Column="1" TextChanged="TextBoxOnTextChanged"/>

<TextBox Name ="Text2" HorizontalAlignment="Left" Height="23" Margin="10,9,-129,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="172" Grid.Column="1" Grid.Row="1" TextChanged="TextBoxOnTextChanged"/>

<Label Name ="Label1" Content="" Grid.Column="1" HorizontalAlignment="Left" Margin="10,6,0,0" Grid.Row="2" VerticalAlignment="Top"/>

</Grid>

</Window>

(XAML)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace CalculateYourLife

{

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

Text1.Text = DateTime.Parse("1980/ 1/ 1").ToShortDateString();

Text2.Text = DateTime.Now.ToShortDateString();

}

private void TextBoxOnTextChanged(object sender, TextChangedEventArgs e)

{

DateTime dtBeg, dtEnd;

if (DateTime.TryParse(Text1.Text, out dtBeg) &&

DateTime.TryParse(Text2.Text, out dtEnd))

{

int iYears = dtEnd.Year - dtBeg.Year;

int iMonths = dtEnd.Month - dtBeg.Month;

int iDays = dtEnd.Day - dtBeg.Day;

if (iDays < 0)

{

iDays += DateTime.DaysInMonth(dtEnd.Year,

1 + (dtEnd.Month + 10) % 12);

iMonths -= 1;

}

if (iMonths < 0)

{

iMonths += 12;

iYears -= 1;

}

Label1.Content =

String.Format("{0} year{1}, {2} month{3}, {4} day{5}",

iYears, iYears == 1 ? "" : "s",

iMonths, iMonths == 1 ? "" : "s",

iDays, iDays == 1 ? "" : "s");

}

else

{

Label1.Content = "";

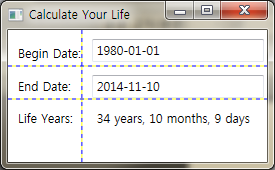
}

}

}

}

(cs)



RowDefinition에는 Height속성을 Auto로 주고 ColumnDefinition에는 Widith속성을 Auto로 주었다.

그 결과 Label의 길이에 맞게 Colum의 길이가 설정된걸 볼 수 있다.

다음은 한 StackPanel안에 두개의 Grid를 삽입하여 만든 정보 입력 프로그램이다.

<Window x:Class="Enter\_the\_Grid.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Enter the Grid" SizeToContent="WidthAndHeight" Width="300">

<StackPanel>

<Grid Margin="5">

<Grid.RowDefinitions >

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"></ColumnDefinition>

<ColumnDefinition Width="100\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Label Content="First name:" HorizontalAlignment="Left" VerticalAlignment="Top" Grid.Row="0" Grid.Column="0" />

<Label Content="Last name:" HorizontalAlignment="Left" VerticalAlignment="Top" Grid.Row="1" Grid.Column="0" />

<Label Content="Social security number:" HorizontalAlignment="Left" VerticalAlignment="Top" Grid.Row="2" Grid.Column="0" />

<Label Content="Credit card number:" HorizontalAlignment="Left" VerticalAlignment="Top" Grid.Row="3" Grid.Column="0" />

<Label Content="Other personal stuff:" HorizontalAlignment="Left" VerticalAlignment="Top" Grid.Row="4" Grid.Column="0" />

<TextBox HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Grid.Row="0" Grid.Column="1" Margin="5"/>

<TextBox HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Grid.Row="1" Grid.Column="1" Margin="5"/>

<TextBox HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Grid.Row="2" Grid.Column="1" Margin="5"/>

<TextBox HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Grid.Row="3" Grid.Column="1" Margin="5"/>

<TextBox HorizontalAlignment="Left" Height="23" TextWrapping="Wrap" VerticalAlignment="Top" Width="120" Grid.Row="4" Grid.Column="1" Margin="5"/>

</Grid>

<Grid Margin="10">

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Button Content="Submit" IsDefault="True" HorizontalAlignment="Center" Width="75" Click="Click"/>

