МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени И.С. ТУРГЕНЕВА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки 01.03.02 Прикладная математика и информатика

направленность (профиль) Системное программирование и компьютерные технологии

Студента Ващенко Артема Тарасовича шифр 192739

Факультет физико – математический

Кафедра информатики

Тема выпускной квалификационной работы

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АППРОКСИМАЦИИ ПОВЕРХНОСТЕЙ

Студент	_____	Ващенко А.Т.
Научный руководитель	_____	к.ф.-м.н., доц. Федяев Ю.С.
Зав. кафедрой	_____	к.ф.-м.н., доц. Дорофеева В.И.

Орёл 2023

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени И.С. ТУРГЕНЕВА»

Факультет физико-математический

Кафедра информатики

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Системное программирование и компьютерные технологии

УТВЕРЖДАЮ:
Зав. кафедрой
Дорофеева В.И.
«02» ноября 2022 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Ващенко Артема Тарасовича шифр 192739

1. Тема ВКР «Разработка программного обеспечения для аппроксимации поверхностей».

Утверждена приказом по университету от «02» ноября 2022 г. №2-3132.

2. Срок сдачи студентом законченной работы «10» июня 2023 г.
3. Исходные данные к работе: учебные пособия и курсы лекций по компьютерной графике, численным методам, разработке программного обеспечения,

электронные источники и ссылки по программированию, алгоритмизации, разработке приложений с применением графического интерфейса QT.

4. Содержание ВКР (перечень подлежащих разработке вопросов)

   ВВЕДЕНИЕ

   ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ

   1.1. Обзор программного обеспечения для аппроксимации поверхностей

   1.2. Постановка задачи

   1.3. Получение данных для аппроксимации

   1.4. Описание исходных данных

   1.5. Параметрическое задание поверхности

   1.6. Методы построения поверхностей

   ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

   2.1. Структура программного обеспечения

   2.2. Состав технологий программного обеспечения

   2.3. Создание графического интерфейса в Qt Design

   2.4. Реализация алгоритмов, необходимых перед аппроксимацией

   2.5. Реализация алгоритмов аппроксимации поверхностей

   2.6. Разработка блока визуализации поверхности

   2.7. Сборка программного обеспечение в монолитное приложение

   ЗАКЛЮЧЕНИЕ

   СПИСОК ЛИТЕРАТУРЫ

   ПРИЛОЖЕНИЯ

5. Перечень графического материала: рисунков – 29, таблиц – 3.

Дата выдачи задания «02» ноября 2022 г.

Студент _____ Ващенко А.Т.

Научный руководитель ВКР _____ Федяев Ю.С.

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов ВКР	Срок выполнения этапов работы	Примечание
Подбор и анализ источников и научных изданий в соответствии с темой исследования	ноябрь – декабрь 2022	
Написание введения	март 2023 г.	
Написание главы 1	апрель 2023 г.	
Написание главы 2	май 2023 г.	
Написание заключения	май 2023 г.	
Оформление ВКР	июнь 2023 г.	
Сдача ВКР	июнь 2023 г.	

Студент _____ Ващенко А.Т.

Научный руководитель ВКР _____ Федяев Ю.С.

АННОТАЦИЯ

ВКР бакалавра на тему «Разработка программного обеспечения для аппроксимации поверхностей» Содержит 62 страницы текста, рисунков – 29, использованных источников – 20.

В настоящий момент подобные задачи востребованы в связи с громоздкостью популярных программных систем автоматизированного проектирования для обычных пользователей, не имеющих полное представление о графическом моделировании.

Ключевые слова: аппроксимация; поверхность; визуализация; Безье; B-сплайн; NURBS; облако точек; контрольные точки

Предмет исследования. Разработка программного обеспечения для аппроксимации поверхностей.

Объект исследования. Методы аппроксимации поверхностей.

Цель работы. Разработка и применение программного обеспечения для аппроксимации поверхностей на языке программирования Python

Разработка приложения. Со стороны разработки используется язык программирования Python и библиотеки как представителей сообщества разработчиков, так и представителей крупных компаний

Результаты работы. Разработано программное обеспечение в виде приложения для персональных компьютеров с использованием языка программирования Python и библиотек как для аппроксимации поверхности, так и для её отображения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Обоснование выбора темы и её актуальность

Трехмерное моделирование становится неотъемлемой частью нашей повседневной жизни. В настоящее время множество проектировщиков и конструкторов используют в начале своих проектов компьютерные модели для менее дорогостоящего представления широкой публике. Также созданные ими проекции реальных объектов можно было немного дешевле изменять, чем изготавливать заново действительную копию объекта. Одним из примеров столь дорогостоящих изменений может послужить проектирование крыла самолета, которое создавать с учетом каждого последующего изменения может быть ресурсозатратным мероприятием. Намного выгоднее проводить тестирование крыла в виде модели в составе общей модели самолета и вносить изменения для получения желаемого результата – воздушного перемещения всего самолета.

Этот пример даёт вам представление об актуальности компьютерного моделирования. Поэтому, для выполнения цели моделирования объекта с определенной высокой точностью и были созданы пакеты программного обеспечения для автоматизированного проектирования. Устаревшие двумерные чертежи также используется конструкторами до сих пор, несмотря на то что они не позволяют изменять отображаемые на них объекты также наглядно и просто, как с помощью систем автоматизированного проектирования (или CAD - систем).

Также построение поверхностей необходимо и для более распространенных сфер человеческой деятельности. Например, моделировать ландшафт места действия для компьютерной игры или современного кинофильма можно также аппроксимируя поверхность с помощью точек, находящихся достаточно далеко на оси Z для построения холмистой области ландшафта. Аппроксимация поверхностей

может пригодиться при построении препятствий для беспилотных автомобилей при помощи технологии обнаружения и определения дальности с помощью света (общее название – LIDAR). Вышеописанной технологией получаем множество точек, характеризующих объект и с помощью аппроксимации мы создадим поверхность объекта из реального мира, с которым будет взаимодействовать беспилотный автомобиль.

Данная тема выбрана по причине практического интереса построения моделей различных объектов окружающего мира для манипуляции ими в программах автоматизированного проектирования и трехмерного моделирования.

Степень разработанности объекта исследования

Объект исследования разрабатывается для различных сфер производства, в котором необходимо моделирование поверхностей для разнообразных целей. Многие частные компании на момент написания работы разработали системы автоматизированного проектирования, в которых встроена возможность аппроксимации поверхностей по набору точек. Позднее в средствах создания компьютерной графики аппроксимация поверхностей также нашла свое применение. Примером могут послужить такие программные обеспечения как Autodesk 3DS Max, Autodesk AutoCAD, PTC Creo и во многих других САПР и средств создания трехмерной графики.

Так же можно найти некоторое количество научных работ, посвященных как аппроксимации в общем, так и применению аппроксимации поверхности для создания конкретных моделей. К примеру, с помощью аппроксимации поверхности были созданы модели как переходные отверстия печатной платы [1], так и лопасти вентиляторов на основе будучи используемом в данной выпускной квалификационной работе алгоритмом [2].

Также следует отметить, что данная тема рассматривалась также и научными деятелями физико-математических наук. Многие исследователи вопроса

моделирования обращались к разделу начертательной и дифференциальной геометрии для построения основных поверхностей. Также требуется раздел теории поверхностей для более детального изучения самих поверхностей для их взаимодействия внутри смоделированной системы. Из наиболее интересных работ можно отметить изогеометрическую аппроксимацию поверхностей, описанную Борисом Ивановичем Квасовым в своей монографии «Методы изогеометричекой аппроксимацией сплайнами». Также рассматривалось и практическое применение построения поверхностей по экспериментальным точкам в языках программирования Turbo C и Fortran благодаря Шикину Е. В. И Плису А.И. Но выпущенное от них учебное издание немного устарело, ввиду малой популярности вышеописанных языков программирования в настоящее время.

Исходя из ранее упомянутого программного обеспечения, продолжается как применение, так и ознакомление простых пользователей и начинающих инженеров с особенностями алгоритмов аппроксимации поверхности через статьи научных журналов. Также на этом программном обеспечении выпускаются по наше время и методические материалы при различных кафедрах. Как от кафедры начертательной геометрии, так и от кафедры машиностроения.

Объект исследования

Объектом данного исследования являются методы аппроксимации поверхностей.

Предмет исследования

Предметом исследования в данной выпускной квалификационной работе является разработка программного обеспечения для аппроксимации поверхностей.

Цель работы

Целью данной выпускной квалификационной работы являются разработка и применение программного обеспечения для аппроксимации поверхностей на языке программирования Python.

Основные задачи исследования

1. Изучение методов аппроксимации поверхностей.
2. Постановка задачи аппроксимации поверхности исходя из набора координат трехмерных точек в текстовом файла.
3. Разработка программного обеспечения для аппроксимации поверхностей.
4. Тестирование и апробация разработанного программного обеспечения.

Структура работы

Данная дипломная работа состоит из введения, двух глав, заключения, а также списка источников и приложения.

В введении данной работы рассматривается обоснование выбора темы дипломной работы, а также определения её актуальности. Также рассматривается степень разработанности проблемы, предмета и объекта исследования. Далее будут сформулированы цели и задачи данной работы и краткое описание методов теоретического и эмпирического исследования и обработки данных, непосредственно затрагивающих данную тему.

В первой главе более подробно описывается критический анализ состояния данной темы и предлагаемые способы её решения с теоретической точки зрения, на основе изученной литературы и научных статей в сети Интернет.

Вторая глава данной работы представляет из себя проведение исследования темы на основе практического применения теоретической информации с указанием результатов этого исследования.

В заключении будут сформулированы выводы по результатам данной работы, пересмотр актуальности данной темы и предложения улучшений практической реализации темы разработки программного обеспечения для аппроксимации поверхностей.

Также будет приведен список используемых источников и приложения, содержащие как исходный код программного обеспечения, так и немного более подробные иллюстрации его работы.

ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ

1.1. Обзор программного обеспечения для аппроксимации поверхностей

В настоящее время, тема «разработка программного обеспечения для аппроксимации поверхностей» является востребованной, поскольку компьютерное моделирование повсеместно применяется инженерами различных областей, а также художниками трехмерной компьютерной графики. Всё активнее моделирование объектов реального мира используется в качестве простой визуализации с целью получение представления об объекте, а также для проведения симуляции различных ситуаций над построенной моделью. Именно для того, чтобы воссоздать различные объекты достаточно объемным определением и выполнением различных условий, и были созданы программные продукты CAD.Именно с их помощью и возможно оптимизировать и упростить процесс конструирования моделей, повысить производительность, улучшить качество и уровень детализации проекта, улучшить связь документации и часто способствовать созданию базы данных производственного проектирования. Примерами таких программных продуктов является Rhinoceros Rhino, Automapki от Autoshaper, Geomagic Wrap от 3D Systems и т.п. Из сегмента 3D моделирования среди программ, работающими с аппроксимацией поверхностей, в частности – NURBS поверхностей, применяются такие программные продукты как Autodesk 3Ds Max, Blender и т.п.

Рисунок 1 – Логотип Rhinoceros

Рассмотрим немного подробнее примеры реализации аппроксимации поверхностей методом NURBS поверхностей. В качестве первого примера выступит программный пакет Rhinoceros 7, поставляемый с пробным периодом в 90 дней при регистрации пользователя.

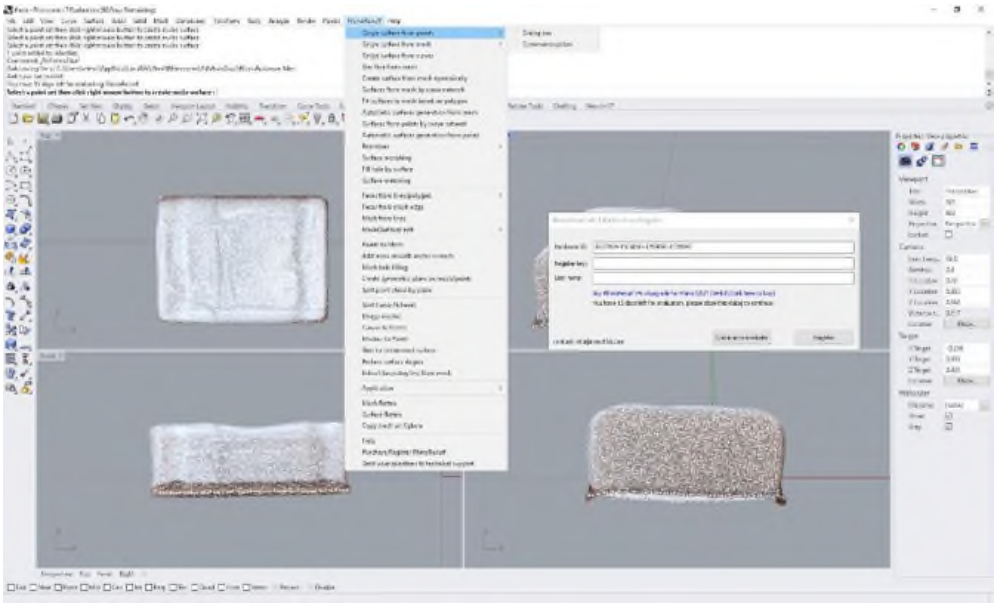

Рисунок 2 – Интерфейс САПР Rhinoceros Rhino 7

Для аппроксимации поверхности с помощью NURBS требуется установка плагина для Rhino 7 – Rhino Resurf, поставляемого на коммерческой основе с 15 дневным бесплатным периодом. Не смотря на минималистичный интерфейс программного продукта, для одной лишь аппроксимации поверхности этого много и, учитывая не лучшую степень проработанности дополнительного модуля (далее дополнения) Rhino Resurf, Конкретнее говоря, хоть и программа предлагает выбрать точки после того, как был выдан запрос о приобретения лицензии с возможностью выбора продолжения пробного периода, аппроксимировать поверхность из облака точек может быть затруднительно (см. рис. 2).

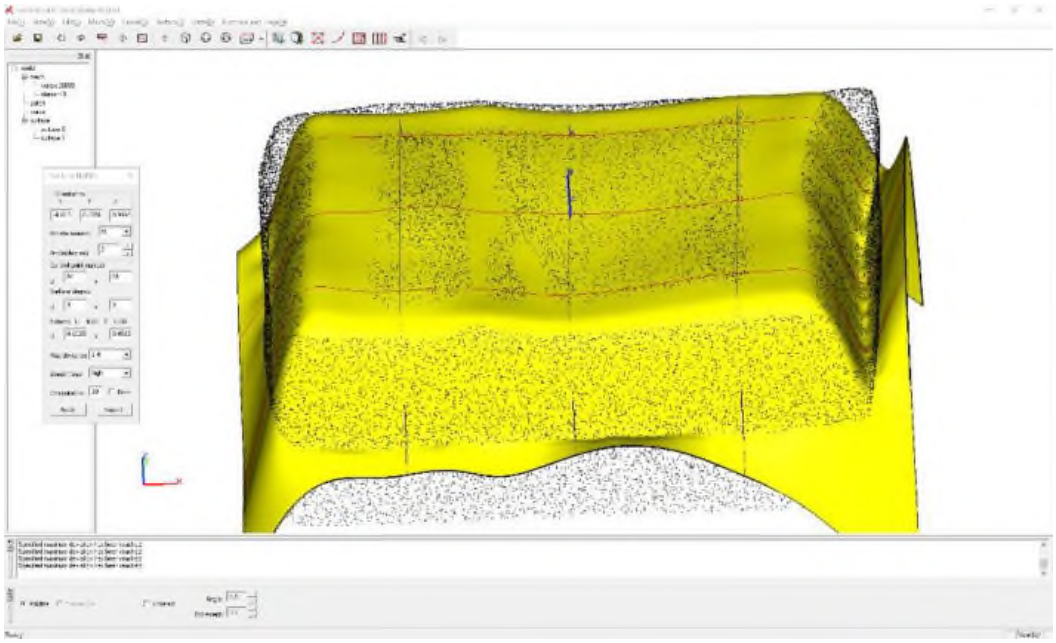

Рисунок 3 – Интерфейс ПО Point Cloud to NURBS (Beta) от RESURF

Отдельное программное обеспечение, поставляемое на сайте разработчика дополнения по идентичной лицензии и находящееся в бета версии, не может похвастаться столь проработанным интерфейсом, в отличии от Rhino 7.К тому же, аппроксимация поверхности из облака точек модели флэш накопителя оставляет желать лучшего по причине искажений итоговой поверхности (см. рис.3).

За другим примером можно обратиться к программному продукту Autoshaper от бельгийской компании Automapki. Она выполняет необходимую функцию аппроксимации поверхности, но тот же файл облака точек, что и при рассмотрении предыдущей CAD системы, однако при построении аппроксимированной поверхности, данное ПО выдало странные нежелательные поверхности, которые оказалось легко убрать с помощью ползунков внизу справа (см рис. 4 и 5).

