МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ФЕДОРЕНКО ЄВГЕНІЙ ОЛЕКСАНДРОВИЧ

Допускається до захисту:
В.о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____ Оксана ЗЕЛІНСЬКА
«_____» _____ 2024р.

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ГЕНЕРАЦІЇ МЕЛОДІЙ НА ОСНОВІ СИСТЕМИ КВАРТО-КВІНТОВОГО КОЛА

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (бакалаврська) робота

Керівник:
Павло РИМАР, старший викладач
кафедри інформаційних технологій

Оцінка: _____ / _____ / _____
(бали за шкалою ЄКТС / за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

АНОТАЦІЯ

Федоренко Є.О. Розробка застосунку для генерації мелодій на основі системи кварто-квінтового кола. Спеціальність 122 «Комп’ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

Основною метою виконання кваліфікаційної роботи є розробка алгоритму та застосунку, який дозволить автоматизувати процес створення музики, оскільки самостійне написання мелодій є трудомістким та вимагає специфічних знань. У першому розділі проводиться постановка задачі та огляд конкурентів. У другому розділі представлена теоретична база та процес створення алгоритму. У третьому розділі представлені технології застосовані при розробці та процес реалізації застосунку.

Ключові слова: Музика, Кварто-квінтового коло, Генерація мелодій, Windows Forms , Desktop application.

ABSTRACT

Fedorenko Y.O. Development of an application for generating melodies based on the circle of fifths system. Specialty 122 "Computer Science". Vasyl Stus Donetsk National University, Vinnytsia, 2024.

The main goal of the qualification work is to develop an algorithm and application that will automate the process of creating music, since writing melodies on your own is time-consuming and requires specific knowledge. The first section describes the problem statement and a review of competitors. The second section presents the theoretical basis and the process of creating the algorithm. The third section presents the technologies used in the development and implementation of the application.

Keywords: Music, Quarto-quintet circle, Melody generation, Windows Forms, Desktop application.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MIDI – (Musical Instrument Digital Interface) цифровий інтерфейс музичних інструментів.

ШІ – штучний інтелект.

WinForms – Windows Forms.

WPF – Windows Presentation Foundation.

DPI – (Dots per inch) роздільна здатність у точках на дюйм.

XML – (Extensible Markup Language) розширювана мова розмітки.

XAML – (Extensible Application Markup Language) розширювана мова розмітки для додатків.

JSON – (JavaScript Object Notation) запис об'єктів JavaScript.

UWP – (Universal Windows Platform) універсальна платформа Windows.

MVVM – (Model-View-ViewModel) модель- вигляд-модель вигляду.

UI – (User Interface) інтерфейс користувача.

UX – (User Experience) досвід користувача.

ВСТУП

Актуальність теми дослідження. На сьогоднішній день, інформаційні технології дозволяють автоматизувати процес створення музики та допомогти людям, які не мають спеціалізованої освіти у галузі музики, створювати власні композиції. У цьому контексті, розробка додатку для створення музики є важливою задачею, оскільки він може допомогти поширити музичну культуру та зробити її більш доступною для широкої аудиторії. Окрім цього, такий продукт буде корисний і для професійних музикантів для виконання рутинної роботи або пошуку нових ідей.

Мета – розробка алгоритму, який дозволить автоматизувати процес створення музики, забезпечивши користувачу можливість створювати прості мелодії на основі обраної тональності, розміру та темпу – параметрами що характеризують настрій та характер музичного твору.

Завдання дослідження:

- розглянути поняття генератора мелодій,
- провести аналіз існуючих аналогів, знайти їх переваги та недоліки й скомпонувати їх у вимоги до застосунку,
- розглянути основні закони милозвучності в мелодіях,
- створити математичну модель на основі розглянутих законів,
- спроектувати застосунок що відповідає заданим вимогам та імплементує алгоритм на основі створеної моделі.

Об’єкт дослідження: генерація мелодій на основі заданих параметрів.

Предмет дослідження: науково-теоретичні основи побудови музичного твору, їх математичне моделювання, алгоритмізація та імплементація алгоритму в застосунку

Апробація: результати дослідження здобули перемогу у «І турі Всеукраїнського конкурсу студентських наукових робіт з «Комп’ютерні науки» у

2022 - 2023 навчальному році». Також робота брала участь у міжнародному конкурсі студентських наукових робіт «Black Sea Science 2024».

Структура кваліфікаційної (бакалаврської) роботи – кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ ЗАСТОСУНКІВ

1.1 Постановка задачі

Мелодія це милозвучний ряд музичних нот, які складають основну частину пісні або музичного твору [1]. Милозвучність мелодій хоч і є суб'єктивною, проте музична теорія містить поняття правила що визначають загально прийняту милозвучність.

Генератор мелодій має використовувати закони музичної теорії для автоматичної побудови музичного ряду. Генератор має приймати параметри, які коротко формулюють настрій та характер очікуваного результату. Вони можуть задаватися використовуючи загальні ознаки описані побутовою мовою або поняття музичної теорії. Перший варіант не потребує опанування теорією, проте процес конкретизації результату є складнішим. Другий варіант надає користувачам більший контроль над результатом, проте він вимагає деяких специфічних знань. Також він є простіший в імплементації оскільки не вимагає системи переходу з абстрактних понять в реальні терміни, що можуть бути використані в алгоритмі. Через ці причини вважаю другий варіант більш підходящим для даної роботи. Також він може в подальшому слугувати плацдармом для реалізації генератору що працює на параметрах заданих загальними поняттями.

Генератор має мати інтуїтивний інтерфейс для комунікації з користувачем. Увесь процес від задання параметрів до прослуховування готової мелодії має бути швидким та зрозумілим.

1.2 Огляд існуючих аналогів

Suno – це генератор музичних творів що побудований на базі ШІ. Даний інструмент є онлайн сервісом. Своєю задачею він ставить максимально легке створення музики, що описується користувачем природньою мовою.

Рисунок 1.1 – Логотип онлайн сервісу «Suno»

Інтерфейс є мінімалістичним й побудований на основі поширеного вигляду додатків що виконують функцію музичного програвача. Таким чином користувачі одразу можуть зорієнтуватися на сайті та зрозуміти його функції.

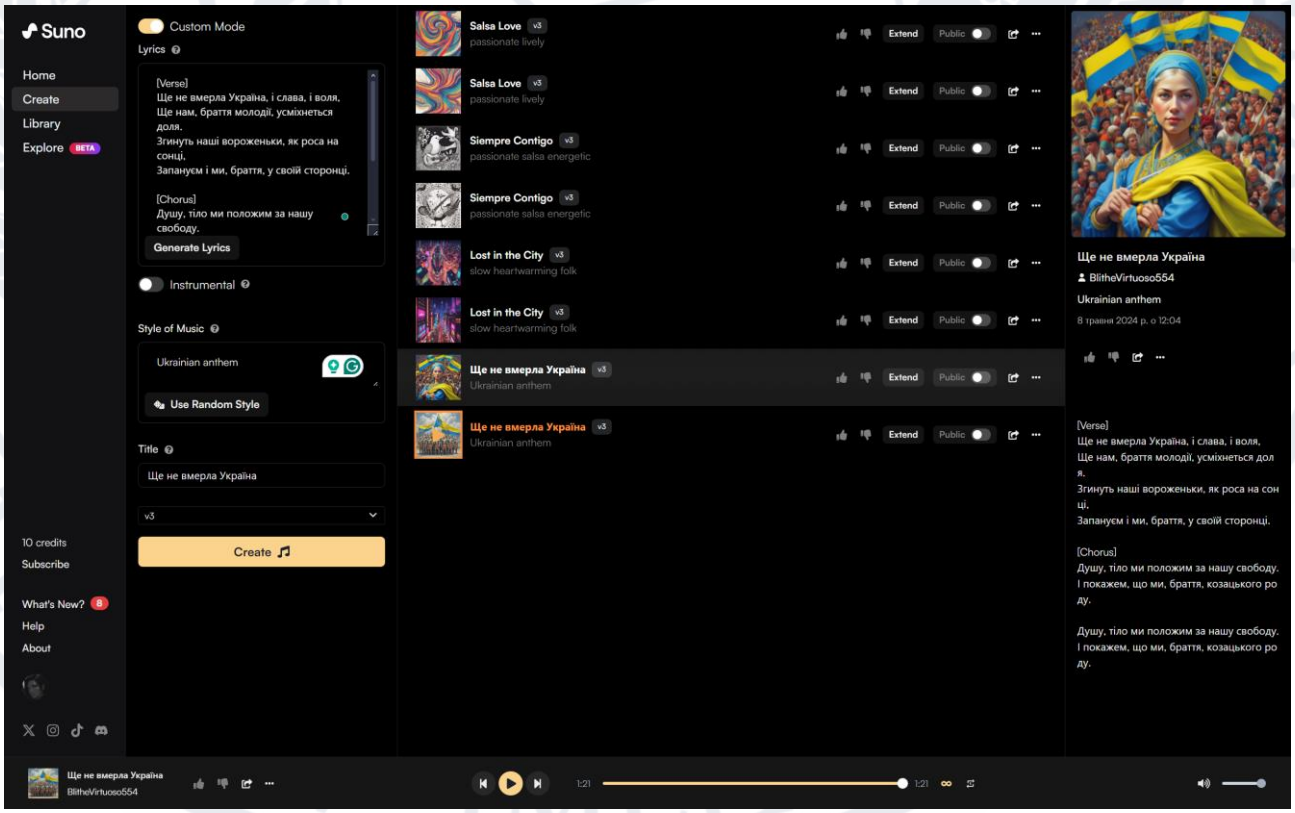

Рисунок 1.2 – Інтерфейс онлайн сервісу «Suno»

В якості вхідних параметрів генератор приймає текст пісні та жанр. Можна вказати назву пісні та перемкнути режим генерації з інструментальної (без вокалу) та не інструментальної (з вокалом).

