ОТЧЕТ

за второй (заключительный) этап выполнения НИОКР по теме:

«Разработка учебно-демонстрационного стенда для изучения  
технологических процессов изготовления деталей из полимерных  
композиционных материалов методом вакуумной инфузии»

# Разработка алгоритмов учебно-демонстрационной программы.

## Моделирование процесса вакуумной инфузии методом клеточных автоматов

### Вычисление давления в узлах

Течение связующего через армирующий наполнитель моделируется законом Дарси, который устанавливает линейное отношение между скоростью потока и градиентом прикладываемого давления:

*u* = – [*K*] / μ · *p*, (1.1)

где u – скорость потока, м/с; [K] – тензор проницаемости, м2; μ – вязкость смолы, Па · с; *p* – давление в смоле, Па;  – оператор Гамильтона.

вверх.

Также необходимо учитывать уравнение неразрывности:

 · *u* = 0. (1.2)

Подставляя (1.1) в (1.2), исключаем *u* и получаем уравнение для определения давления *p*:

 · { ( 1/μ ) [*K*] *p* } = 0. (1.3)

Уравнение (1.3) является эллиптическим уравнением второго порядка в частных производных относительно давления.

В данном продукте используется подход к имитационному моделированию, основанный на идее клеточных автоматов. Клеточные автоматы были предложены в 1940 году Джоном фон Нейманом, Станиславом Уламом и Аланом Тьюрингом.

Клеточные автоматы представляют собой пространственно дискретную систему, зависящую от времени, в которой каждое следующее состояние индивидуальной клетки через заданный период времени зависит от её собственного предыдущего состояния и предыдущих состояний конечного числа заданных соседних клеток.

В сравнении с наиболее популярным методом моделирования физических явлений, описываемых уравнениями в частных производных, методом конечных элементов, метод клеточных автоматов имеет как достоинства, так и недостатки.

Основным и наиболее существенным достоинством является меньший объём необходимых вычислений. Это приводит к сокращению времени моделирования и снижению требований к аппаратному обеспечению компьютера, на котором производится моделирование.

К недостаткам можно отнести более низкую точность. Также при моделировании с использованием клеточных автоматов для сохранения точности плотность расположения клеток (узлов сетки, на которые разбита деталь) должна быть максимально однородной. Метод конечных элементов же требует разбиения детали на треугольники, максимально близкие к равносторонним.

На основе закона Дарси была выведена формула [1] для обновления значения давления в клетке:

, (1.4)

где *pt* – давление в данной клетке на шаге *t*, Па;

*pi,t* – давление в *i*-й соседней клетке на шаге *t*, Па;

*K* - проницаемость для данной клетки, м2;

- средняя проницаемость, м2;

*d* – толщина (высота полости) для данной клетки, м;

*di* – толщина (высота полости) для *i*-й соседней клетки, м;

П – пористость для данной клетки, безразмерная;

П *i* – пористость для *i*-й соседней клетки, безразмерная;

*li* – расстояние до *i*-й соседней клетки, м;

– среднее расстояние между клетками, м;

*q*, *r*, *s* – весовые коэффициенты, приняты равными 1, 1 и 2 соответственно;

*m* – число соседей.

Все влияющие параметры, такие как проницаемость (часть *A* формулы (1.4)), рабочий объём клетки, представленный высотой полости и пористостью (часть *B*), а также расстояния между клетками (часть C) нормализуются. Часть *A* учитывает локальную проницаемость клетки. Часть *B* задаёт вес клетки относительно её соседей, в зависимости от их параметров. Часть *C* делает клеточный автомат независимым от плотности клеток, т.е. позволяет сделать поведение заполнения всегда одинаковым для клеток с одинаковыми параметрами, вне зависимости от плотности.

Параметр *m* предписывает число соседних клеток, учитываемых при расчёте давления, т.е. одно и то же число соседей используется, вне зависимости от их плотности.

Весовые коэффициенты *q*, *r* и *s* должны быть подобраны так, чтобы обеспечить наилучшую аппроксимацию поведения реального потока.

В данном алгоритме используется изотропное значение проницаемости *K*. В случае необходимости клеточный автомат может быть расширен для использования анизотропной проницаемости.

Для улучшения взаимодействия между слоями с различной величиной проницаемости было решено в формулу (1.4) добавить ещё один множитель min(*Ki* / *K*, 1):

. (1.5)

### Определение степени заполнения узла

Изначально все клетки являются пустыми и имеют нулевое давление связующего *p*. В клетках, непосредственно соединённых с устройством подачи связующего, давление является постоянным и в случае вакуумной инфузии равно атмосферному давлению *patm*. Когда расчёт начинается, это давление распространяется через окружающие точки. В точке вакуумирования давление связующего не может подняться выше создаваемого давления вакуума *pvac*. На каждом шаге расчёта оцениваются величина давления *p* в клетках и степень её заполнения *f*. Степень заполнения *f* определяется как отношение давления связующего в клетке *p* к давлению вакуума *pvac*. Клетка считается заполненной, если в ней давление связующего *p* достигло давления вакуума *pvac* (1.6).

. (1.6)

Вычисления оканчиваются, когда все клетки становятся заполненными (для всех клеток *f* = 1), либо на последнем шаге не обновилась величина давления ни для одной клетки (для всех клеток *pt* = *pt*+1).

### Вычисление времени шага

Шаги вычисления имеет переменную длительность. Длительность *t*-го шага определяется по формуле (1.7):

, (1.7)

где τ*t* – длительность шага t, с;

μ – вязкость связующего (смолы), Па · с:

*nt* – число полностью заполненных клеток (для которых *f* = 1) на шаге *t*;

– среднее расстояние между клетками, м;

- средняя проницаемость, м2;

*patm* – давление, с которым осуществляется подвод связующего, Па;

*pvac* – давление вакуума, Па.

Точность подсчёта времени с использованием (1.7) имеет сильную зависимость от однородности плотности пространственного расположения клеток.

### Вычисление вязкости связующего

Вязкость связующего имеет зависимость от температуры. Связь вязкости и температуры может быть выражена при помощи формулы Френкеля-Андраде [2], которое основывается на соображениях о движениях молекул:

, (1.8)

где:

μ – динамическая вязкость, Па · с;

*T* – температура, К;

*E*ав – энергия активации вязкого течения, Дж / моль;

*A* – константа, определяемая природой жидкости, Па · с;

*R* = 8,314 Дж/(К · моль) – универсальная газовая постоянная.

Формулу Френкеля-Андраде (1.8) можно преобразовать для вычисления вязкости относительно некоторой номинальной вязкости μ(*T*0) = μ0, измеренной при номинальной температуре *T*0 (в данном случае нужно принять *T*0 = 298,15 К = 25 **°**C):

, (1.9)

Если разделить (1.9) на (1.8) и выразить , можно получить формулу для вычисления вязкости с учётом температурной зависимости:

. (1.10)

### Вычисление времени жизни связующего

При тепловом воздействии на термореактивные полимеры их вязкость с некоторого момента времени начинает интенсивно возрастать, что свидетельствует о начале процесса гелеобразования.

Использовать связующее для изготовления изделий или для пропитки необходимо в промежутке времени от момента приготовления связующего до момента, когда оно переходит в гелеобразное состояние. Такой промежуток времени называется временем жизни.

**Зависимость времени жизни связующего** идентична зависимости вязкости от температуры (1.8) и описывается уравнением:

, (1.11)

где:

*t*ж – время жизни связующего (время гелеобразования), с;

*T* – температура, К;

*E*ао – энергия активации процесса отверждения, Дж / моль;

*B* – константа, определяемая природой жидкости, с;

*R* = 8,314 Дж/(К · моль) – универсальная газовая постоянная.

Соответственно, формула для вычисления времени жизни связующего с учётом температурной зависимости имеет вид аналогичный (1.10):

,

где *T*0 – номинальная температура (в данном случае *T*0 = 298,15 К = 25 **°**C), *t*ж0 – номинальное время жизни связующего (при температуре *T*0).

## Параметры, необходимые для работы алгоритма моделирования

### Сетка преформы

Сетка должна содержать набор координат узлов (клеточных автоматов), а также набор связей узлов друг с другом (для задания соседних клеточных автоматов).

Координаты задаются в метрах. В разработанном программном обеспечении предусмотрена функция перевода координат в метры из миллиметров при загрузке файла сетки.

При добавлении нескольких слоёв, каждый слой представляется отдельной сеткой, загружаемой из файла. Соединения между узлами различных слоёв создаются автоматически перед началом процесса моделирования.

### Параметры наполнителей преформы

Преформа может состоять из множества слоёв из различных материалов. Используются следующие параметры:

* **Толщина** – толщина ткани материала. Задаёт высоту полости в клетке.

[*d*] = [м]

* **Проницаемость** – параметр, характеризующий свойство вещества пропускать сквозь себя жидкости и газы при наличии перепада давления.

[*K*] = [м2]

* **Пористость** – доля объёма пор в общем объёме пористого тела. Является безразмерной величиной от 0 до 1. 0 соответствует материалу без пор; пористость 1 недостижима. Пористость равна отношению свободного объёма *Vc*, не заполненного элементарными структурными частицами, к общему объёму *V* тела:

*,*

[П] = [м3] / [м3] = []

где *VТ* — объём твёрдого скелета или матрицы.

### Параметры связующего

Связующее характеризуется номинальной вязкостью и температурным коэффициентом.

* **Номинальная вязкость** – динамическая вязкость жидкости, измеренная при температуре 25 °C (298,15 К).

[μ0] = [Па · с]

* **Энергия активации вязкого течения** – величина, используемая для определения вязкости жидкости при заданной температуре. Измеряется в Джоулях в моль (Дж/моль).

[*E*ав] = [Дж / моль]

* **Номинальное время жизни связующего** – величина, характеризующая максимально возможный срок применения смешанного связующего, измеренная при 25 °C (298,15 К). По истечении этого срока при данной температуре связующее станет желеобразным, и, в силу резкого роста вязкости, дальнейшая пропитка станет невозможной. Измеряется в секундах (с).

