

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS

Исследование и разработка метода процедурной анимации фехтования боевым цепом

Обучающийся / Student Левин Даниил Михайлович

Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр

Группа/Group J4223

Направление подготовки/ Subject area 09.04.03 Прикладная информатика

Образовательная программа / Educational program Технологии разработки
компьютерных игр 2022

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Магистр

Руководитель ВКР/ Thesis supervisor Карсаков Андрей Сергеевич, кандидат технических
наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная
категория "ординарный доцент")

Консультант/ Consultant Берзон Давид Ильич, ШРВ, инженер, неосн по совм.

Обучающийся/Student

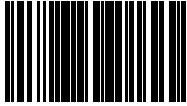
Документ подписан	
Левин Даниил Михайлович	
22.05.2024	

(эл. подпись/ signature)

Левин Даниил
Михайлович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Карсаков Андрей Сергеевич	
22.05.2024	

(эл. подпись/ signature)

Карсаков
Андрей
Сергеевич

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Левин Даниил Михайлович

Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр

Группа/Group J4223

Направление подготовки/ Subject area 09.04.03 Прикладная информатика

Образовательная программа / Educational program Технологии разработки компьютерных игр 2022

Язык реализации ОП / Language of the educational program Русский

Квалификация/ Degree level Магистр

Тема ВКР/ Thesis topic Исследование и разработка метода процедурной анимации фехтования боевым цепом

Руководитель ВКР/ Thesis supervisor Карсаков Андрей Сергеевич, кандидат технических наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная категория "ординарный доцент")

Консультант/ Consultant Берзон Давид Ильич, ШРВ, инженер, неосн по совм.

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

Основные вопросы, подлежащие разработке / Key issues to be analyzed

Цель: автоматизировать процесс создания анимации фехтования боевым цепом при помощи алгоритмов

процедурной генерации.

Задачи:

- изучить алгоритмы процедурной генерации, применимые для создания анимации;
- проанализировать существующие инструменты процедурной анимации;
- разработать метод процедурной генерации анимации;
- апробировать полученные результаты.

Форма представления материалов ВКР / Format(s) of thesis materials:

-текст ВКР в формате pdf,

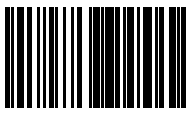
-презентация

Дата выдачи задания / Assignment issued on: 22.02.2024

Срок представления готовой ВКР / Deadline for final edition of the thesis 24.05.2024

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Карсаков Андрей Сергеевич	
03.05.2024	

(эл. подпись)

Карсаков
Андрей
Сергеевич

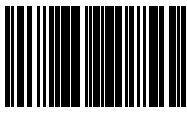
Задание принял к
исполнению/ Objectives
assumed BY

Документ подписан	
Левин Даниил Михайлович	
05.05.2024	

(эл. подпись)

Левин Даниил
Михайлович

Руководитель ОП/ Head
of educational program

Документ подписан	
Карсаков Андрей Сергеевич	
22.05.2024	

(эл. подпись)

Карсаков
Андрей
Сергеевич

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Левин Даниил Михайлович
Факультет/институт/кластер/ Faculty/Institute/Cluster школа разработки видеоигр
Группа/Group J4223
Направление подготовки/ Subject area 09.04.03 Прикладная информатика
Образовательная программа / Educational program Технологии разработки компьютерных игр 2022
Язык реализации ОП / Language of the educational program Русский
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Исследование и разработка метода процедурной анимации фехтования боевым цепом
Руководитель ВКР/ Thesis supervisor Карсаков Андрей Сергеевич, кандидат технических наук, Университет ИТМО, школа разработки видеоигр, доцент (квалификационная категория "ординарный доцент")
Консультант/ Consultant Берзон Давид Ильич, ШРВ, инженер, неосн по совм.

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

Разработка алгоритма, подходящего для создания анимации фехтования боевым цепом

Задачи, решаемые в ВКР / Research tasks

Изучение стилей фехтования боевым цепом; Анализ существующих методов процедурной анимации; Анализ алгоритмов обратной кинематики; Написание алгоритма процедурной анимации; Анализ моделей и определение параметров анимации; Определение зависимостей и вывод формул на основе определенных параметров; Анализ и подбор программного обеспечения; Программная реализация в соответствии с составленным алгоритмом, формулами и определенными параметрами; Тестирование и оценка итогового результата.

Краткая характеристика полученных результатов / Short summary of results/findings

В ходе выполнения работы разработана и реализована система процедурной анимации фехтования боевым цепом. Были выполнены все поставленные задачи и получены необходимые результаты.

Обучающийся/Student

Документ
подписан

Левин Даниил
Михайлович



Левин Даниил

Руководитель ВКР/
Thesis supervisor

22.05.2024	
------------	--

(эл. подпись/ signature)

Михайлович

(Фамилия И.О./ name and surname)

Документ подписан	
Карсаков Андрей Сергеевич	
22.05.2024	

(эл. подпись/ signature)

Карсаков
Андрей
Сергеевич

(Фамилия И.О./ name and surname)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	8
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	11
1.1 Анализ методов процедурной анимации	11
1.2 Биомеханика движений человека	19
1.3 Алгоритмы обратной кинематики	21
1.3.1 Алгоритм CCDIK.....	23
1.3.2 Алгоритм FABRIK	24
1.4 Анализ боевых стилей и оружия	25
1.4.1 Примитивный бой цепом.....	26
1.4.2 Кусари-дзюцу.....	27
1.4.3 Кусаригама-дзюцу.....	28
2 ВЫБОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РЕАЛИЗАЦИИ.....	31
3 РАЗРАБОТКА АЛГОРИТМА ПРОЦЕДУРНОЙ АНИМАЦИИ	34
3.1 Боевые стойки.....	34
3.2 Разработка алгоритма	36
3.3 Расчёт траектории движения руки	38
3.4 Расчёт движения туловища	39
3.5 Расчет движения ноги	41
3.6 Подбор алгоритмов обратной кинематики для генерации различных типов движений.....	44
4 РЕАЛИЗАЦИЯ АЛГОРИТМА ПРОЦЕДУРНОЙ АНИМАЦИИ	47
4.1 Движение руки.....	49
4.2 Движение туловища	50
4.3 Движение ноги.....	51
4.4 Симуляция движения цепи.....	52
5 ТЕСТИРОВАНИЕ И ОЦЕНКА РЕАЛИЗОВАННОЙ СИСТЕМЫ	54
5.1 Определение критериев тестирования	54
5.1.1 Тестирование алгоритма на мощном оборудовании	54
5.1.2 Тестирование алгоритма на слабом оборудовании	58
5.1.3 Натуральность генерируемых движений.....	63
5.2 Перспективы развития решения	64

ЗАКЛЮЧЕНИЕ	65
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	67

ВВЕДЕНИЕ

При разработке компьютерных игр используется большое количество ассетов, которые включают в себя 3D-модели и анимации для них. Однако, их количество необходимое для проекта, зачастую может значительно превышать количество используемых в нем объектов, так как для одной 3D-модели требуется большое количество анимированных движений.

Анимация персонажей, ключевой процесс создания игровых приложений, наравне с созданием механик и уровней. Сама процедура анимации – очень комплексная задача, включающая в себя множество мелких деталей, которые стоит учесть при создании состояний, движений и их последовательности. Для получения убедительного результата необходимо смоделировать множество различных аспектов человеческого поведения. Поведение и внешний вид виртуальных персонажей должны быть узнаваемо человеческими, хотя доскональная реалистичность не является необходимой. Люди умеют распознавать движения и человекоподобное поведение, поэтому действия и внешний вид виртуального персонажа должны приблизительно соответствовать ожиданиям зрителя-человека. Это означает, что движения персонажа должны быть не только естественными, но и соответствовать ситуации, например, правильная реакция на определенные стимулы или раздражители. [1]

Основным методом анимации является – анимация по ключевым кадрам (Keyframes). При её создании указывается начальное и конечное положение в кадрах, а положение промежуточных кадров вычисляется программой и создает путь. Этот процесс более долгий и кропотливый и требует большого опыта для получения реалистичности движения. Так же часто используется процедурная анимация (procedural animation), которая автоматически генерирует анимацию в реальном времени, используя базовые настройки и ограничения программы. [2] Еще одним способом является технология «Motion capture», позволяющая скелетом модели сделать движения реального актера, на котором закреплены оптические маркеры или

магнитные сенсоры. Данная технология, несмотря на высокую стоимость позволяющая получить качественную анимацию в краткие сроки, стала основным инструментом трехмерного анимирования с момента своего успешного введения в широкую практику кино- и игровой индустрии. [3]

Каждый из типов имеет разные методы создания, что разбивает их на дополнительные разделы. Не смотря на обилие вариаций при создании движений в боевых сценах с помощью процедурной генерации, предпочтение отдаётся именно анимации по ключевым кадрам и motion capture, которую используют в высокобюджетных проектах. Оба этих варианта довольно затратные по времени и ресурсам, поэтому стоит остановиться именно на процедурной анимации.

На сегодняшний день известно мало игр, в которых методы процедурной анимации в полной мере использовались бы для генерации движений персонажей, участвующих в ближнем бою. Чаще всего, в играх с помощью процедурной генерации воссоздают перемещение персонажей, физическое взаимодействие с окружением, однако, боевые системы все также реализуют с использованием стандартных методов создания анимации. В качестве процедурной генерации ударов в играх зачастую используют физическую анимацию, которая в свою очередь не позволяет получить правдоподобные движения, так как сами удары больше напоминают неконтролируемые выбрасывания рук или ног в направлении удара.

При использовании оружия ближнего боя, также наблюдается сильная недостоверность движений. Персонаж взаимодействует с ним неестественно, будто бы орудует мечом или кнутом. Также в большинстве проектов используют одинаковые анимации использования разного типа оружия, что тоже ставит под удар реалистичную составляющую боевых сцен.

Использование цепного оружия в игровой индустрии довольно распространено, но его проработке и разнообразию очень часто уделяется мало времени, ввиду дороговизны анимирования как самих частей оружия, таких как цепь и груз, так и самих движений персонажа.

Основываясь на вышесказанном, можно сделать вывод о том, что процедурная анимация является полезным инструментом, позволяющим решить проблему создания большого количества анимированных движений, заодно предоставляя возможность к производству более уникальных и разнообразных анимаций. Таким образом, рассматриваемая тема является достаточно актуальной, так как на данный момент было разработано ни одного похожего решения и структурированного подхода к созданию динамической системы процедурно-генерируемой анимации фехтования персонажа боевым цепом в ближнем бою, который позволил бы получить нужный результат, без использования огромного количества исходных данных и позволяющий предотвратить дополнительные затраты на их хранение и обработку.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ методов процедурной анимации

К одному из наиболее перспективных видов компьютерной анимации можно отнести процедурную анимацию, позволяющую задавать параметры движения в кадре и создавать сложные системы взаимодействия предметов, сил, материалов, освещения с заданными свойствами. Процедурная анимация позволяет генерировать эти процессы в реальном времени, изменяя параметры настроек. Эти черты отличают процедурную анимацию от предопределенной, которая вынуждает дизайнера вручную настраивать каждый кадр. [4]

Отличительными чертами процедурной анимации являются:

- экономичность создания;
- небольшой размер файла;
- большое количество разнообразных настроек, позволяющих до бесконечности изменять полученный результат;
- масштабируемость изображения до необходимого размера.

К недостаткам процедурной анимации относятся:

- высокие требования к техническим характеристикам оборудования,
- возможная неестественность движений в сравнении с предопределенными методами анимации,
- техническая сложность создания многих процедурных анимаций.

Фактором, способствующим широкому использованию процедурной анимации, явилось создание «нодовой» системы (от слова node - узел), состоящей из связанных между собой с определенной закономерностью наборов параметров, задаваемых пользователем. На сегодняшний день эта система является неотъемлемым элементом 3D редакторов, позволяющих добиться максимально точного результата без обращения к языкам программирования. [5]

Однако, даже при всей универсальности и перспективности использования методов процедурной генерации движений, аниматоры очень не скоро откажутся от стандартных методов анимации. [6]

На сегодняшний день существует несколько видов процедурной трехмерной анимации:

- вертексная (vertex - вершина) – анимация вертексов трехмерной сетки объекта, перемещение их в пространстве;
- объектная – анимация передвижения 3D объекта в пространстве;
- скелетная анимация – анимация, основанная на анимационном скелете, который представляет из себя иерархическую связь из множества объектов (костей), положение каждого из которых зависит от своего иерархического предка.

Для анимации человека используют скелетную анимацию, потому что по своей функциональной природе данный метод очень похож на то, что мы привыкли видеть в реальном мире, так как можно провести простую аналогию с человеческим скелетом. [7] Анимационный скелет гуманоидных персонажей представлен на рисунке 1.1.

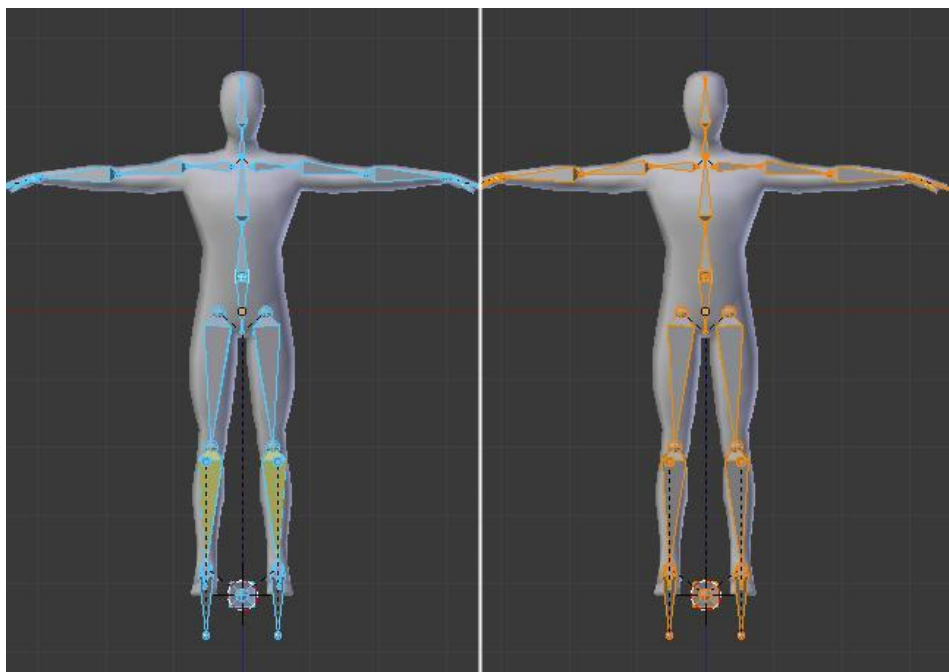


Рисунок 1.1 – Анимационный скелет гуманоидного персонажа

Процедурную скелетную анимацию можно разделить ещё на два вида:

- Заранее сгенерированная процедурная анимация
- Процедурная анимация, генерируемая в режиме реального времени (real-time animation)

Первый тип является анимацией, созданной заранее с помощью алгоритма и сохраненную в виде конечного файла, который затем импортируется в приложение и никак не видоизменяется в реальном времени. Второй тип, вместо того, чтобы использовать predetermined анимации, действия и реакции персонажей синтезируются в реальном времени. Эти анимации каждый раз являются особенными, неидентичными к предыдущим, даже если повторяется одна и та же сцена. К таким условиям можно отнести: неровности ландшафта, столкновение и взаимодействие с объектами и так далее. Процедурная анимация в реальном времени выглядит, как правило, более реалистично и правдоподобно по сравнению с другими типами анимации. [8]

На данный момент, в игровой индустрии больше всего используется метод процедурной анимации, основанный на физических законах, который называют физической анимацией.

Физическую анимацию можно разделить на несколько методов:

- Ragdoll-физика (физика повторяющая поведение тряпичной куклы);
- Симуляция твёрдого тела;
- Инверсивная (обратная) кинематика.

Все вышеперечисленные методы стараются повторять принципы симуляции движение человека. Более реалистичное поведение достигается при соблюдении биомеханики движения человеческого тела. Одним из самых простых, но эффективных способов создания процедурных анимаций является использование физики тряпичной куклы. Идея состоит в том, чтобы создать гуманоидное тело и соединить каждую конечность суставами, которые повторяют степени свободы, которые демонстрируют их реальные

аналоги. Просто полагаясь на физику твердых тел и совместные ограничения, можно смоделировать, как человек будет падать. Это не только экономит бюджет на “анимациях смерти”, но и позволяет создавать персонажей, которые реалистично падают и взаимодействуют со своим окружением. Такую задачу было бы практически невозможно воспроизвести с помощью предопределенного набора анимаций, независимо от того, насколько они точны и проработаны. [9] Пример работы Ragdoll-физики изображен на рисунке 1.2.



