УДК 004.056.5

Программная система для оценки качества настройки струнного инструмента

Леснов Е.В., Андрюхин А.И.
Донецкий национальный технический университет
s4eniel@gmail.com

Леснов Е.В., Андрюхин А.И. Программная система для оценки качества настройки струнного инструмента.*В данной работе рассмотрены различные вариации тюнеров, приведено описание проектирования и реализации собственной программной системы для оценки качества настройки струнных инструментов, реализующей алгоритм считывания звукового потока и его последующую обработку.*

Введение

На сегодняшний день многие люди увлекаются игрой на гитаре. Кто-то более серьезно, кто-то менее, но ни один гитарист не обходится без тюнера. Современные технологии предоставляет возможность настроить музыкальный инструмент практически в любых условиях, имея при себе необходимый гаджет.

В настоящее время, платформа Android является одной из самых быстроразвивающихся мобильных операционных систем. Регулярно выпускаются новые устройства, причем каждое следующее поколение на порядок мощнее предыдущего. Следует отметить, что данная ОС является open source и распространяется по лицензии GNU. Эти факторы в совокупности и создают огромную популярность данной платформы.

Таким образом, с оглядкой на популярность платформы Android, решение создать приложение на базе данной ОС является очень актуальным.

Постановка задачи

Тюнер — отдельное устройство или программа, облегчающие настройку музыкальных инструментов. Использование тюнера облегчает задачу в разы: от музыканта требуется только следить за шкалой или другой индикацией, предусмотренной прибором, и смотреть, чтобы отклонение от вышеупомянутого эталона не было слишком велико[1].

В связи с развитием мобильных устройств не так давно появилась такая разновидность тюнеров, как мобильный тюнер, который представляет собой программное обеспечение для смартфонов. Для настройки инструмента требуется, собственно, инструмент и смартфон.

Далее, музыкант играет каждую струну по очереди и, смотря на экран гаджета, определяет в какую сторону нужно крутить колки. Были проанализированы существующие аналоги и на основании этих данных были разработаны основные требования и поставлены основные задачи.

Основная цель — разработать приложение для начинающих музыкантов, помогающее им настроить инструмент, а также получить быстрый доступ к табулатурам основных типов аккордов. Кроме того, приложение должно иметь интуитивно понятный и максимально информативный интерфейс, иметь широкий выбор строев инструмента, охватывающий все стандарты исполнения.

Приложение должно иметь высокую скорость обработки входящего звукового потока, а также иметь расширение функционала в виде интерфейса взаимодействия с базой данных, содержащей в себе изображения табулатур основных типов аккордов.

Приложение должно работать на платформе Android актуальной версии.

Структура программной системы

Посмотрим на статическую структурную диаграмму, показывающую разбиение программной системы на структурные компоненты и связи между компонентами, изображенную на рисунке 1.

Имеется общий пакет com.saeniel.tuner, внутри которого находятся классы, отвечающие за функциональность, собственно, тюнера, а так же подпакет, содержащий в себе классы, реализующие функциональность помощника по поиску аккордов в базе данных.

Как видно из диаграммы, после компиляции, на выходе получится .apk файл, готовый к установке на устройство Android.

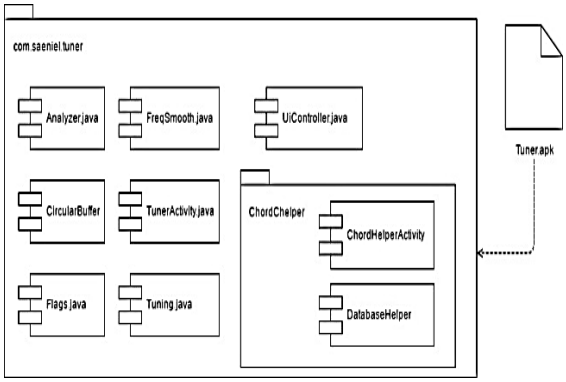

Рисунок 1 — Диаграмма развёртывания

Диаграмма вариантов использования системы представлена на рисунке 2.

