ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

**Отчет о прохождении производственной практики**

**(научно-исследовательской работы)**

|  |
| --- |
| Карандиной Анны Ивановны |

*(Ф.И.О. обучающегося)*

|  |
| --- |
| 4 курс, гр. 3530901/90202 |

|  |
| --- |
| 09.03.01 «Информатика и вычислительная техника» |

*(Направление подготовки (код и наименование)*

|  |
| --- |
| **Место прохождения практики**: Высшая школа интеллектуальных систем и суперкомпьютерных технологий (ВШИСиСТ) ИКНТ ФГАОУ ВО «СПбПУ». |

|  |
| --- |
| **Сроки практики:** с 02 февраля по 14 апреля 2023 г. |

|  |
| --- |
| **Руководитель практики (НИР):** |

|  |
| --- |
| Цыган В.Н., доцент ВШИСиСТ ИКНТ |

*(Ф.И.О., уч.степень, должность)*

|  |
| --- |
| **Оценка (зачет): \_\_\_\_\_\_\_\_\_\_\_** |

|  |
| --- |
| Руководитель практики (НИР): /Цыган В.Н. / |

|  |
| --- |
| Обучающийся: /Карандина А.И./ |

|  |
| --- |
| Дата: |

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

**ИНДИВИДУАЛЬНЫЙ ПЛАН (ЗАДАНИЕ И ГРАФИК)**

**ПРОВЕДЕНИЯ ПРАКТИКИ**

|  |
| --- |
| Ф.И.О. обучающегося Карандина Анна Ивановна |

|  |
| --- |
| **Направление подготовки** (код/наименование): 09.03.01, «Информатика и вычислительная техника» |
| **Профиль** (код/наименование): 09.03.01\_02, «Технологии разработки программного обеспечения» |
| **Вид практики:** производственная |
| **Тип практики:** научно-исследовательская работа |
| **Место прохождения практики**: Высшая школа интеллектуальных систем и суперкомпьютерных технологий (ВШИСиСТ) ИКНТ ФГАОУ ВО «СПбПУ». При необходимости руководителями практики обеспечивается организация практики на базе университета с использованием электронного обучения и дистанционных образовательных технологий. |

|  |
| --- |
|  |
| Руководитель НИР: Цыган В.Н., доцент ВШИСиСТ ИКНТ |
| *(Ф.И.О., уч.степень, должность)* |

**Рабочий график проведения практики (НИР)**

Сроки практики: с 02.02.2023 г. по 14.04.2023 г.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Этапы (периоды) практики | Вид работ | Сроки прохождения этапа (периода) практики |
| 1 | Организационный этап | Установочная лекция для разъяснения целей, задач, содержания и порядка прохождения практики, выдача сопроводительных документов по практике  Основная тема: Разработка программного модуля для автоматической генерации 3D-моделей корпусов для устройств СКАРА. | 02.02.2023 г. |
| 2 | Основной  этап | Сбор информации, обработка, систематизация и анализ фактического и теоретического материала.  Содержание практики: (Изучение предметной области, обзор существующих технологий и аналогов по теме НИР, выбор инструментальных средств для разработки программного модуля, выбор метода создания 3D-модели (топологическая оптимизация и /или генеративный дизайн), разработка программного модуля, а именно инициализация плагина и обработка данных от пользователя).  Планируемые результаты прохождения практики: программный модуль | 03.02.2023 - 15.04.2023 г. |
| 3 | Заключительный этап | Подготовка отчета | 10.04.- 12.04.2023 г. |
| Защита отчета по практике (зачет) | 13.04 - 14.04.2023 г |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Карандина А.И

Руководитель НИР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Цыган В.Н.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| РЕФЕРАТ………...……………………………………………………………... | 5 |
| ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ……………….. | 6 |
| ВВЕДЕНИЕ………...……………………………………………………………...  Глава 1. Технологии создания 3d-моделей…………………………………….   * 1. Обзор существующих технологий создания 3d-моделей..................   2. Понятие топологической оптимизации………………………….......   3. Поиск и анализ существующих решений автоматической генерации 3d-моделей………………………………………………………………   Глава 2. Проектирование программного модуля……………………………….   * 1. . Выбор ПО с модулем топологической оптимизации…………………..   2.2. Разработка принципов и плана решения………………………………...  Глава 3. Разработка программного модуля………………………………………  3.1. Создание плагина для выбранного ПО…………………………………..  3.2. Обработка входных данных………………………………………………  ЗАКЛЮЧЕНИЕ…………….………………………………………………...…...СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ……………….…………….  ПРИЛОЖЕНИЕ……..………………………........................................................ | 7  10  10  12  16  18  18  20  24  24  27  30  31  32 |

**РЕФЕРАТ (Проект)**

3D-МОДЕЛЬ, АВТОМАТИЗАЦИЯ, ГЕНЕРАТИВНЫЙ ДИЗАЙН, ПЛАГИН, API, ДАТЧИКИ, КОРПУС, ПРОЦЕСС ГЕНЕРАТИВНОГО ДИЗАЙНА

В данной работе была изучена информация о технологии проектирования 3d-моделей с использованием методов топологической оптимизации. На основе полученной информации был разработан программный модуль для автоматической генерации 3d-моделей корпусов для устройств СКАРА. Для этого использовалась API Ansys Discovery. Программный модуль выполняет обработку пользовательских данных, построение геометрии, наложение физических характеристик и запуск процесса топологической оптимизации.

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ И ТЕРМИНОВ**

* API (**application programming interface)** - описание способов взаимодействия одной компьютерной программы с другими.
* ПО – программное обеспечение
* САПР – система автоматизированного программирования
* СКАРА - сверхкомпактный автономный робот-автомобиль
* ПСПОД – лаборатория промышленных систем и потоковой обработки данных
* ТО – топологическая оптимизация
* JSON - (JavaScript Object Notation) - это формат передачи данных, основанный на синтаксисе объектов JavaScript.

