введение

Джазовые музыканты на всех этапах своего развития сталкиваются с необходимостью заниматься под метроном, под минус или виртуальный ансамбль. В настоящее время , виртуальный ансамбль пользуется большой популярностью, так как музыкант может себе выбрать темп, тональность, стиль, тактовый размер и другие параметры, чего в заранее записанном минусе сделать крайне проблематично.  
Для таких случаев уже давно существуют программы, которые позволяют генерировать виртуальные барабаны, бас и фортепианный / гитарный аккомпанемент, но они часто генерируют слишком простые, шаблонные и однообразные ритмы и басовые линии, поэтому под них скучно и не интересно играть.  
В рамках данного исследования была разработана первая бетаверсия программы, которая использует простой анализ гармонических последовательностей и позволяет генерировать более связные и интересные джазовые аккомпанементы.

1. Определение требований к разрабатываемому приложению (краткое техническое задание)

ПО разрабатывается на языке программирования Python 3 с использованием библиотеки chordparser.  
В данной бетаверсии программа не имеет графического интерфейса, поэтому для взаимодействия с пользователем решено минимизировать работу в консоли. На данном этапе был добавлен минималистичный вебинтерфейс на fastapi, который позволяет пользователю написать гармоническую сетку в поле и получать на выходе сгенерированный аудиофайл в формате wav.

# 1.1 Назначение и цели создания приложения

Целью данной работы является разработка программы, которая позволяет генерировать джазовые аккомпанементы, которые будут более похожи на то, что играет живой музыкант.  
Задачи:  
- Разработка алгоритма генерации басовых линий, имеющих правильную гармоническую связь,  
- Разработка алгоритма генерации барабанных партий, которые будут включать в себя взбивки, синкопы и триоли,  
- разработка простой консольной программы, которая будет принимать на вход текстовый файл с гармонической сеткой и генерировать аудиофайл с аккомпанементом,  
- Разработка простого вебинтерфейса для взаимодействия с пользователем.

# 1.2 Требования к программе

* Программа должна быть простой в использовании, кроссплатформенной, и доступной для незрячих пользователей.
* Программа должна уметь работать с любыми тактовыми размерами со знаменателем 4.
* Программа должна уметь менять остроту свинга в зависимости от темпа, если она не была задана пользователем.
* Программа должна уметь доводить басовую линию до тоники следующих аккордов без сильных скачков.
* Программа должна создавать интересные барабанные взбивки и переходы, например плавно переходить от щёток в теме к палкам в соло и обратно, делать синкопированные и ровные коды.

# 1.3 Состав и содержание работ по созданию программы

* (x) Разработка алгоритма генерации аккомпанемента на басу.
* (x) Разработка алгоритма генерации аккомпанемента на барабанах, барабанных взбивок, случайных синкоп, триолей и ритмических фигур.
* ( ) Разработка простого вебинтерфейса для взаимодействия с пользователем.
* ( ) Написание документации и инструкций по использованию программы.

# 1.4 Порядок контроля и приемки приложения

Выполненная курсовая работа сдается в несколько этапов в установленные техническим заданием сроки. На первом этапе преподавателем производиться оценка программного кода на соответствие стандарту оформления PEP8 и PEP257, а также работоспособность кода. На втором этапе производится оценка текста пояснительной записки к курсовому проекту на соответствие стандарту СТО 60-02.2.3-2018. На третьем этапе производиться защита курсовой работе в формате выступления с презентацией

# 1.5 Требования к документированию

* (x) Документация к программе должна быть написана ввиде Docstrings ко всем классам и функциям по стандартам PEP257 и СТО 60-02.2.3-2018.

2 Проектирование приложения

# 2.1 Проектирование вариантов использования приложения

При использовании консольного клиента пользователь пишет текстовый файл input.txt на специальном языке, который описывает гармоническую сетку, тональность, темп, смену размеров и другие музыкальные тонкости произведения. Затем запускает программу и получает на выходе файл output.wav, который автоматически откроется в ассоциированном с wav-файлами проигрывателе.  
При использовании вебинтерфейса пользователь открывает страницу, на которой есть поле для ввода гармонической сетки, кнопка для генерации аккомпанемента и кнопка для скачивания сгенерированного аудиофайла. После нажатия кнопки "Сгенерировать" пользователь получает на выходе аудиофайл в формате wav, который можно прослушать в браузере или скачать.

