Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент,

_____
«____»_____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____А.А.Замышляева
«____»_____2019 г.

Математическое моделирование движений пальцев руки человека

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2019.56.ПЗ ВКР

Руководитель работы, ст. преп. кафедры ПМиП
_____/М.Ю. Сартасова
«____»_____2019 г.

Автор работы
Студент группы ЕТ-412
_____/Д.М. Кичеев
«____»_____2019 г.

Нормоконтролер, ассистент
_____/Н.С. Мидоночева
«____»_____2019 г.

Челябинск
2019

АННОТАЦИЯ

Кичеев Д.М. Математическое моделирование движения пальцев руки человека. – Челябинск: ЮУрГУ, ЕТ-412, 72 с., 37 ил., библиогр. список – 13 наим., 1 прил.

Целью данной работы является разработка математической модели движения пальцев руки человека с реализацией в виде компьютерной программы.

В первом разделе был проведен анализ предметной области, рассмотрено анатомическое строение кисти руки человека. Были проанализированы биомеханические протезы, по их строению.

Во втором разделе описана математическая модель движения пальцев руки человека. Выделены примитивы для строения трехмерной модели механического манипулятора кисти руки.

Третий раздел посвящен разработке архитектуры сверточной нейронной сети для решения задачи оценки креативного потенциала личности. Также в данном разделе представлены результаты обучения данной нейронной сети.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Согласно исследованиям в области медицины повреждения пальцев и кисти руки человека составляют 30–57% от общего числа повреждений опорно-двигательного аппарта, и в 12,5% случаев являются причиной инвалидности человека [1].

Люди, получившие серьезные повреждения пальцев и кистей рук, зачастую теряют способность к выполнению трудовой деятельности, самообслуживания; их психологическое состояние также ухудшается.

При серьезных травмах кистей рук производят хирургическое удаление конечностей. Предлагаемые взамен биопротезы либо не выполняют необходимый для человека функционал кисти руки, либо очень высоки по стоимости и не достижимы для среднестатистического человека.

В нашем веке, когда 3D печать становится обыденностью, можно задаться вопросом создания более дешевых протезов кистей рук и пальцев руки человека. Так как сам по себе пластик дешев, а 3D принтеры могут напечатать деталь любой сложности, то можно реализовать приложение, которое по нужным нам параметрам будет печатать полностью функциональный протез кисти руки.

Но чтобы реализовать полностью функционирующий манипулятор, нужно разобрать модель движения пальцев руки человека. С помощью данной модели разработать приложение, которое по введенным параметрам верхних конечностей будет строить трехмерную кисть руки, на которой и будет проверяться физико-математическая модель движения пальца.

Таким образом, можно выделить, что целью данной работы является разработка математической модели движения пальцев руки человека с реализацией в виде компьютерной программы.

Для достижения поставленной цели необходимо решить следующие задачи:

1) изучить и проанализировать анатомию кисти руки, а также рассмотреть строение современных биопротезов;

2) разработать математическую модель движения пальцев руки человека;

3) написать компьютерную программу, реализующую математическую модель;

4) реализовать функционирующий интерфейс программы;

5) проверить работу программы на экспериментальных данных.

1 МЕТОДЫ И СПОСОБЫ МОДЕЛИРОВАНИЯ ДВИЖЕНИЯ ПАЛЬЦЕВ РУК ЧЕЛОВЕКА

1.1 Существующие модели кисти руки

1.1.1 Анатомия руки человека

Для того, чтобы начать строить математическую модель кисти руки человека, нужно разобраться в анатомии строения кисти руки человека. Анатомия кисти руки человека довольно-таки сложна, поскольку это одна из самых подвижных частей тела человека, которую мы используем чуть ли не ежесекундно. В ней множество костей, связок и мышц, также человеческие пальцы, в отличие от конечностей животных, имеют ногти. Также кожа рук отличается от кожного покрова тела человека, имея специфические складки и особую чувствительность. Итак, начнем рассмотрение строения кисти человека с обзора строения костей (рисунок 1.1).

Рисунок 1.1 – Костевое строение кисти человека

Глядя на рисунок можно разделить кости кисти руки человека на три отдела: запястье, включающее восемь костей; пясть, состоящая из пяти длинных костей, и пальцы, насчитывающие в общей сумме четырнадцать фаланг. Рассмотрим все три отдела.

Начнем с запястья. «Кости запястья, располагаются в два ряда. Верхний, или проксимальный, ряд прилегает к дистальному отделу костей предплечья, образуя эллиптическую, выпуклую в сторону предплечья суставную поверхность; другой ряд – нижний, или дистальный, обращен к пясти (рисунок 1.2).

Рисунок 1.2 – Кости пясти кисти человека

К костям первого ряда запястья, если считать от лучевого края кисти к локтевому, относятся следующие кости: ладьевидная, полулунная, трехгранная и гороховидная.

Второй ряд костей запястья составляют соответственно: кость-трапеция, трапециевидная кость, головчатая кость и крючковидная кость, как показано на рисунке выше» [2].

Пясть, состоит из пяти костей – пять коротких трубчатых костей кисти, отходящих в виде лучей от запястья. Нумерация пястных костей соответствует

нумерации сочленяющихся с ними пальцев. В пястных костях различают основание, тело и головку (рисунок 1.3).

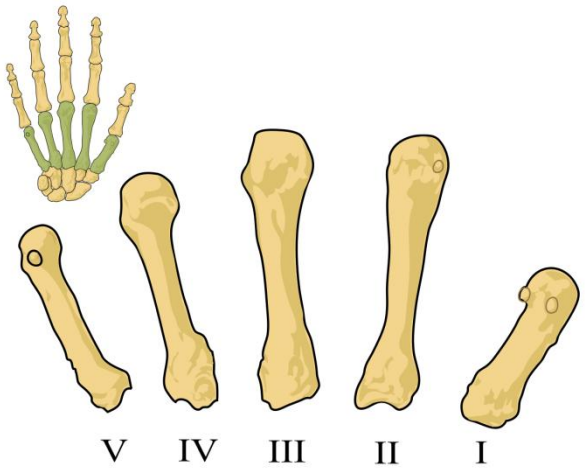


Рисунок 1.3 - Костевое строение пясти

«Утолщенный конец пястной кости называется её основанием. Пястная кость сочленена как с дистальным рядом костей запястья, так и с соседними пястными костями. Тело пястной кости (её основная часть) имеет несколько изогнутую к тылу форму, содержит питательный канал, открывающийся с ладонной стороны кости питательным отверстием. Головка пястной кости шаровидная, её суставная поверхность несколько возвышена с ладонной стороны.

Кости пальцев состоят из фаланг. Фаланги – короткие трубчатые кости, образующие скелет пальцев конечностей позвоночных животных, в том числе человека (рисунок 1.4).

У человека каждый палец, кроме большого, состоит из трёх фаланг, а большой – из двух. Эти три фаланги называются основной, средней и ногтевой. Основная фаланга крепится к пястной кости. На руке самая длинная фаланга – основная третьего пальца, а самая толстая – основная фаланга большого пальца. Каждая фаланга представляет собой удлинённую косточку, имеющую в средней части форму полуцилиндра, плоская часть которого обращена на ладонную, а

выпуклая на тыльную сторону. Концевые части фаланги несут суставные поверхности» [3].

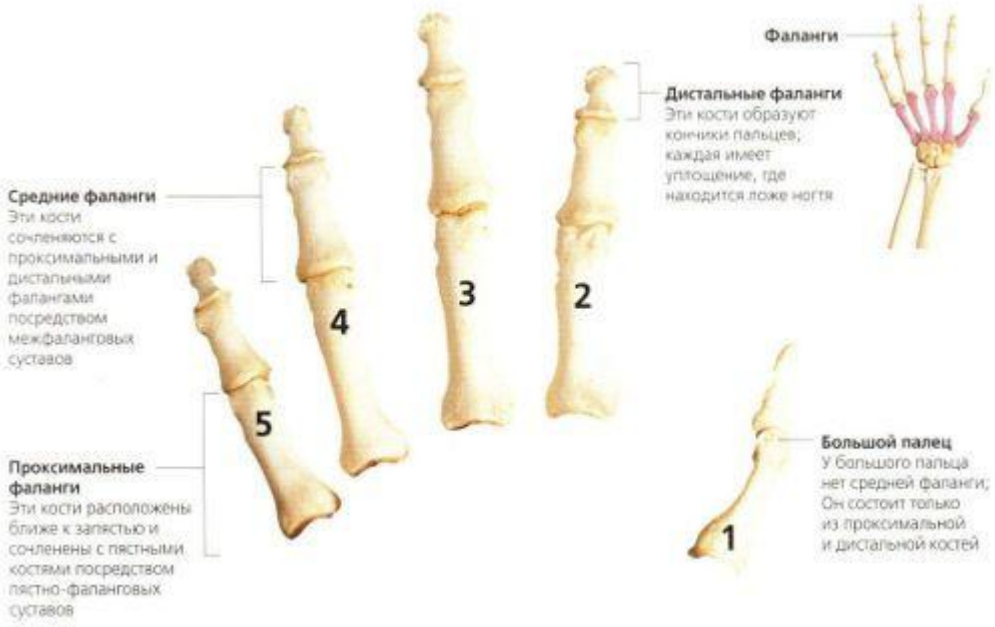


Рисунок 1.4 – Костевое строение фаланг пальцев

«В медицине применяются следующие термины для фаланг кисти:

- проксимальная (основная) фаланга;
- средняя фаланга;
- дистальная (ногтевая) фаланга.

Рассмотрев строение костей пальцев руки человека, приступим к рассмотрению мышц, которые отвечают за подвижность кисти руки человека и за сцепку между костьми» [4].

Мышц в кистях немного. Расположены они на ладони, внутри и снаружи. На долю большого пальца приходится куда больше мышц, чем на остальные. В самих пальцах, являющихся самым подвижным элементом рук, мышц нет. Сгибатели и разгибатели пальцев находятся в ладони и в предплечьях. Если посмотреть на тыльную сторону кисти с пальцами, максимально поднятыми вверх, то можно увидеть связки-разгибатели, тянущиеся от пальцев к запястью (рисунок 1.5).

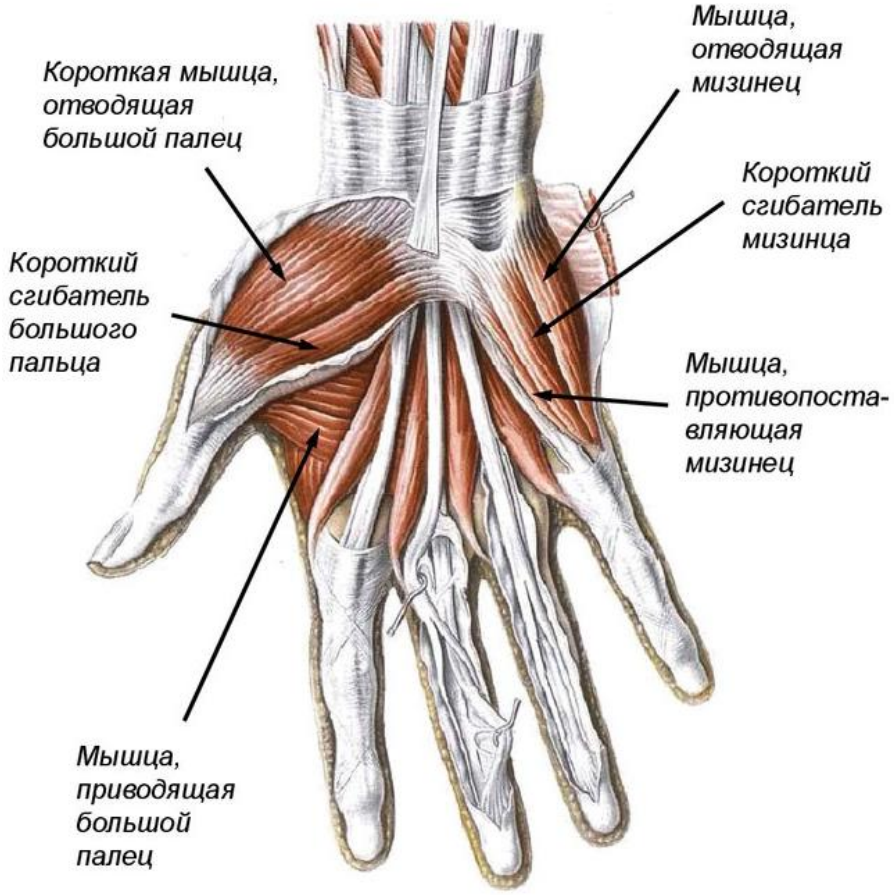


Рисунок 1.5 – Мышцы кисти руки человека

«В кисти человека проходят три основных нерва: срединный, локтевой и лучевой – у каждого человека они расположены по-своему. Каждый нерв на пути прохождения «отдаёт ветви», постепенно истончаясь. Часть этих ветвей будет собирать информацию, а часть – передавать стимулы от центральной нервной системы (рисунок 1.6).

Рисунок 1.6 – Нервы руки человека

Полностью рассмотрев строение кисти руки человека, можем заметить, что костей много, и они не большие, кисть имеет хорошую пластичность и подвижность, из-за этого при травмах работоспособность кисти заметно снижается. Кости запястья образуют как малоподвижные соединения так и суставы» [5]. Лучезапястное соединение имеет форму полукруга и обеспечивает сгибание и разгибание кисти. Это очень сложный сустав, дающий кисти широкий диапазон движений. Кости запястья соединены между собой связками и образуют прочное сочленение, способное выдерживать большие нагрузки при опоре на кисть. У основания большого пальца расположен седловидный сустав, позволяющий совершать движения в двух осях, что делает большой палец очень функциональным. Также еще, в дальнейшем, мы будем использовать факт, что Кости фаланг имеют шаровидные соединения, что позволяет пальцам сгибаться только в одной плоскости.

1.1.2 Биопротезы

Людям, как биологическому виду, у которого есть конечности, в результате несчастных случаев, катаклизмов, болезней и прочих жизненных ситуаций, к сожалению, иногда свойственно их терять. И, в отличие от некоторых животных, нам повезло меньше. Например, тритонов и ящериц природа одарила регенерацией потерянных частей тела, а вот человека (впрочем, как и большинство других видов) оставила наедине с собой в таких ситуациях. Но поскольку человек обладает интеллектом, а цивилизация следует по технологическому пути, то еще с древних времен утраченные конечности научились заменять протезами.

«На нынешней стадии развития технического прогресса и научных достижений люди с физическими недостатками имеют большой выбор различных возможностей и ассортимент продукции протезной индустрии, а также полный ассортимент различного адаптивного оборудования. Сейчас в сфере протезирования, в основном благодаря развитию ИТ и синергии индустрий, наблюдается всплеск новых разработок и достижений. Основная цель, которую пытаются достичь ученые и инженеры всего мира – воплотить в искусственном изделии все функции живой руки или ноги.

Впрочем, все бионические устройства разных фирм, институтов и центров пока что не сильно похожи на свои естественные прототипы. Помимо прочих сложностей, основной элемент, которого не хватает всем разработкам – это похожая на настоящую кожа для наружного покрытия. Впрочем, вполне вероятно, что в скором времени эта проблема будет решена путем изготовления полноценной искусственной кожи – сейчас уже проводятся эксперименты по соединению в единое работающее целое нервной ткани и электронных устройств.

Практически до конца 20-го века все изобретения в области протезирования были механического характера, в некоторых случаях сгибание регулировалось вручную. Основными проблемами протезов тех времен (да и, собственно,

разработанных ранее конструкций, применяющихся во многих случаях до сих пор) были отсутствие какой-либо связи с организмом, негибкость и недолговечность. Протезы, которые заменяли руку или ногу, не могли функционировать, как полноценный их прототип – это всего лишь суррогат, заменяющий активные части тела, но неспособный приблизиться по возможностям к естественному аналогу. Это и есть главный минус протезов – их «внешний» характер и низкая функциональность. Все, что остается делать их обладателю, это использовать их как элемент гардероба, который со временем изнашивается и становится непригодным к дальнейшей эксплуатации» [6].