<Button Content="Cancel" IsCancel="True" HorizontalAlignment="Center" Width="75" Grid.Column="1" Click="Click"/>

</Grid>

</StackPanel>

</Window>

(XAML)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Enter\_the\_Grid

{

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Click(object sender, RoutedEventArgs e)

{

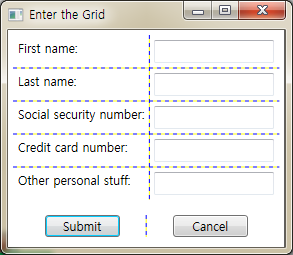
Close();

}

}

}

(cs)



한 개의 StackPanel안에 Grid가 2개 들어가는 걸 볼 수 있다.

위에 입력 부분에 나눠진 선들이 하나의 Grid고 그 밑에 버튼만을 나누는 선이 또 하나의 Grid이다.

다음은 ColumnSpan을 이용해서 2개의 Column을 합친 입력 프로그램이다.

<Window x:Class="Span\_the\_Cells.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MainWindow" SizeToContent="WidthAndHeight">

<Grid Margin="5" ShowGridLines="True">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"></ColumnDefinition>

<ColumnDefinition Width="100\*"></ColumnDefinition>

<ColumnDefinition Width="Auto"></ColumnDefinition>

<ColumnDefinition Width="Auto"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Label Content="First name:" VerticalAlignment="Center" Grid.Row="0" Grid.Column="0" />

<Label Content="Last name:" VerticalAlignment="Center" Grid.Row="1" Grid.Column="0" />

<Label Content="Social security number:" VerticalAlignment="Center" Grid.Row="2" Grid.Column="0" />

<Label Content="Credit card number:" VerticalAlignment="Center" Grid.Row="3" Grid.Column="0" />

<Label Content="Other personal stuff:" VerticalAlignment="Center" Grid.Row="4" Grid.Column="0" />

<TextBox Height="23" TextWrapping="Wrap" Margin="5" Width="120" Grid.Row="0" Grid.Column="1" Grid.ColumnSpan="3"/>

<TextBox Height="23" TextWrapping="Wrap" Margin="5" Width="120" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="3"/>

<TextBox Height="23" TextWrapping="Wrap" Margin="5" Width="120" Grid.Row="2" Grid.Column="1" Grid.ColumnSpan="3"/>

<TextBox Height="23" TextWrapping="Wrap" Margin="5" Width="120" Grid.Row="3" Grid.Column="1" Grid.ColumnSpan="3"/>

<TextBox Height="23" TextWrapping="Wrap" Margin="5" Width="120" Grid.Row="4" Grid.Column="1" Grid.ColumnSpan="3"/>

<Button Content="Submit" IsDefault="True" Margin="5" Width="75" Grid.Row="5" Grid.Column="2" Click="Click"/>

<Button Content="Cancel" IsCancel="True" Margin="5" Width="75" Grid.Row="5" Grid.Column="3" Click="Click"/>

</Grid>

</Window>

(XML)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Span\_the\_Cells

{

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void Click(object sender, RoutedEventArgs e)

{

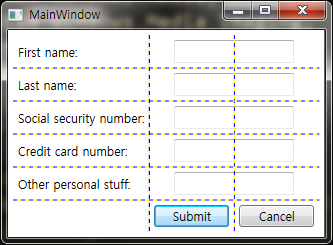
Close();

}

}

}

(CS)



실행 된 화면을 보면 텍스트 박스는 2개의 Column을 사용하는 것을 볼 수 있다.

다음은 GridSplitter를 사용하여 셀의 간격을 나눈 프로그램이다.

