**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](#_Toc461650010)

[Abstract: 1](#_Toc461650011)

[Spis treści 2](#_Toc461650012)

[1. Wykaz stosowanych pojęć i skrótów 4](#_Toc461650013)

[2. Wstęp i cel pracy 5](#_Toc461650014)

[2.1. Wstęp 5](#_Toc461650015)

[2.2. Cel pracy 5](#_Toc461650016)

[3. Opis dziedziny 7](#_Toc461650017)

[3.1. Opis tematu projektu dyplomowego 7](#_Toc461650018)

[3.2. Podstawowe pojęcia muzyczne i funkcje harmoniczne 7](#_Toc461650019)

[3.3. Przegląd dostępnych rozwiązań 16](#_Toc461650020)

[3.3.1. Emily Howell 16](#_Toc461650021)

[3.3.2. Mezzo 18](#_Toc461650022)

[3.3.3. AthTek DigiBand 20](#_Toc461650023)

[4. Opis rozwiązania 22](#_Toc461650024)

[4.1. Opis interfejsu graficznego programu MusicAnalyzer 22](#_Toc461650025)

[4.2. Informacje bazowe – pliki konfiguracyjne 22](#_Toc461650026)

[4.3. Wczytywanie melodii wejściowej i metody określające jej cechy 23](#_Toc461650027)

[4.3.1. Zdekodowanie sygnałów dźwięków 23](#_Toc461650028)

[4.3.2. Wyszukanie zmian tonacji utworu 24](#_Toc461650029)

[4.3.3. Wyszukanie zmian metrum 24](#_Toc461650030)

[4.3.4. Ustalenie prawidłowej tonacji utworu 24](#_Toc461650031)

[4.3.5. Ustalenie prawidłowych oznaczeń dźwięków 26](#_Toc461650032)

[4.3.6. Ustalenie wartości rytmicznych wszystkich nut 26](#_Toc461650033)

[4.4. Prezentacja graficzna zapisu nutowego 28](#_Toc461650034)

[4.5. Algorytm tworzenia akompaniamentu 31](#_Toc461650035)

[4.5.1. Harmony Search 31](#_Toc461650036)

[4.5.2. Wyszukanie współbrzmień wejściowych 31](#_Toc461650037)

[4.5.3. Wybór węzłów dla akordów 31](#_Toc461650038)

[4.5.4. Adaptacja algorytmu HS 31](#_Toc461650039)

[4.5.5. Ustalenie atrybutów nut akompaniamentu 32](#_Toc461650040)

[4.6. Zapis nutowy i dźwiękowy powstałego akompaniamentu 32](#_Toc461650041)

[4.7. Stosowane rozwiązania pomocnicze 34](#_Toc461650042)

[4.7.1. Sanford.Multimedia 34](#_Toc461650043)

[4.7.2. PSAMControlLibrary 35](#_Toc461650044)

[4.7.3. PSAMWPFControlLibrary 35](#_Toc461650045)

[5. Ocena 37](#_Toc461650046)

[5.1. Założenia testów 37](#_Toc461650047)

[5.2. Kryteria oceny 38](#_Toc461650048)

[5.3. Wyniki testów 38](#_Toc461650049)

[5.4. Ocena jakości działania programu MusicAnalyzer 39](#_Toc461650050)

[6. Podsumowanie 40](#_Toc461650051)

[7. Wykaz rysunków 41](#_Toc461650052)

[8. Wykaz tabel 42](#_Toc461650053)

[9. Wykaz literatury pomocniczej 43](#_Toc461650054)

[Odwołania 43](#_Toc461650055)

# 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.
2. *Progresja harmoniczna* – szereg współbrzmień następujących po sobie, które tworzą dobrze zharmonizowany układ i wrażenie spójnego motywu muzycznego u odbiorcy.
3. *MIDI* (ang. Musical Instrument Digital Interface) – kompletny system służący do przekazywania informacji dźwiękowych pomiędzy elektronicznymi instrumentami muzycznymi. Pliki zapisane w tym standardzie mogą być odtwarzane na urządzeniach dzięki stałym próbkom dźwięku różnych instrumentów w nich zapisanych, sam plik zaś nie zawiera nagrania dźwięku w żadnej formie. Posiada za to kolekcję sygnałów, które odzwierciedlają różne muzyczne działania, takie jak włączenie / wyłączenie dźwięku o zadanej wysokości, naciśnięcie pedału w pianinie czy zmiana tempa. W znajdującej się na początku sekcji nagłówkowej zawiera się także zestaw atrybutów pliku potrzebnych do jego prawidłowego odczytania i odtworzenia.
4. *Tessitura* – w odniesieniu do głosów instrumentalnych lub wokalnych, przedział dźwięków najczęściej używany w danej partii.
5. *Bitmapa* – obraz w grafice rastrowej, w którym określona jest jednoznacznie zawartość każdego piksela; umożliwia to szybkie i nieskomplikowane obliczeniowo wyświetlanie obrazu, lecz nie pozwala na zwiększenie dokładności w wyniku przybliżenia.
6. *Strój (muzyczny)* – system uporządkowania dźwięków w ramach jednej oktawy; obecnie powszechnie stosowany jest strój równomiernie temperowany, który dzieli oktawę na 12 równych części. Dodatkowo strój muzyczny wyznacza częstotliwość dźwięków wydawanych przez instrumenty muzyczne. W okresie baroku, dźwięk A4 miał częstotliwość 415Hz, następnie zastąpiono go strojem 432Hz, a od okresu międzywojnia XX w. najczęściej używany jest strój 440Hz (stosowany w urządzeniach elektronicznych zgodnych ze standardem *MIDI*).

# Wstęp i cel pracy

## Wstęp

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 definitywnie przesądzony.

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.

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. 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 wymaga szeregu 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 raczej jako twórczy dodatek? Na te i wiele innych pytań, twórca oprogramowania tworzącego akompaniament musi sobie wcześniej odpowiedzieć a także zgodnie z nimi powinien być oceniany. Problemy z oceną jakości kompozycji przekładają się także na trudności z wytworzeniem ich, w szczególności gdy odpowiada za ten proces bezduszny program komputerowy.

## Cel pracy

Celem tego projektu było zmierzenie się z problemem automatycznego komponowania muzyki z wykorzystaniem technik sztucznej inteligencji bez oparcia na posiadanej głębokiej wiedzy dziedzinowej. Jedyną muzyką znaną przez program komputerowy jest aktualnie wczytany utwór z pliku wejściowego, do którego powstać ma akompaniament. Nie posiada on zbioru kompozycji, by móc się na nich wzorować, gdyż stawiane przed nim wyzwanie, to utworzenie pasującej melodii dzięki metodom sztucznej inteligencji. Tak powstał *MusicAnalyzer,* program tworzący własną muzykę przy wykorzystaniu algorytmu *Harmony Search* w oparciu o zasady klasycznej harmonii muzycznej. Do działań w ramach pracy dyplomowej należy także dodać ocenę efektywności jego działań przy różnej parametryzacji algorytmu.

# Opis dziedziny

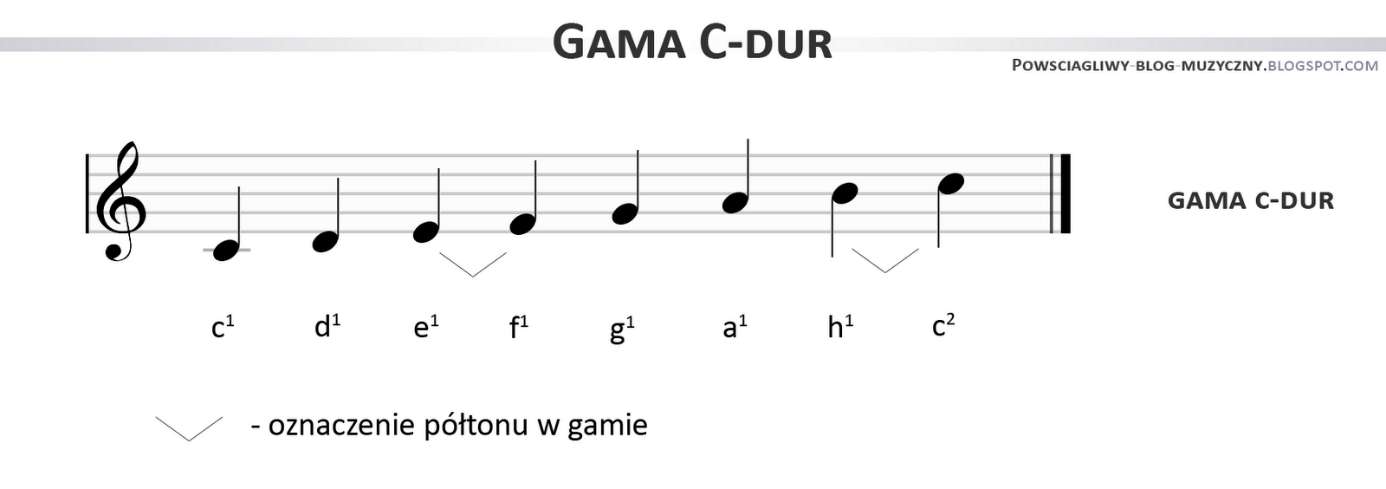
## Opis tematu projektu dyplomowego

Tematem projektu dyplomowego jest automatyczne komponowanie muzyki w postaci tworzenia akompaniamentu do istniejącej melodii. Jak nietrudno się domyślić, jest to problem nieposiadający deterministycznego rozwiązania, które jesteśmy także w łatwy sposób ocenić, porównać z innym. Z tego względu, głównym obiektem pracy nad tym projektem była implementacja algorytmu z dziedziny sztucznej inteligencji (*Harmony Search*), który jest względnie nowym pomysłem i nie posiada wciąż wielu zastosowań. Wybrane podejście do automatycznego generowania muzyki wiąże się z kilkoma przyjętymi założeniami. W odróżnieniu od większości tego typu programów, *MusicAnalyzer* nie posiada bazy kompozycji stworzonych ludzką ręką, by na ich podstawie tworzyć własne utwory korzystając z pewnych modyfikacji i kombinacji. Jego główną siłą jest implementacja algorytmu harmonicznego, który ma za zadanie utworzyć nową ścieżkę dźwiękową będącą w harmonii z linią melodyczną wczytaną w pliku wejściowym. Szukanie prawidłowych współbrzmień a następnie ich ocena zostały oparte na zasadach klasycznej harmonii, którymi posługują się kompozytorzy od wieków. Efektem działań algorytmu jest sekwencja zmieniających się w kluczowych momentach utworu akordów. Element improwizacji i tworzenie wyszukanej wiodącej melodii celowo nie były zgłębiane, gdyż prace skupione były na szukaniu prawidłowego tła dla melodii już istniejącej. Program przystosowany jest do wczytania pliku z zapisem wielu ścieżek dźwiękowych jednocześnie i wszystkie z nich są jednakowo brane pod uwagę w szukaniu współbrzmienia z powstającym akompaniamentem.