Рисунок 1.2 – Поведение персонажа под действием физики тряпичной куклы

Несмотря на удобство ragdoll-физики, она слабо подходит для создания анимаций прыжка или передвижения, так как рассчитана на симуляции именно падений. Для создания более корректного поведения персонажа, используется комбинированный метод, который включает в себя физику тряпичной куклы и симуляцию твёрдого тела для определённых частей тела, на которые приходится опора при движении.

Лучшим методом для детализации действий и движений персонажа является метод инверсивной кинематики. В этом методе создаются дополнительные ограничения, которые обуславливают, по каким траекториям могут перемещаться элементы скелета под действием физики или других взаимодействий. [10]

Благодаря выставленным ограничениям, движения скелета становятся более удобными для симуляции мышц, как это показано на рисунке 1.3.

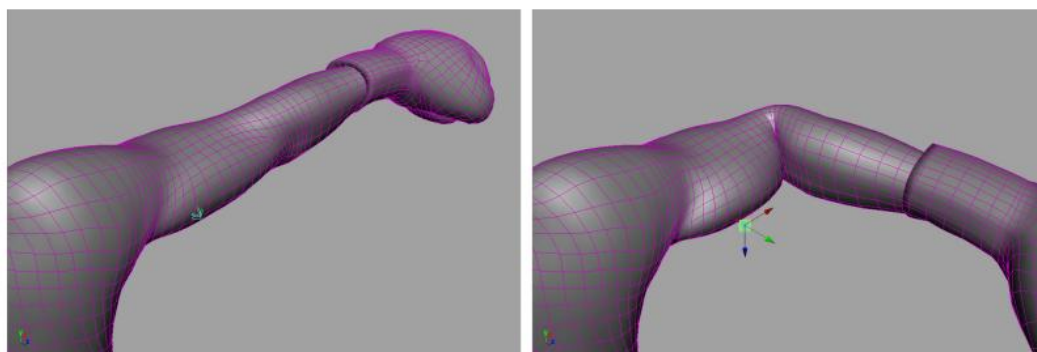


Рисунок 1.3 – Сокращение мышц плеча, при сгибании руки

Не смотря на возможность детализированной проработки мышечной структуры, в игровой индустрии чаще всего, её опускают, так как она становится нецелесообразной, при отсутствии детализированных объектов или создании игр, к примеру, для мобильных устройств. [11] Поэтому обратную кинематику чаще используют для создания реалистичных взаимодействий конечностей с поверхностями или объектами. К примеру, при создании процедурной анимации передвижения по наклонным поверхностям, с помощью инверсивной кинематики генерируется правильное положение стоп, как это показано на рисунке 1.4.



Рисунок 1.4 – использование инверсивной кинематики

На рисунке 4 персонаж изображен в двух состояниях, без использования инверсивной кинематики слева и использованием метода справа. Без обратной кинематики левая нога персонажа проходит сквозь скалу. При её использовании, нога должным образом помещается поверх него. Этот тип системы очень часто можно увидеть в современных играх. Хотя прохождение ногой по объектам, возможно, и не является большой проблемой для игрового процесса, это все же хороший метод для создания большего погружения и визуального качества в игре.

Помимо методов физической анимации в игровой индустрии активно применяются методы процедурной анимации с использованием нейросетевых технологий. Такой метод подразумевает почти полное отсутствие физических алгоритмов при анимации. Все генерируемые движения персонажа являются адаптивными и подстраиваются под разные условия. В статье [12] подробно говорится об одной из таких технологий, созданной на основе базы данных, содержащей в себе огромную библиотеку скелетных движений, захваченных при помощи технологии Motion Capture. Анимации созданные с использованием данного метода выглядят плавно и реалистично, а алгоритм самостоятельно адаптирует анимацию под условия окружающей среды. Персонаж может по-разному двигаться на различной поверхности, преодолевать препятствия и карабкаться. Пример работы данного решения представлен на рисунке 1.5.

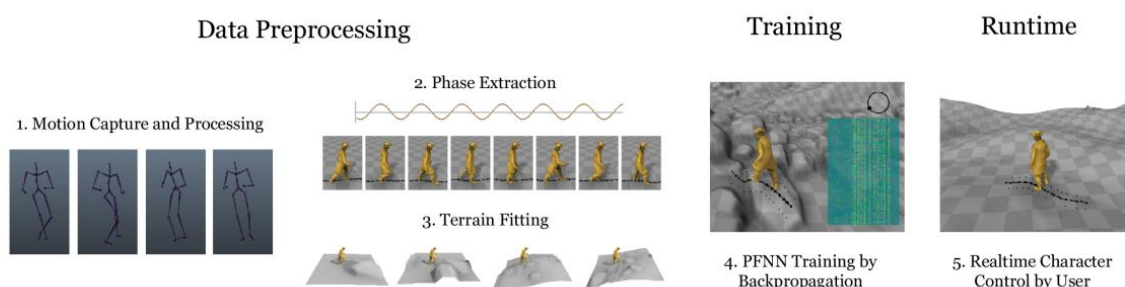


Рисунок 1.5 – Три этапа предлагаемой системы

На этапе предварительной обработки данных (слева) обрабатывается захваченное движение, и извлекаются управляющие параметры. Затем эти

данные сопоставляются с данными карты высот из виртуальных сред. Затем PFNN (Phase-Functioned Neural Network) обучается обратным распространением таким образом, что выходные параметры могут быть сгенерированы из входных параметров (средний). Наконец, во время выполнения, движение персонажа вычисляется в реальном времени с учетом пользовательского управления и геометрии окружающей среды (справа).

Помимо использования нейронных сетей для определения местности, в процедурной анимации их используют для создания универсальных анимаций взаимодействия с объектами. Одна нейросеть способна управлять сразу несколькими действиями в игре. Открывание дверей, перенос предметов, использование мебели. При этом она динамично изменяет положения рук, чтобы персонаж мог реалистично держать предметы разных размеров, садиться на разные по размеру стулья, а также пролезать в проходы разных размеров. [13] В статье [14] описывается подобный метод, созданный на основе технологии Neural State Machine. Учёные из Эдинбургского университета использовали заранее созданные анимации взаимодействия с предметами окружения, что позволило с помощью анализа нейронной сети подстроить движения под разные размеры и высоту объектов. Архитектура работы метода изображена на рисунке 1.6

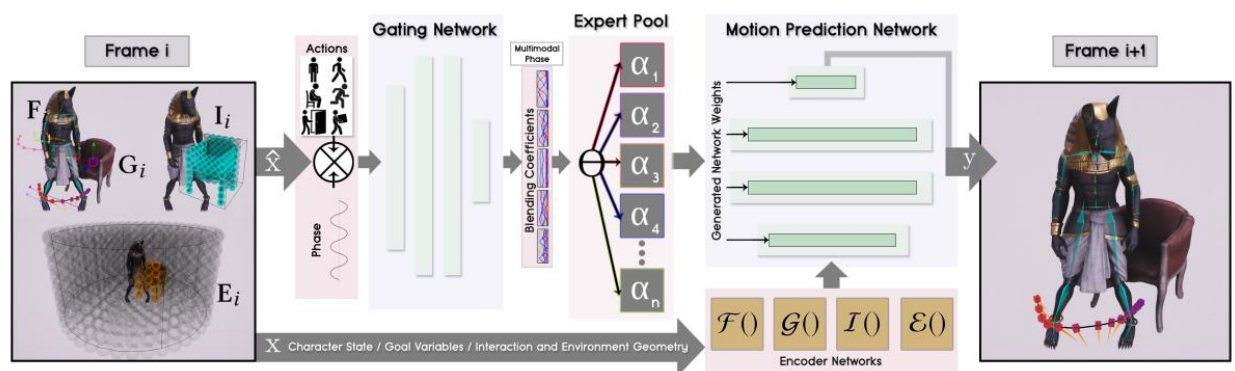


Рисунок 1.6 – Архитектура работы Neural State Machine for Character-Scene Interactions

Архитектура данной системы состоит из сети стробирования и сети прогнозирования движения. Строблирующая сеть принимает в качестве входных данных подмножество параметров текущего состояния и вектор

целевого действия для вывода экспертных коэффициентов смешивания, которые затем используются для генерации сети прогнозирования движения. Сеть прогнозирования движения принимает в качестве входных данных позу, переменные управления траекторией и параметры цели из предыдущего кадра и прогнозирует эти переменные для текущего кадра.

Оба метода использования нейронных сетей, являются довольно удобными и выдают качественный результат, но для их использования требуются огромные затраты времени на обучение и оборудование, а также большая библиотека данных, которые будут использоваться для анализа сетью. Помимо этого, обе сети не являются самостоятельными решениями и используют в процессе обучения технологию Motion capture, что делает проблематичным использование данного метода в небольших проектах и данной работе.

Помимо нейросетей в индустрии существует ещё один метод, который только начинает набирать популярность. Генерация анимаций на основе ключевых кадров. В статье [15] автор приводит обзор системы процедурной анимации, созданной аниматором Дэвидом Розеном из Wolfire Games в игре Overgrowth. Дэвид Розен реализовал систему процедурной анимации, которая генерировала абсолютно все анимации передвижения, которые есть в игре, используя всего лишь 13 ключевых кадров. Генерация происходила путем интерполяции между ключевыми кадрами по определенным формулам, с вычислением линейных и дуговых движений. Главным удобством метода является добавление новых анимаций в игру. Для этого необходимо, чтобы аниматор добавил лишь нескольких ключевых кадров, после чего процедурная генерация сама просчитает переключения между ними.

Несмотря на универсальность и качество анимаций при использовании метода, стоит отметить, что он не является полностью самостоятельным. Для некоторых условий, чтобы движения персонажа выглядели реалистичнее,

Розену было необходимо использовать инверсивную кинематику и систему Ragdoll.

На сегодняшний день количество различных методов создания процедурной анимаций гуманоидных персонажей, поражает своей обильностью и количеством. При этом как уже было подмечено, большая часть из них не являются самостоятельными и базируются на других технологиях, чтобы добиться большей реалистичности движений. При таком подходе сначала создается предопределенная анимация какого-либо действия, а затем в нее процедурно добавляются процедурные алгоритмы, которые вносят определённые динамические изменения. Таким образом, процедурная real-time анимация в сочетании с предопределенной анимацией позволяет получать более правдоподобные результаты.

Исходя из выше описанных методов было решено использовать смешанный подход анимации, состоящий из предопределённой анимации и методов инверсивной кинематики.

1.2 Биомеханика движений человека

Для наиболее точной анимации руки человека в бою требуется приблизить создаваемую нами модель к реальным анатомическим особенностям человека. Одна из таких особенностей - движение в суставах.

Каждый сустав имеет определённые ограничения, из-за которых большая часть может двигаться только в определенных направлениях. Для переноса этих ограничений и применения их в генерации анимаций необходимо определить их положения и значения. В механике существует термин, который называется степени свободы, он обозначает количество независимых переменных, используемых для полного описания механической системы. необходимых для полного описания механической системы. В нашем случае количество степеней свободы — это количество осей, вокруг которых может вращаться сустав. Самый первый сустав при проходе сверху вниз — это плечевой сустав. При проекции на шарниры согласно статье [16], он будет похож на шаровой шарнир, представленный из

двух звеньев: одно звено неподвижно, а второе вращается вокруг первого в любом направлении, а также способно вращаться вокруг своей оси. Таким образом положение свободного звена задается двумя осями, а вращение вокруг своей же оси - третьей. Следовательно, плечевой сустав имеет три степени свободы. Следующий сустав - локтевой. При все той же проекции на шарниры локтевой сустав будет подобен цилиндрическому шарниру - он способен вращаться только вокруг одной оси - оси цилиндра, соответственно имеет одну степень свободы. Последний сустав в руке - лучезапястный. Лучезапястный сустав может вращаться вокруг двух осей, как и плечевой, однако не может вращаться вокруг собственной, что говорит нам о наличии двух степеней свободы.

Подводя итоги определения степеней свободы всей руки (не считая пальцы, рука сжата в кулак) с закрепленным туловищем (неподвижное звено), рука принимается за цепь, состоящую из четырёх звеньев и трёх шарниров: туловище – плечевой сустав – плечо – локтевой сустав – предплечье – лучезапястный сустав – кисть. Степени свободы руки без учёта пальцев представлены на рисунке 1.7. Складывая степени свободы рассматриваемых суставов (плечевой сустав – три, локтевой сустав – одна, предплечье – одна, лучезапястный сустав – две степени свободы) получаем, что рука человека имеет семь степеней свободы.

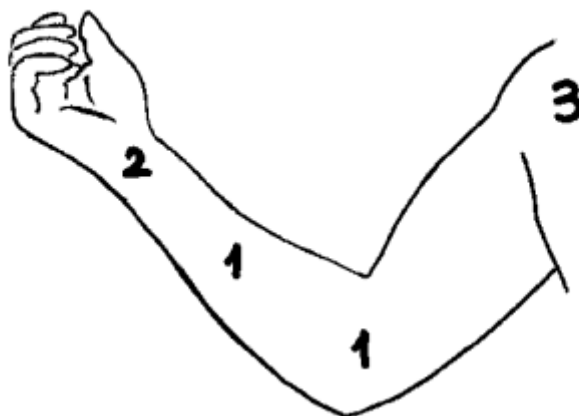


Рисунок 1.7 – Степени свободы руки без учёта движений пальцев

1.3 Алгоритмы обратной кинематики

Принимая во внимание выше обозначенную степень свободы, можно сделать вывод, что задание положения руки в пространстве является довольно непростой задачей, так как имеет множество возможных решений. Данная проблема лежит в области кинематики и может быть решена методами, пришедшими в анимацию из робототехники, а именно методами прямой и обратной кинематики. Данные методы позволяют определить положения цепи связанных звеньев в пространстве. В случае с анимацией движений человека такой цепочкой из звеньев принимается иерархия костей конечностей анимационного скелета. Прямая кинематика (Forward Kinematic, далее FK) определяет положение конечного исполнительного органа цепи (end-effector) с учетом параметров шарниров. Это традиционный подход к анимации, который как правило характеризуется ручной установкой каждой кости в нужное положение. [17] FK отвечает на вопрос: какая поза будет получена, если установить суставы в определенное положение. Обратная кинематика (Inverse Kinematic, далее IK) работает противоположно прямой кинематике. Последнее звено цепи (end-effector) устанавливается в первую очередь, а затем, в зависимости от этого устанавливается остальная часть цепи, сохраняя при этом связь между звеньями и их длину. Таким образом, IK позволяет определить положение всех звеньев цепи в пространстве зная позицию конечной точки. IK отвечает на вопрос: в какие положения надо поместить суставы, чтобы получить определенную позу. Для анимации ударных движений больше всего подходит именно IK анимация, так как в данном случае известна конечная точка движения последнего звена руки – точка нанесения удара.

На сегодняшний день существует множество алгоритмов обратной кинематики и их модификации:

- FABRIK (Forward and Backward Reach Inverse Kinematics) – метод прямого и обратного следования,
- CCDIK (Cyclic Coordinate Descent) – циклический координатный спуск,
- Jacobian Transpose (J.Transpose) – транспонирование Якоби
- Jacobian DLS (Damped Least Squares) – метод наименьших квадратов с демпфированием,
- Jacobian Pseudo-inverse DLS (SVD-DLS) Damped Least Squares with Singular Value Decomposition – псевдообратный затухающий метод наименьших квадратов.

ИК алгоритмы разделяют на итерационно-эвристические (CCDIK, FABRIK) и на алгоритмы Якобиана (J.Transpose, DLS, SVD-DLS). Алгоритмы Якобиана позволяют получить более сглаженные позы, однако, большинство этих подходов страдают от высокой вычислительной стоимости и сложных матричных вычислений. В статье [18] представлено сравнение работы данных алгоритмов. Результаты сравнения их быстродействия представлены в таблице 1.1

Таблица 1.1 – Сравнение быстродействия алгоритмов

Алгоритмы	Число итераций	Время, сек
FABRIK	15	0.01328
CCDIK	26	0.12359
J.Transpose	1311	12.98947
J.DLS	998	10.48501
J.SVD-DLS	808	9.29652

Как заметно по результатам сравнения FABRIK и CCDIK являются самыми быстрыми алгоритмами, поэтому далее будут рассмотрены только они.

