

Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Факультет КОМП'ЮТЕРНИХ НАУК

(повна назва)

Кафедра ПРОГРАМНОЇ ІНЖЕНЕРІЇ  
(повна назва)

## АТЕСТАЦІЙНА РОБОТА (ПРОЕКТ) Пояснювальна записка

БАКАЛАВР

(освітньо-кваліфікаційний рівень)

(позначення документа)

Ігровий додаток для симуляції пересування в спеціальних умовах .

.

.

(тема)

Виконав: студент 4 курсу, групи III-12-1  
напряму підготовки (спеціальності) 6.050103 Програмна інженерія

(шифр і назва напряму, спеціальності)

Шпетний Д.В.

(прізвище, ініціали)

Керівник Туруга О.П.

(прізвище, ініціали)

Рецензент

(прізвище, ініціали)

Допускається до захисту

Зав. кафедри

(підпис)

Дудар З.В.

(прізвище, ініціали)

2016 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук  
Кафедра програмна інженерія  
Освітньо-кваліфікаційний рівень бакалавр  
Напрям підготовки 6.050103 Програмна інженерія  
(шифр і назва)

ЗАТВЕРДЖУЮ:  
Зав. кафедри \_\_\_\_\_  
(підпис)  
«\_\_\_\_» \_\_\_\_\_ 2016 р.

**ЗАВДАННЯ**  
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Шпетному Дмитру Володимировичу  
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Ігровий додаток для симуляції пересування в спеціальних умовах

затверджена наказом по університету від "\_\_\_\_" \_\_\_\_\_ 2016 р. № \_\_\_\_\_

2. Термін подання студентом роботи (проекту) 10 червня 2016 р.

3. Вихідні дані до роботи (проекту) розробити програмну систему для аналіза аудіосигналів та демонстраційний рівень для показу можливостей системи.  
Використовувати технології Unity, C#, Python, UML моделювання.

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) мета роботи, аналіз користувальницьких і розробка функціональних вимог до програмного продукту, опис прийнятих проектних рішень, методи та алгоритми, що використовувались, опис роботи застосування, тестування та аналіз дослідної експлуатації. Додатки: а) слайди презентації, б) приклади програмних кодів, в) електронні матеріали до проекту на CD.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, мініатюр, плакатів) ілюстрації до алгоритмів, діаграма частотних характеристик музичних інструментів, зображення конкуруючих програмних продуктів, діаграма розподілу ринка операційних систем, зображення структури грального рушія, діаграма патерну гральний цикл, схема патерну MVC, структура пакету UML, діаграма діяльності, діаграма прецедентів, ілюстрація реалізованого алгоритму, екранні форми графічного інтерфейсу

---

---

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	доц. Турута О.П.		

7. Дата видачі завдання « 18 » квітня 2016.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів дипломного проекту	Термін виконання етапів проекту (тижень)	Примітка
1	Аналіз предметної галузі	18.04.16 – 24.04.16	виконано
2	Розробка специфікації ПЗ	20.04.16 – 28.04.16	виконано
3	Об'єктний аналіз поставленої задачі	27.04.16 – 30.04.16	виконано
4	Створення коду програми	01.05.16 – 15.05.16	виконано
5	Тестування і налагодження програми	12.05.16 – 19.05.16	виконано
6	Підготовка пояснівальної записки:	20.05.16 – 27.05.16	виконано
7	Підготовка презентації та доповіді	25.05.16 – 30.05.16	виконано
8	Нормоконтроль, рецензування		виконано
9	Попередній захист		виконано
10	Занесення диплома в електронний архів		виконано
11	Допуск до захисту у зав. кафедри		виконано

Студент \_\_\_\_\_

(підпис)

Керівник роботи (проекту) \_\_\_\_\_ доц., Турута О.П.

## РЕФЕРАТ

Пояснювальна записка: 54 стор., 38 рис., 16 дж., 3 дод.

Метою роботи є аналіз існуючих алгоритмів обробки аудіосигналів, їх використання у ігрових цілях та створення програмної реалізації системи для комплексного аналізу аудіосигнала.

Метод розробки – об'єктно-орієнтований підхід до розробки програмного додатку, метод прототипування , застосування паттернів розробки.

В результаті була розроблена ігрова програмна система для аналізу та обробки аудіосигналів та застосування їх при автоматичній генерації рівнів.

**ПРОГРАМНА СИСТЕМА, АНАЛІЗ АУДІО, UNITY, ВІДЛЕННЯ ТАКТИВ, ВІДЛЕННЯ ПІКІВ, ПРОЦЕДУРНА ГЕНЕРАЦІЯ.**

Explanatory note: 54 p., 38 pic., 16 src., 3 ap.

The aim is to analyze the current state of audio processing algorithms, their usage in game development, software implementation of complex audio signal analysis.

Method of the development - an object-oriented approach to application software development, applying of design patterns, prototyping approach.

As a result, a software system for audio analysis and processing was created. It was used for procedural level generation.

**SOFTWARE SYSTEM, AUDIO PROCESSING, UNITY, BEAT DETECTION, ONSET DETECTION, PROCEDURAL GENERATION.**

## СОДЕРЖАНИЕ

Введение.....	5
1 Анализ предметной области.....	6
1.1 Процесс анализа аудиоданных.....	6
1.2 Анализ игрового жанра.....	15
1.3 Анализ аналогов.....	16
1.4 Постановка задачи.....	20
2 Проектирование программного обеспечения.....	21
2.1 Программные средства.....	21
2.2 Архитектура программного обеспечения.....	23
2.3 UML - моделирование программной системы.....	26
3 Описание реализации.....	32
3.1 Описание клиентской части.....	32
3.2 Описание пользовательского интерфейса.....	36
4 Тестирование.....	39
Выводы.....	41
Перечень источников.....	42
Приложение А Слайды презентации.....	44
Приложение Б Листинг части кода.....	52
Приложение В Электронные материалы (CD)	

## ВВЕДЕНИЕ

Обработка сигналов — область прикладной математики, которая исследует теорию преобразования цифровых и аналоговых сигналов, изменяющихся во времени или пространстве. Под обработкой сигналов подразумевают математические, статистические, вычислительные, эвристические и лингвистические аспекты, формализации и методы для представления, моделирования и анализа. С увеличением количества электроники увеличивается и количество оцифрованных сигналов различного типа таких как аудио-, видеосигналов и прочих. В то же время наиболее растущим рынком программного обеспечения является индустрия видеоигр, где использование музыки и аудио в различных проявлениях является неотъемлемым аспектом. Некоторые жанры акцентируют свое внимание на соответствие аудио и игрового процесса.

В большинстве случаев, проектирование игровых уровней является задачей геймдизайнера, который основываясь на своем опыте и понимании задачи перед игроками выстраивает локацию и набор заданий, соответствующий общему настроению игры в данный момент.

В то же время данный процесс может быть автоматизирован при помощи анализа аудиосигнала и использовании данных как основу для проектирования уровня. Такие особенности музыки как ритмический рисунок, темп, пиковые моменты используются в таких жанрах игр как ритм-игры, где игроку обычно необходимо сделать какое-то действие в такт музыке.

Рынок не насыщен такими играми. Хотя попытки создать подобные игры предпринимались давно (первой игрой в жанре считается "Simon" 1978 года), но на текущий момент в игровой индустрии процент аудио-ориентированных игр невелик. Так на площадке Steam всего 51 такая игра. Часть из них используют свои прегенерированные аудио файлы, и лишь немногие дают пользователю возможность использовать свой трек, который будет каким-либо образом обработан. Это связано со сложностью анализа данных в связи со субъективностью восприятия музыки человеком.

Таким образом, задача по обработке аудио сигнала является актуальной в современных условиях и будет рассмотрена в данной работе.

Целью данной работы является создание игровой системы анализа аудиоданных и демонстрации возможности применения данных на демонстрационном уровне.

## 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

### 1.1 Процес анализа аудиоданных

Началом анализа аудиоданных можно считать работы математика Жан Батиста Жозефа Фурье, который доказал, что любую функцию удовлетворяющую условиям непрерывности во времени, периодичности и условиям Дирихле можно представить в виде ряда, который получил название ряда Фурье.

На практике разложение периодических функций в ряд Фурье широко используется, например, в задачах теории цепей: несинусоидальное входное воздействие раскладывают на сумму синусоидальных и рассчитывают необходимые параметры цепей, например, по методу наложения.

Разложение в ряд Фурье позволяет разложить непрерывную функцию в сумму других непрерывных функций. В общем случае, ряд будет иметь бесконечное количество членов. Дальнейшим усовершенствованием подхода Фурье является интегральное преобразование. В отличие от ряда Фурье, преобразование Фурье раскладывает функцию не по дискретным частотам, а по непрерывным.

Спектр преобразования Фурье — в общем случае, функция комплексная, описывающая комплексные амплитуды соответствующих гармоник. То есть, значения спектра являются комплексными числами, чьи модули являются амплитудами соответствующих частот, а аргументы — соответствующими начальными фазами. На практике, рассматривают отдельно амплитудный спектр и фазовый спектр, который представлен на рисунке 1.1.

Из показанного на рисунке графика видно, что коэффициенты ряда Фурье являются ни чем иным, как значениями преобразования Фурье в дискретные моменты времени. Однако, преобразование Фурье сопоставляет непрерывной во времени, бесконечной функции другую, непрерывную по частоте, бесконечную функцию — спектр. В случае с дискретными функциями (представлением цифрового аудио сигнала) используется дальнейшее развитие преобразования Фурье — дискретное преобразование Фурье (ДПФ).

Дискретное преобразование Фурье призвано решить проблему необходимости непрерывности и бесконечности во времени сигнала. По сути, сигнал считается бесконечным, где значимая часть — это анализируемый сигнал, а остальное пространство временной оси заполнено нулевым сигналом, который для анализируемых данных представляет собой вещественный ноль. Математически это означает, что, имея исследуемую бесконечную во времени функцию  $f(t)$ , в процессе анализа она умножается

на некоторую оконную функцию  $w(t)$ , которая обращается в ноль везде, кроме интересующего интервала времени.

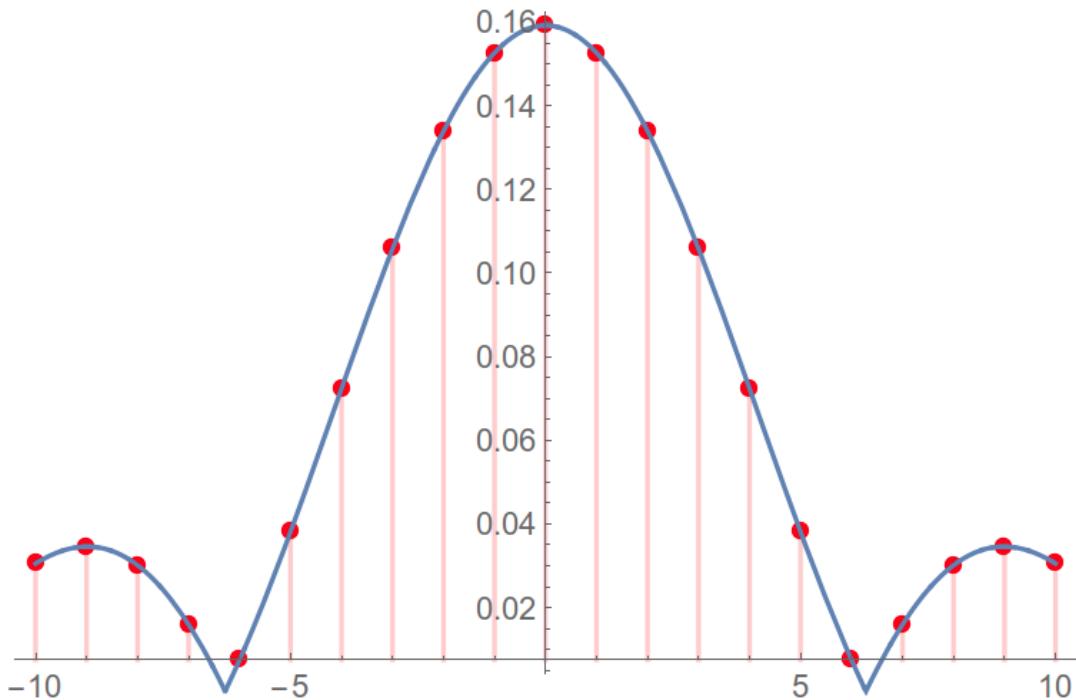


Рисунок 1.1 – Соответствие ряда Фурье и преобразования Фурье на примере амплитудного спектра

Однако в это случае изменяется и получаемое в результате преобразования значение. Результатом дискретного преобразования является не спектр-функция, а дискретный спектр.

Для успешного определения темпа, ритма, пиков и начала атаки в музыке необходимо вычленить спектр, что позволит работать с конкретными инструментами [1]. В основном ноту определяют по четырем следующим фазам:

- началу атаки;
- атаке;
- транзиенте;
- угасанию.

Эти фазы, а также исходный аудиосигнал представлены в схематическом виде на рисунке 1.2.

Большинство алгоритмов использует подход выбора фаз транзиент, поскольку данный этап характеризуется всплеском амплитуды, изменением в кратковременном промежутке спектра. Атакой считается интервал увеличение амплитуды аудиопакета.

Транзиентой называют кратковременный промежуток, во время которого сигнал быстро развивается в направлении, которое нельзя предугадать. Для большинства акустических инструментов транзиента соотносится с возбуждением сигнала, например, ударом молотка по струне в случае фортепиано. Согласно психоакустическим исследованиям, человеческое ухо не может выделить две транзиенты в случае, если они находятся в интервале 10 и меньше миллисекунд [2]. В таком случае, началом атаки может считаться минимальный момент, в который транзиента может быть надежно определена.

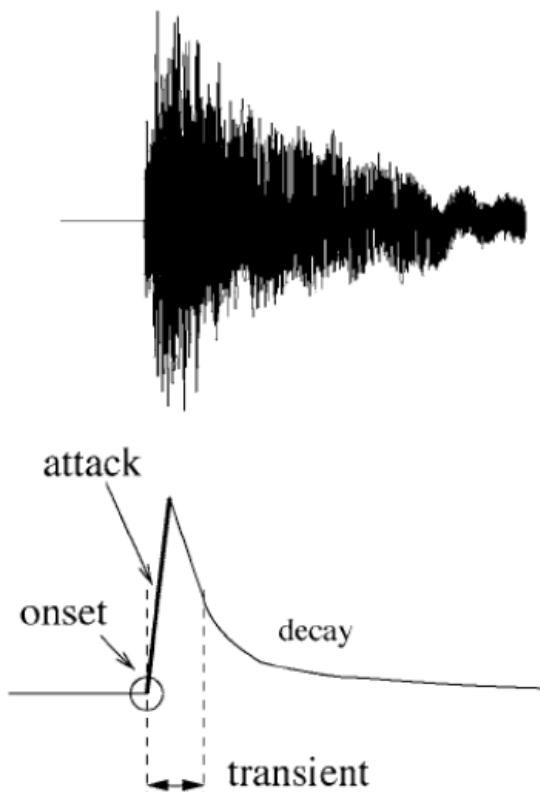


Рисунок 1.2 – Графическое представление пика и начала атаки

В общем случае, подход к анализу пиков в сигнале можно поделить на несколько этапов, которые приведены на рисунке 1.3.