<Window x:Class="Split\_the\_Client.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Split the Client" Height="350" Width="525">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition Width="Auto"></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<Button Content="Button No. 1" Grid.Row="0" Grid.Column="0"/>

<GridSplitter ShowsPreview="True" HorizontalAlignment="Center" VerticalAlignment="Stretch" Width="6" Grid.Row="0" Grid.Column="1"></GridSplitter>

<Grid Grid.Row="0" Grid.Column="2">

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Button Content="Button No. 2"/>

<GridSplitter ShowsPreview="True" HorizontalAlignment="Stretch" VerticalAlignment="Center" Height="6" Grid.Row="1" Grid.Column="0"/>

<Button Content="Button No. 3" Grid.Row="2" Grid.Column="0"/>

</Grid>

</Grid>

</Window>

(XAML)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Split\_the\_Client

{

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

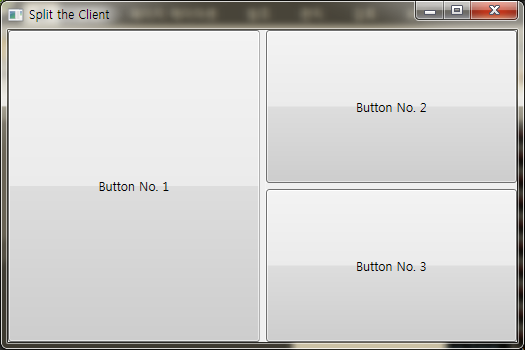
InitializeComponent();

}

}

}

(cs)



실행 화면이다. 버튼 1번과 버튼2, 3번 사이에 간격이 만들어졌고, 2번과 3번 사이에도 간격이 만만들어진 것을 확인할 수 있다.

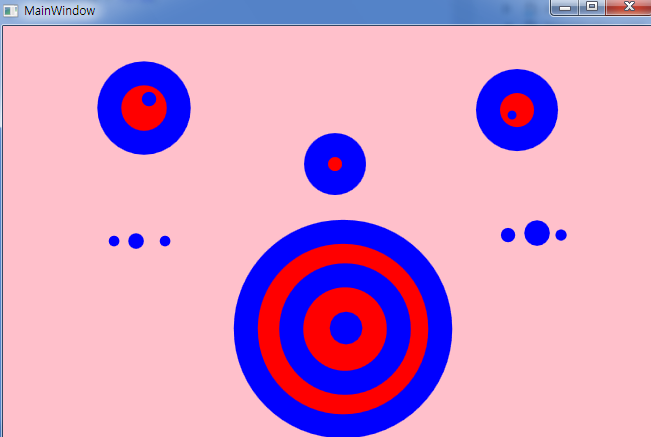
Chapter 09. 라우팅

09-1. DrawCircles

다양한 크기의 원을 그리고, 원하는 곳으로 이동시킬 수 있는

DrawCircles 프로젝트를 만들어 봅시다.

(완성 사진)



(한눈에 보는 메서드들(CS형식))



실습) CS 형식으로 작성된 코드입니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

namespace Petzold.DrawCircles

{

public class DrawCircles : Window

{

Canvas canv;

// 그리는 것과 관련된 필드

bool isDrawing;

Ellipse elips;

Point ptCenter;

// 드래깅과 관련된 필드

bool isDragging;

FrameworkElement elDragging;

Point ptMouseStart, ptElementStart;

[STAThread]

public static void Main()

{

Application app = new Application();

app.Run(new DrawCircles());

}

public DrawCircles()

{

Title = "Draw Circles";

Content = canv = new Canvas();

}

protected override void OnMouseLeftButtonDown(MouseButtonEventArgs args)

{

base.OnMouseLeftButtonDown(args);

if (isDragging)

return;

// 새로운 타원 객체를 생성하고 캔버스에 추가

ptCenter = args.GetPosition(canv); //최초의 클릭으로 원의 중심 지정.

elips = new Ellipse();

elips.Stroke = SystemColors.WindowTextBrush;

elips.StrokeThickness = 1;

elips.Width = 0;

elips.Height = 0;

canv.Children.Add(elips);

Canvas.SetLeft(elips, ptCenter.X);

Canvas.SetTop(elips, ptCenter.Y);

