Лабораторная работа №6:
Работа с медиа файлами

Цель:

Целью данной работы является получение навыков работы с медиа файлами на языке высокого уровня C# в среде программирования Microsoft Visual Studio 2022 Community

Задание:

Разработать и реализовать программу “Mp3 Player” на основе класса MediaPlayer. Программа должна содержать следующий функционал:

1) Выбор нескольких аудио файлов в формате mp3.
2) Отображение названий выбранных файлов в компоненте ListBox.
3) Выбор и воспроизведение файла из компонента ListBox.
4) Последовательное воспроизведение файлов из компонента ListBox.
5) Воспроизведение файлов из компонента ListBox в случайном порядке.
6) Возможность остановить, запустить и поставить на паузу текущий воспроизводимый файл.
7) Возможность перейти к произвольному моменту воспроизводимого файла при помощи компонента Slider.
8) Отображение общей длительности воспроизводимого файла и текущего времени воспроизведения.
9) Возможность регулирования громкости воспроизведения.

Разработать и реализовать программу “Video Player” на основе компонента MediaElement. Программа должна содержать следующий функционал:

1) Выбор и загрузка видео файла.
2) Возможность остановить, запустить и поставить на паузу текущий воспроизводимый файл.
3) Возможность перейти к произвольному моменту воспроизводимого файла при помощи компонента Slider.
4) Отображение общей длительности воспроизводимого файла и текущего времени воспроизведения.
5) Возможность регулирования громкости воспроизведения.

Все сообщения обеих программ (ошибки и уведомления), должны сопровождаться звуками.

Справочная информация:

Класс SoundPlayer

Для воспроизведения простых, одиночных звуков (например – нажатие кнопок или озвучка всплывающего сообщения), может быть использован класс SoundPlayer. Класс поддерживает работу с форматом файлов .WAV, синхронную и асинхронную загрузку, а также, одиночное и цикличное воспроизведение звуков.

Загрузка и одиночное воспроизведение звукового файла могут выглядеть следующим образом:

```
//подключение пространства имён
using System.Media;
…
//диалог выбора файла
OpenFileDialog dlg = new OpenFileDialog();
dlg.ShowDialog();
//создание объекта и указание пути к файлу
SoundPlayer sp = new SoundPlayer();
sp.SoundLocation = dlg.FileName;
//загрузка файла
```

```
sp.Load();
//проигрывание файла
sp.Play();
```

В данном примере, файл загружается синхронно, то есть, работа программы будет остановлена до тех пор, пока не будет загружен файл, а проигрывание звука осуществляется асинхронно. В некоторых случаях, имеет смысл загружать асинхронно, то есть, программа будет продолжать выполнение, не дожидаясь окончания загрузки файлов.

Асинхронная загрузка может быть реализована следующим образом:

```
//команда на асинхронную загрузку и назначение обработчика события на завершение загрузки
sp.LoadAsync();
sp.LoadCompleted += Sp_LoadCompleted;
…
//функция, вызываемая при завершении загрузки
private void Sp_LoadCompleted(object sender, System.ComponentModel.AsyncCompletedEventArgs e)
{
    MessageBox.Show("Звук загружен!");
}
```

Проигрывать звук синхронно (программа будет приостановлена до окончания воспроизведения звука), либо циклически, можно следующим образом:

```
//синхронно
sp.PlaySync();
//циклически
sp.PlayLooping();
```

Помимо запуска воспроизведения звука при вызове одной из соответствующих функций, существует механизм “триггеров”, которые срабатывают автоматически при наступлении какого-либо события (нажатие кнопки, движение курсора мыши и т.д.).