Первый этап — начальная обработка звука. В общем данный этап служит для выделения необходимой части обработки. Например, аудио сигнал может быть разбит на подсигналы по частотным характеристикам; каждый из сигналов оценивается с некоторым коэффициентом влияния на общую характеристику. Также к сигналу могут

быть применены различные модуляции и фильтры для отсеивания шума такие как компрессия (dynamic range compression). Данный этап считается optionalным поскольку сильно зависит от алгоритма поведения дальнейших этапов.

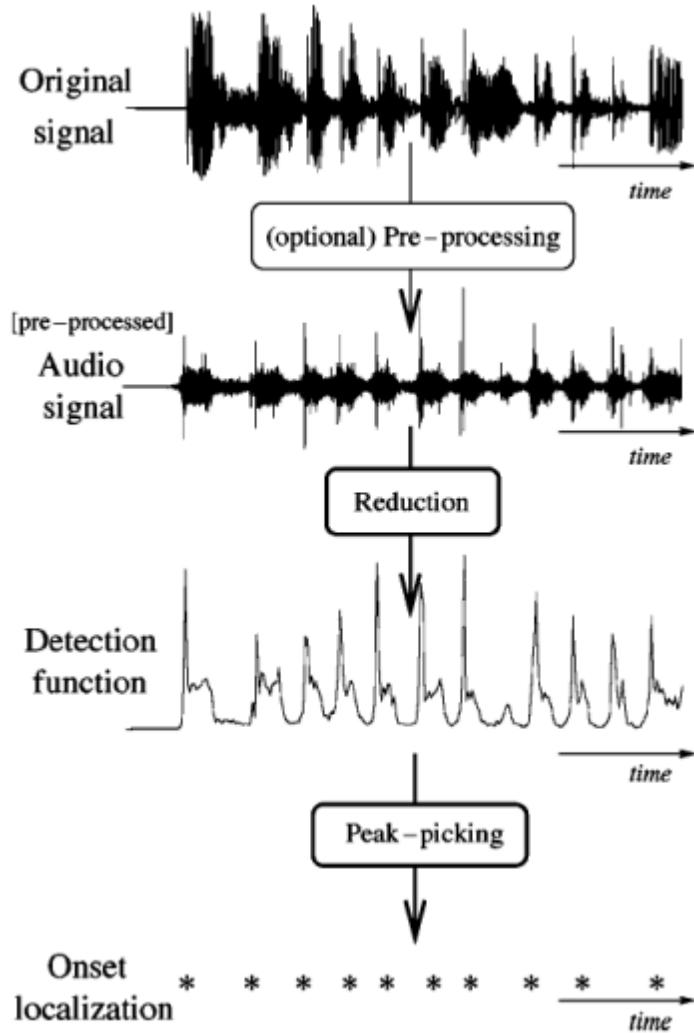


Рисунок 1.3 – Этапы анализа пиков аудио сигнала

Второй этап — редукция. Служит для получения сжатой модели структуры сигнала. Таким образом упрощается задача нахождения музыкальных событий и их начала в сигнале. Данные анализируются не целиком, а во временном промежутке-окне, размер которого зависит от частоты дискретизации аудио и желаемого коэффициента гранулярности анализа, обычно находимого эмпирическим путем [2]. Для данного этапа можно использовать не спектральный, а амплитудный анализ. Недостатки использования такого подхода состоят в том, что нельзя выделить какой-то конкретный инструмент, задающий темп. Для большинства жанров в качестве ритмической основы используют

басовые и ударные, которые имеют свой диапазон частот [3]. Схематически это изображено на рисунке 1.4.

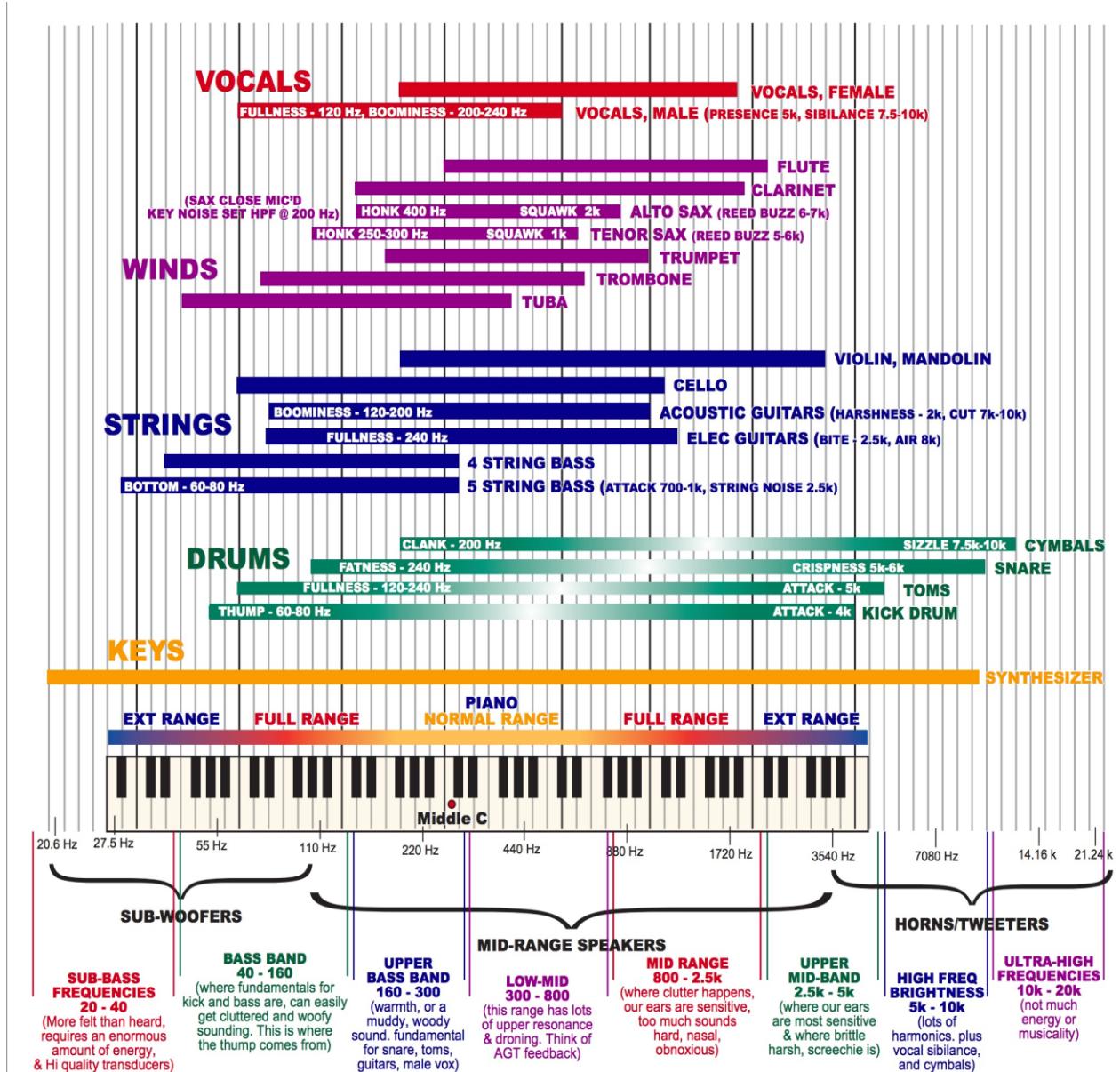


Рисунок 1.4 – Диаграмма частот инструментов

Редукцию можно проводить используя следующие подходы [1]:

- используя характеристики сигнала (временные и спектральные);
- используя вероятностную модель (две соревнующиеся модели и приближение неожиданного момента).

Рассмотрим анализ на основании характеристик сигнала. В случае временной (также известной как темпоральной) характеристики используют подход основанный на

изменении энергии. Для этого определяется производная энергии по времени и производная логарифма энергии [1].

$$\frac{d(\log E)}{dt} = \frac{1}{E} \frac{dE}{dT} \quad (1.1)$$

Преимущества спектральной характеристики в меньшей необходимости предварительной обработки сигнала и лучшей работы в полифонической среде. Рассмотрим трансформацию Фурье сигнала  $x(n)$  [1]:

$$X_k(n) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(nh + m)w(m)e^{-\frac{2j\pi mk}{N}} \quad (1.2)$$

где  $w(m)$  – окно размера  $N$ ,  $h$  – размер преобразования Фурье.

В спектральном домене увеличение энергии транзиент часто представляется как широкополосное событие. Поскольку энергия сигнала обычно сконцентрирована в нижнем диапазоне частот, изменения транзиент более заметны в высоких частотах [3]. Учитывая этот факт, спектр может быть оценен с большим весом высоких частот для получения энергетической совокупности изменения.

Заключительный этап — выборка пиков. На данном этапе могут быть применены различные статистические и эмпирические методики. В общем случае, пиком считается сигнал, значительно большей амплитуды соседних сигналов. Также могут быть учтены такие факторы, как глобальный темп, что больше влияет на восприятие человека, чем действительные пики звука [4].

Таким образом, нахождение звуковых пиков, транзиент, начала атаки и их амплитуды позволяет найти общий темп музыкального фрагмента и изменения в энергии, характеризующие общее развитие звучания.

Эти данные в дальнейшем могут быть использованы для независимого анализа в рамках игрового процесса. Так, например, время и расстояния между тактами может быть использовано в игровых целях для синхронизации скорости и качества действий игрока в зависимости от проигрываемого на фоне игрового процесса аудио фрагмента.

Совокупность подвергнутых анализу данных может быть применена для построения и настройки игровых механик для создания более полного эффекта присутствия.

Отдельно можно выделить процесс выделения ритмических ударов-тактов. Применяется похожая идеология как и при анализе энергии сигнала. Однако эта задача гораздо сложнее предыдущей, поскольку очень просто прочувствовать каждому человеку, но при этом сложно формализовать данный процесс. Поэтому приведенные ниже алгоритмы получают лишь приближенное представление происходящих процессов с определенной долей погрешности, которая варьируется от одного сигнала к другому.

В реализованном алгоритме применяется идея отслеживания локального максимума пиков и отслеживания среднего значения на определенном интервале, что дает неплохие результаты в простейших случаях [4]. Однако в общем случае существует два ограничения на определение такта. С одной стороны, они должны отражать неизменную в локальном промежутке серию значений, которые и определяют понятие ритма в целом. С другой стороны, выбранные в каждый конкретный момент значения должны соответствовать музыкальному событию в сигнале, например, ноте сыгранной на инструменте [2]. В качестве основы для данного алгоритма были взяты статистически лучшие алгоритмы по определению такта, которые сочетают в себе вышеуказанные принципы анализа. Предположим, что задан конечный темп произведения и необходимо рассчитать времена тактов, которые соответствуют пикам в сигнале. Тогда применима функция, достигающая указанных требований.

$$C(\{t_i\}) = \sum_{i=1}^N O(t_i) + \alpha \sum_{i=2}^N F(t_i - t_{i-1}, \tau_p) \quad (1.3)$$

где  $\{t_i\}$  это последовательность из  $N$  найденных локальных тактов,  $O(t)$  — так называемый пакет силы транзиент, найденный из аудио, который велик для моментов когда выбор такта хорош в следствии акустических характеристик,  $\alpha$  — коэффициент взвешенности для балансировки двух взаимосвязей и  $F(\Delta t, \tau_p)$  это функция для измерения временного соответствия междутактового интервала  $\Delta t$  и идеального расстояния между тактами  $\tau_p$ , который определяется исходя из конечного темпа. К примеру, можно применить такую функцию простой квадратичной ошибки примененную к

логарифмическому значению полученных расстояний между тактами и идеального расстояния, заданного темпом.

$$F(\Delta t, \tau) = - \left( \log \frac{\Delta t}{\tau} \right)^2 \quad (1.4)$$

Функция принимает максимальное значение 0 когда  $\Delta t = \tau$  и становится значительно меньше нуля для более крупных отклонений. Она также симметрична на логарифмической оси времени так, что  $F(k\tau, \tau) = F(\tau/k, \tau)$ . В следствии этого, предполагаем, что время было квантифицировано по некоторому значению, к примеру, с шагом в 4 миллисекунды или с частотой дискретизации в 250 Герц.

Ключевым преимуществом данной функции является простота использования ее в рекурсивном подходе для нахождения лучшего результата  $C^*(t)$  всех последовательностей времени  $t$  [4].

$$C^*(t) = O(t) + \max_{\tau=0 \dots t} \{ \alpha F(t - \tau, \tau_p) + C^*(\tau) \} \quad (1.5)$$

Данное уравнение основано на наблюдении того факта, что лучший результат для времени  $t$  является локальное значение силы начала атаки в сумме с лучшим значением предыдущего времени такта  $\tau$  которое максимизирует сумму лучшего результата и стоимость перехода из этого времени. Во время подсчета  $C^*$  также необходимо учитывать настоящее время предыдущего такта, который дал лучший результат:

$$P^*(t) = \arg \max_{\tau=0 \dots t} \{ \alpha F(t - \tau, \tau_p) + C^*(\tau) \} \quad (1.6)$$

Но на практике необходимо ограничить интервал  $\tau$  в связи с быстрорастущим штрафом выражения  $F$ , которое сделает маловероятным что лучшее время предыдущего такта

находится далеко от  $t - \tau_p$ , ищем  $\tau = t - 2\tau_p \dots t - \tau_p/2$ . Рассмотрим применение алгоритма на рисунке 1.5.

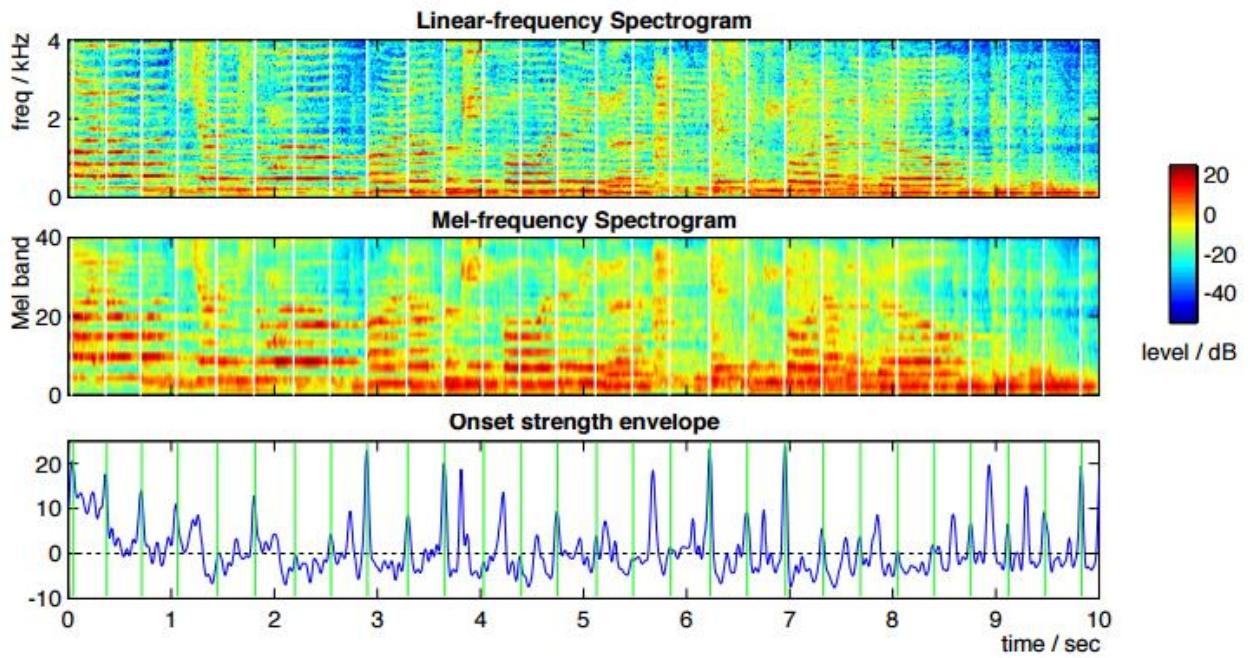


Рисунок 1.5 – Сравнение спектрограмм и силы начала атаки во времени

Как видно из приведенного изображения, времена тактов не всегда соответствуют максимальной силе и энергии сигналов, однако для поддержания ритмического рисунка важно сохранять постоянство временного промежутка.