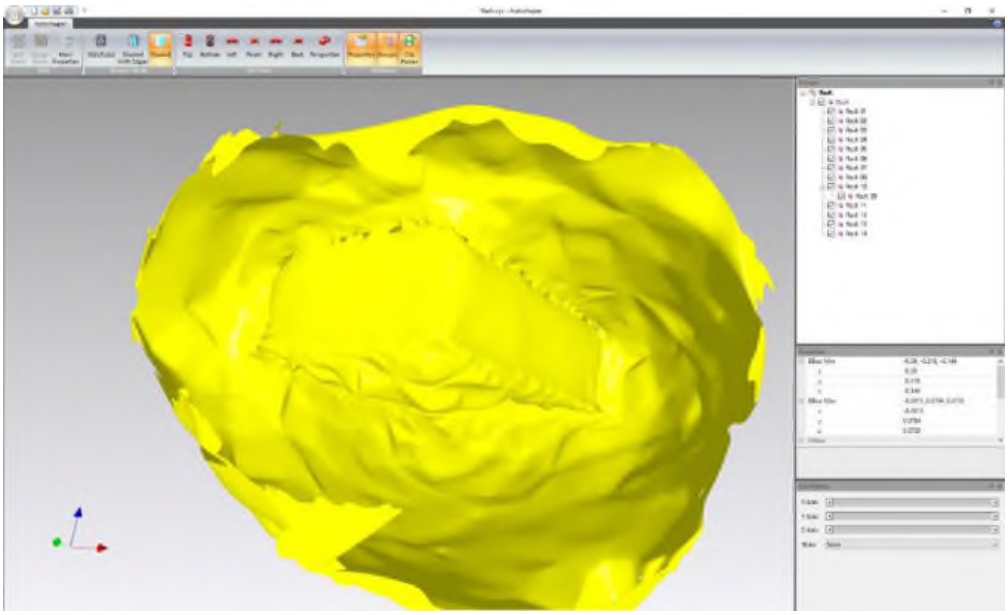

Рисунок 4 – Аппроксимированная поверхность с лишними кусками

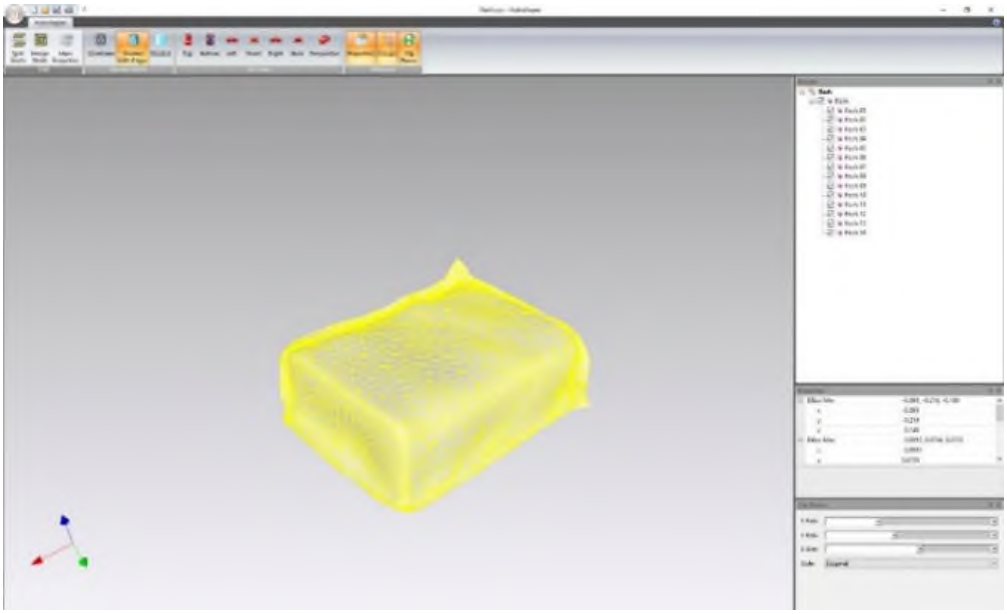

Рисунок 5 – Аппроксимированная поверхность со срезанными лишними кусками.

Несмотря яна то, что данное программное обеспечение удовлетворяет нашей задаче, однако отсутствует контроль в выборе метода аппрксимации и редактирование аппроксимированной поверхности является весьма ограниченным постредством её срезания.

Следующим в качестве примера аналога рассмотрим Geomagic Wrap от компании 3D Systems, продвигаемое как «наиболее простой в использовании, доступный, быстрый, точный путь от облаков точек к трехмерным полигональным и поверхностным моделям, которые могут быть мгновенно использованы в проектировании, производстве, машиностроении, искусстве, промышленном дизайне и т.д.». Несмотря на рекламный слоган с официального сайта компании, облако точек не выйдет максимально просто преобразовать облако точек в аппроксимированную поверхность (конкретно здесь строится NURBS поверхность).

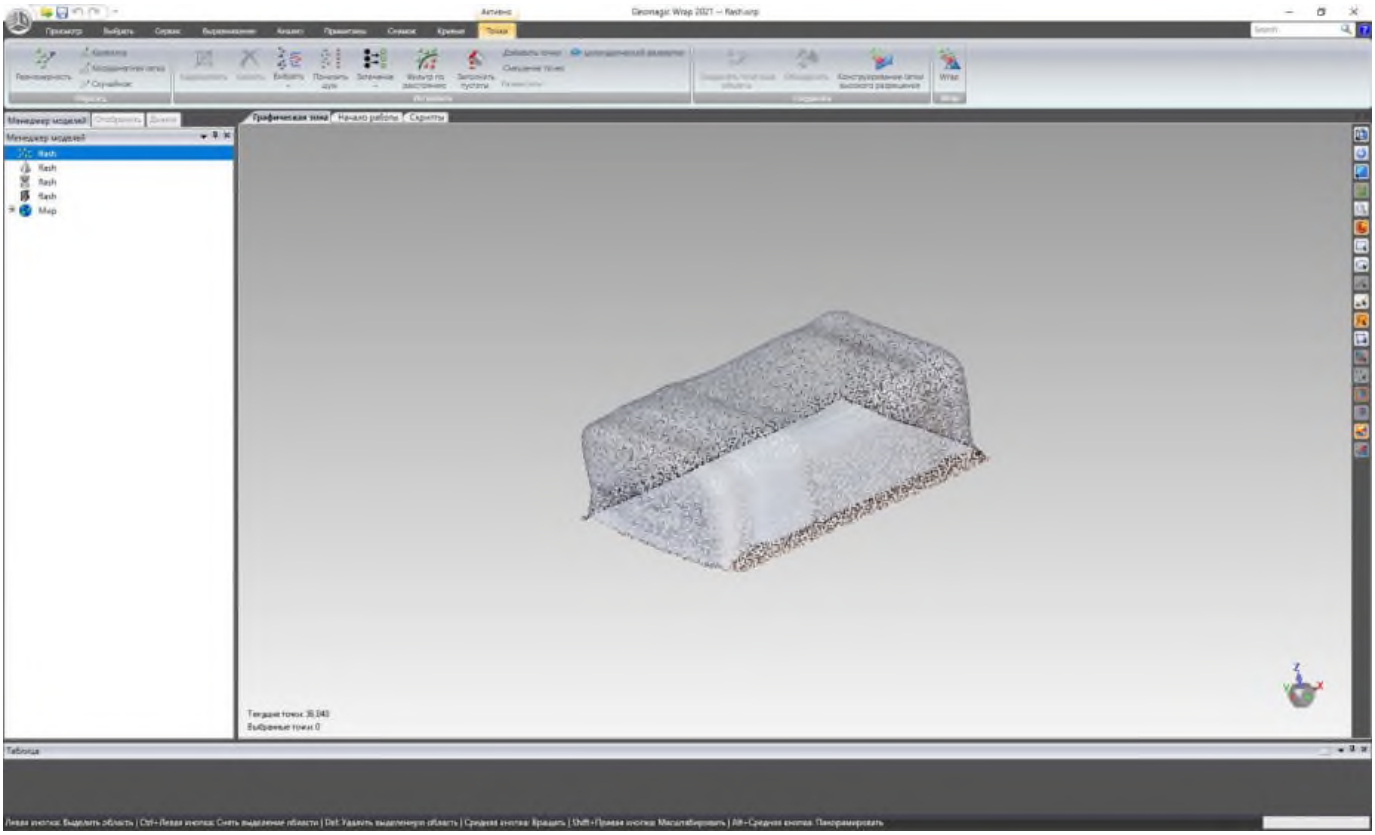

Рисунок 6 – Интерфейс программного продукта Geomegic Wrap 2021

На данном примере для того, чтобы получить аппроксимированную поверхность, необходимо вручную выстраивать вектор нормали для облака точек, потом построить полигональную сетку, выполнить точную обработку поверхности и затем начать аппроксимацию поверхности с помощью функции AutoSurface. По итогу мы и получаем аппроксимированную поверхностью В настоящее время,

данная реализация является наиболее оптимальной для аппроксимации поверхностей. Однако, настоящая реализация задачи выпускной квалификационной работы не позволяет совершить или изменить выбор в пользу аппроксимации с помощью поверхностей Безье или поверхности базисных сплайнов.

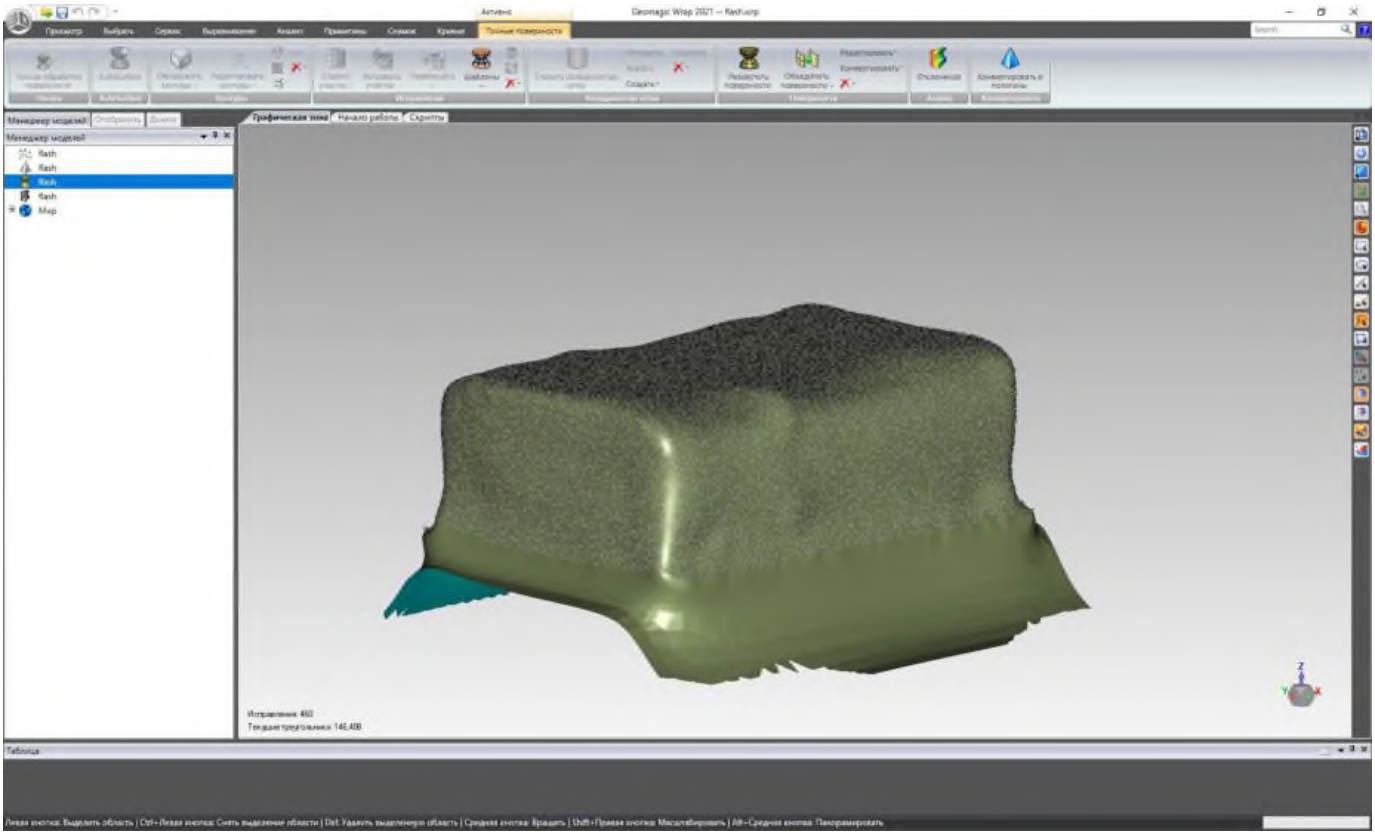

Рисунок 7 – Результат аппроксимации поверхности из облака точек

Также существуют во множестве систем автоматизированного проектирования и генерации компьютерной графики, как и свои реализации аппроксимации поверхностей, так и менее конкретные способы описания поверхностей через более простые кривые. Также в большинстве популярных CAD системах существуют среды реализации скриптов, поддерживающих выбранные разработчиком языки общего или специального назначения и предоставляющие необходимый набор связей между компонентами программного продукта и средой выполнения скриптов, чтобы при должной подготовке пользователь мог управлять

программными продуктами для автоматизированного проектирования посредством написанных для собственных целей скриптов.

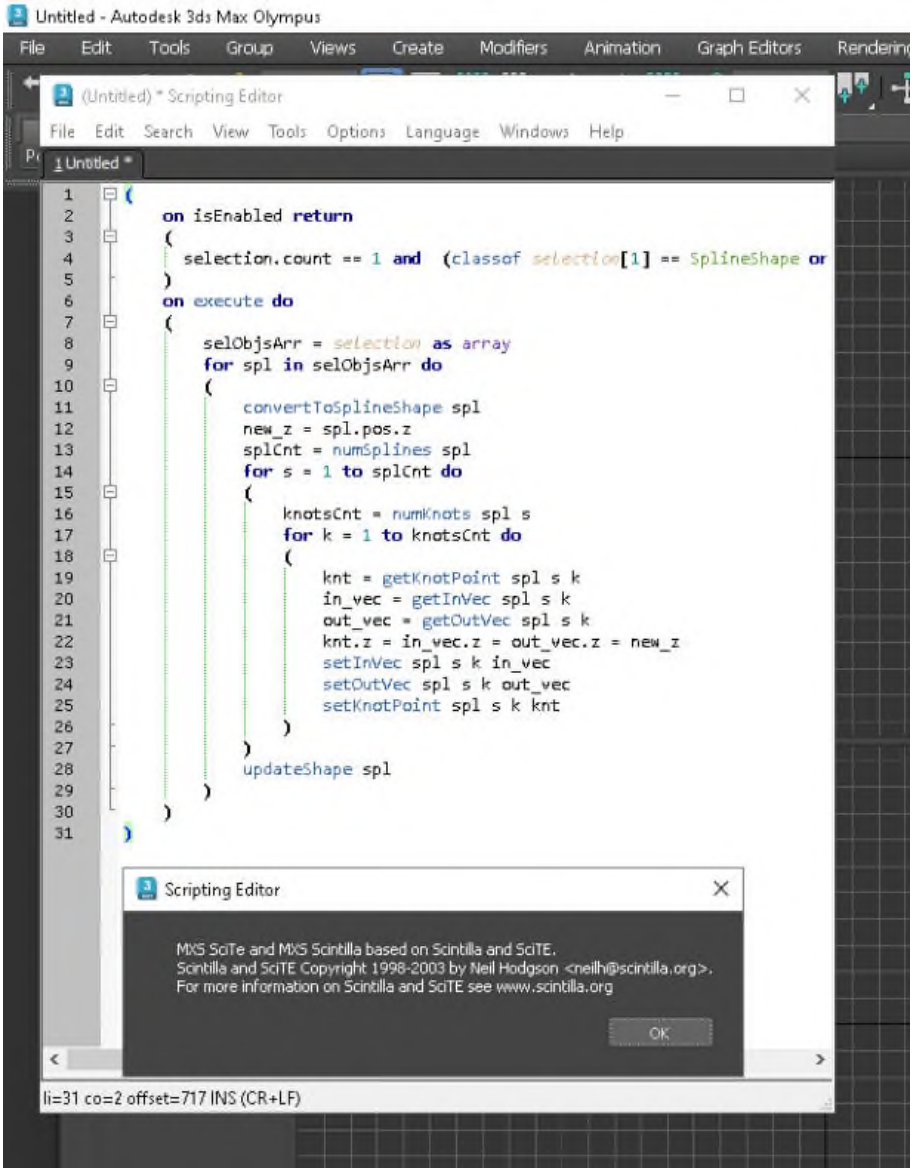

Рисунок 8 – Пример написания скрипта на языке программирования MAXScript внутри программного продукта Autodesk 3dsMax

Также не исключается возможность применить часть разрабатываемого программного обеспечения для аппроксимации поверхности, реализуя в поддерживаемых средах программирования в вышеупомянутых системах.

