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

Факультет Школа разработки видеоигр
Образовательная программа Технологии разработки компьютерных игр

О Т Ч Е Т

о преддипломной практике

Тема задания: Разработка алгоритма процедурной анимации для ленивца

Обучающийся Султанов Тимур Ильдарович, J4221

Согласовано:
Руководитель практики от университета: Карсаков Андрей Сергеевич, кандидат технических наук, факультет школы разработки видеоигр, доцент

Практика пройдена с оценкой _____

Дата _____

Санкт-Петербург
2024

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Инверсная кинематика – процесс определения параметров связанных подвижных объектов (для достижения необходимой позиции, ориентации и расположения этих объектов

Конечный эффектор – устройство на конце манипулятора, предназначенное для взаимодействия с окружающей средой

Риггинг – это подготовка трехмерной модели персонажа к анимации, при которой внутри заранее отрисованной заготовки размешается риг — набор виртуальных суставов и костей, устанавливаются закономерности его функционирования и возможные трансформации

Blueprint – система визуального программирования в Unreal Engine 4 и Unreal Engine 5 на основе нодов с данными: событиями и функциями

FPS – количество сменяемых кадров за единицу времени в компьютерной графике

Line trace – линейная проверка, от точки до точки на наличие препятствий, если на пути линии имеется объект, то она возвращает информацию об этом объекте

Motion capture – метод анимации персонажей и объектов при помощи оцифровки движений реального объекта и последующего переноса их на трёхмерную модель

Runtime анимация – анимация в режиме реального времени

Sphere trace – перемещает сферу вдоль заданной линии и возвращает первое встреченное попадание

ВВЕДЕНИЕ

В современной медиаиндустрии активно применяются различные виды компьютерных анимаций. При компьютерной реализации даже применительно к одной модели разработчики часто используют большое количество различных анимаций. При этом используются различные программные комплексы и различные подходы к реализации анимаций.

Сама анимированная компьютерная модель объекта может реализовываться с использованием различных методов, например, используя покадровую анимацию, вертексную или скелетную. Одним из наиболее часто используемых способов является скелетная анимация, при которой у модели реализуется система костей скелета с привязкой друг к другу и иерархическим подчинением, которая позволяет выставлять степени свободы перемещения для каждой кости.

При этом анимация может реализовываться как с использованием «классического» подхода – предопределенной анимации, так и с помощью процедурной runtime анимации – когда математический алгоритм, реализованный в анимации, подстраивает анимацию модели в зависимости от окружающего ландшафта в режиме реального времени.

Процедурная анимация часто применяется, обычно включает сложные алгоритмы для создания реалистичных движений, особенно с движениями, основанными на физике. Эта методика часто используется для моделирования систем частиц, таких как огонь, дым, жидкостей [1], динамики одежды и волос персонажей [2], растительности [3], атмосферных эффектов [4] и т.п. Но при анимации гуманоидов или животных сложность возрастает, поскольку физические ограничения персонажа с внутренним скелетом, как правило, требуют соответствующего подхода (риггинг).

На практике для сложных моделей часто используется смешанный подход, когда заранее подготовленные анимации реализуются в режиме реального времени в зависимости от окружающей среды.

Преимущества процедурной анимации, по сравнению с классической очевидны – это гибкость в настройке параметров, относительно небольшой размер задействованных данных, экономия времени на создание анимации, реалистичность взаимодействия модели со своим окружением в сцене. Нельзя, однако, не отметить и недостатки, такие как, техническая сложность при создании и более высокие требования к оборудованию.

Но, даже несмотря на более высокую техническую сложность при создании процедурной анимации по сравнению с классической, она остается перспективной и востребованной в медиаиндустрии именно за счет своих преимуществ.

Следует отметить, что при прототипировании в компьютерных играх или, скажем, в реализации 3D сцен часто возникает проблема с нахождением подходящей анимированной модели (ассета), размещенной в публичном доступе. Причем, если данная модель требуется для создания, например, реалистичного лесного окружения и соответствующих существ в сцене, то часто предпочтительно, чтобы анимация была максимально близка к естественной, поскольку это позволяет игроку погружаться в более реалистичный окружающий мир.

Поэтому реализация общедоступной анимации существ, в частности процедурной анимации двупалого ленивца на основе физиологической модели поведения, поможет в решении вышеуказанной проблемы и повысит реалистичность моделирования и возможность применения данного подхода в играх и компьютерных сценах со специфичными механиками и соответствующим окружением.

Целью настоящей работы является разработка алгоритма и имплементации процедурной анимации передвижения на ветвях ленивца на основе исследованных физиологических данных.

В ходе работы над настоящим проектом необходимо выполнить следующие задачи: произвести обзор анатомии и кинематики ленивца, проанализировать существующие методы анимации и алгоритмы инверсной

кинематики. Затем, подобрать программное обеспечения для реализации, разработать алгоритм процедурной анимации и выполнить программную имплементацию алгоритма, протестировать в различных сценариях и оценить полученные результаты.

Актуальность темы обусловлена тем, что в последнее время возникла тенденция большего использования современной медиа-индустрией процедурных анимаций в 3D сценах, прототипировании, поскольку они обеспечивают более высокую реалистичность, а также публично доступные процедурные модели позволят их использовать в низкобюджетных инди-проектах.

1. АНАЛИЗ ПРЕДМЕТА И ОБЛАСТИ РАЗРАБОТКИ

1.1 Изучение анатомии и кинематики ленивца

Для разработки метода процедурной анимации двупалого ленивца целесообразно проанализировать физиологические и анатомические особенности как строения скелета, так и особенностей передвижения двупалого ленивца с целью реализации процедурной анимации физического правильной модели перемещения.

В исследовании [5] отмечаются такие особенности передвижения ленивца на ветвях как:

– диагональный характер поступи (правая задняя (rh), левая передняя (lf), левая задняя (lh), правая передняя (rf));

– структура конечностей ленивца (рисунок 1);

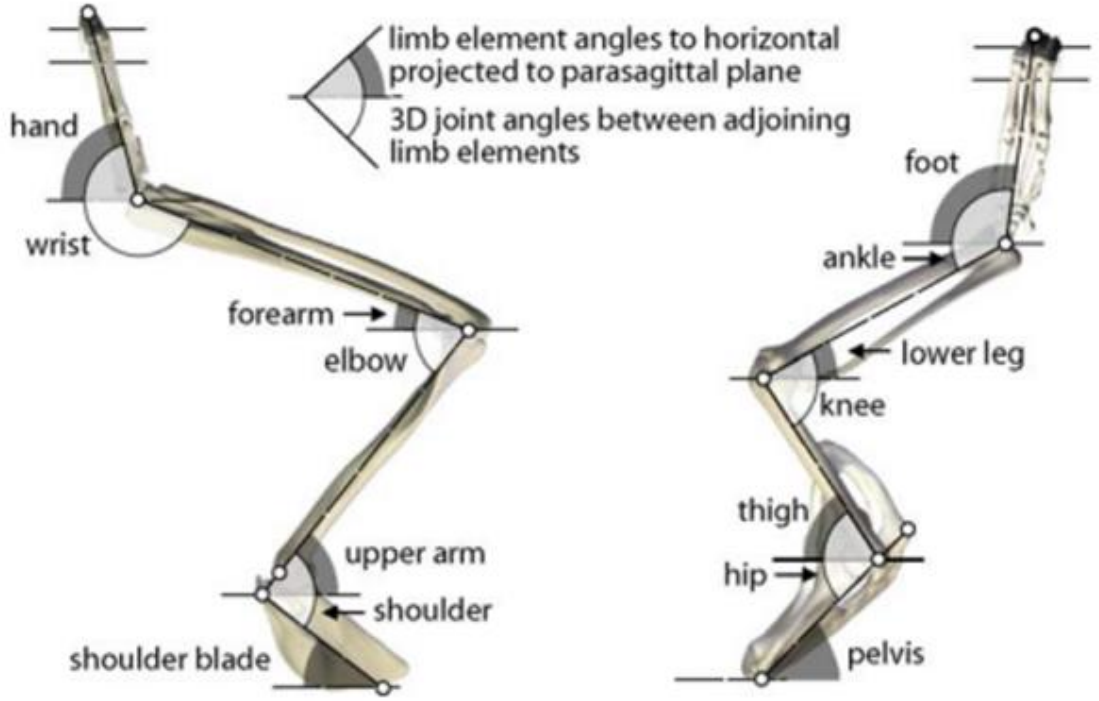

Рисунок 1 – Скелетные ориентиры и анализ углов движения конечностей [5]

– пространственно-временные параметры походки [5] показаны в таблице 1. Продолжительность контакта с веткой варьируется от 0,68 до 17,44 сек. При этом в среднем фаза контакта длилась в 2 раза дольше, чем фаза висения, которая колебалась от 0,19 до 4,54 сек. Длина шага колебалась от 0,21 м до 0,83 м;

Таблица 1 – Пространственно-временные параметры передвижения

	N	Contact duration (mean in s±s.d.)	Swing duration (mean in s±s.d.)	Stride length (mean in m±s.d.)
Unrestrained locomotion (all trials)				
Forelimb	205	2.57 ± 2.50	1.16 ±0.48	0.59 ± 0.12
Hindlimb	205	2.28 ± 2.27	1.33 ± 0.66	0.60 ±0.13
Unrestrained locomotion (0.2-0.3 m/s)				
Forelimb	93	1.55 ±0.33	1.02 ± 0.26	0.62 ±0.10
Hindlimb	93	1.32 ±0.21	1.11 ± 0.31	0.63 ± 0.09
Steady-state locomotion (0.2-0.3 m/s)				
Forelimb	32	1.50 ± 0.20	0.84 ± 0.20	0.59 ± 0.05
Hindlimb	32	1.66 ±0.25	0.67 ±0.15	0.60 ± 0.05

– таблица амплитуд и углов, полученная в ходе экспериментов [5] показана в таблице 2:

Таблица 2 – Таблица амплитуд и углов для конечностей ленивца

Bone name	Amplitude contact phase (º) Mean
1	2
Forelimb elements	
Shoulder blade	41.5
Upper arm	98.1
Forearm	92.3
Hand	36.8
Hindlimb elements	
Pelvis	8.5
Thigh	75.7
Lower leg	53.4
Foot	17.8

Вышеперечисленные данные, полученные в ходе исследования [5], производились на деревянном шесте длиной 4,2 метра и диаметром 40 мм. В качестве объекта исследования выступали несколько животных вида двупалый ленивец Choloepus didactylus со следующими физическими параметрами:

– вес испытуемых составил от 6,5 кг, до 10,6 кг,

– длина тела составил от 78 см до 87 см.

Трехмерный кинематический анализ плечевого пояса при передвижении вверх ногами двупалых ленивцев приведен в статье [6], в ней рассчитаны кинематические данные степеней свободы для элементов плечевого пояса при обычном передвижении двупалого ленивца.

Данные исследования [7] показывают, что передние и задние конечности ленивцев функционируют аналогично друг другу во время передвижения, и что перенос веса на передние конечности, типичный для приматов, не является аналогичным у всех млекопитающих. Таким образом данные этого исследования целесообразно учитывать при рассмотрении реализации передвижения двупалого ленивца на ветвях. Отметим, что в работе авторов [7] приведены усредненные данные о скорости перемещения животного, которая составила по результатам экспериментов 0,11 ± 0,04 м/с при свободном перемещении без ограничений.

Согласно данным исследования [8], крепление рук и ног животного к контактной точке опоры при перемещении осуществляется практически перпендикулярно, как показано на рисунке 2:

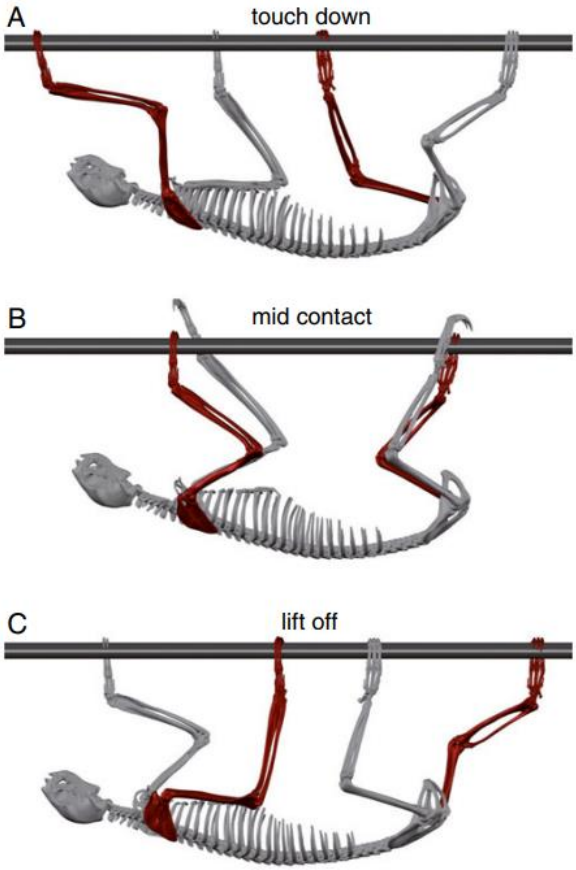

Рисунок 2 – Иллюстрация скелетных данных [8]

Исследователями в статье [8] отмечается, что ленивцы животные не используют маятниковую механику (как например обезьяны гиббоны), т.к. стратегию ленивцев можно сравнить со стратегией человека, интуитивно меняющего походку при ходьбе по тонкому льду. Комбинированные эффекты смоделированных крутящих моментов в суставах приводят к очень контролируемому движению. Очевидно, что уменьшение прилагаемых динамических сил делает поломку опоры (ветки) менее вероятной. Для видов, которые, вероятно, не способны достаточно быстро реагировать на возможный обрыв опоры путем прыжка или немедленного протягивания более надежного захвата, в статье полагают, что что эта стратегия имеет адаптивное значение в древесной среде обитания.

Автором Frank C. Mendel в работе «Use of Hands and Feet of Three-Toed Sloths (Bradypus variegatus) during Climbing and Terrestrial Locomotion» отмечено, что при локомоции по вертикальному стволу ленивец движется в

положении, когда голова направлена вверх и случаев спуска головой вниз не наблюдалось [9].

1.2 Анализ методов анимации

При создании анимации часто традиционно [10] аниматоры используют перемещение персонажей на сцене для создания новых положений по методу **ключевых кадров**, когда для каждого кадра захватываются только отдельные моменты времени, которые соответствуют основным фазам, и с помощью компьютерного алгоритма промежуточные движения интерполируются, чтобы создать плавный переход. Данный подход иногда называют построением промежуточных кадров.

Данный подход, в частности, часто используется в 2D анимации и продолжает развиваться разработчиками в настоящее время [11]. На рисунке 3 приведена оптимизация методики ключевых кадров с использованием “трансформации расстояний”, и показана оптимизация в виде увеличения качества при использовании данного улучшения. Таким образом показано, что использованная разработчиками [11] “трансформация расстояний” позволяет лучше избегать артефактов, а если не использовать данную оптимизацию, то при интерполяции на месте пустого пространства проявляются черные точки-артефакты:

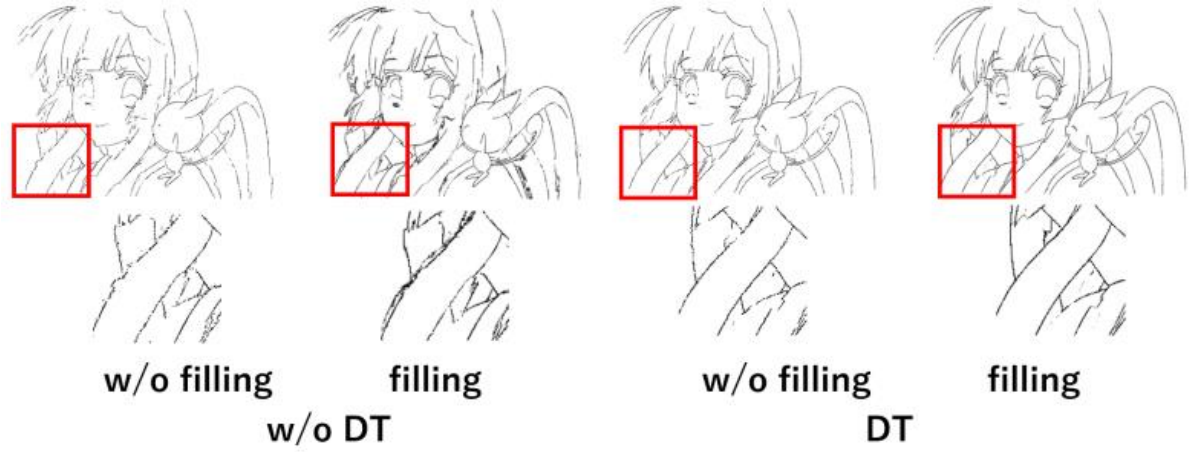

Рисунок 3 – Оптимизация ключевых кадров с использованием «трансформации расстояний» [11]

Для того, чтобы реализовать сложные движения в режиме реального времени используется **метод motion capture** (захват движений) [12]. При использовании данного подхода выделяют две основные техники:

А) Захват движение на основе маркеров. Маркеры нужны для фиксации движения объектов. Маркеры могут быть основаны на различных принципах:

1) Акустические системы. Звуковые датчики расположены на основных суставах актера, а приемники расположены в радиусе захвата. Затем излучатели активируются последовательно и генерируют набор частот, которые улавливаются приемниками и используются для оценки движения в 3D-пространстве.

2) Механические системы. Состоят из ползунков и потенциометра, позволяющие захватывать движения на простой механической основе, на компьютер передаются данные об углах сгибов сочленений со специального костюма, при этом преимуществом подобной системы является ее независимость от внешних искажений.

3) Магнитные системы. Маркеры являются магнитами, а приемники действуют как камеры, для получения движения анализируются магнитные искажения,

4) Оптические системы. Используется костюм с отражателями, и для захвата движения используется высокочастотная видеокамера.

Б) Захват движения без маркеров. Основан на программных технологиях компьютерного зрения и распознавания образов. Образ актера рассматривается несколькими камерами с разных ракурсов и обрабатывается в специальных программных средах [11]. Пример на основе данного подхода показан на рисунке 4:

Рисунок 4 – Захват движения лица без маркеров,
только программными средствами [11]

Вышеперечисленные подходы являются трудоемкими по времени, дорогостоящими и ограничены в использовании физики и взаимодействии с окружающим ландшафтом, что затрудняет их использование в отдельных 3D сценариях (например, в играх и подобных виртуальных мирах), когда необходимо обеспечить интерактивность в режиме реального времени.

В свою очередь, **процедурная анимация** [13] (анимация, основанная на применении математических правил, описывающих поведение объекта), подразделяется на несколько видов, среди которых можно выделить следующие:

1) Неизменяемая процедурная анимация. Анимация в физической среде используется в постоянном виде, без учета окружения.