Також є спрощений режим де замість тексту та жанру можна ввести загальний опис пісні природною мовою.

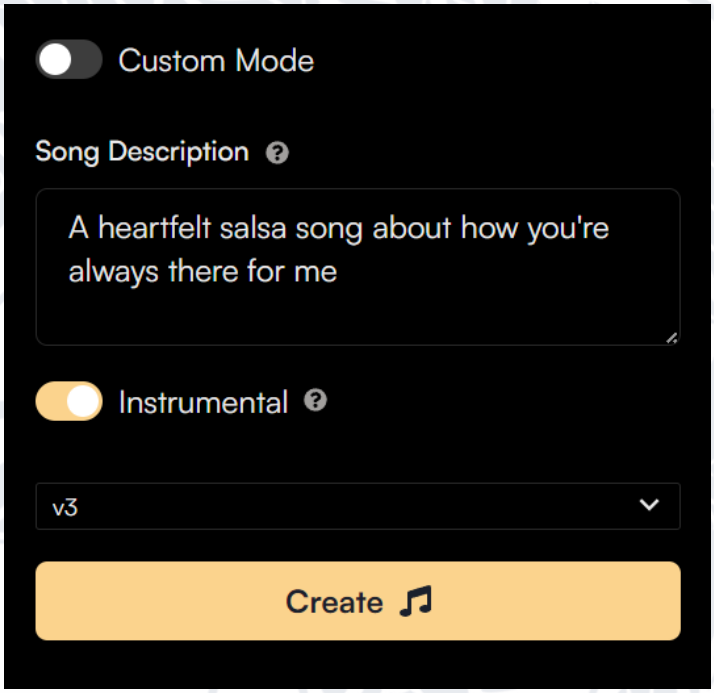

Рисунок 1.3 – Спрощений режим онлайн сервісу «Suno»

В результаті генерації ми отримуємо музичну композицію з кількома інструментами, вокалом (опціонально), та декількома частинами (наприклад куплету та приспіву).

Хоча сервіс є дуже простим й здатним до створення повноцінних пісень, він має кілька недоліків:

– Результат генерації є доволі нестабільним. Генератор може ігнорувати деякі деталі запиту. Наприклад, замість створення пісні у стилістиці гімну, була створена пісня у жанрі поп, що є протилежністю заданого запиту.

– Неможливо задати більш чіткі параметри для генерації. Якщо є потреба створити пісню із конкретним запитом, описаним музичною теорією, результат буде відрізнятися від очікуваного.

Random Music Generators – це онлайн сервіс що містить алгоримтічні генератори мелодій та ритмів. Позиціонує себе як інструмент для музикантів, що шукають незвичні композиції. Має велику кількість параметрів що описують аспекти мелодій мовою музичної теорії.

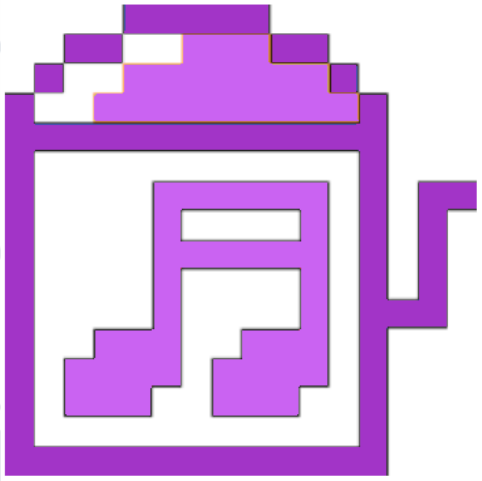

Рисунок 1.4 – Логотип онлайн сервісу «Random Music Generators»

Інтерфейс є навантаженим та несе за собою лише ціль відобразити усі інструменти. Дозволяє налаштувати тональність, максимальну різницю між сусідніми нотами, інтервал дозволених нот, темп, ритм, тривалість та інструмент. Ці параметри є корисними для музикантів, проте людям не знайомим із музичною теорією буде важко зрозуміти даний інтерфейс.

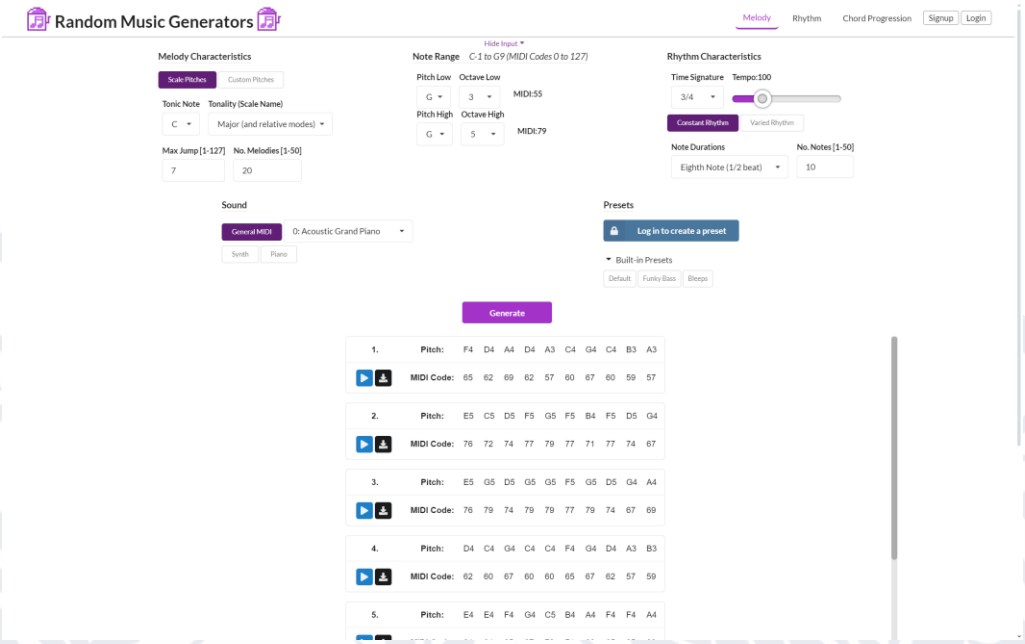

Рисунок 1.5 – Інтерфейс генератора мелодій сервісу «Random Music Generators»

Згенеровані мелодії точно підлаштовуються під задані параметри, проте не завжди є милозвучними. Також довжина мелодій є обмеженою. Результат можна прослухати використовуючи один із даних інструментів або продивитися та скачати у форматі MIDI (стандарт передачі та зберігання музики, розроблений для цифрових музичних синтезаторів [2]).

Окрім генератору мелодій на сайті також представлений генератор прогресій акордів. Він має схожий інтерфейс. Тут можна налаштувати тональність, максимальну різницю між сусідніми акордами, довжину повтору акорду, темп, ритм, тривалість та інструмент.

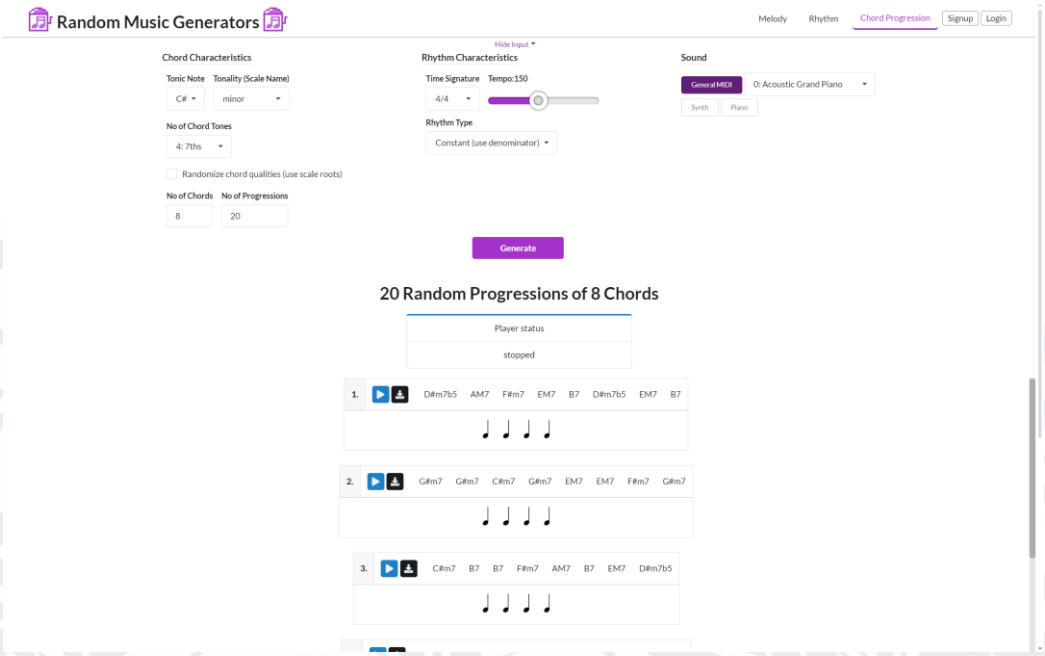

Рисунок 1.6 – Інтерфейс генератора прогресій акордів сервісу «Random Music Generators»

Результат роботи даного генератора є більш милозвучним, оскільки усі ноти в акорді по визначенню є гармонійними. В побудові цієї послідовності прослуховується більша системність. В загалом прослуховування результатів даного генератора є приємнішим.