## Podstawowe pojęcia muzyczne i funkcje harmoniczne

W opisywanym projekcie konieczne było zamodelowanie świata muzycznego, funkcji, figur oraz reguł współbrzmień, aby z nich zbudować własną kompozycję. Tworząc klasy odpowiadające pojęciom muzycznym, utworzono także reguły, które nimi żądzą w nomenklaturze stosowanej przez muzyków. Dzięki temu zabiegowi, proces kompozycji w programie *MusicAnalyzer* jest opisany pojęciami analogicznymi do tych używanych przez rzeczywistych kompozytorów. Na samym początku należy zatem przedstawić terminy ze świata teorii muzyki, które będą używane w dalszej części dokumentu.

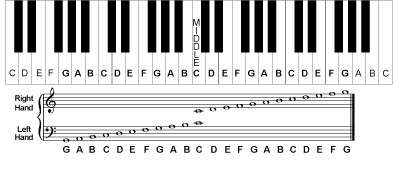
* Tonalność - system uporządkowania dźwięków, dotyczący szczególnie harmoniki i melodyki. Tonalność charakteryzuje występowanie określonego centrum tonalnego[[1]](#footnote-1) (głównego dźwięku) oraz zasad połączeń pomiędzy dźwiękami tworzącymi określone współbrzmienia.
* Nuta – dźwięk w zapisie muzycznym; jego podstawowymi atrybutami jest wysokość (w pojęciu fizycznym odpowiadająca częstotliwości dźwięku) oraz długość trwania
* Półton – najmniejsza odległość między dźwięki określona w teorii muzyki[[2]](#footnote-2). Dźwięk odległy o pół tonu od wskazanego jest większy bądź mniejszy o 1/12 częstotliwości dźwięku podstawowego (ta zależność nie dotyczy każdego stroju muzycznego, lecz ma zastosowanie w stroju równomiernie temperowanym, którym posługują się wszystkie urządzenia elektroniczne i jak strojone są obecnie instrumenty).
* Krzyżyk – oznaczenie podwyższenie nuty o pół tonu względem miejsca, w którym się znajduje na pięciolinii, krzyżyk dodany do nuty (za wyjątkiem dźwięków H i E) wskazuje dźwięki odpowiadające czarnym klawiszom na fortepianie.
* Bemol – ma działanie odwrotne niż krzyk, obniża wskazany dźwięk o pół tonu.
* Oktawa – odległość między dźwiękami odpowiadająca 12-stu półtonom, co daje dwukrotnie większą lub mniejszą częstotliwość pomiędzy przykładowymi dźwiękami[[3]](#footnote-3), należącymi zatem do tego samego szeregu harmonicznego. Ponieważ wszystkie dźwięki w ramach jednej oktawy mają swoich odpowiedników w pozostałych oktawach poprzez zwielokrotnienie bądź podzielenie ich częstotliwości, z punktu widzenia harmonii brzmień, są one tak samo ważne i można je stosować zamiennie w zależności od potrzeby, bądź instrumentu, który daną melodię wykonuje. W programie *MusicAnalyzer* kompozycja wpierw ustala konkretne dźwięki w ramach oktawy, a następnie dopiero dobiera oktawę, w ramach której mają być one dopasowany, aby możliwie najlepiej akompaniowały melodię podstawową.
* Oznaczenia nut w ramach oktawy – kolejne dźwięki w zapisie muzycznym posiadają swoje unikatowe oznaczenia. W notacji używanej w tym dokumencie, są one poukładane alfabetycznie za wyjątkiem dźwięku *H*, który występuje między *A* i *C* (w amerykańskim nazewnictwie można w jego miejscu spotkać *B*)[[4]](#footnote-4). Podstawowe dźwięki (odpowiadające białym klawiszom fortepianu) są oddalone względem siebie (różnica wysokości dźwięku) o odległość całego bądź połowy tonu. Układ ten zatem nie jest spójny, ale powtarzalny w przypadku niższych i wyższych oktaw. Zestaw dźwięków i wzajemnych odległości przedstawia poniższy rysunek.



Rysunek 1 Dźwięki gamy C-dur, źródło http://3.bp.blogspot.com/-ZbOavfTrYJg/T1QALqWFwdI/AAAAAAAAADg/eC5xMCH9c6g/s1600/gama-c-dur-odleglosci.png

Dodatkowo, aby oznaczyć dźwięki znajdujące się pomiędzy tymi odległymi o cały ton (odpowiadające czarnym klawiszom fortepianu), stosuje się krzyżyki i bemole[[5]](#footnote-5), gdyż na pięciolinii zapisane są na tej samej wysokości. I tak nazewnictwo prezentuje się następująco: dla przykładu, dźwięk C z krzyżykiem to Cis, natomiast D z bemolem to Des.

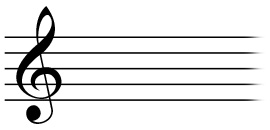
* Absolutne oznaczenie wysokości dźwięku – do opisanego powyżej schematu nazywania dźwięków w ramach danej oktawy, należy dodać jeszcze jej numer, do której z nich należy. Człowiek posiada ograniczoną umiejętność słyszenia dźwięków o bardzo niskich bądź wysokich częstotliwościach[[6]](#footnote-6). Stąd też zakres oznaczeń nie jest zbyt duży i dostosowany jest do możliwości ludzkiego ucha. Najniższa słyszalna oktawa posiada numer 0 (dla przykładu dźwięk C0 ma częstotliwość 16,35 Hz[[7]](#footnote-7)), natomiast najwyższe częstotliwości słyszane przez człowiek to oktawa 10 (C10 ≈ 16 700 Hz[[8]](#footnote-8)).
* Pięciolinia – układ pięciu linii służący do określenia częstotliwości zapisanych na niej dźwięków (dźwięk pokazany wyżej ma wyższą częstotliwość, a niżej – mniejszą). Z tego powodu wywiązał się zwyczaj określania nut jako wysokie lub niskie. Mogą być zapisywane na liniach lub pomiędzy nimi i odpowiadają dźwiękom wydobywanym przez białe klawisze na fortepianie[[9]](#footnote-9).



Rysunek 2 Układ klawiszy fortepianu i odpowiadających im dźwięków na pięcioliniach, źródło: https://tucsonsongstress.files.wordpress.com/2012/11/staffnkeys.gif

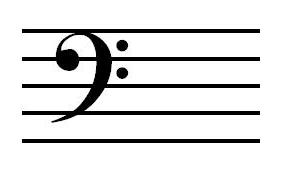
Następstwo dźwięków po sobie obrazuje się poprzez umieszczenie ich obok siebie (od lewej do prawej). Tak też nuty, które mają zabrzmieć jednocześnie, zaznacza się jedna nad drugą.

* Klucz wiolinowy – znak stawiany na początku pięciolinii określający wysokość znajdujących się na niej dźwięków[[10]](#footnote-10). Klucz wiolinowy znajduje się na drugiej linii i wyznacza na niej dźwięk G4 (392 Hz[[11]](#footnote-11)).



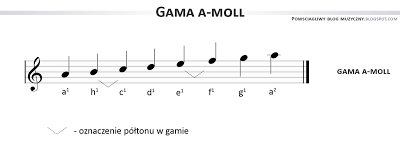
Rysunek 3 Klucz wiolinowy, źródło http://www.music-paper.com/ImagesClefs/treble-clef.jpg

* Klucz basowy – analogicznie jak wiolinowy, określa wysokość dźwięków na pięciolinii. Służy do zapisu niższych dźwięków, umieszczony na trzeciej linii wskazuje nutę F3 (ok. 175 Hz[[12]](#footnote-12))



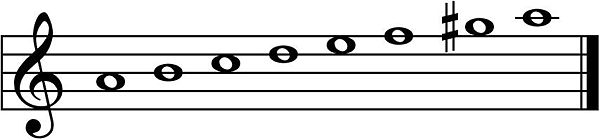
Rysunek 4 Klucz basowy, źródło: http://www.theyrenotourgoats.com/wp-content/uploads/2014/10/Bass\_Clef.jpg

* Akord – jednoczesne współbrzmienie kilku dźwięków (przynajmniej trzech) o różnej wysokości.
* Konsonans – współbrzmienie przynajmniej dwóch dźwięków, które jest odbierane słuchowo jako zgodne. Ma ono swoje uzasadnienie czysto fizyczne, gdyż nasz słuch pozytywnie odbiera dźwięki, których częstotliwości nakładają się na siebie, mają względnie niewielką wspólną wielokrotność[[13]](#footnote-13). Gdy wybrzmiewają jednocześnie, fale akustyczne na siebie nachodzą, wzmacniają się i zachodzi zjawisko dudnienia.
* Dysonans – jest przeciwieństwem konsonansu, a zatem współbrzmieniem niedoskonałym, które sprawia wrażenie niezgodności. W muzyce współczesnej panuje coraz większe przyzwolenie na stosowanie takich współbrzmień, co jednak bywa kontrowersyjne i odrzucane przez część twórców oraz odbiorców[[14]](#footnote-14). Daje to jednak szersze pole działania kompozytorom, gdyż umożliwia tworzenie nieznanych dotąd melodii. W projekcie *MusicAnalyzer* preferowane są jednakowoż klasyczne współbrzmienia harmoniczne, które nie zawierają dysonansów.
* Interwał – określenie odległości między dwoma dźwiękami stosowane w muzykologii. Nazwy interwałów pochodzą z języka łacińskiego[[15]](#footnote-15) i można je podzielić na proste[[16]](#footnote-16) (ich rozpiętość mieści się w obrębie oktawy) i złożone (pozostałe, większe). Do interwałów prostych należą:
  + pryma – odległość zawarta między powtórzonymi stopniami skali
  + sekunda mała – 1 półton
  + sekunda wielka – 2 półtony
  + tercja mała – 3 półtony
  + tercja wielka – 4 półtony
  + kwarta – 5 półtonów
  + tryton – 6 półtonów
  + kwinta – 7 półtonów
  + seksta mała – 8 półtonów
  + seksta wielka – 9 półtonów
  + septyma mała – 10 półtonów
  + septyma wielka – 11 półtonów
  + oktawa – 12 półtonów
* Skala durowa - to siedmiostopniowa skala z charakterystycznym półtonem między stopniami III i IV oraz VII i VIII (stanowiącym powtórzenie I stopnia o oktawę wyżej)[[17]](#footnote-17). Wszystkie dźwięki zawierają się w ramach jednej oktawy. Uznaje się, że skala durowa i utwory na niej oparte mają radosne brzmienie.
* Skala molowa eolska - to siedmiostopniowa skala z charakterystycznym półtonem między stopniami II i III[[18]](#footnote-18). Wszystkie dźwięki zawierają się w ramach jednej oktawy. Skale molowe i oparte na nich fragmenty utworów muzycznych mają smutne brzmienie.



