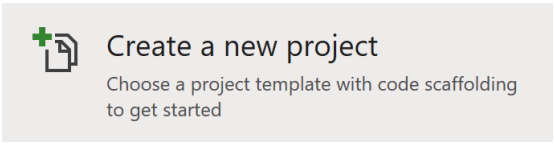
Universal Windows Platform – Sound Game

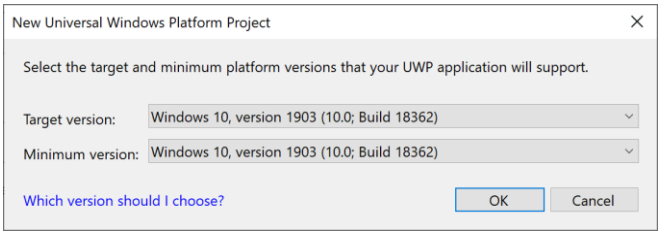
Sound Game shows how to create a simple game to play sounds using a **MediaElement** with a provided frequency as an audio stream

Step 1



Follow **Setup and Start** on how to Install and/or Get Started with **Visual Studio 2019** if not already or in **Windows 10** choose **Start**, find and select **Visual Studio 2019** then from the **Get started** screen select **Create a new project**

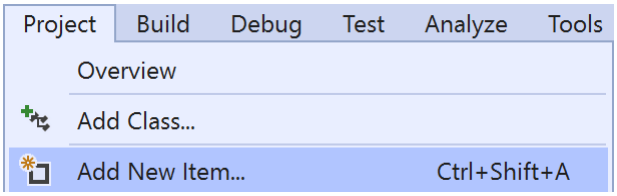
Then choose **Blank App (Universal Windows)** and select **Next** and then in **Configure your new project** enter the **Project name** as **SoundGame** and select **Create**



Finally, in **New Universal Windows Platform Project** pick the **Target version** and **Minimum version** to be at least **Windows 10, version 1903 (10.0; Build 18362)** and then select **OK**

Target Version will control the most recent features of Windows 10 your application can use. To make sure you always have the most recent version, check for any Notifications or Updates in Visual Studio 2019

Step 2



Choose **Project** then **Add New Item...** from the **Menu** in **Visual Studio 2019**

Step 3

Then choose **Code File** from **Add New Item** in **Visual Studio 2019**, enter the **Name** as **Library.cs** and select **Add**

Universal Windows Platform – Sound Game

Step 4

In the **Code** View of **Library.cs** will be displayed and in this the following should be entered:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using Windows.Storage.Streams;
using Windows.UI;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Media;

public class Library
{
    private const short tracks = 1;
    private const short formatType = 1;
    private const short bitsPerSample = 16;
    private const int headerSize = 8;
    private const int formatChunkSize = 16;
    private const int samplesPerSecond = 44100;
    private const short frameSize =
        tracks * ((bitsPerSample + 7) / 8);
    private const int bytesPerSecond =
        samplesPerSecond * frameSize;
    private const int waveSize = 4;
    private const int riff = 0x46464952;
    private const int wave = 0x45564157;
    private const int data = 0x61746164;
    private const int format = 0x20746D66;
    private const int samples = 88200 * 4;
    private const int dataChunkSize =
        samples * frameSize;
    private const int fileSize =
        waveSize + headerSize + formatChunkSize +
        headerSize + dataChunkSize;
    private const string mime = "audio/wav";

    private readonly Dictionary<string, double>
        _notes = new Dictionary<string, double>()
    {
        { "C", 261.6 }, { "C#", 277.2 }, { "D", 293.7 },
        { "D#", 311.1 } , { "E", 329.6 }, { "F", 349.2 },
        { "F#", 370.0 }, { "G", 392.0 }, { "G#", 415.3 },
        { "A", 440.0 }, { "A#", 466.2 }, { "B", 493.9 }
    };
    private readonly MediaElement _playback = new MediaElement();
    private readonly Color _accent =
    (Color)Application.Current.Resources["SystemAccentColor"];


}
```

Universal Windows Platform – Sound Game

There are `using` statements to include necessary functionality and `private const int` that defind the elements to make up the sounds to be played, there's a `Dictionary<string, double>` for each note to be played and a `MediaElement` to allow playback of the sounds

Then below the **`private readonly Color _accent = (Color)Application.Current.Resources["SystemAccentColor"];`** line the following **method** should be entered:

```
private void Play(double note)
{
    IRandomAccessStream stream = new InMemoryRandomAccessStream();
    BinaryWriter writer = new BinaryWriter(stream.AsStream());
    double frequency = note * 1.5;
    writer.Write(riff);
    writer.Write(fileSize);
    writer.Write(wave);
    writer.Write(format);
    writer.Write(formatChunkSize);
    writer.Write(formatType);
    writer.Write(tracks);
    writer.Write(samplesPerSecond);
    writer.Write(bytesPerSecond);
    writer.Write(frameSize);
    writer.Write(bitsPerSample);
    writer.Write(data);
    writer.Write(dataChunkSize);
    for (int index = 0; index < samples / 4; index++)
    {
        double time = index / (double)samplesPerSecond;
        short sample = (short)(10000 *
            Math.Sin(time * frequency * 2.0 * Math.PI));
        writer.Write(sample);
    }
    stream.Seek(0);
    _playback.SetSource(stream, mime);
    _playback.Play();
}
```

`Play` method is used to play a musical note with a `MediaElement` and an `InMemoryRandomAccessStream` which will be used to help create the audio with a `BinaryWriter` to create a wave audio stream that will be written to with the given frequency to produce the samples that will create the required musical note

Universal Windows Platform – Sound Game

Next below the **`private void Play(...) { ... }` method** the following **method** should be entered:

```
private void Add(Grid grid, int column)
{
    Button button = new Button()
    {
        Width = 20,
        Height = 80,
        FontSize = 10,
        Margin = new Thickness(5),
        Padding = new Thickness(0),
        Content = _notes.Keys.ElementAt(column),
        Background = new SolidColorBrush(_accent),
        Foreground = new SolidColorBrush(Colors.White),
        Style = (Style)Application.Current.Resources
        ["ButtonRevealStyle"]
    };
    button.Click += (object sender, RoutedEventArgs e) =>
    {
        button = (Button)sender;
        int note = Grid.GetColumn(button);
        Play(_notes[_notes.Keys.ElementAt(note)]);
    };
    button.SetValue(Grid.ColumnProperty, column);
    grid.Children.Add(button);
}
```

`Add` method will create a `Button` and set the properties for this and when clicked it will trigger the `Play` method

Then below the **`private void Add(...) { ... }` method** the following **method** should be entered:

```
private void Layout(Grid Grid)
{
    Grid.Children.Clear();
    Grid.RowDefinitions.Clear();
    Grid.ColumnDefinitions.Clear();
    // Setup Grid
    for (int Column = 0; (Column < _notes.Count); Column++)
    {
        Grid.ColumnDefinitions.Add(new ColumnDefinition());
        Add(Grid, Column);
    }
}
```

`Layout` method is used to create the look-and-feel using a `Grid` by calling the `Add` method

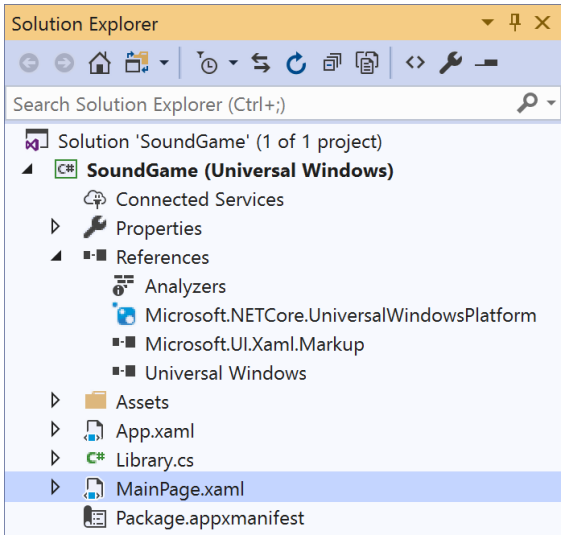
Universal Windows Platform – Sound Game

Finally after the **`private void Layout(...) { ... }` method** the following **`public` method** should be entered:

```
public void New(Grid grid)
{
    Layout(grid);
}
```

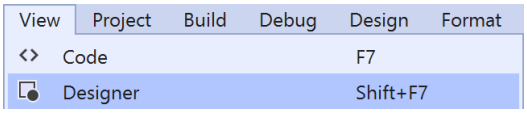
`New` method will setup and start playing the game by calling the `Layout` method

Step 5



In the **Solution Explorer** of **Visual Studio 2019** select **MainPage.xaml**

Step 6



Choose **View** then **Designer** from the **Menu** in **Visual Studio 2019**

Universal Windows Platform – Sound Game

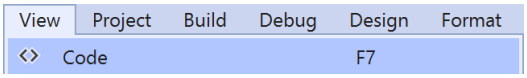
Step 7

In the **Design** View and **XAML** View of **Visual Studio 2019** will be displayed, and in this between the **`Grid`** and **`/Grid`** elements enter the following **XAML**:

```
<Viewbox>
    <Grid Margin="50" Name="Display"
    HorizontalAlignment="Center"
    VerticalAlignment="Center"/>
</Viewbox>
<CommandBar VerticalAlignment="Bottom">
    <AppBarButton Icon="Page2" Label="New" Click="New_Click"/>
</CommandBar>
```

The first block of XAML the main user interface features a Grid to represent the game and the second block of XAML is the CommandBar which contains New to setup and start the game

Step 8



Choose **View** then **Code** from the **Menu** in **Visual Studio 2019**

Step 9

Once in the **Code** View, below the end of **`public MainPage() { ... }`** the following Code should be entered:

```
Library library = new Library();

private void New_Click(object sender, RoutedEventArgs e)
{
    library.New(Display);
}
```

Below the MainPage method an instance of the **`Library`** class is created. In the **`New_Click(...)`** Event handler will setup and play the game using the **`New`** method in the **`Library`** class

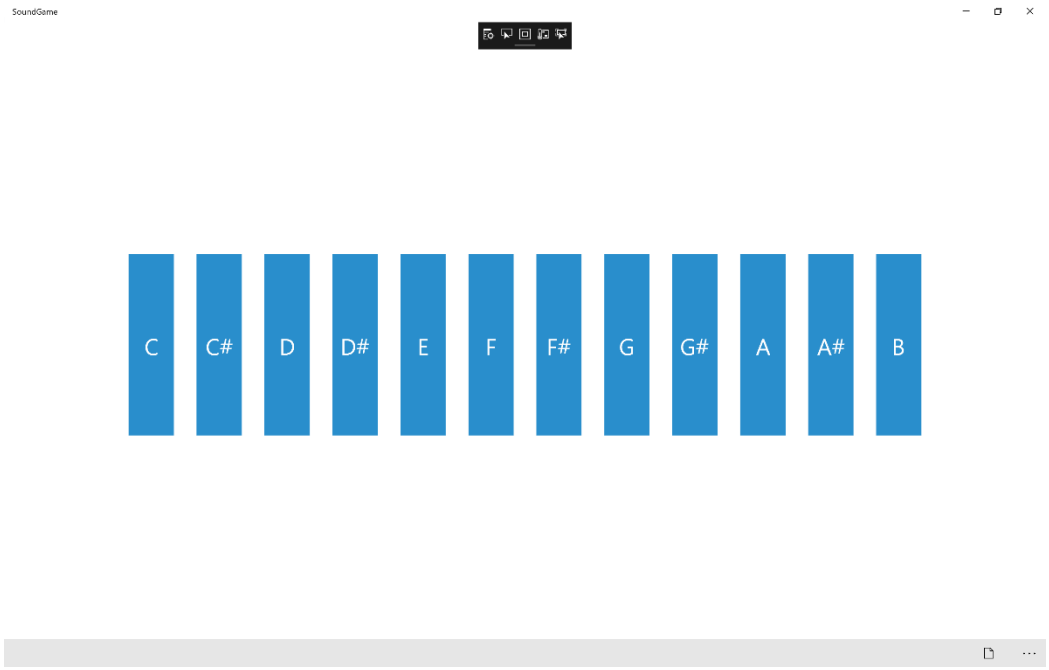
Universal Windows Platform – Sound Game

Step 10

That completes the **Universal Windows Platform** Application, in **Visual Studio 2019** select **Local Machine** to run the Application

Step 11

Once the Application is running use **New** to start then can play sounds by clicking the buttons



Step 12

To Exit the Application, select the **Close** button in the top right of the Application