Не столь давно в сфере протезирования появилось такое направление, как «биомехатроника», которое представляет собой соединение робототехники и нервных клеток человека. «Задачей научных исследований в этом направлении является разработка искусственных конечностей, которыми можно будет управлять лишь силой мысли, а функциональность будет повторять оную у заменяемой конечности человека с максимальной точностью. Кроме создания роботизированных протезов, способных «вести диалог» с нервной системой, важным направлением является остеоинтеграция, то есть сращивание искусственного модуля и кости, что позволит обойтись без гильзы протеза. Эксперименты по сращиванию титановых имплантатов с кожей, мышцами и костной тканью проводятся регулярно, а некоторые компании (в частности, немецкая ESKA Implants с их технологией Endo-Exo) уже представили серийные разработки. Исходя из нынешнего уровня развития технологий, уже в скором времени человек, потерявший конечность, сможет почувствовать себя отчасти киборгом» [7].

Протезирование рук возможно с помощью двух принципиальных типов устройств: механических и биоэлектрических. Механические – протезы, как правило, максимально приближенные к внешнему виду руки, что позволяет человеку не выделяться из толпы. В некоторых случаях протез способен к захвату

и удерживанию предметов с помощью бандажей, которые закрепляются к плечу, а при потребности кисть может заменяться на крюк.

Несмотря на то, что механические протезы существуют уже не одно столетие, предел их функциональности, похоже, давно достигнут. Поэтому дальнейшее развитие связано с биоэлектрическими протезами. Такие механизмы имеют в своей конструкции электроды, считывающие ток, вырабатываемый мускулами при их сокращении. Затем эти данные передаются на микропроцессор, который посредством команд моторам приводит протез в действие. Протез выполняет функции вращения кистью, захвата и удержания предметов. При этом биоэлектрический протез позволяет пользоваться такими миниатюрными вещами, как шариковая ручка, ложка, вилка и т. д. (рисунок 1.7).

Рисунок 1.7 – Бионический протез руки

Протез руки i-LIMB Hand, созданный компанией Touch Bionics, является последним достижением в кибермедицине. Управление им осуществляется интуитивной системой, в основе которой лежит миоэлектрическая технология – сенсор в виде металлической пластинки, соприкасающейся с кожей, улавливает

нервные импульсы от мышц. Благодаря встроенным миниатюрным электромоторам I-Limb способен имитировать множество функций, присущих человеческой руке.

«В своих продуктах Touch Bionics не обделила и тех пользователей, которые не желают скрывать искусственную сущность конечности и выглядеть подобно Терминатору. Специально для таких желающих был разработан вариант протеза в «прозрачной упаковке», через которую видна вся начинка устройства. Спрос на такое исполнение обеспечивают, в основном, мужчины, причем в первую очередь – военные. В то же время и косметическое покрытие i-Limb впечатляет (рисунок 1.8)» [8].

Рисунок 1.8 – протез фирмы Touch Bionics

В области технологии создания искусственных рук трудно не отметить и сверхсовременный протез Luke Arm, созданный Майклом Голдфарбом из Университета Вандербильта и компанией Deka Research (рисунок 1.9). В нем используется компактный однокомпонентный ракетный двигатель, аналогичный по конструкции ранее использовавшимся на космических шаттлах. В качестве

топлива применяется перекись водорода: под воздействием катализатора топливо нагревается и выделяющийся в процессе пар открывает и закрывает клапаны, которые соединены с суставами протеза. Вся эта конструкция заменяет аккумуляторы и электромоторы. Название же Luke Arm было дано в честь Люка Скайуокера из «Звездных войн».

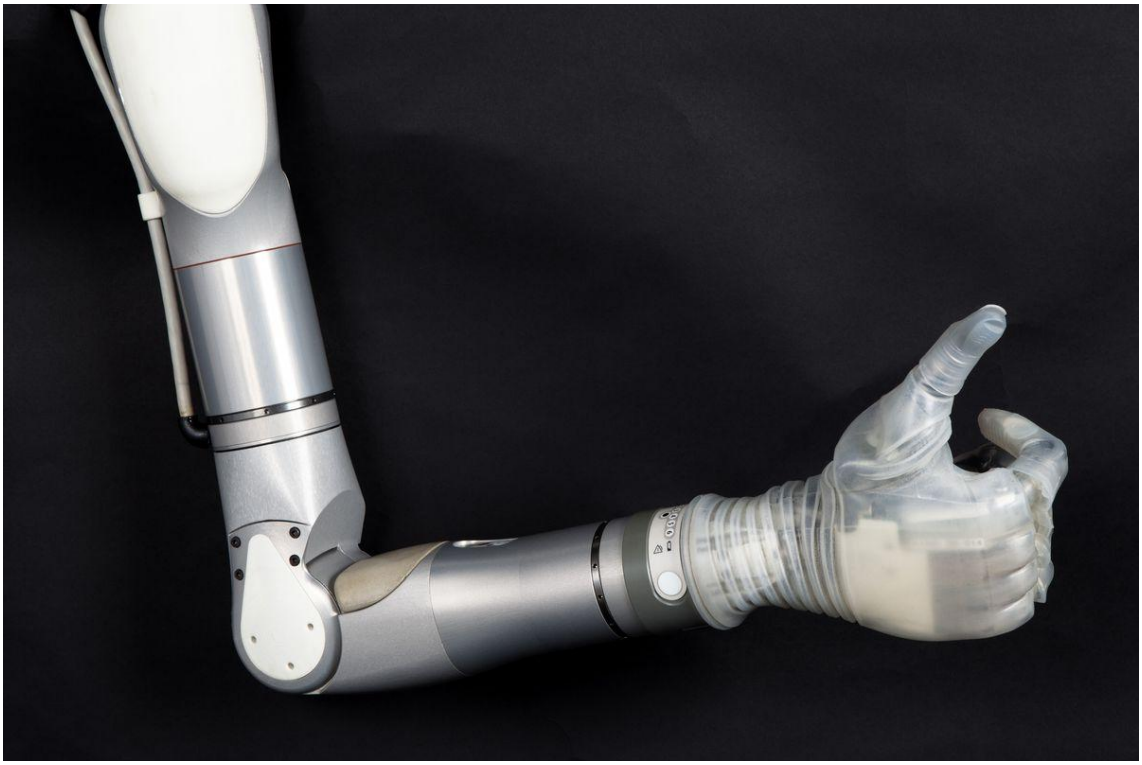


Рисунок 1.9 – Протез Luke Arm

С точки зрения научно-технологического уровня бионической руке Luke Arm не уступает SmartHand, разработанный группой ученых из шести стран: Швеция, Дания, Италия, Исландия, Ирландия, Израиль (рисунок 1.10). «Как и в большинстве других конструкций, для управления SmartHand используются нервные окончаний в культе ампутированной руки. Однако, данный протез уникален тем, что способен имитировать не только движения руки человека, но и воспроизводить ощущения от прикосновения к объекту. Все это реализуется с помощью четырех электродвигателей и 40 датчиков, которые активируются при прикосновении искусственными пальцами к объекту. Первая операция, позволившая пациенту не только пользоваться данным протезом, но и

чувствовать кончиками пальцев, была проведена в 2009 году в Тель-авивском университете» [9].

Рисунок 1.10 – Протез SmartHand

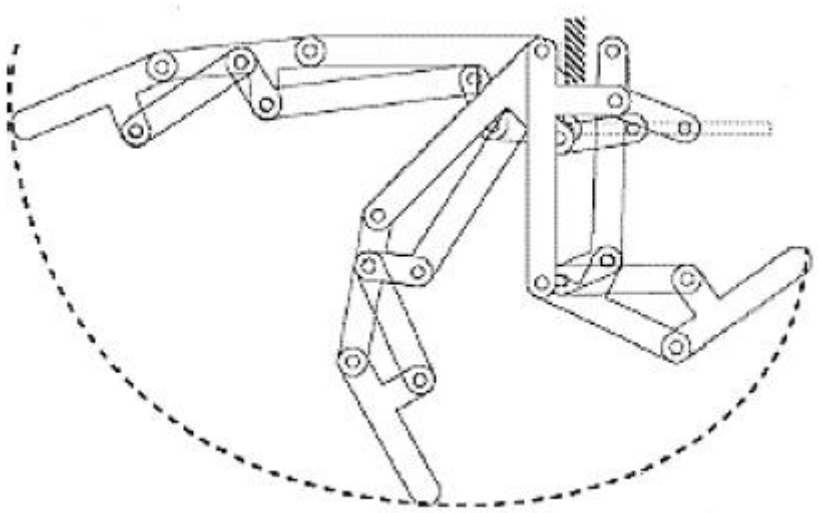
«Судя по всему, британских ученых на создание конструкции протеза также натолкнул фильм «Звездные войны». Они разработали технологию прикрепления к человеческому скелету металлических протезов (пальцы, руки, ноги). При этом им удалось срастить металлические структуры с живой материей с помощью специально выращенных человеческих тканей. Годовые испытания на добровольцах показали, что протез данного вида хорошо приживается, не отторгается организмом, а очевидным преимуществом является то, что его не нужно снимать и он не нуждается в гильзе, таким образом будучи гораздо лучше связан с телом» [10].

«Одной из последних разработок стал миоэлектрический протез руки BeBionic, который способен вращаться на 135 градусов и производить сгибание до 35 градусов. Как и у большинства современных протезов, возможности BeBionic максимально приближаются к естественным движением руки человека. Управление осуществляется микропроцессором, преобразующим в команды для электродвигателя сигналы от датчиков, распознающих движения сохранившихся мышц культи. Еще один плюс BeBionic – специальное программное обеспечение позволяет осуществлять различные виды захватов предметов и регулировать степень сжатия пальцев» [11].

«Создатели роботизированных протезов конечностей, иногда задумываются и о дизайне продукции, который радикально ломает стереотипы. Как и прочие, этот концептуальный протез «Immaculate» авторства дизайнера Ганса Александра Хусклеппа (Hans Alexander Huseklepp) непосредственно связан с нервной системой пользователя. Суставы построены так, чтобы обеспечить максимальную степень свободы – шарнирные элементы позволяют с легкостью осуществлять движения даже в более широких пределах, чем здоровая рука. Но в первую очередь «Immaculate» выделяется достаточно странным видом и позиционируется скорее, как аксессуар для людей, которые хотят выделиться и не скрывают факт потери конечности. Определенная логика в этом есть, ведь, например, очки, также выполняя реабилитационную функцию и будучи весьма заметными (и тем самым очевидно информируя окружающих, что у их хозяина плохое зрение), при этом стараются сделать внешне максимально эстетичными и подходящими к стилю владельца. В таком случае и протез конечности вполне может стать «аксессуаром», раз уж его наличие необходимо» [12].

«В качестве отдельного примера, пожалуй, наивысшего достижения на сегодняшний день – один из самых дорогих протезов мира. Это бионический протез руки, стоимость которого составляет порядка 6 миллионов долларов. Владелец данного протеза при его использовании может вращать им на 360 градусов, поворачивать кисть, а также ощущать прикосновения, кончиками

пальцев различать структуру поверхности и даже температуру объекта. Все это достигается с помощью нейроинтерфейса – этот метод дает значительно большую полноту ощущений и возможностей управления в сравнении с использованием датчиков в более доступных миоэлектрических протезах. Авторы проекта – лаборатория прикладной физики Университета Джона Хопкинса, а финансирование поступает от небезызвестного агентства DARPA, научного подразделения Пентагона» [14].



1.11 – Схема сгибания протеза пальца

«По всем этим протезам можно заметить, что представление механизма сгибания пальца, по сути, один и тот же. Стержни конфигурируются таким образом, что они складываются одновременно, просто перемещая одну планку вперед или назад, с этой системой вам не нужны моторы, а производство является простым и дешевым (рисунок 1.11).

Движения довольно ограничены только потому, что, как уже было сказано, сгибание производится одновременно в трех суставах.

Здесь мы можем увидеть пример этого механизма, который называется механизмом Торонто (рисунок 1.12)» [13].

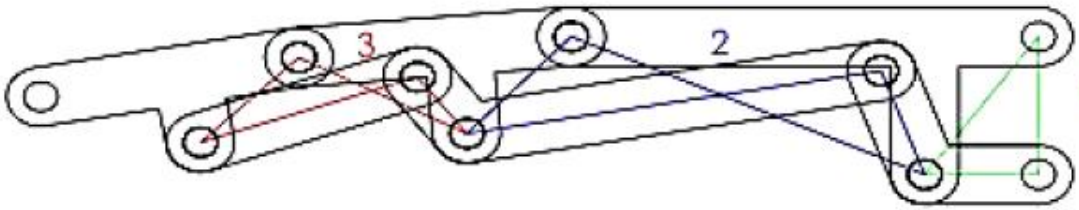


Рисунок 1.12 – Схема Торонтовского механизма

1.2 Геометрическое моделирование 3D объектов

Задача геометрического моделирования является важной областью в 3D моделировании. Поскольку данные о физических объектах реального мира не могут быть целиком введены в компьютер, необходимо априори ограничить объем информации об объекте в рамках интересующего нас вопроса.

Если будет выбрано подходящее представление геометрической модели объекта для оговоренного круга задач, она будет решена эффективно, и наоборот.

Геометрические модели в реальной жизни используются для решения многих задач визуализации, построения расчетных сеток, генерации управляющих программ. В первую очередь они предназначены для хранения информации о форме объектов и их взаимном расположении, и предоставления ее для обработки в удобном для компьютерной программы виде. В этом – ключевое отличие электронной геометрической модели от чертежа, который представляет собой условное символьно-графическое изображение, предназначенное для чтения человеком.

1.3 Выбор среды разработки и языка программирования

Для того чтобы реализовать данную модель нам потребуется открытая библиотек OpenGL и формы, для более легкого создания и вывода полученных результатов и представления таковых в формате 3D.

В данной ситуации рассмотрим такие языки программирования как: C++, C#, Python. Так как эти языки идеально подходят к тем условиям, что были написаны выше. Рассмотрим каждый из них по отдельности и решим, какой из них лучше всего.

Благодаря своей гибкости и простоте, язык Python позволяет легко и быстро писать целый ряд приложений и утилит. Данный язык, в основном, используется для написания небольших скриптов, облегчающих выполнение различных задач, связанных с системным администрированием. Как оказалось, Python можно использовать и для не совсем «традиционных» задач, например, для вывода 3D графики.

Для работы с «непитоновскими» библиотеками (например, OpenGL) необходимы модули, обеспечивающие возможность вызова функций библиотеки непосредственно из программы на языке Python. Библиотека PyOpenGL – модуль, позволяющий в программах на языке Python легко работать с функциями OpenGL, GLU и GLUT, а также с рядом расширений OpenGL.

Но в языке Python есть трудности с формами, поэтому не будем рассматривать этот язык для написания выпускной квалификационной работы.

Язык C# также хорош, как и язык Python, большинство функций в нем, для упрощения написания кода, реализовано и также в VisualStudia уже реализованы формы, с помощью которых можно разработать интерфейс программы.

Заметим, что использование C++ не предполагает строгого объектно-ориентированного программирования, можно использовать его для многих парадигм. Поэтому, если закодируем свой движок на C++, все равно можно

использовать все существующие библиотеки стилей ООП, используя любую парадигму, которая нравится для ядра и написания кода.

Поэтому в данной программе мы будем использовать язык программирования C++, так как в нем уже есть встроенная библиотека OpenGL и реализованы формы для более просто реализации приложения.

1.4 Вывод по разделу

В данной главе была рассмотрена анатомия руки кисти человека. Были выделены основные части руки, такие как: ладонь, сустав, связки, пальцы и сосуды. Также рассмотрены современные протезы их строение и то, как они функционируют.

Разобрано создание графических моделей и выбран язык и среда разработки. В данной выпускной квалификационной работе программа будет написана на языке C++ в среде разработке Borland Turbo C++.