Rysunek 5 Skala molowa eolska w gamie na przykładzie gamy a – moll, źródło: http://3.bp.blogspot.com/-c1\_q-799WgM/T1QOY7uNJGI/AAAAAAAACjI/hpUDBezDOs4/s400/gama-a-moll-odleglosci.png

* Skala molowa harmoniczna – inna skala molowa, charakteryzująca się względem eolskiej tym, iż posiada podwyższony VII stopień o pół tonu. W efekcie półtony występują między II i III stopniem oraz VII i VIII, a między VI i VII występuje interwał sekundy zwiększonej (3 półtony)[[19]](#footnote-19). Ta skala jest najbardziej popularna wśród kompozycji molowych, gdyż podwyższony VII stopień przyjmuje funkcję dźwięku wyraźnie prowadzącego do rozwiązania, akordu toniki.



Rysunek 6 Skala harmoniczna na przykładzie gamy a - moll, źródło: https://upload.wikimedia.org/wikipedia/commons/thumb/4/4b/Moll\_harm.jpg/600px-Moll\_harm.jpg

* Gama - to szereg dźwięków ułożonych według reguł danej skali muzycznej, zaczynający się od konkretnego dźwięku, który zazwyczaj nadaje nazwę tak utworzonej gamie[[20]](#footnote-20). Zawiera ona każdorazowo siedem dźwięków, które mogą należeć do różnych oktaw (zaczynać się na różnych wysokościach), natomiast ich wzajemne odległości są stałe i charakterystyczne dla danej skali.



Rysunek 7 Dźwięki należące do gamy C - dur zapisane w kluczu wiolinowym

* Funkcja harmoniczna - każda tonacja posiada charakterystyczne dla siebie współbrzmienia akordów, na których zbudowane są kompozycje w danej tonacji[[21]](#footnote-21). W tym rozumieniu, akordy posiadają swoją funkcję harmoniczną, gdyż w ramach danej tonacji budują lub rozładowują napięcie, sprawiają wrażenie stabilności lub chaosu w danej frazie. Te same akordy występujące w różnym kontekście są odbierane w inny sposób, stąd też przyporządkowane im funkcje harmoniczne różnią się w zależności od tonacji, w jakiej występują.
* Trójdźwięk durowy – składa się z trzech dźwięków, które oddalone są od siebie w następujący sposób[[22]](#footnote-22):
  + 1. i 2. – o tercję wielką
  + 2. i 3. – o tercję małą
  + 3. i 1. – o kwartę czystą

Wiodąca tercja wielka powoduje, iż współbrzmienie odbierane jest jako pozytywne (durowe).

* Trójdźwięk molowy – składa się z trzech dźwięków, które oddalone są od siebie w następujących interwałach[[23]](#footnote-23):
  + 1. i 2. – o tercję małą
  + 2. i 3. – o tercję wielką
  + 3. i 1. – o kwartę czystą

Tercja mała na początku akordu skutkuje brzmieniem odbieranym jako smutne (molowe).

* Tonika – podstawowy trójdźwięk każdej gamy zbudowany z dźwięków na I, III i V stopniu gamy. Jest ośrodkiem tonalnym, który rozładowuje napięcie zbudowane wewnątrz fraz[[24]](#footnote-24), stąd najczęściej znajduje się na ich końcu lub też na początku. W gamie molowej jest molowy, natomiast w durowej – durowy.
* Subdominanta – akord pełniący tę funkcję harmoniczną zbudowany jest z dźwięków na IV, VI i VIII stopniu gamy. We frazach muzycznych jest wykorzystywany jako akord przejściowy. Ze względu na swoją bliskość do akordu toniki, wykorzystywany jest często w muzyce kościelnej[[25]](#footnote-25), gdyż w zestawieniu z toniką tworzy poważnej, dostojne wrażenie. W gamie molowej jest molowy, natomiast w durowej – durowy.
* Dominanta – akord o budowie tercjowej zbudowany na V stopniu gamy, uzupełniony o stopień VII i II. Jest bardzo charakterystyczną funkcją harmoniczną, występuje niejako w opozycji do ośrodka tonalnego (akordu toniki), ze względu na zawarte w niej stopnie bezpośrednio sąsiadujące z toniką (I stopniem gamy)[[26]](#footnote-26). Pojawienie się jej w danej frazie bywa często momentem kulminacyjnym i zwiastuje rozładowanie zbudowanego napięcia. W gamach molowych często stosuje się skalę harmoniczną, by móc wykorzystać wówczas akord dominanty, w którym podwyższony VII stopień gamy jest oddalony od toniki zaledwie od pół tonu.
* Dominanta septymowa – akord dominanty z dodanym IV stopniem gamy, który wraz z V stopniem gamy, na którym akord jest zbudowany, tworzy interwał septymy małej. Zastosowanie tego akordu potęguje wyrazistość dominanty[[27]](#footnote-27) i powoduje, że jeszcze wyraźniej prowadzi ona do rozwiązania na tonikę.
* Triada harmoniczna (inaczej akordy główne) - trzy podstawowe trójdźwięki zbudowane na pierwszym (tonika), czwartym (subdominanta) i piątym stopniu gamy (dominanta).



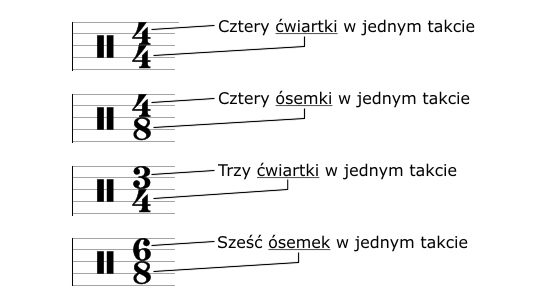
Rysunek 8 Akordy triady harmonicznej w gamie C - dur

* Znaki przy kluczu – są to krzyżyki bądź bemole zapisane przy kluczu na początku pięciolinii, które obowiązują dla wskazanych nutach do odwołania. Zaznaczane są na odpowiedniej wysokości i odnoszą się do wszystkich nut o danej wysokości dźwięku obniżając go lub podwyższając o pół tonu[[28]](#footnote-28). Dzięki temu oznaczeniu na początku systemu, nie jest konieczne prezentowanie znaków przy każdej nucie. Dodatkowo oznaczenie to informuje nas w jakiej tonacji zapisany jest utwór, ponieważ każdy zestaw znaków jest przyporządkowany tylko do jednej tonacji durowej i molowej. Podstawowa para gam (C-dur i a-moll) jako jedyne nie posiadają znaków przy kluczu.



Rysunek 9 Znaki przy kluczu wiolinowym, źródło: https://upload.wikimedia.org/wikipedia/commons/thumb/8/84/A-major\_f-sharp-minor.svg/150px-A-major\_f-sharp-minor.svg.png

* Metrum – schemat opisujący rytm w utworze, obrębie każdego taktu. Określa układ akcentów oraz wartości rytmiczne, na które każdy takt się dzieli[[29]](#footnote-29). W połączeniu z określeniem tempa taktowania utworu, pozwala określić długość trwania każdej z nut. Metrum podaje się poprzez określenie ile oraz jakich wartości rytmicznych mieści się w jednym takcie.



Rysunek 10 Przykłady metrum i ich wyjaśnienie, źródło: http://e-perkusja.pl/plik/nauka\_gry\_na\_perkusji/Artykuly/Czytanie\_z\_nut/04\_Metrum\_gorna\_cyfra.jpg

* Takt – określony poprzez metrum odcinek czasowy, najmniejsza jednostka podziału rytmicznego utworu[[30]](#footnote-30). Na pięciolinii znakiem końca taktu i zarazem początku kolejnego jest pojedyncza kreska taktowa.
* Wartość rytmiczna nuty – oznaczenie pomagające określić względną długość trwania nut w utworze. Ich bezwzględna długość trwania zależy oczywiście od tempa wykonywanego utworu i może być zmienna, lecz przyporządkowanie wartości rytmicznych pozwala zapisać stałą zależność długości trwania zachodzącą między poszczególnymi nutami[[31]](#footnote-31). Dostępne wartości odpowiadają zwielokrotnieniom mniejszych nut w ilości odpowiadającej potędze liczby 2[[32]](#footnote-32). Do podstawowych oznaczeń rytmicznych należą:
  + cała nuta (równa dwóm półnutom)
  + półnuta (równa dwóm ćwierćnutom)
  + ćwierćnuta (równa dwóm ósemkom)
  + ósemka (równa dwóm szesnastkom)
  + szesnastka (równa dwóm trzydziestodwójkom)
  + trzydziestodwójka (równa dwóm sześćdziesięcioczwórkom)
  + itd…

## Przegląd dostępnych rozwiązań

Automatyczne komponowanie muzyki jest wyzwaniem, z którym naukowcy działający w obszarze sztucznej inteligencji mierzą się od 20 - 30 lat. Powstało wiele rozwiązań, które starają się naśladować genialnego w swej pracy rzeczywistego kompozytora, który potrafi stworzyć dzieło przyjemne dla słuchacza, dodatkowo zawierające swój unikatowy styl[[33]](#footnote-33). Prace w tej dziedzinie są na tyle zaawansowane, że powstały już rozwiązania komercyjne umożliwiające generowanie muzyki z bardzo dobrymi efektami, przydatnymi dal osób próbujących układać własne kompozycje muzyczne. Niektóre rozwiązania użyte w programie *MusicAnalyzer* mają swoje odpowiedniki w przedstawionych poniżej przykładach.