2) Процедурная анимация на основе машинного обучения [14]. Для процедурной генерации движений используются нейронные сети.

Данный подход требует наличия большого количества качественной исходной базы данных, полученных с помощью технологии захвата движений или с помощью создания в 3D программе (например, Blender) анимаций всех положений объекта. В статье [15] предложено использование методики CAPTURE (Continuous Appendicular and Postural Tracking Using Retroreflector Embedding) – систему продолжительного анализа поведения животного, совмещающая в себе как захват движения, так и глубокое обучение для непрерывного отслеживания 3D-кинематики головы, туловища и конечностей у свободно ведущих себя животных. С помощью данной системы исследуются животные, для которых сложно получить большой объем информации с датчиков.

Помимо исследованных видов выделяют runtime анимацию – процедурная анимация генерируется в режиме реального времени и использует особенности окружающей среды физической эмуляции и зависит от факторов сцены: в частности, ландшафта – закончилась ветка, на ветке препятствие и т.п.

Этот вид анимации часто не создают полностью процедурной, а используют смешанный подход – при обычной анимации, например, «передвижение по прямой ветке» встречается конец ветки, то при успешном поиске новой подходящей ветки в доступном диапазоне конечностей, будут изменены углы конечностей, для предопределенной анимации перехода на другую ветку.

Анимацию, разделяют на:

– объектную анимацию – используется покадровая съёмка перемещения объектов на сцене,

– вершинную (вертексную) анимацию – анимация формируется за счет перемещения каждой вершины модели от одной позиции к другой,

– скелетную анимацию [16] – скелет, представляющий собой иерархическую структуру костей (bones), в которой каждая последующая кость «привязана» к предыдущей, и, соответственно, каждая последующая

кость повторяет за предыдущей движения и повороты с учётом иерархии. Данные кости между собой связаны, поэтому, когда двигается одна кость, за ней двигаются и все остальные иерархически привязанные к ней кости. При использовании данной технологии достаточно задавать положение и повороты (углы) костей скелета.

Процедурную анимацию авторы [17] подразделяют, в свою очередь, на прямую и инверсную:

– прямая – зная положение костей скелета, можно получить конечную точку в пространстве, которую достигнет конечная кость-потомок (в нашем случае конечность двупалого ленивца), т.е. воздействие на иерархическую цепочку костей передается от предков к потомкам,

– инверсная кинематика работает с зеркальным случаем, когда есть точка в пространстве, которую нужно достигнуть, для этого рассчитываются положения костей обратно - от потомков к предкам.

Исходя из вышеизложенного можно отметить, что процедурная скелетная runtime анимация на основе инверсной (обратной) кинематики сможет оптимально и наиболее физически корректно моделировать поведение двупалого ленивца, т.к. ленивцы обычно в природе большую часть времени находятся и перемещаются на ветке, удерживаясь в подвешенном положении.

1.3 Анализ существующих алгоритмов решения задач инверсной кинематики

Отметим, что обычно авторы [18] выделяют две основные группы методов, с помощью которых решают задачи инверсной кинематики – аналитические и численные.

Аналитические методы используются для поиска возможных решений в зависимости параметров (длины плеча, исходного положения и ограничений вращения). Стоит отметить, что данная группа методов имеет ограничения по масштабируемости, когда инверсная кинематика применяется к большому числу степеней свободы.

Из этого следует, что более универсальными и обеспечивающими необходимый баланс между реалистичными перемещениями и вычислительными ресурсами представляются численные методы [19]. Данная группа методов использует итеративный подход, при котором в ходе итераций достигается удовлетворительное решение.

Среди численных методов алгоритмов инверсной кинематики выделяют [18] три основные группы:

1) Методы Якоби - группа методов, представляющая собой линейную аппроксимацию фактического движения кинематической скелетной цепочки, этот подход основывается на решениях первого порядка, что означает применимость к одной независимой кости. Существует проблема сингулярности – особенность, которая может возникнуть, если матрица содержит ряд нулей, тогда изменение углов костей не сможет привести к желаемому значению положения конечного эффектора.

2) Методы Ньютона, в отличие от решений первого порядка в методах Якоби, данный подход основывается на разложении Тейлора второго порядка и приводят к плавным движениям без колебаний. Однако минусом данного подхода являются высокие вычислительные затраты и сложная реализация.

3) Эвристические алгоритмы инверсной кинематики, такие как CCD - Cyclic coordinate descent, Triangulation, FABRIK - Forward and Backward Reaching Inverse Kinematics, M-FABRIK (усовершенствованный FABRIK).

Рассмотрим подробнее следующие эвристические алгоритмы инверсной кинематики:

CCD (Cyclic coordinate descent) алгоритм был впервые был представлен в 1991 году [20] и представляет собой эвристический метод, работающий итеративно и позволяющий быстро находить решения. И на сегодняшний день является довольно часто используемым алгоритмом в компьютерных играх. Однако при его работе отмечаются и негативные моменты, такие как возможные нереалистичные анимации и сложность его реализации для нескольких конечных эффекторов. Его целесообразно использовать для

работы с последовательными цепочками и, следовательно, не с наборами с несколькими конечными эффекторами.

Суть, метода CCD заключается во вращении каждого сустава на угол вокруг оси вращения, начиная с конечного эффектора, продолжая его родительским, проходя через все кости, и так далее до корневой кости. Это делается итеративно, пока конечный эффектор не окажется в искомом положении или достаточно близко к конечной точке.

Triangulation – эвристический алгоритм, предложенный авторами методики [21] рассчитывает косинус (1, 2) для расчета каждого угла соединения, начиная с корня кинематической цепи для нахождения положения, при котором достигается целевая координата.

Косинус рассчитывается по формуле, в которой длины его сторон с косинусами его углов:

Математическая формула такова:

$$b^2 = a^2 + c^2 - 2ac\ cos\delta_b \quad (1)$$

где a, b, c – длины сторон треугольника, а δ_b– угол противоположный стороне b. Преобразовав эту формулу, ее (2) используют для нахождения углов треугольника:

$$\delta_b = cos^{-1}(\frac{-(b^2-a^2-c^2)}{2ac}) \quad (2)$$

Данный алгоритм итерируется через каждую кость от начальной к конечной. На каждом узле рассчитывается один и тот же набор уравнений, таким образом, к последнему узлу решение будет найдено, если это возможно. Как показано на рисунке 5, при применении данного алгоритма для каждой кости ищется угол, чтобы можно было составить треугольник abc. Таким образом, требуется только одна итерация для достижения цели, если она достижима.

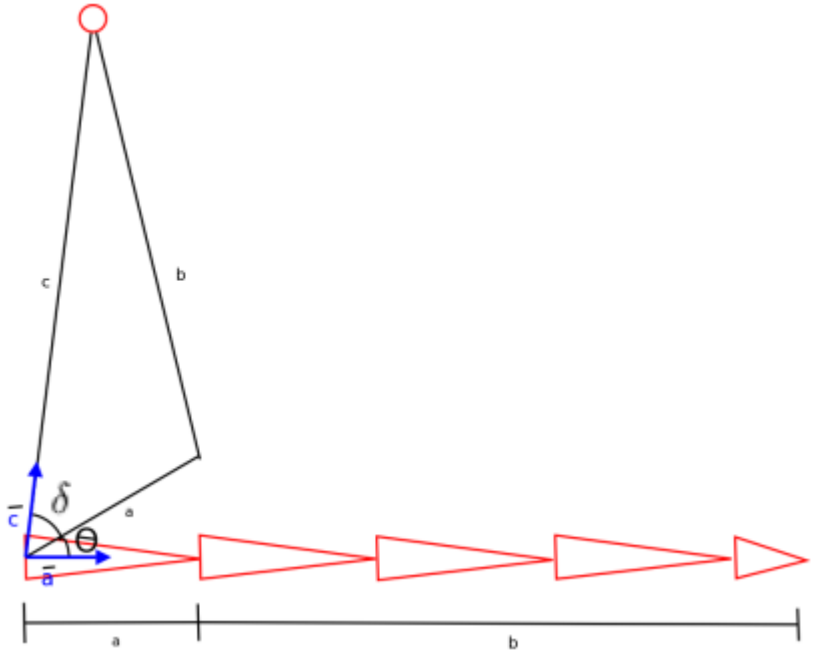

Рисунок 5 – Графическое представление алгоритма Triangulation [21]

Однако минусом алгоритма Triangulation является то, что он не может принимать ограничения по оси вращения, что часто бывает необходимо при реализации задач инверсной кинематики.

FABRIK (Forward and Backward Reaching Inverse Kinematics) эвристический алгоритм [22], с расширением алгоритма в части двухэтапного обхода препятствий [23] – extended FABRIK, и усовершенствованиями степени свободы каждого узла до 1-DOF с учетом родительских и дочерних ограничений узлов [24] – FABRIK-R, использует позиции, уже рассчитанные в режимах прямой и обратной кинематики. В алгоритме FABRIK происходит обход всей цепи, начиная с последнего узла, с поправкой угла каждого обойдённого узла, после чего, происходит обход цепи уже в обратном направлении как показано на рисунке 6:

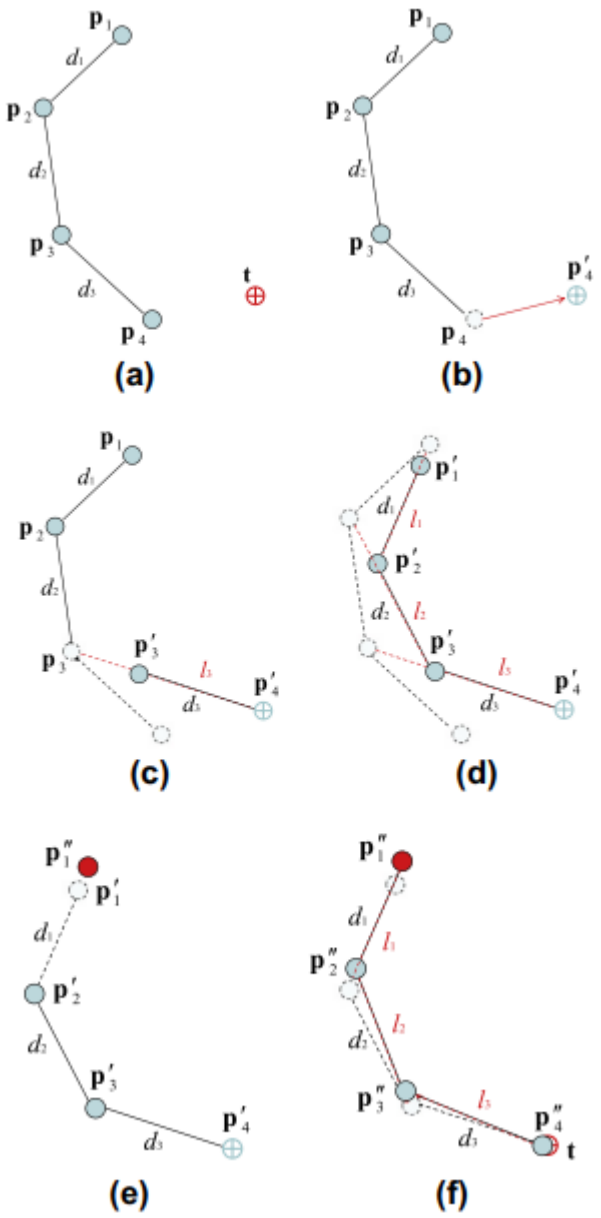

Рисунок 6 – Пример для одного манипулятора с 4 узловыми точками [22]

Данный алгоритм реализуется на основе векторов, когда известны длины плеч. Сначала алгоритм выстраивает на основе инверсной кинематики цепь костей с нодами до цели - конечной точки t, начиная с узла p_4 до начального узла p_1 на новые позиции p'.

Затем, оценивается получившееся смещение $p_1' - p_1$, и если оно больше заранее установленного допустимого значения margin, то теперь передвигаем узлы в прямом порядке, начиная со стартовой позиции до конечной.

При достижении конечной точки, аналогично, оценивается допустимый уровень смещения и процесс производится в обратном порядке и так далее, до достижения приемлемого margin.

При сравнении эвристических методов CCD и FABRIK, FABRIK оказывается быстрее [25] во всех проведённых тестах - на количество итераций, на среднее количество ошибок и на время выполнения функции. FABRIK при тестировании оказывался более плавным в визуальном плане, что приводит к очевидному выводу о более высокой эффективности применения FABRIK для целей компьютерной анимации в режиме реального времени. Однако автор исследования при этом оговаривается, что для случаев, когда цель находилась едва в пределах досягаемости, то FABRIK оказывался менее эффективен.

Авторы работы [22] приводят сравнительные данные о производительности некоторых эвристических алгоритмов как показано в таблице 5:

Таблица 5 – Сравнительная таблица производительности алгоритмов инверсной кинематики

	Reachable Target			
	Number of iterations	Matlab.exe. Time (sec.)	Time per iteration (in msec)	Iterations per second
1	2	3	4	5
FABRIC	15,461	0,01328	0,86	1164
CCD	26,308	0,12356	4,69	213

Продолжение таблицы 5

1	2	3	4	5
Jacobian Transpose	1311,19	12,98947	9,9	101
Jacobian DLS	998,648	10,48051	10,49	95
Jacobian SVD-DLS	808,797	9,29652	11,5	87
FTL	21,125	0,02045	0,97	1033
Triangulation	1,0	0,05747	57,47	21

M-FABRIK – усовершенствование [26] алгоритма FABRIK с ведением понятия “допустимой области” из которой манипулятор может достичь цели.

В примере на рисунке 7 показана полная итерация, с обозначением допустимой области, после стадии d дополнительная итерация не требуется поскольку база может двигаться в обозначенной области, и обратная итерация в данном случае не понадобится:

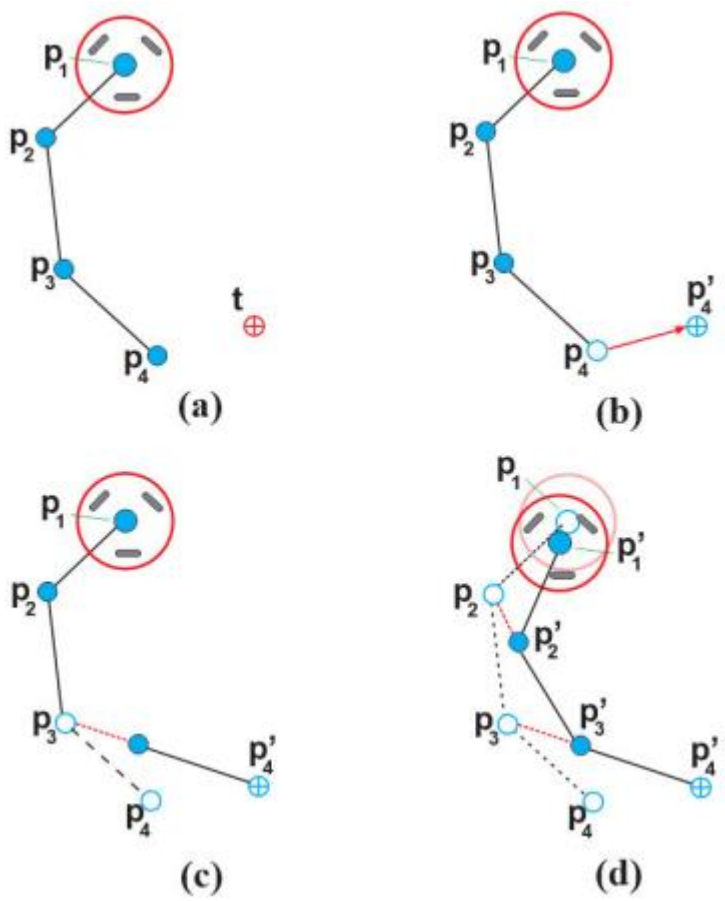

Рисунок 7 – M-FABRIK [26]

Основным преимуществом предлагаемого подхода авторы [26] называют возможность выбора базовой позиции для выполнения дополнительных требований помимо достижения цели: обход препятствий, возможность соблюдения ограничений суставов, повышение манипулятивности, возможность достижения конечной точки если она находится вне доступа, но с учетом “допустимой области” базы манипулятора становится доступной. На рисунке 8 показан результат работы алгоритма по успешному решению отслеживания пути с допустимой областью:

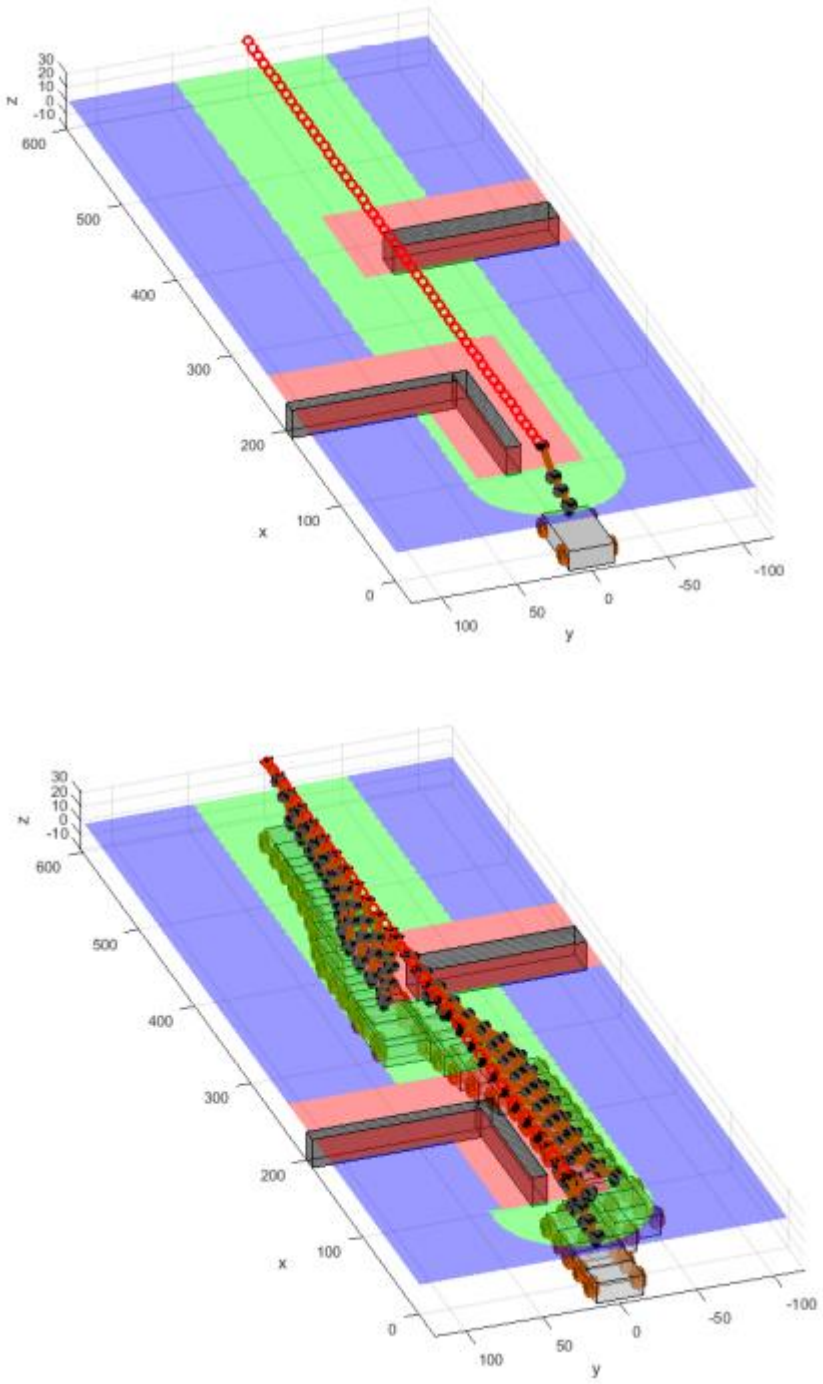

Рисунок 8 – Пример работы M-FABRIK с “допустимой областью” смещения базы (допустимая область показана зеленым цветом) [26]

Обобщим достоинства и недостатки основных алгоритмов инверсной кинематики. Данная информация кратко представлена в таблице 6:

Таблица 6 – Достоинства и недостатки алгоритмов инверсной кинематики

Группа	Особенности: преимущества и недостатки
1	2
1. Методы Якоби (основаны на решениях первого порядка)	Группа методов, представляющая собой линейную аппроксимацию фактического движения кинематической скелетной цепочки, этот подход основывается на решениях первого порядка, что означает применимость к одной независимой кости. Проблема сингулярности – особенность, которая может возникнуть, если матрица содержит ряд нулей, тогда изменение углов костей не сможет привести к желаемому значению положения конечного эффектора.
2. Методы Ньютона (основаны на разложении Тейлора второго порядка)	Достоинством является плавность движений без колебаний. Однако минусом данного подхода являются высокие вычислительные затраты и сложная реализация.
3. Эвристические алгоритмы (CCD, Triangulation, FABRIK, M-FABRIK)	
3.1. CCD	Часто бывают нереалистичные (неплавные) движения, из-за достаточно больших углов, при расчете. Сложность создания ограничений и соответственно реализации для нескольких костей (т.к. действует локально в каждом суставе).

