МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Сравнение алгоритмов создания костной анимации в ООО "ДЖАСТМОБИ"

Обучающийся И.Ю. Морщинин
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., доцент, А.Б. Кузьмичев
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент О.Н. Брега
(ученая степень, звание, И.О. Фамилия)

Тольятти 2023

Аннотация

Тема данной выпускной квалификационной работы: «Сравнение алгоритмов создания костной анимации в ООО "ДЖАСТМОБИ"».

Бакалаврская работа содержит пояснительную записку объемом 46 страница, 11 рисунков, 15 формул, 3 таблицы, 1 блок–схема и список используемой литературы, состоящей из 25 источников.

Объект исследования – костная анимация.

Предмет исследования – алгоритмы костной анимации, реализующие анимации.

Целью работы является сравнение разработанных алгоритмов костной анимации.

Область применения – область мобильных и компьютерных игр, сфера кино, в которую входят мультфильмы, фильмы, видео.

Значимость данной работы – выявление лучшего алгоритма для создания костной анимации.

Abstract

The topic of this final qualifying work: "Comparison of algorithms for creating bone animation in JUSTMOBI LLC".

The bachelor's thesis contains an explanatory note of 46 pages, 11 figures, 15 formulas, 3 tables, 1 flowchart and a list of used literature of 25 sources.

The object of the study is bone animation.

The subject of the study is bone animation algorithms that implement animations.

The purpose of the work is to compare the developed algorithms of bone animation.

The scope of application is the field of mobile and computer games, the field of cinema, which includes cartoons, films, videos.

The significance of this work is to identify the best algorithm for creating bone animation.

Оглавление

Введение

В наши дни сложно представить современную жизнь без анимации, ведь она стала неотъемлемой частью нашей жизни. Анимация окружает нас повсюду – на рекламных баннерах, при использовании смартфона, в мобильных и компьютерных играх, в кино и мультфильмах.

Актуальность данной темы обусловлена тем, что костная анимация очень активно используется в таком популярном на сегодняшний день направлении, как игровая индустрия, которая на данный момент получила огромную популярность среди людей всего мира. Благодаря развитию технологий, удается совершенствовать качество анимации, что очень положительно влияет на конечное качество продукта, а также не менее высокие оценки пользователей. К тому же на данный момент существует множество алгоритмов для создания костной анимации, однако у каждого из них есть свои достоинства и недостатки, поэтому актуальность данной темы сложно переоценить.

В данной выпускной квалификационной работе будут рассмотрены алгоритмы, используемые в создании костной анимации. Также будет произведено математическое описание каждого из них, после чего будет произведено сравнение между ними для выявления удовлетворяющих критериям компании "ДЖАСТМОБИ", после чего с помощью выбранных алгоритмов будет реализована программа для костной анимации. И в заключение будет проведено сравнение для выявления самого производительного алгоритма костной анимации.

Объект исследования – костная анимация.

Предмет исследования – алгоритмы, реализующие анимации.

Целью работы является сравнение разработанных алгоритмов костной анимации.

Задачи данного исследования:

– рассмотреть теоритические сведения по исследуемому объекту,

– исследовать существующие алгоритмы реализации костной анимации,

– провести сравнение выбранных алгоритмов,

– реализовать программу, с использованием выбранных алгоритмов.

В первом разделе данной выпускной работы будут рассмотрены самые распространенные алгоритмы для создания костной анимации, затем выбор нескольких из них на основе критериев компании "ДЖАСТМОБИ" для дальнейшей разработки, а также постановки задач на разработку. Во втором разделе будет подробно описана математическая составляющая выбранных алгоритмов, а также выбор языка программирования, среды разработки, операционной системы, а также средств для реализации программы. В заключительном разделе будет произведена реализация программного обеспечения на основе выбранных алгоритмов, тестирование программы, а также сравнение алгоритмов.

Глава 1 Анализ алгоритмов костной анимации

1.1 Обзор алгоритмов костной анимации

Компьютерная анимация является видом мультипликации, создаваемым с помощью компьютерных технологий. В наше время компьютерная анимация нашла применение не только в области развлечений, но также научной, деловой и производственной сферах.

Юрий Норштейн, режиссер таких мультипликационных фильмов, как «Цапля и журавль», «Ежик в тумане» и многих других, говорил: «мультипликация – это тайны сознания и чувства, помещенные на пленку. Чувства отражаются от материальной поверхности, превращая ее в фантазию. Это обязательное условие любого творчества. В мультипликации же – основное».[8]

Впервые алгоритм с использованием костей в анимации был предложен Э. Катмулом в 1972 году. Описывая проблемы автоматизированной анимации в более поздней статье, Э. Катмул упоминает также об алгоритме двухмерной костной анимации Н. Буртника и М. Вейна, предложенного в 1976 году.

Кроме того, для создания качественной анимации существуют принципы, которые были предложены известной студией The Walt Disney Company в 1930 году, дополняя их с годами. Их знание, а самое главное умение применять на практике может показать квалифицированность человека в данной области. Называются они 12 принципов анимации, и состоят из 12 пунктов:

– сжатие и разжатие: принцип заключается в изменении формы объекта при движении. Он добавляет ощущение жизненности и гибкости объекта;

– подготовка к действию: перед началом действия, объект должен сначала подготовиться к этому действию. Например, анимация бега

начинается с подготовки к бегу, когда тело наклоняется вперед;

– инсценировка: принцип заключается в том, чтобы создать хорошую композицию кадра, чтобы зритель мог легко понять, что происходит на экране;

– прямое действие и постановка по позам: это два различных подхода к созданию анимации. Прямое действие – это анимация, которая создается постепенно, кадр за кадром. Постановка по позам – это анимация, которая создается путем создания ключевых поз и затем заполнения промежутков между ними;

– инерция и наложение движений: этот принцип заключается в том, чтобы добавлять в анимацию дополнительное движение, которое происходит после того, как основное движение уже завершено. Например, при остановке автомобиля находящиеся в нем люди могут сдвинуться вперед;

– замедление в начале и конце: движение должно начинаться медленно, затем ускоряться и замедляться перед тем, как закончиться. Этот принцип помогает придать естественность движению объекта;

– движение по дуге: объекты, движущиеся по прямой линии, кажутся более искусственными. Движение по дуге придает естественность и гармонию движению объекта;

– второстепенные действия: этот принцип используется для усиления и дополнения основного действия в сцене. Иногда, второстепенные действия наоборот, уводят внимание зрителя от основного действия и концентрируют его внимание на деталях;

– частота кадров: частота кадров говорит о том, за сколько кадров будет совершено действие. Данный принцип определяет скорость движения и его продолжительность;

– гиперболизация: это преувеличение некоторых движений персонажа для их более наглядной выразительности. Гиперболизация добавляет жизни и динамики в анимацию;

– прорисовка: принцип подразумевает понятные формы, правильно распределенный вес и центр тяжести объекта. Позы объекта должны передавать состояния, намерения и желания персонажа;

– привлекательность: каждый персонаж должен обладать привлекательностью. Привлекательным персонажам лучше сочувствуют, сопереживают и понимают.

Существует два основных метода создания анимации с использованием костей:

– без использования весовых коэффициентов,

– с использованием весовых коэффициентов.

Алгоритм без использования весовых коэффициентов.

Алгоритм без использования весовых коэффициентов подробно описан в [25]. Трансформированное положение отдельной вершины рассчитывается по формуле:

$$\vec{v}'=B_i^{-1}\vec{v}, \quad (1)$$

где $\vec{v}'$ – итоговое положение вершины,

W – текущая матрица данного сустава, соответствующая трансформации позы скелета,

B_i^{-1} – матрица сустава, трансформации которого наследует рассматриваемая вершина, в позе привязки,

$\vec{v}$ – базовое положение вершины.

Сначала осуществляется перевод координат вершины $\vec{v}$ из глобальной системы координат в локальную систему координат, которая связана с рассматриваемым суставом. При перемещении самого сустава положение вершины в системе координат сустава $\overrightarrow{v_B} = B_i^{-1}\vec{v}$ остается неизменным. После чего производится расчет трансформированного положения сустава – вычисляется матрица W. После чего осуществляется обратный переход в глобальную систему координат: вершина наследует трансформации сустава,

и итоговое положение вершины вычисляется как $\vec{v}'=W\overrightarrow{v_B}$.

