МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**"Южно-Уральский государственный университет
(национальный исследовательский университет) "**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Директор ООО «Статус Безопасности»

_____ Н.П. Макота

“___”_____ 2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2019 г.

РАЗРАБОТКА ПРОГРАММНОЙ СИСТЕМЫ ДЛЯ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ СЕТКИ КОНЕЧНЫХ ЭЛЕМЕНТОВ КОМПОЗИТНЫХ МАТЕРИАЛОВ НА ВОЙЛОЧНОЙ ОСНОВЕ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.04.02.2019.308-153.ВКР

Научный руководитель,
канд. техн. наук, доцент кафедры СП
_____Н.Ю. Долганина

Автор работы,
студент группы КЭ-220
_____И.С. Слободин

Ученый секретарь
(нормоконтролер)
_____ О.Н. Иванова
“___”_____ 2019 г.

Челябинск-2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП

_____Л.Б. Соколинский

09.02.2019

ЗАДАНИЕ
на выполнение выпускной квалификационной работы магистра
студенту группы КЭ-220
Слободину Ивану Сергеевичу,
обучающемуся по направлению
02.04.02 «Фундаментальная информатика и информационные технологии»
(магистерская программа «Технологии разработки высоконагруженных систем»)

1. Тема работы (утверждена приказом ректора от 25.04.2019 № 899)

Разработка программной системы для автоматической генерации сетки конечных элементов композитных материалов на войлочной основе.

2. Срок сдачи студентом законченной работы: 05.06.2019

3. Исходные данные к работе

3.1. Долганина Н.Ю., Сапожников С.Б. Исследование влияния типа переплетения нитей на прочность тканевых преград при локальном ударе. // Вестник Южно-Уральского государственного университета. Серия: Машиностроение, 2013. – Т. 13. – № 2. – С. 95–104.

3.2. Трушин С. Метод конечных элементов. Теория и задачи. – М.: Издательство Ассоциации строительных вузов, 2008. – С. 256.

3.3. Документация языка программирования Kotlin: [https://kotlinlang.ru/docs/reference/kotlin-doc.html].

4. Перечень подлежащих разработке вопросов

4.1. Изучить свойства композитных материалов на войлочной основе.
4.2. Разработать алгоритм создания компьютерной модели композитного материала на войлочной основе.
4.3. Спроектировать программную систему создания компьютерной модели композитного материала на войлочной основе.
4.4. Реализовать программную систему.
4.5. Протестировать программную систему.

5. Дата выдачи задания: 08.02.2019.

Научный руководитель

доцент кафедры системного программирования,
кандидат техн. наук Н.Ю. Долганина

Задание принял к исполнению И.С. Слободин

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность работы

Современные бронезащитные элементы состоят из нескольких слоев различных материалов. В качестве тыльных слоев обычно используются баллистические ткани из прочных арамидных волокон (кевлара). В качестве верхних слоев — металлические или керамические пластины.

С помощью экспериментального исследования невозможно определить влияние множества различных факторов на работу баллистической ткани при нагружении. Также невозможно определить многие свойства материала. Поэтому для решения этой проблемы сегодня часто применяется компьютерное моделирование подобных задач с последующей экспериментальной проверкой полученных данных.

Цель и задачи работы

Целью данной работы является исследование предметной области, способов создания конечно-элементных моделей и разработка программной системы для создания компьютерных моделей композитных материалов на войлочной основе. В программе должен быть пользовательский графический интерфейс, в котором должна быть возможность задать:

- размеры материала;
- длину волокон;
- количество волокон;
- минимальное расстояние между волокнами;
- размер конечного элемента.

Также в графический интерфейс должна выводиться информация о состоянии моделирования, и в нем должна быть возможность прервать процесс моделирования.

Программная система должна создавать k-файл с координатами узлов и конечными элементами, являющийся входным файлом в многоцелевой конечноэлементный комплекс программ LS-DYNA для дальнейших расчетов.

Исходя из поставленной цели, можно сформулировать следующие задачи:

1) изучить свойства композитных материалов на войлочной основе;
2) разработать алгоритм создания компьютерной модели композитного материала на войлочной основе;
3) спроектировать систему создания компьютерной модели композитного материала на войлочной основе;
4) реализовать данную систему;
5) протестировать созданную систему.

Структура и объем работы

Работа состоит из введения, пяти разделов, заключения, библиографии. Объем работы составляет 42 страницы, объем библиографии – 25 источников.

Содержание работы

В первом разделе «Обзор литературы» выполнен анализ типов композитных материалов на войлочной основе, а также исследованы способы создания компьютерных моделей конечно-элементных композитных и войлочных материалов.

Во втором разделе «Анализ требований к программной системе» представлена информация о проектируемой системе, определены и описаны функциональные и нефункциональные требования к системе, варианты использования, и представлена диаграмма и спецификация вариантов использования.

В третьем разделе «Архитектура системы» представлена разработка архитектуры системы. В этом разделе описан общий алгоритм работы системы, пользовательский интерфейс, определены компоненты, составляющие систему.

В четвертом разделе «Реализация» описана реализация алгоритма работы системы и отдельно каждого модуля. Также представлены фрагменты кода.

В пятом разделе «Тестирование» проведена проверка системы на соответствие поставленным ранее требованиям. Раздел содержит различные методы тестирования, например, модульное и функциональное.

В заключении описаны основные достигнутые результаты, полученные при выполнении работы. Подведен итог выполнения задач и целей, поставленных ранее.

1. ОБЗОР ЛИТЕРАТУРЫ

1.1. Композитный материал

Понятие композитных материалов появилось в середине 20 века, но сами композиты вовсе не новое явление. Композитные материалы известны на протяжении тысячелетий. В древности при постройке различных сооружений использовали тростник с глиной, рубленную солому с глиняными кирпичами, железные прутьями внутри мраморных колонн. Железобетон и булат — предшественники современных композитных материалов.

Древесина, кости, панцири — природные композитные материалы. Многие виды природных минералов фактически представляют собой композиты. Они не только прочны, но обладают также превосходными декоративными свойствами. В настоящее время композитными материалами принято считать разнообразные искусственные материалы, которые разрабатываются и внедряются в различных отраслях техники и промышленности. Например, для производства лодок используется стекловолокно, пропитанное смолой (рис. 1).

Рис.1. Изготовление композитного материала для лодок

Композитные материалы используются в современном мире во многих сферах жизни, таких как наука, техника, промышленность, строительство и многих других.

Композитный материал — это неоднородный многокомпонентный сплошной материал, обычно состоящий из пластичной основы (матрицы) и армирующих элементов в качестве наполнителя. Армирующие элементы обычно обеспечивают необходимые механические характеристики материала (прочность, жесткость и т.д.), а матрица — эффективную совместную работу армирующих элементов и защиту их от механических повреждений и агрессивной химической среды. Композитные материалы обладают рядом признаков:

- компоненты материала различаются по химическому составу. Между компонентами имеется четкая граница фаз;
- свойства каждого компонента материала, присутствующего в достаточно большом количестве (больше некоторого критического содержания) в совокупности определяют свойства композита;
- материал состоит из компонентов с различными характеристиками и в результате сам получает свойства, сильно отличающиеся от свойств каждого из составляющих;
- материал неоднороден в микромасштабе и однороден в макромасштабе.

Механические свойства композитного материала определяются соотношением свойств и прочностью связи армирующих элементов и матрицы. Армирующие элементы определяют жесткость, прочность и деформируемость композита, а матрица обеспечивает стойкость к различным внешним воздействиям, монолитность и распределение нагрузки по всему материалу. Правильный выбор исходных компонентов, их соотношений и технологии их совмещения определяет эффективность композита. Меняя их, а также применяя специальные дополнительные реагенты, получают материалы с требуемым набором свойств. Например, существенно повысить тре-

щиностойкость материала можно создав границу раздела между наполнителем и матрицей. В отличие от металлов, в композите повышение статической прочности приводит к повышению характеристик вязкости разрушения.

Преимущества композитных материалов:

- высокая износостойкость;
- высокая удельная прочность;
- высокая жёсткость;
- высокая усталостная прочность;
- высокое сопротивление ползучести.

Матрицами в композитных материалах могут быть металлы, полимеры, керамика и другие. В качестве армирующих элементов используются самые разнообразные вещества. По механической структуре композиты делятся на волокнистые, слоистые, дисперсноупрочненные, упрочненные частицами и нанокомпозиты.

Волокнистые композиты — композиты, в которых в качестве армирующих элементов используются волокна. Даже небольшое их содержание в композитах существенно улучшает механические свойства материала. Также можно варьировать свойства материала, изменяя положение, направление и размеры волокон.

В слоистых композитах, таких как фанера, слоистые пластики, клееное дерево, матрица и наполнитель расположены слоями. Можно менять свойства материалов, укладывая слои под разными углами [1].

Остальные классы композитов различаются размером частиц армирующих элементов. В композитах, упрочненных частицами, их размер больше 1 мкм, а содержание составляет 20-25 % (по объему). В дисперсноупрочненных композитах — от 0,01 до 0,1 мкм с количеством от 1 до 15 %. В нанокомпозитах — 10-100 нм.

Для улучшения свойств волокнистого композита, при его изготовлении должны быть соблюдены следующие условия:

- волокна должны быть распределены равномерно по матрице, не касаясь друг друга;
- на границах разделов должна быть достаточно прочная связь.

Для этого матрица должна обеспечивать хорошую смачиваемость волокна. Также при изготовлении композита не должны быть большими температура и давление. Это поможет избежать изменения свойств композита и его разрушения.

В качестве материала для матриц применяются различные вещества.

Например, в качестве металлов могут использоваться алюминий, магний или медь. Армирующими элементами могут быть высокопрочные волокна или тугоплавкие частицы.

Керамические материалы с волокнами или частицами в качестве наполнителя являются высокопрочными композитами. Часто используют металлические армирующие элементы. Но не все материалы пригодны для использования в качестве них из-за своих свойств, несовместимых со свойствами матрицы.

В полимерных матрицах используются различные смолы или пластмассы. Чтобы при затвердевании матрицы материал не был поврежден, он должен иметь низкую усадку, затвердевать при низких температурах и давлении. Композиты на основе смол обладают хорошими механическими свойствами, имеют большую прочность при сжатии и сдвиге.

1.2. Войлочные материалы

В отличие от тканых, при ударном нагружении прочность нетканых материалов не зависит от размеров исследуемых образцов [15]. По сравнению с тканными войлочные материалы имеют более низкую прочность, но зато показывают лучшие свойства при локальном ударе. Нетканые материалы из высокопрочных волокон могут остановить летящие на высокой скорости фрагменты осколков [2].