Висновок до розділу 1

В якості аналогів були представлені представники двох різних підходів: алгоритмічного та з використанням машинного навчання. Перевага першого полягає в контрольованості, другого – в простоті використання. Вважаю що більш контрольований підхід має перевагу в використанні музикантами або науковцями, наприклад при контролі якості в навчанні ШІ.

З точки зору підходу до створення мелодій, перспективним виглядає поєднання генерації послідовності акордів та випадкового вибору окремих нот в рамках милозвучності. Цей підхід буде досліджений та описаний далі.

Інтерфейс застосунку має бути простим. Хорошою практикою є посилання на існуючі інтерфейси, такі як інтерфейс музичного програвача.

РОЗДІЛ 2
АЛГОРИТМ ГЕНЕРАЦІЇ МЕЛОДІЙ

2.1 Тональність як основа алгоритму

Основною задачею було знаходження системи, що давала би список нот (частот), які звучали би логічно в даному музичному контекст. Ідея цієї системи генерації мелодій побудована на концепті тональності. Тональність - це спосіб визначення того, які ноти та акорди можна використовувати в музиці. Кожна тональність має свою основну ноту, яка є найважливішою, і від неї залежать інші ноти та акорди, які можуть бути використані. Тональності можуть бути яскравими та світлими (мажорні) або темними та меланхолійними (мінорні) [3]. Знання про тональність допоможе зрозуміти, як певні ноти та акорди відносяться один до одного, та допоможе створити більш гармонійну музику. Для спрощення моделі, я використовував тональність саме в контексті акордів.

Акорди є основою музичних творів і використовуються в різних жанрах музики, від класичної до сучасної поп-музики. Це гармонічна комбінація трьох або більше нот, які зазвичай звучать одночасно і створюють акустичний ефект, що може викликати різні емоції у слухачів [4].

У музичній нотації, акорди позначаються спеціальними символами, які показують ноти, які входять до складу акорду. Наприклад, C-акорд, який є одним з найпростіших акордів, складається з нот C, E і G, і позначається як "C" [4].

По своїй суті акорди можуть бути сприйняті як колекція чисел або символів, що представлені відповідними кодами. Для прикладу, C-акорд може бути представлений як набір чисел 60, 64, 67, що відповідають кодам MIDI нот.

Для того щоби в вибраній тональності з доступних акордів створити правильну їх послідовність, необхідно додати ще один базовий музичний концепт: гармонію.

Гармонія - це сукупність звуків в музиці, які зазвичай звучать одночасно та створюють музичну основу для мелодії. Гармонія в музиці може бути створена

за допомогою різних акордів та їх комбінацій, що забезпечують певну сукупність звуків та емоційний вплив на слухача.[5]

У музиці, гармонія може бути описана як відношення між нотами та акордами, які створюють певну логіку та співвідношення між ними. Гармонія може включати в себе використання різних ступенів напруження та розв'язання, які створюють музичну динаміку та сприяють емоційному впливу на слухача.

Таким чином, використовуючи задану тональність як систему допустимих акордів, гармонію як систему послідовностей акордів та самі акорди як систему нот, я отримав модель, що показує які ноти можна використовувати в кожний даний момент часу, а використання яких потрібно запобігати.

2.2 Кварто-квінтове коло як система

Кварто-квінтове коло - це система, яка використовується у музичній теорії для опису відносин між нотами та акордами. Це коло складається з семи основних нот (C, D, E, F, G, A, B), які розташовані на колі в певному порядку.[6]

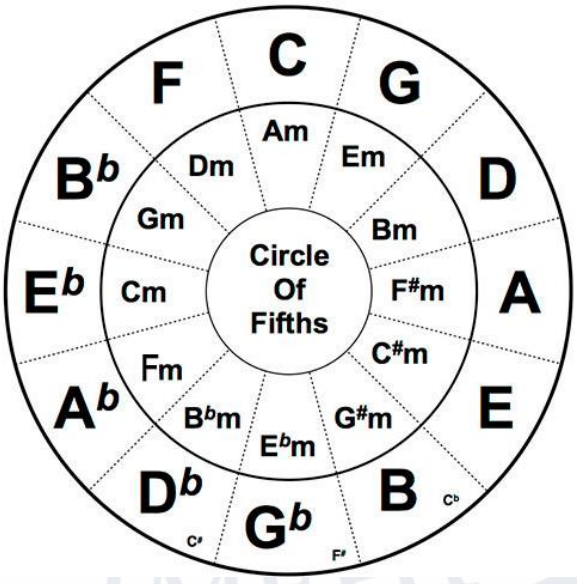

Рисунок 2.1.1 – Кварто-квінтове коло

Кварто-квінтове коло використовується для визначення тональності та побудови акордів у музиці. На колі, нота С знаходиться в верхній частині, а нота В - у нижній. Далі, у напрямку годинникової стрілки, розташовуються інші ноти квартовими та квінтовими інтервалами.

Акорди, які знаходяться поруч на колі, мають схожі звуки та можуть легко поєднуватись у музиці. Наприклад, акорди С, Е та G знаходяться поруч на кварто-квінтовому колі та можуть використовуватись у відповідних тональностях, щоб створити музику зі спокійним та приємним звучанням. Акорди, які розміщені на протилежних боках кола, можуть мати більш дисонансний звук, але можуть додавати більшу напругу та динаміку до музичного твору. Кварто-квінтове коло є важливим елементом музичної теорії, який допомагає розуміти взаємозв'язки між нотами та акордами у музиці.

На квінтовому колі можна виділити три основні акорди - тонічний, домінантний та субдомінантний.

Тонічний акорд розташований на початку квінтового кола, або на нижньому кінці діаметра. Цей акорд є домінантним для своєї тоніки і зазвичай звучить стабільно та спокійно. Найчастіше тонічний акорд є основним акордом у піснях та мелодіях.

Домінантний акорд зазвичай розташовується справа від тонічного. Цей акорд є напруженим та нестабільним, його звучання часто потребує розв'язки у тонічний акорд. Домінантний акорд часто використовують для створення напруження в музичних творах та для підсилення емоційної виразності.

Субдомінантний акорд зазвичай розташовується зліва від тонічного. Цей акорд має менш напружене звучання порівняно з домінантним акордом та може використовуватися для створення відносно спокійної та гармонійної музики. Проте він також потребує розв'язки у тонічний акорд.[6]

Один з підходів до генерації мелодій полягає в тому, щоб спочатку вибрати тоніку, тобто відповідний тон, на якому буде базуватися мелодія, і потім

створювати мелодію, використовуючи ноти та акорди, які відповідають цій тональності на квінтовому колі.

Таким чином, при генерації мелодій можна використовувати домінантні та субдомінантні акорди, щоб створити більш напружені та емоційно насичені моменти у мелодії, а також використовувати тонічні акорди, щоб створити спокійніші та більш гармонійні моменти у мелодії.

Крім того, можна використовувати різні варіації домінантних та субдомінантних акордів, наприклад, замінювати їх на сус-акорди або на альтернативні варіанти, щоб створити більш цікаву та різноманітну гармонію у мелодії.

Враховуючи відносини між акордами на квінтовому колі та їх вплив на гармонію та емоційний настрій музики, можна створити більш складні та виразні мелодії з використанням різних варіацій нот та акордів.

Усі ці залежності є дійсними для будь-яких обраних тональностей та акордів, що дозволяє використовувати їх у алгоритмізації написання мелодій.

2.3 Розташування звуків в часовому просторі

Маючи систему для визначення допустимих звуків, наступним кроком є розташування цих звуків у часі.

Розмірність - це музичний термін, який описує кількість інтервалів часу в музичному творі, що складаються з нот і пауз. Розмірність може бути представлена за допомогою раціональних чисел, які відображають відношення між різними ритмічними фігурами.

Наприклад, розмірність 4/4 означає, що кожен такт (цикл нот, що має визначену тривалість часу та кількість нот) музичного твору складається з чотирьох довжин часу, які можуть бути заповнені різними нотами та паузами. Розмірність може бути також позначена іншими числами, наприклад, 3/4, 6/8, 5/4 тощо.[7]

Для налаштування цього параметру в додатку я використовую дві величини: кількість нот на інтервал та кількість інтервалів на такт.

Розмірність впливає на те, як музичний твір звучить та який він має характер. Наприклад, розмірність 4/4 створює враження ритмічної стабільності, що може надати музиці ритмічний та енергійний характер. З іншого боку, розмірність 3/4 відчувається як менш енергійна і дозволяє створювати музику з більш меланхолійним та витонченим звучанням.

Розмірність може вплинути на музичний твір також тим, як ноти та паузи розташовані в межах кожного такту. Наприклад, якщо ритмічні фігури повторюються рівномірно протягом кожного такту, то звучання музики може бути рівномірним та одноманітним. Якщо ж ритмічні фігури в такту змінюються, це може створити більш складну та цікаву музичну текстуру.

Таким чином, розмірність може вплинути на характер звучання музики, дозволяючи створювати різні настрої та емоційний вплив на слухача. Врахування розмірності в музичному творі може допомогти зробити його більш цікавим, витонченим та виразним.

Окрім розмірності, можна налаштувати і темп – швидкість, з якою музичний твір відтворюється. Він може бути швидким, середнім або повільним. Якщо темп дуже швидкий, то музика звучить енергійно та жваво, а якщо повільний – то мелодія може бути більш заспокійливою та меланхолійною.

2.4 Аналіз необхідного функціоналу

Автоматична генерація мелодій може бути корисною для різних цілей та застосувань.

Наприклад, в сфері музичної композиції автоматична генерація мелодій може допомогти композиторам швидко створювати прототипи та ескізи музичних творів, змінювати темп, ритм та інші параметри музики. Вона може також допомогти знаходити нові музичні ідеї та гармонії, що можуть стати основою для створення повноцінних музичних композицій.