При такой привязке не избежать заломов, растяжений и самопересечений у модели, поэтому данный алгоритм используется в узких направлениях и его невозможно применять в каждом случае для получения качественной анимации.[17]

1. Алгоритм с использованием весовых коэффициентов Subspace–Skeletal Deformation(SSD).

Несмотря на то, что алгоритм с использованием весовых коэффициентов начал широко применяться намного раньше, основные принципы в литературе были изложены только в конце 1990–х годов [18], хотя первые концепты данного алгоритма были опубликованы еще в 1998 году.

Суть алгоритма заключается в том, что каждой кости задается степень влияния на конкретные вершины, и чем выше это влияние, тем сильнее эти вершины будут смещаться под влиянием данной кости. Трансформированное положение отдельной вершины $\vec{v}'$ рассчитывается по формуле (1):

$$\vec{v}'=\sum_i w_i W_i B_i^{-1}\vec{v},\ \sum_i w_i = 1, \quad (2)$$

где w_i – вес i–ой кости,

W_i – текущая матрица данного сустава,

B_i^{-1} – матрица сустава, трансформации которого наследует данная вершина,

$\vec{v}$ – базовое положение вершины.

Данный алгоритм используется для анимации и деформации трехмерных моделей, основанных на скелетных структурах. Вот некоторые области применения алгоритма SSD:

– мобильные и компьютерные игры: алгоритм SSD является популярным методом в игровой индустрии для анимации персонажей;

– анимация фильмов: он применяется в производстве компьютерной

графики для создания анимированных фильмов;

– робототехника: в робототехнике алгоритм SSD может быть использован для управления движением роботов, он обеспечивает гибкость и точное контролируемое движение суставов роботов;

– виртуальная реальность и дополненная реальность: алгоритм SSD может быть использован в средах виртуальной и дополненной реальности для анимации виртуальных персонажей, объектов или роботов.

Достоинства:

– переносимость анимации на другие модели с аналогичной топологией скелета,

– простота расчетов и высокая вычислительная эффективность,

– простота аппаратной реализации алгоритма.

Недостатки:

– необходимость продумывания и создания иерархии скелета для модели,

– ограничения на сложность деформаций,

– трудоемкость корректировки весов.

2. Алгоритм Pose Space Deformation (PSD).

Вершина из исходного положения $\vec{v}_0$ перемещается с помощью алгоритма весовых коэффициентов в положение $\vec{v}_i,\ i = 0,1,\dots,N$. Далее полученная форма компенсируется сдвигами $\vec{d}_i,\ i = 0,1,\dots,N$, которые соответствуют эталонным формам модели $x_i,\ i = 0,1,\dots,N$. Таким образом, вершина $\vec{v}$ в итоге перемещается в положение $\vec{v}_i' = \vec{v}_i + \vec{d}_i = LBS_i(\vec{v}_0) + \vec{d}_i$.[19]

Основная идея алгоритма заключается в том, что при деформации модели она вычисляется как линейная комбинация форм из набора «pose space». Каждая форма в наборе представляет собой некоторое положение

вершин модели в ее исходной форме. При анимации, когда необходимо изменить форму модели, вычисляются линейные коэффициенты для каждой формы из набора, и затем эти коэффициенты используются для генерации новой формы модели.[22]

Алгоритм костной анимации PSD, также как и алгоритм SSD применяется в различных областях для анимации и деформации трехмерных моделей, основанных на скелетных структурах. Вот некоторые области применения алгоритма PSD:

– мобильные и компьютерные игры: алгоритм костной анимации PSD широко применяется в игровой индустрии для анимации персонажей и объектов;

– анимация фильмов: алгоритм PSD используется в производстве компьютерной графики для создания анимации в фильмах. Например, данный алгоритм был применен при моделировании главного персонажа полнометражного анимационного фильма "Вольт" 2008 года выпуска, и заняло это у аниматоров примерно 3 недели;

– медицинская визуализация: алгоритм PSD может применяться в области медицинской визуализации для анимации и деформации трехмерных моделей частей тела или органов;

– виртуальная реальность и дополненная реальность: алгоритм PSD может быть применен в виртуальной и дополненной реальности для анимации виртуальных объектов и персонажей.

Достоинства:

– было разработано большое количество вспомогательных инструментов,

– в 2006 году была реализована аппаратная визуализация на основе данного алгоритма.

Недостатки:

– деформации модели в промежуточных кадрах не отвечают в полной мере соответствующим движениям реального человека,

– большие трудозатраты.

3. Алгоритм Animation Space.

В работе Мерри Б. [21] производится подстановка $p_i = w_i B_i^{-1} \vec{v}$ в уравнении (1): $\vec{v}' = \sum_i W_i p_i$. Так, количество весов, связанных с одной вершиной возрастает до четырех, что соответствует четырем компонентам вектора $p_i = (x, y, z, 0)$. Это позволяет независимым образом распределять влияние костей на различные координаты вершин, что, в итоге, позволяет сгладить артефакты алгоритма весовых коэффициентов. Кроме того, уравнение Animation Space является линейным.

Несмотря на большое количество параметров, время, необходимое для вычисления нового положения вершины алгоритмом Animation Space, сравнимо со временем алгоритма весовых коэффициентов SSD.

За счет того что алгоритм костной анимации Animation Space создан на основе алгоритма SSD, область его применения совпадает с родителем:

– мобильные и компьютерные игры,

– анимация фильмов,

– робототехника,

– виртуальная реальность и дополненная реальность.

Достоинства:

– скорость вычислений,

– сглаживание заломов.

Недостатки:

– трудоемкость корректировки весов.

4. Алгоритм Multi–Weight Enveloping.

Алгоритм Multi–Weight Enveloping (MWE) [23] подразумевает подстановку матрицы преобразования $M_i = W_i B_i^{-1}$ в уравнении (2):

$\vec{v}'=\sum_i w_i M_i \vec{v}$. Далее весовые коэффициенты w_i вносятся в матрицу преобразования M_i, давая, таким образом, «весовую матрицу» и увеличивая количество весовых коэффициентов до двенадцати:

$$\vec{v}' = \sum_{i=1}^{b} \begin{pmatrix} w_{i,11}m_{i,11} & w_{i,12}m_{i,12} & w_{i,13}m_{i,13} & w_{i,14}m_{i,14} \\ w_{i,21}m_{i,21} & w_{i,22}m_{i,22} & w_{i,23}m_{i,23} & w_{i,24}m_{i,24} \\ w_{i,31}m_{i,31} & w_{i,32}m_{i,32} & w_{i,33}m_{i,33} & w_{i,34}m_{i,34} \\ 0 & 0 & 0 & 1/b \end{pmatrix} \vec{v} \qquad (3)$$

Дополнительные веса позволяют рассчитывать влияние на каждый компонент вершины независимым образом по каждому из компонентов движения кости. Это способствует повышению гибкости алгоритма, и как результат сократить заломы как в алгоритме весовых коэффициентов.[20]

Алгоритм MWE имеет несколько преимуществ по сравнению с другими методами анимации, так как он позволяет более точно управлять формой объекта на каждом кадре анимации, учитывая его геометрические особенности и скелетную структуру. Это особенно полезно для объектов с большим количеством деталей и суставов, таких как персонажи в компьютерных играх.

Однако, алгоритм MWE также является более ресурсоемким и сложным в реализации, требуя большого количества вычислительной мощности для расчета анимации. Кроме того, настройка весов вершин может потребовать дополнительного времени и усилий для достижения желаемых результатов.

Алгоритм костной анимации Multi–Weight Enveloping также как и алгоритм Animation Space создан на основе алгоритма SSD, поэтому область его применения совпадает с алгоритмами SSD и AS:

- мобильные и компьютерные игры,
- анимация фильмов,
- робототехника,
- виртуальная реальность и дополненная реальность.

Достоинства:

– гибкость,

– сглаживание заломов.

Недостатки:

– трудность настройки весового коэффициента по причине того, что их двенадцать вместо одного,

– время вычислений.

Таким образом, были кратко описаны алгоритмы костной анимации Subspace–Skeletal Deformation(SSD), Pose Space Deformation (PSD), Animation Space(AS) и Multi–Weight Enveloping (MWE), указали область применения каждого из них, а также их достоинства и недостатки.

