WPF Control Template

lifeisforu@naver.com
최도경

- 이 문서는 다음의 내용을 요약하고 보충한 것입니다.
  - Styling and Templating
  - Customizing the Appearance of an Existing Control by Creating a ControlTemplate.
  - WPF Graphics Rendering Overview.

Role of the Visual Object

- Visual 클래스는 FrmeworkElement 오브젝트가 상속해야 하는 기본 추상 클래스이다.
- Visual 오브젝트는 core WPF 오브젝트로, 다음과 같은 일을 수행한다.
  - Output display : 화면에 출력한다.
  - Transformations : 변환을 수행한다.
  - Clipping : 보이지 않게 잘라내는 영역을 제공한다.
  - Hit testing : 좌표나 기하도형이 visual 의 영역에 포함되는지 확인한다.
  - Bounding gox calculations : visual 의 bounding 영역을 결정한다.

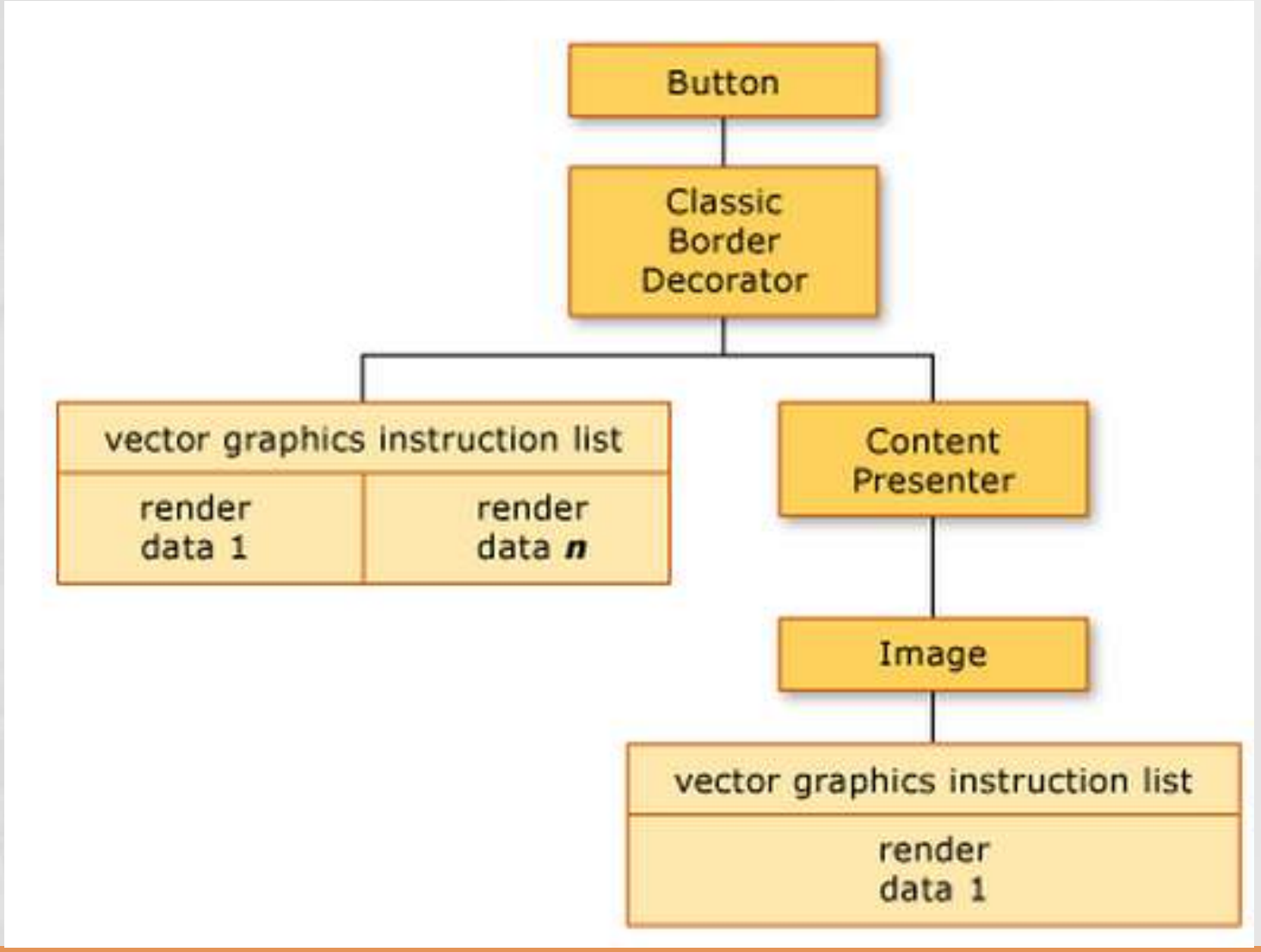
ControlTemplate

- Control 의 계층 구조의 핵심은 ControlTemplate 이다.
- Control template 은 visual 계층구조를 지정하는 역할을 수행한다.

```
<Button Click="OnClick">
  <Image Source="images\greenlight.jpg"></Image>
</Button>
```

- 위 code 의 계층구조는 어떻게 되어 있을까?

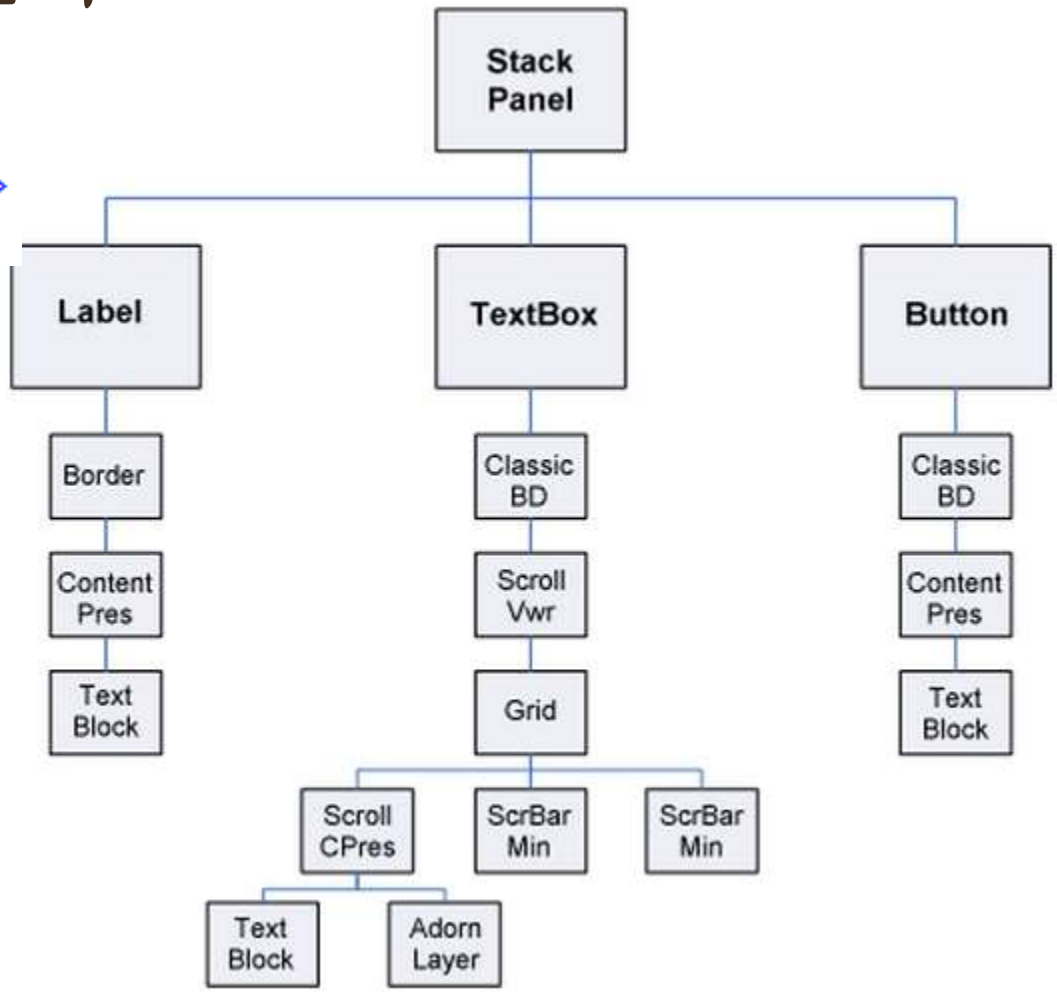
ControlTemplate(cont)