Автоматична генерація мелодій також може бути корисною у сфері відеоігор. Вона може допомогти створювати відповідну атмосферу в грі, підсилювати напруження та динаміку геймплею, а також допомагати генерувати різні музичні теми для різних рівнів гри чи епізодів.

Також автоматична генерація мелодій може бути корисною для використання у рекламних та медіа-проектах, наприклад, в якості фонової музики для відеороликів, презентацій або підкастів.

Підсумовуючи, в різних ситуаціях потрібна генерація мелодій які передаватимуть слухачеві різні емоції, почуття, а також різний характер: швидкий, спокійний, фоновий або той, що привертає на себе увагу. Тому змінюючи параметри додатку, користувач має отримувати результати, які відрізнятимуться по описаним вище критеріям. Для цього в програмі дозволяється змінювати аспекти мелодій, про які йшла мова в попередньому розділі, а саме: тональність, розмірність та ритм. Додатково є можливість налаштування гучності.

Таким чином тональність відповідає за передачу потрібної емоції. Вона може бути мажорною чи мінорною, передаючи відповідні стани, а також може мати різні тонічні ноти/акорди, що можуть більше підходити під той чи інакший контекст. Також це дозволяє гармонійно поєднувати створену мелодію з іншими інструментами, що гратимуть в тій самій тональності.

Розмірність, темп та гучність більше впливають на характер звучання. Різні комбінації можуть давати пряму, зрозумілу, звичну слухачам картину, або навпаки ударити незвичним перебором нот, що одразу буде вирізнятися.

Для визначення тривалості мелодії, користувач може встановлювати кількість тактів, повторюваних інтервалів. Використання звичного часу в секундах тут не може бути використано, оскільки це може зламати структуру композиції. Час фактично буде залежати від цієї кількості тактів в комбінації з темпом, що визначає швидкість їх зміни.

2.5 Алгоритм підбору нот мелодії

Моєю метою було створення базового алгоритму, що створюватиме композицію з одного музичного інструменту, щоби вже надалі мати потенціал мульти-інструментальності та менш тривіальних мелодій на основі створеної бази. Задача генерації мелодію по своїй суті є задачею підбору послідовності нот. Основна складність полягає в збереженні милозвучності композиції, через що ми не можемо випадкові послідовності під час генерації. В попередньому розділі ми визначили що тональність визначає систему акордів, акорди визначають систему нот, а кварто-квінтове коло є описом цієї системи.

Почнемо опис системи з тональності, оскільки для визначення нот потрібно спочатку визначити акорди. Першим акордом мелодії варто використовувати тоніку щоби слухач одразу розумів контекст для подальшої мелодії та не відчував дисонансу. Коли тоніка задана, ми можемо використовувати будь-які інші акорди, що належать даній тональності. У випадку мажорної тоніки, це сусідні акорди справа, зліва, а також акорди, що знаходяться під ними. У випадку мінорної - акорди справа, зліва та над ними. На початку кожного такту тонічний акорд буде повторюватися щоби зберегти цілісність тональності та яскравіше окреслити розмірність такту. Останнім акордом такту оптимально зробити домінанту або субдомінанту, оскільки вони найбільше тяжіють до тоніки, що очікується на початку наступного такту. Також потрібно слідкувати за послідовністю акордів та запобігати повторів, аби мелодія не здавалася монотонною.

Даний алгоритм здатний визначити гармонію мелодії. Потрібно лише циклічно його повторювати поки ми не досягнемо ліміту тактів, визначеного користувачем. Оскільки кожний такт закінчується акордом що тяжіє до тоніки, ми не можемо просто обірвати послідовність акордів на цьому. В такому випадку в слухача виникатиме відчуття незавершеності. Тому після генерації усіх тактів, наприкінці обов’язково звучатиме тоніка.

Робота з нотами доволі схожа до описаного вище. В кожного акорду є своя основна тонічна нота. Вона є однойменною з назвою відповідного акорду. Окрім неї, є ще дві додаткові ноти, що визначають характер акорду. Також, щоби розширити кількість нот, з якою ми працюємо, в програмі використовується тоніка на одну октаву вище. Це така нота, що має вищу частоту, проте має такий самий музичний контекст.

Послідовність нот на інтервалі акорду починається з тоніки в різних октавах, щоби точно окреслити акорд, що грається. Це випадок, коли грається кілька нот одночасно, що є індикатором початку нового інтервалу в такті. Далі у випадковому порядку обираються інші ноти акорду, поки інтервал не закінчується. Після цього з алгоритму вище обирається наступний акорд та процес повторюється.

В такий спосіб гарантується милозвучність мелодії та явно окреслюється музичний контекст, через що композиція відчувається цілісною та правильною.

2.6 Математична інтерпретація нотних систем

Описані вище зв’язки між нотами можуть бути представлені математично. Усі ноти відрізняються один від одного на один півтон. Усього є 12 нот. 13-та нота у такому випадку буде контекстуально рівна 1-ій, але у вищій октаві. Так, можемо сказати, що нота С у першій октаві + 12 півтонів дає нам ноту С у другій октаві.[8]

Акорди, що складаються з 3 та більше контекстуально різних нот, можна побудувати, використовуючи наступні формули:

$x_1 = t + 12n;$

$x_2 = t + 4 + 12n$ для мажорних акордів

або $x_2 = t + 3 + 12n$ для мінорних акордів;

$x_3 = t + 7 + 12n;$

де x_i – нота, виражена в порядковому номеру ноти в півтонах, починаючи з С першої

октави; t – тонічна (основна) нота; n – натуральне число.

Знаходження нот акорду

Таким чином, рахуючи, що тоніка (основна нота) акорду співпадає з його ім'ям, ми отримуємо можливість знайти усі інші його ноти, що можна використовувати.[8]

У випадку з кварто-квінтовим колом, теж спостерігається деяка математична закономірність. Кожна наступна тоніка акорду відрізняється від попередньої на 7 півтонів. Також для мажорних акордів щоби знайти тоніки акорду знизу, потрібно відняти 3 півтони, а для мінорних щоби знайти тоніку акорду зверху, треба додати 3 півтони.[6] Таким чином, маючи тонічний акорд тональності, ми можемо знайти інші доступні акорди математично:

$x_1 = t + 12n;$ (тонічний акорд)

$x_2 = t + 7 + 12n;$ (домінантний акорд)

$x_3 = t - 7 + 12n;$ (субдомінантний акорд)

$x_4 = t - 3 + 12n$ для мажорних тональностей або $x_4 = t + 3 + 12n$ для мінорних;

$x_5 = x_4 + 7 + 12n;$

$x_6 = x_4 - 7 + 12n;$

де x_i – тонічна нота відповідного акорду; t – тонічна нота тонічного акорду; n – натуральне число.

Формула 2.2.2 – знаходження акордів тональності

2.7 Математичне представлення алгоритму

Користувач задає тонічний акорд. З нього ми маємо його тонічну ноту x_1. З формули 2.2.2 отримуємо інші акорди тональності, які об'єднаємо у множину $C = \{c_1; c_2; c_3; c_4; c_5; c_6\}$. Для кожного з цих акордів ми знаємо усі ноти, що

використовуватимуться з формули 2.2.1. Їх представимо у множині $N_i = \{n_1; n_2; n_3; n_4;\}$, де i – відображає номер відповідного акорду, а додаткова нота $n_4 = n_1 + 12$.

Також маємо кількість тактів – t, кількість інтервалів на такт - i, та кількість нот на інтервал - m.

