|  |  |
| --- | --- |
| Герб МГТУ | **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)» |

**НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА**

**НА ТЕМУ:**

***«Создание видеоигровой боевой системы на Unreal Engine 4»***

|  |  |  |
| --- | --- | --- |
| Студент РК6-84Б | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Шендрик Д.А.** |
|  | (Подпись, дата) | (И.О. Фамилия) |
| Руководитель ВКР | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Витюков Ф.А.** |
|  | (Подпись, дата) | (И.О. Фамилия) |

*2024 г.*

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc164703868)

[ПОСТАНОВКА ЗАДАЧИ 6](#_Toc164703869)

[ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 7](#_Toc164703870)

[1. The Witcher III: Wild Hunt 8](#_Toc164703871)

[2. Elden Ring 11](#_Toc164703872)

[ПРАКТИЧЕСКАЯ ЧАСТЬ 15](#_Toc164703873)

[1. Анимации боя 15](#_Toc164703874)

[2. Программирование боевой логики игрового персонажа 20](#_Toc164703875)

[3. Программирование боевой логики противника 25](#_Toc164703876)

[ЗАКЛЮЧЕНИЕ 29](#_Toc164703877)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc164703878)

**ОПИСАНИЕ**

Отчет 30 с., 5 источников

Объект исследования является видеоигровая боевая система и ее компоненты: персонажи, логика перемещения и взаимодействия.

Цель работы - сравнительное изучение уже существующих боевых систем и разработка собственного аналога на Unreal Engine 4.

В процессе работы изучены особенности видеоигровой боевой системы.

В результат была создана программная реализация простой боевой системы сражения двух персонажей.

Степень внедрения - программа впоследствии может быть доработана и использована как компонент полноценной видеоигры.

## 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.

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

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 Elden Ring и Witcher III.

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.

# **ПОСТАНОВКА ЗАДАЧИ**

Необходимо изучить существующие игры в контексте их боевой системы, декомпозировать их в виде конечного автомата и рассмотреть возможности применения их частей для разработки собственной программной реализации.

Требуется разработать программную реализацию боевой системы на языке С++ в Unreal Engine 4. В программе должны быть реализованы следующие функции:

* Система перемещения: ходьба, бег, удары, увороты/перекаты, защитная стойка.
* Система атак/обороны: удар (несколько типов), уворот/перекат, защитная стойка.

Должна быть разработана отдельная программа для персонажа, которым будет управлять реальный человек, и для противника, полностью управляемая программным кодом.

# **ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ**

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. 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 2-3, 5-6 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

Игра 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 (см. рисунок 1). 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 1 – Скриншот боя из игры The Witcher III: Wild Hunt

Хотя в игре и 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 2 – 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 рисунке 2). 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 3 – 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 рисунке 3), 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.

Для борьбы с некоторыми наиболее сложными противниками необходима некоторая подготовка: необходимо найти, где он скрывается, изучить его слабые места посредством расследования, заготовить зелья и метательные предметы, заточить износившееся оружие. Только таким образом можно сразить сложных противников игры на высоком уровне сложности.

### Elden Ring

****Игра вышла в 2022 году и является представителем особого жанра игр, которые предлагают игроку сложные бои, требующие невероятной концентрации (см. рисунок 4), в игре часто необходимо заучивать последовательность ударов противников и находить их уязвимые места.

Рисунок 4 – Скриншот боя из игры Elden Ring

Бои в игре могут проходить очень разнообразно (упрощенная диаграмма состояний персонажа представлена на рисунке 5), 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 5 – 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 (коротки, 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, блокировка 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 закончиться, 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. Особыми способами можно увеличить скорость восстановления выносливости или же увеличить ее общее количество повышая стойкость персонажа.

Рисунок 6 – Упрощенная диаграмма возможных состояний противника в игре Elden Ring

Сами бои в игре очень часто напряженные и сложные, пропустив один сильный удар, можно сразу погибнуть (упрощенная диаграмма состояний противника представлена на рисунке 6). 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 будет в какой-то момент бесполезны против него.

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

# **ПРАКТИЧЕСКАЯ ЧАСТЬ**

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 состояние оппонента. В некоторых случаях переходы являются случайными (переходы в некоторые состояния противника), и характеризуются только вероятностью перейти в данное состояние в каждый такт. Общая логика переключений состояний персонажа и его противника в контексте конечных автоматов приведены на рисунках 11 и 18 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.