Продолжение таблицы 6

1	2
3.2. Triangulation	Преимущество в том, что он может достичь цели всего за одну итерацию. На каждом узле рассчитывается один и тот же набор уравнений, таким образом, к последнему узлу решение будет найдено, если это возможно. Имеет низкие вычислительные затраты, однако его результаты часто визуально неестественны. Минусом данного алгоритма является то, что он не может принимать ограничения по оси вращения, что часто бывает необходимо при реализации задач инверсной кинематики.
3.3. FABRIK	Достоинства - простота, гибкость, позволяющая легко адаптироваться к различным задачам, низкие вычислительные затраты и его эффективность при решении замкнутых циклов или задач с несколькими концевыми эффекторами В отличие от CCD, в которой основное внимание уделяется движению вблизи конечного эффектора, он распределяет движения по всем суставам.
3.4. M-FABRIK	Данный алгоритм - усовершенствование алгоритма FABRIK с ведением понятия “допустимой области” из которой манипулятор может достичь цели. Основное преимущество: возможность выбора базовой позиции для выполнения дополнительных требований помимо достижения цели: обход препятствий, возможность соблюдения ограничений суставов, повышение манипулятивности, возможность достижения конечной точки если она находится вне доступа, но с учетом “допустимой области” базы манипулятора становится доступной.

2. РАЗРАБОТКА АЛГОРИТМА ПРОЦЕДУРНОЙ АНИМАЦИИ И ВЫБОР ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РЕАЛИЗАЦИИ

2.1 Анализ и подбор программного обеспечения

При выполнении данной работы важен выбор игрового движка для реализации системы процедурной анимации в реальном времени. На данный момент самыми распространенными игровыми движками в медиа индустрии являются Unity 3D и Unreal Engine 5. Оба движка практически схожи по своим возможностям, включая область процедурной анимации. Можно отметить, что в источниках [27] и [28] проводится сравнительный анализ этих игровых движков, отмечая их преимущества и недостатки.

Для реализации алгоритма был выбран Unreal Engine 5, ввиду его ряда преимуществ:

- Встроенная в движок система визуального программирования Blueprint, которая облегчает разработку и упрощает внесение изменений в анимационную систему (рисунок 9).

– Встроенная реализация алгоритмов обратной кинематики, что обеспечивает готовые и удобные инструменты для создания реалистичных процедурных анимаций и управления движениями конечностей и туловища (рисунок 9).

– Опыт разработки на данном игровом движке, что облегчает адаптацию и использование существующих ресурсов и инструментов разработки.

Выбор Unreal Engine 5 с учетом указанных преимуществ позволит упростить и ускорить процесс разработки процедурной анимации для данной системы.

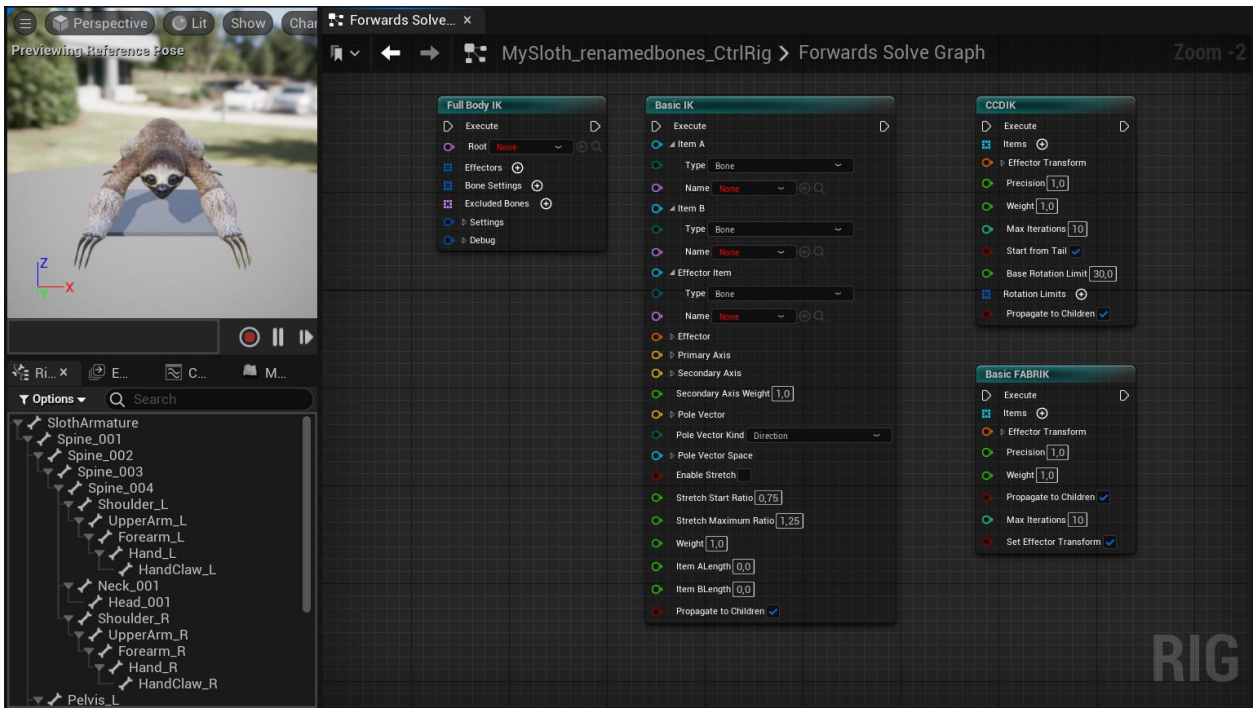

Рисунок 9 – Система визуального программирования Blueprint в Unreal Engine 5

Оптимальным представляется реализация риггинга при помощи ПО Blender.

Риггинг – это процесс создания структуры (рига) для трехмерной модели, в контексте данной работы – модели ленивца, которая позволяет управлять и анимировать ее частями, такими как кости, суставы и деформационные элементы. Риггинг играет ключевую роль в анимации персонажей и объектов, позволяя их анимировать, в том числе процедурно. В данной книге авторы подробно рассматривают процесс риггинга в Blender [29].

Риггинг включает в себя создание и настройку костной и суставной структуры модели, присваивание весов костям, установку контроллеров для управления движениями и деформациями, а также настройку ограничений и иерархии.

Программа Blender предлагает ряд уникальных преимуществ для риггинга, которые делают ее удобным инструментом по сравнению с другими программами для моделирования. Преимущества и недостатки наиболее

распространенных программ для 3D моделирования рассмотрены в статьях [30] и [31].

Отметим следующие преимущества, позволяющих реализовать риггинг на основе Blender:

– Бесплатность и открытый исходный код. Blender является одним из немногих профессиональных 3D-пакетов с открытым исходным кодом.

– Интегрированная анимационная система. Blender предлагает мощные инструменты для создания анимаций, интегрированные в единый рабочий процесс. Внутри Blender можно моделировать, риггировать и анимировать объекты без необходимости переключения между различными программами. Это значительно повышает эффективность работы и упрощает взаимодействие между различными аспектами создания 3D-контента.

– Гибкость и масштабируемость. Blender обладает гибкостью и масштабируемостью в создании ригов. Программой предоставляется широкий спектр инструментов для создания каркаса кости, управления искажениями, настройки контроллеров и определения ограничений движения.

Эти преимущества делают Blender оптимальным выбором для риггинга, обеспечивая мощные инструменты, гибкость и возможности расширения.

2.3 Разработка схемы алгоритма

В ходе исследования анатомических особенностей ленивцев и особенностей их перемещения на ветвях, проведенные в части анализа предметной области, были выявлены следующей особенности локомоции ленивцев:

Диагональность перестановки конечностей. По данным научных исследований кинематики ленивца [5, 32] диагональность локомоции часто свойственна двупалым ленивцам при перемещении на ветвях, очередность перестановки конечностей при этом выявлена следующая:

– правая задняя (rh), левая передняя (lf), левая задняя (lh), правая передняя (rf), то есть сначала ленивец передвигает заднюю лапу вперед и

полностью вытягивает ее, чтобы достичь следующего участка ветки, когда задняя лапа достигает новой точки опоры, ленивец опускает переднюю лапу и приступает к перемещению передних конечностей и так далее.

В статье [5] приводятся данные проведенных авторами испытаний (рисунок 10), где видно, что наиболее часто при довольно быстром движении ленивцами используется тип перемещения DSDC (диагональная последовательность), в отличие от LSDC (боковая последовательность).

На данном рисунке 10 авторами [5] черными кружками обозначены результаты испытаний при неограниченной локомоции, а белыми кружками - результаты испытаний на стационарном стенде.

Таким образом, с учетом вышеизложенных данных, для цели реализации передвижения ленивца при реализации процедурной анимации используем наиболее часто применяемый тип последовательности перестановки конечностей – диагональный. Для отдельных сценариев поведения, таких как переход со ствола на ветку, применим LSDC для комбинирования видов походки и большей реалистичности.

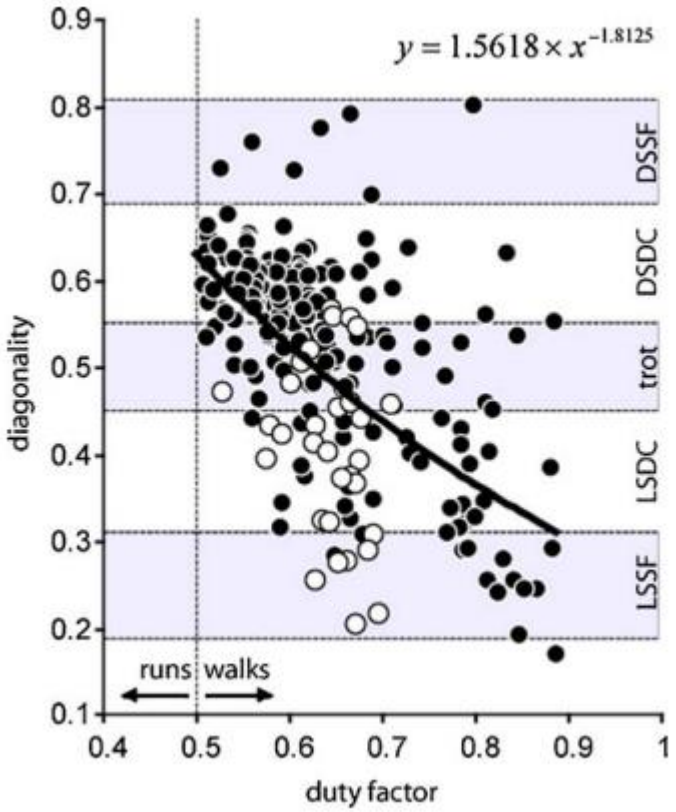

Рисунок 10 – Результаты испытаний видов локомоции ленивца [5]

Установив ограничения на углы суставов в риге для соблюдения физиологических ограничений ленивца на основе таблицы 2 мы получим модель ленивца и сможем контролировать данные ограничения при процедурной анимации при помощи инструмента в Unreal Engine 5 Control Rig.

Общий алгоритм перемещения конечности ленивца можно представить в виде схемы, как это показано на рисунке 11. При этом, как уже отмечалось выше, при создании процедурной анимации будет использоваться диагональный тип перестановки конечностей в последовательности DSDC и LSDC для отдельных сценариев поведения:

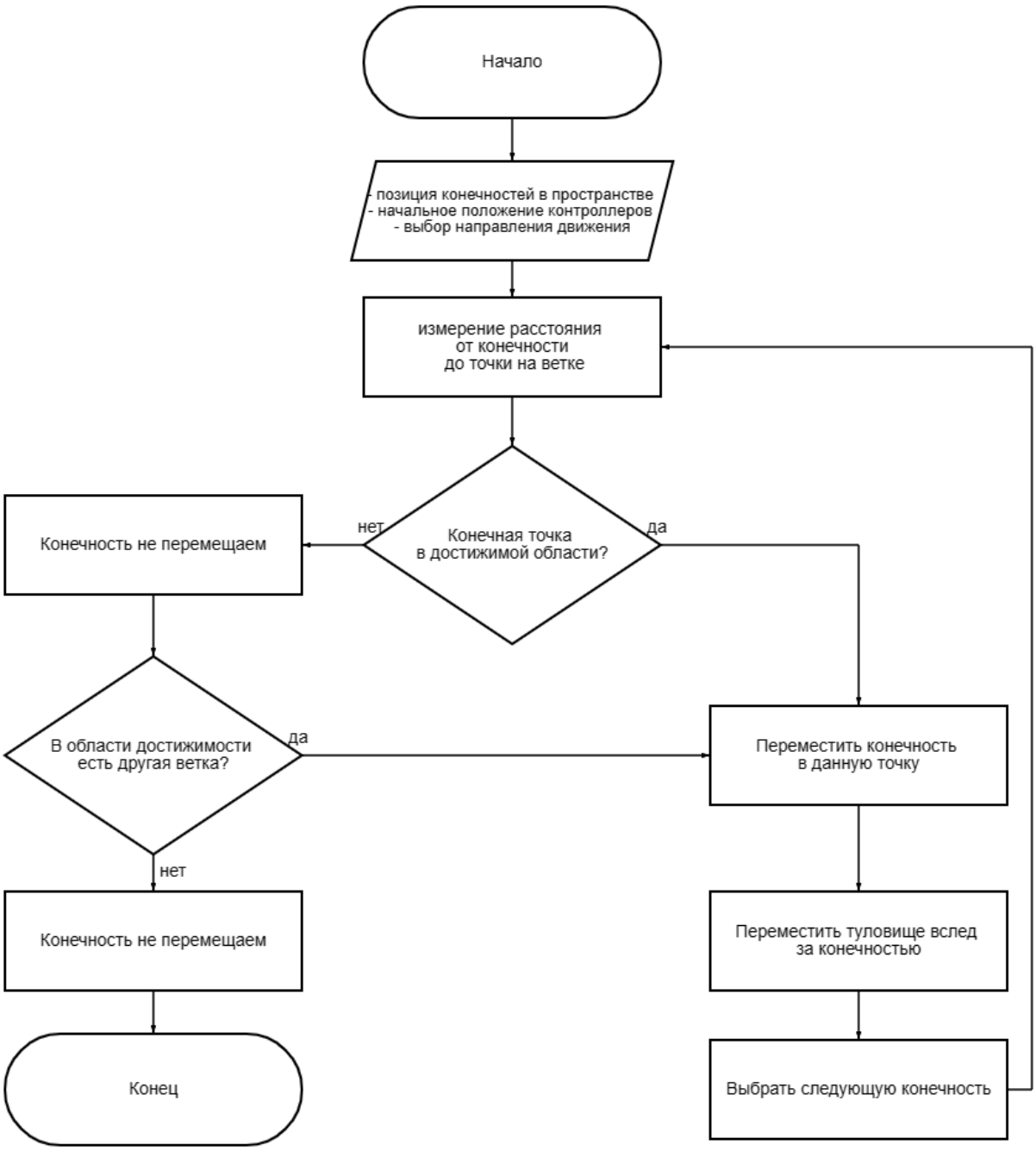

Рисунок 11 – Алгоритм перемещений конечностей ленивца

Общую схему реализации функции поиска следующего положения на ветке эффектора, то есть функции перестановки конечностей, можно представить в следующем виде:

Рисунок 12 – Алгоритм поиска точки для смещения конечности

При реализации алгоритма необходимо учитывать вращение и местоположение тела при перемещении, для вертикального смещения тела блок-схема будет иметь следующий вид на рисунке 13:

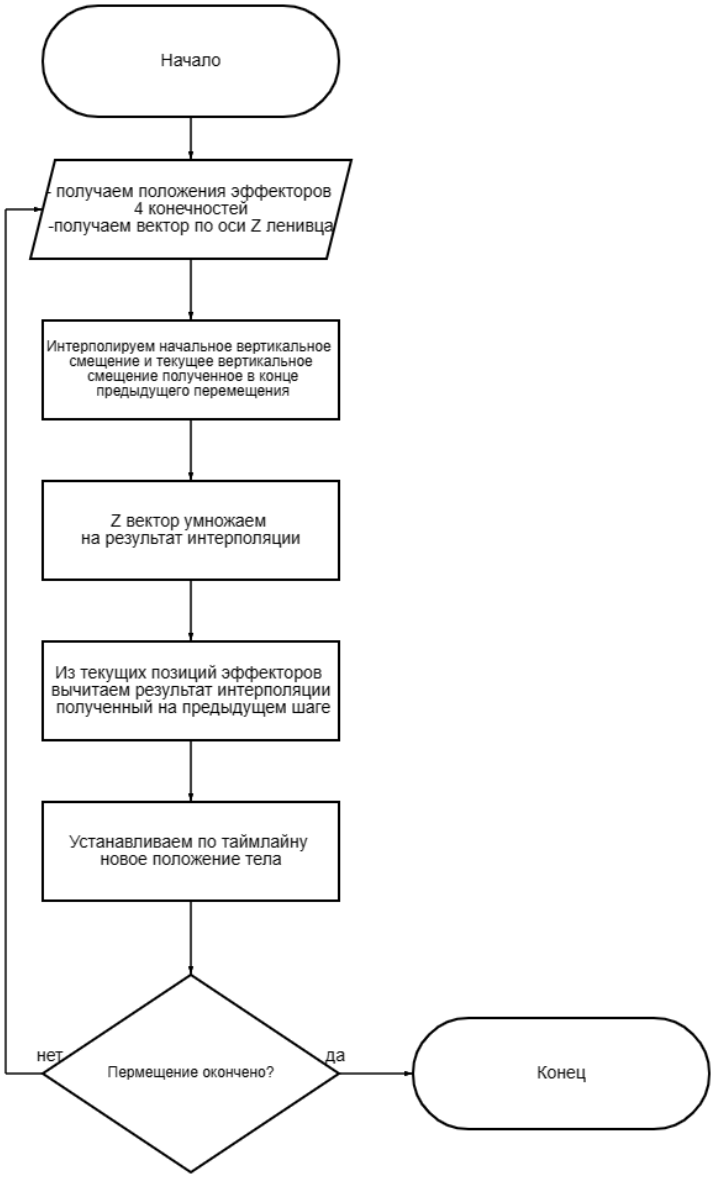

Рисунок 13 – Алгоритм вертикального смещения тела при перестановке эффекторов

Алгоритм горизонтально смещения примет аналогичный вид с учетом того, что для расчета используется вектор по оси У, на основе которого рассчитывается смещение между стартовой позицией эффектора и полученным положением в ходе работы функции расчета следующей позиции.