1.3.1 Алгоритм CCDIK

Самым известным методом ИК является алгоритм циклического спуска по координатам (CCDIK), который на данный момент широко применяется в современной анимации. [19] CCDIK — это эвристический итерационный метод с низкими вычислительными затратами для каждого соединения на итерацию, который может решить задачу ИК без матричных манипуляций. Таким образом, он очень быстро формулирует решение. Главным недостатком алгоритма является создание изломов цепи. При этом данный алгоритм позволяет задавать ограничения вращения звеньев, что является весомым аргументом и именно поэтому CCDIK выбирают в качестве главного метода при создании реалистичных анимаций и симуляций биомеханики. Пример работы алгоритма представлен на рисунке 1.8.

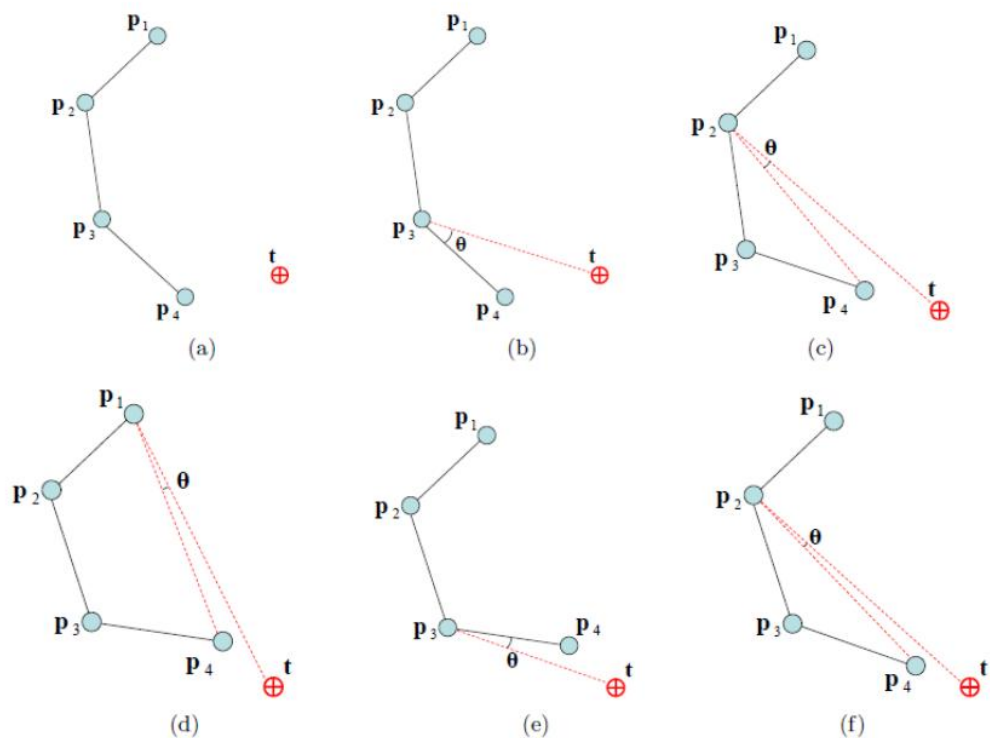


Рисунок 1.8 – Пример работы алгоритма CCDIK: (a) первоначальное положение манипулятора и мишени; (b) нахождение угла θ между конечным звеном (end-effector), шарниром p_3 и целевой точкой и поворот сустава p_3 на этот угол; (c) нахождение угла θ между конечным звеном, суставом p_2 и целевой точкой и поворот сустава p_2 и p_3 на этот угол; (d), (e) и (f) повторение всего процесса до тех пор, пока концевой эффектор не достигнет цели или подойдет достаточно близко

1.3.2 Алгоритм FABRIK

Алгоритм FABRIK является самым быстродействующим методом, среди упомянутых. Его часто сравнивают с CCDIK, поскольку оба являются довольно популярными алгоритмами. В отличие от CCDIK, FABRIK создаёт более сглаженные и натуральные движения среди эвристических методов, не генерируя лишние разрывы и изломы.

Сам алгоритм довольно похож на конкурента, однако вместо вращения используется перемещение. Находится угол между конечным эффектором и целью, далее кость, включающая конечный эффектор перемещается так, чтобы угол этой кости соответствовал найденному, а конечный эффектор находился в точке цели. Такая процедура повторяется до начала цепочки (так как начинается с конца), что приводит к смещению всех костей относительно их начальных позиций. Затем алгоритм делает то же самое, но в обратную сторону, благодаря чему начальное сочленение остается на месте. Прохождения алгоритма вперед и назад считается одной итерацией. Можно повысить количество итераций для достижения лучших результатов.

Алгоритм использует прямой и обратный итеративный подход, итерируя по цепи сначала в одном, потом в другом направлении, находя положение каждого сустава путем определения точки на линии [20]. Пример работы алгоритма представлен на рисунке 1.9.

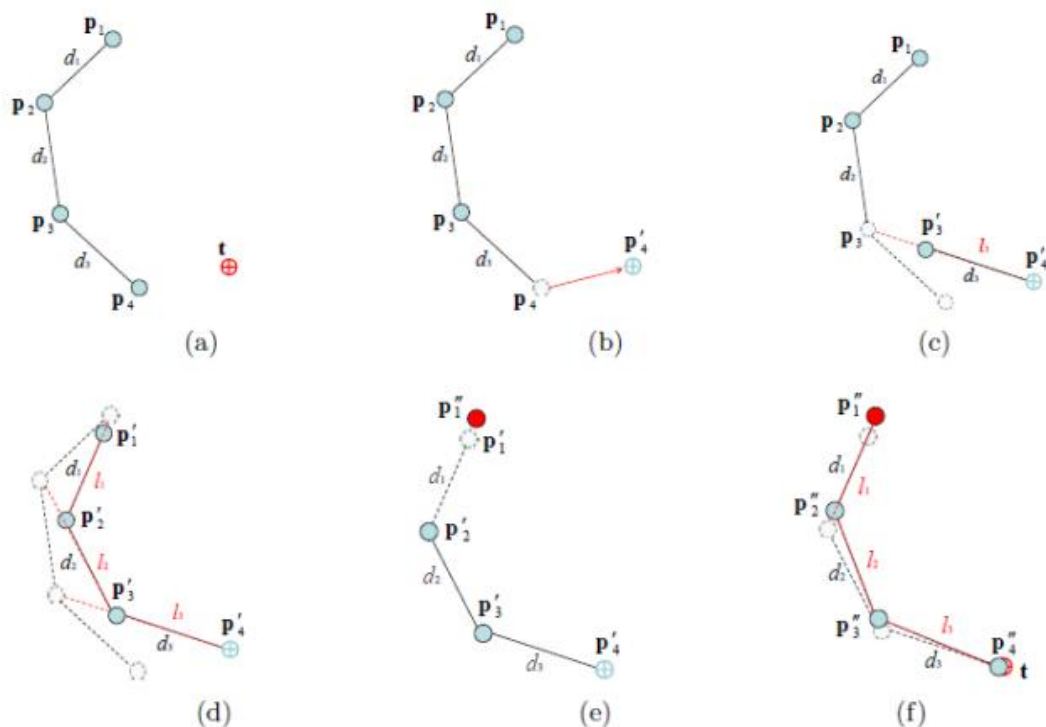


Рисунок 1.9 – Пример полной итерации FABRIK: (а) исходное положение манипулятора и целевой точки; (б) перемещение концевой эффектора p_4 в целевую точку; (с) нахождение шарнира p'_3 , лежащего на прямой l_3 , проходящей через точки p'_4 и p_3 , с расстоянием d_3 от точки p'_4 ; (д) продолжение алгоритма для остальных суставов; (е) второй этап алгоритма: перемещение корневой сустава p'_1 в исходное положение; (ф) повторение процедуры, в другом направлении, от основания к концевому звену. Алгоритм повторяется до тех пор, пока положение конечного звена не достигнет цели или не переместится достаточно близко.

1.4 Анализ боевых стилей и оружия

Среди всех боевых стилей, в которых используется цепное оружие, можно выделить три основных:

- Кусари-дзюцу
- Кусаригама-дзюцу
- Примитивный бой цепом

Каждый из этих боевых стилей, имеет уникальные особенности, оружие, которое используется в процессе и другие нюансы, которые будут рассмотрены дальше.

1.4.1 Примитивный бой цепом

Примитивный бой цепом является самым распространённым вариантом использования цепного оружия.

Зародился стиль в средние века, когда появилась необходимость пробивать вражеский прочный доспех, огибая щит врага.

Основной задачей во время атаки являлось нанесение точного удара по корпусу противника. Зачастую у использующего цеп бойца, была всего одна попытка на это, после чего единственное, что могло помочь, это то, что иногда грузов было три, а не один. Малым щитом было сложно отражать удар такого оружия, а если и получалось, то у бойца с цепом был дополнительный шанс на повторение атаки, прежде чем отражающий поймёт, что произошло. [21]

Помимо того, что бой таким оружием, представлял из себя почти возможность одного шанса, главной проблемой являлось то, что груз предварительно нужно было раскрутить, для более эффективного нанесения удара. Поэтому со временем, бой цепом перешёл в иной формат. Большая часть столкновений, происходила в конных боях, где такое оружие показало себя значительно лучше. [22]



Рисунок 1.10 – Бой цепом в действии

Если разбирать движения человека во время боя цепом или кистенем, можно выделить несколько основных моментов:

- Ноги почти всегда находились в одной позиции;
- Выпад оружием производился с опорой на ведущую ногу;
- При нанесении удара, груз необходимо было раскручивать, поэтому основное движение руки приходилось на вращение предплечья;
- Вторая рука часто держала щит и поэтому почти не двигалась, защищая корпус бойца;
- При нанесении удара, даже если он блокировался, движение атакующей руки не прерывалось в виду специфики оружия.

1.4.2 Кусари-дзюцу

Кусари-дзюцу, как и любой другой японский боевой стиль состоит из огромного количества школ и различных техник, не смотря на это, большая часть сводится к одному принципу и виду оружия.

Для данного боевого стиля изначально использовалась веревка с камнем. С развитием кузнечного дела в Японии, верёвка сменилась на цепь, а камень на железное грузило. [23]

В основном, сам боевой стиль использовался городской стражей для подавления бунтов, обезоруживания и обездвиживание оппонента. Позже цепь стала обязательным оружием самураев, охранников главных ворот. Цепь оборачивалась вокруг пояса и в мгновение могла стать петлей на шее или нанести нокаутирующий удар. [24] Исходя из этого, движения бойца не являлись атакующими, скорее задачей было дождаться нападения и произвести контратаку.



Рисунок 1.11 – Контратака в момент удара при помощи кусари-дзюцу

Основываясь на этом, можно сделать следующие выводы относительно важных моментов движений персонажа при анимации:

- При произведении контратаки, используются обе руки, в которых удерживается цепь.
- Позиция ног сохраняется до момента контратаки.
- В момент контратаки используются приемы из единоборств, к примеру захват или бросок через кувырок.
- Учитывая, что основная задача заключается в опутывании рук оппонента, движения верхней части состоят из большого количества комплексных и мелких движений большого количества суставов.

1.4.3 Кусаригама-дзюцу

Развитие кусаригама-дзюцу началось еще с двенадцатого века, и вплоть до семнадцатого века оставалось одним из популярнейших боевых искусств феодальной Японии. Количество школ кусаригама-дзюцу исчислялось десятками, и развитие это вполне объяснимо эффективностью данного оружия – серпа на цепи, позволявшего нейтрализовать длинное самурайское и солдатское оружие тяжелым грузом. [25]

При использовании кусаригамы, боец мог метнуть цепь таким образом, чтобы та обмотала руку держащую оружие или же другую часть тела оппонента. Благодаря этому, у использующего серп появлялась возможность для того, чтобы подтянуть к себе врага или же при сильном натяжении цепи, наступив на неё выбить из равновесия. [26]

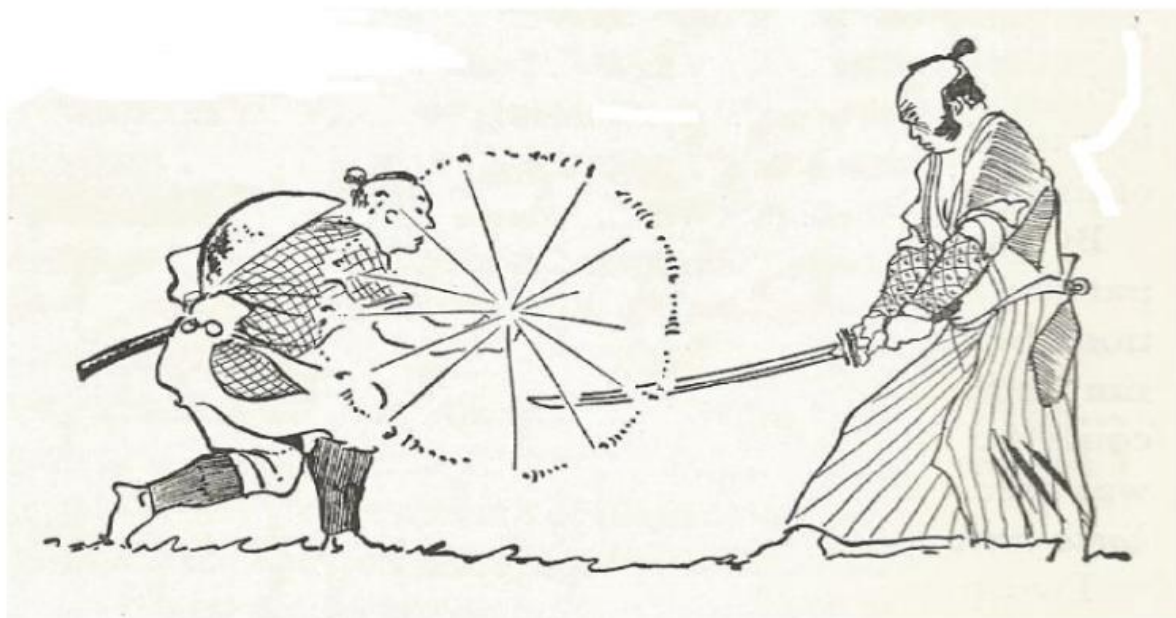


Рисунок 1.12 – Кусари-дзюцу. Приём Катава-Фури

Стоит отметить, что сам стиль боя является довольно разнообразным. Как как оружие состоит не только из цепи и древка, но и имеет лезвие, его применение не оканчивалось на защите или дистанционных атаках. Также допускалось использование как двух серпов соединённых одной длинной цепью, что делало стиль боя еще разнообразнее.

Основываясь на этом, можно сделать следующие выводы относительно движений персонажа при его анимации:

- Учитывая подвижность персонажа при использовании этого стиля боя, его реализация в виде процедурной анимации очень трудоёмка
- Помимо анимации движений, стоит учитывать поведение не только цепи и самого персонажа, но и серпа
- Ноги персонажа всегда находятся в активном состоянии
- При нанесении ударов, часто применяются элементы акробатики, состоящие из большого количества комплексных движений

По итогу анализа всех трех боевых стилей, было решено выбрать для следующей работы примитивный бой цепом, так как этот стиль более подходит для создания процедурной анимации движения персонажа, не затрачивая ресурсы алгоритмов на акробатические приемы или же прямую зависимость от атак противника.

2 ВЫБОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РЕАЛИЗАЦИИ

Для реализации процедурной анимации и получения наилучших результатов, необходимо выбрать подходящий для этого инструмент. Учитывая требования и специфику задачи, лучшим вариантом для создания инструментария, который обеспечит процедурную составляющую процесса анимации, является игровой движок. Его использование позволит не только упростить работу при разработке, но и не будет требовать дополнительных переносов результатов из одной программы в другую. Большая часть других программ по созданию трехмерных моделей не подойдет ввиду того, что реализация работы с изменением анимации в реальном времени там почти невозможна, либо же требует сохранения в отдельный файл и переноса с шансом ошибок.

Для того, чтобы выбрать конкретный игровой движок, необходимо провести сравнение среди наиболее популярных и доступных вариантов.