## Emily Howell

*Emily Howell[[34]](#footnote-34)* to nazwa programu komputerowego autorstwa Davida Cope’a powstałego w latach 90-tych na uniwersyteciew Santa Cruz w Kalifornii. 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[[35]](#footnote-35), wykształciła własny styl kompozytorski.

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

|  |  |
| --- | --- |
| https://patentimages.storage.googleapis.com/US7696426B2/US07696426-20100413-D00001.png  Rysunek 11 Schmeta działania Emily Howell, źródło https://www.google.com/patents/US7696426 | 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 6. Wybór najlepszego utworu i zwrócenie go na wyjście programu |

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[[36]](#footnote-36) na terenie Stanów Zjednoczonych chroniący jego autorski algorytm komponowania muzyki. Ponadto program *EMI* w 1997 roku na Uniwersytecie w Oregonie zmierzył się w teście[[37]](#footnote-37) przypominającym muzyczny odpowiednik Testu Turinga*.* Na podstawie posiadanej bazy utworów J. S. Bachastworzył własną kompozycję (miniaturę fortepianową) imitującą styl geniusza epoki baroku. Podobne zadanie otrzymał szanowany, współczesnykompozytorSteve 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* (2009) i *Breathless* (2012). Niemniej jednak, sztuczne tworzenie muzyki jest wciąż źle odbierane przez środowisko artystyczne, przez co wielu muzyków odmawia wykonywania utworów autorstwa *Emily*.

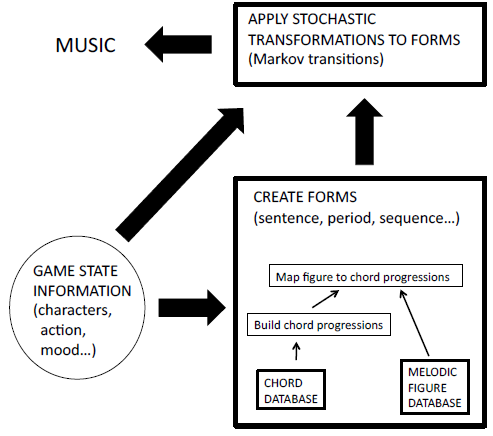
## Mezzo

W 2012 roku na tej samej uczelni, na której działał David Cope, pojawił się kolejny projekt z gatunku automatycznego komponowania muzyki. Wówczas Daniel Lankford Brown ukończył swój przewód doktorski o temacie *Expressing narrative function in adaptive, computer-composed music[[38]](#footnote-38)*. W ramach swoim badań stworzył program komputerowy *Mezzo*, który generuje na żywo tło muzyczne do dowolnej gry komputerowej prowadzonej przez użytkownika komputera. Powstająca muzyka pochodzi z gatunku neoromantyzmu, gdyż zgodnie z zamierzeniem, zasady harmonii i komponowania melodii zapisane w programie były konsultowane z takimi muzykologami jak V. Kofi Agawu, M. Grabócz, W. Caplin, R. Hatten czy B. Almén.

Projekt ten starał się sprostać przede wszystkim dwóm wyzwaniom. Jak stworzyć melodię, która ciągle ewoluuje, lecz mimo powtarzanych czynności przez gracza nie staje się nudna? Drugim wyzwaniem jest problem dopasowania muzyki do emocji przeżywanych przez gracza. Według autora, można im sprostać, gdy sposób powtarzania motywów muzycznych będzie czymś więcej niż repetycją, gdy zawarte w nich frazy i dźwięki będą miały określone funkcje, które będą się stopniowo zmieniać, ewoluować a niektóre z nich będą się wyróżniać tworząc główną linię melodyczną.

Tworzenie własnych kompozycji opiera się wpierw na wczytaniu istniejących motywów muzycznych oraz progresji harmonicznych. Następnie przy pomocy algorytmu genetycznego powstają indywidualne progresje tworzone jednak na wzór tych wejściowych, które dzięki temu posiadają wspólny styl muzyczny. Wówczas, kiedy gracz uruchamia grę i w nią gra, program w czasie rzeczywistym stara się dopasować posiadane frazy muzyczne do odpowiednich momentów w grze, zachowując przy tym ich ciągłość, łagodne przejścia i podobieństwo między kolejnymi motywami. W ramach kolejnych fraz generowane są kolejne zaburzenia, kulminacje, aby zsynchronizować je z tym, co się dzieje w grze. Dla przykładu, w odpowiedzi na osiągnięcie sukcesu przez gracza, pojawia się progresja wstępująca, która odzwierciedla emocje związane radością i tryumfem. Analogicznie, w momencie porażki pojawia się melodia zstępująca, często w tonacji molowej, która daje odczucie zawiedzenia, straconej szansy. Aby uniknąć zbytniej powtarzalności tych modyfikacji, są one dobierane stochastycznie: możliwe transformacje są zamodelowane w postaci łańcuchów Markowa. Powtórzenia motywów i przejścia między określonymi frazami są oznaczone pewnym prawdopodobieństwem (zgodnie z parametrami dla danego fragmentu gry), przez co modyfikacja regresji harmonicznych nawet w tym samym fragmencie gry jest za każdym razem nieco inna.

Poniższy schemat przedstawia sposób generowania akompaniamentu muzycznego przez program *Mezzo* podczas gry.



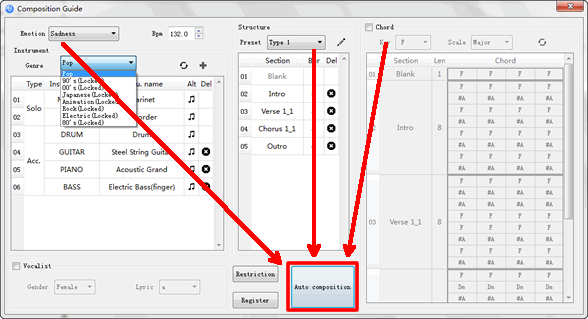
Rysunek 12 Schemat działania programu Mezzo, źródło http://www.danielbrownmusic.com/uploads/1/3/2/3/13234393/final\_dissertation\_final\_edit.pdf

s. 77

Na początku każdej iteracji następuje pobranie informacji o stanie gry (emocje, osiągnięcia, przewidywany humor gracza itd.). Następnie przy wykorzystaniu posiadanej bazy współbrzmień i figur melodycznych, powstają progresje muzyczne na najbliższe kilka lub kilkanaście taktów, które tworzą zdania muzyczne i całe frazy. W ostatnim etapie zachodzą transformacje stochastyczne aby wprowadzić do powstałych fraz element twórczy, niepowtarzalny.

## AthTek DigiBand

*DigiBand*[[39]](#footnote-39) to rozwiązanie komercyjne, które umożliwia łatwe generowanie linii melodycznych dla wybranych instrumentów. Można wskazać rytm, metrum, instrumenty, styl muzyki oraz charakter (klimat emocjonalny) utworu, lub też nagrać próbkę dźwiękową i program sam ułoży pozostałe ścieżki dźwiękowe w dopasowaniu do wczytanej melodii. Jest to narzędzie reklamowane jako pomoc przy tworzeniu własnych kompozycji, gdyż wedle wyboru instrumentów i gatunku pożądanej muzyki, resztę pracy wykonuje zamiast człowieka. Jego zaletą jest także szybkie działanie, lecz niestety brak informacji o algorytmie z jakiego korzysta przy generowaniu muzyki.



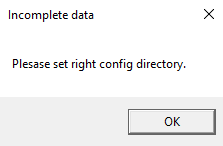
Rysunek 13 Interfejs programu DigiBand, źródło http://www.athtek.com/image/digiband/composition.jpg, strona odwiedzona 1.09.2016

# Opis rozwiązania

## Opis interfejsu graficznego programu MusicAnalyzer

## Informacje bazowe – pliki konfiguracyjne

Do prawidłowego działania programu potrzebny jest zestaw plików konfiguracyjnych, które są dołączone także do projektu *MusicAnalyzer*. Aplikacja wymaga wskazania lokalizacji, w której mają się one znajdować. Jeśli ich tam nie znajdzie, podczas wczytywania pliku wejściowego pokaże się komunikat:



Rysunek 14 Komunikat o braku plików konfiguracyjnych w MusicAnalyzer

Dalsze działanie programu zostanie przerwane, konieczne będzie wskazanie innej lokalizacji na komputerze zawierającej wszystkie niżej wskazane pliki:

* **basicPitches.txt** – plik zawiera 12 wierszy, w których wyszczególnione są wszystkie podstawowe dźwięki gamy (C-dur) w postaci pary nr kolejny i oznaczenie muzyczne np. „0 C”; dzięki niemu możliwe jest np. przyporządkowanie nazw dźwięków do określonej wartości zapisanej w sygnale MIDI
* **minorSequence.txt** – plik ten zawiera 7 wierszy (tyle dźwięków zawiera każda gama), a każdy z nich zawiera numer kolejny dźwięku gamy molowej z ciągu wszystkich dźwięków dostępnych na klawiaturze fortepianu w obrębie oktawy (zawartych w pliku *basicPitches.txt*); umożliwia to rozpoznanie, czy dany dźwięk należy do skali oraz z jakich dźwięków można składać akordu tonalne
* **majorSequence.txt** – ten plik zawiera analogiczne 7 wierszy, jak opisany powyżej, z tym, że jest to ciąg dźwięków dowolnej gamy durowej
* **majorChords.txt** – zawiera spis akordów (trójdźwięków) charakterystycznych dla gamy durowej:
  + tonika
  + subdominanta
  + dominanta (w skali harmonicznej)
  + trójdźwięk na 2-im stopniu gamy
  + trójdźwięk na 3-im stopniu gamy
  + trójdźwięk na 6-ym stopniu gamy

widza ta pozwala stworzyć powyższe akordy odpowiednie dla każdej tonacji

* **minorChords.txt** – zawiera spis akordów (trójdźwięków) charakterystycznych dla gamy molowej:
  + wszystkie trójdźwięki wskazane dla gamy durowej
  + dominanta (w skali eolskiej)
  + trójdźwięk na 3-im stopniu gamy (w skali eolskiej)

## Wczytywanie melodii wejściowej i metody określające jej cechy

Proces wczytywania i analizowania informacji zawartych w wejściowym pliku *MIDI* w programie *MusicAnalyzer* składa się z kilku etapów:

## Zdekodowanie sygnałów dźwięków

Obiekt pliku dźwiękowego przechowuje klasa *Sequence* (z biblioteki *Sanford[[40]](#footnote-40)*), która jest kolekcją obiektów typu *Track*, które odpowiadają ścieżkom dźwiękowym, do których przypisane są sygnały dźwiękowe. Należy zatem iterować każdą z nich, aby uzyskać wszystkie dźwięki. Jednak do ścieżki dźwiękowej przypisane są nie tylko sygnały pojawienia się i wygaśnięcia dźwięku, dlatego z kolekcji obiektów *MidiEvent* należy wybrać tylko te będące typu *ChannelMessage*. Następnie jeśli mamy do czynienia z sygnałem o komendzie *NoteOn*, to jest to sygnał pojawienia się dźwięku, który dodatkowo posiada atrybuty:

* Wysokość dźwięku (możliwe wartości z zakresu *<A0 = 21; C8 = 109>*)
* Poziom głośności dźwięku (0 – 100)
* Umiejscowienie w czasie (ilość taktowań od początku utworu)
* Nr ścieżki dźwiękowej

Niestety w plikach *MIDI* moment zakończenia dźwięku jest przez różne urządzenia oznaczany dwojako: poprzez nadanie sygnału *NoteOff* lub sygnału *NoteOn*, którego poziom natężenia dźwięku = 0. W przypadku spełnienia jednego z powyższych warunków zapisywane są dodatkowe atrybuty wskazanej nuty:

* Oznaczenie znakowe nuty (np. C4)
* Jej odpowiednik w podstawowej skali (C)
* Oktawa
* Koniec trwania
* Długość trwania

## Wyszukanie zmian tonacji utworu

Podobnie, jak odszyfrowywanie dźwięków, wyszukiwanie zmian tonacji utworu wymaga przejrzenia wszystkich ścieżek w pliku i przypisanych doń sygnałów. W tym przypadku należy jednak odnaleźć obiekty *MidiEvent[[41]](#footnote-41)* będące typu *MetaMessage*, których atrybut *MetaType = KeySignature*. Tak powstają instancje klasy *Tonation*, które następnie są uzupełniane między innymi o atrybuty:

* *Offset* – przesunięcie tonacji względem skali C-dur
* Zbiór dźwięków gamy należących do tej skali
* Dźwięki wszystkich akordów należących do tej skali

Dodatkowo, po znalezieniu wszystkich zmian tonacji, dla każdej z nich ustalany jest moment jej początku i końca (ilość taktowań od początku utworu).

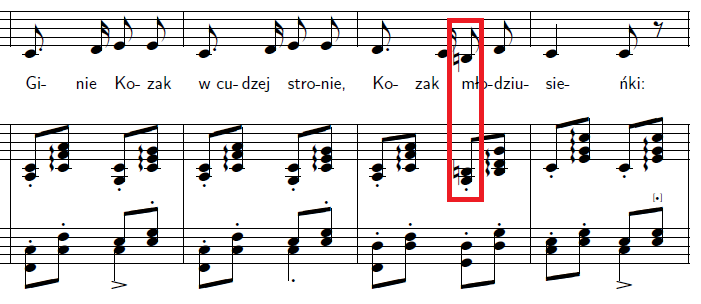
## Wyszukanie zmian metrum

Wyznaczanie metrum i jego zmian odbywa się analogicznie jak dla tonacji, z tą różnicą, że atrybut *MetaType* musi przyjmować wartość *TimeSignature*. Po znalezieniu wszystkich takich zdarzeń, zapisywane jest także momenty początku i końca okresu ich obowiązywania.

## Ustalenie prawidłowej tonacji utworu

Niestety tonacja utworu odczytana z pliku *MIDI* przy pomocy narzędzi z biblioteki *Sanford.Multimedia*[[42]](#footnote-42), nie daje nam gwarancji, że jest prawidłowa. Wynika to z faktu, iż w pliku zapisana jest wartość przesunięcia danej tonacji względem tonacji bazowej (C-dur bądź a–moll). Dlatego, aby odróżnić tonację molową od durowej, która posiada te same znaki przy kluczu, musimy wykonać analizę występowania charakterystycznych dźwięków skali.

1. Dla dowolnej pary tonacji dur – moll, posiadających te same znaki przy kluczu, większość dźwięków występując w zapisie nutowym jest taka sama[[43]](#footnote-43). Istotna różnica może tkwić w siódmym podwyższonym dźwięku w skali molowej harmonicznej, który w takim przypadku nie ma swojego odpowiednika w skali durowej. Kompozytorzy stosują ten zabieg, aby uwydatnić brzmienie dominanty, która wówczas buduje napięcie w utworze dążące do rozwiązania na akordzie toniki. Przykładem może być pieśń S. Moniuszki „Kozak”, w której przed zakończeniem każdego wersu w zwrotce występuje akord dominanty z podwyższonym VII stopniem gamy i następnie rozwiązuje się on na tonikę.



Rysunek 15 Fragment zapisu nutowego pieśni "Kozak" S. Moniuszki z akompaniamentem pianina (c-moll).

Oczywiście, ten zabieg nie jest obowiązkowy, lecz występuje w ogromnej większości kompozycji klasycznych[[44]](#footnote-44), co w porównaniu z faktem, iż odpowiadający mu piąty dźwięk w skali durowej praktycznie nigdy nie jest podwyższany, umożliwia dość celnie określić, czy dany utwór jest w skali durowej czy molowej. Skonstruowano zatem warunek realizujący to założenie:

Jeśli *ilość wystąpień VII stopnia* <= (2 \* *ilość wystąpień podwyższonego VII st.*)

To jest to gama molowa

Inaczej jest to gama durowa

1. Drugą cechą utworu poddawaną analizie, jest statystyka akordów na początków taktów, czyli w miejscach najistotniejszych dla budowania rytmu i układu harmonicznego. Są to węzły, które osadzają rytm, występują na nich dłuższe wartości rytmiczne i to one w dużej mierze charakteryzują tonację, w jakiej napisany zostały badany utwór. Dlatego równorzędnym warunkiem dla analizy występowania podwyższonego VII stopnia gamy, jest określenie, czy częściej na początku taktu występuje akord toniki w tonacji dur czy moll:

Jeśli *ilość wystąpień toniki durowej na początku taktu* <= *ilość wystąpień toniki molowej na początku taktu*

To jest to gama molowa

Inaczej jest to gama durowa

Oba opisane powyżej warunki zostały połączone alternatywą, to znaczy aby gamę uznać za molową, wystarczy spełnić tylko jeden z nich.

## Ustalenie prawidłowych oznaczeń dźwięków

Kolejną niejednoznacznością, z jaką należy się zmierzyć korzystając z plików *MIDI*, jest fakt, iż posiadając jedynie bezwzględną wysokość każdego z dźwięków, nie jesteśmy w stanie od razu zaprezentować ich na pięciolinii w zapisie nutowym. Dla przykładu, dźwięk o wysokości 49 (w formacie *MIDI*) może być jednocześnie oznaczony jako Cis4 i Des4[[45]](#footnote-45). Jest to uzależnione od tonacji utworu w którym te dźwięki występują.

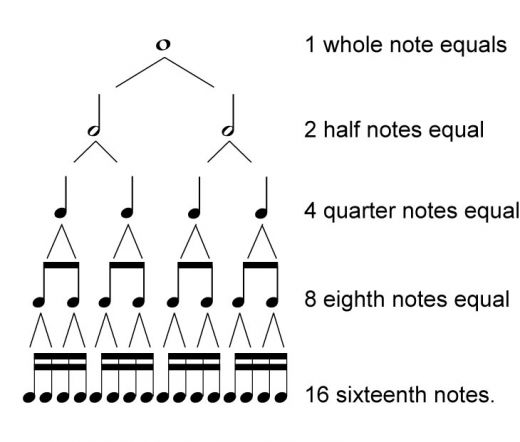


Rysunek 16 Tonacje i znaki przy kluczu wiolinowym, źródło http://blog.gitarius.pl/kurs\_gitarowy/chro4.png

Jeśli wskazana tonacja przy kluczu posiada krzyżyki, wówczas dźwięk ten oznaczylibyśmy jako Cis4, w przeciwnym razie (posiada bemole lub nie posiada znaków przy kluczu), to będzie on oznaczony jako Des4. Ma to znaczenie przede wszystkim pod kątem prawidłowej prezentacji zapisu nutowego, lecz także pozwala uniknąć problemów z porównywaniem dźwięków w innych etapach prowadzonych analiz.

## Ustalenie wartości rytmicznych wszystkich nut

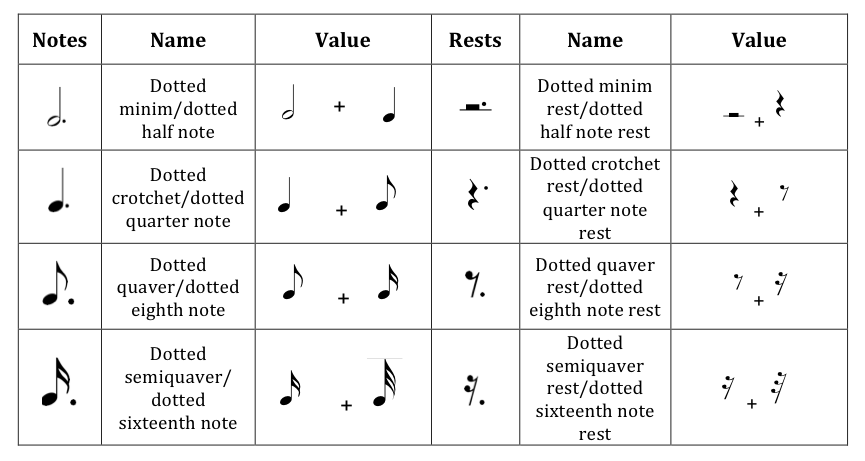
Ostatnim etapem uzupełniania atrybutów wczytanych nut jest prawidłowe dopasowanie wartości rytmicznych względem długości ich trwania. Jest to możliwe dzięki uzyskaniu we wcześniejszych etapach informacji o metrum utworu oraz zapisanej w pliku wartości *Division* – ilości taktowań zegara w ramach jednej ćwierćnuty[[46]](#footnote-46). Mając tą wiedzę jesteśmy w stanie przypisać wartości rytmiczne dźwiękom w zależności od długości ich trwania. Ten krok jest konieczny dla prawidłowego zaprezentowania zapisu nutowego, dlatego też *MusicAnalyzer* przechowuje ten atrybut w formie typu wyliczeniowego *PSAMControlLibrary.MusicalSymbolDuration[[47]](#footnote-47)*, który jest rozpoznawany przez komponent graficzny (*IncipitViewerWPF*) prezentującego nuty na pięciolinii.



