



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ**

НА ТЕМУ:

***«Способы моделирования и создания анимаций для
использования на движке Unreal Engine»***

Студент РК6-84Б

(Подпись, дата)

Шендрик Д.А.

(И.О. Фамилия)

Руководитель ВКР

(Подпись, дата)

Витюков Ф.А.

(И.О. Фамилия)

Нормоконтролёр

(Подпись, дата)

Грошев С.В.

(Фамилия И.О.)

**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)
(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой РК6
(Индекс)

Карпенко А.П.
(Фамилия И.О.)

« » 2024 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра

Студент группы РК6-84Б

PK6-84Б

Шендрек Даниил Андреевич

(фамилия, имя, отчество)

Тема квалификационной работы: «Способы моделирования и создания анимаций для использования на движке Unreal Engine»

Источник тематики (НИР кафедры, заказ организаций и т.п.): кафедра.

Тема квалификационной работы утверждена распоряжением по факультету РК №__ от __ __

Техническое задание

Часть 1. Аналитическая часть

Изучить возможности 3D моделирования в программе Blender. Изучить полный цикл создания статических 3d-моделей. Изучить средства разработки, предоставляемые трехмерным движком Unreal Engine. Изучить и сравнить различные программы для захвата движений.

Часть 2. Практическая часть 1. 3d-моделирование

Создать простые 3d-модели, пройдя полный цикл создания статических моделей, провести их оптимизацию. Выполнить экспорт ассетов и их импорт в движок Unreal Engine 4.2 и решить все возникшие при этом ошибки.

Часть 3. Практическая часть 2. Создание видеоигровой боевой системы

Разработать систему боя для игрового персонажа, включающую в себя следующие действия: ходьба, бег, перекаты, удары, блок. Разработать аналогичную систему действий для неигрового персонажа, как ответные реакции на действия игрового персонажа, под управлением внутриигрового ИИ. Создать игровую арену для демонстрации взаимодействия персонажей с объектами окружающего мира.

Оформление выпускной квалификационной работы

Расчетно-пояснительная записка на **XX** листах формата А4.

Перечень графического (илюстративного) материала (чертежи, плакаты, слайды и т.п.):

Работа содержит X графических листов формата А4.

Дата выдачи задания: «23» февраля 2024 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до «11» июня 2024 г.

Руководитель квалификационной работы

(Подпись, дата)

Витюков Ф.А.

(Фамилия И.О.)

Студент

(Подпись, дата)

Шендрек Д.А.

(Фамилия И.О.)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ РК
КАФЕДРА РК6
ГРУППА РК6-84Б

УТВЕРЖДАЮ
Заведующий кафедрой РК6
(Индекс)
Карпенко А.П.
(Фамилия И.О.)
« » 2024 г.

КАЛЕНДАРНЫЙ ПЛАН
ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Студент группы РК6-84Б

Шендрик Даниил Андреевич
(фамилия, имя, отчество)

Тема квалификационной работы: «Способы моделирования и создания анимаций для использования на движке Unreal Engine»

| № п/п | Наименование этапов выпускной квалификационной работы | роки выполнения этапов | | Отметка о выполнении | |
|----------|--|------------------------|-------|----------------------|--------------|
| | | план | факт | Должность | ФИО, подпись |
| 1. | Задание на выполнение работы. Формулирование проблемы, цели и задач работы | 23.02.2024 | _____ | Руководитель ВКР | Витюков Ф.А. |
| 2. | 1 часть. Аналитическая часть | XX.0X.2024 | _____ | Руководитель ВКР | Витюков Ф.А. |
| 3. | 2 часть. Практическая часть 1 | 22.03.2024 | _____ | Руководитель ВКР | Витюков Ф.А. |
| 4. | 3 часть. Практическая часть 2 | XX.0X.2024 | _____ | Руководитель ВКР | Витюков Ф.А. |
| 5. | 1-я редакция работы | XX.0X.2024 | _____ | Руководитель ВКР | Витюков Ф.А. |
| 6. | Подготовка доклада и презентации | XX.0X.2024 | _____ | Студент | Шендрик Д.А. |
| 7. | Заключение руководителя | XX.06.2024 | _____ | Руководитель ВКР | Витюков Ф.А. |
| 8. | Допуск работы к защите на ГЭК (нормоконтроль) | XX.06.2024 | _____ | Нормоконтролер | Грошев С.В. |
| 9. | Внешняя рецензия | XX.06.2024 | _____ | | |
| 10. | Защита работы на ГЭК | XX.06.2024 | _____ | | |

Студент _____
(подпись, дата)

Шендрик Д.А.
(Фамилия И.О.)

Руководитель ВКР _____
(подпись, дата)

Витюков Ф.А.
(Фамилия И.О.)

РЕФЕРАТ

Расчетно-пояснительная записка 76 с., 42 рис., 1 табл., 12 источников.

Работа посвящена области 3D моделирования, а именно создания моделей в программе Blender, перенос их на движок Unreal Engine и создание анимации “вручную” или средствами motion capture. Был рассмотрен полный цикл создания 3d-моделей и применены на практике простые инструменты для моделирования, анимации и создания системы частиц. В работе созданы несколько простых 3D моделей для демонстрации возможностей программы Blender и проведен их импорт в Unreal Engine для дальнего использования. Создана боевая система для игрока и неигрового персонажа под управлением ИИ, реализованы анимации перемещения и боя, создана программный код на языке программирования C++ для Unreal Engine 4 для демонстрации всех аспектов боевой системы и для демонстрации взаимодействия персонажей с объектами окружающего мира.

Тип работы: выпускная квалификационная работа.

Тема работы: Способы моделирования и создания анимаций для использования на движке Unreal Engine.

Объекты исследований: 3d-моделирование, разработка внутриигровой боевой систем, создание анимации.

СОДЕРЖАНИЕ

| | |
|---|----|
| ВВЕДЕНИЕ | 8 |
| 1 Создание 3D-моделей | 11 |
| 1.1 Обзор Blender..... | 12 |
| 1.2 Процесс создания 3D моделей в Blender | 13 |
| 1.2.1 Модель “Фабрика” | 14 |
| 1.2.2 Модель “Owl-Cat” | 19 |
| 1.3 Импорт в Unreal Engine: связь между Blender и UE4 | 22 |
| 1.4 Альтернативные средства создания 3D-моделей и анимации | 24 |
| 1.4.1 Средства создания моделей: Metahuman Creator | 25 |
| 1.4.2 Новые возможности создание анимации в Unreal Engine 5.4 | 27 |
| 1.4.3 Сравнение программ для motion capture | 30 |
| 2 Создание видеоигровой боевой системы | 36 |
| 2.1 Обзор существующих боевых систем | 36 |
| 2.2 Создание анимации и логики боевой системы | 43 |
| 2.2.1 Анимации боя | 44 |
| 2.2.2 Движение: ходьба, бег, перекаты | 48 |
| 2.2.3 Бой: удар, блок, получение урона..... | 51 |
| 2.3 Программирование боевой логики противника..... | 54 |
| 2.3.1 Передвижение противника..... | 55 |
| 2.3.2 Логика боя противника | 56 |
| 2.4 Реализация системы здоровья..... | 58 |
| 2.4.1 Здоровье персонажей | 58 |
| 2.4.2 Визуальные элементы, показывающие здоровье..... | 59 |
| 2.4.3 Процесс респавна противников | 62 |
| 2.4.4 Смерть игрового персонажа..... | 63 |
| 2.5 Реализация системы атак с умным буфером..... | 64 |
| 2.6 Реализация системы визуального захвата противника | 67 |
| 2.6.1 Работа камеры в режиме фиксации | 67 |

| | |
|---|----|
| 2.6.2 Анимации в режиме фиксации камеры | 68 |
| ЗАКЛЮЧЕНИЕ | 72 |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 74 |
| ПРИЛОЖЕНИЕ А | 76 |

ВВЕДЕНИЕ

В современном мире трехмерное моделирование стало неотъемлемой частью индустрии развлечений, виртуальной реальности, дизайна и многих других областей. Создание трехмерных моделей предоставляет уникальные возможности в визуализации и взаимодействии с окружающим миром, а также играет ключевую роль в разработке современных компьютерных игр и виртуальных симуляторов. А с взрывным развитием виртуальных технологий в последнее десятилетие встает необходимость в поиске инновационных решений для создания максимально реалистичных и взаимодействующих анимированных виртуальных персонажей.

Одним из важных инструментов в арсенале 3D-художников и разработчиков является программа Blender. Blender – это мощный, свободно распространяемый инструмент для трехмерного моделирования, анимации и создания визуальных эффектов [1]. Его богатый набор инструментов позволяет создавать сложные модели и эффекты, а открытый исходный код способствует активному распространению и постоянному улучшению функционала.

В контексте визуальной разработки и создания игрового контента, Blender часто используется для моделирования объектов, персонажей и окружения. Однако, чтобы в полной мере раскрыть потенциал этих модели и внедрить их в некоторый проект, необходимо использовать мощные средства разработки, такие как движок Unreal Engine.

Unreal Engine предоставляет разработчикам инструменты для создания интерактивных 3D-приложений, использующих передовые технологии визуализации и физики [4]. Он имеет множество прикладных функций, таких как работа с текстурами и материалами, рендеринг графики, создание и управление объектами в игровом поле, мощный высокоуровневый язык C++ для написания игровой логики и многое другое.

Однако, чтобы персонажи стали по-настоящему живыми, требуется качественная и реалистичная анимация движений. По данному вопросу в

исследовании сосредоточено внимание на ключевых инструментах, необходимых для создания и анимации виртуальных персонажей в Unreal Engine, а именно Metahuman Creator и программы захвата движений. Этот аспект исследования фокусируется на программном обеспечении, использующем VMC Protocol. Это семейство приложений предоставляет разработчикам средства захвата движений, от лицевой анимации до движений рук и пальцев, что придает виртуальным персонажам естественность и реализм.

Для создания полноценной и захватывающей видеоигрового опыта была создана боевая система. В рамках данной работы планируется разработка боевой видеоигровой системы, вдохновленной характеристиками и механиками, присущими играм подобного жанра, таким как Witcher 2-3 и Dark Souls.

Боевая система будет включать в себя базовый набор действий и маневров для игрового персонажа под управлением пользователя, обеспечивающих подвижность персонажа и его взаимодействие с миром. Персонаж сможет выполнять различные действия, такие как ходьба, бег, удары, перекаты, а также занимать защитную стойку для блокировки ударов.

Кроме того, для создания демонстрации возможностей созданной боевой системы необходимо обеспечить умное поведение противника. В этой работе будет разработан и реализован искусственный интеллект, управляющий противником, который будет обладать аналогичным набором действий и умениями, как и у игрового персонажа.

Для полного погружения игроков в игровой мир также важно учитывать взаимодействие персонажей с окружающим ландшафтом. В рамках исследования будет изучено и реализовано взаимодействие персонажей с различными элементами окружающего мира, такими как поверхности, препятствия и объекты, что добавит реализма и тактических возможностей в боях.

Работа направлена на определение оптимальных комбинаций этих инструментов и программ для достижения выдающегося уровня

интерактивности и реализма виртуальных персонажей и игровых механик, для взаимодействия с игровым миром в Unreal Engine.

Цели работы:

1. создать и оптимизировать 3d-модели в программе Blender для отработки полного цикла 3d-моделирования;
2. импортировать модели в движок Unreal Engine и исправить все возникшие ошибки визуализации;
3. разработать боевую систему для игрового персонажа, включающую следующие действия: ходьба, бег, перекаты, удары, блок;
4. разработать систему ИИ с аналогичной боевой системой для неигрового противника;

Для выполнения работы были использованы средства следующих программ:

- Unreal Engine 4 – разработка внутриигровых систем;
- Blender – создание 3d-моделей и анимации;

1 Создание 3D-моделей

В мире современной компьютерной графики 3D-моделирование и анимация играют важную роль, становясь неотъемлемой частью процесса создания различного цифрового контента, начиная от анимационных фильмов и компьютерных игр, и заканчивая архитектурным проектированием и медицинской визуализацией. В этом вступлении мы рассмотрим различные программы и методы 3D-моделирования, а также основные инструменты и области применения этих технологий.

3D-моделирование включает в себя широкий спектр программ и методов, позволяющих создавать трехмерные объекты и сцены. Среди самых популярных программ для 3D-моделирования следует выделить Blender[1], Autodesk Maya, 3ds Max, Cinema 4D и ZBrush. Каждая из этих программ обладает своими уникальными особенностями и предоставляет различные инструменты для создания и редактирования 3D-моделей.

3D-моделирование находит применение в различных областях, включая анимационное кино, разработку компьютерных игр, архитектурное проектирование, медицинскую визуализацию, индустриальный дизайн и многое другое. Благодаря возможности создания реалистичных трехмерных объектов и сцен, 3D-моделирование позволяет воплощать в жизнь самые смелые творческие идеи и решать разнообразные задачи.

Среди множества программ для 3D-моделирования и анимации особое место занимают Blender и движок Unreal Engine [4]. Blender – это мощный и бесплатный инструмент с открытым исходным кодом, который предоставляет широкие возможности для создания 3D-моделей, анимации и спецэффектов. Unreal Engine, в свою очередь, является одним из ведущих игровых движков, который используется для разработки высококачественных игр и интерактивных визуальных проектов.

1.1 Обзор Blender

Blender предоставляет впечатляющий инструментарий для трехмерного моделирования, анимации и визуализации. Одним из фундаментальных понятий в моделировании Blender является структура модели, состоящая, как и в большинстве программ для 3D моделирования из вершин, ребер и граней. Вершины – это точки в пространстве, ребра – линии, соединяющие вершины, и грани – поверхности, образованные ребрами [1].

Одним из первых этапов создания любой модели в Blender является выбор базовой формы, которую можно легко изменять с использованием инструментов преобразования. Моделирование в Blender часто начинается с применения таких базовых примитивов, как кубы, сферы или цилиндры, которые затем могут быть изменены и детализированы [2]. Возможности моделирования включают в себя создание сложных геометрических форм, а также настройку поверхностей и текстур для достижения желаемого визуального эффекта.

Одним из ключевых инструментов является "Edit Mode", который позволяет пользователю манипулировать вершинами, гранями и ребрами модели непосредственно. Инструменты перемещения, вращения и масштабирования позволяют точно настраивать форму объекта. Более сложные инструменты экструдирования, выдавливания, создания фасок и разрезов добавляют новые вершины, ребра и грани, делая модель более сложной и настраиваемой. Дополнительно, Blender предоставляет мощные инструменты для создания сложных форм с использованием модификаторов.

Модификаторы представляют собой инструменты, которые позволяют применять различные эффекты и изменения к геометрии модели. Например, "Mirror Modifier" отражает часть модели, создавая симметричные объекты; или модификатор "Array", который создает массив тел, которые можно расставлять по направлению кривой, с помощью еще одного модификатора "Curve". Это лишь одни из множества модификаторов, которые предоставляются Blender, каждый из которых обеспечивает уникальные возможности для моделирования.

Кроме того, инструменты скульптурирования в Blender предоставляют манипулировать большим количеством вершин модели одновременно, но с разной силой, таким образом получая очень детализированные модели уникальной формы. Этот инструмент дает 3D-художнику цифровой аналог глины, с которой можно легко формировать взаимодействовать.

Одной из ключевых особенностей Blender является его многозадачность. В программе можно создавать не только статичные 3D модели, но и разрабатывать анимации с использованием удобного редактора ключевых кадров. Анимационные инструменты включают в себя возможность управления скелетной архитектурой (риггинг), создание и редактирование кадров анимации, а также применение различных эффектов и переходов.

Одним из сильных аспектов Blender является его открытость и активное сообщество пользователей. Благодаря этому, пользователи имеют доступ к богатой базе бесплатных ресурсов, таких как плагины, текстуры и модели, что существенно упрощает процесс творчества.

Важно отметить, что Blender обладает удивительной гибкостью в решении различных задач. Например, программа поддерживает не только создание статических объектов, но и моделирование поверхностей для последующего 3D печати. Это демонстрирует широкий спектр применений Blender в индустрии дизайна, искусства и технической разработки.

Таким образом, Blender представляет собой мощное и универсальное средство для творчества в трехмерной графике. В дальнейшем, мы рассмотрим, каким образом созданные в Blender 3D модели интегрируются в игровой мир при помощи Unreal Engine.

1.2 Процесс создания 3D моделей в Blender

В практической части работы были созданы две модели в программе для 3D моделирования Blender. Первая модель – минималистичная фабрика с конвейерной линией, где вырабатывались основные навыки создания моделей, текстурирования, а также создание простых анимаций. Второй моделью является статическая модель мифического пушистого существа, в этой модели использовался скульптинг для создания сложной формы модели и система частиц для создания шерсти.

1.2.1 Модель “Фабрика”

Создание практически любой модели требует сбора референсов, которые играют ключевую роль в создании реалистичной модели [2]. На основе изученных изображений и концептуальных скетчей формируется общая композиция будущей модели, композиция и способ создания частей проекта.

Для определения общей композиции и размеров конвейерной линии были размещены базовые геометрические примитивы: кубы, цилиндры и сферы. Этот позволяет быстро оценить пропорции будущей модели и начать придавать ей форму. **Ошибка! Источник ссылки не найден.** демонстрирует данный этап моделирования.

Следующим этапом – добавление крупных деталей, таких как скругления и вырезы, с помощью модификаторов булевых функций, чтобы создать форму и структуру основных объектов модели. Используя инструменты экструдера, создания скругленных фасок и других, формируются элементы, которые придают модели характер и функциональность.

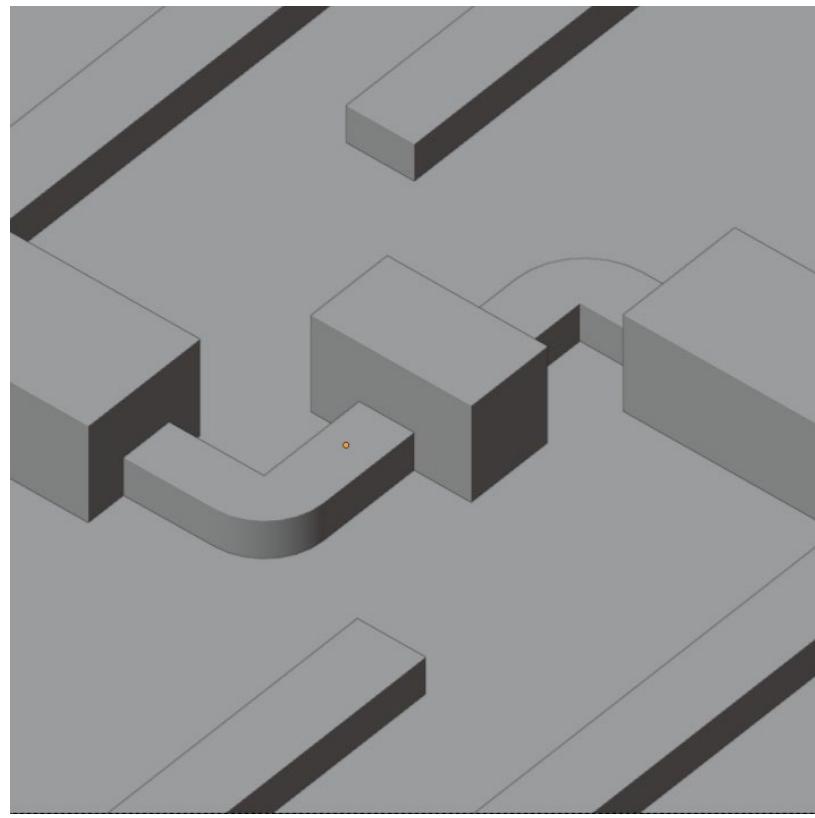


Рисунок 1 – Размещение примитивов для будущей модели

После создания основной структуры конвейера фокус переносится на добавление более мелких деталей, создающих более сложную и интересную картинку (Рисунок 2). Трубы, лампы и вентиляторы внедряются с помощью инструментов Blender, учитывая их расположение и функциональность в рамках общего дизайна. Трубы создаются за счет превращения кривых в объемные тела. Лампы – несколько объединенных цилиндров друг над другом. С помощью деформации и массива, получаются лопасти, добавление еще нескольких примитивов создает еще более проработанные модели вентиляторов. Примерно также создается модель движущейся конвейерной линии, только размноженные массивом модели затем располагаются по направлению заданной кривой с помощью модификатора “Curve”.

