

[](https://www.tutorialr.com/tutorials/)

Windows App SDK

Ruler Control





# Ruler Control

**Ruler Control** shows how to create an on-screen **Ruler** for **Metric** or **Imperial** using **Windows App SDK**

## Step 1

Follow **Setup and Start** on how to get **Setup** and **Install** what you need for **Visual Studio 2022** and **Windows App SDK**.

|  |  |
| --- | --- |
| In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it. | Text  Description automatically generated |
| Once **Visual Studio 2022** has started select **Create a new project**. | **Graphical user interface, text  Description automatically generated** |
| Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**. | **Graphical user interface, text  Description automatically generated** |
| After that in **Configure your new project** type in the **Project name** as *RulerControl*, then select a Location and then select **Create** to start a new **Solution**. | **Graphical user interface, text, application, email  Description automatically generated** |

## Step 2

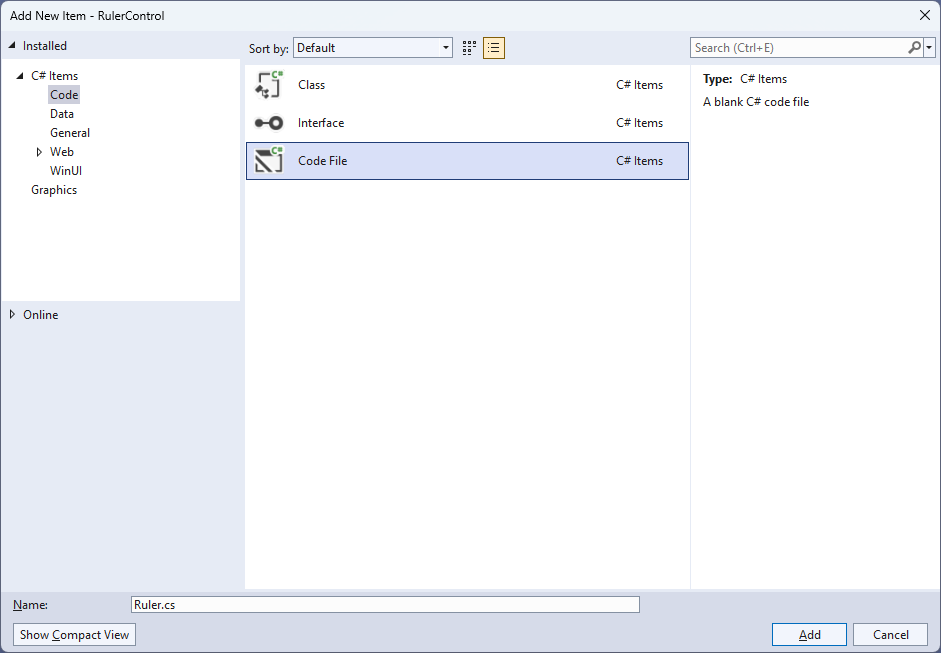
Then in **Visual Studio** within **Solution** **Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item…**

Table

Description automatically generated with low confidence

## Step 3

Then in **Add New Item** from the **C# Items** list, select **Code** and then select **Code File** from the list next to this, then type in the name of *Ruler.cs* and then **Click** on **Add**.



## Step 4

|  |  |
| --- | --- |
| Then from **Solution** **Explorer** for the **Solution** double-click on **Ruler.cs** to see the **Code** for the **User Control**. |  |

## Step 5

You will now be in the **View** for the **Code** of *Ruler.cs*, within this type in the following **Code**:

using Microsoft.UI;

using Microsoft.UI.Xaml;

using Microsoft.UI.Xaml.Controls;

using Microsoft.UI.Xaml.Media;

using Microsoft.UI.Xaml.Shapes;

using Windows.Foundation;

namespace RulerControl;

public enum Units

{

Cm,

Inch

};

public class Ruler : Canvas

{

private const double default\_height = 40.0;

private double \_originalHeight;

// Static Methods

// Dependency Properties & Properties

// Layout Method

// Constructor & Measure Override Event Handler

}

There are **using** statements for the **User Control**, a **namespace** for **RulerControl** with an **enum** for the **Units** of the **Ruler** along with a **class** of **Ruler** that will represent the **User Control**.

## Step 6

Then in the **namespace** of **RulerControl** in the **class** of **Ruler** after the **Comment** of **// Static Methods** type the following **Static Methods**:

private static double CmToDip(double cm) =>

cm \* 96.0 / 2.54;

private static double InchToDip(double inch) =>

inch \* 96.0;

private static Path GetLine(Brush stroke, double thickness,

Point start, Point finish) => new()

{

Stroke = stroke,

StrokeThickness = thickness,

Data = new LineGeometry()

{

StartPoint = start,

EndPoint = finish

}

};

These **Static Methods** include **CmToDip** and **InchToDip** which will perform the conversions for the measurements to be displayed on the **Ruler** to device independent pixels. There is also **GetLine** which will return the **Path** used to display the lines on the **Ruler**.