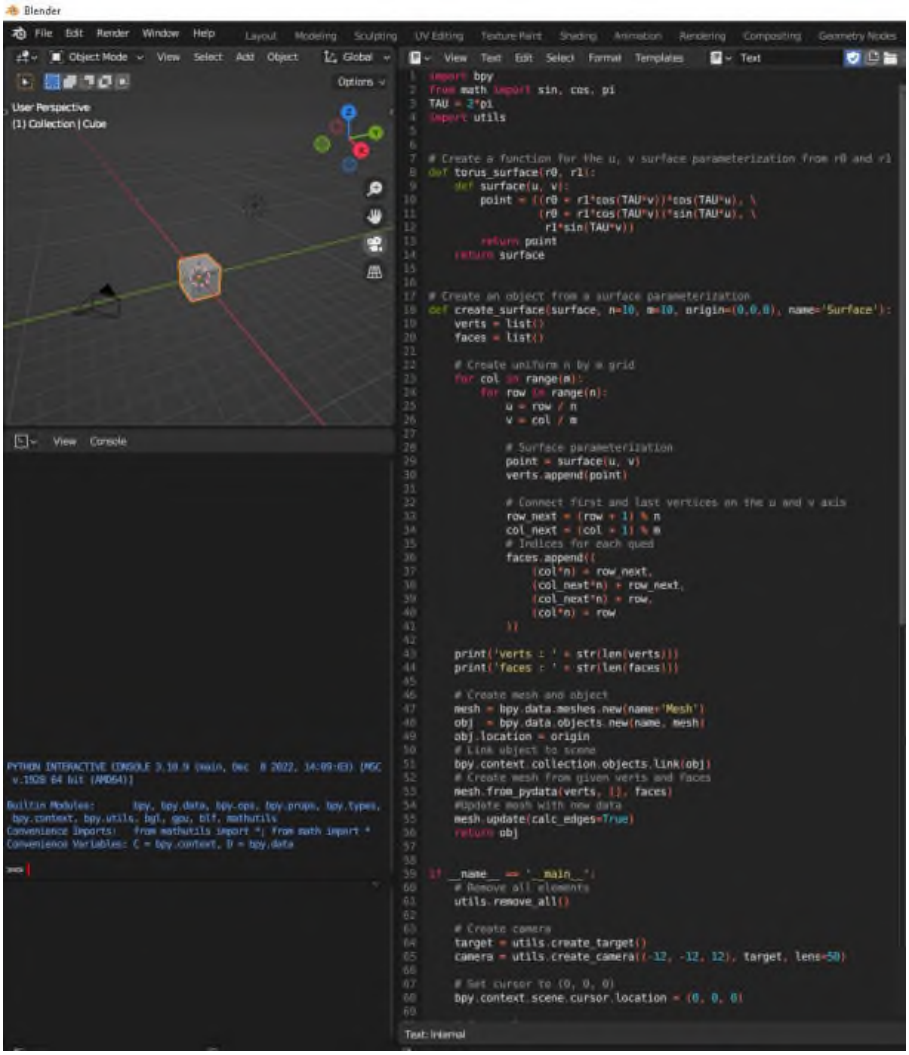

Рисунок 9 – Пример написания скрипта на языке программирования Python в среде работы программного продукта Blender

1.2. Постановка задачи

С помощью данных способов аппроксимации поверхностей, можно сделать вывод о том, что необходимыми минимальными данными являются контрольные точки. Однако, с каждым последующим из представленных способов аппроксимации поверхности потребуется некоторые дополнительные данные для получения поверхности необходимого вида. Рассматривая, к примеру тот же базисных сплайн требуется набор узловых векторов, которого нет в файлах наборов точек. Также потребуется ряд решений для возможности проведения манипуляций

результирующей поверхностью после её аппроксимации с целью подгонки поверхности для получения желаемого результата самим пользователем. Проблема заключается в том, что пользователь может и не знать как всех тонкостей технологии систем автоматизированного проектирования о общем, так и методов аппроксимации поверхностей, в частности, ввиду различных причин. поэтому будут сгенерированы дополнительные для последующего изменения поверхности, дабы удовлетворить требования пользователя. Подробнее о способах генерации данных можно будет изучить во главе 2.

В данной работе поставлена задача разработка программного обеспечения для аппроксимации поверхностей. Его возможности будут заключаться в том, чтобы по набору контрольных точек или же по неструктурированному облаку точек, описывающему реальный объект, создать поверхность, которая и является моделью реального объекта с максимально возможной точностью, которую и дают методы аппроксимации, в отличии от метода реконструкции с помощью полигональной сетки. Перед непосредственной разработкой самого программного обеспечения стоит обозначить следующие требования:

1. Получение облака точек или сети контрольных точек различной степени разреженности, олицетворяющее подобный объект из реального мира.

Получение облака или сети точек осуществляется за счет использования методов фотограмметрии, которая представляет из себя доступный метод получения облака точек или сети контрольных точек.

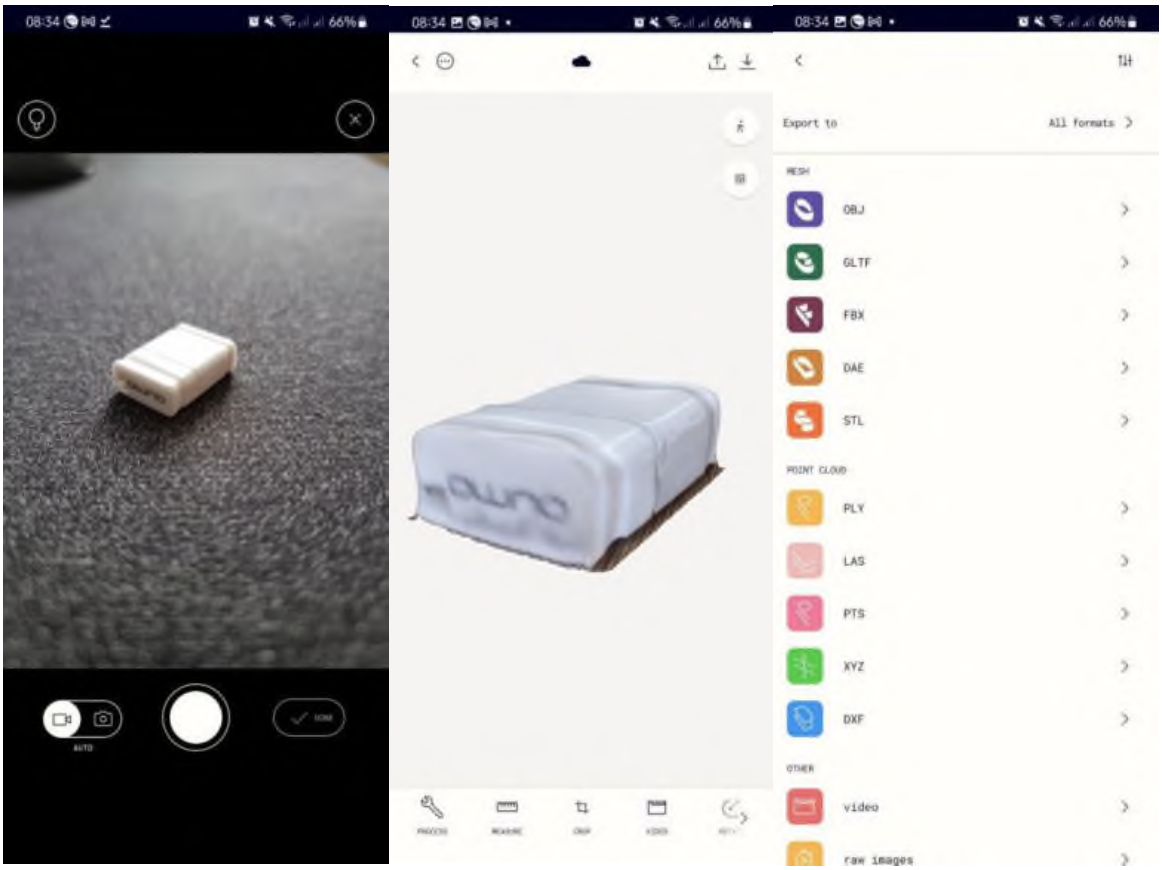

Рисунок 10 – Использование программного обеспечения для получения облака точек с помощью метода фотограмметрии

2. Сортировка полученного облака точек для создания контрольной сетки.

   При аппроксимации поверхности важно, чтобы итоговая поверхность не имела нежелательных искажении, которые могут возникать в следствии наличия точек вне контрольной сетки. При аппроксимации по набору контрольных точек важен порядок описания координат соответствующих точек. Если порядок не учитывать, аппроксимированная поверхность может построиться со значительными искажениями и слабо представлять модель, а порой и вообще не представлять, модель реального объекта.

3. Применение необходимых пользователю способов аппроксимации поверхностей.

От применения необходимых алгоритмов аппроксимации поверхности напрямую зависит от степени свободы в последующем изменении (доработке) поверхности и её применение в других сферах человеческой деятельности.

4. Аппроксимация самой поверхности.

   Сама аппроксимация должна проходить с максимально приемлемой точностью.

5. Отображение аппроксимированной поверхности пользователю.

   Отображение аппроксимированной поверхности должна соответствовать требованиям удобства в применении и возможности манипулировать результирующей поверхностью для соответствия требованиям точности, зависящим напрямую от пользователя.

6. Коррекция поверхности по с помощью доступных дополнительных данных, соответствующими способами.

   При аппроксимации с помощью поверхностей Безье достаточно выполнить требование в возможности перемещать контрольные точки в соответствующей контрольной сети. При аппроксимации с помощью базисных сплайнов потребуется не только иметь возможность передвигать контрольные точки в пространстве, но и иметь возможность изменения расстояния между узлами, что также изменит аппроксимированную поверхность нужным для пользователя способом. Наконец, при аппроксимации с помощью ненормированных рациональных базисных сплайнов требуется изменить весовой коэффициент, влияющий на отдаленность сегмента поверхности от ближайшей к нему контрольной точки.

7. Экспорт результирующей поверхности в 3D объект для работы с другими программными продуктами группы систем автоматизированного проектирования (computer aided design programs).

   Для дальнейшего применения аппроксимированной поверхности, её будет необходимо применить в иных областях человеческой деятельности, поэтому важно сохранить данные о поверхности в максимально приближенном виде для

дальнейшего её использования в качестве дополнительного элемента в программных CAD системах, а также использования в том же ключе и для систем генерации графики компьютером (CGI).

Программное обеспечение представляет из себя приложение для персональных компьютеров, использующие только локальные мощности персонально го компьютера. Данная реализация имеет небольшой, но существенный ряд преимуществ:

- Доступ без использования сети Интернет
- Максимально возможная производительность за счет использования локальных ресурсов компьютера

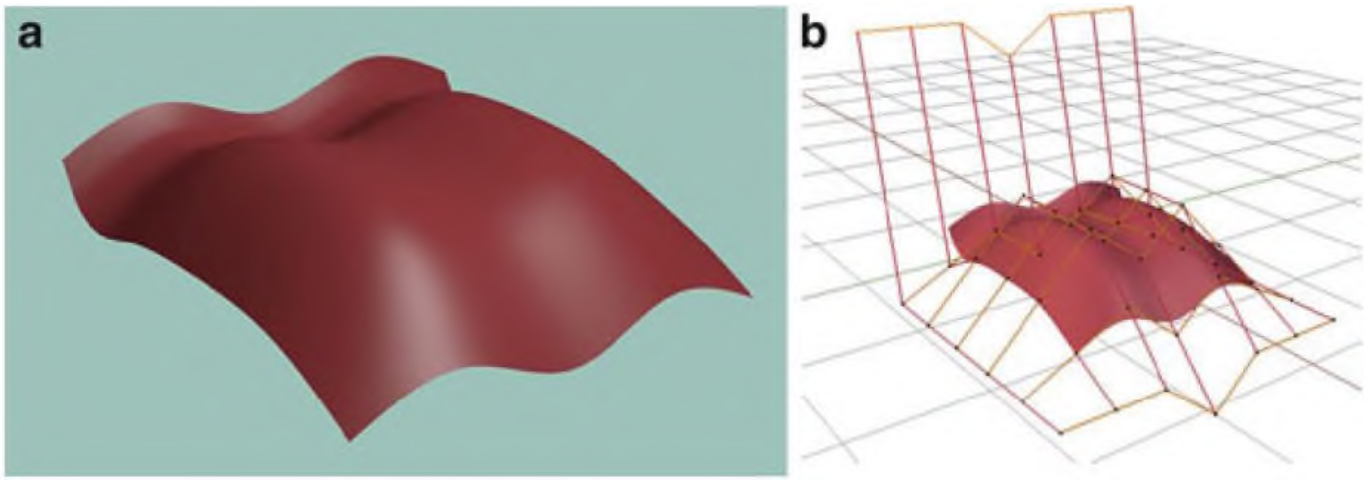

Рисунок 11 – Пример реконструкции поверхности: a – реконструкция с помощью полигональной сетки; b – аппроксимация NURBS – поверхностью

Конкретизируя вышеперечисленные все вышеперечисленные требования, нюансы в разработке и возможности, в данной выпускной квалификационной работе требуется создать программное обеспечение для аппроксимации поверхностей в виде приложения для персональных компьютеров с реализации графического интерфейса.

1.3. Получение данных для аппроксимации

Для воссоздания модели реального объекта потребуются его координаты в трехмерном пространстве. Они будут представлять из себя набор контрольных точек для аппроксимации поверхности. Однако, набор контрольных точек требуется для начала получить. Одними из способов получения контрольных точек является

- Моделирование группировкой простых геометрических объектов
- Сканирование объекта с помощью фотограмметрии
- Сканирование объекта специальным 3D сканером,
- Добавление точек в среде графического редактора, отображающие форму реального объекта.
- Модификация базовых геометрических объектов

Был выбран способ сканирования объекта с помощью фотограмметрии по причине доступности данного способа, ввиду наличия модуля камеры в подавляющем большинстве мобильных устройств, а также несколько большей доступности фотоаппаратов, в отличии от 3D сканеров. Одним из положительных факторов можно выделить практическое удобство построения массива контрольных точек без необходимости прямого контакта с объектом для его измерения, как в случае ручного добавления точек. Так же нет необходимости в реконструкции исходного объекта через простые геометрические фигуры чтобы получить координаты точек из группы геометрических примитивов или их измененных версий, что также является одним из неточных способов моделирования.

Сканирование с помощью 3D сканеров не обладает большой доступностью в связи с необходимостью приобретения специфического оборудования: различные виды 3D сканеров; LiDAR сканеров или телефонов с модулем LiDAR [3].

Рисунок 12 – Примеры 3D Сканеров

Рисунок 13 – Получение 3D модели с помощью множества снимков – фотограмметрия

С помощью технологии фотограмметрии, доступной в специально разработанных программных продуктах (прим. PolyCam [4], Autodesk ReCap, Ajisoft Metashape и другие) мы можем получить облако точек, описывающее

исходный объект. Для дальнейшей работы с полученными данными потребуется их структурировать в сетку с набором координат. В следующем параграфе опишем необходимое для работы описание данных контрольных точек.

1.4. Описание исходных данных

С математической точки зрения задача строится следующим образом. По полученному множеству вершин в виде координат точек в трехмерном пространстве, построить гладкую поверхность. Одним из востребованных условием является плавное изменение получаемой поверхности и прохождение результирующей поверхности вблизи вышеупомянутых вершин.

Сначала по данному набору точек необходимо построить многогранную поверхность, вершины которого совпадают с вершинами исходного набора. Данную поверхность называют ***опорной*** или ***контрольной***. С её помощью можно увидеть, в какой последовательности будет построена итоговая поверхность.

Наконец, рассмотрим массив, организованный в виде двумерного графа таким образом, чтобы он был эквивалентен прямоугольной сетке. То есть, в массиве

$$\begin{matrix} P_{00}, & P_{01}, & \dots & P_{0n}, \\ \dots & \dots & \dots & \dots \\ P_{m0}, & p_{m1}, & \dots & P_{mn} \end{matrix}$$

вместе с вершинами P_{ij} принимаются во внимание также связующие их ребра

$$P_{ij}P_{i,j+1} \quad i = 0,1,\dots,m-1,m; \quad j = 0,1,\dots,n-1$$
$$P_{ij}P_{i+1,j} \quad i = 0,1,\dots,m-1; \quad j = 0,1,\dots,n-1,n.$$

Вершины

$$P_{ij} \quad i = 1,\dots,m-1 \quad j = 1,\dots,n-1$$

являются *внутренними*, а вершины

$$P_{\boldsymbol{0j}} \quad j = 0,1,\dots,n-1;$$
$$P_{\boldsymbol{in}} \quad i = 0,1,\dots,m-1;$$
$$P_{\boldsymbol{mj}} \quad j = 1,2,\dots,n;$$
$$P_{\boldsymbol{i0}} \quad i = 1,2,\dots,n;$$

называются ***граничными***. Легко можно пронаблюдать, что есть 4 соседние вершины для каждой внутренней вершины, для каждой граничной – 3 вершины. Только у четырех вершин, называемых ***угловыми***,

$$\mathrm{P}_{00},\quad \mathrm{P}_{0n},\quad \mathrm{P}_{mn},\quad \mathrm{P}_{m0},$$

существует только 2 соседние вершины. Организованный таким образом массив данных является ***опорным графом*** искомой поверхности.

В случае воссоздания геометрически более сложных объектов, имеет место разделение их набора вершин на более простые наборы. С помощью таких уменьшенных наборов вершин можно построить более простые поверхности, которые можно состыковать между собой. Тем самым, для реконструкции объекта в трехмерную модель используется весь набор данных, взятый отдельными частями.