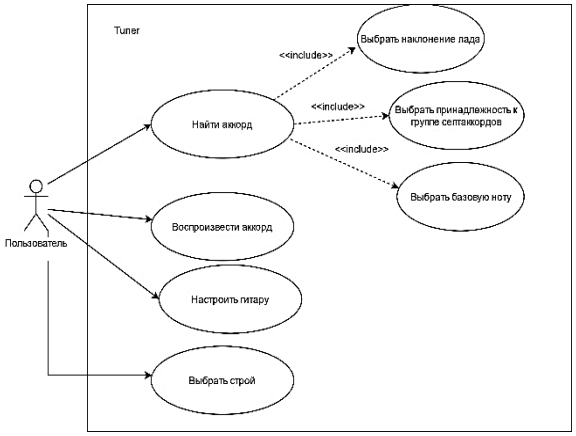

Рисунок 2 — Диаграмма вариантов использования

Пользователь имеет возможность совершить ряд действий: найти заданный аккорд, воспроизвести его звучание, выбрать строй, по которому будет настраиваться инструмент и, собственно, настроить гитару.

Данная программная система каждый момент времени находится в определенном состоянии. На рисунке 3 приведена диаграмма состояний.

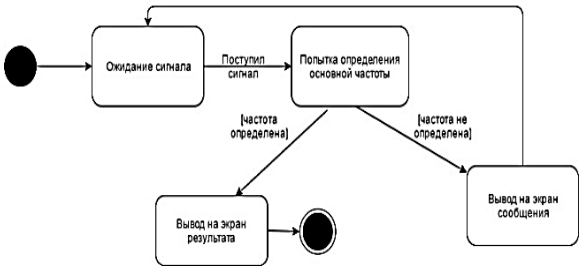

Рисунок 3 — Диаграмма состояний

Проектирование структур системы

При разработке приложения были использованы средства Android SDK для реализации логики приложения. Для хранения информации необходимой для работы помощника по поиску аккордов была спроектирована база данных. Для работы базы данных используется свободная, кроссплатформенная система управления базами данных SQLite.

Исходя из требований, выдвинутых к разрабатываемой системе, была спроектирована база данных. На рисунке 4 представлена схема базы данных, отвечающая за хранение аккордов

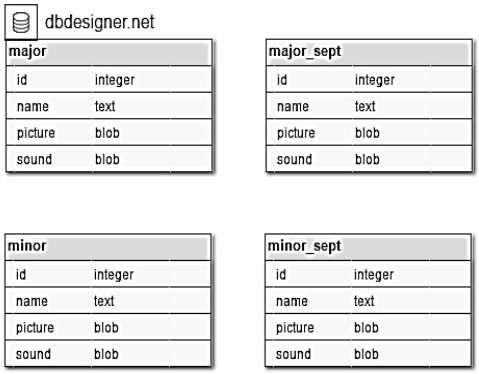

Рисунок 4 – Схема БД

В базе данных содержится четыре таблицы, которые отображают четыре основные типы аккордов: мажорные аккорды, мажорные септаккорды, минорные аккорды, минорные септаккорды. В таблицах хранится такая информация, как название аккорда, его графическое отображение в виде табулатуры и звуковой файл, позволяющий услышать звучание аккорда.

Как видно из представленной схемы базы данных, между таблицами отсутствуют связи. Это легко объясняется тем фактом, что все таблицы являются справочниками, что, по сути, означает их неизменяемость.

И хоть между таблицами нет связей, их все объединяет одинаковая структура. Во все таблицы входят только следующие поля:

– id — уникальный порядковый номер, идентификатор;

– name — название аккорда;

– picture — графическое изображение аккорда, его табулатура;

– sound — звучание аккорда.

Программная реализация

Для разработки системы был выбран высокоуровневый язык программирования Java. В качестве интегрированной среды разработки была выбрана Android Studio.

При создании данной программной системы был использован объектно-ориентированный подход.

Данное приложение использует шаблон «Наблюдатель». Этот поведенческий шаблон создает механизм у класса, который позволяет получать экземпляру объекта этого класса оповещения от других объектов об изменении их состояния, тем самым наблюдая за ними

На этапе проектированы отдельные сущности были выделены в классы, согласно самой концепции ООП. На рисунке 5 изображена диаграмма классов разработанного приложения.