1.2 Выбор области применения алгоритмов костной анимации на примере ООО "ДЖАСТМОБИ"

Компания "ДЖАСТМОБИ" была зарегистрирована в 2018 году, и находится в центральном районе города Тольятти. На данный момент количество сотрудников насчитывает 80 человек работающих как в самом офисе, так и удаленно по всей стране. Сайт компании "https://justmoby.com/". Компания активно участвует в различных выставках игровых продуктов для презентации собственных проектов и идей, а также для обмена опытом с другими представителями компаний в сфере мобильных приложений.

Компания занимается разработкой мобильных игр, а также приложений для операционных систем Android и IOS. Самым популярным проектом на данный момент является мобильная игра "Raft Survival: Ocean Nomad", количество скачиваний которого уже превышает 100 миллионов по всему миру. Также имеется популярная мобильная игра "Let’s Survive", набравшая уже более 10 миллионов скачиваний и только набирающая популярность

мобильная игра "Raft Survival: Desert Nomad", у которой в данный момент более 1 миллиона скачиваний.

В данной компании анимации используются во многих частях продукта: персонажи, окружение, в которое входят деревья, трава, облака, вода, и так далее, а также интерфейс. Созданием анимаций занимаются как сами разработчики, так и квалифицированные аниматоры. Разработчики занимаются простыми и не сложными анимациями, в то время как аниматоры создают сложные и качественные.

Областью применения алгоритмов костной анимации в компании "ДЖАСТМОБИ" являются персонажи. Это могут быть люди, животные или монстры. Для анимации каждого из персонажа используют алгоритм костной анимации исходя из того, какие движения требуются от данного персонажа, а также от сложности его формы и количества деталей. Также костную анимацию применяют в анимации интерфейса, если на нем, например, находится персонаж или несколько персонажей, а также с помощью костной анимации могут анимировать элементы окружения, для плавной и гибкой анимации.

Исходя из области применения алгоритмов костной анимации в компании "ДЖАСТМОБИ", в данной выпускной квалификационной работе рассмотрим применение костной анимации на примере персонажей, поскольку персонажи намного чаще являются объектом костной анимации относительно интерфейса.

В данном разделе была рассмотрена краткая информация о компании "ДЖАСТМОБИ", а также область применения костной анимации в компании, ей являются персонажи и интерфейс, в результате чего была выбрана костная анимация персонажей.

1.3 Выбор алгоритмов костной анимации для проведения

сравнения

Выбор алгоритмов будет основан на критериях компании "ДЖАСТМОБИ" к алгоритмам костной анимации, они выглядят следующим образом:

1. вычислительная способность: сюда относится скорость вычислений алгоритма, она сильно сказывается на оптимизации продукта.

2. трудоемкость: данный критерий включает в себя время реализации анимации с помощью конкретного алгоритма костной анимации.

3. качество сглаживаний: этот критерий подразумевает наименьшее количество артефактов у анимируемого объекта при сложных деформациях.

В сравнении алгоритмов по критериям компании "ДЖАСТМОБИ" примут участие рассмотренные алгоритмы костной анимации: Subspace–Skeletal Deformation(SSD), Pose Space Deformation (PSD), Animation Space(AS) и Multi–Weight Enveloping (MWE).

Ниже представлена сравнительная таблица 1, в которой указаны рассматриваемые алгоритмы костной анимации, критерии компании "ДЖАСТМОБИ", а также баллы от 1 до 5, где 1 – наименьшая оценка, а 5 – наивысшая.

Таблица 1 – Сравнение алгоритмов по критериям

Критерии\алгоритм	SSD	PSD	AS	MWE
Выч. способность	5	3	5	2
Трудоемкость	5	2	4	2
Качество сглаживаний	3	3	4	4
Итог	13	8	13	8

Исходя из итогов сложения баллов в сравнительной таблице 1, было

выбрано 2 алгоритма для разработки и дальнейшего сравнения, а именно: Animation Space(AS) и Subspace–Skeletal Deformation(SSD).

1.4 Постановка задач на разработку алгоритмов костной анимации

Для сравнения алгоритмов костной анимации необходимо выполнить следующие задачи:

– описать математически алгоритмы Animation Space(AS) и Subspace–Skeletal Deformation(SSD),

– реализовать каждый из алгоритмов программно,

– выполнить тестирование выполненной программы на работоспособность,

– выполнить сравнение на основе реализованных алгоритмов,

– подвести краткий итог выполненных работ и выбрать самый производительный алгоритм, основываясь на результатах сравнения.

Глава 2 Разработка алгоритмов костной анимации

2.1 Математическое описание алгоритма костной анимации Animation Space

Алгоритм Animation Space в анимации представляет собой линейную модель для представления и управления анимацией персонажей. Он основывается на математических преобразованиях объектов.

Список инструментов, которые используется совместно с алгоритмом Animation Space:

1. вектор анимации: основной элемент в алгоритме Animation Space – это вектор анимации. Он представляет собой вектор, который содержит информацию о трансляции, вращении и масштабировании объектов в пространстве. Этот вектор может быть использован для определения позы или движения персонажа;

2. линейная комбинация: алгоритм Animation Space может использовать линейную комбинацию векторов анимации для создания новых поз и движений. Это позволяет аниматорам комбинировать и смешивать различные анимационные действия, чтобы получить желаемый результат;

3. интерполяция: для плавного перехода между разными позами и движениями алгоритм Animation Space может использовать методы интерполяции, такие как линейная интерполяция или сплайны, чтобы заполнить промежуточные кадры между ключевыми кадрами анимации;

4. матрицы преобразования: для применения трансляции, вращения и масштабирования объектов, алгоритм Animation Space может использовать матрицы преобразования, такие как матрицы трансляции, матрицы вращения и матрицы масштабирования. Эти матрицы могут быть перемножены или комбинированы для применения нескольких преобразований одновременно.

Рассматривая уравнение (2), мы видим, что атрибуты w_i и $\vec{v}$

умножаются вместе. Алгоритм Animation Space основан на идее объединения их в единый атрибут.

При выводе некоторых свойств широко используется однородное пространство, из–за чего необходимо часто извлекать части 4–векторов и аффинных матриц размером 4×4, то есть тех, последняя строка которых равна 0 0 0 1.

Для лучшего понимания введем следующие обозначения:

– если $v = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$ тогда $\overline{v} \coloneqq \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ и $\underline{v} \coloneqq w$,

– если $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}$ тогда $\overline{A} \coloneqq \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$ и $\underline{A} \coloneqq \begin{pmatrix} a_{14} \\ a_{24} \\ a_{34} \end{pmatrix}$.

Для матриц мы ссылаемся на матрицы $\overline{A}$ и $\underline{A}$ как на линейный и трансляционный компоненты соответственно. Также отметим следующие тождества для аффинных матриц, которые возможно легко проверить путем подстановки:

– $\overline{Av} = \overline{A}\overline{v} + \underline{Av}$,

– $\underline{Av} = \underline{v}$,

– $\overline{AB} = \overline{A}\overline{B}$,

– $\underline{AB} = \underline{A} + \overline{A}\underline{B}$.

Как мы ранее выяснили, формула трансформированного положения отдельной вершины $\vec{v}'$, в алгоритме Animate Space, равна $\vec{v}'=\sum_i W_i\, p_i$, где $p_i = w_i B_i^{-1}\vec{v}$. Данная сумма может быть преобразована в виде матрично–векторного произведения:

$$v = (W_0 W_1 \dots W_{i-1}) \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{i-1} \end{pmatrix} = \wp. \quad (4)$$

Рисунок 1 – Системы координат и преобразования между ними

Вектор p можно рассматривать как координаты v в многомерном пространстве, которое называется «анимационным пространством». Это пространство является 4b–мерным. Левая матрица преобразуется из пространства анимации в пространство модели. Обозначим ее G и назовем «анимационной матрицей», а ее работу – «анимационной проекцией» (рисунок 1).

Для общего элемента пространства анимации находим, что $\underline{v} = \underline{p_0} + \cdots + \underline{p_{b-1}}$.