Алгоритм вращения тела в зависимости от текущего положения конечностей, в тех случаях если ветка имеет поворот примет следующий вид:

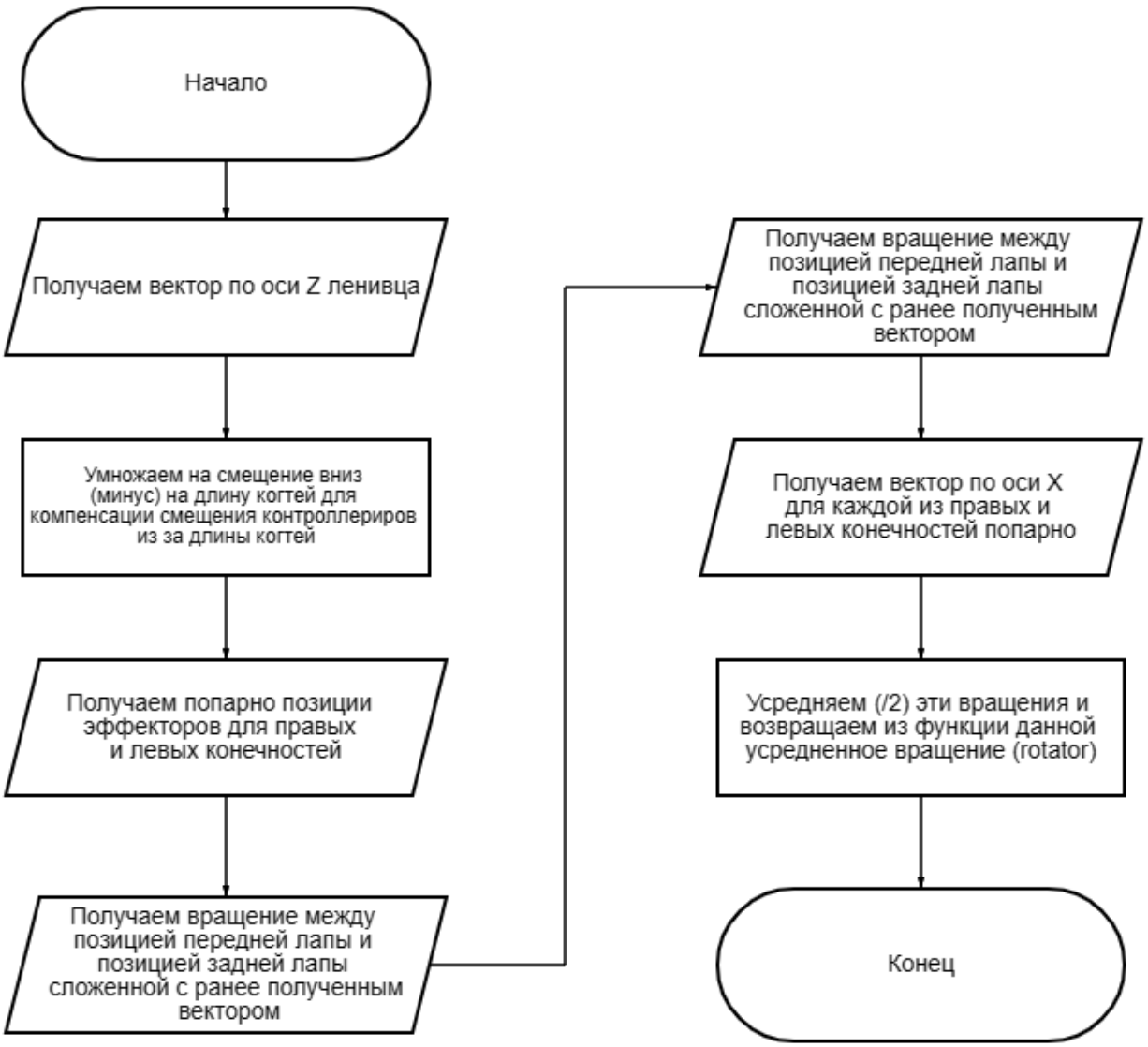

Рисунок 14 – Алгоритм расчета вращения тела при перестановке эффекторов, в случаях, когда ветка имеет поворот

В ходе разработки планируется реализовать комбинированный подход к созданию процедурной анимации: обратной кинематики при перестановке конечностей и усреднения положения тела для смещения корпуса при локомоции, а также использование инверсной кинематики для небольшого смещения контроллеров голов и спины, для особенностей при переходах с ветки на ствол и обратно. Данный подход позволит получить оптимальный результат.

2.3 Подбор алгоритмов для реализации особенностей перемещения ленивца

В процессе реализации алгоритма процедурной анимации для ленивца, требуется принять решение относительно выбора методов обратной кинематики, которые будут применяться к генерируемым движениям животного. Это шаг, который влияет на реалистичность визуальной составляющей анимации.

Для достижения этой цели, необходимо учесть особенности анатомии ленивца и его типичные движения на ветвях. Рассмотрение различных алгоритмов позволяет определить, каким образом будут вычисляться положения и углы суставов, чтобы достичь желаемых движений и позиций.

В настоящей работе ранее рассматривались различные алгоритмы кинематики: как прямая, так и обратная кинематика. В силу особенностей реализации алгоритмов они дают различающиеся результаты работы костной иерархии.

Одним из возможных подходов может быть применение алгоритма FABRIK (Forward and Backward Reaching Inverse Kinematics), который базируется на последовательном расчете позиций суставов для достижения желаемой конечной точки, графическое представление автора Andreas Aristidou [22] показано на рисунке 15:

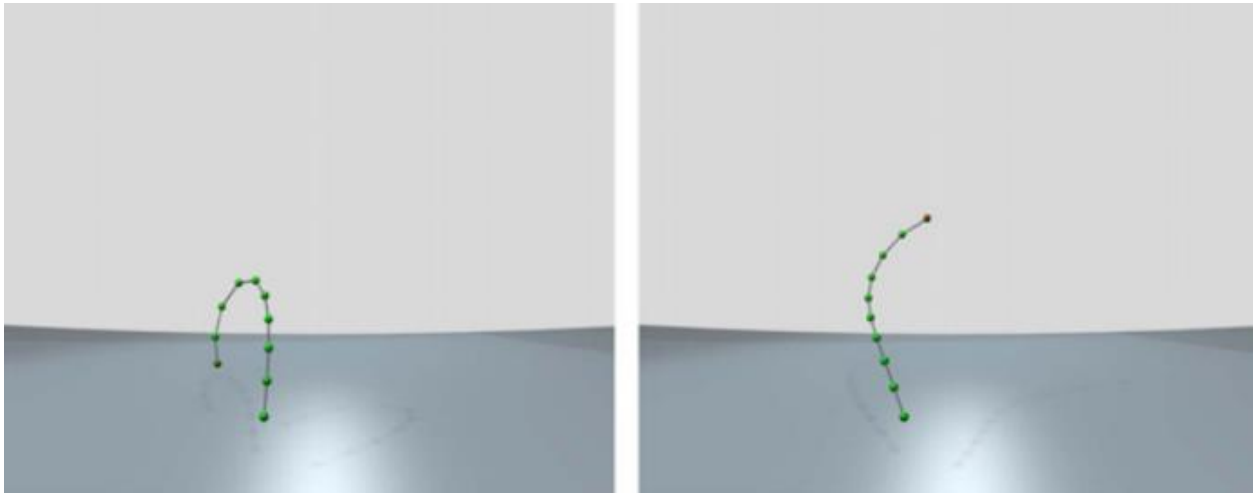

Рисунок 15 – Графическое представление работы алгоритма FABRIK [22]

Другим вариантом можно рассмотреть использование метода CCD (Cyclic Coordinate Descent), который основывается на итеративном обновлении углов сочленений конечностей с целью достижения заданных целевых точек. Графическое представление, выполненное Andreas Aristidou [22] показано на рисунке 16:

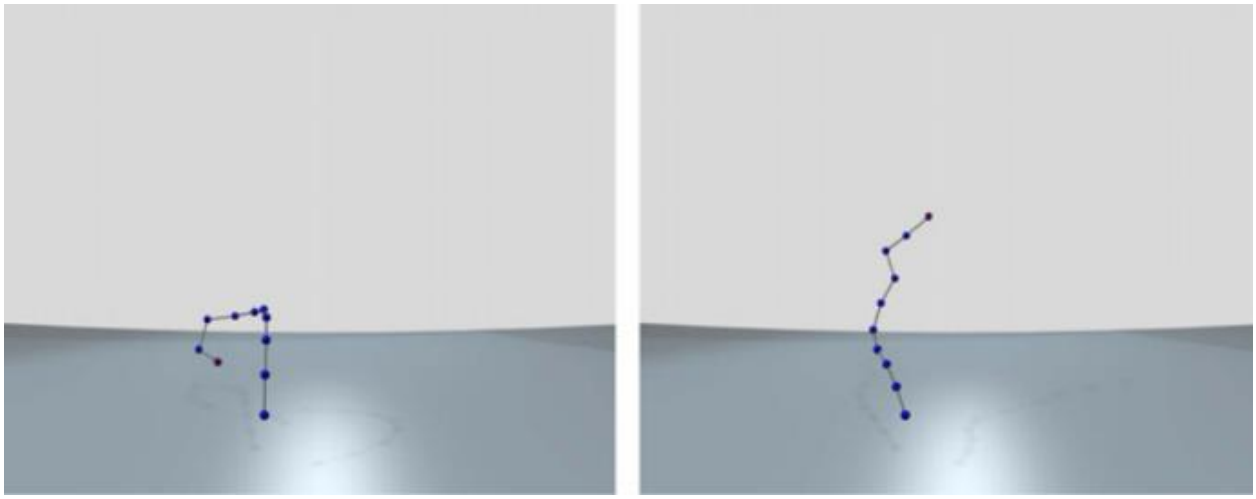

Рисунок 16 – Графическое представление работы алгоритма CCD [22]

Исходя из вышеизложенного, можно отметить, что CCD позволяет моделировать ломаную линию в трехмерном пространстве, представляя конечность как последовательность соединенных суставов. Через итеративные шаги, алгоритм CCD приближается к целевой точке, корректируя углы суставов, чтобы достичь желаемого положения. С другой стороны, алгоритм FABRIK стремится привести систему в гладкое дугообразное состояние. Он основывается на двух проходах: прямом и обратном. Прямой проход начинается с начального положения и стремится достичь целевой точки, а обратный проход обеспечивает вычисление углов суставов, необходимых для достижения желаемого положения.

Таким образом, для реализации обратной кинематики конечностей ленивца лучше подойдет алгоритм CCD, поскольку руки и ноги ленивца имеют разнонаправленные сгибы костей конечности.

При экспериментального тестировании инверсной кинематики на ПО Unreal Engine 5 заметно, что CCD реализуется более физиологически корректно в части плечевых костей, как это можно увидеть на рисунке 17:

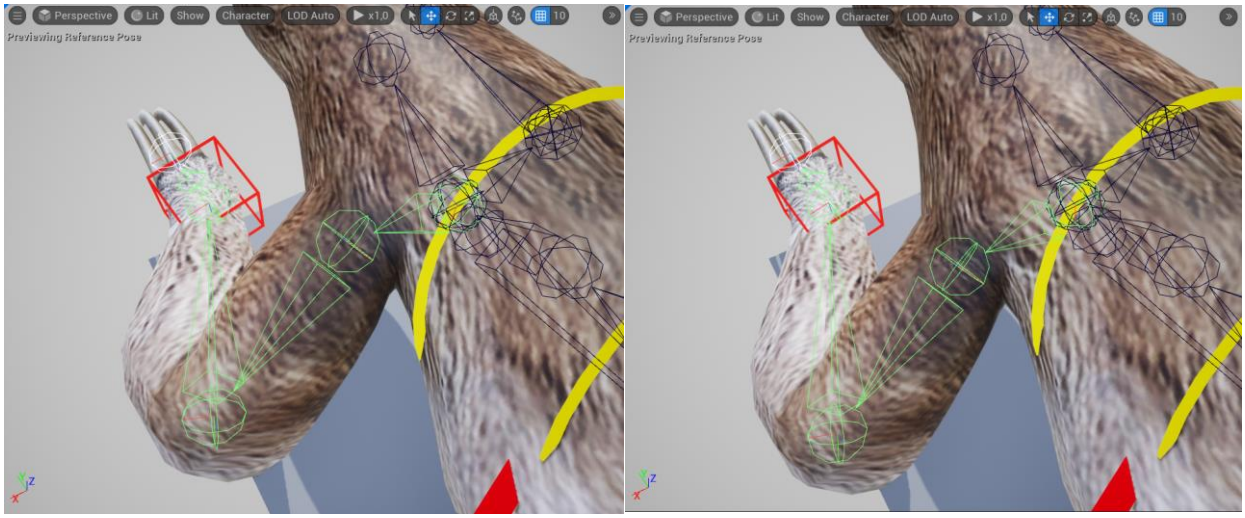

Рисунок 17 – Пример работы алгоритма на цепочке костей, образующих руку ленивца: слева для алгоритма CCD, справа для алгоритма FABRIK

Для реализации костей позвоночника применена система прямой кинематики, поскольку применение инверсной кинематики дает неестественные результаты, как это показано на рисунке 18:

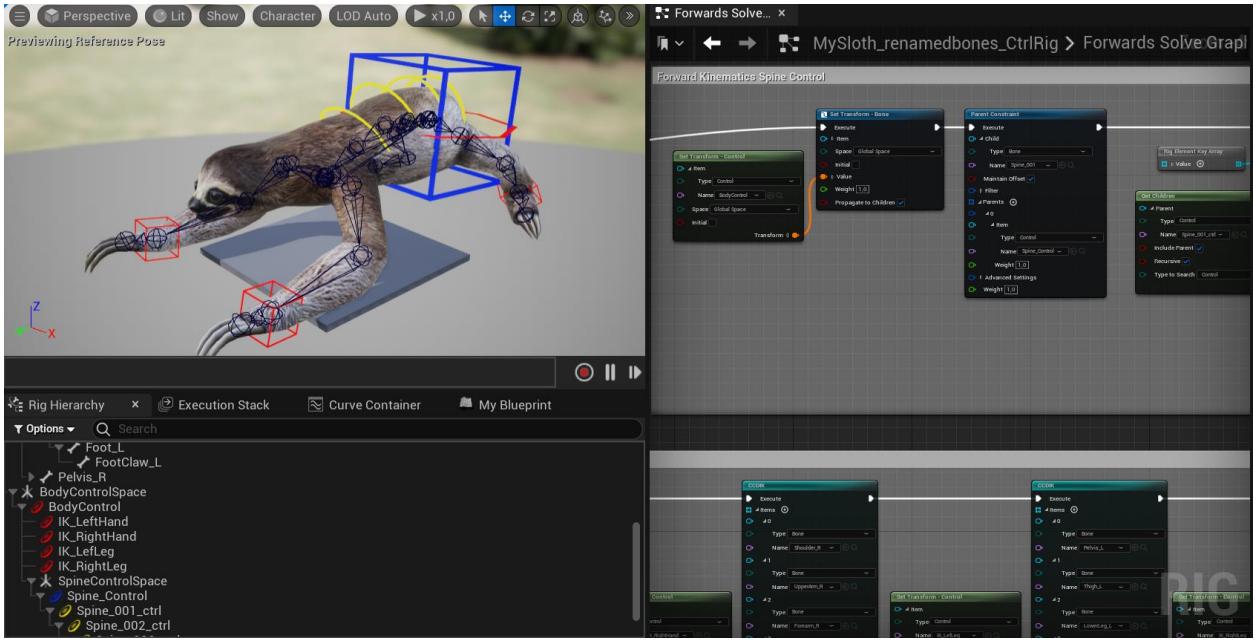

Рисунок 18 – Пример с прямой кинематикой для цепочки костей позвоночника

Подытоживая вышеизложенное, оптимальным представляется реализовать кинематику локомоции ленивца применив алгоритм инверсной кинематики CCD (Cyclic Coordinate Descent) для конечностей и соответствующее усреднение координат конечностей и вращения для цели интерполяции координат и вращения тела.

3. ПРОГРАММНАЯ ИМПЛЕМЕНТАЦИЯ АЛГОРИТМА ПРОЦЕДУРНОЙ АНИМАЦИИ ЛЕНИВЦА

3.1 Проектирование алгоритма модели

Для реализации системы контроля конечностей (контроллеров) модели ленивца реализуем следующие шаги:

1) Назначим иерархию для каждой из четырех конечностей животного с четырехзвенной иерархической структурой, начиная с родительской кости:

– начало: Shoulder\Pelvis для передних\задних конечностей,

– следующая кость: UpperArm\Thigh для передних\задних конечностей,

– следующая кость: ForeArm\LowerLeg для передних\задних конечностей соответственно,

– конечные эффекторы: Hand\Foot.

