Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

К ЗАЩИТЕ ДОПУСТИТЬ
Зав. каф. ЭВМ
_____ Б.В. Никульшин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту
на тему
РИТМ ИГРА НА UNREAL ENGINE 4, ПОСТРОЕННАЯ НА АЛГОРИТМАХ ОБРАБОТКИ ЦИФРОВОЙ СПЕКТРОГРАММЫ

БГУИР ДП 1–40 02 01 01 304 ПЗ

Студент А.Д. Бирюков

Руководитель Е.В. Богдан

Консультанты:

от кафедры ЭВМ Е.В. Богдан

по экономической части В.Г. Горовой

Нормоконтролер Н.О. Туровец

Рецензент

МИНСК 2024

ВВЕДЕНИЕ

Музыкальное сопровождение в видеоиграх является крайне важным аспектом в разработке игр. Правильно выбранная или написанная музыка позволяет авторам передать больше эмоций, лучше раскрыть персонажей, сделать акцент на каком-то важном для игры аспекте, или же просто задавать настроение по мере прохождения этой игры. Но кроме эстетического и художественного аспекта, музыка может выступать основной составляющей, вокруг которой строятся все взаимодействия, атаки, реакции игрового мира на игрока, либо на внешние взаимодействия с ним.

Игры, в которых музыка лежит в основе геймплея, то есть отвечает или влияет на важнейшие для игры аспекты взаимодействия, управления или подходы к взаимодействию с игрой относят к жанру ритм игр. На рынке видеоигр, игры данного жанра могут выступать в форме тренировочной площадки для обучения игры на каком-либо инструменте, способ потренировать реакцию, заняться разминкой или же продуктом для простого получения удовольствия от игры.

Разработку всех ритм игр объединяет необходимость разработчиков в выборе готовых или написании собственных алгоритмов для анализа используемых аудиофайлов. Существуют разные подходы к решению проблемы интерпретации ударов в минуту, в дальнейшем bpm (от английского beats per minute), задаваемых ритмом аудиофайла, в интерактивные события, происходящие или создаваемые игроком. Данные подходы могут сильно отличатся от используемых технологий, а так же от количества используемых аудиофайлов в проекте и политики компании.

Целью данного дипломного проекта является разработка игрового прототипа, с использованием алгоритмов обработки цифровой спектрограммы аудиофайлов, позволяющих анализировать файл и вычислять bpm для создания событий и дальнейшего построения взаимодействий в игре. Такие алгоритмы позволяют сэкономить большое количество человеко-часов и ресурсов компании на поддержании игры в последующем в качестве сервиса.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор существующих аналогов игр

В данном разделе рассмотрим ярких представителей жанра ритм игр и игр, использующих алгоритмы для анализа аудио сигналов.

1.1.1 Guitar Hero

Одним из известнейших примеров жанра ритм игр является серия игр Guitar Hero (рисунок 1.1).

Рисунок 1.1 – Ритм игра «Guitar Hero» [1]

Первая игра серии вышла в 2005 году и известна тем, что управление в игре осуществлялось посредством специального контроллера в виде уменьшенной копии гитары. В качестве примера, первая часть серии включала в себя 47 композиций. Данная серия игр стала основой для множества подобного рода игр, в последующем занявших крупную долю рынка мобильных игр.

Основной целью игры являлось своевременное нажатие необходимых кнопок на специализированном контроллере (рисунок 1.2). Все кнопки контроллера были окрашены 5 различных цветов, означающих определённый лад. Кроме того на корпусе имелась педаль, отвечающая за активацию струн. Успешным считалось нажатие на данную педаль, с одновременно нажатыми кнопками ладов, отображаемых на экране игрока. Нажатие было необходимо производить именно в тот момент, когда окружность некоторого цвета, доходила до находящихся зон внизу экрана. По итогу успешных нажатий, игроку начислялись очки, влияющие на рейтинг данной попытки прохождения.

Основной целью игры являлось своевременное нажатие необходимых кнопок на специализированном контроллере (рисунок 1.2). Все кнопки контроллера были окрашены 5 различных цветов, означающих определённый лад. Кроме того на корпусе имелась педаль, отвечающая за активацию струн. Успешным считалось нажатие на данную педаль, с одновременно нажатыми кнопками ладов, отображаемых на экране игрока. Нажатие было необходимо производить именно в тот момент, когда окружность некоторого цвета, доходила до находящихся зон внизу экрана. По итогу успешных нажатий, игроку начислялись очки, влияющие на рейтинг данной попытки прохождения.

Рисунок 1.2 – Контроллер для Guitar Hero

При разработке данного продукта планировалось использовать лишь определенное количество лицензированных треков. В связи с этим, при создании набора проигрываемых нот, разработчики не использовали алгоритмы для анализа аудиофайлов. Вместо этого акцент был сделан на ручное воссоздание гитарных аккордов под специализированный контроллер, для создания ощущения игры на настоящей гитаре. Такое решение могло быть так же обосновано финансовыми возможностями компании и ограничениями технологий того времени.

Данный подход является более выгодным и точным по сравнению с использованием алгоритмов, при условии, что количество аудиофайлов не будет превышать некоторый порог. В своё время порог определяется штатом и финансовыми возможностями компании.

Минусом данного подхода можно выделить сложность к масштабируемости продукта в будущем. Алгоритмы для частотного анализа аудиофайлов позволили бы в дальнейшем сэкономить на разработке данного проекта, а так же удешевить выпуск дополнительного контента.

1.1.2 Osu!

«Osu!» – это бесплатная ритм игра, в которой имеется четыре официальных игровых режима, отличающихся игровыми механиками (рисунок 1.3). Управление в игре осуществляется посредством различных

периферийных устройств: компьютерной мыши, клавиатуры, графического планшета, сенсорного устройства.

Рисунок 1.3 – Ритм игра «Osu!»

Отличительной особенностью данного проекта является то, что «Osu!» является платформой, позволяющей игрокам добавлять свои композиции, предоставляя игрокам инструменты для добавления собственного фона, композиций, настройки времени появления, времени задержек и позиций интерактивных объектов.

При установке данного продукта, пользователь обнаружит лишь пару тестовых композиций. Это связано с тем, что библиотека доступных треков хранится на официальном сайте продукта и обладает отрытым доступом для добавления и скачивания, готовых к игре композиций (рисунок 1.4). Данные композиции хранятся в специальных файлах формата, хранящих в себе саму композицию, настройки, а так же порядок и положение интерактивных элементов.

Цель игры заключается в своевременном нажатии на появляющиеся в такт музыки ноты, удержание курсора в пределах подвижных слайдеров и раскручивание спиннеров на максимально возможное значение. За все вышеперечисленные действия игроку начисляются очки рейтинга. Чем точнее игрок попадает в такт или удерживает курсор в центре указанных зон, тем больше очков рейтинга ему начисляется. Полученные очки влияют на позицию игрока в мировом рейтинге игроков.