#### 1. **Анимации боя**

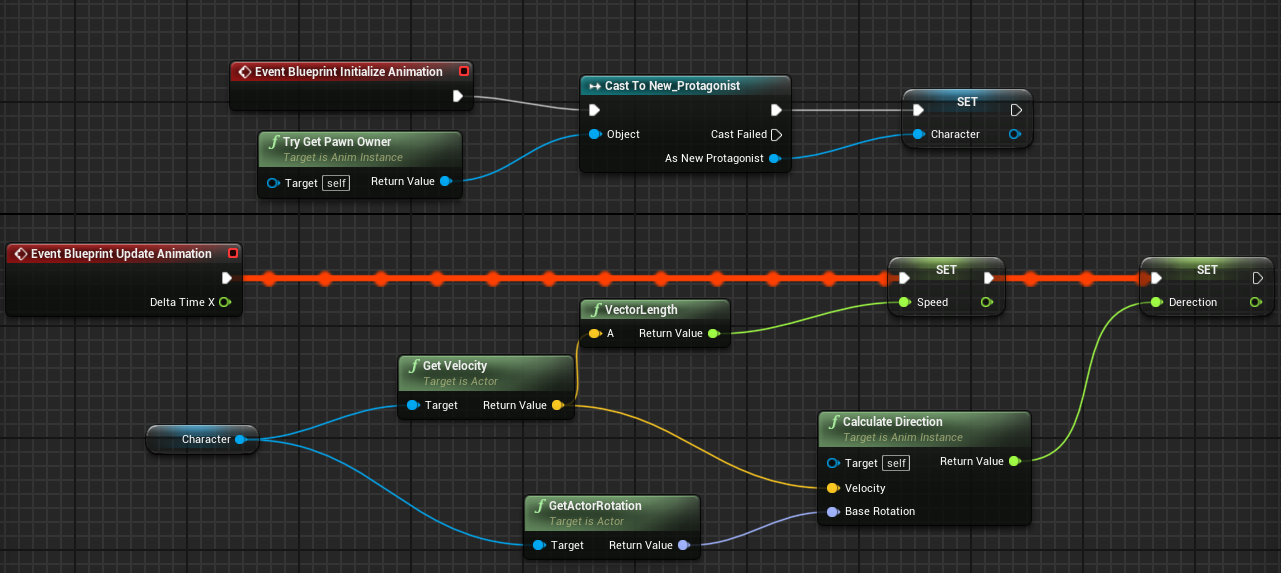
Для реализации анимации в Unreal Engine были использованы анимации и Animation Blueprint (AB), отвечающий за логику анимации. Для AB характерны две составляющие: EventGraph и AnimGraph. EventGraph отвечает за изменения самого AB, вычисляет локальные переменные. AnimGraph необходим для построения логики анимации, характеризуется состояниями, которые могут быть как просто анимациями так и сложными blend-анимациями; и связями, на которые накладываются некие условия. На рисунках 18, 19, 20 приведены EventGraph и AnimGraph для персонажа и противника. Event Graph будет идентичн для обоих персонажей, отличие лишь в том, к какому типу будет приведена переменная character, представляющая собой ссылку на другого персонажа в Blueprint (см. рисунок 7).

Рисунок 7 – 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 8 и 9, 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 AnimGrapth. Также в AnimGraph используются созданные в родительском классе булевы переменные, отвечающие за состояния (например состояние бега или состояние получение урона), поэтому они применяются в условиях связей графа, для переключения анимации (См. рис. 10).