1.5. Параметрическое задание поверхности

Параметрически заданная поверхность - множество *S* точек *M (x, y, z)* пространства, декартовые координаты которых определяются с помощью соотношения

$$x = f_1(u,v),\quad y = f_2(u,v),\quad z = f_3(u,v),$$
$$a \le u \le b,\quad c \le v \le d,$$

где $x = f_1(u,v),\quad y = f_2(u,v),\quad z = f_3(u,v)$ – функции, непрерывные заданном прямоугольнике

$$R = \{(u,v) | a \le u \le b, c \le v \le d\},$$

или в векторно-матричной форме:

$$\mathrm{R} = \mathrm{R}(u,v) = \begin{pmatrix} f_1(u,v) \\ f_2(u,v) \\ f_3(u,v) \end{pmatrix}, (u,v) \in \mathrm{R}$$

Данные соотношения являются *параметрическими уравнениями поверхности S* или *параметризацией поверхности S.* Величины *u* и *v* – называются

внутренними криволинейными координатами поверхности S. С их помощью можно описать координатную сетку, описываемую следующими уравнениями: $\mathrm{R} = \mathrm{R}(u, v_0 = const)$ – уравнение линии u; $\mathrm{R} = \mathrm{R}(u_0 = const, v)$ – уравнение линии v.

1.6. Методы построения поверхностей

Для начала определим, каким образом можно построить искомую поверхность по контрольным точкам, описанным в параметрическом виде. В системах автоматизированного проектирования применяются следующие, довольно популярные методы построения поверхностей:

1. С помощью поверхностей Безье;
2. С помощью поверхностей базисных сплайнов (B- сплайнов);
3. С помощью поверхностей ненормированных рациональных базисных сплайнов (NURBS - поверхностей);

Рассмотрим уравнения вышеописанных поверхностей.

Основная формула параметрический поверхности Безье:

$$S(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u) B_j^m(v) P_{i,j} \quad 0 \le u \le 1 \quad 0 \le v \le 1$$

Где $B_i^n(u), B_j^m(v)$ являются полиномами Бернштейна для соответствующего параметра u, v. Вышеперечисленные полиномы вычисляются по следующей формуле:

$$B_j^n(u) = \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i}.$$

Данная формула является аналогичной для параметра v при количестве столбцов m вместо n. Рассмотрим основные свойства поверхностей Безье:

- произведение полиномов Бернштейна неотрицательно на протяжении всей итерации для каждого параметра;

- Свойство разбиения единицы. Сумма произведений всех полиномов Бернштейна для каждого параметра u и v равна единице;
- поверхность Безье содержится в выпуклой оболочке её контрольных точек;
- 4 угловые контрольные точки лежат на самой поверхности;
- данная поверхность не отображается сама в себя (аффинно - инвариантна);
- изменение по крайней мере одной контрольной точки в начально м массиве контрольных точек приведет к заметному изменению всей поверхности;
- добавление новых контрольных точек приведет к пересчету всех параметрических уравнений поверхностей Безье;
- степень функциональных коэффициентов прямо связано с количеством контрольных точек и растет при их увеличении;

Последние 3 свойства приводят к необходимости использовать другой способ построения поверхности – построение с помощью базисных сплайнов (далее - B-сплайнов). В отличии от поверхностей Безье, B-сплайны позволяют изменять координаты контрольной точки без перерисовки всей поверхности. Также, количество контрольных точек не зависит от степени функциональных коэффициентов поверхности.

Прежде чем перейти непосредственно перейти к определению формулы B-сплайна, рассмотрим множество U из $t+1$ неубывающих чисел

$$u_0 \le u_1 \le \cdots u_{i-1} \le u_i \le u_{i+1} \le \cdots \le u_h, \;\; i = 0 \ldots h.$$

называемое *узлами*. Ближайшие между собой узлы образуют *узловой диапазон* $[u_i, u_{i+1})$, в котором предстоит выполнить вычисление базисных функций до заданной степени. Однако, узловых диапазонов может и не существовать, ы связи с равенством соседних узлов между собой. Если такие узлы повторяются *s* раз, то в таком случае узел является множественным. Узловые векторы могут быть двух видов и отличаются удаленностью между диапазонами. Если диапазоны между всеми узлами равны, то узловой вектор, как и сам сплайн называют однородным. В

остальных случаях, считается неоднородным. Для получения однородного узлового вектора, достаточно просчитать сумму степени p и количества контрольных точек $n+1$: $h=p+n+1$. Можно задать данное множество двумя видами:

1. в явном виде: при p=2, n=5, h=2+5+1 => h=8 => U = (0,1,2,3,4,5,6,7,8)
2. в нормализованном виде, при тех же k и n => $U = (0, \frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}, 1)$

В случае неоднородного узлового вектора, требуется применить формулу:

$$u_i = \begin{cases} 0, & i < p \\ i - p + 1, & p < i < n \\ n - p + 2, & i > n \end{cases}$$

Однако, также возможно изменение узлов, не попадающих под условие 1 и 3 в уравнении выше, если они соответствуют узловому диапазону. Неоднородный узловой вектор также можно преобразовать у нормализованному виду.

Далее вычислить базисную функцию внутри диапазона в нулевой степени:

$$N_i^0(u) = \begin{cases} 1, & u_i \le u < u_{i+1} \\ 0, & \text{в остальных случаях} \end{cases}.$$

для дальнейшего вычисления базисной функции для B-сплайна требуемой степени p:

$$N_i^p(u) = \frac{u - u_i}{u_{i+p} - u_i} N_i^{p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1}^{p-1}(u).$$

Формула аналогична при замене количества контрольных точек n на m, параметра контрольной точки u на v, порядка i на j и степени p на q. Данные замены пригодятся для вычисления базисной функции по направлению v для вычисления поверхности.

В итоге составим формулу построения поверхности из тензорного произведения двух B-сплайновых кривых степени p и q соответственно:

$$S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} N_i^p(u) N_j^q(v) P_{ij}, u_0 \le u \le u_h, v_0 \le v \le v_k.$$

Свойства поверхности из базисных сплайнов.

- Если количество контрольных точек идентично степени базисной функции и узловой вектор состоит из нулей и единиц, количество которых равно $p+1$ (по $u-(q+1)$), то получится поверхность Безье
- Результат её базисных функций не является отрицательным для всех их коэффициентов *u* и *v* внутри узловых диапазонов $[u_i, u_{i+1})$, *i* и *j*, элементов и степеней p и q.
- Свойство разбиения единицы. Сумма всех $N_i^p(u)N_j^q(v)$ для $0 \leq (u,v) \leq 1$ равна единице.
- Обладает локальной модификацией за счет узловых диапазонов, т.е. при изменении положения точки $P_{i,j}$, поверхность изменится в полудиапазоне $[u_i, u_{i+1}) \times [v_j, v_{j+1})$.
- Свойство сильно выпуклой оболочки. Поверхность базисного сплайна содержится в выпуклой оболочке её контрольной сетки.
  Если *u* лежит в узловом диапазоне $[u_i, u_{i+1})$, то есть *p+1* базисных функций, что не равны нулю в данном диапазоне. Следовательно, и причастные к ним контрольные точки имеют ненулевые коэффициенты. Аналогичное можно проверить и для диапазона $[v_j, v_{j+1})$. *q+1* точек будет иметь ненулевые значения. Выпуклая оболочка названа сильной по причине выбора точки на построенной поверхности в диапазоне из полуоткрытого прямоугольника $[u_i, u_{i+1}) \times [v_j, v_{j+1})$, в которой ненулевое базисное значение имеют 4 контрольные точки.
- $S(u,v)$ является C^{p-s} непрерывной по направлению u (по v - C^{q-t}) при условии, что $u(\text{соотв.}\, v)$ является множественным узлом $s(\text{соотв.}\, t)$.
- Если к поверхности базисного сплайна применить аффинное преобразование, то результат можно построить из аффинных образов ее контрольных точек. При желании применить геометрическое или аффинное преобразование, нам достаточно применить преобразование к контрольным точкам и после получения

преобразованных контрольных точек новая B-сплайновая поверхность будет определяться этими новыми точками.

Последним способом аппроксимации поверхности является построение НеРавномерной Рациональной Базисной Сплайновой (Non-Uniform Rational B-spline) поверхности. Ранее в данной работе рассматривалось построение поверхности с помощью B-сплайнов и свойство неравномерности узловых векторов, влияющего на вышеописанную поверхность. Далее рассмотрим свойство рациональности B-сплайновой поверхности, но для начал следует рассмотреть **однородную систему координат.** Она отличается от декартовой системы координат наличием $D + 1$ координатой для каждого пространства. К примеру, для описания кривой на плоскости в однородной системе нужно 3 координаты (x, y, w), которые переносят на декартовую систему, разделив все координаты на последнюю $\left(\frac{x}{w}, \frac{y}{w}\right)$. Аналогичная ситуация происходит и с определением точки на поверхности: (x, y, z, w) . Поскольку одной из целей данной работы является получение поверхности, то потребуется перевод из однородной системы координат в декартовую.

По итогу NURBS поверхность строится по той же формуле, что и B-spline поверхность, но с применением обозначенного выше перехода:

$$\mathrm{S}(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} N_i^p(u) N_j^q(u) P_{i,j} \omega_{i,j}}{\sum_{i=0}^{n} \sum_{j=0}^{m} N_i^p(u) N_j^q(u) \omega_{i,j}}, \qquad u_0 \le u \le u_h, v_0 \le v \le v_k.$$

В одном из источников [5] можно заметить приведение базисных функций заменяется одной переменной:

$$S(u, v) = \sum_{i=0}^{n} \sum_{J=0}^{m} R_{i,j}(u, v) P_{i,j} \,,$$

$$R_{i,j}(u, v) = \frac{N_i^p(u) N_j^q(u) \omega_{i,j}}{\sum_{f=0}^{n} \sum_{g=0}^{m} N_f^p(u) N_g^q(u) \omega_{f,g}},$$

Где $R(u,v)$ – кусочная рациональная базисная функция.

При ненормированных узловых векторах

$$U = \{u_0, u_1, \dots, u_{i-1}, u_i, u_{i+1}, \dots, u_{h-1}, u_h\}, \qquad h = n + p + 1,$$

$$V = \{v_0, v_1, \dots, v_{j-1}, v_j, v_{j+1}, \dots, v_{k-1}, v_k\}, \qquad k = m + q + 1,$$

$$u_i = \begin{cases} 0, & i < p \\ i - p + 1, & p < i < n \\ n - p + 2, & i > n \end{cases}, \qquad v_j = \begin{cases} 0, & j < q \\ j - q + 1, & q < j < m \\ m - q + 2, & j > m \end{cases}.$$

Основные свойства NURBS поверхностей:

- Сумма $R(u,v)$ для всех значений u, v внутри узлового вектора = 1
- Каждая $R(u,v) \geq 0$ для на всех значениях u, v
- За исключением $p = 1$ или $q = 1$, каждая рациональная поверхностная базисная функция имеет ровно один максимум.
- Максимально возможный порядок рациональной B-сплайновой поверхности в каждом параметрическом направлении равен числу вершин управляющей сетки в этом направлении.
- NURBS поверхность порядка p, q (степень $p-1, q-1$) является функцией с непрерывностью C^{p-2}, C^{q-2} на всей её области.
- Неравномерная рациональная B-сплайновая поверхность инвариантна относительно проективного преобразования; т.е. любое проективное преобразование может быть применено к поверхности путем применения его к управляющей сети. Обратите внимание, что это более сильное условие, чем условие для нерациональной B-сплайновой поверхности.
- При $w_{i,j} \geq 0$ для всех i, j поверхность лежит внутри выпуклой оболочки контрольной сетки, образованного объединением всех выпуклых оболочек p, q соседних вершин полигональной сетки.
- Свойство уменьшения вариации в настоящее время не известно для рациональных B-сплайновых поверхностей.

- Влияние одной вершины контрольной сетки ограничено $\pm\frac{p}{2}, \pm\frac{l}{2}$ пролетами в каждом параметрическом направлении.
- Если число вершин контрольной сетки равно порядку в каждом параметрическом направлении и нет дублирующих значений внутренних узлов, неравномерная рациональная B-сплайновая поверхность является рациональной поверхностью Безье

ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1. Структура программного обеспечения

При эффективной работе с аппроксимированной поверхностью, как и при эффективной работе с компьютерной графикой в общем важны 2 фактора.

- Возможность изменить как необходимые контрольные точки, так и дополнительные данные, которые также учитываются при аппроксимации поверхности.
- Возможность рассмотреть каждую деталь аппроксимированной поверхности интуитивно понятным для пользователя способом.

Для начала потребуется возможность чтения набора отсканированных контрольных точек, необходимая для их компоновки в контрольную сеть. Порой в системах фотограмметрии контрольные точки называют облаком точек ввиду их хаотичного расположения относительно друг друга. В большинстве форматов облаков точек существует как свои методы получения и группировки контрольных точек, так и группировка на основе поиска одинаковых точек среди фото и решения системы нелинейных уравнений на основе найденных соответствий. Один из примеров описан в данной статье [6]. Из – за неоднозначности данных, полученных при фотограмметрии, следует провести сортировку точек для получения вышеописанной контрольной сети. Стоит учесть необходимость для пользователя возможность сохранять итоговую поверхность в том виде, который легко читается как системами автоматизированного проектирования, так и системами компьютерной графики.

Для получения аппроксимированной поверхности потребуется применить один из способов аппроксимации, описанных в п. 1.6. Т.к. в таких методах аппроксимации как аппроксимация с помощью поверхностей B-сплайна и NURBS

– поверхностей в качестве начальных данных применяются не только контрольная сеть, но и узловой вектор, а также весовые коэффициенты. Такие расчеты потребуется проводить заранее и позже дать возможность пользователю корректировать сгенерированные варианты под свои нужды.

Аппроксимированную поверхность и сгенерированные данные потребуется отобразить и редактировать в простом интерфейсе, чтобы пользователю не было необходимости путаться в нем. Немаловажным фактором является возможность рассмотрения поверхности под разным углом и с разным приближением, дабы результирующая поверхность могла удовлетворить потребность пользователя в создании гладкой поверхности.

Основываясь на данном описании функциональных блоков, можно представить модель взаимодействия пользователя с разрабатываемым программным обеспечением на схеме ниже

Данное приложение будет состоять из трех функциональных блоков, которые потребуется связать между собой.

1. Блок чтения — записи файлов. Необходим для чтения облака точек, создания контрольной сетки и сохранения готовой к дальнейшей работе поверхности.
2. Блок аппроксимации поверхности. Требуется для применения выбранного способа аппроксимации поверхности и генерации необходимых данных для выполнения своей задачи.
3. Блок отображения и редактирования аппроксимированной поверхности. Даст возможность настроить под нужды пользователя результат работы программы для сохранения через блок чтения — записи. Блок схема стандартной работы с разрабатываемым программным обеспечением приводится на рисунке ниже.

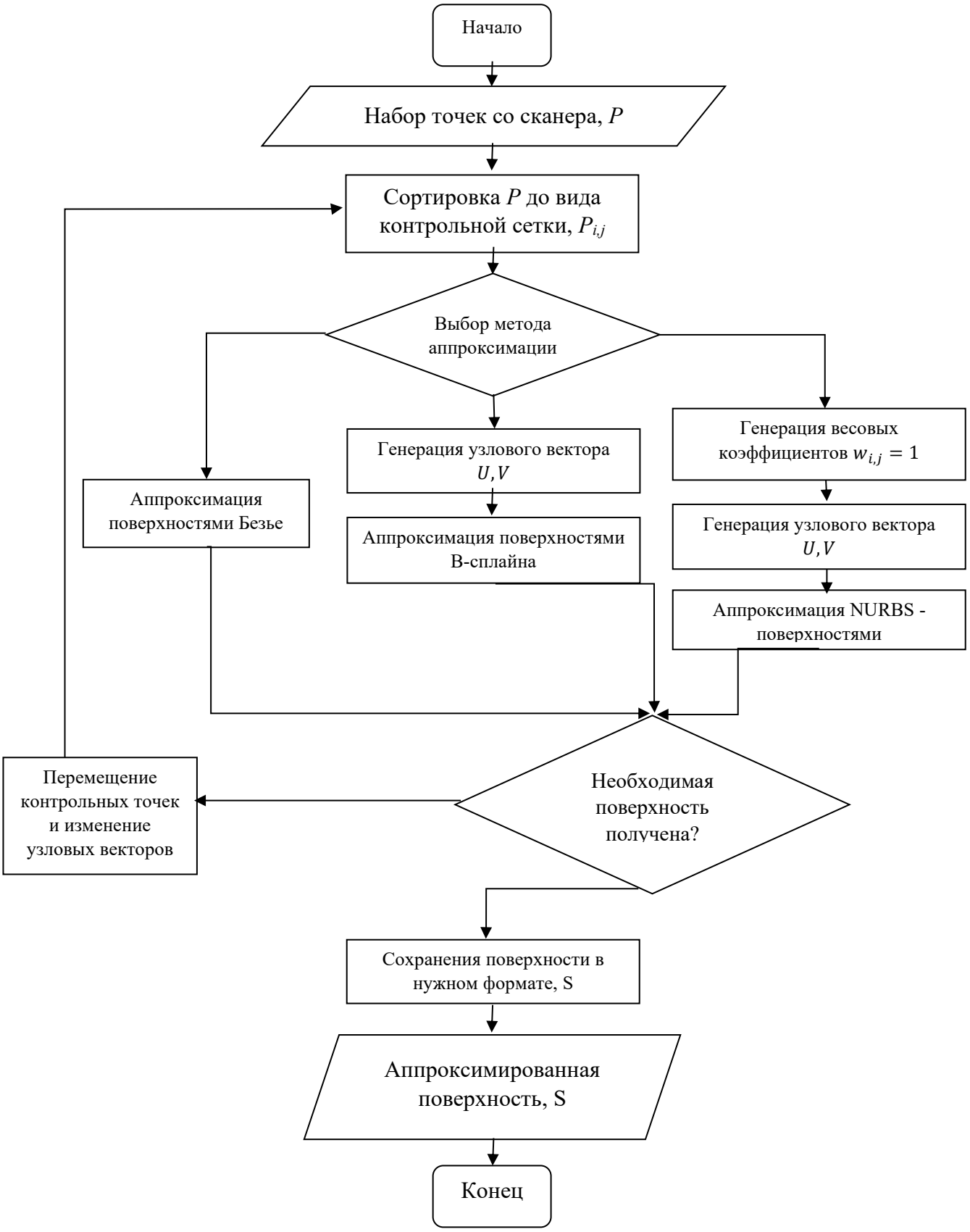

Рисунок 14 – Блок схема взаимодействия пользователя с ПО

Программное обеспечение представляет из себя приложение, содранное под операционной системой Windows 10. Доступ к исходному коду приложения доступен по ссылке к репозиторию из личного профиля на ресурсе GitHub по данному URL: https://github.com/artmen57/Graduate-work-App-development-for-surface-approximation.