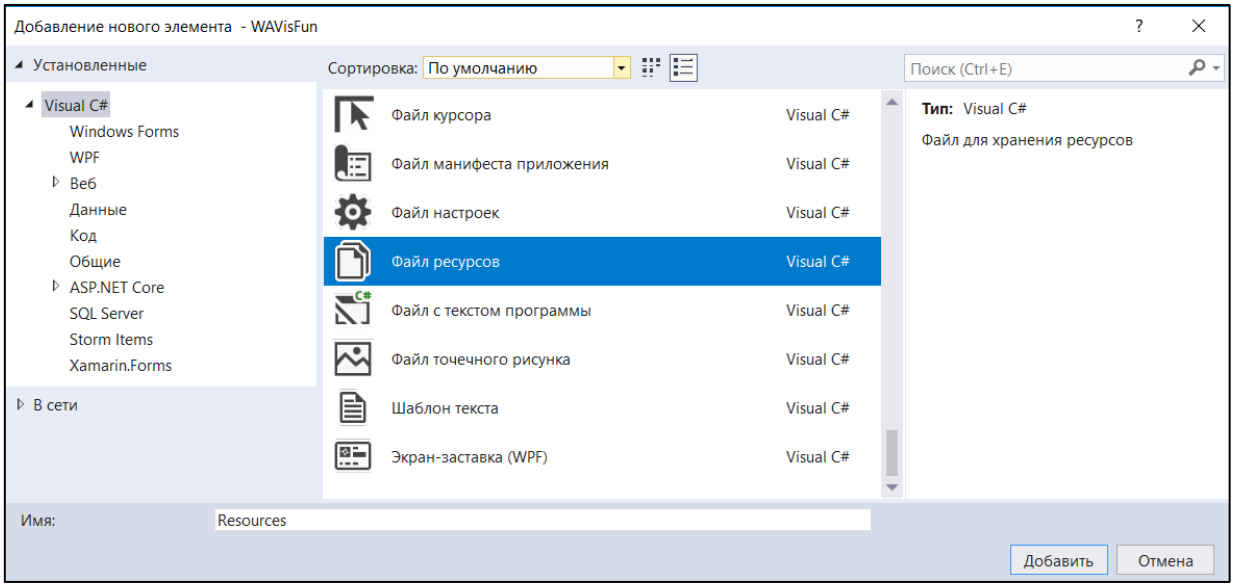
Простой триггер может быть описан следующим образом:

В приведённом примере, звук “tada.wav” будет проигрываться в момент попадания курсора мыши в область кнопки с именем “play”. Для того, чтобы звук проигрывался при нажатии на кнопку, нужно изменить код EventTrigger следующим образом:

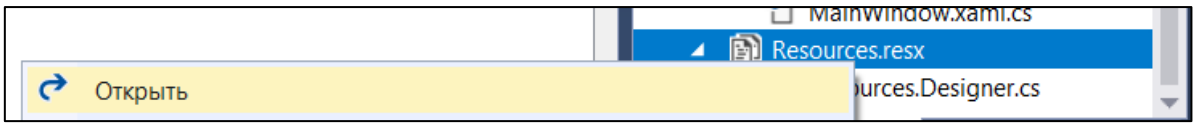
```
<EventTrigger RoutedEvent="Button.Click">
    <SoundPlayerAction Source="Resources/tada.wav" />
</EventTrigger>
```

Поскольку обычно, файлы типа .wav занимают не много места, их можно добавлять в проект как компилируемые ресурсы.

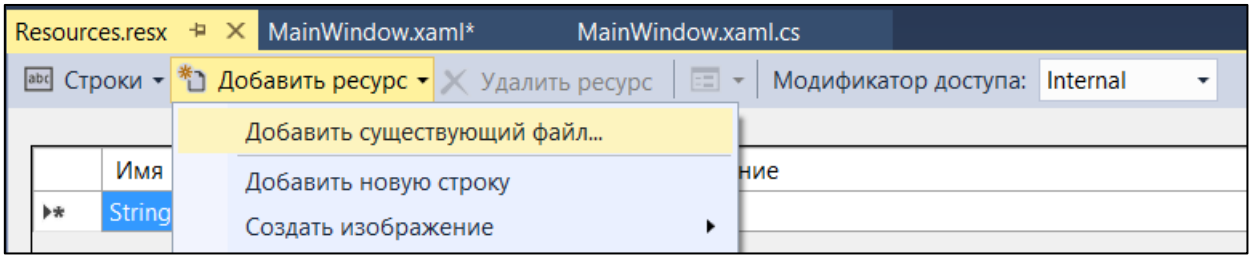
Для этого добавьте к проекту ресурс:

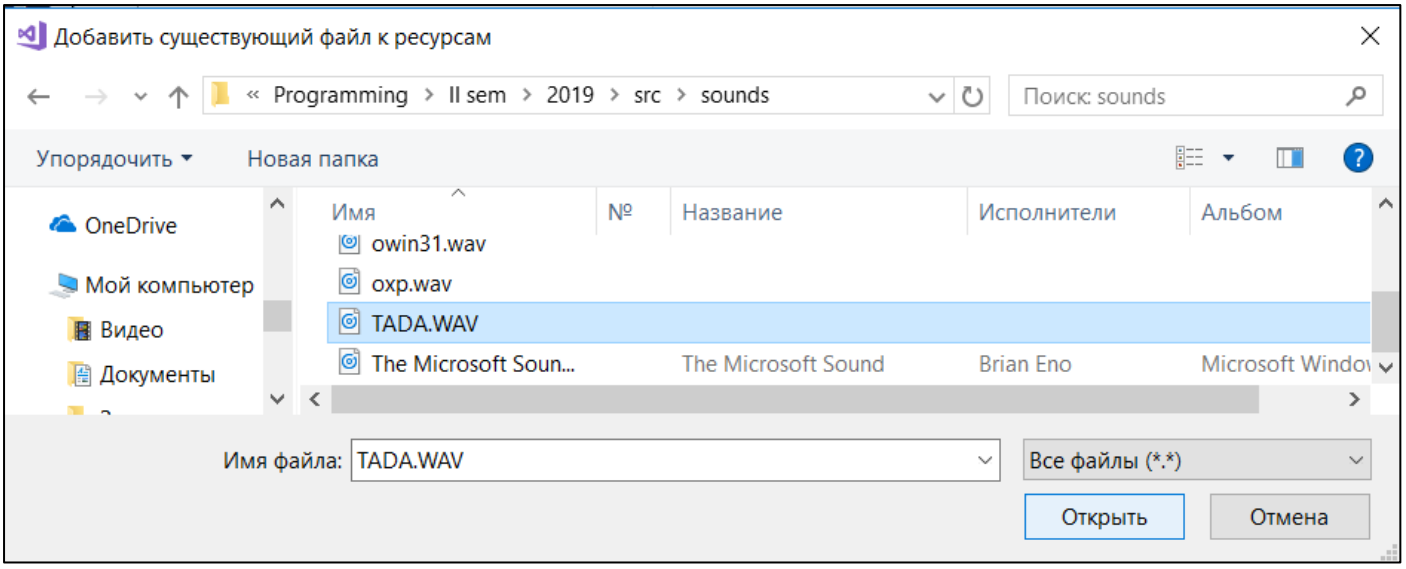


После чего, нажмите правой кнопкой мыши на ресурс и выберите “Открыть”:

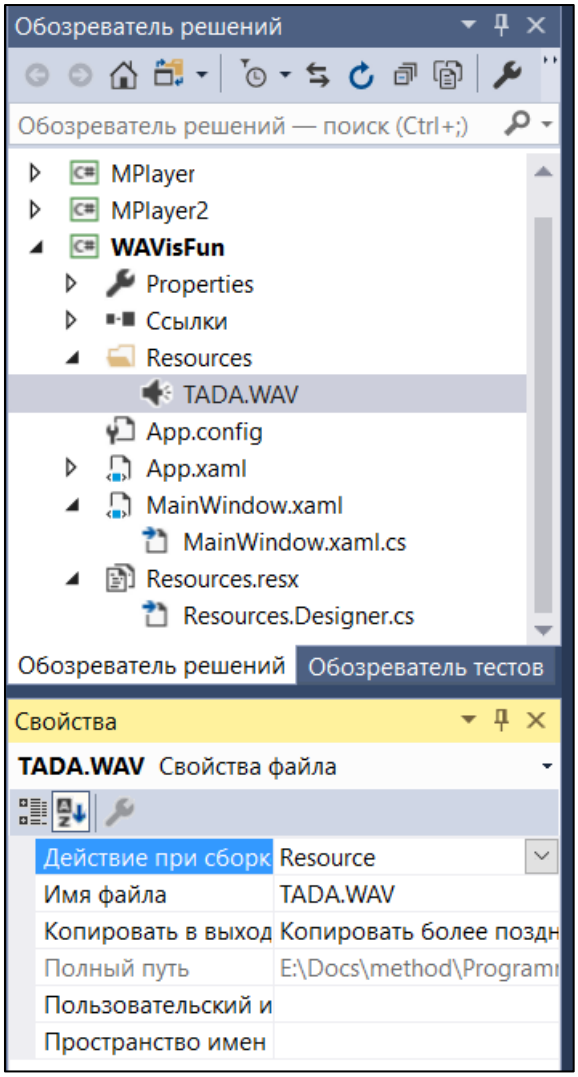


А затем “Добавить ресурс” -> Добавить существующий файл…:





Затем, выберите загруженный звук в и выберите “Действие при сборке” – Resource:



После этого, загружать звук можно будет следующим образом:

```
//создание объекта
SoundPlayer sp = new SoundPlayer();
//загрузка звука из ресурсов
sp.Stream = Properties.Resources.TADA;
//проигрывание звука
sp.Play();
```

Помимо загрузки пользовательских звуков, можно использовать стандартный набор звуков операционной системы Windows:

```
SystemSounds.Asterisk.Play();    //Вопросительный знак
SystemSounds.Beep.Play();        //Уведомление о получении почты
SystemSounds.Exclamation.Play();//Восклицание
SystemSounds.Hand.Play();        //Критическая ошибка
SystemSounds.Question.Play();    //Вопрос
```

Класс MediaPlayer

Не смотря на простоту использования, объект SoundPlayer имеет ряд недостатков. Например, в один момент времени может проигрываться только один звук. Кроме того, объект плохо подходит для воспроизведения длинных аудио файлов и не имеет настроек воспроизведения.

Для подобных задач существует класс MediaPlayer:

```
//создание объекта, обычно глобального
MediaPlayer player = new MediaPlayer();
```

```
…
//выбор медиа файла, например, в формате .mp3
OpenFileDialog dlg = new OpenFileDialog();
dlg.ShowDialog();
//загрузка выбранного файла
player.Open(new Uri(dlg.FileName, UriKind.Relative));
//воспроизведение
player.Play();
```

Класс MediaPlayer предоставляет множество параметров, отображающих тип воспроизводимого файла, его параметры, а также настройки управления воспроизведением:

Balance – устанавливает баланс между левым и правым каналом как число от -1 (только левый канал) до 1 (только правый канал).

Volume – устанавливает громкость в виде числе от 0 (полная тишина) до 1 (полная громкость). Значение по умолчанию равно 0.5.

SpeedRatio – устанавливает повышенную скорость при воспроизведении звука (или видео). Значение по умолчанию равно 1, что означает нормальную скорость, в то время как 2 — двойную скорость 0.5 — половину нормальной скорости и т.д. Можно использовать любое положительное значение типа double.

HasAudio и **HasVideo** – указывает на то, содержит ли текущий загруженный медиафайл, соответственно, аудио- и видеосоставляющие.

NaturalDuration, **NaturalVideoHeight** и **NaturalVideoWidth** – содержит длительность содержимого, а так же высоту и ширину видео, если это видео файл.

Position – объект TimeSpan, указывающий текущее местоположение в медиафайле. Это свойство можно устанавливать для пропуска части файла и продолжения воспроизведения с указанного места.

DownloadProgress и **BufferingProgress** – показывает процент загружаемого файла (для случаев, когда источником является URL, указывающий на местоположение в Интернете). Процент представлен в виде числа от 0 до 1

Clock – получает или устанавливает часы MediaClock, ассоциированные с проигрывателем. MediaClock используется только тогда, когда аудио синхронизируется с временной шкалой.

Open() - загружает новый медиафайл.

Play() – Начинает воспроизведение.

Pause() – временно приостанавливает воспроизведение, не меняя его позиции. Для продолжения воспроизведения необходимо вызвать Play().

Stop() – останавливает воспроизведение и сбрасывает позицию на начало файла.

События: **MediaOpened, MediaEnded** и **MediaFailed**.