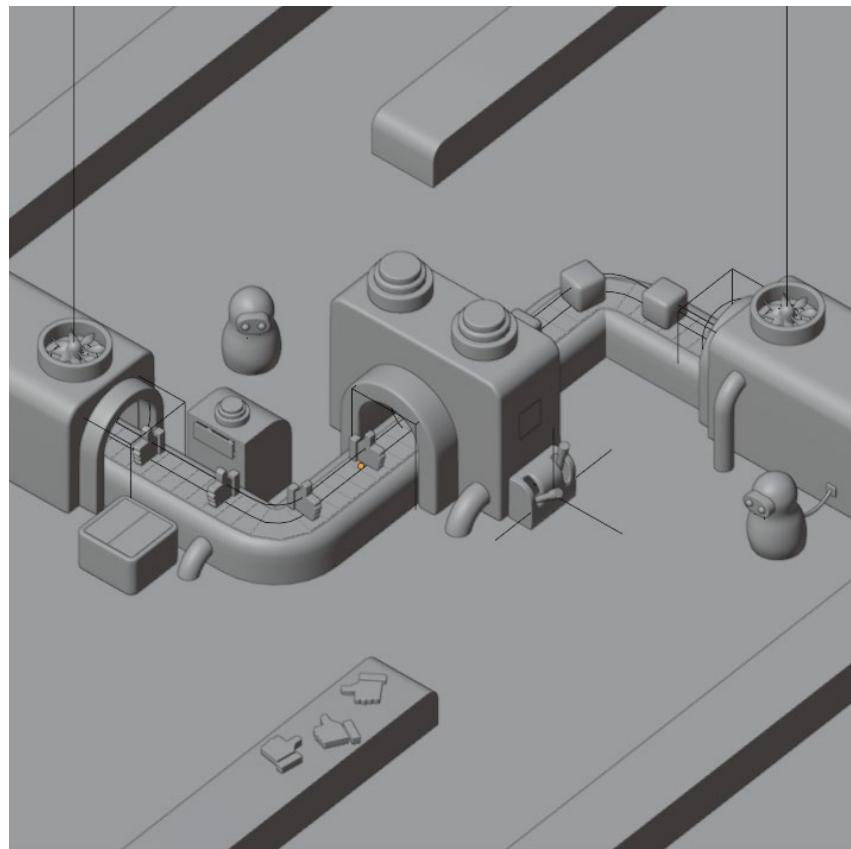


Рисунок 2 – Модель с полной детализацией без текстур и материалов

Далее были добавлены более сложные объекты, например роботы и “лайки”. Эти модели были созданы практически вручную, передвижением вершин или групп вершин, чтобы добиться определенной формы.

Следующий этап работы: текстурирование и присвоение материалов. С использованием текстур и шейдеров создаются детали, придающие модели фотorealистичность. Также регулируются освещение и тени для достижения желаемого визуального эффекта.

Большинству тел достаточно присвоить цвет и определенные свойства материалов. Однако некоторым требуется учен UV разверток. Blender позволяет очень удобно создавать и изменять данные развертки, выбирая грани прямо на модели, что делает этот процесс очень удобным, так как сразу виден результат.

Помимо присвоения всем телам своих материалов в модель были добавлены источники света и камера. Результат работы изображен ниже (Рисунок 3).

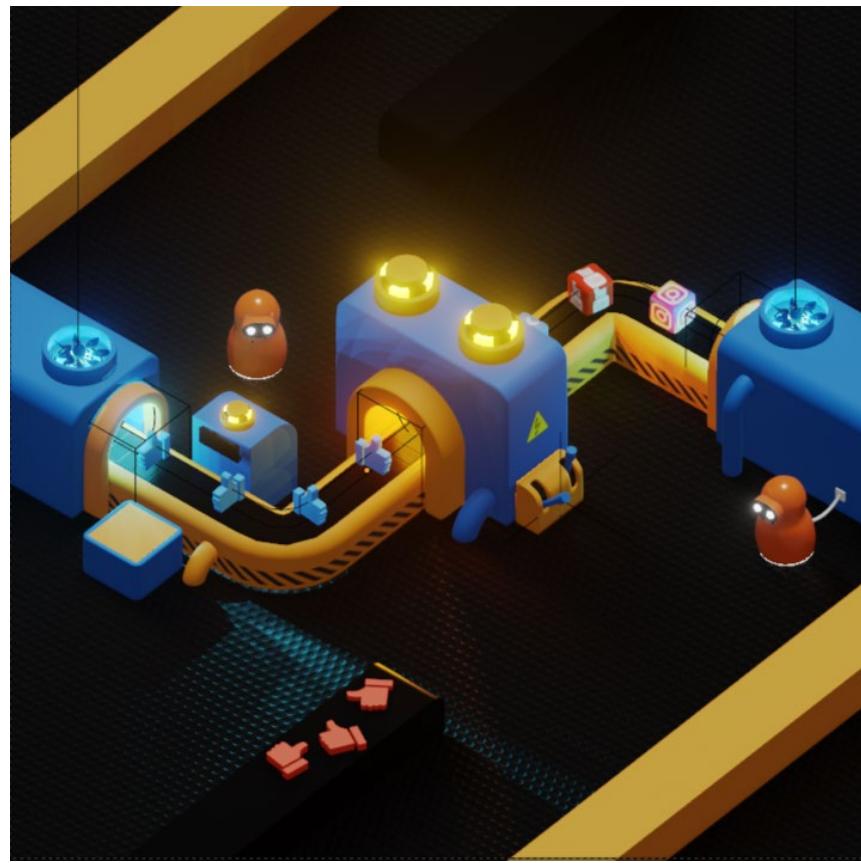


Рисунок 3 – Модель “Фабрика” с текстурами и настроенными источниками света

После завершения моделирования конвейерной линии в Blender следующий этап - создание анимации движения. Blender предоставляет широкий спектр инструментов для анимации, и в данном случае, создается анимация передвижения блоков вдоль конвейера.

Настройка движения включает определение скорости, направления и взаимодействия объектов между собой. Каждому объекту можно задать положение в определенный момент времени - ключевой кадр (keyframe). Другой способ создать анимацию – задать кривую, по которой будет двигаться объект. В практическом задании были использованы оба способа: движение по кривой для ленты и основных объектов на ней и ключевые кадры для выпадающего “дизлайка”, робота и света. Механизмы анимации в Blender позволяют создавать плавные и реалистичные движения, что существенно улучшает визуальный

аспект разрабатываемой сцены. После завершения этапа анимации получается динамичный и интересный объект, пригодный для интеграции в игровой мир.

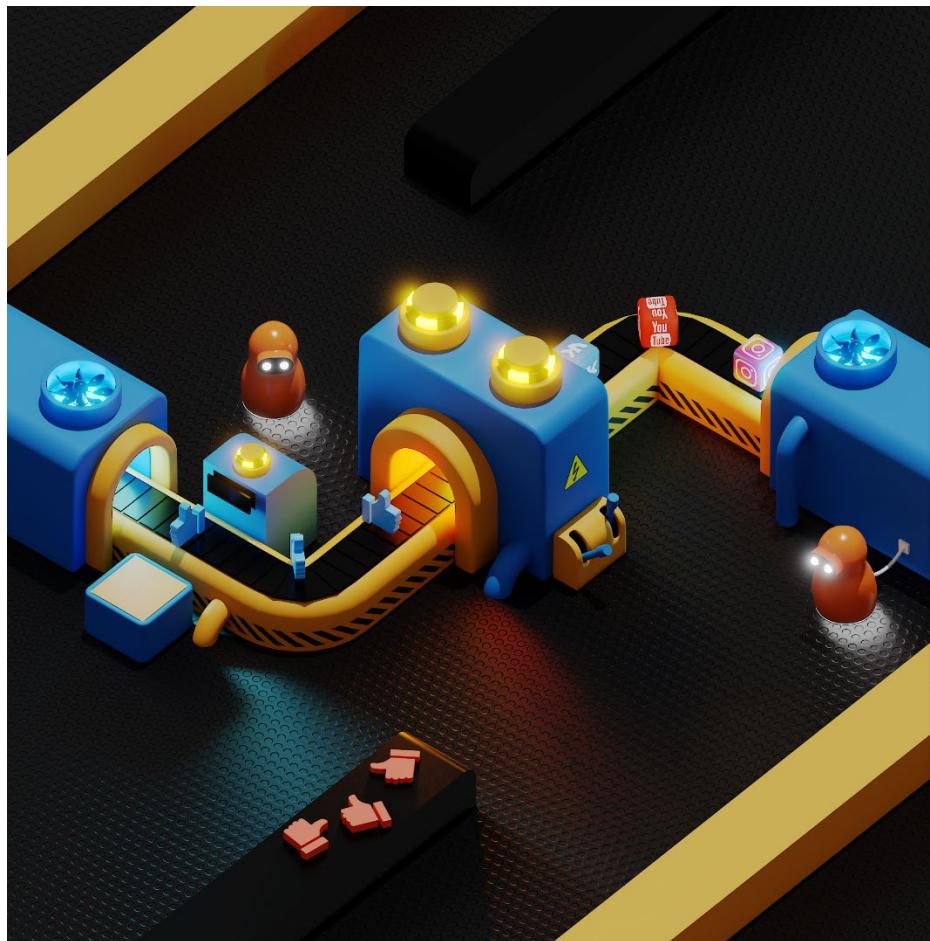


Рисунок 4 – Результат рендеринга готовой модели на движке Cycles

После успешного завершения процесса моделирования и анимации наступает этап рендеринга (Рисунок 4). Blender предлагает несколько встроенных движков рендеринга, таких как Cycles и Eevee, обеспечивающих высококачественные графические результаты.

Движок Cycles предоставляет фотoreалистичные изображения, основанные на трассировке лучей, что делает его идеальным выбором для создания высококачественных визуальных эффектов. С другой стороны, Eevee обеспечивает быстрый превью и интерактивный рендеринг, что ускоряет процесс разработки.

Для создания одного кадра и короткой анимации был выбран движок Cycles с добавлением небольшого количества эффектов, например эффект Glare для увеличения свечения ламп.

Этап рендеринга в Blender завершает процесс создания 3D модели фабрики. Далее готовый контент можно интегрировать в Unreal Engine.

1.2.2 Модель “Owl-Cat”

Как и в предыдущей модели, первый шаг в создании уникальной модели, мы провели сбор референсов и осуществили начальное моделирование, определяя общую форму существа [3]. Далее, для детализации и добавления выразительности, была применена техника скульптинга.

Для увеличения плотности сетки и обеспечения большей детализации, мы использовали инструменты сабдивизии Blender. Этот метод позволяет увеличить количество полигонов и более точно отобразить форму существа, что необходимо для последующего скульптинга.

Следующий этап включал в себя процесс скульптинга, где мы добавляли дополнительные детали, такие как когти, клюв, и выражение мордочки. С инструментами сглаживания и формирования деталей, Blender предоставил удобное и гибкое пространство для художественного творчества. Регулируя уровни детализации, мы добивались нужного баланса между реализмом и эстетикой существа.

Таким образом, процесс скульптинга в Blender является ключевым этапом в создании уникальной модели полукота-полусовы, придавая ей индивидуальность и характер.

Для перехода от высоко детализированной High-poly модели к оптимизированной Low-poly версии, мы применили метод Texture baking, обеспечивая сохранение общей формы существа при снижении количества полигонов.

Сначала создавалась копия High-poly модели, содержащая все детали и высокое количество полигонов. Далее, с использованием модификаторов, таких как Decimate, количество полигонов в копии модели уменьшалось, сохраняя при этом основные формы и пропорции. Этот этап требует баланса между оптимизацией и сохранением ключевых деталей.

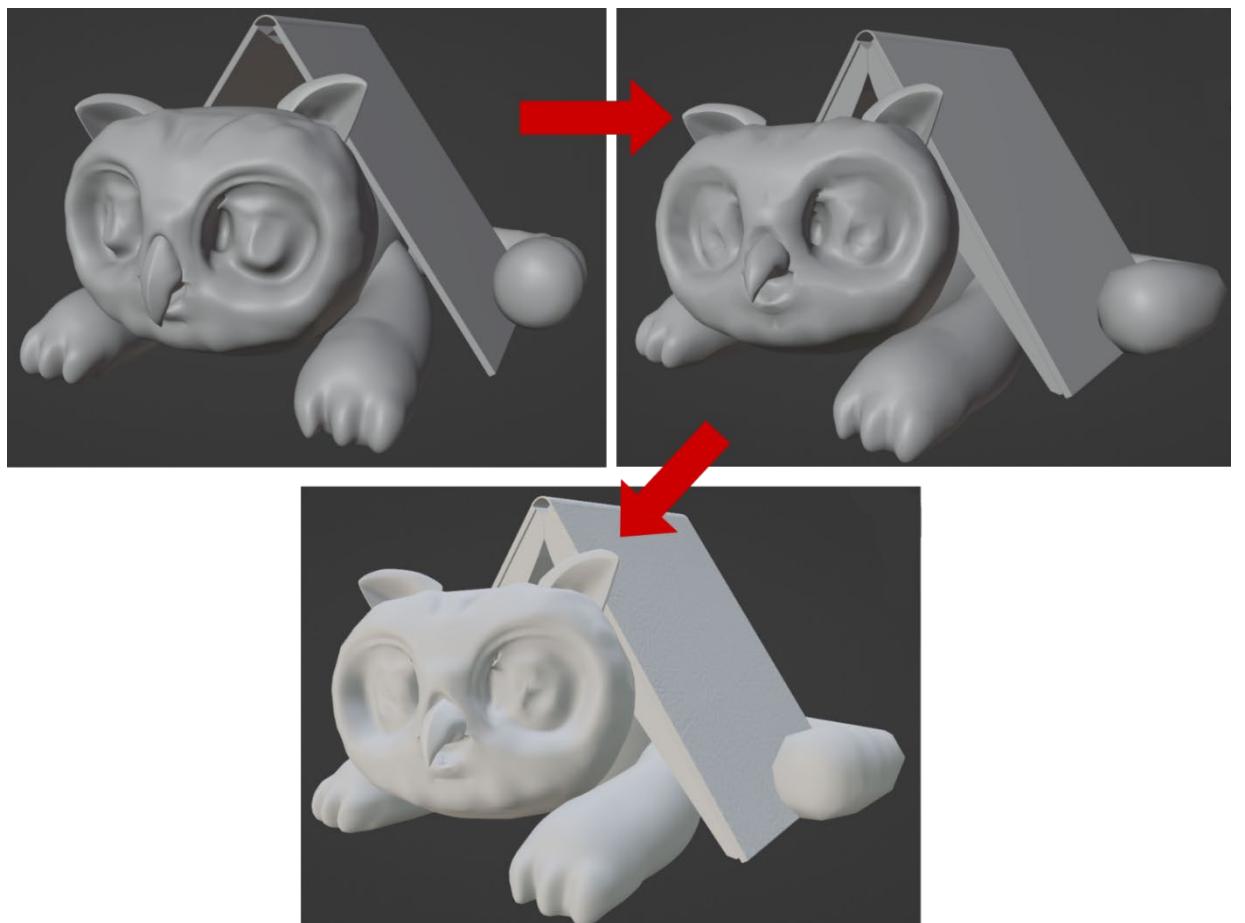


Рисунок 5 – Переход от High-poly модели к Low-poly с “запеченной” картой нормалей

Следующим шагом было создание карты нормалей. Мы запекли детали высоко полигональной модели на текстуру, а затем применили эту карту к Low-poly модели. Это создало впечатление высокой детализации, необходимой для качественного отображения на экране, но с уменьшенным числом полигонов, что оптимизировало производительность модели в реальном времени.

Процесс оптимизации от High-poly к Low-poly в Blender (Рисунок 5) предоставляет эффективный инструмент для создания высококачественных моделей с минимальной нагрузкой на систему, что особенно важно в контексте разработки игр и виртуальной реальности.

Для добавления реалистичной шерсти к модели полукота-полусовы в Blender, мы воспользовались встроенной системой частиц, предоставляющей широкий спектр возможностей для создания объемных и естественных текстур.

В первую очередь, создали систему частиц на Low-poly модели, выбрав "Hair" в качестве типа частиц. Это позволяет создавать объемные волосы, реагирующие на окружающее освещение и движение существа.



Рисунок 6 – Полученное изображение в программе Blender

Система частиц в Blender предоставляет множество параметров для настройки волос. Регулируя длину, плотность, толщину и кручение, мы добивались естественного и разнообразного внешнего вида шерсти.

Экспериментировали с цветом, чтобы соответствовать общей цветовой гамме существа.

Для создания более реалистичного вида шерсти, применили текстуры к частицам. Это позволило добавить волосам различные оттенки и учесть различия в цвете и яркости по всей поверхности.

Система частиц в Blender также предоставляет инструменты для управления направлением и распределением волос. Мы настраивали их, чтобы следовать естественной линии роста волос на существе и создавать желаемую текстурную форму.

Использование системы частиц в Blender позволяет создавать реалистичные визуальные эффекты, такие как шерсть, и при этом обеспечивает высокую степень контроля и настраиваемости. Это является важным шагом в процессе придания модели большей визуальной глубины и живости.

Далее к готовой модели добавляется небольшое количество объектов окружения с текстурами [5] и свет. После начинается процесс рендеринга, чтобы получить качественное изображение (Рисунок 6).

1.3 Импорт в Unreal Engine: связь между Blender и UE4

После завершения создания и настройки модели в Blender, следующим шагом является ее импорт в Unreal Engine (UE4). Этот этап является критическим в процессе интеграции созданных объектов в разрабатываемую игру.

Для эффективного и беспроблемного обмена данными между Blender и Unreal Engine мы использовали формат файла FBX. FBX — это открытый формат Autodesk, который поддерживается большинством 3D приложений, что обеспечивает совместимость и переносимость моделей между разными платформами.

При импорте в Unreal Engine, некоторые аспекты модели могут потребовать дополнительной настройки для правильного отображения в игровом мире. Проблемы могут включать в себя неправильное применение материалов,

смещение текстур, или несоответствие масштаба, а также неправильное отображение анимаций. В процессе разработки были внесены соответствующие изменения в модели, чтобы обеспечить их правильную интерпретацию в Unreal Engine.

Один из основных аспектов, требующих внимания, — это коррекция масштаба модели. Для этого при сохранении модели в Blender нужно выполнять действие “Apply all transforms”, чтобы обеспечить правильное восприятие размеров объектов в игровой сцене. Подобное же применяется и к модификаторам модели, они должны быть применены к модели до ее экспорта.

Еще одним важным аспектом является настройка материалов. В некоторых случаях, особенно при использовании сложных шейдеров в Blender, требуется перенастройка материалов в Unreal Engine для достижения согласованного визуального эффекта. Другие материалы могут просто теряться, и их приходится переприсваивать.

Еще одной важной проблемой является то, что констрайнты (constraints) применяются в Blender для анимации движения по кривой и системы частиц, не могут быть напрямую переданы в Unreal Engine. В этом случае необходим переделать эти части моделей: в анимациях движения нужно сделать дополнительные keyframe'ы для анимации в моделе “Фабрик” (Рисунок 7); для системы частиц второй модели - частицы волос преобразуются в меш и затем в тела, которые корректно (но не так визуально эстетично) отображаются в Unreal Engine (Рисунок 8).

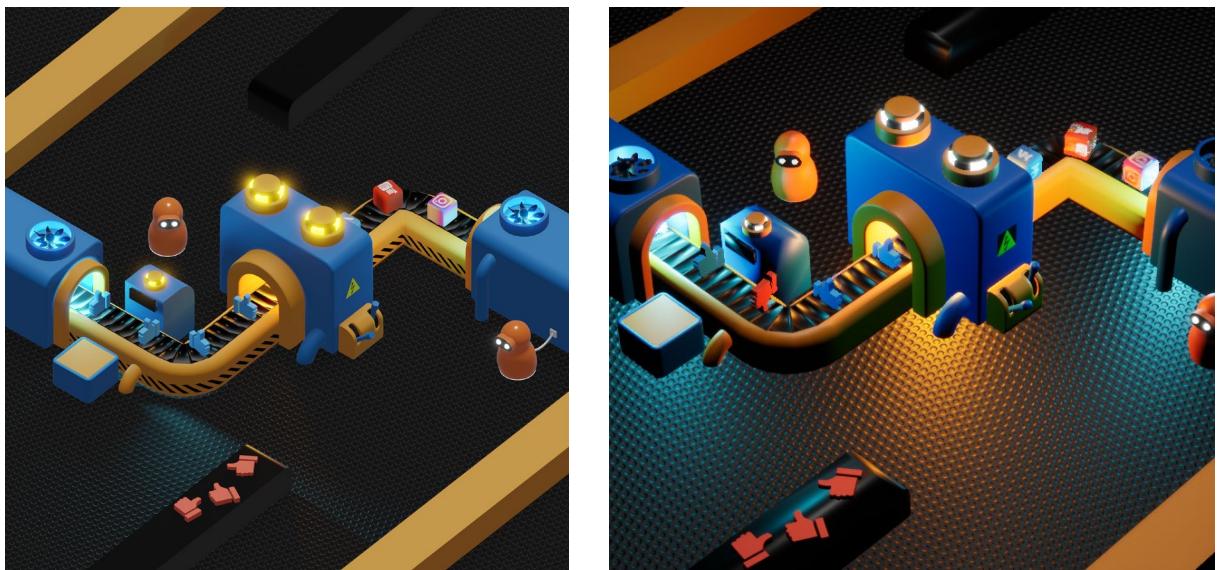


Рисунок 7 – Модель “Фабрика” в Blender (слева) и в UE4 (справа)



Рисунок 8 – Модель “OwlCat” в Blender (слева) и в UE4 (справа)

1.4 Альтернативные средства создания 3D-моделей и анимации

В данной главе будут рассмотрены различные методы, предлагающие альтернативные подходы к созданию трехмерных моделей и анимации. Вместо традиционного метода ручного моделирования и анимации, основанного на манипуляции мешами и ключевыми кадрами, эти подходы используют

сторонние средства и технологии для более быстрого, эффективного и реалистичного создания контента.