В данной игре частично используются алгоритмы для частотного анализа композиций, но остаётся необходимость вручную определять положение нот на экране, а так же других интерактивных элементов. Данных аспект является основным минусом такого подхода, потому как конечному пользователю хочется как можно быстрее получить желаемый результат в

виде готовой интерактивной композиции, а не задумываться о расположении элементов по мере игры.

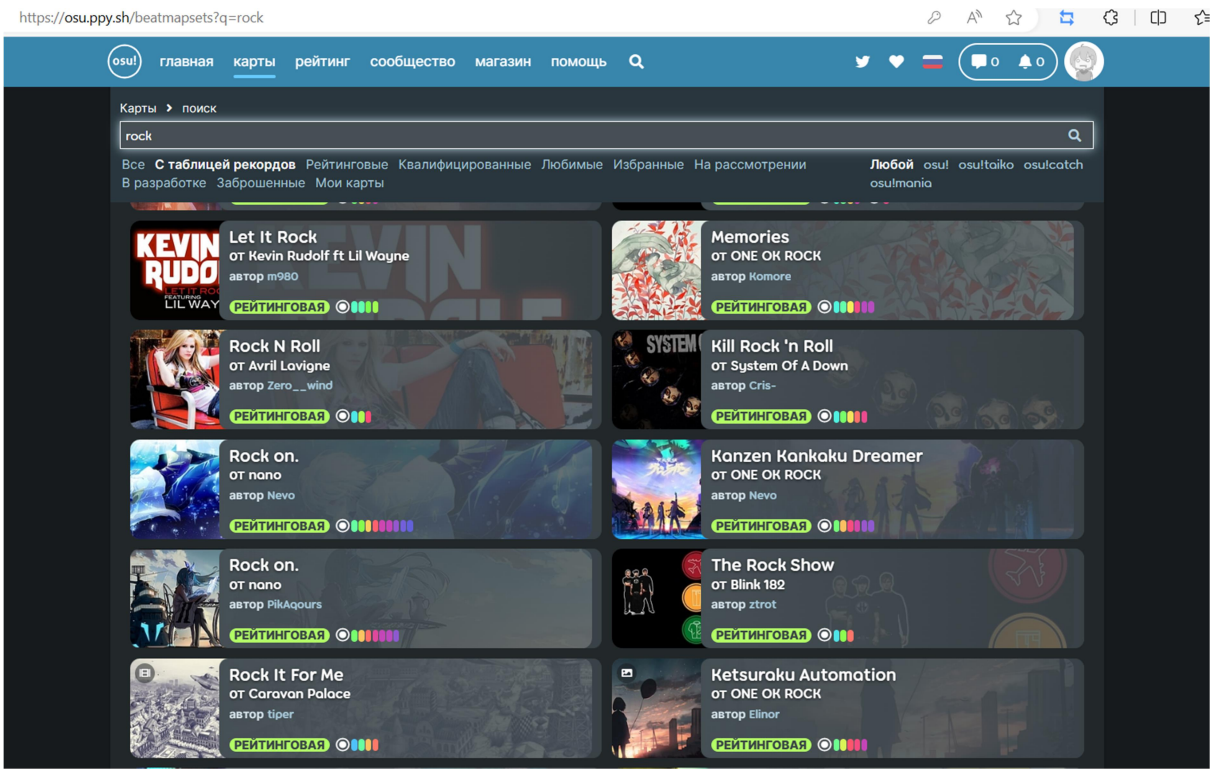

Рисунок 1.4 – Библиотека игры «Osu!» [2]

В связи со своей доступностью и наличием большой библиотеки композиций, данный проект стал крайне популярен как среди фанатов жанра игры и определённых прослоек пользователей, увлекающихся восточной музыкой, так и среди пользователей, использующих данную платформу для тренировки реакции и точности нажатия клавиш.

1.1.3 One Hand Clapping

One Hand Clapping – это вокальный 2D платформер от небольшой инди студии (рисунок 1.5). Несмотря на небольшую известность проекта, хочется отметить нестандартный подход разработчиков к реализации взаимодействия игрока с внутри игровым миром.

В отличие от вышеперечисленных примеров, в данной игре отсутствует фактор борьбы с другими игроками за позицию в локальном или мировом рейтинге. Так же в данной игре не делается фокус на точности нажатия кнопок в соответствии с музыкальным сопровождением. One Hand Clapping предлагает игрокам использовать свой голос как инструмент для преодоления препятствий и решения головоломок. Игра погружает в фантастический мир, где правильные звуковые частоты помогают освещать путь, создавать платформы и управлять элементами окружения. Это не

только новый способ взаимодействия с игровым миром, но и возможность для игроков тренировать свои вокальные навыки и чувство ритма.

Рисунок 1.5 – Игра One Hand Clapping

Управление в игре осуществляется посредством клавиатуры и микрофона. Все головоломки в игре построены так, что для их решения необходимо пропевать ноты определенной тональности. За счёт встроенных алгоритмов анализа аудио сигнала, поступающий в микрофон сигнал, преобразуется в частотно временной график, кривая которого, как пример, может послужить мостом для прохождения препятствий (рисунок 1.6). Все взаимодействия с персонажами и объектами в игре, так же зачастую осуществляют посредством голосового ввода.

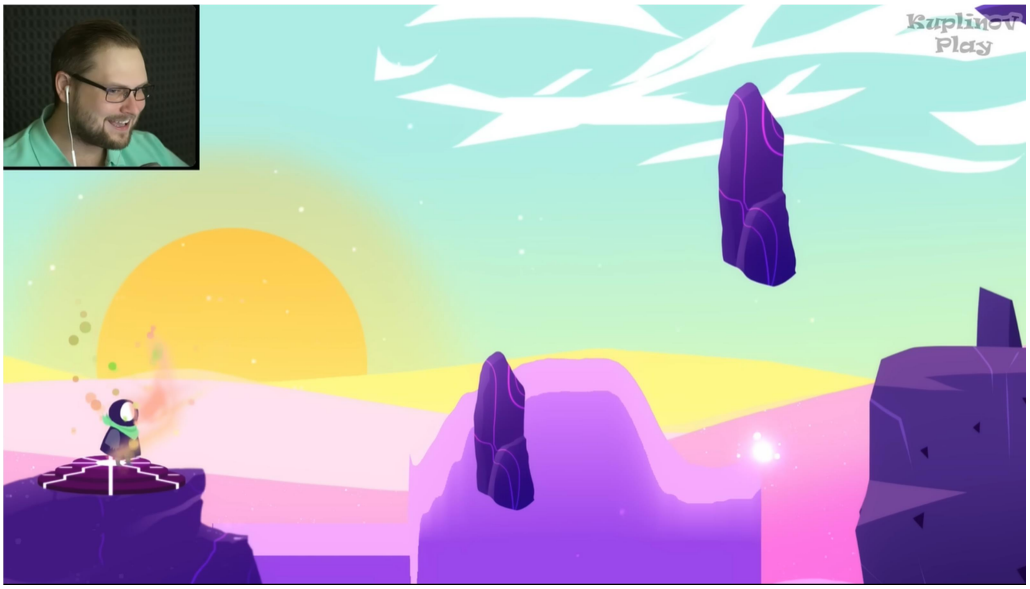

Рисунок 1.6 – Пример использования голоса в One Hand Clapping [3]

1.2 Обзор инструментов и технологий

Данный подраздел включает в себя краткий обзор инструментов и технологий, которые использовались при разработке дипломного проекта.

