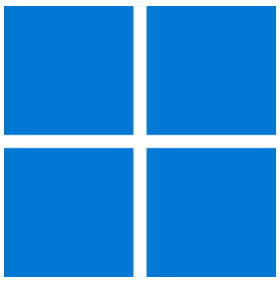
tutorial

Windows App SDK



Matrix Control

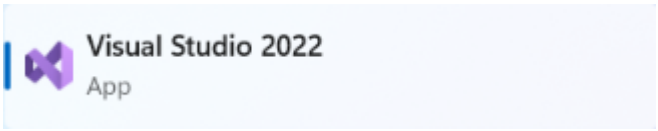
Matrix Control

Matrix Control shows how to create a **Matrix** to display the **Time** or **Date** using **Windows App SDK**

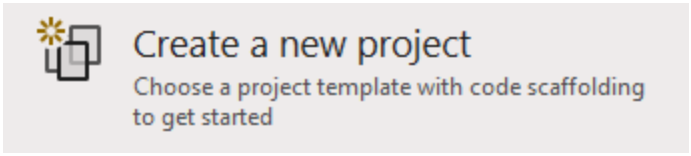
Step 1

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

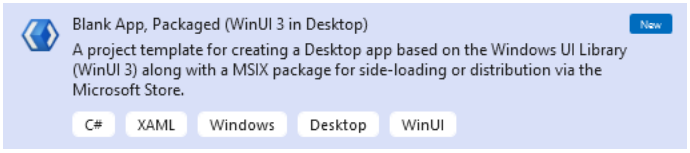
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



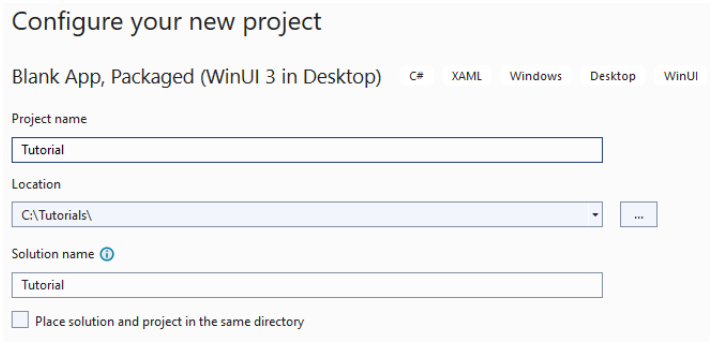
Once **Visual Studio 2022** has started select **Create a new project**.



Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.