Более наглядно визуализация эффекторов представлены на рисунке 19:

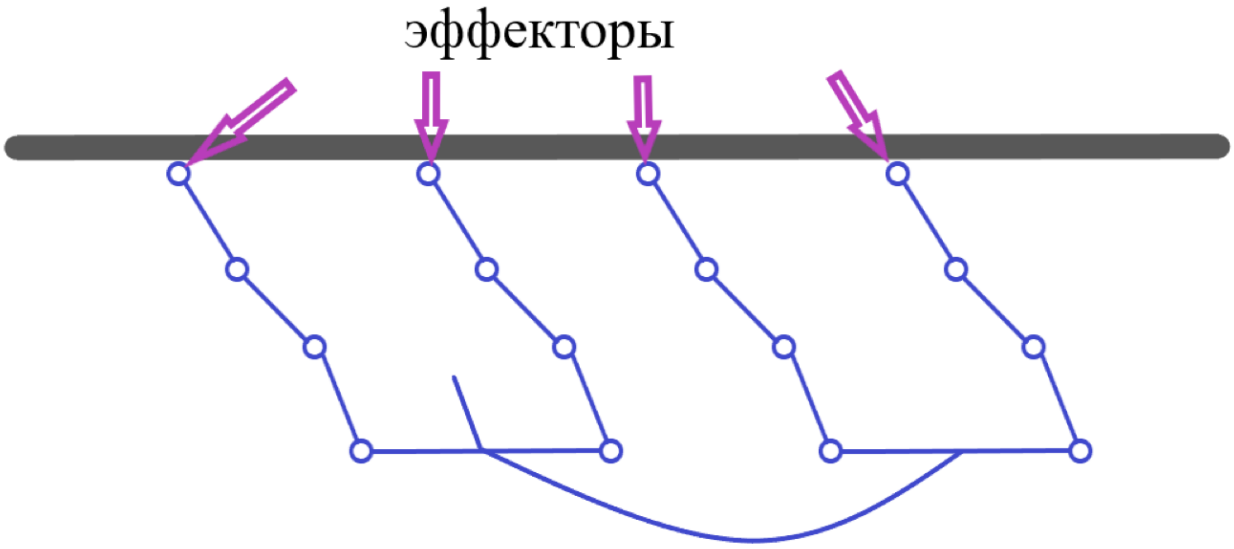

Рисунок 19 – Конечные эффекторы при реализации инверсной кинематики CCD для четырех конечностей

2) Применим функцию инверсной кинематики CCD, которая позволит передвинуть иерархическую структуру костей в положение конечного эффектора для каждой конечности.

3) Выставим в функции CCD углы ограничений, соответственно для передних и задних конечностей, согласно значениям полученных экспериментальных данных [5], показанных в настоящей работе в таблице 2.

Поскольку скорость ленивца в естественной среде обитания согласно данных исследования [7] варьируется в пределах 0,11 ± 0,04 м/с при свободном перемещении без ограничений и в связи с тем, что время контакта с веткой длилась в 2 раза дольше, чем неподвижная фаза в среднем, то необходимо задать соответствующие переменные с плавающей точкой в blueprint для реализации времени перемещения и значения выставить в этом диапазоне, чтобы получить реалистичную визуализацию.

Для реализации в алгоритме наиболее присущего типа локомоции ленивца, т.е. перестановки конечностей, отметим, что наиболее частый тип перестановки конечностей – DSDC (диагональный тип, как это показано на рисунке 20 ниже):

– правая задняя (rh),

– левая передняя (lf),

– левая задняя (lh),

– правая передняя (rf).

При этом для отдельных сценариев используем второй по частоте тип перестановки конечностей – LSDC (боковая последовательность).

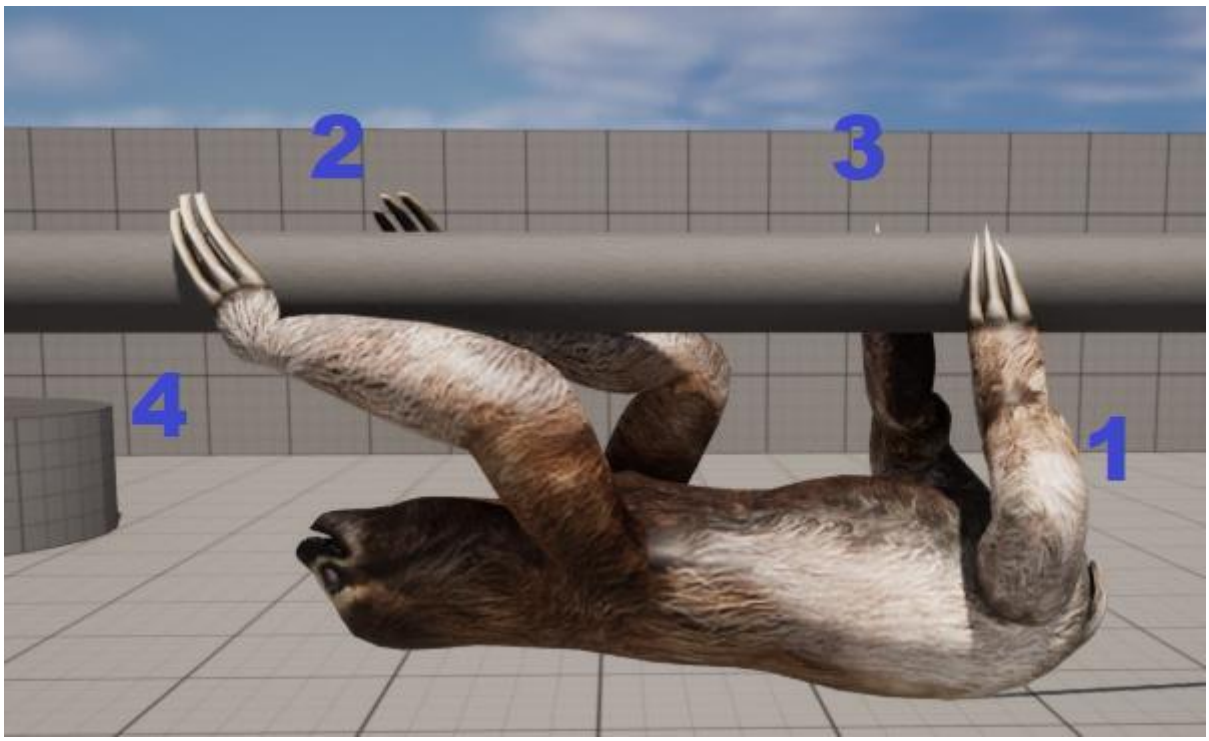

Рисунок 20 – Очередность перестановки конечностей ленивца

3.2 Разработка решения

Для практического применения алгоритма реализации процедурной анимации была импортированная трехмерная модель ленивца в программу Blender. Эта модель включала в себя геометрию тела ленивца и текстуры, в ПО Blender реализована система ригов, позволяющая управлять его конечностями и движениями (рисунок 21).

Для этого создается «арматура» модели, которая представляет собой скелет модели, затем необходимо добавить необходимые кости и настроить связи между моделью и системой костей. Целесообразно реализовать весовое распределение, чтобы модель в процессе анимирования правильно реагировала на движения костей.

Риги были созданы с учетом анатомической структуры ленивца и оптимизированы для реализации процедурной анимации.

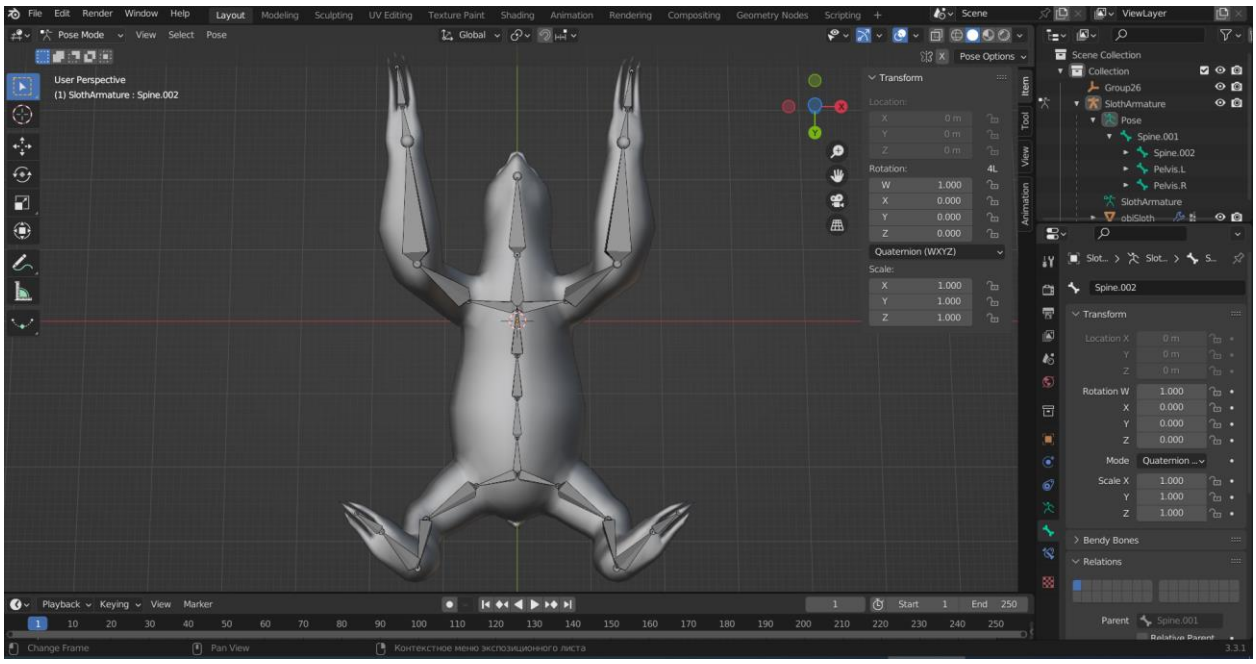

Рисунок 21 – Система ригов в ПО Blender, реализованная для модели ленивца

Следующим шагом реализована интеграция этой модели ленивца с ригами в Unreal Engine 5. С помощью Unreal Engine 5, применить различные алгоритмы процедурной анимации, основанные на физиологических параметрах ленивца, в частности, с использованием системы Control Rig.

Как уже исследовалось ранее, реализация процедурной анимации осуществляется в программе Unreal Engine 5, на основе удобного инструментария визуального программирования blueprints, который предоставляет удобный набор функций и готовые компоненты для визуальной реализации анимаций на сцене.

Редактор Control Rig, поставляемый в составе программы Unreal Engine 5 предоставляет удобный функционал для программной имплементации работы с контроллерами.

Для передачи контроллеров в blueprints модели из Control Rig и управления ими, зададим контроллеры для каждой конечности и присвоим соответствующие наименования:

IK_LeftHand,

IK_RightHand,

IK_RightLeg,

IK LeftLeg,

Head_ctrl,

Sloth_Spine.

Для получения контроля над конечностью в глобальной системе координат над функцией инверсной кинематики CCD с помощью функции «Get Transform – Control» при этом используя следующие параметры: Type Control, Space Global Space.

Пример реализации решения на основе CCDIK в редакторе Control Rig показан на рисунке 22:

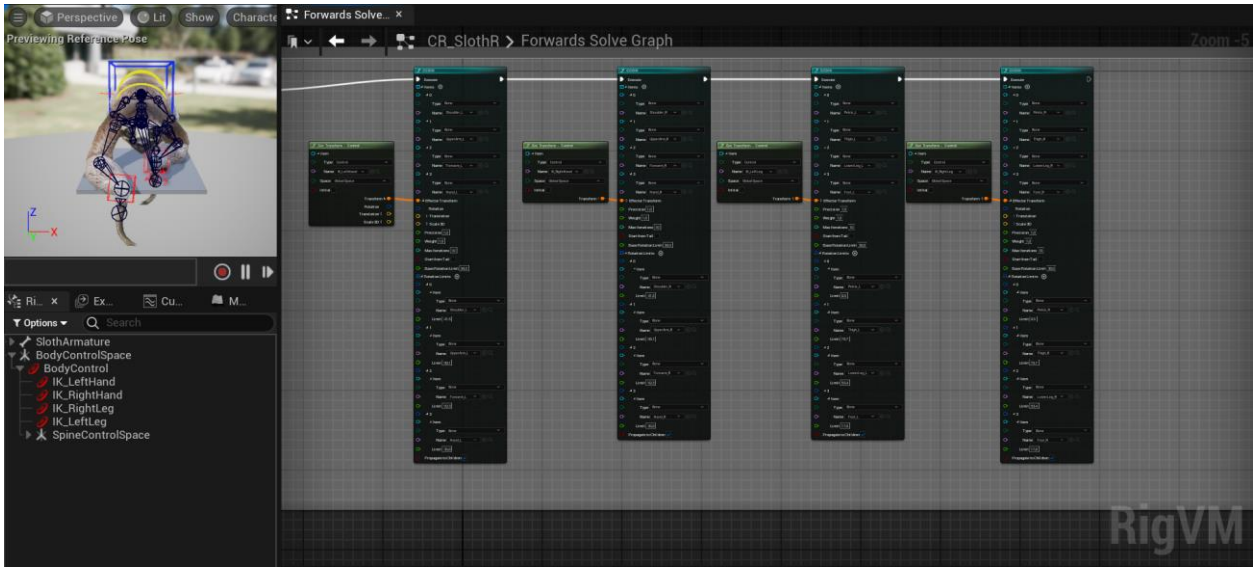

Рисунок 22 – Пример реализации инверсной кинематики CCD в редакторе Control Rig для четырех конечностей

Чтобы имплементировать ограничения углов костей конечностей передадим параметры амплитуд углов, полученные в ходе исследования группы авторов [5], как это было казано в таблице 2, в функцию CCD для каждой кости иерархии соответствующих конечностей.

Целесообразно реализовать перемещение конечностей по типу DSTC через систему компонентов, унаследованных от Actor component, и передать в blueprints модели, в свою очередь унаследованной от родительского класса Pawn, через массив корректную очередность компонентов, присвоив индекс

от 0 до 3 в заданном порядке, а также необходимо инициализировать Control Rig с помощью функции Add Mapped Skeletal Mesh, задать начальные значения relative rotation скелетного меша.

3.3 Работа с моделью и программная имплементация алгоритма на основе физиологических особенностей кинематики ленивца

При реализации процедурной анимации перемещения ленивца очевидно, исходя из его особенностей поведения в окружающей среде, а в частности на деревьях, основным окружением являются ствол и ветви деревьев. Следовательно, при программной имплементации логично разделить пространственную реализацию перемещения на три основных элемента:

- перемещение по ветвям, по их ответвлениям, учитывая возможность того, что ветви в пространстве могут находиться под разными углами,
- перемещение по стволу, в том числе перестановку лап, если надо перейти на ветку, которая находится под иным углом к стволу,
- соответственно, переход со ствола на ветви и обратно.

В дальнейшем при имплементации, программный алгоритм реализуется на основе вышеуказанной поэлементной логики.

Для перемещения конечностей необходимо оценить область достижимости конечности, создав отдельный канал Branch для мешей веток, чтобы обеспечить перемещение эффектора на достижимую область в пределах ветки и обеспечить логику перемещения по ветвям, но не по другому окружению и пустому пространству. Для этого в циклах проверяем дистанцию 50 единиц, с помощью и соответствующих итеративных трассировок, чтобы обеспечить локомоцию как на ветку и следующую ветку, так и перемещение и закрепление конечности с корректной стороны ветки, как это показано на рисунке 23:

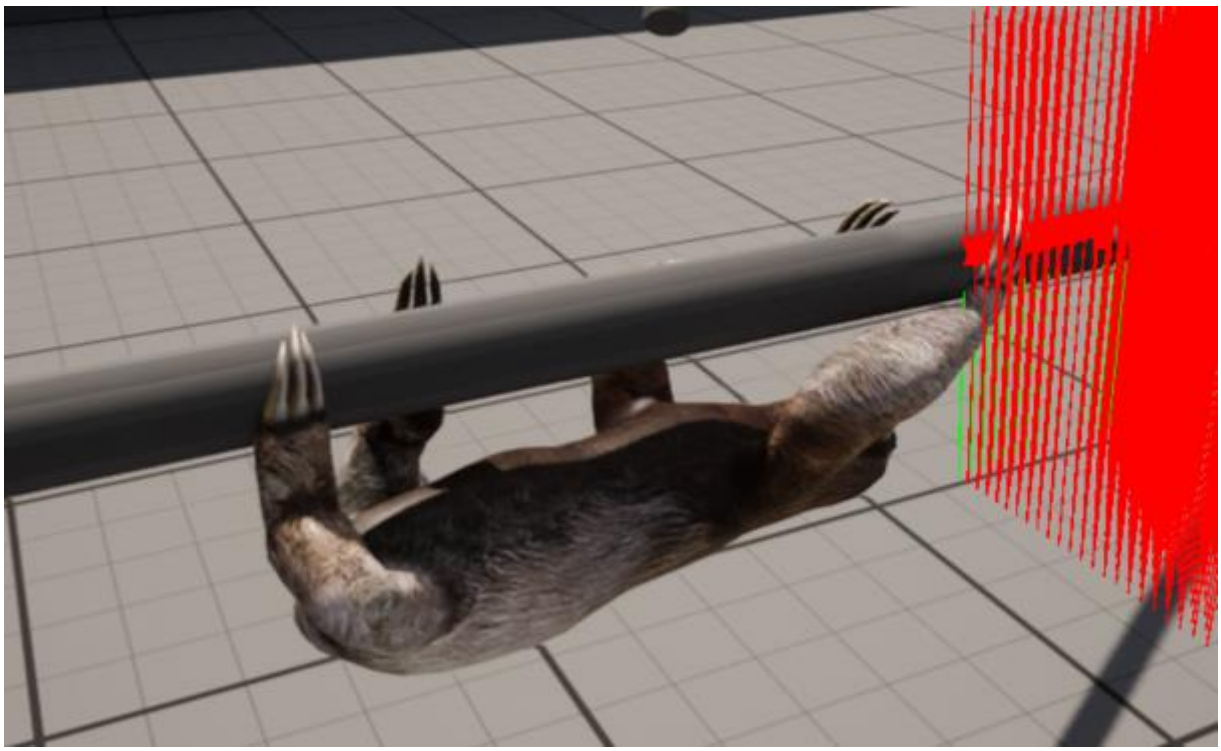

Рисунок 23 – Пример проверки с помощью Line trace достижимости области для перемещения конечности

На вышеуказанном рисунке можно отметить debug trace окрашенный в зеленый цвет, который показывает нахождение точки для перемещения эффектора.

