МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИХАРКІВСЬКИЙ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет комп'ютерних наук
Кафедра програмної інженерії

ЗВІТ З ПЕРЕДДИПЛОМНОЇ ПРАКТИКИ

Місце проходження практики «ДЖИ ФАЙВ ХОЛДИНГ УКР»
у період з «18» квітня 2016 р. по «14» травня 2016 р.
Тема індивідуального завдання:
Програмна система «Ігровий додаток для симуляції пересування в спеціальних умовах»

Студент гр. ПІ-12-1 _____ Шпетний Д.В.

Керівник з практики _____ доц. Мельнікова Р.В.

Роботу захищено «__»_____2016р.
з оцінкою _____

Керівник випускної кваліфікаційної роботи _____ Турута О.П.
Рекомендована оцінка _____

Харків 2016

РЕФЕРАТ

Пояснювальна записка: 30 сторінок, 2 розділи, 15 рисунків, 6 джерел та один додаток.

Метою роботи є аналіз існуючих алгоритмів обробки аудіосигналів, їх використання у ігрових цілях та створення програмної реалізації процедурної генерації рвіня на основі аналізу аудіо.

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.

ВИПУСКНА РОБОТА БАКАЛАВРА, ПРОГРАМНА СИСТЕМА, АНАЛІЗ АУДІО, UNITY, ВИДІЛЕННЯ ТАКТІВ, ВИДІЛЕННЯ ПІКІВ, ПРОЦЕДУРНА ГЕНЕРАЦІЯ.

Explanatory note: 30 pages, 2 sections, 15 pictures, 6 sources and 1 application.

The aim is to analyze the current state of audio processing algorithms, their usage in game development, software implementation of procedural level generation based on audio 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.

FINAL WORK OF BACHELOR, SOFTWARE SYSTEM, AUDIO PROCESSING, UNITY, BEAT DETECTION, ONSET DETECTION, PROCEDURAL GENERATION.

СОДЕРЖАНИЕ

ВСТУПЛЕНИЕ

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 выпускной 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 дальнейшей 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.

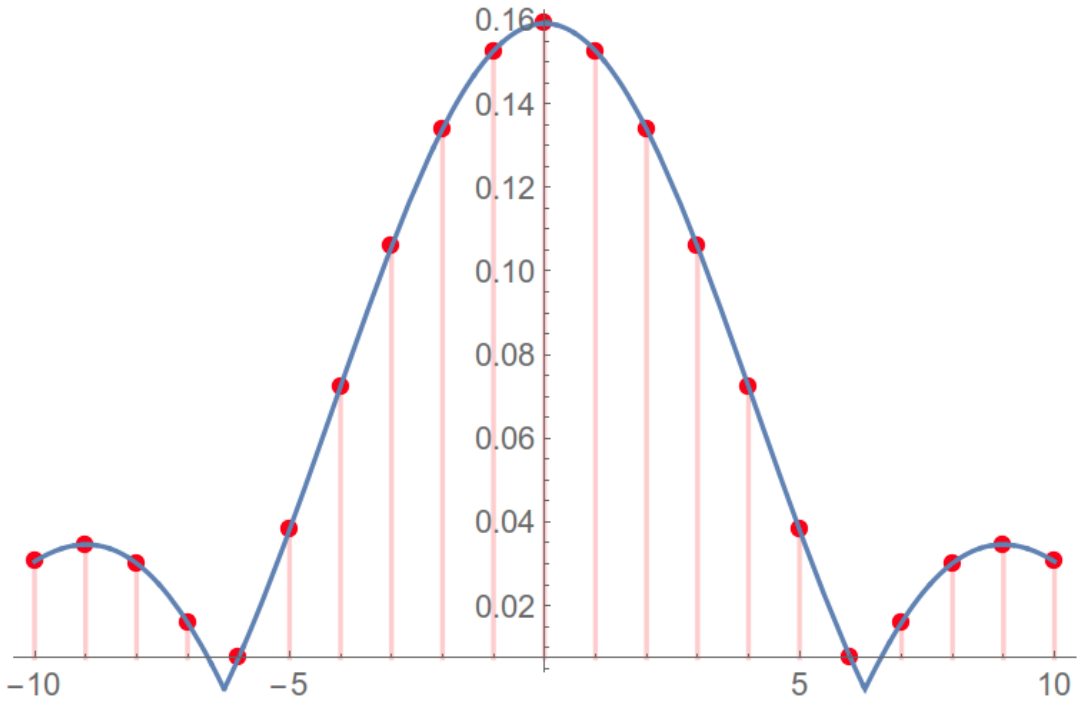

Рис. 1.1 - Соответствие ряда Фурье и преобразования Фурье на примере амплитудного спектра.

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.