**ВВЕДЕНИЕ**

3D-моделирование – это процесс создания геометрической модели, путем описания ее координат в пространстве, имеющим три однородных измерения: высота, длина, ширина, задающихся в трех плоскостях: X, Y, Z. С развитием 3D-технологий, 3D-моделирование начало внедрятся в системы автоматизированного проектирования, создавая новый подход к процессу проектирования. У авторов появилась возможность создавать модели объектов практически любой сложности. С инновационными технологиями процесс проектирования становится все более автоматизированным, что позволяет инженеру сосредоточится на идее, а не на реализации задуманного. В настоящее время в лаборатории ПСПОД ведется работа над проектом беспилотного транспортного средства СКАРА (рисунок 1).

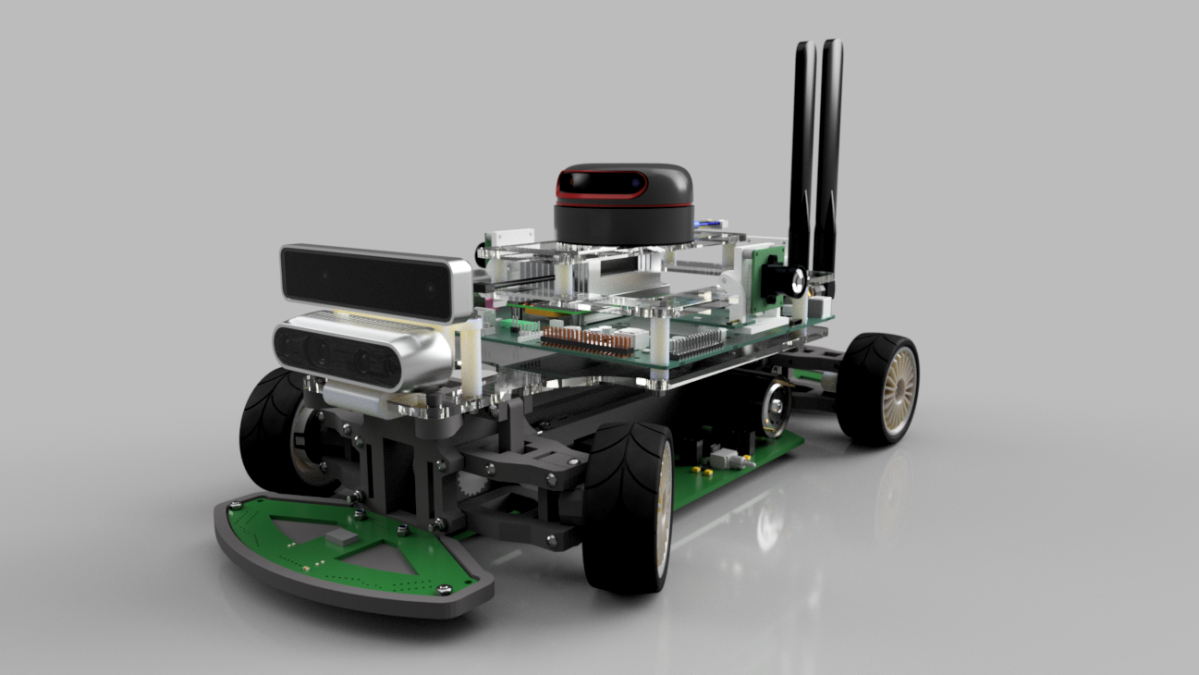


Рис. 1. Модель беспилотного робота-автомобиля СКАРА

Беспилотный робот-автомобиль — транспортное средство, оборудованное системой автоматического управления, которое может передвигаться без участия человека. Беспилотные транспортные средства активно внедряются в различные сферы — от космических исследований до подземных сред. Такие устройства позволяют сделать деятельность человека более безопасной, так как могут функционировать в неблагоприятных условиях внешней среды. Еще одним преимуществом служит экономия драгоценного времени, затрачиваемого на управление транспортным средством, которое пользователь может потратить по своему усмотрению. Задачи, которые может решать СКАРА, ограничены установленными на нем датчиками. Это может быть сбор информации о рельефе, создание карты местности и другие исследовательские цели.

Модель автономного робота-автомобиля спроектирована с помощью САПР и состоит из шасси и прикрепленной к нему платформы с датчиками. Немаловажным элементом для данного устройства является корпус, который подлежит разработке. В дальнейшем планируется производить несколько моделей робота-автомобиля, отличающихся размерами шасси, набором датчиков и предназначенных для использования в разных средах (сухая, влажная, каменистая и т.д.) Так как для каждой новой модели устройства необходимо разрабатывать корпус, было принято решение автоматизировать этот процесс с помощью разработки программного модуля для автоматической генерации 3D-моделей с использованием методов топологической оптимизации – что и стало основной целью данной работы. Вариаций датчиков и устройств может быть большое множество, и оно не обязательно лимитировано тем, что на данный момент есть тестовом образце робота-автомобиля. Как следствие, проектирование каждый раз нового устройства и корпуса для него – в перспективе более затратная задача по времени в сравнении с написанием программного модуля, который будет разработан только один раз и поможет создавать множество различных вариантов корпусов. Актуальность работы также заключается в том, что разрабатываемый программный модуль будет учитывать специфику проекта СКАРА и станет уникальным решением конкретной задачи.

Для проектирования корпуса необходимо определится, какая САПР будет наиболее подходящей. Главным критерием является наличие модуля топологической оптимизации и существование API для возможности использования ПО в программном модуле. Одной из задач, которую предстоит решить в процессе работы, является выявление требований к программному модулю. Также необходимо изучить API выбранного программного средства.