[*t*ж] = [с]

* **Энергия активации процесса отверждения** – величина, используемая для определения времени жизни связующего при заданной температуре. Измеряется в Джоулях в моль (Дж/моль).

[*E*ао] = [Дж / моль]

### Параметры подачи и съёма связующего

Источник связующего определяется положением на детали, диаметром трубки и давлением подачи.

* **Диаметр источника** – диаметр трубки, подключённой к преформе, по которой подаётся связующее.

[*Dinj*] = [м]

* **Давление подачи** – давление, с которым осуществляется впрыск связующего. Для процесса вакуумной инфузии соответствует атмосферному давлению. По умолчанию принимается *patm* = 105 Па.

[*patm*] = [Па]

Источник связующего определяется положением трубки откачки на детали, диаметром этой трубки и давлением вакуумирования.

* **Диаметр источника** – диаметр трубки, подключённой к преформе, по которой подаётся связующее.

[*Dvac*] = [м]

* **Давление вакуума** – давление, создаваемое компрессором в рабочей области детали. Величина больше нуля и меньше атмосферного давления.

[*pvac*] = [Па]

# Разработка учебно-демонстрационной программы

## Основные концепции, использованные при разработке учебно-демонстрационной программы

### Особенности реализации алгоритма моделирования на языке С++

Описанный выше метод моделирования был реализован на языке C++ с использованием возможностей стандарта C++11.

В разработанной программе каждый узел (клеточный автомат) хранится в отдельном объекте класса *VSimNode*. В узле хранится следующая информация:

* *x*, *y*, *z* – координаты узла, м;
* текущее значение давления, Па;
* значение давления, рассчитанное для следующего шага, Па;
* тип узла: обычный, точка подвода связующего, точка съёма связующего;
* список указателей на соседние узлы;
* указатель на объект, хранящий параметры материала (структура *VCloth*).

Поверхность трёхмерной модели разбивается на треугольники. Информация о треугольниках хранится в объектах класса *VSimTriangle*. Треугольник содержит указатели на узлы, являющиеся его вершинами. Информация о степени заполнения элемента поверхности, представленного треугольником, характеризуется цветом треугольника. Цвет треугольника характеризует среднюю степень заполнения узлов, находящихся в его вершинах. Шкала цветов представлена на рисунке Рисунок 4.



Рисунок 1 – Шкала цветовых обозначений степени пропитки.

За процесс симуляции отвечает объект класса *VSimulator*. Он содержит указатели на все активные узлы и треугольники, параметры симуляции (класс *VSimulationParameters*), информацию о протекании симуляции (структура *VSimulationInfo*).

На каждом шаге процесса симуляции, происходящего в объекте класса *VSimulator*, осуществляются следующие действия:

1. расчёт длительности последнего шага τ по формуле (1.7) , с;
2. измерение времени, ушедшего на моделирование последнего шага τм, с;
3. расчёт фактора реального времени: τ / τм;
4. определение текущей степени пропитки детали, %;
5. определение среднего давления связующего в детали, Па;
6. вычисление новых величин давления для каждого из узлов по формуле (1.5);
7. обновление значения давления в каждом из узлов;
8. определение, обновилось ли значение давления хотя бы в одном из узлов;
9. определение количества полностью заполненных узлов *n*, используемого при подсчёте времени по формуле (1.7);
10. обновление цветов треугольников;
11. оповещение объекта графической визуализации (*VGraphicsViewer*) о том, что был произведён новый шаг симуляции.

Если за шаг не обновилось значение давления ни для одного из узлов, либо все узлы заполнились, симуляция останавливается.

### Автоматическое соединение узлов различных слоёв

Одной из задач, возникшей при разработке программы имитационного моделирования, является соединение (добавление в соседи) близлежащих узлов различных слоёв. Необходимо это для обеспечения взаимного влияния процессов пропитки в различных слоях.

Для сохранения точности расчётов при использовании метода клеточных автоматов, необходимо соединять только те узлы, расстояние между которыми не сильно отличается от среднего расстояния между узлами внутри слоя. Было принято решение соединять узлы, расстояние между которыми находится в диапазоне ±10% относительно среднего расстояния между узлами одного из соединяемых слоёв.

Очевидным подходом к выполнению соединения узлов является их последовательный перебор. Необходимо перебрать все узлы первого слоя и для каждого из них вычислить расстояние до каждого из узлов второго слоя. Т.е. для каждого из узлов первого слоя перебираются все узлы второго слоя. Таким образом, сложность вычислений составляет *O*(*n* · *m*), где *n* и *m* – число узлов на первом и на втором слое соответственно. При большой степени детализации число узлов на каждом из слоёв может достигать десятков тысяч. В этом случае для соединения слоёв потребуется длительное время.

Для решения данной проблемы было решено создать структуру данных с константной сложностью поиска *O*(1) узлов по координатам. Такая структура данных была реализована в классе *VNodesVolume*.

В основе реализации данного класса лежит построение вокруг слоя описывающего параллелепипеда, который разбивается на кубические ячейки одинакового размера. В каждой ячейке содержится список указателей на узлы, координаты которых попадают в данную ячейку. Такой параллелепипед хранится в трёхмерном массиве.

В таком массиве при поиске узлов можно не перебирать все элементы, а просматривать только те, которые находятся в ячейках, удовлетворяющих заданным требованиям (рисунок 5).

При соединении слоёв рассматриваются узлы только из тех ячеек, которые попадают в пространство между двумя сферами радиусами и . Центр сфер – координата узла другого слоя, с которым необходимо выполнить соединение. Количество рассматриваемых ячеек не зависит от общего числа узлов в слое.

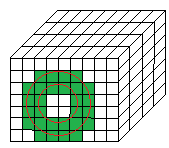


Рисунок 2 – Выбор списков узлов,   
потенциально удовлетворяющих критерию.

При использовании данного алгоритма сложность соединения слоёв составляет *O*(*n*). Также *O*(*n*) составляет сложность построения параллелепипеда, содержащего списки указателей на узлы.

Недостатком является увеличение объёма потребляемой оперативной памяти. Для обеспечения максимального быстродействия требуется подбирать такой размер стороны ячейки, чтобы в каждую ячейку попадало по одному узлу, и при этом число пустых ячеек было минимальным. Также при возрастании числа ячеек возрастает размер потребляемой памяти.

Пустой объект класса список на 64-х разрядном компьютере занимает 24 байта. Таким образом, например, для слоя с габаритами 2 м x 4 м x 0,04 м при создании ячеек со стороной 0,01 м потребуется выделить 7,32 МБ памяти.

Эмпирическим путём было получено, что наилучшее значение размера стороны ячейки составляет .

### Многопоточность

Процесс моделирования может занимать длительное время. Так как подавляющее большинство современных компьютеров оснащено многоядерными процессорами, если требуется, чтобы программа выигрывала от увеличения вычислительной мощности, то её необходимо проектировать как набор параллельных задач [3].

Параллелизм – это одновременное выполнение двух или более операций. Наиболее применяемый в наше время подход к реализации параллелизма – запуск нескольких потоков в одном процессе (многопоточность).

Многопоточность – свойство платформы, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени.

Сутью многопоточности является квазимногозадачность на уровне одного исполняемого процесса, то есть все потоки выполняются в адресном пространстве процесса. Кроме этого, все потоки процесса имеют не только общее адресное пространство, но и общие дескрипторы файлов. Любой выполняющийся процесс имеет как минимум один (главный) поток.

Существует два способа применить распараллеливание для повышения производительности [3]. Первый подход – *распараллеливание по задачам*. Он заключается в разбиении задачи на части и их параллельном запуске, в результате чего уменьшается общее время её выполнения. Трудности при реализации этой процедуры на практике возникают из-за наличия многочисленных зависимостей между разными частями.

Второй подход – *распараллеливание по данным*. В этом случае каждый поток выполняет одну и ту же операцию, но с разными данными. Для обработки одной порции данных требуется столько же времени, сколько и раньше, но за фиксированное время можно обработать больше данных. Очевидно, что и у этого подхода есть ограничения, и не во всех случаях он дает выигрыш, но достигаемое повышение производительности иногда открывает новые возможности.

В C++11 работа с потоками осуществляется по средствам класса *std::thread*. Функция, подаваемая в качестве аргумента конструктору *std::thread*, начинает свое исполнения сразу по окончании работы конструктора. Завершение потока происходит по завершении работы исполняемой функции. После того, как объект *std::thread* создан возможны три варианта развития событий:

1. Другой поток исполнения будет ожидать завершения исполнения созданного потока. Для этого в ожидающем потоке осуществляется вызов *thread.join()*. Этот вызов блокирует вызывающий поток.
2. Главный поток исполнения может завершится до того, как будет завершён созданный поток. Блокировки вызывающего потока происходить не будет. Для этого осуществляется вызов *thread.detach().*
3. В случае, если не было выполнено ни то, ни другое, при вызове деструктора объекта thread произойдёт аварийное завершение.

Использование join является более предпочтительным, так как detach представляет меньше контроля за исполнением.

В классе thread, также существует статическая функция std::thread::hardware\_concurrency, которая может вернуть количество потоков, которые могут выполняться параллельно на аппаратном уровне.

Когда появляется возможность параллельного исполнения кода, очень часто возникает необходимость конкурентного доступа к разделяемому ресурсу. Для получения предсказуемого результата от операции над разделяемым ресурсом, необходимо упорядочивать доступ к оному. Существует несколько программных инструментов, помогающих защитить разделяемые данные и сделать код потокобезопасным. Их называют примитивами синхронизации. Наиболее распространенными являются следующие примитивы: мьютексы, семафоры, условные переменные и спин-блокировки. Все они являются вариантами одной и той же концепции: они защищают кусок кода, давая только определенному потоку право получать доступ к данным и блокируя остальные.

Наиболее распространенным (и используемым) примитивом является мьютекс. Мьютекс расшифровывается как «Взаимное исключение». Это гарантия, что только один поток может выполнять код.

