Sprawozdanie

Nazwa grupy: Stowarzyszenie Kodujących Dżentelmenów

Członkowie: Patryk Adler 235389, Maksymilian Kicki 235310

Nazwa aplikacji: Your Music Player – YMP

Użyte narzędzia, technologie i wtyczki: Microsoft Visual Studio 2017, C#, .Net, Windows Forms, NAudio, TagLib
Sposób współdzielenia plików: GitHub

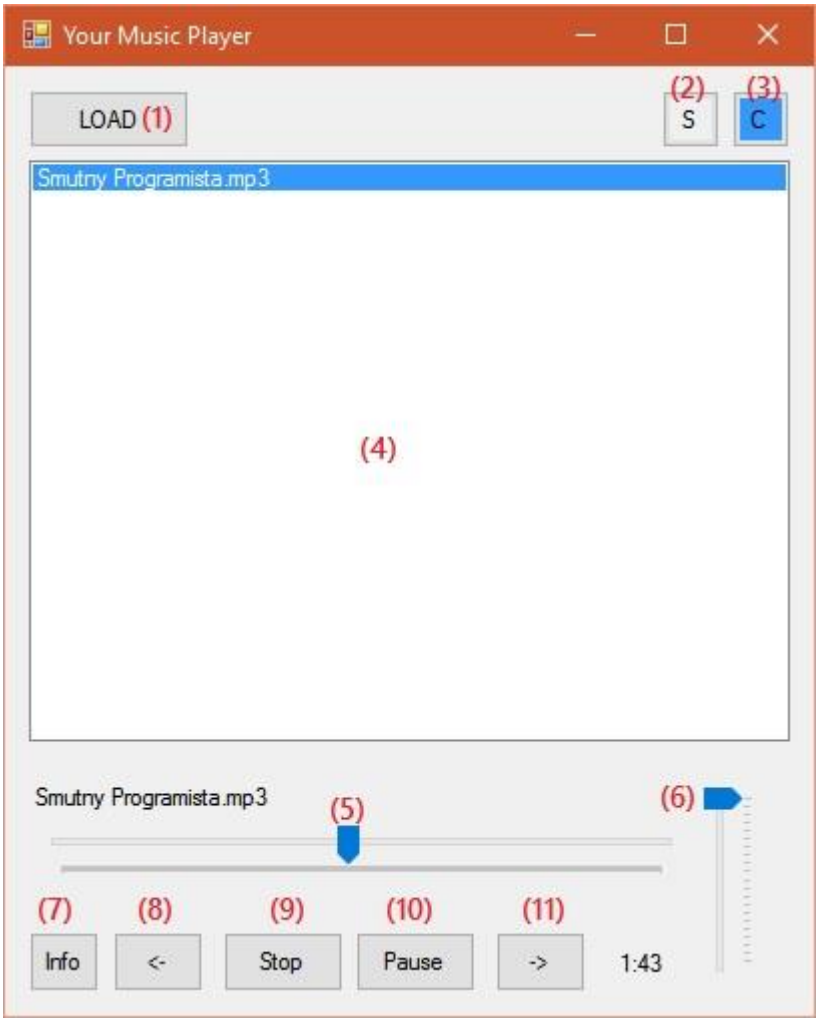
Założenia i cel projektu:
Celem projektu było stworzenie aplikacji do odtwarzania plików muzycznych z funkcją przewijania i obsługą wielu plików i folderów. Aplikacja z założenia miała być wykonana w języku C# przy użyciu technologii .Net wraz z Windows Forms. YMP miał wyświetlać nazwy odtwarzanych plików, jak i pozwalać na odczytanie należytych danych o wczytywanych plikach.

Realizacja projektu:
Aplikacja wykonaliśmy przy użyciu odpowiedniego języka oraz dzięki wybranej technologii. Dzięki wykorzystaniu wyżej wymienionej technologii mogliśmy sobie pozwolić na stworzenie przejrzystego interfejsu użytkownika. Tło pozostawiliśmy podstawowe ze względu na brak talentu artystycznego, jednak pozwoliliśmy sobie na wprowadzenie pewnej wariacji w postaci zmieniającego się tła w zależności od tagów dodanych do plików muzycznych (tło zmienia się na obrazek, który jest przypisany do danego utworu). Oprócz zaimplementowania podstawowych funkcjonalności takich jak odtwarzanie, stopowanie i pauzowanie utworu, dodaliśmy również możliwość przewijania utworów. Poruszanie się pomiędzy odpowiednimi utworami umożliwiają przyciski „<-” (do tyłu) oraz „->” (do przodu). Aby nie trzeba było posługiwać się wbudowaną w Windows'a regulacją głośności, wprowadziliśmy własną regulację. Dodatkowo zaimplementowaliśmy funkcje takie jak ciągłe odtwarzanie utworów oraz losowe wybieranie utworu. Długo zastanawialiśmy się również nad metodą wczytywania utworów, jednak postawiliśmy na wczytywanie „hurtowe”, to znaczy, że wczytywać można całe foldery z muzyką. Program obsługuje następujące rozszerzenia plików muzycznych: *.mp3, *.wav, *.acc, *.wma, *.afi, *.mp4 (tylko dźwięk). Niestety, w przeciwieństwie do założeń, nie udało nam się zaimplementować equalizera. Testy przeprowadzaliśmy na bieżąco w sposób manualny, testując przy tym każdą z wprowadzonych funkcjonalności.