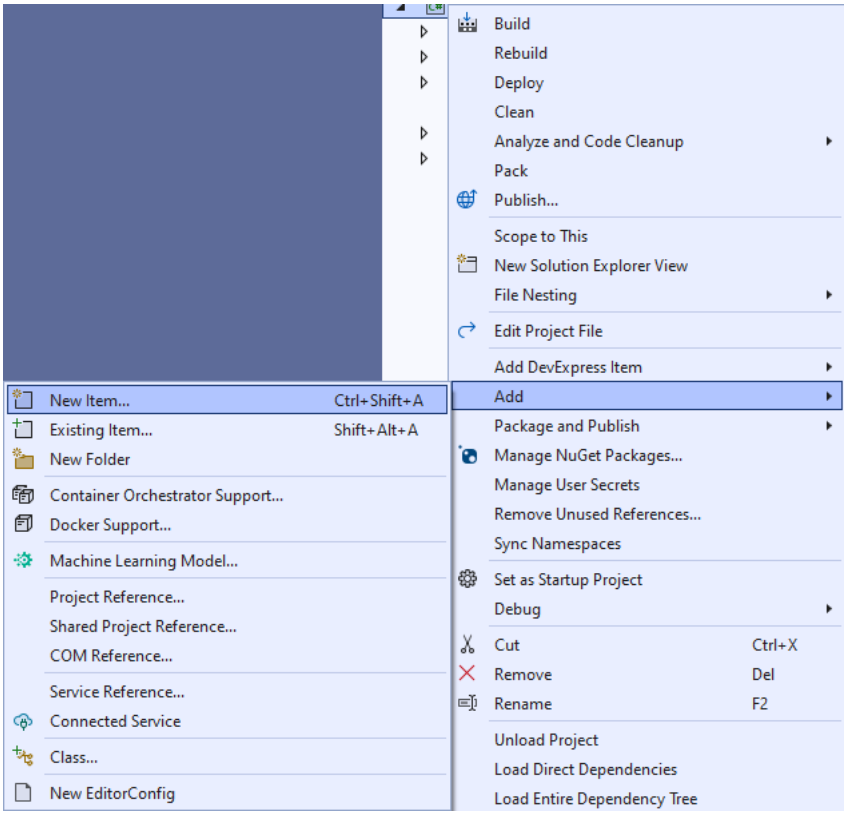


After that in **Configure your new project** type in the **Project name** as *MatrixControl*, then select a Location and then select **Create** to start a new **Solution**.



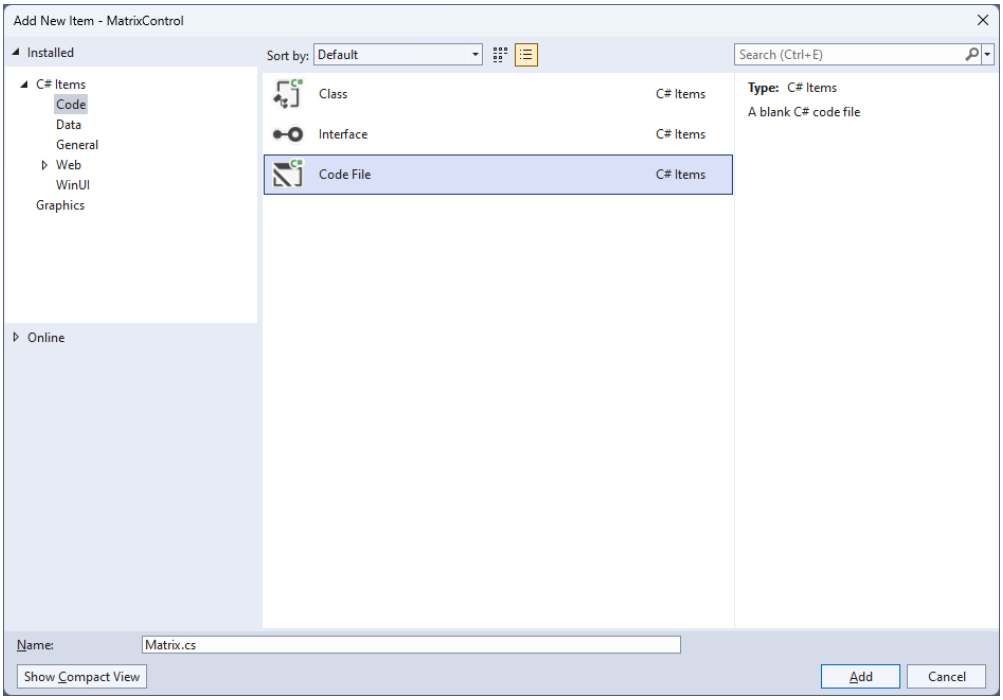
Step 2

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...**



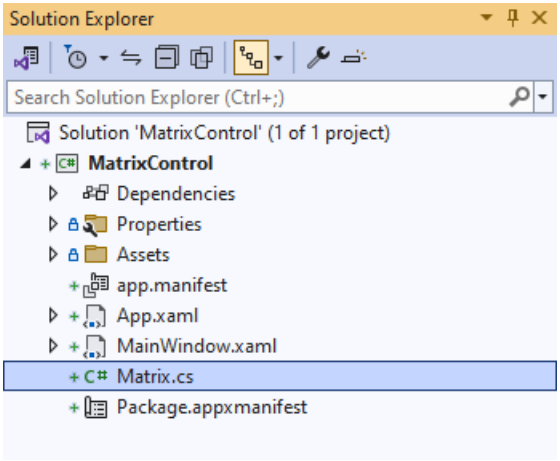
Step 3

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 *Matrix.cs* and then **Click** on **Add**.



