**Automatyczne komponowanie muzyki**

**Filip Czajkowski**

**Nr albumu 131481**

**Katedra Architektury Systemów Komputerowych**

**Praca dyplomowa z kierunku Informatyka, studia stacjonarne II stopnia**

# Streszczenie:

Poniższy dokument opisuje powstały projekt dyplomowy magisterski, którego celem było stworzenie aplikacji komponującej muzykę w dopasowaniu do zadanej linii melodycznej. W tym celu powstał *MusicAnalyzer*, program układający zapis nutowy akordów dla fortepianu, który może służyć jako akompaniament przy wykonywaniu utworu podanego na wejściu. Dokumentacja wykonanego projektu opisuje interfejs aplikacji, logikę działania programu, wykonane testy uzyskanego rozwiązania oraz ocenę jakości wskazaną przez osoby testujące.

# Abstract:

This document describes a graduation project, which aim was to create an application capable of composing an accompaniment to a melody read from input file. Thus *MusicAnalyzer* arose, a program creating music notation of chords for a piano, which can be played as an accompaniment to a melody from input file. The documentation of a given project describes user interface of a desktop application, algorithm composing the music, tests of few final result music compositions and their feedback given by people assessing program’s efficiency.

# Spis treści

[Streszczenie: 1](#_Toc459763139)

[Abstract: 1](#_Toc459763140)

[Spis treści 2](#_Toc459763141)

[1. Wykaz stosowanych pojęć i skrótów 4](#_Toc459763142)

[2. Wstęp i cel pracy 5](#_Toc459763143)

[2.1. Wstęp 5](#_Toc459763144)

[2.2. Cel pracy 5](#_Toc459763145)

[3. Opis dziedziny 6](#_Toc459763146)

[3.1. Opis tematu projektu dyplomowego 6](#_Toc459763147)

[3.1. Przegląd dostępnych rozwiązań 7](#_Toc459763148)

[3.1.1. Emily Howell 7](#_Toc459763149)

[3.1.2. AthTek DigiBand 8](#_Toc459763150)

[3.1.3. Ludwig 8](#_Toc459763151)

[3.2. Podstawowe pojęcia muzyczne i funkcje harmoniczne 8](#_Toc459763152)

[4. Opis rozwiązania 9](#_Toc459763153)

[4.1. Opis interfejsu graficznego programu MusicAnalyzer 9](#_Toc459763154)

[4.2. Informacje bazowe – pliki konfiguracyjne 9](#_Toc459763155)

[4.3. Wczytywanie melodii wejściowej i metody określające jej cechy 9](#_Toc459763156)

[4.4. Wyznaczanie tonacji i metrum w utworze 9](#_Toc459763157)

[4.5. Prezentacja graficzna zapisu nutowego 9](#_Toc459763158)

[4.6. Algorytm tworzenia akompaniamentu 9](#_Toc459763159)

[4.7. Zapis nutowy i dźwiękowy powstałego akompaniamentu 9](#_Toc459763160)

[4.8. Stosowane rozwiązania pomocnicze 9](#_Toc459763161)

[4.8.1. Sanford.Multimedia 9](#_Toc459763162)

[4.8.2. PSAMControlLibrary 10](#_Toc459763163)

[4.8.3. PSAMWPFControlLibrary 11](#_Toc459763164)

[5. Ocena 12](#_Toc459763165)

[5.1. Założenia testów 12](#_Toc459763166)

[5.2. Wyniki testów 12](#_Toc459763167)

[5.3. Kryteria oceny 12](#_Toc459763168)

[5.4. Ocena jakości określona przez osoby testujące 12](#_Toc459763169)

[6. Podsumowanie 13](#_Toc459763170)

[7. Wykaz literatury pomocniczej 14](#_Toc459763171)

[8. Wykaz rysunków 15](#_Toc459763172)

[9. Wykaz tabel 16](#_Toc459763173)

# Wykaz stosowanych pojęć i skrótów

1. *Test Turinga* - sposób mający dowodzić opanowania przez maszynę umiejętności myślenia w sposób podobny do ludzkiego. Zaproponowany w 1950 roku przez Alan Turinga polega na prowadzeniu rozmowy z człowiekiem i konkurującą maszyną przez osobę mającą tu rolę sędziego. Jeśli nie jest on w stanie prawidłowo wskazać, które z nich jest maszyną, to przyjmuje się, że wówczas zdaje ona test.