2 ПОСТРОЕНИЕ МАТЕМАТИЧЕСКОЙ 3D МОДЕЛИ КИСТИ РУКИ

Для создания математической модели необходимо решить несколько задач:

- разобрать структуру кисти руки;
- геометрическое моделирование кисти руки с использованием примитивов из набора;
- расчет динамики движения элементов модели кисти за счет сокращения связок.

Рассмотрим каждую задачу отдельно.

2.1 Структура кисти руки

После анатомии руки человека, можно выделить следующие элементы кисти руки:

– ладонь;

– сустав;

– пальцы;

– связки и сосуды.

Для построения модели не будем учитывать сосуды, а ладони представим в виде прямоугольного каркаса. В точках крепления пальцев к ладони установим сферы – элементы, соответсвующие суставам. Каждый палец разобьем на фаланги. Также, соединим фаланги пальцев с помощью сфер. Примерная схема структуры кисти представлена на рисунке 2.1.

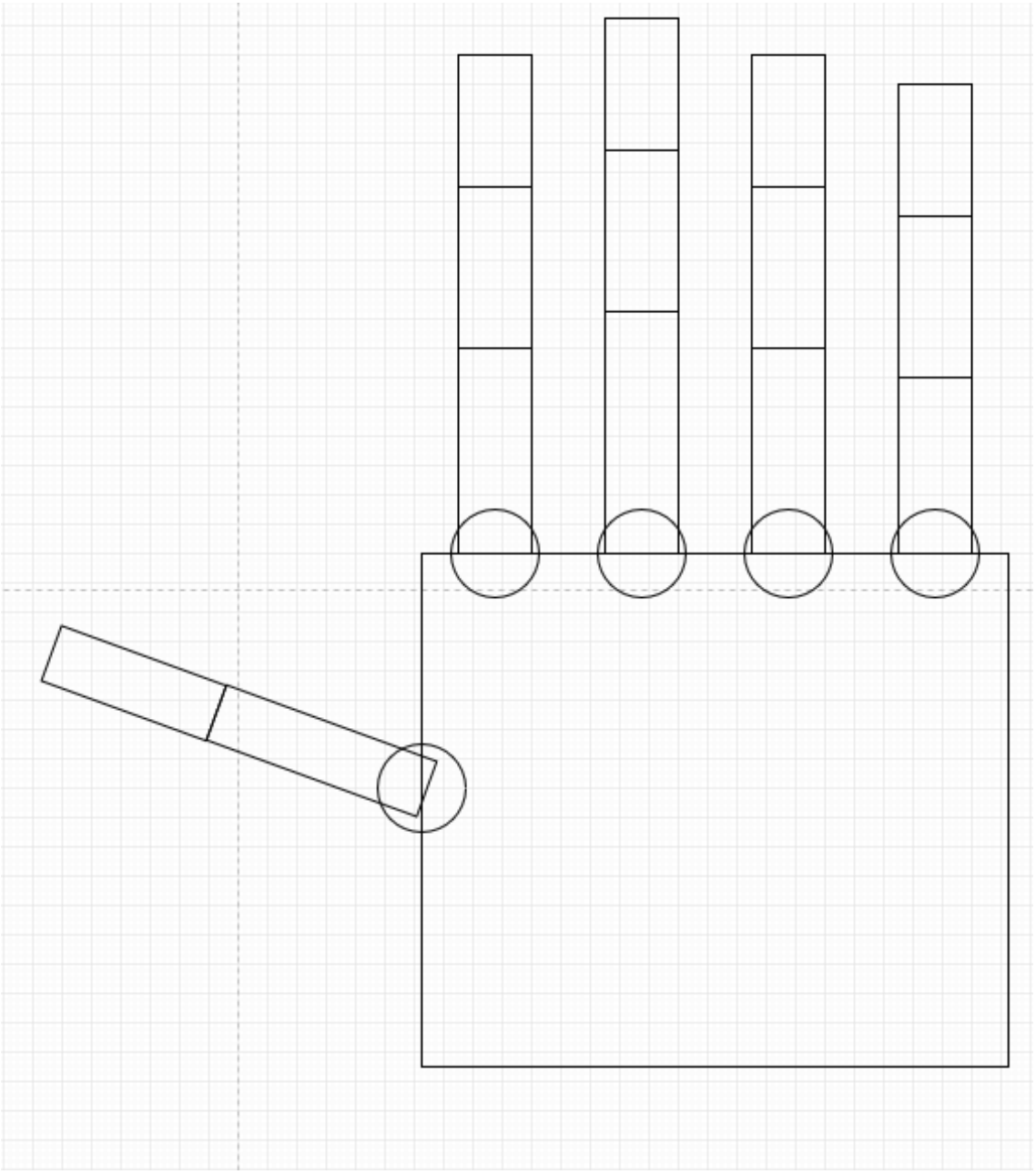


Рисунок 2.1 – Примерная структура кисти руки

Рассчитаем параметры ладони. Для нее заданы начальные размеры ширины и длины. Введем следующие обозначения: *wl* – ширина, *hl* – длина ладони (рисунок 2.2).

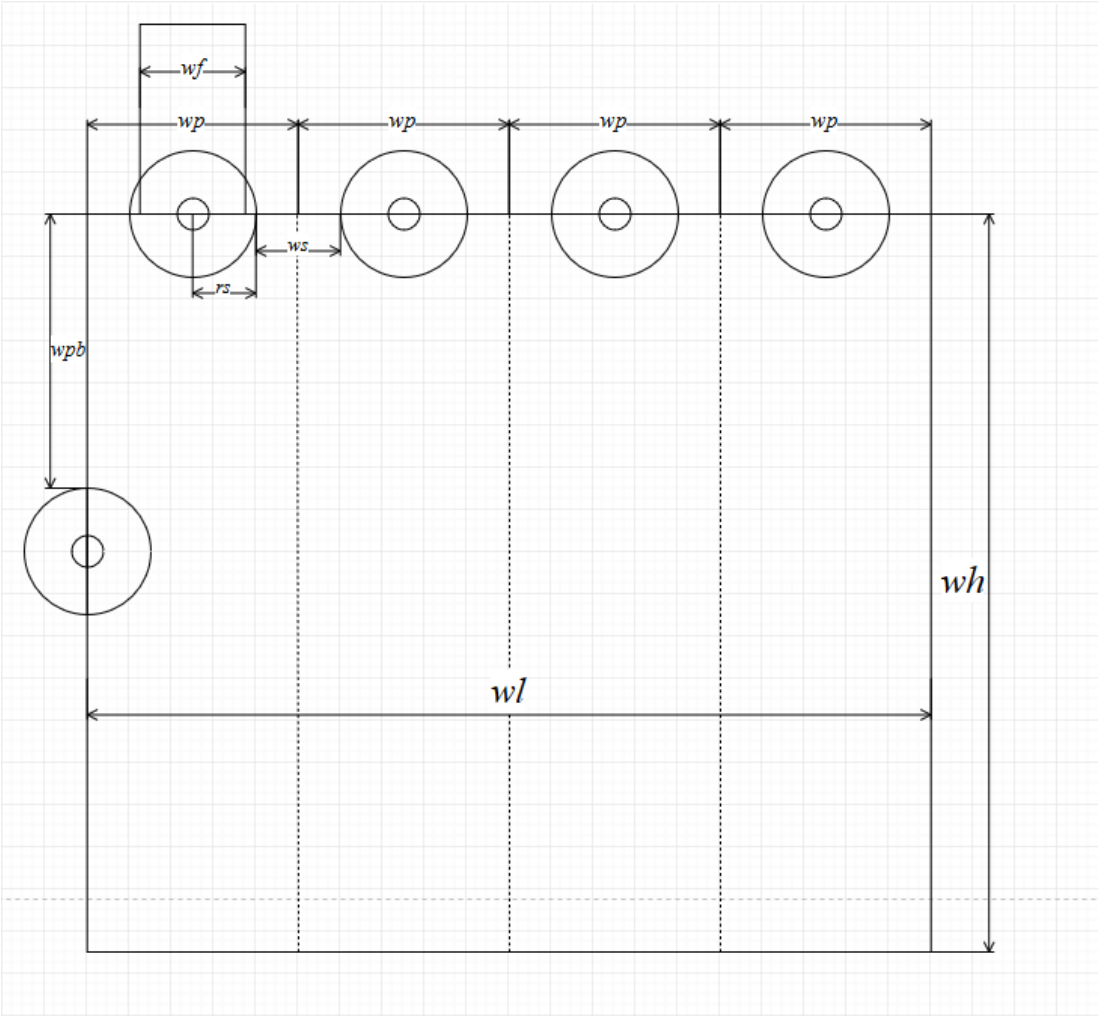


Рисунок 2.2 – Схема ладони

Необходимо определить положение суставов для крепления пальцев. Вычислим расстояние – ширину для крепления одного пальца (2.1).

— (2.1)

где *wp* – ширина для крепления пальца.

Также необходимо учесть движение самих суставов, поэтому зададим расстояние между суставами (2.2).

— (2.2)

где *ws* – расстояние между суставами.

Следовательно, радиус сферы для сустава можно рассчитать по формуле (2.3).

——— (2.3)

где *rs* – радиус сферы.

В каждый сустав кисти вставляется модель пальца, виде цилиндров. Для того, чтобы палец мог хорошо крепится к суставу, ширина фаланги будет рассчитываться по формуле (2.4).

— (2.4)

Пальцы руки состоят из трех фаланг. Для каждого пальца мы будем задавать вектор из трех параметров обозначающие длины фаланг (2.5).

–

– (2.5)

–

где l – длина соответствующего пальца;

hf – вектор длин фаланг пальца.

Также рассчитаем длину на которой должен находиться большой палец от указательного (2.6).

– (2.6)

где – длина соответствующего пальца;

– длина ладони.

Мы рассчитали все параметры, которые потребуются нам для построения модели руки человека.

Из анатомии руки известно, что пальцы двигаются за счет сокращения мышц. Направление движения фаланг пальцев и суставов представлено на рисунке 2.3 .

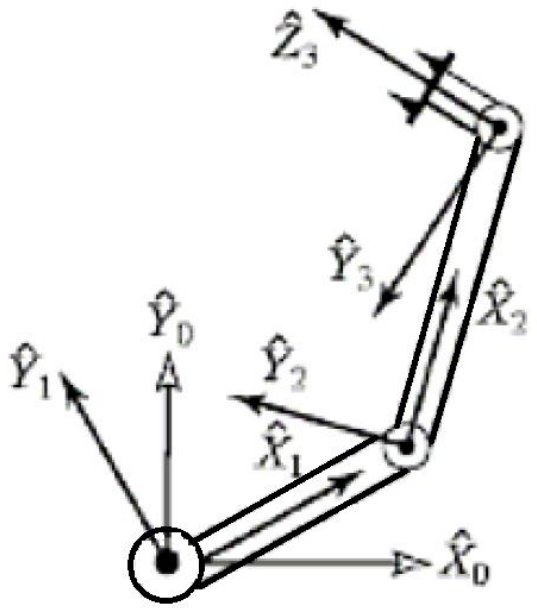


Рисунок 2.3 – Направление движения пальцев

Рассмотрим действие физических сил применительно к создаваемой модели пальцев руки (рисунок 2.4).

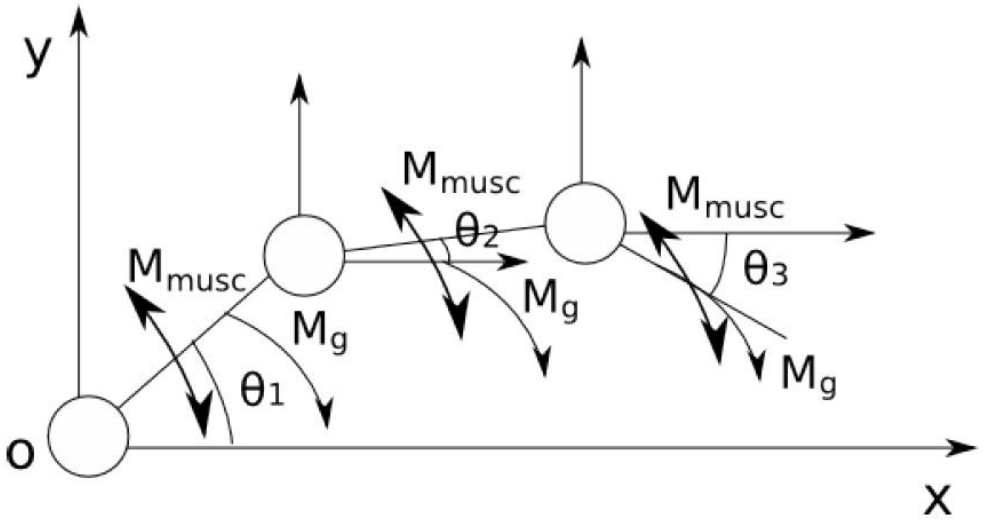


Рисунок 2.4 – Моменты сил, действующие на палец

Для расчета движения пальца руки используем систему дифференциальных уравнений Лагранжа второго рода (2.7).

(2.7)

где N – количество степеней свободы системы;

– обобщенные координаты;

– обобщенные скорости;

T – полная кинетическая энергия системы;

– обобщенные силы, действующие в системе.

В качестве обобщенных координат были выбраны углы в абсолютной системе координат: между фалангой и горизонталью, совпадающей с положением пальца в состоянии при котором ни одна из мышц сгибателей/разгибателей не претерпевает укорочения. Выбор абсолютной системы координат, обусловлен тем, что система (2.7) имеет в ней легкочитаемый и удобный для программирования на ЭВМ вид.

Преобразуем данную систему и получим, что левая часть системы уравнений (2.7) в матричной форме, для рассматриваемой системы, примет вид:

Обозначив:

– –

– –

– –

Запишем систему уравнений (2.6) в новом виде:

(2.8)

где , , – искомые функции управления.

Таким образом, подставляя в (2.8) законы движений для , возможно получить значения действующих в системе сил, то есть решить прямую задачу динамики.

Также, так как мы работаем с трехмерным объектом, потребуется матрица видовых преобразований, для изменения координат модели и ее перерисовки. Полученная модель будет поворачиваться по оси y, поэтому матрица (2.9) представлена в виде:

(2.9)

где угол α – угол поворота модели;

– матрица поворота по оси y.

С помощью данной матрицы, при перемножении ее на координаты точек модели происходит перерасчет координат и поворот модели на заданный угол α.

Определим места крепления связок к элементам модели кисти. Так как связки это основное, за счет чего двигаются пальцы, то рассмотрим рисунок 2.5. на данном рисунке показана схематичная модель пальца с прикрепленными к ней связками.

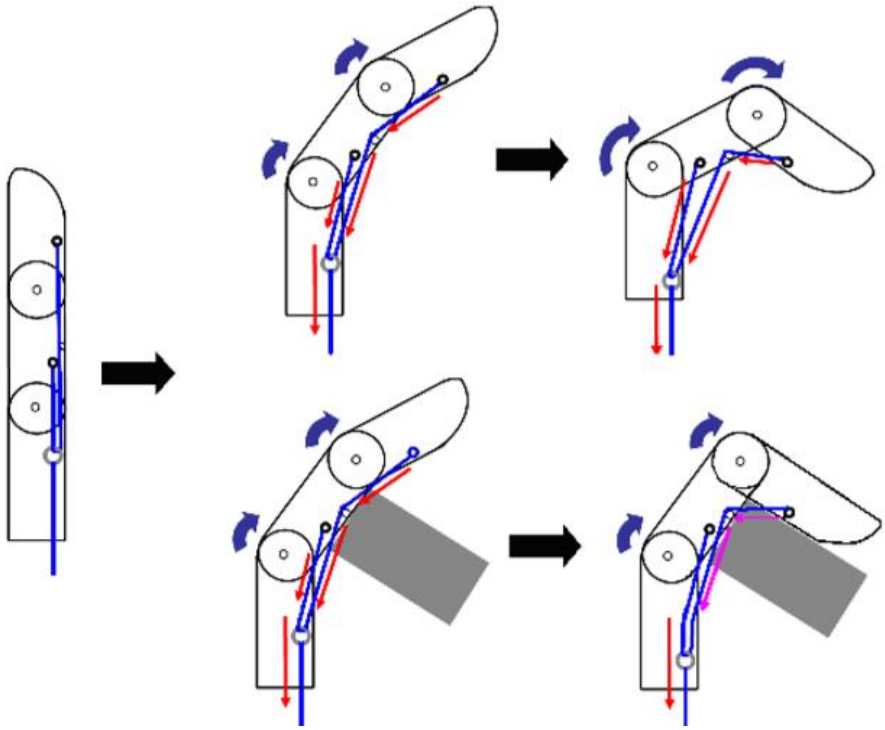


Рисунок 2.5 – Крепление связок

Как показано на изображении, лучше всего связки крепить внутри пальца и также пропускать их в центре через сустав, чтобы они не торчали снаружи. Внутри сустава будет расположен подвижный механизм, который будет помогать связкам продуктивнее, с меньшими затратами сил, сокращаться.