В статье [27] приведена статистика по наиболее популярным игровым движкам и их сравнение [28]. Исходя из представленных данных, наиболее часто используемыми на данный момент игровыми движками являются Unity 3D, Unreal Engine. Два этих движка практически равнозначны по своему функционалу, в частности в области процедурной анимации. Ряд статей [29, 30] проводит сравнение данных движков и выявляет их плюсы и минусы.

Основываясь на минусах и плюсах обоих движков приведённых в статье было решено для реализации алгоритма выбрать Unreal Engine 5, так как он обладает рядом преимуществ, такими как:

- Встроенная система визуального программирования Blueprint, которая облегчит работу с созданием рабочего алгоритма (рисунок 2.1);
- встроенная реализация алгоритмов обратной кинематики, описанных в предыдущем исследовании;
- Имеет встроенную систему создания анимаций, которая обладает продвинутыми инструментами и набором утилит для разработки сложных анимаций;
- Имеющийся опыт работы с движком.

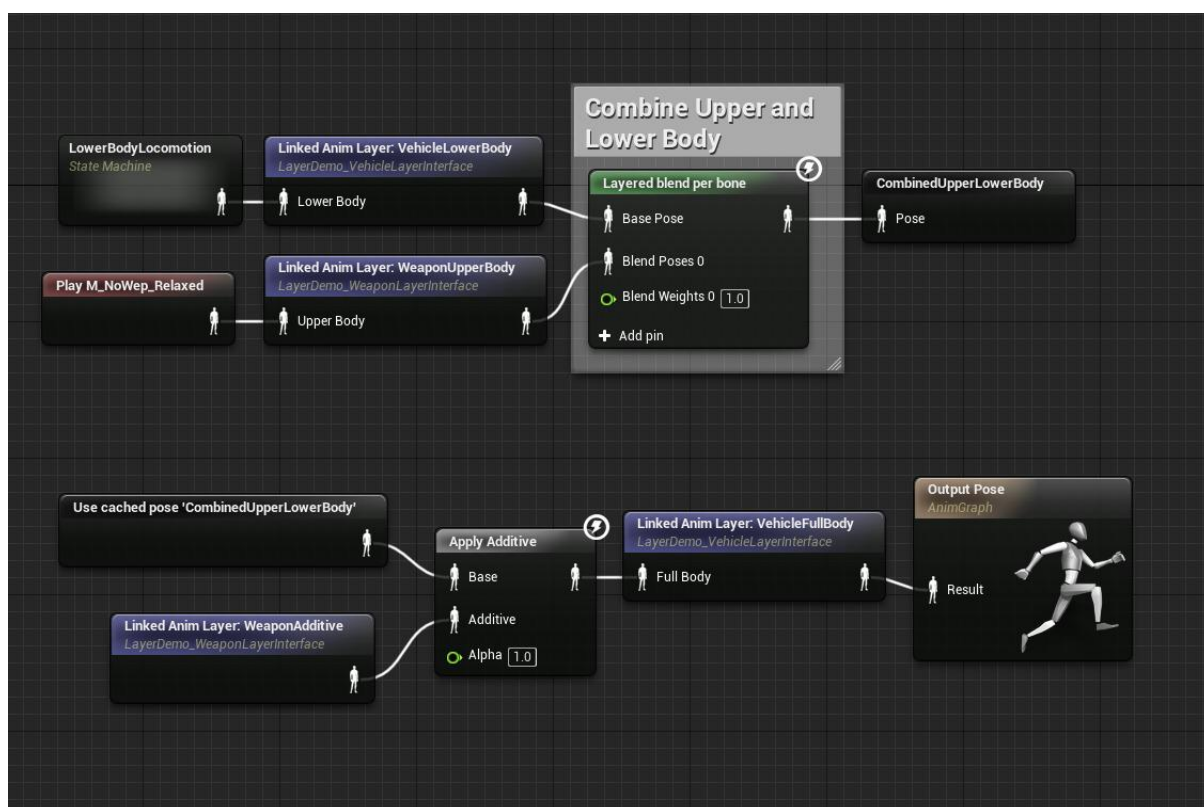


Рисунок 2.1 – Система Blueprint в движке Unreal Engine 5

Для прототипирования алгоритма будет использована стандартная модель гуманоидного персонажа из Unreal Engine 5 (рисунок 2.2), которая уже содержит в себе готовый скелет и риг, позволяющий создавать и применять любые анимации.

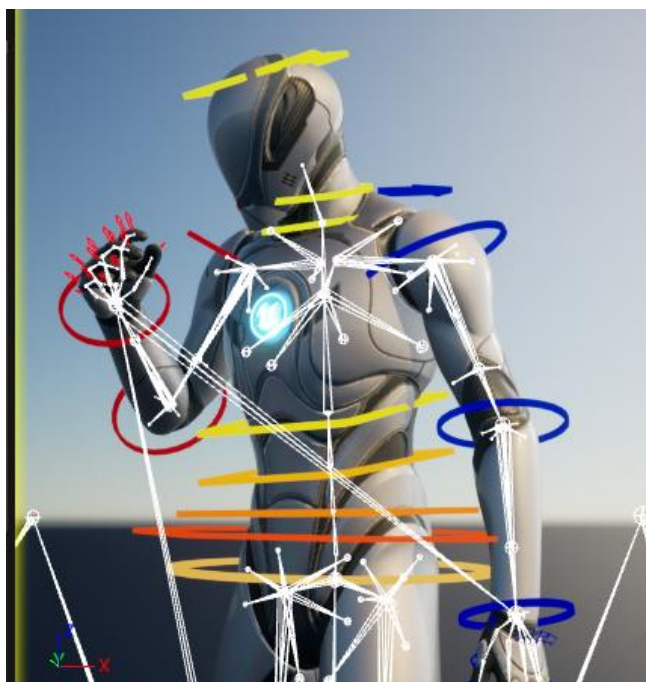


Рисунок 2.2 – Стандартная модель гуманоидного персонажа в Unreal Engine 5.1

При необходимости, инструментарий движка дает возможность быстрого переноса на любую другую модель, чей скелет в большинстве случаев совпадает с имеющимся у используемого примера.

3 РАЗРАБОТКА АЛГОРИТМА ПРОЦЕДУРНОЙ АНИМАЦИИ

Прежде чем приступать к разработке алгоритма, стоит выделить основную боевую стойку, которая будет использована при создании анимации.

Для этого, следует обратиться к записи тренировочного боя между двумя бойцами и проанализировать их стойки. [31]

3.1 Боевые стойки

При фехтовании оружием с цепью и грузом, боец находится в нескольких боевых стойках:

- Защитная стойка. Когда боец защищается от удара, он раскручивает груз, чтобы иметь возможность быстрой контратаки, так же этот прием позволяет обезоружить противника, если тот не использует оружие с цепью.



Рисунок 3.1 – Защитная стойка

- Атакующая стойка. При атаке, боец большую часть времени находится в состоянии замаха, для нанесения удара. Сам же удар происходит либо из-за спины, хлестким движением, либо простым боковым движением.



Рисунок 3.2 – Атакующая стойка

Для создания алгоритма и последующей реализации лучше всего подойдет атакующая стойка.

3.2 Разработка алгоритма

Тип движения анимируемого объекта будет зависеть от некоторых параметров. А именно: ввода игрока, от позиции персонажа, длины цепи у оружия и нахождения противника в пространстве.

Таким образом, при использовании атакующей стойки, алгоритм движения в момент нанесения удара будет выглядеть следующим образом:

- 1) Движение начинается из начальной позы, в момент которой персонаж наносит удар в зависимости от удаленности цели.
- 2) Проверяется дистанция от персонажа до противника методом Line Trace
- 3) Если в момент проверки противник находится дальше, чем расстояние максимальной досягаемости груза кистеня при вытянутой руке, но в зоне досягаемости, то происходит перемещение персонажа на шаг вперед, он же выпад и начинается фаза удара.
- 4) Если в момент проверки противник находится слишком близко, то происходит перемещение персонажа на шаг назад, он же отход и начинается фаза удара.
- 5) Если же в момент проверки противник уже находится в зоне нанесения удара, то персонаж сразу переходит в фазу удара.
- 6) После удара персонаж возвращается к начальной позе.

Первичная блок-схема алгоритма процедурной анимации удара цепом представлена на рисунке 3.3.

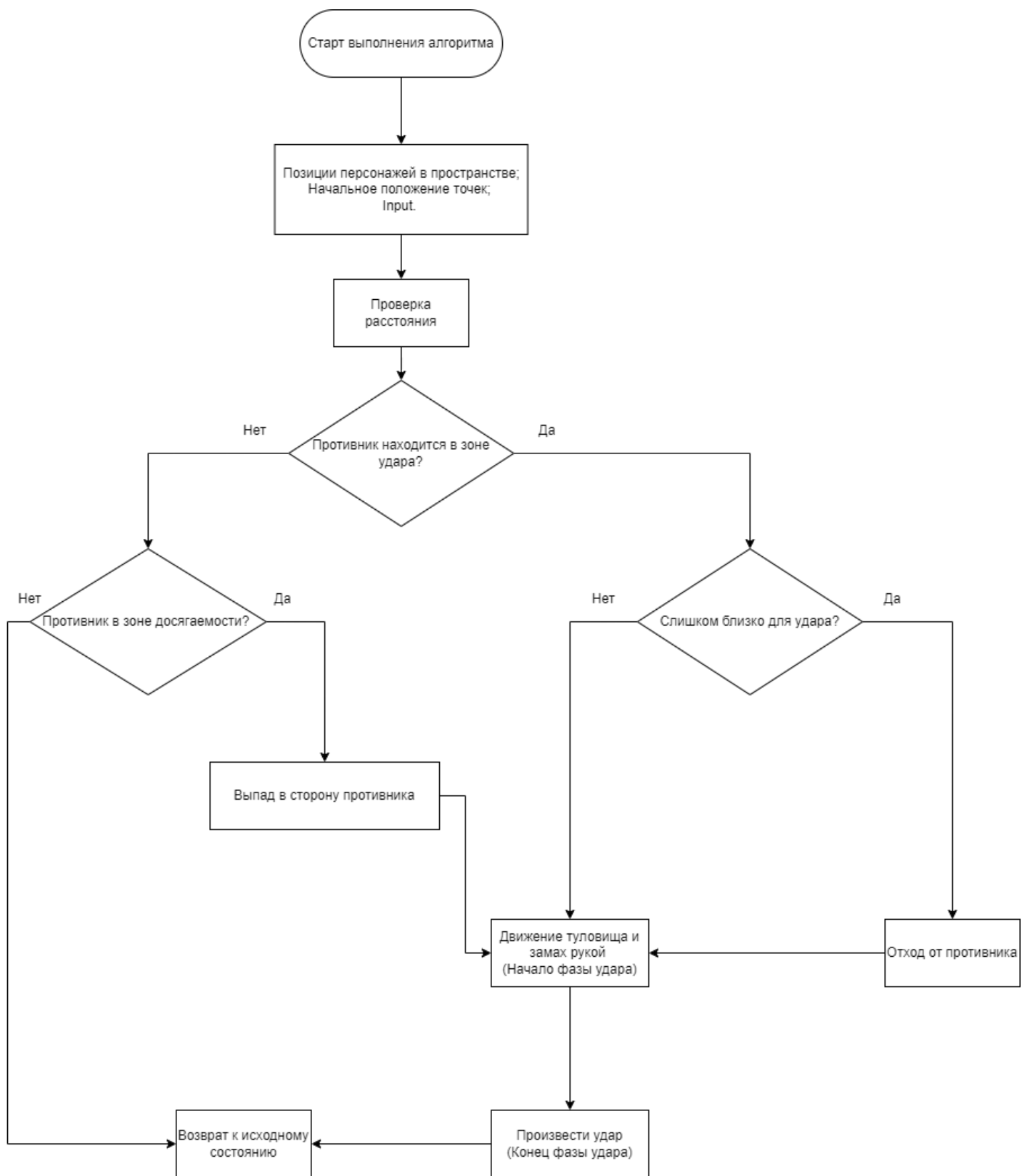


Рисунок 3.3 – Блок-схема алгоритма

3.3 Расчёт траектории движения руки

Для построения траектории движения руки в соответствии с составленным алгоритмом необходимо иметь как минимум три точки в пространстве:

- начальная точка (начальное положение руки),
- конечная точка удара (цель),
- промежуточная точка, определяющая тип удара.

Первая точка определяется начальным положением руки с учетом замаха. Определяется по формуле (3.1)

$$Sp = i \cdot r \cdot q, \quad (3.1)$$

где i - отклонение руки при замахе для нужной позиции удара; r - коэффициент, отражающий величину отклонения; q - коэффициент качества исполнения удара изменяющийся в пределах $[0;1]$.

Конечная точка удара определяется исходя из позиции противника. Для её расчета используется следующая формула (3.2).

$$Ep = \frac{hk}{2}, \quad (3.2)$$

где h - высота половины капсулы противника; k - коэффициент смещения.

Средняя точка зависит от угла замаха и считается как отклонение от центра отрезка между первой и последней траекторией. Для её расчета используется следующая формула (3.3)

$$Mp = \frac{s \tan(\alpha)}{2}, \quad (3.3)$$

где α - угол замаха; s - расстояние от начальной точки удара до конечной.

Для упрощения реализации, расчет и построение траектории происходит в 2D пространстве. После этого, полученный сплайн поворачивается в необходимый угол.

Таким образом, объединяя все выше описанное, алгоритм построения траектории руки выглядит следующим образом:

- 1) Определение начальной точки А (начальная позиция руки);
- 2) Определение конечной точки В по формуле 2;
- 3) Определение длины отрезка АВ;
- 4) Определение угла между точками А и В относительно нуля координат;
- 5) Переход в 2D плоскость, где точка А остается неизменной, но значение оси Z не воспринимается для расчета, а вторая точка рассчитывается как первая, но смещается на длину отрезка оси X;
- 6) Вычисление средней точки С по формуле 3;
- 7) Вычисление первой точки A^1 с учетом замаха по формуле 1;
- 8) Построение сплайна по трем точкам A^1 , С, В;
- 9) Поворот всего сплайна на изначальный угол, определенный в 3 пункте алгоритма.

3.4 Расчёт движения туловища

Ударное движение туловища происходит практически синхронно с ударным движением руки и состоит из выпада в сторону противника и закручивания туловища.

Поворот туловища осуществляется в зависимости от положения противника. Для определения общего угла поворота используется forward вектор, который отражает направление взгляда персонажа и enemy вектор в направлении противника.

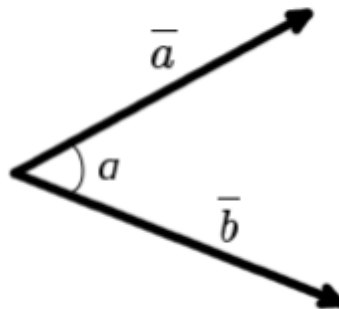


Рисунок 3.4 – Forward вектор(a) и вектор к противнику(b)

В ходе анализа ударных движений и боевых ситуаций было выделено 4 возможных положения противника относительно персонажа в условиях рукопашного боя:

- 1) противник стоит напротив игрока (угол между forward вектором игрока и enemy вектором: $\alpha \sim 0$ [30; 0] [0; -20]);
- 2) противник стоит левее игрока (угол между forward вектором игрока и enemy вектором: $\alpha > 30$ and < 90);
- 3) противник стоит левее игрока (угол между forward вектором игрока и вектором к противнику: $\alpha > -20$ and < -80);
- 4) противник стоит за спиной игрока.

Для каждого случая эмпирическим путем и из соображений биомеханики и ограничений в суставах был определен диапазон углов поворота туловища в момент удара:

- 1) поворот таза на -90 градусов от начальной стойки;
- 2) поворот таза на -20 градусов от начальной стойки;
- 3) поворот таза на -150 градусов от начальной стойки;
- 4) отсутствие поворота или полный разворот всего игрока.

Остальные кости туловища поворачиваются на угол, зависимый от общего угла поворота туловища α в процентном соотношении.

Для того, чтобы определить угол между forward и enemy векторами, необходимо рассчитать оба вектора.

Если в случае с forward вектором игрока, расчет происходит внутри движка, благодаря встроенным инструментам, то для enemy вектора необходимо найти разность между координатами двух точек, определяемых векторами.

Для расчета косинуса угла между векторами рассчитывается по формуле (3.4)

$$\cos \alpha = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}, \quad (3.4)$$

где a - forward вектор игрока; b - enemy вектор противника.