Rysunek 17 Podział wartości rytmicznych, źródło http://whittier.mpls.k12.mn.us/uploads/rhythm\_tree\_1.jpg

Powyższy schemat przedstawia jedynie podstawowe wartości, których długość jest pewną potęgą dwójki, co oczywiście nie wystarcza, by odpowiednio zaprezentować wszystkie nuty. Dlatego zapis nutowy umożliwia przedłużanie wskazanej wartości o połowę poprzez dodanie obok kropki[[48]](#footnote-48). Możliwe jest także dodawanie kolejnych kropek, wówczas każda z nich przedłuża o dwukrotnie mniejszą wartość (1/4, 1/8 bazowej wartości rytmicznej itd.). Ilustruje to poniższa tabela.

Tabela 1 Wartości rytmiczne przedłużane, źródło https://musictheorysite.files.wordpress.com/2015/06/dotted-value.png



W programie *MusicAnalyzer* w procesie dopasowywania wartości rytmicznych przyjęto założenie, że nuta może mieć maksymalnie dwie kropki przedłużające jej wartość, gdyż w praktyce nie spotyka się ich w większej ilości. Występuje tu jednak dodatkowa trudność w związku z zapisem dźwięków w formacie *MIDI*. Mimo, że w zapisie nutowym dwa dźwięki występują bezpośrednio po sobie, to jednak, aby nie powodować wrażenia nachodzenia ich na siebie oraz by odtwarzacz pliku prawidłowo wyłączył pierwszy dźwięk i włączył kolejny, wszystkie dźwięki zazwyczaj są trochę krótsze (o ok. 10-15% czasu)[[49]](#footnote-49). Powoduje to problem matematyczny, gdyż wskazane wartości należy odpowiednio zaokrąglić w górę, lecz jednocześnie odróżnić od siebie różne wartości o zbliżonych do siebie długościach (np. półnutę od ćwierćnuty z dwiema kropkami).

## Prezentacja graficzna zapisu nutowego

Po ustaleniu wszystkich atrybutów nut, zmian tonacji i metrum, należy odpowiednio przygotować każdą z pięciolinii. Biblioteka *PSAMWPFControlLibrary[[50]](#footnote-50)* umożliwia prezentowanie zapisu nutowego w elemencie graficznym typu *IncipitViewerWPF*. Jego interfejs wymaga podawania po kolei wszystkich oznaczeń, które mają się pokazać na pięciolinii, a zatem nie tylko nut, ale też klucza, oznaczenia metrum czy kresek taktowych. Do tego celu służy przeciążona metoda:

*PSAMWPFControlLibrary*.*IncipitViewerWPF.AddMusicalSymbol(MusicalSymbol symbol)*

Umożliwia dodanie do zapisu dowolnych obiektów, których klasy dziedziczą po klasie *MusicalSymbol.*

*MusicAnalyzer* prezentuje nuty przypisane do osobnych ścieżek dźwiękowych na oddzielnych pięcioliniach. W tym celu, dla każdego kanału należy wskazać w jakim kluczu przedstawić zapis nutowy (wiolinowym czy basowym).



Rysunek 18 Fragment partii tenora z chóralnej aranżacji utworu ludowego "Oberek powiślański"

Powyższy przykład dobrze ilustruje problem z wyborem klucza dla wskazanej melodii. Tessitura głosu tenorowego jest często zbyt wysoka do zapisu w kluczu basowym, natomiast bywa też za nisko, jak na klucz wiolinowy (przykład na rysunku). W obu przypadkach łączy się to z koniecznością dorysowywania dodatkowych linii powyżej bądź poniżej pięciolinii. Prowadzona jest zatem analiza wysokości wszystkich nut należących do każdej ścieżki dźwiękowej, aby określić, który z kluczy pozwala zaprezentować je bardziej przejrzyście. Program zlicza nuty, które występują powyżej i poniżej wartości granicznej, za którą przyjęto dźwięk C4 – dźwięk równie odległy od obu pięciolinii. Jeśli więcej jest dźwięków wyższych lub równych C4, to zapis będzie prowadzony w kluczu wiolinowym, w przeciwnym wypadku - w kluczu basowym.

Po ustaleniu klucza danej pięciolinii, program iteruje po wszystkich nutach należących do wskazanej ścieżki dźwiękowej. Zanim doda ją do kolekcji elementów należących do pięciolinii, każdorazowo sprawdza, czy w momencie pojawienia się kolejnej nuty nie jest także przypisany sygnał zmiany tonacji lub metrum. W takim przypadku, najpierw należy wstawić znaki przy kluczu bądź nowe metrum a dopiero później nutę[[51]](#footnote-51). Dodatkowo następuje sprawdzanie, czy na ten moment nie przypada koniec taktu. Wówczas należy postawić znak kreski taktowej. Ostatnim elementem, który może poprzedzać nutę jest znak pauzy wynikający z faktu, iż poprzednia nuta skończyła się wcześniej. Jeśli zbiega się to dodatkowo z momentem stawiania kreski taktowej, należy sprawdzić, czy pauza należy jeszcze do poprzedniego taktu, czy już do następnego i umieścić znaki w odpowiedniej kolejności. Program każdorazowo wylicza odpowiednią długość przerwy i przyporządkowuje jej wartość rytmiczną. W programie *MusicAnalyzer* przyjęto uproszczenie, iż wstawiane pauzy będą miały swoje nominalne wartości rytmiczne i nie będą przedłużane.

Gdy na pięciolinii należy jednak umieścić nutę, program uzupełnia jej wcześniej wyliczone atrybuty jeszcze o bezpośrednie cechy widoku. Należy do nich połączenie nut zaczynających się w tym samym momencie i należących do tego samego kanału dźwiękowego w pionowy akord. Tę zależność i jej prezentację przedstawia poniższa przykładowa ilustracja:



Rysunek 19 Trójdźwięk zapisany na pięciolinii, źródło http://www.codeproject.com/KB/miscctrl/psamcontrollibrary/examplechords.png, strona odwiedzona 1.09.2016

Aby nuta była w ten sposób zaprezentowana, musi otrzymać wartość wskaźnika *isChordElement = true*. Konstruowanie akordów odbywa się poprzez porównanie aktualnej nuty z poprzednią. Jeśli obie zaczynają się w tym samym czasie, to należy im ustawić powyższą flagę, w przeciwnym razie przyjmuje ona domyślną wartość *false*.

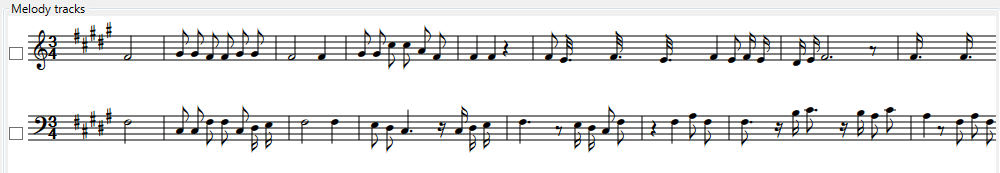
Ostatnim atrybutem nuty pozostającym do ustalenia przed wygenerowaniem jej widoku jest kierunek ogonka nuty, jeśli dana wartość rytmiczna taki posiada. Dotyczy to zatem wszystkich nut za wyjątkiem całej nuty. Możliwe różnice w prezentacji przedstawia poniższy rysunek:



Rysunek 20 Fragment zapisu nutowego z nutami o różnym kierunku

Widać wyraźnie, że nuta o niższej wysokości dźwięku prezentowana w dolnej części pięciolinii jest prezentowana bardziej przystępnie, gdy jej ogonek jest skierowany do góry, a dla wyższych nut analogicznie w dół. *MusicAnalyzer* dla każdej nuty określa ten kierunek w zależności od wysokości nuty i klucza pięciolinii, na jakiej się znajduje[[52]](#footnote-52). Nuty z ogonkami ‘w dół’ w kluczu wiolinowym to od H4 i wyżej, a niższe mają ogonek ‘do góry’. W kluczu basowym ogonek ‘w dół’ mają nuty od D3 i wyżej, a niższe ‘do góry’. Za przechowanie tej informacji na potrzeby wyświetlenia nuty w obiekcie *IncipitViewerWPF* odpowiada typ wyliczeniowy *PSAMControlLibrary.NoteStemDirection*. W przyjętym rozwiązaniu zastosowano także powiązanie nut należących do tego samego akordu, które są prezentowane w pionie. W takim przypadku, wiążąca jest pozycja najniższej nuty w akordzie, ona determinuje kierunek ogonka, a pozostałe prezentowane są z tym samym kierunkiem.

Po ustaleniu opisanych wyżej atrybutów związanych z prezentacją zapisu nutowego, obiekt klasy *IncipitViewerWPF* wyświetla wszystkie elementy zgodnie z kolejnością podania ich na wejściu. Biblioteka *PSAMControlLibrary* umożliwia też wiązanie sąsiadujących ze sobą nut poprzez określenie ich atrybutu w postaci typu wyliczeniowego *PSAMControlLibrary*.*NoteBeamType*, lecz w tym projekcie to rozwiązanie nie jest stosowane. Jeśli nuty nie są częścią akordu granego jednocześnie, każdorazowo prezentowane są osobno.



Rysunek 21 Przykładowy zapis nutowy dwóch głosów w programie MusicAnalyzer

## Algorytm tworzenia akompaniamentu

## Harmony Search

## Wyszukanie współbrzmień wejściowych

## Wybór węzłów dla akordów

## Adaptacja algorytmu HS

## Ustalenie atrybutów nut akompaniamentu

## Zapis nutowy i dźwiękowy powstałego akompaniamentu

Po zakończeniu działania algorytmu *Harmony Search*, *MusicAnalyzer* wybiera najlepsze utworzone rozwiązanie i prezentuje jego zapis nutowy jako kolejną ścieżkę dźwiękową widoczną na ekranie głównym aplikacji. Zawarte w niej nuty posiadają analogicznie te same atrybuty, co zdekodowane dźwięki ze ścieżek z pliku wejściowego.