В войлочном композите одновременно происходит большое количество механических процессов:

- непрерывная перестройка волокон;
- выпрямление волокон;
- трение;
- разрыв связей;
- перестройка композита.

В экспериментальном исследовании все это не позволяет изучить свойства материала и влияние различных факторов при локальном ударном нагружении [9, 14].

1.3. Существующие решения

Для исследования свойств нетканых материалов при разрушении в различных условиях было проведено множество исследований экспериментального и компьютерного анализа.

Баллистические свойства нетканых материалов были исследованы в работах [6, 22]. Также в этих работах были разработаны программы для создания компьютерных моделей нетканых материалов, но они предназначены для создания войлочных материалов и не могут генерировать композитные.

В работах [10–12, 23] были изучены свойства и поведение геотекстиля и стекловолоконных войлоков из термически скрепленных волокон. В результате была разработана модель в виде сплошной среды (рис. 2). Но она не может показать работу отдельных волокон в материале.

В работах [4, 5] рассмотрен войлочный материал из хаотически расположенных полимерных волокон, скрепленные термическим способом по толщине. Волокна разбиты на конечные элементы двух типов с разными свойствами. Одни описывают участки с термической обработкой, а другие — без нее. Но эти модели также представляют собой сплошную среду.

В работе [8] была создана модель материала, состоящего из триангулированной структуры, также представляющего собой сплошную среду. Было исследовано макроскопическое поведение созданной модели.

Авторы работ [7, 16] создали двумерную модель войлочного материала из прямолинейных волокон в пакете программ Marc. Для этого они написали подпрограмму. Также они провели исследование полученных моделей на растяжение. Однако эта программа для поставленной задачи не подходит, так как композитные материалы имеют трехмерную структуру.

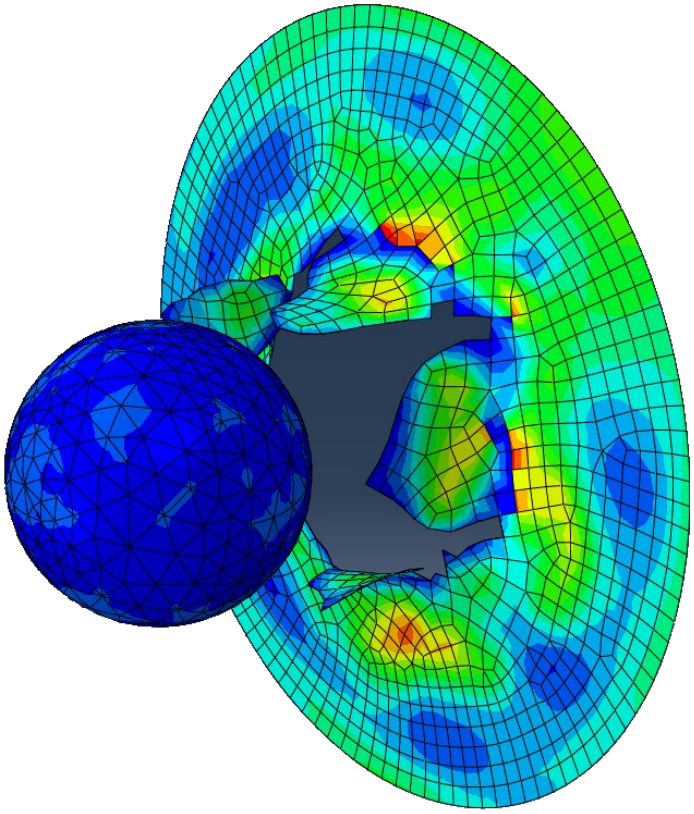

Рис.2. Модель в виде сплошной среды

В настоящее время для создания компьютерных моделей исследуемых объектов часто используются CAD/CAE программы. CAD (computer aided design) — программный пакет для создания чертежей, документации к ним и 3D моделей. CAE (computer aided engineering) — программный пакет для инженерных расчётов, анализа и симуляции физических процессов. Они позволяют оценить поведение компьютерной модели в условиях, приближенных к реальным. Но такой метод не позволяет построить модель с очень сложной геометрией, потому что при переносе модели из пакета программ CAD в конечно-элементный пакет программ CAE не вся модель правильно импортируется.

Задачу динамического нагружения композитного материала эффек-

тивнее считать в пакете программ LS-DYNA, так как это многоцелевой конечноэлементный комплекс для разработки и анализа высоконелинейных и быстротекущих процессов в задачах механики твердого и жидкого тела. Но в этом пакете невозможно построить модель разрабатываемого композитного материала с хаотически расположенными волокнами автоматически. Необходимо каждое волокно задавать вручную. Поэтому этот способ не подходит для решения задачи.

Digimat — пакет программ, позволяющий создавать модели сложных многокомпонентных композитных материалов (рис. 3). Также он дает возможность производить конечно-элементный расчет над созданными моделями. Но данная программа платная и не специализируется на решении динамических задач.

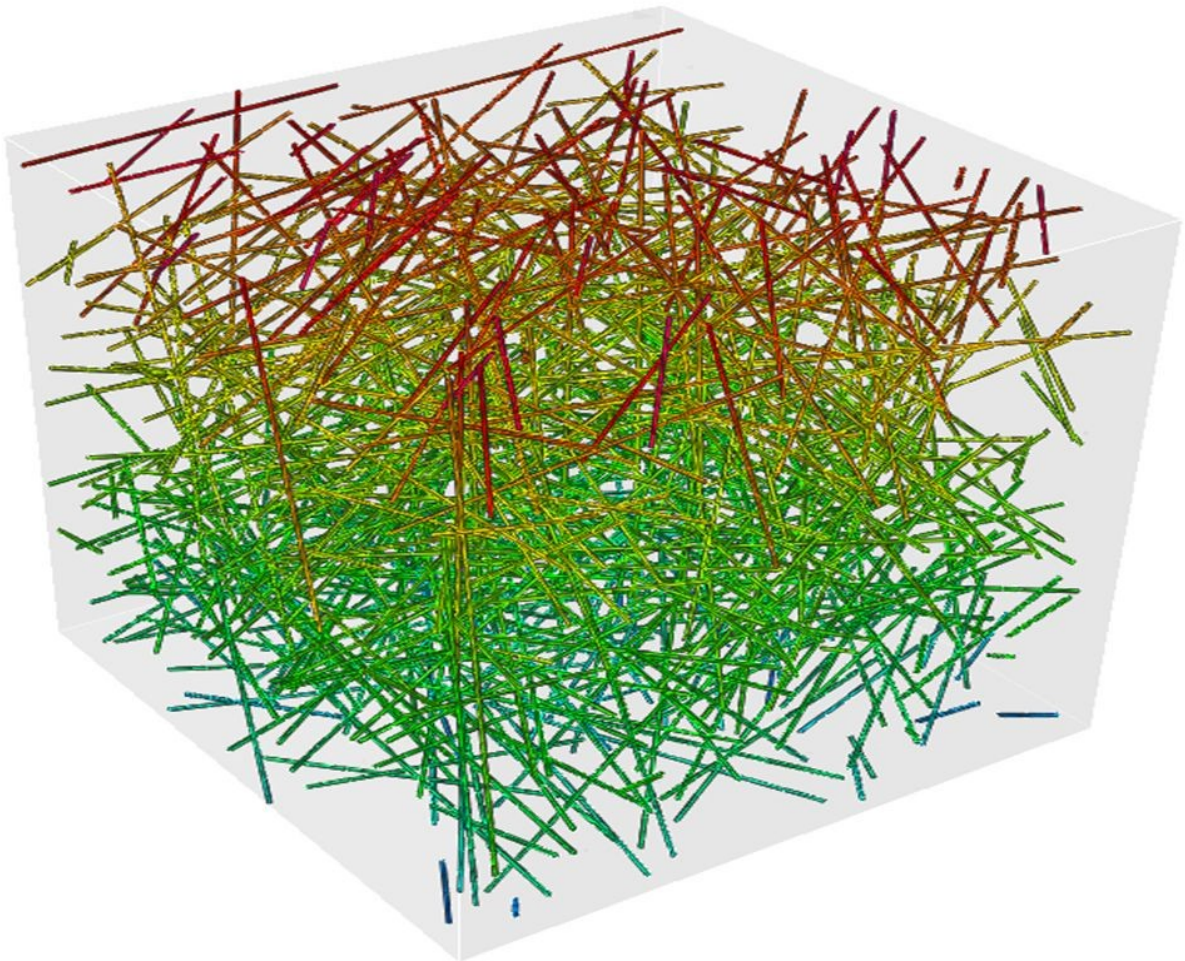

Рис.3. Модель, созданная в пакете программ Digimat

Таким образом, нет готовых решений, в полной мере учитывающих микроструктуру материала. Также не существует готовых решений для моделирования композитного материала на войлочной основе с хаотически расположенными волокнами по всему объему. Поэтому было принято решение разработать и реализовать программную систему, решающую эту задачу.

1.4. Метод конечных элементов

Метод конечных элементов — это численный метод решения дифференциальных и интегральных уравнений, возникающих при решении задач прикладной физики. Область, занимаемая объектом, разбивается на конечное количество подобластей (конечных элементов) [13]. В каждом из элементов произвольно выбирается вид аппроксимирующей функции. Затем собирается система линейных уравнений и решается одним из известных методов. Этот метод имеет очень простую физическую интерпретацию, поэтому он получил очень широкое использование. Еще одним достоинством метода конечных элементов является то, что область, в которой происходит исследование, может быть любой формы [3, 24]. В качестве конечных элементов на плоскости часто используются треугольники, в пространственном — тетраэдры. Для моделирования композитного материала на войлочной основе для волокон используются отрезки, для матрицы — тетраэдры.

2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ

2.1. Функциональные требования к системе

К системе предъявляются следующие функциональные требования:

1) система должна предоставлять пользователю графический интерфейс с возможностью задавать:

a) размеры материала;

b) длину волокон;

c) количество волокон;

d) минимальное расстояние между волокнами;

e) размер конечного элемента.