2.2 Вывод по разделу

В данном разделе были разобраны примитивы строения модели пальцев руки человека, построение и изменение 3D моделей, математическое движение пальцев руки человека, за счет приложенной силы к системе и крепление связок.

Мы вычислили матрицу видовых преобразований, которая нужна для поворота и изменения 3D модели руки. Также рассчитали систему уравнений зависимости силы внешней системы на связки пальцев руки человека.

За счет этого, мы смогли определить, куда лучше всего зацепить на модели пальца руки связки. Для того чтобы данная модель выдавала наибольшую продуктивность.

3 РАЗРАБОТКА ПРОГРАММЫ ВИЗУАЛИЗАЦИИ МОДЕЛИ КИСТИ РУКИ ЧЕЛОВЕКА

Приложение, разрабатываемое под операционную систему Windows, должно строиться по основному принципу: после инициализации приложения и создания окон должен запускаться цикл обработки сообщений (Рисунок 3.1).

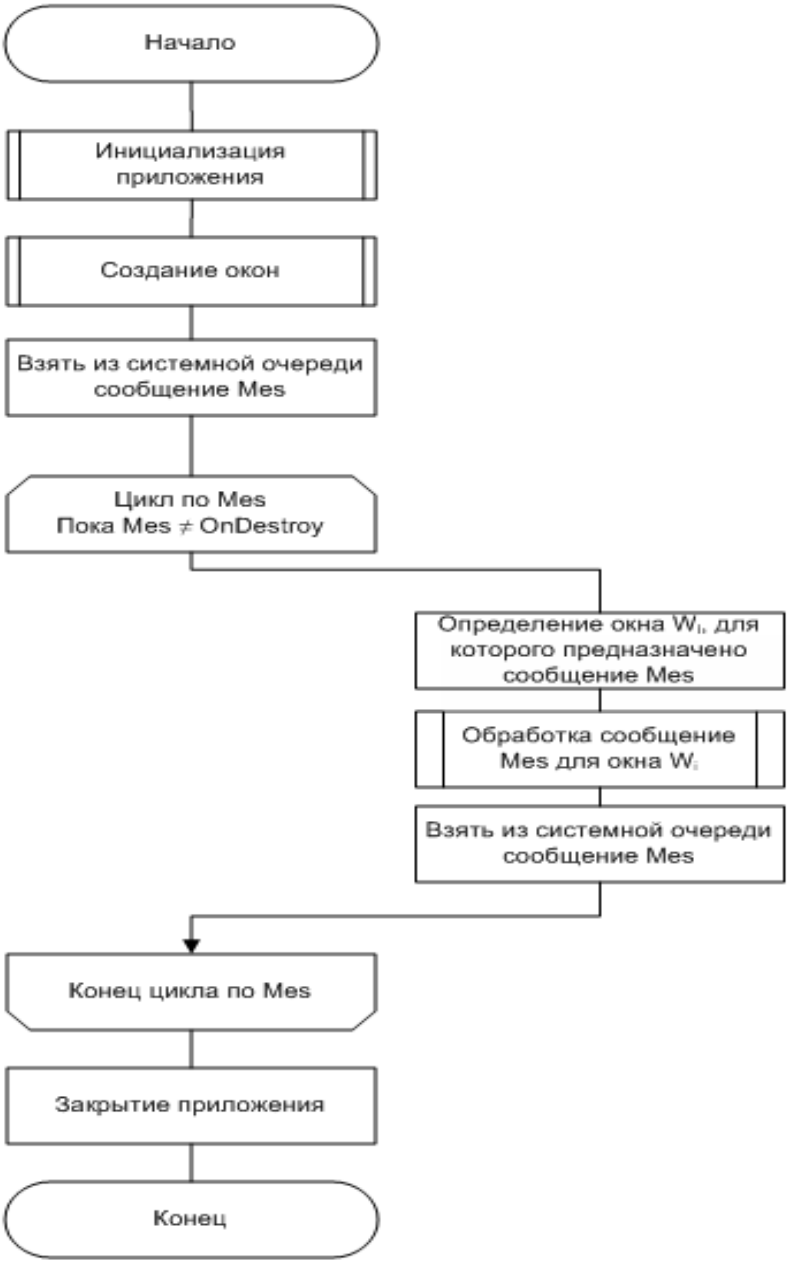


Рисунок 3.1 – Схема основного алгоритма

При получении сообщения из системной очереди необходимо определить событие, соответствующее данному сообщению. Все события можно привязать к трем группам:

– выбор пользователем пункта из меню «Файл» или «Модель»;

– выбор пункта меню «Справка»;

– нажатие элементов управления моделью.

Для каждого события вызывается соответствующий обработчик. Схема вспомогательного алгоритма, в котором реализована обработка событий данной программы представлена на рисунке 3.2.

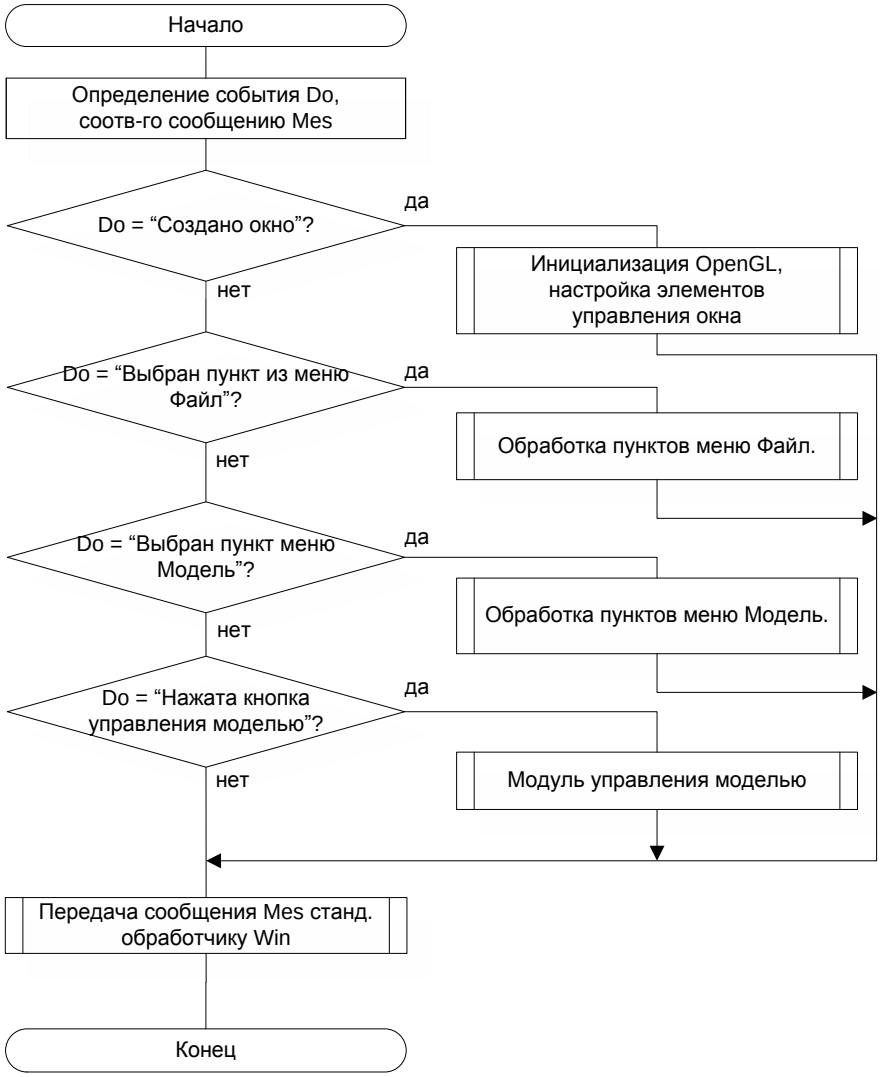


Рисунок 3.2 – Схема вспомогательного алгоритма «Обработка событий»

При «Создании окна» происходит инициализация библиотеки OpenGL , а также настройка элементов управления окна. При выборе пункта «Файл» осуществляется работа с файлами. С помощью стандартных окон происходит как загрузка так и сохранение файлов в формате .txt с параметрами модели. Определено событие «Выбран пункт меню модель» происходит обработка пунктов меню Модель (рисунок 3.3). При выборе данного пункта происходит расчет модели на основе заданных параметров. Если модель построена успешно то происходит визуализация модели на окне, если нет, то выдается сообщение об ошибке построения.

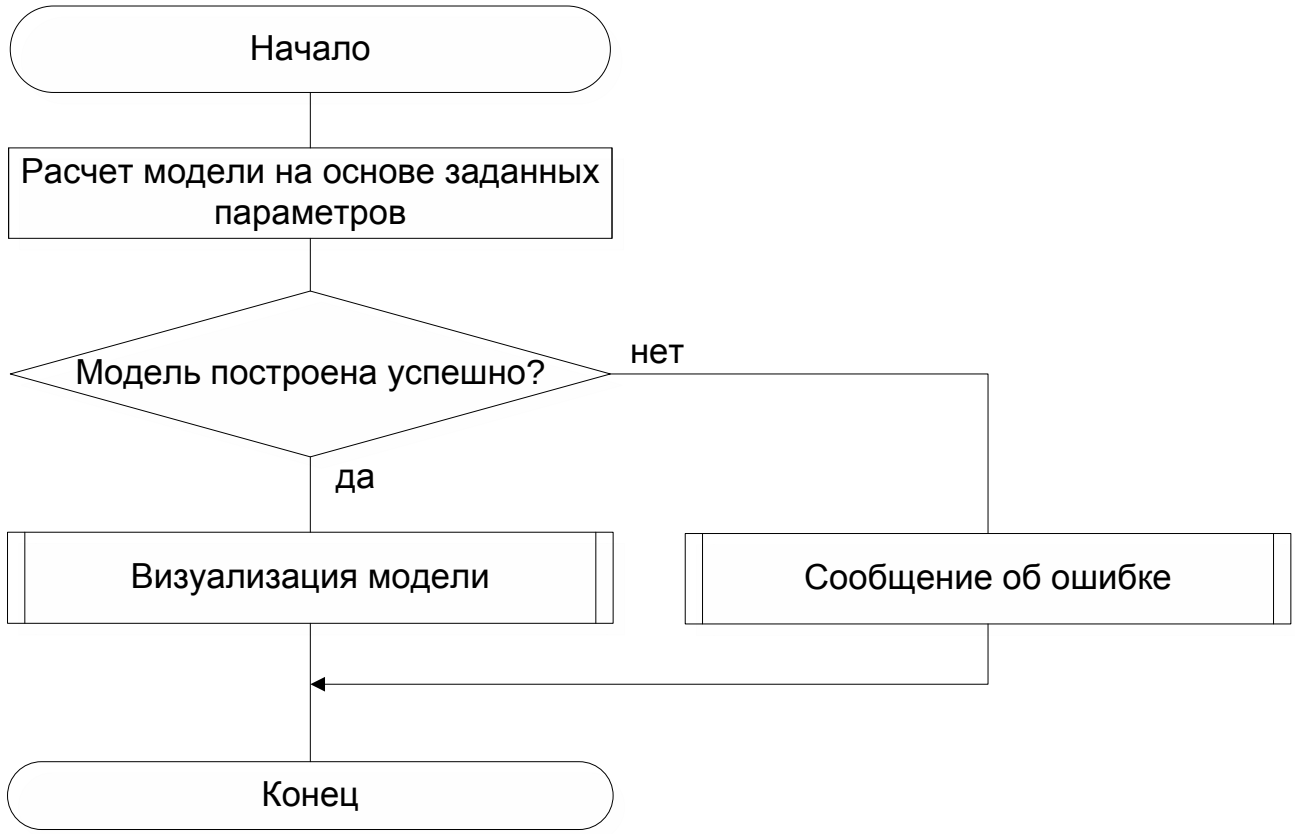


Рисунок 3.3 – Схема вспомогательного алгоритма
«Обработка пунктов меню Модель»

При нажатии кнопки управления модели происходит инициализация параметров модели, поворот модели и прорисовка самой модели кисти руки.

Также имеются клавиши «Инструкция» и «О программе» на которых при нажатии на экран выводится MessageBox с информацией о том, как пользоваться

программой либо о самой программе. При нажатии на клавишу «Выход» происходит завершение приложения.

Рассмотрим алгоритм построения модели пальца руки (рисунок 3.4).

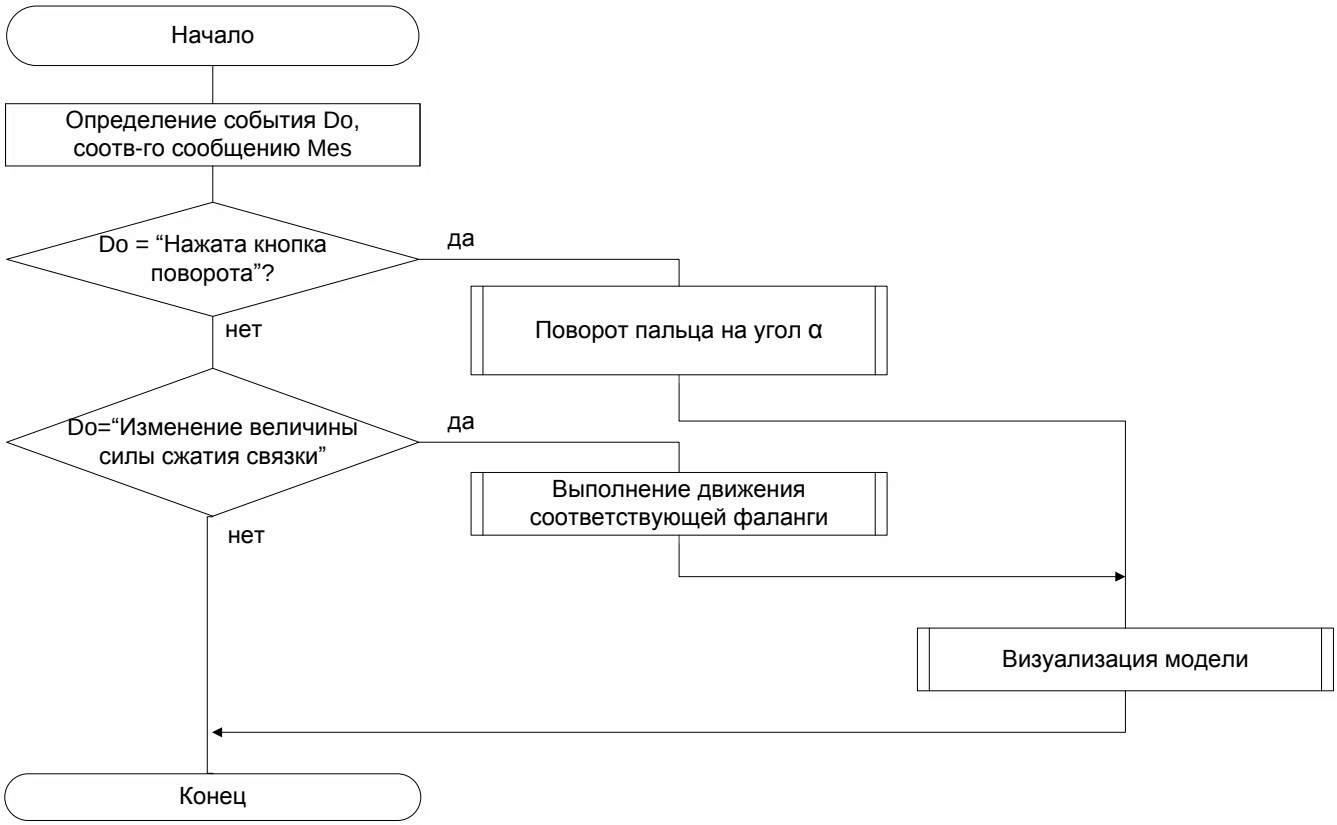


Рисунок 3.4 – Схема вспомогательного алгоритма «Обработка управления модели пальца»

Так как работа с пальцами введется в новом окне, то проверяется посылка сообщений на данную форму. Если нажата кнопка поворота модели, то происходит поворот пальца, на заданный угол α и визуализация данной модели. При изменении величины сжатия связки осуществляется выполнение движения соответствующей фаланги за счет силы мышц и в конце также осуществляется визуализация модели.