Step 4

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



Step 5

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

```
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Data;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Shapes;
using System;
using System.Collections.Generic;
using System.Linq;

namespace MatrixControl;

public enum Sources
{
    Value, Time, Date, TimeDate
}

public class Matrix : StackPanel
{
    private readonly byte[][] table =
    {
        // Table 0 - 4

        // Table 5 - 9

        // Table Minus, Slash, Colon & Space

    };

    // Constants & Members

    // Dependency Properties & Properties

    // Add Element & Add Section Methods

    // Set Layout & Add Layout Methods

    // Value Property & Constructor

}
```

There are **`using`** statements for the **User Control**, a **`namespace`** for **`MatrixControl`** with an **`enum`** for the **`Sources`** of the **Matrix Control** along with a **`class`** of **`Matrix`** that will represent the **User Control**.

Step 6

Then in the **`namespace`** of **`MatrixControl`** in the **`class`** of **`Matrix`** after the **Comment** of **`// Table 0 - 4`** type the following **Code** for the **`table`** which will represent values between *0* and *4*:

```
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0
}, // 0
new byte[] {
0,0,0,0,0,0,0,0,
0,0,0,1,1,0,0,0,
0,1,1,1,1,0,0,0,
0,0,0,1,1,0,0,0,
0,0,0,1,1,0,0,0,
0,0,0,1,1,0,0,0,
0,0,0,0,0,0,0,0
}, // 1
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0
}, // 2
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0
}, // 3
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,0,0,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,1,1,0,
0,0,0,0,0,1,1,0,
0,0,0,0,0,0,0,0
}, // 4
```

Step 7

While still in the **`namespace`** of **`MatrixControl`** in the **`class`** of **`Matrix`** after the **Comment** of **`// Table 5 - 9`** type the following **Code** for the **`table`** which will represent values between *5* and *9*:

```
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0
}, // 5
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0
}, // 6
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,1,1,0,
0,0,0,0,0,1,1,0,
0,0,0,0,0,1,1,0,
0,0,0,0,0,1,1,0,
0,0,0,0,0,0,0,0
}, // 7
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0
}, // 8
new byte[] {
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,1,1,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,1,1,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0
}, // 9
```

Step 8

While still in the **`namespace`** of **`MatrixControl`** in the **`class`** of **`Matrix`** after the **Comment** of **`// Table Minus, Slash, Colon & Space`** type the following **Code** for the **`table`** which will represent a *Minus*, *Slash*, *Colon*, and *Space*:

```
new byte[] {
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,1,1,1,1,1,1,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0
}, // Minus
new byte[] {
0,0,0,0,0,0,0,0,
0,0,0,0,0,1,1,0,
0,0,0,0,1,1,0,0,
0,0,0,1,1,0,0,0,
0,0,1,1,0,0,0,0,
0,1,1,0,0,0,0,0,
0,0,0,0,0,0,0,0
}, // Slash
new byte[] {
0,0,0,0,0,0,0,0,
0,0,0,1,1,0,0,0,
0,0,0,1,1,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,1,1,0,0,0,
0,0,0,1,1,0,0,0,
0,0,0,0,0,0,0,0
}, // Colon
new byte[] {
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0
} // Space
```

Step 9

While still in the **`namespace`** of **`MatrixControl`** in the **`class`** of **`Matrix`** after the **Comment** of **`// Constants & Members`** type the following **Constants** and **Members**:

```
private readonly List<char> glyphs = new()
{
    '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '-', '/', ':', ' '
};

private const string time = "HH:mm:ss";
private const string date = "dd/MM/yyyy";
private const string date_time = "HH:mm:ss dd/MM/yyyy";
private const string invalid_source = "Invalid argument";
private const int padding = 1;
private const int columns = 8;
private const int rows = 7;

private string _value;
private int _count;
```

The **Constants** include an **Array** of **`glyphs`** that will represent what can be displayed including *Digits* or *Minus*, *Slash*, *Colon*, and *Space* along with **Members** for the **Matrix Control**.