2) в графическом интерфейсе должна быть возможность видеть в реальном времени количество добавленных волокон;

3) в графическом интерфейсе должна быть возможность прервать моделирование и записать текущую модель в файл;

4) система должна создавать k-файл (входной файл с готовой моделью для расчета в пакете программ LS-DYNA) с координатами узлов и конечными элементами (рис. 4).

```
*KEYWORD
*NODE
       1 0.000000000E+00 0.000000000E+00 0.000000000E+00       0       0
       2 1.000000000E+00 0.000000000E+00 0.000000000E+00       0       0
                                    ...
   46358 9.900000000E+01 5.000000000E+01 4.000000000E+00       0       0
   46359 1.000000000E+02 5.000000000E+01 4.000000000E+00       0       0
*ELEMENT_BEAM
       1       1   15328   15226       0
       2       1   15226   15225       0
                                   ...
     599       1   26105   26003       0
     600       1   26003   26002       0
*ELEMENT_SOLID
     601       3       1       2     102    5152    5152    5152    5152    5152
     602       3       2     103     102    5254    5254    5254    5254    5254
                                   ...
  200599       3   41208   41106   41107   46258   46258   46258   46258   46258
  200600       3   46358   46258   41106   41208   41208   41208   41208   41208
*END
```

Рис.4. Пример выходного файла

2.2. Нефункциональные требования к системе

К системе предъявляются следующие нефункциональные требования:

1) программа должна представлять собой исполняемый файл;

2) программа должна работать в операционных системах семейства Windows.

2.3. Диаграмма вариантов использования

На рис. 5 представлена диаграмма вариантов использования.

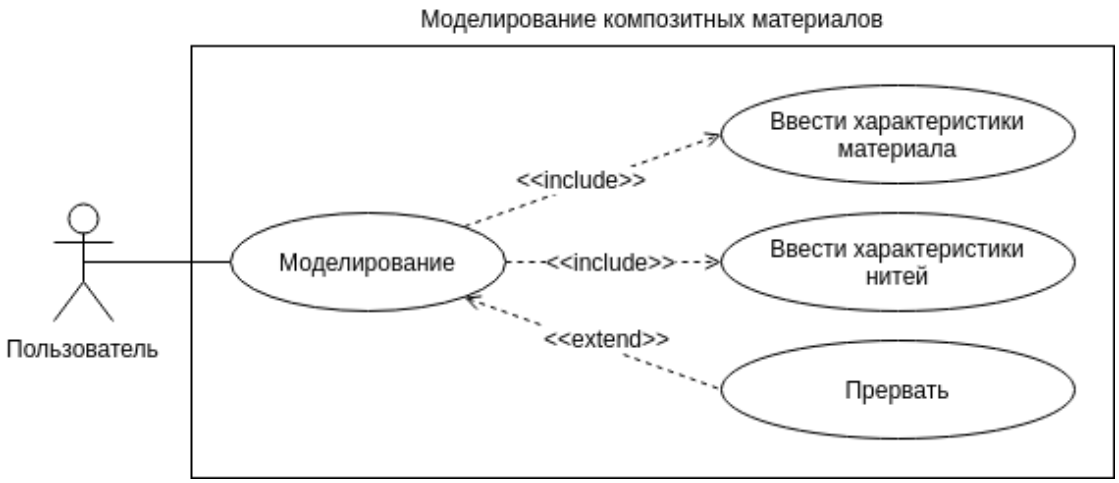

Рис.5. Диаграмма вариантов использования

Основные актеры, взаимодействующие с системой.

Пользователь – актер, который вводит характеристики материала, нитей и запускает моделирование.

Краткое описание вариантов использования.

Моделирование — процесс моделирования композитного материала на войлочной основе на основе входных данных.

Ввести характеристики материала — вариант использования, предоставляющий пользователю возможность ввести размер материала.

Ввести характеристики волокон — вариант использования, предоставляющий пользователю возможность ввести характеристики волокон: количество, длину, размер конечного элемента.

Прервать — вариант использования, предоставляющий пользователю возможность прервать процесс моделирования.

2.4. Спецификация основных вариантов использования

Основные варианты использования системы указаны в табл. 1.

Табл.1. Спецификация вариантов использования

UseCase: **Моделирование**
ID: 1
Аннотация: Моделирование композитного материала.
Главные актеры: Пользователь
Второстепенные актеры: Нет
Предусловия: Введены характеристики материала и волокон.
Основной поток: • Прецедент начинается, когда пользователь введет характеристики материала и нитей и нажмет на кнопку запуска моделирования. • Моделирование. • Система записывает получившуюся модель в k-файл.
Постусловия: Создан k-файл с моделью.
Альтернативные потоки: Нет

3. АРХИТЕКТУРА СИСТЕМЫ

3.1. Общее описание архитектуры системы

Система создания компьютерных моделей композитного материала на войлочной основе имеет графический интерфейс. В нем пользователь вводит входные данные и нажимает кнопку запуска моделирования. На первом шаге система генерирует тетраэдры, заполняющие весь объем модели, на втором шаге система генерирует волокна, а на третьем смещает волокна на случайную величину по всем трем осям. Во время генерации волокон в графический интерфейс выводится информация о количестве создаваемых волокон. Если пользователь нажимает на кнопку прерывания моделирования, программа создает последнее волокно и переходит к третьему этапу. После окончания 3 этапа создается выходной k-файл с моделью и пользователю выводится информация о результате моделирования.

При проектировании были рассмотрены несколько алгоритмов создания модели.

Первый алгоритм заключался в том, что изначально объем заполняется волокнами, после чего волокна разбиваются на конечные элементы. Каждый конечный элемент окружается тетраэдрами. После чего у каждой свободной грани создается тетраэдр. Новый тетраэдр сам создает новые грани. Таким образом количество свободных граней быстро растет. Добавление каждого нового тетраэдра занимает очень много времени, так как необходимо проверять пересечения со всеми соседними тетраэдрами. Таким образом алгоритм работает очень долго и не может создать модель необходимых размеров.

Второй алгоритм заключается в том, чтобы создать сетку точек и тетраэдров, а потом случайным образом генерировать волокна и пытаться привязать к ним тетраэдры, не смещая само волокно. Но этот алгоритм сильно меняет геометрию близко-лежащих тетраэдров, и для каждого

нового волокна остается меньше места на добавление. Таким образом не получается сгенерировать достаточное количество волокон в такой модели.

Таким образом был выбран третий алгоритм для реализации.

На рис. 6 представлена диаграмма классов анализа.

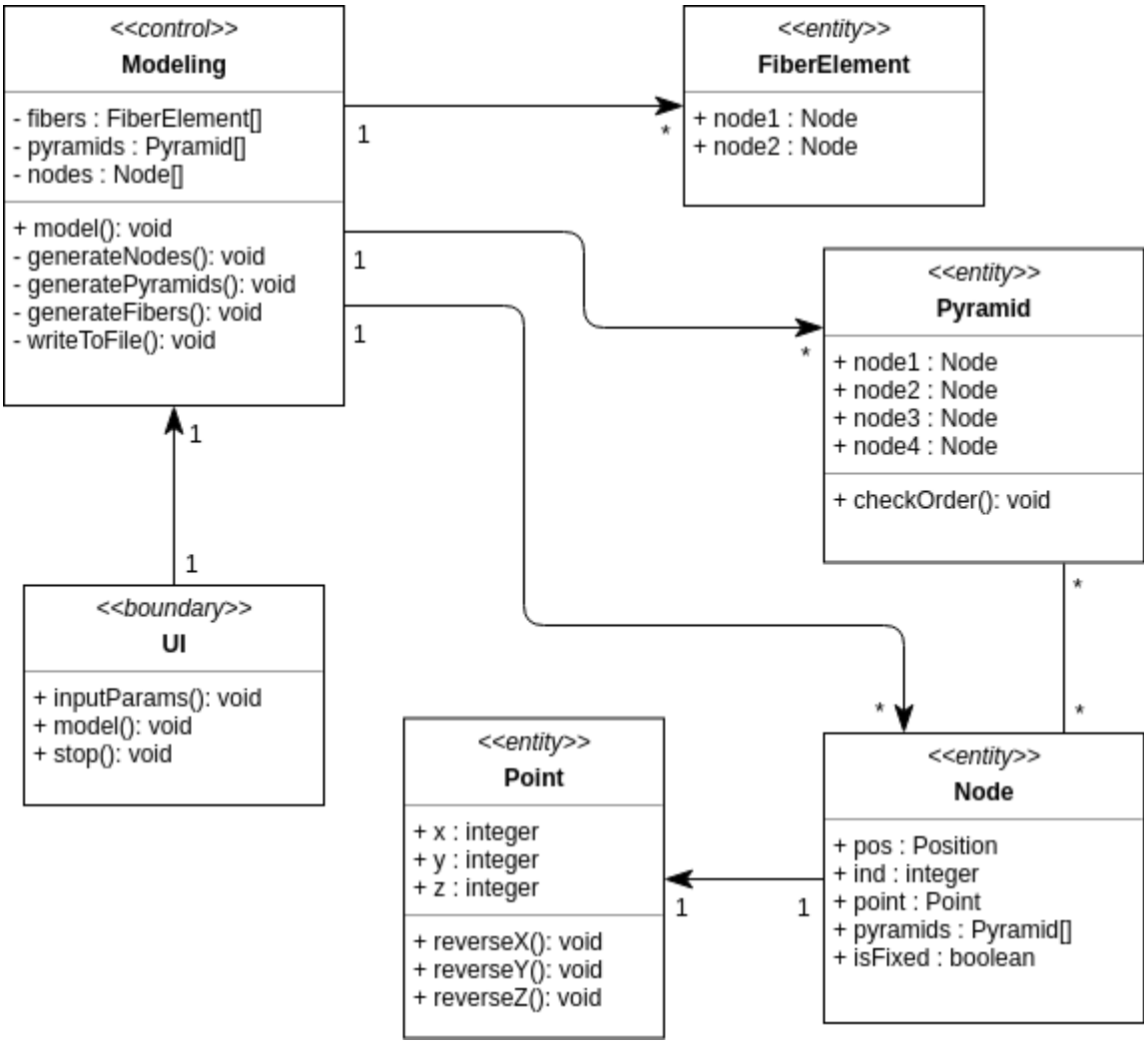

Рис.6. Диаграмма классов анализа

3.2. Описание компонентов, составляющих систему

UI – класс предоставляющий графический интерфейс.

Modeling – класс, отвечающий за:

- заполнение пространства точками;
- генерацию тетраэдров;
- генерацию волокон;
- проверку пересечения волокон;
- проверку пересечения тетраэдров;

- проверку расстояния между волокнами;
- запись модели в выходной k-файл.

FiberElement – класс, хранящий информацию о начале и конце конечного элемента — части волокна.

Point – класс, хранящий координаты точки в 3-мерной матрице.

Node – класс, хранящий информацию об абсолютных координатах вершины в пространстве.

Pyramid – класс, хранящий информацию о тетраэдре, координатах ее вершин.

3.3. Алгоритм работы

Алгоритм работы системы включает в себя следующие шаги.

1. Заполнение параллелепипеда с размерами, заданными пользователем, точками, находящимися на определенном расстоянии друг от друга.