Модель начинает строиться с пальцев рук, фаланги пальцев рассчитываются исходя из «золотого сечения», после формируется сама кисть за счет введенных параметров длины и ширины.

Для того, чтобы понять, почему прорисовка происходит именно так, обратимся к объектно-ориентированному анализу построения модели кисти руки.

3.1 Объектно-ориентированный анализ для построения модели кисти руки

Рассмотрим UML диаграмму объектно-ориентированного анализа для построения модели кисти руки человека (Рисунок 3.4).

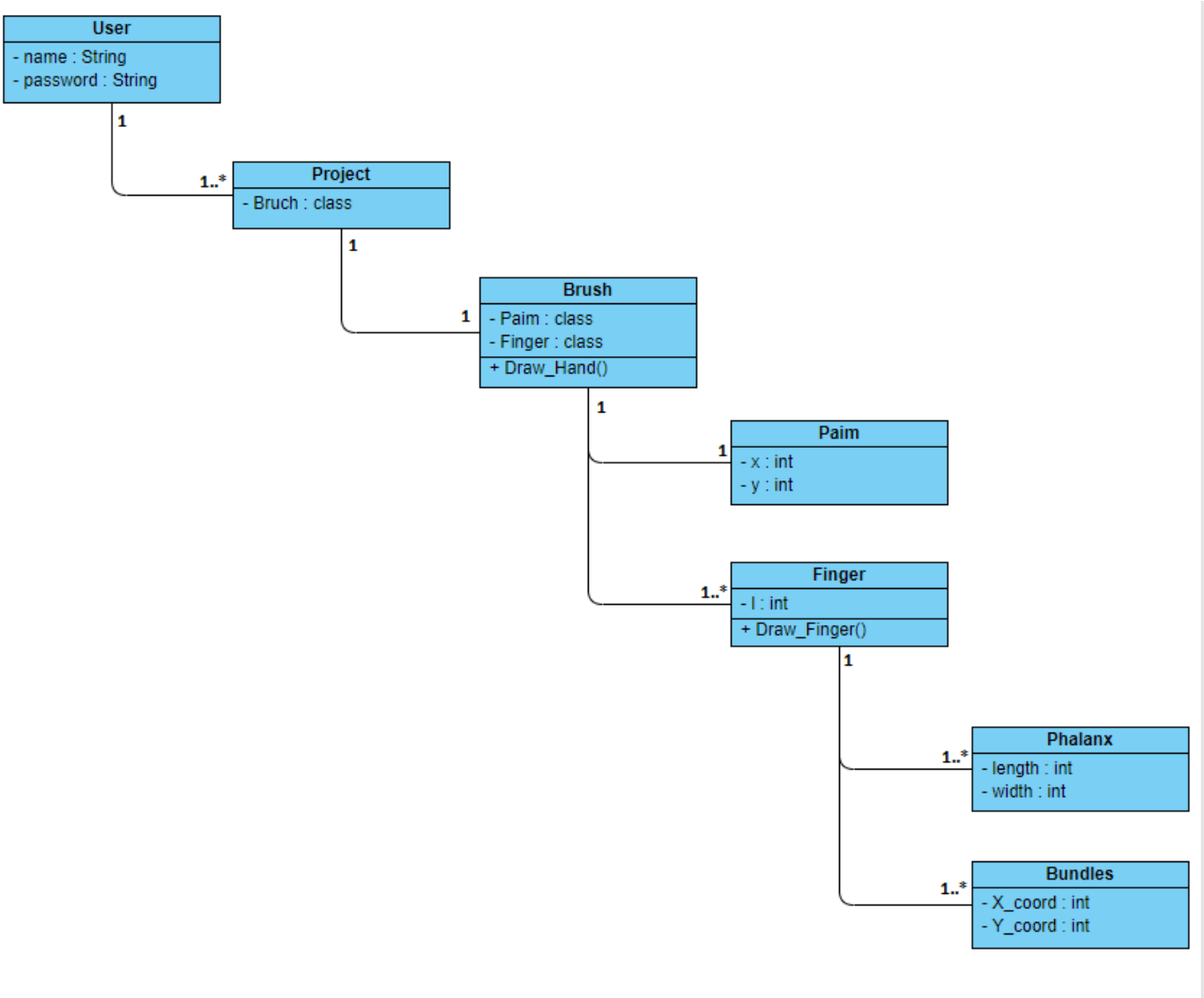


Рисунок 3.5 – UML диаграмма объектно-ориентированного анализа для построения модели кисти руки человека

Пользователь может создать любое количество проектов, в каждом проекте создается лишь одна модель руки. Модель руки состоит из двух элементов: из

одной ладони и множества пальцев. А палец, в свою очередь, состоит из множества фаланг и связок.

Рассмотрим каждый элемент подробнее. Фаланга, в нашем случае фаланга представляется виде цилиндра, имеет свою длину и ширину, так как фаланги пальцев человека имеют различные данные параметры. Связки имеют параметры координат по осям x и y. Так как связки связывают фаланги пальцев между собой, то эти координаты еще и обозначают начало строение фаланги.

Пальцы, между тем, получают параметры длины. Данный параметр с помощью аффинных координат и «золотого сечения» преобразуются и посылаются для построения в функцию Draw_Finger(). В ладонь посылаются параметры координат x и y, с помощью которых строится каркасная модель.

В кисть входят как раз таки ладонь и пальцы. С помощью, которых строится графическая модель движения пальцев руки. Это реализовано в функции Draw_Hand(). Также каждая математическая модель кисти руки человека, есть один отдельный проект. А для каждого пользователя можно создавать какое угодно количество проектов.

Рассмотрев UML модель и схемы алгоритмов, рассмотрим разработку пользовательского интерфейса данной программы.

3.2 Разработка пользовательского интерфейса

После запуска программы появляется главное окно, содержащее следующие пункты: «Файл», «Модель», «Помощь» и клавиши поворота математической модели кисти руки (рисунок 3.5).

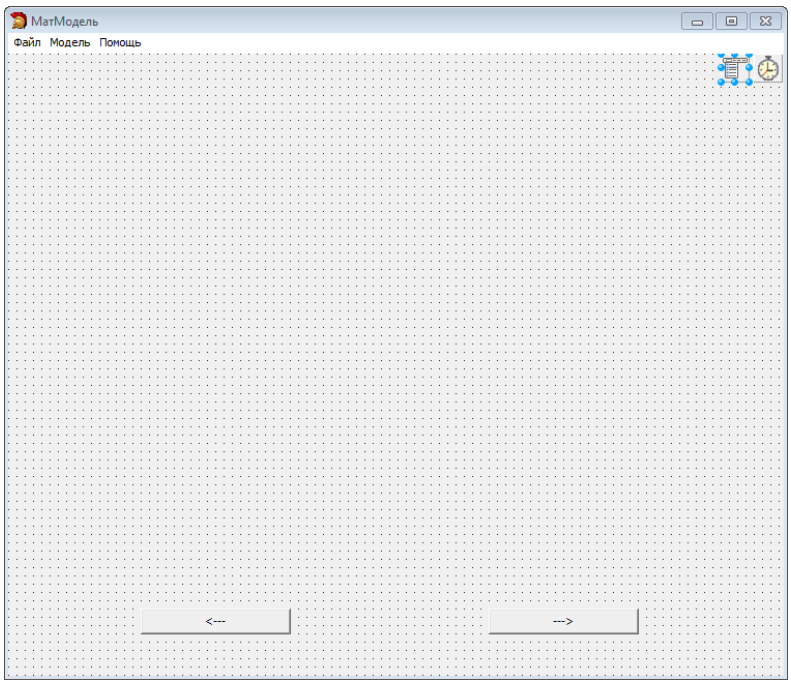


Рисунок 3.6 – Главное окно программы

При нажатии кнопку меню «Файл» выползает подменю, в котором находятся пункты: «Создать», «Сохранить», «Загрузить» и «Выход». При нажатии на пункт «Создать» создается трехмерная модель руки на форме. Когда нажимается пункт «Сохранить» появляется окно сохранения параметров модели (рисунок 3.7).

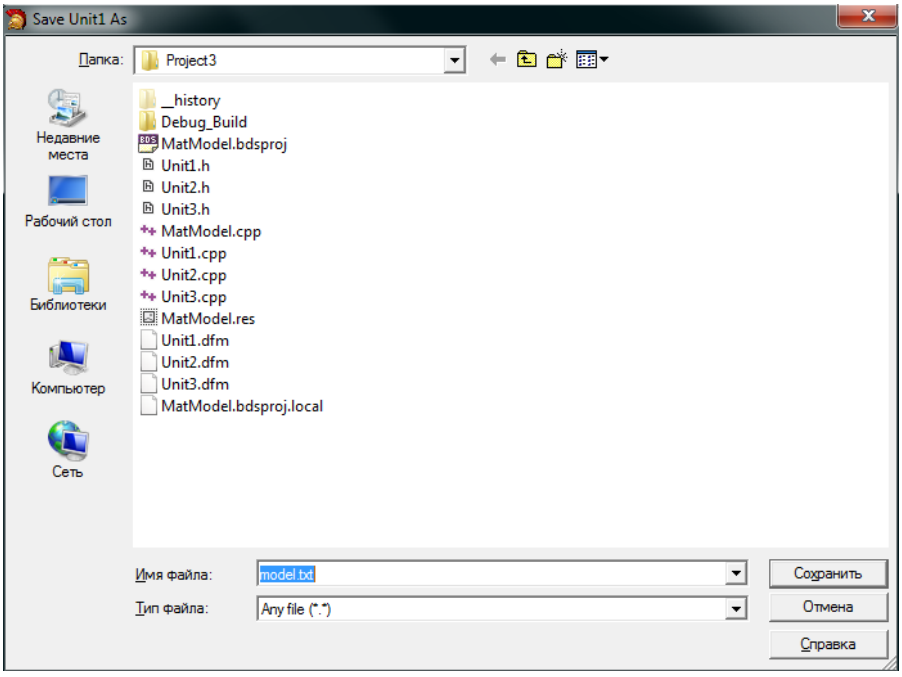


Рисунок 3.7 – Окно сохранения параметров модели

С помощью которого полученные параметры модели сохраняются в файл формата .txt. При выборе пункта меню «Загрузить» создается окно загрузки файла формата .txt (рисунок 3.8). С помощью которого строится 3D-модель кисти руки. При нажатии пункта «Выход» происходит завершение приложения.

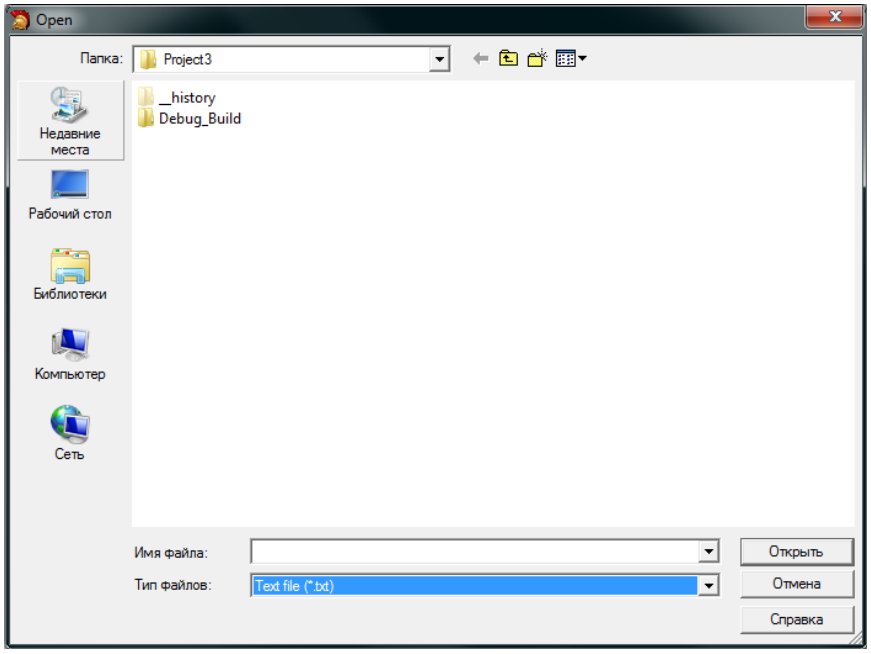


Рисунок 3.8 – Окно загрузки параметров модели

При выборе пункта меню «Модель» выходит подменю с выбором подпунктов: «Параметры» и выбор прорисовки пальцев. При выборе пункта «Параметры» создается окно ввода параметров кисти руки человека (рисунок 3.9).

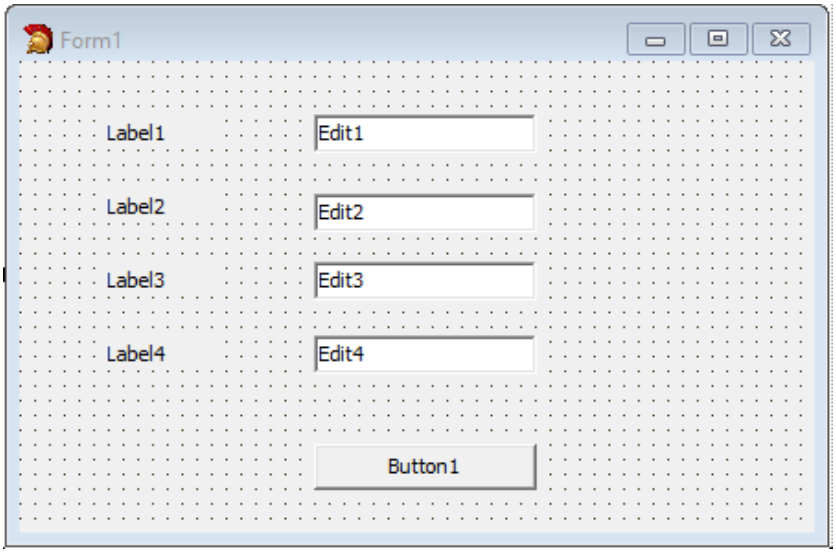


Рисунок 3.9 – Окно ввода параметров

В этом окне задаются такие параметры как: длина пальцев, длина и ширина кисти руки человека. После того как заполнили все параметры и нажата клавиша «Ок» на главном окне (рисунок 3.6) строится модель кисти руки человека с помощью цилиндров и сфер.

При выборе одного из пальцев создается новое окно, в котором строится модель подвижная модель пальца (рисунок 3.10).