Step 10

While still in the **`namespace`** of **`MatrixControl`** in the **`class`** of **`Matrix`** 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(Matrix), null);

public static readonly DependencyProperty SourceProperty =
DependencyProperty.Register(nameof(Source), typeof(Sources),
typeof(Matrix), new PropertyMetadata(Sources.Time));

public static readonly DependencyProperty SizeProperty =
DependencyProperty.Register(nameof(Size), typeof(UIElement),
typeof(Matrix), new PropertyMetadata(4));

public Brush Foreground
{
    get { return (Brush)GetValue(ForegroundProperty); }
    set { SetValue(ForegroundProperty, value); }
}

public Sources Source
{
    get { return (Sources)GetValue(SourceProperty); }
    set { SetValue(SourceProperty, value); }
}

public int Size
{
    get { return (int)GetValue(SizeProperty); }
    set { SetValue(SizeProperty, value); }
}
```

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

Step 11

While still in the **namespace** of **MatrixControl** in the **class** of **Matrix** after the **Comment** of **// Add Element & Add Section Methods** type the following **Methods**:

```
private Rectangle AddElement(string name, int left, int top)
{
    var element = new Rectangle()
    {
        Tag = name,
        Opacity = 0,
        RadiusX = 1,
        RadiusY = 1,
        Width = Size,
        Height = Size,
        Margin = new Thickness(2)
    };
    element.SetBinding(Shape.FillProperty, new Binding()
    {
        Path = new PropertyPath(nameof(Foreground)),
        Mode = BindingMode.TwoWay,
        Source = this
    });
    Canvas.SetLeft(element, left);
    Canvas.SetTop(element, top);
    return element;
}

private void AddSection(string name)
{
    int x = 0;
    int y = 0;
    int index = 0;
    var section = new Canvas()
    {
        Tag = name,
        Height = rows * Size,
        Width = columns * Size
    };
    for (int row = 0; row < rows; row++)
    {
        for (int column = 0; column < columns; column++)
        {
            section.Children.Add(AddElement($"{name}.{index}", x, y));
            x = x + Size + padding;
            index++;
        }
        x = 0;
        y = y + Size + padding;
    }
    Children.Add(section);
}
```

The **Method** of **AddElement** will create an element for the **Matrix Control** which is used by **AddSection**.

Step 12

While still in the **namespace** of **MatrixControl** in the **class** of **Matrix** after the **Comment** of **// Set Layout & Add Layout Methods** type the following **Methods**:

```
private void SetLayout(string name, char glyph)
{
    var layout = Children.Cast<Canvas>()
        .FirstOrDefault(f => (string)f.Tag == name);
    int pos = glyphs.IndexOf(glyph);
    byte[] values = table[pos];
    for (int index = 0; index < layout.Children.Count; index++)
    {
        layout.Children.Cast<Rectangle>()
        .FirstOrDefault(f => (string)f.Tag == $"{name}.{index}")
        .Opacity = values[index];
    }
}

private void AddLayout()
{
    var array = _value.ToCharArray();
    var length = array.Length;
    var list = Enumerable.Range(0, length);
    if (_count != length)
    {
        Children.Clear();
        foreach (int item in list)
        {
            AddSection(item.ToString());
        }
        _count = length;
    }
    foreach (int item in list)
    {
        SetLayout(item.ToString(), array[item]);
    }
}
```

The **Method** of **SetLayout** will display the appropriate value for the **Matrix Control** by setting the **Opacity** and **AddLayout** will setup the display of the **Matrix Control**.

Step 13

While still in the **namespace** of **MatrixControl** in the **class** of **Matrix** after the **Comment** of **// Value Property & Constructor** type the following **Property** and **Constructor**:

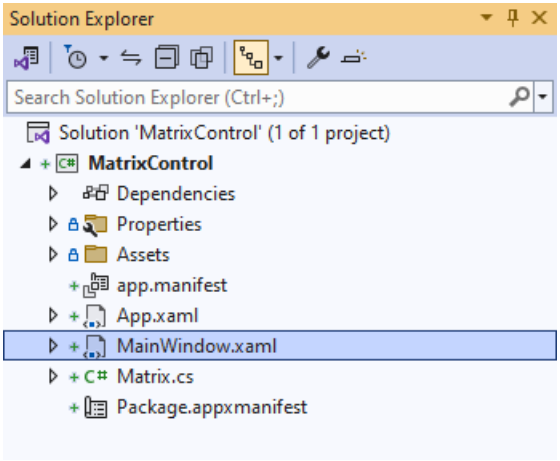
```
public string Value
{
    get { return _value; }
    set { _value = value; AddLayout(); }
}