Один из таких подходов – это система motion capture, которая позволяет захватывать движения реальных объектов или персонажей и преобразовывать их в анимацию [6]. С помощью специального оборудования, такого как камеры и сенсоры, система motion capture записывает движения и выражения лица, а затем преобразует их в трехмерную анимацию. Этот метод обеспечивает высокую степень реализма и естественности анимации, что делает его особенно полезным для создания персонажей и сцен с большим количеством движения.

Другим альтернативным средством создания 3D-моделей и анимации является сканирование реальных объектов. Этот метод позволяет создавать трехмерные модели на основе реальных предметов или существ, сканируя их с помощью специализированных сканеров или фотографирования в различных ракурсах. Такой подход особенно полезен для создания детализированных и реалистичных моделей, например, при создании архитектурных объектов или персонажей для видеоигр и кино.

Кроме того, одним из последних достижений в области создания реалистичных 3D-персонажей является система Metahuman Creator, разработанная командой Unreal Engine. Эта система предлагает пользователю возможность создавать высококачественные и гиперреалистичные персонажи с помощью набора готовых инструментов и настроек. Благодаря передовым технологиям визуализации и анимации, Metahuman Creator становится все более популярным инструментом среди разработчиков видеоигр, кинематографистов и визуальных художников.

1.4.1 Средства создания моделей: Metahuman Creator

Metahuman Creator, разработанный Epic Games, представляет собой выдающийся инструмент для создания высококачественных персонажей с фотопрералистичной внешностью. Используя этот инструмент, разработчики

могут быстро и эффективно создавать уникальные модели персонажей, настраивая их внешний вид в соответствии с проектными потребностями. Гибкие параметры Metahuman Creator позволяют легко настраивать черты лица, волосы, одежду и другие детали, предоставляя богатый выбор для кастомизации персонажей.

Среди ключевых преимуществ инструмента стоит выделить простоту использования и удивительную детализацию создаваемых персонажей. За считанные минуты можно создать уникального персонажа с высоким уровнем реализма, что делает Metahuman Creator идеальным выбором для различных виртуальных проектов.

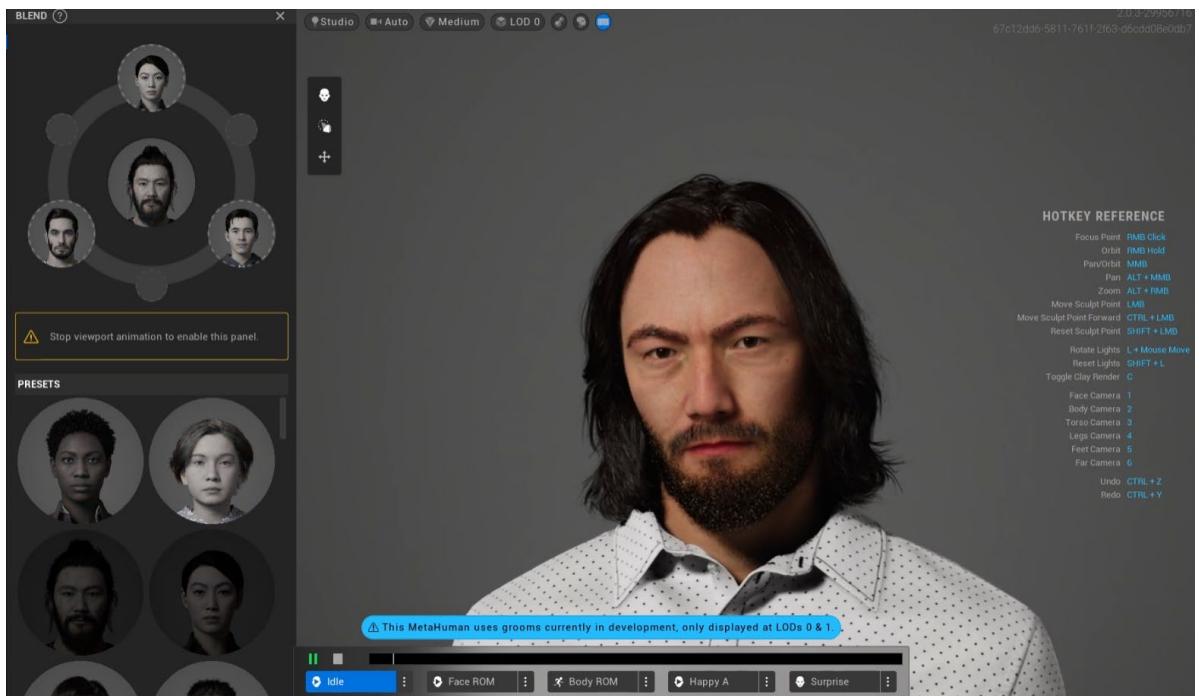


Рисунок 9 – Созданный с помощью Metahuman Creator фотореалистичный персонаж

В рамках исследования было проведено изучение возможностей браузерного редактора и создана уникальная модель Metahuman'a (Рисунок 9). Дополнительно был проведен тестирование плагина Quixel Bridge в рамках Unreal Engine, который позволяет интегрировать объекты из библиотеки и

созданных персонажей Metahuman непосредственно в проект без необходимости скачивания и распаковки архивов.

Однако, создание виртуальных персонажей — это только первый шаг. Существенная часть реализма и выразительности виртуальных персонажей достигается через анимацию движений. В этом контексте, область Motion Capture становится ключевым аспектом. Даже самая красочная и реалистичная модель может показаться недостоверной без убедительной анимации.

1.4.2 Новые возможности создания анимации в Unreal Engine 5.4

Компания Epic Games, в рамках GDC в марте 2024, представила новую версию своего движка – Unreal Engine 5.4 [12]. Релиз самого обновления состоялся в конце апреля 2024 года. Начав как движок для шутеров, сейчас в Unreal Engine есть функционал позволяющий создавать практически всё: анимации, риггинг, модели, текстуры, визуализацию, моушен дизайн. В новом обновлении разработчики сфокусировались на улучшении инструментов анимации и рендеринга.

Ретаргетинг – это перенос анимации, с одного персонажа, на другого, при этом не важны ни структура скелета, ни пропорции. В ранних версиях ретаргетинг был связан со множеством проблем, когда при переносе анимация выглядела неправильно и таким образом портила работу. Теперь же ретаргетинг делается невероятно просто, для этого необходимо нажать все пару кнопок. С новой функцией Auto Generate Retargeter этот процесс стал более простым и менее непредсказуемым в контексте результирующей анимации.

Изменения в Control Rig теперь позволяют начать анимировать, не выходя из UE, достаточно перейти в редактор Control Rig (Рисунок 10).

В данном редакторе создаются контролеры, с помощью которых можно управлять различными частями тела персонажа. В прошлых версиях этот редактор был очень неудобен, и поэтому мало кто им пользовался. В этом же обновлении,

он стал намного дружелюбнее к пользователю. Теперь просто для работы достаточно перенести нужные контроллеры на модель и все начнет работать.

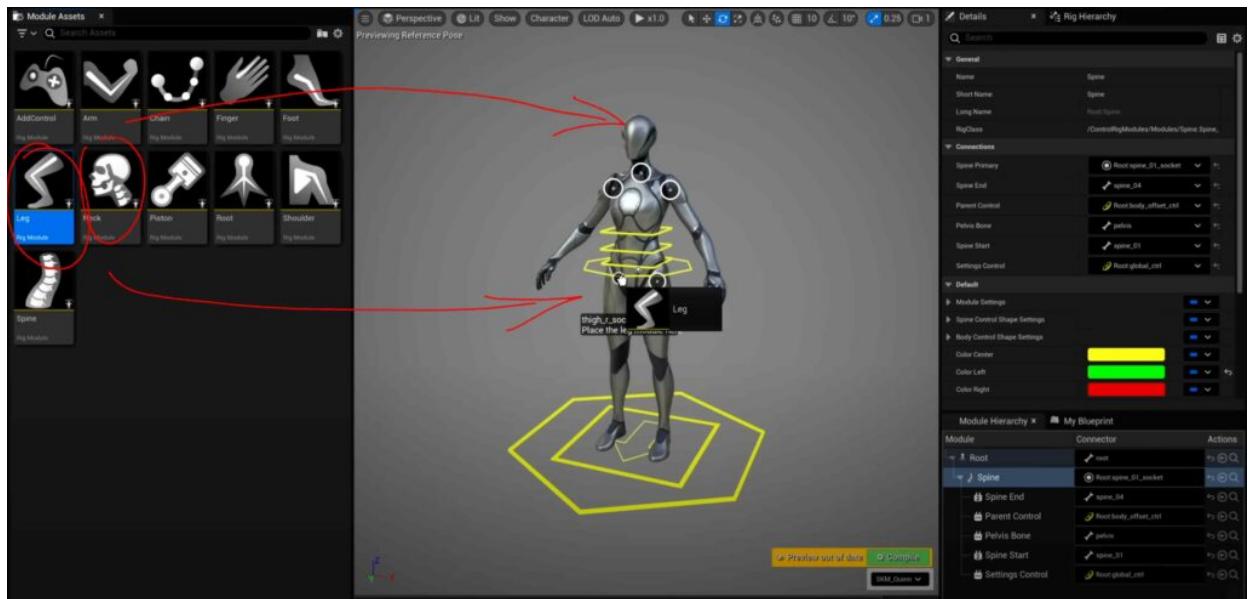


Рисунок 10 – Окно редактора Control Rig

Также использовать Control Rig можно не «запекая» анимацию в отдельные ключи. Для этого можно добавить Control Rig как отдельный слой и изменять анимацию, не используя сторонний софт.

Анимация в Unreal Engine 5.4 дополнилась новой технологией Motion Matching, вместо ручной настройки анимации пользователем, автоматически выбирается подходящая анимация. Прыжки, падения, сложные элементы передвижения – все это берется из базы данных. А для плавных переходов вся траектория просчитывается заранее.

В этом же году, эпик добавят пресеты 500 бесплатных анимаций, AAA-качества (Рисунок 11), созданных на базе высококачественных данных полученных посредством захвата движений, которые ещё будут работать и с технологией Metahuman.



Рисунок 11 – Окно редактора анимации с набором
пресетов из базы данных

Список всех нововведений в Unreal Engine 5.4 для анимации:

- Modular Control Rig. Инструмент для более интуитивной настройки рига отдельных частей тела.
- Automatic Retargeting. Функция для более быстрого перенаправления анимаций для двуногих персонажей.
- Новые функции Skeletal Editor. Упрощают деформацию меша во время движения, например при скручивании туловища. Также можно изменять формы отдельных участков полигональной сетки.
- Обновление интерфейса Sequencer. UI стал более интуитивным, появилась поддержка создания кастомных инструментов анимации для продвинутых пользователей.
- Фреймворк для геймплейной анимации Motion Matching. Система Motion Matching может проанализировать информацию о движении персонажа в игре и подобрать подходящие движения из библиотеки заранее подготовленных анимаций.

- Новая экспериментальная функция Tessellation для технологии Nanite. Добавляет мелкие детали (трещины и неровности) во время рендеринга без изменения исходной полигональной сетки.
- Улучшение стабильности и производительности Temporal Super Resolution. Также добавлены новые режимы визуализации для более точной настройки и отладки этой системы.
- Система нодов Movie Render Graph. Позволяет рендерить отдельные элементы сцены с помощью построения сети нодов, что упрощает рабочий процесс, в который вовлечено много специалистов.
- Улучшение производительности во время рендеринга в целом.

Таким образом Unreal Engine делает очень большие шаги, чтобы стать монополистом на рынке. На сегодняшний день, не выходя из одной программы можно реализовать практически любые задачи от создания игры на любую платформу, до создания кино.

1.4.3 Сравнение программ для motion capture

Процесс захвата движений (Motion Capture) становится ключевым элементом при создании реалистичных анимаций виртуальных персонажей.

Virtual Motion Capture (VMCProtocol, OSC/VMC Protocol) — это протокол передачи данных о движении аватара для виртуального захвата движения, созданный японскими разработчиками. С его помощью можно легко перемещать своего аватара, используя простую в использовании библиотеку, не реализуя обработку устройств VR. Также есть возможность отправлять и получать движения в различных приложениях и из них. Схема работы протокола представлена на рисунке ниже (Рисунок 12), взятом с официального сайта разработчиков [7].

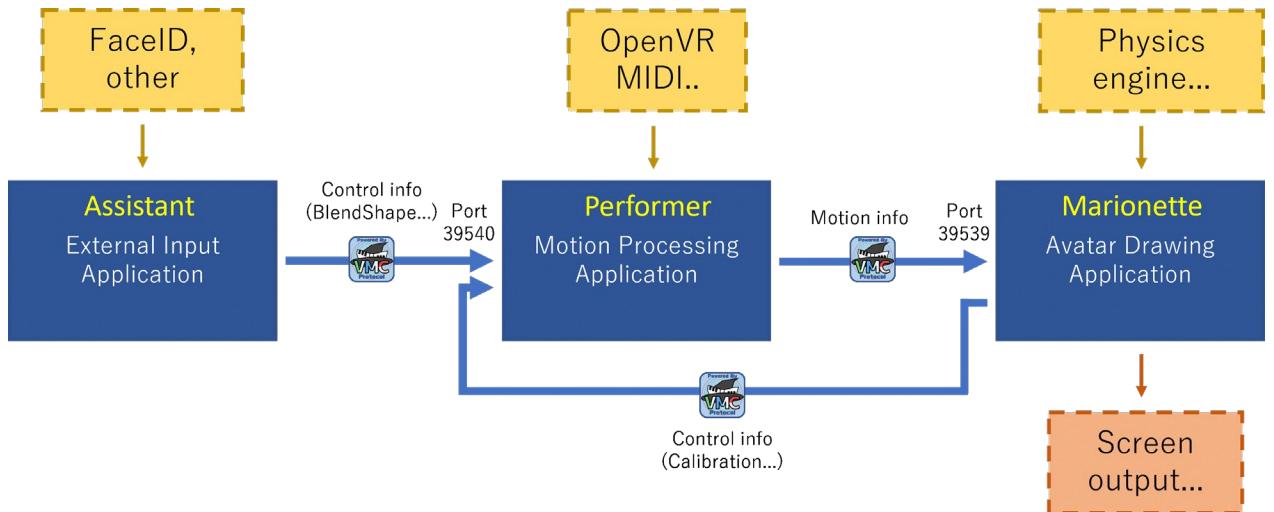


Рисунок 12 – Схема работы протокола VMC

Марионетка (Marionette) — занимается получением движения и отрисовкой его на экране (обязателен). Работает сервер для Performer, обычно использует порт 39539. Например: EVMC4U, VMC4UE.

Исполнитель (Performer) — выполняет процесс считывания и переработки движения. Отправляет все “кости” и дополнительную информацию Марионетке. (обязателен). Работает клиентом для Марионетки, но работает сервером для Ассистента, который является необязательной частью схемы. Например: Virtual Motion Capture, Waidayo, VSeeFace, MocapForAll, TDPT.

Ассистент (Assistant) — отправляет Исполнителю некоторые “кости”, сетку лица и т.д. (необязателен). Он работает как клиент для Performer. Например: Waidayo, Sknuckle, Simple Motion Tracker, Uni-studio.

VMC – это простая реализация с использованием Open Sound Control и VRM, которая может взаимодействовать с различными средами, такими как Windows, Mac, Linux и iOS, во внутренней или локальной сети компьютера.

На сайте разработчика [7] представлен список программ, которые поддерживают протокол и могут отправлять и принимать данные. Основываясь на этом списке, были выбраны несколько программ для исследования: Virtual Motion Capture, TDPT, LuppetX, XR Animation.

В дополнение были взяты программы, также предназначенных для считывания и передачи движений, но не обозначенные в списке, среди них: WebCam Motion Capture, VSeeFace и StrongTrack. Выбор последних программ основывался на публикацию Федора Андреевича Витюкова [6], которая предоставляет ценный обзор этих программных инструментов для захвата движений. В данной же работе упор был сделан на возможность поддержки передачи данных по протоколу VMC и доступности программ рядовому пользователю.

WebCam Motion Capture – это приложение предоставляет отличную возможность захвата движений лица и пальцев рук, используя встроенные вебкамеры. Важным преимуществом является его интуитивно-понятный интерфейс и эффективность в захвате нюансов лицевой анимации. Также программа может повторять движения с записанного видео и с трансляции камеры. Однако, для полной интеграции с VMC протоколом, требуется полная версия приложения, доступная за символическую плату в 2 доллара.

VSeeFace – приложение, ориентированное на создание реалистичных лицевых анимаций и предоставляет богатый набор инструментов для этой цели. VSeeFace легко захватывает экспрессии лица, придавая виртуальным персонажам выразительность и естественность. Для анимации рук потребуется Leap Motion устройство, что может стать ограничением в использовании в некоторых случаях.

StrongTrack не совместим с VMC протоколом. Для его интеграции в UE необходимы сторонние протоколы, поэтому в дальнейшем данное приложение не рассматривается.

Virtual Motion Capture (VMC Protocol) – приложение, разработанное создателями одноименного протокола, предоставляет гибкую платформу для считывания движений. Однако основной упор там делается на VR и для считывания движения нужны соответствующие девайсы, что добавляет сложность и стоимость, но в то же время предоставляет более точные и широкие возможности считывания движений.



Рисунок 13 – Считывание движений с видео в программе XR Animation

TDPT (Two Dimensional Pose Tracker) – приложение специализируется на считывании движений всего тела. Однако оно не включает в себя функциональность для считывания пальцев и мимики лица. TDPT может быть идеальным выбором для общих планов, где не требуется детализированное воспроизведение движений лица. Но для исследуемых задач данное приложение не подходит.

LuppetX – приложение LuppetX предоставляет короткий бесплатный период использования и специализируется на считывании мимики лица, для считывания движений рук потребуются дополнительные устройства.

XR Animation – среди опробованных программ XR Animation выделяется как наиболее функциональное. Это приложение может считывать движения всего тела, включая пальцы и мимику, прямо с камеры. Его способность точно воспроизводить движения делает его эффективным инструментом для создания реалистичной анимации. Данное приложение нацелено для стримеров, но может

считывать движения и с заранее записанного видео (Рисунок 13). Программа может в режиме реального времени передавать движения по VMC протоколу.

Таблица 1 – Сравнение различных программ для захвата движений

| Название программы | Считывание мимики | Считывание пальцев рук | Поддержка VMC protocol | Цена |
|------------------------|----------------------------|------------------------------|------------------------|-------------------------|
| WebCam Motion Capture | Хорошее | Хорошее | +/- | 2\$ для полного доступа |
| VSeeFace | Хорошее | Требует ся доп. оборудование | + | Полностью бесплатна |
| StrongTrack | Хорошее | Приемлемо | - | Полностью бесплатна |
| Virtual Motion Capture | Необходимо VR оборудование | Необходимо VR оборудование | + | Полностью бесплатна |
| TDPT | нет | нет | + | Полностью бесплатна |
| Luppet X | Посредственно | Требует ся доп. оборудование | + | Free Trial на час |
| XR Animation | Хорошее | Хорошее | + | Полностью бесплатна |

Для наглядности, результаты сравнения 8-ми программ представлены в виде таблицы (Таблица 1). Из данной таблицы можно видеть, что для дальнейшего использования программы захвата движения в связке с Unreal Engine, и с учетом того, что программа должна быть бесплатной и обладать возможностью считывать как мимики лица, так и движения пальцев рук; подходящей можно считать лишь одну программу – XR Animation. В дальнейшем развитии работы планируется применять именно эту программу.

Все программы тестировались на компьютере (PC) с использованием камеры смартфона на Android через DroidCam. Это обеспечивает удобство использования и мобильность, делая процесс считывания движений более гибким и доступным.

Отдельно стоит выделить недавно появившиеся, но активно развивающиеся технологии искусственного интеллекта и машинного обучения в данной области. Например, MediaPipe4U [8] предоставляет набор библиотек и инструментов для быстрого применения методов искусственного интеллекта и машинного обучения в проекте Unreal Engine. Данный инструмент позволяет захватывать движения лица и всего тела прямо с камеры смартфона и передать их сразу в проект Unreal Engine. Помимо захвата движений данный инструмент может распознавать текст, преобразовывать речь и многое другое. Все функции работают как в режиме реального времени, так и в автономном режиме, с низкой задержкой и достаточно просты в использовании.