**ГЛАВА 1. ТЕХНОЛОГИИ СОЗДАНИЯ 3D-МОДЕЛЕЙ**

* 1. **Обзор существующих технологий создания 3d-моделей**

Создание и дальнейшее взаимодействие с 3D-моделями требовало значительных вычислительных мощностей. Сперва оно было доступно только крупным компаниям и исследовательским учебным заведениям, где использовались суперкомпьютеры и новейшие по тем временам технологии. В наши дни процессом 3D-моделирования повсеместно используется, оставаясь перспективной отраслью развития для многих сфер.

Существует множество технологий для создания 3D-моделей:

* CAD-программы (Computer-Aided Design) - это программы, предназначенные для создания 3D-моделей с помощью компьютера. Они могут быть использованы для создания различных объектов, от простых предметов до сложных машин и архитектурных конструкций с помощью редактирования точек, линий и поверхностей. Некоторые из популярных CAD-программ включают в себя AutoCAD, SolidWorks и CATIA.
* Моделирование с использованием сканирования - это процесс создания 3D-модели путем сканирования реальных объектов и преобразования их в цифровую форму. Для этого используются специальные устройства, такие как 3D-сканеры и технологии, такие как лазерное сканирование, структурированный свет или фотограмметрию.
* Моделирование с использованием фотограмметрии - это технология, которая позволяет создавать 3D-модели путем обработки множества фотографий объекта. Программы, использующие фотограмметрию, могут определять точки на объекте, а затем соединять их в 3D-модель.
* Скульптурирование - это процесс создания 3D-моделей путем непосредственного моделирования объекта в компьютерной программе. Эта технология основана на создании модели из блока материала и затем оттачивании ее до желаемой формы. Некоторые программы для скульптурирования включают в себя ZBrush и Mudbox.
* VR-моделирование - это технология, которая позволяет создавать 3D-модели в виртуальной реальности. Она может использоваться для создания объектов и архитектурных конструкций с помощью специальных программ и оборудования, таких как HTC Vive и Oculus Rift.
* Моделирование с использованием математических методов - это процесс создания 3D-моделей путем использования математических алгоритмов и уравнений. Эта технология используется для создания сложных геометрических форм и объектов.
* Программы-конструкторы, которые позволяют пользователям выбирать из библиотеки готовых объектов, чтобы создавать свои модели, например, Tinkercad, SketchUp.
* Подходы, использующие геометрические формы, такие как примитивы и NURBS (Non-Uniform Rational B-Splines). Такие инструменты, как Blender, могут использовать эти техники.

В данной работе полученные 3D-модели корпусов планируется создавать аддитивным способ (печать на 3D-принтере), поэтому целесообразно выбирать технологию, поддерживающую данную опцию. Также важно учитывать габариты устройства СКАРА и создаваемого корпуса, строго позиционировать готовый корпус относительно устройства, чтобы совпадали точки крепления. Кроме того, для автоматизации процесса создания 3D-моделей нужно свести к минимуму ручное проектирование модели и последующую ручную оптимизацию. Технология должна быть доступной для студентов и не требовать сверхмощных вычислительных средств.

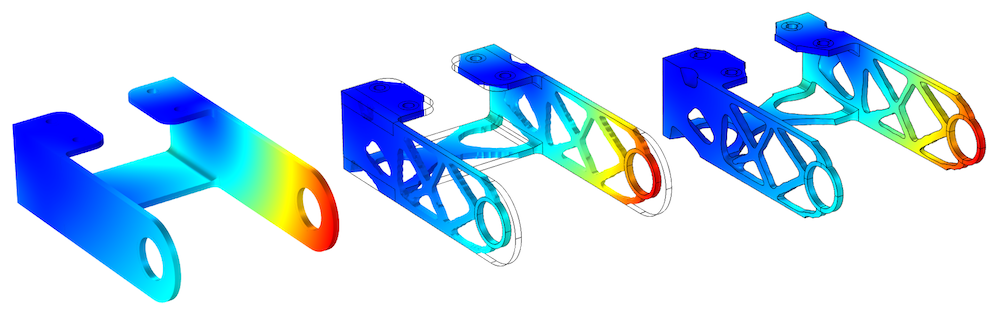
Вышеописанным требованиям удовлетворяет технология САПР. Множество программ для создания 3D-моделей предоставляют удобный графический интерфейс с возможностью создания и гибкой настройки 3D-объектов. Поддерживает опция подготовки модели к 3D-печати, алгоритмы оптимизации и генерации 3D-моделей. Также технология САПР считается общедоступной, так как ее можно использовать на обычных ПК.

* 1. **Понятие топологической оптимизации**

Топологическая оптимизация - это процесс, который использует симуляции для оптимизации формы и расположения проекта на основе заданных требований к производительности. В топологической оптимизации компьютер анализирует структуру проекта и систематически удаляет материал из областей, где он не нужен, сохраняя при этом структурную целостность проекта. В результате получается дизайн, который оптимизирован по таким факторам, как вес, жесткость и прочность, при минимальном использовании материала.

Обычная оптимизация топологии использует анализ конечных элементов для оценки производительности проекта и создания структур, удовлетворяющих следующим требованиям:

* Сниженное соотношение жесткости к весу
* Лучшее соотношение энергии деформации к весу
* уменьшенное соотношение объема материала к коэффициенту запаса прочности,
* Отношение собственной частоты к массе.

Рис. 2. Топологической оптимизации монтажной скобы (осталось только 50% материала)

Сгенерированные конструкции произвольной формы TO часто трудно изготовить с использованием традиционных методов производства. Но из-за роста и технического прогресса в [аддитивном производстве](https://engineeringproductdesign.com/knowledge-base/additive-manufacturing-processes/) (или 3D-печати) результаты проектирования с помощью оптимизации топологии могут быть загружены непосредственно в 3D-принтер.