## Step 7

While still in the **namespace** of **RulerControl** in the **class** of **Ruler** after the **Comment** of **// Dependency Properties & Properties** type the following **Dependency Properties** and **Properties**:

public static readonly DependencyProperty ForegroundProperty =

DependencyProperty.Register(nameof(Foreground), typeof(Brush),

typeof(Ruler), new PropertyMetadata(new SolidColorBrush(Colors.Black)));

public static readonly DependencyProperty LengthProperty =

DependencyProperty.Register(nameof(Length), typeof(double),

typeof(Ruler), new PropertyMetadata(10.0));

public static readonly DependencyProperty SegmentProperty =

DependencyProperty.Register(nameof(Segment), typeof(double),

typeof(Ruler), new PropertyMetadata(20.0));

public static readonly DependencyProperty UnitProperty =

DependencyProperty.Register(nameof(Unit), typeof(double),

typeof(Ruler), new PropertyMetadata(Units.Cm));

public Brush Foreground

{

get { return (Brush)GetValue(ForegroundProperty); }

set { SetValue(ForegroundProperty, value); }

}

public double Length

{

get { return (double)GetValue(LengthProperty); }

set { SetValue(LengthProperty, value); }

}

public double Segment

{

get { return (double)GetValue(SegmentProperty); }

set { SetValue(SegmentProperty, value); }

}

public Units Unit

{

get { return (Units)GetValue(UnitProperty); }

set { SetValue(UnitProperty, value); }

}

The **Dependency Properties** or **Properties** for the **User Control** can be customised for the **Ruler**.

## Step 8

While still in the **namespace** of **RulerControl** in the **class** of **Ruler** after the **Comment** of **// Layout Method** type the following **Method**:

private void Layout()

{

Children.Clear();

for (double value = 0.0; value <= Length; value++)

{

double dip;

if (Unit == Units.Cm)

{

dip = CmToDip(value);

if (value < Length)

{

for (int i = 1; i <= 9; i++)

{

if (i != 5)

{

var mm = CmToDip(value + 0.1 \* i);

Children.Add(GetLine(Foreground, 0.5, new Point(mm, Height),

new Point(mm, Height - Segment / 3.0)));

}

}

var middle = CmToDip(value + 0.5);

Children.Add(GetLine(Foreground, 1.0, new Point(middle, Height),

new Point(middle, Height - Segment \* 2.0 / 3.0)));

}

}

else

{

dip = InchToDip(value);

if (value < Length)

{

var quarter = InchToDip(value + 0.25);

Children.Add(GetLine(Foreground, 0.5, new Point(quarter, Height),

new Point(quarter, Height - Segment / 3.0)));

var middle = InchToDip(value + 0.5);

Children.Add(GetLine(Foreground, 1.0, new Point(middle, Height),

new Point(middle, Height - 0.5 \* Segment \* 2.0 / 3.0)));

var division = InchToDip(value + 0.75);

Children.Add(GetLine(Foreground, 0.5, new Point(division, Height),

new Point(division, Height - 0.25 \* Segment / 3.0)));

}

}

Children.Add(GetLine(Foreground, 1.0, new Point(dip, Height),

new Point(dip, Height - Segment)));

}

}

This **Method** creates the look-and-feel for the **User Control** for either *Metric* or *Imperial* units to be displayed accordingly for the **Ruler**.

## Step 9

While still in the **namespace** of **RulerControl** in the **class** of **Ruler** after the **Comment** of **// Constructor & Measure Override Event Handler** type the following **Constructor** and **Event Handler**:

public Ruler() =>

Loaded += (object sender, RoutedEventArgs e) => Layout();

protected override Size MeasureOverride(Size availableSize)

{

Height = !double.IsNaN(Height) ? Height : default\_height;

var desiredSize = (Unit == Units.Cm) ?

new Size(CmToDip(Length), Height) :

new Size(InchToDip(Length), Height);

if(Height != \_originalHeight)

{

Layout();

\_originalHeight = Height;

}

return desiredSize;

}

The **Constructor** will call the **Method** of **Layout** when the **User Control** has been **Loaded** and the **Event Handler** of **MeasureOverride** will manage the resizing of the **Ruler**.

## Step 10

|  |  |
| --- | --- |
| Within **Solution** **Explorer** for the **Solution** double-click on **MainWindow.xaml** to see the **XAML** for the **Main Window**. |  |

## Step 11

In the **XAML** for **MainWindow.xaml** there be some **XAML** for a **StackPanel**, this should be **Removed** by removing the following:

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Center" VerticalAlignment="Center">

<Button x:Name="myButton" Click="myButton\_Click">Click Me</Button>

</StackPanel>

## Step 12

While still in the **XAML** for **MainWindow.xaml** above **</Window>**, type in the following **XAML**:

<Viewbox Margin="50">

<local:Ruler Length="15.0" Unit="Cm"

Background="{ThemeResource SystemControlHighlightAccentBrush}"

Foreground="{ThemeResource SystemControlBackgroundAltHighBrush}"/>

</Viewbox>

This **XAML** contains a **ViewBox** including the **User Control** of **Ruler** with the **Length** and **Unit** set.

