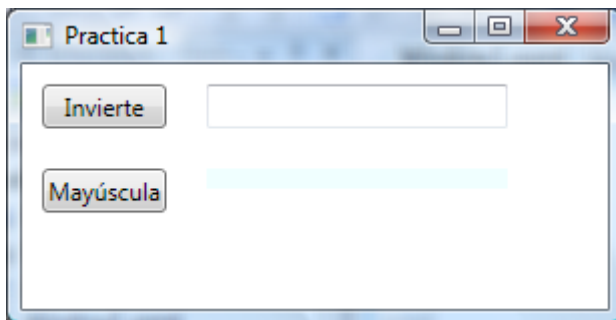


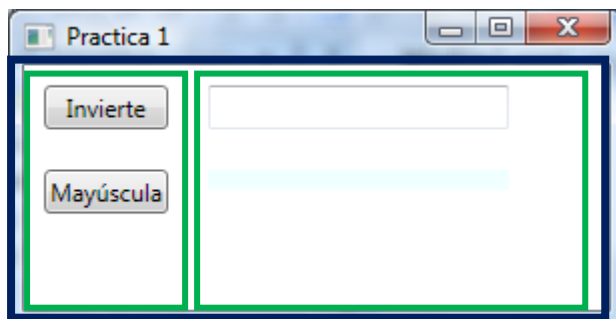
## Sencilla aplicación interactiva

### Diseño de la interfaz

Creemos una aplicación con dos botones (clase `Button`), una caja de texto (`TextBox`) y una etiqueta (`Label`) sin contenido (con algún color de fondo para poder distinguirla). Su apariencia debe ser algo como esto:



Para ello vamos a usar un `StackPanel` (en vez del `Grid` que se crea por defecto) de orientación horizontal que, a su vez contendrá dos `StackPanel` de orientación vertical, el izquierdo para situar los botones y el derecho para situar la caja de texto y la etiqueta)



```
<StackPanel Orientation="Horizontal">
  <StackPanel>
  </StackPanel>
  <StackPanel >
  </StackPanel>
</StackPanel>
```

Situar los controles dándoles un nombre a cada uno. Por ejemplo:

```
<Button Name="BotonInvierte" Content="Invierte" />
```

Ajustar la propiedad de `Margin` para que quede una apariencia correcta.

Compilar y ejecutar.

## Gestionar los botones

Escribimos los gestores que se ocuparán de los eventos Click de los botones: en el editor de diseño del archivo XAML de la ventana, hacemos doble click sobre el botón invierte. Automáticamente, se crea un método en la clase de la ventana para gestionar el evento.

```
<Button Name="BotonInvierte" ... Click="BotonInvierte_Click" />

private void BotonInvierte_Click(object sender, RoutedEventArgs e)
{

}
```

Queremos que cuando se pulse el botón con el texto “Invierte”, la etiqueta muestre el texto que se haya introducido en la caja de texto invertido. Para ello, vamos a aprovechar que las instancias de los objetos que están contenidos en la ventana, son datos miembro de la clase derivada de `Window` y que el nombre que se le da en el archivo XAML a cada elemento funciona como referencia de la instancia. Así, un código similar al siguiente haría que el comportamiento del botón fuese el deseado:

```
TextBox tb = CajaTexto;
Label etiq = Etiqueta;
int i;
string aux="";
for (i = 0; i < tb.Text.Length; i++)
    aux += tb.Text[tb.Text.Length-i-1];
etiq.Content = aux;
```

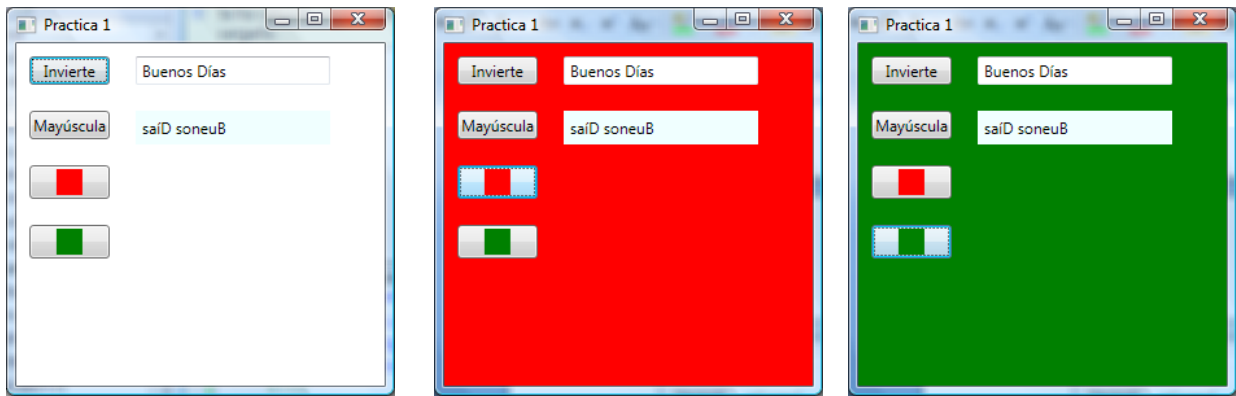
Consúltase la documentación de referencia de las clases `string`, `TextBox` y `Label` para comprender en detalle el código anterior.

De modo similar, creamos el gestor para que un click del botón “Mayúscula” provoque que se muestre en la etiqueta el texto de la caja pasado a mayúsculas. Revisar los métodos de la clase `string` para escribir el código adecuado.