Существенным недостатком и ограничением данного алгоритма является сильная зависимость от предопределенного идеального темпа. Конечно, уменьшение значимости темпа при определении такта при помощи весового коэффициента делает возможным поиск локальных пиков и тактов, очень сильно отличающихся от идеальных, но это все еще не позволяет использовать данный метод при определении медленно варьирующегося темпа [4]. Для решения данной проблемы необходимо динамически отслеживать и изменять  $\tau_p$ . В прочем, это не решает проблему анализа внезапного изменения темпа, что требует более существенных изменений алгоритма. В данном случае существует возможность параллельного анализа различных значений  $\tau_p$  и внедрения еще одного параметра для функции выбора лучшего результата. Такой подход улучшит результаты для аудио подобного рода, но существенно увеличит вычислительную нагрузку. Более подходящим способом является вычисление доминантного темпа заново при

существенном изменении времени получения пиков, что требует отдельного места в алгоритме.

## 1.2 Анализ игрового жанра

Игра разрабатывается в жанре «музыкальная игра». Игровой процесс большей частью ориентирован на взаимодействие игрока с музыкальной темой или отдельными аудио треками. Часто жанр объединяют с головоломкой в следствии использования ритмически генерируемых уровней.

Концепция интеграции теории игр и музыки не нова. Первые исследования можно отнести к «Musikalischs Würfelspiel» (музыкальная игра с игральными костями), где система использовала значения кубиков-костей для случайного генерирования музыкального произведения из составленных вручную частей. Первым представителем такого рода игры можно считать «Der allezeit fertige Menuetten- und Polonaisencomponist» 1757 года авторства Иоганна Филиппа Кирнбергера [5]. В прочем, данные игры не были предназначены для случайного использования. Игроку необходимо было знать базовые музыкальные концепции, математику и самостоятельно подбирать недостающие или неподходящие части. Сама игра схожа с порождающей грамматикой - формализмом генеративной лингвистики, связанной с изучением синтаксиса. В рамках подхода порождающей грамматики формулируется система правил, при помощи которых можно определить, какая комбинация слов оформляет грамматически правильное предложение.

Правила игры можно объяснить по аналогии с генерацией строки стихотворения, представленному на рисунке 1.6.

1	The	cow	ran	past	the	field.
2	The	pig	walked	through	the	yard.
3	The	sheep	ran	into	the	marsh.

Рисунок 1.6 – Таблица генерации строки стихотворения

Каждый бросок кости определяет строку, из которой на данном этапе будет выбрано слово. Каждая прогрессия одинаковая по сути, но разнится в мелких деталях.

Компьютерные аналоги подобных игр постепенно переросли в отдельный жанр. На данный момент наиболее популярным поджанром является ритм-игра, где пользователь осуществляет игровые действия с периодичностью, совпадающей с темпом музыкального произведения. Относительно популярными являются и другие поджанры такие как:

- игра на запоминание;
- игра свободной формы («Freeform music game»), где создание аудио является целью игры, часто в совокупности с открытым миром. Часто представители поджанра близки к профессиональному программному обеспечению для синтезирования и обработки аудио;
- гибридные, где элементы аудио приемов вплетаются в игру другого жанра. Часто звуковые эффекты взаимодействия игрока и окружающей среды заменяются на изменения музыкальной темы, что побуждает использовать набор действий, подходящий к субъективному восприятию гармонии звучания. Отдельно выделяют поджанр игр, в которых аудио используется для определения геймплея - динамики, темпа и других не относящихся к музыке компонентов игры.

### 1.3 Анализ аналогов

Одной из подобных игр является Audiosurf. Это серия аркадных музыкальных игровых программ-головоломок. У игрока в распоряжении есть супермобиль (как гласят игровые тексты) и трасса, которую необходимо пройти, набрав как можно больше очков. Игра создана независимым разработчиком Invisible Handlebar — компанией Дилана Фитерера. Игра распространялась в качестве бесплатной бета-версии, а 15 февраля 2008 года состоялся запуск полного издания через систему Steam. Название игры является совмещением английских слов audio — звукозапись, трек и surf — скользить по волнам. Вместе эти два слова образуют фразу «Скользить по волнам музыки», что очень точно отражает игровой процесс.

В марте 2008 года Audiosurf стала самой продаваемой игрой на Steam, обогнав по продажам The Orange Box, Counter-Strike и несколько сотен других игр.

Хотя поначалу игра кажется гонкой, на деле это головоломка. После выбора одного из дюжины корабликов игрок должен выбрать трассу. В Audiosurf трассами служат песни, причём пользователь может выбрать любой трек, находящийся на его жёстком диске.

Каждая трасса представляет собой ленту, изогнутую в некоем пространстве согласно ритму выбранной музыки; на ленте есть три колеи с блоками. Фигуры, как и кораблик, движутся вперёд, но с заметно меньшей скоростью. Каждая выбранная пользователем аудиодорожка анализируется игрой, в результате создаётся и сохраняется в специальный ASH-файл, содержащий сведения о динамике звука, геометрии трассы и расположении блоков и ассоциированный с песней (его размер составляет около 30 килобайт). Это позволяет ускорить время повторной загрузки того же звукового файла. Высота и форма трассы отражают динамику проигрываемой музыки. Например, если игрок выбрал медленную и тихую песню, трасса будет идти в гору, движение будет медленным, окружение наполнено холодными цветами. Если была выбрана интенсивная и громкая песня, то во время игры кораблик будет спускаться с горы с большой скоростью, а трасса будет заполнена блоками с большим числом тёплых цветов, что изображено на рисунке 1.7.



Рисунок 1.7 – Игровой процесс AudioSurf

Другая похожая игра - Crypt of the NecroDancer. Это независимая видеоигра, разработанная студией Brace Yourself Games и вышедшая для операционной системы Windows 23 апреля 2015 года. В качестве основы игры взят жанр «roguelike»,

характерными особенностями которого являются генерируемые случайным образом уровни, пошаговость и необратимость смерти персонажа. Часто используется изометрическая проекция, нарочитая двухмерность и стилизация под ретро.

Особенностью игры является использование ритм-паттернов саундтрека. Действия игрока наиболее эффективны, когда выполняются в такт музыки и наказываются штрафом случае не попадания.

В качестве контроллера можно использовать танцевальную платформу — плоский игровой контроллер, используемый в танцевальных играх. Большая часть таких платформ выполнена в виде матриц 3 на 3 квадрата, на которые может становиться игрок. В остальном игра представляет собой типичный «dungeon crawler», где игрок проходит подземелья попутно сражаясь с противниками и подбирая награды для усиления. С прохождением игры усложняется и саундтрек, наращивая темп и заставляя игрока быстрее принимать решения. Интерфейс игры представлен на рисунке 1.8.

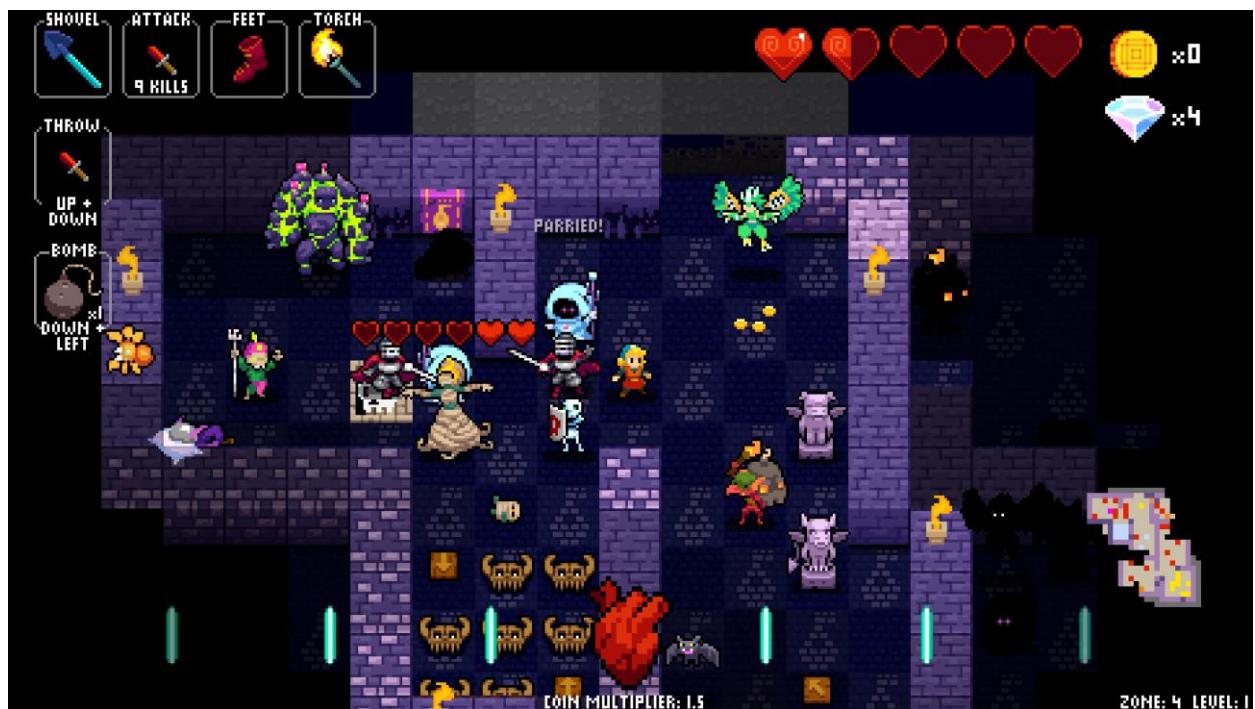


Рисунок 1.8 – Игровой процесс Crypt of the NecroDancer

Игра osu! – это бесплатная музыкальная ритм-игра, написанная Dean "peppy" Herbert'ом в 2007 году.

Игровой процесс состоит в нажатии появляющихся на экране нот, ведению мячика по слайдерам, а также вращению спиннера максимально быстро. В зависимости от

точности попадания в такт музыки, начисляются очки, за неправильное выполнение действий у игрока отнимаются «жизни», по истечению которых засчитывается поражение.

Игра поддерживает многопользовательский режим и в ней используются следующие игровые элементы:

- ноты — представлены в виде кругов, по которым необходимо нажимать.
- слайдеры — нужно провести мячик по определённой траектории.
- спиннера — необходимо раскрутить «спиннер» как можно быстрее для получения очков.

Для игры, помимо клавиатуры и компьютерной мыши, могут быть использованы планшетный компьютер и графический планшет. Графический интерфейс изображен на рисунке 1.9.

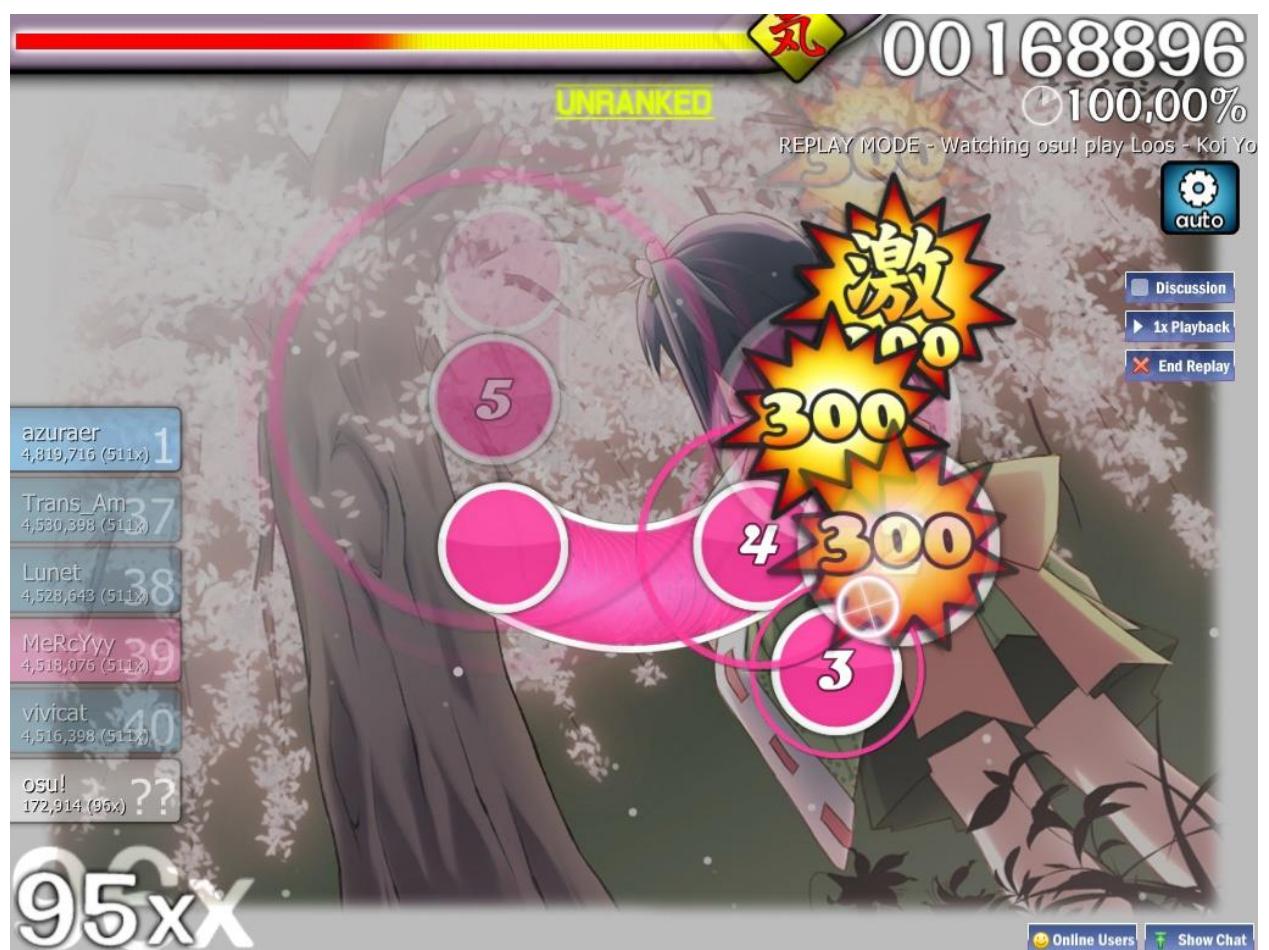


Рисунок 1.9 – Игровой процесс osu!

Существенным недостатком игры является необходимость пользователям составлять карты уровней, которые создаются на выбранное музыкальное произведение

при помощи встроенного внутриигрового редактора. С одной стороны, это позволяет создавать различные наборы уровней под один и тот же фрагмент, что вносит разнообразие и безусловно является преимуществом, но с другой стороны возлагает ответственность на сообщество на заполнение игры контентом. Таким образом, неправильно составленная карта уровня сводит на нет соответствие аудио и видео ряда, что нарушает целостность и задумку игры. Полученная карта никаким образом не валидируется и не верифицируется, не существует автоматически генерируемого каркаса, что значительно усложняет пользовательскую разработку и возлагает на человека теоретически выполняемую программно работу.