2 Создание видеоигровой боевой системы

Среди видеоигр распространен жанр Action RPG, это трехмерные игры, где игрок управляет лишь одним персонажем с видом от первого или третьего лица, когда основной упор в игре сделан на бои в ближнем бою с использованием различного вооружения. Для подобных игр характерна возможность передвижения персонажа по поверхности карты и свободный обзор камеры. В контексте боев могут быть несколько типов атак с разной скоростью, уроном и анимацией; также и защитные действия также могут быть различными: блокирование оружием или щитом, уклонения или перекаты.

Для учебной разработки был выбран тип разрабатываемой игры от третьего лица и с определенным набором действий для персонажей: ходьба, бег, перекаты, удар, блок. Была разработана программа на C++ для Unreal Engine 4.2, которая управляет действиями персонажей, считывает нажатие определенных клавиш игроком и создает неплохой прототип боевой системы, способный в дальнейшем вырасти до уровня неплохой трехмерной игры.

Перед непосредственной разработкой был проведен анализ уже существующих аналогов в жанре action RPG от разных разработчиков.

2.1 Обзор существующих боевых систем

Обзор проводился для двух игр класса action RPG: The Witcher III: Wild Hunt от польского разработчика CD Project Red и Elden Ring от японской компании FromSoftware.

На рисунках рисунок 15-рисунок 16 и рисунок 18-рисунок 19 показаны примерные диаграммы состояний, в которых может находиться как противник, так и игровой персонаж. Так как доступ к исходному коду рассматриваемых игр не доступен, то диаграммы были сделаны только на основе умозрительного анализа игрового процесса. Однако даже на основе визуального анализа легко выделить основные состояния и возможные переходы, и так как некоторые из

них невозможны в отдельных случаях (например, некоторые типы противников не способны атаковать издалека, а атаки других не могут быть заблокированы), то такие переходы отображаются прерывистой линией на диаграммах. По той же причине, что отсутствует доступ к исходному коду, нельзя точно установить условия перехода между состояниями, поэтому в схемах переходы остаются без подписей.

The Witcher III: Wild Hunt вышла в 2014 году и в первую очередь знаменита своим сюжетом и проработкой истории и персонажей. Однако продвижение по сюжету не обходится без сражений и путешествий, которые занимают весомую часть игрового процесса.

Бои в *The Witcher* – динамичные и эффектные, и не требуют особых навыков от игрока (Рисунок 14). Главной проблемой продвижения далее по игре может служить только уровневая система в игре, когда от уровня противника зависит количество урона, который он наносит и количество его здоровья, что приводит к практически невозможному бою, при сильно меньшем уроне героя в сравнении с противником.

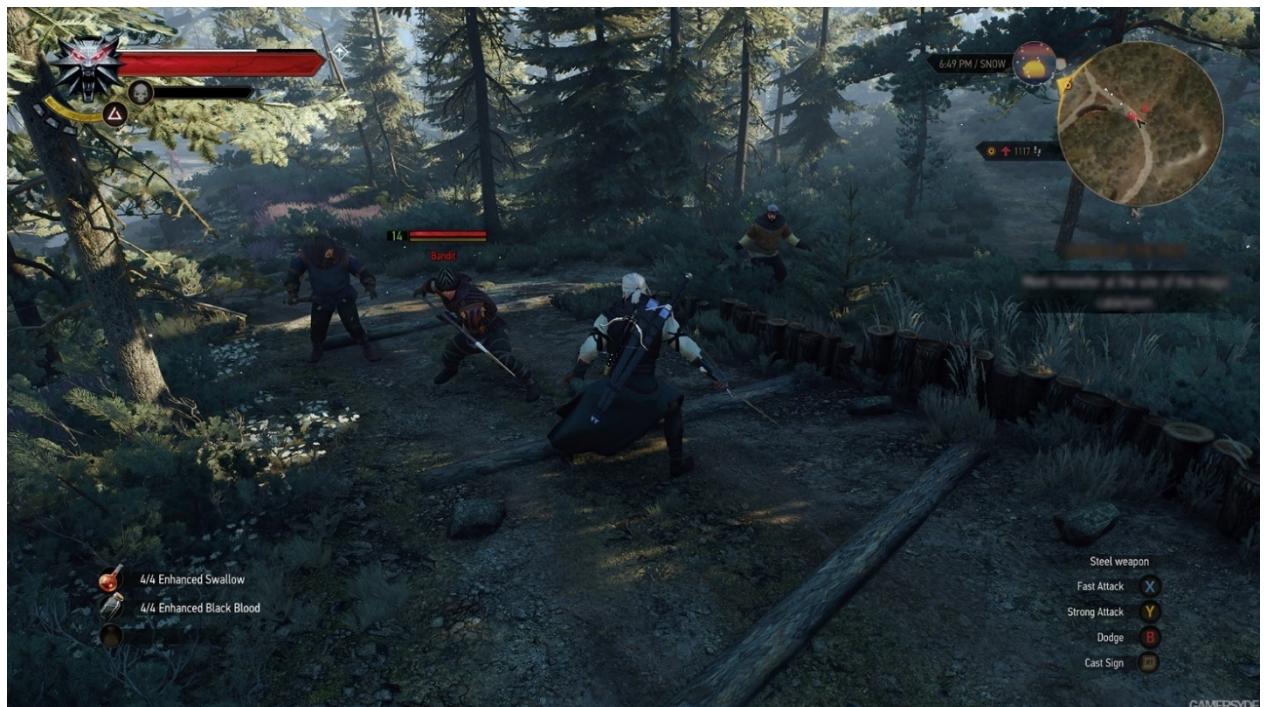


Рисунок 14 – Скриншот боя из игры *The Witcher III: Wild Hunt*

Хотя в игре присутствует большое количество разнообразного оружия, но движения персонажа практически не зависят от того, чем он вооружен (за исключением малого количества особых инструментов), что делает бои, в некоторый момент игры, довольно однообразными. Смена оружия обусловлена только тем, что некоторые противники более восприимчивы к оружию определенного типа: монстров нужно бить серебряным оружием, а людей – стальным.

Также в игре присутствует элемент развития, где с каждым уровнем можно получить какой-то навык или способность, будь то большее количество здоровья, новый удар или сопротивление некоторому виду урона. Таким образом игрок может помимо обычных ударов и блока изучить “заряженный” удар, для которого нужно удерживать кнопку удара; длинную серию ударов или же может получить способность контратаковать из блока, если нажмет кнопку блока в последний момент перед атакой.

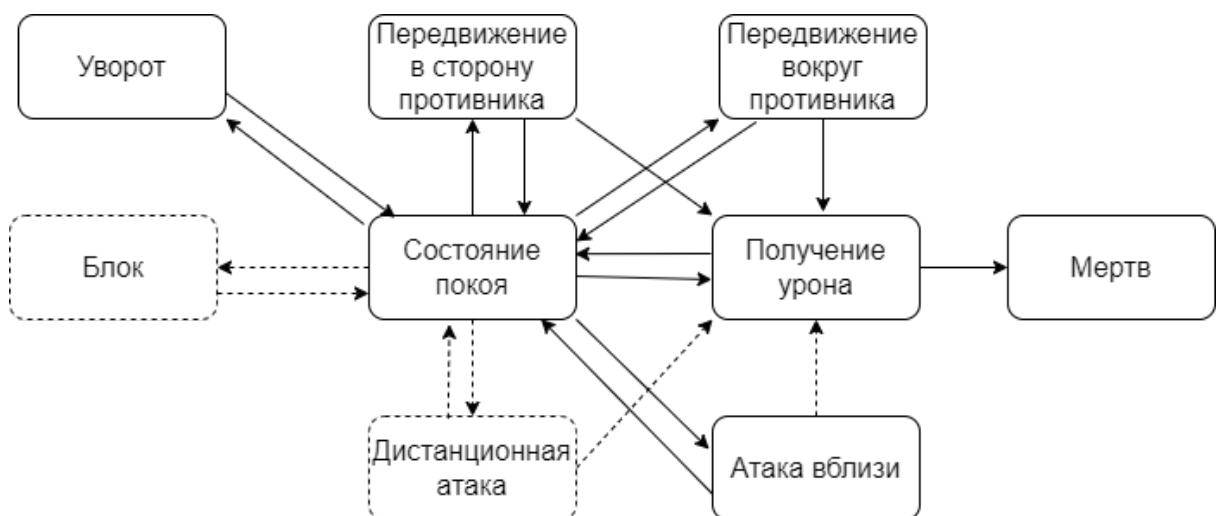


Рисунок 15 – Упрощенная диаграмма возможных состояний игрового персонажа в игре The Witcher III

Помимо сражений на мечах в игре присутствует система магии: пять “глифов”, которыми можно как атаковать, так и обороняться. Применение этих способностей тратит особую характеристику персонажа – выносливость, которая

после применения магии будет медленно восстанавливаться даже в бою. По мере развития, эти способности приобретают новые свойства, становясь более эффективным средством в арсенале игрока, позволяя атаковать сразу группу противников или не получать урон, даже не находясь в блоке.

Также в игре присутствует система алхимии, игрок может приготовить из собранных в мире материалов особые зелья и увеличить некоторые характеристики, восстановить здоровье или выносливость или же получить способность видеть в темноте. Приготовив зелье один раз, восстановить его количество после применения можно с помощью медитации и траты определенного материала.

Бои в игре нельзя назвать достаточно реалистичными или сложными (упрощенная диаграмма персонажа состояний представлена на рисунок 15). В момент, когда игрок нажимает кнопку атаки, вокруг героя возникает сфера, попав в нее, противник автоматически получает урон, независимо от того, стоял ли он за спиной игрока или прямо перед ним. Также эти атаки плохо контролируются и не могут быть остановлены по необходимости, так, если была начата серия атак, то при необходимости уклониться от дистанционной атаки, сделать это нет возможности, и герой точно получит урон.

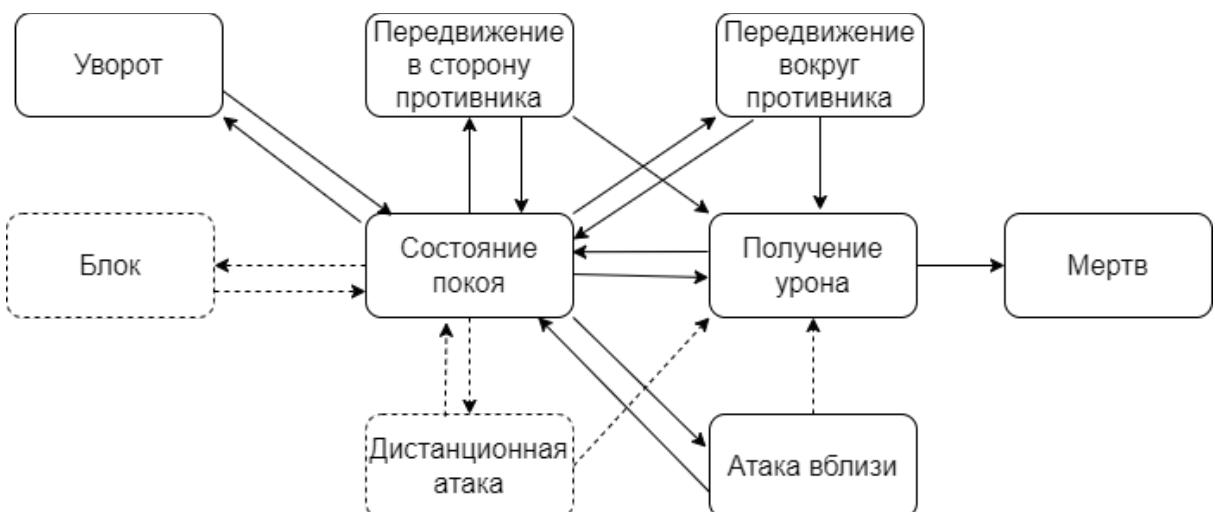


Рисунок 16 – Упрощенная диаграмма состояний возможного поведения противника в игре The Witcher III

Противники в игре представлены в виде обычных воинов с мечами, копьями или арбалетами и монстрами, вид и способности которых могут быть совершенно разнообразны. Одни монстры могут становиться невидимыми, вторые способны летать, а третьи же просто настолько большие, что бить можно только по их конечностям. Все противники наделены неплохим ИИ (упрощенная диаграмма противника представлена на рисунок 16), что позволяет им приспосабливаться к манере боя игрока, заставляя его в свою очередь действовать более изобретательно. Противники могут применять окружение, прятаться за укрытиями в случае применения магии или дистанционных атак, могут пытаться заходить за спину игрока; если нападают большим числом, то нападают сообща.

Elden Ring вышла в 2022 году и является представителем особого жанра игр, которые предлагают игроку сложные бои, требующие невероятной концентрации (Рисунок 17), в игре часто необходимо заучивать последовательность ударов противников и находить их уязвимые места.



Рисунок 17 – Скриншот боя из игры Elden Ring

Бои в игре могут проходить очень разнообразно (упрощенная диаграмма состояний персонажа представлена на рисунок 18), все зависит от того стиля боя,

который применяет игрок. Это могут быть быстрые атаки и уклонения от ответных атак, или же выверенные сильные удары, после заблокированной серии атак противника, когда он наиболее уязвим. Игра позволяет экспериментировать со стилем боя, хотя не все из них могут быть в одинаковой степени эффективны.

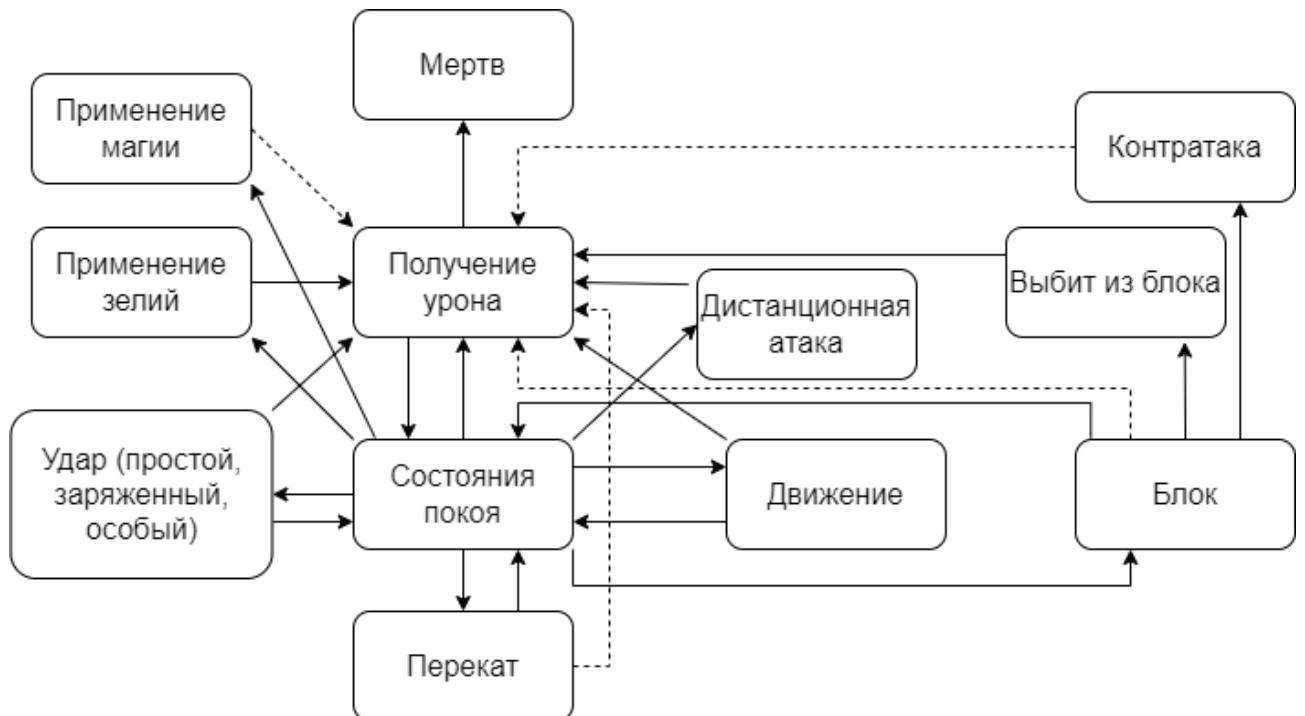


Рисунок 18 – Упрощенная диаграмма состояний игрового персонажа в игре Elden Ring

В игре есть различные классы, зависящие от наиболее развитых характеристик, основанные на силе, стойкости, ловкости, магии или вере. Каждая характеристика влияет не только на самого персонажа, но и на его оружие и даже тип урона, который он будет наносить, а это уже может быть полезно против определенных противников, так как они могут иметь сильное сопротивление определенному типу урона.

Как и классов, в игре присутствует огромное количество различного вооружения: мечи (короткие, средние, двуручные и просто гигантские), ножи, копья, цепы, когти, дубины, луки, арбалеты, магические посохи. При этом каждое оружие имеет три типа атаки: слабый удар, сильный удар (который

можно дополнительно зарядить) и специальный прием, который можно изменить и наделить особыми свойствами. Помимо этого, герой может держать оружие как в одной, так и в двух руках, отчего изменяется сила удара и свойства блока. Держа основное оружие в одной руке, вторая может быть занята дальнобойным оружием (например арбалетом или магическим посохом) или щитом, блокировкой которым дает защитные показатели намного выше, чем блок оружием в двух руках.

Магические способности игрок может получить, развивая соответствующие характеристики, а определенные заклинания может изучить, найдя свитки где-то в мире игры. На применение магии тратится особая характеристика героя – мана, восстановить ее можно только использовав специальное зелье, просто так она не восстановиться.

Другой характеристикой, важной в бою, является выносливость, она, в отличие от маны, автоматически восполняется в бою. Она тратиться на перекаты, атаки и блоки. Если она закончится, то герой не сможет больше действовать и станет уязвим для атак противника. Также если выносливости мало, то противник сможет даже пробить поставленный блок, нанеся сильные повреждения.

Сами бои в игре очень часто напряженные и сложные, пропустив один сильный удар, можно сразу погибнуть (упрощенная диаграмма состояний противника представлена на рисунок 19). Помимо обычных противников (рыцарей, зверей или мелких монстров), существуют так называемые боссы, охраняющие особые сокровища или преграждающие путь далее по сюжету. Сражения с ними – наиболее сложные, но и самые интересные в игре. Один раз поняв, как нужно действовать с этим противником, какие удары он может применять и, главное, как от них защищаться, в следующий раз будет намного проще сражаться с ним или ему подобными. У каждого босса есть особый список атак, которые он может применять в бою. Эти атаки могут начинаться случайно, но чаще всего они зависят как от количества здоровья, которое осталось у босса (чем его меньше, тем агрессивнее становятся атаки), так и от действий игрока,

босс легко может приспособиться к стилю игрока и одни и те же атаки будут в какой-то момент бесполезны против него.

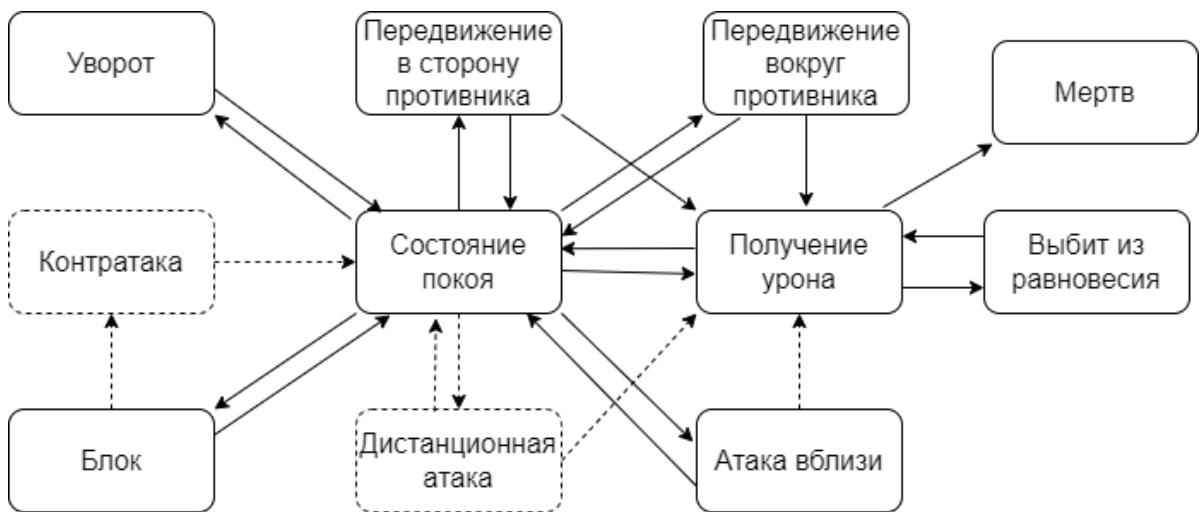


Рисунок 19 – Упрощенная диаграмма возможных состояний противника в игре Elden Ring

Это делает игру, с одной стороны, очень сложной в освоении, но, с другой стороны, невероятно интересной, если понять, как в нее правильно играть.