1.2.1 Unreal Engine 4

Unreal Engine 4 – является программной платформой, разработанной компанией Epic Games. Имеет в наличие широкий спектр инструментов, для создания визуальных эффектов, 3D графики, игр и симуляций. Четвертая версия данной платформы была разработана в 2012 году. Именно с четвертой версии данное программное обеспечение перешло на бесплатную форму распространения. С момента выпуска Unreal Engine 4 стала одной из самых популярных платформ для разработки игр по всему миру (рисунок 1.7).

Основные аспекты, выделяющие Unreal Engine 4 на фоне других платформ для разработки игр:

1. Большое сообщество. Unreal Engine 4 обладает активным сообществом разработчиков. Существует форум разработчиков Epic, где обсуждаются различные темы, связанные с разработкой на Unreal Engine 4. Специалисты компании часто отвечают на вопросы по разработке, а так же осуществляют консультацию и помощь с разработкой для компаний, работающих с продуктом компании. Кроме того, Unreal Engine предлагает магазин, где пользователи могут приобрести и продать различные плагины и контент, созданный другими пользователями.

Рисунок 1.7 – Рабочая среда Unreal Engine 4

2. Языки программирования. Unreal Engine 4 поддерживает C++ и проприетарного Blueprint Visual Scripting для создания игровых элементов. Blueprint Visual Scripting – это визуальная система скриптов, которая

позволяет разработчикам значительно увеличить скорость прототипирования и создания полноценных игровых элементов, без написания кода.

3. Качественная документация. Документация Unreal Engine 4 доступна на многих языках и содержит подробные сведения по установке и работе с программным обеспечением. Она включает в себя введение в программирование на C++, обзоры предустановленных функций и технологий, руководства по работе с определёнными аспектами платформы, будь то графика или физические симуляции, и многое другое. Также имеется поддержка следующих инструментов программирования на C++:

3.1 C++ Class Wizard: Инструмент для быстрого создания новых классов и компонентов;

3.2 Console Variables: Инструмент для управления различными параметрами с использованием консоли;

3.3 Low-Level Memory Tracker: Инструмент для детального отслеживания используемой создаваемым продуктом памяти устройства.

4. Поддержка современных технологий компьютерной графики. Unreal Engine 4 поддерживает множество современных технологий компьютерной графики. Он включает в себя системы освещения и теней, материалы и текстуры, визуальные эффекты и постобработку. Кроме того, Unreal Engine 4 поддерживает технологии RTX, Dolby Atmos, VR, ARCore, Niagara, Houdini и многие другие.

5. Поддержка всех известных платформ. Unreal Engine 4 поддерживает разработку для множества платформ, включая Windows, macOS, Linux, iOS, Android, PlayStation, Xbox, Nintendo Switch и другие.

6. Интеграция со средой разработки. Для разработки на C++ в Unreal Engine 4 используется Microsoft Visual Studio или Apple Xcode. Для корректной работы данных сред разработки, Epic Games были разработаны специализированные плагины, которые находятся в открытом доступе.

7. Модульная архитектура. Unreal Engine 4, построен на модульной архитектуре. Имеется возможность добавления собственных модулей и расширений функциональности платформы.

1.2.2 Microsoft Visual Studio

Microsoft Visual Studio – это интегрированная среда разработки (IDE), разработанная компанией Microsoft (рисунок 1.8). Является одним из важнейших инструментов, используемых при разработке данного дипломного проекта и для разработки программного обеспечения в целом.

Данная интегрированная среда разработки поддерживает широкий спектр языков программирования, таких как C++, Node.js, Python и R, что позволяет разработчикам создавать приложения для различных платформ, включая Windows, Azure и многие другие. Одной из выдающихся особенностей Visual Studio является её способность улучшать продуктивность разработчиков за счёт таких функций, как IntelliSense,

которая предлагает авто дополнения кода, и CodeLens, которая предоставляет полезную информацию прямо в редакторе кода.

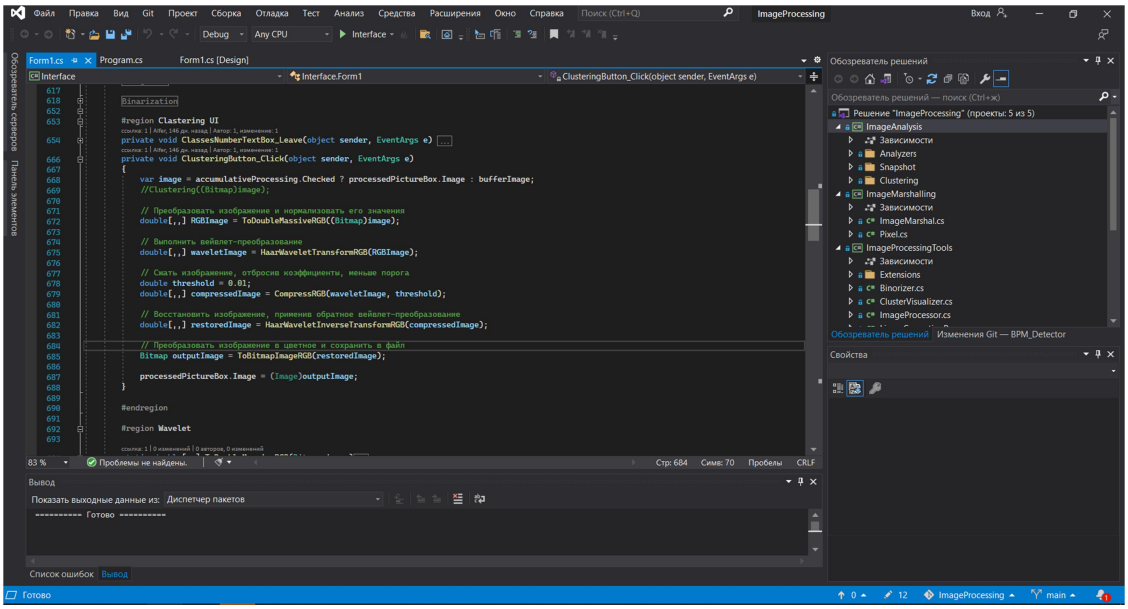

Рисунок 1.8 – Интегрированная среда разработки Microsoft Visual Studio

Visual Studio интегрируется с GitHub и Unreal Engine 4, что упрощает управление версиями и работу над проектом. Кроме этого, данное ПО обладает детальными настройками, как самой среды разработки, так и разрабатываемого проекта. Для более тонкой настройки можно использовать доступные к установке плагины и дополнительные инструменты от других пользователей. Эти инструменты и возможности интеграции делают Visual Studio идеальным выбором для командных и одиночных проектов.