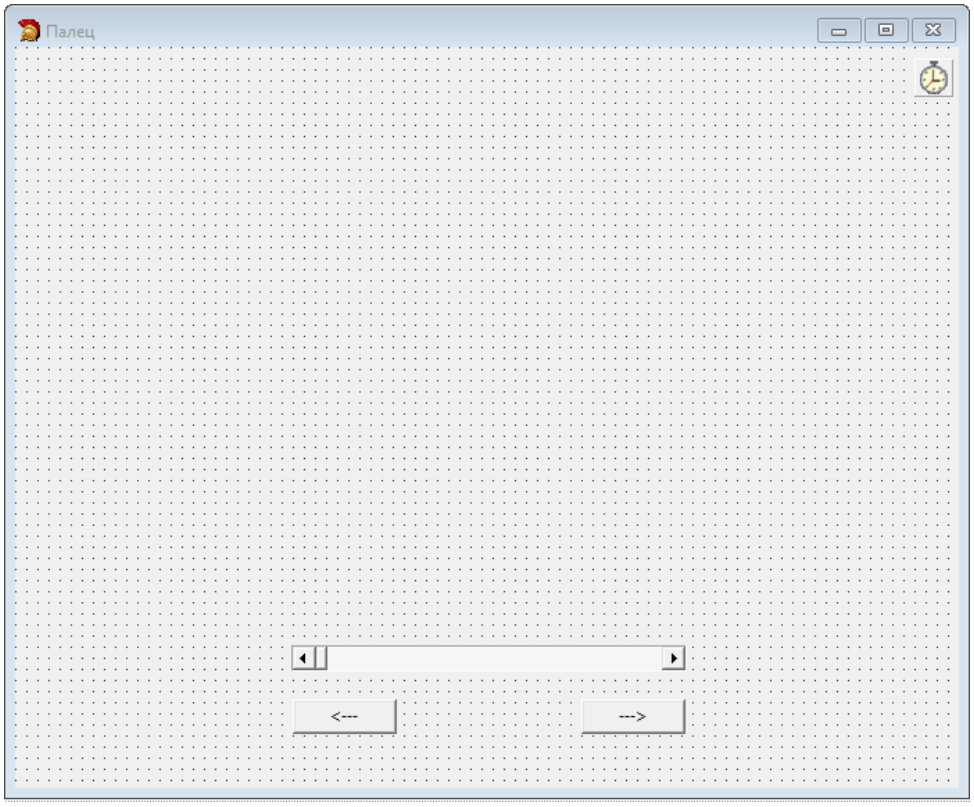


Рисунок 3.10 – Окно рисования модели пальца

С помощью скроллбара можно выбрать силу, с которой будет сгибаться палец. И также с помощью кнопок поворачивается модель вокруг своей оси. За счет этого можно посмотреть, как изгибается модель пальца.

При нажатии пункта меню «Помощь» появляется подменю, в котором указаны такие пункты как: «Инструкция» и «О программе». При выборе пункта «Инструкция» создается окно, в котором написано, как пользоваться данным приложением (рисунок 3.11). При нажатии кнопки «ОК» окно закрывается.

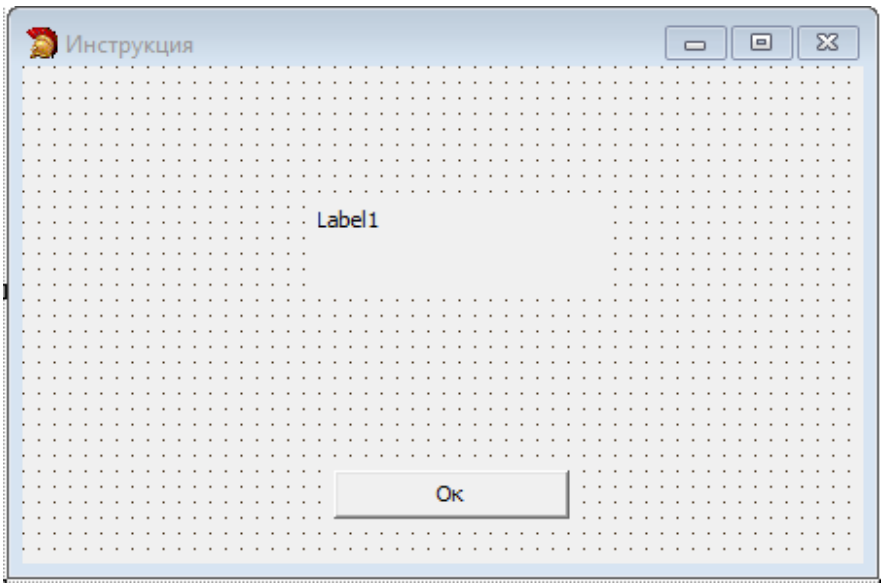


Рисунок 3.11 – Окно инструкции

При нажатии на кнопку «О программе» появляется окно, в котором можно увидеть информацию о программе и разработчике (рисунок 3.12). Также при нажатии на клавишу «ОК», окно закрывается.

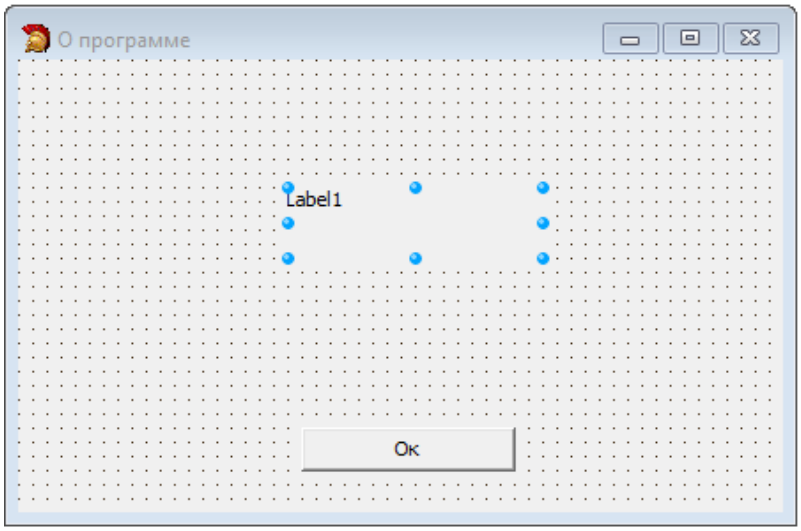
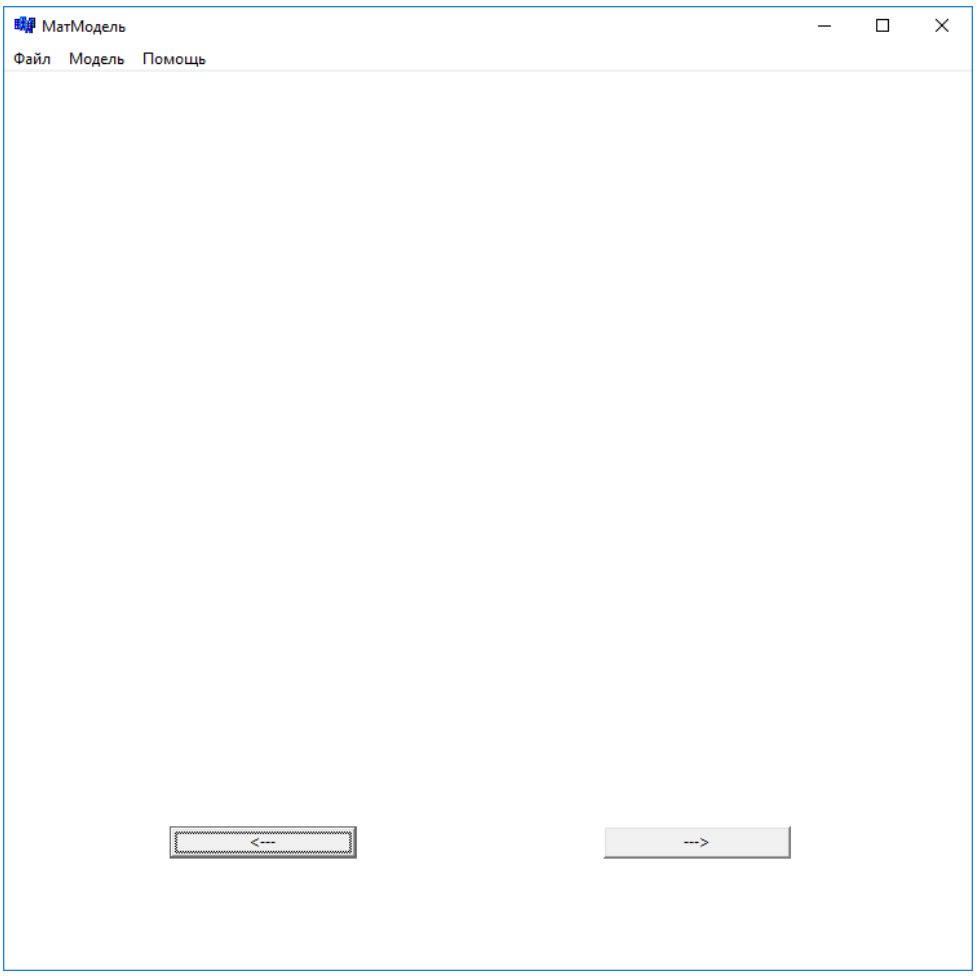


Рисунок 3.12 – Окно информации «О программе»

3.3 Проверка работы программы на экспериментальных данных

Для проверки работы программы необходимо запустить файл MathModel.exe. После запуска программы открывается главное окно, представленное на рисунке 3.13.



Для того, чтобы задать параметры модели выбираем пункт меню «Модель» и из подменю выбираем «Параметры». Открывается диалоговое окно (рисунок 3.14), в котором вводим следующие значения:

– мизинец – 69мм;

– безымянный – 85мм;

– средний – 92мм;

– указательный – 90мм;

– большой – 75мм;

– ширина ладони – 90мм;

– длина ладони – 83мм.

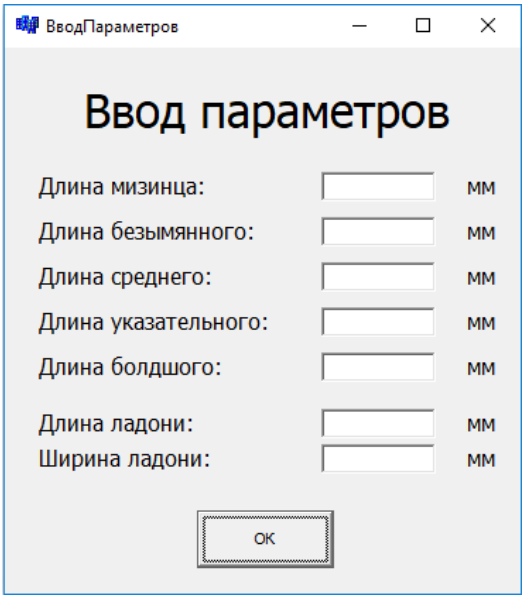


Рисунок 3.14 – Окно ввода параметров

Нажмём кнопку «ОК». С помощью введенных значений параметров строится модель руки (рисунок 3.15).

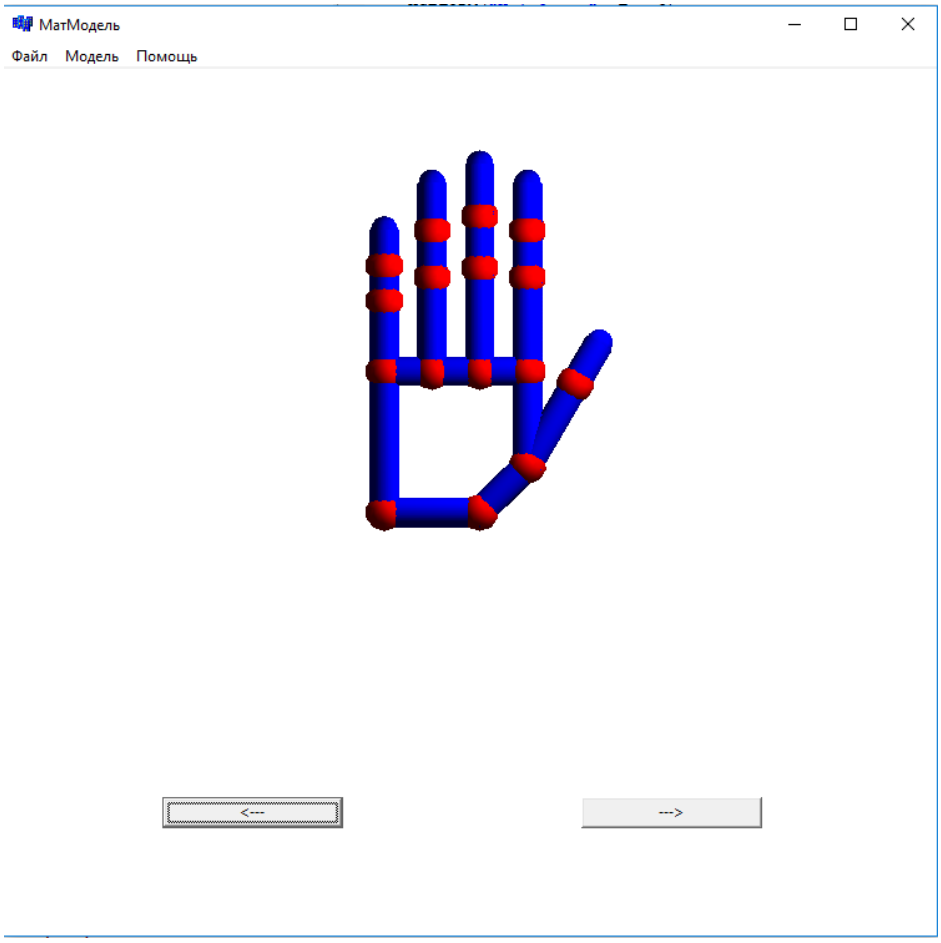


Рисунок 3.15 – Построенная модель руки

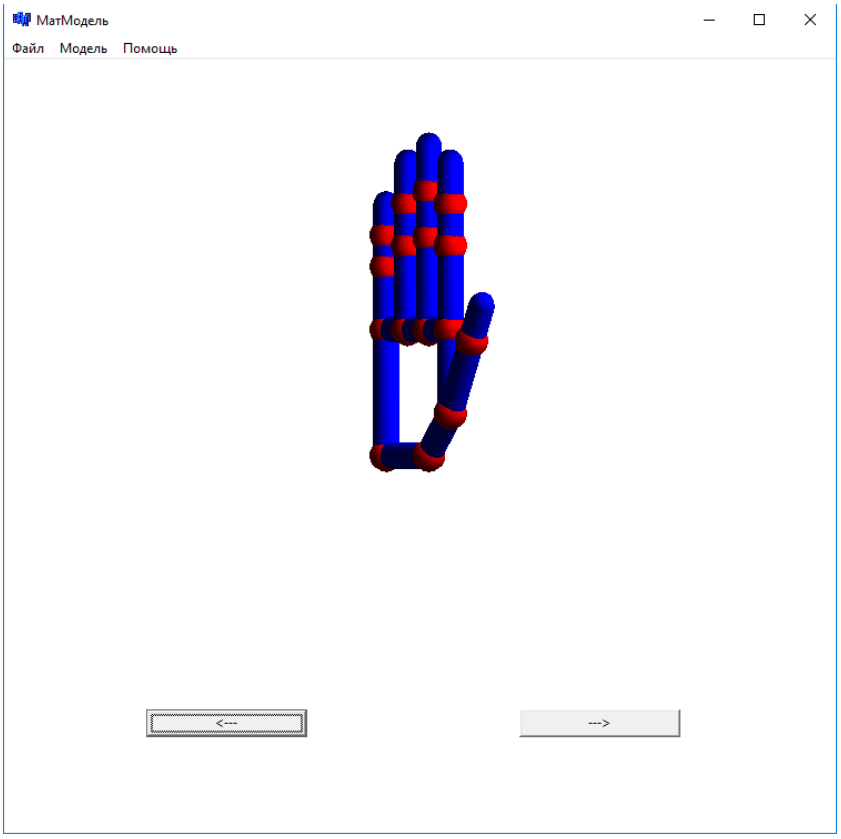


Рисунок 3.16 – Работа программы

Далее, с помощью кнопок управления поворачиваем модель, чтобы рассмотреть ее с разных сторон (рисунок 3.16).