Możliwe jest także wydrukowanie zapisu nutowego akompaniamentu w formacie pionowym i w podziale na strony w zależności od długości zapisu. Opcja ta dostępna jest z menu głównego ekranu (*Menu -> Print accompaniment*). Należało w tym przypadku dokonać pewnych przekształceń, gdyż element graficzny w oknie aplikacji prezentuje nuty w jednym wierszu, co w przypadku prezentacji na wydruku nie byłoby funkcjonalne. W tym celu należało podzielić jedną pięciolinię na wiele systemów tak, aby w pełnej wielkości mogły się one pomieścić na kartce papieru. Aby podzielić element graficzny na wiele części należało wpierw zapisać jego zawartość jako bitmapę a następnie kolejne fragmenty skopiować do nowej bitmapy, której wymiary umożliwiają pomieszczenie wielu systemów z pięcioliniami. Dodatkowo należało zastosować podział na strony w przypadku, gdy zapis miałby zajmować więcej niż 1 stronę. Pomocny okazał się artykuł[[53]](#footnote-53) Benjamina Walkera, w którym opisuje on sposób implementacji podziału wydruku na strony przy konwersji elementów graficznych na bitmapy.

Przy użyciu narzędzi systemowych *Windows,* wydruk można zapisać do pliku *.pdf*. Przykładowy wydruk jest przedstawiony na poniższej ilustracji. Brak bocznych marginesów jest zdeterminowany tym, iż wskazane narzędzie zapisujące plik w formacie *.pdf* umożliwia wydruk na całej stronie, bez marginesów. Przy użyciu fizycznej drukarki, ilość wierszy i ich szerokość będzie dostosowana do możliwości drukarki.



Rysunek 22 Przykładowy wydruk zapisu nutowego akompaniamentu

Program pozwala także zapisać powstałą ścieżkę dźwiękową do pliku *MIDI*. Zostaje ona dodana do ścieżek istniejących i w takiej formie możemy nadpisać plik wejściowy lub utworzyć nowy. Program umożliwia wówczas także odtworzenie melodii bazowej wraz z akompaniamentem, co uławia ocenę jego dopasowania.

Zapisanie nowej ścieżki do pliku *MIDI* jest wspierane przez bibliotekę *Sanford.Multimedia[[54]](#footnote-54)* poprzez dodanie obiektu klasy *Track* do obiektu *Sequence*, który jest kolekcją ścieżek dźwiękowych. Wpierw należy stworzyć ścieżkę z odpowiednimi dźwiękami oraz z oznaczeniem wyboru instrumentu, którego próbki mają być wykorzystane do odtworzenia melodii. W tym celu należy dodać na samym początku ścieżki obiekt *ChannelMessage*, dla którego typ komendy to *ProgramChange* (jest to oznaczenie zmiany instrumentu zgodne ze standardem *MIDI*). Obiekt ten należy stworzyć przy pomocy obiektu *ChannelMessageBuilder*, aby utworzyć odpowiedni ciąg bajtów do zapisu w pliku binarnym. Jednym z argumentów jest numer instrumentu z zakresu *<0; 127>* zgodny z podstawowym standardem *MIDI[[55]](#footnote-55)* (np. *AcousticGrandPiano* = 0). Następnie należy dodać wszystkie nuty należące do wskazanej ścieżki. Muszą być umieszczone we wskazanym miejscu w czasie oraz dla każdej z nich należy nadać komendy *NoteOn* (włączenie dźwięku) i *NoteOff* (wyłączenie). Do ich wygenerowania program korzysta tak samo jak we wcześniejszym przypadku z obiektu *ChannelMessageBuilder*.

## 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[[56]](#footnote-56)* to biblioteka dla platformy .Net udostępniająca klasy i metody ułatwiające odczytywanie i analizę plików dźwiękowych w formacie *Musical Instrument Digital Interface (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[[57]](#footnote-57) 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[[58]](#footnote-58)*, 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

Sposób oceny tworzenia akompaniamentu przez program *MusicAnalyzer*:

* Jako dane źródłowe zostały użytych 5 różnych melodii jedno- bądź wielogłosowych:
  + Oberek powiślański (dwugłos: sopran i bas) – *OberekPowiślańskiSB.mid*
  + Marsz Turecki (melodia prawej ręki na fortepianie) – *TurkishMarch.mid*

Wszystkie pliki dźwiękowe są dostępne w folderze *\Midis* dostępnym w głównym folderze projektu *MusicAnalyzer*.

* Dla każdego testu zostały wywołane trzy uruchomienia programu, a ocenie zostało poddane jedynie to dające najlepsze efekty, co zostało określone na podstawie funkcji oceny w algorytmie harmonicznym. Uczyniono tak, gdyż *Harmony Search* to algorytm dający niedeterministyczne rozwiązania, dzięki czemu przy każdym jego uruchomieniu nawet przy tych samych parametrach, wyniki mogą nieznacznie różnić się od siebie.
* Wszystkie testy programu będą wykonywane na tym samym komputerze, bez obciążenia go pracą innych aplikacji, przy zastosowaniu dwóch ustawień parametrów:
  + I wariant – test jakościowy (przy wykorzystaniu ograniczonych, lecz swobodnych zasobów)
    - HMS
    - hmcr
    - Par
    - delta
    - MaxIter
    - BestTrackUnchanged
    - MaxTime

Dobranie powyższych wartości zostało utworzone na podstawie badań empirycznych, przy jakich ustawieniach program jest w stanie stworzyć najbardziej wydajną kompozycję.

* + II wariant – test jakości przy zachowaniu wydajności działania; głównym kryterium ograniczającym będzie czas działania algorytmu = 1 s.
    - HMS
    - hmcr
    - Par
    - delta
    - MaxIter
    - BestTrackUnchanged
    - MaxTime

Dobranie powyższych wartości zostało utworzone na podstawie badań empirycznych, przy jakich ustawieniach program działa w przystępnym czasie dla użytkownika przy zachowaniu maksymalnej możliwej jakości.

* Do oceny działania każdej wygenerowanej kompozycji zostało powołanych 5 osób na co dzień związanych z wykonywaniem muzyki klasycznej, aby oceniły działanie programu wedle ustalonych kryteriów. Nie wszystkie osoby potrafią czytać zapis nutowy. Uśrednienie wyników ich oceny umożliwi wskazanie względnie obiektywnej oceny działania programu w różnych aspektach i pod różnymi ustawieniami.

## Kryteria oceny

Osoba testująca ocenia każdą kompozycję nadając punkty <1; 6> pod kątem następujących kryteriów:

* Harmoniczne współbrzmienie akompaniamentu z melodią główną (1 – bardzo słabe, 6 – perfekcyjne)
* Rytm zmieniania się akordów (1 – rytm akompaniamentu zupełnie nie odpowiada rytmowi melodii bazowej, 6 – perfekcyjne dopasowanie momentów zmian akordów)
* Podobieństwo akompaniamentu do melodii wejściowej (1 – akompaniament jest uderzająco podobny do melodii bazowej, 6 – akompaniament sprawia wrażenie osobnej lecz dobrze współbrzmiącej melodii)
* Różnorodność współbrzmień (1 – powtarzalne akordy tworzone na tonice dźwięku melodii bazowej, 6 – ciekawe, innowacyjne współbrzmienia, które też nie wykraczają poza ramy harmonii klasycznej)
* Ogólna ocena współbrzmienia akompaniamentu z melodią główną (1 – akompaniament zupełnie nie pasuje i jest nieciekawy, sztuczny, 6 – świetnie dopasowany akompaniament naśladujący twórczość rzeczywistego kompozytora)

## Wyniki testów

## Ocena jakości działania programu MusicAnalyzer

# Podsumowanie

# Wykaz rysunków

[Rysunek 1 Dźwięki gamy C-dur, źródło http://3.bp.blogspot.com/-ZbOavfTrYJg/T1QALqWFwdI/AAAAAAAAADg/eC5xMCH9c6g/s1600/gama-c-dur-odleglosci.png 9](#_Toc461650056)

[Rysunek 2 Układ klawiszy fortepianu i odpowiadających im dźwięków na pięcioliniach, źródło: https://tucsonsongstress.files.wordpress.com/2012/11/staffnkeys.gif 10](#_Toc461650057)

[Rysunek 3 Klucz wiolinowy, źródło http://www.music-paper.com/ImagesClefs/treble-clef.jpg 10](#_Toc461650058)

[Rysunek 4 Klucz basowy, źródło: http://www.theyrenotourgoats.com/wp-content/uploads/2014/10/Bass\_Clef.jpg 10](#_Toc461650059)

[Rysunek 5 Skala molowa eolska w gamie na przykładzie gamy a – moll, źródło: http://3.bp.blogspot.com/-c1\_q-799WgM/T1QOY7uNJGI/AAAAAAAACjI/hpUDBezDOs4/s400/gama-a-moll-odleglosci.png 12](#_Toc461650060)

[Rysunek 6 Skala harmoniczna na przykładzie gamy a - moll, źródło: https://upload.wikimedia.org/wikipedia/commons/thumb/4/4b/Moll\_harm.jpg/600px-Moll\_harm.jpg 12](#_Toc461650061)

[Rysunek 7 Dźwięki należące do gamy C - dur zapisane w kluczu wiolinowym 13](#_Toc461650062)

[Rysunek 8 Akordy triady harmonicznej w gamie C - dur 14](#_Toc461650063)

[Rysunek 9 Znaki przy kluczu wiolinowym, źródło: https://upload.wikimedia.org/wikipedia/commons/thumb/8/84/A-major\_f-sharp-minor.svg/150px-A-major\_f-sharp-minor.svg.png 15](#_Toc461650064)

[Rysunek 10 Przykłady metrum i ich wyjaśnienie, źródło: http://e-perkusja.pl/plik/nauka\_gry\_na\_perkusji/Artykuly/Czytanie\_z\_nut/04\_Metrum\_gorna\_cyfra.jpg 15](#_Toc461650065)

[Rysunek 11 Schmeta działania Emily Howell, źródło https://www.google.com/patents/US7696426 17](#_Toc461650066)

[Rysunek 12 Schemat działania programu Mezzo, źródło http://www.danielbrownmusic.com/uploads/1/3/2/3/13234393/final\_dissertation\_final\_edit.pdf 20](#_Toc461650067)

[Rysunek 13 Interfejs programu DigiBand, źródło http://www.athtek.com/image/digiband/composition.jpg, strona odwiedzona 1.09.2016 21](#_Toc461650068)