С++11 предоставляет 3 типа операций над базовыми мьютексами:

* *lock –* если мьютекс не принадлежит никакому потоку, тогда поток, вызвавший *lock*, становится его обладателем. Если же некий поток уже владеет мьютексом, то текущий поток (который пытается овладеть им) блокируется до тех пор, пока мьютекс не будет освобожден и у него не появится шанса овладеть им.
* *try\_lock -* если мьютекс не принадлежит никакому потоку, тогда поток, вызвавший *try\_lock*, становится его обладателем и метод возвращает *true.* В противном случае возвращает *false*. *try\_lock* не блокирует текущий поток.
* *unlock –* освобождает ранее захваченный мьютекс.

В C++11 представлено четыре вида мьютексов:

* *std::mutex* – базовый *mutex*, которым может владеть один поток в единицу времени. При попытке повторного овладения *мьютексом* потоком, уже владеющим им, произойдёт взаимная блокировка (*deadlock)* или будет брошено исключение с кодом ошики  *resource\_deadlock\_would\_occur*.
* *std::recursive\_mutex* – обладает теми же свойствами, что и *std::mutex*, но позволяет рекурсивное овладение мьютексом, т.е. многократный вызов метода *lock()* в потоке, который владеет мьютексом. При этом метод *unlock()* должен быть вызван не меньшее количество раз, чем был вызван *lock().* В противном случае произойдёт взаимная блокировка, т.к. этот поток никогда не освободит *мьютекс* и остальные потоки будут находиться в вечном ожидании.
* *std::timed\_mutex –*обладая свойствами *std::mutex, std::timed\_mutex,* также*,* обладает дополнительными методами позволяющими блокировку на время.
* std::recursive\_timed\_mutex – рекуррентная версия *std::timed\_mutex.*

Для облегчения работы с мьютексами в C++11 существуют средства для применения для них принципа «**Получение ресурса есть инициализация» (RAII).** Эти средства реализованы в std::lock\_guard. При создании объекта lock\_guard захватывается мьютекс, переданный ему в конструкторе. В деструкторе, же, происходит освобождение мьютекса.

Ещё одним блокирующим примитивом в C++11 является условная переменная – *std::condition\_variable*. Привычным шаблоном использования подобного примитива является ожидание одного потока, наступления события в другом потоке. Этот примитив можно рассматривать как некий сигнал, появления которого необходимо ожидать.

Основными методами *condition\_variable* являются:

* *wait –* ставит поток в ожидание сигнала. Ожидание не лимитировано временем. Может принимать в качестве аргумента предикат, от результата которого будет зависеть выход потока из ожидания. Т.е. если даже *wait* был завершен благодаря сигналу, происходит проверка предиката после чего поток снова становится в ожидание, если предикат ложен. Нужно это, в первую очередь, для того, чтобы избежать реагирования на *фальшивое* пробуждение.
* *wait\_for –* ожидание лимитировано согласно аргументу.
* *wait\_until* - ожидание лимитировано согласно аргументу.
* *notify\_one –*посылает сигнал одному из ожидающих потоков; т.е. разблокирует один поток. Какой поток будет разбужен – не известно. Гарантировано лишь то, что один из них будет.
* *notify\_all –*посылает сигнал всем ожидающим потокам; т.е. разблокирует все потоки ожидающие на данном объекте *condition\_variable.*

В разработанной программе для сообщения другим потокам информации о наступлении событий был создан класс *VNotify*, реализующий обёртку над *std::condition\_variable.*

Также в C++11 реализованы атомарные переменные *std::atomic*. Они характеризуются тем, что как только начинается процесс их чтения или записи, ничто не может его прервать и ничто не может произойти с этой переменной в середине.

В разработанной программе были применены оба подхода к распараллеливанию.

Распараллеливание по задачам реализовано за счёт одновременного расчёта данных и их визуализации на трёхмерной модели. После каждой итерации расчёта объекту, отвечающему за визуализацию, при помощи условной переменной отправляется сигнал о завершении итерации расчёта. Если объект визуализации в данный момент уже завершил отрисовку результатов предыдущего шага расчёта, он готов к отрисовке результатов нового. Если итерация расчётов занимает в несколько меньше времени, чем итерация отрисовки, визуализация будет происходить не для каждого шага. Данные о визуализируемых цветах (массив элементов *VSimTriangle*) защищаются мьютексом.

Распараллеливание по данным реализовано за счёт одновременно расчёта новых давлений и степеней заполнения для различных частей узлов. Для этого массив указателей на узлы разбивается на приблизительно равные части, расчёт которых осуществляется параллельно. Число параллельно рассчитываемых частей равно числу возможных выполняемых потоков на аппаратном уровне минус единица (но не менее одной части).

Было проведено сравнение результатов выполнения многопоточного кода и однопоточного. Для этого было измерено время моделирования пропитки трёх деталей (таблица Таблица 1). Использовались следующие параметры:

* давление подачи связующего: 105 Па;
* давление вакуума: 103 Па;
* вязкость связующего при 25°C: 0,1 Па·с;
* температура: 25°C;
* проницаемость 2·10-9м2;
* толщина (высота полости): 0,002 м;
* пористость: 0,5.

Таблица 1 – Измерение времени моделирования   
с использованием многопотончости и без него

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Имя файла | Число узлов | Число треугольников | Габариты, *x, y, z*, м | Диаметр источника, м | Время однопоточного моделирования, с | Время многопоточного моделирования, с |
| gmsh\_simple.msh | 5589 | 10832 | 2; 4; 0,04 | 0,1 | 492 | 25 |
| 1x0.5m.msh | 6764 | 13226 | 1; 0,5; 0 | 0,05 | 256 | 12 |
| paddel\_einfach.db | 526 | 949 | 0,22; 0,98; 0,05 | 0,02 | 8 | 2 |

По таблице Таблица 1 видно, что использование многопоточности позволяет добиться в десятки раз большей производительности, особенно при использовании слоёв с большим количеством узлов. Во многом столь большая разница в производительности обусловлена тем, что при однопоточном выполнении визуализация результата осуществлялась на каждом шаге, а при многопоточном часть шагов визуализации пропускалась.

### Визуализация при помощи трёхмерной графики

Для визуализации результатов моделирования требуется отображать на трёхмерной сцене слои, состоящие из множества окрашенных треугольников. Для этой цели применяется графический движок. Для взаимодействия с графическим движком и управления процессом отображения элементов на экране был создан класс *VGraphicsViewer*.

Для управления отображением каждого слоя был создан класс *VGraphicsLayer*. Он включает в себя указатели на элементы, управляющие отображением каждым треугольником и узлом (отображается как небольшой кубик).

За управление отображением треугольников отвечает класс *VGraphicsTriangle*. Он содержит указатель на объект класса *VSimTriangle*, имеет те же координаты вершин и при необходимости обновляет свой отображаемый цвет на цвет, записанный в объекте класса *VSimTriangle*.

За управление отображением узлов отвечает класс *VGraphicsNode*. Он содержит указатель на объект класса *VSimNode* и отображает небольшой куб в позиции, соответствующей данному узлу.

В графических движках отображаемые на экране элементы описываются графом сцены.

Граф сцены представляет структуру, которая содержит логическое и зачастую пространственное представление графической сцены. Граф сцены представляет собой набор узлов такой структуры, как граф или дерево. Узел дерева (в предельной структуре дерева графа сцены) может иметь множество потомков, но зачастую только одного предка, причём действие предка распространяется на все его дочерние узлы; эффект действия, выполненного над группой, автоматически распространяется на все её элементы. Во многих программах ассоциирование матрицы преобразования на уровне любой группы и умножение таких матриц представляет собой эффективный и естественный способ обработки таких действий. Общей особенностью, к примеру, является способность группировать связанные формы/объекты в составной объект, который можно перемещать, трансформировать, выбирать и т.д. так же просто, как и одиночный объект.

Узлы в графе считываются сверху вниз и слева направо. Каждый узел, в зависимости от своего типа определяет, каким образом будет нарисована сцена. Узлы графа наследуют своё состояние от посещённых ранее узлов.

Граф сцены, используемой для визуализации процесса пропитки, представлен на рисунке Рисунок 6.

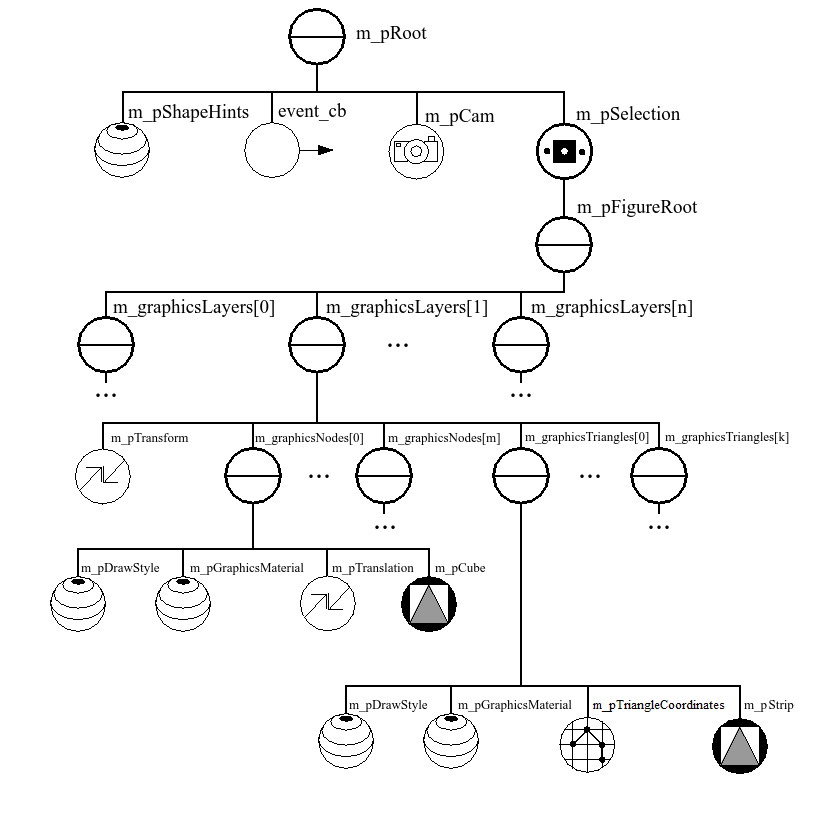


Рисунок 3 – Граф сцены для визуализации процесса пропитки.