Как и любой другой инструмент проектирования или процесс проектирования продукта, TO имеет свои преимущества и недостатки. К преимуществам можно отнести:

* *Оптимизированный дизайн.* В большинстве случаев дизайн продукта требует балансировки факторов и определения наилучшего решения. Оптимизация топологии помогает с количественной оценкой оптимального проекта. FEA учитывает множество факторов и избегает опасных предположений, которые могут привести к дефектным элементам.
* *Минимизация материала*. Наиболее привлекательным преимуществом оптимизации топологии является ее способность уменьшать количество ненужного материала и повышать соотношение жесткости к весу. Меньший вес и размер также приводят к меньшему размеру продукции и меньшему потреблению энергии. Кроме того, оптимизированная конструкция сократит количество необходимого сырья, а значит, будет экологичным для планеты.
* *Рентабельность*. Проекты оптимизации топологии будут экономически эффективными при наличии подходящего производственного процесса. Потому что вы размещаете материал только там, где это необходимо, сводя к минимуму использование материала и стоимость. Это также экономит другие факторы, такие как упаковка, меньше энергии для перемещения и транспортировки. Многие сложные геометрические формы, возникающие в результате оптимизации топологии, делают стандартные производственные процессы непрактичными. Однако в сочетании с 3D-печатью эта сложность не требует дополнительных затрат.
* *Уменьшение воздействия на окружающую среду*. Дизайн с оптимизацией топологии является устойчивым, поскольку он производит меньше отходов. Но это может быть не так, если выбрано неправильное производство. Например, схемы оптимизации топологии больше подходят для аддитивного производства. Для субтрактивного производства жизненно важно начинать проектную модель как можно меньшего размера, чтобы свести к минимуму удаление материала и минимизировать потери.
* *Ускорение итеративного процесса проектирования*. Использование различных режимов и учет нагрузки на компоненты снижает риск отказа.
* Более короткое время выхода на рынок

Топологическая оптимизация является ценным инструментом для инженеров и дизайнеров, поскольку он может помочь создавать легкие, но прочные дизайны, используя материалы эффективно. Удаляя материал из областей, где он не нужен, дизайнеры могут достичь требуемых требований производительности, минимизируя при этом общий вес дизайна. Это может иметь значительные преимущества в приложениях, где вес является критическим фактором, таких как в авиакосмическом и автомобильном дизайне, где снижение веса может повысить топливную эффективность и производительность.

Существует несколько методов для выполнения топологической оптимизации, включая оптимизацию на основе градиента, эволюционные алгоритмы и методы уровня набора. Эти методы используют математические алгоритмы и симуляции для оптимизации дизайна, и результаты могут быть визуализированы и проанализированы с помощью программного обеспечения для 3D-моделирования.

Конец формы

Оптимизацию топологии часто неправильно понимают и смешивают с генеративным проектированием. Оба метода проектирования продукта стали популярными в последнее десятилетие, и часто ошибочно полагают, что они одинаковы. Генеративный дизайн — это методология проектирования, при которой автор использует компьютерные технологии, позволяющие творить посредством управления процессом автоматизированной генерации создаваемых объектов [1].  Генеративный дизайн использует оптимизацию топологии, но делает шаг вперед, устраняя необходимость в исходной модели. Генеративный дизайн создает свою модель, используя такие входные данные, как проектное пространство, силы и ограничения. Затем он использует оптимизацию формы для анализа и создания нескольких проектов для оценки инженером



Рис. 3. Набор моделей на выходе модуля генеративного дизайна

* 1. **Обзор существующих решений для автоматической генерации 3d-моделей**

Автоматическое создание 3D-моделей - это процесс создания трехмерных моделей объектов с помощью компьютерных программ и алгоритмов, которые используются для генерации моделей автоматически. Этот процесс обычно основан на данных, полученных с помощью различных технологий сканирования, таких как лазерное сканирование, фотограмметрия, томография и другие методы. Алгоритмы и программное обеспечение анализируют эти данные и генерируют 3D-модель.

Так, программы Agisoft Metashape и RealityCapture создают 3D-модели из серии фотографий, сделанных с разных углов, использую алгоритмы фотограмметрии для выравнивания фотографий и генерации 3D-модели. ПО Pix4Dmapper разработано специально для создания 3D-моделей из аэрофотоснимков, сделанных дронами. Оно может генерировать 3D-облачные точки, текстурированные меши и ортомозаики.

К популярным программа, создающим с помощью лазерного сканирования 3D-модели, а также меши и облака точек, относятся Leica Cyclone , Trimble RealWorks, FARO Scene

Стоит упомянуть о программах, которые генерируют 3D-модели на основе чертежей зданий, планов местности. Также существуют онлайн генераторы трехмерных объектов, где можно на основе существующих в библиотеки моделей создать свой 3D-объект, выполняя настройку параметров.

Что касается аналогов автоматической генерации 3D-моделей в ПО типа САПР, то популярными являются Autodesk Fusion 360 и Autodesk Dreamcatcher. Autodesk Dreamcatcher - это экспериментальное программное обеспечение, которое позволяет пользователям задавать параметры дизайна и желаемые характеристики продукта, а затем использовать алгоритмы генеративного дизайна для автоматического создания оптимальных форм и структур. Программа может использоваться для создания различных продуктов, от мебели до автомобилей, и может помочь ускорить процесс проектирования, сократить затраты на материалы и повысить эффективность конструкции.

Изучив предметную область на наличие аналогов и существующих решений, не было найдено средств, способных сгенерировать 3D-модель корпуса с минимальным участием пользователя и использующих входные данные в виде описания объектов и их местоположения. Однако имеется возможность использовать некоторые САПР в качестве инструмента в предстоящей разработке программного модуля.