Visual Tree

- Control template 이 표현하는 계층구조를 visual tree 라고 한다.

```
<StackPanel>
  <Label>User name:</Label>
  <TextBox />
  <Button Click="OnClick">OK</Button>
</StackPanel>
```



Root Visual

- Visual tree 에서 최상위에 있는 요소를 root visual 이라고 한다.
- 일반적으로는 Window 나 NavigationWindow 이다.

Viewing the Visual Tree

- XAML 편집기에서 Show Visual Tree 를 클릭하면 visual tree 를 볼 수 있다.

```
<StackPanel>
  <Label>User name:</Label>
  <TextBox />
  <Button>OK</Button>
</StackPanel>
```

Visual Tree Explorer

- :StackPanel
  - :Label
    - :Border
      - :ContentPresenter
        - :TextBlock
  - :TextBox
    - Bd:ListBoxChrome
      - PART_ContentHost:ScrollViewer
        - :Grid
          - :Rectangle
          - :ScrollContentPresenter
            - :TextBoxView
              - :DrawingVisual
            - :AdornerLayer
          - :ScrollBar
          - :ScrollBar
  - :Button
    - Chrome:ButtonChrome
      - :ContentPresenter
        - :TextBlock

Visual Tree Helper

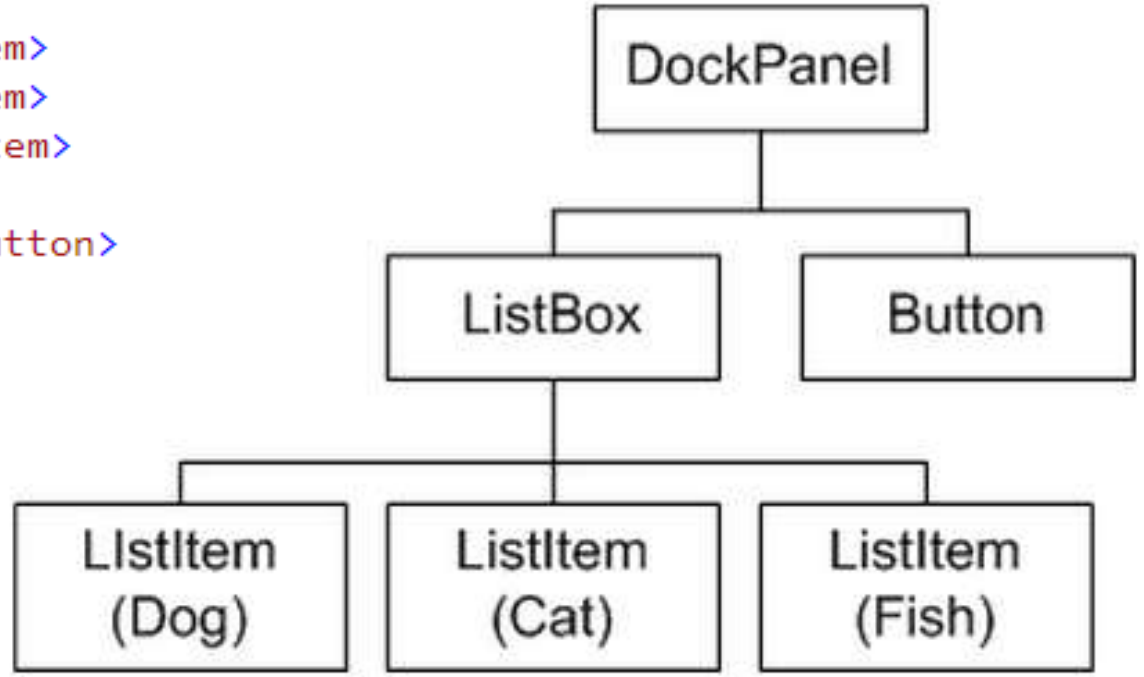
- VisualTreeHelper 는 visual tree 에 접근하는 방법을 제공한다.

```
// Enumerate all the descendants of the visual object.
static public void EnumVisual(Visual myVisual)
{
    for (int i = 0; i < VisualTreeHelper.GetChildrenCount(myVisual); i++)
    {
        // Retrieve child visual at specified index value.
        Visual childVisual = (Visual)VisualTreeHelper.GetChild(myVisual, i);

        // Do processing of the child visual object.

        // Enumerate children of the child visual object.
        EnumVisual(childVisual);
    }
}
```