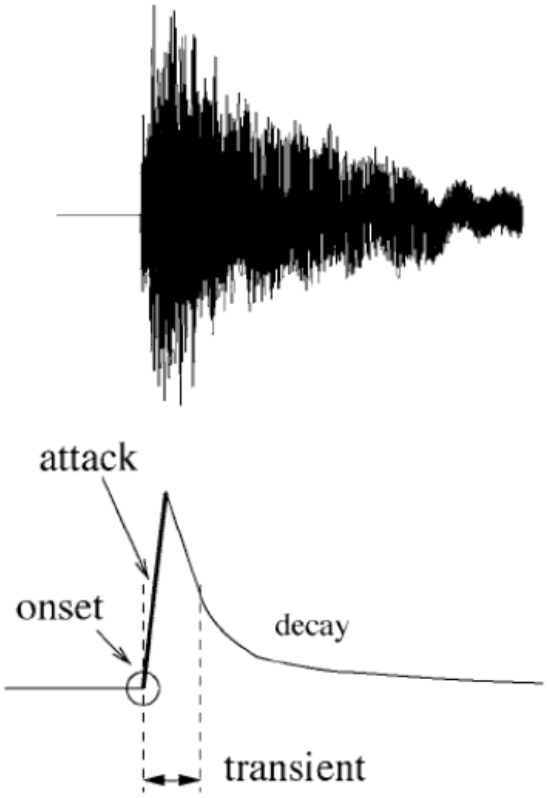

Рис. 1.2 - Графическое представление пика и начала атаки.

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 опциональным поскольку сильно зависит от алгоритма поведения дальнейших этапов.

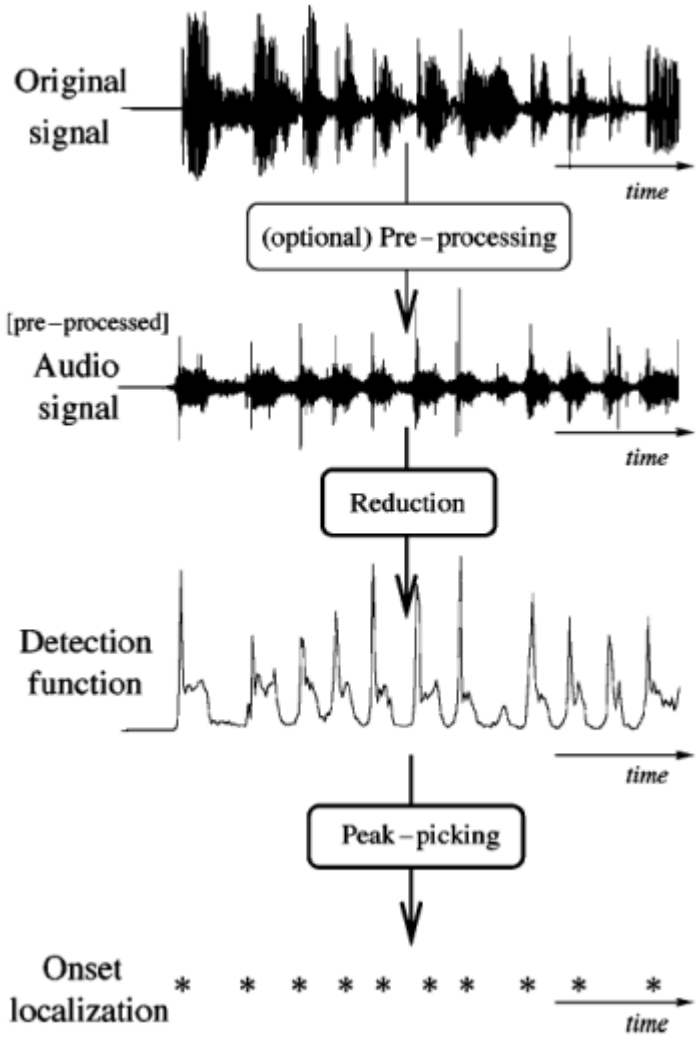

Рис. 1.3 - Этапы анализа пиков аудио сигнала

Второй этап анализа пиков аудио сигнала — его редукция. Она служит для получения сжатой модели структуры аудио 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 [6]. Схематически это изображено на рисунке 1.4.