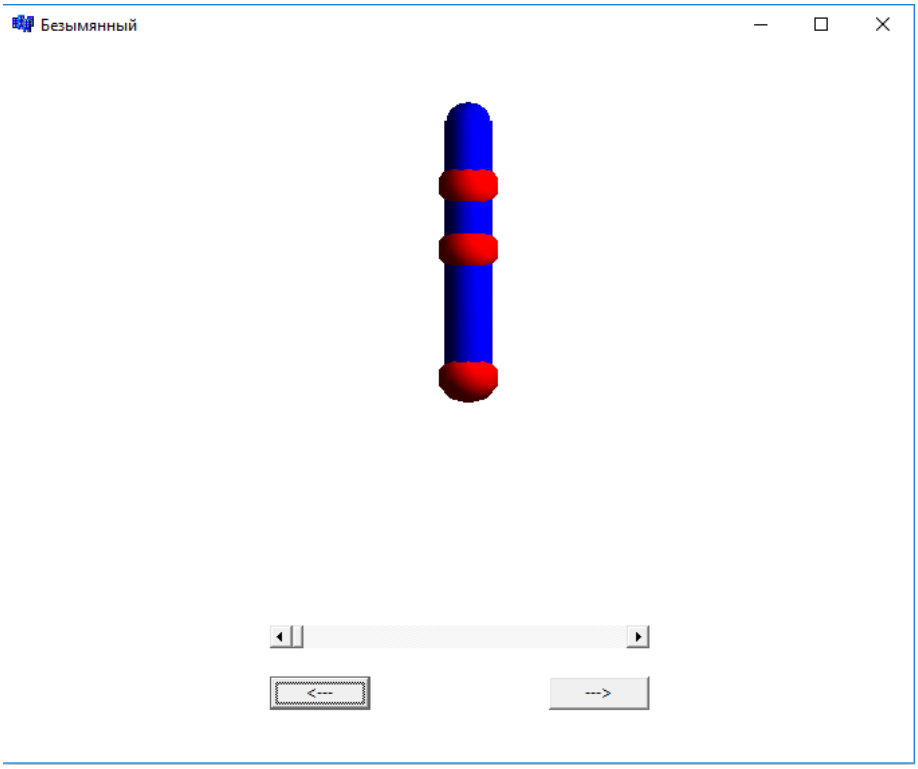


Рисунок 3.17 – Модель безымянного пальца

Также с помощью пункта меню «Файл» можно загрузить параметры модели, для этого нужно выбрать подпункт меню «Загрузка», либо сохранить имеющиеся параметры, нажав на «Сохранить». При нажатии на клавишу «Создать» создается трехмерная модель.

Реализовано и отдельное рассмотрение каждого пальца модели кисти. Для просмотра модели пальцев в меню выбираем пункт: «Модель» и выбираем, ниже пункта «Параметры» тот палец, который мы хотим рассмотреть. После этого появляется окно (рисунок 3.17), на котором рисуется выбранный палец.

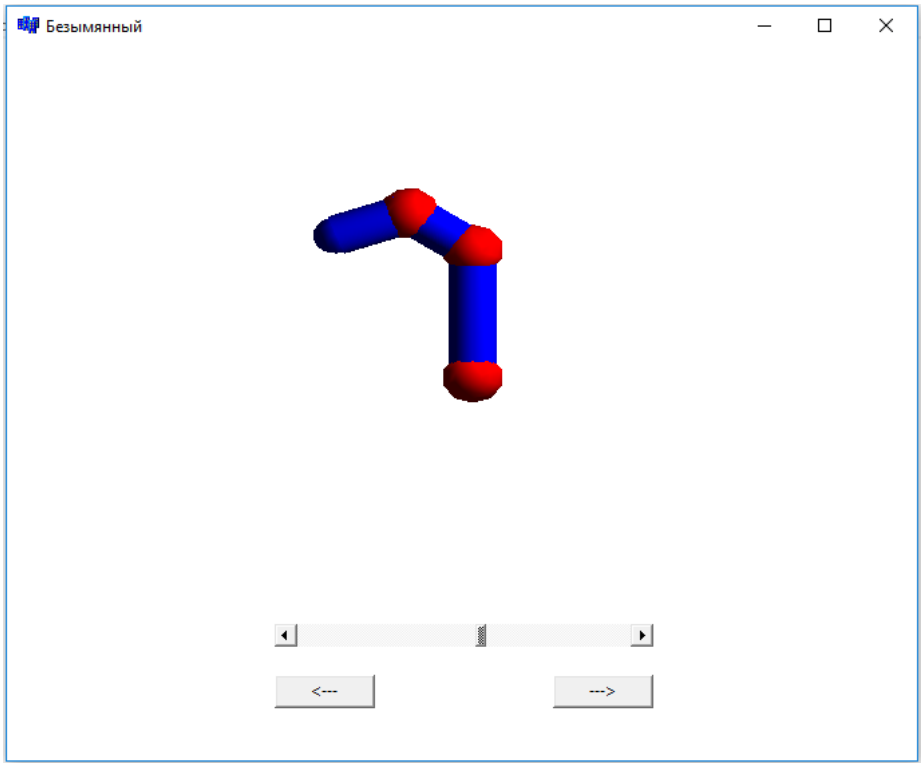


Рисунок 3.18 – Работа модели безымянного пальца

С помощью скроллбара можно задать любое значение силы благодаря которой палец будет сгибаться и разгибаться. Также есть кнопки поворота модели пальца руки, с их помощью можно под любым углом посмотреть, как себя ведет модель пальца при разных значениях силы (рисунок 3.18).

При выборе, в основном окне, пункта меню «Помощь» выйдет подменю в котором есть следующие клавиши: «Инструкция» и «О программе». При нажатии пункта «Инструкция» появляется окно (рисунок 3.19), в котором находится информация, как пользоваться программой.

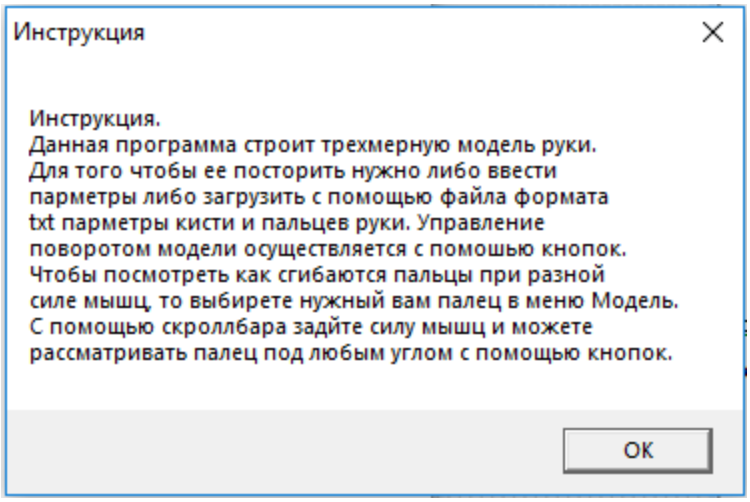


Рисунок 3.19 – Окно «Инструкция»

Работают и функционируют клавиши «О_программе» и «Инструкция» с помощью которых пользователь узнает о том, как пользоваться программой и узнает информацию о разработчике приложения.

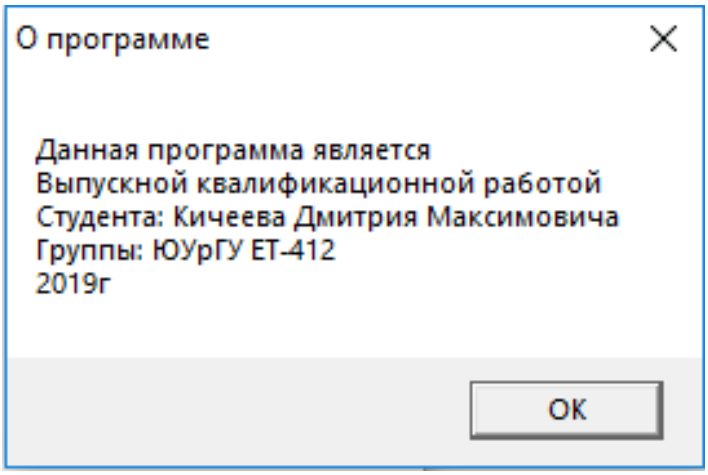


Рисунок 3.20 – Окно «О программе»

3.4 Вывод по разделу

В данном разделе были представлены алгоритмы и диаграммы, связанные с программой, разработан интерфейс программы и описаны его функциональные возможности. Также были представлены результаты тестирования программы на экспериментальных данных.

Анализируя данные результаты, можно сделать вывод, что разработанная модель функционирует корректно. Все цели и задачи выполнены.

ЗАКЛЮЧЕНИЕ

Данная работа посвящена разработке математической модели движения пальцев руки человека.

В результате работы были решены следующие задачи:

1) изучены и проанализирована анатомия кисти руки человека, и также рассмотрены различные строения современных биопротезов;

2) разработана математическая модель движения пальцев руки человека;

3) выполнена компьютерная программа, реализующая данную математическую модель;

4) реализован понятный для пользователя функционирующий интерфейс;

5) осуществлена проверка работа программы на экспериментальных данных.

Разработанная модель имеет ряд преимуществ: разработана визуализация модели кисти и пальцев руки человека, возможность рассмотреть зависимость силы от сгибания/разгибания пальца, а также взглянуть на кисть и пальцы под разным углом обзора.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Гринягин И.В. Компьютерная диагностика активности пальцев руки человека на основе биомеханического моделирования: автореф. дис. канд. мед.наук: 03.01.09. М., 2011.

2 Синельников Р. Д., Синельников Я. Р., Синельников А. Я. Атлас анатомии человека. Том первый. М. : Новая волна, 2009. 345 с

3 Анатомия кисти руки – URL: https://www.oum.ru/literature/anatomiya-cheloveka/anatomiya-kisti/ (дата обращения: 05.02.2019)

4 Биомеханический анализ движений травмированной кисти (опыт клинического применения). / Е.В. Бирюкова, А.А. Фролов, Гринягин И.В. и др. //Российский медицинский журнал №2. - 2010. - С. 14-19.

5 Коновалова А.Н. Неврология и нейрохирургия: учебник: т. 1. / А.Н. Коновалова, А.В. Козлова, Е.И. Гусев, В.И. Скворцова. - М. : ГЭОТАР-Медиа, 2009. – 624 с.

6 Механические протезы [Электронный ресурс]. – Электрон. текстовые дан. – ФГУП "Калужское протезно-ортопедическое предприятие", 2014. – Режим доступа: http://ortopedica-kaluga.ru/protezyi-verhnih-konechnostey, свободный.

7 Протезы верхних конечностей [Электронный ресурс]. – Электрон. текстовые дан. – ФГУП "Калужское протезно-ортопедическое предприятие", 2014. – Режим доступа: http://ortopedica-kaluga.ru/protezyi-verhnih-konechnostey, свободный.

8 Кисть с микропроцессорной системой управления «Миотея» [Электронный ресурс]: научно-производственная фирма «Галатея». – Электрон. текстовые дан. – М.: НПФ. «Галатея», 2011. – Режим доступа: http://www.npfgalatea.ru/catalog/id/80, свободный.

9 Кужекин А.П. Конструкции протезно-ортопедических изделий. — М.: Легкая промышленность, 1984, с. 172

10 Dorador Gonzalez, J.M. (2004). ROBOTICA Y PROTESIS INTELIGENTES. Retrieved March 15th 2015 from: http://www.revista.unam.mx/vol.6/num1/art01/art01-2d.htm

11 Baker, Scheme, Englehart, K., Hutcinson, & Greger. (2010). Continuous Detection and Decoding of Dexterous Finger Flexions with Implantable MyoElectric Sensors. IEEE Transactions on Neural Systems and Rehabilitation Engineering.

12 Englehart, K., & Hudgins, B. (2003). A robust, real-time control scheme for multifunction myoelectric control. 50(7), 848-854.

13 Farry, K. A., & Walker, I. D. (1993). Myoelectric teleoperation of a complex robotic hand. Paper resented at the IEEE International Conference on Robotics and Automation.

ПРИЛОЖЕНИЕ 1

Исходный текст программы

В файле Project1.cpp:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("Unit1.cpp", Form1);
USEFORM("Unit2.cpp", Form2);
USEFORM("Unit3.cpp", Form3);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
                Application->Initialize();
                Application->CreateForm(__classid(TForm1), &Form1);
                Application->CreateForm(__classid(TForm2), &Form2);
                Application->CreateForm(__classid(TForm3), &Form3);
                Application->Run();
        }
        catch (Exception &exception)
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
```