Logical Tree

- Logical tree 는 run time 에 응용프로그램의 element 들을 표현한다.

```
<DockPanel>
  <ListBox>
    <ListBoxItem>Dog</ListBoxItem>
    <ListBoxItem>Cat</ListBoxItem>
    <ListBoxItem>Fish</ListBoxItem>
  </ListBox>
  <Button Click="OnClick">OK</Button>
</DockPanel>
```



LogicalTree(cont)

- Logical tree 는 property 계층구조와 event routing 을 을 이해하는데 유용하다.
- Visual tree 와는 다르게 logical tree 는 non-visual object 들을 포함한다(ex : ListItem).
- Logical tree 는 ContentPresenter 를 확장하지 않는다. 이는 logical tree 와 visual tree 가 1:1 관계가 아님을 의미한다.
- LogicalTreehelper 는 logical tree 에 접근하는 방법을 제공한다.

ContentPresenter & ItemsPresenter

- 기본 content 나 items 를 출력하고자 한다면, ContentPresenter 와 ItemsPresenter 를 사용한다.
- 자세한 내용은 "ContentPresenter 란? 그저 간단한 컨트롤껍데기?"를 참조하라.

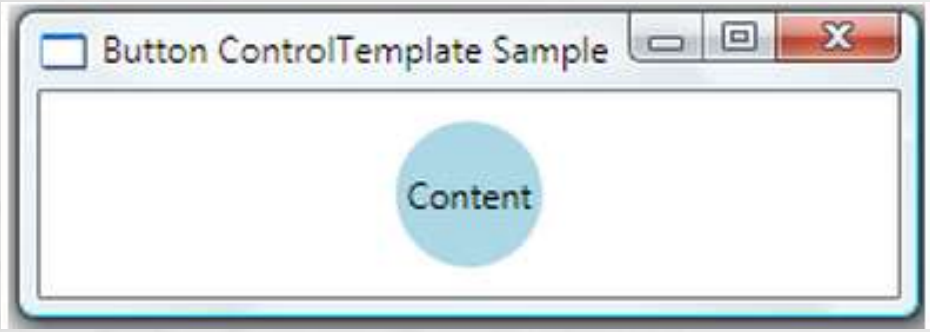
Defining Control Template

- ControlTemplate 은 control 의 visual tree 를 지정한다. Control 의 Template property 에 ControlTemplate 개체를 할당해 control 의 외형을 지정한다.

```
<Style TargetType="Button">
  <!--Set to true to not get any properties from the themes.-->
  <Setter Property="OverridesDefaultStyle" Value="True"/>
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="Button">
        <Grid>
          <Ellipse Fill="{TemplateBinding Background}"/>
          <ContentPresenter HorizontalAlignment="Center"
                            VerticalAlignment="Center"/>
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

Defining Control Template(cont)

- 앞의 xaml 은 다음과 같은 button 을 생성한다.



- 다음 예제를 받아서 분석해 보자.
  - ControlTemplateExample.zip

TemplatePartAttribute

- Control template 에 named part 를 식별할 수 잇는 정의를 추가한다.