**Глава 2. Проектирование программного модуля**

**2.1. Выбор ПО с модулем топологической оптимизации**

Первая задача, которая подлежит рассмотрению – анализ существующих программных средств и выбор подходящего ПО с модулем ТО. Можно выделить следующие критерии по выбору ПО, удовлетворяющего требованиям данной работы:

1. Наличие модуля топологической оптимизации
2. Наличие API и документации по использованию
3. Доступность
4. Комфортный пользовательский интерфейс, не требующий глубокого изучения
5. Технические требования к устройству, на котором будет запускаться ПО

Были проанализированы следующие программные средства: Autodesk Fusion 360, PTC Creo Generative Design, Catia 3dexperience, SIEMEN’S NX, Ansys Discovery. Вышеперечисленные ПО являются лидерами рынка и предоставляют весь необходимый функционал для моделирования и топологической оптимизации. Рассмотрим каждое более внимательно.

* Altair OptiStruct: Это программа для анализа конечных элементов, которая включает в себя возможности топологической оптимизации. Она позволяет пользователям задавать цели и ограничения проектирования, а затем генерировать оптимизированные конструкции на основе этих параметров. Altair OptiStruct широко используется в авиационной, автомобильной и производственной промышленности. Она известна своими надежными алгоритмами оптимизации, а также способностью обрабатывать сложные ограничения проектирования и требования производства.
* SolidThinking Inspire: Эта программа специально разработана для топологической оптимизации и позволяет пользователям создавать и оценивать оптимизированные конструкции для различных приложений. SolidThinking Inspire использует уникальный подход к топологической оптимизации, который заключается в создании грубой геометрии желаемой конструкции, а затем использовании алгоритмов оптимизации для уточнения дизайна. Он известен своим интуитивно понятным интерфейсом и способностью генерировать легкие и эффективные конструкции.
* COMSOL Multiphysics: COMSOL Multiphysics - это инструмент для моделирования, который включает в себя возможности топологической оптимизации. Он позволяет пользователям оптимизировать расположение и форму конструкций на основе различных критериев проектирования. COMSOL Multiphysics часто используется в авиационной, автомобильной и химической промышленности
* Topology Optimization Creo. В Creo пользователи могут задавать различные ограничения на конструкцию, такие как масса, прочность и жесткость, а затем использовать топологическую оптимизацию для создания оптимальной конструкции, учитывающей эти ограничения. Пользователи могут также включать дополнительные параметры, такие как методы производства, материалы и технические ограничения, чтобы создавать детали, которые соответствуют требованиям производства. Creo также предоставляет возможность быстро проверять и анализировать различные варианты оптимизированных конструкций, чтобы определить наилучший вариант. К сожалению, настоящее время могут возникнуть трудности с приобретением лицензии на этот программный продукт.
* Ansys Discovery - это программное обеспечение для 3D-моделирования с оптимизацией топологии, приближенной к процессу генеративного дизайна, как заверяет компания. Discovery — это надежное программное обеспечение от Ansys, в котором реализовано то, что компания называет «проектированием, основанным на моделировании».  Discovery сочетает в себе мгновенное физическое моделирование и интерактивное геометрическое моделирование на одной платформе. Стоимость годовой подписки 1200$.

В качестве программного средства для данного курсового проекта был выбран Ansys Discovery. Для его установки требуются устройства с высокими техническими характеристиками, так как вычисления проводятся на самом устройстве без использования облака. Ansys Discovery предлагает бесплатную студенческую версию продукта, что вполне подходит для поставленной задачи. Однако, у студенческой версии применяются академические условия использования, к которым относится отсутствие экспорта геометрии и возможность сохранить файл только в формате Discovery(\*.scdoc). Если лаборатория будет запускать коммерческое производство СКАРА, то необходимо будет приобрести подписку Ansys Discovery, чтобы не нарушать лицензионное соглашение. Ansys Discovery имеет API и предоставляет документацию по его использованию. Стоит заметить, что в открытом доступе почти нет примеров использования API Discovery, так как оно появилось только в 2021 году. Особое внимание стоит обратить на модуль топологической оптимизации. Программа генерирует большое количество вариантов дизайна и оценивает их на основе заданных пользователем критериев и ограничений, что приводит к оптимизированному дизайну, который соответствует желаемым целям производительности. Кроме того, Discovery имеет удобный пользовательский интерфейс, не требующий долгого изучения и практики.

**2.2. Разработка принципов и плана решения**

Создание продукта, лежащего в основе этой квалификационной работы, требует глубокого изучения предметной области и проектирования решений. В основе решения будет лежать алгоритм автоматической генерации и оптимизации 3D-модели, на вход которого будут поступать пользовательские данные, на основе которых и будет происходить построение объектов.

Начнем с задания пользовательских данных. Это ключевой момент, так как конечный продукт напрямую зависит от пожелания пользователя и корректности задаваемых данных. Необходимо выбрать формат, в котором данные будут передаваться на вход программного модуля. После изучения существующих вариантов, принято решение использовать JSON-формат. JSON представляет собой текстовый формат, который можно легко читать и создавать как человеком, так и машиной. Он использует синтаксис пар "ключ-значение" и может быть представлен в виде объектов, массивов, строк, чисел, логических значений и значений null.

Размеры корпуса, расположение датчиков относительно корпуса, размер, масса датчиков и форма датчиков, точки крепления их к корпусу – переменные величины, которые зависят от модели устройства СКАРА и задаются пользователем. Датчики могут быть двух форм: прямоугольная и цилиндрическая. Размер прямоугольных датчиков указывается с помощью трех параметров: длины датчика по осям x, y, z. Размер цилиндрических – с помощью радиуса и высоты по оси z. Положение датчика, а именно его центра, описывается тремя координатами по осям x, y, z. Также опционально можно задать вращение датчика, указав угол поворота в градусах и ось (x, y, z), относительно которой производиться вращение. Масса датчика является опциональным параметром, ее необходимо указать в ньютонах. Помимо описания корпуса и датчиков необходимо указать места крепления этих объектов друг к другу, к шасси СКАРА, для это нужно указать размер отверстия под бедующий винт и его расположение.