Мы ссылаемся на $\underline{v}$ как на вес p, а также обозначим его $\underline{p}$. Точно также, как обычно работают с однородными точками в гиперплоскости 4D $\underline{v} = 1$, обычно работают с точками в гиперплоскости пространства анимации $\underline{p} = 1$. В контексте уравнения $\vec{v}' = \sum_i W_i \, p_i$ это говорит о том, что веса, влияющие на вершину, равны 1.

Мы можем применить технику присоединения к оболочке с несколькими весами, пусть $u_i = w_i \vec{v}$, тогда:

$$\vec{v}' = \sum_{i=1}^{b} \begin{pmatrix} u_{i,11}m_{i,11} + u_{i,12}m_{i,12} + u_{i,13}m_{i,13} + u_{i,14}m_{i,14} \\ u_{i,21}m_{i,21} + u_{i,22}m_{i,22} + u_{i,23}m_{i,23} + u_{i,24}m_{i,24} \\ u_{i,31}m_{i,31} + u_{i,32}m_{i,32} + u_{i,33}m_{i,33} + u_{i,34}m_{i,34} \\ 1/b \end{pmatrix} \quad (5)$$

Также, для нахождения промежуточных кадров между ключевыми кадрами, используется интерполяция. Интерполяция – это метод создания плавного движения между двумя и более точками в анимации. Она используется для создания плавных переходов между кадрами анимации. Существуют различные методы интерполяции, в том числе линейная, квадратичная, кубическая и более сложные методы.

Линейная интерполяция – это наиболее простой и распространенный метод. Он использует простую линейную функцию для определения промежуточных значений между начальной и конечной точками. Формула линейной интерполяции выглядит так[7]:

$$result = p_0 + (p_1 - p_0) * t, \quad (6)$$

где $result$ – искомое положение,

p_0 – начальное положение,

p_1 – конечное положение,

t – значение, которое находится между «p_0» и «p_1», или равно одному из них и определяющее положение промежуточной точки.

Квадратичная интерполяция – это более сложный метод по сравнению с линейной интерполяцией, который использует квадратичную функцию для определения промежуточных значений. Он может быть полезен для создания более плавных движений. Формула квадратичной интерполяции:

$$result = p_0(1-t)^2 + 2p_1t(1-t) + p_2t^2, \quad (7)$$

где $result$ – искомое положение,

p_0 – начальное положение,

p_1 – промежуточное положение,

p_2 – конечное положение,

t – значение, которое находится между «p_0» и «p_1», или равно одному из них и определяющее положение промежуточной точки.

Кубическая интерполяция – это еще более сложный метод, который использует кубическую функцию для определения промежуточных значений. Он может быть полезен для создания еще более плавных движений. Формулы кубической интерполяции:

$$result = p_0(1-t)^3 + 3p_1t(1-t)^2 + 3p_2t^2(1-t) + p_3t^3, \quad (8)$$

где $result$ – искомое положение,

p_0 – начальное положение,

p_1 – промежуточное положение,

p_2 – промежуточное положение,

p_3 – конечное положение,

t – значение, которое находится между «p_0» и «p_1», или равно одному из них и определяющее положение промежуточной точки.

Одним из сложных методов интерполяции является бикубическая интерполяция. Бикубическая интерполяция – это метод интерполяции, который используется в анимации для создания плавных переходов между кадрами. Она основана на кубической интерполяции, но включает в себя дополнительные шаги для улучшения качества анимации.[4]

В бикубической интерполяции значения для каждой точки на новом кадре вычисляются на основе значений вокруг нее на предыдущем и следующем кадрах. Это позволяет создавать более плавное движение и более точную интерполяцию формы объектов.[12]

Формула для бикубической интерполяции представляет собой кубическую функцию, которая вычисляет значение точки на новом кадре на основе значений на предыдущем и следующем кадре вблизи этой точки. В этой формуле используются шестнадцать значений, которые представляют значения функции в шестнадцати точках вокруг целевой точки. Эти значения

вычисляются на основе значений на предыдущем и следующих кадрах.

Бикубическая интерполяция может быть более ресурсоемкой, чем другие методы интерполяции, такие как линейная или квадратичная интерполяции. Однако она может обеспечить более точную и плавную интерполяцию, что особенно важно для объектов со сложной формой и текстурой.[10]

Также, одним из сложных методов интерполяции являются сплайны.

Сплайны – это математические кривые, которые используются в анимации и компьютерной графике для плавного перехода между ключевыми кадрами. Они позволяют создавать более органичную и естественную анимацию, управляя траекторией движения объектов.[1]

Существует несколько видов сплайнов, но основные из них – это кубический сплайн и катмулл–ромов сплайн.

Кубический сплайн – это сплайн третьего порядка, который определяется через 4 точки (2 начальные и 2 конечные). Он плавно проходит через каждую из этих точек, а также имеет плавный переход между ними, что создает более органичное движение объекта.[2]

Катмулл–ромов сплайн – это еще более гладкий сплайн, который также определяется через 4 точки, но в отличие от кубического сплайна, он проходит через эти точки, используя кривые Безье. Это позволяет создавать более естественные и органичные траектории движения объектов.[3]

Оба вида сплайнов широко используются в анимации и компьютерной графике для создания плавных и органичных анимаций. Они могут быть заданы и рассчитаны в программном коде с помощью различных алгоритмов, например, алгоритма де Кастельжо для кубических сплайнов и алгоритма катмулл–рома для катмулл–ромов сплайнов.[6]

Еще одним инструментом в создании костной анимации являются «Кривые Безье». Кривые Безье – это математический инструмент для описания гладких кривых, который широко используется в компьютерной

графике, дизайне и анимации.

Кривые Безье определяются на основе контрольных точек и векторов, которые задают направление кривой в каждой точке. Кривая Безье порождается в результате построения линейной комбинации контрольных точек с использованием базисных функций Безье.[14]

Формула для вычисления точки на кривой Безье имеет следующий вид:

$$B(t) = \sum 0^n B_i, n(t) * P_i, \quad (9)$$

где $B_i, n(t)$ – базисная функция Безье порядка n,

P_i – контрольная точка,

t – параметр, который меняется от 0 до 1.

Базисные функции Безье определяются рекурсивно и являются полиномами степени n:

$$B_i, n(t) = C(n, i) * (1 - t)^{n-i} t^i, \quad (10)$$

где $C(n, i)$ – биномиальный коэффициент, равный количеству способов выбрать i элементов из n возможных.

При $n = 0$ базисная функция $B_i, 0(t)$ равна 1 для $t = 0$ и 0 для всех остальных значения параметра t. При $n > 0$ базисная функция $B_i, n(t)$ является комбинацией базисных функций порядка $n - 1$:

$$B_i, n(t) = (1 - t) * B_i, n - 1(t) + t * B_{i-1}, n - 1(t) \quad (11)$$

Таким образом, кривая Безье первого порядка $(n = 1)$ определяется двумя контрольными точками p_0 и p_1, а формула для точки на кривой имеет следующий вид:

$$B(t) = (1 - t) * p_0 + t * p_1 \quad (12)$$

Кривая Безье второго порядка $(n = 2)$ определяется тремя контрольными точками p_0, p_1 и p_2, а формула для точки на кривой имеет следующий вид:

$$B(t) = (1 - t)^2 * p_0 + 2 * (1 - t) * t * p_1 + t^2 * p_2 \quad (13)$$

Кривая Безье третьего порядка $(n = 3)$ определяется четырьмя

контрольными точками p_0, p_1, p_2 и p_3, а формула для точки на кривой имеет следующий вид:

$$B(t) = (1-t)^3 * p_0 + 3 * (1-t)^2 * t * p_1 + 3 * (1-t) * t^2 * p_2 + t^3 * p_3 (14)$$

Чем выше порядок кривой Безье, тем больше контрольных точек требуется для ее определения, но и тем более гладкой она будет выглядеть. Кривые Безье также могут быть использованы для создания кривых поверхностей, например, в 3D моделировании.

Также в костной анимации существуют кинематические ограничения. Они используются для ограничения движения объектов в соответствии с определенными правилами. Они позволяют контролировать поведение объектов в зависимости от различных условий, таких как ограничение на силу, направление движения, ограничения на скорость и ускорение, ограничение на угол поворота и так далее.

Одними из самых распространенных видов кинематических ограничений являются «инверсная кинематика» и «прямая кинематика».