2.2. Состав технологий программного обеспечения

При разработке программного обеспечения для аппроксимации поверхностей есть возможность применить практически любой современный язык программирования, поддерживающий применение программных платформ для создания графического интерфейса и поддержкой систем научной визуализаций данных.

При разработке программного обеспечения были использованы следующие инструменты:

- Язык программирования Python 3 (Далее —Pythons)
- Библиотека для построения поверхностей NURBS — Python (geomdl)
- Программный пакет Qt (Версия для Python 3 - PyQt)
- Программный пакет Visualization toolkit (VTK)

Python — язык программирования общего назначения с высокой степенью абстракции от особенностей компьютерных технологий. Будучи языком общего назначения, с помощью данного языка можно написать большое количество программ разной специфики и объема работ. Он набрал большую популярность на момент написания данной работы и входит в лидирующие позиции по популярности среди языков программирования [7]. Поддерживает все основные парадигмы программирования, тем самым не ограничивает разработчика в применении таких парадигмах программирования, как объектно – ориентированное программирование, функциональное программирование и др. Данный язык

реализован 20 февраля 1991 года благодаря датскому программисту Гвидо Ван Руссо. Хоть и язык быстро развивается за счет создания объемных библиотек разной направленности и активного сообщества, чья численность неуклонно растет, финальные решения по главным изменениям в языке программирования остается за Руссо.

Рисунок 15 – Логотип Python.

При выполнении данной выпускной квалификационной работы Python был выбран по причине удобства синтаксиса в разработке ПО. Также он удобен при использовании вместе с большими наборами библиотек, разработанных под использования операции, требующих большие объемы данных или объемные вычисления. Несмотря на вышеописанную возможность, также удобно использовать более простые функции из других файлов исходного кода Python.

На данный момент, разработано немалое количество небольших библиотек аппроксимации и интерполяции сплайнами. Одними из примеров являются SpliPy [8], nurbspy [9], NURBS-Python [10]. Именно последняя из перечисленных библиотек и применяется в данной работе.

NURBS-Python – объектно ориентированная библиотека расчета NURBS примитивов на чистом Python без внешних зависимостей с открытым исходным кодом. Автором данной библиотеки является Онур Рауф Бингол и создал он её в мае 2016 года, но в научной сфере о ней стало известно в январе 2019 года, когда он написал одноименную статью. Данная библиотека нам потребуется при аппроксимации поверхностей в соответствующем блоке. Также её модули визуализации могут послужить основой для разработки блока отображения и

редактирования поверхности. Для блока чтения — записи она особенно пригодится при сохранении итоговой поверхности для различных систем автоматизированного проектирования и компьютерной графики. Вышеописанные программные обеспечения оперируют такими форматами как .obj, .stl, .3dm, .smesh.

Основой разработки для блока отображения аппроксимированной поверхности стала Visualization Toolkit (VTK) – программная система с открытым исходным кодом для трехмерной компьютерной графики, обработки изображений и научной визуализации. Авторами данной системы являются Уильям Шредер, Кеннет Мартин и Билл Лоренсен и выпустили они её в 1994 году. Изначально данная система была разработана для языка программирования C++, однако при нарастающей популярности в кругах специалистов по научной визуализации данных и поддержке компании Kitware inc. стали доступны интерфейсы для других языков программирования, в том числе и для Python. Также в 2006 году было представлено книжное описание системы для языка программирования Java [11].

Выбор был сделан в её пользу по 2 причинам:

1. Возможность использовать её вместе с библиотекой NURBS-Python. Автор NURBS-Python предоставил для отображения аппроксимированной поверхности интерфейс VTK, который потребовалось незначительно отредактировать.
2. Повышенная производительность. В отличии от Matplotlib, VTK отображает результирующую поверхность намного отзывчивее и не требует стороннего ПО для отображения (веб — браузера), как того требует библиотека Plotly.

Рисунок 16 – Логотип VTK.

Основным компонентом, связывающим все блоки вместе является программное обеспечения для создания графического интерфейса Qt.

Разработанной в мае 1995 году Ховардом Нордом и Эйриком Чамбе-Энг. С его помощью можно поместить как виджет отображения аппроксимированной поверхности, так и виджеты данных для самой поверхности, которые можно редактировать в реальном времени. Также с помощью сигнально — слотной связи можно передавать команды как на чтение файлов контрольных точек, так и для сохранения итоговой поверхности. Поставляется данное программное обеспечение через интегрированную среду разработки Qt Creator, в которой использовались некоторые программы для выполнения данной работы. Конкретизируя, были использованы:

- Программа Qt Designer и для создания формы графического интерфейса без какого — либо взаимодействия с пользователем;
- утилита pyuic5 для конвертации .ui формы, созданной в Qt Designer, в .py файл исходного кода. Готовый для взаимодействия со всеми блоками программного обеспечения для аппроксимации поверхности.

Рисунок 17 – Логотип PyQt

Написание исходного кода было выполнено при помощи текстового редактора Visual Studio Code. Помимо того, что он является одним из самых популярных редакторов кода, также имеется возможность собрать из него IDE для разных языков программирования с помощью расширений. Также пригождается и привязка в системе контроля версии Git, с помощью которой можно публиковать исходный код своего программного обеспечения в создаваемый репозиторий.

Также прилагается простой файловый менеджер и подсветка синтаксиса, дающее дополнительное удобство при разработке ПО.

2.3. Создание графического интерфейса в Qt Design

Как и упоминалось ранее, создание графического интерфейса для программного обеспечения для аппроксимации поверхностей выполнялось с помощью программы Qt Designer. Qt Designer изначально работает формами формата XML в интерактивном режиме и сохраняет их в виде .ui файла.

После создания файла формы интерфейса, потребуется его использование в исходный код Python. Для этого потребовалась утилита pyuic5. Она транслирует формат XML, используемый в Qt Creator, в синтаксис исходного кода Python для дальнейшего взаимодействия блоков программного обеспечения с пользователем, использующего его, а также является связующим звеном между блоком отображения и редактирования поверхности и блоком аппроксимации поверхности.

Требуемый метод конвертации описан в листинге 1.

```
pyuic5 -x -o interface.py form.ui
```

Листинг 1. Команда преобразования файла XML формы из Qt Designer в файл кода Python

На рисунке ниже можно увидеть интерфейс программы Qt Designer внутри IDE Qt Creator с готовой формой для использования в разрабатываемом программном обеспечении.

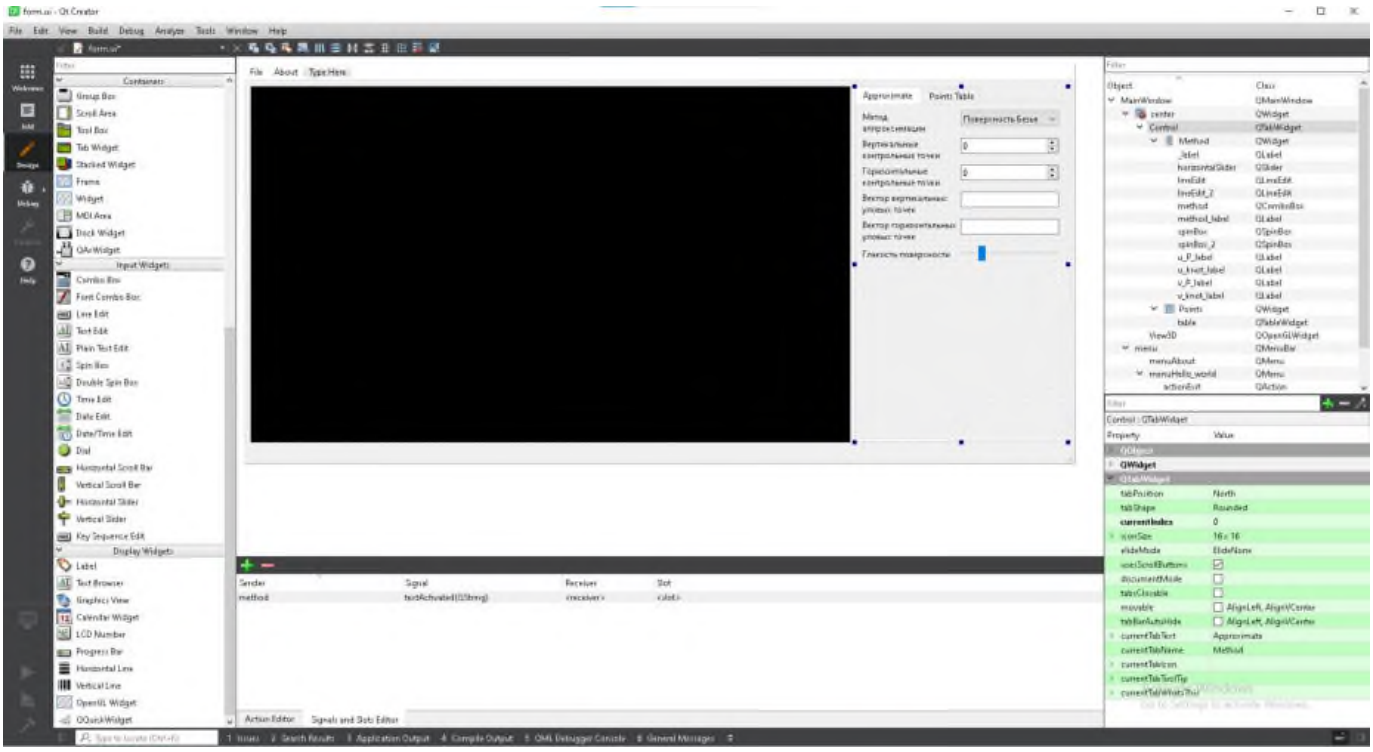

Рисунок 18 – Разработка интерфейса в IDE Qt Creator

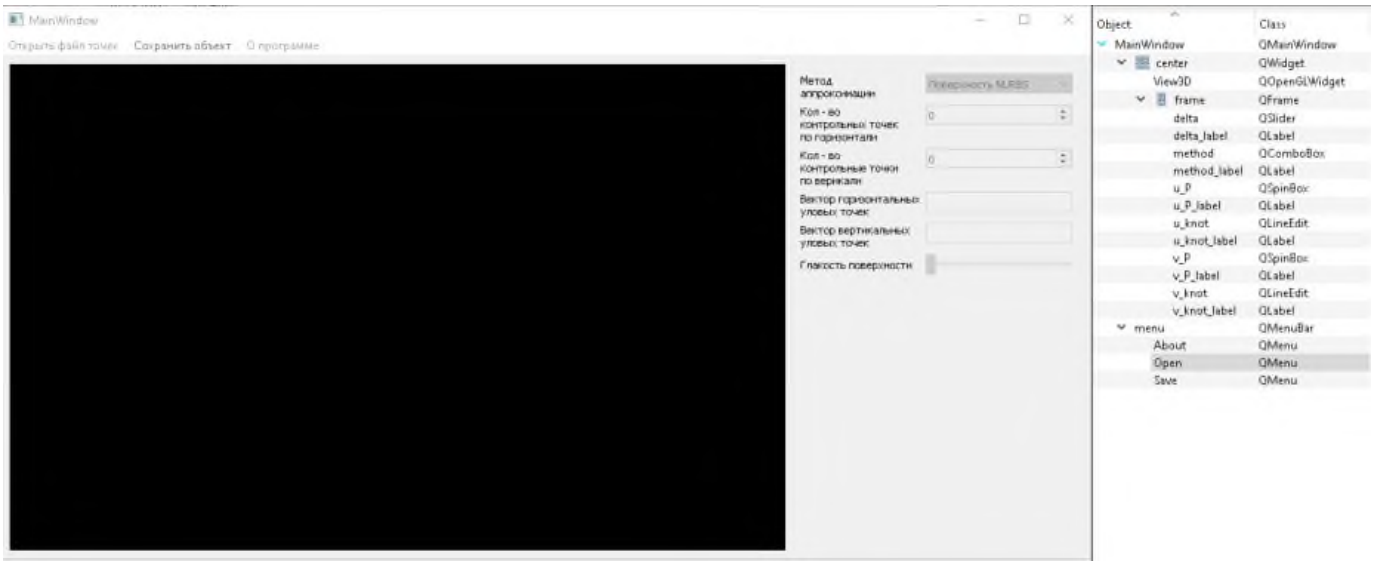

Рисунок 19 – Готовый шаблон интерфейса ПО для аппроксимации поверхности

Ниже приведена таблица основных элементов, которые будут необходимы для работы с аппроксимированной поверхностью. Названия всех виджетов расположено на рис.19.

Таблица 1 – Состав виджетов интерфейса

Название виджета в интерфейсе	Назначение виджета
Frame	Содержание параметров, управляющих аппроксимацией.
View3D	Общее название виджета для отображение аппроксимированной поверхности
Method	Выбор метода аппроксимации поверхности
u_P	Выбор количества строк для аппроксимации поверхности
v_P	Выбор количества столбцов для аппроксимации поверхности
u_knot	Изменение значений узлового вектора для всех точек в строке
v_knot	Изменение значений узлового вектора для всех точек в столбце
About	Вывод сообщения о программе
Open	Вывод диалогового окна с выбором файла облака точек
Save	Вывод диалогового окна для сохранения файла поверхности на диске

Примечание. Виджет View3D является виджетом отрисовки изображения OpenGL и будет заменен на виджет VTK для визуализации поверхности и её редактирования ввиду несколько большей простоты его использования.

2.4. Реализация алгоритмов, необходимых перед аппроксимацией

Для аппроксимации поверхностей первоначально требуются контрольные точки. Из условий с главы 1 получаем начальные данные в виде облака точек, но не известно, отмечены ли они так, чтобы построить контрольную сеть. В случае, если они описывают какую – либо поверхность, но не расположены в удобном для

алгоритмов аппроксимации виде, то получится несвязный геометрический объект вместо модели реального объекта. Для подготовки к аппроксимации поверхности, необходимо построить точки так чтобы получилась контрольная сетка

Способ построения контрольной сетки из данных линейных координат приводится ниже в следующем листинге:

```
def reader (f:str):
    val7=[];val8=[]
    for xyz in open(f,"r"):
        try:
            val=[float(coord) for coord in xyz.split()]
            val7.append(val)
            if val!=[] and val!=[[type(str)]]:
                val8.append(val)
        except(ValueError):
            continue
    new_list_ = [list(l) for i, l in groupby(val7, bool) if i]
    ready_list=val8
    return ready_list,new_list_,
```

Листинг 2 – Сбор (x, y, z) файлов сканеров

Полученная контрольная сетка должна получиться таким образом, каким она показана на рисунке 20.

	v0	v1	v2	v3	v4
u0	(x, y, z)	(x, y, z)	(x, y, z)	(x, y, z)	(x, y, z)
u1	(x, y, z)	(x, y, z)	(x, y, z)	(x, y, z)	(x, y, z)
u2	(x, y, z)	(x, y, z)	(x, y, z)	(x, y, z)	(x, y, z)

Рисунок 20 – Вид контрольной сетки

После получения контрольной сетки следующим обязательным набором данных будет узловой вектор, с помощью которого можно будет редактировать поверхностью, если того пожелает пользователь. В файлах наборов точек обычно не встречается никаких узловых векторов, следовательно, необходимо их сгенерировать. Для генерации узловых векторов потребуется сначала определить

количество контрольных точек по осям X и Y, но случае применения алгоритмов аппроксимации, по формулам из п.1.6. видно, что необходимо использовать сгенерированную контрольную сетку по координатам в параметрическом виде, т.е. в *u* и *v*. Соответственно, составим узловые векторы *U* и *V* соответственно.