## Step 13

|  |  |
| --- | --- |
| Then, within **Solution** **Explorer** for the **Solution** select the arrow next to **MainWindow.xaml** then double-click on **MainWindow.xaml.cs** to see the **Code** for the **Main Window**. |  |

## Step 14

In the **Code** for **MainWindow.xaml.cs** there be a **Method** of **myButton\_Click(...)** this should be **Removed** by removing the following:

private void myButton\_Click(object sender, RoutedEventArgs e)

{

myButton.Content = "Clicked";

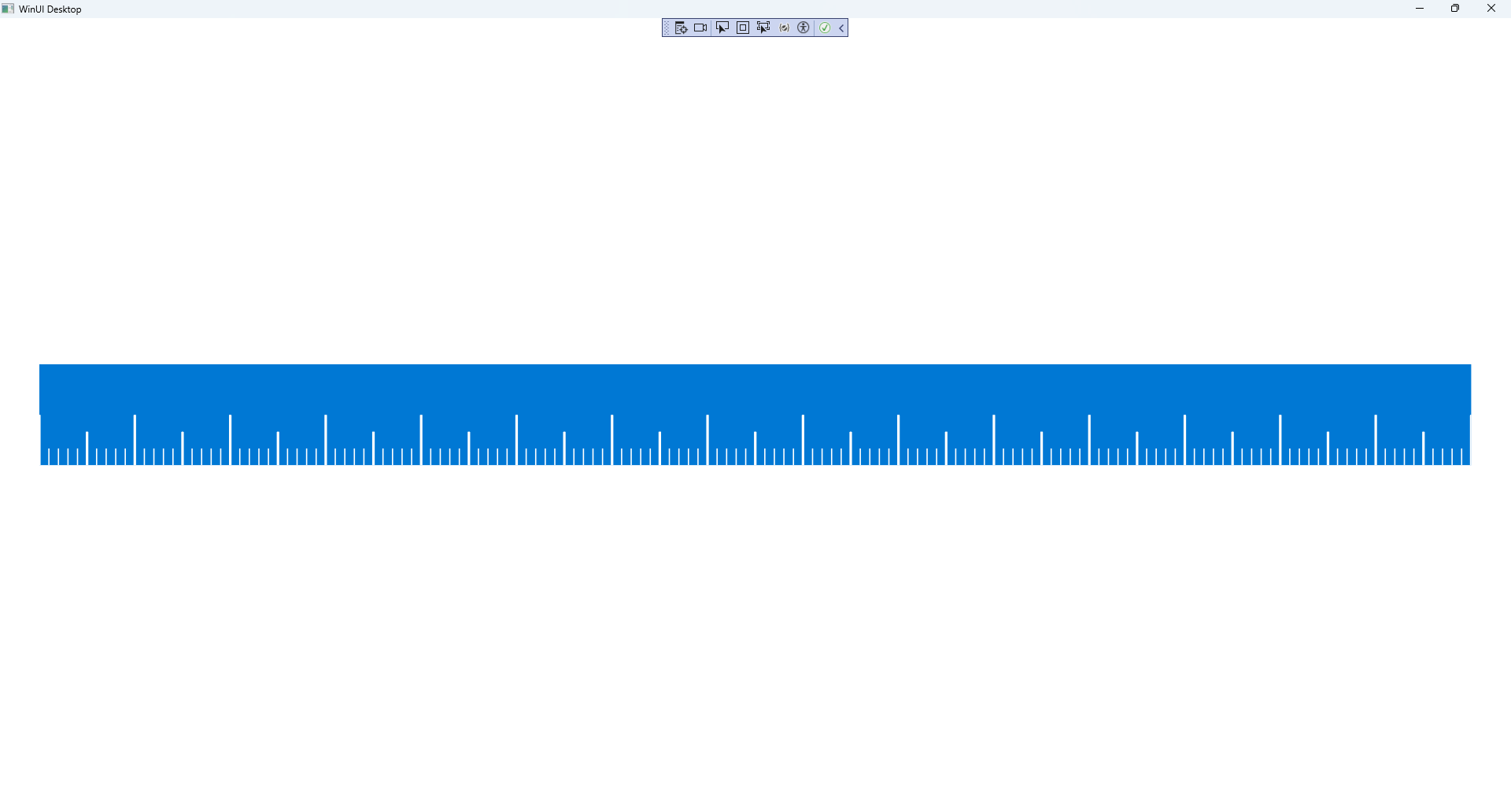
}

## Step 15

|  |  |
| --- | --- |
| That completes the **Windows App SDK** application. In **Visual Studio 2022** from the **Toolbar** select **RulerControl (Package)** to **Start** the application. |  |

## Step 16

Once running you will see the **Ruler Control** displayed with the given *Length* and *Unit*.

****

## Step 17

|  |  |
| --- | --- |
| To **Exit** the **Windows App SDK** application, select the **Close** button from the top right of the application as that concludes this **Tutorial** for **Windows App SDK** from [tutorialr.com](https://tutorialr.com)! |  |