// 마우스를 캡처하고, 앞으로의 이벤트를 준비

CaptureMouse();

isDrawing = true;

}

protected override void OnMouseRightButtonDown(MouseButtonEventArgs args)

{

base.OnMouseRightButtonDown(args);

if (isDrawing)

return;

// 클릭된 엘리먼트를 얻어오고, 앞으로의 이벤트를 준비

ptMouseStart = args.GetPosition(canv);

elDragging = canv.InputHitTest(ptMouseStart) as FrameworkElement;

if (elDragging != null)

{

ptElementStart = new Point(Canvas.GetLeft(elDragging),

Canvas.GetTop(elDragging));

isDragging = true;

}

}

protected override void OnMouseDown(MouseButtonEventArgs args)

{

base.OnMouseDown(args);

if (args.ChangedButton == MouseButton.Middle)

{

Shape shape = canv.InputHitTest(args.GetPosition(canv)) as Shape;

if (shape != null)

shape.Fill = (shape.Fill == Brushes.Red ?

Brushes.Transparent : Brushes.Red);

}

}

protected override void OnMouseMove(MouseEventArgs args)

{

base.OnMouseMove(args);

Point ptMouse = args.GetPosition(canv);

// 타원의 크기 재조정

if (isDrawing)

{

double dRadius = Math.Sqrt(Math.Pow(ptCenter.X - ptMouse.X, 2) + // 마우스를 움직일 때는 마우스 커서 위치를 추적해 경계선을 지정한다.

Math.Pow(ptCenter.Y - ptMouse.Y, 2));

Canvas.SetLeft(elips, ptCenter.X - dRadius);

Canvas.SetTop(elips, ptCenter.Y - dRadius);

elips.Width = 2 \* dRadius;

elips.Height = 2 \* dRadius;

}

// 타원을 이동

else if (isDragging)

{

Canvas.SetLeft(elDragging,

ptElementStart.X + ptMouse.X - ptMouseStart.X);

Canvas.SetTop(elDragging,

ptElementStart.Y + ptMouse.Y - ptMouseStart.Y);

}

}

protected override void OnMouseUp(MouseButtonEventArgs args)

{

base.OnMouseUp(args);

// 그리기 동작을 종료

if (isDrawing && args.ChangedButton == MouseButton.Left)

{

elips.Stroke = Brushes.Blue;

elips.StrokeThickness = Math.Min(24, elips.Width / 2);

elips.Fill = Brushes.Red;

isDrawing = false;

ReleaseMouseCapture();

}

// 드래깅을 종료

else if (isDragging && args.ChangedButton == MouseButton.Right)

{

isDragging = false;

}

}

protected override void OnTextInput(TextCompositionEventArgs args)

{

base.OnTextInput(args);

// Esc 키를 누르면 그리기나 드래깅을 종료

if (args.Text.IndexOf('\x1B') != -1)

{

if (isDrawing)

ReleaseMouseCapture();

else if (isDragging)

{

Canvas.SetLeft(elDragging, ptElementStart.X);

Canvas.SetTop(elDragging, ptElementStart.Y);

isDragging = false;

}

}

}

protected override void OnLostMouseCapture(MouseEventArgs args)

{

base.OnLostMouseCapture(args);

// 그리기의 비정상적 종료 : 자식 타원을 제거

if (isDrawing)

{

canv.Children.Remove(elips);

isDrawing = false;

}

}

}

}

XAML 형식으로 작성된 코드입니다.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

XAML 코드

<Window x:Class="Routing\_DrawCircles.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="MainWindow" Height="350" Width="525">

<Canvas Name="canvas" Background="Pink" MouseLeftButtonDown="canvas\_MouseLeftButtonDown" MouseRightButtonDown="canvas\_MouseRightButtonDown" MouseDown="canvas\_MouseDown" MouseMove="canvas\_MouseMove" MouseUp="canvas\_MouseUp" TextInput="canvas\_TextInput" LostMouseCapture="canvas\_LostMouseCapture">