## Configurar el fondo de la ventana

Queremos que el fondo de la ventana se pueda elegir también mediante botones.

En este caso, los botones no tendrán como contenido un texto, sino un rectángulo del color correspondiente.



Para diseñar los botones hay que hacer que el contenido sea un objeto de la clase `Rectangle`, con los valores apropiados en las propiedades `Fill`, `Height` y `Width`. Por ejemplo:

```
<Button Name="Rojo" Margin="10">
    <Rectangle Width="20" Height="20" Fill="Red"/>
</Button>
```

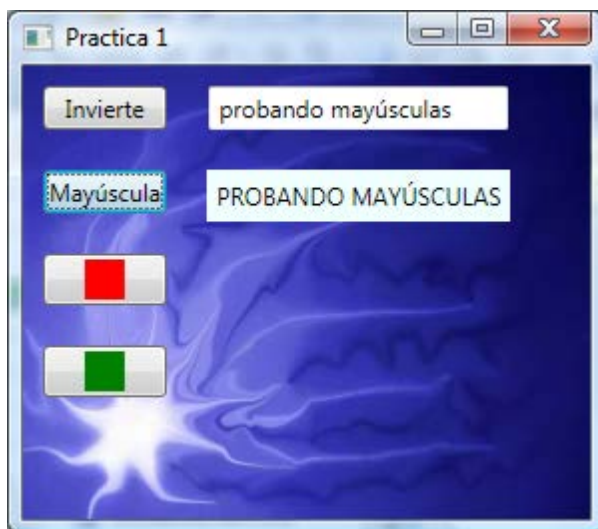
Crear los gestores para los Click de los nuevos botones. Para ello solo hay que tener en cuenta que la propiedad `Background` de la ventana es la que se utiliza para determinar cómo se va a pintar el fondo.

Revisar la documentación de referencia sobre las clases `Window` (propiedad `Background`), `Brush` y `Brushes`.

Probar el funcionamiento de la aplicación.

### Imagen de fondo inicial

Ahora vamos a hacer que, inicialmente, en vez de mostrarse un fondo blanco, aparezca una imagen.



Primero tenemos que añadir el archivo de imagen al proyecto: apuntar en el visor de soluciones al proyecto, pulsar el botón derecho del ratón y elegir Agregar / "Elemento existente"; seleccionar el archivo de imagen que se desee.

Ahora solo hay que dar el valor apropiado a la propiedad `Background`. En este caso, como el valor a asignar es un objeto de tipo `ImageBrush` que a su vez toma la imagen a través de su propiedad `ImageSource`, no podemos usar simplemente la sintaxis de asignar una simple cadena de caracteres. Tenemos que recurrir a usar un elemento de propiedad:

```
<Window.Background>
  <ImageBrush
    ImageSource="nombrearchivo.jpg" />
</Window.Background>
```

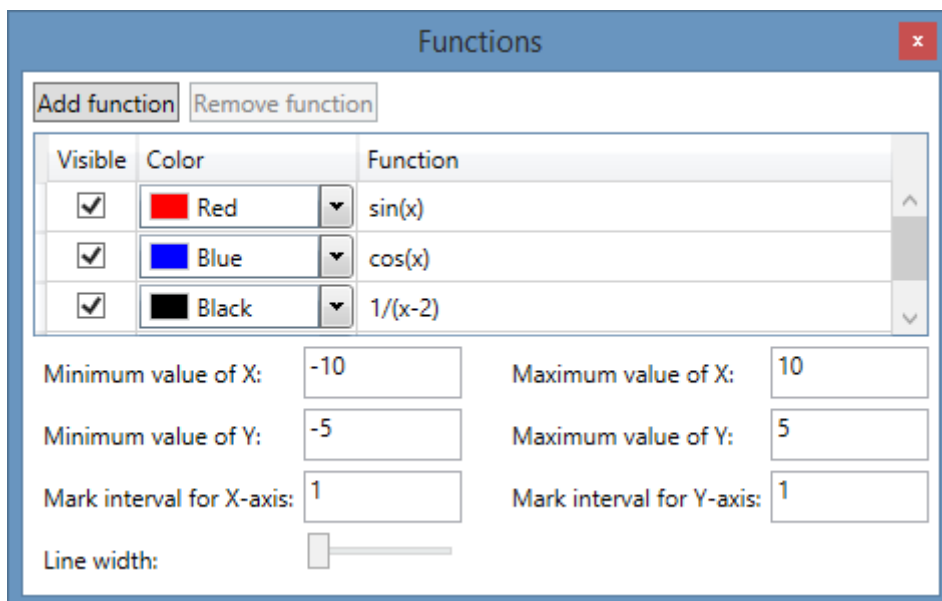
## Variaciones

Crear una aplicación similar a la anterior que permita hacer una conversión de grados Celsius a Fahrenheit y viceversa.

## Uso de paneles para distribución de elementos

Crear los siguientes diseños de ventanas usando la combinación de paneles que se estime más oportuna.

Comprobar que el diseño tiene una buena adaptación a las variaciones de tamaño de la ventana o de resolución del dispositivo.



Remove


X: 167.4Y: 7.8


Vx: 34.5Vy: 14.8

$\theta$ : 266.7

$\omega$ : -36.9

w 33.2h: 23.8

Relleno:  ▼

Borde:  ▼ 2

Forma: Ellipse ▼ 3 ▼

Add