Инверсная кинематика – это техника, которая позволяет управлять движением кинематической цепи путем установки конечной позиции и/или ориентации кинематического эффектора, а затем вычисления соответствующих угловых значений суставов цепи, чтобы достичь заданной целевой позиции.

Для реализации инверсной кинематики в анимации используются алгоритмы, которые вычисляют углы суставов на основе заданной конечной позиции/ориентации кинематического эффектора. Существует несколько подходов к решению этой задачи, включая метод Якоби, методы Лагранжа и Ньютона, а также генетические алгоритмы.

Прямая кинематика – это метод решение кинематических задач, при котором путем последовательного применения матриц преобразования координат определяются положения и ориентации всех элементов в цепочке кинематических соединений.

Таким образом, если имеется цепочка из нескольких соединенных между собой кинематических элементов, то прямая кинематика позволяет определить положение конечного элемента или эффектора в пространстве в зависимости от положения базовых элементов и углов поворота соединений.

Для решения задач прямой кинематики в анимации часто используются матрицы трансформации, описывающие преобразования координат между различными элементами цепочки. Для применения матриц трансформации к векторам и точкам в пространстве используются различные операции, например, умножение матрицы на вектор или точку.

Таким образом, было подробно рассмотрено математическое описание алгоритма костной анимации Animation Space, а также используемые на практике инструменты для реализации качественной костной анимации.

2.2 Математическое описание алгоритма костной анимации Subspace–Skeletal Deformation

Алгоритм Subspace–Skeletal Deformation в анимации является методом костной деформации, который использует подпространства для представления и управления анимации персонажей.

Список инструментов, которые используется совместно с алгоритмом Subspace–Skeletal Deformation:

1. подпространства деформации: алгоритм Subspace–Skeletal Deformation предполагает наличие набора подпространств деформации, которые представляют собой некоторые локальные вариации формы искажений модели персонажа. Каждое подпространство деформации описывается набором базовых функций, таких как радиальные базисные функции или сплайны;

2. матрица весов: для каждой вершины модели персонажа вводится матрица весов, которая определяет, какие подпространства деформации

влияют на данную вершину. Эти веса могут быть заданы вручную или вычислены автоматически с использованием методов, таких как алгоритм взвешенных ближайших соседей или радиальные базисные функции;

3. деформация вершин: для каждой вершины модели персонажа происходит деформация, комбинируя влияние подпространств деформации с их соответствующими весами. Это достигается вычислением линейной комбинации базовых функций подпространств деформации, умноженных на соответствующие веса;

4. анимация: для создания анимации с использованием алгоритма Subspace–Skeletal Deformation, аниматоры могут изменять веса матрицы весов во времени, чтобы контролировать деформацию модели персонажа на разных кадрах анимации. Это может быть достигнуто с помощью различных методов интерполяции, таких как линейная интерполяция или сплайны, чтобы обеспечить плавные переходы между различными весами.

Основополагающей формулой данного алгоритма является: $\vec{v}'=\sum_i w_i W_i B_i^{-1}\vec{v}$, $\sum_i w_i = 1$.

Алгоритм SSD допускает возможность влияния не одной, а нескольких костей на одну вершину. Для каждой такой кости присуждается свой вес, то есть степень влияния данной кости на перемещения вершины. Чем больший вес будет дан определенной кости, тем сильнее вершина будет смещаться под ее влиянием.

Для упрощения работы были разработаны вспомогательные инструменты, которые способны автоматически построить кости по модели персонажа [15]. Также, для удобства работы с распределением весов были виртуальный аэрограф, позволяющий работать с весами через изображение [16], а также алгоритмы автоматического распределения весов [24].

Для каждой вершины модели алгоритм SSD определяет набор костей, которые оказывают влияние на эту вершину. Затем для каждой кости вычисляется соответствующая радиальная базисная функция, которая

определяет весовой коэффициент влияния этой кости на вершину. Эти весовые коэффициенты образуют матрицу весов. Матрица весов, полученная с использованием радиальных базисных функций, определяет, как каждая кость влияет на деформацию модели при анимации. Эта матрица затем используется для обновления позиции вершин модели в соответствии с перемещением костей.

Радиальные базисные функции зависят только от радиуса и являются скалярными функциями, определенными на некотором пространстве. Примером радиальной базисной функции является функция Гаусса, которая имеет следующий вид:

$$\varphi(R) = e^{(-\alpha R^2)}, \quad (15)$$

где R – радиус,

α – параметр, определяющий форму функции.

Алгоритм взвешенных ближайших соседей является методом классификации или регрессии, который используется для прогнозирования значения целевой переменной на основе ближайших соседей с учетом их весовых коэффициентов. Шаги выполнения алгоритма взвешенных ближайших соседей в рамках алгоритма костной анимации SSD выглядит следующим образом:

1. для каждой вершины модели определяются ближайшие кости. Это можно сделать путем вычисления расстояния от каждой вершины до каждой кости и выбора нескольких ближайших костей;

2. вычисляются расстояния между каждой вершиной и ее ближайшими костями. Это позволит определить, насколько каждая вершина близка к каждой кости;

3. применяется функция взвешивания, чтобы назначить весовые коэффициенты для каждой вершины и ее ближайших костей. Чем ближе вершина к кости, тем больший вес ему будет назначен;

4. собираются все весовые коэффициенты в матрицу весов. Каждая строка матрица соответствует вершине модели, а каждый столбец соответствует кости.

Матрица весов, полученная с использованием алгоритма взвешенных ближайших соседей, определит влияние каждой кости на деформацию вершин модели при анимации. Эта матрица затем используется в алгоритме SSD для обновления позиции вершин с учетом перемещений костей.

К алгоритму Subspace–Skeletal Deformation применимы те же вспомогательные инструменты, что и для алгоритма костной анимации Animation Space. В алгоритме SSD также используется интерполяция для нахождения промежуточных кадров, сплайны и кривые Безье для плавности движения между кадрами, а также кинематические ограничения движения.

Таким образом, был математически описан алгоритм костной анимации Subspace–Skeletal Deformation, а также используемые на практике инструменты для реализации качественной костной анимации.

2.3 Разработка алгоритмов костной анимации

Благодаря проведенному исследованию по теоритическим аспектам, можно построить блок–схему программы, изображенную на рисунке 1, с помощью которой будут реализованы выбранные алгоритмы костной анимации.

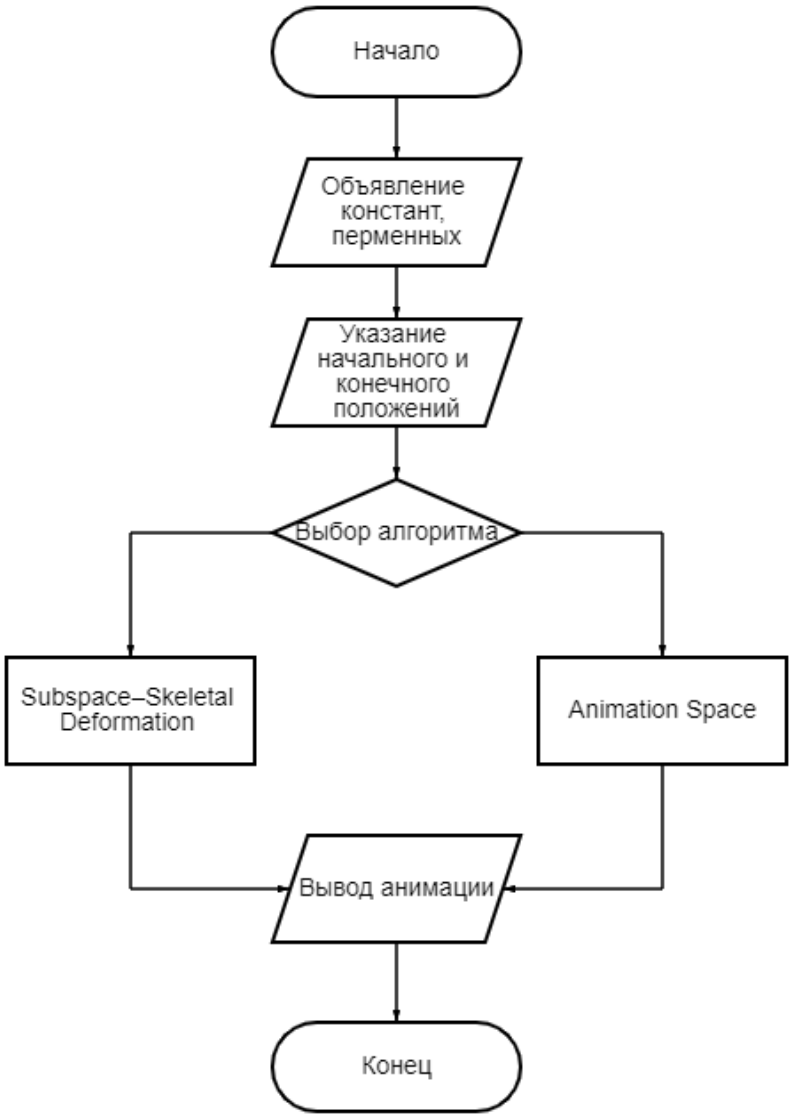

Рисунок 2 – Блок–схема программы с использованием алгоритмов костной анимации SSD и AS

Таким образом, была разработана блок–схема программы, с помощью которой можно приступить к реализации программного обеспечения.