</Canvas>

</Window>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

실습) XAML 형식의 CS 코드

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Routing\_DrawCircles

{

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

public partial class MainWindow : Window

{

Ellipse elips;

// 그리는 것과 관련된 필드

bool isDrawing;

Point ptCenter;

// 드래깅과 관련된 필드

bool isDragging;

FrameworkElement elDragging;

Point ptMouseStart, ptElementStart;

public MainWindow()

{

InitializeComponent();

}

[STAThread]

private void canvas\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

base.OnMouseLeftButtonDown(e);

if (isDragging)

return;

ptCenter = e.GetPosition(canvas); //최초의 클릭으로 원의 중심 지정.

elips = new Ellipse();

// 새로운 타원 객체를 생성하고 캔버스에 추가

elips.Stroke = SystemColors.WindowTextBrush;

elips.StrokeThickness = 1;

elips.Width = 0;

elips.Height = 0;

canvas.Children.Add(elips);

Canvas.SetLeft(elips, ptCenter.X);

Canvas.SetTop(elips, ptCenter.Y);

isDrawing = true;

}

private void canvas\_MouseRightButtonDown(object sender, MouseButtonEventArgs e)

{

base.OnMouseRightButtonDown(e);

if (isDrawing)

return;

// 클릭된 엘리먼트를 얻어오고, 앞으로의 이벤트를 준비

ptMouseStart = e.GetPosition(canvas);

elDragging = canvas.InputHitTest(ptMouseStart) as FrameworkElement;

if (elDragging != null)

{

ptElementStart = new Point(Canvas.GetLeft(elDragging),

Canvas.GetTop(elDragging));

isDragging = true;

}

}

private void canvas\_MouseDown(object sender, MouseButtonEventArgs e)

{

base.OnMouseDown(e);

if (e.ChangedButton == MouseButton.Middle)

{

Shape shape = canvas.InputHitTest(e.GetPosition(canvas)) as Shape;

if (shape != null)

shape.Fill = (shape.Fill == Brushes.Red ?

Brushes.Transparent : Brushes.Red);

}

}

private void canvas\_MouseMove(object sender, MouseEventArgs e)

{

base.OnMouseMove(e);

Point ptMouse = e.GetPosition(canvas);

// 타원의 크기 재조정

if (isDrawing)

{

double dRadius = Math.Sqrt(Math.Pow(ptCenter.X - ptMouse.X, 2) + // 마우스를 움직일 때는 마우스 커서 위치를 추적해 경계선을 지정한다.

Math.Pow(ptCenter.Y - ptMouse.Y, 2));

Canvas.SetLeft(elips, ptCenter.X - dRadius);

Canvas.SetTop(elips, ptCenter.Y - dRadius);

elips.Width = 2 \* dRadius;

elips.Height = 2 \* dRadius;

}

// 타원을 이동

else if (isDragging)

{

Canvas.SetLeft(elDragging,

ptElementStart.X + ptMouse.X - ptMouseStart.X);

Canvas.SetTop(elDragging,

ptElementStart.Y + ptMouse.Y - ptMouseStart.Y);

}

}

private void canvas\_MouseUp(object sender, MouseButtonEventArgs e)

{

base.OnMouseUp(e);

// 그리기 동작을 종료

if (isDrawing && e.ChangedButton == MouseButton.Left)

{

elips.Stroke = Brushes.Blue;

elips.StrokeThickness = Math.Min(24, elips.Width / 2);

elips.Fill = Brushes.Red;

isDrawing = false;

ReleaseMouseCapture();

}

// 드래깅을 종료

else if (isDragging && e.ChangedButton == MouseButton.Right)

{

isDragging = false;

}

}

private void canvas\_TextInput(object sender, TextCompositionEventArgs e)

{

base.OnTextInput(e);

//Esc 키를 누르면 그리기나 드래깅을 종료

if (e.Text.IndexOf('\x1B') != -1)