Przebieg współpracy:
Realizacja projektu przebiegała bardzo sprawnie. Dzięki systemowi kontroli wersji (GitHub) mogliśmy bez najmniejszego problemu wymieniać się . Kontaktowaliśmy się poprzez FaceBook'a oraz Skype żeby wspólnie ustalać zasadę działania programu, jego funkcjonalności oraz aby wspólnie rozdzielać zadania między sobą, w taki sposób, żeby praca jednego członka grupy nie utrudniała pracy drugiemu programiście. Dzięki luźnej atmosferze, wcześniejszej wspólnej współpracy i wzajemnej motywacji udało nam się osiągnąć zadowalające nas wyniki i stworzyć aplikacje zgodnie z większością naszych założeń.

Napotkane trudności:
W trakcje realizowania projektu napotkaliśmy wiele trudności, od czystej niechęci do tworzenia, poprzez lenistwo, aż do problemów z implementacją, ale to te ostatnie okazały się najtrudniejsze do rozwiązania. Jednym z większych problemów, jakie napotkaliśmy, było rozgraniczenie dlaczego odtwarzanie utworu ma zostać zatrzymane (użytkownik zatrzymuje, przełącza na kolejny utwór lub utwór się kończy i program powinien przejść do następnego). Ostatecznie poradziliśmy sobie z tym poprzez wprowadzenie zmiennej, która przetrzymuje informacje o powodzie zatrzymania pliku. Kolejną większą trudnością było zaimplementowanie możliwości przewijania utworu.

Działanie programu:
Interfejs użytkownika w istocie jest bardzo intuicyjny, wszystkie przyciski są podpisane. Po naciśnięciu przycisku „LOAD”(1) mamy możliwość wskazania folderu, z którego mają być odtwarzane utwory. Kolejno, po prawej znajdują się dwa przyciski, „S”(2) – odtwarzanie w losowej kolejności, „C”(3) – odtwarzanie ciągłe. Poniżej znajduje się lista plików(4), a pod nią suwak odpowiadający za przewijanie utworu(5) oraz drugi suwak(6) pozwalający regulować głośność. Na samym dole interfejsu znajdziemy przyciski: „Info”(7) – informacje o pliku, „<-”(8) – poprzedni utwór, „Stop”(9) – zatrzymanie odtwarzania, „Play/Pause”(10) – odtwarzanie i pauzowanie utworu, „->”(11) – następny utwór.



Opis kodu:

Poniżej znajduje się zrzut ekranu fragmentu programu. Ilustruje on działanie metody obsługującej naciśnięcie przycisku „Play/pause”. Jeśli na liście jest zaznaczony plik i odtwarzacz ma odtwarzać, program pobiera ścieżkę do pliku, zaczyna odtwarzanie zaznaczonego pliku oraz ustawia zmienne informacyjne na nowe.

```
private void playBtn_Click(object sender, EventArgs e)
{
    if (playList.SelectedIndex >= 0)
    {
        audioPlayer.playing = !audioPlayer.playing;
        if (audioPlayer.playing)
        {
            String filePath = filePaths[playList.SelectedIndex];
            currentIndex = playList.SelectedIndex;
            if(audioPlayer.playSound(filePath))
            {
                String name = filePath;
                if (audioPlayer.audioFile.FileName.Equals(name))
                {
                    setLabel(getFilePath(name, 2));
                    loadMetaPicture(filePath);
                }
            }
        }
        else
        {
            playBtn.Text = "Play";
            audioPlayer.outputDevice.Pause();
        }
    }
}
```

Poniższa metoda odpowiada za odtwarzanie utworu. Jako argument przyjmuje ścieżkę do pliku. Jeśli wszystkie dane są poprawne, inicjuje AudioFileReader z biblioteki NAudio, a potem poprzez klauzulę try/catch odtwarza utwór.

```
public bool playSound(String filePath)
{
    if (playing)
    {
        if (outputDevice == null)
        {
            outputDevice = new WaveOutEvent();
            outputDevice.PlaybackStopped += OnPlaybackStopped;
        }
        if (audioFile == null)
        {
            try
            {
                audioFile = new AudioFileReader(filePath);
                outputDevice.Init(audioFile);
            }
            catch (FormatException)
            {
                MessageBox.Show("Format exception, file not supported!");
                return false;
            }
        }
        try
        {
            outputDevice.Play();
            outputDevice.Volume = volume / 20f;
            //Debug.Print("VOLUME: " + (volume / 20f).ToString());
            PlaybackStopType = PlaybackStopTypes.PlaybackStoppedReachingEndOfFile;
            return true;
        }
        catch (InvalidOperationException ex)
        {
            Debug.Print("playSound() InvalidOperationException : " + ex.ToString());
            return false;
        }
    }
    return false;
}
```

Poniższy zrzut ekranu prezentuje metodę, która odpowiada za zastopowanie odtwarzania utworu. Jeśli program odtwarza aktualnie utwór to zmienia się wartość zmiennej „playing”, a odtwarzanie zostaje przerwane.

```
public bool stopSound()
{
    PlaybackStopType = PlaybackStopTypes.PlaybackStoppedByUser;
    if (playing)
    {
        outputDevice?.Dispose();
        outputDevice = null;
        try
        {
            audioFile?.Dispose();
        }
        catch (Exception ex)
        {
            Debug.Print("exception: " + ex.ToString());
        }
        audioFile = null;
        playing = false;
        return true;
    }
    else
    {
        outputDevice?.Stop();
        audioFile?.Dispose();
        audioFile = null;
        return false;
    }
}
```