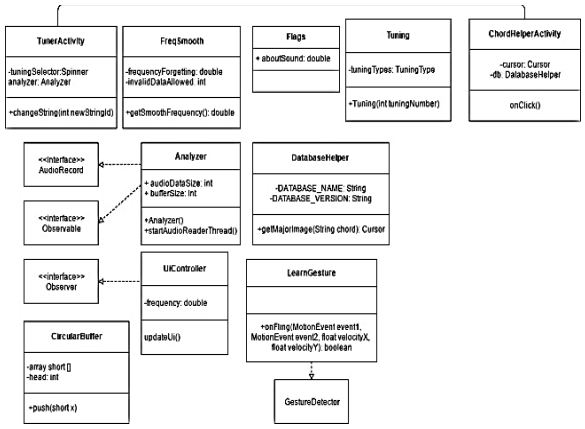

Рисунок 5 – Диаграмма классов

Рассмотрим более детально классы данного приложения.

Класс TunerActivity отвечает за функцию настройки инструмента. Он отвечает за принятие информации из классов, реализующих логику обработки входного звукового сигнала, а также за отображение результатов проанализированного сигнала на экране устройства.

Класс Flags содержит в себе флаги информирования пользователя.

Класс CircularBuffer реализует кольцевой буфер. Кольцевой буфер — это структура данных, использующая единственный буфер фиксированного размера, как будто бы после последнего элемента сразу же снова идет первый. Такая структура легко предоставляет возможность буферизации потоков данных.

Класс FreqSmooth позволяет «сгладить» частоту, тем самым отделив полезную частоту от ненужного сигнала.

Класс Analyzer отвечает за считывание анализ входного потока аудиоданных. Этот класс является «наблюдаемым» относительно класса UiController, согласно шаблону «наблюдатель». После проведенного анализа, результат передается в класс UiController, где происходит визуализация проанализированных данных.

Класс Tuning отвечает за графическую интерпретацию логики работы тюнера. Содержит в себе механизм, позволяющий перейти от понятия «частота» к понятию «нота».

Класс UiController отвечает отображение на экране смартфона обработанного звука. Является «наблюдателем» относительно класса Analyzer, согласно шаблону «наблюдатель».

Класс LearnGesture отвечает за распознавание жестов, позволяющих переключиться между тюнером и помощником аккордов.

Класс DatabaseHelper осуществляет взаимодействие между базой данных и приложением. Содержит функции получения информации из базы.

Класс ChordHelperActivity отвечает за логику взаимодействия пользователя с помощником аккордов. Основную логику содержит в себе обработчик кнопки «Поиск». Нажатие на данную кнопку запускает процесс парсинга аккорда. Определяется базовая нота аккорда, затем определяется наклонение лада (мажор или минор) и является ли аккорд септаккордом.

Алгоритм работы тюнера изображен на рисунке 6.

Рисунок 6 – Алгоритм работы тюнера

При запуске приложения создается объект класса, отвечающий за запись звука. Далее, происходит загрузка строя инструмента и начинается чтение звукового потока. Считывается порция сигнала и сразу определяется его громкость. Если значение громкости недостаточно велико, считывание сигнала происходит заново. При условии, что громкость сигнала удовлетворительная начинается вычисление автокорелляции первого порядка. Вычисляется прямое преобразование Фурье, после которого значение частоты заменяется

вычисленным значением. Затем, вычисляется обратное преобразование Фурье и от полученного результата берется вещественная часть. После проведенных вычислений начинается процесс удаления «ложных» сэмплов, после чего на экране отображается сыгранная пользователем нота.

Непрерывная функция и представление её рядом Фурье

Математической основой спектрального анализа сигналов является преобразование Фурье.

Математически, сигнал длительностью T секунд является некоторой функцией f(x), заданной на отрезке {0, T} (X в данном случае — время). Такую функцию всегда можно представить в виде суммы гармонических функций (синусоид или косинусоид) вида:

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{+\infty} A_k \cos\left(2\pi \frac{k}{\tau} x + \theta_k\right) \quad (1), \text{ где}$$

k – номер тригонометрической функции (номер гармонической составляющей, номер гармоники)
T – отрезок, где функция определена (длительность сигнала)
A_k – амплитуда k-ой гармонической составляющей,
θ_k – начальная фаза k-ой гармонической составляющей.