# Wstęp i cel pracy

## Wstęp

*MusicAnalyzer* to program tworzący własną muzykę przy wykorzystaniu metod sztucznej inteligencji w oparciu o zasady klasycznej harmonii muzycznej.

## Cel pracy

# Opis dziedziny

## Opis tematu projektu dyplomowego

Tematem projektu dyplomowego było automatyczne komponowanie muzyki. Jak nietrudno się domyślić, jest to problem nieposiadający deterministycznego rozwiązania, które jesteśmy także w łatwy sposób ocenić, porównać z innym. Muzyka, podobnie jak inne dziedziny sztuki, dostarczają wrażeń (w tym przypadku dźwiękowych), które nie są jednoznacznie odczytywane przez wszystkich odbiorców. Dodatkowo, ocena tego samego utworu muzycznego w zależności od gatunku muzyki, w ramach którego go odbieramy, może się znacząco różnić. Problemy z oceną jakości kompozycji przekładają się także na trudności z wytworzeniem ich, w szczególności gdy odpowiada za ten proces bezduszny program komputerowy.

Świat nauk ścisłych wielokrotnie w przeszłości stawiał sobie za cel zrozumienie i określenie stałymi regułami fenomenu osiągnięć nauk humanistycznych. Czy można w deterministyczny sposób opisać proces tworzenia przez poetę wiersza albo obrazu rysowanego przez malarza? Czy metody sztucznej inteligencji są w stanie naśladować, wydaję się, cudownie twórczy proces zachodzący w mózgu artysty? Na te pytanie naukowcy starają się odpowiadać twierdząco i popierają to coraz doskonalszymi osiągnięciami w tej dziedzinie, lecz na dzień dzisiejszy nie wydaje się, by los artystów został już definitywnie przesądzony.

Podobne wyzwania stają przed naukowcami pracującymi nad automatycznym komponowaniem muzyki. Problem ten można podzielić na dwa rodzaje wyzwań.

Pierwsze podejście zakłada zupełne zastąpienie pracy kompozytora i umożliwienie tworzenia kompletnej muzyki od podstaw przez komputer. <<coś o tym konkursie>>

Drugim rodzajem zmagań z tworzeniem muzyki w sposób sztuczny jest układanie akompaniamentu do istniejącej już melodii, lecz w taki sposób, by obie ze sobą dobrze współbrzmiały. Problem ten wydaje się na pierwszy rzut oka nieco prostszy, gdyż posiadamy jakiś punkt zaczepienia, wzór, do którego staramy się dopasować. Jednakże osiągnięcie sukcesu w tym zadaniu niesie ze sobą podobne problemy, co tworzenie muzyki bez wzorca. Ponadto, to podeście niesie ze sobą szereg założeń, które należy przyjąć, aby móc później ocenić jakość stworzonego dopasowania. Czy obie melodie mają być harmonicznie dopasowane? Jakiego typu harmonia ma je wiązać? Czy bardzo podobne do siebie melodie są pożądane, czy raczej oceniane jako nudne, nieciekawe? Czy dysonanse będą odbierane jako nieudolność programu czy właśnie twórczy dodatek? Na te i wiele innych pytań, twórca oprogramowania tworzącego akompaniament musi wcześniej sobie odpowiedzieć a także zgodnie z nimi powinien być oceniany.

## Przegląd dostępnych rozwiązań

## Emily Howell

*Emily Howell* to nazwa programu komputerowego autorstwa Davida Cope’a powstałego w latach 90-tych na uczelni *University of California* w Santa Cruz. Projekt jest kontynuacją pracy uczonego nad programem *EMI* (Experiments in Musical Intelligence). Zasilany był bazą utworów wybranego kompozytora a następnie tworzył nową kompozycję, która stylistyką nawiązywała do utworów z zestawu wejściowego. *Emily Howell* funkcjonuje w analogiczny sposób, natomiast jako dane wejściowe posiada jedynie utwory wytworzone przez *EMI*, dzięki czemu, przynajmniej według jego twórcy, wykształciła własny styl kompozytorski.

