实验四 Windows桌面编程

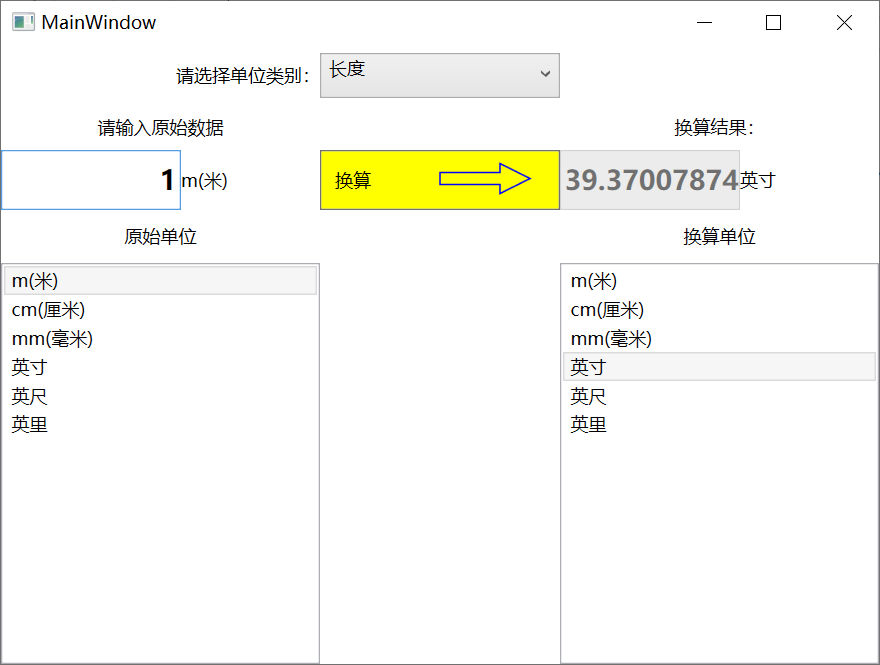
一、实验目的

1. 熟悉使用WPF进行界面编程的基本过程；
2. 掌握WPF布局、控件、事件的使用。

二、实验内容

（可将整个工程打包后上传）

1. 使用WPF技术，进行合理布局，设计一个窗体应用程序，完成一些常用度量单位的转换，基本样式如下图：



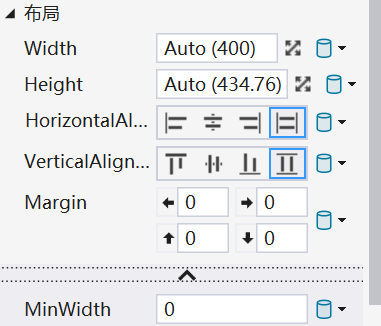
主要设计过程包括：

（1）新建WPF工程。

（2）设置控件布局。保留默认的Grid布局控件，并设置其行数和列数。操作如下：

a. 将光标停在XAML的Grid标签上，或者打开左侧“文档大纲”窗口并选择Grid，再切换到属性窗口，可以查看并编辑该Grid的各种属性。

b. 展开“布局”组



c. 点击右侧“…”按钮，在弹出的窗口中，添加三个ColumnDefinition，即添加三个列。为每个列选择尺寸分配规则。Auto自动，Pixel固定像素，Star按星数平分。

d. 通过添加行。

e. 第2行第0列包含一个文本框TextBox和一个文本块TextBlock，应将其用StackPanel组织起来：将StackPanel设置Grid.Column="0" Grid.Row="2"，堆叠方向为Horizontal。在该StackPanel下层放置文本框和文本块。

（3）选择各类控件，为它们安排好位置，设置它们的各类属性，如颜色、大小、靠左还是靠右等。一般是首先拖放控件，然后在属性窗口中编辑，默认设置的Margin大小可调整。

a. 中间的“换算”Button用下面的方式面出来：

<Button Grid.Column="1" Grid.Row="2" Width="160" Height="40">

<StackPanel Orientation="Horizontal">

<TextBlock Width="40" VerticalAlignment="Center">换算</TextBlock>

<Canvas Width="100" Height="30">

<Path Stroke="Blue" Data="M 30, 10 l 40,0 l0,-6 l 20,10 l-20,10 l0,-6 l-40,0 Z">

</Path>

</Canvas>

</StackPanel>

</Button>

如果箭头位置不理想，可调整Data中M后的两个坐标。

b. 要让后台cs代码中访问列表控件，需要给它命名，如

<ListBox x:Name="lstSource"></ListBox>

c. 在App.xaml文件中添加资源，如：

<SolidColorBrush x:Key="blueBrush" Color="Blue"></SolidColorBrush>

该画刷可用于任一带颜色属性的控件。

<Style TargetType="TextBlock" x:Key="redText">

<Setter Property="Foreground" Value="Red"/>

</Style>

该样式专用于TextBlock，还可添加其它Property设置，在TextBlock的属性中添加以下属性进行应用。

Style="{StaticResource redText}

要求至少生成自己的一个画刷和一个样式并应用于控件。

（4）在原始数据和换算结果后的两个TextBlock，分别绑定到原始单位列表和换算单位列表的当前值

<TextBlock Text="{Binding SelectedValue, ElementName=lstSource}" VerticalAlignment="Center"/>