Идея создания корпуса заключается в том, что сначала создается ограничивающий объем корпуса, другими словами – указывается область, которую будет занимать корпус, и за пределы которой он не должен выходить. Под корпусом, сверху и по бокам могут находиться датчики, имеющие определенный размер, вес, точки крепления. Далее определятся объем, который занимают датчики внутри корпуса – это пространство нельзя затрагивать во время проектирования. Следует учитывать, что датчики, располагающиеся сверху на корпусе, имеет массу и оказывают нагрузку на него. Поэтому нужно выдвигать требования к жесткости конструкции – корпус должен быть достаточно прочным, чтобы выдержать вес датчиков. Так же, датчики могут проходить насквозь корпуса, поэтому для некоторых нужно предусмотреть соответствующие отверстия.

После обработки входных данных и построения начальной геометрии можно перейти к процессу топологической оптимизации. На этом этапе необходимо задать материал для корпуса и ограничения и нагрузки, необходимые для запуска процесса топологической оптимизации. В зависимости от реализованного алгоритма ТО, для запуска процесса необходимо указать ограничения (фиксированные части модели, не затрагиваемые процессом оптимизации), способы крепления деталей (шарнирный, фиксированный, может двигаться относительно осей), нагрузки (вектор и значение нагрузки, оказываемый другой деталью). В последнюю очередь ставится цель для топологической оптимизации, определяющая, каким способом алгоритм будет вычитать лишний объем. Чтобы предоставить пользователю выбор из нескольких вариантов, как при генеративном дизайне, в данной работе принято решение запустить процесс ТО несколько раз изменяя при этом начальную цель.

После окончания процесса топологической оптимизации необходимо определить, в какой форме возвращается результат алгоритма, по необходимости преобразовать его для дальнейшего сохранения в выбранном формате. Использования студенческой версии Ansys Discovery накладывает ограничения на формат выходных данных – они могут быть сохранены только в формате Discovery(\*.scdoc). Этот формат предполагает геометрии и других настроек, производимых в процессе работы над моделью в Discovery и может быть открыт только в приложении Discovery.

**ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ**

**3.1. Создание плагина для выбранного ПО**

Ansis Discovery предоставляет возможности для создания плагина или надстройки (Add-in). Готовый плагин отображается в интерфейсе при открытии Discovery и предоставляет прописанный в нем функционал. Для создания плагина в данной работе использовались ПО Visual Studio 2022 и язык программирования С# версии 10.0.

Плагин для Discovery представляет из себя библиотеку DLL с управляемым кодом, использующий API Discovery версии API.23. API состоит из двух частей: SpaceClaim.Api.V23.dll отвечает за построение и редактирование геометрии, Discovery.Api.V23.dll отвечает за задание физических характеристик, создание симуляции и запуск процессов, доступных в разделе Simulation.

Плагин, являясь сборкой библиотеки классов .NET, реализует как минимум интерфейс IExtensibility. Этот интерфейс предоставляет точку входа для надстройки и позволяет ей выполнять любую инициализацию. Чтобы плагин отображался в пользовательском интерфейсе Discoveri, его нужно разместить в ленте. Лента — это графический элемент управления, который проходит по горизонтальной полосе в верхней части окна приложения. Как правило, лента включает в себя панели инструментов с вкладками, которые заполнены графическими кнопками и другими графическими элементами управления и сгруппированы по функциональным возможностям. Для этого необходимо реализовать интерфейсы ICommandExtensibility и IRibbonExtensibility.

Во время запуска плагина Discovery выполняет следующие шаги для загрузки и инициализации надстройки:

1. Находит все XML-файлы манифеста, описывающие доступные надстройки.
2. Использует имя сборки и типа, указанные в файле манифеста, для создания объекта надстройки.
3. Проверяет, реализует ли плагин интерфейс IExtensibility, а затем вызывает IExtensiblity.Connect. В это время надстройка выполняет любую внутреннюю инициализацию.
4. Проверяет, реализует ли надстройка дополнительные интерфейсы ICommandExtensibility и IRibbonРасширяемость.
5. Когда Discovery закрывается, вызывается метод надстройки IExtensibility.Disconnect.

В Visual Studio 2022 был создан новый проект, в который были добавлены ссылки на сборки API: SpaceClaim.Api.V23, Discovery.Api.V23, SpaceClaim.Api.V23.Intarnal. Был написан класс, служащий точкой входа в программу, наследующийся от класса AddIn и реализующий интерфейсы IExtensibility, ICommandExtensibility, IRibbonExtensibility.

Discovery использует XML-файл манифеста для описания надстройки, который содержит методанные (имя и описание плагина) и данные об использовании (сборка, название типа класса плагина, вариант хостинга). Атрибут хост имеет значение по умолчанию, все остальные атрибуты необходимо указать. Был написан следующий XML-файл манифеста:

Листинг 3.1

|  |
| --- |
| <AddIns>  <AddIn name="BasicTemplate"  description="BasicTemplate Add-in for Disco"  assembly="BasicTemplate.dll"  typename="BasicTemplate.BasicTemplate"  host="SameAppDomain"  allowOnDemandLoad="true" />  </AddIns> |

Так же в созданном классе была произведена настройка ленты. При запуске Discovery вызывается метод IRibbonExtensibility.GetCustomUI, который возвращает строку, содержащую XML-описание изменений, которые нужно внести в ленту.