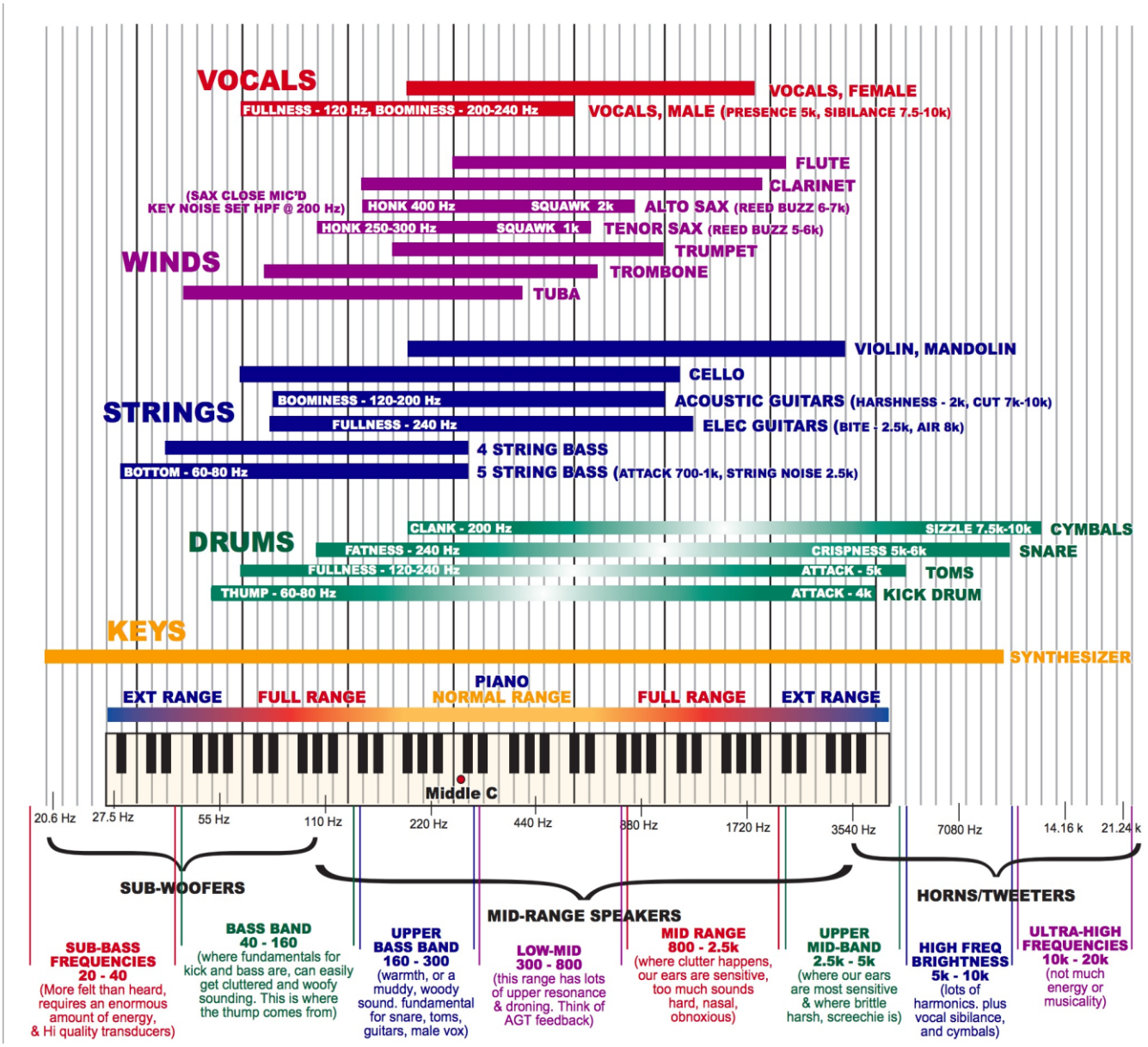

Рис. 1.4 - Диаграмма частот инструментов

Редукцию можно проводить используя следующие подходы [1]:

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

Рассмотрим аудио анализ на основании характеристик входящего 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, что позволяет отследить изменение энергии на протяжении длительных временных промежутков [1].

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

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

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

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.

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 «Musikalisches 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 Филлипа 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 1.5.

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

Рис. 1.5. Таблица генерации строки стихотворения

Каждый бросок кости определяет строку, из который 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.

Рис. 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.