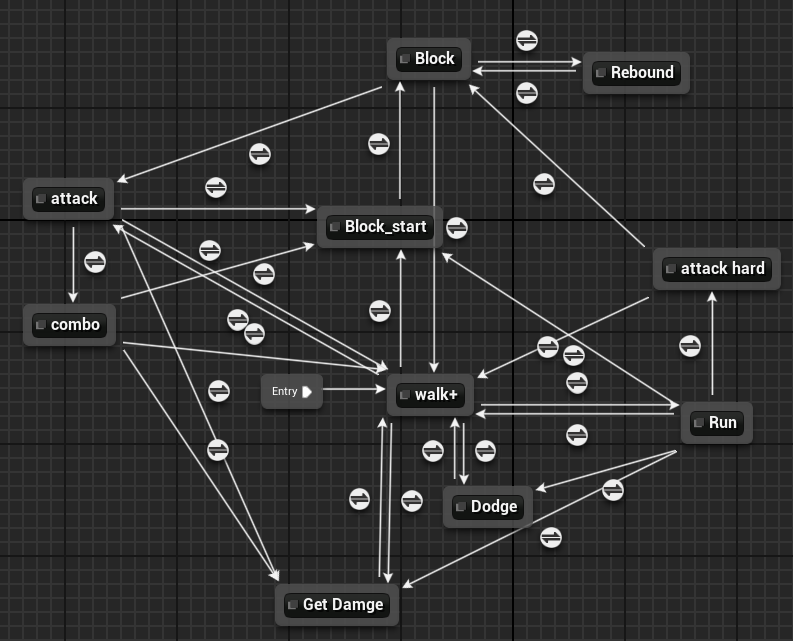
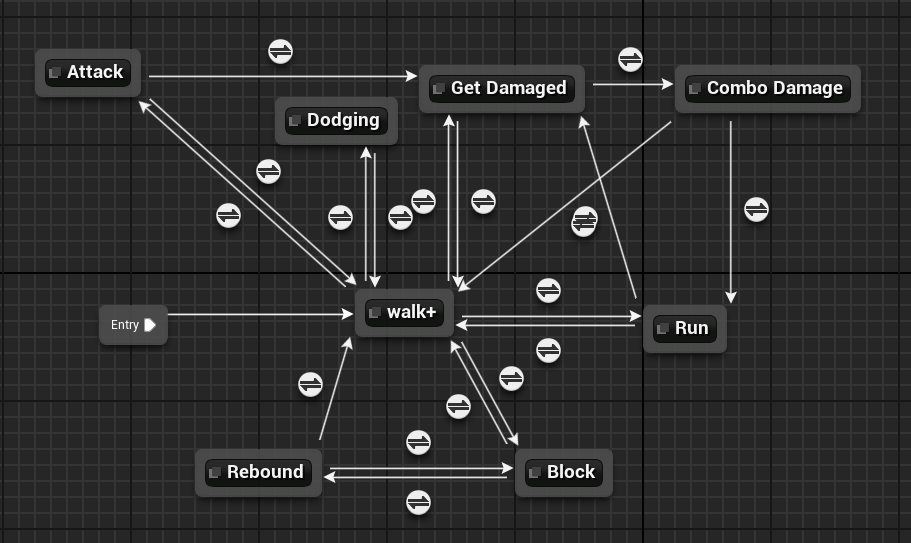
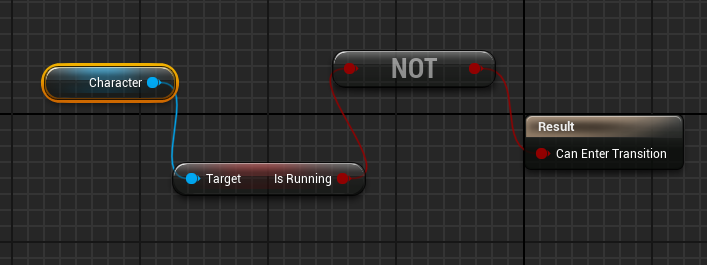
Рисунок 8 – AnimGraph для игрового персонажа

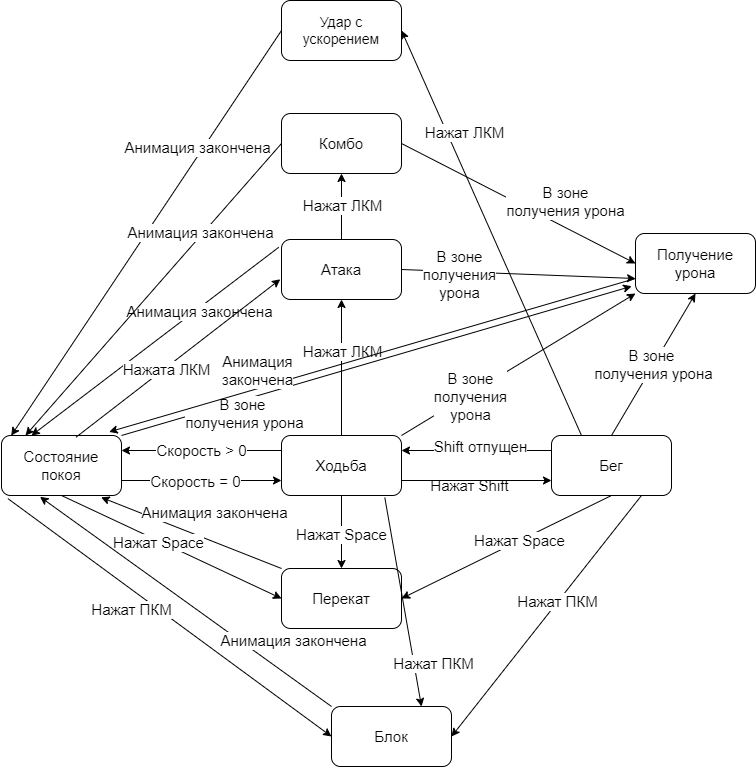
Рисунок 9 – AnimGraph для противника игрока