#### 1.4 Постановка задачи

Целью данной работы является создание игровой системы анализа аудиоданных и показа возможности применения данных на демонстрационном уровне. Создание программной системы состоит из пунктов:

- анализ алгоритмов обработки сигналов;
- разработка программного комплекса анализа аудиоданных;
- применения проанализированных данных в контексте игры.

Для написания программной системы был выбран игровой движок Unity и языки программирования C# и Python. Для разработки активно использовался платформенный класс AudioSource, содержащий необходимые исходные звуковые данные. Данный класс является единственной альтернативой для проигрывания аудиофайлов в данном игровом движке.

Также необходимо произвести анализ и проектирование программного обеспечения используя язык UML.

## 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### 2.1 Программные средства

Для написания данной работы был выбран игровой движок Unity, который позволяет разрабатывать приложения, работающие под операционными системами Windows, OS X, Windows Phone, Android, Apple iOS, Linux, а также на игровых приставках Wii, PlayStation 3, PlayStation 4, Xbox 360, Xbox One. Приложения, созданные с помощью Unity, поддерживают графические библиотеки DirectX и OpenGL.

Предоставляемый инструментарий бесплатен для некоммерческой разработки и поддерживает большой набор технологий, в частности для написания игровой логики могут быть использованы следующие языки программирования: C#, JavaScript, Boo.

C# - это объектно-ориентированный язык программирования. Он был разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270. Относится к семье языков с С-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML. Unity официально поддерживает фреймворк .NET 2.0, что дает определенную свободу действия при написании приложения. В следствии кроссплатформенности некоторые вызовы платформенных функций заменены на движковые вызовы Unity.

JavaScript - прототипно-ориентированный сценарный язык программирования. Является реализацией языка ECMAScript (стандарт ECMA-262). Обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса. Как и в случае C# код выполняется не привычным браузером, а средой Unity. Но при этом не проходит процесс компиляции и сопутствующие ему этапы, что негативно сказывается на вероятности ошибок и производительности [6].

В процессе анализа альтернатив был выбран C#, как компилируемый и строго типизированный язык, что положительно скажется на производительности и минимизирует количество ошибок, отсеиваемых на этапах компиляции кода [7].

Расчеты физики производит физический движок PhysX от NVIDIA. Это кроссплатформенный физический движок для симуляции ряда физических явлений, а также комплект средств разработки (SDK) на его основе. Первоначально разрабатывался компанией Ageia для своего физического процессора PhysX. NVIDIA адаптировала движок для ускорения физических расчётов на своих графических чипах с архитектурой CUDA. PhysX может также производить параллельные высокоеффективные вычисления с использованием обычного процессора. В настоящее время PhysX доступен на следующих платформах: Windows, Linux, Mac OS X, Wii, PlayStation 3, Xbox 360 (аппаратное ускорение возможно только на платформе Windows).

Также сообщество разработчиков активно развивается, в следствии чего существует большое количество материалов и данных для разработки. Не в последнюю очередь это связано с удобством использования сторонних разработок — изначально существует унифицированная платформа распространения Unity Asset Store.

Основной платформой разработки была выбрана операционная система Windows поскольку под управлением операционных систем семейства Windows по данным ресурса NetMarketShare работает более 91% персональных компьютеров, что представлено на рисунке 2.1.

## Desktop Operating System Market Share

April, 2016

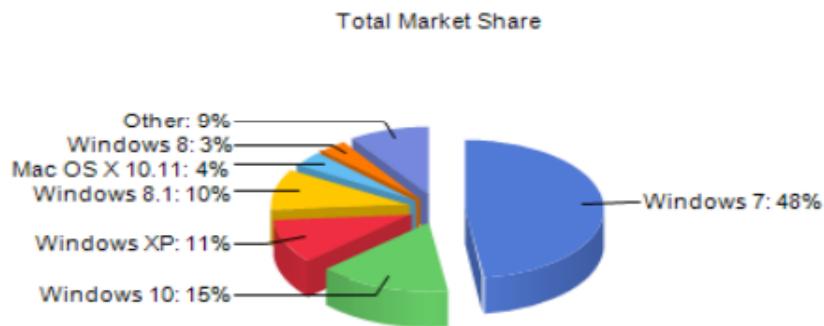


Рисунок 2.1 – Операционные системы персональных компьютеров

Однако, использование кроссплатформенного движка позволяет с минимальными изменениями выпустить игру для различных устройств и операционных систем.

Помимо основной разработки в среде Unity для быстрого прототипирования был выбран Python. Это высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули (они в свою очередь могут быть объединены в пакеты). Решение про использование этого инструмента разработки было обусловлено необходимостью визуализировать часть аудиоданные, что в кратчайшие сроки можно было сделать при помощи сторонних библиотек. Также для языка написано большое количество математических библиотек. Анализ существующих реализаций позволил найти слабые и сильные стороны алгоритмов.

В качестве системы контроля версий была выбрана система git, позволяющая легко отслеживать изменения и вести учет правок в полуавтоматическом режиме, что положительно сказывается на удобстве работы [8].

## 2.2 Архитектура программного обеспечения

В качестве архитектурного архитектуры была выбрана многослойная система, обеспечиваемая игровым движком. Большинство современных движков имеют общие архитектурные концепции, поскольку большинство из них проверены временем и опробованы на практике.

Например, хорошей практикой считается абстрагирование от платформенных вызовов даже в случае поддержки одной платформы, что позволит с гораздо меньшими усилиями добавить поддержку новой операционной системы или платформы.

Также наиболее производительным считается модульный подход при проектировании. Таким образом можно понизить связность логически независимых компонентов, что обеспечивает высокую скорость разработки и повышает вероятность возможности оптимизации компилятором частых вызовов. Однако следует помнить, что

чем больше функционала предоставляет движок, тем медленнее данный функционал будет работать при конкретных настройках. В общем случае, в каждом современном движке можно выделить такие компоненты как на рисунке 2.2.

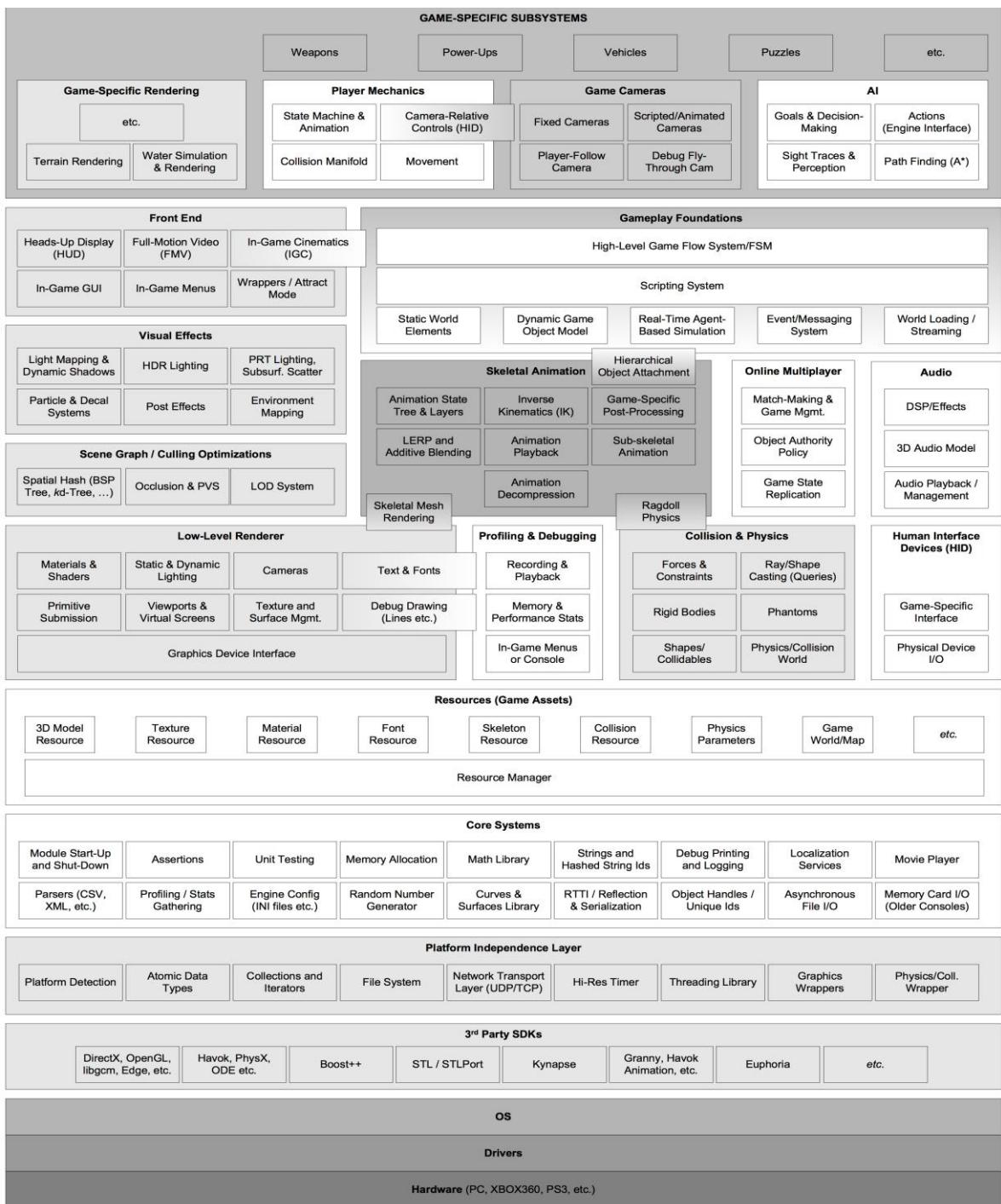


Рисунок 2.2 – Архитектура игрового движка

Например, полный отказ от скелетной анимации невозможен, поскольку по умолчанию предусмотрен движком. Следовательно появляются ненужные накладные

расходы, что следует учитывать при дальнейшей оптимизации производительности и потреблении других вычислительных ресурсов.

Unity поддерживает все необходимые компоненты для создания 3D игры и на уровне движкового рендера предусмотрены оптимизации для шлемов виртуальной реальности.

Как и во всех играх, в данной работе был применен паттерн Игровой цикл (Game loop), задача которого состоит в том, чтобы устранить некоторые зависимости игрового времени от пользовательского ввода и скорости процессора. Шаблон схематически показан на рисунке 2.3.

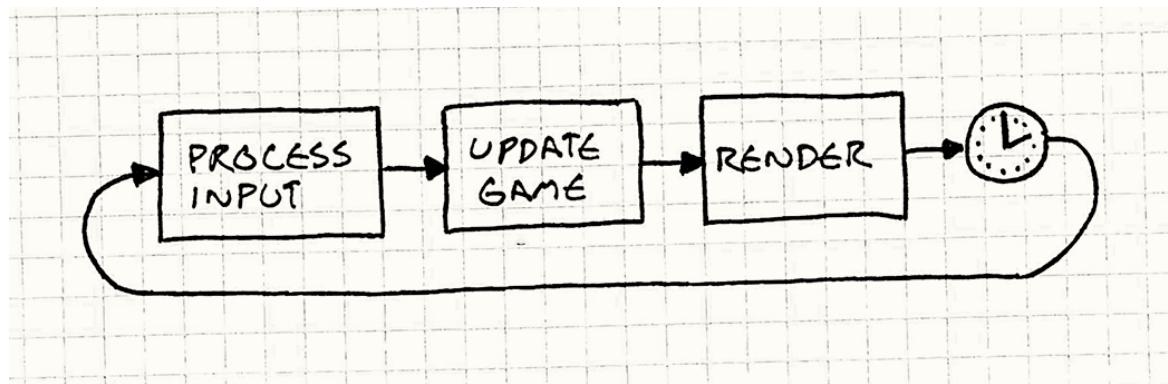


Рисунок 2.3 – Игровой цикл

Этот цикл является важнейшей частью игрового программного обеспечения, так как именно он выполняется на протяжении 90% всего времени. Скорость, а также плавность игры зависит непосредственно от быстродействия и количества выполнения функций из цикла в секунду.

Архитектура разрабатываемого приложения также относится к событийному типу, поскольку зависящие от пользователя действия происходят только при его взаимодействии с игровым контроллером. Данная схема позволяет абстрагировать логику работы игры и ее функциональность, что позволяет легко модифицировать игровые правила и окружение.

В процессе разработки работы были использованы идеи шаблона MVC. В таком шаблоне модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные.

Полный цикл работы данной MVC-триады: модели, представления и контроллера переключателя – можно описать следующим образом. При инициализации представления

пользователем оно обращается к модели и устанавливает текст метки в соответствии с текущим состоянием переключателя. Пользователь инициирует изменение переключателя, нажимая на определенную кнопку. При этом представление отправляет соответствующую команду контроллеру: включить или выключить. Контроллер интерпретирует команду и изменяет модель. Представление регистрирует изменение модели: по этому событию оно изменяет текст метки для соответствия новому состоянию модели переключателя. Диаграмма такого шаблона представлена на рисунке 2.4.

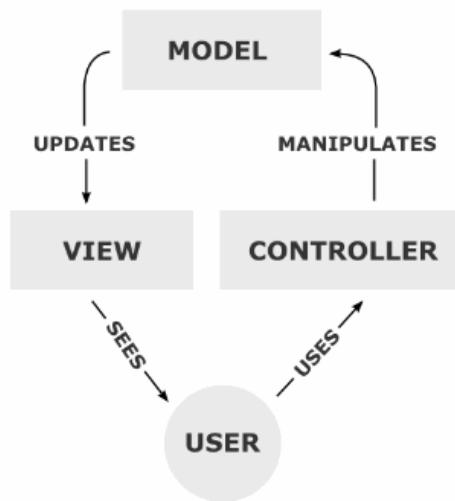


Рисунок 2.4 – Шаблон MVC

Преимущества данного подхода в возможности легко модифицировать логику работы (модель) не меняя представления (вида) и использовать другие пользовательские обработчики событий (контроллеры) не связываясь с другими частями.

### 2.3 UML - моделирование программной системы

UML – это унифицированный графический язык моделирования для описания, визуализации, проектирования и документирования ОО систем. UML призван поддерживать процесс моделирования ПС на основе ОО подхода, организовывать взаимосвязь концептуальных и программных понятий, отражать проблемы масштабирования сложных систем. Модели на UML используются на всех этапах жизненного цикла ПС, начиная с бизнес-анализа и заканчивая сопровождением системы.

Разные организации могут применять UML по своему усмотрению в зависимости от своих проблемных областей и используемых технологий [9].

Также UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

Он также позволяет разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий (таких как класс, компонент, обобщение, агрегация и поведение) и больше сконцентрироваться на проектировании и архитектуре.

В UML стандарта версии 2.3 используются и стандартизирован большой набор диаграмм, которые делятся на такие категории:

- структурные диаграммы;
- диаграммы поведения;
- диаграммы взаимодействия.

Различные типы диаграмм и их классовая принадлежность представлены на рисунке 2.5.