2.4 Разработка диаграмм

Для реализации программы необходимо выбрать, какой язык программирования будет использоваться. Для этого приведем сравнение по

нескольким критериям. Оценка языка программирования по конкретному критерию представлена значениями от 1 до 5, где 1 – наименьшая оценка, а 5 – наивысшая. В таблице 2.1 приведено сравнение языков программирования. [5]

Таблица 2 – Сравнение языков программирования

Критерии\язык	C++	C#	Python
Скорость работы	5	5	4
Объем занимаемой ОП	4	4	4
Скорость написания	3	5	4
Простота	2	3	4
Итог	14	17	16

На основе итоговых баллов, было решено реализовать программу на языке C#, в среде разработки Visual Studio 2019.

C# – это высокоуровневый объектно–ориентированный язык программирования, разработанный компанией Microsoft. C# был создан как часть платформы .NET Framework, которая предоставляет разработчикам инструменты для создания различных типов приложений, включая Windows–приложения, веб–приложения и другие.[9]

Основные особенности языка C# включают в себя:

– объектно–ориентированный подход: C# поддерживает основные принципы объектно–ориентированного программирования, такие как наследование, полиморфизм и инкапсуляция;

– безопасность типов: C# требует, чтобы переменные и объекты имели явно определенные типы данных, что позволяет избежать ошибок типизации во время выполнения программы;

– сборка мусора: C# использует автоматическую систему сборки мусора, которая автоматически удаляет объекты, которые больше не

используются в программе что упрощает процесс управления памятью;

– многопоточность: C# имеет встроенную поддержку многопоточности, что позволяет разработчикам создавать приложения, которые могут обрабатывать несколько задач одновременно.

Язык C# используется для разработки широкого спектра приложений, включая приложения для Windows и мобильных устройств, веб–приложения, игры и другие типы приложений.

Visual Studio – это интегрированная среда разработки от Microsoft, которая используется для создания различных типов приложений, включая Windows–приложения, веб–приложения, мобильные приложения, игры и многое другое.

При разработке будет использоваться библиотека OpenTK, которая предоставляет обертку над OpenGL для языка C#.

OpenGL – это кроссплатформенный стандарт для создания графических приложений, который позволяет разработчикам создавать высококачественную графику и 3D–моделирование. OpenGL был разработан компанией Silicon Graphics в 1992 году и с тех пор был расширен и развит сообществом разработчиков.

Для реализации будет использоваться OpenGL версии 3.[11]

OpenTK – это библиотека для разработки приложений с использованием OpenGL, которая предоставляет удобный доступ к функциям OpenGL и другим системным вызовам на языках программирования C# и VB.NET.

Выполнение работы будет производится на операционной системе Windows 10 версии.

В данном разделе было произведено сравнение языков программирования, в результате которого выбран язык C# для реализации программного обеспечения. Была определена среда для разработки, операционная система и средства для реализации программы.

Глава 3 Реализация и сравнение алгоритмов костной анимации

3.1 Реализация алгоритмов костной анимации

Для использования библиотеки OpenTK, необходимо сначала добавить OpenGL в проект.

Рисунок 3 – Файлы, необходимые для работы с OpenGL

На рисунке 3 изображены файлы, которые необходимо переместить в папку со средой разработки Visual Studio, а также в пару корневых папок системы Windows.

Следующим шагом будет установка самой библиотеки OpenTK непосредственно в проект. Ее можно установить с помощью консоли диспетчера пакетов, используя код «Install–Package OpenTK», как показано на рисунке 4:

```
Узел консоли диспетчера пакетов версии 5.11.4.13

Чтобы увидеть все доступные команды NuGet, введите «get-help NuGet».

PM> Install-Package OpenTK
Идет восстановление пакетов для C:\Users\Илья\source\repos\ConsoleApp2\ConsoleApp2\ConsoleApp2.csproj...
Установка пакета NuGet OpenTK 4.7.7.
Фиксация восстановления...
Запись файла ресурсов на диск. Путь: C:\Users\Илья\source\repos\ConsoleApp2\ConsoleApp2\obj\project.assets.json
Восстановлен C:\Users\Илья\source\repos\ConsoleApp2\ConsoleApp2\ConsoleApp2.csproj (за 48 ms).
"OpenTK 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.Compute 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.Core 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.Graphics 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.Input 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.Mathematics 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.OpenAL 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.redist.glfw 3.3.8.30" успешно установлено в ConsoleApp2
"OpenTK.Windowing.Common 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.Windowing.Desktop 4.7.7" успешно установлено в ConsoleApp2
"OpenTK.Windowing.GraphicsLibraryFramework 4.7.7" успешно установлено в ConsoleApp2
"System.Runtime.CompilerServices.Unsafe 5.0.0" успешно установлено в ConsoleApp2
Выполнение действий NuGet заняло 276 ms
Прошло времени: 00:00:00.5685323
PM>
```

Рисунок 4 – Установка библиотеки OpenTK через консоль диспетчера пакетов

Теперь производим подключение библиотек:

```
using OpenTK.Graphics.OpenGL;
using OpenTK.Mathematics;
using OpenTK.Windowing.Common;
using OpenTK.Windowing.Desktop;
using System;
using System.Drawing;
```

Рисунок 5 – Подключение необходимых библиотек

Все библиотеки, изображенные на рисунке 5 необходимы для работоспособности программы.

```
public MainForm(int width, int height, string title) : base(GameWindowSettings.Default, new NativeWindowSettings() { Size = (width, height), Title = title })
{
    VSync = VSyncMode.On;
    // Initialize the root bone
    rootBone = new Bone(Vector3.Zero, Quaternion.Identity, Vector3.One)
    {
        Name = "Root Bone"
    };

    // Create child bones
    for (int i = 0; i < NumBones; i++)
    {
        Bone childBone = new Bone(new Vector3(0, BoneLength, 0), Quaternion.Identity, Vector3.One)
        {
            Name = "Child Bone " + (i + 1)
        };

        rootBone.AttachChildBone(childBone);
    }

    // Initialize the bone matrices
    boneMatrices = new Matrix4[NumBones + 1];

    using MainForm game = new MainForm(320, 480, "Comparison of algorithms");

    game.Run();
}
```

Рисунок 6 – Создание костей и окна вывода программы

Создаем окно для вывода объекта анимации, как показано на рисунке 6. Используем конструктор base, после чего создаем родительскую кость rootBone, а также дочерние кости childBone. Устанавливаем размер окна 320×480, и называем его «Comparison of algorithms».[13]

```
GL.MatrixMode(MatrixMode.Projection);
GL.LoadIdentity();
Matrix4 perspective = Matrix4.CreatePerspectiveFieldOfView((float)Math.PI / 4f, Width / Height, 0.1f, 100f);
GL.LoadMatrix(ref perspective);

GL.MatrixMode(MatrixMode.Modelview);
GL.LoadIdentity();
Matrix4 lookat = Matrix4.LookAt(new Vector3(0, 0, 5), Vector3.Zero, Vector3.UnitY);
GL.LoadMatrix(ref lookat);
```