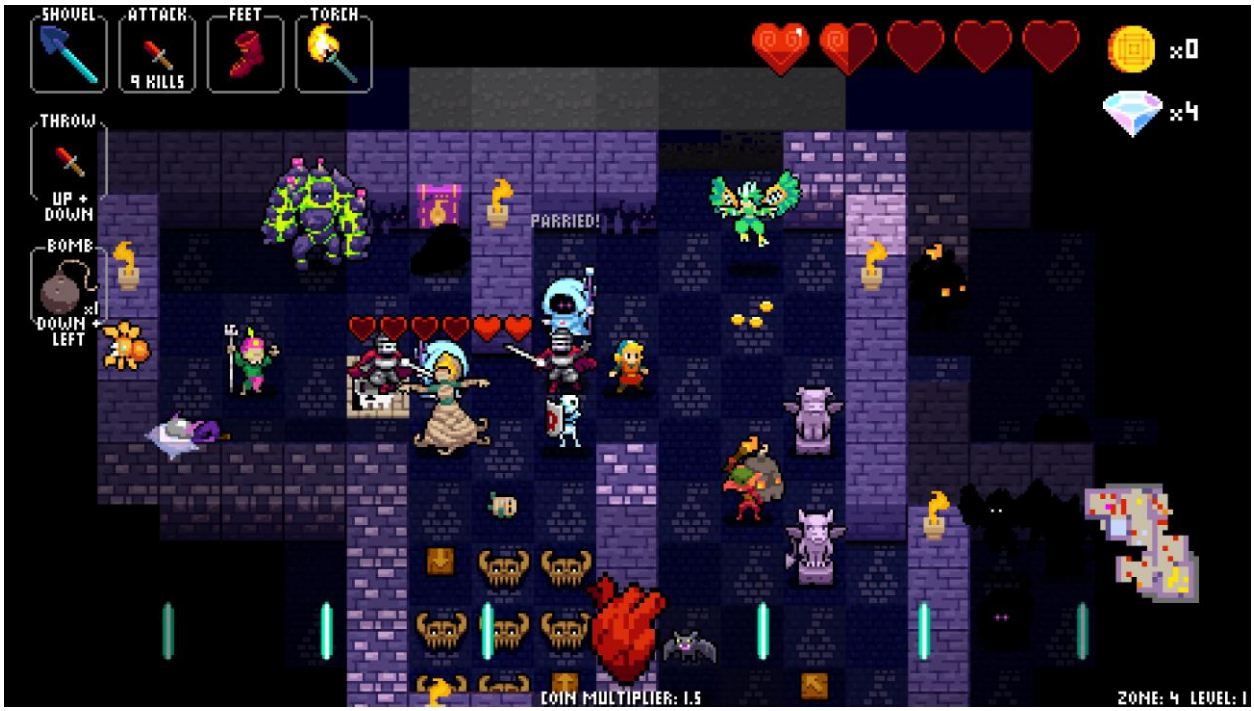

Рис. 1.8 - Игровой процесс Crypt of the NecroDancer

Игра osu! - 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.

– спиннеры — 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.

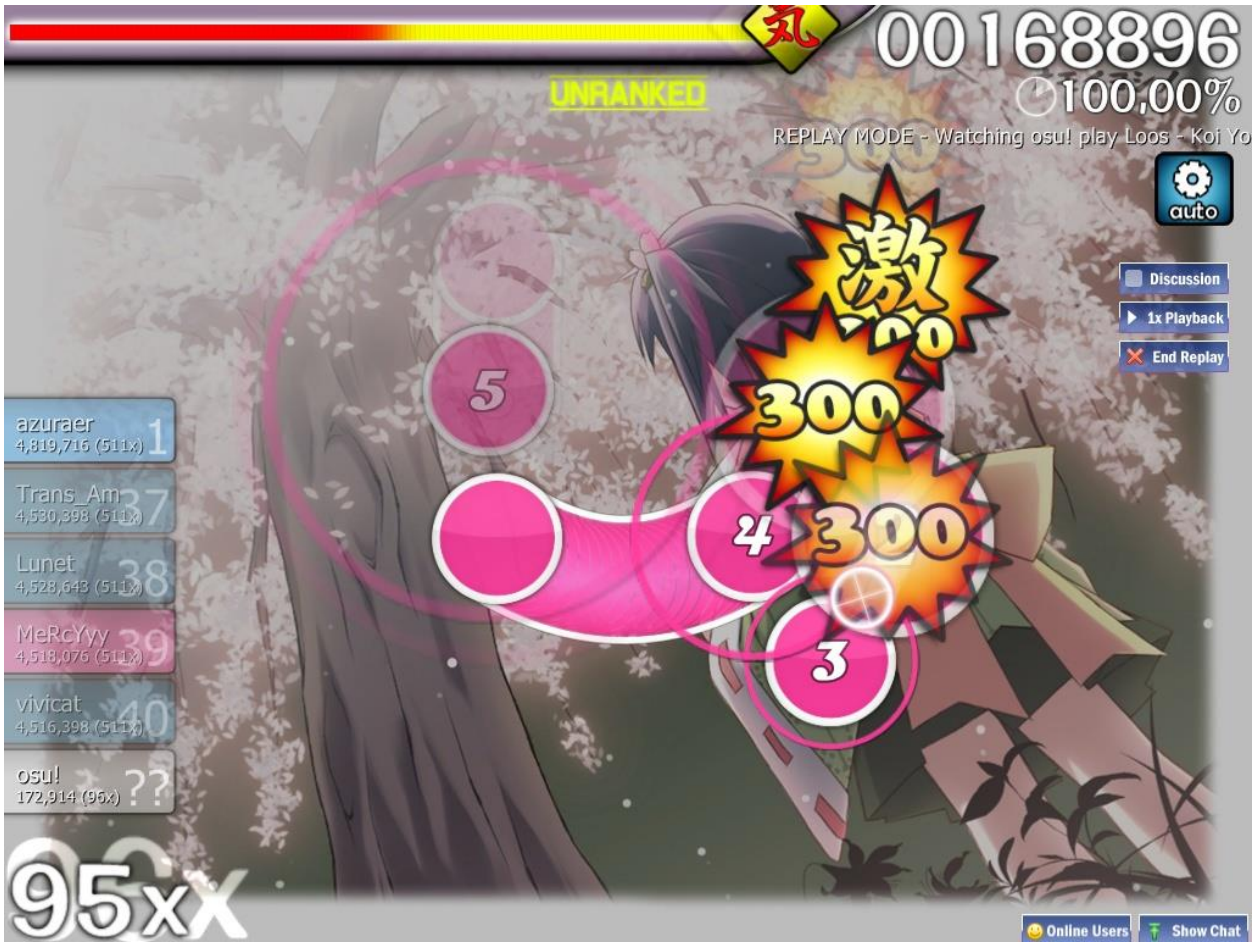

Рис. 1.9 - Игровой процесс osu!