Visual Studio также выделяется на фоне конкурентов благодаря своим возможностям для отладки и диагностики, предоставляя разработчикам инструменты для эффективного поиска и устранения ошибок в коде. Это включает в себя визуализации для асинхронных операций и автоматические анализаторы, которые помогают оптимизировать производительность и качество кода. Такие возможности делают Visual Studio не только инструментом для написания кода, но и платформой для обеспечения высокого качества разработки программного обеспечения.

Использование Visual Studio в дипломном проекте позволяет значительно ускорить процесс разработки и повысить его качество, благодаря широкому спектру инструментов и функций, которые она предлагает.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

После анализа всех требований к разрабатываемому проекту, мы переходим к разбиению системы на функциональные блоки. Этот метод позволяет создавать гибкую архитектуру приложения, что в свою очередь позволяет изменять существующие и добавлять новые функциональные блоки, не затрагивая общую работу системы.

Структура проекта состоит из следующих блоков:

- блок игровой логики;
- блок игрового контроллера;
- блок игрового интерфейса;
- блок управления персонажем;
- блок игрового меню;
- блок чтения аудиофайла;
- алгоритм анализа аудиосэмплов;
- алгоритм вычисления музыкального ритма.

Взаимосвязь основных блоков проекта отображена на структурной схеме ГУИР.400201.304 С1.

2.1 Блок игровой логики

Данный блок содержит в себе базовую логику игровых правил и взаимодействий игрока с игровыми объектами. В блоке игровой логики описываются основные логические элементы игры, такие как условия проигрыша, подсчёт очков игрока. В данный блок так же входит установление взаимосвязей между другими блоками данного проекта. Реализация нескольких вариаций игровой логики, позволяет в дальнейшем быстро переключаться между различными режимами игры. В том числе и на игровое меню.

Для реализации данного блока будет использован язык визуального программирования Blueprints.

2.2 Блок игрового контроллера

Данный блок отвечает за всевозможные взаимодействия игроком, с игровым миром, через игрового персонажа. В данном блоке описываются основные принципы передвижения персонажа, взаимодействия с предметами, реакции персонажа на события и так далее. Кроме того блок игрового контроллера позволяет разработчику абстрагироваться от персонажа или пешки, над которым производится управление данным контроллером. Функционал Unreal Engine 4 позволяет при определённых условиях производить переключение между управляемыми пешками и персонажами. Такой подход позволяет сохранить гибкость разрабатываемых блоков кода.

Для реализации данного блока, также будет использован язык визуального программирования Blueprints.

2.3 Блок игрового интерфейса

Блок игрового интерфейса отвечает за интерфейс главного меню игры, интерфейса для взаимодействия с аудиофайлами, а так же за интерфейс самого игрового процесса, счётчика очков и других элементов индикации и вывода информации пользователю.

Для реализации данного блока будет использован набор внутренних инструментов Unreal Engine 4, для создания графических пользовательских интерфейсов, а так же язык визуального программирования Blueprints.

2.4 Блок управления персонажем

Для осуществления управления игровой пешкой необходимо при считывании нажимаемых клавиш, создавать события, которые в дальнейшем будут передаваться в блок игрового контроллера. Нажатия должны считываться как с клавиатуры, так геймпада. В данном блоке должна быть реализована реакция на нажатие определённых, заранее установленных разработчиком клавиш, отвечающих за исполнение игровой пешкой конкретных действий.

Для реализации данного блока будет использован язык визуального программирования Blueprints, а также настройки проекта Unreal Engine 4.

2.5 Блок игрового меню

Данный блок представляет собой реализацию игрового меню и окон для взаимодействия с игроком, разработанных с помощью внутренних инструментов Unreal Engine 4, для создания графических пользовательских интерфейсов. В данный блок входят:

- главное меню игры;
- меню настроек;
- меню выбора аудиофайла.

Логика взаимодействия пользователя с интерфейсом описывается на языке визуального программирования Blueprints.

2.6 Блок чтения аудиофайла

Блок алгоритма осуществляющего чтение заголовка и блока данных аудиофайла. Первоначальной задачей алгоритма является определение формата файла, валидация событий, приводящих к возникновению ошибок при чтении и открытие файла в режиме чтения. Аудиофайл должен соответствовать формату .wav, так как данный формат файла используется для хранения несжатого аудиосигнала в импульсно-кодовой модуляции.

После открытия файла, главными задачами блока будут являться, чтение заголовка, инициализация структуры для сохранения данных из заголовка аудиофайла, таких как количество каналов, частоту дискретизации, аудиоформат, наличие и тип кодировки, количество байт для хранения одного сэмпла, а так же общий размер файла без учета первых 16 байт. Данная информация необходима для корректного чтения блока данных конкретного аудиофайла. Так же полученная информация будет использована для анализа композиции в блоке анализа аудиофайла.

Данный алгоритм будет реализован с использованием языка программирования C++.

2.7 Алгоритм анализа аудиосэмплов

Блок алгоритма анализа аудиосэмплов отвечает за обработку и анализ блока данных. Используя информацию, полученную в заголовке файла в алгоритме чтения файла, производится быстрое преобразование Фурье с использованием окна Гаусса. Использование окна Гаусса позволит избавиться от возможного появления шумов, после применения БПФ функции. Входными значениями алгоритма являются массив с амплитудно-временными значениями, а так же размер данного массива. Выходным значением является массив с амплитудно-частотными значениями, отображающими перепады амплитуд конкретных диапазонов частот в определённый промежуток времени. На основе данных значений производится анализ и вычисление ритма музыкальной композиции, который в дальнейшем отправляется в блок игровой логики.

Данный алгоритм будет реализован с использованием языка программирования C++.

2.8 Алгоритм вычисления музыкального ритма

В данном алгоритме, исходя из полученных данных алгоритмов чтения аудиофайла, а также из алгоритма анализа аудиосэмплов, происходит вычисление музыкального ритма композиции. В учёт берутся количество каналов, а так же пиковые амплитудно-частотные значения. Результатом данного алгоритма являются данные, на основе которых создаются игровые события взаимодействующие с игроком через игрового персонажа.

Данный алгоритм будет реализован с использованием языка программирования C++.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Функциональная структура проекта во многом аналогична структурной. Взаимосвязь между основными компонентами представлена на диаграмме классов ГУИР.400201.304 РР.1.

3.1 Блок игровой логики

В данном блоке находятся описания базовых логических законов и правил игры. Определение этих базовых элементов осуществляется в классе `GM_GameMode`, который является наследуемым классом от базового класса `GameMode`. Ниже будут рассмотрены отдельные элементы и переменные, определённые в данных классах и используемые в данном дипломном проекте.