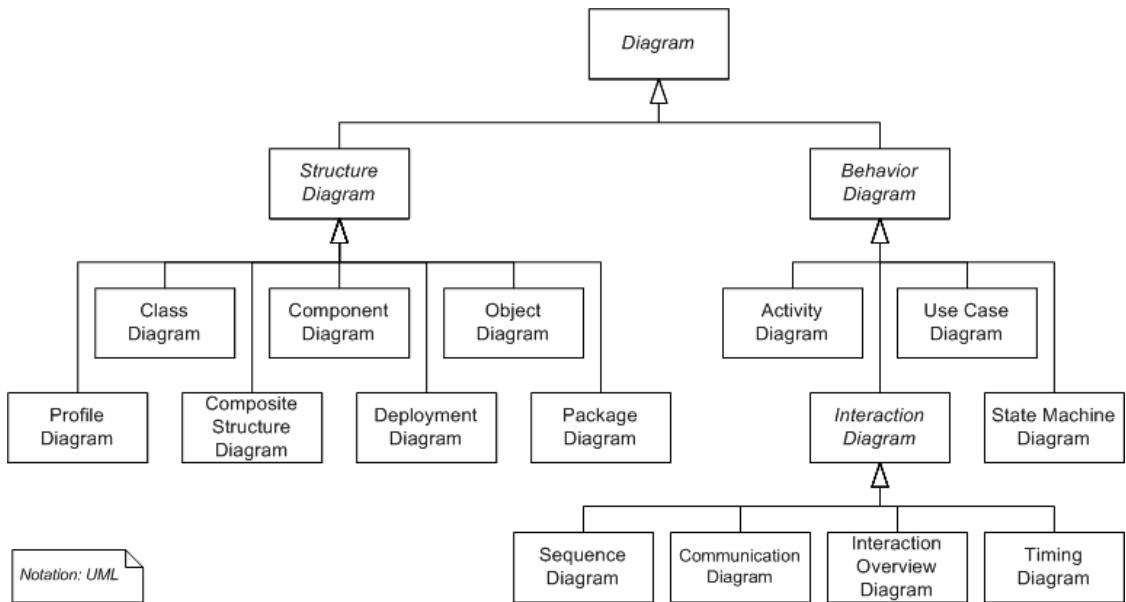


Рисунок 2.5 – Структура пакета UML

Они характеризуют систему с разных сторон, позволяя наглядным и лаконичным образом охарактеризовать общую структуру и архитектуру программного приложения,

цели, достигаемые при помощи использования продукта, перечислить и формализовать различных бизнес-пользователей продукта, ознакомиться с процессом и набором правил при использовании данных.

В качестве диаграммы поведения была выбрана диаграмма сценариев использования (use case diagram). Это диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствуетциальному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних требований к системе, однако четкое понимание требований к системе также важно и для внутреннего анализа и проектирования [10].

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

При моделировании системы с помощью диаграммы прецедентов необходимо чётко отделить систему от её окружения, определить действующих лиц (актёров), их взаимодействие с системой и ожидаемый функционал системы, и определить в глоссарии предметной области понятия, относящиеся к детальному описанию функционала системы (то есть, прецедентов).

При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов не учитываются.

В данной диаграмме используются следующие элементы:

- рамки системы — это прямоугольник с названием в верхней части и эллипсами (прецедентами) внутри. Часто может быть опущен без потери полезной информации;
- актёр — это стилизованный человечек, обозначающий набор ролей пользователя (понимается в широком смысле: человек, внешняя сущность, класс, другая система), взаимодействующего с некоторой сущностью (системой, подсистемой, классом). Актёры не могут быть связаны друг с другом (за исключением отношений обобщения/наследования);
- прецедент — это эллипс с надписью, обозначающий выполняемые системой действия, которые могут включать возможные варианты, приводящие к наблюдаемым актёрам результатам. Надпись может быть именем или описанием (с точки зрения

актёров) того, «что» делает система (а не «как»). Имя прецедента связано с непрерываемым (атомарным) сценарием — конкретной последовательностью действий, иллюстрирующей поведение. В ходе сценария актёры обмениваются с системой сообщениями. Сценарий может быть приведён на диаграмме прецедентов в виде UML-комментария. С одним прецедентом может быть связано несколько различных сценариев. Часть дублирующейся информации можно устраниТЬ путём указания связей обобщения, включения или расширения.

Диаграмма вариантов использования для данной индивидуальной работы представлена на рисунке 2.6.

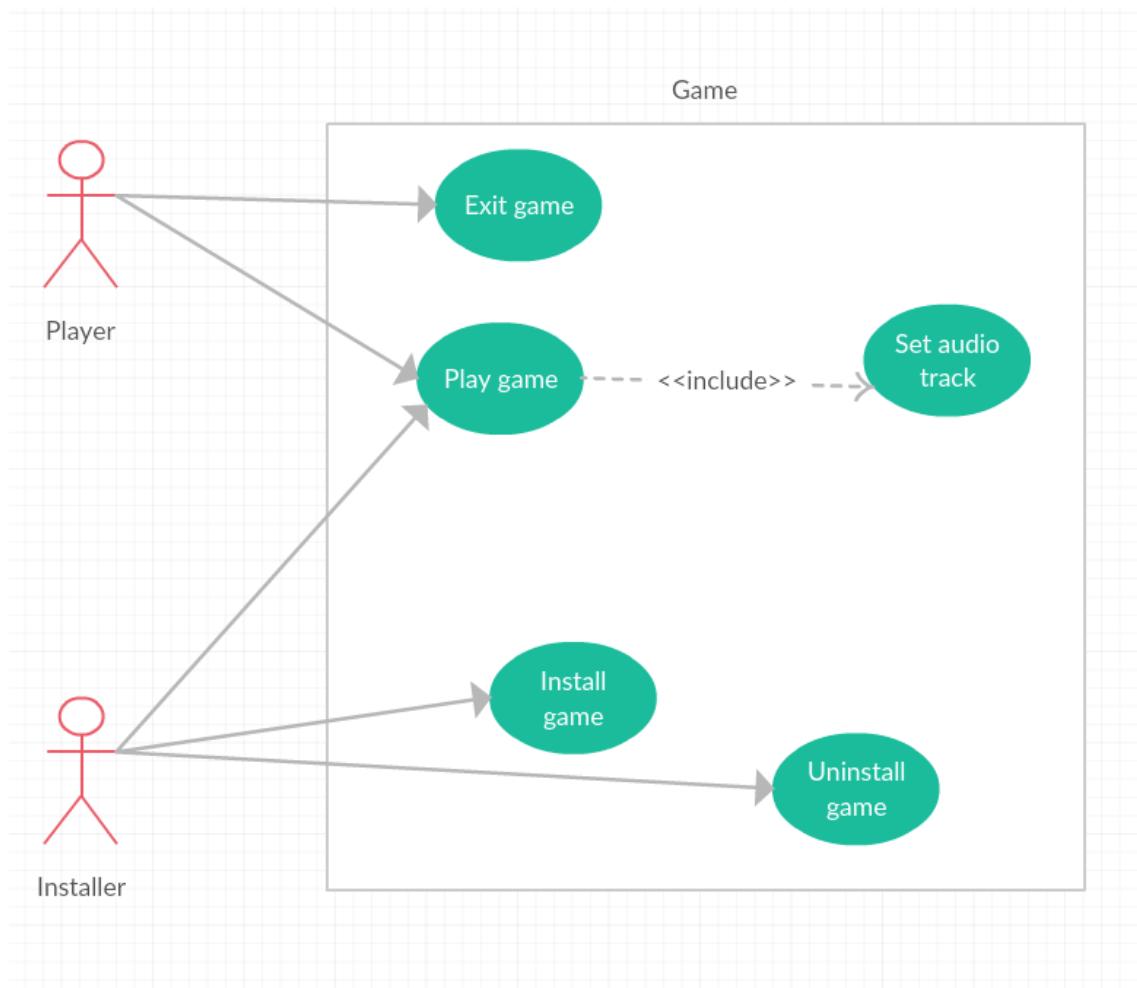


Рисунок 2.6 – Use case диаграмма приложения

Пользователи разбиты на два логических участника — непосредственно игрок и игровой установщик. Они оба имеют возможность начать игру. Непосредственно частью

игры является выбор аудиотрека, что отображено на диаграмме соответствующим отношением включением компонентов.

Также для показа была выбрана диаграмма активности. Это вид UML-диаграммы, на которой показано разложение некоторой деятельности на её составные части. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий англ. *action*, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений. Диаграмма данной работы изображена на рисунке 2.7.

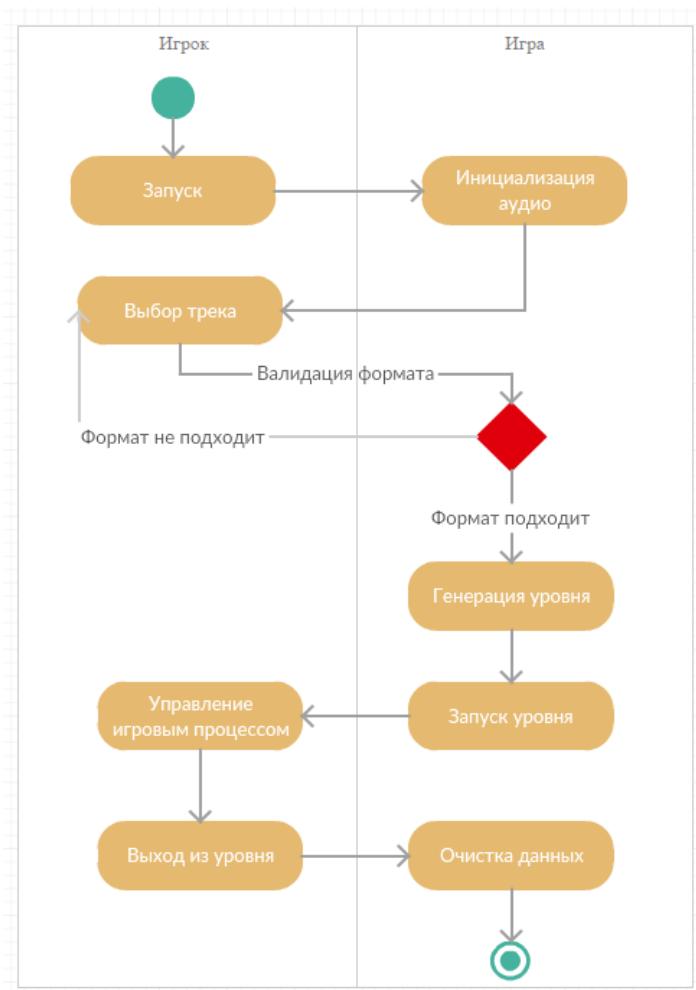


Рисунок 2.7 – Диаграмма активности

На данной диаграмме рассмотрен более низкий логический уровень непосредственного взаимодействия с программным продуктом. Игрок инициирует запуск

приложения; приложение в свою очередь дает пользователю выбрать аудио файл для дальнейшей обработки и анализа. Затем проводится ряд первичных валидаций, в частности валидация формата, накладываемая реализацией аудио модуля игрового движка Unity текущей версии. В случае успеха процедурно генерируется игровой уровень и игрок получает возможность действия обратно для управления процессом игры. В случае непрохождения валидации игроку будет выдано сообщение об ошибке формата выбранного файла и заново будет предложено сделать выбор. После окончания игрового процесса происходит очистка памяти в виде удаления скопированных аудио файлов, которые использовались во время их обработки, и выход из приложения.

Таким образом, процесс проектирования позволил четко формализовать требования к программному обеспечению и показать основные сценарии использования. И хотя реализация вариантов использования диаграммами является наиболее трудоемким и сложным методом, но этот метод лучше всего согласован с объектно-ориентированным подходом и в наибольшей мере приближает к конечной цели, а моделирование использования – это универсальный способ представления функциональных требований к программному продукту.

Диаграмма активности, в свою очередь, показала игровой процесс на более низком уровне, обеспечив таким образом более глубокое понимание игровой сессии и выполнение действий системы шаг за шагом.

### 3 ОПИСАНИЕ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

#### 3.1 Описание реализации клиентской части

Для реализации клиентской части было выбрано язык C# и интегрированную среду разработки Microsoft Visual Studio 2015 Express Edition от компании Microsoft. Выбор среды разработки связан с поддержкой игрового движка Unity интеграции с IDE. Помимо этого Visual Studio – это практически единственный способ отладки приложения в реальном времени.

При работе над системой анализа аудио было принято решение минимизации зависимостей и дополнительных фреймворков для повышения производительности игры и уменьшения вероятности проблем работы на разных платформах. Для данных целей был использован компонент Audio Source (источник звука), который воспроизводит компонент Audio Clip в сцене. Аудио клипы содержат непосредственные данные аудио. Unity поддерживает моно, стерео и мультиканальные звуковые ассеты (до восьми каналов). На текущий момент времени Unity может импортировать следующие типы файлов: .aif, .wav, .mp3, and .ogg. Также, Unity может импортировать трекерные модули из следующих типов файлов: .xm, .mod, .it, and .s3m.

Unity поддерживает различные сценарии воспроизведения звука. Так, если Audio Clip является 3D клипом, источник проигрывается в заданном положении в пространстве и будет приглушаться в зависимости от расстояния. Аудио может быть распределено по колонкам (например, из стерео в 7.1) с помощью свойства Spread и трансформироваться между 3D и 2D с помощью свойства PanLevel. Можно контролировать зависимость этих эффектов от расстояния с помощью кривых затухания. Также если слушатель находится в одной или нескольких зонах реверберации, то к источнику применяются реверберации. Для обогащения аудио ряда, к источнику можно применять отдельные аудио фильтры.

Для реализации системы были использованы только данные аудио выборки, взятые из источника аудио и представленные в виде вещественных чисел в диапазоне от -1 до 1 включительно. Помимо массива данных для анализа регулируются следующие параметры системы:

- размер анализируемого блока в единицах выборки квантования (sample). Этот параметр может быть равен только степени двойки в следствии реализации алгоритма быстрого преобразования Фурье. Параметр влияет на гранулярность анализа, что непосредственно связано с особенностями слухового восприятия человеком [11]. В общем и целом параметр оказывает значительное влияние на процесс анализа сигнала в

следствии искусственного ограничения размера теоретически бесконечного во времени сигнала [12]. Если выбрать слишком маленький размер, то можно столкнуться с проблемой локального минимума [13], при выборе слишком большого размера теряется точность и скорость анализа. Так, например, размер выборки в 1024 сэмпла равняется порядка одной секунде аудио при частоте дискретизации 44100 Гц, что показывает оптимальные результаты в большинстве жанров музыки [14];

- количество анализируемых блоков за один проход функции. Важный параметр с точки зрения оптимизации производительности. Для целей данной работы не принципиальный фактор, поскольку анализ выполняется до передачи игрового управления пользователю, а вычислительные мощности потенциальной аудитории одни из лучших в сегменте вычислительных устройств;

- количество частотных разбиений. Это свойство позволяет разбить спектр блока на части по их частотной характеристике. Таким образом можно анализировать звуки разной частоты, что позволяет выделить пик на низких, средних и высоких частотах, что может быть полезно для отсеивания шума и дальнейшей оптимизации работы алгоритма;

- чувствительность детектора тактов. Константа, служащая для отсеивания ложных тактов-шумов. Эмпирическим путем для данного алгоритма работы выбрано значения в промежутке от 1.3 до 1.6 включительно [15];

- размер выборки блоков для вычисления среднего значения. Важнейший критерий, определяющий значение локального максимума для подсчета энергии звуковой волны в данном блоке [16]. Слишком большое значение приведет к невозможности проанализировать пики в приемлемом для человеческого восприятия временном диапазоне. В то же время слишком маленькое значение даст ложную информацию в следствии шумов в аудио (для тяжелых жанров музыки) или же в результате изменения темпа и мотива (характерно для классической музыки и джаза);

- чувствительность детектора пиков. Константа, служащая для отсеивания ложных пиков-шумов. Эмпирическим путем для данного алгоритма работы выбрано значения в промежутке от 1.3 до 1.6 включительно [15].