Рисунок 7 – Создание матриц модели для алгоритма Animation Space

На рисунке 7 представлены вычисления матриц для алгоритма Animation Space. Здесь мы применяем значения объекта для матрицы модели.

```
private void Timer_Tick(object sender, EventArgs e)
{
    currentFrame++;
    if (currentFrame >= AnimationFrames)
        currentFrame = 0;

    float t = (float)currentFrame / AnimationFrames;

    // We use the AS algorithm to interpolate the position of the object
    Vector3 interpolatedPosition = AnimationSpace(controlPoints, weights, t);

    Refresh();
}


private Vector3 AnimationSpace(List<Vector3> controlPoints, List<float> weights, float t)
{
    if (controlPoints.Count != weights.Count)
        throw new ArgumentException("The number of control points must match the number of weights.");

    Vector3 interpolatedPosition = Vector3.Zero;

    // We perform summation of weighted control points
    for (int i = 0; i < controlPoints.Count; i++)
    {
        interpolatedPosition += weights[i] * controlPoints[i];
    }
    return interpolatedPosition;
}
```

Рисунок 8 – Программная реализация алгоритма Animation Space

На рисунке 8 изображена реализация алгоритма Animation Space. В методе Animation Space выполняется нахождение трансформированной позиции объекта, на основе формулы $\vec{v}'=\sum_i W_i\, p_i$.

В методе Timer_Tick используется текущий кадр анимации t для вызова метода Animation Space и получения трансформированной позиции объекта. После чего позиция объекта обновляется и происходит перерисовка сцены.

```
public void UpdateAnimation()
{
    foreach (Vertex vertex in vertices)
    {
        Vector3 interpolatedPosition = Vector3.Zero;

        foreach (WeightedBone weightedBone in vertex.WeightedBones)
        {
            Bone bone = weightedBone.Bone;
            float weight = weightedBone.Weight;

            Matrix4 inverseBindPoseMatrix = Matrix4.Invert(bone.BindPoseMatrix);

            //using the SSD algorithm formula to iterate over all vertices and calculate the interpolated position
            Matrix4 weightedMatrix = weight * bone.CurrentPoseMatrix * inverseBindPoseMatrix;

            Vector4 transformedPosition = Vector4.Transform(new Vector4(vertex.Position, 1.0f), weightedMatrix);
            interpolatedPosition += transformedPosition.Xyz;
        }

        vertex.Position = interpolatedPosition;
    }
}
```

Рисунок 9 – Программная реализация алгоритма SSD

На рисунке 9 представлен класс алгоритма Subspace–Skeletal Deformation, с использованием основной формулы данного алгоритма.

В данном разделе были реализованы алгоритмы создания костной анимации Subspace–Skeletal Deformation и Animation Space. Был приведен листинг программы с пояснениями.

3.2 Тестирование выполненной программы

При компиляции программы мы получаем результат, изображенный на рисунке 10:

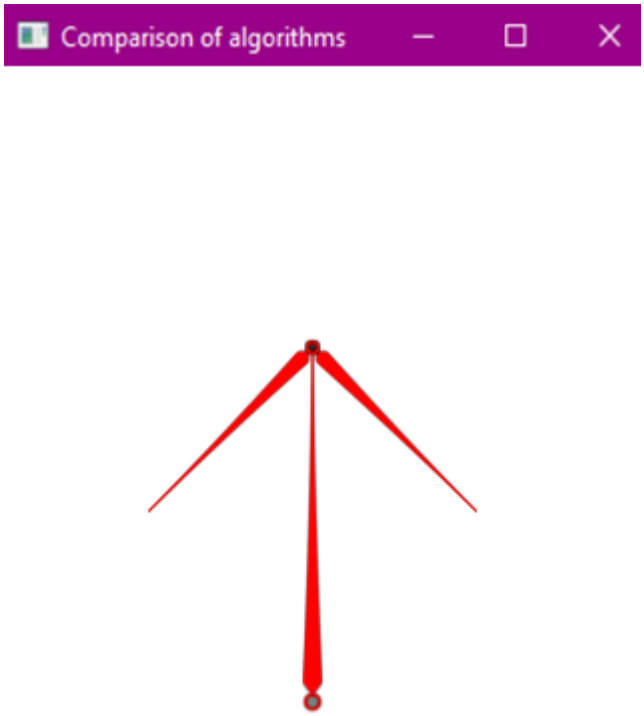

Рисунок 10 – Результат работы программы алгоритмом AS

На рисунке 10 изображен результат выполнения программы с помощью алгоритма костной анимации Animation Space.

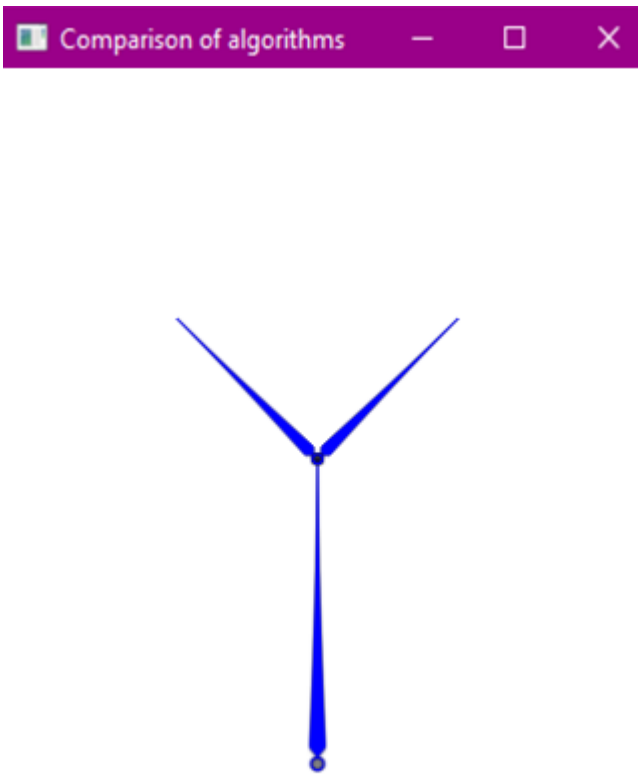

Рисунок 11 – Результат работы программы алгоритмом SSD

На рисунке 11 изображен результат выполнения программы с помощью алгоритма костной анимации Subspace–Skeletal Deformation.

В результате работы программы мы получаем двигающиеся кости, которые перемещаются путем реализации выбранных алгоритмов костной анимации. Также этот результат свидетельствует о корректной работе программы.

Благодаря реализации выбранных алгоритмов костной анимации, удалось получить анимацию костей.

3.3 Сравнение разработанных алгоритмов костной анимации

Теперь, на основе реализованных алгоритмов костной анимации, проведем сравнение их производительности. Сравнивать будем такие алгоритмы, как: с использованием весовых коэффициентов (SSD – subspace–skeletal deformation), и AS (animation space).

Алгоритм SSD – самый простой и распространенный метод создания анимации. Однако он имеет несколько недостатков, наиболее существенный из которых заключается в том, что сетки, созданные данный методом, теряют объем при повороте соединений на большие углы. Это происходит из–за его недостаточной гибкости. При нахождении положения вершины в новой позе матрицы преобразования влияющих костей интерполируются линейным образом. Линейная интерполяция этих матриц не эквивалентна линейной интерполяции их вращений. Но, несмотря на этот существенный недостаток, алгоритм SSD остается популярным благодаря своей простоте и вычислительной эффективности.

Алгоритм AS, в сравнении с алгоритмом SSD, имеет большую гибкость, изменяя одиночное положение позы покоя каждой вершины. Комбинация обратного преобразования кости, положения покоя вершины и весового коэффициента увеличивает количество переменных весов на влияние кости до четырех. Это позволяет каждой кости независимо влиять на каждый компонент положения вершины, уменьшая проявления заломов, как в алгоритме SSD. Еще одно преимущество данного алгоритма заключается в том, что он позволяет создавать вершины, которые являются линейными комбинациями существующих вершин. В результате этого пропадает необходимость пересчета весов модели.

Сравнение производительности рендеринга алгоритмов SSD и AS представлено в таблице 3. В ней представлены следующие параметры:

– количество вершин,

– количество костей,

– показания кадров в секунду(FPS),

– алгоритм Subspace–Skeletal deformation (SSD),

– алгоритм Animation Space (AS).