Рисунок 10 – Пример условия для перехода из состояния бега в исходное состояние ходьбы в AnimGraph игрового персонажа

Полная диаграмма состояний игрового персонажа представлена на рисунке 11. 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 11 – Диаграмма состояний персонажа под управлением игрока

#### 2. **Программирование боевой логики игрового персонажа**

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

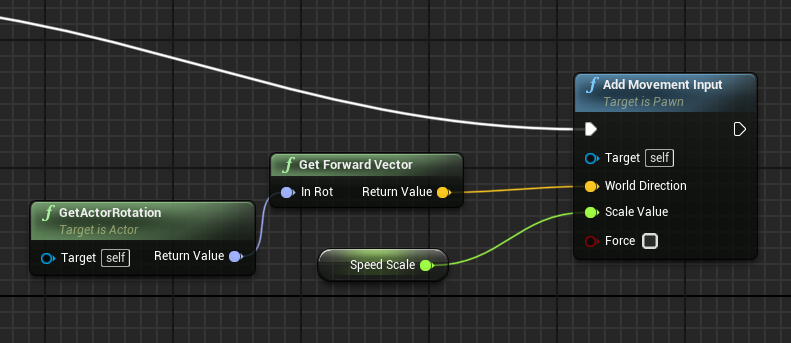
Для передвижения в различные стороны персонажем под управлением игрока, происходит захват нажатия клавиш WASD, отвечающих за соответствующие стороны. Удержание данных клавиш приводит к постоянному движению персонажа в соответствующую сторону, в зависимости от того, куда смотрит камера. Как только клавиша 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 12 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 12 – Функциональный блок Blueprint для передвижения персонажа

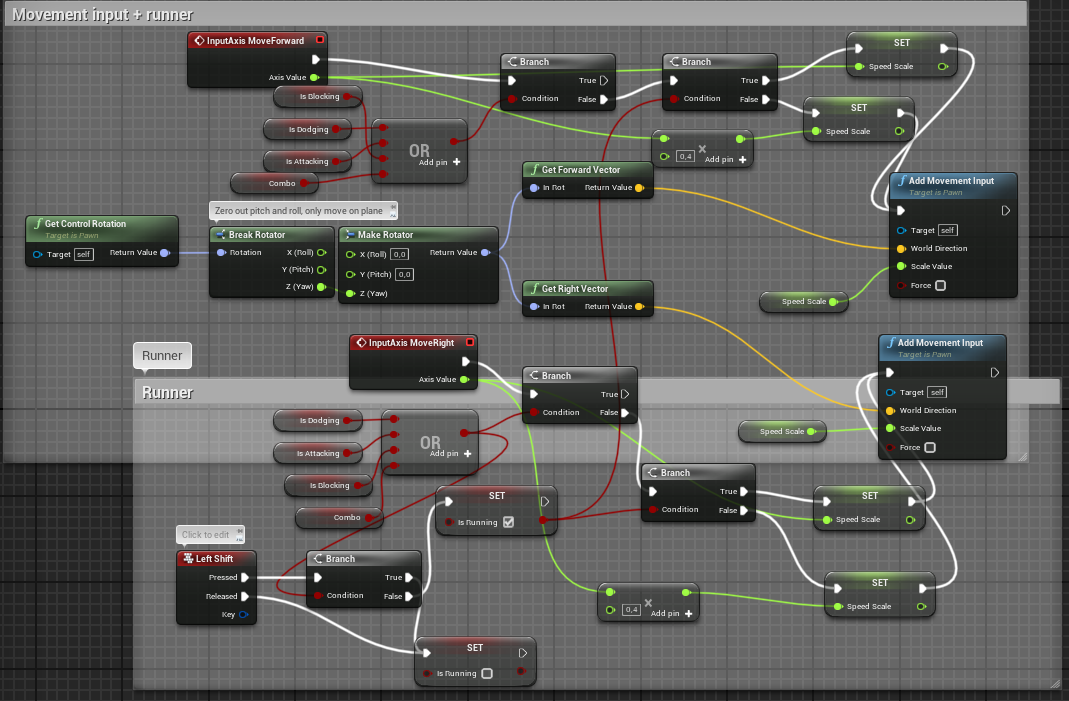
Если же при уже нажатой клавиши из списка WASD дополнительно удерживать клавишу Shift, то персонаж из медленного движения переходит на бег с большой скоростью, булева переменная Is\_running в родительском классе, отвечающая за индикацию переключения в состояние бега, станет “истиной”, а переменная скорости возрастет. Простое нажатие клавиши Shift, без указания направления движения с помощью WASD, не приведет к изменению состояния персонажа. На рисунке 13 изображен аналог алгоритма в виде графа событий Blueprint.