```
[TemplatePartAttribute(Name = "ContentPresenter", Type = typeof(ContentPresenter))]
[TemplatePartAttribute(Name = "Popup", Type = typeof(Popup))]
public class ComboBox : ItemsControl
{
}
```

- 일반적으로는 PART_ 라는 prefix 를 붙인다(예 : PART_ContentPresenter, PART_Popup).

TemplatePartAttribute(cont)

```
<ControlTemplate TargetType="ComboBox">
  <Grid>
    <Border x:Name="ContentPresenterBorder">
      <Grid>
        <ToggleButton x:Name="DropDownToggle"
                      HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
                      Margin="-1" HorizontalContentAlignment="Right">
          <Path x:Name="BtnArrow" Height="4" Width="8"
                Stretch="Uniform" Margin="0,0,6,0"  Fill="Black"
                Data="F1 M 300,-190L 310,-190L 305,-183L 301,-190 Z " />
        </ToggleButton>
        <ContentPresenter x:Name="ContentPresenter" Margin="6,2,25,2">
          <TextBlock Text=" " />
        </ContentPresenter>
      </Grid>
    </Border>
    <Popup x:Name="Popup">
      <Border x:Name="PopupBorder"
              HorizontalAlignment="Stretch" Height="Auto"
              BorderThickness="{TemplateBinding BorderThickness}"
              BorderBrush="Black" Background="White" CornerRadius="3">
        <ScrollViewer x:Name="ScrollViewer" BorderThickness="0" Padding="1">
          <ItemsPresenter/>
        </ScrollViewer>
      </Border>
    </Popup>

  </Grid>
</ControlTemplate>
```

TemplateBindingExtension

- TemplateBinding 은 RelativeSource property 에다가 RelativeSource.TemplatedParent 를 지정한 것과 동일하다.

```
<Style TargetType="ListBox">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="ListBox">
        <Border CornerRadius="5" Background="{TemplateBinding ListBox.Background}">
          <ScrollViewer HorizontalScrollBarVisibility="Auto">
            <StackPanel Orientation="Horizontal"
                        VerticalAlignment="Center"
                        HorizontalAlignment="Center"
                        IsItemsHost="True"/>
          </ScrollViewer>
        </Border>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

Finding Element

- Template 에서 생성한 element 를 찾는 방법은?

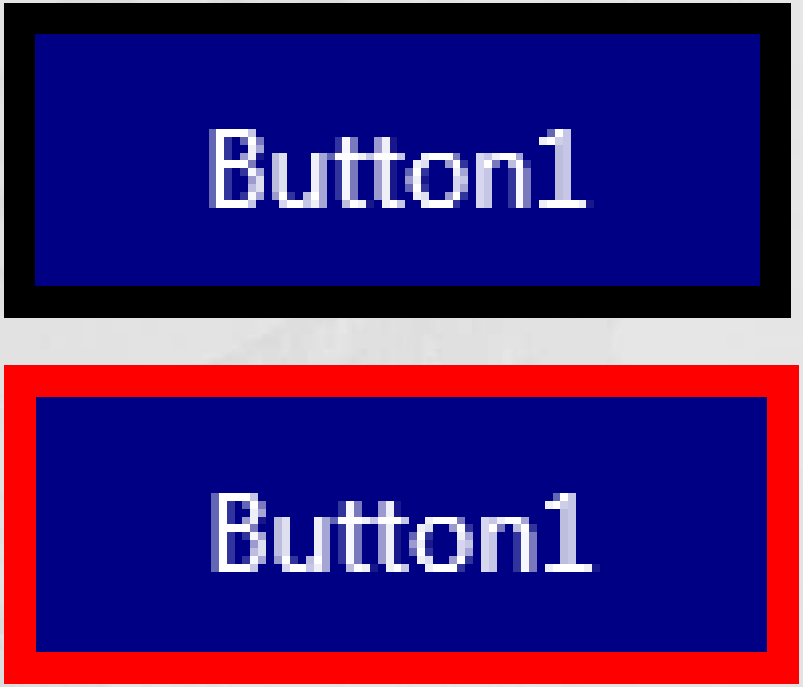
```
<Style TargetType="{x:Type Button}">
  <Setter Property="Template">
    <Setter.Value>
      <ControlTemplate TargetType="{x:Type Button}">
        <Grid Margin="5" Name="grid">
          <Ellipse Stroke="DarkBlue" StrokeThickness="2">
            <Ellipse.Fill>
              <RadialGradientBrush Center="0.3,0.2" RadiusX="0.5" RadiusY="0.5">
                <GradientStop Color="Azure" Offset="0.1" />
                <GradientStop Color="CornflowerBlue" Offset="1.1" />
              </RadialGradientBrush>
            </Ellipse.Fill>
          </Ellipse>
          <ContentPresenter Name="content" Margin="10"
                            HorizontalAlignment="Center" VerticalAlignment="Center"/>
        </Grid>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
</Style>
```