Существенным недостатком игры является необходимость пользователям составлять карты уровней, которые создаются на выбранную песню при помощи встроенного игрового редактора. С одной стороны, это позволяет создавать различные наборы уровней под один и тот же фрагмент, но с другой стороны возлагает ответственность на сообщество на заполнение игры контентом.

1.4 Постановка задачи

Целью выпускной 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 выпускной 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 C-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 кросс-платформенности 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.

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.

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 высокоэффективные 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.

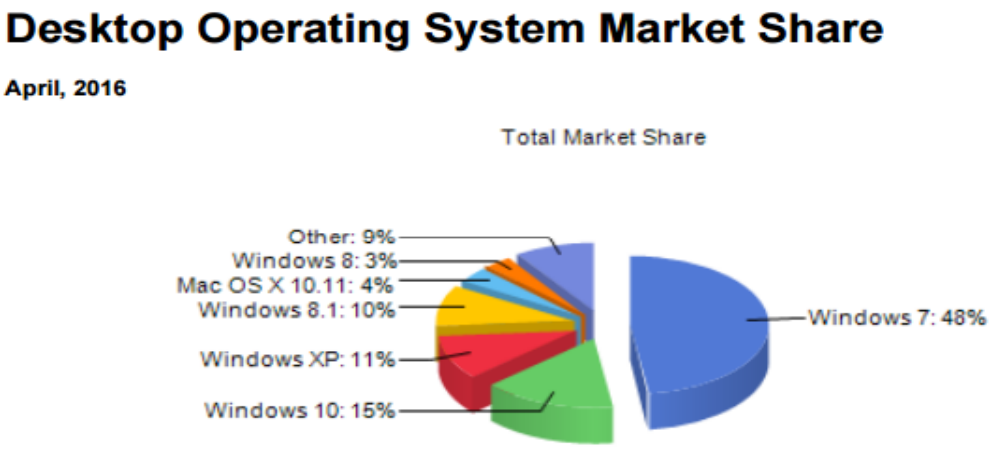

Рис. 2.1 — Операционные системы персональных компьютеров

Однако, использование кросс-платформенного 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.

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. 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. В общем случае, в каждом современном движке можно выделить такие компоненты как на рисунке 2.2.