Рисунок 13 – EventGraph реализующий аналог алгоритма движения персонажа: ходьба и бег

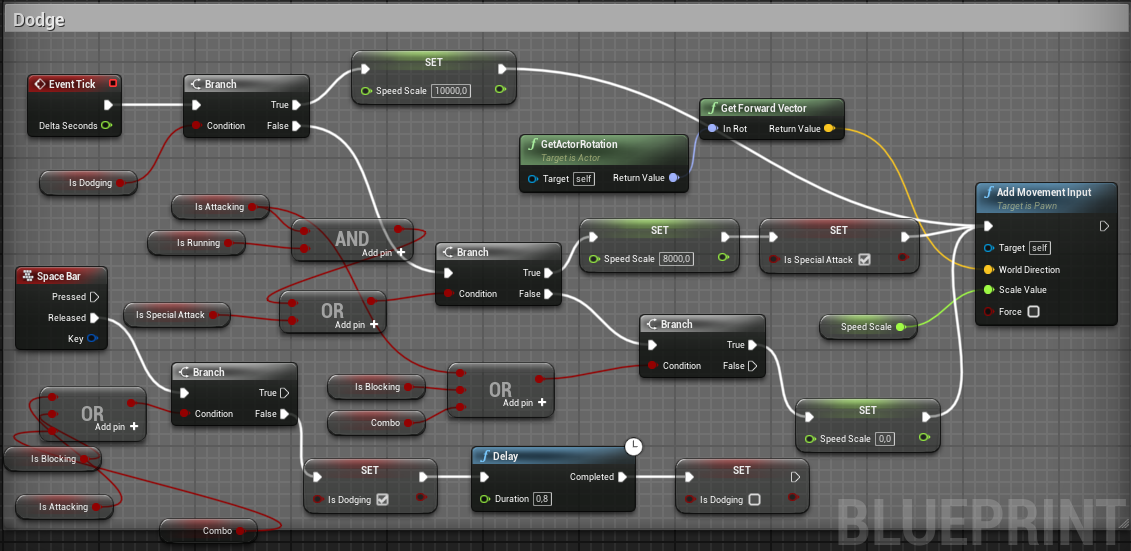
Для использования переката, используется клавиша Space (на рисунке 14 представлен аналог алгоритма в виде графа событий Blueprint). Направление переката зависит от поворота персонажа по оси Z (получается путем применения функции getActorRotation()). После нажатия персонаж перемещается на определенное 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 14 – EventGraph реализующий аналог алгоритма переката