2.2 Создание анимации и логики боевой системы

Поведение и игрового персонажа, и противника можно рассматривать в контексте конечных автоматов. Каждое его состояние – это какое-либо действие и связанная с ним анимация, связи – переходы между состояниями по определенным условиями. Входным сигналом для такого конечного автомата может быть как нажатие непосредственным игроком клавиши на клавиатуре или мыши, так и изменение состояния оппонента. В некоторых случаях переходы являются случайными (переходы в некоторые состояния противника), и характеризуются только вероятностью перейти в данное состояние в каждый тик. Общая логика переключений состояний персонажа и его противника в контексте конечных автоматов приведены на рисунок 24 и иллюстрация 31 соответственно.

Конечные автоматы также применяются и в самом Unreal Engine – особые Blueprint'ы для анимации имеют аналогичную структуру и используют в своей основе правила конечных автоматов, чтобы переключаться между различными анимациями. Находясь в одном состоянии, анимация зацикливается. Но если условие перехода выполняется, то начинает проигрываться совершенно другая анимация. Такой конечный автомат для анимации можно связать с состояниями самого объекта, его переменными, тогда каждое действие, которое способен сделать персонаж или его противник, будут сопровождаться соответствующей анимацией.

2.2.1 Анимации боя

Для реализации анимации в Unreal Engine были использованы анимации и Animation Blueprint (AB), отвечающий за логику анимации [9]. Для AB характерны две составляющие: EventGraph и AnimGraph. EventGraph отвечает за изменения самого AB, вычисляет локальные переменные. AnimGraph необходим для построения логики анимации, характеризуется состояниями, которые могут быть как просто анимациями так и сложными blend-анимациями; и связями, на которые накладываются некие условия. На рисунках Рисунок 20, Рисунок 21, Рисунок 22 приведены EventGraph и AnimGraph для персонажа и противника. Event Graph будет идентичен для обоих персонажей, отличие лишь в том, к какому типу будет приведена переменная character, представляющая собой ссылку на другого персонажа в Blueprint (Рисунок 20).

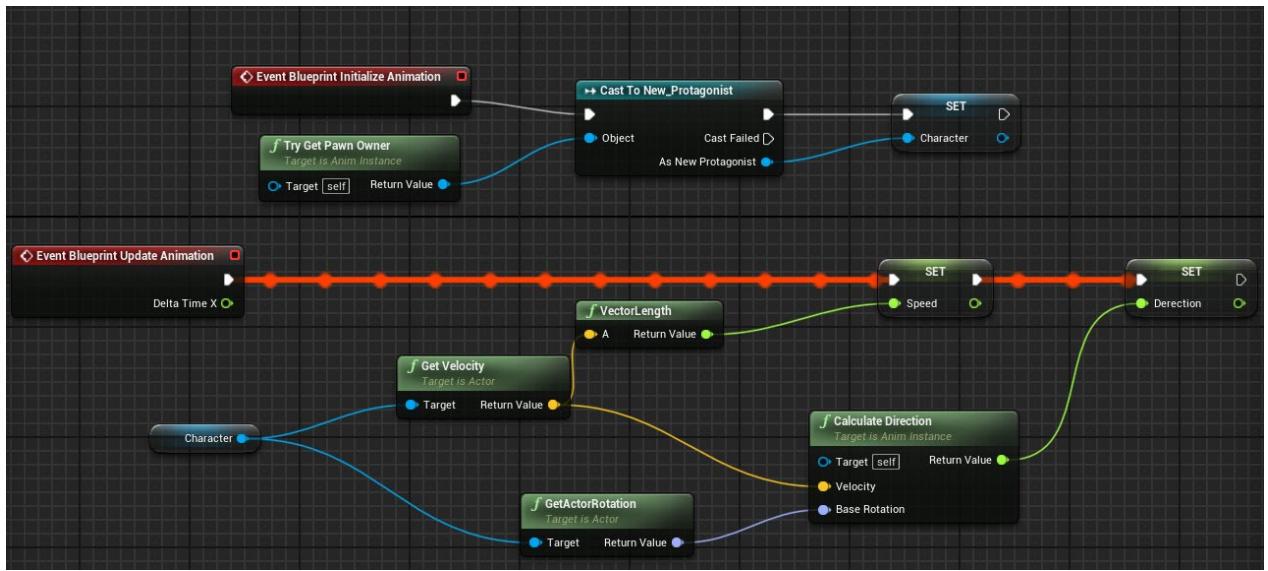


Рисунок 20 – EventGraph игрового персонажа

AnimGraph игрока и его противника будут слегка отличаться, это связано с различной логикой их поведения, и разным набором действий, которые они могут выполнять. Данные графы представлены на рисунке Рисунок 21 и Рисунок 22, как видно количество и сложность переключения между анимациями игрового персонажа больше, так как и набор его действий шире, и они зависят от действий реального игрока.

Для движения персонажа в различных направлениях необходимо считывание его скорости и разворота по оси Z из родительского класса, передаваемого в EventGraph. Далее эти параметры используются в анимациях внутри AnimGraph. Также в AnimGraph используются созданные в родительском классе булевые переменные, отвечающие за состояния (например состояние бега или состояние получение урона), поэтому они применяются в условиях связей графа, для переключения анимации (Рисунок 23).

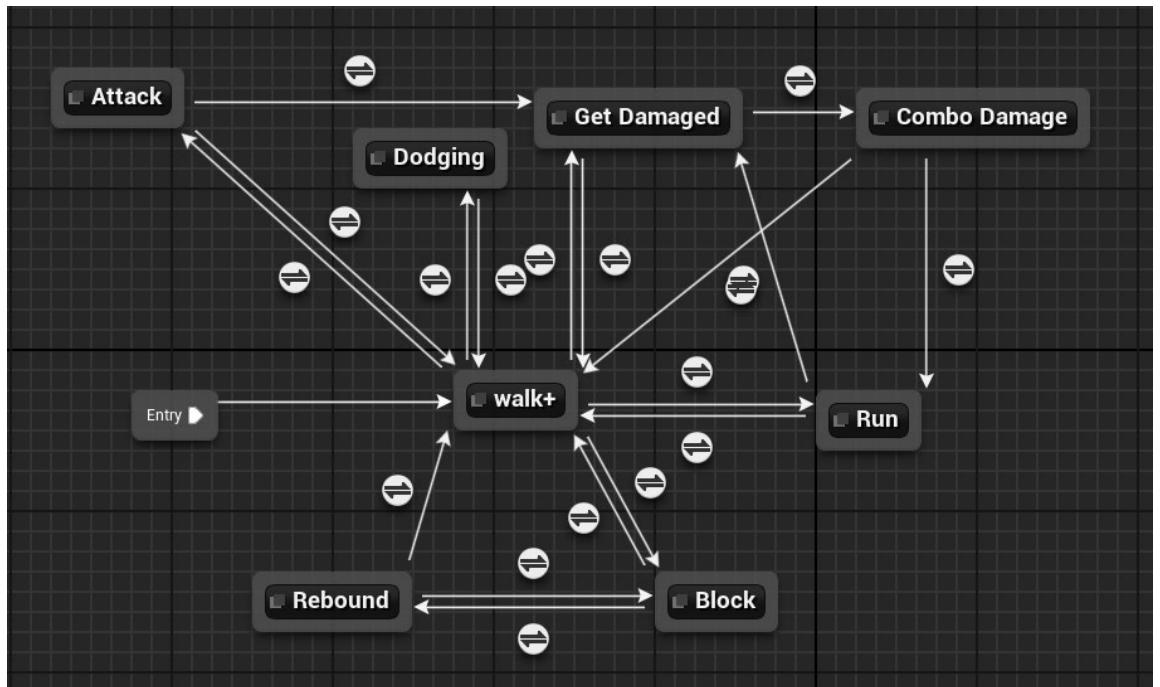


Рисунок 21 – AnimGraph для игрового персонажа

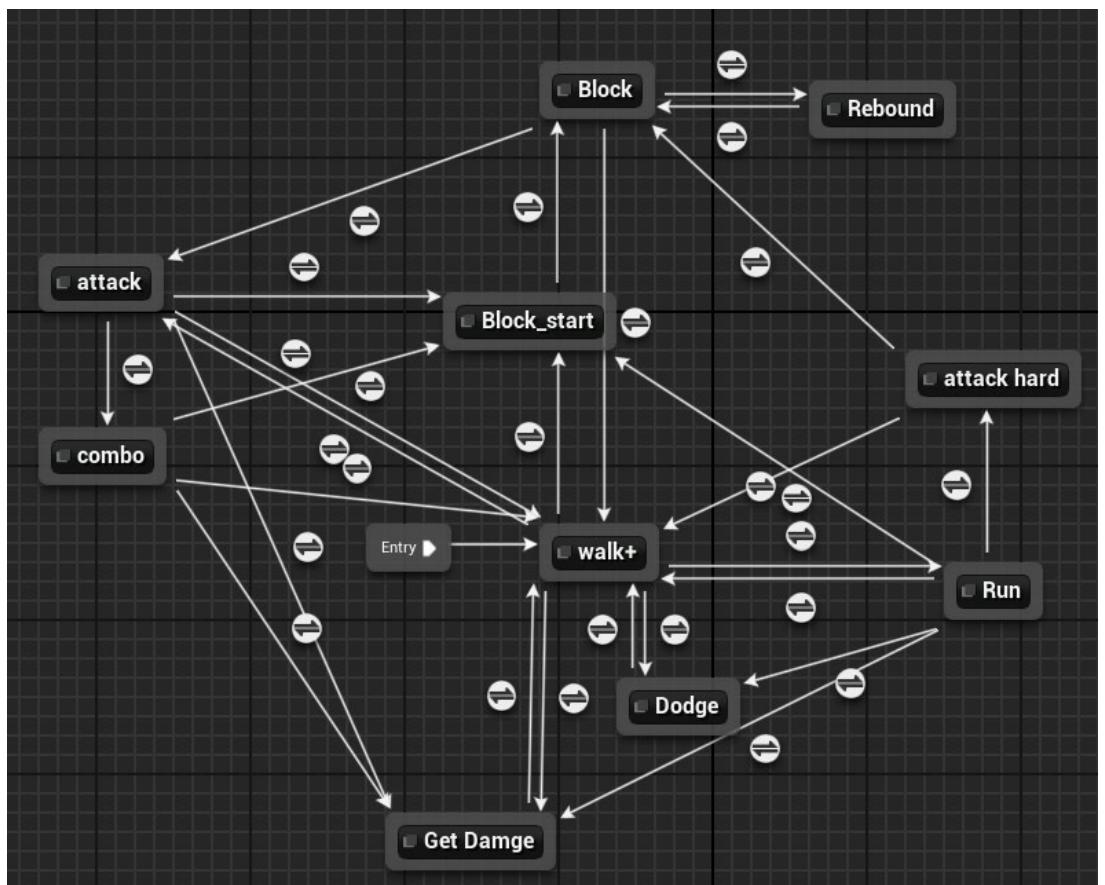


Рисунок 22 – AnimGraph для противника игрока

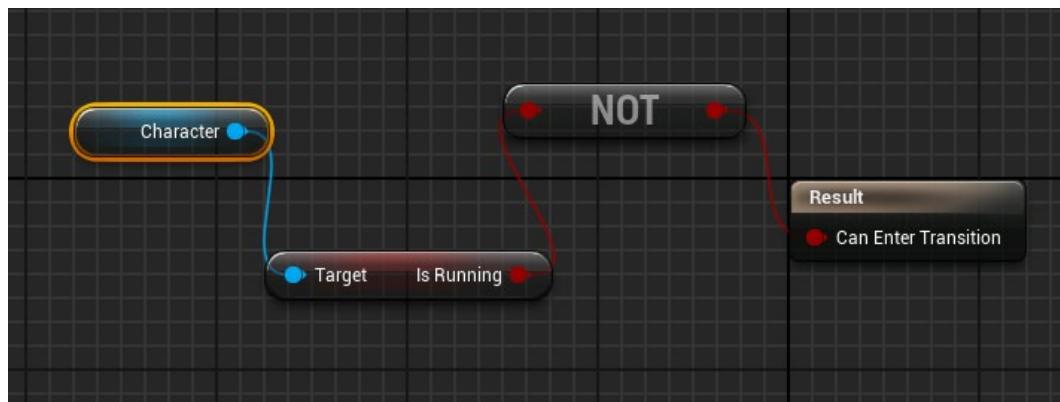


Рисунок 23 – Пример условия для перехода из состояния бега в исходное состояние ходьбы в AnimGraph игрового персонажа

Полная диаграмма состояний игрового персонажа представлена на рисунке Рисунок 24. Все действия начинаются с “исходного состояния”, то есть состояния покоя. Далее любой переход (кроме получения урона) связан с действиями реального игрока, который нажимает те или иные клавиши. Выход из состояния – это конец соответствующей анимации, кроме режимов движения, где переход связан с отпусканием клавиши.

Атакующие действия всегда связаны с нажатием левой клавиши мыши. Если ее нажать несколько раз подряд, то начнется комбо, с более длительной анимацией. А если клавиша нажата во время бега, то пройдет уникальная анимация удара, в которой герой еще и неуязвим для атак.

Во время перекатов и блока герой также неуязвим. Поэтому нет перехода в состояние получения урона.

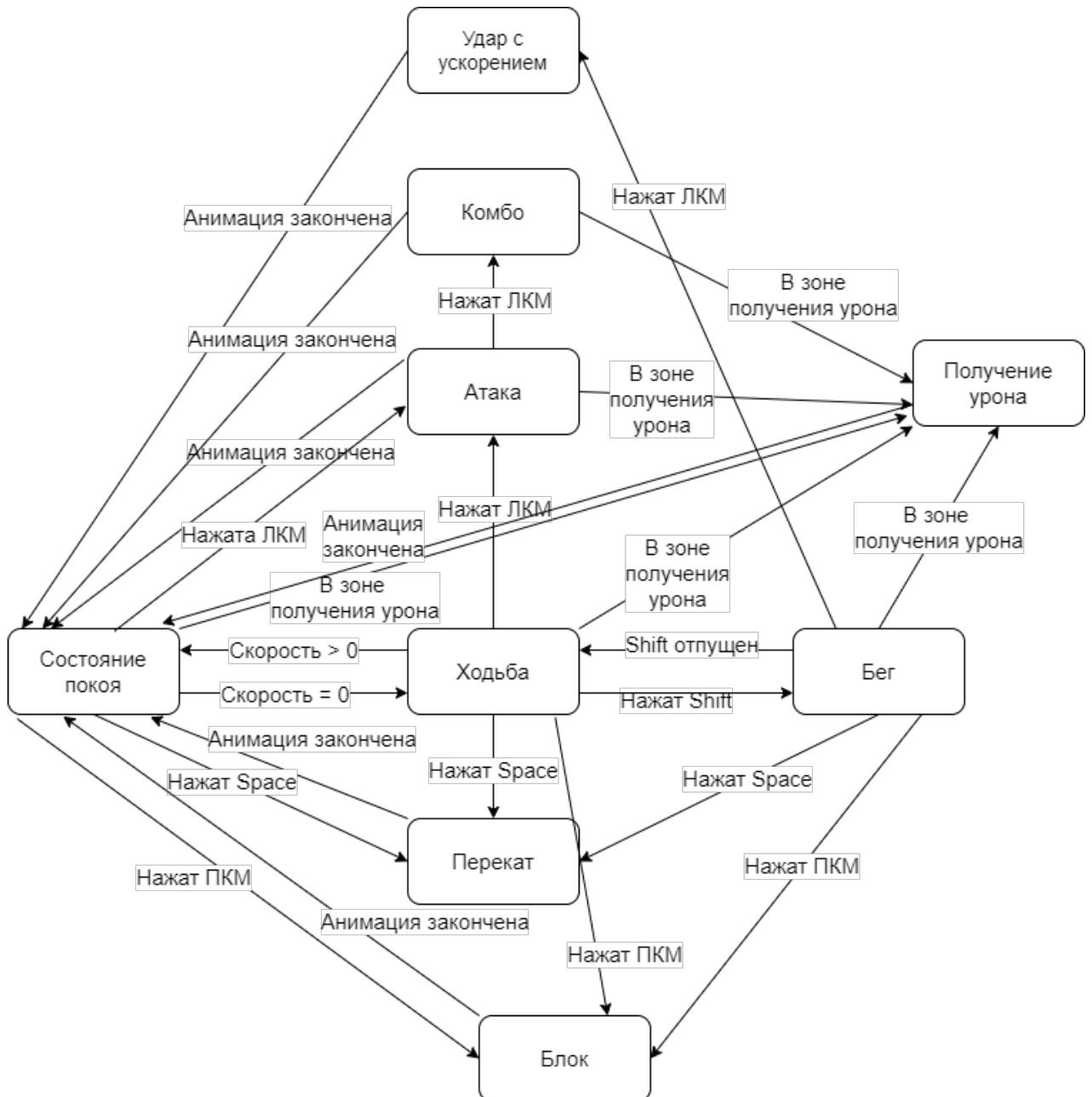


Рисунок 24 – Диаграмма состояний персонажа под управлением игрока

2.2.2 Движение: ходьба, бег, перекаты

Для передвижения персонажа существует всего три состояния: медленная ходьба, бег и перекат/кувырок, для быстрого перемещения вперед.

Для передвижения в различные стороны персонажем под управлением игрока, происходит захват нажатия клавиш WASD, отвечающих за

соответствующие стороны. Удержание данных клавиш приводит к постоянному движению персонажа в соответствующую сторону, в зависимости от того, куда смотрит камера. Как только клавиша перестает быть нажатой, персонаж перестает двигаться в заданном направлении, при условии того, что ни одна другая клавиша WASD не нажата; и переходит в исходное состояние. В программном коде, для того чтобы изменять положение персонажа в пространстве с постоянной скоростью применяется функция Add Movement Input (на рисунке Рисунок 25 представлен функциональный блок в Blueprint), которая принимает на вход направление и скорость. Тогда каждый такт, если нажата соответствующая клавиша направления персонаж будет перемещаться. При этом будет отображаться анимация ходьбы, зависящая от направления движения и от скорости.

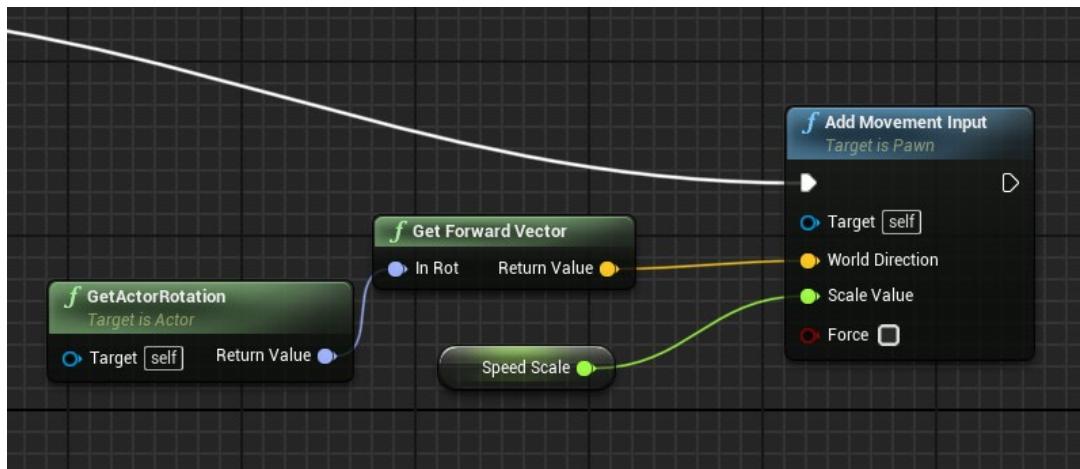


Рисунок 25 – Функциональный блок Blueprint для передвижения персонажа

Если же при уже нажатой клавиши из списка WASD дополнительно удерживать клавишу Shift, то персонаж из медленного движения переходит на бег с большой скоростью, булева переменная Is_running в родительском классе, отвечающая за индикацию переключения в состояние бега, станет “истиной”, а переменная скорости возрастет. Простое нажатие клавиши Shift, без указания направления движения с помощью WASD, не приведет к изменению состояния

персонажа. На рисунке Рисунок 26 изображен аналог алгоритма в виде графа событий Blueprint.