Используемые на рисунке 6 обозначения типов узлов:

 – разделитель. Сохраняет состояние пред посещением своих дочерних элементов, а после их обхода вновь восстанавливает его.

 – отображение. Определяет внешний вид элементов.

 – функция обратного вызова. Задаёт функцию, которая будет вызвана в результате реакции сцены на событие.

 – камера. Задаёт способ отображения (перспективная или ортографическая проекции) и позицию обзора сцены.

 – выбор. Позволяет пользователю при помощи мыши выбирать дочерние узлы.

 – трансформация. Задаёт пространственное положение узла при помощи матрицы трансформации.

 – фигура. Определяет геометрию отображаемого объекта.

 – топология. Определяет координаты в пространстве.

### Формат хранения детали в файле

В программе присутствует возможность сохранения обрабатываемой детали в файл с возможностью последующей загрузки и продолжения моделирования. Для хранения данных применяется язык разметки XML.

Стандарт XML был впервые опубликован Консорциумом Всемирной паутины в 1998 году. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. *Расширение XML* – это конкретная грамматика, созданная на базе XML и представленная словарём тегов и их атрибутов, а также набором правил, определяющих какие атрибуты и элементы могут входить в состав других элементов. Преимуществами XML являются сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов.

Для реализации сохранения и загрузки детали из файла было разработано три класса. Используемые теги хранятся в классе *VModelImportExport*. За сохранение в файл отвечает класс *VModelExport*. За загрузку из файла отвечает класс *VModelImport*.

В сохраняемый файл записываются:

* информация о состоянии симуляции: время процесса, время моделирования, фактор реального времени, процент выполнения задачи, среднее давление связующего, номер итерации;
* параметры моделирования: диаметр трубки подачи связующего, диаметр трубки съёма связующего, температура, давление подачи, давление вакуума, коэффициенты *q*, *r*, *s*, средняя дистанция между клетками, средняя проницаемость, число узлов,
  + характеристики связующего: энергия активации вязкого течения, номинальная вязкость, энергия активации процесса отверждения, номинальное время жизни, имя материала.
* параметры стандартного рабочего поля стенда:
  + размеры;
  + параметры портов подачи и вакуума: диаметр порта подачи, диаметр порта вакуума,
    - положение порта подачи;
    - положение порта вакуума;
* положение используемых параметров подачи и вакуума: использовать ли параметры рабочего поля,
  + параметры заданных портов подачи и вакуума: диаметр порта подачи, диаметр порта вакуума,
    - положение порта подачи;
    - положение порта вакуума;
* находится симуляция на паузе или была остановлена;
* ограничение на время моделирования, необходимо ли его ограничивать;
* список слоёв:
  + характеристики слоя: используется ли при расчётах, отображается ли на экране, минимальный идентификатор узла в слое, максимальный идентификатор узла в слое, минимальный идентификатор треугольника в слое, максимальный идентификатор треугольника в слое, число узлов, число треугольников,
    - характеристики материала слоя: высота полости, проницаемость, пористость, цвет (до пропитки), имя,
    - список узлов:
      * характеристики узла: идентификатор, давление, роль (обычный, связан с устройством подачи, связан с устройством снятия связующего), *x*, *y*, *z* координаты,
    - список треугольников:
      * характеристики треугольника: идентификатор, цвет, три идентификатора узлов, находящихся в вершинах треугольника;
* список соединений узлов:
  + характеристики соединения: идентификатор узла, список идентификаторов соседних узлов.

## Внутренняя структура программного обеспечения

Весь алгоритм моделирования реализован в модуле *v\_sim*. Диаграмма классов данного модуля представлена на рисунке Рисунок 7.

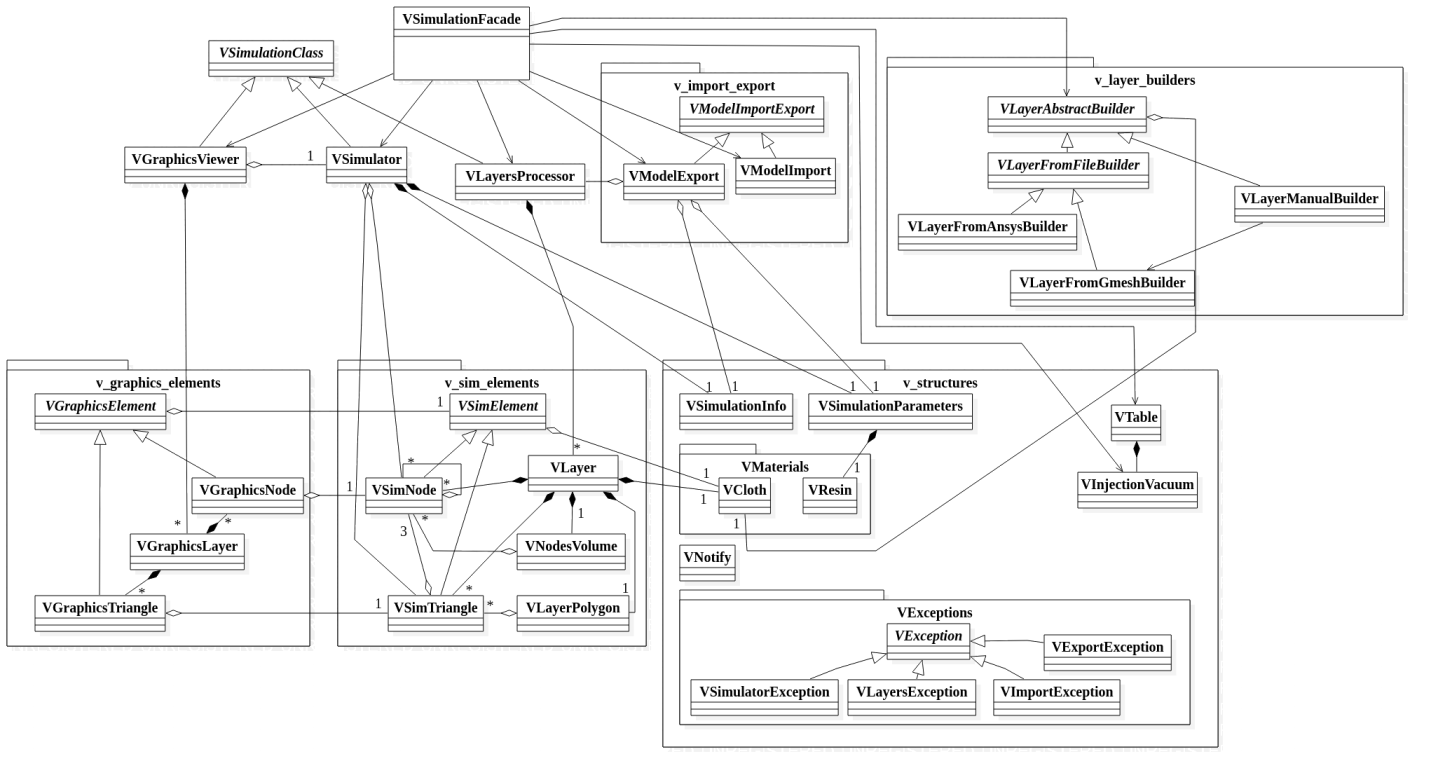


Рисунок 4 – Диаграмма классов программы моделирования.

### Основные классы

Основные классы, реализующие процесс моделирования:

***VSimulationFacade*** – класс, обеспечивающий внешний интерфейс взаимодействия со всей системой моделирования и организующий её работу.

***VSimulationClass*** – абстрактный класс. Является базовым для классов *VSimulator*, *VGraphicsViewer*, *VLayersProcessor*, отвечающих за моделирование. Реализует функции для работы с мьютексами для защиты наборов узлов и треугольников, общие для этих трёх классов.

***VSimulator*** – основной класс. Реализует алгоритм моделирования, описанный в пункте 4.1.

***VGraphicsViewer*** – класс, реализующий визуализацию процесса пропитки. Строит трёхмерную сцену, описанную в пункте 2.1.4.

***VLayersProcessor*** – класс для работы со слоями. Содержит в себе набор всех слоёв. Предоставляет интерфейсы для добавления, удаления, включения, отключения, отображения, скрытия, соединения, сортировки слоёв; для определения габаритов детали; для указания на слоях позиций подвода и съёма связующего.

### Классы элементов пропитываемой детали, используемые в расчётах

Классы, характеризующие структурные единицы пропитываемой детали, используемые в процессе расчёта, хранятся в модуле *v\_sim\_elements*. Модуль содержит следующие классы:

***VSimElement*** – абстрактный базовый класс, определяющий основные методы взаимодействия с элементарными структурными единицами, на которые разбивается пропитываемая деталь: узлами (класс *VSimNode*) и треугольниками (класс *VSimTriangle*).

***VSimNode*** – класс, созданный для хранения информации, характеризующей узел моделирования (клеточный автомат) на текущей итерации. Подробнее об информации, хранящейся в экземплярах данного класса, написано в пункте 4.1.

***VSimTriangle*** – класс, созданный для хранения информации о треугольниках, на которые разбивается трёхмерная деталь (см. пункт 2.1.4).

***VLayer*** – класс, объекты которого позволяют манипулировать слоями в целом. Содержит в себе указатели на все узлы и все треугольники слоя.

***VNodesVolume*** – структура данных, хранящая указатели на узлы в ячейках трёхмерного массива (представляющего параллелограмм). Содержит методы для быстрого поиска узлов в заданном геометрическом месте точек.

***VLayerPolygon*** – класс, отвечающий за формирование и хранение контура проекции слоя на горизонтальную плоскость. Также содержит методы, позволяющие определить, находится заданная точка внутри или вне контура.

### Классы визуализации элементов пропитываемой детали

Классы, отвечающие за графическое отображение структурных единиц пропитываемой детали, хранятся в модуле *v\_graphics\_elements*. Данные классы реализуют подграфы графа сцены (пункт 2.1.4). Модуль содержит следующие классы:

***VGraphicsElement*** – абстрактный базовый класс, определяющий основные свойства отображения элементов детали: узлов (класс *VGraphicsNode*) и треугольников (класс *VGraphicsTriangle*).