2. Заполнение параллелепипеда тетраэдрами, вершины которых являются созданные на предыдущем этапе точки.

3. Получение случайной начальной точки и вычисление конечной на определенном расстоянии для построения волокна.

4. Вычисление кратчайшего пути между точками вдоль ребер тетраэдров.

5. Изменение координат точек, расположение их на одной прямой и на одинаковом расстоянии друг от друга, равном размеру конечного элемента.

6. Поиск пересечений тетраэдров и сдвиг соседних точек так, чтобы убрать эти пересечения.

7. Проверка пересечения нового волокна с уже добавленными ранее. Если пересечение найдено, то новое волокно не добавляется.

8. Проверка расстояния между новым волокном и уже добавленными ранее. Если оно меньше минимального, то новое волокно не добавляется.

9. Если не создано необходимое количество волокон и не нажата кнопка прерывания моделирование, то переход к пункту 3.

10. Перемещение волокна на случайные величины по всем трем осям. Если при перемещении образуются пересечения тетраэдров или расстояние между волокнами становится меньше минимального, то волокно не перемещается.

11. Запись координат узлов и конечных элементов в выходной k-файл.

4. РЕАЛИЗАЦИЯ

4.1. Технологии разработки

Было принято решение реализовывать программу на языках программирования Java [17] и Kotlin [18, 19]. Для сборки проекта в исполняемый jar файл и подключения зависимостей была выбрана система автоматической сборки Gradle [20]. Также было решено использовать математическую библиотеку Apache Commons Math [21], которая упрощает работу с трехмерными объектами, и в которой реализованы наиболее распространенные алгоритмы, например, поиск точки пересечения прямой и плоскости. Графический интерфейс был создан с помощью библиотеки Swing [25].

4.2. Пользовательский интерфейс

Пользовательский интерфейс (рис. 7) представляет собой форму ввода данных, необходимых для моделирования. Пользователь должен ввести характеристики материала и волокон. После этого по нажатию на кнопку «Вычислить» система начинает процесс моделирования на основе введенных параметров.

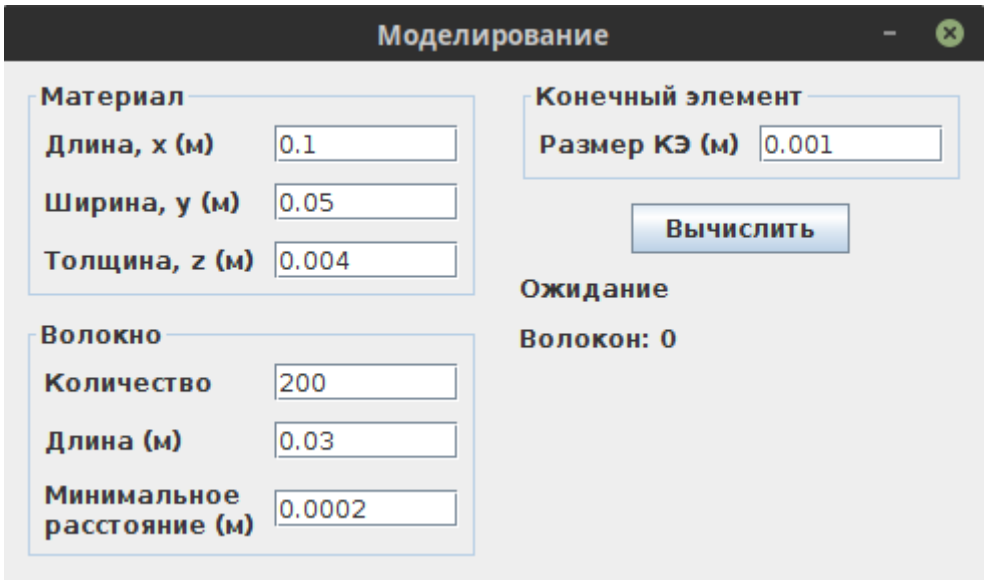

Рис.7. Пользовательский интерфейс

После начала моделирования вместо кнопки «Вычислить» появляется кнопка «Остановить» (рис. 8). По нажатию на нее программа

генерирует последнее волокно, после чего переходит на следующий этап моделирования.

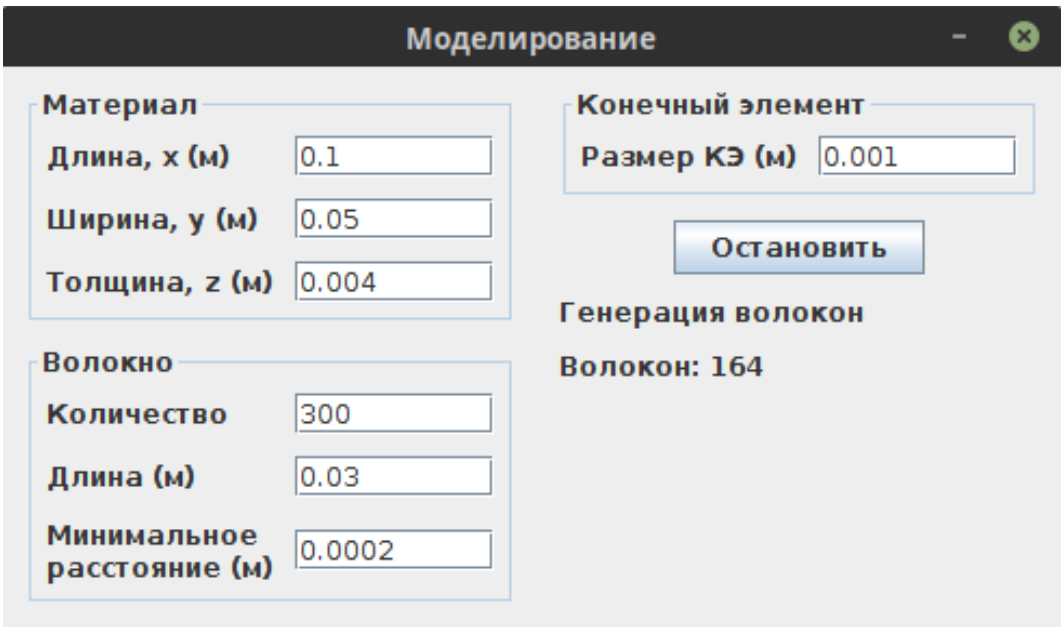

Рис.8. Пользовательский интерфейс во время моделирования

4.3. Модульная структура

Модульная структура системы представлена на рис. 9.

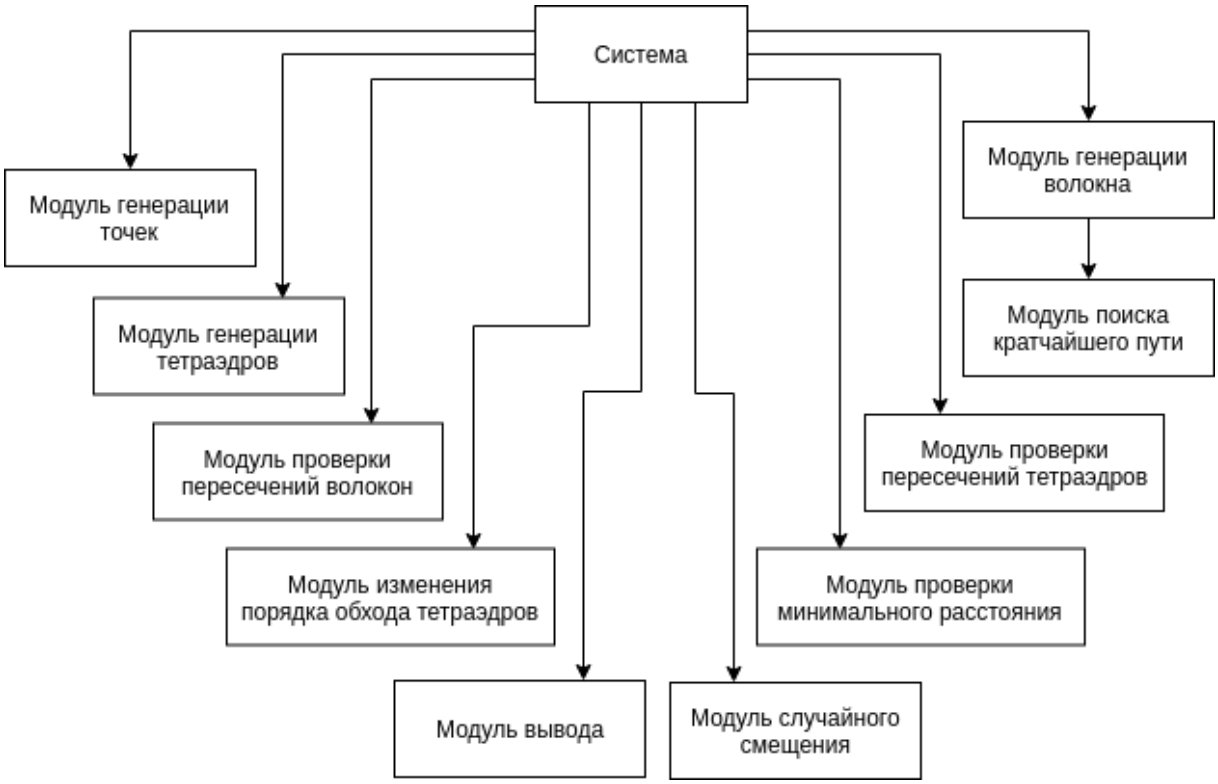

Рис.9. Модульная структура системы

Модуль генерации точек

Модуль заполняет заданный пользователем параллелепипед точками на расстоянии размера конечного элемента друг от друга (рис. 10).

```
fun generateNodes(nodeCountX: Int, nodeCountY: Int,
        nodeCountZ: Int, lX: Double, lY: Double,
        lZ: Double): Nodes {
    val nodes = Nodes(nodeCountX, nodeCountY, nodeCountZ)
    for (z in 0 until nodeCountZ)
        for (y in 0 until nodeCountY)
            for (x in 0 until nodeCountX)
                nodes[x, y, z] = Vector3D(x * lX, y * lY, z * lZ)
    return nodes
}
```