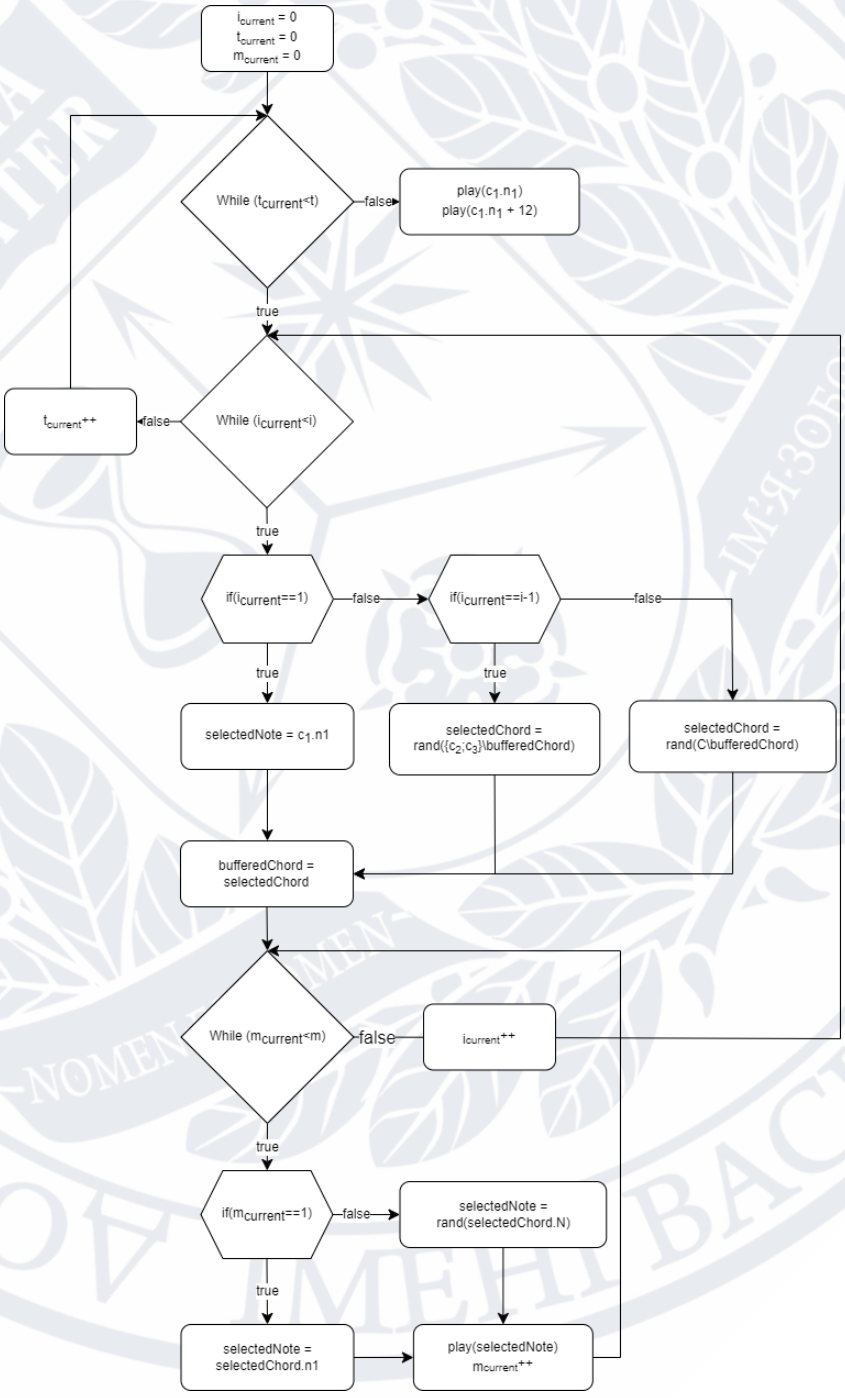

Рисунок 2.2.1 – Алгоритм генерації мелодій

Висновок до розділу 2

Деякі поняття музичної теорії були виведені у математичну модель. Вона може використовуватися у будь-яких застосунках для роботи над музикою у вигляді окремих нот чи акордів. Наприклад, розумні підказки, перевірка на милозвучність та, власне, генерація мелодій.

Створений на основі цієї моделі алгоритм створює мелодії у вигляді послідовності нот. Створені мелодії завжди є милозвучними.

Надалі ця модель та алгоритм можуть бути покращені шляхом введення нових музичних понять, наприклад новими формами акордів.

РОЗДІЛ 3
ЗАСТОСУНОК ЩО РЕАЛІЗУЄ АЛГОРИТМ ГЕНЕРАЦІЇ МЕЛОДІЙ

3.1 Фреймворк .NET

Додаток націлений на просту демонстрацію використання алгоритму на платформі Windows. Тому для розробки був обраний фреймворк .NET.

.NET - це платформа з відкритим вихідним кодом для створення десктопних, веб та мобільних додатків, які можуть працювати на будь-якій операційній системі. Система .NET включає інструменти, бібліотеки та мови, які підтримують сучасну, масштабовану та високопродуктивну розробку програмного забезпечення. Активна спільнота розробників обслуговує та підтримує платформу .NET.

Рисунок 3.1.1 – Логотип фреймворку .NET

.NET виконує такі завдання:

– Перекладає код мови програмування .NET в інструкції, які може обробити обчислювальний пристрій.
– Надає утиліти для ефективної розробки програмного забезпечення. Наприклад, визначати поточний час або виводити текст на екран.

– Визначає набір типів даних для зберігання на комп'ютері такої інформації як текст, числа і дати.

NET має модульну та оптимізовану архітектуру. Користувачі можуть вибирати різні компоненти, щоб задовольнити свої вимоги до розробки програмного забезпечення.

Основні компоненти .NET:

– Мови .NET
– Фреймворки моделей додатків
– Середовище виконання .NET

Розробники використовують мови програмування .NET та фреймворки моделей додатків для створення своїх .NET-додатків. Потім середовище виконання .NET виконує і запускає їх. [9]

3.2 Мова програмування C#

C# – це об'єктно-орієнтована мова програмування, створена компанією Microsoft для розробки додатків на платформі .NET. Вона дозволяє розробникам використовувати широкий спектр .NET бібліотек та інструментів для створення потужних, високоякісних додатків. [10]

Рисунок 3.1.2 – Логотип мови програмування C#

Переваги використання мови C#:

- **Легка крива вивчення:** Це мова високого рівня з легкою кривою вивчення, що робить її доступною для програмістів усіх рівнів кваліфікації.
- **Безпека:** Мова має вбудовані функції безпеки, які допомагають захистити додаток від атак та вразливостей.
- **Швидка розробка:** Розробники можуть писати код швидше та зменшити кількість помилок і багів.
- **Продуктивність:** він пропонує високопродуктивні інструменти розробки, такі як IntelliSense, та розширені засоби налагодження, що полегшують написання та налагодження коду.
- **Крос-платформна підтримка:** Сприяє створенню додатків для Windows, Linux та macOS.
- **Сильна спільнота:** Сильна та активна спільнота розробників, яка надає підтримку, ресурси та рішення спільних проблем.

Недоліки використання мови C#:

- **Нижча продуктивність, ніж у нативних мов:** менш ефективна, ніж мови, скомпільовані у машинний код.
- **Накладні витрати на виконання:** віртуальна машина Common Language Runtime (CLR) може спричинити деякі додаткові витрати під час виконання.
- **Менш придатна для систем з низьким енергоспоживанням:** не ідеально підходить для високооптимізованих додатків на пристроях з низьким енергоспоживанням. [10]

3.3 Вибір графічної бібліотеки

Windows Forms - це фреймворк інтерфейсу користувача для створення настільних додатків для Windows. Це один з найефективніших способів створення десктопних додатків на основі візуального дизайнера, що входить до

складу Visual Studio. Завдяки таким функціям як перетягування візуальних елементів керування, створювати десктопні програми є нескладною задачею. [11]

Рисунок 3.1.3 – Логотип бібліотеки WinForms

До переваг Windows Forms відносяться:

- **Простота процесу розробки:** за допомогою Windows Forms дуже легко розробляти десктопні програми для Windows. Процес розробки спрощується завдяки інструменту візуального проектування за допомогою перетягування, відомому як Конструктор Windows Forms. Все, що вам потрібно зробити, це вибрати елементи керування за допомогою курсору і розмістити їх там, де ви хочете. Дизайнер має такі інструменти, як лінії сітки та лінії прив'язки, тому вам не доведеться турбуватися про вирівнювання елементів керування. Розробники можуть не лише легко створювати користувацькі інтерфейси, але й легко проектувати макети форм, елементів керування, компонентів тощо, що полегшує процес розробки. Крім того, існують такі інструменти, як TableLayoutPanel та елементи керування SplitContainer, які допомагають швидко створювати складні макети форм.
- **Простота відображення та маніпулювання даними:** дуже легко відображати та маніпулювати даними з бази даних, XML веб-сервісу, JSON-файлу або будь-якого іншого подібного джерела даних. Тип проекту поставляється з функцією DataGridView, яка дозволяє

відображати дані в традиційній табличній формі, де кожному дню відповідає своя комірка.

- **Багатий набір елементів керування:** у Windows Forms існує величезна бібліотека готових елементів управління. Ці елементи управління можуть бути налаштовані відповідно до конкретного типу програми, що дозволяє розробникам створювати багатофункціональні додатки з вражаючим набором можливостей, причому розробникам не потрібно створювати елементи управління з самого початку.
- **Інтеграція з .NET:** інтеграція з фреймворком .NET забезпечує надійну основу для створення додатків для Windows. Таким чином, розробники можуть використовувати численні можливості платформи .NET для створення надійних і масштабованих додатків.
- **Узгодженість з іншими додатками Windows:** Windows Forms має набір елементів що звичний для користувачів Windows. Це надає додатку звичний користувацький інтерфейс, що полегшує адаптацію при користуванні.
- **Стабільність і надійність:** Windows Forms відома своєю стабільною. Фреймворк також дуже надійний, враховуючи той факт, що він був випробуваний і протестований протягом багатьох років.
- **Гнучкість та розширюваність:** Windows Forms є дуже гнучким та розширюваним. Розробники можуть розширювати функціональність додатків за допомогою кастомних елементів управління.
- **Простий у вивченні:** Windows Forms легко вивчити. Крім того він має велику спільноту розробників та багато ресурсів в Інтернеті, таких як навчальні посібники та документація.

З недоліків можна визначити:

- **Є старою технологією:** з появою нових технологій багато розробників вважають, що додатки Windows Forms застаріли, а користувацькі

інтерфейси, побудовані за допомогою сучасних технологій, є більш привабливими.

- **Не підходить для мобільних додатків:** оскільки Windows Forms не працює на смартфонах і планшетах, його конкурентність з іншими технологіями падає.
- **Більші потреби у пам’яті:** Windows Forms додатки займають більше пам'яті, ніж додатки побудовані на WPF (сучасний аналог). Ця різниця може бути пов'язана з різними типами рендерингу які вони використовують.
- **Відсутність розвитку у майбутньому:** Windows Forms все ще використовується, але Microsoft приділяє більше уваги іншим технологіям, таким як UWP (Universal Windows Platform) і WPF. Тому оновлення для Windows Forms можуть бути рідшими у майбутньому. Це може спричинити проблеми з підтримкою та сумісністю в майбутньому.[12]

WPF – це фреймворк інтерфейсу користувача для створення настільних додатків для Windows.

Рисунок 3.1.4 – Логотип бібліотеки WPF

Рендеринг WPF є векторним, що дозволяє додаткам чудово виглядати на моніторах з високим DPI, оскільки вони можуть бути нескінченно масштабовані.