{

if (isDrawing)

ReleaseMouseCapture();

else if (isDragging)

{

Canvas.SetLeft(elDragging, ptElementStart.X);

Canvas.SetTop(elDragging, ptElementStart.Y);

isDragging = false;

}

}

}

private void canvas\_LostMouseCapture(object sender, MouseEventArgs e)

{

base.OnLostMouseCapture(e);

// 그리기의 비정상적 종료 : 자식 타원을 제거

if (isDrawing)

{

canvas.Children.Remove(elips);

isDrawing = false;

}

}

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. canvas\_MouseLeftButtonDown

최초의 클릭을 중심으로 새로운 타원 객체 생성 / 캔버스에 추가 / 마우스캡쳐

1. canvas\_MouseRightButtonDown

클릭된 좌표값 얻기

1. canvas\_MouseDown

클릭된 좌표값 얻기

1. canvas\_MouseMove

타원 크기 재조정 / 타원 위치 이동

1. canvas\_MouseUp

그리기 동작 종료 / 드래깅 종료

1. canvas\_TextInput

ESC 버튼 클릭 시 그리기 / 드래깅 종료

1. canvas\_LostMouseCapture

그리기 비정상적 종료 시 타원 제거

**라우팅(스타일러쉬)**

**라우팅이란** 이벤트가 엘리먼트 트리를 따라 위나 아래로 이동될 때 이를 라우팅된다고 말함.

**라우트된 이벤트의 정의**

-기능측면 : 이벤트를 발생시킨 특정 개체뿐 아니라 요소 트리의

여러 수신기에서 처리기를 호출할수 있는 이벤트 형식.

-구현측면 : RoutedEvent클래스 인스턴스의 지원을 받으며 WPF이벤트

시스템에 의해 처리되는 CLR이벤트임.

엘리먼트 트리에서 이벤트를 전달하는 방법!

\*버블링 : 이벤트를 이벤트가 발생된 엘리먼트부터 최상위 엘리먼트까지

위쪽 방향으로 올려주는 "버블 업"을한다.

\*터널링 : 발생된 이벤트를 이벤트가 발생된 엘리먼트를 포함하는 최상위

엘리먼트에서 이벤트가 발생된 엘리먼트까지 아래 방향으로

내려주는 "터널 다운"을 한다

**##<스타일러쉬 펜 예제>**

사용목적:마우스로만 되던 그리기 기능을 태블릿에다가 스타일러쉬펜(터치펜) 이용해 그리기 기능을 추가한것이다.

C# 코드

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Shapes;

namespace ShadowTheStylus

{

public class ShadowTheStylus : Window

{

// 스타일러스 폴리라인에 대한 몇 가지 상수의 정의

static readonly SolidColorBrush brushStylus = Brushes.Blue;

static readonly SolidColorBrush brushShadow = Brushes.LightBlue;

static readonly double widthStroke = 96 / 2.54; // 1 cm

static readonly Vector vectShadow =

new Vector(widthStroke / 4, widthStroke / 4); // 그림자효과

// 스타일러스 이동에 대한 추가적인 필드

Canvas canv;

Polyline polyStylus, polyShadow;

bool isDrawing;

[STAThread]

public static void Main()

{

Application app = new Application();

app.Run(new ShadowTheStylus());

}

public ShadowTheStylus()

{

Title = "Shadow the Stylus";

// Window 컨텐트로 사용할 Canvas 생성

canv = new Canvas();

Content = canv;

}

protected override void OnStylusDown(StylusDownEventArgs args)

{

base.OnStylusDown(args);

Point ptStylus = args.GetPosition(canv);

// 끝이 둥근 Polyline을 생성해 전경에 사용

polyStylus = new Polyline();

polyStylus.Stroke = brushStylus;

polyStylus.StrokeThickness = widthStroke;

polyStylus.StrokeStartLineCap = PenLineCap.Round;

polyStylus.StrokeEndLineCap = PenLineCap.Round;

polyStylus.StrokeLineJoin = PenLineJoin.Round;

polyStylus.Points = new PointCollection();

polyStylus.Points.Add(ptStylus);