Пример использования свойств и методов MediaPlayer:

```
//назначение обработчика события на открытие медиа файла
player.MediaOpened += Player_MediaOpened;
…
//обработчик события открытия медиа файла
private void Player_MediaOpened(object sender, EventArgs e)
{
    //получение длительности медиа файла (свойство доступно только после события открытия)
    Duration d = player.NaturalDuration;
    //получение половины длительности медиа файла в секундах
    int half = Convert.ToInt32(d.TimeSpan.TotalSeconds / 2);
    //установка проигрывания со середины медиа файла
    player.Position = new TimeSpan(0, 0, half);
}
```

Для воспроизведения видео файлов необходимо использовать объект типа VideoDrawing, а также визуальный компонент типа Image:

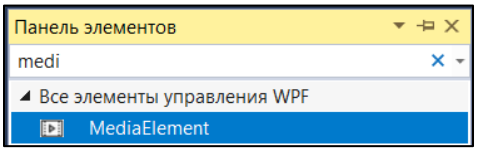
```
//загрузка видео файла
```

```
player.Open(new Uri(dlg.FileName, UriKind.Relative));
//создание объекта для рисования кадров
VideoDrawing aVideoDrawing = new VideoDrawing();
//установка области рисования и ассоциированного проигрывателя
aVideoDrawing.Rect = new Rect(0, 0, 200, 200);
aVideoDrawing.Player = player;
//отрисованный кадр
DrawingImage exampleDrawingImage = new DrawingImage(aVideoDrawing);
//настройка компонента типа Image с именем img
img.Source = exampleDrawingImage;
img.Stretch = Stretch.Uniform;
img.HorizontalAlignment = HorizontalAlignment.Left;
//начало воспроизведения
player.Play();
```

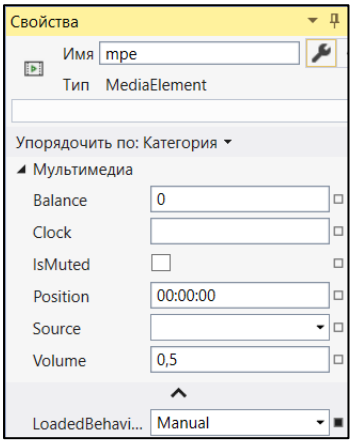
В один момент времени может существовать более одного медиа плейера и воспроизводиться больше одного медиа файла.

Компонент MediaElement

Несмотря на то, что класс MediaPlayer может быть использован для воспроизведения видео файлов, гораздо удобнее это делать при помощи визуального компонента MediaElement:



После размещения компонента на форме, имеет смысл сразу задать свойство LoadedBehavior – Manual, для того, чтобы иметь возможность управлять воспроизведением медиа файлов:



Работа с MediaElement во многом похожа на работу с MediaPlayer:

```
//выбор файла
OpenFileDialog dlg = new OpenFileDialog();
dlg.ShowDialog();
//установка источника
mpe.Source = new Uri(dlg.FileName, UriKind.Relative);
//установка звука в половину от максимума
mpe.Volume = 50.0/100.0;
//начало проигрывания медиа файла
mpe.Play();
//приостановка воспроизведения
mpe.Pause();
//остановка воспроизведения
mpe.Stop();
```

Для получения длительности медиа файла, необходимо использовать обработчик события загрузки:

```
private void mpe_MediaOpened(object sender, RoutedEventArgs e)
{
```

```
    //где timeline – компонент типа Slider
    timeline.Maximum = mpe.NaturalDuration.TimeSpan.TotalMilliseconds;
}
```

Для “плавного” переключения по “таймлайну” видео, имеет смысл использовать компонент Slider следующего вида:

```
<Slider x:Name="timeline" Thumb.DragCompleted="timeline_ValueChanged"/>
```

И соответствующий ему обработчик события:

```
using System.Windows.Controls.Primitives;
…
private void timeline_ValueChanged(object sender, DragCompletedEventArgs e)
{
    int SliderValue = (int)timeline.Value;

    TimeSpan ts = new TimeSpan(0, 0, 0, 0, SliderValue);
    mpe.Position = ts;
}
```

Простой видео плейер на основе компонентов MediaElement, Menu, Slider и Button может выглядеть следующим образом:

Список литературы:

Класс SoundPlayer:

https://docs.microsoft.com/ru-ru/dotnet/api/system.media.soundplayer?view=netframework-4.7.2

Класс MediaPlayer:

https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.media.mediaplayer?view=netframework-4.7.2

Класс MediaElement:

https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.controls.mediaelement?view=netframework-4.7.2

Синтез речи при помощи класса SpeechSynthesizer:

https://docs.microsoft.com/ru-ru/dotnet/api/system.speech.synthesis.speechsynthesizer?view=netframework-4.7.2

Выбор нескольких файлов через OpenFileDialog:

https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.openfiledialog.multiselect?view=netframework-4.7.2