Исходная имплементация алгоритма процедурной анимации основана на линейной трассировке нахождения точки достижимости конечности модели с помощью функции Line trace by channel, при этом произведено тестирование поведения в 3D окружении с помощью функции Sphere trace by channel, что теоретически должно было привести к улучшению производительности за счет меньшего количества трассировок. Применение в алгоритме показано на рисунке 24:

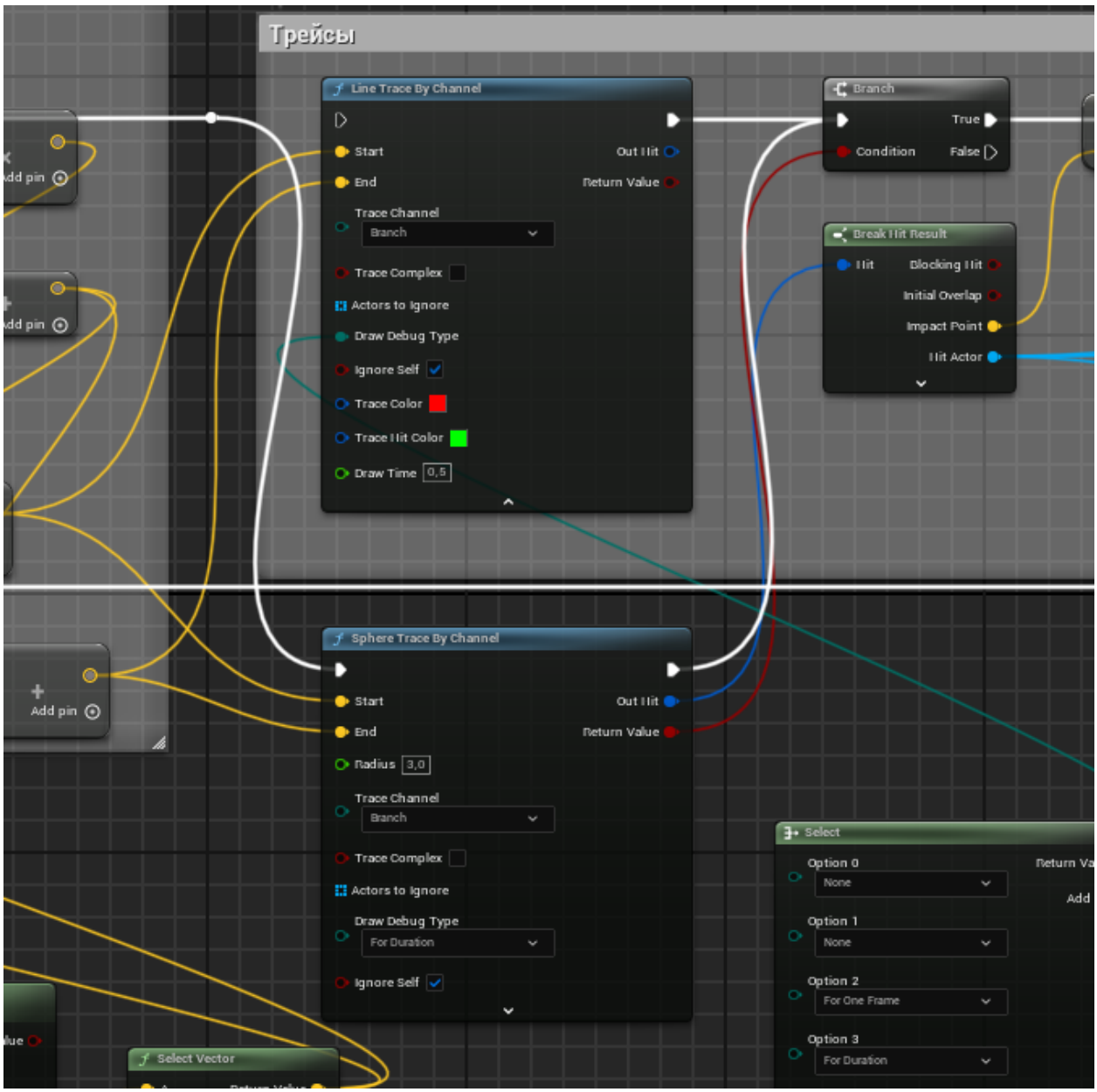

Рисунок 24 – Экспериментальная реализации функции поиска точки достижимости на основе Sphere trace

Однако в ходе экспериментов установлено, что примерно в 20% случаев, когда меши веток находятся рядом и пересекаются, то алгоритм приводит к некорректному нахождению конечной точки, то есть ленивец переходит на другой ветку хотя не должен. Поэтому использовано другое решение для улучшения производительности, а именно: уменьшено количество и дистанция Line trace by channel с 50 до 40 итераций.

Таким образом, алгоритм нахождения точки достижимости будет следующим:

1) определение начальной точки нахождения конечности,
2) определение точки достижимости. Согласно полученным данным, длина шага в ходе экспериментов [5] колебалась от 0,21м. до 0,83м., то в переменную, полученную в результате пересечения Line trace с мешем ветки целесообразно установить в данном диапазоне, определяем позицию с нужной стороны ветки, с помощью right vector,
3) перемещение конечности в найденную точку пересечения, если она находится в допустимом диапазоне. Отметим, что при постановке конечности на меш ветки она должна соответствовать следующей формуле и учитывать толщину ветки по формуле (3) и устанавливаться в зависимости от того, перемещаем мы правую или левую конечности:

$$finPos = startPos+m/2, \quad (3)$$

где posC – конечная позиция эффектора, v – стартовая позиции эффектора, m - толщина меша ветки,
4) если точка пересечения не найдена, то модель либо останавливается, либо перемещается в обратном направлении, либо переходит на следующую ветку,
5) если конечность перемещается в найденную точку, то при смещении скелетной структуры, необходимо учитывать наличие когтей и реализовать смещение на их длину по формуле (4):

$$finHandPos = effPos – clawLen, \quad (4)$$

где finHandPos – конечная позиция эффектора, effPos – текущая расчетная точка конечной позиции, clawLen – длина когтей животного.

Точность нахождения точки на ветке зависит от количества итераций функции Line trace by channel и, соответственно, влияет на частоту кадров.

В работе реализована трассировка дистанции на величину 0,50 м., которая находится в диапазоне от 0,21 м. до 0,83м., выведена в переменную, которую можно изменять в заданном диапазоне, либо выбирать случайное значение из диапазоне.

Для реализации в программе создадим функцию FindNextStep которая будет вызываться в компонентах модели для каждой конечности и принимать следующие параметры:

- vector StartFootPosition – начальная позиция эффектора,
- int Direction – принимает значения 1 и -1 для определения направления движения,
- vector ParentBonePosition – начальная позиция родительской кости конечности,
- boolean ChangeBranch – принимает значения true\false в зависимости от перехода на следующую ветку.

Вышеуказанные параметры начинаются передаваться в функцию FindNextStep при нажатии клавиш ввода (input) в зависимости от автоматического движения с помощью итеративного макроса M_Autorun либо удержания клавиши перемещения.

Возвращаемые значения функции FindNextStep следующие:

- vector ImpactPoint – точка, находящаяся в достижимой области в случае, если Line trace by channel нашел при итерациях точку пересечения,
- boolean PointExist – возвращает true в случае нахождения точки в достижимой области,
- actor Branch – в случае если соседняя ветка находится в достижимой области.

Логика работы данной функции реализована следующим образом:

1) Формула (5) использована для расчета позиции для перестановки лапы:

$$NewHandPos = CurrHandPos + ForwardVector * MaxStep, \quad (5)$$

где NewHandPos – новая позиция эффектора, CurrHandPos – текущая позиция перемещаемого эффектора, ForwardVector – вектор направления движения, MaxStep – максимальная дальность возможного поиска координат точки для перемещения конечности (причем, если не найдена точка, то при следующей итерации трассировка осуществляется со смещением на 1 см ближе текущей позиции, но не ближе параллельной соответствующей конечности, передней или задней, для обеспечения поступательного движения).

2) В качестве потенциального Impact Point (точки для перемещения конечности) учитывается трассировка точки пересечения с мешем ветки с соответствующей стороны лапы, например, первое пересечение справа, если мы пытаемся сместить правую конечность.

3) Осуществляем проверку условий:

- найденная точка существует,
- проверяем, что ветка та же самая (по каналу Branch), если не меняем ветку,
- проверка CheckLegsOrder не ставим задние лапы перед передними, в том числе текущую перемещаемую, например, переднюю лапу, не ставим позади второй передней лапы,
- дистанция удовлетворяет длине конечности, которая ранее задана переменной, она задается для каждой модели в переменной, в частности для текущей модели установлено значение 40,
- дополнительная проверка минимального шага, чтобы алгоритм не пытался сместить конечность назад.

Визуализация поиска Impact Point функции FindNextStep в текущей модели ленивца приведена на рисунке 25 ниже:

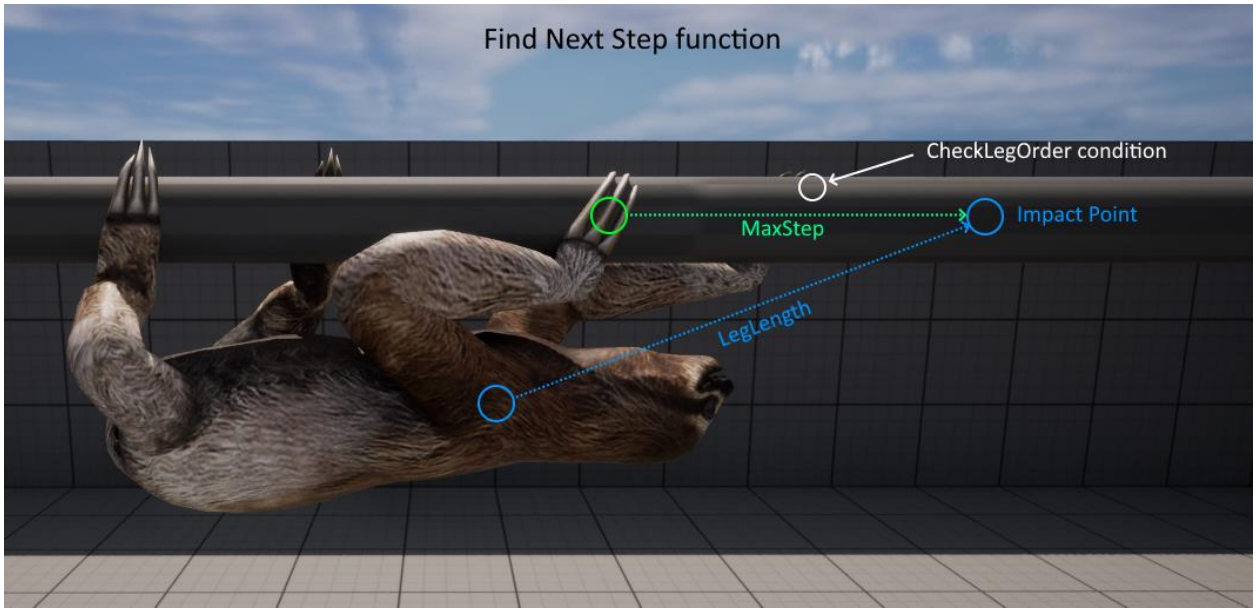

Рисунок 25 – Визуализация функции FindNextStep, возвращающей ImpactPoint

Для перехода на следующую ветку слева и справа от конечностей используем Line trace с дистанцией в области достижимости, получив, соответственно, right vector для трассировки для правых конечностей и умножив правый вектор на -1 для левых конечностей.

Чтобы реализовать смещение тела ленивца используется следующий итеративный подход:

1) Для нахождения изгиба тела при изгибании ветки, для получения параметра rotation, усредняем в blueprints модели rotation, исходя из положения конечностей с помощью встроенной функции Set Actor Location and Rotation.

2) Для поиска среднего вращения реализована функция, которая постоянно вызывается, возвращающая rotation и усредняющая положение тела относительно конечностей. Реализация данной функции показана в формуле (6):

$$AvgRotation = (DirRightBackLegToRightFwdLeg + DirLeftBackLegToLeftFwdLeg)/2, \quad (6)$$

где, DirRightBackLegToRightFwdLeg - направление от задней лапы с правой стороны тела к передней лапе с той же правой стороны тела, DirLeftBackLegToLeftFwdLeg - направление от задней лапы с левой стороны тела к передней лапе с той же левой стороны тела.

3) Полученное в результате усредненное направление конечностей, преобразуем в поворот модели в мировых координатах,

Для нахождения нового положения (New Location) тела ленивца используется аналогичный усредняющий подход:

– получаем положение конечного эффектора в данный момент времени,

– находим усреднённое значение позиции между четырьмя конечностями,

– получаем вектор по оси Z направленный вверх,

– делаем поправку полученного вектора на смещение тела по вертикали и вычитаем из полученной ранее усредненной позиции.

Формула поиска нового положения (7) выглядит следующим образом:

$$AverageLocation = (FootPosition1 + FootPosition2 + FootPosition3 + FootPosition4)/4 – ActorUpVector*BodyMaxVerticalOffset, \quad (7)$$

где FootPosition1, FootPosition2, FootPosition3, FootPosition4 – координаты соответствующих конечностей, ActorUpVector – вектор модели направленный вверх, переменная BodyMaxVerticalOffset – смещение тела ленивца относительно положений конечностей (поскольку найдено усреднённое положение лап, а тело должно быть смещено вниз относительно лап, т.е. не

находиться в одной плоскости с лапами), для выбранной модели данное значение составляет 7.

Перемещение по стволу, в том числе перестановка лап, если надо перейти на ветку, которая находится под иным углом к стволу - соответственно, переход со ствола на ветви и обратно, реализовано с помощью дополнительной логики: чтобы выделить акторов по которым будет осуществляться вертикальное движение, введем в соответствующее поле актора, выполняющего роль ствола, тэг «Trunk» и в дальнейшем, для применения алгоритма движения по стволу, проверяем условие при помощи функции blueprints Actor has tag.

Для цели определения вертикальности движения, по оси Y проверяем угол расположения актора ствола на сцене, и если наклон больше 70 градусов, то определяем это как вертикальное движение по стволу.

Затем, определяя направление в переменной direction 1 или -1 (вперед\вверх), и, определяя в функциях is Foots on Trunk и is Foots on Branch с помощью условий, находятся ли лапы еще на стволе (учитываем подходит ли ленивец к ветке передними лапами или задними, т.к. переход на ветку и с ветки ленивцем в природе осуществляется головой вперед\вверх, как это исследовалось в 1 главе настоящей работы).

Далее, с помощью box line trace с шириной в 2 раза больше ширины меша ленивца, чтобы учесть изгиб ствола и доступность его для перестановки конечности, реализуем функцию Transfer to Branch в компоненте конечности.

В данной функции Transfer to Branch реализован line trace в зависимости от того какую конечность перемещаем по аналогии как это реализовано для перемещения по ветке в функции FindNextStep, с учетом того, что если хотя бы одна из передних лап ещё на стволе, то задние не переносим на ветку.

Обратная функция Transfer to Trunk дополнительно учитывает условие, что спуск с ветки возможен только при условии, что голова животного находится под веткой, поскольку, согласно исследованию [9], этот тип спуска свойственен ленивцу, когда голова животного направлена вверх.

В качестве входных параметров функция Transfer to Trunk принимает ID текущей конечности животного и ссылку на текущий меш и возвращает координаты точки, на которую может быть переставлена конечность и true, в случае если данная точка существует.

Следует отметить, что при спуске с ветки на ствол и подъеме со ствола на ветку, для удобства контроля над головой ленивца добавлен сокет Neck_Socket, чтобы получать положения головы скелетного меша, и при смещении головы ленивца в сторону от ветки и чтобы не возникало коллизий с мешем, проверяем в тике положение головы и с помощью интерполяции, векторов вправо и вверх, плавно ее смещаем в сторону от ветки до момента перехода всех конечностей со ствола на ветку и обратно.

Для контроля же задней части ленивца при спуске на ветку добавлена функция Offset Back, использующая контроллер Sloth_Spine Control Rig, которая в timeline добавляет интерполированное смещение задней части ленивца, для визуальной достоверности параметр установлен на 0,15 м. в ходе тестирования, к основной функции усреднения положения теля при смещении конечностей. А при подъеме на ветку задние лапы ленивца аналогично в timeline для плавности поднимаем с помощью функции Offset back, однако с тем отличием, что в функции Transfer to Branch ограничиваем максимальную дистанцию line trace до 10 см. для улучшения визуальной составляющей, а затем поднимаем (с интерполяцией) одновременно обе задние конечности, как это ленивцы осуществляют в живой природе.

Для реализации вращения вокруг оси ствола, в тех случаях, когда ветка расположена под другим углом относительно положения модели ленивца на стволе, используем функции Find Side Step (для нахождения точки для перестановки конечности) и Rotate Body on Trunk.

Функция Find Side Step находит точку и с помощью line trace возвращает координаты для нового положения лапы с учетом направления вправо\влево.

А функция Rotate Body on Trunk, соответственно, смещает тело, по двум осям, поскольку движение боковое, игнорируя наклоны вперед\назад, что иначе приводило бы к неестественным наклонам.

Для поворота тела вокруг ствола, мы смещаем текущую лапу в зависимости от направления поворота, на величину переменной, которая зависит от физических характеристик, толщины ствола. А затем смещаем туловище на усредненное значение координат конечностей, по аналогии с нахождением усредненного положения тела.

Скорость ленивца регулируется в проекте в виде параметра StepSpeed, которая с учетом timeline длительностью в 1 секунду и задержки в переменной TimeBetweenSteps 0,2 секунды на сцене, а также максимальной величины шага в 27 сантиметров, позволяет управлять скоростью модели ленивца, и, если установить значение StepSpeed 0,5, то получим скорость передвижения ленивца в 0,11 м\с как это отмечалось в исследовании [7], где значения средней скорости ленивца составила в ходе экспериментов 0,11 ± 0,04 м/с.

Для удобства использования модели, используются соответствующие привязки клавиш ввода для перемещения объекта на сцене. Причем как с возможностью выбора возможности перехода на ветку, так и с возможностью автоматического передвижения с остановкой алгоритма при невозможности нахождения точки для перестановки очередной конечности.

4. ЭКСПЕРИМЕНТАЛЬНОЕ ТЕСТИРОВАНИЕ И ОЦЕНКА РАЗРАБОТКИ

В результате работы по реализации алгоритма процедурной анимации ленивца следует оценить визуально поведение получившейся процедурной анимации при различных сценариях его типичного поведения и взаимодействия с окружением на сцене, в частности, в цикле игрового движка Unreal Engine 5.

Поскольку средой обитания ленивца преимущественно являются деревья и ветви, то рассмотрим следующие сценарии, которые возникают при его перемещении на деревьях.

При тестировании для оценки успешности визуализации процедурной анимации модели ленивца принималась в расчет визуальная составляющая, когда контроллеры позволяли модели выглядеть естественно в различных условиях.

Сценарий 1. Перемещение ленивца на поворотах веток деревьев.