Режим дизайну в Visual Studio, а також Visual Studio Blend дозволяють легко створювати WPF-додатки за допомогою перетягування та/або прямого редагування розмітки XAML.[13]

Переваги WPF:

- **Просунутий інтерфейс користувача:** завдяки візуально проробленому та високоінтерактивному користувацькому інтерфейсу розробка є прямолінійною та зрозумілою, а широка підтримка технології через мультимедіа, анімацію, візуалізацію даних, векторну графіку і т.д. дає змогу забезпечити високий рівень візуальної насиченості.
- **Тісна мультимедійна інтеграція:** фреймворк забезпечує широку мультимедійну інтеграцію, що дозволяє розробникам використовувати декілька незалежних технологій разом.
- **Використання XAML:** WPF відомий тим, що використовує XAML для визначення користувацького інтерфейсу, прив'язки даних, елементів інтерфейсу та подій. Це дає розробникам свободу у створенні та проектуванні елементів інтерфейсу, а також спрощує співпрацю між розробниками та дизайнерами.
- **Чітке розділення між бізнес-логікою та інтерфейсом користувача:** Однією з цілей Microsoft при створенні WPF було гнучке поєднання бізнес-логіки та користувацьких інтерфейсів в додатку без змішування відповідальності у коді. Фреймворк дозволяє використовувати такий патерни проектування як MVVM (Model-View-View-Model).

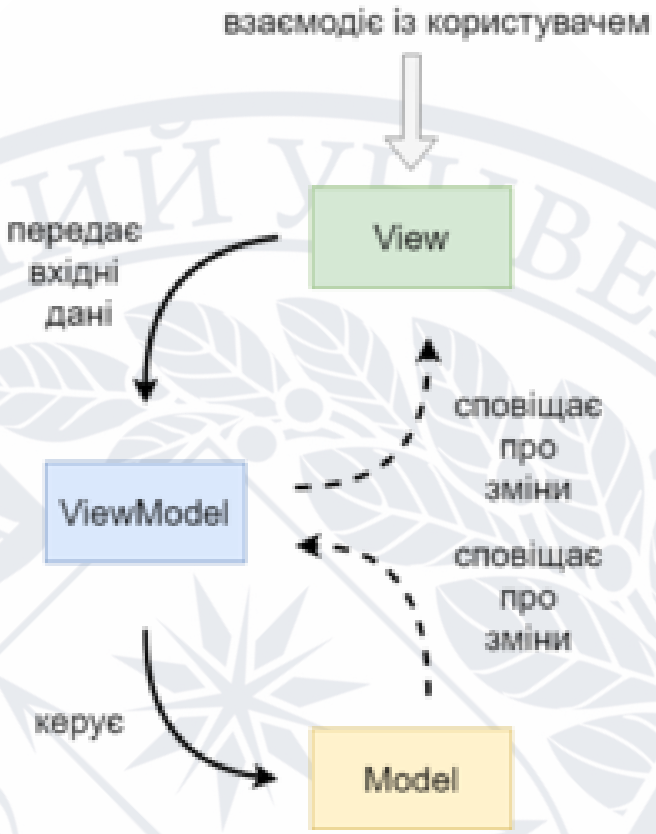

Рисунок 3.1.5 – Діаграма взаємодії між компонентами шаблону MVVM

– **Доступ до анімацій:** анімація потрібна для того, щоб привернути увагу користувача, а також для того, щоб зробити додаток зручним для користувача. WPF реалізує їх у XAML, де створюється простий опис руху без необхідності писати код.
– **Підтримка від третіх сторін та спільнот розробників:** WPF має активну спільноту програмістів та велику кількість сторонніх постачальників, що дозволяє отримати велику кількість додаткових елементів та функцій без необхідності їх самостійної імплементації.
– **Широкі можливості кастомізації та тематизації:** WPF добре відомий своїми можливостями кастомізації, гнучкості та тематизації користувацьких інтерфейсів. Розробник може визначати стилі, шаблони і теми.

Недоліки включають в себе:

- **Крута крива вивчення:** WPF має криву вивчення, що є крутішу за інші фреймворки графічного інтерфейсу. Новому розробнику може знадобитися деякий час, щоб зрозуміти концепції та освоїтися з концепціями, XAML і наявної MVVM структури.
- **Не забезпечує крос-платформну сумісність:** оскільки WPF в першу чергу призначений для додатків на базі Windows, його використання у крос-платформених додатках є неможливим.
- **Розрахунок на апаратне прискорення:** хоча апаратне прискорення є перевагою WPF і дає змогу отримати більш плавну анімацію та покращену швидкість відгуку, воно також має і недоліки. Таке апаратне прискорення може бути ресурсоємним і забирати занадто багато обчислювальної потужності та пам'яті. Це може бути шкідливим для додатків що цілять на старі або слабкі комп’ютери.
- **Великий розмір додатків:** додатки WPF є значно більші за розміром. При завантаженні вони можуть мати потребу і більшій пропускній здатності та обсягу оперативної пам’яті. Це є ще одним фактором повільної роботи на машинах без графічних прискорювачів і на машинах нижчого цінового класу.[14]

Опираючись на вище описане порівняння та задачу у створенні простого але функціонального прототипу, була обрана бібліотека Windows Forms. Вона не потребує додаткового вивчення мови розмітки XAML, має простішу криву вивчення в загальному, є оптимальною для більшої кількості комп’ютерів та надає увесь мінімально необхідний функціонал для додатку.

3.4 Середовище розробки Visual Studio

Microsoft Visual Studio - це повнофункціональне інтегроване середовище розробки (IDE) для програмування, налагодження, тестування та розгортання додатків.

Рисунок 3.1.6 – Логотип середовища розробки Visual Studio

Платформа забезпечує контроль версій, гнучке планування та управління додатками для великих команд. Microsoft Visual Studio також має версію для індивідуальних програмістів, яка дозволяє їм безкоштовно працювати над кодом. Платформа має редактор коду, який підтримує IntelliSense (інструмент для автоматичного завершення коду), а також рефакторинг коду. Вона також має вбудовані інструменти, такі як конструктор форм для створення графічних додатків, веб-дизайнер, конструктор класів і конструктор схем баз даних. Microsoft Visual Studio підтримує різні мови програмування, у тому числі C#. [15]

До основних функцій входить:

– **Розробка:** платформа дозволяє розробникам писати код з меншою кількістю помилок. Завдяки підказкам IntelliSense користувачі можуть швидко і точно вводити змінні за допомогою коду. Програмісти можуть вести розробку швидко, незалежно від складності коду. Програмісти можуть покращувати свій код за допомогою лампочок, які підказують дії, наприклад, перейменування функції або додавання параметра.
– **Аналіз:** Microsoft Visual Studio дозволяє програмістам дізнатися більше про свій код. Функція CodeLens допомагає знайти важливу інформацію, таку як зміни, внесені до коду, вплив цих змін і те, чи був метод протестований. Користувачі можуть швидко знайти посилання, авторів, тести, історію коментарів та іншу важливу інформацію. Користувачі

можуть переміщатися по коду, щоб знаходити типи, відкривати файли та ідентифікувати будь-яке місце в коді, де є посилання на тип. Аналіз коду платформи перевіряє код C# та Visual Basic на предмет стилю, якості, підтримки та інших якостей. Аналіз коду виконується під час проектування для всіх відкритих файлів, тому користувачі отримують негайний зворотній зв'язок про порушення коду під час набору коду і застосовують один з рефакторингів, швидких дій та виправлень коду, щоб позбутися порушення.

- **Дебагінг:** Microsoft Visual Studio дозволяє програмістам призупинити виконання коду в той момент, коли вони хочуть перевірити помилку, використовуючи точку зупинки. Користувачі також можуть повернутися до будь-якого конкретного рядка коду без необхідності перезапускати сеанси або відтворювати стани. Розробники можуть переглядати інформацію про змінні в редакторі під час дебагінгу і використовувати підказки, щоб побачити ім'я і поточне значення змінної, розгорнути об'єкт і побачити його елементи, а також відредагувати значення змінної. Користувачі можуть перевіряти проблеми в автономному режимі у виробничому середовищі, використовуючи такі можливості, як IntelliTrace і глибокий аналіз дамп файлів.
- **Інструменти для тестування:** Платформа дозволяє програмістам писати високоякісний код за допомогою комплексних інструментів тестування. Розробники можуть писати, виконувати та налагоджувати модульні тести обраною мовою та тестовим фреймворком. Microsoft Visual Studio має набір вбудованих шаблонів проектів і тестових фреймворків, які підтримують різні платформи. Функція IntelliTest може зменшити зусилля зі створення та підтримки модульних тестів для нового або існуючого коду. Користувачі можуть генерувати цікаві вхідні-вихідні значення для своїх методів і зберігати їх у вигляді невеликого тестового набору з великим покриттям коду.[15]

– **Windows Forms Designer:** Платформа має рішення для швидкої розробки додатків на основі Windows Forms - Windows Forms Designer, що дозволяє легко додавати елементи керування до форми, впорядковувати їх та писати код для їхніх подій. За допомогою конструктора можна додавати до форми компоненти, налаштовувати їх вигляд, функціонал та дії з ними за допомогою гарячих клавіш та зручного інтерфейсу.[16]