// 그림자용으로 쓸 Polyline

polyShadow = new Polyline();

polyShadow.Stroke = brushShadow;

polyShadow.StrokeThickness = widthStroke;

polyShadow.StrokeStartLineCap = PenLineCap.Round;

polyShadow.StrokeEndLineCap = PenLineCap.Round;

polyShadow.StrokeLineJoin = PenLineJoin.Round;

polyShadow.Points = new PointCollection();

polyShadow.Points.Add(ptStylus + vectShadow);

// 전경의 모든 폴리라인 이전에 그림자 폴리라인을 삽입

canv.Children.Insert(canv.Children.Count / 2, polyShadow);

// 끝에 전경으로 사용할 폴리라인을 추가

canv.Children.Add(polyStylus);

CaptureStylus();

isDrawing = true;

args.Handled = true;

}

protected override void OnStylusMove(StylusEventArgs args)

{

base.OnStylusMove(args);

if (isDrawing)

{

Point ptStylus = args.GetPosition(canv);

polyStylus.Points.Add(ptStylus);

polyShadow.Points.Add(ptStylus + vectShadow);

args.Handled = true;

}

}

protected override void OnStylusUp(StylusEventArgs args)

{

base.OnStylusUp(args);

if (isDrawing)

{

isDrawing = false;

ReleaseStylusCapture();

args.Handled = true;

}

}

protected override void OnTextInput(TextCompositionEventArgs args)

{

base.OnTextInput(args);

// Esc 키를 누르면 그리기를 종료

if (isDrawing && args.Text.IndexOf('\x1B') != -1)

{

ReleaseStylusCapture();

args.Handled = true;

}

}

protected override void OnLostStylusCapture(StylusEventArgs args)

{

base.OnLostStylusCapture(args);

// 비정상적인 종료 : 해당 폴리라인을 제거

if (isDrawing)

{

canv.Children.Remove(polyStylus);

canv.Children.Remove(polyShadow);

isDrawing = false;

}

}

}

}

>>이것을 Xmal 코드로 바꾼다.

#소스 코드

필요한 참조.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

#코드

namespace \_1110

{

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

public partial class MainWindow : Window

{

static readonly SolidColorBrush brushStylus = Brushes.Blue; //브러쉬 색깔(파란색),읽기 전용 속성

static readonly SolidColorBrush brushShadow = Brushes.LightBlue;//브러쉬 쉐도우 색깔(밝은 파랑), 읽기 전용 속성

static readonly double widthStroke = 96 / 2.54; // 1 cm, 읽기 전용 속성

static readonly Vector vectShadow =new Vector(widthStroke / 4, widthStroke / 4); //그림자효과,읽기전용

bool isDrawing; //그렸는지 안그렸는지

Polyline polyStylus, polyShadow; //변수들

public MainWindow()

{

InitializeComponent();

}

//=================================함수구현=================================

private void Canvas\_StylusDown\_1(object sender, StylusDownEventArgs e) //터치펜누른상태

{

base.OnStylusDown(e);

Point ptStylus = e.GetPosition(canv); //캔버스 영역에서 선택한 좌표들을 ptStylus변수에 저장

// 끝이 둥근 Polyline을 생성해 전경에 사용

polyStylus = new Polyline();//PolyLine 생성

polyStylus.Stroke = brushStylus; //윤곽선 지정

polyStylus.StrokeThickness = widthStroke;//윤곽석 넓이

polyStylus.StrokeStartLineCap = PenLineCap.Round;//윤곽선 시작부분에 끝에 모양 설정

polyStylus.StrokeEndLineCap = PenLineCap.Round;//윤곽선 마지막부분에 끝에 모양 설정

polyStylus.StrokeLineJoin = PenLineJoin.Round;//꼭짓점 부분 모양 설정

polyStylus.Points = new PointCollection();//좌표들 생성

polyStylus.Points.Add(ptStylus);//좌표들은 저장한다

// 그림자용으로 쓸 Polyline

polyShadow = new Polyline();