***VGraphicsNode*** – класс, определяющий параметры отображения узлов. Узлы отображаются в виде небольших кубов. Объекты данного класса содержат указатели на соответствующие им объекты класса *VSimNode*.

***VGraphicsTriangle*** – класс, определяющий параметры отображения треугольников. Объекты данного класса содержат указатель на соответствующие им объекты класса *VSimTriangle*.

***VGraphicsLayer*** – класс, определяющий параметры отображения слоя в целом. Объекты данного класса содержат указатели на весе объекты классов *VGraphicsNode* и *VGraphicsTriangle*, относящихся к данному слою.

### Классы создания сеток слоёв

Классы, отвечающие за создание сеток слоёв, хранятся в модуле *v\_layer\_builders*. На данный момент реализовано только создания сеток слоёв за счёт их импорта из файлов. Классы модуля:

***VLayerAbstractBuilder*** – абстрактный базовый класс, определяющий интерфейс класса, создающего слой.

***VLayerFromFileBuilder*** – класс, содержащий основные методы, необходимые для импорта сетки слоя из файла, вне зависимости от формата файла.

***VLayerFromGmeshBuilder*** – класс, содержащий методы, необходимые для импорта сетки слоя из файла формата *Gmsh*.

***VLayerFromAnsysBuilder*** – класс, содержащий методы, необходимые для импорта сетки слоя из файла формата *Ansys*.

***VLayerManualBuilder*** – класс, позволяющий создать сетку слоя по заданному замкнутому контуру.

### Классы сохранения и загрузки созданной детали

Классы, отвечающие за сохранение пропитываемых деталей в файл и их последующую загрузку, хранятся в модуле *v\_import\_export*. Классы модуля:

***VModelImportExport*** – абстрактный базовый класс, описывающий XML-теги, используемые при сохранении и загрузке детали.

***VModelExport*** – класс, содержащий методы, необходимые для сохранения модели в файл.

***VModelImport*** – класс, содержащий методы, необходимые для загрузки сохранённой модели из файла.

### Вспомогательные структуры и классы

Вспомогательные классы хранятся в модуле *v\_structures*. Классы (структуры) модуля:

***VSimulationInfo*** – структура, служащая для хранения информации о текущем состоянии процесса моделирования. Информация включает: прошедшее время пропитки, время моделирования, фактор реального времени, процент выполнения задачи, среднее давление, номер итерации.

***VSimulationParameters*** – класс, служащий для хранения текущих параметров, необходимых для моделирования. Такими параметрами являются: диаметр трубки подачи связующего, диаметр трубки съёма связующего, температура, давление подачи, давление вакуума, коэффициенты *q*, *r*, *s*, средняя дистанция между клетками, средняя проницаемость, число узлов, характеристики связующего (экземпляр структуры *VResin*).

***VCloth*** – структура, служащая для хранения информации о параметрах материала наполнителя. Информация включает: проницаемость, высоту полости, пористость, цвет (которым обозначается слой до начала пропитки), имя материала.

***VResin*** – структура, служащая для хранения информации о параметрах материала связующего. Информация включает: номинальную вязкость (вязкость при температуре 25 °C), энергия активации вязкого течения, номинальное время жизни (при температуре 25 °C), энергия активации процесса отверждения, имя материала.

***VInjectionVacuum*** – структура, служащая для хранения информации о положениях портов подачи связующего и вакуума, их диаметров.

***VTable*** – класс, служащий для хранения информации о параметрах рабочего поля стенда. Параметрами являются размеры, положения портов подачи связующего и вакуума, их диаметры.

***VNotify*** – класс, созданный для оповещения других потоков о произошедших событиях.

***VException*** – базовый класс исключения, вырабатываемого модулем симуляции.

***VSimulatorException*** – класс исключения, возникающего при некорректном изменении модели в то время, как запущен процесс симуляции.

***VLayersException*** – класс исключения, возникающего при некорректном изменении конфигурации слоёв детали.

***VImportException*** – класс исключения, возникающего в результате ошибки импорта информации из файла.

***VException*** – класс исключения, возникающего в результате ошибки сохранения информации в файл.

# Разработка пользовательского интерфейса учебно-демонстрационной программы

Для реализации взаимодействия пользователя с программой было решено применить оконный интерфейс. Всего в программе имеется пять видов окон:

* **Основное окно** содержит элементы настройки параметров пропитки, управления процессом моделирования и, непосредственно, элементы визуализации процесса моделирования.
* **Окно «Создание слоя»** содержит элементы, необходимые для добавления нового слоя.
* **Окно «Задание границ слоя»** позволяет пользователю задавать контуры новых слоёв.
* **Окно «База данных тканей»** позволяет выбирать для слоя материал из базы данных, а также производить редактирование параметров базы данных: менять параметры материалов, добавлять и удалять материалы.
* **Окно «База данных связующих»** позволяет выбирать из базы данных материал связующего, которым будет пропитываться деталь, а также производить редактирование параметров базы данных.

Разработанная программа доступна для операционных систем Windows, GNU/Linux и macOS.

## Основное окно программы

Основное окно (Рисунок 5) открывается при запуске программы. Оно содержит элементы настройки параметров пропитки и визуализации (вкладки «Параметры моделирования», «Дополнительно»), управления процессом моделирования (панель инструментов) и, непосредственно, элементы визуализации процесса моделирования.

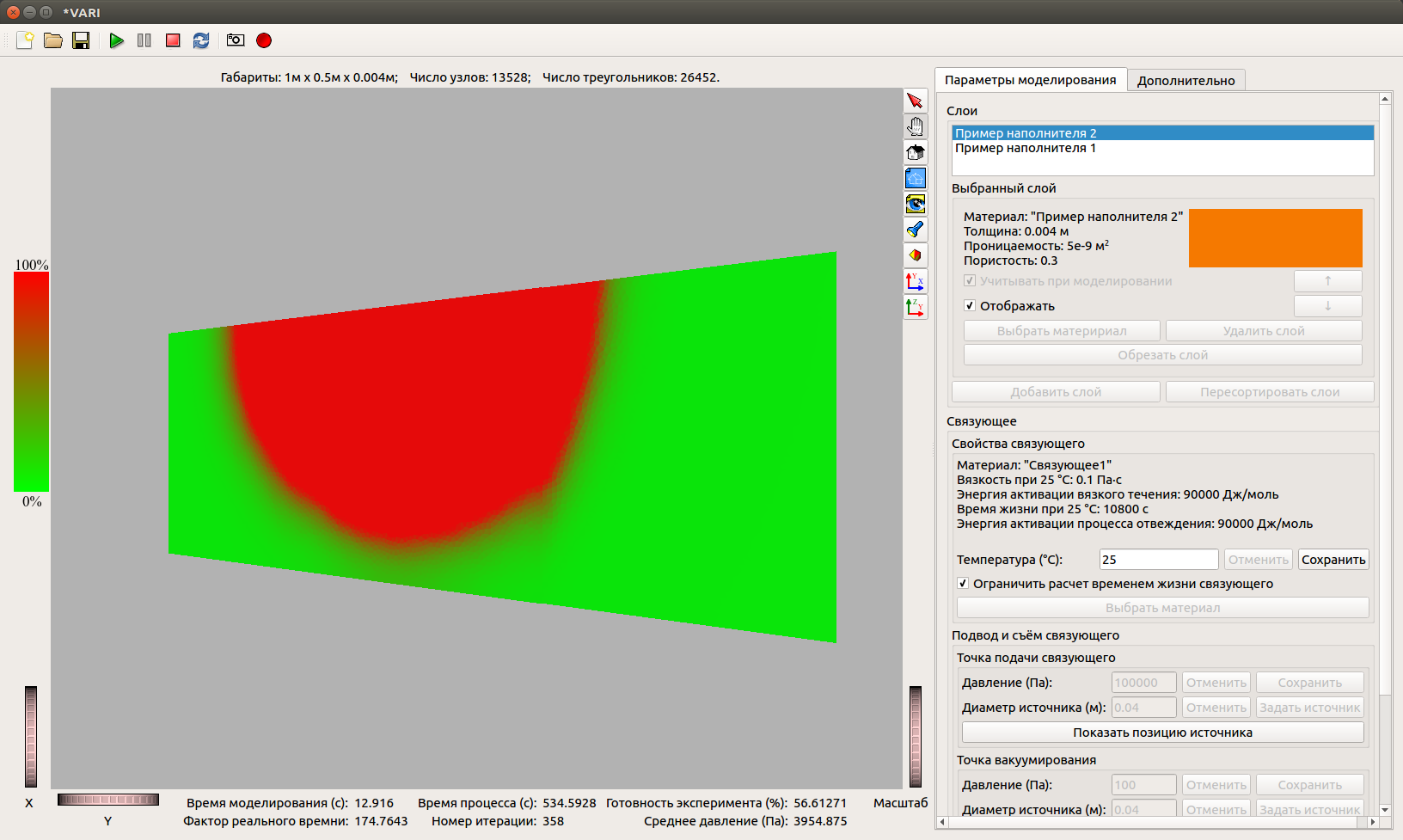


Рисунок 5 – Основное окно программы моделирования.

### Панель инструментов

У верхнего края основного окна по умолчанию располагается панель инструментов (Рисунок 6). Панель инструментов может быть перемещена пользователем и оставлена в любом месте экрана либо прикреплена к любому краю окна.



Рисунок 6 – Панель инструментов

Панель инструментов содержит три группы действий:

* создание, загрузка, сохранение модели;
* управление процессом моделирования: запуск, пауза, остановка, остановка со сбросом раскраски;
* включение режимов записи серий изображений и видео.

Запись серий изображений осуществляется путём сохранения в новую папку серий снимков части окна, отвечающей за визуализацию процесса пропитки. Поверх создаваемых кадров наносится информация о текущем состоянии процесса пропитки.