Базовый класс `GameMode` содержит в себе следующие поля:

– `GameSessionClass` хранит в себе объект класса `GameSession`, предназначенный для реализации сетевых игровых сессий. По умолчанию инициализируется объектом базового класса `GameSession`, что соответствует одиночному режиму игры.

– `GameStateClass` хранит в себе объект класса `GameState`, предназначенный для хранения специфичных аспектов игрового мира, а так же некоторых данных каждого отдельного пользователя в сетевых игровых сессиях. По умолчанию инициализируется объектом базового класса `GameState`, что соответствует одиночному режиму игры.

– `PlayerControllerClass` хранит в себе объект класса `PlayerController`, предназначенный для реализации всевозможных взаимодействий и передвижений подконтрольного персонажа или пешки. По умолчанию инициализируется объектом базового класса `PlayerController`, что приравнивается к отсутствию возможности взаимодействия управляемым персонажем или пешкой с игровым миром.

– `PlayerStateClass` хранит в себе объект класса `PlayerState`, предназначенный для хранения некоторых аспектов состояний и параметров подконтрольного персонажа или пешки. По умолчанию инициализируется объектом базового класса `PlayerState`, что соответствует отсутствию начальных состояний персонажа.

– `HUDClass` хранит в себе объект класса `HUD`, предназначенный для реализации пользовательского интерфейса игрока. По умолчанию инициализируется объектом базового класса `HUD`, что соответствует отсутствию как такового пользовательского интерфейса.

– `DefaulPawnClass` хранит в себе объект класса `Pawn`, предназначенный для хранения ссылки на базовый управляемый игроком

объект. По умолчанию инициализируется объектом базового класса Pawn, что соответствует отсутствию подконтрольного объекта для PlayerController.

– SpectatorClass хранит в себе объект класса Pawn, предназначенный для реализации возможности наблюдения за игровым процессом со стороны. Чаще всего используется при разработке сетевых игр. По умолчанию инициализируется объектом базового класса Pawn, что равняется отсутствию наблюдателя.

– ReplaySpectatorPCClass хранит в себе объект класса PlayerController, предназначенный для реализации возможности управления для наблюдателя. По умолчанию инициализируется объектом базового класса PlayerController, что приравнивается к отсутствию возможности взаимодействия в роли наблюдателя.

– ServerStatReplicatorClass хранит в себе объект класса ServerStatReplicator, предназначенный для реализации хранения сетевых данных для репликации другим игрокам в сетевой игре. По умолчанию инициализируется объектом базового класса ServerStatReplicator, что соответствует одиночному режиму игры.

Метод ConstructionScript() является базовым методом всех создаваемых классов в Unreal Engine 4, и представляет собой инициализацию параметров при создании объекта класса.

Производный класс GM_GameMode содержит следующие поля:

– PlayerStart содержит в себе ссылку на объект класса PlayerStart. Представляет собой точку в пространстве XYZ для порождения базовой пешки игрока.

Метод GetGameMode() возвращает ссылку на объект данного класса.

Метод ChangeGameMode(GameMode) осуществляет замену нынешнего игрового режима на передаваемый объект базового класса GameMode.

3.2 Блок управления

Блок управления представлен в виде класса BP_PlayerController. Данный класс наследуется от базового класса PlayerController, и отвечает за управление объектом BP_PlayerPawn, посредством использования следующих системных асинхронных вызовов: InputPitchScale, InputYawScale, InputRollScale.

Базовый класс PlayerController содержит следующие поля:

– InputPitchScale отвечает за масштабирование ввода, который влияет на поворот камеры или персонажа по оси X.

– InputYawScale отвечает за масштабирование ввода, который влияет на поворот камеры или персонажа по оси Y.

– InputRollScale отвечает за масштабирование ввода, который влияет на поворот камеры или персонажа по оси Z.

Метод ConstructionScript() инициализирует базовые параметры класса.

Метод GetInputPitchScale() возвращает значение переменной InputPitchScale.

Метод GetInputYawScale() возвращает значение переменной InputYawScale.

Метод GetInputRollScale() возвращает значение переменной InputRollScale.

Метод SetInputPitchScale(Float) присваивает передаваемое значение переменной InputPitchScale.

Метод SetInputYawScale(Float) присваивает передаваемое значение переменной InputYawScale.

Метод SetInputRollScale(Float) присваивает передаваемое значение переменной InputRollScale.

Производный класс BP_PlayerController включает в себя следующие поля:

Метод SetupGameHUD() вызывается в начале игры и осуществляет инициализацию пользовательского интерфейса.

Метод EventEndPlay() вызывается в конце игрового процесса и возвращает игрока в главное меню.

Метод GetPlayerController() возвращает ссылку на объект используемого контроллера класса PlayerController.

Метод ChangeGameMode() осуществляет замену игрового режима.

3.3 Блок игрового интерфейса

Данный блок реализован с использованием базовых инструментов Unreal Engine 4, для создания графических интерфейсов. Логика, описывающая функционал интерфейса описана в классах: UserWidget, WB_MainMenu, WB_ChooseFileMenu.

Базовый класс UserWidget включает в себя следующие поля:

– ShowMouseCursor это булева переменная, отвечающая за отображение курсора мыши при использовании графического интерфейса.

– IsEnabled это булева переменная, отвечающая за активное состояние элемента пользовательского интерфейса.

– ESlateVisibility набор состояний видимости для элементов пользовательского интерфейса. Имеет набор предустановленных константных значений.

Метод SetVisibility(ESlateVisibility) принимает и переводит элемент интерфейса в переданное состояние.

Метод GetVisibility() возвращает активное состояние видимости для текущего элемента пользовательского интерфейса.

Метод SetIsEnabled(Boolean) инициализирует переменную IsEnabled, переданным булевым значением.

Метод GetIsEnabled() возвращает активное булево состояние для текущего элемента пользовательского интерфейса.

Производный класс WB_MainMenu имеет следующие поля:

Поля PlayButton и QuitButton хранят в себе объекты класса Button, а так же размеры и цвета данных кнопок.

Методы OnPlayButtonClick() и OnQuitButtonClick() описывают реакцию интерфейса на нажатия данных кнопок.

Метод ChangeGameMode() осуществляет замену игрового режима.

Производный класс WB_ChooseFileMenu имеет следующие поля:

Поля SelectButton, ApplyButton и BackButton хранят в себе объекты класса Button, а так же размеры и цвета данных кнопок.

Поле Filename предназначено для хранения полученного названия исполняемого аудиофайла.

Методы OnSelectButtonClick(), OnApplyButtonClick() и OnBackButtonClick() описывают реакцию интерфейса на нажатия данных кнопок.

Метод ClearInputRow() осуществляет очистку окна с названием аудиофайла.

Метод ChangeGameMode() осуществляет замену игрового режима.