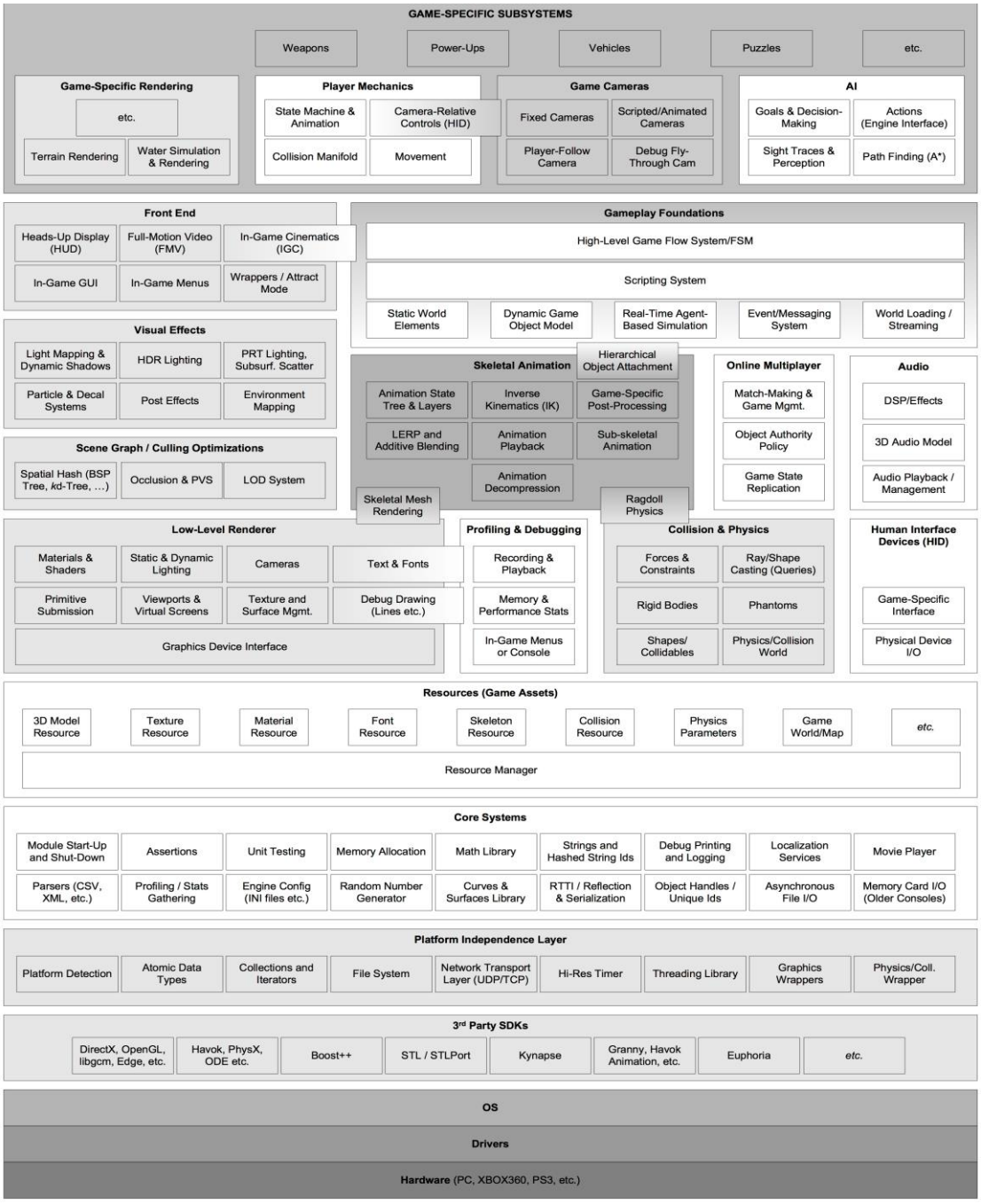

Рисунок 2.2 — Архитектура игрового движка

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 устраненить некоторые зависимости игрового времени от пользовательского ввода и скорости процессора. Шаблон схематически показан на рисунке 2.3.

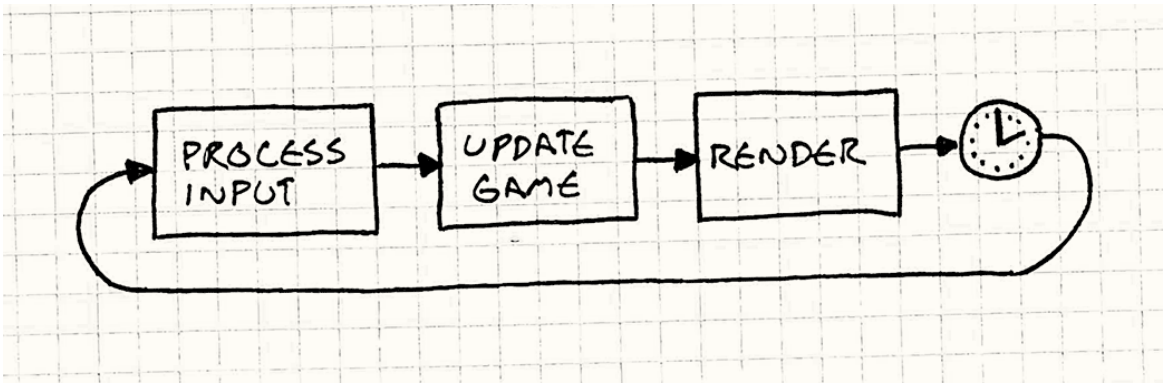

Рисунок 2.3 — Игровой цикл

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 зависимые 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.

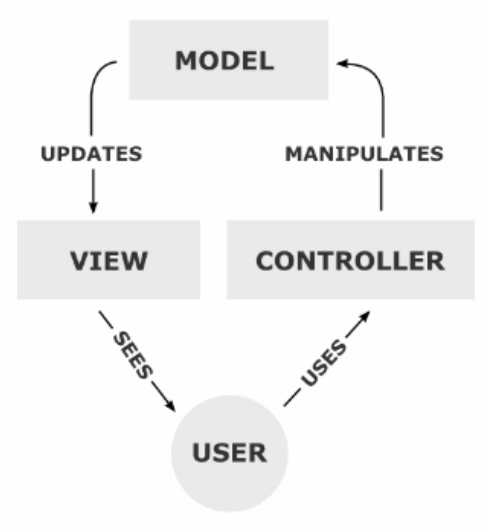

Рис. 2.4 — Шаблон MVC

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.

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.