Рис.10. Метод, заполняющий объем точками

Модуль генерации тетраэдров

Модуль заполняет заданный пользователем параллелепипед тетраэдрами, вершина которых являются созданные точки на предыдущем этапе (рис. 11). В каждый параллелепипед, образованный восемью соседними точками входит 5 тетраэдров (рис. 12).

```
for (z in 0 until pyramidCountZ) {
    for (y in 0 until pyramidCountY) {
        for (x in 0 until pyramidCountX) {
        //задаются относительные координаты точек параллелепипеда
            val nd1 = Point(0, 0, 0)
            val nd2 = Point(1, 0, 0)
            val nd3 = Point(0, 1, 0)
            val nd4 = Point(1, 1, 0)
            val nd5 = Point(0, 0, 1)
            val nd6 = Point(1, 0, 1)
            val nd7 = Point(0, 1, 1)
            val nd8 = Point(1, 1, 1)

            //инвертирование нечетных тетраэдров по всем осям,
чтобы у соседних тетраэдров грани совпадали
            if (x % 2 == 1) {
                nd1.reverseX()
                nd2.reverseX()
                nd3.reverseX()
                nd4.reverseX()
                nd5.reverseX()
                nd6.reverseX()
                nd7.reverseX()
                nd8.reverseX()
            }
            if (y % 2 == 1) {
                nd1.reverseY()
                nd2.reverseY()
                nd3.reverseY()
```

```
                nd4.reverseY()
                nd5.reverseY()
                nd6.reverseY()
                nd7.reverseY()
                nd8.reverseY()
            }
            if (z % 2 == 1) {
                nd1.reverseZ()
                nd2.reverseZ()
                nd3.reverseZ()
                nd4.reverseZ()
                nd5.reverseZ()
                nd6.reverseZ()
                nd7.reverseZ()
                nd8.reverseZ()
            }

            //вычисление абсолютных координат вершин
параллелепипеда
            val n1 = nodes[x + nd1.x, y + nd1.y, z + nd1.z]
            val n2 = nodes[x + nd2.x, y + nd2.y, z + nd2.z]
            val n3 = nodes[x + nd3.x, y + nd3.y, z + nd3.z]
            val n4 = nodes[x + nd4.x, y + nd4.y, z + nd4.z]
            val n5 = nodes[x + nd5.x, y + nd5.y, z + nd5.z]
            val n6 = nodes[x + nd6.x, y + nd6.y, z + nd6.z]
            val n7 = nodes[x + nd7.x, y + nd7.y, z + nd7.z]
            val n8 = nodes[x + nd8.x, y + nd8.y, z + nd8.z]

            //создание пяти тетраэдров, формирующих параллелепипед
            pyramids.add(Pyramid(n1, n2, n3, n5))
            pyramids.add(Pyramid(n2, n3, n4, n8))
            pyramids.add(Pyramid(n5, n6, n8, n2))
            pyramids.add(Pyramid(n5, n7, n8, n3))
            pyramids.add(Pyramid(n2, n3, n8, n5))
        }
    }
}
```

Рис.11. Участок кода, генерирующий тетраэдры

Модуль поиска кратчайшего пути

Модуль ищет кратчайший путь между двумя точками, проходящий по ребрам тетраэдров. Это рекурсивный метод, которому на вход подается текущая и конечная точка. Из всех ребер, выходящих из текущей точки, выбираются те, другие концы которых находятся ближе к конечной точке, и делается рекурсивный вызов для них. При этом считается сумма расстояний от каждой точки до волокна и в конце выбирается путь с наименьшей суммой расстояний. При выборе точек, учитывается то, что путь не может содержать два ребра одного тетраэдра.

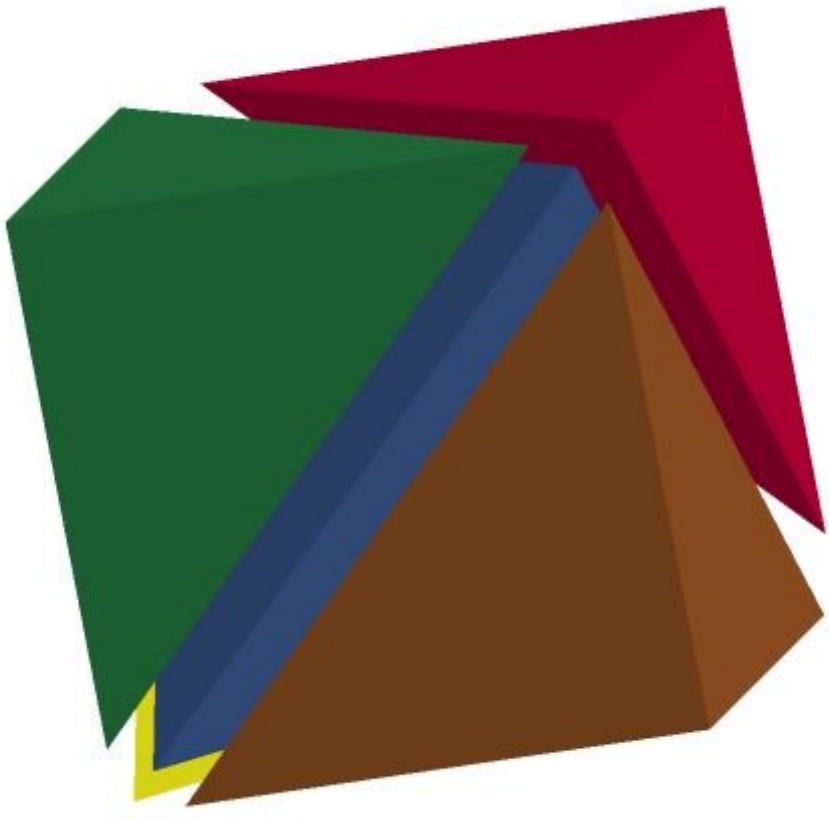

Рис.12. Параллелепипед, образованный пятью тетраэдрами

Модуль генерации волокна

Модуль создает волокно со случайными начальными координатами, смещением по одной из оси (x или y), равным длине волокна и случайным смещением по другой оси (y или x) (рис. 13). Диапазон случайного смещения меняется во время работы программы. Сначала модуль пытается добавлять волокна с большим наклоном, а когда он не может найти место для нового такого волокна, то пытается добавлять уже более горизонтальные и вертикальные волокна. После этого вычисляется кратчайший путь между начальной и конечной точками с помощью предыдущего модуля.

Модуль проверки пересечений тетраэдров

Модуль проверяет, пересекают ли одни тетраэдры другие после изменения их геометрии во время привязки волокна к тетраэдру (рис. 14). Если пересечения найдены, то он пытается перенести точки, не

принадлежащие волокну, в противоположную от него сторону, чтобы пересечения исчезли. Новые перенесенные точки также могут пересечь тетраэдры. Поэтому они тоже проверяются. Если после переносов пересечения все равно остаются, то волокно не добавляется в итоговую модель.

```
val dir = Random.nextInt(8)

val c = Random.nextInt(max - min + 1) + min

val dirX = Random.nextInt().sign()
val dirY = Random.nextInt().sign()

val lengthX: Int
val lengthY: Int
if (dir == 0) {
    lengthX = c
    lengthY = finiteElementCount
} else {
    lengthX = finiteElementCount
    lengthY = c
}

var x = Random.nextInt(nodeCountX - lengthX - 2) + 1
if (dirX < 0)
    x += lengthX
var y = Random.nextInt(nodeCountY - lengthY - 2) + 1
if (dirY < 0)
    y += lengthY
val z = Random.nextInt(nodeCountZ - 2) + 1

val endX = x + lengthX * dirX
val endY = y + lengthY * dirY
val endZ = z

val startNode = nodes[x, y, z]
val endNode = nodes[endX, endY, endZ]
```

Рис.13. Участок кода, выбирающий две точки для построения волокна

```
val node = nodes[x - 1, y, z]
if (selectedNode.pos.x < node.pos.x) {
    if (node.isFixed)
        return false

    node.add(Vector3D(
        selectedNode.pos.x - node.pos.x - finiteElementSize / 10,
        selectedNode.pos.y - node.pos.y, 0.0
    ))
    node.isFixed = true
    checkNodes.add(node)
}
```

Рис.14. Участок кода, проверяющий пересечение с левым тетраэдром

Модуль проверки пересечений волокон

Модуль проверяет пересечения созданного волокна с уже существующими (рис. 15). Если пересечения есть, то данное волокно не добавляется в модель.

```
private fun checkFixedElementNodes(finiteElementNodes: List<Node>):
        Boolean {
    for (node in finiteElementNodes)
        if (node.isFixed)
            return false

    for (node in finiteElementNodes) {
        node.pyramids.forEach { pyramid ->
            val node1 = pyramid.node1
            val node2 = pyramid.node2
            val node3 = pyramid.node3
            val node4 = pyramid.node4
            if (node1.pos.z == node.pos.z && node1.isFixed)
                return false
            if (node2.pos.z == node.pos.z && node2.isFixed)
                return false
            if (node3.pos.z == node.pos.z && node3.isFixed)
                return false
            if (node4.pos.z == node.pos.z && node4.isFixed)
                return false
        }
    }

    return true
}
```

Рис.15. Метод, проверяющий пересечения нового волокна с существующими

Модуль проверки минимального расстояния

Модуль вычисляет минимальное расстояние между двумя волокнами.

Модуль случайного смещения

Модуль выбирает случайное волокно и случайные вектора перемещения для первой и последней точки волокна. После этого перемещения точек алгоритм строит прямую по ним и перемещает конечные элементы волокна вдоль этой прямой. Если при перемещении появились новых пересечений тетраэдров, то волокно не перемещается.

Модуль изменения порядка обхода тетраэдров

Модуль определяет порядок обхода вершин тетраэдров таким

образом, чтобы первая грань обходилась по часовой стрелке (рис. 16). Для этого он строит плоскость из трех первых точек. Вычисляет вектор нормали к ней. Строит прямую вдоль этого вектора, проходящую через четвертую вершину. Вычисляет точку пересечения прямой и плоскости. Вычисляет вектор нормали в сторону от 4 вершины. Вычисляет векторное произведение двух векторов двух прямых плоскости. Сравнивает вектора, если они не совпадают, то необходимо поменять две вершины плоскости местами.

```
fun checkOrder() {
    val node1Pos = node1.pos
    val node2Pos = node2.pos
    val node3Pos = node3.pos
    val node4Pos = node4.pos

    val plane = Plane(node1Pos, node2Pos, node3Pos, TOLERANCE)

    val line = Line(node4Pos, node4Pos.add(plane.normal),
        TOLERANCE)
    //точка пересечения плоскости и нормали к ней, проходящей через
четвертую точку тетраэдра
    val point = plane.intersection(line)

    val vec12 = node2Pos.subtract(node1Pos)
    //векторы двух сторон тетраэдра, принадлежащих плоскости
    val vec23 = node3Pos.subtract(node2Pos)

    //вектор, направленный от четвертой точки к плоскости
    val vec1 = node4Pos.subtract(point).normalize()
    //векторное произведение - нормаль к плоскости
    val vec2 = vec12.crossProduct(vec23).normalize()

    //проверка совпадения векторов и изменение направления обхода
плоскости
    if (vec1.x.sign() != vec2.x.sign() || vec1.y.sign() !=
            vec2.y.sign() || vec1.z.sign() != vec2.z.sign()) {
        val temp = node2
        node2 = node3
        node3 = temp
    }
}
```

Рис.16. Метод, меняющий порядок обхода вершин тетраэдра

Модуль вывода

Модуль создает новый k-файл в папке, в которой находится программа, и записывает в него координаты узлов и конечных элементов.