Среднеквадратичное отклонение ряда от функции f(x) будет стремиться к нулю, но несмотря на среднеквадратичную сходимость, ряд Фурье функции, вообще говоря, не обязан сходиться к ней поточечно[3].

Этот ряд может быть также записан в виде:

$$f(x) = \sum_{-\infty}^{+\infty} \hat{f}_k e^{i2\pi \frac{k}{\tau} x} \quad (2), \text{ где}$$

$\hat{f}_k$ – k-ая комплексная амплитуда.

Или так:

$$f(x) = \frac{a_0}{2} + \sum_{k=1} [a_k \cos\left(2\pi \frac{k}{\tau} x\right) + b_k \sin(2\pi \frac{k}{\tau} x)] \quad (3)$$

Связь между коэффициентами (1) и (3) выражается следующими формулами:

$$A_k = \sqrt{a_k^2 + b_k^2}$$

и

$$\theta_k = arctg \frac{b_k}{a_k}.$$

Пример использования системы

Разработанный программный продукт является приложением, предназначенным для начинающих музыкантов и совмещает в себе функционал тюнера и интерфейс по поиску табулатур аккордов. Проводя циклически анализ входного потока данных из микрофона устройства, приложение определяет основную частоту, звучащую в реальном времени. Результаты сравнение фактической частоты с эталонной выводятся на экран, что позволяет увидеть отклонение от эталона.

Помощник поиска аккордов представляет из себя интерфейс взаимодействия с базой данных. Заполнив соответствующие поля, пользователь имеет возможность найти в базе данных изображение табулатуры искомого аккорда, а также его звуковой файл, содержащий звучание аккорда. Изображения и звуковые файлы хранятся в базе как массивы байт, которые в приложении приводятся в воспроизводимые форматы.

После запуска приложения, по умолчанию запускается тюнер. Скриншот тюнера приведен на рисунке 7.

Рисунок 7 – Экран тюнера

На данном этапе пользователю следует поднести смартфон к резонатору гитары (если речь идет о классической или акустической гитаре) или же динамику комбоусилителя, если настраиваемый инструмент — электрогитара. Затем, требуется играть открытые струны медленно, чтобы дать время приложению на анализ. Проанализировав звук, приложение выдаст результат в строку состояния и отобразит графически отклонение от эталона.

Изображение работы тюнера приведено на странице 8.

Рисунок 8 – Работа тюнера

По окончанию настройки инструмента, пользователь имеет возможность просмотреть табулатуру того или иного аккорда, а также услышать, как он звучит. На выбор пользователя предоставлены четыре основные типа аккордов: минорный, мажорный, минорный септаккорд, мажорный септаккорд.

Для того, чтобы перейти из режима тюнера в режим помощника поиска аккордов, пользователь должен совершить действие «свайп влево» или «свайп 46 вправо». Загрузится помощник поиска аккордов, изображение экрана которого показано на рисунке 8.

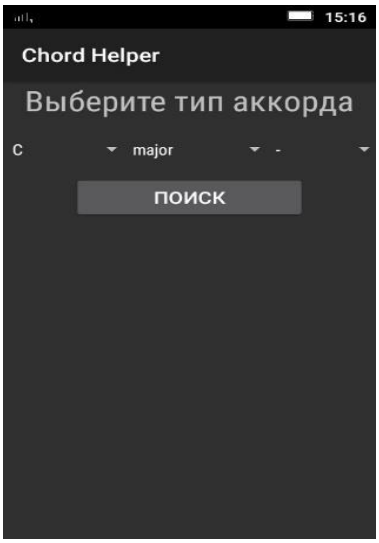

Рисунок 9 – Экран помощника поиска аккордов

Пользователь имеет возможность выбора базовой ноты аккорда, наклонение лада (минор или мажор), а также принадлежность аккорда к группе септаккордов.

После выбора соответствующих вариантов в списках, требуется нажать кнопку «Поиск». После поиска в базе данных будет произведена загрузка изображения и звукового файла.

По нажатию на изображение табулатуры запустится воспроизведение данного аккорда встроенным плеером.

На рисунке 10 показан результат поиска в базе данных по запросу «до мажор септаккорд».