Данные параметры позволяют гибко настраивать систему для соответствия нуждам разработчика и непосредственного игрового процесса внедряемой игры. Ниже представлен фрагмент программного кода вычисления пиков.

```
private void CalculateFluxThresholds ()
{
    for (int i = 0; i < m_soundParser.TotalSamples; i++)
```

```

    {
        int start = Math.Max(0, i - m_thresholdSize / 2);
        int end     = Math.Min(m_soundParser.SpectralFlux.Length -
1, i + m_thresholdSize / 2);
        double mean = 0;
        for (int j = start; j <= end; j++)
            mean += m_soundParser.SpectralFlux[j];
        mean /= (end - start);
        m_threshold[i] = mean * m_thresholdMultiplier;
    }
}

```

В пределах заданной границы рассчитывается среднее арифметическое значение, которое таким образом сглаживается и в следствии чего получаем более равномерное, а значит и более приемлемое для человеческого восприятия распределение пиков. Также были опробованы и проанализированы алгоритмы определения тактов для различных жанров. Например, визуализация тактов для рок-жанра представлена на рисунке 3.1.

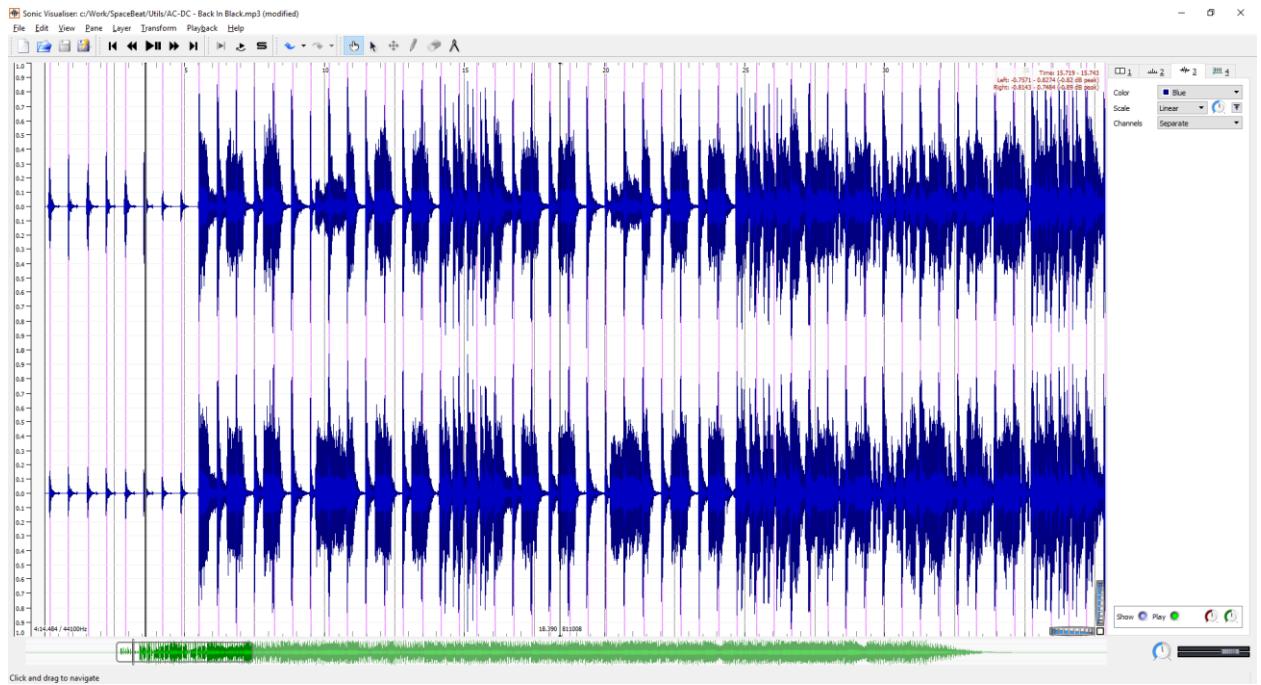


Рисунок 3.1 — Визуальное представление тактов Back in Black

Синим обозначены амплитудные представления аудио для левого и правого каналов, а вертикальные фиолетовые полосы обозначают такт музыкального

произведения. Однако для данного жанра характерен постоянный и четко уловимый ритмический рисунок. Для классической музыки ситуация гораздо сложнее в следствии радикального изменения характера мелодии и длины произведения. Рассмотрим визуализацию работы алгоритма для произведения Иоганна Себастьяна Баха «Токатта и фуга в ре минор» представленную на рисунке 3.2.

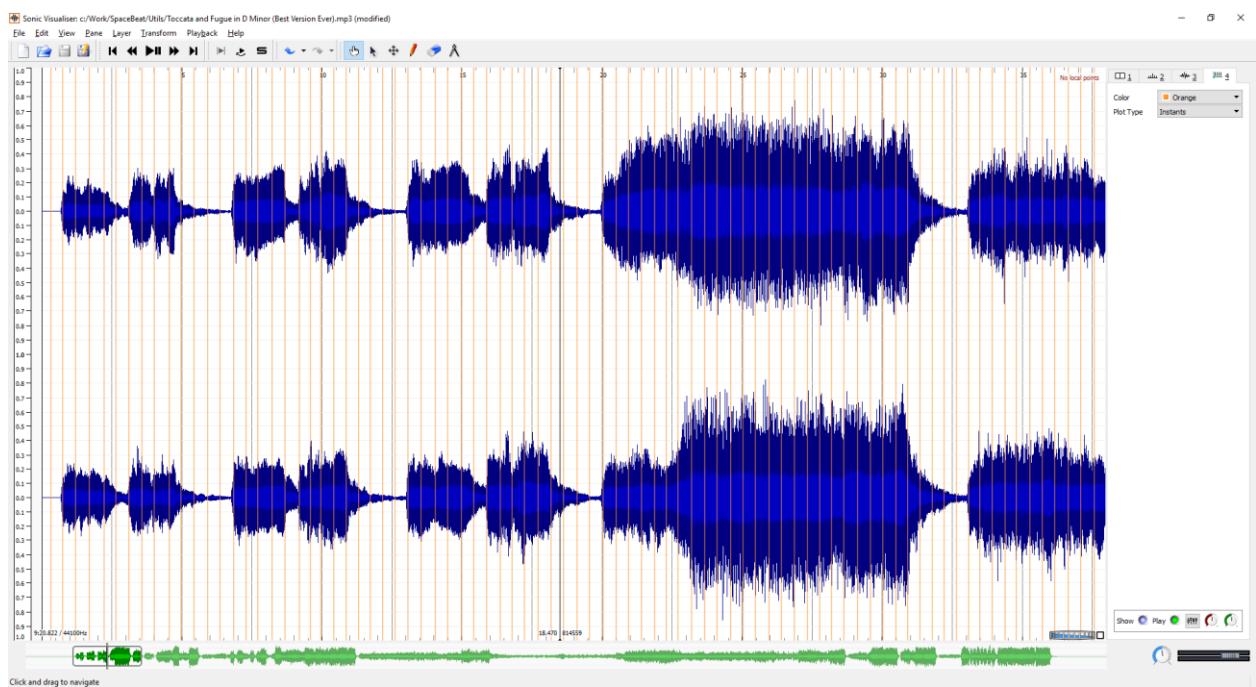


Рисунок 3.2 — Визуальное представление тактов фрагмента Токатты и фуги ре минор

Для данного произведения сложно выделить ярко уловимые такты, что не позволяет полноценно использовать музыку данного типа для ритм-игр, но отслеживание энергии музыкального сочинения все так же позволяет использовать данные полученные в результате комплексного анализа системой.

В качестве промежуточного варианта рассмотрим джазовое произведение, а именно переход от одной темы к другой с затиханием звука и постепенным изменением мотива. Данный фрагмент представлен на рисунке 3.3.

Поскольку человеку характерно воспринимать совокупность звуков, то темп оценивается не единственно логическим импульсно-моментальным образом, а как совокупность предыдущей мелодии и текущего воспринимаемого отрывка размером порядка 1 секунды во временном интервале [2]. Данный тезис нашел свое отражение в алгоритме, что видно на представленном фрагменте. Хотя темп произведения изменился,

но при анализе он оценивается в накопительной манере и предпочтение отдается постоянству ритмического рисунка.

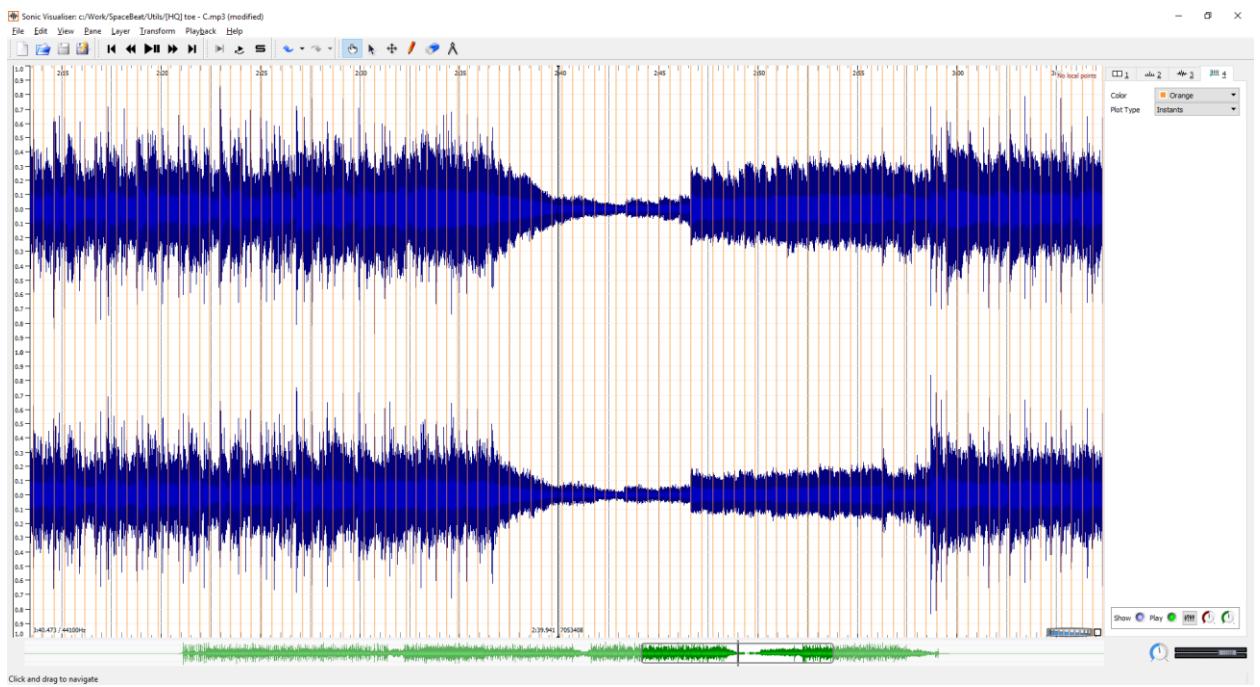


Рисунок 3.3 — Фрагмент джазового произведения

Для визуализации результатов анализа и разбора аудио-волны в данном разделе использовалась программное обеспечение с открытым программным кодом Sonic Visualiser издданное под лицензией GNU General Public License.

### 3.2 Описание пользовательского интерфейса

Интерфейс пользователя разрабатывался при помощи стандартных средств UnityEngine UI. В общем случае под пользовательским интерфейсом подразумевается разновидность интерфейсов, в котором одна сторона представлена человеком (пользователем), другая — машиной/устройством. Данный интерфейс представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными, чаще всего сложными, машинами, устройствами и аппаратурой. На текущий момент игровой движок Unity работает и поддерживает только 3D режим, что означает дополнительные накладные расходы при отсутствии необходимости третьего измерения, что актуально для таких жанров как, например, платформер. Это также касается и

пользовательского интерфейса, поскольку привычный для игроков формат — это двухмерные окна и меню, что реализовано в большинстве операционных систем и прикладных программ. Однако, не смотря на это, базовые средства разработки пользовательских интерфейсов в платформе Unity удовлетворяют требованиям задачи и являются достаточно удобными в использовании. В частности были использованы такие компоненты как полотно (canvas) — основной элемент служащий для отрисовки двухмерных объектов. На рисунке 3.4 можно увидеть пример внутриигрового пользовательского интерфейса.

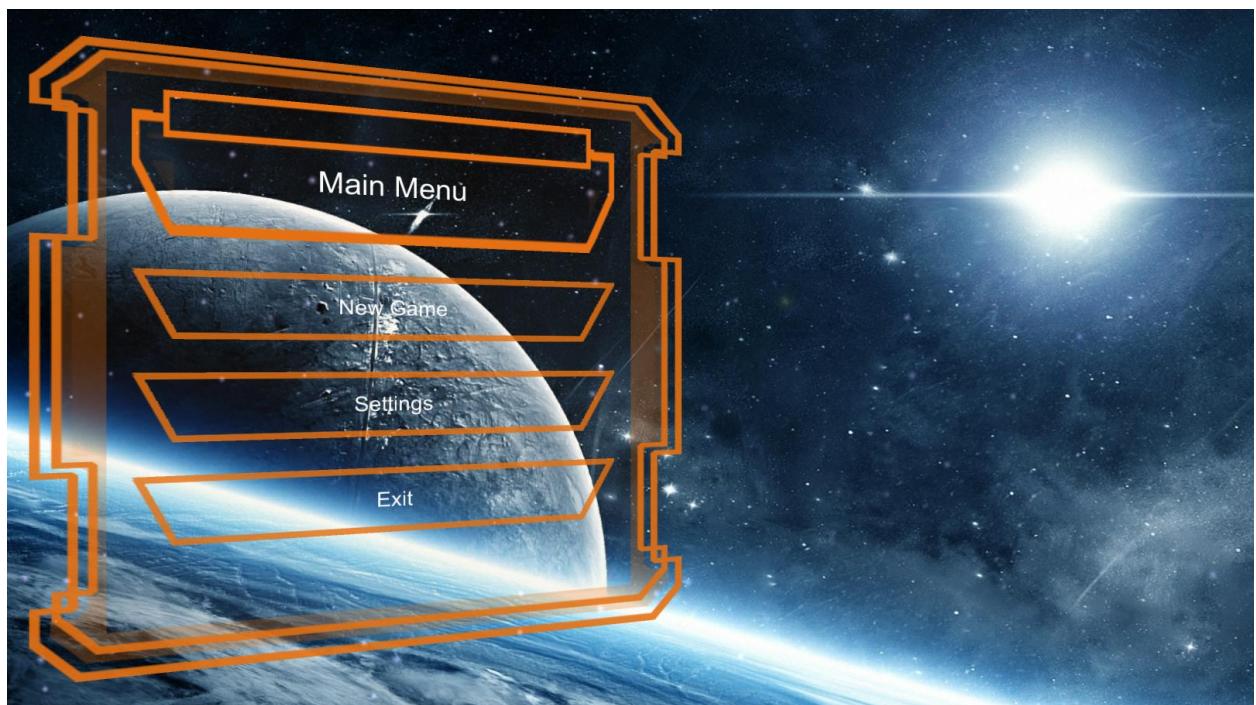


Рисунок 3.4 — Главное меню игры

Также данный интерфейс не статический, а динамический, что было достигнуто с помощью использования системы анимации, предоставляемой платформой. Как известно, на текущий момент существуют несколько направлений таких как:

- анимация по ключевым кадрам;
- скелетная анимация;
- процедурная анимация;
- анимация захвата движения (motion capture).