（5）添加事件处理方法。通过事件标签页或双击控件添加。

a. 响应ComboBox的SelectionChanged事件，获取到选定内容的方法是：

ComboBox cb = sender as ComboBox;

ComboBoxItem item = cb.SelectedItem as ComboBoxItem;

string selected = item.Content.ToString();

然后通过switch语句分别处理长度、重量等选项。

b. 响应换算按钮的Click事件，首先获取文本框的内容（textSource.Text textSource.Text），转换为数值；然后获取列表框的当前选项（lstSource.SelectedValue.ToString()），根据下拉列表框的当前选项（comboTypes.SelectedItem）选择转换功能。

b. 响应TextBox的TextChanged事件。当在TextBox输入内容时，换算结果内容清空。

c. 在列表中选择了新的选项时，换算结果内容清空。

另外：

1. 如果原始数据格式有误，要提示用户。
2. 可在颜色、布局、功能等方面按自己的想法进行完善。
3. 对于长度等单位，可构建如下字典对象辅助转换：

Dictionary<string, double> length = new Dictionary<string, double>

{ {"m(米)", 1 }, {"cm(厘米)", 0.01 }, {"mm(毫米)", 0.001 }};

但温度不行，需自行转换。

1. 现有一小游戏叫2048，是采用Windows Form技术开发，已提供代码，现将其改造为采用WPF技术进行开发，用嵌套的Grid来组织各类控件。

游戏的基本方法是：用上下左右4个方向键，将屏幕上的数字块进行“移动”，如果移动后相同的则进行数字的合并，每次移动后又产生一个新的数（2或4）。要求用户得到数字2048就赢了。

另外，还有从以下方向改进（可选）：

* 示例游戏中方块较少，可由用户选择N的数值，据此创建N\*N的控件；

可以命名一个Grid：

<Grid x:Name="gamePad">

为其设置行与列：

for (int i = 0; i < N; i++)

{

gamePad.ColumnDefinitions.Add(new ColumnDefinition());

gamePad.RowDefinitions.Add(new RowDefinition());

}

在代码文件里，就可以对它进行手动添加里面各个Button。

Border border = new Border();//加个边框

border.BorderThickness = new Thickness(2);

TextBlock textBlock = new TextBlock();//框内加文本块

textBlock.HorizontalAlignment = HorizontalAlignment.Center;

textBlock.VerticalAlignment = VerticalAlignment.Center;

textBlock.FontSize = 16;

textBlock.Margin = new Thickness(3);

border.Child = textBlock;

Grid.SetRow(border, i);

Grid.SetColumn(border, j);

gamePad.Children.Add(border);

* 示例游戏中方向键的规则比较乱，可以用自己的规则替代；
* 可以在界面上展示当前最高数值；
* 可以给程序加上选择模式的功能，2🡪4🡪8…，夏🡪商🡪周…，班🡪排🡪连…
* 可以添加上下左右按钮

（按游戏改善程度给分。如果直接将WinForm程序塞入WPF则给0分）

三、实验心得与体会

这次实验让我学习到了WPF技术的灵活性和强大功能。通过使用XAML进行界面布局设计，我发现可以更直观地描述界面的结构和外观，并且可以轻松实现各种大小和复杂度的布局。嵌套的Grid让我可以更好地组织和管理界面元素，提高了界面的可维护性和可扩展性。

在实现度量单位转换应用程序时，我将TextBox和Button等控件与代码逻辑结合，实现了简洁而高效的交互效果。而在改造2048小游戏时，我使用了嵌套的Grid来组织游戏面板和数字块，这使得界面布局更加灵活，并且为游戏操作提供了良好的交互体验。

总的来说，通过这次实验，我更加熟练地掌握了WPF技术的使用方法，对于界面设计和开发有了更深入的理解。我相信在未来的项目中，我能够更快速地实现各种复杂的界面需求，同时为用户带来更加优秀的交互体验。 WPF 技术的学习和应用对我的工作和职业发展都具有重要意义。