Предположим, у нас есть $n+1$ параметров $t0, t1, \ldots, tn$ и степень p (которую может изменить пользователь, но в программном обеспечении она будет выставлена по умолчанию со значением, равным 2). Для B-сплайновой кривой степени p нам нужно $m+1$ узлов, где $m = n+p+1$ (в случае поверхности Безье, количество степеней равно количеству точек на оси). Если кривая зажата (что необходимо, исходя из свойств аппроксимации для каждого метода в п.1.6.), то эти узлы имеют вид:

$$u_0 = u_1 = \ldots = u_p = 0, u_{p+1}, \ldots, u_{m-p-1}, u_{m-p} = u_{m-p+1} = \ldots = u_m = 1.$$

Точнее, первые $p+1$ и последние $p+1$ узлы — это 0 и 1, соответственно. Что касается оставшихся $n-p$ узлов (если не выбран метод аппроксимации с помощью поверхности Безье), то они могут быть равномерно распределены или выбраны надлежащим образом для достижения некоторых желаемых условий.

Предположим, что оставшиеся $n-p$ внутренних узлов равномерно распределены. Тогда, $u_p = 0, u_{p+1}, \ldots, u_{m-p-1}, u_{m-p} = 1$ делят [0,1] на $n-p+1$ подынтервалов. Следовательно, узлы

$$u_0 = u_1 = \cdots = u_p = 0$$

$$u_{j+p} = \frac{j}{n-p+1} \text{ для } j = 1,2, \ldots n-p$$

$$u_{m-p} = u_{m-p+1} = \cdots = u_m = 1$$

При поверхности Безье:

$$u_0 = u_1 = \cdots = u_{m-p} = 0$$

$$u_{m-p} = u_{m-p+1} = \cdots = u_m = 1$$

Например, если у нас 6 ($n = 5$) параметров и степень $p = 3$, то мы должны найти $(n + p + 1) + 1 = (5 + 3 + 1) + 1 = 10$ узлов (т. е. $m = 9$). Поскольку мы используем зажатые кривые, первые и последние четыре (т. е. $p + 1$) узла должны быть равны 0 и 1, соответственно. Точнее, мы имеем

$$0, 0, 0, 0, u_4, u_5, 1, 1, 1, 1,$$

а два внутренних узла делят [0,1] на три подынтервала, каждый из которых имеет длину 1/3. Следовательно, вектор узлов имеет вид

$$\{0, 0, 0, 0, 1/3, 2/3, 1, 1, 1, 1\}.$$

Равномерно распределенный вектор узлов не требует знания параметров, и его очень просто генерировать. Однако этот метод не рекомендуется использовать.

Другой популярный метод генерации вектора узлов, предложенный де Буром, заключается в "усреднении" параметров. Вот формула расчета:

$$u_0 = u_1 = \cdots = u_p = 0$$

$$u_{j+p} = \frac{1}{p} \sum_{i=j}^{j+p-1} t\_i \text{ для } j = 1,2, \ldots n - p$$

$$u_{m-p} = u_{m-p+1} = \cdots = u_m = 1$$

Таким образом, первый внутренний узел является средним из p параметров $t_1, t_2, \ldots, t_p$; второй внутренний узел является средним из следующих p параметров, $t_2, t_3, \ldots, t_{p+1}$.

Таблица 2 – Пример использования алгоритма усреднения

t_0	t_1	t_2	t_3	t_4	t_5
0	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{3}{4}$	1

Предположим, нам нужно сгенерировать вектор для степени $p = 3$. Как упоминалось ранее, требуется 10 узлов $(m = 9)$. Из этих 10 первые четыре и последние четыре узла — это нулевой и единичный соответственно. Таким

образом, первый внутренний узел — это среднее значение параметров $\frac{1}{4}$, $\frac{1}{3}$ и $\frac{2}{3}$, а второй внутренний узел — это среднее значение следующих трех параметров $\frac{1}{3}$, $\frac{2}{3}$ и $\frac{3}{4}$. Вычисленный вектор узлов имеет вид

Таблица 3 – Пример использования алгоритма «усреднения» Де Бура

$t_0 = t_1 = t_2 = t_3$	t_4	t_5	$t_6 = t_7 = t_8 = t_9$
0	$\frac{\frac{1}{4}+\frac{1}{3}+\frac{2}{3}}{3} = \frac{5}{12}$	$\frac{\frac{1}{3}+\frac{2}{3}+\frac{3}{4}}{3} = \frac{7}{12}$	1

Следующая диаграмма иллюстрирует положение параметров и сгенерированных узлов. Обратите внимание, что 0(4) и 1(4) означают, что 0 и 1 являются четверными узлами (т.е. кратность = 4). Как видно, первый ненулевой узел $[0, \frac{5}{12})$ содержит два параметра, второй ненулевой узел $[\frac{5}{12}, \frac{7}{12})$ не содержит ни одного параметра, а третий ненулевой узел $[\frac{7}{12}, 1)$ содержит два параметра.

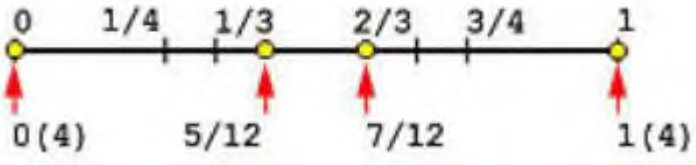

Рисунок 21 – Визуализация узловых точек алгоритма Де Бура

После сгенерированных узловых векторов по краям поверхностей, в случае с NURBS – поверхностями потребуется генерация весовых коэффициентов. В случае их отсутствия, можно будет использовать единичные значения весов, тем самым получая B-сплайновую поверхность. Позднее из можно будет использовать в качестве интерактивного элемента для редактирования поверхности [12].

Однако для генерации узлового вектора был использован несколько иной алгоритм, основанный на усредненном алгоритме.

На листинге ниже приведена его реализация, подробно описанная в приложении 1

2.5. Реализация алгоритмов аппроксимации поверхностей

В данном пункте буду выведены упрощенные алгоритмы реализации поверхности Безье, поверхности базисного сплайна. Для начала потребуется привести также листинги алгоритмов.

```
Входные данные: массив P[0:n] из n+1 точек и вещественное
число u в [0,1].
Получаемый результат: точка на кривой, C(u)
Тело функции:
    массив точек Q[0:n]
    от i:= 0 до n выполнить
        Q[i] := P[i]; // сохраняем входные данные
    от k:= 1 до n выполнить
        от i := 0 до n - k выполнить
            Q[i] := (1 - u)Q[i] + u Q[i + 1];
вернуть Q[0];
```

Листинг 4 – Обобщенный алгоритм Де Кастельжо для кривой Безье

```
Входные данные: контрольная сетка (m+1)x(n+1) и параметры
(u,v).
Получаемый результат: точка на поверхности p(u,v).
Тело  функции:
    от i := 0 до m выполнить
        начало:
        алгоритм де Кастельжо к i-ой строке контрольных
точек с v;
        вернуть точку qi(v);
        конец
    алгоритм де Кастельжо к q0(v), q1(v), ..., qm(v) с u;
вернуть точку p(u,v);
```

Листинг 5 – Построение поверхности Безье

Для построения поверхности базисных сплайнов потребуется алгоритм Кокса — де Бура. Однако, данный алгоритм также работает и при построении NURBS — поверхностей.

```
Входные данные: значение u
Получаемый результат: точка на кривой, p(u).
Тело программы
Если u лежит в [uk,uk + 1) и u!= uk: пусть h = p (т.е.
вставка u p раз) и s = 0;
Если u = uk и uk - узел кратности s:       пусть h = p - s
(т.е. вставка u p - s раз);
pk-s, pk-s-1, pk-s-2, ..., pk-p+1,pk-p в новый массив и
переименуйте их в pk-s,0, pk-s-1,0, pk-s-2,0, ..., pk-
p+1,0;
от r := 1 до h выполнить
     от i := k-p+r до k-s выполнить
          начало:
          Пусть ai,r = (u - ui) / ( ui+p-r+1 - ui )
          Пусть pi,r = (1 - ai,r) pi-1,r-1 + ai,r pi,r-1
          конец
pk-s,p-s - точка p(u).
```

Листинг 6 – Алгоритм Де Бура

```
Входные данные: контрольная сетка (m+1)x(n+1), векторы узлов в
направлениях u и v и (u,v);
Получаемый результат: точка на поверхности p(u,v);
Тело программы:
     Пусть u находится в [uc,uc+1); Пусть v находится в [vd,vd+1);
     Если u не равно uc, то пусть s будет нулем,
          иначе пусть s будет кратностью uc;
     Если v не равно vd, то пусть t будет нулем,
          иначе пусть t будет кратностью vd;
     от i := c-p до c-s выполнить
          начало
             алгоритм де Бура к контрольным точкам pi,d-q, pi,d-
q+1, ..., pi,d-t относительно v;
             вернуть qi(v);
          конец
алгоритм де Бура к точкам qc-p(v),qc-p+1(v), ... ,qc-s(v)
относительно u;
вернуть точку p(u,v);
```

Листинг 7 – Алгоритм построения поверхности B — сплайна

Для алгоритма построения Ненормированного рационального базисного сплайновой поверхности аналогичен принципу построения идентичной кривой. А для построения NURBS кривой потребуется всего – лишь умножить каждую контрольную точку на ее вес (являющийся только положительным числом), преобразуя кривую NURBS в четырехмерную B-сплайн кривую. Выполняем алгоритм де Бура на полученной четырехмерной B-сплайн кривой, а затем проецируем данную кривую обратно в трехмерное пространство, разделив компоненты координат для декартовой системы на четверную координату и сохранив четвертый компонент в качестве нового веса.

2.6. Разработка блока визуализации поверхности

Как упоминалось в п.2.2, в качестве основы для отображения аппроксимированной поверхности была использована программная система Visualization Toolkit. Отображение аппроксимированной поверхности сразу можно задать во внутреннем модуле библиотеки NURBS — Python, однако, данная реализация является ограниченной тем. что не имеет возможности управлять контрольными точками и весовыми коэффициентами. Однако, VTK является достаточно гибкой системой, чтобы её можно было реализовать внутри графического интерфейса.

Для возможного рассмотрения со всех сторон по умолчанию доступны следующие последовательности управления. Примечание: список приведенных команд действует только при английской раскладке клавиатуры.

- Поворот объекта по перемещению мыши, удерживая её левую кнопку
- Приближение объекта по перемещению мыши вверх — вниз или же прокручивании колеса вперед — назад;
- Изменение заднего фона пространства, на котором отрисована поверхность по нажатию кнопки «B»;

- Выбор объекта через наведение курсором мыши на сам объект и нажатием кнопки «P»;
- Выход из приложения по нажатию кнопки «E»;
- Включение вида отображения проволочного каркаса при активном виде поверхности для объекта по нажатию кнопки «W»;
- Включение вида отображения поверхности при активном виде отображения проволочным каркасом для объекта по нажатию кнопки «S»;
- Перемещение к точке через наведение курсором мыши на точку и нажатием кнопки «F»;
- Сброс приближения камеры по нажатию кнопки «R»
- Смена цвета для выбранного объекта по нажатию «M»;
- Смена видимости для выбранного объекта по нажатию «H»;
- Сброс параметров видимости для всех объектов по нажатию «N»;
- Перемещение модели по осям по нажатию кнопок — стрелок;

В автоматическом режиме выполняется отображение нескольких аппроксимированных поверхностей соединяя их между собой, дабы получить реконструкцию реального объекта.

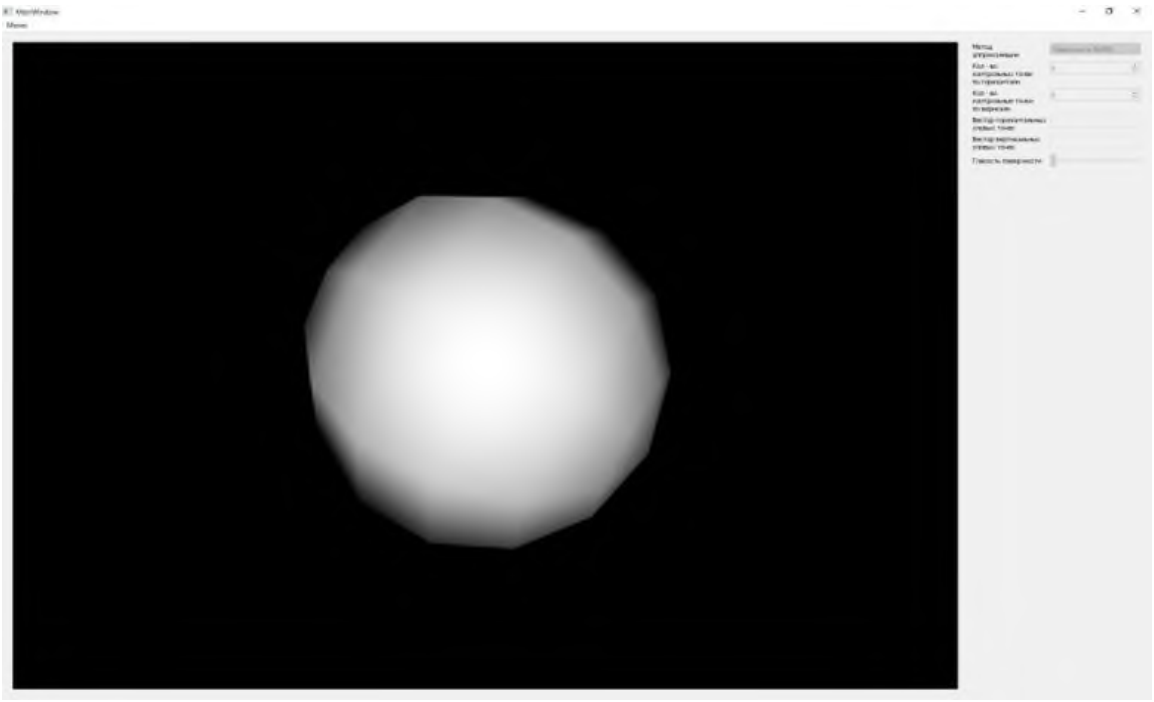

Рисунок 22 – Пример работы VTK внутри итоговой программы

Для получения возможности управления контрольными точками, воспользуемся одним из примеров на официальном сайте курсов по использованию VTK [13]. Конкретнее, потребовался пример ContourWidget, реализованный на Python. Краткое описание. В данном примере создается набор точек, лежащих на окружности, и контур через эти точки. Этот контур можно интерактивно изменять, перетаскивая контрольные точки.

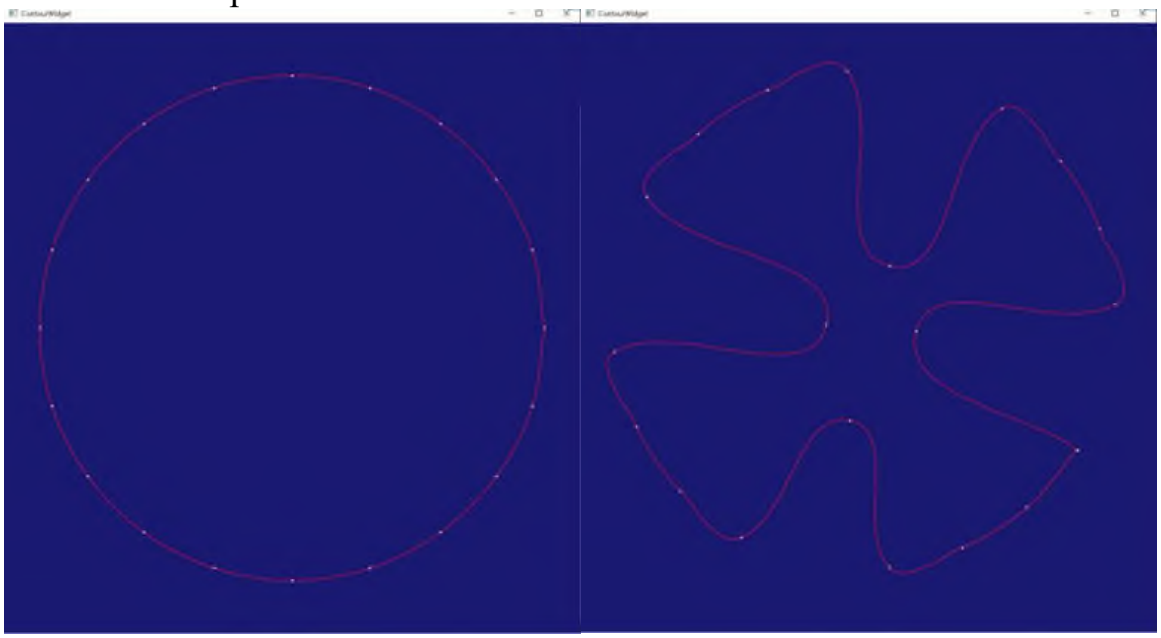

Рисунок 23 – Интерактивная линия: a – неизмененная; b – измененная.

Несмотря на то, что можно сбросить расстояние от камеры до объекта, было бы неплохо иметь возможность для опознания положения камеры в пространстве. Точнее говоря, хорошо бы знать, с какой стороны пользователь смотрит на аппроксимированную поверхность.

С данной задачей справляется пример объекта vtkCameraOrientationWidget, используемый также в примерах на рисунке ниже.