Знание угла позволит определить, с какой стороны находится цель для удара и соответственно, в какую сторону необходимо будет повернуть корпус персонажа при нанесении удара.

Угол поворота туловища рассчитывается исходя из процентного соотношения углов поворота частей скелета туловища (звеньев), основанного на биомеханических характеристиках человека в момент произведения движения удара. Финальный угол, на который необходимо довернуть корпус, рассчитывается по формуле (3.5)

$$\alpha = 0.6 \cdot \alpha + 0.4 \cdot \alpha + 0.3 \cdot \alpha + 0.2 \cdot \alpha, \quad (3.5)$$

где $0.6 \cdot \alpha$ - угол поворота pelvis (кость таза); $0.4 \cdot \alpha$ - угол поворота spine1; $0.3 \cdot \alpha$ - угол поворота spine2; $0.2 \cdot \alpha$ - угол поворота spine3.

Комбинируя все вычисления вместе, можно определять в какую сторону и на сколько градусов необходимо повернуть корпус в зависимости от положения противника в пространстве.

3.5 Расчет движения ноги

Кроме поворота туловища, необходимо также приводить в движение нижнюю часть корпуса, так как она также участвует в ударном движении. Кроме того, может возникнуть ситуация, когда противник находится вне зоны досягаемости, например, находясь за спиной или на отдалении. В таком случае необходимо сориентировать положение персонажа относительно противника. Для этого необходимо развернуть туловище персонажа в

направлении противника, а также выставить правильную стойку, передвинув ноги.

Способы определения позиций противника относительно анимируемого персонажа, а также расчет его углов поворота уже были рассмотрены ранее, так что далее будет только рассмотрена методика задания позиции у ног.

Само движение, так или иначе, можно описать, как обычный шаг в любом направлении. То есть сам процесс можно описать как подъем ноги с начальной точки и её перенос на конечную, при этом с легким смещением всего персонажа [32]. Начальная точка определяется положением ноги в момент замаха руки на удар, а конечная точка может быть рассчитана по формуле сложения векторов (3.6)

$$\bar{a} + \bar{b} = \{ax + bx; ay + by; az + bz\}, \quad (3.6)$$

где a – вектор из начальной позиции персонажа к начальной позиции ноги; b – вектор из начальной позиции персонажа к центру дистанции между персонажами.

Сам же перенос ноги может быть описан движением по синусоиде, в самом простом случае. Для симуляции шагового движения ноги по синусоиде необходимо интерполировать перемещение по Z от точки A до B по формулам (3.7, 3.8)

$$l_z = l_z + \sin(lerp)h, \quad (3.7)$$

$$lerp = lerp + deltaTime \cdot V, \quad (3.8)$$

где l_z - координата точки ноги относительно оси Z ; $lerp$ - приращение времени движения; h - высота на которую необходимо поднять ногу во время шага; $deltaTime$ - время между двумя кадрами; V - скорость перестановки ноги или же скорость с которой совершается шаг.

Для плавности движения, координаты x и y можно изменять линейно по формуле линейной интерполяции. Уравнение которой выглядит следующим образом [33] (3.9)

$$y = y_1 + \left(\frac{x-x_1}{x_2-x_1} \cdot (y_2 - y_1) \right), \quad (3.9)$$

где y - искомое; x - показатель, для которого определяется значение (искомое); x_1 - наименьший показатель; x_2 - наибольший показатель; y_1 - значение наименьшего показателя; y_2 - значение наибольшего показателя.

3.6 Подбор алгоритмов обратной кинематики для генерации различных типов движений

Для реализации алгоритма процедурной анимации необходимо определиться, какие методы обратной кинематики будут применяться к генерируемым движениям.

В пункте 1.3 данной работы были рассмотрены различные алгоритмы обратной кинематики. В силу специфики их работы они позволяют получить различный результат на одной и той же цепочке звеньев. Например, на рисунке 4.5 наглядно изображена работа алгоритма CCDIK [19]. Видно, что алгоритм позволяет получить ломанную линию в пространстве.

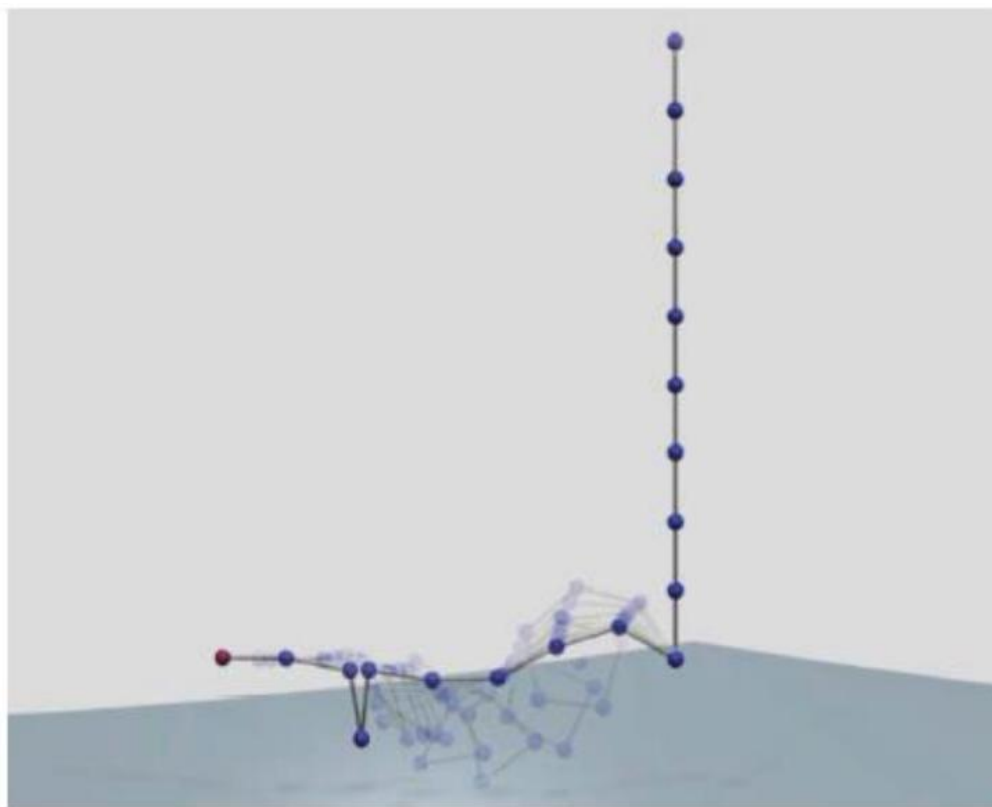


Рисунок 3.5 – Визуализация работы алгоритма CCDIK

Алгоритм FABRIK наоборот стремится привести систему в дугообразное состояние [19] (рисунок 3.6).

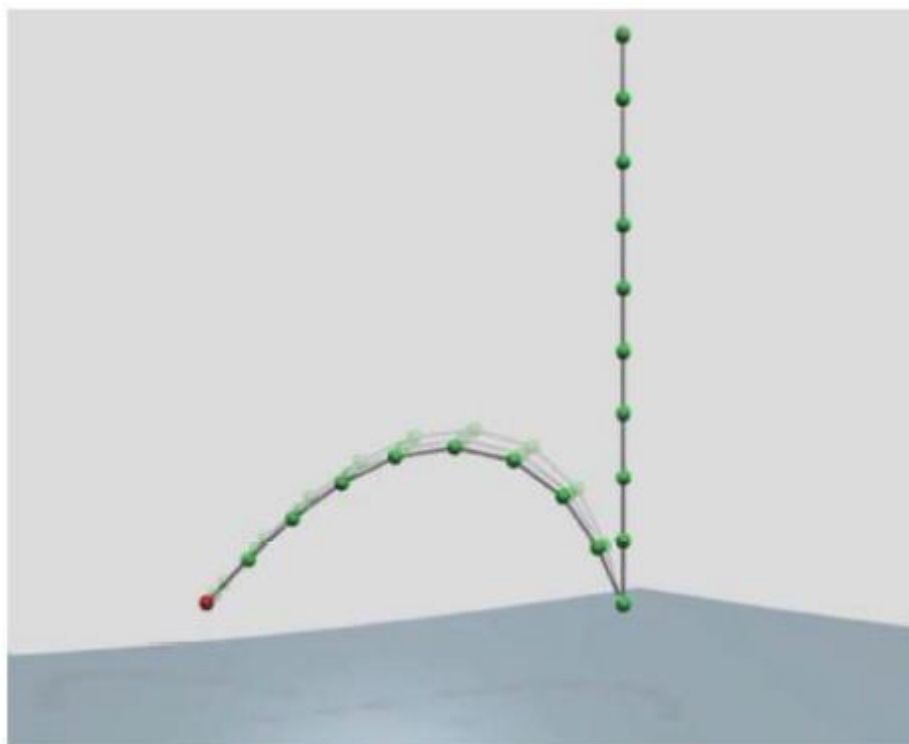


Рисунок 3.6 – Визуализация работы алгоритма FABRIK

Анализируя получаемые результаты, можно сделать вывод о том, что алгоритм FABRIK хорошо подходит, например, для создания движения руки, так как все звенья костной цепи плечо – предплечье – запястье – кисть имеют суставы, сгибающиеся в момент удара в одну сторону и теоретически, могут быть представлены в виде дуги. В то время как ноги имеют голеностопный сустав, который из соображения биомеханики сгибается в противоположную от коленного сустава сторону и, следовательно, в данном случае лучше подойдет алгоритм CCDIK.

Кроме вышеописанных алгоритмов IK, в движке Unreal Engine 5 также есть методы TwoBoneIK и LegIK. Они представляют собой модификацию алгоритма CCDIK для двух костей с возможностью регулировки положения центрального шарнира в цепочке звеньев, что позволяет использовать их для удобства генерации движений рук и ног. Возможность регулировать положение центрального шарнира хорошо накладывается на реальное движение руки в момент удара и позволяет, таким образом, изменяя положение центрального локтевого шарнира, создавать различные типы ударов

Алгоритм LegIK работает аналогично TwoBoneIK за исключением того, что в нем не нужно задавать положение центрального шарнира так как он всегда находится в плоскости звеньев цепи. Поэтому анимированная данным методом нога всегда будет сгибаться корректно в соответствии с биомеханической степенью свободы коленного сустава.

В ходе экспериментов с подбором алгоритмов для генерации различных движений, с целью получения наиболее правдоподобного с точки зрения биомеханики результата, были определены следующие пары движение-алгоритм:

- удар рукой – TwoBoneIK,
- выпад ногой – FootIK,
- толкающая нога – CCDIK,
- скручивание корпуса – FK Control Rig.

Таким образом, в силу различия получаемых решений задач обратной кинематики, описанные выше алгоритмы могут быть применены к созданию разных типов движения. Предполагается, что, комбинируя различные алгоритмы обратной кинематики в разные моменты удара и для различных звеньев анимационного скелета, можно добиться реалистичных движений.

4 РЕАЛИЗАЦИЯ АЛГОРИТМА ПРОЦЕДУРНОЙ АНИМАЦИИ

Для реализации прототипа будет использоваться стандартный манекен, предлагаемый Unreal Engine 5 для ригинга и анимации [34]. Помимо этого для первичного прототипа будет использоваться готовая движения одной руки, которую уже с помощью методов ИК алгоритм будет корректировать. Основываясь на формулах и расчетах движения приведенных в предыдущем разделе, был доработан алгоритм и создана новая блок-схема. Итоговый результат доработанного алгоритма приведен на рисунке.

Далее будет рассмотрена реализация алгоритма процедурной анимации удара боевым цепом с использованием формул, определенных в 4 разделе данной работы. Все вычисления производятся в движке Unreal Engine 5.

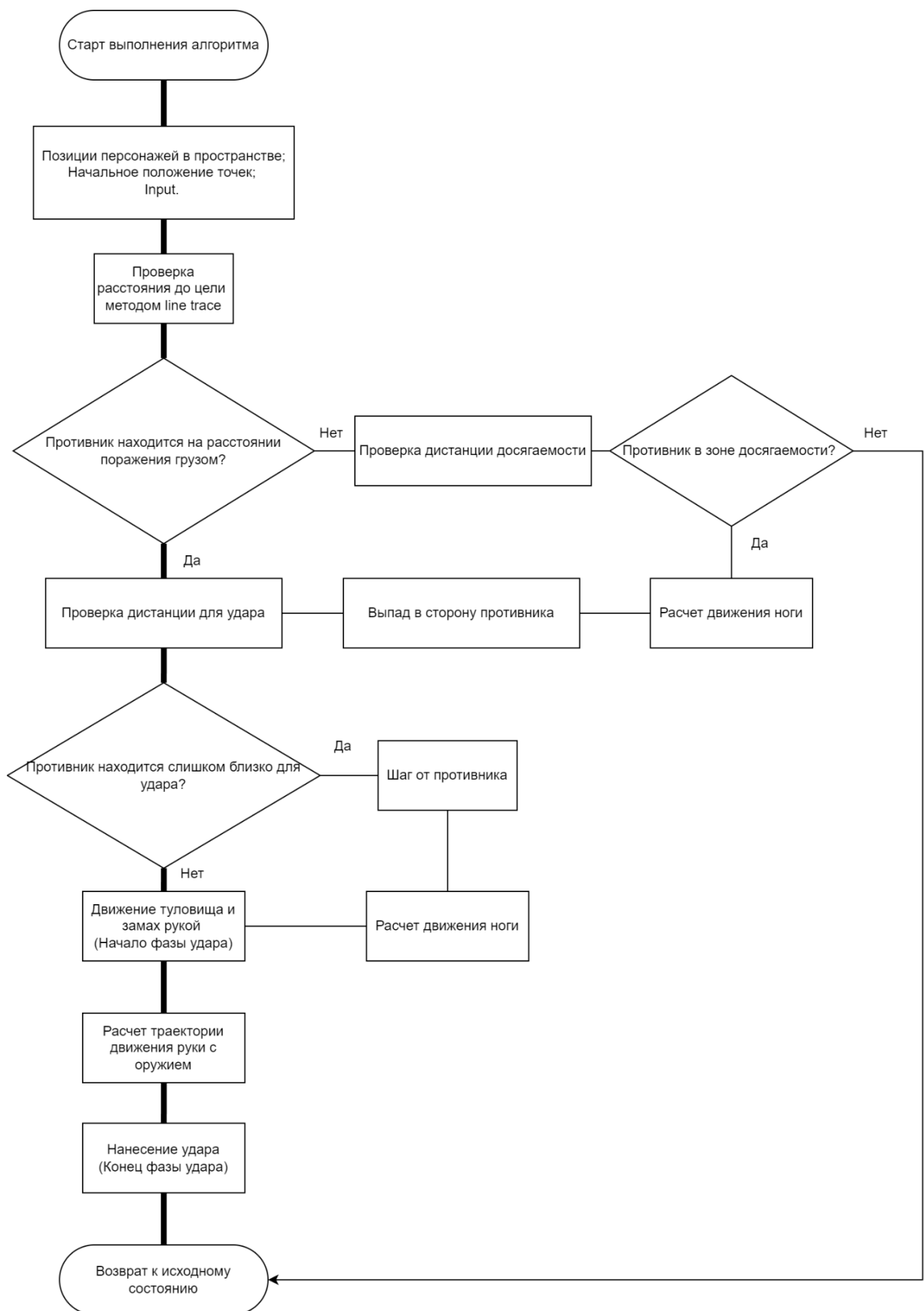


Рисунок 4.1 – Итоговая блок-схема алгоритма

4.1 Движение руки

Чтобы построить траекторию движения кулака в пространстве можно воспользоваться уравнением сплайна. Однако в Unreal Engine уже есть удобный инструмент для работы со сплайнами, который позволяет построить траекторию по точкам в пространстве. Для удобства и ускорения процесса разработки был использован данный инструмент.

В 4 разделе работы были выведены формулы построения траектории движения руки в соответствии с составленным алгоритмом. Формулы приведены ниже (4.1-4.3)

$$Sp = i \cdot r \cdot q, \quad (4.1)$$

где i - отклонение руки при замахе для нужной позиции удара; r - коэффициент, отражающий величину отклонения; q - коэффициент качества исполнения удара изменяющийся в пределах $[0;1]$

$$E_p = \frac{hk}{2}, \quad (4.2)$$

где h - высота половины капсулы противника; k - коэффициент смещения.

$$M_p = \frac{\sin(\alpha)}{2}, \quad (4.3)$$

где α - угол замаха; s - расстояние от начальной точки удара до конечной.