```
            try
            {
                throw Exception("");
            }
            catch (Exception &exception)
            {
                Application->ShowException(&exception);
            }
        }
        return 0;
}
//---------------------------------------------------------------------------
```

В файле Unit1.cpp:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
bool TForm1::bSetupPixelFormat(HDC hDC)
{
  PIXELFORMATDESCRIPTOR pfd; //Создаем структуру
  int pixelformat;
  pfd.nSize = sizeof (PIXELFORMATDESCRIPTOR); //Размер структуры
  pfd.nVersion = 1;  //Версия структуры
  pfd.dwFlags = PFD_DOUBLEBUFFER | PFD_SUPPORT_OPENGL |
PFD_DRAW_TO_WINDOW;
  pfd.iLayerType =  PFD_MAIN_PLANE;  //Тип поверхности
  pfd.iPixelType = PFD_TYPE_RGBA;  //Формат указания цвета
  pfd.cColorBits = 16;  //Глубина цвета
  pfd.cDepthBits = 16;  //Размер буфера глубины
  pfd.cAccumBits = 0; //Общее число битовых плоскостей в буфере аккумулятора
  pfd.cStencilBits = 0;  //Размер буфера трафарета
  if (!(pixelformat=ChoosePixelFormat(hDC, &pfd))) {
        MessageBox(NULL, "Невозможно выбрать формат пикселей", "Error",
MB_OK);
        return false;
  }
  if (!SetPixelFormat (hDC, pixelformat, &pfd)) {
        MessageBox(NULL, "Невозможно установить формат пикселей", "Error",
MB_OK);
        return false;
  }
  return true;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
```

```
  hDC = GetDC(Handle); //Handle – дескриптор окна (hwnd в WinAPI)
if (!bSetupPixelFormat(hDC)) //Устанавливаем формат пикселей
  return;
ghRC = wglCreateContext(hDC); //Создаем контекст воспроизведения
wglMakeCurrent(hDC, ghRC);  //Делаем его текущим
glClearColor(1.0, 1.0, 1.0, 1.0); //Цвет экрана при очищении
glEnable(GL_COLOR_MATERIAL);  //Разрешаем задавать цвет объектам
glEnable(GL_DEPTH_TEST);  //Тест глубины для объемности изображения
glEnable(GL_LIGHTING);  //Разрешаем освещение
glEnable(GL_LIGHT0);  //Включили освещение 0
float p[4]={3,3,3,1},
                d[3]={-1,-1,-3};
glLightfv(GL_LIGHT0,GL_POSITION,p);  //Установка позиции освещения
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormDestroy(TObject *Sender)
{
  if(ghRC)
{
  wglMakeCurrent(hDC,0);
  wglDeleteContext(ghRC);
}
if(hDC) ReleaseDC(Handle, hDC);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormResize(TObject *Sender)
{
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);
```

```
  glLoadIdentity();
  glOrtho(-5, 5, -5, 5, 2, 12);
  gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);
  glMatrixMode(GL_MODELVIEW);
}
//---------------------------------------------------------------------------
 void TForm1::Draw_Hand()
{
  GLUquadricObj *quadObj;
  GLfloat rquad=90;
  quadObj = gluNewQuadric();
  glClear(GL_DEPTH_BUFFER_BIT |  GL_COLOR_BUFFER_BIT);
  glPushMatrix();
  Finger(quadObj,1.5, -1, 1, false);
  glRotated (90, 0, 1, 0);
  gluQuadricDrawStyle(quadObj, GLU_LINE);
  gluCylinder(quadObj, 0.15, 0.15, 0.5, 100, 100);
  glRotated (90, 0, -1, 0);
  Finger(quadObj,2, 0.5, 0, false);
  glRotated (90, 0, 1, 0);
  gluQuadricDrawStyle(quadObj, GLU_LINE);
  gluCylinder(quadObj, 0.15, 0.15, 0.5, 100, 100);
  glRotated (90, 0, -1, 0);
  Finger(quadObj,2.2, 0.5, 0, false);
  glRotated (90, 0, 1, 0);
  gluQuadricDrawStyle(quadObj, GLU_LINE);
  gluCylinder(quadObj, 0.15, 0.15, 0.5, 100, 100);
  glRotated (90, 0, -1, 0);
  Finger(quadObj,2, 0.5, 0, false);
```

```
glRotated (90, 1, 0, 0);
gluQuadricDrawStyle(quadObj, GLU_LINE);
gluCylinder(quadObj, 0.15, 0.15, 1, 100, 100);
glRotated (90, -1, 0, 0);
Finger(quadObj,2, 0, -1, true);

glRotated (90, 1, -1, 0);
gluQuadricDrawStyle(quadObj, GLU_LINE);
gluCylinder(quadObj, 0.15, 0.15, sqrt(0.5), 100, 100);
glRotated (90, -1, 1, 0);

glTranslated(-0.5, -0.5, 0);
gluQuadricDrawStyle(quadObj, GLU_FILL);
glColor3f(1,0,0);
gluSphere(quadObj, 0.2,10,10);

glRotated (90, 0, -1, 0);
gluQuadricDrawStyle(quadObj, GLU_LINE);
glColor3f(0,0,1);
gluCylinder(quadObj, 0.15, 0.15, 1, 100, 100);
glRotated (90, 0, 1, 0);

glTranslated(-1, 0, 0);
gluQuadricDrawStyle(quadObj, GLU_FILL);
glColor3f(1,0,0);
gluSphere(quadObj, 0.2,10,10);

glRotated (90, -1, 0, 0);
gluQuadricDrawStyle(quadObj, GLU_LINE);
```

```
  glColor3f(0,0,1);
  gluCylinder(quadObj, 0.15, 0.15, 1.5, 100, 100);//*/
  glRotated (90, 1, 0, 0);

  glTranslated(1,0.5,0);
  gluDeleteQuadric(quadObj);

  SwapBuffers(hDC);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{
   //Form1->Draw_Hand();
}
//---------------------------------------------------------------------------
void   TForm1::Finger(GLUquadricObj *quadObj,double l, double x, double y, bool b)
{
//рисование связок пальцев
        gluQuadricDrawStyle(quadObj, GLU_FILL);
        glTranslated(x, y, 0);
        if(b==true)
                glRotated (30, 0, 0, -1);
        glColor3f(1,0,0);
        gluSphere(quadObj, 0.2,10,10);
        glTranslated(0, l/2, 0);
        gluSphere(quadObj, 0.2,10,10);
        glTranslated(0, l/4, 0);
        if(b==false)
        {
```

```
            gluSphere(quadObj, 0.2,10,10);
            glTranslated(0, l/4, 0);
        }
        glRotated (90, 1, 0, 0);
        glColor3f(0,0,1);
        gluSphere(quadObj, 0.15,10,10);
        gluQuadricDrawStyle(quadObj, GLU_LINE);
//рисование фаланг пальцев
        if(b==false)
        {
            gluCylinder(quadObj, 0.15, 0.15, l/4, 100, 100);
            glRotated (90, -1, 0, 0);
            glTranslated(0, -l/4, 0);
            glRotated (90, 1, 0, 0);
        }
        gluCylinder(quadObj, 0.15, 0.15, l/4, 100, 100);
        glRotated (90, -1, 0, 0);
        glTranslated(0, -l/4, 0);
        glRotated (90, 1, 0, 0);
        gluCylinder(quadObj, 0.15, 0.15, l/2, 100, 100);
        glRotated (90, -1, 0, 0);
        glTranslated(0, -l/2, 0);
        if(b==true)
            glRotated (30, 0, 0, 1);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  GLfloat rquad=30;
```

```
  glRotatef(rquad,0.0f,1.0f,0.0f);
  Form1->FormPaint(Sender);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
  GLfloat rquad=30;
  glRotatef(rquad,0.0f,-1.0f,0.0f);
  Form1->FormPaint(Sender);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
        Form1->Draw_Hand();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N1Click(TObject *Sender)
{
                Form2->Caption="Мизинец";
                Form2->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N2Click(TObject *Sender)
{
        Form2->Caption="Безымянный";
        Form2->Show();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N3Click(TObject *Sender)
{
        Form2->Caption="Средний";
        Form2->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N4Click(TObject *Sender)
{
        Form2->Caption="Указательный";
        Form2->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N5Click(TObject *Sender)
{
        Form2->Caption="Большой";
        Form2->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N9Click(TObject *Sender)
{
       Form1->Close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N10Click(TObject *Sender)
{
              Form3->Show();
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N12Click(TObject *Sender)
{
        Application->MessageBox("Инструкция.\n"
        "Данная программа строит трехмерную модель руки.\n"
        "Для того чтобы ее посторить нужно либо ввести\n"
        "парметры либо загрузить с помощью файла формата\n"
        "txt парметры кисти и пальцев руки. Управление\n"
        "поворотом модели осуществляется с помошью кнопок.\n"
        "Чтобы посмотреть как сгибаются пальцы при разной\n"
        "силе мышц, то выбирете нужный вам палец в меню Модель.\n"
        "С помощью скроллбара задйте силу мышц и можете\n"
        "рассматривать палец под любым углом с помощью кнопок.\n"
,"Инструкция",MB_OK);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::N13Click(TObject *Sender)
{
        Application->MessageBox("Данная программа является\n"
                        "Выпускной квалификационной работой\n"
                         "Студента: Кичеева Дмитрия Максимовича\n"
                         "Группы: ЮУрГУ ЕТ-412\n"
                                "2019г\n" , "О программе" , MB_OK);
}
//---------------------------------------------------------------------------
```

В файле Unit1.h:

```
//---------------------------------------------------------------------------
```

```
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <Menus.hpp>
#include <Dialogs.hpp>

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
#include "math.h"

#include "Unit2.h"
#include "Unit3.h"
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:        // IDE-managed Components
        TTimer *Timer1;
        TButton *Button1;
        TButton *Button2;
        TMainMenu *MainMenu1;
        TMenuItem *SetParameters1;
        TMenuItem *Save1;
```

```
TMenuItem *Instruction1;
TMenuItem *N1;
TMenuItem *N2;
TMenuItem *N3;
TMenuItem *N4;
TMenuItem *N5;
TMenuItem *N6;
TMenuItem *N7;
TMenuItem *N8;
TMenuItem *N9;
TMenuItem *N10;
TMenuItem *N11;
TMenuItem *N12;
TMenuItem *N13;
void __fastcall FormCreate(TObject *Sender);
void __fastcall FormDestroy(TObject *Sender);
void __fastcall FormResize(TObject *Sender);
void __fastcall FormPaint(TObject *Sender);
void __fastcall Button1Click(TObject *Sender);
void __fastcall Button2Click(TObject *Sender);
void __fastcall Timer1Timer(TObject *Sender);
void __fastcall N1Click(TObject *Sender);
void __fastcall N2Click(TObject *Sender);
void __fastcall N3Click(TObject *Sender);
void __fastcall N4Click(TObject *Sender);
void __fastcall N5Click(TObject *Sender);
void __fastcall N9Click(TObject *Sender);
void __fastcall N10Click(TObject *Sender);
void __fastcall N12Click(TObject *Sender);
```

```
        void __fastcall N13Click(TObject *Sender);
private:        // User declarations
HGLRC ghRC;  // указатель на контекст воспроизведения (Rendering Context)
HDC   hDC; // дескриптор (контекст) устройств
public:                 // User declarations
        __fastcall TForm1(TComponent* Owner);
        bool bSetupPixelFormat(HDC hDC);
        void Draw_Hand();
        void Finger(GLUquadricObj *quadObj,double l, double x, double y, bool b);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

В файле Unit2.cpp:

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
bool TForm2::bSetupPixelFormat(HDC hDC)
{
  PIXELFORMATDESCRIPTOR pfd; //Создаем структуру
  int pixelformat;
  pfd.nSize = sizeof (PIXELFORMATDESCRIPTOR); //Размер структуры
  pfd.nVersion = 1;  //Версия структуры
  pfd.dwFlags = PFD_DOUBLEBUFFER | PFD_SUPPORT_OPENGL |
PFD_DRAW_TO_WINDOW;
  pfd.iLayerType =  PFD_MAIN_PLANE;  //Тип поверхности
  pfd.iPixelType = PFD_TYPE_RGBA;  //Формат указания цвета
  pfd.cColorBits = 16;  //Глубина цвета
  pfd.cDepthBits = 16;  //Размер буфера глубины
  pfd.cAccumBits = 0; //Общее число битовых плоскостей в буфере аккумулятора
  pfd.cStencilBits = 0;  //Размер буфера трафарета
  if (!(pixelformat=ChoosePixelFormat(hDC, &pfd))) {
        MessageBox(NULL, "Невозможно выбрать формат пикселей", "Error",
MB_OK);
        return false;
  }
  if (!SetPixelFormat (hDC, pixelformat, &pfd)) {
        MessageBox(NULL, "Невозможно установить формат пикселей", "Error",
MB_OK);
        return false;
  }
  return true;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormCreate(TObject *Sender)
{
```

```
  hDC2 = GetDC(Handle); //Handle – дескриптор окна (hwnd в WinAPI)
if (!bSetupPixelFormat(hDC2)) //Устанавливаем формат пикселей
  return;
ghRC2 = wglCreateContext(hDC2); //Создаем контекст воспроизведения
//
wglMakeCurrent(hDC2, ghRC2);  //Делаем его текущим
glClearColor(1.0, 1.0, 1.0, 1.0); //Цвет экрана при очищении
glEnable(GL_COLOR_MATERIAL);  //Разрешаем задавать цвет объектам
glEnable(GL_DEPTH_TEST);  //Тест глубины для объемности изображения
glEnable(GL_LIGHTING);  //Разрешаем освещение
glEnable(GL_LIGHT0);  //Включили освещение 0
float p[4]={3,3,3,1},
                d[3]={-1,-1,-3};
glLightfv(GL_LIGHT0,GL_POSITION,p);  //Установка позиции освещения
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormDestroy(TObject *Sender)
{
  if(ghRC2)
{
  wglMakeCurrent(hDC2,0);
  wglDeleteContext(ghRC2);
}
if(hDC2) ReleaseDC(Handle, hDC2);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormResize(TObject *Sender)
{
  glViewport(0, 0, Width, Height);
```

```
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(-3, 3, -3, 3, 2, 12);
  gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);
  glMatrixMode(GL_MODELVIEW);
}
//---------------------------------------------------------------------------
void   TForm2::Finger(GLUquadricObj *quadObj,double l, double x, double y, bool b)
{
//рисование связок пальцев
        gluQuadricDrawStyle(quadObj, GLU_FILL);
        glTranslated(x, y, 0);
        glColor3f(1,0,0);
        gluSphere(quadObj, 0.2,10,10);
        glTranslated(0, l/2, 0);
        gluSphere(quadObj, 0.2,10,10);
        glRotated (i, 0, 0, 1);
        glTranslated(0, l/4, 0);
        if(b==false)
        {
                gluSphere(quadObj, 0.2,10,10);
                glRotated (i, 0, 0, 1);
                glTranslated(0, l/4, 0);
        }
        glRotated (90, 1, 0, 0);
        glColor3f(0,0,1);
        gluSphere(quadObj, 0.15,10,10);
        gluQuadricDrawStyle(quadObj, GLU_LINE);
//рисование фаланг пальцев
```

```
        if(b==false)
        {
                gluCylinder(quadObj, 0.15, 0.15, l/4, 100, 100);
                glRotated (90, -1, 0, 0);
                glTranslated(0, -l/4, 0);
                glRotated (i, 0, 0, -1);
                glRotated (90, 1, 0, 0);
        }
        gluCylinder(quadObj, 0.15, 0.15, l/4, 100, 100);
        glRotated (90, -1, 0, 0);
        glTranslated(0, -l/4, 0);
        glRotated (i, 0, 0, -1);
        glRotated (90, 1, 0, 0);
        gluCylinder(quadObj, 0.15, 0.15, l/2, 100, 100);
        glRotated (90, -1, 0, 0);
        glTranslated(0, -l/2, 0);
        if(b==true)
                glRotated (30, 0, 0, 1);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Timer1Timer(TObject *Sender)
{
        i=ScrollBar1->Position;
        GLUquadricObj *quadObj;
        GLfloat rquad=90;
        quadObj = gluNewQuadric();
        glClear(GL_DEPTH_BUFFER_BIT | GL_COLOR_BUFFER_BIT);
        glPushMatrix();
        if(Form2->Caption=="Мизинец")
```

```
            Form2->Finger(quadObj ,1.5 , 0, 0, false);
        if(Form2->Caption=="Безымянный")
            Form2->Finger(quadObj ,2 , 0, 0, false);
        if(Form2->Caption=="Средний")
            Form2->Finger(quadObj ,2.2 , 0, 0, false);
        if(Form2->Caption=="Указательный")
            Form2->Finger(quadObj ,2 , 0, 0, false);
        if(Form2->Caption=="Большой")
            Form2->Finger(quadObj ,2 , 0, 0, true);
        gluDeleteQuadric(quadObj);
        SwapBuffers(hDC2);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
        GLfloat rquad=30;
  glRotatef(rquad,0.0f,1.0f,0.0f);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{
        GLfloat rquad=30;
  glRotatef(rquad,0.0f,-1.0f,0.0f);
}
//---------------------------------------------------------------------------
```

В файле Unit2.h:

```
//---------------------------------------------------------------------------

#ifndef Unit2H
```

```
#define Unit2H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>

#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/glaux.h>
#include "math.h"
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:        // IDE-managed Components
        TTimer *Timer1;
        TButton *Button1;
        TButton *Button2;
        TScrollBar *ScrollBar1;
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall FormDestroy(TObject *Sender);
        void __fastcall FormResize(TObject *Sender);
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
private:        // User declarations
        HGLRC ghRC2;   // указатель на контекст воспроизведения (Rendering Context)
```

```
        HDC   hDC2; // дескриптор (контекст) устройств
        int i;
public:                 // User declarations
        __fastcall TForm1(TComponent* Owner);
        bool bSetupPixelFormat(HDC hDC);
        __fastcall TForm2(TComponent* Owner);
        void Finger(GLUquadricObj *quadObj,double l, double x, double y, bool b);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```

В файле Unit3.cpp:

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm3::Button1Click(TObject *Sender)
{
        Form3->Close();
}
//---------------------------------------------------------------------------
```

В файле Unit3.h:

```
//---------------------------------------------------------------------------

#ifndef Unit3H
#define Unit3H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm3 : public TForm
{
__published:        // IDE-managed Components
        TLabel *Label1;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
        TLabel *Label8;
        TButton *Button1;
        TLabel *Label9;
```

```
        TLabel *Label10;
        TLabel *Label11;
        TLabel *Label12;
        TLabel *Label13;
        TLabel *Label14;
        TLabel *Label15;
        TEdit *Edit1;
        TEdit *Edit2;
        TEdit *Edit3;
        TEdit *Edit4;
        TEdit *Edit5;
        TEdit *Edit6;
        TEdit *Edit7;
        void __fastcall Button1Click(TObject *Sender);
private:        // User declarations
public:                 // User declarations
        __fastcall TForm3(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm3 *Form3;
//---------------------------------------------------------------------------
#endif
```