Szczegóły algorytmu zastosowanego w programie *EMI* nie są znane, natomiast wiadomo, iż jest on oparty na kilku krokach:

1. Utworzenie bazy źródłowej kompozycji, których styl chcemy naśladować
2. Znalezienie wzorców (motywów dźwiękowych i rytmicznych)
3. Podział dzieła na segmenty zgodnie ze znalezionym rytmem
4. Analiza hierarchiczna – wszystkim współbrzmieniom przypisywane są ich funkcje, jakie pełnią w danej frazie; pomaga to określić np. które akordy po sobie występują;
5. Nieliniowe rekombinacje i modyfikacje powstających fragmentów z posiadanych wzorców

Dodatkowo program jest wciąż ulepszany, gdyż powstające utwory są oceniane przez dr Cope’a i wyniki jego szczegółowej oceny wpływają na proces tworzenia kolejnych wyników.

Rozwiązania stosowane w obu projektach okazały się na tyle wydajne, iż *David Cope* posiadł patent na terenie Stanów Zjednoczonych chroniący jego autorski algorytm komponowania muzyki. Ponadto program *EMI* w 1997 roku na Uniwersytecie w Oregonie zmierzył się w teście przypominającym muzyczny odpowiednik *Testu Turinga.* Na podstawie posiadanej bazy utworów *J. S. Bacha* stworzył własną kompozycję (miniaturę fortepianową) imitującą styl geniusza epoki baroku. Podobne zadanie otrzymał szanowany, współczesnykompozytor *Steve Larson.* Następnie pianistka wykonała przed oceniającą publicznością powyższe utwory oraz oryginalny utwór *Bacha*. Zebrani mieli za zadanie ocenić, która z kompozycji jest oryginałem, która stworzona przez współczesnego kompozytora, a która przez program *EMI*. Ich werdykt był bardzo zaskakujący, gdyż to komputerowo wytworzony utwór został uznany za oryginalne dzieło z XVII wieku, za to utwór *Larsona* wzięto za wytwór sztucznej inteligencji, natomiast pierwotna kompozycja *Bacha* została odebrana jako imitacja autorstwa współczesnego kompozytora.

Wyniki przedstawionego powyżej konkursu nie dają nam pewności co do pozytywnej oceny walorów artystycznych kompozycji programu *EMI*, gdyż publiczność w swoim głosowaniu źle oceniła utwór samego *Bacha*. Można jednak sądzić, że efekty obliczeń *Emily Howell* można przyjąć za dobrej jakości muzykę, ponieważ doczekały się one dwóch albumów na płytach: „*From Darkness, Light”* w 2009 i „*Breathless*” w 2012 roku. Niemniej jednak, sztuczne tworzenie muzyki jest wciąż źle odbierane przez środowisko artystyczne, przez co wielu muzyków odmawia wykonywania utworów autorstwa *Emily*.

## AthTek DigiBand

## Ludwig

## Podstawowe pojęcia muzyczne i funkcje harmoniczne

# Opis rozwiązania

## Opis interfejsu graficznego programu MusicAnalyzer

## Informacje bazowe – pliki konfiguracyjne

## Wczytywanie melodii wejściowej i metody określające jej cechy

## Wyznaczanie tonacji i metrum w utworze

## Prezentacja graficzna zapisu nutowego

## Algorytm tworzenia akompaniamentu

## Zapis nutowy i dźwiękowy powstałego akompaniamentu

## Stosowane rozwiązania pomocnicze

W aplikacji *MusicAnalyzer* stosowane były rozwiązania udostępniane przez ogólnodostępne biblioteki działające zgodnie z licencją *Open Source*, dzięki czemu mogą być używane do celów niekomercyjnych. Dołączenie ich referencji jest konieczne do skompilowania projektu.