5. ТЕСТИРОВАНИЕ

В ходе тестирования приложения использовались следующие методы:

- тестирование интерфейса пользователя;
- модульное тестирование;
- функциональное тестирование.

Во время тестирования графического интерфейса была проведена проверка работы всех графических элементов, таких как форма, поля ввода, кнопка, на всех стадиях моделирования.

Во время модульного тестирования были проверены на корректную работу отдельные модули программы. Несколько тестов приведены ниже.

```
fun checkOrder()
```

Этот метод проверяет порядок обхода вершин тетраэдра. Если первая грань обходится против часовой стрелки, то две вершины этой грани меняются местами. Node1, node2, node3, node4 – четыре вершины тетраэдра, являющиеся внешними переменными. Результат тестирования приведен в табл. 2.

Табл.2. Результат модульного тестирования

№ теста	Входные данные	Выходные данные	Ожидаемый результат	Тест пройден?
1	node1=(0, 0, 0) node2=(1, 0, 0) node3=(0, 1, 0) node4=(0, 0, 1)	node1=(0, 0, 0) node2=(1, 0, 0) node3=(0, 1, 0) node4=(0, 0, 1)	node1=(0, 0, 0) node2=(1, 0, 0) node3=(0, 1, 0) node4=(0, 0, 1)	Да
2	node1=(0, 0, 0) node2=(0, 1, 0) node3=(1, 0, 0) node4=(0, 0, 1)	node1=(0, 0, 0) node2=(1, 0, 0) node3=(0, 1, 0) node4=(0, 0, 1)	node1=(0, 0, 0) node2=(1, 0, 0) node3=(0, 1, 0) node4=(0, 0, 1)	Да
3	node1=(5, 2, -10) node2=(10, 2, 0) node3=(1, 10, 0) node4=(-10, -10, 0)	node1=(5, 2, -10) node2=(10, 2, 0) node3=(1, 10, 0) node4=(-10, -10, 0)	node1=(5, 2, -10) node2=(10, 2, 0) node3=(1, 10, 0) node4=(-10, -10, 0)	Да
4	node1=(5, 2, -10) node2=(10, 2, 0)	node1=(5, 2, -10) node2=(1, 10, 0)	node1=(5, 2, -10) node2=(1, 10, 0)	Да

	node3=(1, 10, 0) node4=(10, 10, 0)	node3=(10, 2, 0) node4=(10, 10, 0)	node3=(10, 2, 0) node4=(10, 10, 0)	
5	node1=(0, 0, 0) node2=(0, 1, 0) node3=(1, 1, 1) node4=(0, 0, 1)	node1=(0, 0, 0) node2=(1, 1, 1) node3=(0, 1, 0) node4=(0, 0, 1)	node1=(0, 0, 0) node2=(1, 1, 1) node3=(0, 1, 0) node4=(0, 0, 1)	Да

```
fun calcHash(node1: Node, node2: Node, node3: Node, node4: Node): Int
```

Этот метод отвечает за вычисление хэш-функции для тетраэдра из четырех вершин. Вершины сортируются по ее индексу и эти индексы складываются с определенными коэффициентами. Node1, node2, node3, node4 – четыре вершины тетраэдра. Результат тестирования приведен в табл. 3.

Табл.3. Результат модульного тестирования

№ теста	Входные данные	Выходные данные	Ожидаемый результат	Тест пройден?
1	node1=1 node2=2 node3=3 node4=4	102034	102034	Да
2	node1=10 node2=5 node3=3 node4=1	103060	103060	Да
3	node1=1 node2=0 node3=3 node4=8	1038	1038	Да
4	node1=10 node2=9 node3=8 node4=7	708100	708100	Да

```
fun matrixToLinear(x: Int, y: Int, z: Int): Int
```

Этот метод отвечает за перевод координат точки в 3-мерной матрице в индекс одномерного массива. Для вычисления используются глобальные

переменные countX, countY, countZ, определяющие размер 3-мерной матрицы. Результат тестирования приведен в табл. 4.

Табл.4. Результат модульного тестирования

№ теста	Входные данные	Выходные данные	Ожидаемый результат	Тест пройден?
1	x=0 y=0 z=0 countX=10 countY=10 countZ=10	0	0	Да
2	x=1 y=2 z=3 countX=10 countY=10 countZ=10	321	321	Да
3	x=9 y=9 z=9 countX=10 countY=10 countZ=10	999	999	Да
4	x=24 y=25 z=2 countX=100 countY=50 countZ=4	12524	12524	Да

`fun getDistanceBetweenLines(p11: Vector3D, p12: Vector3D, p21: Vector3D, p22: Vector3D): Double`

Этот метод вычисляет расстояние между двумя отрезками в пространстве. p11 и p12 — координаты начала и конца первого отрезка, p21 и p22 — второго. Результат тестирования приведен в табл. 5.

Во время функционального тестирования была проведена проверка соответствия реализованной системы поставленным требованиям.

В первом тесте была создана модель размером по 10 тетраэдров в длину и ширину и 8 тетраэдров в высоту. Для визуализации тетраэдры были окрашены в разные цвета. Визуализация входных данных изображена на рис. 17, а результат тестирования на рис. 18.

Табл.5. Результат модульного тестирования

№ теста	Входные данные	Выходные данные	Ожидаемый результат	Тест пройден?
1	node1=(0, 0, 0) node2=(0, 0, 1) node3=(0, 0, 0) node4=(0, 1, 0)	0	0	Да
2	node1=(0, 0, 0) node2=(0, 0, 1) node3=(1, 0, 0) node4=(1, 1, 1)	1	1	Да
3	node1=(0, 0, 0) node2=(0, 0, 1) node3=(0.5, 0, 0.5) node4=(1, 0, 0.5)	0.5	0.5	Да
4	node1=(0, 0, 0) node2=(1, 0, 1) node3=(0.5, 1, 0) node4=(0.5, -1, 0)	0.353553	0.353553	Да

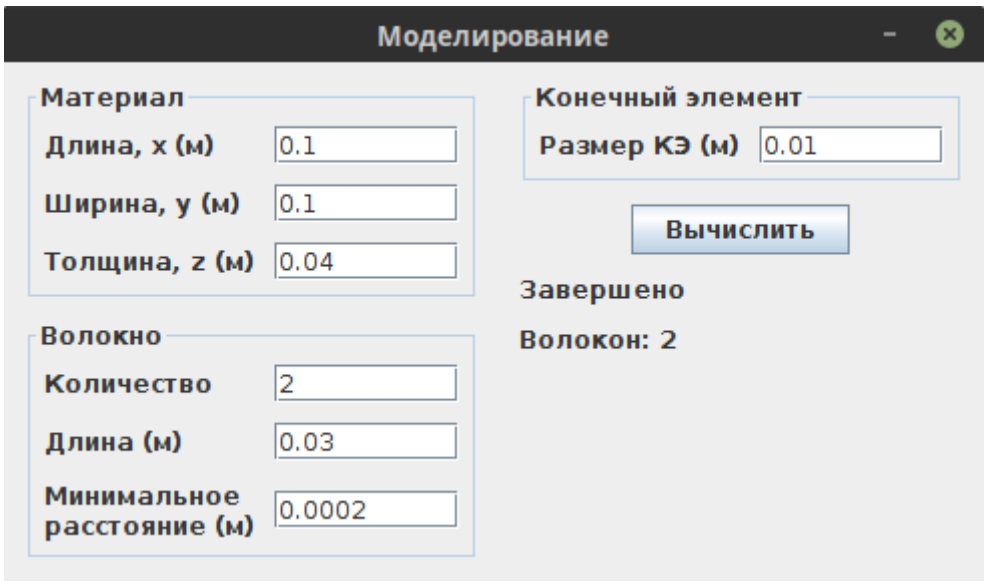

Рис.17. Пример входных данных для визуализации небольшого материала

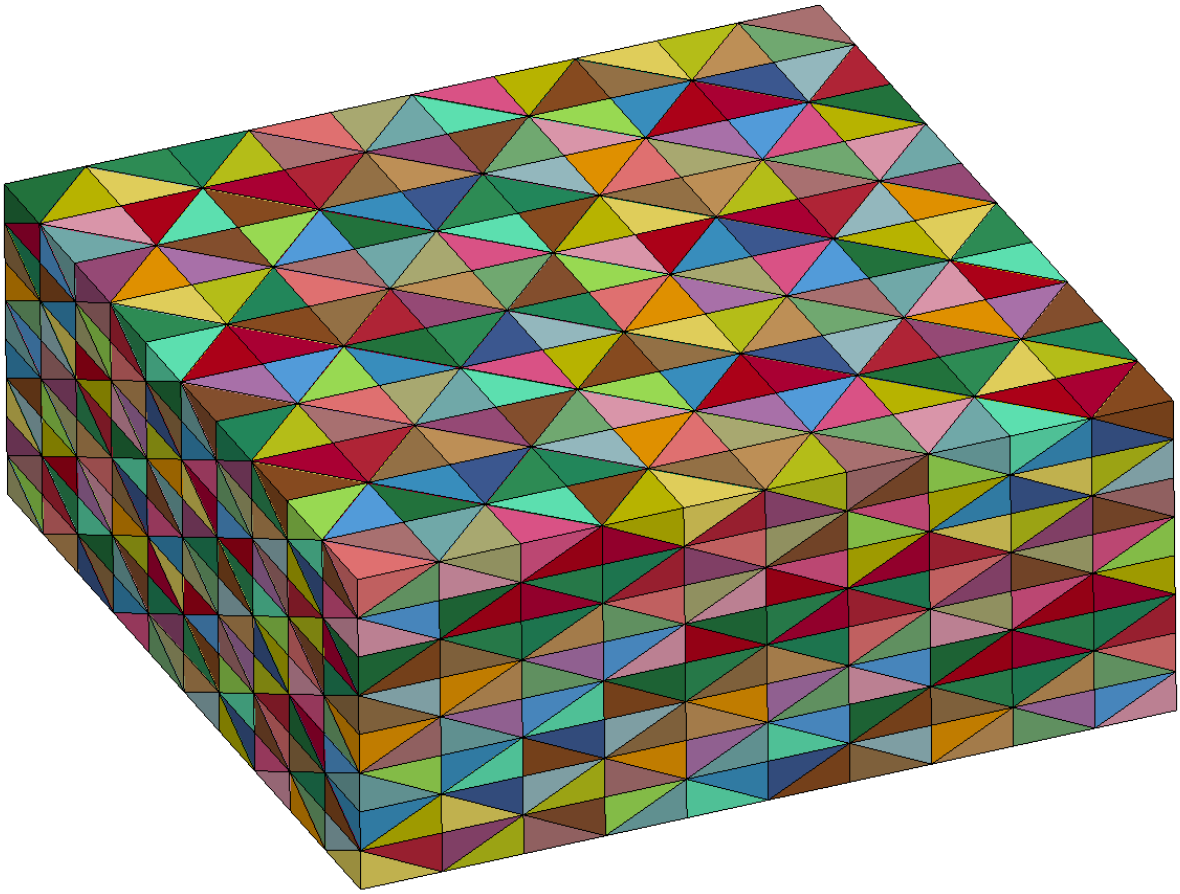

Рис.18. Результат моделирования материала с разноцветными тетраэдрами

Во втором и третьем тестах было создано по одному волокну для проверки изменения координат вершин. Визуализация входных данных изображена на рис. 19, результат тестирования на рис. 20, 21.