3.4 Алгоритм чтения аудиофайла

Данный алгоритм, является частью подключаемого плагина для работы и анализа аудио файла. К данному алгоритму относятся следующие классы: WavHeader, WavReader.

В классе WavHeader задана структура WAVHEADER содержащая следующие поля:

– chunkId поле содержит в себе информацию о начале RIFF-цепочки.

– chunkSize хранит в себе размер последующей RIFF-цепочки.

– format содержит в себе символьное краткое название формата файла.

– subchunk1Id содержит символьный идентификатор подцепочки.

– subchunk1Size поле хранит размер подцепочки, начиная с этой позиции.

– audioFormat содержит номер аудиоформата файла.

– numChannels хранит в себе количество каналов для воспроизведения аудиофайла.

– sampleRate частота дискретизации аудиосигнала.

– byteRate количество байт, передаваемых за секунду воспроизведения.

– blockAlign количество байт для одного сэмпла, включая все каналы.

– bitsPerSample количество бит, необходимых для хранения одного сэмпла.

– subchunk2Id содержит символьный идентификатор подцепочки

– subchunk2Size поле хранит размер подцепочки, начиная с этой позиции.

– subchunk2ListType содержит тип хранящейся информации о лицензии аудиофайла.

– subchunk2Text хранит информацию о лицензии аудиофайла.

Метод GetHeader(String) возвращает структуру WAVHEADER, хранящий в себе всю информацию аудиофайла, необходимую для чтения файла.

Метод PrintHeader(WavHeader) выводит информацию из полей структуры в консоль программы.

В классе WavReader содержаться следующие поля для работы с файлами:

– file хранит ссылку на файл, с которым производятся операции.

– err поле используется для вызова системных ошибок при валидации файла.

– cplxData_Lch массив для хранения сэмпла для левого канала, в случае если аудиозапись представлена в стерео формате.

– cplxData_Rch массив для хранения сэмпла для правого канала, в случае если аудиозапись представлена в стерео формате.

– Filename хранит в себе название файла, с которым производятся операции.

– DSISE хранит размер одного сэмпла в байтах.

– bassIntensities массив для хранения частотных пиков, полученных в результате анализа аудиофайла.

– numSamples количество сэмплов, читаемых методом ReadFromFile().

– sampleBuff массив для чтения данных из аудиофайла.

– bytesRead хранит количество байт, полученное при чтении аудиофайла.

– PI содержит константное значение.

Метод ReadFromFile(String, WavHeader) производит чтение аудиофайла и возвращает сэмлы для левого и правого каналов. Эти данные в дальнейшем используются для анализа аудиосигнала.

Метод ValidateFile(String) производит валидацию названия, пути, а так же формата аудиофайла.

3.5 Алгоритм анализа аудиофайла

Данный алгоритм также является частью подключаемого плагина для работы и анализа аудио файла. К данному алгоритму относится класс FFT.

В классе FFT инициализированы следующие поля и методы:

– N частота дискретизации аудиосигнала.

– even и odd массивы хранят четные и нечетные значения на время выполнения быстрого преобразования Фурье.

– temp поле предназначено для хранения временного значения.

– bassIntensity хранит значение вычисленного частотного пика.

Метод ConvertToComplex(int16_t*) реализует преобразование массива значений типа int16_t, в массив комплексных чисел типа double. Данный метод необходим для последующей реализации метода быстрого преобразования Фурье.

Метод FFT(Double*) производит рекурсивную операцию преобразования Фурье.

Метод CalculateBassIntensity(int16_t*) осуществляет анализ массива, полученного из метода FFT(), и определяет существенные изменения нижних частотных диапазонов в конкретный интервал времени.

Метод CalculateBPM(Double*) возвращает значение ударов в минуту, вычисленных из корреляции последних нескольких результатов метода CalculateBassIntensity().

4 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ РИТМ ИГРЫ НА UNREAL ENGINE 4, ПОСТРОЕННОЙ НА АЛГОРИТМАХ ОБРАБОТКИ ЦИФРОВОЙ СПЕКТРОГРАММЫ

4.1 Характеристика разработанного проектного решения

Созданный дипломный проект представляет собой ритм игру на Unreal Engine 4 для операционной системы Windows 10, в основе которой лежат алгоритмы для чтения и обработки данных из WAV аудиофайлов. Основной разработанной частью является программная реализация, позволяющая обрабатывать аудиофайлы и использовать полученные данные, для задания ритма взаимодействия игрока, непосредственно с игровым персонажем и его окружением в самой игре.

Отличительная особенность проекта, заключается в минимизации необходимости создания собственных музыкальных композиций и лицензировании имеющихся треков, для последующего продвижения и продажи продукта.

Целью разработки данного проекта является упрощение этапа получения музыкального ритма из композиций и последующая экономия ресурсов на дальнейшем расширении и развитии продукта. Приложение предназначено для использования в развлекательных целях потребителей и коммерческой выгоды разработчика с внедренных способов монетизации.

Целевой аудиторией данного приложения являются люди, заинтересованные индустрией игр, а так же жанром ритм игр в частности. Продукт предлагает проведение досуга, минимальное время которого составляет около 3 минут, что является средним временем продолжительности проигрывания композиции.

Планируется распространение платной версии приложения через Steam.

4.2 Расчёт инвестиций в разработку программного средства

4.2.1 Расчёт зарплат на основную заработную плату разработчиков

Расчёт затрат на основную заработную плату разработчиков производится исходя из количества людей, которые занимаются разработкой программного продукта, месячной зарплаты каждого участника процесса разработки и сложности выполняемой ими работы. Затраты на основную заработную плату рассчитаны по формуле:

$$З_о = К_{пр} \sum_{i=1}^{n} З_{чi} \cdot t_i, \quad (4.1)$$

где $К_{пр}$ – коэффициент премий и иных стимулирующих выплат;

$n-$ категории исполнителей, занятых разработкой программного средства;

$З_{чi}$ – часовая заработная плата исполнителя *i*-й категории, р;

t_i – трудоемкость работ, выполняемых исполнителем *i*-й категории, ч.

Разработкой всего приложения занимается инженер-программист, Обязанности тестирования приложения лежат на инженере-тестировщике. Задачами инженера-программиста, являются написание алгоритма для чтения аудиофайла, разработки алгоритма для обработки цифровой спектрограммы аудиосигнала, разработка графического интерфейса игры, написание основной игровой логики, связи между данными, полученными в результате обработки аудиофайла и игровой логики. Инженер-тестировщик занимается выявлением неработоспособных частей приложения, а также оценивает пользовательский опыт, получаемый от использования приложения.

Месячная заработная плата основана на медианных показателях для Junior инженера-программиста за 2024 год по Республике Беларусь, которая составляет 1000 Долларов США в месяц, а для Junior инженера-тестировщика – 900 Долларов США[4]. По состоянию на 16 апреля 2024 года, 1 Доллар США по курсу Национального Банка Республики Беларусь составляет 3.2663 Белорусских рублей [5].