Запись видео осуществляется также как запись серии кадров. Только после её окончания все полученные снимки объединяются в видеофайл. Видеофайл кодируется кодеком MJPEG и сохраняется в контейнер AVI. MJPEG (Motion JPEG) представляет собой покадровый метод видеосжатия, основной особенностью которого является сжатие каждого отдельного кадра видеопотока с помощью алгоритма сжатия изображений JPEG. При сжатии методом MJPEG межкадровая разница не учитывается. Основными преимуществами данного кодека является высокая распространённость, а также простота реализации, что делает MJPEG подходящим для реализации в устройствах с ограниченными вычислительными ресурсами.

### Вкладка «Параметры моделирования»

В правой части основного окна располагаются две вкладки, содержащие элементы, позволяющие пользователю сформировать деталь, задать параметры моделирования и настроить программу. Элементы, отвечающие за формирование детали и задание параметров процесса моделирования, располагаются во вкладке «Параметры моделирования» (Рисунок 7).

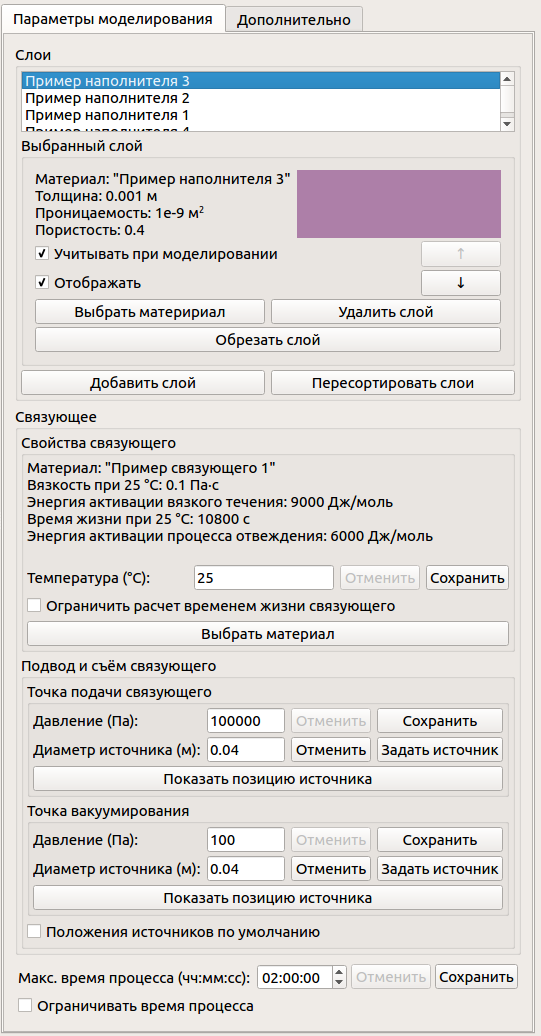


Рисунок 7 – Вкладка «Параметры моделирования»

В верхней части вкладки «Параметры моделирования» находится раздел «Слои» (Рисунок 8), элементы которого отвечают за формирование детали.

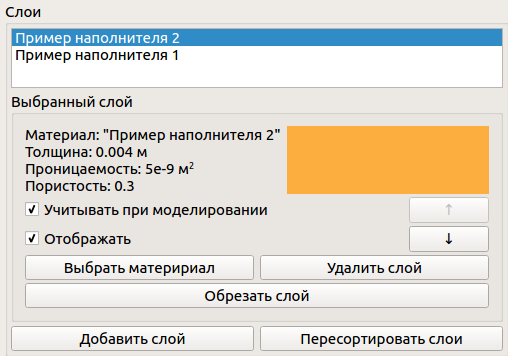


Рисунок 8 – Раздел «Слои» с выбранным слоем

В данном разделе присутствуют следующие возможности:

* добавление слоёв;
* отображение списка добавленных слоёв;
* отображение характеристик материала слоя;
* изменение материала слоя;
* отображение цвета слоя по умолчанию;
* смена цвета слоя по умолчанию;
* включение и исключение слоя из расчётной модели (учёт при моделировании);
* включение и отключение отображения слоя при визуализации пропитки;
* изменение позиций слоя внутри формируемой детали;
* включение режима обрезки слоя;
* удаление слоя;
* вызов принудительной пересортировки слоёв, в случай если какие-либо слои был вручную перемещены по вертикальной оси.

В верхней части раздела находится список всех добавленных слоёв. Слои в списке располагаются в том же порядке, что и в формируемой детали (верхние слои сверху, нижние – снизу). Каждый слой обозначается названием материала, из которого он сформирован. Слои, включённые в расчётную модель и отображаемые при визуализации процесса пропитки, занесены в список чёрным цветом. Слои, включённые в расчётную модель, но не отображаемые при визуализации, занесены серым цветом. Не включённые в расчётную модель слои занесены в список красным цветом.

При выборе пользователем конкретного слоя, у пользователя появляется возможность произвести с ним перечисленные выше действия при помощи элементов управления, расположенных в подразделе «Выбранный слой».

Под разделом «Слои» располагается раздел «Связующее» (Рисунок 9). Элементы данного раздела позволяют задавать параметры, характеризующие свойства связующего и способ его подачи.

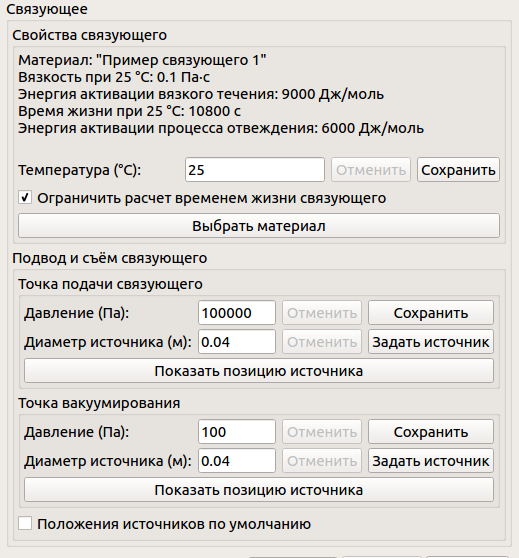


Рисунок 9 – Раздел «Связующее»

В разделе «Связующее» присутствуют следующие возможности:

* выбор материала связующего;
* отображение характеристик материала связующего;
* задание температуры связующего;
* включение/выключение режима, когда по окончании времени жизни связующего происходит автоматическая остановка моделирования;
* задание давления подачи связующего;
* задание диаметра и положения источника подачи связующего на сформированной детали;
* отображение синим цветом положения источника подачи на детали;
* задание давления вакуумирования;
* задания диаметра и положения порта вакуума на сформированной детали;
* отображение голубым цветом положения порта вакуума на детали;
* переключение режима, при котором положение и диаметр портов подачи и вакуума задаются автоматически, в соответствии с характеристиками рабочего поля стенда.

Для ввода параметров было решено использовать конструкцию следующего вида:

*Название параметра (единица измерения) – Поле ввода – Кнопка «Отменить» – Кнопка «Сохранить».*

Когда пользователь изменяет величину параметра в поле ввода, кнопка «Отменить» справа от поля ввода становится активной. При нажатии на неё в поле ввода будет возвращено значение, установленное в данный момент в модели. При нажатии кнопки «Сохранить» в модель будет записано значение параметра, заданное в поле ввода. После нажатия кнопки «Отменить», также как и кнопки «Сохранить», кнопка «Отменить» становится неактивной. Неактивная кнопка «Отменить» свидетельствует о том, что в поле ввода в данный момент записано то же значение параметра, что и установлено в модели.

Ниже раздела «Связующее» располагаются элементы управления ограничением времени процесса (Рисунок 10).



Рисунок 10 – Элементы управления ограничением времени процесса

Если установлена галочка-переключатель «Ограничивать время процесса», то когда время моделируемого процесса превысит заданное ниже в поле «Макс. время процесса» значение, симуляция автоматически встанет на паузу.

### Вкладка «Дополнительно»

На вкладке «Дополнительно» (Рисунок 11), в первую очередь, располагаются элементы, ответственные за настройку программы.

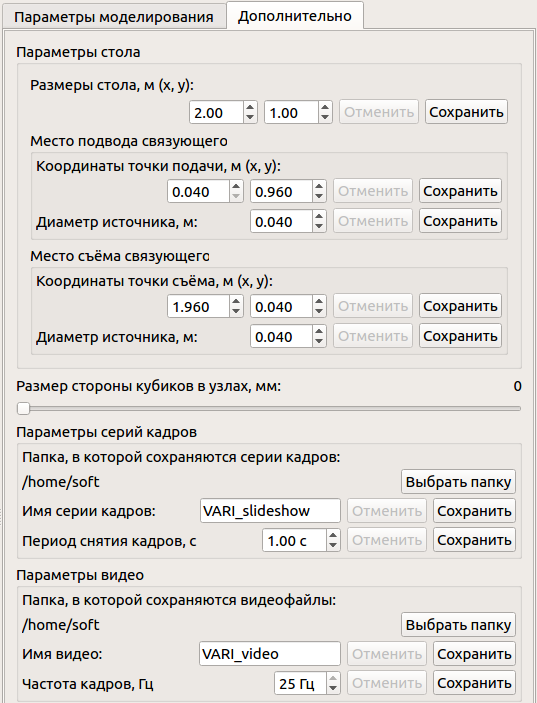


Рисунок 11 – Вкладка «Дополнительно»

В верхней части вкладки «Дополнительно» в разделе «Параметры стола» задаются параметры рабочего поля стенда. Здесь задаются размеры стола, координаты портов подачи и съёма связующего, их диаметры. Все размеры задаются в метрах. Началом координат является левый нижний угол стола.

Ниже отображается ползунок «Размер стороны кубиков в узлах, мм». При помощи данного ползунка пользователь может задавать размер стороны кубиков, которые отображаются в узлах сетки детали. Кубики в узлах окрашены в выбранный для слоя стандартный цвет, и их цвет не зависит от степени пропитки области, в которой находится узел. Выставление большого размера кубиков может понадобиться пользователю для возможности определения того, какой именно слой он наблюдает во время пропитки. При выставлении пользователем нулевого размера стороны кубики в узлах отображаться перестают.