polyShadow.Stroke = brushShadow;

polyShadow.StrokeThickness = widthStroke;

polyShadow.StrokeStartLineCap = PenLineCap.Round;

polyShadow.StrokeEndLineCap = PenLineCap.Round;

polyShadow.StrokeLineJoin = PenLineJoin.Round;

polyShadow.Points = new PointCollection();

polyShadow.Points.Add(ptStylus + vectShadow);

// 전경의 모든 폴리라인 이전에 그림자 폴리라인을 삽입

canv.Children.Insert(canv.Children.Count / 2, polyShadow);

// 끝에 전경으로 사용할 폴리라인을 추가

canv.Children.Add(polyStylus); //캔버스영역에 도형을 그림

CaptureStylus();//스타일러스가 갭쳐가 되면 반환값이 true 아니면 false

isDrawing = true;

e.Handled = true;

}

private void canv\_StylusMove(object sender, StylusEventArgs e) //터치펜이 이동중인 상태

{

base.OnStylusMove(e);//이벤트가 발생

if (isDrawing)

{

Point ptStylus = e.GetPosition(canv);//이동중인 좌표값을 ptStylus에다가 저장

polyStylus.Points.Add(ptStylus);//저장

polyShadow.Points.Add(ptStylus + vectShadow);//좌표값과 그림자좌표를 더함

e.Handled = true;//처리상태를 truefh 설정

}

}

private void canv\_StylusUp(object sender, StylusEventArgs e)//터치펜을 떴을때

{

base.OnStylusUp(e);//이벤트 발생

if (isDrawing)

{

isDrawing = false; //false 설정

ReleaseStylusCapture();//스타일러스 장치가 갭쳐가된경우 해체한다

e.Handled = true;//처리상태를 truefh 설정

}

}

private void canv\_TextInput(object sender, TextCompositionEventArgs e) //Esc누르면 그리기를 종료

{

base.OnTextInput(e);

// Esc 키를 누르면 그리기를 종료

if (isDrawing && e.Text.IndexOf('\x1B') != -1)//Esc가 눌러진다면...

{

ReleaseStylusCapture();//스타일러스 장치가 갭쳐가된경우 해체한다

e.Handled = true;//처리상태를 truefh 설정

}

}

private void canv\_LostStylusCapture(object sender, StylusEventArgs e)

{

base.OnLostStylusCapture(e);

// 비정상적인 종료 : 해당 폴리라인을 제거

if (isDrawing)

{

canv.Children.Remove(polyStylus);//전경 제거

canv.Children.Remove(polyShadow);//그림자 제거

isDrawing = false;//false로 설정

}

}

}

}

#Xmal 코드

<Window x:Class="\_1110.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

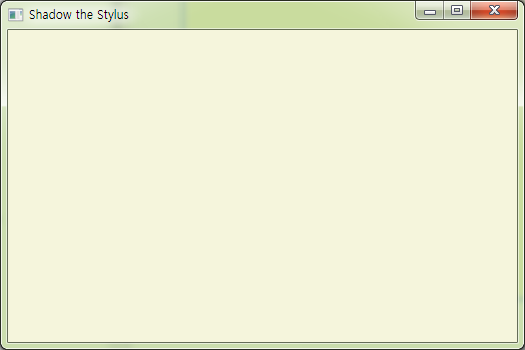
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="Shadow the Stylus" Height="350" Width="525">

<Canvas Name="canv" Background="Beige" StylusDown="Canvas\_StylusDown\_1" StylusMove="canv\_StylusMove" StylusUp="canv\_StylusUp" TextInput="canv\_TextInput" LostStylusCapture="canv\_LostStylusCapture">

</Canvas>

</Window>

**>>아래 와 같이 나옴. **