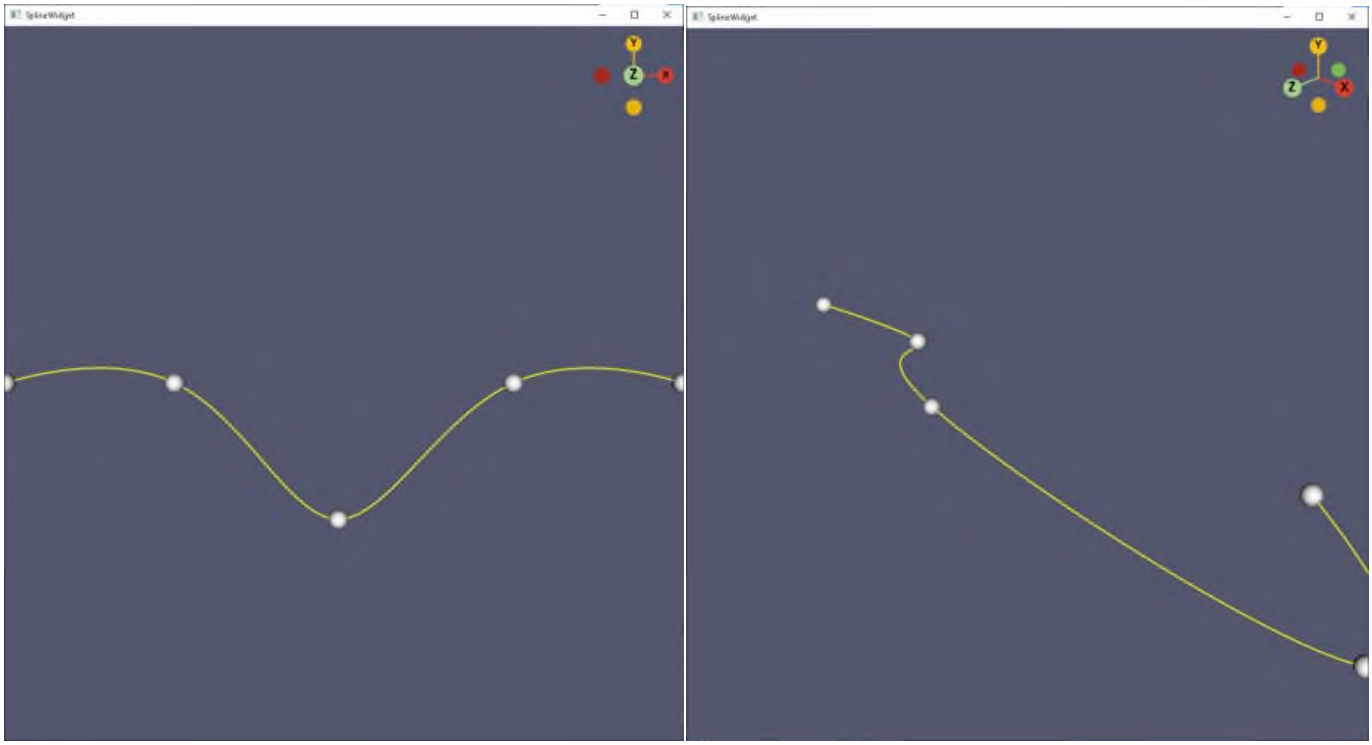

Рисунок 24 – Пример использования ориентировочной оси и перемещения контрольных точек

Данные, проиллюстрированные выше, примеры позволяют реализовать как управление контрольными точкам, так и ориентированность в пространстве. Остается применит их вне примеров для программного обеспечения, требующееся для данной выпускной квалификационной работы.

Прежде чем использовать блок отображения в графический интерфейс, его потребуется заменить вместо QtOpenGL виджета, используя данный листинг, описанный в приложении 2.

После замены виджета можно будет приступать к переносу данных об аппроксимированной поверхности на виджет QVTKRenderWindowInteractor для отображения внутри программного обеспечения в интерактивном режиме.

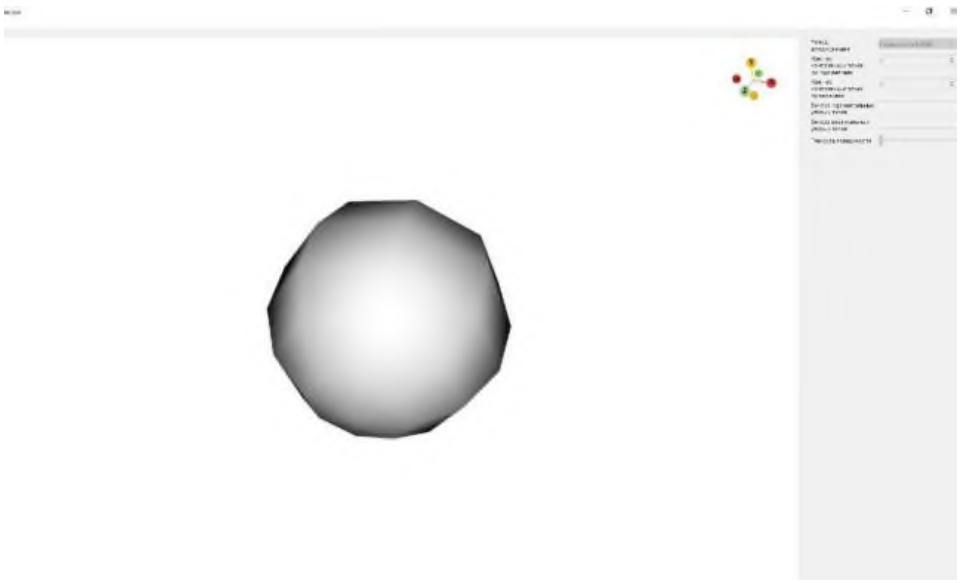

Рисунок 25 – Снимок вращаемой сферы, аппроксимированной по точкам

Далее после вызова из меню действия открытия файла точек можно увидеть поверхности, аппроксимированные на основе разделенных наборов точек.

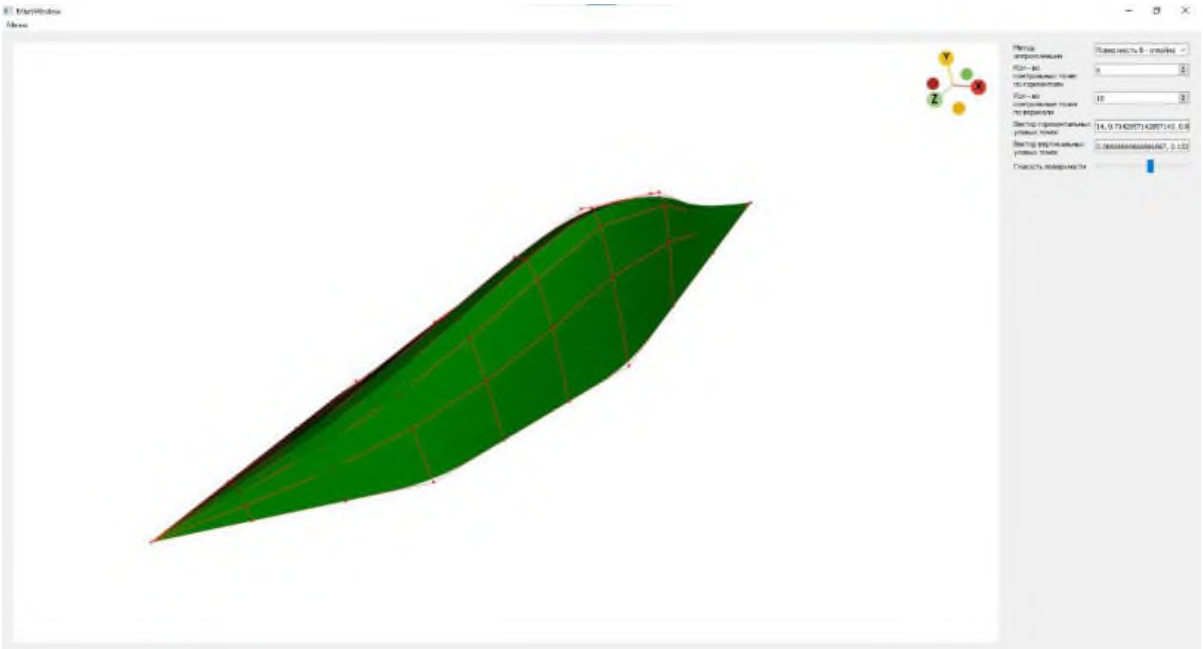

Рисунок 26 – Аппроксимированная поверхность хвостового плавника кита

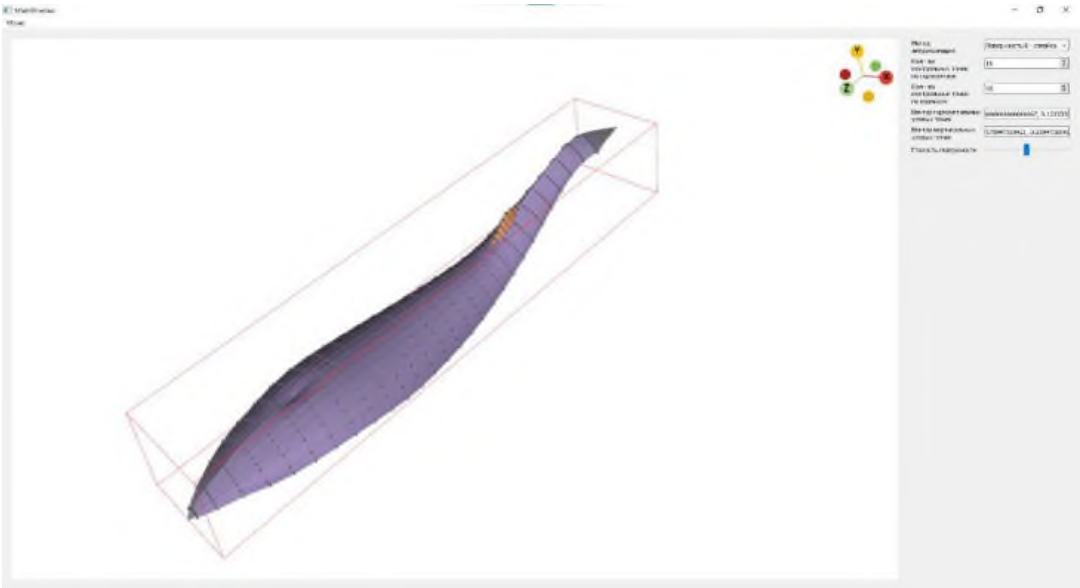

Рисунок 27 – Аппроксимированные поверхности спины кита

Соединяются кусочно-гладкие поверхности посредством специального контейнера, который способен содержать все возможные поверхности. Обычно такая функция пригождается, когда данные с координатами контрольных точек сильно разделены

В дальнейшей разработке после первого выпуска программного обеспечения может потребоваться множественный выбор точек из разных объектов аппроксимированных поверхностей.

Рисунок 28 – Изменение положения контрольной точки поверхности

2.7. Сборка программного обеспечение в монолитное приложение

После того, как было разработано программное обеспечение для аппроксимации поверхностей, стоит задуматься о его методе его распространения. В данный момент лидирующую позиция в мире операционных систем занимает семейство операционных систем Windows. Однако, не уступают в конкуренции и операционные системы семейства GNU/Linux. Тем не менее одним из популярным, среди среднестатистических пользователей персонального компьютера, видом распространения программ является установочные пакеты. Однако, необходимости, а порой и времени, чтобы установить тот или иной продукт, поэтому порой прибегают к одиночным исполняемым файлам. Благодаря обширному сообществу Python – разработчиков, были также созданы и специальные модули, позволяющие конвертировать файлы исходного кода данного языка в исполняемые файл.

Для данной работы нам потребуются следующие загружаемые модули:

- Pyinstaller
- Auto-py-to-exe

Первый упомянутый модуль нам пригодится для самой конвертации. Также пользователям GNU/Linux чаще всего удобно работать с программами через эмулятор терминала. Второй модуль является аналогичной реализацией с графическим веб – интерфейсом (что удобно для пользователей Windows) для создания исполняемых файлов через pyinstaller. С помощью данного программного обеспечения и будут создаваться исполняемые файлы. Итоговой реализацией будет приложение под Windows, однако можно будет использовать исходный код программы, помещённый в репозиторий на GitHub для конвертации на другие операционные системы, где возможна установка Python. Чтобы конвертировать

программное обеспечение в исполняемый файл из исходных кодов в репозитории программы, необходимы следующие компоненты:

- Python 3.9.17
- Numpy1.19.5
- NURBS – Python 5.3.1
- PyQt 5.15.9
- VTK 9.2.6

После установки как требуемых выше в списке включая pyinstall, так и необязательного, но удобного в использовании auto-py-to-exe, с помощью пакетного менеджера pip или аналогичного ему продукта с графическим интерфейсом Anaconda, можно приступить к сборке исполняемого файла. Именно в нем и будет храниться все необходимое.

Для начала работы с auto-py-to-exe необходимо открыть командную строку и ввести название самой программы без каких – либо дополнительных параметров.

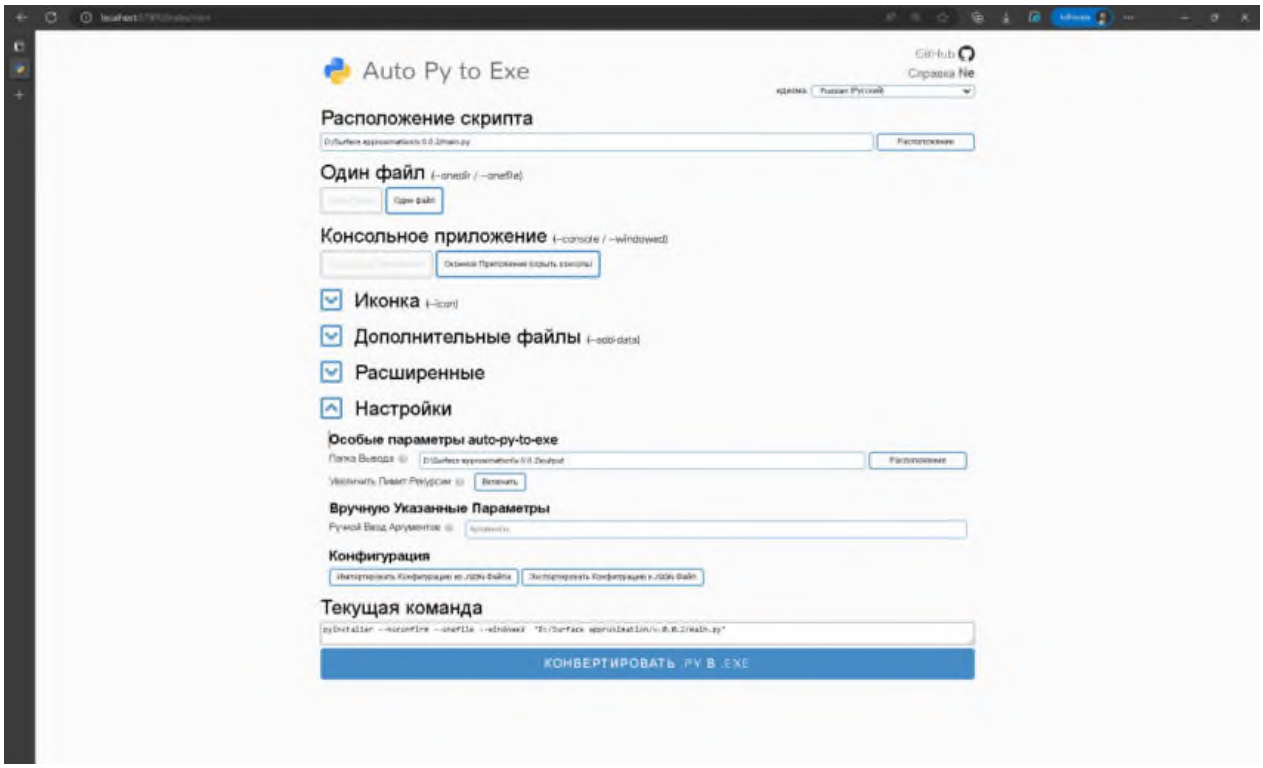

Рисунок 29 – Интерфейс программы Auto-py-to-exe

Вам откроется веб – интерфейс приложения, выполненный с достаточно минимальным количеством параметров для выполнения конвертации.

Рассмотрим его чуть более подробно по порядку:

- «Расположение скрипта» – поле для ввода пути к скрипту, который необходимо конвертировать;
- Параметр конвертации в один файл или в одну директорию. Параметр одного файла удобен при скриптах малого объема и с минимальным количеством подключаемых модулей, но значительно замедляется конвертация при наличии обширных библиотек, используемых в коде.
- Параметр отображения консоли. «Консольное приложение» подойдет для программы, предназначенной для работы с терминалом или командной строкой. «Оконное приложение (скрыть консоль)» предназначено для приложений с настроенным графическим интерфейсом.
- «Иконка» – позволяет установить свою иконку для программы. Вместо стандартной.
- «Дополнительные файлы» – позволяет конвертировать выбранные файлы вместе с основным скриптом в случае, если необходимы примерные данные, не подключаемые к основному скрипту с помощью import.
- «Расширенные» – позволяют глубже отслеживать ошибки в конвертированном приложении, задавать параметры, зависимые т операционные системы. И многие другие параметры для поиска внутри приложения, а также некоторые настройки самого конвертера.
- «Настройки» – в данном списке доступен путь результирующего скрипта, возможность ограничения лимита на рекурсию, а также возможность конфигурирования pyinstaller специфически необходимым для разработчика образом. Такие же настройки можно сохранить, экспортируя их в текстовый JSON файл или наоборот, использовать другие настройки, уже импортируя JSON файл.