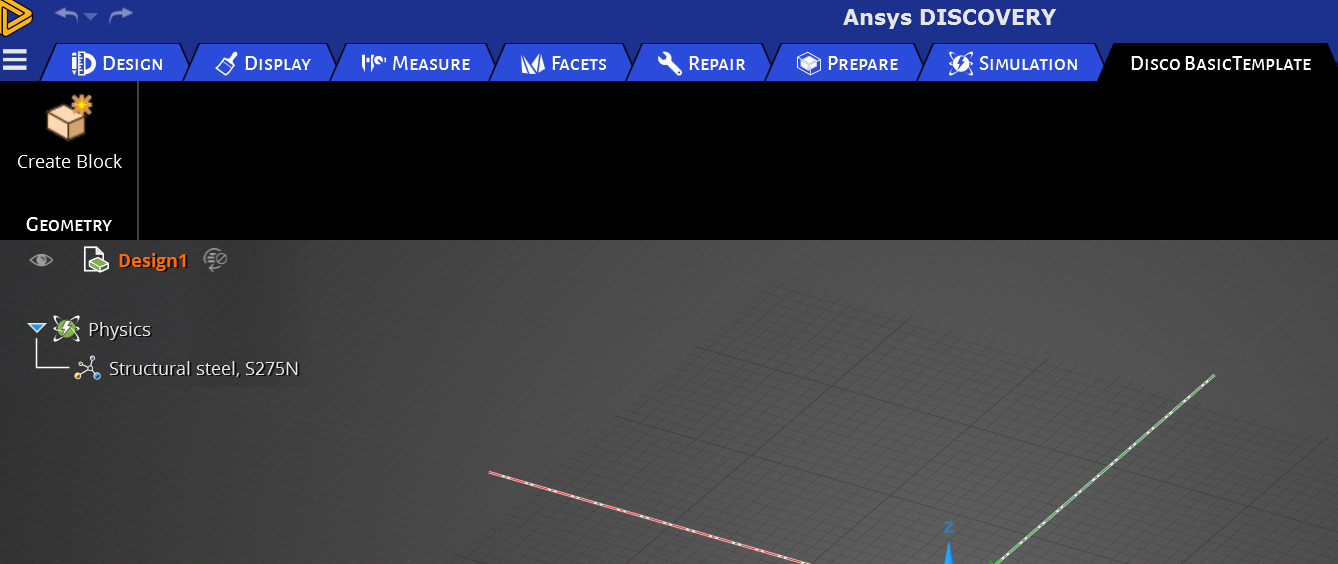
Листинг 3.2

|  |
| --- |
| <ribbon>  <tabs>  <tab id="Disco BasicTemplate" command="BasicTemplate.RibbonTab">  <group id ="Geometry" command="BasicTemplate.Geometry" label="Geometry">  <button id="BasicTemplate.CreateBlock" size="large" command="BasicTemplate.CreateBlock"/>  </group>  </tab>  </tabs>  </ribbon>  </customUI> |

Из листинга 3.2 можно увидеть, что в ленте в группе геометрия создается графический элемент, за которым закрепляется команда «create block». Все объекты пользовательского интерфейса (UI) в Discovery должны быть связаны с командой. Команда управляет текстом, всплывающей подсказкой, изображением и обрабатывает события «Выполнение» и «Обновление». Для изменения и добавления команд реализован интерфейс ICommandExtensibility. Обнаружение команд выполняет метод ICommandExtensibility.Initialize при запуске. Ниже приведен листинг класса инициализации плагина:

Листинг 3.3

|  |
| --- |
| public class BasicTemplate : AddIn, IExtensibility, ICommandExtensibility, IRibbonExtensibility  {  readonly CommandCapsule[] capsules = new[] {  new CreateBlockCapsule(),  };  public bool Connect()  {  return true;  }  public void Disconnect()  {    }  public void Initialize()  {  foreach (CommandCapsule capsule in capsules)  {  capsule.Initialize();  }  }  public string GetCustomUI()  {  return Resources.Ribbon;  }  } |

 Рис. 3.1. Новый раздел в Discovery, содержащий кнопку «создать блок»

**3.2. Обработка входных данных**

Начальные данные от пользователя передаются в программу в формате JSON. Создается текстовый файл .txt, где пользователь описывает входные данные, в плагине указывается путь к этому файлу. С помощью метода ReadAllText считывается содержимое файла, после чего конвертируется в объект динамического типа при помощи метода DeserializeObject класса JsonConvert. Далее в созданный массив запиваются значения корпуса. Блок обращения к полученному динамическому объекту обернут кострукцией try – catch для выявления ввода некорректных данных.

|  |
| --- |
| {"cube": [0.35, 0.19, 0.058],  "hole": [  {"radius":0.003, "loc":[0.1645, 0.036, 0.067], "extrude": 0.002, "count":4},  {"radius":0.0015, "loc":[0.1589, 0.018, 0.092], "extrude": 0.008, "rotation": [90, 1], "count":1},  {"radius":0.0015, "loc":[0.1589, -0.018, 0.092], "extrude": 0.008, "rotation": [90, 1], "count":1},  ],  "sensors": [  {"name":"frontCameraB", "size":[ 0.0889, 0.025, 0.031], "mass":20, "location":[-0.16847, 0, 0.08363], "shape":"s"},  {"name":"frontCameraT", "size":[0.108, 0.0155, 0.0265], "mass":20, "location":[-0.173, 0, 0.11159], "shape":"s"}  ]  } |

Рис. 3.2. Пример входных данных

Для более удобного хранения и использования распарсенных входных данных были созданы классы HoleT и Sensor. Создавая экземпляр класса HoleT вызывается конструктор, в который необходимо передать параметры типа Double: радиус, длину отверстия (extrude), массив чисел Double, задающий расположение точки относительно центра координат и количество точек. Если количество точек не задано пользователем, считается что это 1 точка. В классе реализованы get-методы, возвращающие значения полей класса. Также написан метод setRotation, так как пользователь опционально может задать вращение для отверстия. Он также используется при парсинге входных данных, при вызове устанавливает значение угла поворота (angle) и оси, вдоль которой будет проводится вращение (axic) соответствующим полям класса.

По той же логике реализован класс Sensor. С помощью конструктора заполняются поля класса name, mass, size, location, IsSquare, angl, axis. Написаны get-методы, чтобы получать безопасно доступ значения полей класса.