При реализации алгоритма применена инверсная кинематика конечностей ленивца. В ходе экспериментов производилось визуальное тестирования перемещения модели при поворотах веток, когда ветки находятся под различным углом к текущей. Диапазон постановки углов меше ветки относительно текущей осуществлялся в диапазоне от 95 до 175 градусов с шагом в 20 градусов для нахождения оптимального диапазона значений углов веток относительно текущей ветки.

Отметим, что сами границы диапазона предоставляют пограничные значения, когда 180 градусов является фактически продолжением текущей ветки, а при угле в 90 градусов алгоритм показывал значительное смещение передней конечности и показывал пересечения когтей с мешем ветки. до 175 градусов. На рисунке 26 видно, что поворот туловища и расположение конечностей при изгибе ветки осуществляется корректно и нужной стороны ветки:

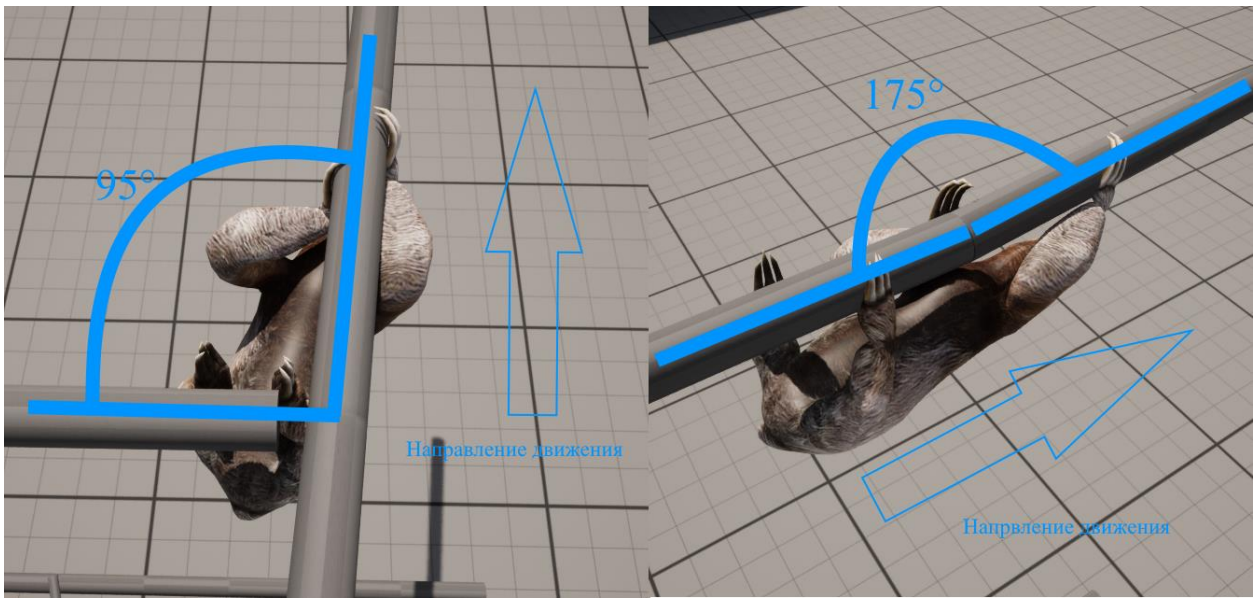

Рисунок 26 – Поворот ветки под углами 95 и 175 градусов (слева направо)

Успешная визуализация демонстрировалась в диапазоне углов от 95 градусов

Сценарий 2. Перемещение ленивца на параллельных ветках.

При движении по параллельным веткам поведение модели ленивца осуществляется без ошибок line trace, то есть либо движение осуществляется по текущей ветки, при необходимости смены ветки модель перемещает конечности в правильной очередности, если точка достижимости на новой ветке доступна.

В ходе экспериментов модель тестировалась на параллельных ветках при угле 0 градусов относительно друг друга.

На рисунке 27 показано начало движения по исходной ветке и постановка конечностей при смене ветки на параллельную (слева направо):

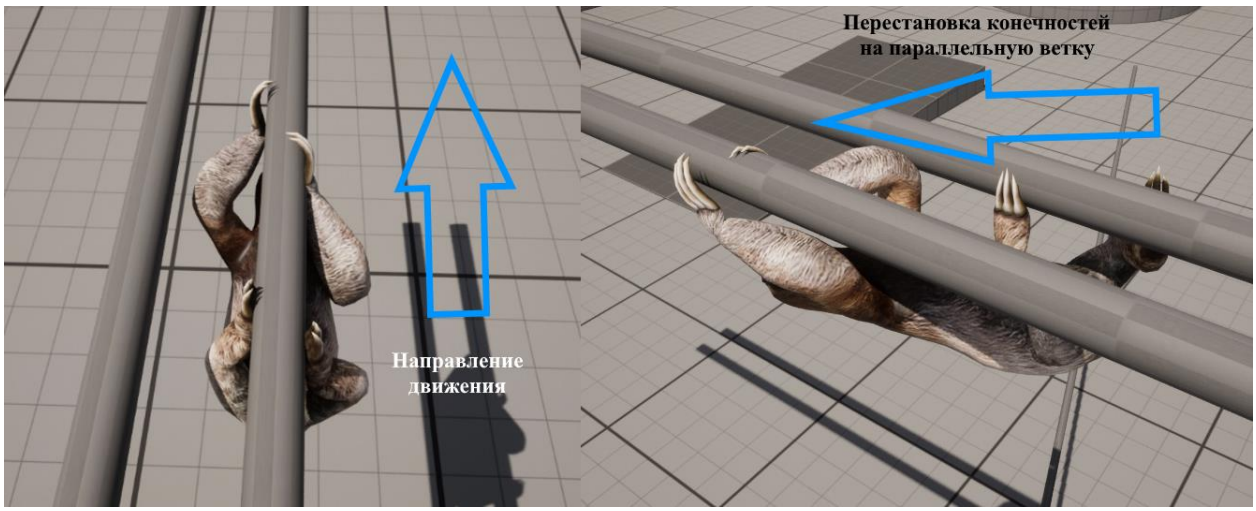

Рисунок 27 – Движение по параллельным ветвям

По результатам проведенных экспериментов по сценарию 2 получена успешная визуализация.

Сценарий 3. Перемещение ленивца по стволу дерева.

При перемещении по стволу реализована возможность движения вокруг оси ствола, с помощью бокового смещения лап, для тех случаев, когда переход может понадобиться для перехода на ветку, которая находится под другим углом относительно положения тела ленивца, как показано на рисунке 28 (слева на скриншоте модель находится под веткой, справа – под другим углом относительно той же ветки, за счет смещения положения тела):

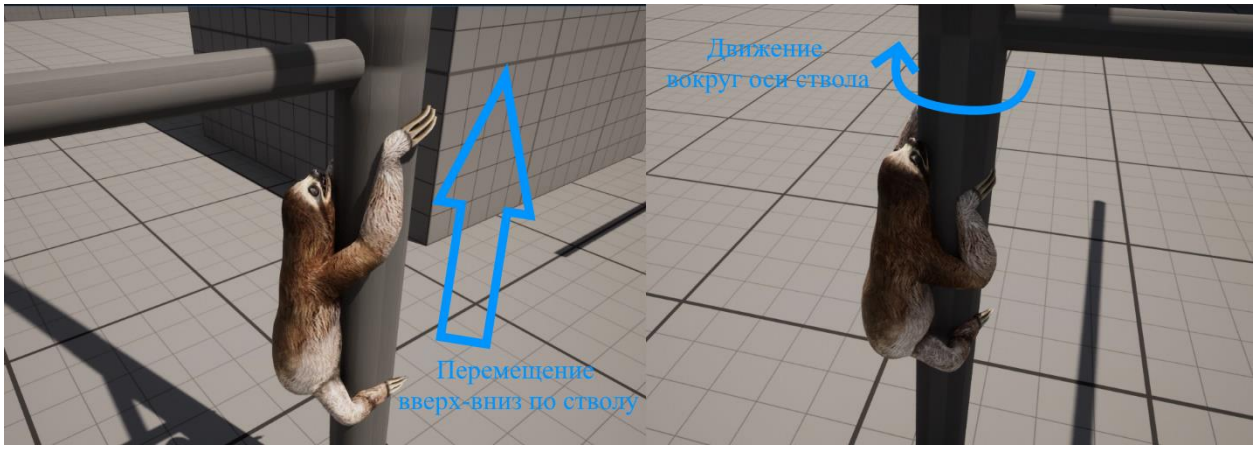

Рисунок 28 – Перемещение по стволу

По результатам тестирования можно отметить, что перемещение модели осуществлялось успешно во всех направлениях.

Сценарий 4. Перемещение ленивца на ветках, переходящих на другую высоту и на конце ветки.

При продолжении движения по ветке, которая находится под углом относительно исходной перестановка лап осуществляется с помощью инверсной кинематики и без сбоев и при достижении конечной точки перемещение останавливается, при этом остается возможность перемещения либо назад с сохранением очередности перестановки конечностей, либо перехода на другую ветку.

Тестирование осуществлялось на ветках, переходящих на другую высоту в диапазоне от 130 градусов до 200 градусов относительно текущего положения модели ленивца с шагом 5 градусов. Пограничным положениями явились углы в 130 градусов, по причине того, что при значениях менее 130 градусов тело модели пересекалось с веткой, что является неестественным, и 200 градусов в связи с тем, что для ленивцев не является типичным спуск вниз головой, как это изучалось авторами исследования [9], поэтому в алгоритме ограничен подобный вид движения.

Примеры при экспериментальном тестировании показан на рисунке 29:

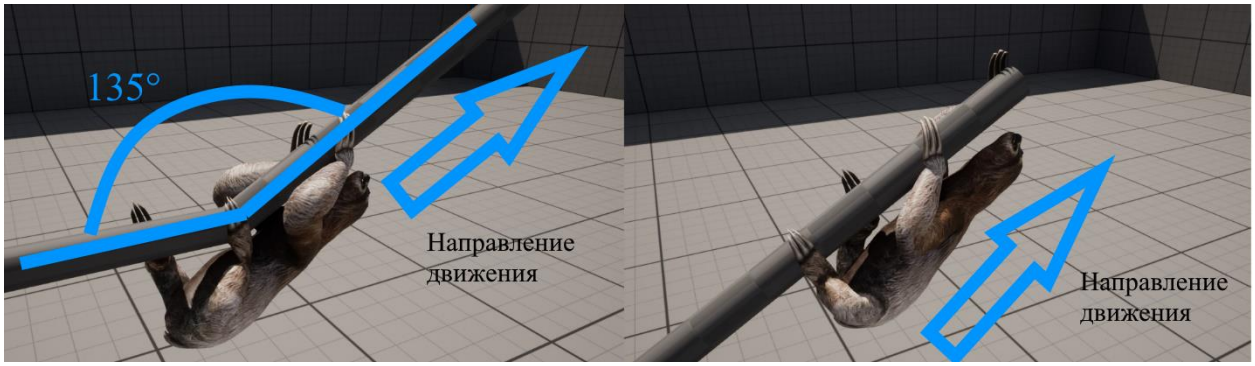

Рисунок 29 – Перемещение по ветке 135 градусов и достижение конца ветки (слева направо)

Сценарий 5. Переход ленивца с ветки на ствол деверева и обратно.

При спуске с ветки на ствол модели ленивца реализовано смещение головы в сторону от ветки, и подтягивание задних ног при подъеме на ветку для того, чтобы визуально правильно модель работала в этих сценариях.

В ходе экспериментов производилось визуальное тестирования перемещения модели по данному сценарию, когда ветки находятся под различным углом к стволу. Диапазон постановки углов меша ветки относительно вертикального ствола для проведения экспериментов находился в диапазоне от 90 до 140 градусов с шагом 10 градусов для нахождения оптимального диапазона значений углов веток относительно ствола.

На рисунке 30 приведены примеры поведения модели ленивца на сцене под разными углами в случаях, которые визуально могут восприниматься как реалистичное поведение:

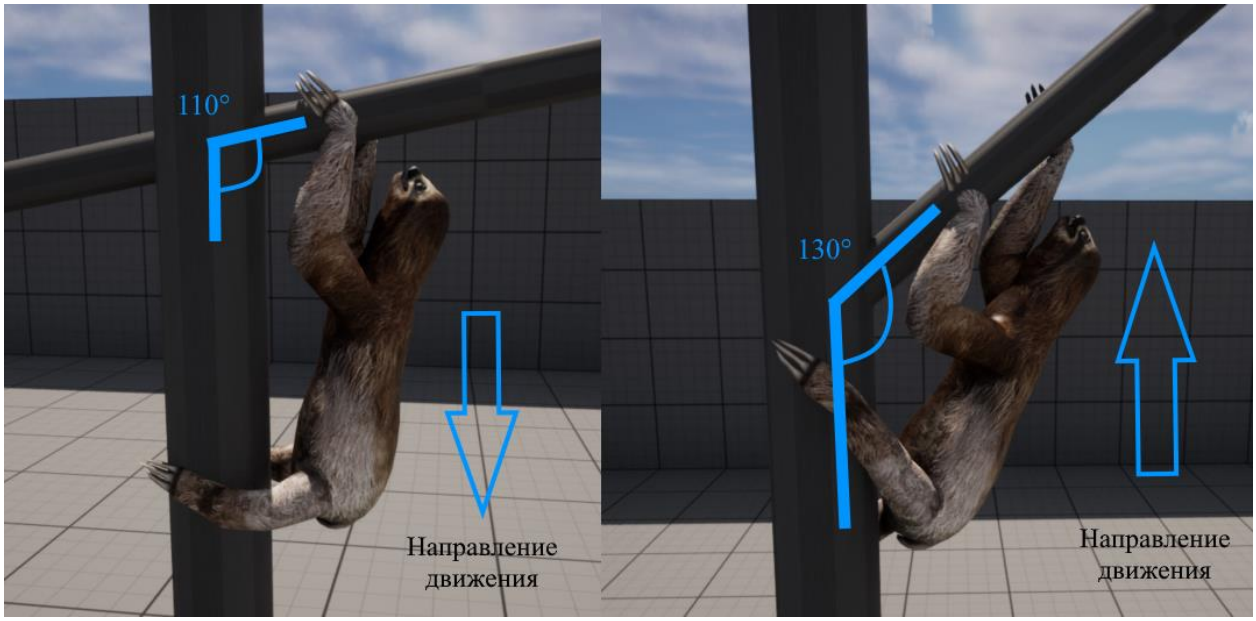

Рисунок 30 – Спуск с ветки на ствол и обратно под углами 110 и 130 градусов (слева направо)

Пограничные случаи показаны на рисунке 31:

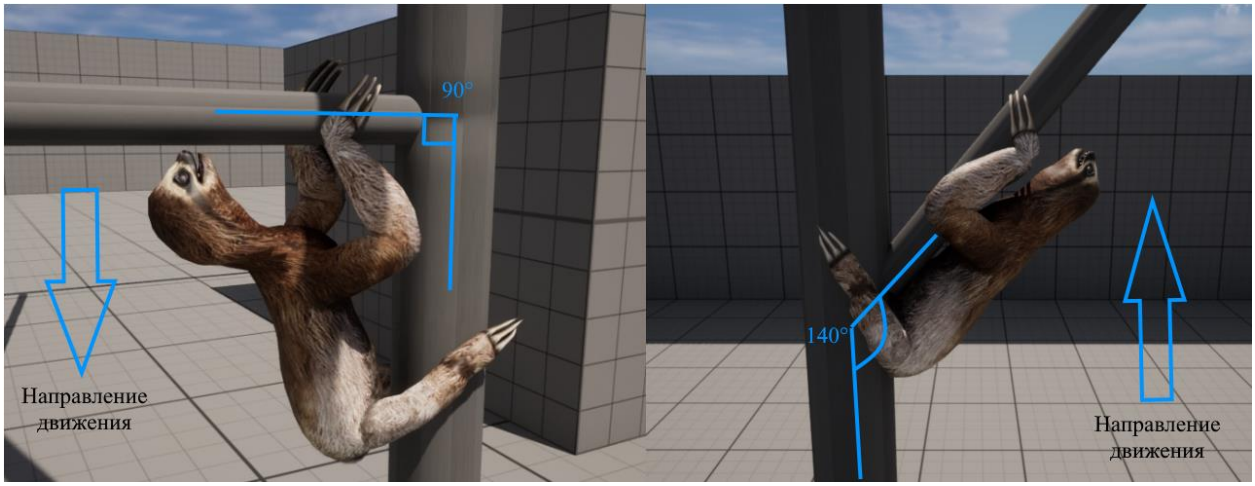

Рисунок 31 – Спуск с ветки на ствол и обратно под углами 90 и 140 градусов (слева направо)

По результатам проведенных экспериментов по сценарию 5 получены следующие данные: успешная визуализация от более 90 градусов относительно ствола до 140 градусов относительно ствола. В случаях за границей данного диапазона визуальное поведение модели воспринимается как пограничное, а не абсолютно успешное.

По результатам проведенного тестирования в различных сценариях обобщим полученные результаты в сводную таблицу:

Таблица 7 – Обобщенные результаты визуального тестирования модели

Сценарий	**Условия**	**Результат**
1	2	3
Перемещение ленивца на поворотах веток деревьев	Углы между ветками в диапазоне от 90 до 180 градусов относительно текущей ветки	Успешная визуализация в диапазоне более 95 и менее 175 градусов
Перемещение ленивца на параллельных ветках	Угол параллельных веток относительно друг друга 0 градусов	Успешная визуализация
Перемещение ленивца по стволу дерева.	Движение вокруг оси ствола и вверх-вниз	Успешная визуализация

Продолжение таблицы 7

1	2	3
Перемещение ленивца на ветках, переходящих на другую высоту и на конце ветки	Углы между ветками от 135 до 200 градусов относительно исходной ветки	Успешная визуализация в диапазоне более 135 и менее 200 градусов
Переход ленивца с ветки на ствол деверева и обратно	Углы веток в диапазоне от 90 до 140 градусов относительно ствола	Успешная визуализация в диапазоне более 90 и менее 140 градусов

Нагрузочная производительность тестировалась на персональном компьютере со следующими характеристиками:

– процессор: Intel Core i7-9750H CPU 2.60GHz,

– озу: 16GB,

– видеокарта: NVIDIA GeForce RTX 2070 with Max-Q Design 8GB.