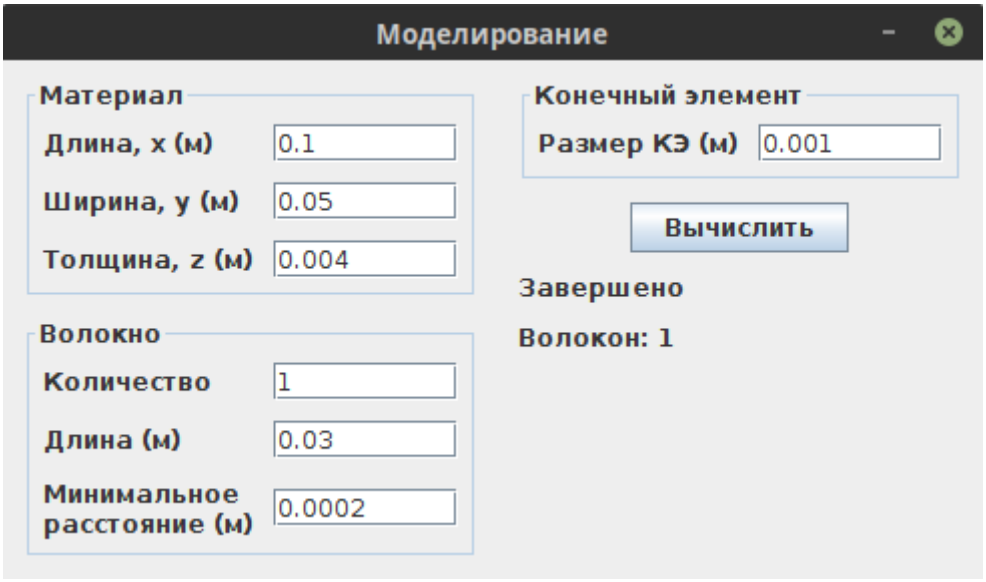

Рис.19. Пример входных данных для материала с одним волокном

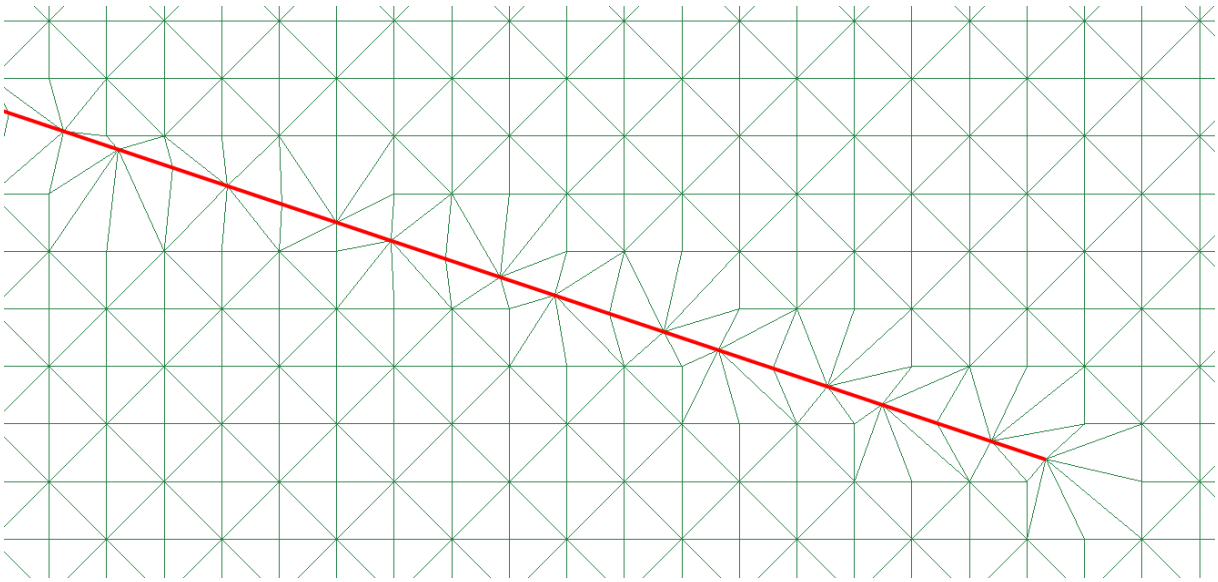

Рис.20. Результат моделирования привязки волокна к тетраэдрам с изменением их геометрии

В четвертом тесте была создана небольшая модель для проверки пересечений волокон и тетраэдров. Визуализация входных данных изображена на рис. 22, а результат на рис. 23.

В пятом тесте была создана модель с реальными размерами. Визуализация входных данных изображена на рис. 24, результат на рис. 25.

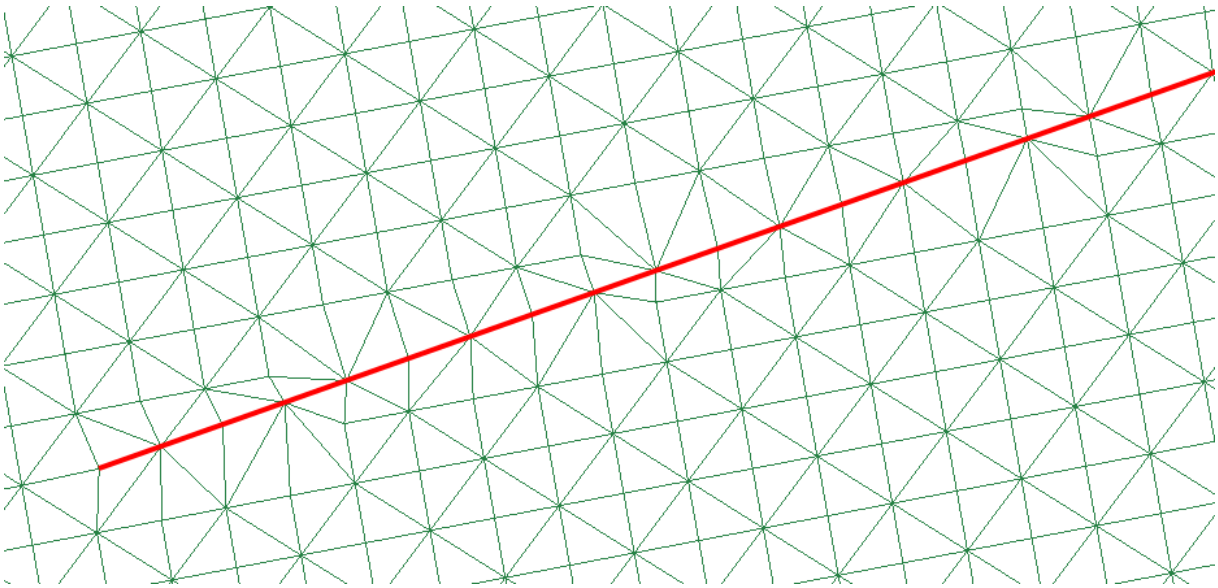

Рис.21. Результат моделирования привязки волокна к тетраэдрам с изменением их геометрии

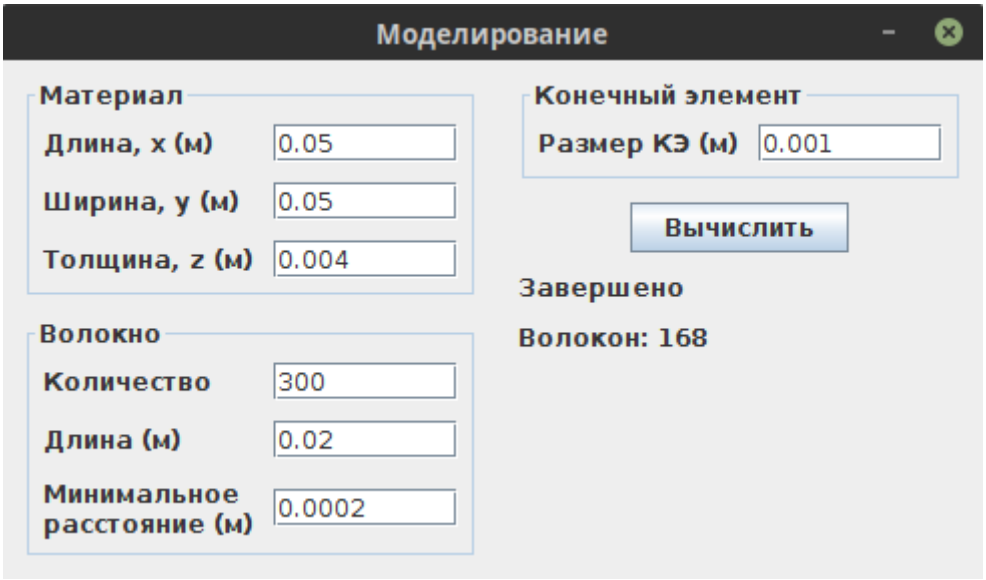

Рис.22. Пример входных данных небольшой модели для проверки пересечений тетраэдров