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

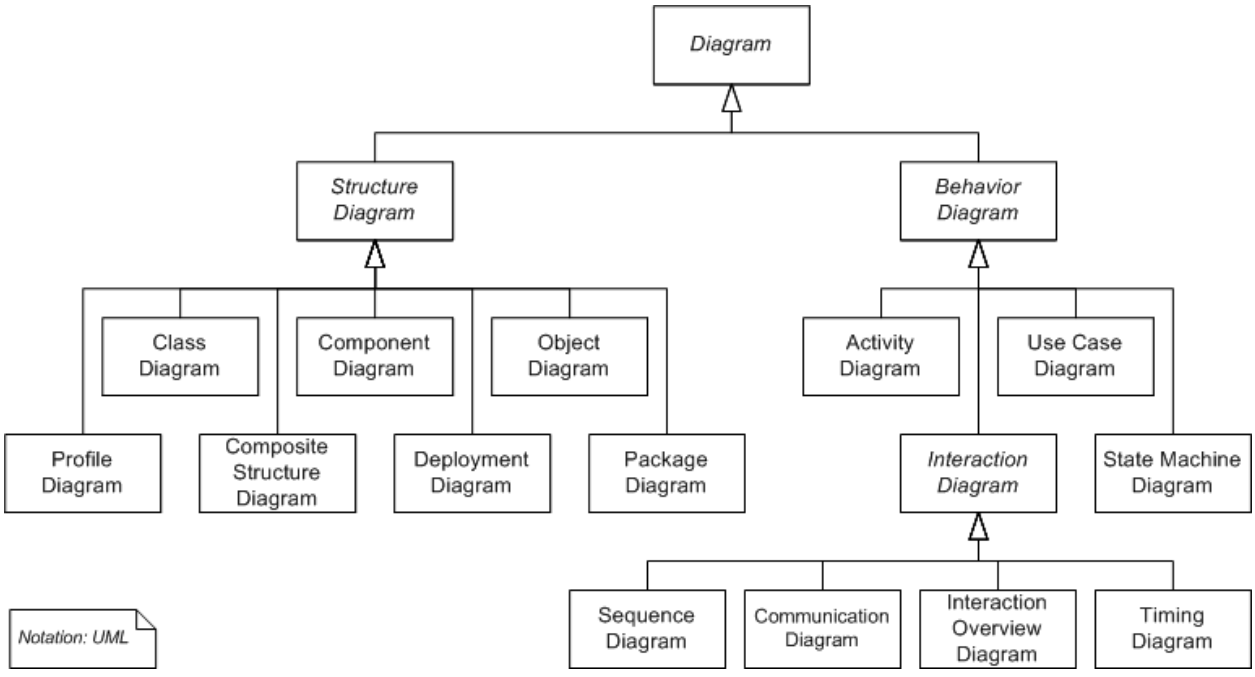

Рис. 2.5 — Структура пакета UML

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 соответствует отдельному 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 система также важно и для внутреннего анализа и проектирования.

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 актёрами 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 путем указания связей обобщения, включения или расширения.

Диаграмма вариантов использования для данной выпускной работы представлена на рисунке 2.8.