Далее идёт раздел «Параметры серий кадров». В нём указывается папка, в которую будут сохраняться серии кадров, какие имена будут получать серии кадров, период снятия кадров.

Аналогичным образом задаётся параметры видео в разделе ниже.

### Визуализация процесса пропитки

Визуализация трёхмерной модели пропитываемой преформы (Рисунок 12) находится в левой части основного окна.

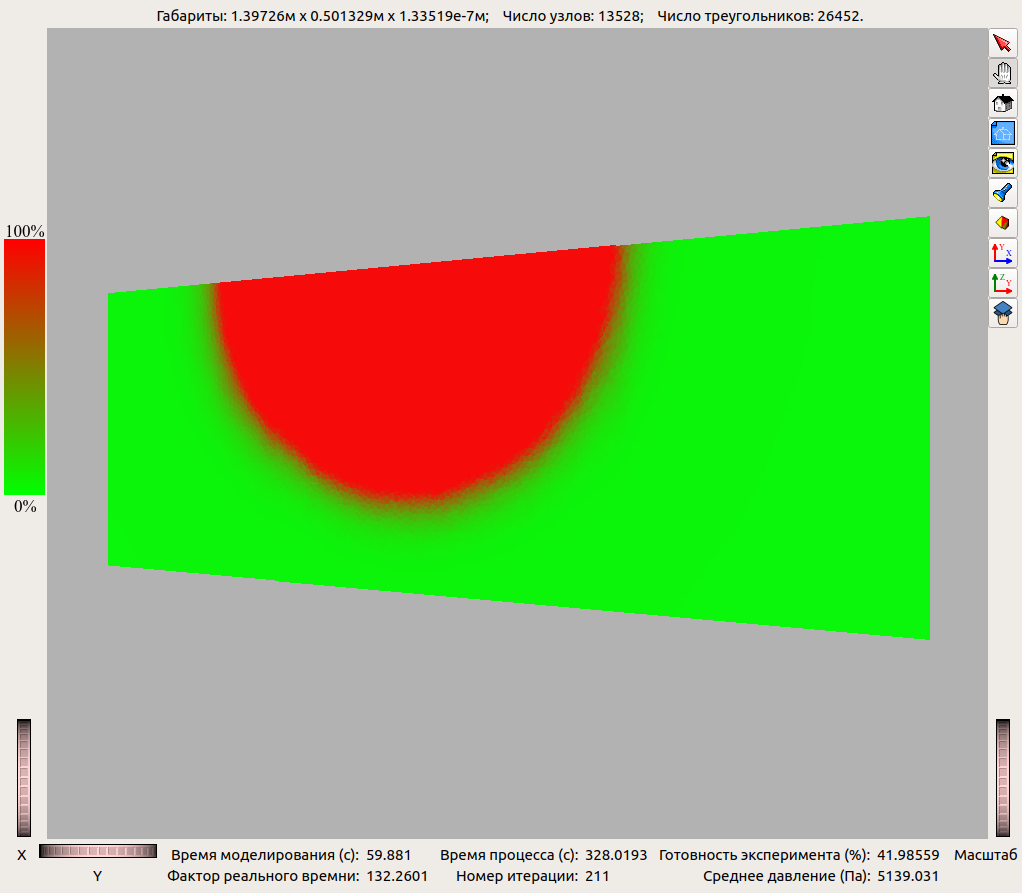


Рисунок 12 – Элементы визуализации модели преформы

Визуальное представление модели имеет ряд элементов управления.

Кнопки:

 - режим взаимодействия с моделью;

 - режим перемещения и вращения вида на модель;

 - переход к заранее сохранённой точке обзора;

 - сохранение текущей точки обзора;

 - полностью уместить модель на экране;

 - режим приближения;

 - переключение типа проекции ( – перспективная,  – ортографическая);

 - вид сверху;

 - вид справа;

 - режим перемещения слоёв;

 - режим задания контура обрезки.

Вращательные элементы (Рисунок 13):

* вращение модели вокруг горизонтальной оси;
* вращение модели вокруг вертикальной оси;
* приближение и отдаление модели.

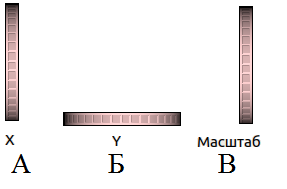


Рисунок 13 –Вращение модели вокруг горизонтальной оси (А), вокруг вертикальной оси (Б), приближение и отдаление (В).

При включённом режиме перемещения и вращения вида на модель () пользователь может вращать, перемещать, приближать и отдалять вид на модель помощи мыши.

Вверху отображается информация о габаритах трёхмерной модели преформы, число узлов, число треугольников, составляющих модель.

В процессе моделирования ячейки детали раскрашиваются в соответствии с их уровнем заполнения связующим. Зелёному цвету соответствует отсутствие заполнения, а красному – полное заполнение. Шкала цветовых обозначений степени пропитки располагается слева от трёхмерной модели преформы.

Под отображением модели выводится информация о текущем состоянии процесса моделирования:

* расчётное время моделирования, в секундах;
* время моделирования, в секундах;
* отношение расчётного времени за итерацию к измеренному времени итерации – фактор реального времени;
* номер итерации;
* средняя степень заполнения всех узлов в процентах – готовность эксперимента;
* среднее давление во всех узлах в паскалях.

При выборе пользователем слоя в списке и последующем нажатии кнопки **«Обрезать слой»** включается режим обрезки слоя. В результате, все слои, кроме выбранного, перестают отображаться и на панели инструментов визуализации модели появляется кнопка переключения выделения ().

Когда кнопка переключения выделения  утоплена, на экране можно задать контур, по которому будет выполнена обрезка. Для этого на экране отображения модели щелчками мыши задаются точки контура. Задаваемый контур отображается пунктирной линией (Рисунок 15).

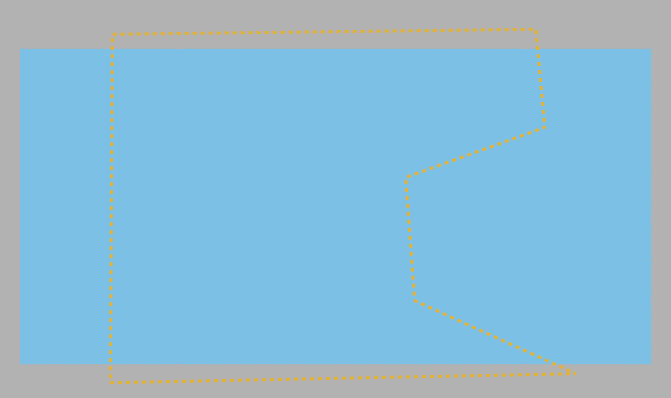


Рисунок 14 – Процесс задания контура для обрезки

Для окончания выбора контура обрезки необходимо осуществить **двойной щелчок** мышью в последней точке контура. Далее программа начнёт выбирать все элементы, попавшие в данный контур. После окончания данного процесса все элементы, попавшие внутрь контура, будут подсвечены фиолетовым цветом, и появится диалог подтверждения обрезки.

В случае нажатия пользователем в диалоге подтверждения кнопки «Да», от слоя останется только та часть, попавшая внутрь контура. В случае нажатия кнопки «Нет», слой обрезан не будет, но выделенная часть будет иметь фиолетовую окраску. Впоследствии у него будет возможность вновь провести контур вдоль фиолетовой части и выполнить обрезку.

Для включения режима перемещения слоёв служит кнопка  в панели инструментов визуализации модели. Когда включён данный режим и у визуализации включён режим выбора (), пользователь имеет возможность кликнуть мышью по изображению любого слоя. В результате вокруг выбранного слоя появится сфера управления перемещением (Рисунок 16).

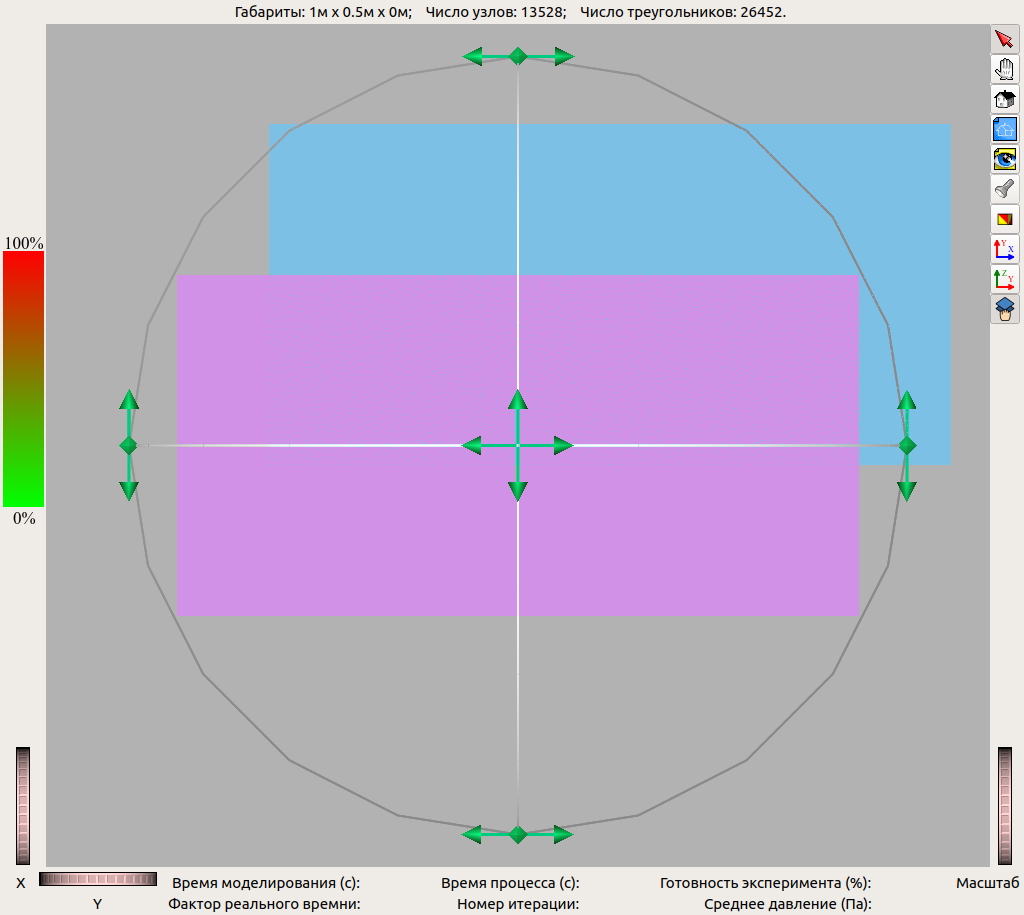


Рисунок 15 – Сфера управления перемещением слоя

Линейное перемещение слоя пользователь может выполнить, зажав левой кнопкой мыши перекрести стрелок () на сфере и с зажатой кнопкой двигать мышь в нужную сторону. Для вращательного перемещения слоя пользователю необходимо зажать левой кнопкой мыши любую точку сферы сфере и с зажатой кнопкой двигать мышь в нужную сторону. В результате слой будет поворачиваться относительно центра сферы (Рисунок 17).