Результатом расчетов этих формул является движение руки персонажа, с последующим возвращением в нормальное состояние. (Рисунок 4.2)

Для сохранения биомеханических свойств конечности во время удара дополнительно используется метод TwoBoneIK, который двигает руку и плечевой сустав вслед за кулаком.

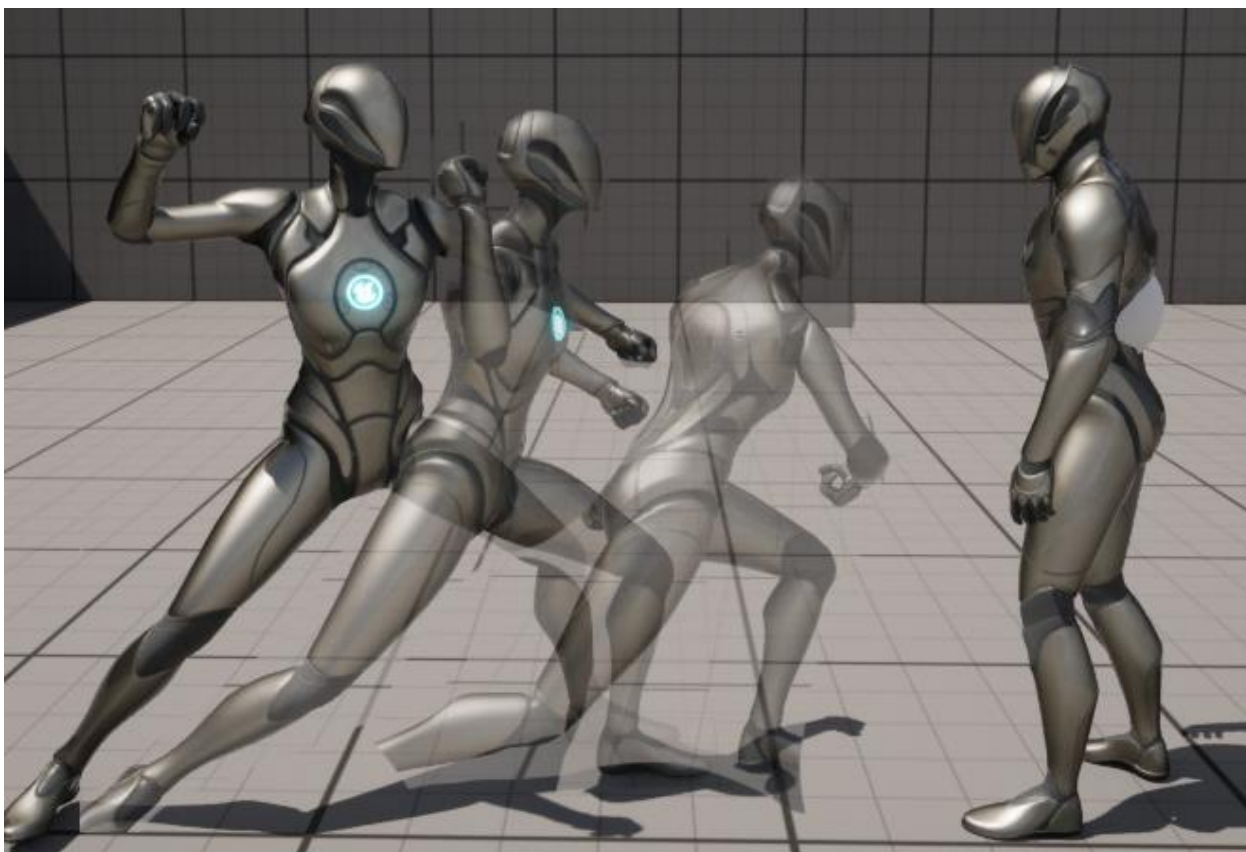


Рисунок 4.2 – Процесс нанесения удара рукой

4.2 Движение туловища

Ударное движение туловища происходит практически синхронно с ударным движением руки и состоит из выпада в сторону противника и закручивания туловища.

В 4 разделе работы были подобраны следующие формулы, которые позволяют рассчитать необходимые процессы (4.4-4.5)

$$\cos \alpha = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \cdot |\vec{b}|}, \quad (4.4)$$

где \vec{a} - forward вектор игрока; \vec{b} - enemy вектор противника.

$$\alpha = 0.6 \cdot \alpha + 0.4 \cdot \alpha + 0.3 \cdot \alpha + 0.2 \cdot \alpha, \quad (4.5)$$

где $0.6 \cdot \alpha$ - угол поворота pelvis (кость таза); $0.4 \cdot \alpha$ - угол поворота spine1; $0.3 \cdot \alpha$ - угол поворота spine2; $0.2 \cdot \alpha$ - угол поворота spine3.

Результатом является скручивание туловища во время удара, которое реализуется с помощью FK Control Rig инструмента встроенного в Unreal Engine 5 (Рисунок 4.3)

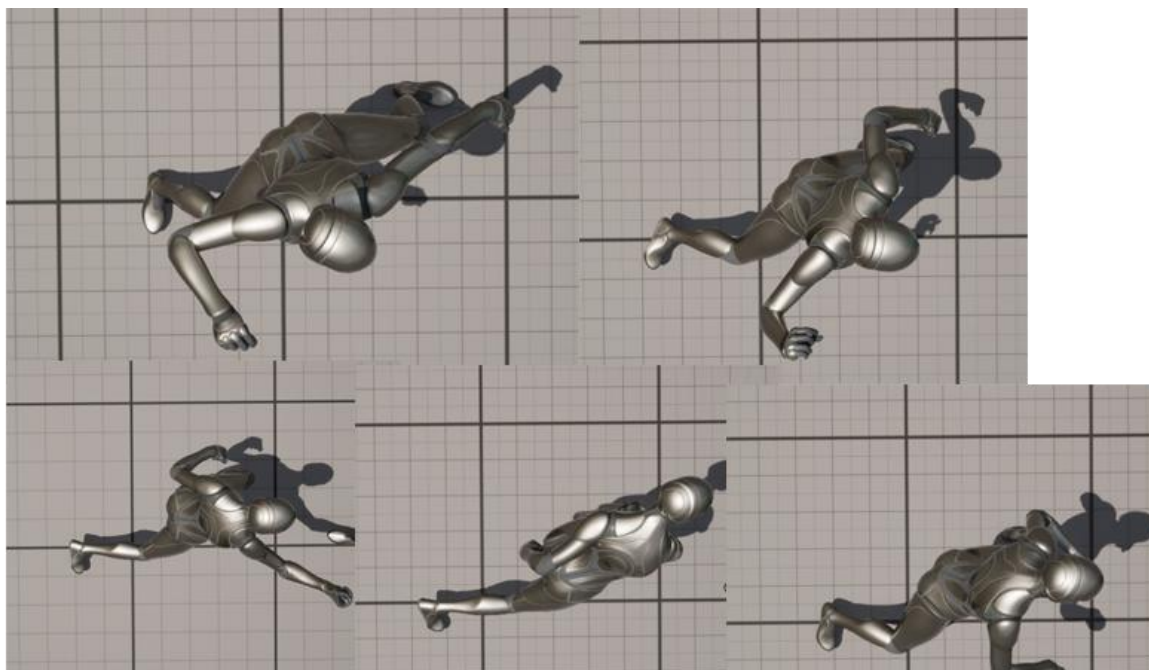


Рисунок 4.3 – Скручивание корпуса

4.3 Движение ноги

Для движения ноги также были выведены необходимые формулы, которые в сочетании с кинематикой будут двигать конечность в необходимом направлении в соответствии с расчетом по формулам приведённым ниже (4.6-4.8)

$$\bar{a} + \bar{b} = \{ax + bx; ay + by; az + bz\}, \quad (4.6)$$

где a – вектор из начальной позиции персонажа к начальной позиции ноги; b – вектор из начальной позиции персонажа к центру дистанции между персонажами.

$$l_z = l_z + \sin(lerp)h, \quad (4.7)$$

$$lerp = lerp + deltaTime \cdot V, \quad (4.8)$$

где l_z - координата точки ноги относительно оси Z; $lerp$ - приращение времени движения; h - высота на которую необходимо поднять ногу во время шага; $deltaTime$ - время между двумя кадрами; V - скорость перестановки ноги или же скорость с которой совершается шаг.

$$y = y_1 + \left(\frac{x - x_1}{x_2 - x_1} \cdot (y_2 - y_1) \right), \quad (3.9)$$

где y - искомое; x - показатель, для которого определяется значение (искомое); x_1 - наименьший показатель; x_2 - наибольший показатель; y_1 - значение наименьшего показателя; y_2 - значение наибольшего показателя.

Результат позволяет смещать стопу на заданную дистанцию, в то время, как остальная часть костей ноги двигается вслед за ней с помощью инструментария FootIK (Рисунок 4.4)

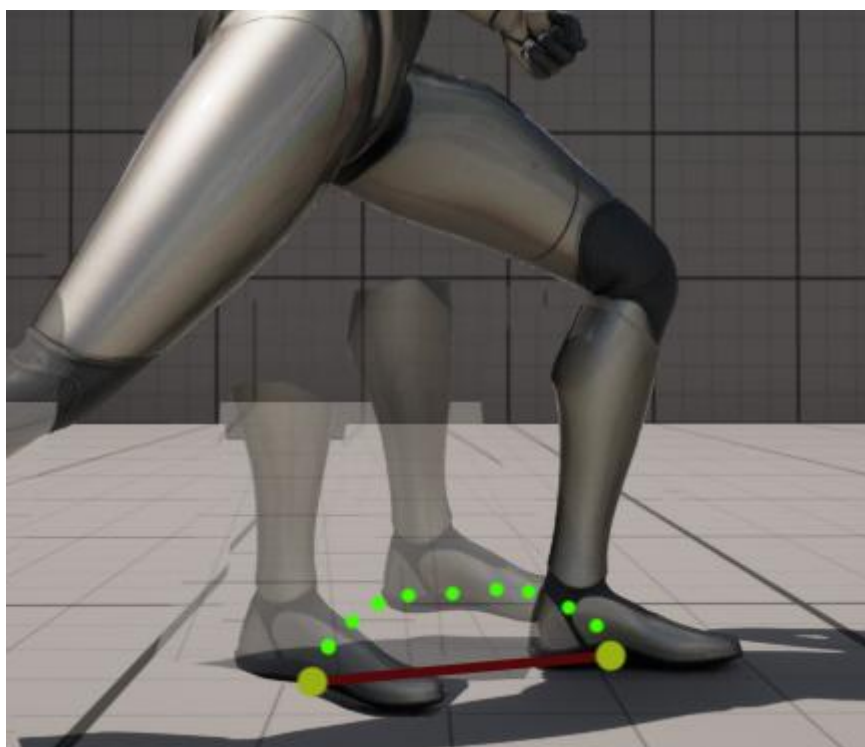


Рисунок 4.4 – Смещение ноги по дуге с помощью FootIK

4.4 Симуляция движения цепи

Для реализации реалистичного поведения цепи может подойти встроенный инструментарий Unreal Engine. Physics Constraints позволяет соединить два объект между собой, что открывает возможность к симуляции цепи и груза на ней.

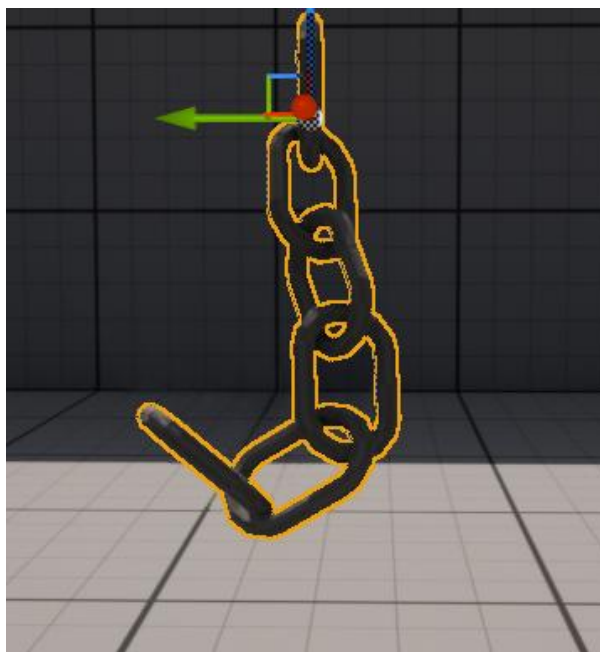


Рисунок 4.5 – Пример работы Physics Constraints

За счет своей универсальности, инструментарий позволяет работать со своими компонентами как и в Blueprint системе, так и использоваться при написании программного кода на C++.

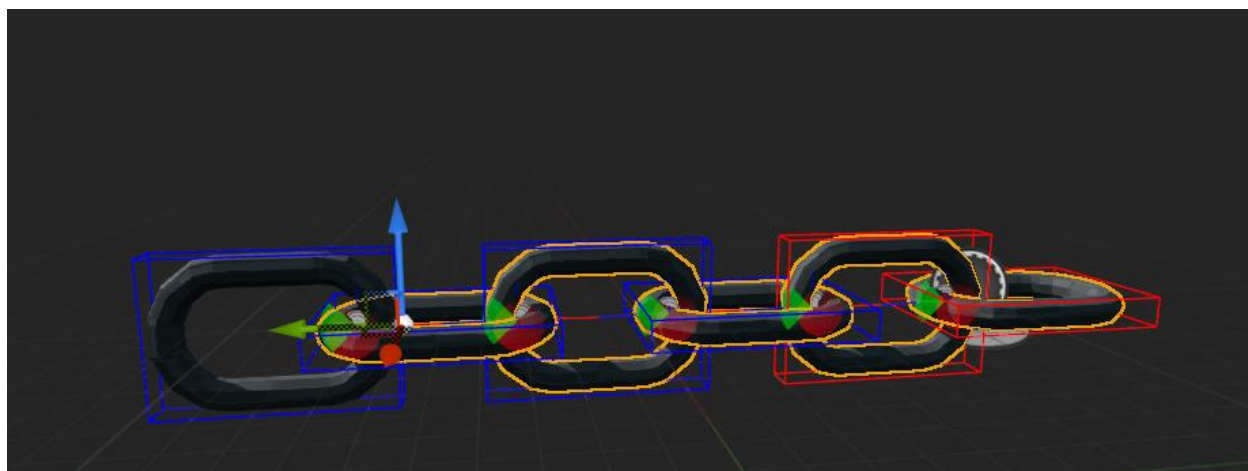


Рисунок 4.6 – Сочленение цепей методом Physics Constraints

Данный метод подойдет для тестового использования, но по результатам тестирования стало ясно, что для полноценной реализации может потребоваться иной способ работы с цепью, ввиду его сильной нагрузки на ресурсы компьютера для просчета каждой цепи.

5 ТЕСТИРОВАНИЕ И ОЦЕНКА РЕАЛИЗОВАННОЙ СИСТЕМЫ

5.1 Определение критериев тестирования

Полученный в ходе прошлой научно исследовательской и практической работ результат, должен удовлетворять некоторым критериям достижения цели. При разработке системы процедурной анимации, важно понимать, насколько эта система будет применима в условиях игровых ситуаций, а именно, как она будет вести себя в реальном времени в цикле игрового движка, и как будут выглядеть генерируемые движения. Для дальнейшего анализа итогового результата были определены следующие критерии:

- производительность разработанного решения,
- реалистичность генерируемых движений.

5.1.1 Тестирование алгоритма на мощном оборудовании

Неотъемлемым моментом при любой разработке инструментария или решения является производительность. Данное тестирование направлено на выявление стабильности и работоспособности решения в различных условиях. В случае системы процедурной анимации это быстрота работы алгоритма в игровых условиях. Во время запущенной игры, решение не должно сильно влиять на количество кадров в секунду, иначе это может повлечь ухудшения пользовательского опыта. На данный момент игровые приложения работают со скоростью от 60 FPS (кадров в секунду) до 144 FPS, где последнее является эталонным значением, а первое минимально допустимым. Поэтому разработанная система должна выполняться достаточно быстро, чтобы не влиять на количество кадров в секунду значительно. Так как система работает в реальном времени, вычисления, в большинстве своем, выполняются каждый кадр, к ней предъявляются особые требования к производительности.