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

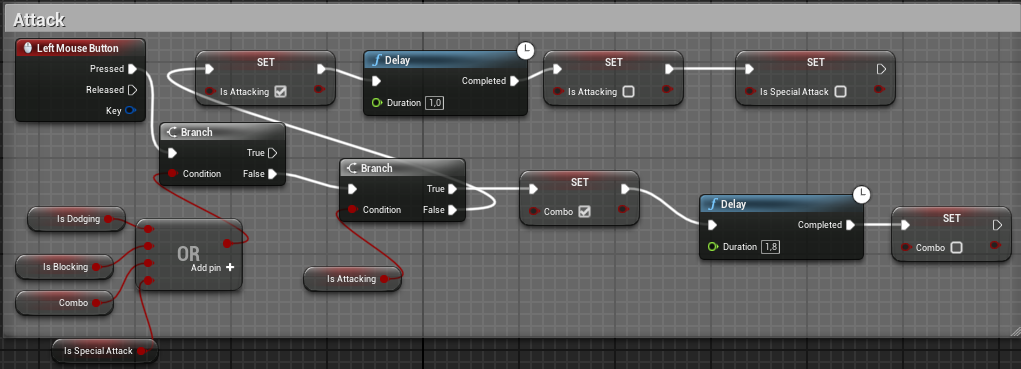
Для нанесения удара используется левая клавиша мыши (весь алгоритм нанесения ударов представлен в виде аналогичного графа событий Blueprint на рисунке 15). Далее начинается анимация удара, в процессе удара никакие перемещения или иные действия невозможны. Начало удара характеризуется переключением переменной 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 3). После завершения анимации удара персонаж снова переходит в исходное состояние, открывая возможность для взаимодействия.

Рисунок 15 – 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 находиться 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.

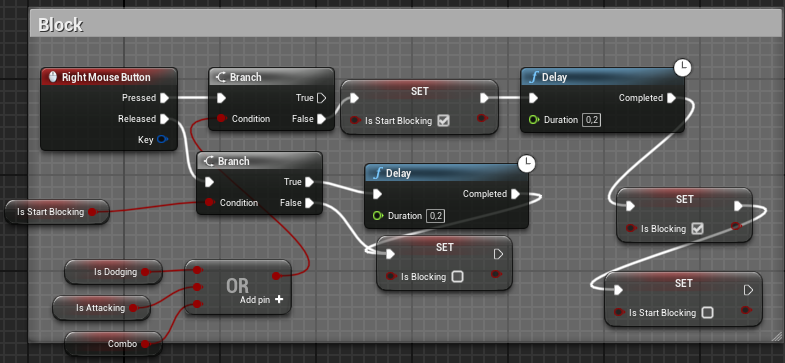
Для использования блока (на рисунке 16 представлен аналог алгоритма в виде графа событий Blueprint) используется правая клавиша мыши. Срабатывает анимация блока и переменная Is\_block становиться “истиной”. Если же в момент блока противник начинает атаку в сторону персонажа игрока, и они повернуты друг к другу лицом, то срабатывает дополнительная анимация, показывающая получение удара; но урона здоровью игрока не наноситься.

Рисунок 16 – EventGraph реализующий аналог алгоритма блокировки

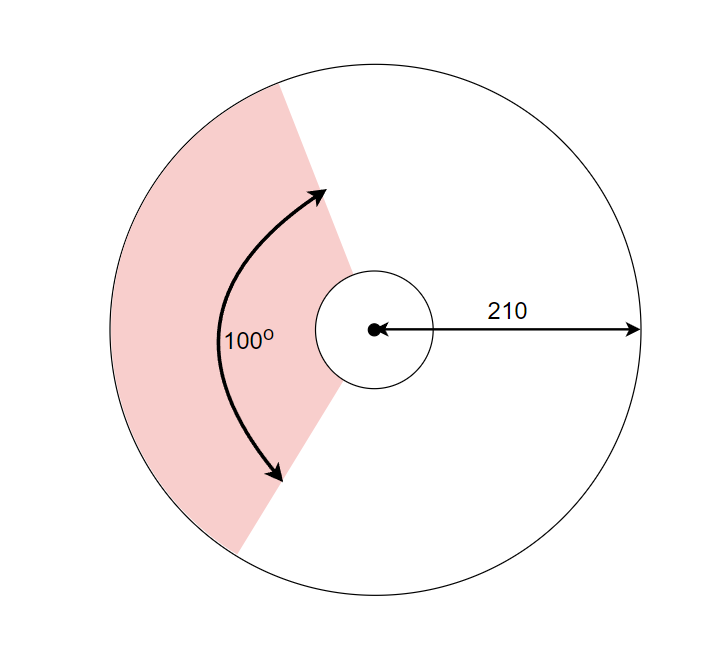
Если противник начинает атаку, и персонаж игрока находится в зоне получения урона (схема данной зоны изображена на рисунке 17), то персонаж переходит в состояние “получение урона”, проигрывается соответствующая анимация и персонаж теряет некоторое кол-во здоровья. Если же здоровья стало меньше или равным 0, то персонаж игрока уничтожается и вся карта перезагружается.