- «Текущая команда» – окно для отображения аналогичной настройки auto–py–to–exe в среде выполнения программы pyinstaller.

Конвертировав исходный код программного обеспечения для аппроксимации поверхностей, были получены 1 файл и 1 директория. Оба файла представляют собой одно и то же программное обеспечение. Однако, в директории находятся все библиотеки, установленные в среде программирования в которой пользователь конвертирует программное обеспечение. Но и есть 1 сильная сторона при конвертации в директорию для запуска приложения – скорость работы.

По сравнению с монолитным исполнительным файлом, директория оказывается намного быстрей, что позволяет проще использовать программное обеспечение несколько раз, в случае повторного использования данного программного обеспечения.

Рассматривать подробно консольную программу pyinstaller не представляется целесообразным. Поскольку auto–py–to–exe способен в полной мере использовать все его свойства.

Итогом данной конвертации, спустя чуть более продолжительное время, по сравнению с обычным запуском программного кода в любой Python – поддерживаемой интегрированной средой разработки, мы получаем независимое от внешней среды приложение для аппроксимации поверхностей, которое можно отправить в ветку выпуска на GitHub – репозитории.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе была рассмотрена задача разработки программного обеспечения для аппроксимации поверхностей.

В качестве результата выполнения данной работы была достигнута следующая цель: было разработано программное обеспечение для аппроксимации поверхности на высокоуровневом языке программирования общего назначения Python.

Данная цель была выполнена благодаря поочередному выполнению нижеперечисленных основных задач:

1. Были изучены основные методы аппроксимации поверхностей, а также методы их построения.
2. Поставлена задача аппроксимации поверхностей исходя из набора трехмерных координат евклидовом пространстве, содержащихся в текстовом файле в виде массива координат на каждую из осей.
3. Программное обеспечение было разработано на основе языка программирования Python с применением дополнительных модулей различной направленности и связано по смыслу для удобной работы с ним для пользователя.
4. Разработанное программное обеспечение в виде директории с исполняемым файлом и связанными дополнительными модулями было протестировано и апробировано, а результирующие объекты от данного приложения можно использовать в других, более популярных системах автоматизированного проектирования и компьютерной графики.

В результате разработки вышеописанного программного обеспечения учитывались все необходимые детали, прямо или косвенно связанные с получением и применением исходными данных, взаимодействия с пользователем, применяющее данное программное обеспечение, и взаимодействием с другим программным обеспечением, связанным с компьютерной графикой. Данная выпускная

квалификационная работа предоставляет практический смысл в удобстве работы, по сравнению с другими CAD или CGI системами. Сравнение приводится здесь в необходимости изучать программное обеспечение, в чем выигрывает данная работа.

Разработанное программное обеспечение в качестве результата данной выпускной квалификационной работы доступно по ссылке: https://github.com/artmen57/Graduate-work-App-development-for-surface-approximation

СПИСОК ЛИТЕРАТУРЫ

1. Аширбакиев Р. И. Аппроксимация поверхности переходного отверстия печатной платы ортогональными прямоугольниками для вычисления емкости / Р. И. Аширбакиев, И. Ф. Калимулин, О. М. Кузнецова-Таджибаева // Доклады ТУСУР. – 2013. – № 4(30). – С. 58–61.
2. A three-dimensional freeform blading tool based on NURBS curves and surfaces implemented in Python. / L. Weihua, X Shenren. – http://dx.doi.org/10.33737/gpps21-tc-91 // Proceedings of Global Power and Propulsion Society – 2021. – URL: https://www.researchgate.net/publication/364397168_A_three-dimensional_freeform_blading_tool_based_on_NURBS_curves_and_surfaces_implemented_in_Python (date accessed: 10.06.2023).
3. Быковский, С. МОБИЛЬНЫЕ СИСТЕМЫ 3D СКАНИРОВАНИЯ СРЕДНЕГО РАДИУСА ДЕЙСТВИЯ / Быковский С., Кустарев П., Дидин Е., Громов В., Драница А. // Control Engineering – 2018 – URL: https://controleng.ru/wp-content/uploads/7334.pdf – Дата доступа: 10.06.2023.
4. PolyCam: Система фотограмметрии [Для создания моделей с помощью фотограмметрии] - Режим доступа: https://poly.cam/.- Дата доступа:10.06.2023.
5. Пигль, Л. «The NURBS Book» / Л. Пигль, У. Тиллер; – 2-е изд., Springer-Verlag Berlin Heidelberg 1995 – 649 с. – ISBN 978-3-642-97385-7.
6. «Технология построения 3D-моделей объектов по набору изображений» / Хабр. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/143094/. – Дата доступа: 10.06.2023.
7. «Stack Overflow Developer Survey 2022,» / Stack Exchange Inc. [Электронный ресурс]. – Режим дотсупа: https://survey.stackoverflow.co/2022#most-popular-technologies-language. – Дата доступа: 10.6.2023.

8. «Splipy» / GitHub [Электронный ресурс]. – Режим доступа: https://github.com/SINTEF/Splipy. – Дата доступа: 10.6.2023.
9. «nurbspy» / GitHub. [Электронный ресурс]. – Режим доступа: https://github.com/RoberAgro/nurbspy. – Дата доступа: 10.6.2023.
10. Bingol, O.R., «NURBS-Python» / GitHub [Электронный ресурс]. – Режим доступа: https://github.com/orbingol/NURBS-Python. – Дата доступа: 10.06.2023.
11. Шредер У., The Visualization Toolkit: An Object-oriented Approach to 3D Graphics / Шредер Уильям Дж., Мартин К. М., Лоренсен У. Е., – 4-е изд. – Калифорния, США: Kitware, 2006. – 528 с. – ISBN 1-930934-19-X.
12. Shene, C.-K., «CS3621 Introduction to Computing with Geometry Notes». [Электронный ресурс]. – Режим доступа: https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/. – Дата доступа: 10.06.2023.
13. VTK Python Examples: [Электронный ресурс]. – Режим доступа: examples.vtk.org/site/Python/. – Дата доступа: 10.06.2023.
14. Python 3: [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/. – Дата доступа: 10.06.2022.
15. Qt for Python Documentation: [Электронный ресурс]. – Режим доступа: https://doc.qt.io/qtforpython-6/ – Дата доступа: 10.06.2022.
16. Тросиненко, А., Компьютерное моделирование / А. Тросиненко, // численные методы моделирования физических систем. – 2018. [Электронный ресурс]. Режим доступа: https://cc.dvfu.ru/ru/компьютерное-моделирование/. – Дата доступа: 10.06.2023.
17. Salomon, D. Curves and Surfaces for Computer Graphics / D. Salomon; New York: Springer-Verlag, 2006. – 460 с. – ISBN 978-0-387-28452-1.
18. Rogers, D. F., An introduction to NURBS With Historical Perspective / D. F. Rogers; Annapolis: Elsevier Science, 2000. – 324 с – ISBN 9781558606692.

19. Bingol, O.R. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python / O. R. Bingol, A. Krishnamurthy // SoftwareX Original software publication| volume 9, P85-94, 2019 – Режим доступа: https://www.softxjournal.com/article/S2352-7110(18)30177-8/fulltext. – Дата доступа: 10.06.2023.
20. Шикин Е.В. Кривые и поверхности на экране компьютера. Руководство по сплайнам для пользователей / Е.В. Шикин, А.И. Плис; - Москва; Диалог-МИФИ, – 1996. – 240 с.

Приложение 1

Алгоритм генерации узлового вектора

```
def linspace(start, stop, num, decimals=18):
    """ Returns a list of evenly spaced numbers over a specified
interval. Inspired from Numpy's linspace function:
    :param start(float): starting value; stop(float): end value;
num(int): number of samples to generate; decimals(int): number of
significands;
    :return: a list of equally spaced numbers; rtype: list """
    start = float(start); stop = float(stop)
    if abs(start - stop) <= 10e-8:
        return [start]
    num = int(num)
    if num > 1:
        div = num - 1; delta = stop - start
return[float(("{:."+str(decimals)+"f}").format((start+(float(x)*flo
at(delta)/float(div)))))
for x in range(num)]
    return [float(("{:." + str(decimals) + "f}").format(start))]

def generate(degree, num_ctrlpts, **kwargs):
""" Generates an equally spaced knot vector. It uses the following
equality to generate knot vector:`m = n + p + 1`where;`p`, degree;
:math:`n + 1`, number of control points;:`m + 1`, number of knots
Keyword Arguments:
* ``clamped``: Flag to choose from clamped or unclamped knot vector
options. *Default: True*
:param degree(int): degree; num_ctrlpts(int): number of control
points;
:return: knot vector; rtype: list """
    if degree == 0 or num_ctrlpts == 0: raise ValueError("Input values
should be different than zero.")
    clamped = kwargs.get('clamped', True)    # Get keyword arguments
    num_repeat = degree   # Number of repetitions at the start and
end of the array
    num_segments = num_ctrlpts - (degree + 1)     # Number of knots
in the middle
    if not clamped:
        num_repeat = 0        # No repetitions at the start and end
        num_segments = degree + num_ctrlpts - 1 # Should conform the
rule: m = n + p + 1
    knot_vector = [0.0 for _ in range(0, num_repeat)] # First knots
    knot_vector += linspace(0.0, 1.0, num_segments + 2)    # Middle
knots
    knot_vector += [1.0 for _ in range(0, num_repeat)] # Last knots
    return knot_vector     # Return auto-generated knot vector
```

Приложение 2

Фрагмент программного кода для замены OpenGL виджета на VTK виджет рендера с тестовой фигурой

```
class VTK_Render(QtWidgets.QMainWindow):
    def __init__(self, parent = None):
        QtWidgets.QMainWindow.__init__(self, parent)
        colors = vtk.vtkNamedColors()
        colors.SetColor('ParaViewBkg', [255, 255, 255, 255])
        self.frame = QtWidgets.QFrame()
        self.vl = QtWidgets.QVBoxLayout()
        self.vtkWidget = QVTKRenderWindowInteractor(self.frame)
        self.vl.addWidget(self.vtkWidget)
        self.ren = vtk.vtkRenderer()
        self.ren.SetBackground(colors.GetColor3d('ParaViewBkg'))
        cam_orient_manipulator = vtk.vtkCameraOrientationWidget()
        cam_orient_manipulator.SetParentRenderer(self.ren)
        # Enable the widget.
        cam_orient_manipulator.On
        self.vtkWidget.GetRenderWindow().AddRenderer(self.ren)
        self.iren = self.vtkWidget.GetRenderWindow().GetInteractor()
        mapper = vtk.vtkPolyDataMapper()
        # Create sourcecв
        source = vtk.vtkSphereSource()
        source.SetCenter(0, 0, 0)
        source.SetRadius(5.0)
        mapper.SetInputConnection(source.GetOutputPort())
        actor = vtk.vtkActor()
        actor.SetMapper(mapper)
        self.ren.AddActor(actor)
        self.ren.ResetCamera()
        #self.iren.SetRenderWindow(self.ren)
        # Create an actor
        self.frame.setLayout(self.vl)
        self.setCentralWidget(self.frame)
        self.iren.Initialize()
        self.iren.Start()
…
class MainWindow(QtWidgets.QMainWindow):
    def __init__(self,*args, **kwargs):
        """ Initialize Main Window \n
            & create connection signals """
        super().__init__(*args, **kwargs)
        self.ui=Ui_MainWindow()
        self.ui.setupUi(self)
        self.ui.View3D.close()
        self.vtkWidget=VTK_Render()
```

```
        self.ui.horizontalLayout.insertWidget(0,self.vtkWidget)
        self.ui.actOpen.triggered.connect(self.Open_file)
        self.ui.actExit.triggered.connect(self.closeEvent)
        timer = QtCore.QTimer(self)
        timer.setInterval(20)   # period, in milliseconds
        #timer.timeout.connect(self.ui.View3D.updateGL)
        #timer.start()
        self.show()
…
```

РЕЦЕНЗИЯ

на выпускную квалификационную работу
Ващенко Артема Тарасовича

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АППРОКСИМАЦИИ ПОВЕРХНОСТЕЙ

представленную к защите
по направлению подготовки 01.03.02 Прикладная математика и информатика
направленность (профиль) Системное программирование и компьютерные технологии

В выпускной квалификационной работе изучается задача разработки и применения программного обеспечения для аппроксимации поверхностей. Эти задачи используются для моделирования различных процессов, проектирования и конструирования компьютерных моделей.

В первой главе подробно описывается анализ состояния данной темы и предлагаемые способы её решения с теоретической точки зрения, на основе изученной литературы и научных статей в сети Интернет. Вторая глава данной работы представляет из себя проведение исследования темы на основе практического применения теоретической информации с указанием результатов этого исследования. Рассматриваются различные модели аппроксимации.

В работе разработано программное обеспечение для решения задачи аппроксимации поверхностей.

Содержание ВКР соответствует теме и цели исследования. Тема разработана достаточно полно и качественно. Полученные по результатам исследования практические выводы являются обоснованными и достоверными.

Автор продемонстрировал умение работать с информационными источниками, т.к. были проанализированы и систематизированы публикации по данной теме.

Работа оформлена в соответствии с требованиями к оформлению ВКР. В процессе рецензирования было сделано существенное количество замечаний, которые автор учел и исправил. Однако в работе имеются некоторые недоработки в представленном материале, а именно, нарушена логика изложения разделов в главе 1.

Работа выполнена на хорошем уровне и соответствует требованиям, предъявляемым Федеральным государственным образовательным стандартом высшего образования по направлению подготовки 01.03.02 Прикладная математика и информатика (квалификация (степень) бакалавр) к выпускным квалификационным работам.

Считаю, что ВКР заслуживает оценки «хорошо».

Рецензент работы, к.ф.-м.н., доцент В.И.Дорофеева
26.06.2023

АНТИПЛАГИАТ
ОБНАРУЖЕНИЕ ЗАИМСТВОВАНИЙ

СПРАВКА

о результатах проверки текстового документа
на наличие заимствований

Орловский государственный университет
имени И.С. Тургенева

ПРОВЕРКА ВЫПОЛНЕНА В СИСТЕМЕ АНТИПЛАГИАТ.ВУЗ

Автор работы: Ващенко Артем Тарасович
Самоцитирование рассчитано для: Ващенко Артем Тарасович
Название работы: РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ АППРОКСИМАЦИИ ПОВЕРХНОСТЕЙ
Тип работы: Выпускная квалификационная работа
Подразделение: кафедра информатики, физико-математический факультет

РЕЗУЛЬТАТЫ

■ **ОТЧЕТ О ПРОВЕРКЕ КОРРЕКТИРОВАЛСЯ:** НИЖЕ ПРЕДСТАВЛЕНЫ РЕЗУЛЬТАТЫ ПРОВЕРКИ ДО КОРРЕКТИРОВКИ

СОВПАДЕНИЯ	2.48%	СОВПАДЕНИЯ	2.48%
ОРИГИНАЛЬНОСТЬ	97.52%	ОРИГИНАЛЬНОСТЬ	97.52%
ЦИТИРОВАНИЯ	0%	ЦИТИРОВАНИЯ	0%
САМОЦИТИРОВАНИЯ	0%	САМОЦИТИРОВАНИЯ	0%

ДАТА ПОСЛЕДНЕЙ ПРОВЕРКИ: 29.06.2023 **ДАТА И ВРЕМЯ КОРРЕКТИРОВКИ:** 29.06.2023 09:33

Структура документа: Проверенные разделы: основная часть с.5, 7-62

Модули поиска: ИПС Адилет; Библиография; Сводная коллекция ЭБС; Интернет Плюс*; Сводная коллекция РГБ; Цитирование; Переводные заимствования (RuEn); Переводные заимствования по eLIBRARY.RU (EnRu); Переводные заимствования по коллекции Гарант: аналитика; Переводные заимствования по коллекции Интернет в английском сегменте; Переводные заимствования по Интернету (EnRu); Переводные заимствования по коллекции Интернет в русском сегменте; Переводные заимствования издательства Wiley ; eLIBRARY.RU; СПС ГАРАНТ: аналитика; СПС ГАРАНТ: нормативно-правовая документация; Медицина; Диссертации НББ; Коллекция НБУ; Перефразирования по eLIBRARY.RU; Перефразирования по СПС ГАРАНТ: аналитика; Перефразирования по Интернету; Перефразирования по Интернету (EN); Перефразированные

Работу проверил: Черкасова Владлена Владиславовна
ФИО проверяющего

Дата подписи: 29.06.2023г.

Подпись проверяющего

Чтобы убедиться в подлинности справки, используйте QR-код, который содержит ссылку на отчет.

Ответ на вопрос, является ли обнаруженное заимствование корректным, система оставляет на усмотрение проверяющего. Предоставленная информация не подлежит использованию в коммерческих целях.