В перерасчёте на Белорусские рубли месячные оклады для инженера-программиста и инженера-тестировщика соответственно составляет 3 266,3 и 2 939,67 Белорусских рублей соответственно.

Часовой оклад исполнителей высчитывается путём деления их месячного оклада на количество рабочих часов в месяце, то есть 160 часов.

За количество рабочих часов в месяце для инженера-программиста и инженера-тестировщика принято соответственно 196 и 32 часа.

Коэффициент премии приравнивается к единице, так как она входит сумму заработной платы. Затраты на основную заработную плату приведены в таблице:

Таблица 4.1 – Затраты на основную заработную плату

Категория исполнителя	Месячный оклад, р	Часовой оклад, р	Трудоёмкость работ, ч	Итого, р
Инженер-программист	3 266,3	20,41	196	4 000,36
Инженер-тестировщик	2 939,67	18,37	32	587,84
Итого				4 588,2
Премия и иные стимулирующие выплаты (0%)				0
Всего затраты на основную заработную плату разработчиков				4 588,2

4.2.2 Расчёт затрат на дополнительную заработную плату разработчиков

Расчёт затрат на дополнительную заработную плату команды

разработчиков рассчитывается по формуле:

$$З_д = \frac{З_о \cdot Н_д}{100}, \quad (4.2)$$

где $Н_д$ – норматив дополнительной заработной платы.

Значение норматива дополнительной заработной платы принимает за 10 %.

4.2.3 Расчёт отчислений на социальные нужды

Размер отчислений на социальные нужды определяется согласно ставке отчислений, которая на апрель 2024 г. равняется 35%: 29% отчисляется на пенсионное страхование, 6% – на социальное страхование. Расчёт отчислений на социальные нужды вычисляется по формуле:

$$Р_{соц} = \frac{(З_о + З_д) \cdot Н_{соц}}{100}, \quad (4.3)$$

где $Н_{соц}$ – норматив отчислений в ФСЗН.

4.2.4 Расчёт прочих расходов

Расчёт затрат на прочие расходы определяется при помощи норматива прочих расчётов. Эта величина имеет значение 30%. Расчёт прочих расходов вычисляется по формуле:

$$Р_{пр} = \frac{З_о \cdot Н_{пр}}{100}, \quad (4.4)$$

где $Н_{пр}$ – норматив прочих расходов.

4.2.5 Расчёт расходов на реализацию

Для того, чтобы рассчитать расходы на реализацию, необходимо знать норматив расходов на неё. Принимаем значение норматива равным 3%. Формула, которая использована для расчёта расходов на реализацию:

$$Р_р = \frac{З_о \cdot Н_р}{100}, \quad (4.5)$$

где $Н_р$ – норматив расходов на реализацию.

4.2.6 Расчёт общей суммы затрат на разработку и реализацию

Определяем общую сумму затрат на разработку и реализацию как сумму ранее вычисленных расходов: на основную заработную плату разработчиков, дополнительную заработную плату разработчиков, отчислений на социальные нужды, расходы на реализацию и прочие расходы. Значение определяется по формуле:

$$З_р = З_о + З_д + Р_{соц} + Р_{пр} + Р_р \quad (4.6)$$

Таким образом, величина затрат на разработку программного средства высчитывается по указанной выше формуле и указана в таблице:

Таблица 4.2 – Затраты на разработку

Название статьи затрат	Формула/таблица для расчёта	Значение, р.
1. Основная заработная плата разработчиков	См. таблицу 4.1	4 588,2
2. Дополнительная заработная плата разработчиков	$З_д = \frac{4\,588{,}2 \cdot 10}{100}$	458,82
3. Отчисление на социальные нужды	$Р_{соц} = \frac{(4\,588{,}2 + 458{,}82) \cdot 35}{100}$	1 766,46
4. Прочие расходы	$Р_{пр} = \frac{4\,588{,}2 \cdot 30}{100}$	1 376,46
5. Расходы на реализацию	$Р_р = \frac{4\,588{,}2 \cdot 3}{100}$	137,65
6. Общая сумма затрат на разработку и реализацию	$З_р = 4\,588{,}2 + 458{,}82 + +1\,766{,}46 + 1\,376{,}46 + 137{,}65$	8 327,59

4.3 Расчёт экономического эффекта от реализации программного средства на рынке

Для расчёта экономического эффекта организации-разработчика программного средства, а именно чистой прибыли, необходимо знать такие параметры как объем продаж, цену реализации и затраты на разработку.

Соответственно необходимо создать обоснование возможного объёма продаж, количества проданных лицензий программного средства, купленного пользователями. По данным платформы, 14 апреля 2024 года платформой Steam активно пользуются 35,28 миллиона человек. На март этого же года, по статистике используемых устройств на платформе Steam, 54.40% всех пользователей используют операционную систему Windows 10 [6]. Предположим, что количество пользователей, заинтересованных жанром

ритм игр, составляет 4% от общего количества пользователей. Итоговое количество пользователей, соответствующих системным требованиям проекта, а так же заинтересованных жанром, составляет около 765 тысяч пользователей.

Учитывая небольшое количество конкурентов на платформе, небольшие бюджеты проекта, а так же отсутствие известности разработчиков, предположим, что 10% от общего количества пользователей Steam станут покупателями данного продукта. Это составляет около 76,500 человек.

Стоимость низкобюджетных игр от мало известных и неизвестных студий разработчиков на платформе Steam варьируется от 6 до 12 Долларов США. Расчёты в белорусском регионе Steam ведутся в Долларах США. Наиболее оптимальной ценой полной версии продукта предполагается 7 Долларов США, с вычетом комиссии платформы Steam в 10% составляет 6,3 Доллара США. Таким образом, отпускная цена копии программного средства составляет 20,58 Белорусских рубля.

Для расчёта прироста чистой прибыли необходимо учесть налог на добавленную стоимость, который высчитывается по следующей формуле:

$$\text{НДС} = \frac{\text{Ц}_{\text{отп}} \cdot N \cdot \text{Н}_{\text{д.с}}}{100\% + \text{Н}_{\text{д.с}}}, \tag{4.7}$$

где N – количество копий(лицензий) программного продукта, реализуемое за год, шт.;

$\text{Ц}_{\text{отп}}$ – отпускная цена копии программного средства, р. ;

N – количество приобретённых лицензий;

$\text{Н}_{\text{д.с}}$ – ставка налога на добавленную стоимость, %.