Рис.23. Небольшая модель для проверки пересечений тетраэдров

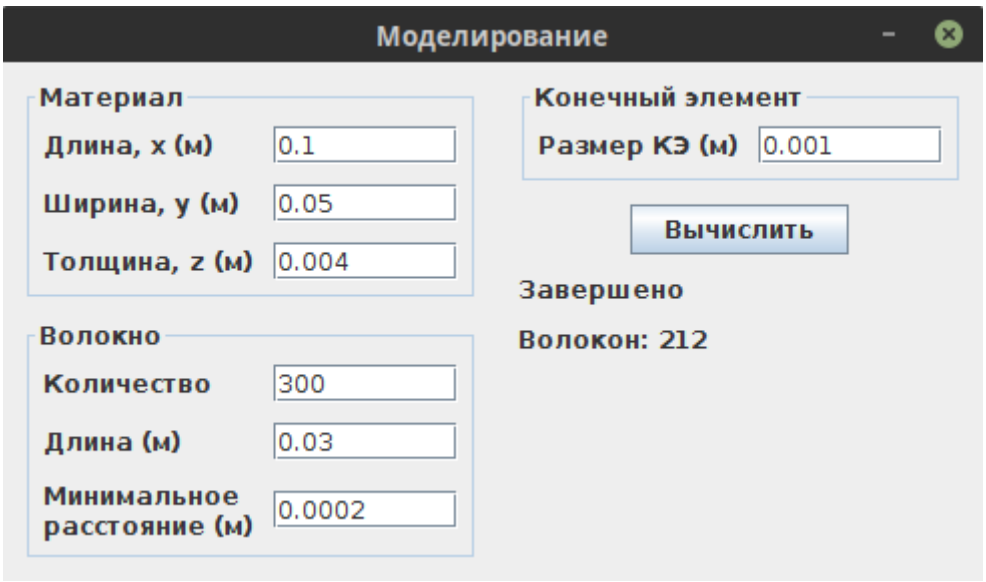

Рис.24. Пример входных данных материала реального размера

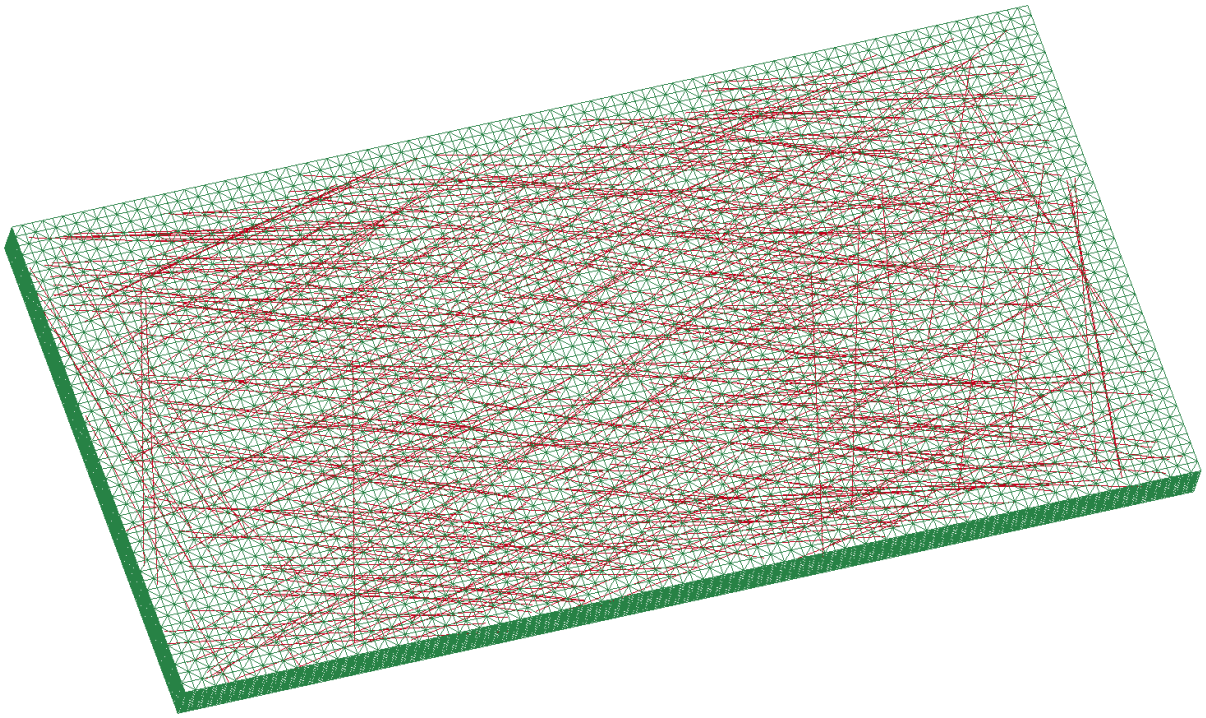

Рис.25. Результат моделирования материала реального размера

ЗАКЛЮЧЕНИЕ

В результате работы была разработана программная система для создания компьютерных моделей композитных материалов на войлочной основе.

В ходе работы были выполнены следующие задачи:

1) изучены свойства композитных материалов на войлочной основе;
2) разработан алгоритм создания компьютерных моделей композитных материалов на войлочной основе;
3) спроектирована и реализована программа;
4) программа протестирована.

Таким образом, все поставленные цели были достигнуты.

ЛИТЕРАТУРА

1. Бондалетова Л.И. Полимерные композиционные материалы (часть 1): учебное пособие / Л.И. Бондалетова, В.Г. Бондалетов. – Томск: Изд-во Томского политехнического университета, 2013. – 118 с.

2. Chocron S. Characterization of Fraglight non-woven felt and simulation of FSP’s impact in it. / S. Chocron, F. Gálvez, D. Cendón, C. Roselló. // Final report for the European Research Office of the US Army, Universidad Politecnica de Madrid, 2002. – P. 59.

3. Bouchoucha F., Ichchou M., Haddar M. Diffusion matrix through stochastic wave finite element method. // Finite Elements in Analysis and Design, 2013. – № 64. – P. 97–107.

4. Demirci E. Computation of mechanical anisotropy in thermally bonded bicomponent fibre nonwovens. / E. Demirci, M. Acar, B. Pourdeyhimi, V.V. Silberschmidt. // Computational Materials Science, 2012. – Vol. 52. – P. 157–163.

5. Demirci E. Finite element modelling of thermally bonded bicomponent fibre nonwovens: tensile behaviour. / E. Demirci, M. Acar, B. Pourdeyhimi, V.V. Silberschmidt. // Computational Materials Science, 2011. – Vol. 50. – P. 1286–1291.

6. Dolganina N.Yu., Kudryavtsev O.A., Sapozhnikov S.B. An assessment of the aramid felt high velocity impact resistance. // Proceedings of 20th International Conference on Composite Materials ICCM20 (19–24 July 2015, Copenhagen, Denmark), 2015. – P. 1–10.

7. Farukh F. Numerical Analysis of Progressive Damage in Nonwoven Fibrous Networks under Tension. / F. Farukh, E. Demirci, B. Sabuncuoglu, M. Acar, B. Pourdeyhimi, V.V. Silberschmidt. // International Journal of Solids and Structures, 2014. – Vol. 51. – P. 1670–1685.

8. Silberstein M.N. Elastic–plastic behavior of non-woven fibrous mats. / M.N. Silberstein, Chia-ling Pai, G.C. Rutledge, M.C. Boyce. // Journal of the Mechanics and Physics of Solids, 2012. – Vol. 60. – P. 295–318.

9. Jearanaisilawon P. A continuum model for needlepunched nonwoven fabrics. Ph.D. thesis. – USA: Massachusetts Institute of Technology, 2008. – P. 27–56.

10. Ridruejo A., González C., Llorca J. A constitutive model for the in-plane mechanical behavior of nonwoven fabrics. // International Journal of Solids and Structures, 2012. – Vol. 49. – P. 2215–2229.

11. Ridruejo A., González C., Llorca J. Damage micromechanisms and notch sensitivity of glass-fiber non-woven felts: an experimental and numerical study. // Journal of the Mechanics and Physics of Solids, 2010. – Vol. 58. – P. 1628–1645.

12. Ridruejo A., González C., Llorca J. Failure locus of polypropylene nonwoven fabrics under in-plane biaxial deformation. // Comptes Rendus Mécanique, 2012. – Vol. 340. – P. 307–319.

13. Трушин С. Метод конечных элементов. Теория и задачи. – М.: Издательство Ассоциации строительных вузов, 2008. – С. 256..

14. Ridruejo A. Mechanical behaviour of nonwoven felts. // Ph.D. thesis, Universidad Politécnica de Madrid, 2011.

15. Долганина Н.Ю., Сапожников С.Б. Исследование влияния типа переплетения нитей на прочность тканевых преград при локальном ударе. // Вестник Южно-Уральского государственного университета. Серия: Машиностроение, 2013. – Т. 13. – № 2. – С. 95–104.

16. Sabuncuoglu B., Acar M., Silberschmidt V.V. Finite Element Modelling of Fibrous Networks: Analysis of Strain Distribution in Fibres under Tensile Load. // Computational Materials Science, 2013. – Vol. 79. – P. 143–158.

17. Документация языка программирования Java. [Электронный ресурс] URL: https://docs.oracle.com/javase/7/docs/ (дата обращения: 18.05.2019).

18. Документация языка программирования Kotlin. [Электронный ресурс] URL: https://kotlinlang.ru/docs/reference/kotlin-doc.html (дата обращения: 18.05.2019).

19. Руководство по языку программирования Kotlin. [Электронный ресурс] URL: https://metanit.com/java/kotlin/ (дата обращения: 18.05.2019).

20. Система автоматической сборки. [Электронный ресурс] URL: https://docs.gradle.org/current/userguide/userguide.html (дата обращения: 18.05.2019).

21. Математическая библиотека Apache Math3. [Электронный ресурс] URL: https://commons.apache.org/proper/commons-math/javadocs/api-3.6.1 (дата обращения: 18.05.2019).

22. Долганина Н.Ю., Кибель М.О. Разработка компьютерных моделей войлочных материалов, применяемых в бронезащитных элементах. // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика, 2016. – Т. 5. – № 2. – С. 95–104.

23. Ridruejo A., González C., Llorca J. Micromechanisms of deformation and fracture of polypropylene nonwoven fabrics. // International Journal of Solids and Structures, 2011. – Vol. 48. – P. 153–162.

24. Heighway E.A., Biddlecombe C.S. Two-dimensional authomatic triangular mesh generation for the finite element electromagnetics package PE2D. // IEEE Trans. on Mag, 1982. – P. 594–598.

25. Руководство по библиотеке для создания графического интерфейса Swing. [Электронный ресурс] URL: https://docs.oracle.com/javase/tutorial/uiswing (дата обращения: 18.05.2019).