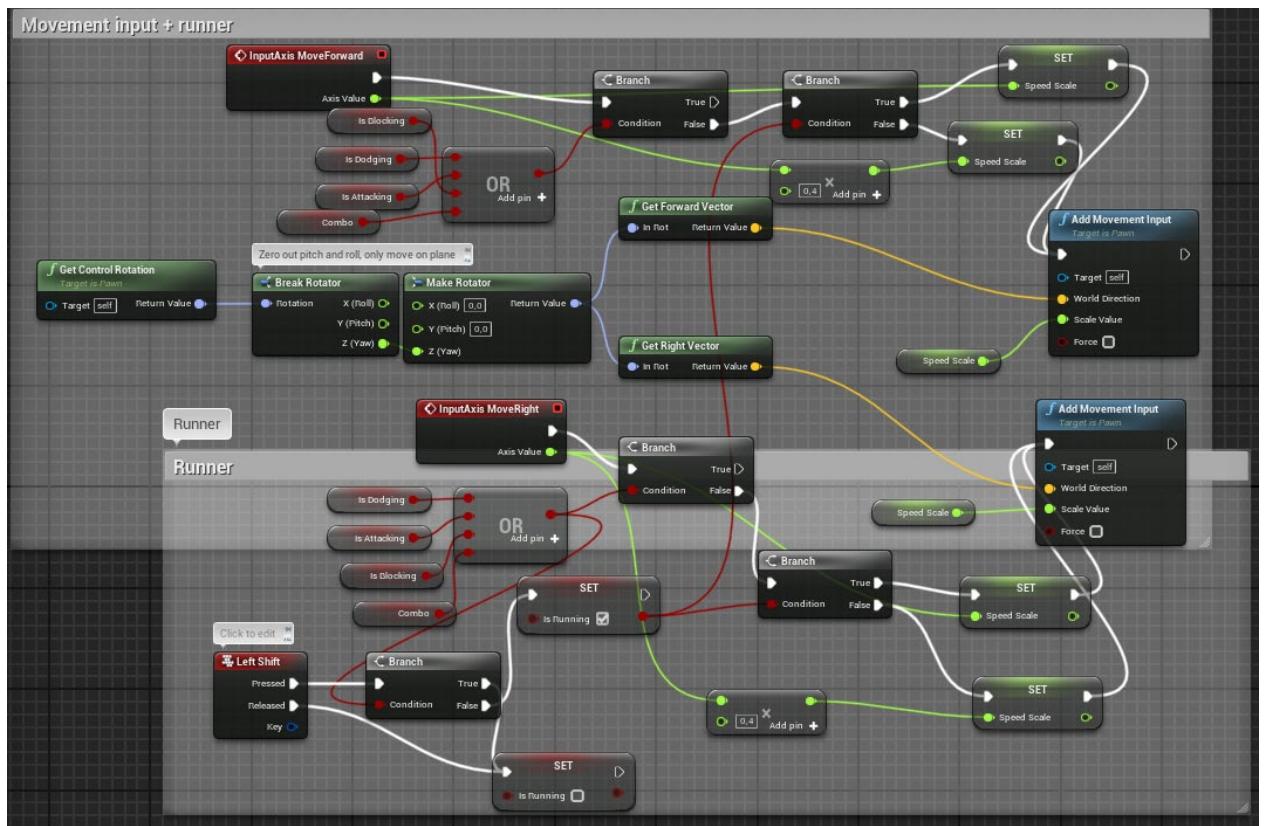


Рисунок 26 – EventGraph реализующий аналог алгоритма движения персонажа: ходьба и бег

Для использования переката, используется клавиша Space (на рисунке Рисунок 27 представлен аналог алгоритма в виде графа событий Blueprint). Направление переката зависит от поворота персонажа по оси Z (получается путем применения функции `getActorRotation()`). После нажатия персонаж перемещается на определенное длительностью анимации расстояние в направление его поворота. В процессе выполнения кувырка изменить его направление нельзя, как и невозможны никакие воздействия на персонажа, пока он не выйдет из этого состояния. Это происходит потому, что как только была нажата клавиша Space, булева переменная `Is_dodging` становится “истиной” и активируется часть кода внутри функции `Tick`, которая отвечает за прямолинейное движение персонажа в течении определенного числа тактов пока

идет анимация переката; остальная же часть кода не активируется т.к. стоит проверка на “ложность” данной переменной. Далее переменная снова становится “ложью”, и персонаж снова способен перейти в другое состояние, т.к. условие с данной переменной перестает выполняться, и соответствующая часть кода снова становится активной.

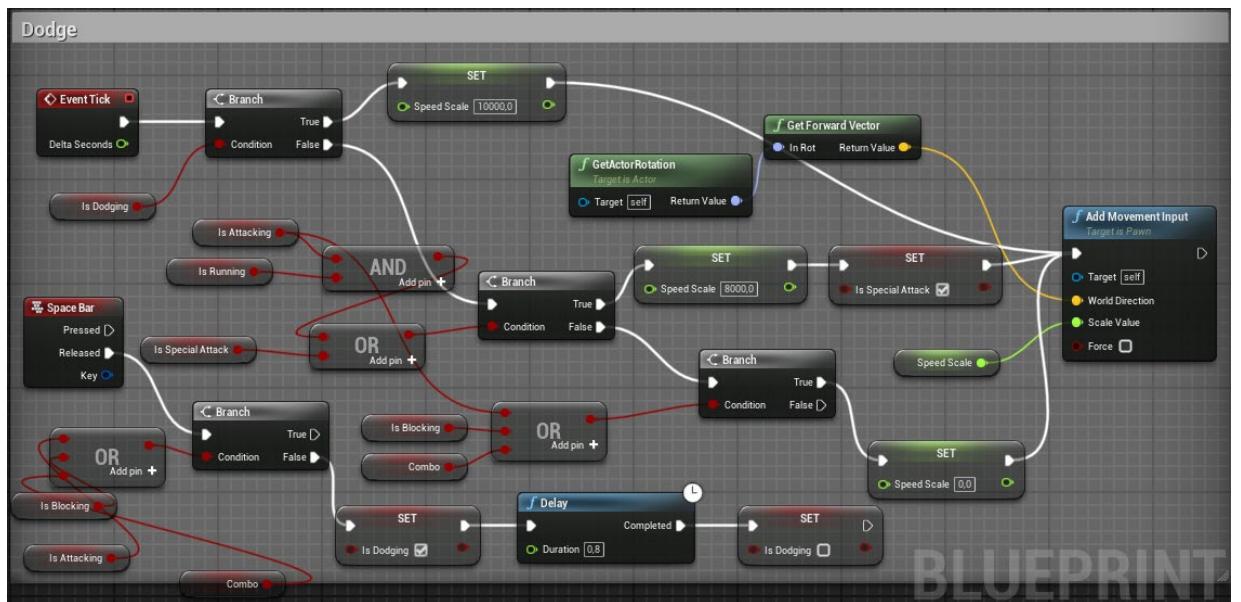


Рисунок 27 – EventGraph реализующий аналог алгоритма переката

2.2.3 Бой: удар, блок, получение урона

Боевая система невозможна без основного элемента: системы нанесения и приема ударов.

Для нанесения удара используется левая клавиша мыши (весь алгоритм нанесения ударов представлен в виде аналогичного графа событий Blueprint на рисунке Рисунок 28). Далее начинается анимация удара, в процессе удара никакие перемещения или иные действия невозможны. Начало удара характеризуется переключением переменной `Is_attacking` в “истину”, однако момент возможного получения удара противником задается другой переменной – `Make_damage`, данную переменную может применять противник для расчета того, попал ли персонаж по нему или нет (См. раздел 2.3.2). После завершения

анимации удара персонаж снова переходит в исходное состояние, открывая возможность для взаимодействия.

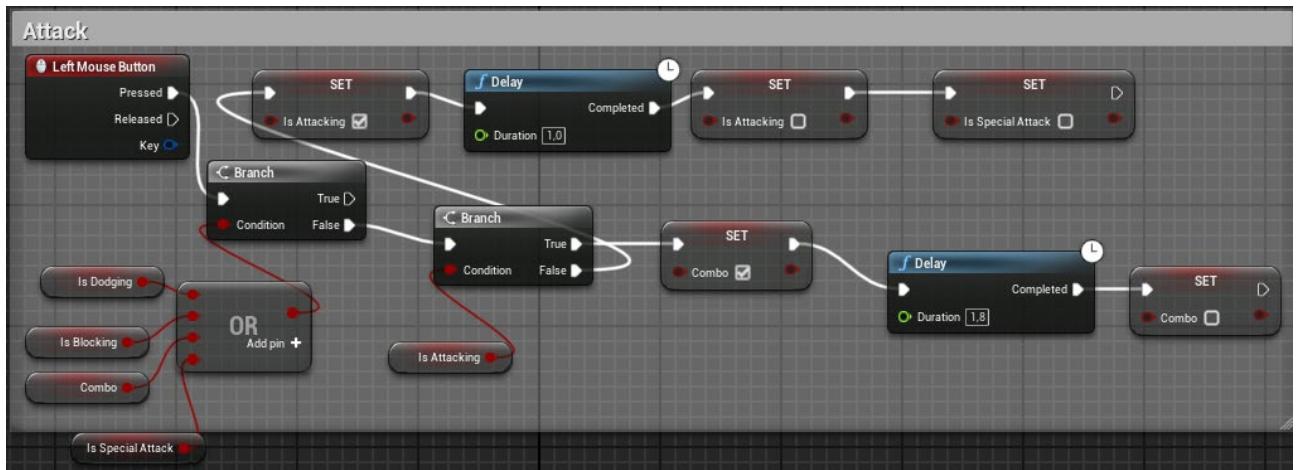


Рисунок 28 – EventGraph реализующий аналог алгоритма удара

Также помимо простого одиночного удара присутствует два особых приема: комбо и удар после ускорения.

Для использования комбо-удара (два последовательных удара) необходимо быстро два раза подряд нажать левую клавишу мыши. Это приведет к последовательному переходу сначала в состояние простого удара и проигрывания соответствующей анимации и далее в состояние второго удара и новой анимации. Для индикации перехода в данное состояния также существует своя булева переменная `Is_combo`.

Удар после ускорения — это длинный удар, который срабатывает только когда персонаж находится в состоянии бега (`WASD + Shift`) и в этот момент нажимается левая кнопка мыши. Тогда проигрывается анимация того, как персонаж подпрыгивает и, переместившись на значительное расстояние, ударяет о землю. Данное состояние характеризуется булевой переменной `Is_special_attack`.

Для использования блока (на рисунке Рисунок 29 представлен аналог алгоритма в виде графа событий Blueprint) используется правая клавиша мыши. Срабатывает анимация блока и переменная `Is_block` становится “истиной”. Если

же в момент блока противник начинает атаку в сторону персонажа игрока, и они повернуты друг к другу лицом, то срабатывает дополнительная анимация, показывающая получение удара; но урона здоровью игрока не наноситься.

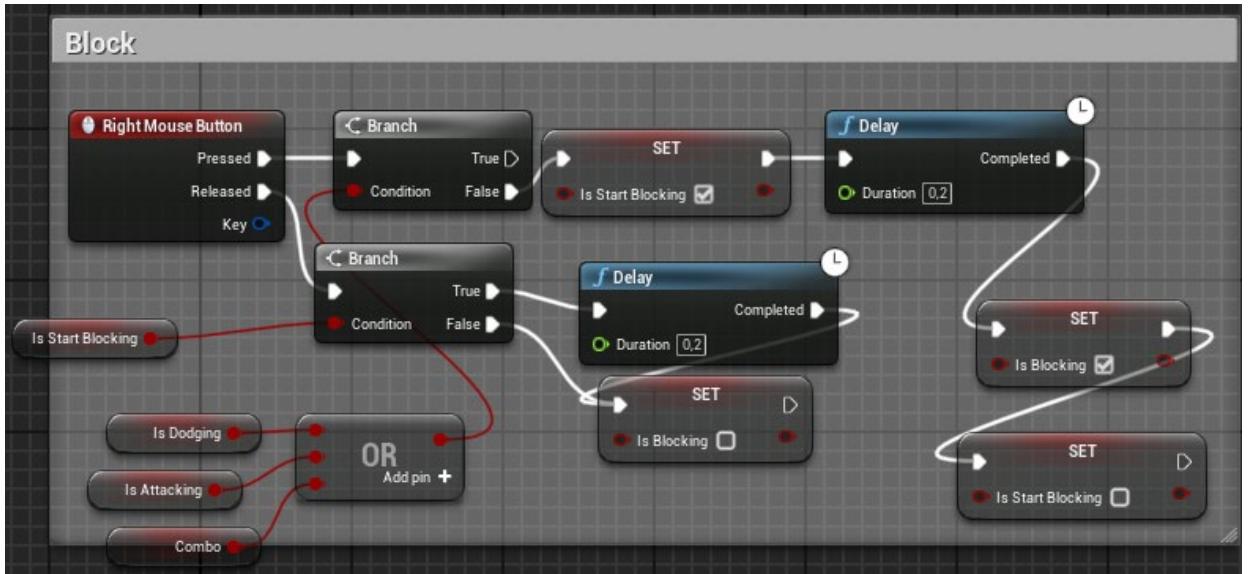


Рисунок 29 – EventGraph реализующий аналог алгоритма блокировки

Если противник начинает атаку, и персонаж игрока находится в зоне получения урона (схема данной зоны изображена на рисунке Рисунок 30), то персонаж переходит в состояние “получение урона”, проигрывается соответствующая анимация и персонаж теряет некоторое кол-во здоровья. Если же здоровья стало меньше или равным 0, то персонаж игрока уничтожается и вся карта перезагружается.

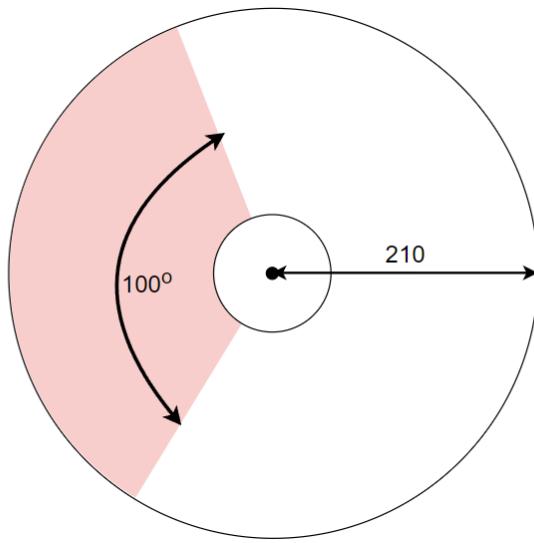


Рисунок 30 – Схема зоны вокруг противника, где персонаж может получить урон

2.3 Программирование боевой логики противника

Для персонажа под управлением игрока должен существовать какой-либо персонаж, который будет с ним сражаться, получая или нанося ему урон. В разработанной системе объект `character`, являющийся противником, полностью управляемый программным кодом, и, хотя его действия сильно зависят от действий персонажа игрока, игрок не может отдавать противнику команды напрямую.

Как и персонаж игрока, противник обладает теми же возможностями в движении и бою: ходьба, бег, перекаты, удар, блок. Также есть и особые состояния, все они изображены на графе состояний на рисунке 31. Однако, если движение противника строго определено: всегда направлено на сближение с персонажем игрока, то боевые действия - в основном случайные события.

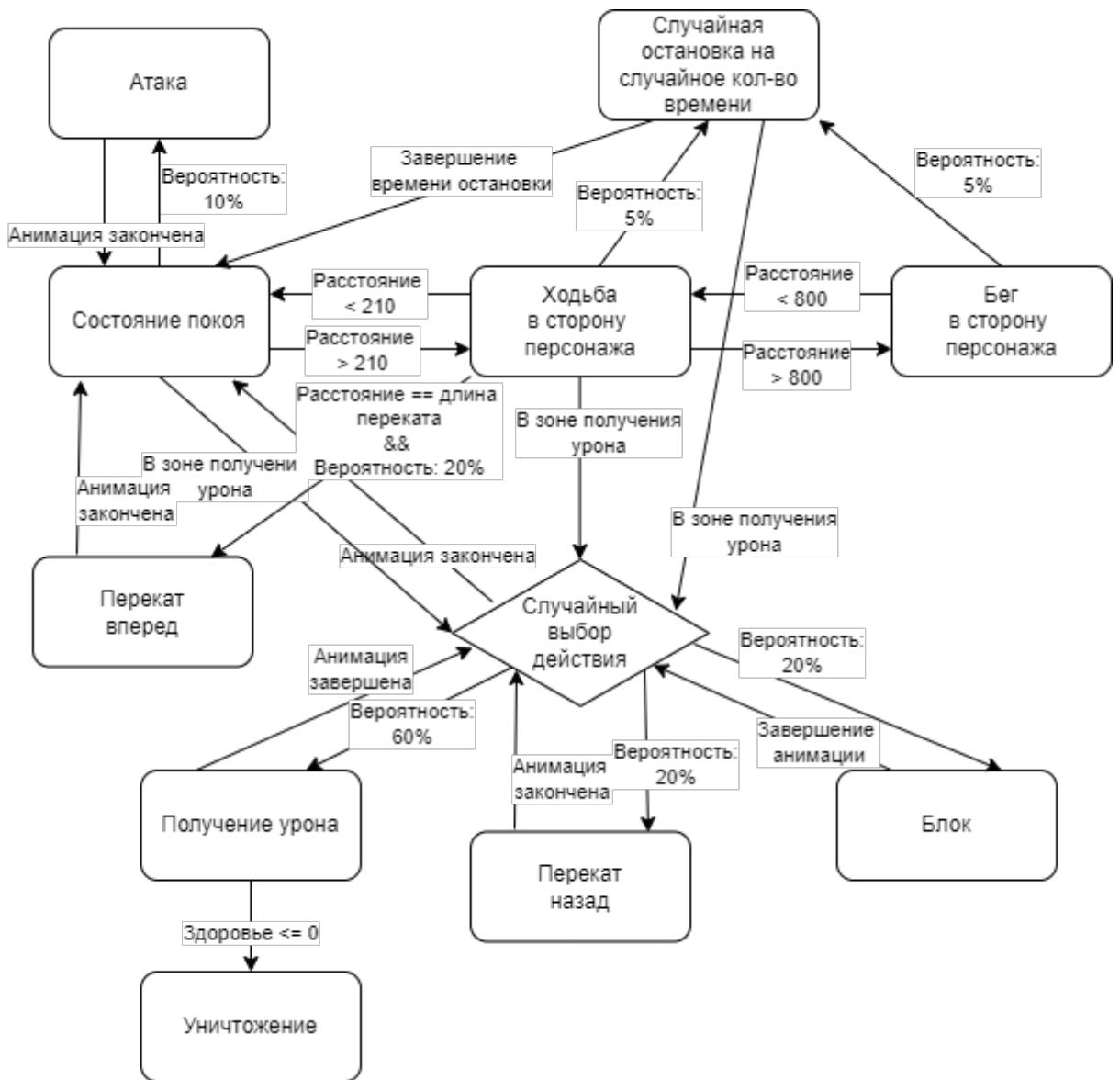


Рисунок 31 – Диаграмма состояний персонажа-противника

2.3.1 Передвижение противника

Движение противника практически всегда направлено в сторону персонажа игрока (аналог алгоритма передвижения представлен на рисунке Рисунок 32). Переход с бега на ходьбу и наоборот зависит лишь от расстояния между двумя персонажами: если оно больше 800, то противник переходит на бег(переключение переменной Is_running в “истину”); если расстояние от 210 до 800, то противник медленно идет (исходное состояние с небольшой скоростью);

если же противник находится в радиусе 210 от персонажа игрока, то он перестает двигаться в его сторону (исходное состояние и отсутствующая скорость).

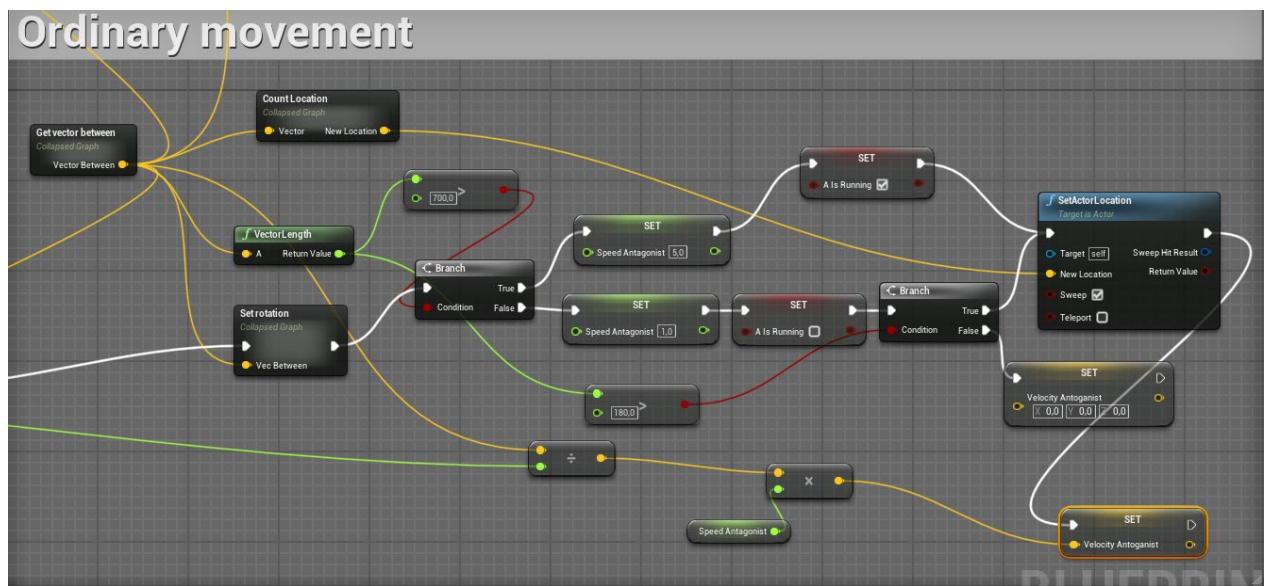


Рисунок 32 – Аналог алгоритма передвижения противника в сторону героя

Также в процессе движения с вероятностью 0.05 противник может остановиться на случайное кол-во времени от 0 до 5 секунд, это делается для разнообразия движения противника и дает игроку возможность напасть.