3.5 Дизайн додатку

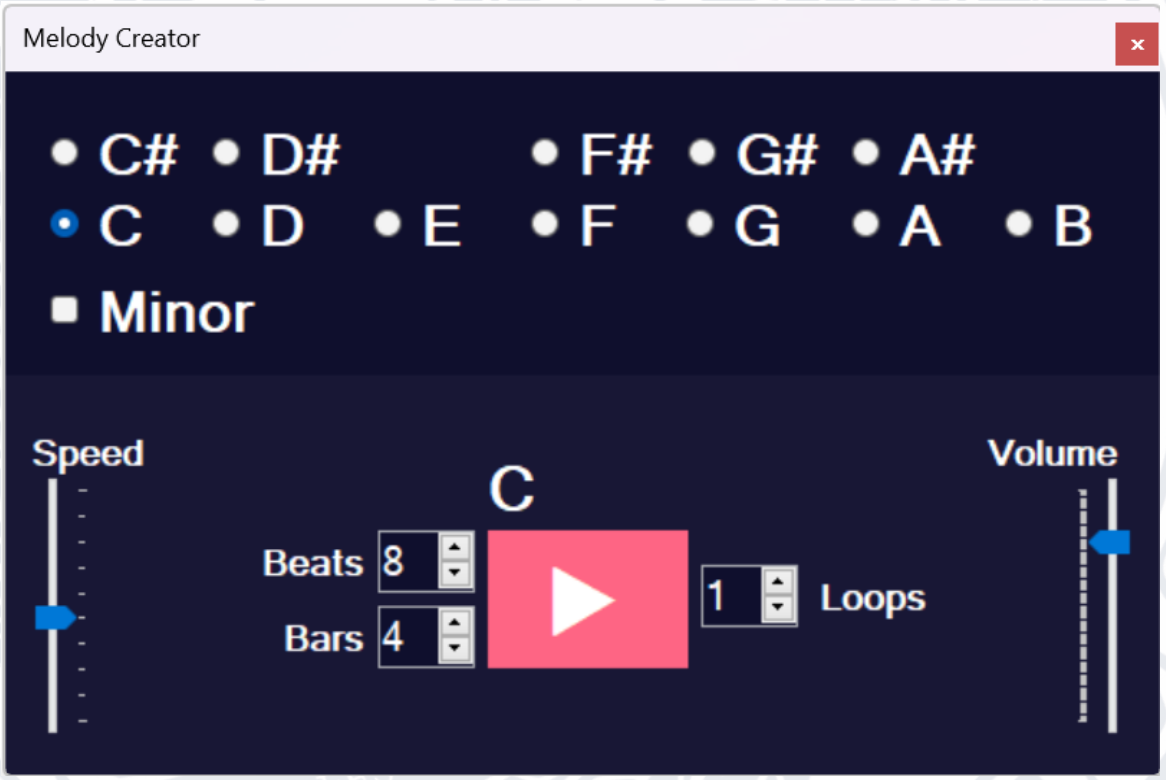

Рисунок 3.2.1 – Дизайн інтерфейсу в додатку «Melody Creator»

Дизайн є наближеним до знайомого вигляду музичного програвача аби спростити адаптацію користувачів до інтерфейсу. Кольори меню тональності та інших налаштувань відрізняються настільки, щоби розділити ці секції, але при цьому не привертати на себе занадто багато уваги. Акценти видно на елементах

налаштування та кнопці старту. Вона є центральним елементом дизайну оскільки генерація це основна функціональність додатку.

Зі сторони UX в додатку пророблено розділення різних налаштувань по їх групам. Тональність це основний параметр що регулює настрій та висоту мелодії, тому це меню розташоване зверху й відокремлене від інших налаштувань. Основні акорди розташовані нижче за # (дієз) акорди. Це схоже на те, як ноти розташовані на усім знайомим клавішам фортепіано. Мажор та мінор визначаються за допомогою простого чек-боксу. Також обрана тональність дублюється біля кнопки «плей». Таким чином навколо неї можна побачити усі визначальні налаштування.

Налаштування знизу відповідають за розмірність, темп та гучність, тобто характер мелодії. В центрі ми бачимо спінери для визначення елементів розмірності, отже візуально видно що вони відносяться до однієї групи. Зліва визначається темп. Він є менш визначальним, тому знаходиться в стороні, проте його роль важливіша за гучність, тому вона знаходиться правіше, ніби другою по визначальності.

Створений дизайн є доволі детальним та відритим з боку налаштувань. Це корисно якщо користувач розуміє як працює система, проте для людей незнайомих з музикою це може бути незручно. В майбутньому можна створити спрощений режим, де потрібно лише описати абстрактний характер та емоцію мелодії, а інші параметри підлаштуються автоматично.

3.6 Імплементація додатку

Створення додатку з Windows Forms у Visual Studio починається з інструмента під назвою «Дизайнер».

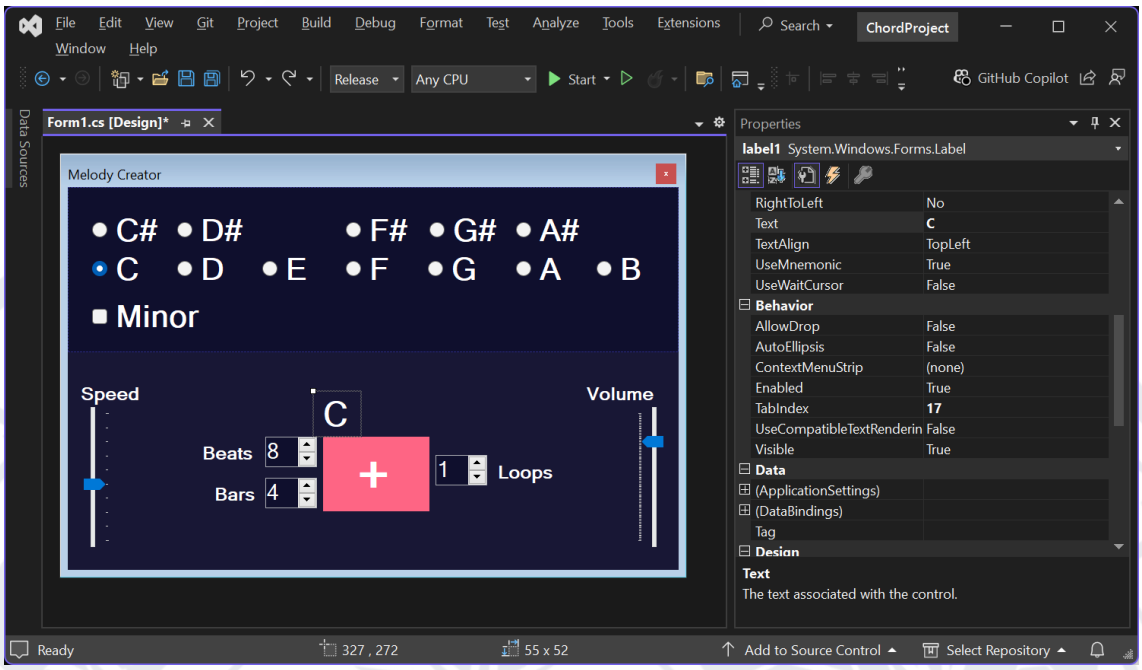

Рисунок 3.2.2 – Інтерфейс інструменту «Дизайнер» у Visual Studio

Тут можна створювати, розташовувати та налаштовувати елементи інтерфейсу користувача. Елементи дизайну були імплементовані за допомогою відповідних класів елементів:

– System.Windows.Forms.RadioButton – використовується для вибору однієї з кількох опцій що представлені іншими елементами керування RadioButton у даній групі елементів. Максимум 1 з елементів може бути обраним через інтерфейс. Якщо користувач обирає інший елемент групи, попередній відмічений втрачає цей статус й він передається до нового елементу. Клас може містити у собі назву що буде відображатися поруч із інтерактивним елементом вибору [17]. Цей клас був застосований для реалізації вибору тональностей мелодії. Він був обраний тому що кожна окрема мелодія може мати лише одну тональність з n можливих.

– System.Windows.Forms.CheckBox – (також відомий як прапорець) використовується, щоби надати користувачеві вибір, наприклад, істина/хиба або так/ні. На нього можна клацнути аби змінити стан з поточного на протилежний. Він є незалежним від усіх інших елементів

цього класу [18]. Цей елемент був застосований для позначення тональності як мінорної. Коли прапорець відмічений й знаходиться у стані «істина», тональність вважається мінорною. Коли прапорець знаходиться у стані «хиба», тональність вважається мажорною. Таким чином даний елемент репрезентує два можливих стани тональності.

- System.Windows.Forms.Label – використовується для відображення тексту на формі і не бере участі у користувацькому введенні або в подіях миші чи клавіатури [19]. У застосунку цей клас був застосований для відображення поточної обраної тональності в центрі інтерфейсу й також для підпису деяких інтерактивних елементів, таких як «Speed», «Volume», «Beats», «Bars» та «Loops».
- System.Windows.Forms.NumericUpDown – використовується для створення елемента керування, який відображає числові значення. Значення змінюється за допомогою стрілок вгору та вниз і утримує деяке попередньо визначене користувачем числове значення [20]. У додатку цей елемент використовується для визначення «Beats», «Bars» та «Loops», тобто кількості ударів, інтервалів та повторів у згенерованій мелодії. Ці значення мають бути точними й можуть змінюватися з дельтою що рівна 1. Це робить даний елемент ідеальною імплементацією потрібного функціоналу.
- System.Windows.Forms.TrackBar – це елемент керування, який має смугу значень та вказівник, що ковзає між мінімальним і максимальним значенням. Користувач може перетягувати вказівник вздовж лінії аби обрати необхідне значення [21]. Клас зберігає останнє вказане значення. Цей елемент дозволяє не дуже точно, але швидко та інтуїтивно налаштувати певний параметр, що підходить для параметрів швидкості та звуку. Для них і був застосований даний елемент.
- System.Windows.Forms.Button – використовується для обробки події натиснення миші у межах елементу кнопки [22]. У додатку кнопка

слугує тригером для зчитування параметрів, генерації та програвання мелодії.