[Rysunek 14 Komunikat o braku plików konfiguracyjnych w MusicAnalyzer 22](#_Toc461650069)

[Rysunek 15 Fragment zapisu nutowego pieśni "Kozak" S. Moniuszki z akompaniamentem pianina (c-moll). 25](#_Toc461650070)

[Rysunek 16 Tonacje i znaki przy kluczu wiolinowym, źródło http://blog.gitarius.pl/kurs\_gitarowy/chro4.png 26](#_Toc461650071)

[Rysunek 17 Podział wartości rytmicznych, źródło http://whittier.mpls.k12.mn.us/uploads/rhythm\_tree\_1.jpg 27](#_Toc461650072)

[Rysunek 18 Fragment partii tenora z chóralnej aranżacji utworu ludowego "Oberek powiślański" 29](#_Toc461650073)

[Rysunek 19 Trójdźwięk zapisany na pięciolinii, źródło http://www.codeproject.com/KB/miscctrl/psamcontrollibrary/examplechords.png, strona odwiedzona 1.09.2016 30](#_Toc461650074)

[Rysunek 20 Fragment zapisu nutowego z nutami o różnym kierunku 30](#_Toc461650075)

[Rysunek 21 Przykładowy zapis nutowy dwóch głosów w programie MusicAnalyzer 31](#_Toc461650076)

[Rysunek 22 Przykładowy wydruk zapisu nutowego akompaniamentu 33](#_Toc461650077)

# Wykaz tabel

[Tabela 1 Wartości rytmiczne przedłużane, źródło https://musictheorysite.files.wordpress.com/2015/06/dotted-value.png 28](#_Toc461578776)

# Wykaz literatury pomocniczej

**Brak źródeł w bieżącym dokumencie.**

# Odwołania

**Brak źródeł w bieżącym dokumencie.**

– użyć bibteX szukać przez google scholars

BibteX 2 Word

1. A. Poszowski, *Harmonia systemu tonalnego dur-moll*, Wydaw. Akademii Muzycznej w Gdańsku, Gdańsk 2001, s. 24 [↑](#footnote-ref-1)
2. *Zanim zaczniesz grać na... : ABC muzyki*, Polskie Wydawnictwo Muzyczne, Kraków 2002, s.12 [↑](#footnote-ref-2)
3. Andrzej Rakowski, *Percepcja wysokości dźwięku*, Państwowa Wyższa Szkoła Muzyczna, Warszawa 1978, s. 47 [↑](#footnote-ref-3)
4. *Zanim zaczniesz grać na...*, dz. cyt., s.14 [↑](#footnote-ref-4)
5. *Zanim zaczniesz grać na...*, dz. cyt., s.38 [↑](#footnote-ref-5)
6. Andrzej Rakowski, *Percepcja wysokości…, dz. cyt.,* s. 24 [↑](#footnote-ref-6)
7. https://pl.wikipedia.org/wiki/MIDI, (dostęp 14.09.2016) [↑](#footnote-ref-7)
8. https://pl.wikipedia.org/wiki/MIDI, (dostęp 14.09.2016) [↑](#footnote-ref-8)
9. *Zanim zaczniesz grać na...*, dz. cyt., s.7 [↑](#footnote-ref-9)
10. *Zanim zaczniesz grać na...*, dz. cyt., s.18 [↑](#footnote-ref-10)
11. https://pl.wikipedia.org/wiki/MIDI, (dostęp 14.09.2016) [↑](#footnote-ref-11)
12. https://pl.wikipedia.org/wiki/MIDI, (dostęp 14.09.2016) [↑](#footnote-ref-12)
13. Andrzej Rakowski, *Percepcja wysokości…, dz. cyt.,* s. 67 [↑](#footnote-ref-13)
14. Andrzej Rakowski, *Percepcja wysokości…, dz. cyt.,* s. 72 [↑](#footnote-ref-14)
15. Irena Poniatowska, *Dzieło muzyczne, teoria, historia, interpretacja*, PWM, Kraków 1984, s. 74 [↑](#footnote-ref-15)
16. *Zanim zaczniesz grać na...*, dz. cyt., s.26 [↑](#footnote-ref-16)
17. A. Poszowski, *Harmonia systemu tonalnego…*, dz. cyt., s. 37 [↑](#footnote-ref-17)
18. A. Poszowski, *Harmonia systemu tonalnego…*, dz. cyt., s. 43 [↑](#footnote-ref-18)
19. A. Poszowski, *Harmonia systemu tonalnego…*, dz. cyt., s. 46 [↑](#footnote-ref-19)
20. *Zanim zaczniesz grać na...*, dz. cyt., s.34 [↑](#footnote-ref-20)
21. A. Poszowski, *Harmonia systemu tonalnego…*, dz. cyt., s. 78 [↑](#footnote-ref-21)
22. Irena Poniatowska, *Dzieło muzyczne…, dz. cyt.*, s. 226 [↑](#footnote-ref-22)
23. Irena Poniatowska, *Dzieło muzyczne…, dz. cyt.*, s. 227 [↑](#footnote-ref-23)
24. Jacek Targosz, *Podstawy harmonii funkcyjnej*, PWM, Kraków 2011, s. 16 [↑](#footnote-ref-24)
25. Jacek Targosz, *Podstawy harmonii…*, dz. cyt., s. 19 [↑](#footnote-ref-25)
26. Jacek Targosz, *Podstawy harmonii…*, dz. cyt., s. 21 [↑](#footnote-ref-26)
27. Jacek Targosz, *Podstawy harmonii…*, dz. cyt., s. 25 [↑](#footnote-ref-27)
28. Irena Poniatowska, *Dzieło muzyczne…, dz. cyt.*, s. 186 [↑](#footnote-ref-28)
29. Witold Rudziński, *Nauko o rytmie muzycznym. C 1*, PWM, Kraków 1987, s. 14 [↑](#footnote-ref-29)
30. Witold Rudziński, *Nauko o rytmie…*, dz. cyt., s. 10 [↑](#footnote-ref-30)
31. Irena Poniatowska, *Dzieło muzyczne…, dz. cyt.,* s. 124 [↑](#footnote-ref-31)
32. Andrzej Rakowski, *Percepcja wysokości…, dz. cyt.,* s. 27 [↑](#footnote-ref-32)
33. G. Johnson, *Undiscovered Bach? No, a Computer Wrote It*, 11.11.1997, http://www.nytimes.com/1997/11/11/science/undiscovered-bach-no-a-computer-wrote-it.html?pagewanted=all, dostęp 20.08.2016 r. [↑](#footnote-ref-33)
34. http://artsites.ucsc.edu/faculty/cope/Emily-howell.htm, dostęp 20.08.2016 r. [↑](#footnote-ref-34)
35. Jacqui Cheng, *Virtual composer makes beautiful music - and stirs controversy*, 30.09.2009, http://arstechnica.com/science/2009/09/virtual-composer-makes-beautiful-musicand-stirs-controversy/, dostęp 20.08.2016 r. [↑](#footnote-ref-35)
36. https://www.google.com/patents/US7696426, dostęp 22.08.2016 r. [↑](#footnote-ref-36)
37. G. Johnson, *Undiscovered Bach…*, dz. cyt. [↑](#footnote-ref-37)
38. D. L. Brown, *Expressing narrative function in adaptive, computer-composed music*, 06.2012, http://www.danielbrownmusic.com/uploads/1/3/2/3/13234393/final\_dissertation\_final\_edit.pdf, dostęp 22.08.2016 r. [↑](#footnote-ref-38)
39. http://www.athtek.com/digiband/music-composition-software.html, strona odwiedzona 1.09.2016 [↑](#footnote-ref-39)
40. Zob. Punkt 4.7.1 niniejszej pracy [↑](#footnote-ref-40)
41. Zob. Punkt 4.7.1 niniejszej pracy [↑](#footnote-ref-41)
42. Zob. Punkt 4.7.1 niniejszej pracy [↑](#footnote-ref-42)
43. *Zanim zaczniesz grać na...*, dz. cyt., s.19 [↑](#footnote-ref-43)
44. A. Poszowski, *Harmonia systemu tonalnego…*, dz. cyt., s. 33 [↑](#footnote-ref-44)
45. *Zanim zaczniesz grać na...*, dz. cyt., s.23 [↑](#footnote-ref-45)
46. J. P. Bello, *MIDI Code*, https://www.nyu.edu/classes/bello/FMT\_files/9\_MIDI\_code.pdf (dostęp 14.09.2016) [↑](#footnote-ref-46)
47. Zob. Punkt 4.7.2 niniejszej pracy [↑](#footnote-ref-47)
48. *Zanim zaczniesz grać na...*, dz. cyt., s.27 [↑](#footnote-ref-48)
49. J. P. Bello, *MIDI Code*, https://www.nyu.edu/classes/bello/FMT\_files/9\_MIDI\_code.pdf (dostęp 14.09.2016) [↑](#footnote-ref-49)
50. Zob. Punkt 4.7.3 niniejszej pracy [↑](#footnote-ref-50)
51. Irena Poniatowska, *Dzieło muzyczne…, dz. cyt.*, s. 144 [↑](#footnote-ref-51)
52. Irena Poniatowska, *Dzieło muzyczne…, dz. cyt.*, s. 103 [↑](#footnote-ref-52)
53. http://www.codeproject.com/Articles/339416/Printing-large-WPF-UserControls, strona odwiedzona w dniu 08.09.2016 [↑](#footnote-ref-53)
54. Zob. Punkt 4.7.1 niniejszej pracy [↑](#footnote-ref-54)
55. https://www.midi.org/specifications/item/gm-level-1-sound-set, (dostęp 14.09.2016) [↑](#footnote-ref-55)
56. L. Sanford, *C# MIDI Toolkit*, 18.04.2007, http://www.codeproject.com/Articles/6228/C-MIDI-Toolkit, dostęp 14.09.2016 [↑](#footnote-ref-56)
57. Jacek Salamon, *PSAM Control Library*, 24.06.2010, http://www.codeproject.com/Articles/87329/PSAM-Control-Library, dostęp 22.03.2016 r. [↑](#footnote-ref-57)
58. Jacek Salamon, *PSAM WPF Control Library*, 24.06.2010, http://www.codeproject.com/Articles/89582/PSAM-WPF-Control-Library, dostęp 22.03.2016 [↑](#footnote-ref-58)