Система для тестирования алгоритма имеет следующие характеристики:

- ОС: Windows 10 x64
- процессор: Intel Core i7-12700f 2.1GHz,
- видеокарта: NVIDIA GeForce RTX 3080 11GB,
- оперативная память: 32 GB DDR4.

Для сравнения и выявления работоспособности решения, было проведено тестирование в двух случаях:

- прототип с персонажем, использующим стандартную анимацию,
- прототип с персонажем, использующим процедурную анимацию.

Для обоих прототипов были созданы условия использования, в которых были исключены факторы дополнительно влияющие на производительность, а именно:

- отключено лишнее освещение,
- отключены лишние эффекты,
- отключены тени,
- убраны лишние статические объекты.

Обе сцены настроены идентично, за исключением самого персонажа. Тестирование также проводилось на сборках проекта. Таким образом исключается любое постороннее воздействие на производительность, что позволяет получить наиболее чистые замеры.

Так как компьютер, на котором производились замеры, имеет большую производительность, результат может содержать в себе как погрешности, так и неточности. Поэтому за эталон были приняты замеры, произведенные на прототипе с обычными анимациями.

Помимо этого, для более точных и приближенных результатов к использованию на менее мощном оборудовании, был выставлен лимит кадров в 80 FPS. С этими замерами будут сравниваться замеры производительности системы процедурной анимации.

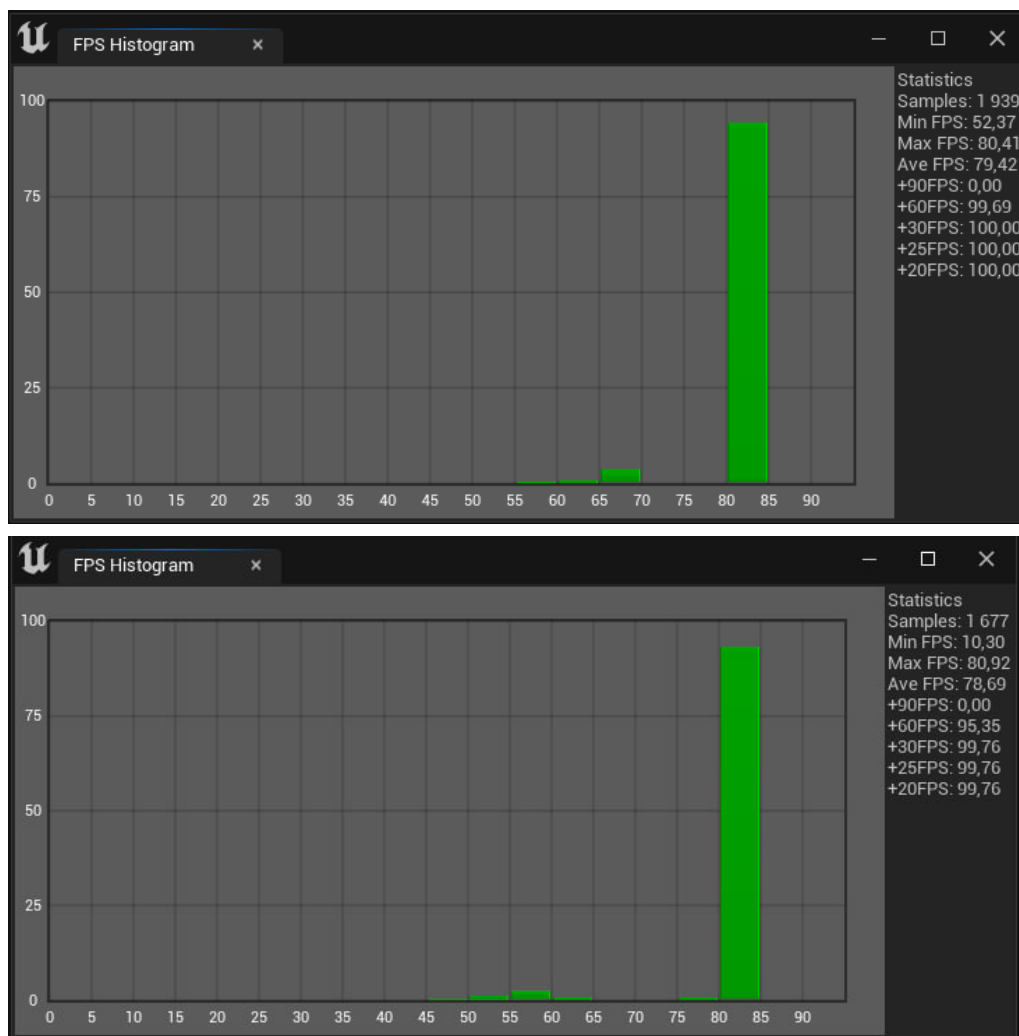


Рисунок 5.1 – Гистограммы FPS на мощном компьютере для одного персонажа с обычной анимацией (а) и с процедурной анимацией (б)

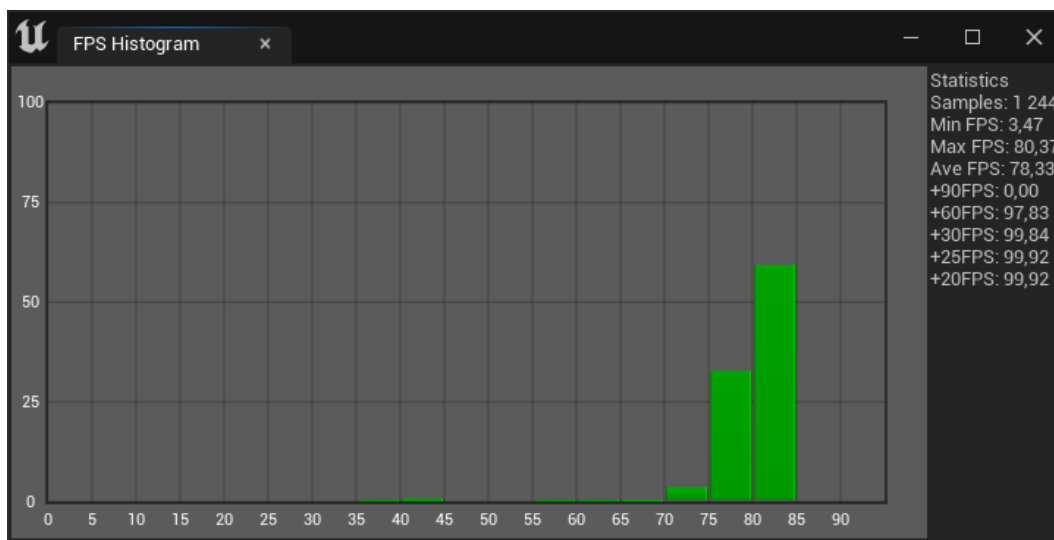
В ходе тестов производительности средствами встроенного профайлера Unreal Engine 5 были получены следующие результаты:

- для прототипа с обычными анимациями для одного анимируемого персонажа с оружием на сцене средняя частота кадров составила 79.42 FPS со средней скоростью отрисовки кадра 12.67 мс,
- для прототипа с процедурными анимациями для одного анимируемого персонажа с оружием на сцене средняя частота кадров составила 78.69 FPS со средней скоростью отрисовки кадра 12.93 мс.

Гистограммы с замерами FPS приведены на рисунке 5.1.

Таким образом, разница между двумя случаями составила 0.73 FPS или 0.26 мс на отрисовку кадра. Как можно заметить, на мощной системе, производительность для одного персонажа с оружием почти не меняется. Теперь необходимо произвести тестирование решения на масштабируемость. Для этого было увеличено количество персонажей одновременно использующих процедурную анимацию до 36. В ходе тестирования были получены результаты замеров производительности для 36 персонажей с оружием на одной сцене с обычной анимацией и на другой сцене с процедурной анимацией (рисунок 5.2):

- для прототипа с обычными анимациями для 36 анимированных персонажей с оружием на сцене средняя частота кадров составила 78.33 FPS со средней скоростью отрисовки кадра 13.03 мс,
- для прототипа с процедурными анимациями для 36 анимированных персонажей с оружием на сцене средняя частота кадров составила 62.47 FPS со средней скоростью отрисовки кадра 16.80 мс.



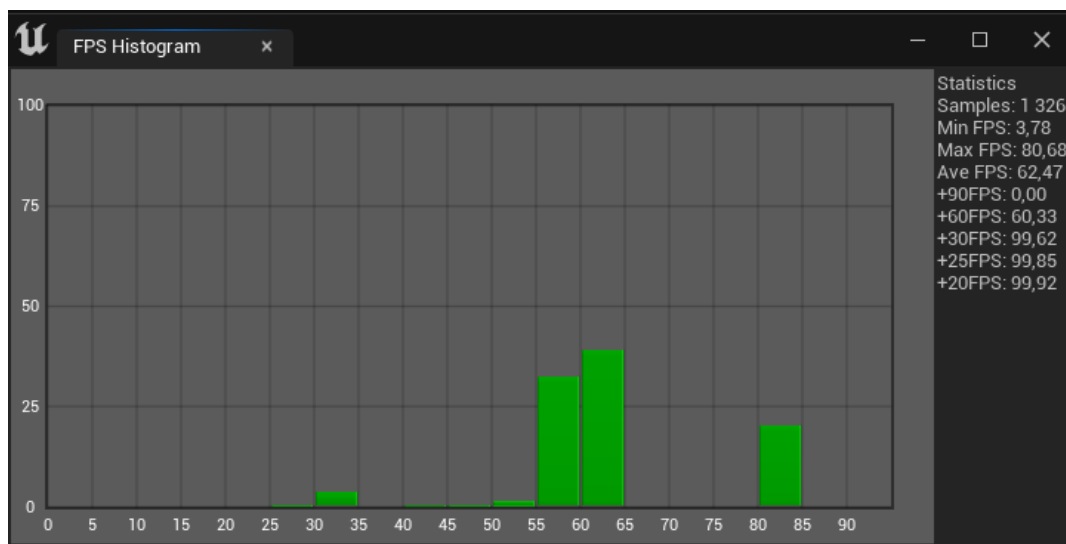


Рисунок 5.2 – Гистограммы FPS на мощном компьютере для 36 анимированных персонажей с оружием на сцене с обычной анимацией (а) и с процедурной анимацией (б)

По результатам проверки получилось, что разница между двумя случаями составила 15.86 FPS, это значит, что процедурная анимация обрабатывалась в 1.2 раза медленнее обычной, что является весьма неплохим результатом, учитывая погрешность и также то, что большинство вычислений алгоритма происходят на каждом кадре. Также, дополнительной нагрузкой оказался расчет движком физики цепи у оружия. Из-за большого количества сочленений цепи, повышается нагрузка как на видеокарту для отрисовки, так и на процессор для просчета физики.

5.1.2 Тестирование алгоритма на слабом оборудовании

Помимо тестирования на мощной системе, необходимо произвести такие-же замеры на компьютере с более слабыми характеристиками. Для этого, была выбрана система со следующими показателями:

- ОС: Windows 10 x64
- процессор: Intel Core i7-6700k 4.0 GHz,
- видеокарта: NVIDIA GeForce GTX 1070 8GB,
- оперативная память: 16 GB DDR4.

Для сравнения и выявления работоспособности решения на более слабой конфигурации, также, были проведены тесты в двух случаях:

- прототип с персонажем, использующим стандартную анимацию,

- прототип с персонажем, использующим процедурную анимацию.

Для обоих прототипов были созданы условия использования, в которых были исключены факторы дополнительно влияющие на производительность, а именно:

- отключено лишнее освещение,
- отключены лишние эффекты,
- отключены тени,
- убраны лишние статические объекты,
- убран лишний код,
- выключены лишние программы,
- понижены настройки окна визуализации.

Обе сцены настроены идентично, за исключением самого персонажа. Тестирование также проводилось на сборках проекта. Таким образом исключается любое постороннее воздействие на производительность, что позволяет получить наиболее чистые замеры.

Так как компьютер, на котором производились замеры, имеет небольшую производительность, замеры имеют погрешности. Поэтому за эталон были приняты замеры, произведенные на прототипе с обычными анимациями. С этими замерами будут сравниваться замеры производительности системы процедурной анимации.

В ходе тестов производительности средствами встроенного профайлера Unreal Engine 5 были получены следующие результаты:

- для прототипа с обычной анимацией одного анимируемого персонажа с оружием на сцене средняя частота кадров составила 59.19 FPS со средней скоростью отрисовки кадра 17.92 мс,
- для прототипа с процедурной анимацией одного анимируемого персонажа с оружием на сцене средняя частота кадров составила 48.05 FPS со средней скоростью отрисовки кадра 21.19 мс.

Гистограммы с замерами FPS приведены на рисунке 5.3.

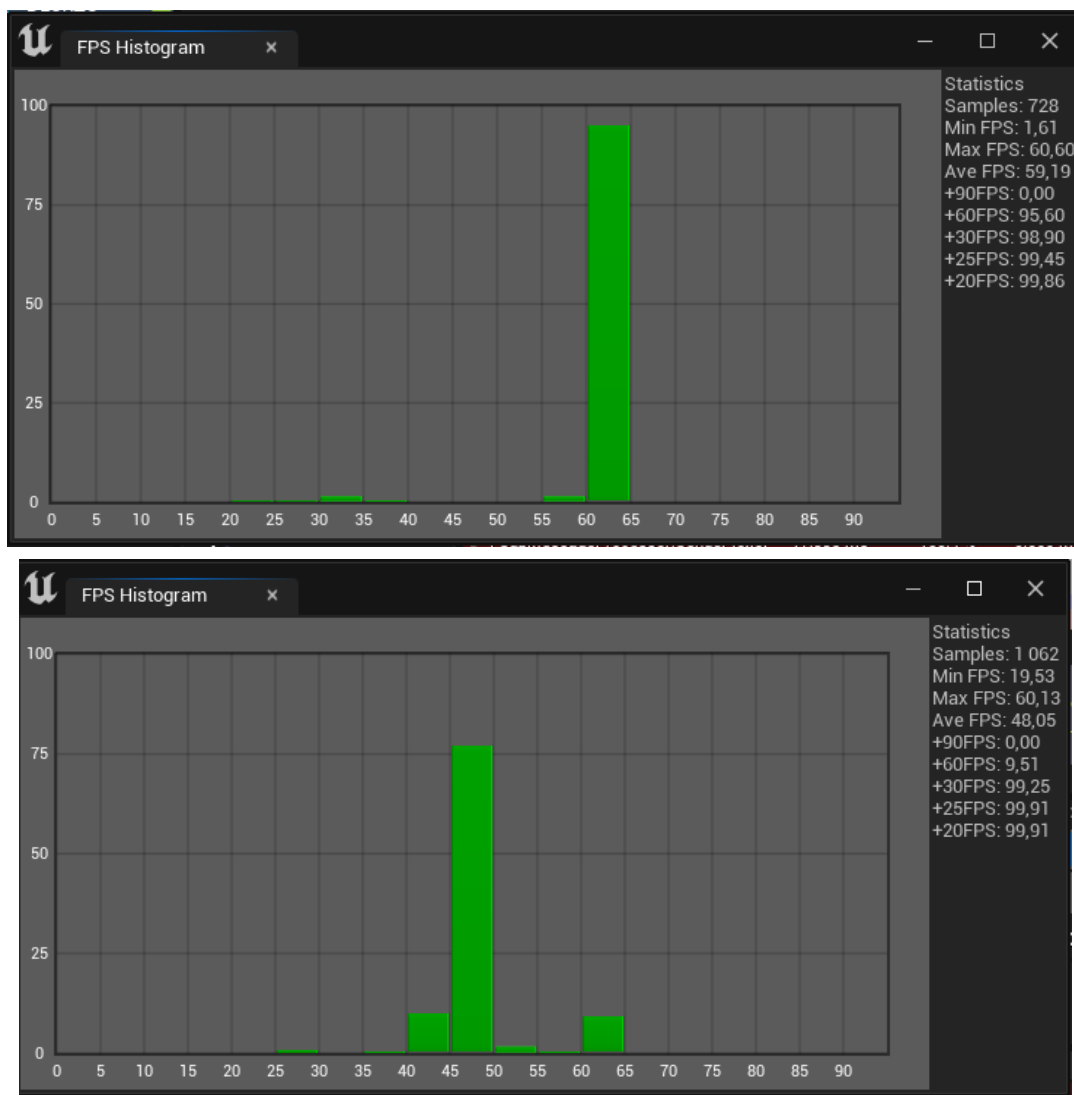


Рисунок 5.3 – Гистограммы FPS на слабом компьютере для одного персонажа с обычной анимацией (а) и с процедурной анимацией (б)

В первом тестировании на слабом компьютере, разница между двумя случаями составила 11.14 FPS или 7.27 мс на отрисовку кадра. Как можно заметить, в этом тестировании, производительность даже для одного персонажа с оружием довольно сильно отличается. Симуляция работы цепи и написание большей части кода на blueprint, довольно сильно влияет на производительность. Так что необходимо в будущем рассмотреть варианты по оптимизации решения.