Рисунок 17 – Схема зоны вокруг противника, где персонаж может получить урон

### 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 управляется программным кодом, и, хотя его действия сильно зависят от действий персонажа игрока, игрок не может отдавать противнику команды напрямую.

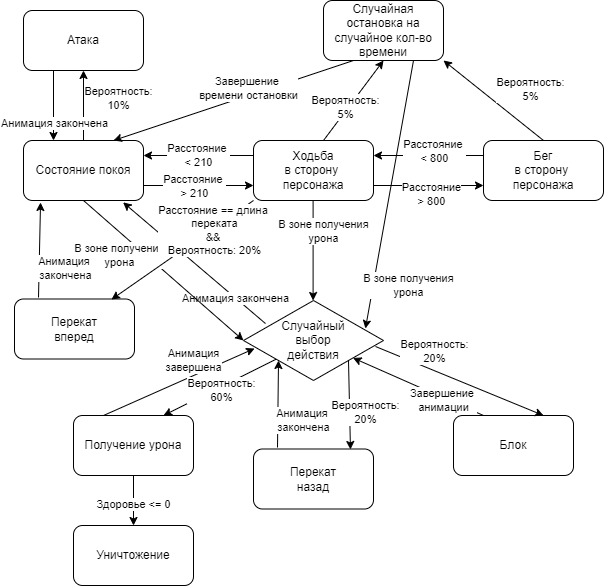
Как и персонаж игрока, противник обладает теми же возможностями в движении и бою: ходьба, бег, перекаты, удар, блок. Также есть и особые состояния, все они изображены на графе состояний на рисунке 18. Однако, если движение противника строго определено: всегда направлено на сближение с персонажем игрока, то боевые действия - в основном случайные события.

Рисунок 18 – Диаграмма состояний персонажа-противника

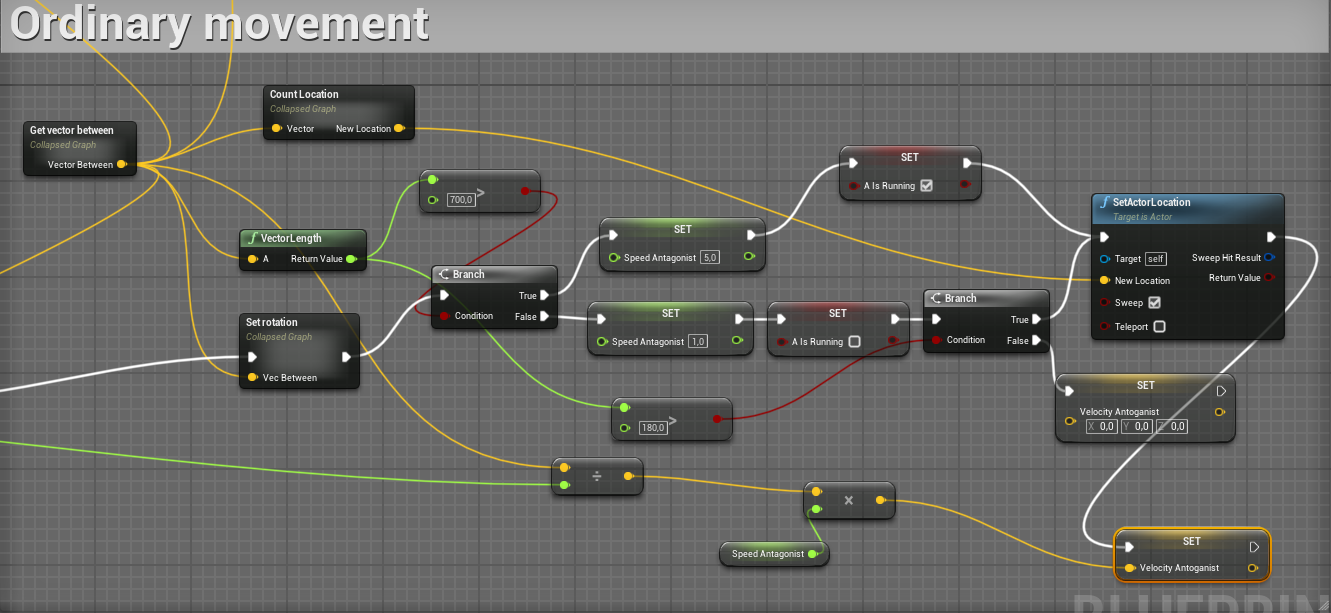
Движение противника практически всегда направлено в сторону персонажа игрока (аналог алгоритма передвижения представлен на рисунке 19). Переход с бега на ходьбу и наоборот зависит лишь от расстояния между двумя персонажами: если оно больше 800, то противник переходит на бег(переключение переменной Is\_running 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 19 – 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.