Во всех сценариях нагрузочного тестирования частота кадров находилась в диапазоне 96-100 FPS, на рисунке 32 на скриншоте показан FPS во время тестирования:

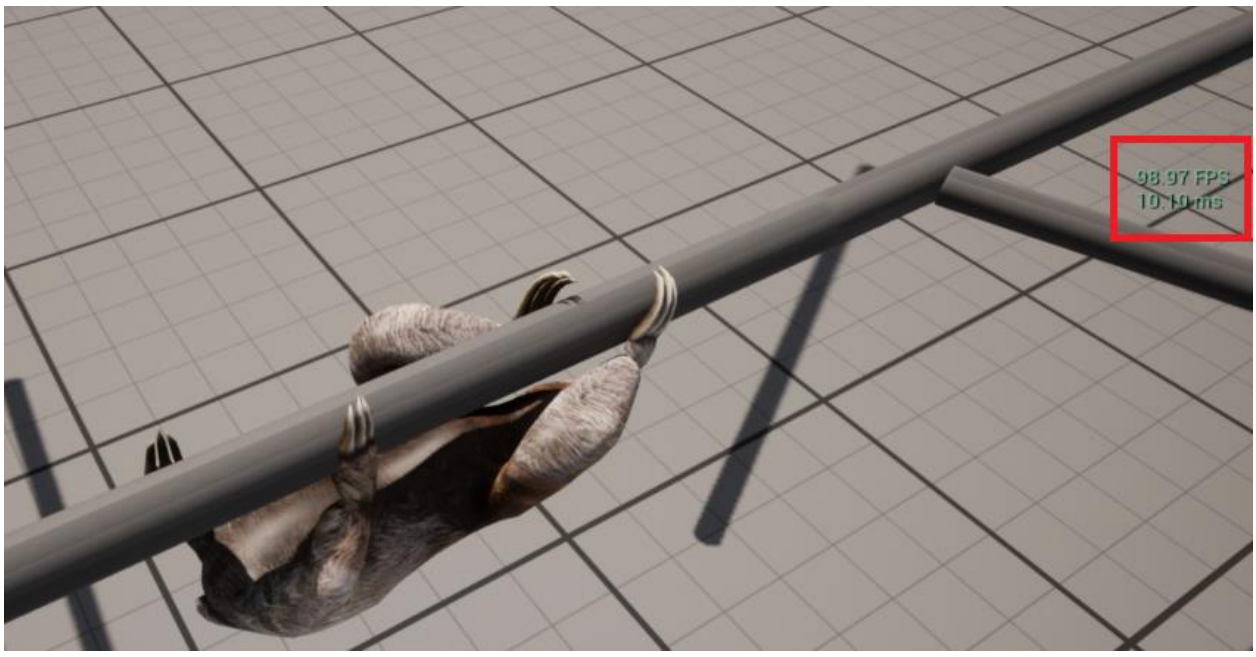

Рисунок 32 – FPS при работе процедурной анимации

В ходе оценки производительности тестировалась работа анимации с помощью системы профилирования Unreal Insights. По результатам установлено, что оценка производительности алгоритма во фрейме, включая вызовы из основной функции находилась в диапазоне от 12,4 до 14 миллисекунд (рисунки 33 и 25).

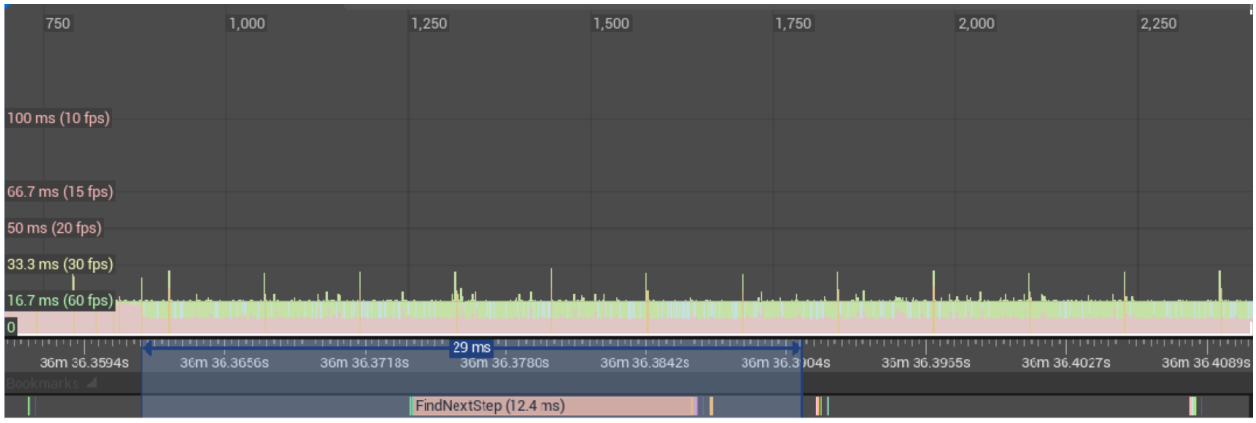

Рисунок 33 – Производительность алгоритма при работе процедурной анимации с одной моделью ленивца

А по результатам стресс-тестирования при нахождении на сцене одновременно 10 одновременно работающих моделей ленивца оценка производительности алгоритма находилась в диапазоне от 126,6 до 154.4 миллисекунд (рисунки 34 и 35).

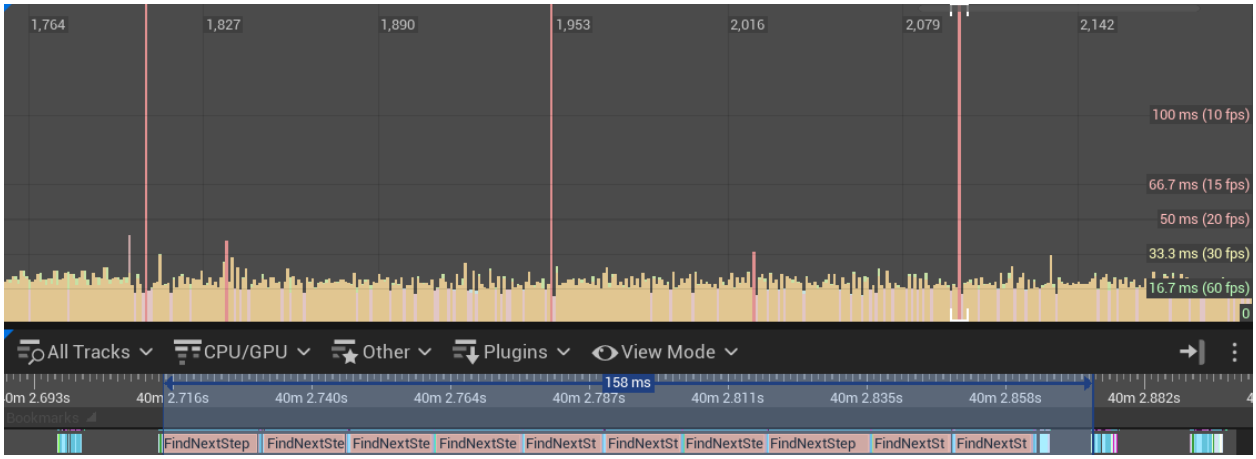

Рисунок 34 – Производительность алгоритма при работе процедурной анимации с 10 моделями ленивцев на сцене

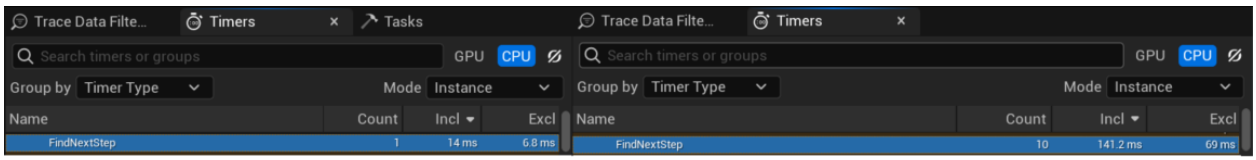

Рисунок 35 – Пример результатов оценки производительности при работе процедурной анимации с 1 моделью (слева) и 10 моделями (справа)

В целом по результатам проведенного экспериментального тестирования можно отметить, что реализованная модель ленивца работает плавно и визуально правильно и завяленный функционал выполняется. Таким образом, данную модель можно использовать на ветвях и стволах разной толщины в рамках установленных ограничений по достижимости конечностями мешей веток и стволов с сохранением работоспособного состояния.

ЗАКЛЮЧЕНИЕ

В рамках настоящей работы было изучено строение тела ленивца и особенности его локомоции в окружающей среде обитания. В ходе данного анализа было выявлено, что основная среда его обитания — это деревья, и для перемещения животное использует два основных типа перестановки конечностей: диагональный и боковой, при этом скорость перемещения является небольшой, в среднем около 0.11 м\с.

Далее, в ходе исследования процедурной анимации изучались преимущества и недостатки различных алгоритмов кинематики, как прямой, так и инверсной с целью подбора оптимального алгоритма, которым явился алгоритм инверсной кинематики cyclic coordinate descent. В ходе выбора среды разработки для имплементации были изучены плюсы и минусы различных программ и сделан выбор в пользу Blender для риггинга и Unreal Engine 5 для генерации анимации в режиме реального времени.

В ходе проектирования и разработки алгоритма процедурной анимации был предложены структурные элементы моделирующие поведения ленивца и требующие реализации: перемещение по ветвям в трехмерном пространстве, перемещение по стволу, переходы между веткой и стволом, с учетом особенностей его физиологии и локомоции на основе проведенного ранее исследования биологии ленивца.

При реализации модели ленивца применялась система Control Rig встроенная в Unreal Engine 5 для управления контроллерами конечностей и функционалом системой инверсной кинематики для управления костями модели с помощью blueprints. В дальнейшем при тестировании модели были выявлены допустимые диапазоны, позволяющие обеспечить визуально достоверное поведение модели в различных сценариях поведения в окружающем пространстве.

По результатам проведенной работы можно отметить, что реализованная процедурная анимация ленивца позволяет генерировать движения ленивца на дереве и ветвях. Таким образом, девелопер может,

изменяя параметры модели, управлять движением и генерируемой анимацией в режиме реального времени, при этом движения будут коррелировать с поведением животного в живой природе.

Исходя из вышеизложенного, можно отметить, что цель была достигнута, в дальнейшем возможно усложнение логики поведения ленивца и добавление новых сценариев и окружения, оптимизация производительности, реализация проекта на C++, как вариант, возможно реализовать разработанную систему в виде плагина для программ моделирования либо игрового движка.

Результаты настоящей работы, исходный код, инструкции к запуску опубликованы в репозитории на GitHub, получить доступ к ним можно по ссылке https://github.com/TimArVR/ITMO_ProceduralSloth.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Jozef Hladký Fire Simulation in 3D Computer Animation with Turbulence Dynamics including Fire Separation and Profile Modeling / Jozef Hladký, Roman Ďurikovič // International Journal of Networking and Computing. – 2018.- Volume 8, Number 2. – P.186-204.

2. Naoaki Kataoka Procedural animation of waving cartoon hair by emulating animator's techniques [Электронный ресурс] / Naoaki Kataoka, Tomokazu Ishikawa, Yusuke Kameda, Ichiro Matsuda, Susumu Itoh // International Core Journal of Engineering. - Volume 7 Issue 4. – 2021. – Режим доступа: https://doi.org/10.1117/12.2521535. - (Дата обращения: 24.04.2024).

3. Rodrigues Nelson Influence of the underwater environment in the procedural generation of marine alga Asparagopsis Armata [Электронный ресурс] / Rodrigues Nelson, Sousa António Augusto, Rodrigues Rui, Coelho António // WSCG 2022: full papers proceedings: 30. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision. – 2022. – P. 273-281. - Режим доступа: http://hdl.handle.net/11025/49605. - (Дата обращения: 24.04.2024).

4. Goswami P. A survey of modeling, rendering and animation of clouds in computer graphics [Электронный ресурс] / Goswami P. // Vis Comput 37, 1931–1948. - 2021. – Режим доступа: https://doi.org/10.1007/s00371-020-01953-y. - (Дата обращения: 24.04.2024).

5. John A.Nyakatura Limb kinematics during locomotion in the two-toed sloth (Choloepus didactylus, Xenarthra) and its implications for the evolution of the sloth locomotor apparatus / John A.Nyakatura, Alexander Petrovitch, Martin S.Fischer // Zoology. – 2010. - Volume 113, Issue 4. P. 221-234.

6. John A Nyakatura Three-dimensional kinematic analysis of the pectoral girdle during upside-down locomotion of two-toed sloths (Choloepus didactylus, Linné 1758) / John A Nyakatura, Martin S Fischer // Frontiers in Zoology. – 2010. - Volume 7.

7. M. C. Granatosky Forelimb and hind limb loading patterns during below branch quadrupedal locomotion in the two-toed sloth / M. C. Granatosky, D. Schmitt // Journal of Zoology. – 2017. – Volume 302, Issue 4. P. 271-278.

8. John A. Nyakatura A mechanical link model of two-toed sloths: no pendular mechanics during suspensory locomotion / John A. Nyakatura, Emanuel Andrada // Acta Theriol. – 2013. – Vol. 58. P. 83–93.

9. Frank C. Mendel Use of Hands and Feet of Three-Toed Sloths (Bradypus variegatus) during Climbing and Terrestrial Locomotion // Journal of Mammalogy.- 1985. – Vol. 66, No. 2. P. 359-366

10. Степанова Я.Д. Компьютерная анимация и мультипликация. Создание анимационного персонажа // Культура и технологии. - 2020. - Том 5. Вып. 4. - С. 215-216.

11. Rei Narita, Optical Flow Based Line Drawing Frame Interpolation Using Distance Transform to Support Inbetweenings [Электронный ресурс] / Rei Narita, Keigo Hirakawa, Kiyoharu Aizawa // 2019 IEEE International Conference on Image Processing (ICIP). – 2019. – Режим доступа: https://doi.org/10.1109/ICIP.2019.8803506. - (Дата обращения: 24.04.2024).

12. Shubham Sharma, Use of Motion Capture in 3D Animation: Motion Capture Systems, Challenges, and Recent Trends [Электронный ресурс] / Shubham Sharma, Shubhankar Verma, Mohit Kumar, Lavanya Sharma // 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). – 2019. – Режим доступа: https://ieeexplore.ieee.org/abstract/document/8862448. - (Дата обращения: 24.04.2024).

13. Шарикова Марина Владимировна, Технологии анимации [Электронный ресурс] // Постулат. - 2022. - №9. – Режим доступа: http://www.e-postulat.ru/index.php/Postulat/article/view/4423/4483. – (Дата обращения: 24.04.2024)

14. Sebastian Silva Neuranimation: Reactive Character Animations with Deep Neural Networks / Sebastian Silva, Sergio Sugahara, Willy Ugarte // In Proceedings

of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. – 2022. – P.252-259

15. Jesse D. Marshall Continuous Whole-Body 3D Kinematic Recordings across the Rodent Behavioral Repertoire / Diego E. Aldarondo, Timothy W. Dunn, William L. Wang, Gordon J. Berman, Bence P. O¨ lveczky // Neuron. – 2021. - Volume 109, Issue 3. – P. 420-437

16. Rohmer D. Velocity Skinning for Real-time Stylized Skeletal Animation / Rohmer D., Tarini M., Kalyanasundaram N., Moshfeghifar F. // Computer Graphics Forum. – 2021. - Volume 40, Issue 2. P.549-561

17. Zucconi A. An Introduction to Procedural Animations [Электронный ресурс] // 2017. – Режим доступа: http://www.alanzucconi.com/2017/04/17/procedural-animations/. – (Дата обращения: 24.04.2024).

18. Aristidou A. Inverse kinematics techniques in computer graphics: A survey / Aristidou, A., Lasenby, J., Chrysanthou, Y., and Shamir, A. // In Computer Graphics Forum. – 2018. - Vol. 37. P.35–58.

19. Cadevall Soto Lydia Procedural generation of animations with inverse kinematics [Электронный ресурс] // Universitat de Barcelona. – 2021. – Режим доступа: http://hdl.handle.net/2445/182237. – (Дата обращения: 24.04.2024).

20. Li Chun Tommy Wang A Combined Optimization Method for Solving the Inverse Kinematics Problem of Mechanical Manipulators / Li Chun Tommy Wang and Chih Cheng Chen // IEEE Transactions on Robotics and Automation. – 1991. - Vol. 7, 4. P. 489–499.

21. R. Muller-Cajar Triangualation: A new algorithm for inverse kinematics / R. Muller-Cajar, R. Mukundan // Proc. Image Vis. Comput. New Zealand. – 2007. - P.181–186.

22. Andreas Aristidou FABRIK: A fast, iterative solver for the Inverse Kinematics problem / Andreas Aristidou, Joan Lasenby // Graphical Models. – 2011. – Vol. 73, Issue 5. P. 243-260.

23.Tao S. Extending FABRIK with Obstacle Avoidance for Solving the Inverse Kinematics Problem / Tao S., Tao H., Yumeng Y., // Journal of Robotics. – 2021. – Vol. 2. P.1-10.

24. Santos M. C. FABRIK-R: An extension developed based on FABRIK for robotics manipulators / Santos M. C., Molina L., Carvalho E., Freire E. O. // IEEE Access PP. – 2021. – Vol 99. P. 1-1.

25. Richardsson Matilda Most efficient Inverse Kinematics algorithm for Quadruped models: Comparing FABRIK to CCD [Электронный ресурс] // Degree project in Computer Science with specialisation in Interactive Media technology Stockholm, Sweden. – 2022. – Режим доступа: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1642293&dswid=-7883. – (Дата обращения: 24.04.2024).

26. Phillipe Cardoso Santos M-FABRIK: A New Inverse Kinematics Approach to Mobile Manipulator Robots Based on FABRIK / Phillipe Cardoso Santos, Raimundo Carlos Silvério Freire, Elyson Adan Nunes Carvalho, Lucas Molina, Eduardo Oliveira Freire // IEEE Access. – 2020. - Vol. 8. P. 208836-208849.

27. Barczak Andrzej Marian, Woźniak Hubert Comparative study on game engines// Studia Informatica. System and information technology – 2020.- Vol. 23, No. 1-2. P.5-24.

28. Al Lawati, H. A. J. The Path of UNITY or the Path of UNREAL? A Comparative Study on Suitability for Game [Электронный ресурс] – 2020. – Режим доступа: https://doi.org/10.47611/jsr.vi.976. – (Дата обращения: 24.04.2024).

29. Ruan Lotter Taking Blender to the Next Level. – Packt Publishing, 2022.- P.520.

30. Назарова Е.Н. Конференция Современные технологии: проблемы инновационного развития и внедрения результатов “Сравнительная характеристика программы Blender и другого редактора трехмерной графики – Maya // Сборник статей XII Международной научно-практической

конференции. - Петрозаводск: Международный центр научного партнерства «Новая Наука» (ИП Ивановская И.И.), 2022

31. Абдушукуров, Ф. А. Сравнение программ трёхмерной графики 3ds Max и Blender / Ф. А. у. Абдушукуров, И. Н. Голицына // Интернаука. – 2020. – № 19-1(148). – С. 23-26.

32. Andrew J. McKamy Pump the brakes! The hindlimbs of three-toed sloths decelerate and support suspensory locomotion Icon for The Forest of Biologists/ Andrew J. McKamy, Melody W. Young, Angela M. Mossor, Jesse W. Young, Judy A. Avey-Arroyo, Michael C. Granatosky, Michael T. Butcher//Journal of Experimental Biology. – 2023.- Volume 226, Issue 8