Таблица 3–Таблица сравнения производительности алгоритмов

Вершины	Кости	FPS	
		SSD	AS
1000	3	291	458
2000	3	263	437
4000	5	238	414
8000	7	201	386

Исходя из перечисленных плюсов и минусов алгоритмов и показаниям их производительности, самым продуктивным является алгоритм Animation Space.

Подводя итог по данному разделу можно с уверенностью сказать, что в ходе сравнения разработанных алгоритмов лучше себя проявил алгоритм костной анимации Animation Space.

Заключение

В выпускной квалифицированной работе представлена разработка программного обеспечения для создания костной анимации с помощью алгоритмов костной анимации, а также проведено сравнение между ними на основе результатов тестирования.

В первой главе подробно описаны самые распространенные алгоритмы, использующиеся при создании анимации.

Была рассмотрена краткая математическая часть каждого из приведенных алгоритмов. Также были перечислены их достоинства и недостатки, на основе которых можно выбрать подходящий алгоритм под определенный проект.

Также ознакомились с компанией "ДЖАСТМОБИ", узнали ее деятельность, а также области применения костной анимации. Ознакомились с критериями выбора алгоритмов костной анимации.

Далее было выбрано 2 алгоритма для разработки и дальнейшего сравнения, а это: Animation Space и SSD.

В заключении были поставлены задачи на разработку алгоритмов, которые решались по ходу выполнения выпускной квалификационной работы.

Во второй главе подробно описана математическая часть алгоритмов костной анимации Animation Space и Subspace–Skeletal deformation. Также были подробно описаны инструменты, которые используются при создании анимации данными алгоритмами.

Была разработана блок–схема программы, с помощью которой можно приступить к реализации программного обеспечения.

И было произведено сравнение языков программирования, в результате которого был выбран язык C# для реализации программного обеспечения. Была определена среда для разработки, операционная система и средства для

реализации программы.

В третьей главе представлен листинг программы, результат ее выполнения и сравнение алгоритмов костной анимации на основе тестирования.

Были реализованы алгоритмы создания костной анимации Subspace–Skeletal Deformation и Animation Space. Был приведен листинг программы с пояснениями.

Так же был произведен тест работы программы, в результате которого выяснили, что программа работает корректно, а алгоритмы улучшили плавность анимации.

Благодаря реализации выбранных алгоритмов костной анимации, удалось убедиться в корректной работе программы, а также мы получили двигающиеся кости, которые перемещаются путем реализации выбранных алгоритмов костной анимации.

В заключении было проведено сравнение между алгоритмами костной анимации Animation Space и Subspace–Skeletal Deformation, в результате которого лучшим оказался алгоритм костной анимации Animation Space.

В ходе проведенных исследований, были получены математические знания, а также приобретены широкие знания в области компьютерной анимации.

Список используемой литературы и используемых источников

1. Алберг Дж. Теория сплайнов и ее приложения / Дж. Алберг, Э. Нильсон, Дж. Уолш; пер. с англ. Ю. Н. Субботина; под ред. С. Б. Стечкина; с доб. С. Б. Стечкина, Ю. Н. Субботина. – Москва: Мир, 1972. – 316 с.: ил. – Библиогр.: с. 267-269. – Предм. указ.: с. 310–311. – Имен указ.: с. 312–313.

2. Б.В.Соболь, Б.Ч.Месхи, И.М.Пешхоев. Практикум по вычислительной математике. – Ростов–на–Дону: Феникс, 2008.

3. Вычислительная математика [Электронный ресурс]: учебное пособие в двух частях. Ч. 1 / В. Н. Варапаев [и др.]. – Москва: МГСУ: Ай Пи Эр Медиа, 2017. – 88 с. – (Прикладная математика). – ISBN 978–5–7264–1455–3.

4. Воеводин В. В. Вычислительная математика и структура алгоритмов. – Москва: Издательство МГУ, 2006. – 112 с.

5. Джейсон Visual C# .NET. Полное руководство / Джейсон, Майк Прайс; Гандэрлой. – М.: Корона Принт, 2004. – 960 с.

6. Задорожный А.Г., Киселев Д.С. Построение сплайнов с использованием библиотеки OpenGL, 2019. –13с.

7. Интерполяция алгебраическими многочленами. Сплайн–интерполяция: учебное пособие / Министерство науки и высшего образования РФ, Федеральное государственное бюджетное образовательное учреждение высшего образования "Воронежский государственный университет" составители: А. П. Карпова, М. Н. Силаева. – Воронеж: Издательский дом ВГУ, 2022. – 64 с.

8. Курило Л.В. Теория и практика анимации: Ч.1. Теоретические основы туристской анимации: Учебное пособие, М. Советский спорт, 2006. – 195с.

9. Нейгел, К. C# 2005 для профессионалов / К. Нейгел. – М.: Вильямс, 2006. – 966 с.

10. Осипа, Дж. 3D–моделирование и анимация лица. Методики для

профессионалов / Дж. Осипа. – М.: Диалектика, 2008. – 400 с.

11. Официальный сайт OpenGL – https://www.opengl.org/

12. Пулькин С. П. Вычислительная математика: учебное пособие для вузов / С. П. Пулькин, Л. Н. Никольская, А. С. Дьячков. – Москва: Просвещение, 1980. – 176 с.

13. Рихтер CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C# / Рихтер, Джефри. – М.: Питер, 2007. –656 с.

14. Цисарж, В. Математические методы компьютерной графики / В. Цисарж, Р. Марусик. – Киев: Факт, 2004. – 464 с.

15. Baran, I. Automatic rigging and animation of 3D characters / I. Baran, J. Popovic // ACM SIGGRAPH 2007 papers.— SIGGRAPH ’07.— New York, NY, USA: ACM, 2007

16. Hagland, T. A Fast and Simple Skinning Technique / T. Hagland // Game Programming Gems / Ed. by M. DeLoura.— Rockland, MA, USA: Charles River Media, 2000.— 614 pp.

17. Kry, P. G. Eigenskin: real time large deformation character skinning in hardware / P. G. Kry, D. L. James, D. K. Pai // Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation. – SCA ’02. – New York, NY, USA: ACM, 2002. – Pp. 153–159.

18. Lander, J. Skin them bones: Game programming for the web generation / J. Lander // Game Developer Magazine. –1998. –May. – Pp. 11–16.

19. Lewis, J. P. Pose space deformation: a unified approach to shape interpolation and skeleton driven deformation / J. P. Lewis, M. Cordner, N. Fong // Proceedings of the 27th annual conference on Com71 puter graphics and interactive techniques. – SIGGRAPH ’00. –New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. – Pp. 165–172.

20. Magnenat–Thalmann, N. Joint-dependent local deformations for hand animation and object grasping / N. Magnenat-Thalmann, R. Laperri`ere, D. Thalmann // Proceedings on Graphics interface ’88. – Toronto, Ont., Canada,

Canada: Canadian Information Processing Society, 1988. – Pp. 26–33.

21. Merry, B. Animation space: A truly linear framework for character animation / B. Merry, P. Marais, J. Gain // ACM Trans. Graph. – 2006. –October. – Vol. 25. – Pp. 1400–1423.

22. Sloan, P. – P. J. Shape by example / P. – P. J. Sloan, C. F. Rose, III, M. F. Cohen // Proceedings of the 2001 symposium on Interactive 3D graphics. – I3D ’01. – New York, NY, USA: ACM, 2001. – Pp. 135–143.

23. Wang, X. C. Multi-weight enveloping: least-squares approximation techniques for skin animation / X. C.Wang, C. Phillips // Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation. – SCA ’02. – New York, NY, USA: ACM, 2002. –Pp. 129–138.

24. Wareham, R. Bone glow: An improved method for the assignment of weights for mesh deformation / R. Wareham, J. Lasenby // AMDO 2008 / Ed. by F. J. P. L’opez, R. B. Fisher. – Vol. 5098. – Berlin Heidelberg: Springer–Verlag, 2008.

25. Woodland, R. Filling the Gaps – Advanced Animation Using Stitching and Skinning / R. Woodland // Best of Game Programming Gems / Ed. by M. DeLoura. – Boston, MA, USA: Course Technology, Cengage Learning, 2008. – 549 pp.