Finding Element(cont)

```
// Finding the grid that is generated by the ControlTemplate of the Button
Grid gridInTemplate = (Grid)myButton1.Template.FindName("grid", myButton1);

// Do something to the ControlTemplate-generated grid
MessageBox.Show("The actual width of the grid in the ControlTemplate: "
    + gridInTemplate.GetValue(Grid.ActualWidthProperty).ToString());
```

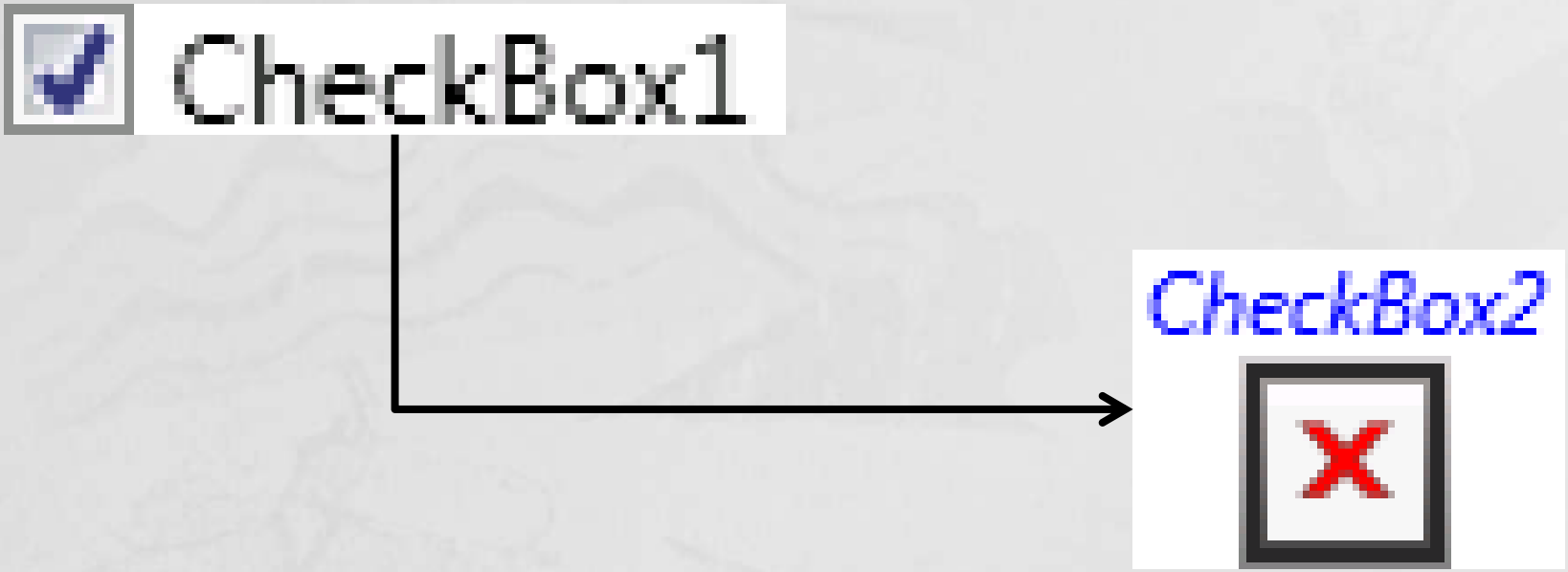
Exercise 1

- 다음과 같은 button 을 만들어 보자. 마우스를 올리면 빨간 테두리가 나온다.



Exercise 2

- CheckBox 를 다음과 같이 바꿔보자.



- 귀찮의 그림을 넣지 말고 그냥 텍스트 X 와 V 를 사용하라.