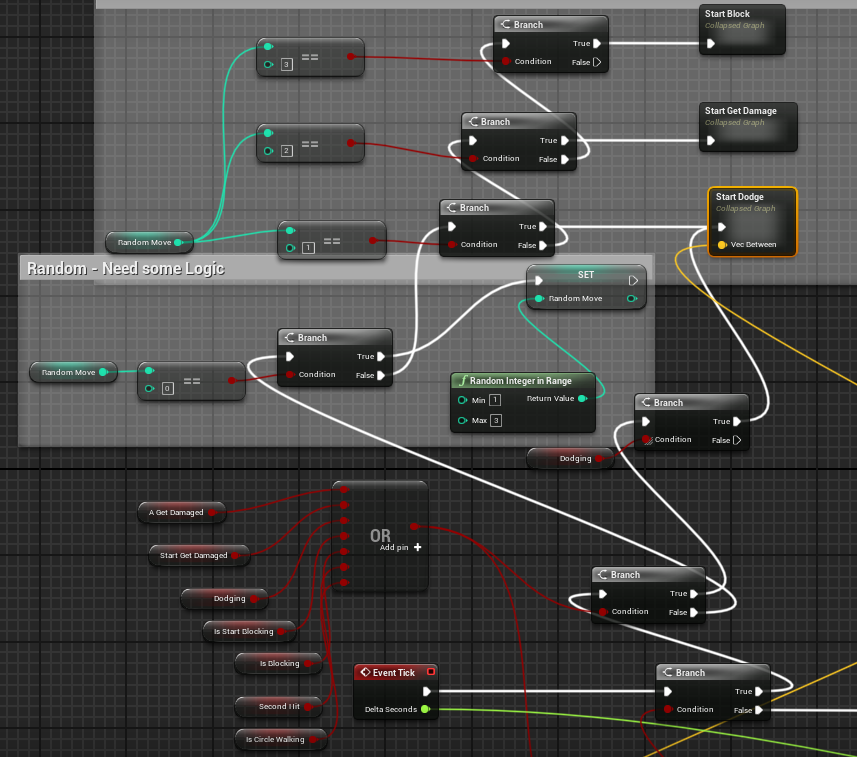
Агрессивные и защитные действия противника начинают действовать только когда персонаж находится на расстоянии менее 210 от противника. Случайные действия в ответ на атаку персонажа игрока представлены на рисунке 20.

Рисунок 20 – 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.

## ЗАКЛЮЧЕНИЕ

В процессе данной работы были исследованы различные игровые боевые системы, разработан собственный аналог видеоигровой боевой системы.

Используя движок 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 управляется 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. Обучающий курс “Вводный курс по анимации в UE” [Электронный ресурс] // YouTube. URL: [https://www.youtube.com/playlist?list=PLzn9qrbm2X8OuNjFPw 2rIj8O1h1Kt8JX0](https://www.youtube.com/playlist?list=PLzn9qrbm2X8OuNjFPw2rIj8O1h1Kt8JX0). (Дата обращения: 28.02.2024)
2. Обучающий курс “Продвинутый курс по анимации в UE” [Электронный ресурс] // YouTube. URL: [https://www.youtube.com/playlist?list=PLUi8nuTU EtTvTb8Wk6cBfvw8IEFAPZYqd](https://www.youtube.com/playlist?list=PLUi8nuTUEtTvTb8Wk6cBfvw8IEFAPZYqd). (Дата обращения: 03.03.2024)
3. Обучающий курс “Основы управление анимацией из C++” [Электронный ресурс] // YouTube. URL: <https://www.youtube.com/watch?v=ftjflBJwFAU>. (Дата обращения: 03.03.2024)
4. Public asset library [Электронный ресурс] // Poly Heaven. URL: <https://polyhaven.com/>. (Дата обращения: 22.11.2023)
5. Unreal Engine 4 Documentation [Электронный ресурс] // Unreal Engine Documentation. URL: <https://docs.unrealengine.com/>. (Дата обращения: 27.11.2023)