Теперь необходимо произвести такое же тестирование решения на масштабируемость, но для слабой системы. Для этого точно так же было увеличено количество персонажей одновременно использующих

процедурную анимацию до 36. В ходе тестирования были получены результаты замеров производительности для 36 персонажей с оружием на одной сцене с обычной анимацией и на другой сцене с процедурной анимацией (рисунок 5.4):

- для прототипа с обычными анимациями для 36 анимированных персонажей с оружием на сцене средняя частота кадров составила 51.75 FPS со средней скоростью отрисовки кадра 20.30 мс,
- для прототипа с процедурными анимациями для 36 анимированных персонажей с оружием на сцене средняя частота кадров составила 39.14 FPS со средней скоростью отрисовки кадра 27.31 мс.

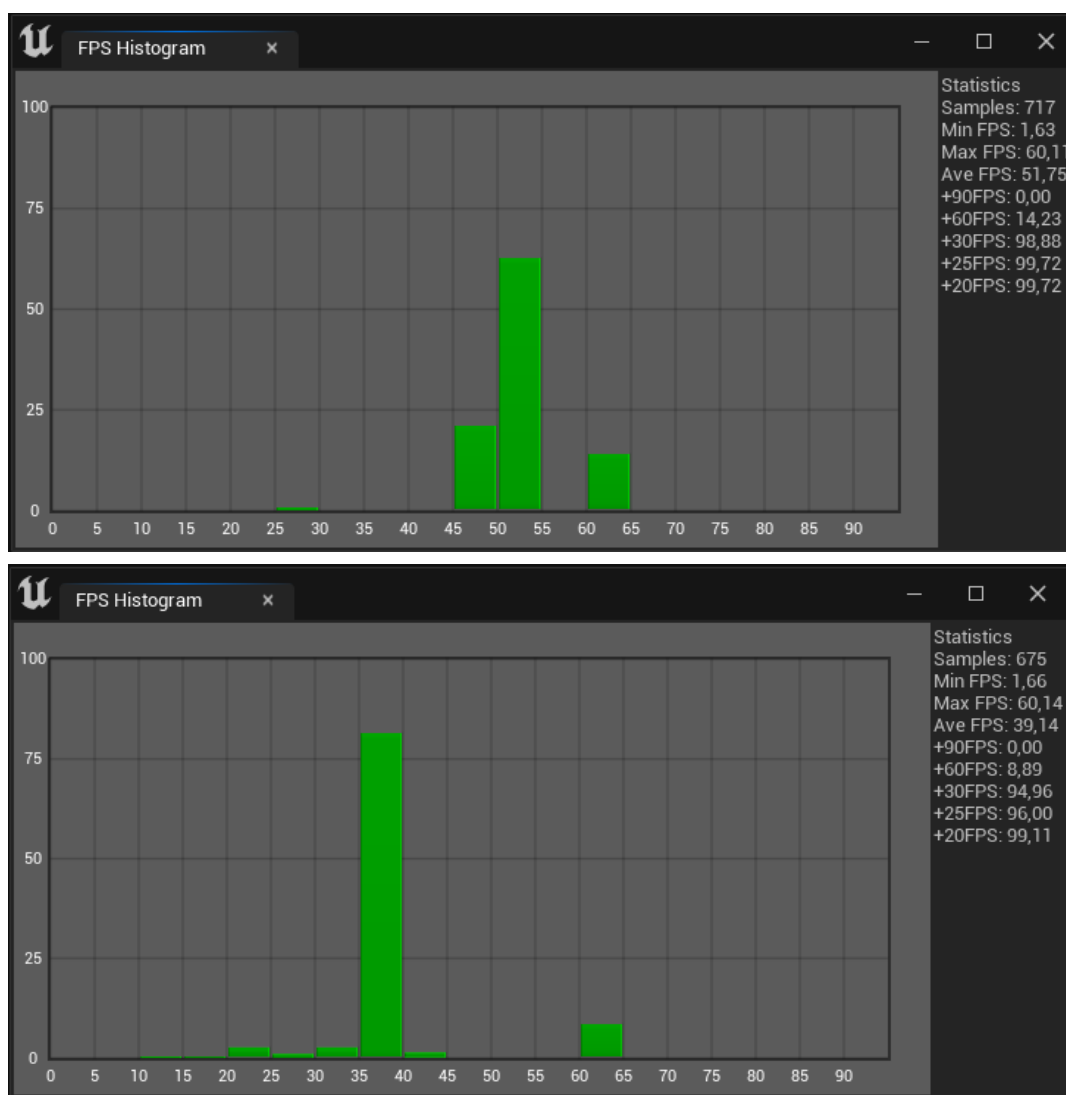


Рисунок 5.4 – Гистограммы FPS на слабом компьютере для 36 анимированных персонажей с оружием на сцене с обычной анимацией (а) и с процедурной анимацией (б)

По результатам тестирования, уже можно заметить, что масштабирование решения становится более ограниченным на слабой конфигурации компьютера. В дальнейшем необходимо произвести оптимизацию решения, для получения более стабильных показателей

Несмотря на все полученные результаты тестирования позволяют сказать, что система может быть использована как для анимации одного, так и для множества персонажей находящихся одновременно на одной игровой сцене. Однако, с ростом количества персонажей на сцене будет наблюдаться снижение частоты кадров, что может привести к зависаниям. Поэтому для

дальнейшего использования системы в играх с массовыми боями необходимо провести оптимизацию.

В рамках дальнейшего улучшения системы могут быть произведены предложенные шаги оптимизации [34]:

- полный перевод системы на C++,
- рефакторинг программного кода, удаление лишнего,
- распараллеливание выполнения функций (асинхронность),
- оптимизация и пересмотр симуляции цепи у оружия.

5.1.3 Натуральность генерируемых движений

Как уже упоминалось ранее, генерируемые движения должны выглядеть реалистично, чтобы улучшить пользовательский опыт во время игры и упростить работу аниматоров. Для этого была проведена субъективная оценка, которая будет состоять из следующих критериев:

физический и биологический реализм (как анимация визуально соответствует ограничениям реального мира),

стиль (конкретный способ, в котором движение будет выполнено, в зависимости от заданных параметров).

Так как данная оценка по большей мере не имеет конкретной метрики, было проведено визуальное сравнение движений, полученных в ходе процедурной генерации с движениями реального человека и с обычной анимацией. В ходе тестов, эмпирических экспериментов и сравнительной оценки было выявлено, что генерируемые движения выглядят достаточно реалистично и не уступают в этом обычной анимации, однако в отличие от нее позволяют внести разнообразие и уникальность. Все движения генерируются в соответствии с биомеханической и физиологической моделью реального человека и визуально похожи на реалистичные ударные движения боя с использованием кистеня или цепя.

5.2 Перспективы развития решения

Рассматривая перспективы системы, можно отметить различные варианты того, как можно было бы развить и улучшить разработанное решение. На базе разработанного подхода к созданию процедурной анимации ударов, в систему могут быть добавлены и другие ударные движения. В перспективе предполагается произвести следующие улучшения:

- провести оптимизацию системы,
- добавить другие типы движений,
- доработать имеющиеся тип движения,
- улучшить логику работы симуляции цепи у оружия
- добавить алгоритм просчета движений в зависимости от траектории цепи
- реализовать простой игровой прототип с применением разработанной системы.

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно-исследовательской работы были выполнены все поставленные задачи и получены следующие результаты:

- 1) изучены основные принципы биомеханики ударных движений человека, определена биомеханическая модель движений;
- 2) проанализированы существующие методы процедурной анимации, осуществлен выбор метода для применения в разработке системы;
- 3) проанализированы алгоритмы обратной кинематики, осуществлен выбор наиболее подходящих для реализуемой системы алгоритмов;
- 4) написан алгоритм процедурной анимации фехтования боевым цепом;
- 5) проанализированы физико-математическая и биомеханическая модели удара человека, на их основе определены параметры анимации;
- 6) определены зависимости влияния параметров на ударные движения, на их основе выведены уравнения и формулы для регулирования анимации в реальном времени;
- 7) проведен анализ и подбор программного обеспечения, подходящего для реализации системы процедурной анимации;
- 8) проведена программная реализация системы на движке Unreal Engine 5 в соответствии с составленным алгоритмом, формулами и определенными параметрами;
- 9) проведено тестирование системы, выявлены имеющиеся недостатки;
- 10) определены перспективы дальнейшего улучшения разработанной системы.

Разработанная система процедурной анимации позволяет генерировать движения для персонажей, фехтующих боевым цепом. Таким образом, аниматор может легко и быстро, путем задания параметров вручную или через код в реальном времени, создавать уникальные движения для каждого конкретного персонажа. Сами параметры при этом коррелируют с

представлениями об ударных движениях из реальной жизни, что делает их настройку понятной для аниматора.

Помимо этого, программная реализация алгоритма, текст и инструкция к запуску были опубликованы в репозитории на GitHub, доступ к которому можно получить по ссылке:

<https://github.com/Cobuch/Research-and-development-of-a-method-for-procedural-animation-of-fencing-with-a-combat-flail>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Shapiro A. Building a Character Animation System – 2011 – P. 1-12.
2. Казакова Н. Ю. Основные принципы разработки персонажа в рамках гейм-дизайна //Вестник Кемеровского государственного университета культуры и искусств. – 2016. – №. 35. – С. 146-157.
3. Никитиных Е. И., Конурина Г. А. РАЗРАБОТКА 3D-МОДЕЛЕЙ И АНИМАЦИИ ДЛЯ ИГРОВОЙ ИНДУСТРИИ. – 2021.
4. Кидалова Т. С. ДОСТОИНСТВА И НЕДОСТАТКИ ПРОЦЕДУРНОЙ АНИМАЦИИ //АРТ-ВИЗУАЛИС. – 2016. – С. 12-14.
5. Кидалова Т. С., Фомина А. В., Дзюбанов Д. С. ГЕНЕРАТИВНОЕ ИСКУССТВО И ПРОЦЕДУРНАЯ АНИМАЦИЯ //ТЕХНОЛОГИИ XXI ВЕКА: ПРОБЛЕМЫ И ПЕРСПЕКТИВЫ РАЗВИТИЯ. – 2016. – С. 238- 241.
6. Процедурная анимация и почему она не заменит ручной труд – 2020 URL: <https://dtf.ru/u/48338-akkaunt-ne-ispolzuetsya/165600-procedurnaya-animaciya-i-pochemu-ona-ne-zamenit-ruchnoy-trud> – (Дата обращения: 21.05.2024).
7. Букатов А. А., Гридчина Е. Е., Заставной Д. А. Методы скелетной анимации для трансформации полигональных поверхностей трёхмерных моделей //Инженерный Вестник Дона. – 2012. – Т. 21. – №. 3.
8. Ручкин К. А., Егурнов Н. О. Обзор систем процедурной генерации //Информатика, управляющие системы, математическое и компьютерное моделирование (ИУСМКМ-2017). – 2017. – С. 511- 5
9. Zucconi A., An Introduction to Procedural Animations. – 2017 URL: <http://www.alanzucconi.com/2017/04/17/procedural-animations/>
10. Ruuskanen A. Inverse Kinematics in Game Character Animation // Information Technology. – 2018 – P. 1-40.

11. McLaughlin T., Butler L., Coleman D. Character Rigging, Deformations, and Simulations in Film and Game Production // SIGGRAPH '11: ACM SIGGRAPH 2011 Courses August 2011 – P. 1– 18.
12. Holden D., Komura T., Saito J., Phase-functioned neural networks for character control // CM Transactions on Graphics 36(4), 2017 – P. 1 – 13
13. Реалистичная анимация персонажей в играх с помощью ИИ – 2019 URL: <https://habr.com/ru/post/474252/> – (Дата обращения: 21.05.2024).
14. Starke S., Zhang H., Komura T. Neural State Machine for CharacterScene Interactions // Trans. Graph. 38, 6 – 2019 –P. 1 – 14
15. Создание процедурной анимации из опорных кадров – опыт разработчика Overgrowth. – 2019 URL: <https://dtf.ru/gamedev/79680-sozдание-procedurnoy-animacii-vsego-iz-13-keyfreymovopytrazrabotchika-overgrowth> – (Дата обращения: 21.05.2024).
16. Дубровский В. И., Федорова В. Н. Биомеханика. – 2008.
17. Character Animation: Skeletons and Inverse Kinematics - 2017. URL: <https://venturebeat.com/pc-gaming/character-animation-skeletons-andinverse-kinematics/> – (Дата обращения: 21.01.2024).
18. Aristidou A., Lasenby J. Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver. – 2009.
19. Martin A., Barrientos A., Del Cerro J. The natural-CCD algorithm, a novel method to solve the inverse kinematics of hyper-redundant and soft robots //Soft robotics. – 2018. – Т. 5. – №. 3. – С. 242-257.
20. Aristidou A., Lasenby J. FABRIK: A fast, iterative solver for the Inverse Kinematics problem //Graphical Models. – 2011. – Т. 73. – №. 5. – С. 243- 260
21. Моргенштерн, боевой цеп. Как сражались этим страшным оружием– 2021 URL: <https://www.mirf.ru/science/morgenshtern-boevoj-cep-kisten/> – (Дата обращения: 21.05.2024).

- 22.Как это было. Часть 14. Дробящее оружие на цепи - цепи – 2015
URL:<https://fishki.net/1769613-kak-jeto-bylo-chast-14-drobjawee-oruzhie-na-seri---seru.html> – (Дата обращения: 21.05.2024).
- 23.Кусарифундо и манрикикусари (утяжеленная цепь) – 2011 URL:
<https://mitsumono.ru/weaponandequip/108-kusarifundo/> – (Дата обращения: 21.05.2024).
- 24.КУСАРИ-ДЗЮЦУ – 2012 URL:
https://ninja2.my1.ru/news/kusari_dzjucu/2012-02-25-87 – (Дата обращения: 21.05.2024).
- 25.Кусаригамадзюцу (кусарикама-дзюцу) - искусство сражения серпом на цепи – 2011 URL: <https://mitsumono.ru/skills/133-kusarigamajutsu/> – (Дата обращения: 21.05.2024).
- 26.Bishop, Mark D. Okinawan Weaponry, Hidden Methods, Ancient Myths of Kobudo & Te. N.p., Lulu.com, 2017.
- 27.Rajapaksha N., The Best Game Engines For Advanced Game Development, 2020.
- 28.Petrichenko L. V., Albekova Z. M., Game Engine Comparison // Научные тенденции: Вопросы точных и технических наук, 2018. 79
- 29.Al Lawati H. A. J., The Path of UNITY or the Path of UNREAL? A Comparative Study on Suitability for Game Development, 2020.
- 30.Lee H. S., Ryoo S. T., Seo S. H., A Comparative Study on the Structure and Implementation of Unity and Unreal Engine 4 // Journal of the Korea Computer Graphics Society 25(4):17-24, 2019
- 31.Спарринг кистень [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=88FDF3NsU3I&ab_channel=%D0%90%D0%BB%D0%B5%D0%BA%D1%81%D0%B0%D0%BD%D0%B4%D1%80%D0%97%D0%B5%D0%BD%D0%B4%D1%80%D0%B8%D0%BA%D0%BE%D0%B2 – (Дата обращения: 21.05.2024).

32. Zihao Yang, Minghai Yuan, Xinhui Shi, Zenan Yang and Mengyuan Li, Mechanism Design and Kinematics Analysis of Spider-like Octopod Robot// Journal of Physics: Conf. Series 1314 (2019)
33. Р.М. Минькова ВЕКТОРНАЯ АЛГЕБРА И АНАЛИТИЧЕСКАЯ ГЕОМЕТРИЯ// Екатеринбург: ГОУ ВПО УГТУ–УПИ, 2006. 42 с
34. Касьянов В. Н. Методы оптимизации программ. – 1984.