## Sanford.Multimedia

*Sanford*.*Multimedia* to biblioteka dla platformy .Net udostępniająca klasy i metody ułatwiające odczytywanie i analizę plików dźwiękowych w formacie MIDI. Projekt autorstwa Leslie Sanford umożliwia wczytanie pliku wejściowego, odkodowanie zawartych w nim sygnałów oraz odtworzenie zapisanej w nim muzyki. Poniżej przedstawione zostały wybrane klasy, które były używane w projekcie MusicAnalyzer i dostarczały funkcjonalności potrzebnych do operacji na pliku MIDI:

* **MidiEvent** – odzwierciedla zdarzenia (sygnały) zachodzące w określonych momentach czasu, które zawierają różnorakie informacje związane z daną chwilą; przykładem takich zdarzeń są przykładowo: zmiana tonacji, zmiana tempa, początek trwania określonego dźwięku lub jego koniec;
* **Track** – odpowiada pojedynczej ścieżce dźwiękowej zapisanej w pliku; zawiera w sobie kolekcję obiektów klasy **MidiEvent**, czyli zdarzeń zaistniałych na danej ścieżce;
* **Sequence** – odpowiada całości zawartości pliku MIDI i posiada kolekcję obiektów typu **Track**, przez co umożliwia zapisywanie, odczytywanie i odtwarzanie pliku muzycznego w całości;
* **ChannelMessage** – wiadomość zawarta w zdarzeniu **MidiEvent**, która jest przypisana do konkretnego kanału dźwiękowego (stereo); są to głównie wiadomości o rozpoczęciu (*NoteOn*) lub zakończeniu (*NoteOff*) dźwięku o wskazanej wysokości i głośności;
* **MetaMessage** – wiadomości pozostałych typów, które nie są bezpośrednio związane z danym kanałem, lecz ze ścieżką dźwiękową (np. zmiana tonacji)
* **OutputDevice** – klasa opisująca urządzenie umożliwiające wysyłanie sygnałów MIDI, aby móc odsłuchać utwór muzyczny
* **Sequencer** – umożliwia odtwarzanie sygnałów muzycznych zapisanych w obiekcie **Sequence**; tworzy kolejkę sygnałów i wysyła je poprzez **OutputDevice**

## PSAMControlLibrary

Jest to biblioteka funkcji umożliwiających prezentację graficzną zapisu nutowego w technologii *Windows Forms* na platformie *.Net*. Projekt autorstwa Jacka Salamona posiada też bardziej rozbudowaną wersję komercyjną, lecz używana w tym projekcie biblioteka jest dostępna w ramach licencji *BSD*. Udostępnia ona obiekty odwzorowujące elementy zapisu melodii na pięciolinii oraz umożliwia wygenerowanie widoku zapisu nutowego w formatce w technologii Windows Forms. Kilka używanych obiektów oraz typów wyliczeniowych:

* **Clef** – klucz na pięciolinii (basowy lub wiolinowy)
* **Key** – zestaw znaków przy kluczu odpowiadający tonacji utworu (krzyżyki lub bemole)
* **TimeSignature** – oznaczenie metrum (np. ¾)
* **Note** – nuta na pięciolinii o odpowiedniej wysokości i wartości rytmicznej
* **MusicSymbolDuration** – wartość rytmiczna nuty (np. ósemka, ćwierćnuta)
* **BarLine** – kreska taktowa
* **Rest** – pauza odpowiedniej długości (np. pauza ósemkowa, ćwierćnutowa)

## PSAMWPFControlLibrary

Opisywana powyżej biblioteka została stworzona z myślą o interfejsie graficznym w technologii Windows Forms. Jako że *MusicAnalyzer* tworzy okno aplikacji dzięki *Windows Presentation Foundation*, konieczne było wykorzystanie biblioteki *PSAMWPFControlLibrary*, która jest przystosowana do budowy interfejsu w tej technologii i daje też dużo szersze możliwości graficzne. Dzięki dobrej kompatybilności *Windows Forms* i *WPF*, można było użyć elementów muzycznych z pierwotnej biblioteki, natomiast konieczne było jedynie osadzenie ich w elemencie typu *IncipitViewerWPF*, który gromadził na pięciolinii dodawane kolejno do niego elementy.

Aby wymienione powyżej znaki pokazywały się prawidłowo, należy w systemie operacyjnym zainstalować czcionkę *Polihymnia* dołączonej do projektu i dystrybuowanej zgodnie z licencją *Sil Open Font Licence*.

# Ocena

## Założenia testów

## Wyniki testów

## Kryteria oceny

## Ocena jakości określona przez osoby testujące

# Podsumowanie

# Wykaz literatury pomocniczej

– użyć bibteX szukać przez google scholars

BibteX 2 Word

# Wykaz rysunków

# Wykaz tabel