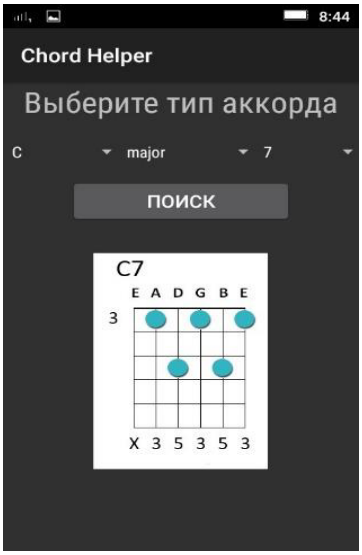

Рисунок 10 – Работа помощника по поиску аккордов

Тестирование

При разработке данного программного обеспечения было использовано тестирование юзабилити.

Юзабилити-тестирование — это исследование, выполняемое с целью определения, удобен ли некоторый искусственный объект (такой как веб- страница, пользовательский интерфейс или устройство) для его предполагаемого применения[2].

Для оценки качества интерфейса или пригодности использования продукции были использованы следующие юзабилити-метрики:

– результативность;

– эффективность;

– удовлетворенность.

В результате тестирования, фокус-группа определила, что приложение является интуитивно-понятным и эффективным. Последний респондент проводил тест с экстремально низким строем инструмента, и в следствие этого, приложение не всегда могло адекватно определить ноту.

Результаты тестирования приведены на рисунке 11.

Результативность	Эффективность	Удовлетворенность
90 %	30%	+
90%	30%	+
100%	30%	+
90%	30%	+
95%	30%	+
85%	45%	+-

Рисунок 11 – Результаты юзабилити-тестирования

Выводы

В ходе выполнения данной работы было разработано программное обеспечение, предназначенное для начинающих музыкантов. Данный программный продукт представляет из себя тюнер и помощник поиска табулатур, написанный с использованием языка Java, СУБД MySQL для платформы Android. Приложение позволяет выполнить основные функции, связанные с настройкой струнных музыкальных инструментов.

Помимо этого, в рамках программного продукта реализован помощник поиска аккордов, реализующий поиск заданного аккорда в базе данных.

В дальнейшем планируется улучшать и дорабатывать систему оптимизируя основной алгоритм путем уменьшения времени обработки сигнала и повышени я точности обработки, т.к. именно эти два параметра являются самыми важными в тюнере.

В настоящее время основная функциональность разработки была расширена за счет добавления помощника по поиску аккордов, планируется еще больше улучшить представленный помощник, превратить его в полноценный справочник по теории музыки в мобильном приложении, содержащий в себе исчерпывающие описания и звуковые примеры.

Реализация вышеописанных планов позволит сделать данное приложение конкурентноспособным на рынке аналогичных программных продуктов, что поможет вывести распространение разработанного приложения на коммерческий уровень.

Литература

1. Тюнер [Электронный ресурс] // guitarlesson. – Режим доступа: http://guitarlesson.ru/stati/tyuner-dlya-gitary.html. – Загл. с экрана.
2. Тестирование юзабилити [Электронный ресурс] // artw. – Режим доступа: https://artw.ru/blog/archives/3537/. – Загл. с экрана
3. Ряд Фурье [Электронный ресурс] // Wikipedia. – Режим доступа: https://ru.wikipedia.org/wiki/Ряд_Фурье
4.Эдмунд Лэй.Цифровая обработка сигналов для инженеров и технических специалистов / Эдмунд Лэй. – Москва: ООО «Группа ИДТ», 2007

Леснов Е.В., Андрюхин А.И. Программная система для оценки качества настройки струнного инструмента.*В данной работе рассмотрены различные вариации тюнеров, приведено описание проектирования и реализации собственной программной системы для оценки качества настройки струнных инструментов, реализующей алгоритм считывания звукового потока и его последующую обработку.*
Ключевые слова:*качество, настройка, струнный инструмент, тюнер,Java, Android*

Lesnov E.V., Andryukhin A.I. A software system for evaluating the quality of tuning a string instrument*. In this paper, various variations of tuners are considered. A description of the design and implementation of its own software system for evaluating the quality of tuning of stringed instruments is given. This system implements the algorithm of reading the audio stream and its subsequent processing.*
Keywords: *quality, tuning, string instrument, tuner, Java, Android*

Статья поступила в редакцию 20.11.2017
Рекомендована к публикации доктором технических наук В.Н. Павлышом