Также, находясь на расстоянии длины переката от персонажа игрока, противник с вероятностью 0.2 может сделать перекат вперед чтобы быстро сократить расстояние между противниками.

2.3.2 Логика боя противника

Агрессивные и защитные действия противника начинают действовать только когда персонаж находится на расстоянии менее 210 от противника. Случайные действия в ответ на атаку персонажа игрока представлены на рисунке Рисунок 33.

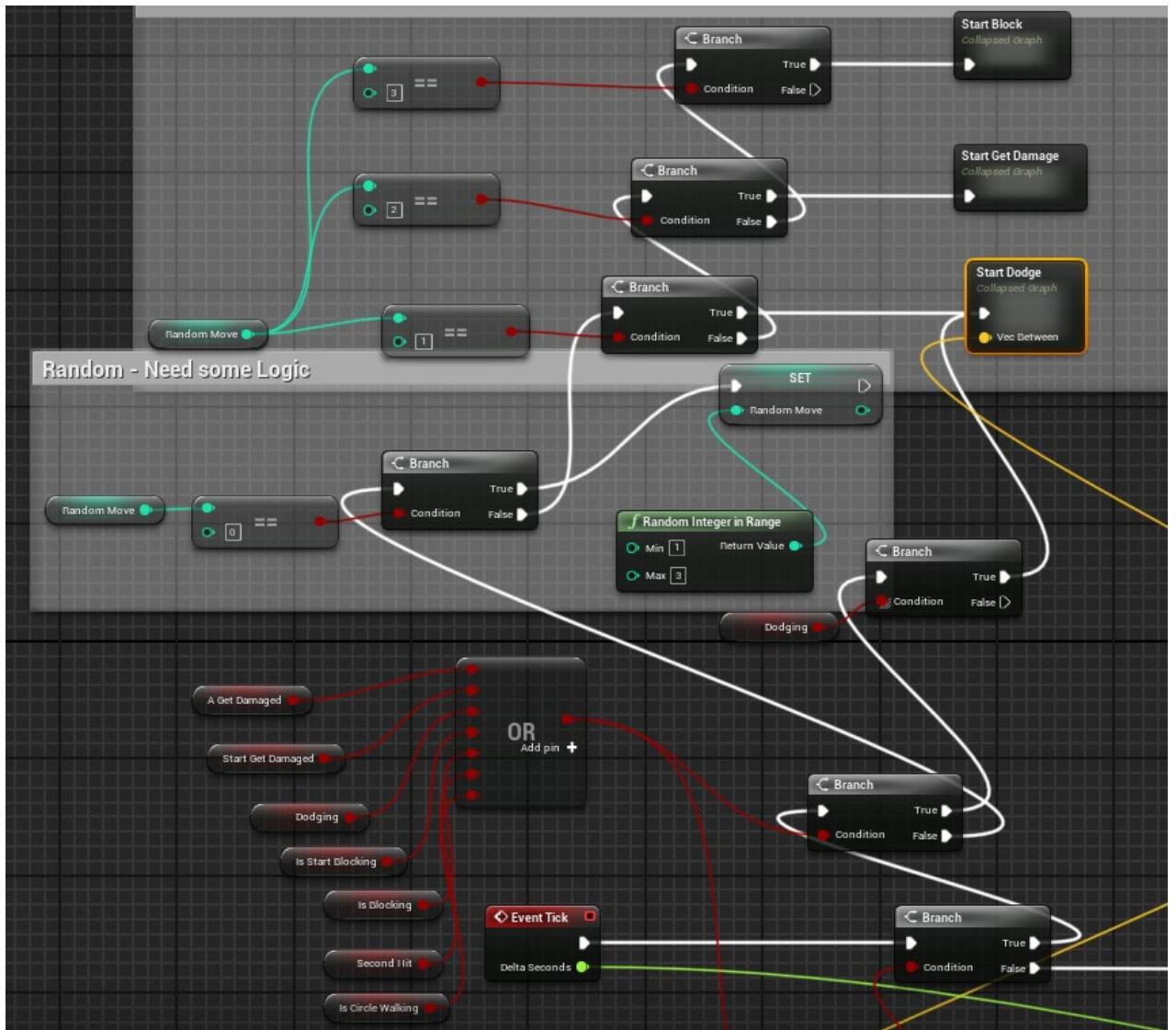


Рисунок 33 – Аналог алгоритма случайных ответных действий на атаку героя

Противник с вероятностью 0.1 может начать атаковать, при этом переключение булевой переменной, отвечающей за нанесение урона, срабатывает только через определенный промежуток времени. Таким образом анимация срабатывает так, чтобы был виден характерный замах меча, и это позволяет игроку успеть выполнить защитные действия.

Если же противник является целью атаки, то происходит случайный выбор между тремя состояниями: получение урона (вероятность 0.6), блок (вероятность 0.2), уклонение (вероятность 0.2).

При переходе в состояние получения урона проигрывается анимация в зависимости от того, какой удар нанес игрок: обычный или комбо-удар. При этом, если здоровье противника стало меньше или равным 0, то он уничтожается и на карте создается новый противник.

При переходе в состояние блока противник встает в блокирующую стойку на время, зависящее от типа удара: обычный или комбо-удар. При этом каждый удар сопровождается небольшим отклонением противника.

При переходе в состояние уклонения противник сразу же начинает делать перекат в противоположную от игрока сторону, при этом учитывается, чтобы он сделал только один перекат и при пересечении с препятствием не проходил сквозь него.

2.4 Реализация системы здоровья

Система здоровья является одной из ключевых механик боевой системы, обеспечивая реалистичное и интуитивное взаимодействие персонажей в игре. В данной системе предусмотрены различные компоненты, которые позволяют как игроку, так и противникам эффективно управлять своим здоровьем и реагировать на полученные повреждения.

2.4.1 Здоровье персонажей

Каждый персонаж, будь то герой или противник, обладает переменной здоровья, которая определяет их жизнеспособность в игре. Для реализации этой механики в классах персонажей (`New_Protagonist` и `New_Antagonist`) введены две основные переменные:

- `Health` – текущее количество здоровья персонажа. Эта переменная изменяется в ходе игры при получении урона или лечении.
- `Max_Health` – максимальное количество здоровья, которое может иметь персонаж. По умолчанию это значение установлено на уровне 120, однако

разработчики оставили возможность изменения этого параметра через редактор, чтобы обеспечить гибкость в настройке персонажей.

Уменьшение здоровья происходит при получении персонажем урона. В этот момент переменная `Is_get_damage` устанавливается в значение `true`, что инициирует процесс уменьшения текущего здоровья (`Health`). Важно отметить, что внутри классов персонажей не вызываются дополнительные события при изменении здоровья, за исключением достижения здоровья нулевого уровня. Когда здоровье персонажа падает до нуля, переменная `Is_dead` устанавливается в значение `true`, сигнализируя о смерти персонажа. Это событие запускает анимацию умирания и последующие действия, такие как респавн противника или появление окна с оповещением для героя.

2.4.2 Визуальные элементы, показывающие здоровье

Для обеспечения визуальной обратной связи игроку реализованы различные элементы интерфейса, отображающие текущее состояние здоровья персонажей.

Для отображения здоровья игрового персонажа используется виджет здоровья героя (`Health_Protagonist`), который был разработан с использованием системы Unreal Motion Graphics (UMG). Этот виджет включает в себя элемент `Progress Bar`, который графически отображает текущее состояние здоровья героя (Рисунок 34). При инициализации виджет получает значение максимального здоровья (`Max_Health`). Далее, каждый игровой тик, виджет обновляет процент заполнения `Progress Bar`, деля текущее здоровье (`Health`) на максимальное. Blueprint виджета, который выполняет процесс его изменения изображен на рисунке Рисунок 35. Это позволяет игроку всегда видеть актуальное состояние здоровья своего персонажа, что важно для принятия тактических решений в бою.



Рисунок 34 – Виджет здоровья персонажа

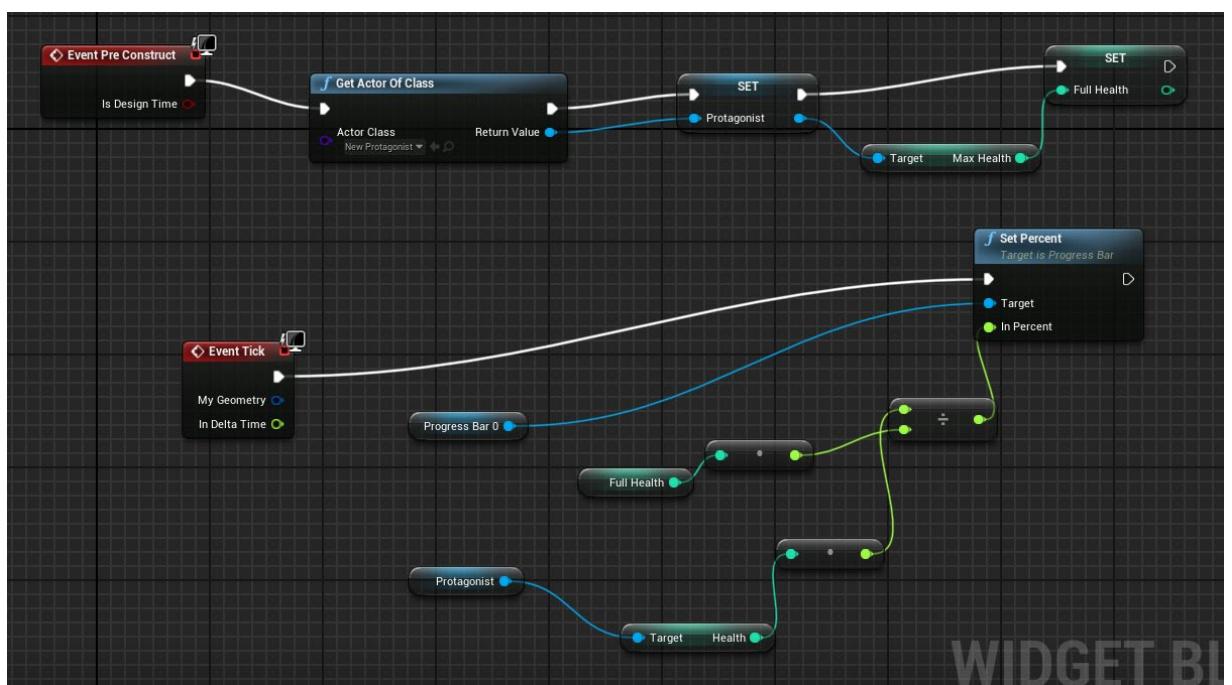


Рисунок 35 – Blueprint для виджета здоровья игрока

Для отображения здоровья противников используется физический объект (Health_Bar_Antagonist), представляющий собой элемент класса Plane. Этот объект отображается над головой противника и динамически изменяется в зависимости от его текущего здоровья (Рисунок 36).



Рисунок 36 – Отображение здоровья противника с помощью объекта Plane

При инициализации объект получает значение максимального здоровья противника и затем, каждый игровой такт, масштабируется по оси Y, деля текущее здоровье на максимальное. Положение Health_Bar_Antagonist обновляется через вызов GetActorLocation у объекта класса New_Antoganist, обеспечивая точное соответствие позиции шкалы и противника. Поворот объекта высчитывается через разницу между положением игрока и противника. Затем данный вектор преобразуется к типу FRotator, отвечающий за угол поворота объекта по всем трем осям. Весь процесс изменения объекта можно увидеть на рисунке Рисунок 37, где изображен EventGraph шкалы здоровья противника.

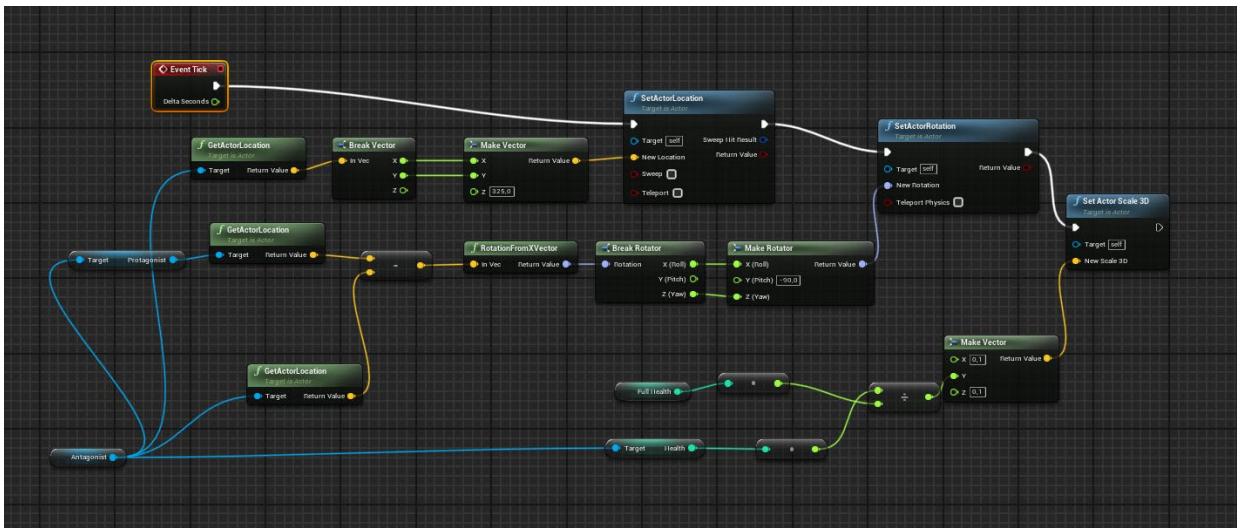


Рисунок 37 – Blueprint для виджета здоровья противника

2.4.3 Процесс респавна противников

Процесс респавна противников был разработан для поддержания постоянного уровня вызова для игрока и динамичного игрового процесса. Когда здоровье противника достигает нулевого уровня, и заканчивается анимация смерти, система вводит паузу на 3 секунды. Это создаёт реалистичный эффект задержки перед исчезновением тела. Затем вызывается функция `DestroyActor`, которая удаляет противника из игрового мира.

Для выбора новой позиции респавна используется случайный выбор точки в радиусе 1000 метров от исходной позиции. Новый противник создаётся с помощью функции `SpawnActor` в этой позиции и получает все параметры, аналогичные уничтоженному персонажу. Такой подход обеспечивает постоянное присутствие противников в игре, поддерживая напряжение и динамику боя. Фрагмент Level Blueprint, осуществляющий данную функцию представлен на рисунке Рисунок 38.

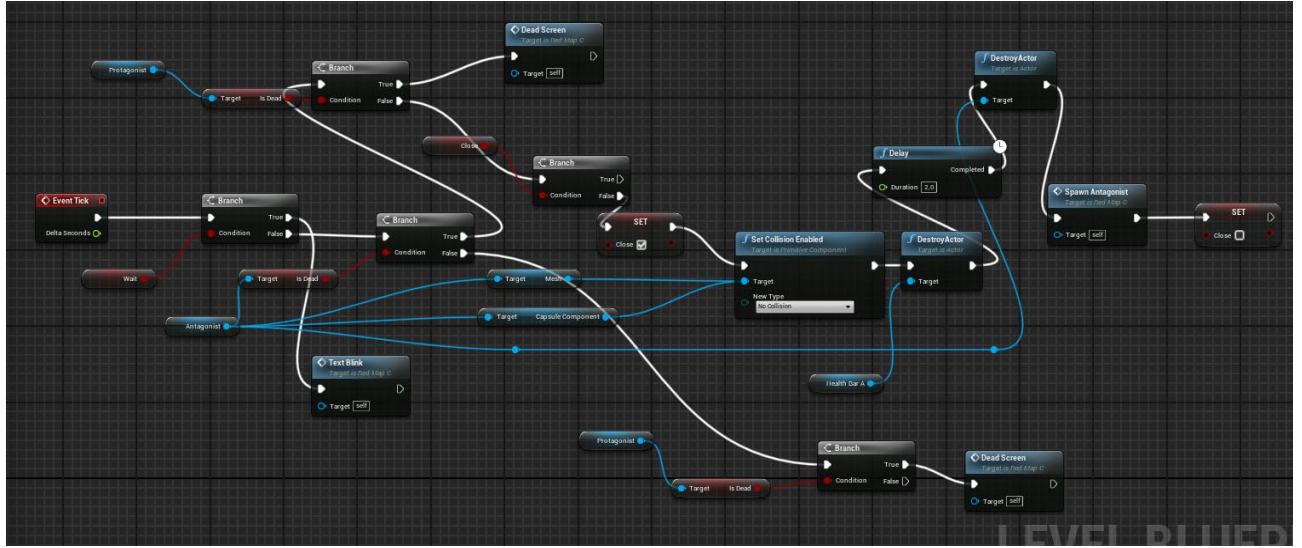


Рисунок 38 – Level Blueprint для выбора действий после смерти персонажа

2.4.4 Смерть игрового персонажа

После смерти игрового персонажа выводится специальный интерфейсный элемент, информирующий игрока о гибели героя. Для этого используется виджет Died_Screen, который состоит из трёх элементов Image. Когда переменная Is_dead становится истиной, срабатывает анимация смерти персонажа. Также это событие вызывает срабатывание соответствующего кода в Level Blueprint (Рисунок 38). Виджет начинает плавно изменять свою прозрачность от 0 до 1, заполняя весь экран.

В этот момент все действия игрока, включая вращение камеры, становятся невозможными, что реализовано в классе New_Protagonist. На экране появляется картинка с оповещением о смерти персонажа и мигающая надпись "Press Enter to continue". По нажатию клавиши Enter переменная Is_dead сбрасывается в значение false, позволяя игроку снова управлять персонажем. Виджет становится прозрачным, и игровой процесс продолжается (Рисунок 39). Этот элемент интерфейса создан для обеспечения четкой и понятной обратной связи игроку в случае гибели его персонажа.

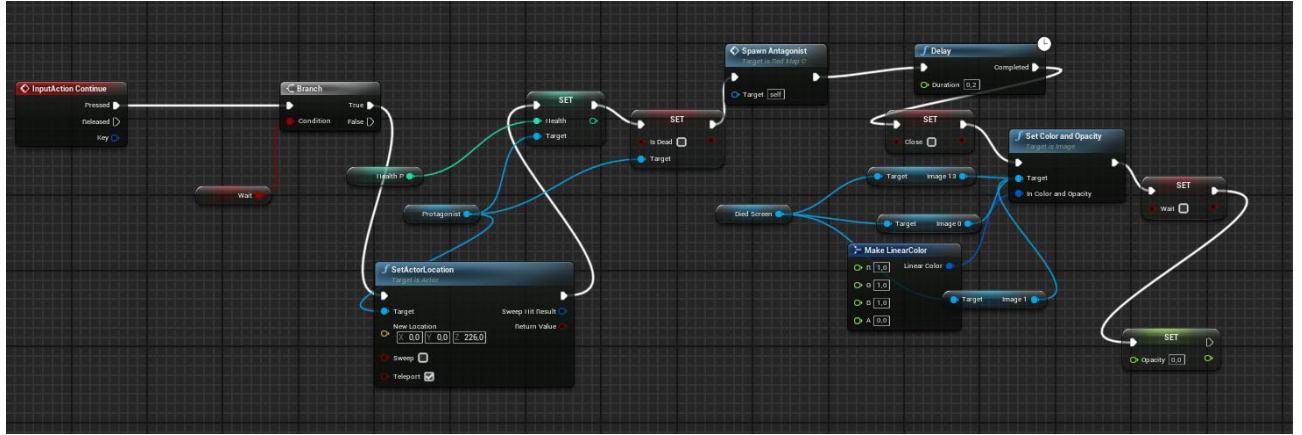


Рисунок 39 – Фрагмент Level Blueprint для «возрождения» игрока

2.5 Реализация системы атак с умным буфером

Система умной буферизации атак в игре была разработана для повышения отзывчивости и плавности боевых действий. В отличие от традиционного режима атак, где последовательные нажатия кнопки мыши могут прерывать текущие анимации ударов, умная буферизация позволяет игроку непрерывно кликать, и анимации будут корректно воспроизводиться. Это достигается за счет использования простого буфера, который заполняется в момент нажатия кнопки мыши во время выполнения атаки. Когда первая анимация удара завершается, система проверяет состояние буфера и, если он активен, запускает следующую анимацию атаки. В противном случае персонаж переходит в состояние покоя.