public Matrix()
{
    Orientation = Orientation.Horizontal;
    var timer = new DispatcherTimer()
    {
        Interval = TimeSpan.FromMilliseconds(250)
    };
    timer.Tick += (object s, object args) =>
    {
        if (Source != Sources.Value)
        {
            var format = Source switch
            {
                Sources.Time => time,
                Sources.Date => date,
                Sources.TimeDate => date_time,
                _ => throw new ArgumentException(invalid_source)
            };
            Value = DateTime.Now.ToString(format);
        }
    };
    timer.Start();
}
```

The **Property** of **Value** will setup the display of the **Matrix Control** using the **Method** of **AddLayout** and the **Constructor** will setup a **DispatcherTimer** to be used to display the **Value** of the **Matrix Control**.

Step 14

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



Step 15

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 16

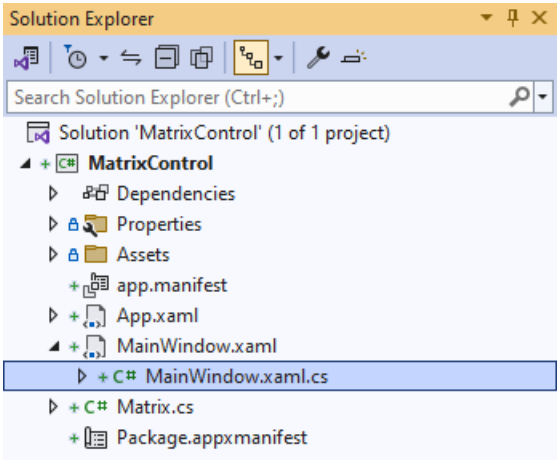
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

```
<Viewbox>
    <local:Matrix Padding="50" Source="Time"
    Foreground="{ThemeResource AccentButtonBackground}"/>
</Viewbox>
```

This **XAML** contains a **`ViewBox`** including the **User Control** of **`Matrix`** with the **`Source`** set to **`Time`**.

Step 17

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 18

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

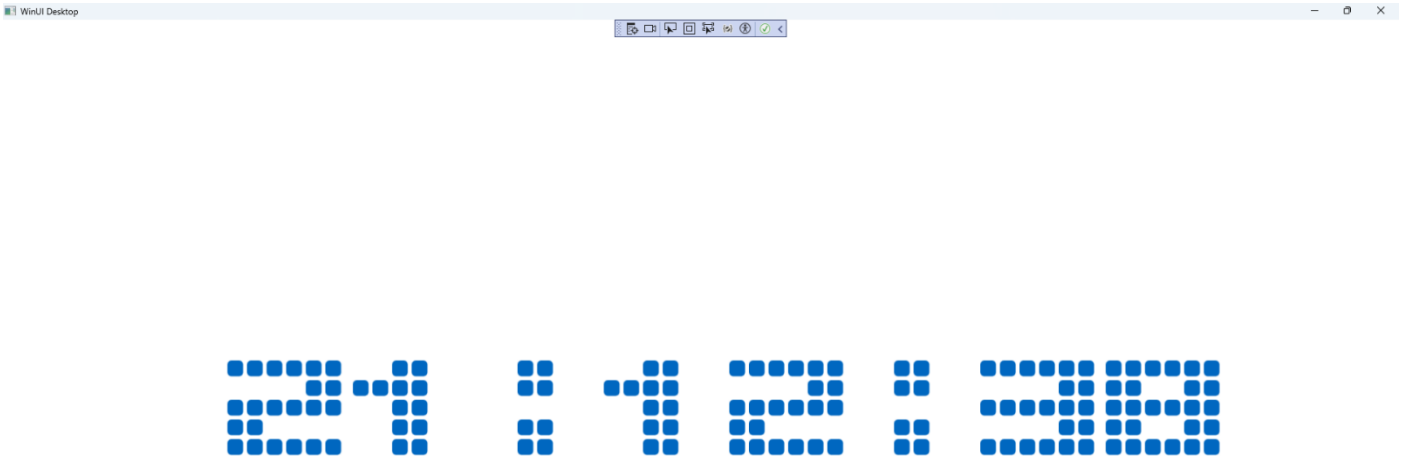
```
private void myButton_Click(object sender, RoutedEventArgs e)
{
    myButton.Content = "Clicked";
}
```

Step 19

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

Step 20

Once running you will see the **Matrix Control** displaying the current *Time*.



Step 21

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!