# 2.2 Описание выделенных компонентов

Программа состоит из следующих компонентов:  
- bass.py - логика генерации басовых линий,  
- drums.py - логика генерации барабанных партий,  
- drum\_sounds.py - вспомогательный модуль для комбинирования звуков ударных,  
- sound\_combiner.py - модуль для комбинирования звуков в один wav файл по тактам,  
- harmony.py - вспомогательный модуль для работы с гармоническими последовательностями,  
- notes\_with\_octaves.py - модуль-обёртка над библиотекой Chordparser, описывающий функции для работ с нотами в октавах, гаммами и midi значениями,  
- main.py - основная консольная программа.  
- app.py - вебинтерфейс на fastapi, который позволяет пользователю взаимодействовать с программой через браузер.

3. Реализация приложения

# 3.1. Выбор языка программирования и среды разработки

Программа написана на языке Python 3.13. Выбор языка обусловлен его простотой, наличием большого количества библиотек и кросплатформенностью. Первая попытка реализации программы была сделана на языке Rust, но из-за недостатка библиотек и сложности работы с аудио библиотеками, было решено вернуться к Python.  
- достоинства  
 - Простой и понятный синтаксис.  
 - множество библиотек, разработанных сообществом.  
 - Поддержка ООП.  
 - Кроссплатформенность.  
- недостатки  
 - Низкая скорость выполнения.  
 - динамическая типизация.

Программа разрабатывается в Visual Studio Code, так как это единственная IDE, которая доступна для незрячих.  
- достоинства  
 - Бесплатная.  
 - Поддержка множества языков программирования.  
 - Поддержка плагинов.  
 - Кроссплатформенность.  
 - поддержка github copilot.

# 3.2. Выбор и обоснование используемых библиотек

* chordparser
* - библиотека для работы с аккордами и нотами. Была выбрана из-за своей простоты и полного функционала. Для расширения функционала были написаны обёртки.
* pydub
* - библиотека для работы с аудио файлами. Позволяет легко комбинировать звуки, изменять громкость и т.д. Была выбрана из-за своей сверхпростоты и кросплатформенности.
* fastapi
* - библиотека для создания веб-приложений на Python. Была выбрана из-за своей простоты и скорости разработки. Позволяет легко создавать RESTful API и веб-интерфейсы.
* uvicorn
* - асинхронный сервер для запуска приложений на fastapi. Выбран из-за своей скорости и простоты использования.
* Для тестирования используется библиотека
* pytest
* . Выбрана из-за того, что она позволяет легко писать тесты и имеет множество плагинов.
* для компиляции программы в exe используется библиотека
* pyinstaller
* . Самая популярная библиотека для компиляции программ на Python в exe.
* Для управления зависимостями используется новый менеджер пакетов
* UV
* . Был выбран из-за его скорости и многофункциональности,. Он позволяет устанавливать зависимости прямо в виртуальное окружение, управлять версиями Python, а так же устанавливать инструменты командной строки. UV написан на rust, что обеспечивает высокую скорость при установке нескольких десятков пакетов.

# 3.3. Описание разработанных алгоритмов

## 3.3.1 Алгоритм генерации басовых линий

* Программа получает на вход гармоническую сетку произведения и попарно итерирует по аккордам. Для каждого аккорда генерируется линия баса по гамме этого аккорда, а последняя нота в линии текущего аккорда выберается та, которая тяготеет к тонике следующего. Например пятая ступень, вторая пониженная (по хроматизму) или большая септима.
* В случайных местах редко вставляются хроматические переходы по гамме, синкопы и синкопированное остенато (пока не реализовано).

## 3.3.2 Алгоритм генерации барабанных партий

Пока что используется упрощённый алгоритм генерации барабанов. Тарелка бьёт на 1, 3 и 4 восьмую в каждой половинной, большой барабан на первую долю, малый барабан иногда синкопирует, дробит или делает триоль на 2 и 3 треть четверти.

В будущем планируется добавить алгоритм, который будет генерировать более сложные барабанные партии, которые будут включать в себя взбивки и переходы в новые фразы, а так же переход от счёток к палкам и обратно.

# 3.4. Описание реализованных классов, их свойств и методов.

СМ. Документация

4. Тестирование

# 4.1. Написание Unit-тестов

Юниттесты написаны для всех функций, которые имеют определённый результат. Для модуля note with octave написан файл с тестами, именованный test\_<имя\_модуля>.py. Тесты написаны с использованием библиотеки pytest. Для запуска тестов нужно выполнить команду uv run pytest в терминале.