Для реализации буфера используется булева переменная `Attack_buf` в классе игрового персонажа `New_Protagonist`. Эта переменная служит индикатором нажатия кнопки мыши и становится истиной каждый раз при нажатии левой кнопки мыши (ЛКМ). Переменная проверяется в момент завершения текущей анимации атаки, определяя, должна ли начаться следующая анимация.

Нажатия ЛКМ регистрируются с помощью системной функции `BindAction`, что позволяет корректно отслеживать все попытки игрока атаковать. Несмотря на то, что игрок может многократно нажимать кнопку мыши во время выполнения одной атаки, переменная `Attack_buf` является булевой, что

гарантирует запуск только одной следующей атаки после завершения текущей. Таким образом, система эффективно предотвращает переполнение буфера и обеспечивает стабильное поведение атак.

Переход в состояние атаки определяется переменной `Is_attacking`, которая становится истиной при начале атаки. Это состояние контролируется и управляет кодом, обеспечивая четкое разграничение между активной фазой атаки и фазой покоя. Когда персонаж атакует, `Is_attacking` установлена в значение `true`, что сигнализирует системе об активном выполнении боевых действий.

На текущий момент система поддерживает две основные анимации атаки: удар справа и удар слева. Эти анимации плавно переходят друг в друга при постоянном нажатии ЛКМ, создавая иллюзию непрерывной атаки. Переходы между анимациями реализованы в AnimGraph (Рисунок 40), что позволяет гибко управлять последовательностью атак и обеспечивать плавные переходы между движениями. При этом вначале всегда идет анимация первого удара справа, но закончится и перейти в состояние покоя можно из любой из двух анимаций атаки.

Конец текущей анимации определяется переменной `Is_attacking`, которая переходит в значение `false` после определенной задержки, установленной в коде класса игрового персонажа. Этот механизм позволяет точно определить момент завершения атаки и проверить состояние буфера `Attack_buf`. Если буфер активен, запускается следующая анимация атаки; если нет – персонаж возвращается в состояние покоя.

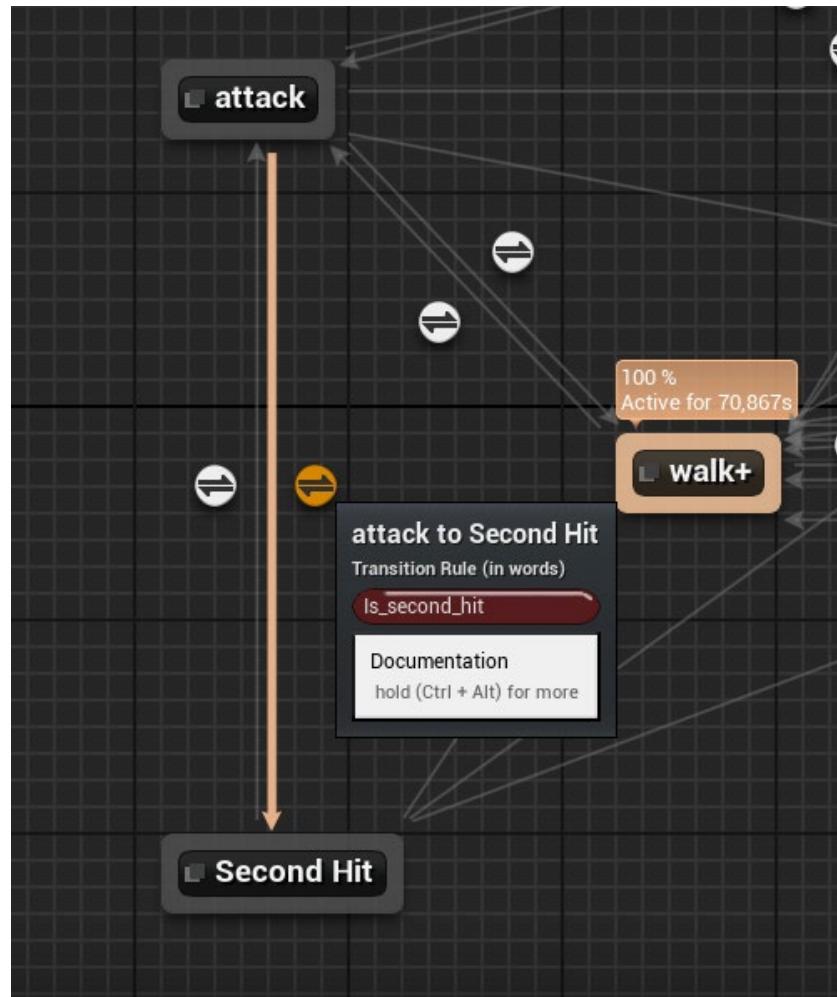


Рисунок 40 – Фрагмент AnimGraph, реализующий переход между анимациями атак и состоянием покоя

Система умной буферизации атак значительно улучшает игровой опыт, делая боевые действия более плавными и интуитивно понятными. Возможность непрерывного нажатия кнопки мыши и автоматического воспроизведения анимаций атак повышает уровень вовлеченности и удовлетворенности игроков. Тщательная реализация и оптимизация системы обеспечивают ее надежную работу и высокую производительность, что является важным аспектом успешного игрового проекта.

2.6 Реализация системы визуального захвата противника

Режим фиксации камеры, также известный как Attach-режим, предоставляет игроку возможность сосредоточиться на конкретном противнике, обеспечивая постоянное наблюдение за ним и упрощая управление боевыми действиями. Этот режим делает бой более интуитивным и удобным, позволяя игроку удерживать камеру и персонажа направленными на выбранную цель.

2.6.1 Работа камеры в режиме фиксации

Режим фиксации камеры может быть активирован двумя способами:

1) Самостоятельная активация. Игрок может вручную активировать режим фиксации камеры, нажав среднюю кнопку мыши (СКМ). В этот момент переменная `Is_target_set` класса игрового персонажа устанавливается в значение `true`, что сигнализирует системе о необходимости фиксации на цели.

2) Активация при блокировании. Переход в режим фиксации камеры также происходит автоматически при переходе в режим блока с помощью правой кнопки мыши (ПКМ). Ближайший противник становится целью, и система начинает фиксировать камеру на нем.

В обоих случаях целевым противником становится ближайший к игроку враг. Это позволяет игроку быстро и эффективно сосредоточиться на текущей угрозе.

Для реализации фиксации камеры используется функция `SetRotation` контроллера игрового персонажа. Угол поворота контроллера вычисляется как вектор между игроком и ближайшим противником. Эта функция вызывается каждый тик, что обеспечивает постоянное направление камеры на цель, независимо от ее движений.

При активации режима фиксации камеры персонаж автоматически поворачивается лицом к выбранному противнику. Это достигается путем установки ротации контроллера в каждый тик, что синхронизирует направление

персонажа и камеры с позицией противника. Таким образом, игрок всегда видит своего врага и может точно контролировать боевые действия.

Режим фиксации камеры может быть деактивирован двумя способами:

1) Повторное нажатие СКМ. Игрок может вручную отключить режим фиксации камеры, снова нажав среднюю кнопку мыши. Это позволяет быстро вернуться к обычному режиму управления камерой.

2) Выход из режима блока. При выходе из режима блока, активированного правой кнопкой мыши, режим фиксации камеры также автоматически отключается.

2.6.2 Анимации в режиме фиксации камеры

В режиме фиксации камеры анимации персонажа играют ключевую роль, обеспечивая реалистичность и плавность его движений. Поскольку в этом режиме персонаж всегда повернут лицом к выбранному противнику, а его движения могут быть направлены в разные стороны, используются специальные анимации, зависящие от направления движения [10]. Это достигается за счет сочетания нескольких технологий и инструментов, доступных в Unreal Engine.

Одним из ключевых элементов системы анимации является использование вектора скорости (Velocity), который задается персонажу при нажатии клавиш направления [11]. Хотя сам поворот персонажа остается неизменным и всегда ориентирован на противника, его движения осуществляются в различных направлениях, что позволяет использовать разнообразные анимации для каждой ситуации.

Персонажу задается вектор скорости, соответствующий направлению его движения. Этот вектор обновляется каждый такт, когда игрок нажимает клавиши движения (W, A, S, D). Вектор Velocity позволяет определить точное направление движения персонажа относительно его текущего поворота.

В Animation Blueprint направление движения вычисляется как угол от -180 до 180 градусов, который получается путем преобразования вектора Velocity и

поворота персонажа (Рисунок 41). Этот угол используется для выбора соответствующей анимации, что делает движения персонажа реалистичными и адаптивными к текущей ситуации.

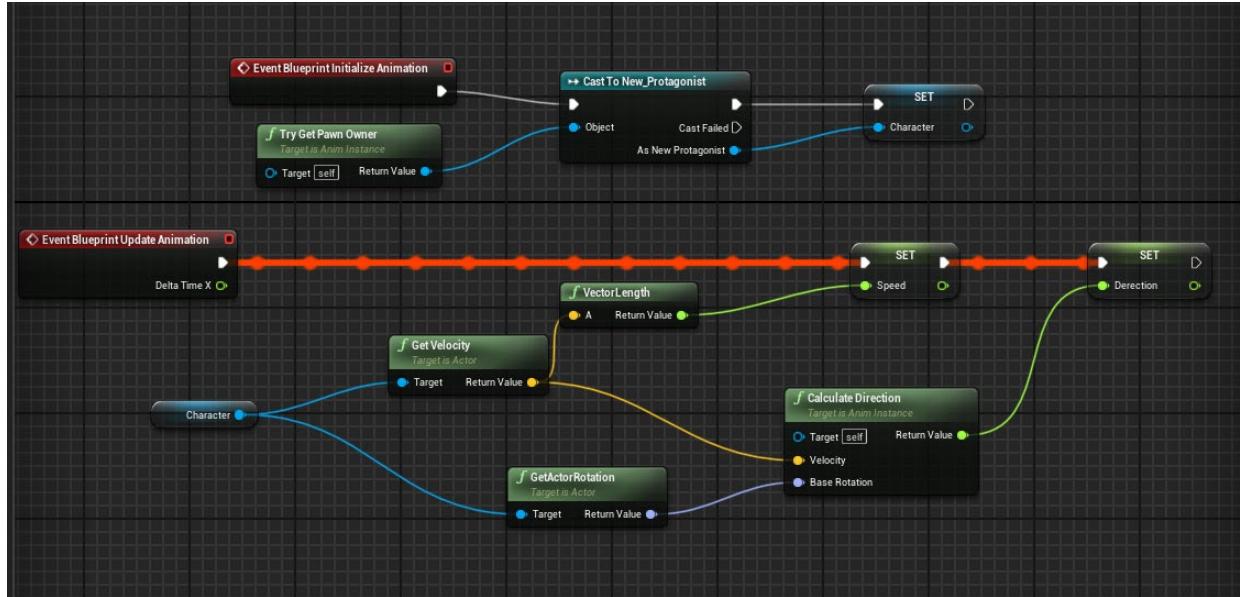


Рисунок 41 – EventGraph, вычисляющий направления и скорости анимации

Для обеспечения плавных переходов между анимациями используется структура Blend Space. Blend Space представляет собой набор анимаций, организованных по шкале направления и скорости. В зависимости от угла направления и величины скорости персонажа Blend Space выбирает подходящую анимацию или комбинацию анимаций, создавая плавный и непрерывный переход между различными состояниями движения (Рисунок 42).

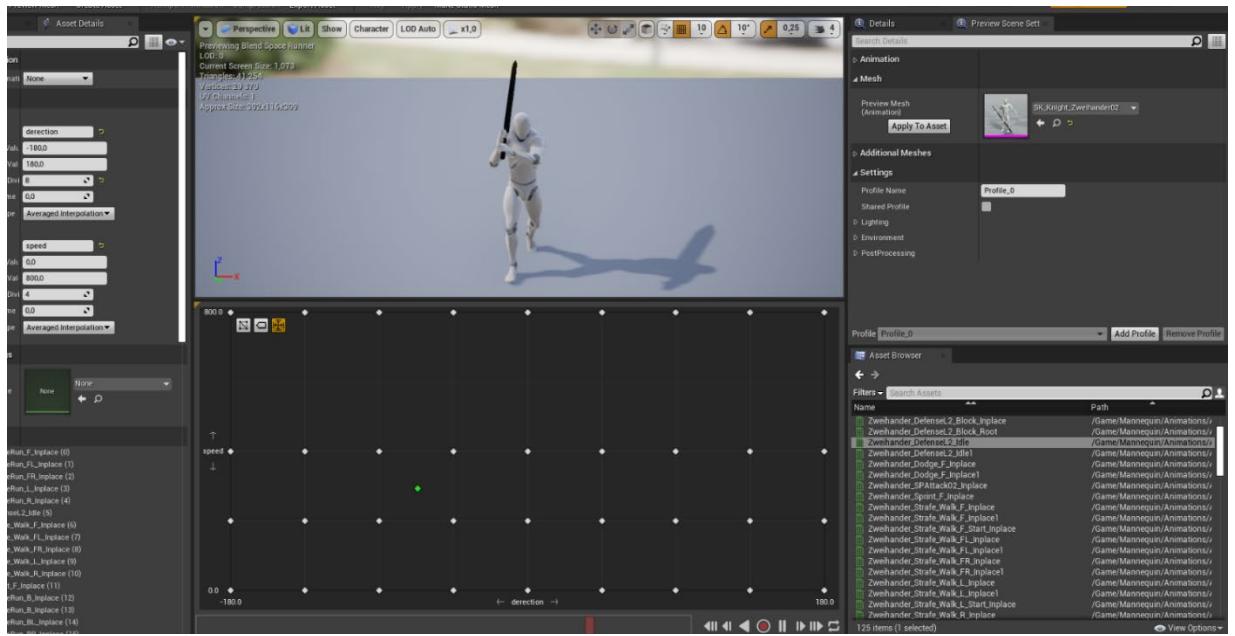


Рисунок 42 – Окно редактора Blend Space

Например, при движении персонажа вперед с высокой скоростью будет использована анимация бега вперед, в то время как при движении влево или вправо – соответствующие анимации бокового шага. При этом, если персонаж движется под углом, Blend Space позволяет смешивать анимации, создавая плавный переход, который соответствует реальному движению.

В режиме фиксации камеры предусмотрены следующие типы анимаций, зависящих от направления движения:

- Ходьба: Анимации ходьбы адаптированы для движения в различных направлениях. Например, при движении вперед используется анимация ходьбы вперед, а при движении назад – анимация шагов назад. Боковые шаги также имеют свои уникальные анимации, которые используются при движении влево и вправо.

- Бег: В режиме фиксации камеры бег сопровождается анимациями, соответствующими направлению движения. Например, бег вперед, назад, влево и вправо имеют свои специфические анимации. Blend Space позволяет плавно переходить между этими анимациями в зависимости от направления и скорости движения.

- Перекаты: Перекаты выполняются с учетом направления и позиции противника. Анимации перекатов направлены на то, чтобы персонаж мог быстро уклониться от атаки или переместиться в более выгодное положение. При этом направление переката зависит от текущего движения персонажа и его ориентации относительно противника.

ЗАКЛЮЧЕНИЕ

В процессе данной работы был исследован многогранный инструментарий Blender в контексте создания 3D-моделей. Обзор возможностей Blender раскрывает его важную роль как мощного инструмента для дизайнеров и разработчиков, предоставляя широкий набор инструментов для моделирования, текстурирования, и анимации.

Был проведен общий обзор 3D-моделирования в Blender и в практической части были созданы две уникальные модели: конвейерная линия и мифическое существо: полукот-полусова.

Работа с Blender выявила его многослойные возможности: от базового моделирования до тонкой настройки анимации и визуальных эффектов.

Для дальнейшего использования модели был проведен ее перенос в Unreal Engine, который представил свои уникальные вызовы. Несмотря на технологические преимущества FBX-формата, модели не всегда переносятся без проблем. Это потребовало дополнительной работы и творческого решения проблем для обеспечения корректного отображения и воспроизведения анимации и моделей в Unreal Engine.

Иным вариантом создания фотorealистичных моделей был рассмотрен инструмент Metahuman Creator, генерирующий высококачественные модели, но лишь примерно похожих людей.

Для придания моделям реалистичных движений были рассмотрены методы motion capture и сравниены несколько программ, объединенных методом VMC, который способен передать движения напрямую в Unreal Engine.

Объединяя усилия Blender, программ для захвата движения и Unreal Engine, можно создать не только модели, но и визуализацию, готовую к интеграции в игровую или иную среду. Этот путь подчеркнул не только важность технических навыков, но и творческого мышления в процессе создания виртуальных миров. Все эти этапы в совокупности формируют полноценный цикл разработки - от идеи и моделирования до добавления его в реальный проект.

Комплексное использование моделей и анимации было продемонстрировано в рамках разработки боевой системы для 3-х мерного видео игрового экшена для Unreal Engine 4.2. В рамках разработки был написан C++ код управляющий как действиями персонажа под управлением игрока, так и противника, который полностью управляет с помощью написанного кода.

Разработана несложная система атак и защиты, которая в динамике позволяет игроку как насладиться процессом, так и испытать свои навыки и реакцию. Такой проект, разработанный на Unreal Engine при дальнейшей разработке, может вырасти до полноценного видео игрового экшена, не только с неплохой боевой системой, но и с красивой визуальной составляющей и, возможно, интересным сюжетом и сеттингом.

Таким образом продемонстрирован весь процесс создания цифрового продукта на Unreal Engine, начиная от идеи и проектирования, до готового рабочего проекта, который может запустить пользователь.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Modeling – Blender Manual [Электронный ресурс] // Blender Manual. URL: <https://docs.blender.org/manual/en/latest/modeling/index.html>. (Дата обращения: 14.11.2023)
- 2) Обучающий курс “Фабрика” [Электронный ресурс] // YouTube. URL: https://www.youtube.com/playlist?list=PLn6DikVGbeEiJFNb2_wfV2zg4BDm8xvsQ. (Дата обращения: 16.11.2023)
- 3) Обучающий курс “Енот” [Электронный ресурс] // YouTube. URL: https://www.youtube.com/playlist?list=PLn6DikVGbeEgMvn_JJyX1Rnrt3Wlj0rvk. (Дата обращения: 22.11.2023)
- 4) Unreal Engine 4 Documentation [Электронный ресурс] // Unreal Engine Documentation. URL: <https://docs.unrealengine.com/>. (Дата обращения: 27.11.2023)
- 5) Public asset library [Электронный ресурс] // Poly Heaven. URL: <https://polyhaven.com/>. (Дата обращения: 22.11.2023)
- 6) Сравнение систем захвата движения для создания анимированных персонажей в Unreal Engine: научная публикация (не опубликованная) / Витюков Ф.А., Эвоян Э.Б. – Москва, 2023. – 2 с.
- 7) Official VMC Protocol webpage [Электронный ресурс] // VMC Protocol specification. URL: <https://protocol.vmc.info/english.html>. (Дата обращения: 12.12.2023)
- 8) Плагин Unreal Engine для использования методов искусственного интеллекта в проектах Unreal Engine [Электронный ресурс] // Github. URL: <https://github.com/endink/Mediapipe4u-plugin>. (Дата обращения: 22.12.2023)
- 9) Обучающий курс “Вводный курс по анимации в UE” [Электронный ресурс] // YouTube. URL: <https://www.youtube.com/playlist?list=PLzn9qrbm2X8OuNjFPw2rIj8O1h1Kt8JX0>. (Дата обращения: 28.02.2024)
- 10) Обучающий курс “Продвинутый курс по анимации в UE” [Электронный ресурс] // YouTube. URL: <https://www.youtube.com/playlist?list=PLUi8nuTUEtTvTb8Wk6cBfvw8IEFAPZYqd>. (Дата обращения: 03.03.2024)

- 11) Обучающий курс “Основы управление анимацией из C++” [Электронный ресурс] // YouTube. URL: <https://www.youtube.com/watch?v=ftjflBJwFAU>. (Дата обращения: 03.03.2024)
- 12) Unreal Engine 5.4 release note [Электронный ресурс] // Unreal Engine Official Web-site. URL: <https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5.4-release-notes>. (Дата обращения: 28.04.2024)

ПРИЛОЖЕНИЕ А

Графическая часть дипломного проекта

В графическую часть дипломного проекта входят:

- 1) Схема взаимодействия технологий в стеке ELK (рисунок 6);
- 2) Программная архитектура SIEM-системы (рисунок 7).