Кожний зі створених елементів автоматично створює об’єкт відповідного класу й поміщає його у поле, ім’я якого може бути налаштовано.

Тригери можна додати до будь-якого елементу клацнувши на нього двічі. Таким чином для центральної кнопки «Play» в інтерфейсі був створений метод що має наступну сигнатуру: «private void buttonPlay_Click(object sender, EventArgs e)». Цей метод й буде викликатися при натисненні на кнопку. У ньому ми визначаємо які значення присутні в об’єктах інших елементів й передаємо ці дані у метод що запускає імплементацію створеного алгоритму. Алгоритм підбирає необхідні ноти які треба відтворити.

На рисунку 3.2.2 можна помітити що в дизайнері кнопка програвання мелодії має символ «+», в той час як на дизайні використовується символ «▶». Справа у обмеженнях роботи із інструментом «Дизайнер» у Visual Studio. Він обмежує множину символів що можливо додати у якості тексту на елементі. Проте Unicode символи все ж підтримуються у Windows Forms. Для того аби обійти обмеження інструменту розробки, до елементу був доданий символ-заповнювач, що замінений на символ «▶» при ініціалізації додатку. Для цього поле Text елементу buttonPlay присвоюється значенню «\u25B6», що є кодом для «▶» у системі Unicode.

Для відтворення нот використовується клас Midi.OutputDevice. Кожен екземпляр цього класу описує пристрій аудіо виводу, встановлений у системі. Midi.OutputDevice.InstalledDevices може бути використаний для того аби дізнатися список доступних аудіо девайсів в системі. Кожний з цих об’єктів матиме поле Midi.DeviceBase.Name аби дати користувачеві змогу вибрати необхідний девайс [23]. Проте для спрощення додатку, береться перший доступний девайс.

Для використання обраного об’єкту Midi.OutputDevice його необхідно ініціювати за допомогою методу void OutputDevice.Open() [23]. Після цього з ним

можна проводити різного роду інтеракції. Наприклад, одразу після ініціації викликається метод void OutputDevice.SendControlChange(Channel channel, Control control, int value), в який ми передаємо канал девайсу що буде використовуватися для подальших викликів, параметр для зміни та значення цього параметру [23]. В додатку ми використовуємо Midi.Channel.Channel1 в якості основного каналу. На початку роботи програми нам необхідно задати значення 127 параметру Midi.Control.SustainPedal. Це переключає програвання нот у режим подовжених звуків. Тобто після запиту на програвання ноти, вона продовжить грати й затухати у гучності протягом певного часу. Це створює ефект гри на фортепіано із зажатою педаллю витримки. В інакшому випадку ноти б програвалися протягом короткого інтервалу й різко затихали, що звучить не природньо.

Надалі в додатку використовується метод void outputDevice.SendNoteOn(Channel channel, Pitch pitch, int velocity), де задається канал девайсу де нота буде програна, висота ноти та швидкість натискання [23]. Як зазначено вище, усі інтеракції з пристроєм виводу проводяться через перший канал. Висота ноти визначається алгоритмом. Швидкість натискання по своїй суті впливає на гучність програної ноти й береться із значення в об’єкті відповідного параметра на інтерфейсі.

Кожна нота має мати паузу перед програванням іншої. Ці паузи імплементовані за допомогою методу void Thread.Sleep(int millisecondsTimeout) що переводить потік у режим очікування на вказаний проміжок часу [24]. Час очікування задається через формулу:

$x = \frac{1000}{v}$, де v-значення параметру швидкості

Формула 3.2.1 – Формула визначення часу пауз між програванням нот

Можна також зазначити що додаткова затримка створюється під час роботи самого алгоритму, проте вона є не суттєвою та не впливає на сприйняття.

Висновок до розділу 3

Розробка додатку була розбита на дві частини: створення дизайну та імплементація функціоналу.

Дизайн був створений з урахуванням сильних та слабких сторін конкурентів. Він є простим для адаптації, контрастним й детальним у налаштуваннях застосунку.

Імплементація була розпочата з огляду технологій. Були обрані ті, що мають найбільшу простоту, але які надають увесь необхідний функціонал для реалізації алгоритму та дизайну. За заданими критеріями був використаний Windows Forms на базі .NET та C#. Опираючись на обрані інструменти середовищем розробки був обраний Visual Studio що забезпечує підтримку усіх заданих технологій. Для імплементації аудіо складової застосунку була обраний midi-dot-net як бібліотека що покриває потребу у відтворенні заданих нот та конфігурації відтворення звуку.

Були описані особливості роботи із даними технологіями в контексті створення застосунку для генерації мелодій. У застосунку був повноцінно імплементований дизайн, функціонал та алгоритм генерації.

ВИСНОВКИ

Створений проект є прикладом алгоритмізації реальної діяльності та поєднання різних за своєю природою наук в одному продукті. Реалізований застосунок побудований на основі аналізу UI/UX дизайну для додатків й технологіях .NET, C# та Windows Forms.

Додаток може бути використаний як професійними музикантами для автоматичного виповнення простих рутинних задач з написання музичних партій, так і незнайомими з музикую людьми, які натомість потребують створення композиції за персональними характеристиками. Також цю роботу можна використати в іншій області, а саме навчанні машинного інтелекту. В цьому випадку додаток може аналізувати ноти від ШІ та перевіряти їх на мелодичність.

Алгоритм можна розширити шляхом подальшого дослідження зв’язків між акордами кварто-квінтового кола для більш унікальних та персоналізованих партій. Також можна поліпшити параметризацію мелодій, використовуючи загально вживані терміни для опису бажаної композиції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Melody. Encyclopedia Britannica. URL: https://www.britannica.com/dictionary/melody (дата звернення: 16.05.2024).

2. MIDI (Musical Instrument Digital Interface). TechTarget. URL: https://www.techtarget.com/whatis/definition/MIDI-Musical-Instrument-Digital-Interface (дата звернення: 16.05.2024).

3. Mark DeVoto. Key. Encyclopedia Britannica. URL: www.britannica.com/art/key-music (дата звернення: 16.05.2024).

4. Chord Progressions. Music Theory Academy. URL: www.musictheoryacademy.com/understanding-music/chord-progressions (дата звернення: 16.05.2024).

5. The Circle Of Fifths. Music Theory Academy. URL: www.musictheoryacademy.com/understanding-music/the-circle-of-fifths (дата звернення: 16.05.2024).

6. Time Signatures. Music Theory Academy. URL: www.musictheoryacademy.com/understanding-music/the-circle-of-fifths (дата звернення: 16.05.2024).

7. A Guide To Semitones & Tones (Half & Whole Steps). Hello Music Theory. URL: hellomusictheory.com/learn/semitones-tones (дата звернення: 16.05.2024).

8. How to Make Chords From Scales. Flypaper. URL: flypaper.soundfly.com/write/how-to-make-chords-from-scales (дата звернення: 16.05.2024).

9. What is .Net. AWS. URL: https://aws.amazon.com/what-is/net/?nc1=h_ls (дата звернення: 16.05.2024).

10. Introduction to C#: definition and utilities. MyTaskPanel Consulting. URL: https://www.mytaskpanel.com/introduction-to-csharp/ (дата звернення: 16.05.2024).

11. Desktop Guide (Windows Forms .NET). Microsoft. URL: https://learn.microsoft.com/en-

us/dotnet/desktop/winforms/overview/?view=netdesktop-8.0 (дата звернення: 16.05.2024).

12. Sascha Thattil. Advantages and Disadvantages of Winforms. Yuhiro. URL: https://www.software-developer-india.com/advantages-and-disadvantages-of-winforms/ (дата звернення: 16.05.2024).

13. Windows Presentation Foundation (WPF). GitHub. URL: https://github.com/dotnet/wpf (дата звернення: 16.05.2024).

14. Sascha Thattil. Advantages and Disadvantages of WPF. Yuhiro. URL: https://www.software-developer-india.com/advantages-and-disadvantages-of-wpf (дата звернення: 16.05.2024).

15. Microsoft Visual Studio Product Overview. TechnologyAdvice. URL: https://technologyadvice.com/products/microsoft-visual-studio-reviews/ (дата звернення: 16.05.2024).

16. Windows Forms Designer overview. Microsoft. URL: https://learn.microsoft.com/en-us/visualstudio/designers/windows-forms-designer-overview?view=vs-2022 (дата звернення: 16.05.2024).

17. RadioButton Class. Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.radiobutton?view=windowsdesktop-8.0 (дата звернення: 16.05.2024).

18. Using CheckBox In Windows Forms. C# Corner. URL: https://www.c-sharpcorner.com/article/working-with-checkbox-control-in-winforms-application2/ (дата звернення: 16.05.2024).

19. Label in C#. GeeksforGeeks. URL: https://www.geeksforgeeks.org/label-in-c-sharp/ (дата звернення: 16.05.2024).

20. C# | NumericUpDown Class. GeeksforGeeks. URL: https://www.geeksforgeeks.org/c-sharp-numericupdown-class/ (дата звернення: 16.05.2024).

21. Advanced TrackBar. Syncfusion. URL: https://www.syncfusion.com/winforms-ui-controls/trackbar (дата звернення: 16.05.2024).

22. Button Class. Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.button?view=windowsdesktop-8.0 (дата звернення: 16.05.2024).

23. OutputDevice.cs. GitHub. URL: https://github.com/jstnryan/midi-dot-net/blob/master/Midi/OutputDevice.cs (дата звернення: 16.05.2024).

24. Thread.Sleep Method. Microsoft. URL: https://learn.microsoft.com/en-us/dotnet/api/system.threading.thread.sleep?view=net-8.0 (дата звернення: 16.05.2024).