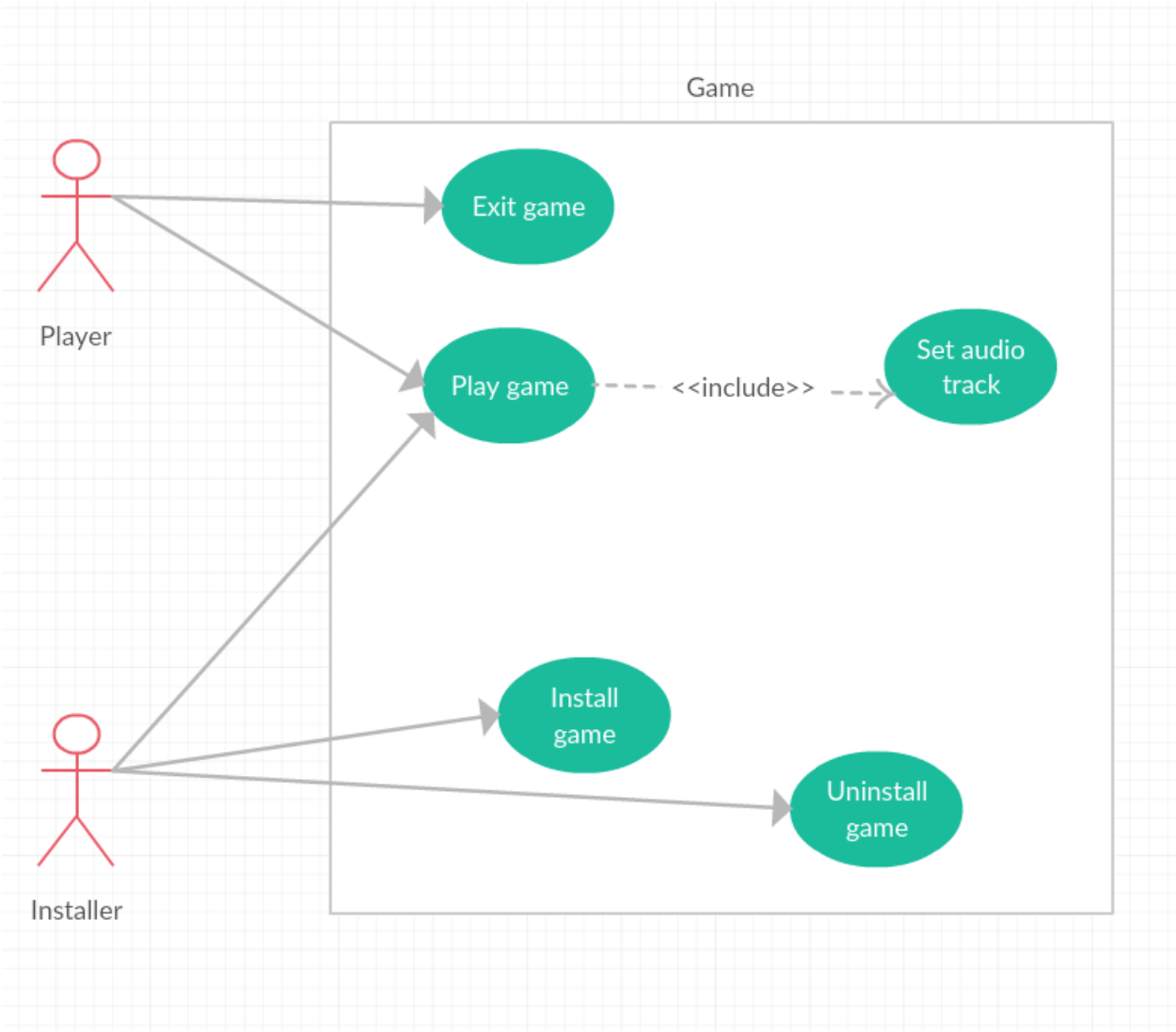

Рис. 2.8 — Use case диаграмма приложения

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 2.9.

Рис. 2.9 — 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.

ВЫВОДЫ

Я, Шпетный Дмитрий Владимирович, проходил преддипломную практику на базе компании «ДЖИ ФАЙВ ХОЛДИНГ УКР» в период с 18 апреля 2016 года по 14 мая 2016 года. Во время прохождения практики я получил опыт работы на реальном производстве, получил новые знания и закрепил их на выполнении индивидуального задания, научился применять теоретические знания при проектировании, разработке и тестировании программного обеспечения на практике. Во время выполнения индивидуального задания, выданного на период преддипломной практики, я закрепил и увеличил количество изученного материала по технологиям C#, Unity, Python, операционным системам Windows, Linux, OS X, мобильным платформам Android и iOS, а также углубил общие алгоритмические навыки.

В результате выполнения индивидуального задания была проанализирована предметная область и спроектирована общая архитектура и концепция игрового приложения. Также были рассмотрены существующие алгоритмы анализа и теоретическая их подоплека.

Результаты выполненной работы полностью удовлетворяют поставленным требованиям. В ходе разработке была имплементирована программная система анализа аудио сигнала. Система была интегрирована в игровой движок Unity и проверена на совместимость на различных платформах и операционных системах.

В ходе разработки индивидуального задания были использованы архитектурные шаблоны, практики написания «чистого кода» и лучшие практики разработки на всех этапах построения приложения.

Игровая отрасль является наиболее быстроразвивающимся сегментом программного обеспечения, а жанр музыкальных игр одной из перспективных веток для разработки в связи с недостатком представителей на рынке и уверенными продажами аналогов. Геймплей, построенный на аудиовизуальном взаимодействии является интуитивно понятным и все еще необычным для целевых игроков.

ПЕРЕЧЕНЬ ИСТОЧНИКОВ

1. Juan Pablo Bello. A Tutorial on Onset Detection in Music Signals / Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, Mark B. Sandler // University of London, 2005. – ISBN 0-78039331-720-4

2. B. C. J. Moore. An Introduction to the Psychology of Hearing, 5th ed. / B. C. J. Moore // New York: Academic, 1997 – 268с. – ISBN 978-0125056281.

3. X. Rodet. Detection and modeling of fast attack transients / X. Rodet and F. Jaillet // Havana, 2001, 30–33с. – 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_W%C3%BCrfelspiel – 08.03.2016 р. – Загол. з экрана.

6. N. Ono. Separation of a monaural audio signal into harmonic/percussive components by complementary diffusion on spectrogram / N. Ono, K. Miyamoto, H. Kameoka, S. Sagayama // European signal processing conference (EUSIPCO), 2008.

ПРИЛОЖЕНИЕ А

Описание предприятия

Компания «ДЖИ ФАЙВ ХОЛДИНГ УКР» основа в 2001 году как независимый разработчик игр для мобильных платформ и домашних игровых консолей в России. Сейчас Джи Файв является международной компанией, разрабатывающей и издающей игры. Целевыми платформами компании являются iPhone, iPad, Android, Mac и Windows. Распространение контента идет через платформенные магазины приложений, такие как Windows Store или Play Market.

Компания разрабатывает и издает семейно-ориентированные игры, где игровой процесс является простым для обучения, что позволяет охватывать широчайший круг аудитории, куда входят как опытные игроки, так и новички.

Компания официально зарегистрирована на Стокгольмской бирже NASDAQ OMX под торговой маркой G5EN. Интернациональная команда G5 из более чем 230 сотрудников включает опытных менеджеров, специалистов в области маркетинга, игровых продюсеров и дизайнеров, client и server-side программистов, программистов баз данных, художников, моделеров, дизайнеров уровней и инженеров качества. Офисы компании открыты в Украине, России, Мальте, Швеции и США.

В портфолио компании входят такие игры-бестселлеры как Virtual City, Supermarket Mania, Special Enquiry Detail, Stand O’Food, and Mahjongg Artifacts. Также Джи Файв разрабатывает игры основанные на сторонних лицензиях и издает игры сторонних разработчиков. Среди сотрудничающих компаний можно отметить такие как Capcom, Electronic Arts, Square Enix, Disney и THQ.