Ставка налога на добавленную стоимость по состоянию на 15 апреля 2024 года, в соответствии с действующим законодательством Республики Беларусь, составляет 20%. Используя данное значение, посчитаем НДС:

$$\text{НДС} = \frac{20{,}58 \cdot 76\,500 \cdot 20\%}{100\% + 20\%}, = 262\,395 \text{ р.}$$

Посчитав налог на добавленную стоимость, можно рассчитать прирост чистой прибыли, которую получит разработчик от продажи программного продукта. Для этого используется формула:

$$\Delta\text{П}_{\text{ч}}^{\text{р}} = (\text{Ц}_{\text{отп}} \cdot N - \text{НДС}) \cdot \text{Р}_{\text{пр}} \cdot \left(1 - \frac{\text{Н}_{\text{п}}}{100}\right), \tag{4.8}$$

где *N* – количество копий(лицензий) программного продукта, реализуемое за год, шт.;

$\text{Ц}_{\text{отп}}$ – отпускная цена копии программного средства, р.;

НДС – сумма налога на добавленную стоимость, р.; $Н_п$ – ставка налога на прибыль, %;

$Р_{пр}$ – рентабельность продаж копий;

$Р_{пр}$ – рентабельность продаж копий.

Ставка налога на прибыль, согласно действующему законодательству, по состоянию на 16.04.2023 равна 20%. Рентабельность продаж копий взята в размере 40%. Зная ставку налога и рентабельность продаж копий (лицензий), рассчитывается прирост чистой прибыли для разработчика:

$$\Delta П_ч^р = (20{,}58 \cdot 76\,500 - 262\,395) \cdot 40\% \cdot \left(1 - \frac{20}{100}\right) = 41\,983\,200$$

4.4 Расчёт показателей экономической эффективности разработки и реализации программного средства на рынке

Для того, чтобы оценить экономическую эффективность разработки и реализации программного средства на рынке, необходимо рассмотреть результат сравнения затрат на разработку данного программного продукта, а также полученный прирост чистой прибыли за год.

Сумма затрат на разработку меньше суммы годового экономического эффекта, поэтому можно сделать вывод, что такие инвестиции окупятся менее, чем за один год.

Таким образом, оценка экономической эффективности инвестиций производится при помощи расчёта рентабельности инвестиций (Return on Investment, ROI). Формула для расчёта ROI:

$$ROI = \frac{\Delta П_ч^р - З_р}{З_р} \cdot 100\% \quad (4.9)$$

где $\Delta П_ч^р$ – прирост чистой прибыли, полученной от реализации программного средства на рынке информационных технологий, р.;

$З_р$ – затраты на разработку и реализацию программного средства, р.

$$ROI = \frac{41\,983\,200 - 8\,327{,}59}{8\,327{,}59} \cdot 100\% = 504045{,}86\%$$

4.5 Вывод об экономической целесообразности реализации проектного решения

Проведённые расчёты экономического обоснования позволяют сделать предварительный вывод о целесообразности разработки данного программного продукта. Общая сумма затрат на разработку и реализацию составила 8 327,59 Белорусских рублей, а отпускная цена была установлена

на уровне 20,58 Белорусских рублей.

Прирост чистой прибыли за год, исходя из предполагаемого объёма продаж в размере 76 500 базовых версий в год, составляет 41 983 200 Белорусских рублей. Рентабельность инвестиций за год составляет 504045,86%.

Это результаты вычислений показывают, что разработка данного программного продукта является целесообразной и реализация программного средства по установленной цене имеет смысл.

Однако, следует учитывать возможные риски, связанные с конкуренцией со стороны крупных и средне бюджетных продуктов для выбранной целевой платформы, а так же крупных продуктов на других популярных платформах с большой аудиторией. Релиз продукта в тот же временной период может привести к незамеченности данного продукта на рынке, на фоне информационного шума, созданного конкурентными рекламными компаниями. Кроме того, высокая рентабельность связана с рисками, и расчётные результаты были получены при предполагаемом объёме продаж в 76 500 копий в год.

Тем не менее, при поддержке проект может получить долгосрочное и успешное развитие, благодаря чему количество проданных копий может превысить предполагаемое количество. Добавление, после релиза базовой версии продукта, дополнительной модели монетизации, посредством покупки внутриигрового контента или рекламы, позволит удержать прибыль в долгосрочной перспективе. В целом, инвестирование в предложенный проект хоть и рискованно, но способно привести к большой прибыли с минимальным количеством затрат на разработку, что делает инвестицию вполне оправданной.

ЗАКЛЮЧЕНИЕ

Во время работы над преддипломной практикой была изучена предметная область, рассмотрены аналоги реализуемого проекта, как с технической, так и экономической точки зрения, детально рассмотрен процесс разработки программных продуктов, начата разработка программной части проекта.

Также в ходе преддипломной практики была разработаны следующие разделы дипломного проектирования: введение, обзор литературы, системное проектирование, структурная схема, функциональное проектирование, схема диаграммы классов, а так же экономическое обоснование разработки данного программного продукта для мирового рынка.

Среди достоинств разрабатываемого программного продукта можно выделить следующее:

– независимость элементов проекта;

– возможность использования любой композиции;

– возможность быстрого портирования продукта на другие платформы;

– поддержка клавиатуры и геймпада, как устройств управления;

– отсутствие встроенной рекламы на релизе.

К минусам разрабатываемого программного продукта можно отнести следующее:

– выпуск продукта для единственной операционной системы;

– поддержка малого количества аудиоформатов;

– отсутствие визуальных эффектов;

– малый набор базовых игровых механик;

– отсутствие собственного саундтрека.

Благодаря системному подходу к проектированию возможно дальнейшее улучшение и расширение функциональности проекта в целом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Видеоигра «Guitar Hero» фото [Электронный ресурс] – Режим доступа: https://www.youtube.com/watch?v=slJA7xwDRWk – Дата доступа: 02.04.2024

[2] Официальная Web-библиотека для игры «Osu!» фото [Электронный ресурс] – Режим доступа: https://osu.ppy.sh/beatmapsets?q=rock – Дата доступа: 02.04.2024

[3] Видеоигра «One Hand Clapping» фото [Электронный ресурс] – Режим доступа: https://www.youtube.com/watch?v=b9j8baO7nlU – Дата доступа: 04.04.2024

[4] Интернет-издание «Dev.by» фото [Электронный ресурс] – Зарплата в ИТ – Режим доступа: https://salaries.devby.io – Дата доступа: 16.04.2024

[5] Сайт Национальный банк Республики Беларусь фото [Электронный ресурс] – Официальные курсы белорусского рубля по отношению к иностранным валютам, устанавливаемые Национальным банком Республики Беларусь ежедневно, на 16.04.2024 – Режим доступа: https://www.nbrb.by/statistics/rates/ratesdaily.asp – Дата доступа: 16.04.2024

[6] Web-сервис «Steam» фото [Электронный ресурс] – Статистика устройств пользователей – Режим доступа: https://store.steampowered.com/hwsurvey/ – Дата доступа: 16.04.2024

ПРИЛОЖЕНИЕ А

(обязательное)
Вводный плакат

ПРИЛОЖЕНИЕ Б

(обязательное)
Схема структурная

ПРИЛОЖЕНИЕ В

(обязательное)
Диаграмма классов