Также существуют и другие типы анимаций, которые не являются широко используемыми при разработке игры. Платформа Unity поддерживает все вышеперечисленные типы анимаций, однако для поставленных целей была выбрана

только анимация по ключевым кадрам, позволяющая изменять свойства объекта в зависимости от условий. При разработке интерфейса игры в качестве управляющей изменениями в объекте характеристиками было выбрано время и квадратичная кривая для плавного изменения состояния. Пример еще одного пользовательского интерфейса можно увидеть на рисунке 3.5.

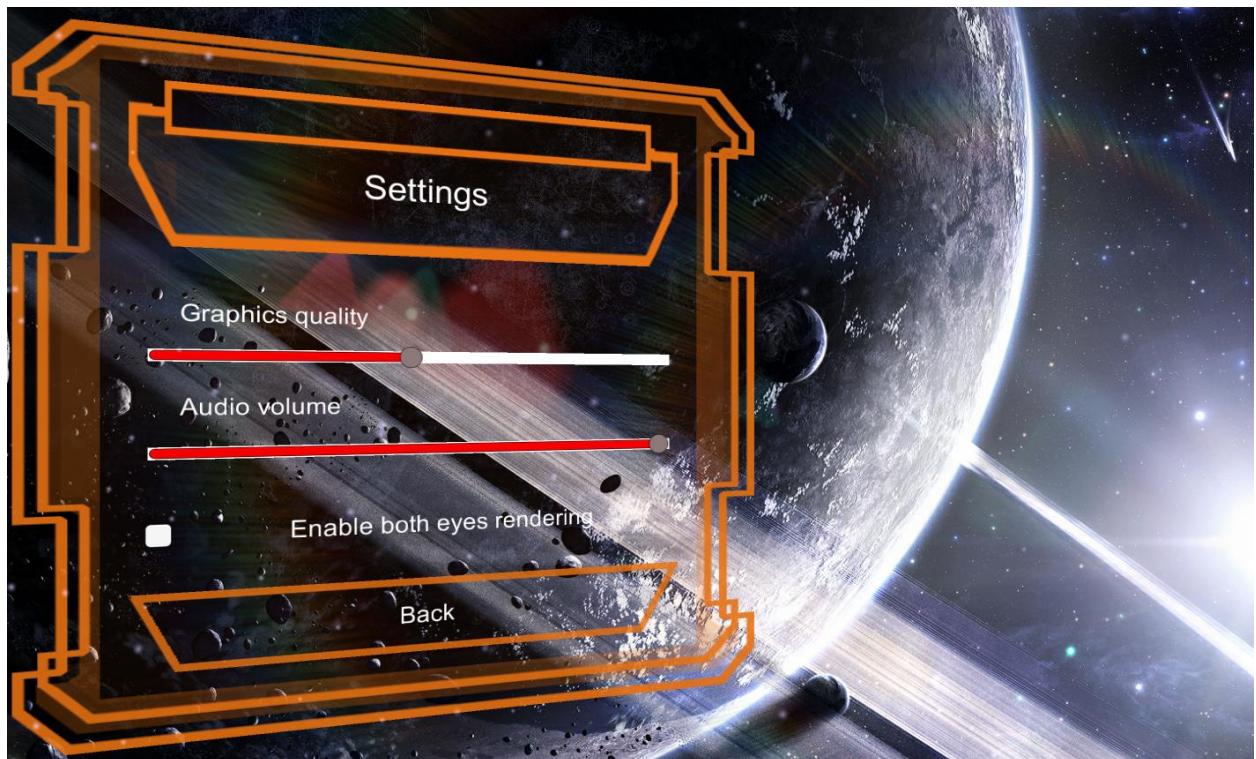


Рисунок 3.5 — Фрагмент окна настроек

На данном фрагменте видна запланированная для дальнейшей работы функция по изменению схемы работы отрисовки игрового уровня. В следствии актуальности направления виртуальной реальности в разработке программного обеспечения и популярности шлемов виртуальной реальности для игровых приложений был проведен ряд подготовительных работ для упрощения процедуры миграции при использовании платформенных средств для рендеринга в шлемы виртуальной реальности.

## 4 ТЕСТИРОВАНИЕ

В общем случае тестированием называют процесс исследования или испытания программного продукта, имеющий под собой такие цели как демонстрация разработчикам и заказчикам, что программа соответствует требованиям или выявление ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

Для данного программного продукта согласно поставленному заданию нежелательным поведением является нехарактерное человеческому восприятию выбор пиков и отдельных тактов для чего проведена мануальная проверка на соответствие такта выданного алгоритмом, пиков в сигнале, выявленных при помощи реализованного программного комплекса. Результатом работы алгоритма на выходе являются совокупность временных отметок музыкальных событий в случае для тактов и кортежа значений времени и силы пиков для алгоритма анализа пиков. Пример работы алгоритма приведен на рисунке 4.1.

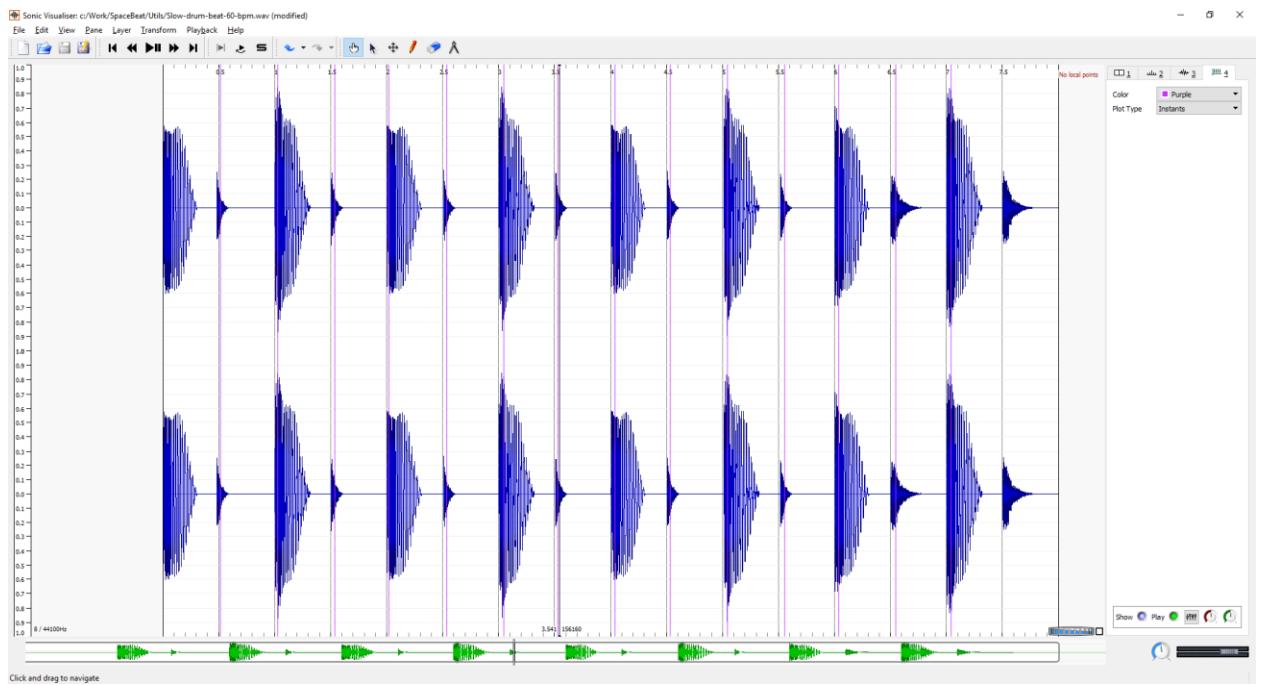


Рисунок 4.1 — Результат тактового анализа барабанного фрагмента

На приведенном фрагменте четко видно соответствие амплитуды сигнала и выбранных алгоритмом тактов, что подтверждает способность работы алгоритма.

Помимо данного фрагмента для тестирования были взяты такие виды аудио как такт метронома со скоростью сто ударов в минуту, фрагмент барабанной партии в темпе шестдесят ударов в минуту и ряд музыкальных произведений различного жанра.

Как и предполагалось на этапе анализа и проектирования, слабость алгоритма проявляется в граничных значениях, таких как, например, идеально математически поломанный такт, что проявляется в неровном расстоянии между отдельными нотами и случайной амплитуде и частоте сигнала. Данный вид сигналов используется в радарах и различного рода локаторов, поскольку позволяет отслеживать промежуток между излученным и полученным сигналом. Также алгоритм показывает не лучшие результаты в сложных музыкальных произведениях, таких как, например «Спящая красавица» Чайковского. В связи с колоссальной длительностью данного произведения в почти 4 часа раскрывается неоптимальность оптимизации алгоритма. Поскольку логически произведение разбито на 4 части с различным темпом и музыкальной идеей, то оценка такого аудио в совокупности дает не лучшие результаты.

В целом алгоритм показал свою состоятельность для большинства представителей таких жанров как поп, рок, электронная музыка и других популярных в настоящее время. Можно отметить безотказную работу алгоритма для аудио с ярко выраженным ритмическим рисунком и минимальным количеством инструментов.

Таким образом, тестирование программной системы помогло аналитическим путем оценить правильность работы и выявить ряд частей программной системы для дальнейшей работы. В результате тестирования можно вынести вердикт о правильности работы системы и соответствии результатов поставленному на этапе проектирования заданию.

## ВЫВОДЫ

В ходе данной аттестационной работы был проведен анализ предметной области и математической базы рассматриваемых алгоритмов в разделе 1.3, рассмотрены ближайшие аналоги существующих музыкальных игр в разделе 1.4, выявлены преимущества и пути возможных доработок игр. Задача для выполнения была сформулирована в разделе 1.5.

В рамках работы была спроектирована и разработана комплексная система анализа музыкальных файлов, которая может стать удобным сервисом при разработке музыкальных и ритм-игр. Помимо самой системы разработан небольшой демонстрационный уровень для показа применения проанализированных данных в реальном проекте. Во время выполнения задачи были использованы приобретенные знания из курсов “Разработка игр”, “Параллельное программирование”, “Разработка интерактивного медиа” и других. Улучшились навыки проектирования программного обеспечения, в частности составления UML-диаграмм, были рассмотрены возможности игровых движков Unity, UnrealEngine, CryEngine.

Также в рамках поставленной задачи был проведен концептуальный анализ предметной области, сравнительные достоинства и недостатки подходов в анализе аудиосигнала. Выполнена программная реализация системы анализа аудио данных и демонстрационный уровень. Конечный продукт удовлетворяет требованиям поставленной задачи.

В результате разработки удалось достичь поставленной на этапе бизнес-анализа задачи по реализации программного продукта, реализовав запланированный функционал программной системы. Система имеет удобный для разработчика интерфейс и документированный код, написанный с использованием проверенных шаблонов и практик проектирования. В частности, активно использовалась методология “чистого кода” и выдержан единый стандарт кодирования.

Однако выполненная работа имеет большие перспективы в плане дальнейших доработок. В первую очередь необходимо постоянно проводить работы по улучшению алгоритмов анализа сигнала и оценивать больше возможных характеристик помимо ритма и общей энергии. Следующей важной частью станет создание полноценной игры на базе доработанной системы с использованием технологий виртуальной реальности для соответствия тенденциям рынка и более тесной связи игрока с игровым миром.

## ПЕРЕЧЕНЬ ИСТОЧНИКОВ

1. Bello, J. P. A Tutorial on Onset Detection in Music Signals [Текст] / Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, Mark B. Sandler // University of London, 2005. – ISBN 0-78039331-720-4
2. Moore, B. C. J. An Introduction to the Psychology of Hearing, 5th ed. [Текст] / B. C. J. Moore // New York: Academic, 1997 – 268c. – ISBN 978-0125056281.
3. Rodet, X. Detection and modeling of fast attack transients [Текст] / X. Rodet, F. Jaillet // Havana, 2001, 30–33c. – ISBN 978-0132136037
4. Beat Tracking by Dynamic Programming [Электронный ресурс] / Columbia University – США – Режим доступа: <http://www.ee.columbia.edu/~dpwe/pubs/Ellis07-beattrack.pdf> – 16.07.2007 р. – Загол. з экрана.
5. Musikalisches Würfelspiel [Электронный ресурс] / Wikipedia - Режим доступа: [https://en.wikipedia.org/wiki/Musikalisches\\_Würfelspiel](https://en.wikipedia.org/wiki/Musikalisches_Würfelspiel) – 08.03.2016 р. – Загол. з экрана.
6. Фаулер, М. Архитектура корпоративных программных приложений [Текст]/ Мартин Фаулер, Дэвид Райс; – М.: Вильямс. – 544 с.
7. Мартин, Р. Чистый код. Создание, анализ и рефакторинг [Текст]/ Роберт Мартин. – М.: Питер. – 464 с.
8. Chacon S. Pro Git [Текст] / Scott Chacon – М.: Ид-во «Символ-плюс», 2015. – 1012 с.
9. Фаулер, М. UML. Основи, 3-є видання: пер. з англ./ Мартин Фаулер. – СПб: Символ-Плюс, 2004. – 192 с. – ISBN 5-93286-060-X.
10. Діаграма варіантів використання (use case diagram) [Електронний ресурс]/ UML Теорія – Режим доступу: [http://www.info-system.ru/designing/methodology/uml/theory/use\\_case\\_diagram\\_theory.html/](http://www.info-system.ru/designing/methodology/uml/theory/use_case_diagram_theory.html/). – Загол. з екрану.
11. Ono, N. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram [Текст] / N. Ono, K. Miyamoto, H. Kameoka, S. Sagayama // Tokyo : MANOGAKO, 2008. – 322 с. - ISBN 978-0233768486.
12. Shlien, S. The modulated lapped transform, its time-varying forms, and its applications to audio coding standards, 1st ed. [Текст] / S. Shlien // Boston: Academic, 1997 – 359-366c. - ISBN 978-0134555286.
13. Bello, J. P. Usage of phase and energy for musical onset detection in the complex domain [Текст] T.11. Usage of phase and energy for musical onset detection in the complex

domain: учеб. пособие / J. P. Bello, C. Duxbury, M. Davies, M. Sandler – 1<sup>st</sup> ed. –New York: Academic, 2004. – 556 с.

14. von Brandt, A. Detecting and estimating parameter jumps using ladder algorithms and likelihood ratio test [Текст] / A. von Brandt. – Boston.: MA, 1983. - 1023 с. - ISBN 978-0245788486

15. Kauppinen, I. Methods for detecting impulsive noise in speech and audio signals [Текст] / I. Kauppinen. – Greece.: Greece Mendacium, 2002. - 977 с. - ISBN 978-0264576486

16. Система запитань та відповідей Stack Overflow [Електронний ресурс] stackoverflow.com: Stack Overflow FAQ. – Режим доступу: www/URL: <http://stackoverflow.com>. – Загол. з екрану.

ПРИЛОЖЕНИЕ А  
СЛАЙДЫ ПРЕЗЕНТАЦИИ

Выпускная аттестационная работа бакалавра

Тема: “Игровая программа для симуляции передвижения в специальных условиях”

Выполнил:  
ст. гр. ПИ-12-1  
Шпетный Д.В.

Руководитель:  
доцент  
Турута А.П.

Цель работы

Анализ существующих алгоритмов обработки аудиосигналов, их использование в игровых целях и создание программной реализации комплексной системы анализа аудиосигнала.