В методе getParametrs после получения и преобразования данных от пользователя вызываются методы parserHole и parserSensor, которым в качестве параметра передается dynamic объект. В этих методах создаются объекты соответствующих классов и добавляются в объявленные в классе CreateBlockCapsule листы – формы хранения для полученных пользовательских данных. При парсинге используется приведение чисел к типу Double. При парсинге датчиков сначала определяется их форма, в зависимости от того, прямоугольный ли датчик или цилиндрический, применяются разные варианты парсинга. У прямоугольных датчиков 3 параметра размера, а цилиндрических – 2 параметра (радиус и длина). При этом угол и ось вращения задаются равными нулям, если пользователь не указал их для датчика или отверстия.

**4. Вопросы, подлежащие дальнейшей разработке**

В следующей итерации работы над отчетом будет описаны методы создания геометрии датчиков и отверстий, наложение физических характеристик на полученную геометрию, создание симуляции для запуска ТО, определение цели ТО и запуск процесса, обработка полученных данных, экспорт результатов моделирования.

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения научно исследовательской работы были обоснованы цель и актуальность выбранной темы. Исследована предметная область, выбрана технология создания 3D-модели. Так же были изучены статьи о генеративном дизайне и топологической оптимизации, после чего было принято решение, что возможностей топологической оптимизации достаточно для данного проекта. Исследованы и проанализированы различные ПО, реализующие методы топологической оптимизации, и выбрана наиболее подходящая по сформированным критериям. Разработаны принципы и план решения. Проведено ознакомление с API Ansys Discovery, после чего была написаны класс инициализации плагина и методы парсинга пользовательских данных (код которых приведен в Приложениях 1-3). Сформирован список вопросов, подлежащих последующей разраьотки.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Дё Ю.С. Тектоника и генеративный дизайн / Дё Ю.С., Кремлёв А.Ю. // Молодёжь и современные информационные технологии: сборник трудов XIV Международной научно-практической конференции студентов, аспирантов и молодых учёных. Томск, 2016. т. 2. с. 203–204.
2. Озеров А. Топологическая оптимизация и 3D‑печать как пример взаимного развития // Журнал об аддитивном производстве – 2022 – №4 – С.22 – 30
3. Ерофеев, П. С. Применение технологии BIM в архитектурном учебном проек­тировании зданий и сооружений / П. С. Ерофеев, А. И. Меркулов, В. Ф. Манухов // Вестник Мордов­ского университета. – 2015. – Т. 25, № 1. – С. 105–109. DOI: 10.15507/VMU.025.201501.105

Приложение 1

Листинг класса HoleT

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace BasicTemplate  {  internal class HoleT  {  Double radius;  Double extrude;  List<Double> location = new List<Double>();  Double angle = 0;  int axis = 0; //x = 1, y = 2, z = 3  int count = 1;  public HoleT(Double radius, Double extrude, List<Double> location, int count)  {  this.radius = radius;  this.extrude = extrude;  this.location = location;  this.count = count;  }  public void setRotation(Double angle, int axic) {  this.angle = angle;  this.axis = axic;  }  public int getHoleCount()  {  return count;  }  public Double getRadius()  {  return radius;  }  public List<Double> getLocation()  {  return location;  }  public Double getExtrude()  {  return extrude;  }  public int getAxic()  {  return axis;  }  public double getAngle()  {  return angle;  }  }  } |

Приложение 2

Листинг класс Sensor

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace BasicTemplate  {  internal class Sensor  {  String name;  List<Double> size;  Double mass;  List<Double> location;  bool IsSquare;  Double angle = 0;  int axis = 0;  public Sensor(String name, Double mass, List<Double> size, List<Double> location, Boolean b)  {  this.name = name;  this.mass = mass;  this.size = size;  this.location = location;  this.IsSquare = b;  }  public void setRotation(Double angle, int axic)  {  this.angle = angle;  this.axis = axic;  }  public String getName()  {  return name;  }  public bool getShape()  {  return IsSquare;  }  public List<Double> getSize()  {  return size;  }  public List<Double> getLocation()  {  return location;  }    public Double getAngl()  {  return angle;  }  public int getAxis()  {  return axis;  }  }  } |

Приложение 3

Парсинг данных от пользователя

|  |
| --- |
| protected void getParametrs()  {  string fileText = File.ReadAllText(path);  var parametrs = JsonConvert.DeserializeObject<dynamic>(fileText);  try  {  for (int i = 0; i < cubeSize.Length; i++)  {  cubeSize[i] = (Double) parametrs.cube[i];  }  parserHole(parametrs);  parserSensor(parametrs);  }  catch(Exception e){  Debug.Fail(e.Message);  }  }  private void parserHole(dynamic parametrs)  {  if (parametrs.hole == null)  Debug.Fail("hole == null");  foreach (var h in parametrs.hole)  {  List<Double> holeLoc = new List<Double>();  foreach (Double loc in h.loc)  {  holeLoc.Add((Double)loc);  }  HoleT hole = new HoleT((Double)h.radius, (Double)h.extrude, holeLoc, (int) h.count);  if (h.rotation != null)  {  hole.setRotation((Double)h.rotation[0], (int)h.rotation[1]);  }  holes.Add(hole);  }  }  private void parserSensor(dynamic parametrs)  {  try  {  foreach (var i in parametrs.sensors)  {  Boolean b = i.shape == "s" ? true : false;  int j;  List<Double> size = new List<Double>();  List<Double> loc = new List<Double>();  if (b)  {  size.Add((Double)i.size[0]);  size.Add((Double)i.size[1]);  size.Add((Double)i.size[2]);  }  else  {  size.Add((Double)i.size[0]);  size.Add((Double)i.size[1]);  }  foreach (Double l in i.location)  loc.Add((Double) l);    Sensor sensor = new Sensor((String)i.name, (Double)i.mass, size, loc, b);  if (i.rotation != null)  sensor.setRotation((Double)i.rotation[0], (int)i.rotation[1]);  sensorList.Add(sensor);  }  }catch (Exception e)  {  Debug.WriteLine(e.Message);  }  } |