## Аналоги



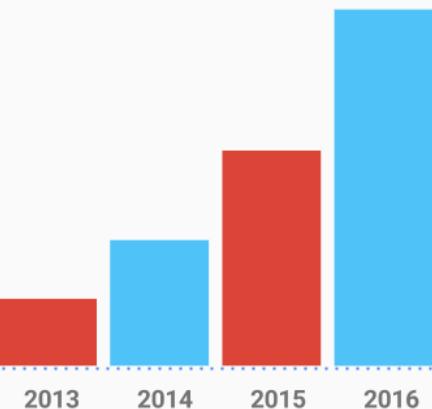
## Актуальные проблемы

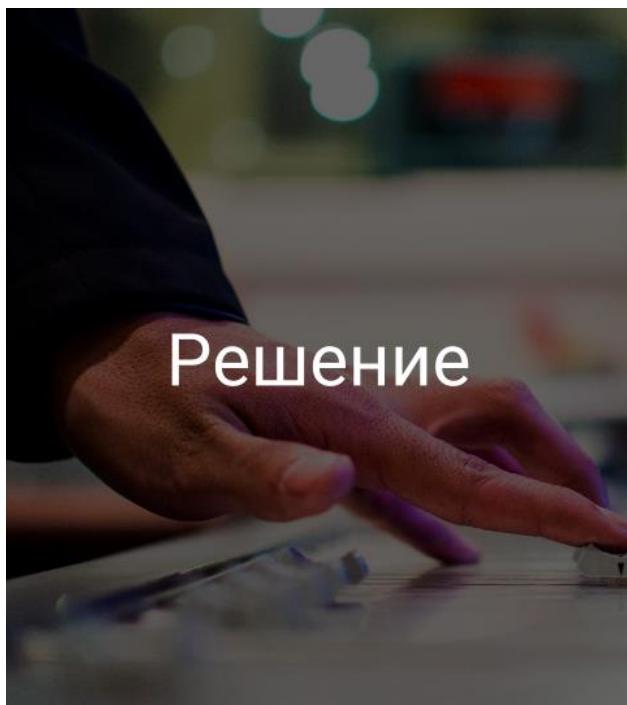
Геймплей по большей части  
завязан на координацию ритма и  
действия.

Несовершенство алгоритмов.

Малое количество игр жанра.

Отсутствие более полного  
игрового взаимодействия

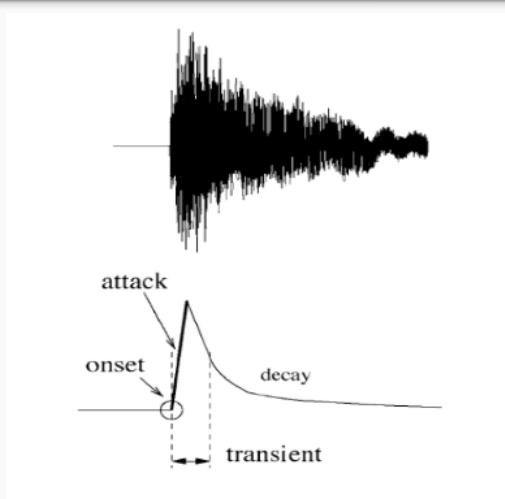




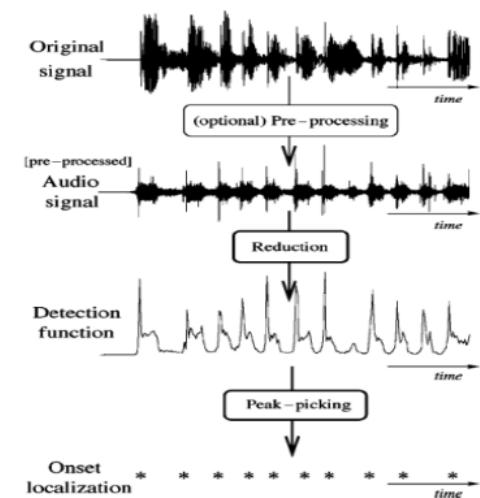
## Решение

Использовать комплексный анализ аудиосигнала, охватывающий не только анализ тактов, но и отслеживающий общую энергию произведения.

### Теоретические сведения

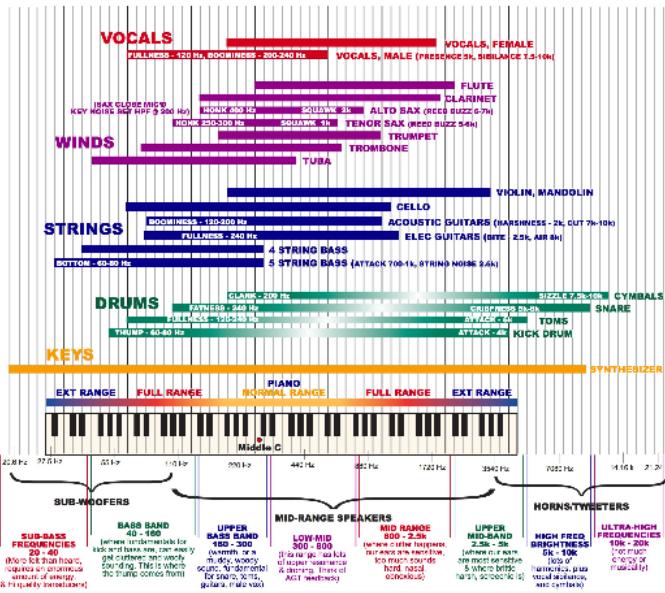


Схематическое представление события



Алгоритм анализа

## Спектр музыкальных инструментов



- Некоторые инструменты являются ключевыми для общей энергии
  - Часть инструментов сложно отслеживать в связи с наложением сигналов
  - Цифровые эффекты, накладываемые на сигнал мешают анализу

## Алгоритм анализа тактов

$$C(\{t_i\}) = \sum_{i=1}^N O(t_i) + \alpha \sum_{i=2}^N F(t_i - t_{i-1}, \tau_p)$$

где  $\{t_i\}$  это последовательность из  $N$  найденных локальных тактов,

$O(t)$  – пакет силы транзиент, найденный из аудио, который велик для моментов когда выбор такта хорош в следствии акустических характеристик,

$\alpha$  — коэффициент взвешенности для балансировки двух взаимосвязей,

$F(\Delta t, t_p)$  это функция для измерения временного соответствия междутактового интервала  $\Delta t$  и идеального расстояния между тактами  $t_p$ , который определяется исходя из конечного темпа

## Алгоритм анализа тактов

$$F(\Delta t, \tau) = - \left( \log \frac{\Delta t}{\tau} \right)^2$$

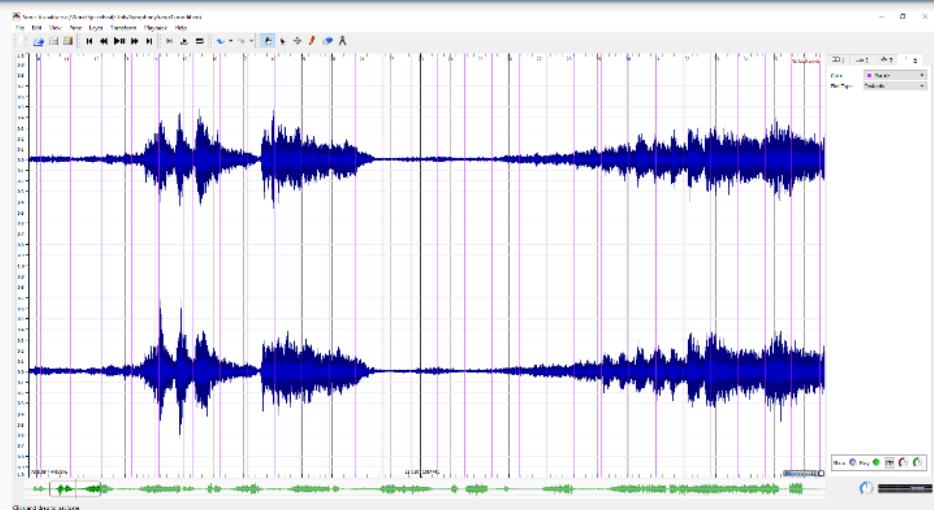
Функция принимает свое максимальное значение 0 когда  $\Delta t = \tau$  и становится значительно меньше нуля для более крупных отклонений. Она также симметрична на логарифмической оси времени так, что  $F(k\tau, \tau) = F(\tau/k, \tau)$

## Алгоритм анализа тактов

$$C^*(t) = O(t) + \max_{\tau=0 \dots t} \{ \alpha F(t - \tau, \tau_p) + C^*(\tau) \}$$

Используем локальное значение и стоимость перехода от ожидаемого идеального значения.

## Недостатки подхода



Первая часть симфонии №5 Бетховена

## Средства разработки



- Языки программирования - C#, Python
- Среда разработки - Microsoft Visual Studio 2015 Express
- Платформа разработки, игровой движок - Unity 3D

## Фрагмент программной реализации отслеживания пиков

```
private void DetectPeaks()
{
    double[] prunnedSpectralFlux = new double[m_threshold.Length];

    for (int i = 0; i < m_threshold.Length; i++)
        prunnedSpectralFlux[i] = Math.Max(0, m_soundParser.SpectralFlux[i] - m_threshold[i]);

    for (int i = 0; i < prunnedSpectralFlux.Length - 1; i++)
    {
        if (prunnedSpectralFlux[i] > prunnedSpectralFlux[i + 1])
            m_peaks[i] = prunnedSpectralFlux[i];
        else
            m_peaks[i] = 0;
    }
}
```

Для человеческого уха пиком в аудио можно считать то, что больше предыдущего значения.

## Реализация на демонстрационного игрового уровня

Желательно  
доделать еще перед  
вставкой картинки

## Выводы

- проведен анализ алгоритмов обработки аудиосигнала
- рассмотрены преимущества и недостатки аналогов
- использован комплексный подход в анализе музыкального произведения
- реализована программная система комплексного анализа аудиосигнала
- разработан демонстрационный уровень

ПРИЛОЖЕНИЕ Б  
ЛИСТИНГ ЧАСТИ КОДА

```

Класс анализатора музыки

using System;
using System.Linq;
using UnityEngine;

namespace SpaceBeat.Sound
{
    public class MusicAnalyzer
    {
        public SoundParser m_soundParser;

        private float m_thresholdMultiplier;
        private double[] m_threshold;
        private double[] m_peaks;
        private double m_sumOfFluxThresholds;
        private int m_thresholdSize;

        /// <summary>
        /// Initializes a new instance of the <see cref="SpaceBeat.Sound.MusicAnalyzer"/>
        class.

        /// MusicAnalyzer creates array of 3d track heights (0..1) analyzing sound parameters.
        /// </summary>
        /// <param name="sound">Main audioClip object for analyzing soundwave</param>
        /// <param name="sampleSize">Size of block in samples (can be 1024, 2048, 4096
        etc.)</param>
        /// <param name="soundFeed">How many blocks analyzed in one Update's method
        call</param>
        /// <param name="beatSubbands">"How many spectrum divisions are made</param>
        /// <param name="beatSensitivity">Beat sensitivity of Beat Detector</param>
        /// <param name="thresholdSize">Threshold size</param>
        /// <param name="thresholdMultiplier">Threshold multiplier</param>
        public MusicAnalyzer(

```

```

    AudioClip sound,
    int sampleSize = 1024,
    int soundFeed = 40,
    int beatSubbands = 3,
    double beatSensitivity = 1.5,
    int thresholdSize = -1,
    float thresholdMultiplier = 1.5f
)
{
    m_thresholdMultiplier = thresholdMultiplier;

    m_thresholdSize = (thresholdSize < 0) ? (int)(sound.length / 100) : thresholdSize;

    m_soundParser = new SoundParser(sound, sampleSize, soundFeed, beatSubbands,
beatSensitivity);
    m_threshold = new double[m_soundParser.TotalSamples];
    m_peaks = new double[m_soundParser.TotalSamples];
}

/// <summary>
/// Analyze method calls when MonoBehaviour calls Update method.
/// It call main Parse method of the m_soundParser object. After parse
/// MusicAnalyzer calculate FluxThresholds, smooth it by Kalman's filter
/// and convert in 1..0 representation.
/// </summary>
public bool Analyze()
{
    if (m_soundParser.Parse())
    {
        CalculateFluxThresholds();
        CalculateKalmanFilter();
        DetectPeaks();
        ConvertToPercents();
        CalculateSumOfThresholds();
    }
}

```

```

        return true;
    }

    Debug.Log("Parsed " + 100 * Math.Round(m_soundParser.ParseSampleCount /
(float)m_soundParser.TotalSamples, 2) + "% of sound");

    return false;
}

/// <summary>
/// Calculates the flux thresholds.
/// </summary>
private void CalculateFluxThresholds()
{
    for (int i = 0; i < m_soundParser.TotalSamples; i++)
    {
        int start = Math.Max(0, i - m_thresholdSize / 2);
        int end   = Math.Min(m_soundParser.SpectralFlux.Length - 1, i + m_thresholdSize /
2);

        double mean = 0;
        for (int j = start; j <= end; j++)
            mean += m_soundParser.SpectralFlux[j];

        mean /= (end - start);

        m_threshold[i] = mean * m_thresholdMultiplier;
    }
}

/// <summary>
/// Alternate way to detect peaks.
/// </summary>
private void DetectPeaks()
{

```

```

double[] prunnedSpectralFlux = new double[m_threshold.Length];

for (int i = 0; i < m_threshold.Length; i++)
    prunnedSpectralFlux[i] = Math.Max(0, m_soundParser.SpectralFlux[i] -
m_threshold[i]);

for (int i = 0; i < prunnedSpectralFlux.Length - 1; i++)
{
    if (prunnedSpectralFlux[i] > prunnedSpectralFlux[i + 1])
        m_peaks[i] = prunnedSpectralFlux[i];
    else
        m_peaks[i] = 0;
}
}

/// <summary>
/// Kalman's filter for smoothing xy function. In this case filter will smooth flux
thresholds function.

/// More: http://en.wikipedia.org/wiki/Kalman\_filter
/// </summary>
/// <param name="q">Measurement noise</param>
/// <param name="r">Environment noise</param>
/// <param name="f">Factor of real value to previous real value</param>
/// <param name="h">Factor of measured value to real value</param>
private void CalculateKalmanFilter(double q = .35, double r = 35, double f = 1, double
h = 1)
{
    double state = m_threshold[0];
    double covariance = .1;

    for (int i = 0; i < m_threshold.Length; i++)
    {
        double x0 = f * state;
        double p0 = f * covariance * f + q;
        double k = h * p0 / (h * p0 * h + r);
    }
}

```

```

state = x0 + k * (m_threshold[i] - h * x0);
covariance = (1 - k * h) * p0;
m_threshold[i] = state;
}

}

/// <summary>
/// Convert all flux thresholds values into 0..1 representation.
/// </summary>
private void ConvertToPercents()
{
    double maxFlux = m_threshold.Max();

    for (int i = 0; i < m_threshold.Length; i++)
        m_threshold[i] = m_threshold[i] / maxFlux;
}

private void CalculateSumOfThresholds()
{
    m_sumOfFluxThresholds = 0;

    for (int i = 0; i < m_threshold.Length; i++)
        m_sumOfFluxThresholds += m_threshold[i];
}

public double[] Thresholds { get { return m_threshold; } }
public double[] Peaks { get { return m_peaks; } }
public double[,] Beats { get { return m_soundParser.Beats; } }
public double SpeedFactor { get { return m_threshold.Length /
    m_sumOfFluxThresholds; } }
}

```