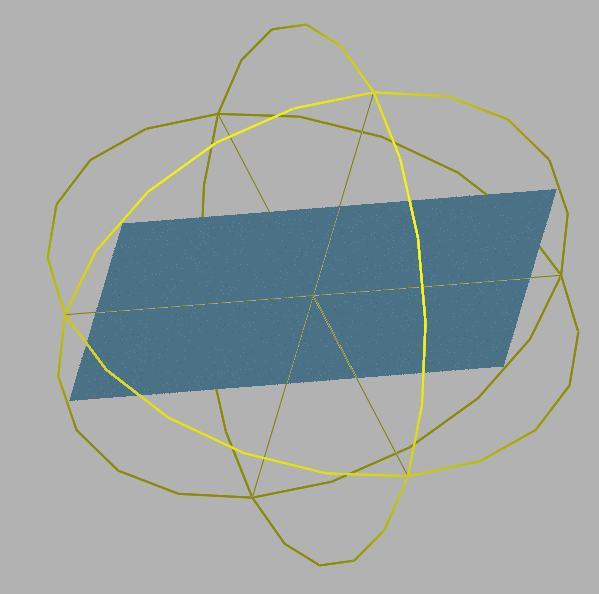


Рисунок 16 – Сфера управления перемещением во время процесса вращательного смещения слоя

## Окно «Создание слоя»

При добавлении пользователем нового слоя открывается окно «Создание слоя» (Рисунок 17).

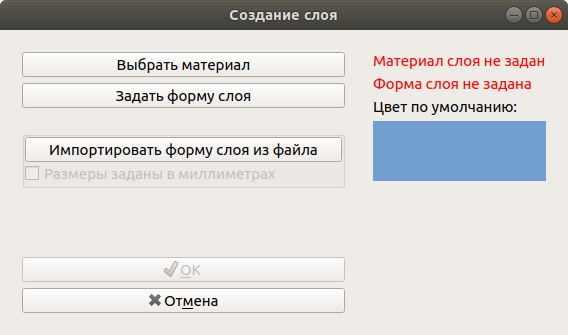


Рисунок 17 – Исходный вид окна создания слоя

Окно создания слоя предоставляет пользователю возможности выбора материала нового слоя и задания его формы. Форма слоя может быть задана пользователем вручную (2D – плоский слой или 1D – спиральная трубка) либо импортирована им из файла. Импортируемый слой из файла, теоретически, может иметь и трёхмерную форму.

Для выбора материала открывается окно интерфейса взаимодействия с базой данных наполнителей. После успешного выбора материала красная надпись «Материал слоя не задан» заменяется чёрным описанием характеристик выбранного для слоя материала.

При задании пользователем контура слоя вручную красная надпись «Форма слоя не задана» заменяется чёрной надписью «Форма слоя задана пользователем».

Для загрузки формы слоя из файла пользователю необходимо нажать на кнопку «Импортировать форму слоя из файла». В результате откроется диалог выбора файла, в котором ему необходимо выбрать файл формата трёхмерной модели Gmsh (версия формата не выше 2.2). После успешного выбора файла надпись «Форма слоя не задана» заменится путём к выбранному файлу. При импорте модели, размеры которой заданы в миллиметрах, необходимо отметить галочку-переключатель «Размеры заданы в миллиметрах».

Для выбора цвета слоя по умолчанию пользователю необходимо нажать на цветной прямоугольник под надписью «Цвет по умолчанию». В открывшемся диалоге пользователю нужно задать желаемый цвет. Цвет прямоугольника соответствует выбранному цвету слоя.

## Окно «Задание границ слоя»

Окно «Задание границ слоя» (Рисунок 18) содержит элементы управления, позволяющие пользователю задать контур слоя. Поддерживаются режимы 2D – плоский слой или 1D – спиральная трубка.

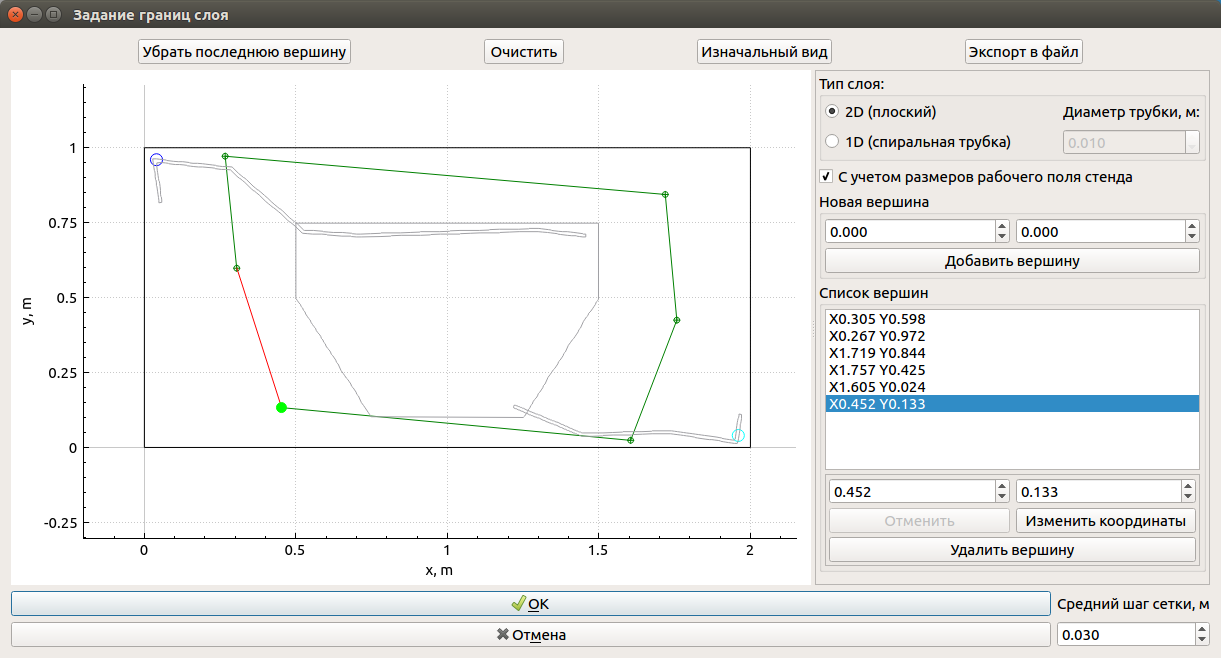


Рисунок 18 – Окно «Задание границ слоя»

В левой части окна находится холст. На нём чёрным цветом изображён прямоугольник, обозначающий границы стола. Синим цветом изображена окружность, обозначающая стандартное положение на столе трубки подачи связующего. Голубым цветом – трубки откачки воздуха. Серым цветом на холсте изображены контуры уже добавленных ранее слоёв. Зелёным цветом изображается контур, создаваемого пользователем слоя. Красным цветом – линия, соединяющая начальную и конечную вершину контура (только для 2D слоёв). Светло-зелёный круг наносится на выбранную в данный момент вершину.

На холсте пользователь может при помощи мыши задавать положения вершин, образующих контур. Пользователь при помощи однократных нажатий левой кнопкой мыши может создавать новые вершины. Также, нажатием на уже добавленную вершину, он может её выбрать. При удержании левой кнопки мыши пользователь может перемещать вершину по холсту. При этом программа не даёт пользователю координаты вершин, которые приведут к появлению в контуре самопересечений. Также, если отмечена галочка-переключатель «С учётом размеров рабочего поля стенда», у пользователя не будет возможности задать координаты вершины в области, находящейся за пределами рабочей зоны.

У пользователя имеется возможность масштабировать холст путём вращения колёсика мыши. Также холст можно перемещать, зажав на нём правую кнопку или нажав на колёсико мыши. Координаты холста в метрах отображаются на осях снизу и слева.

С правой стороны располагается панель с управляющими элементами. С её помощью реализуются следующие возможности:

* выбор режима задания слоя;
* включение/отключение учёта размеров рабочего поля стенда;
* добавление новых вершин по введённым координатам;
* отображение списка добавленных вершин;
* изменение координат добавленных вершин;
* удаление добавленных вершин.

В верхней части окна располагается набор кнопок, выполняющих следующие действия:

* удаление последней добавленной вершины;
* удаление всех добавленных вершин (очистка);
* возвращение первоначального масштаба и положения холста;
* генерация по заданному контуру сетки и экспорт её в файл формата Gmsh.

В правом нижнем углу задаётся величина среднего шага генерируемой сетки.

## Окно «База данных тканей»

Для взаимодействия пользователя с базой данных наполнителей служит окно «База данных тканей» (Рисунок 19).



Рисунок 19 – Интерфейс взаимодействия с базой данных тканей

В правой части окна располагается список имён всех материалов, загруженных из базы данных. При выборе материала в полях ввода слева отображаются его параметры.

Параметры в полях ввода можно редактировать. Для сохранения изменённых параметров в базе данных служит кнопка «Сохранить». Для возвращения в поля ввода параметров, в настоящий момент записанных в базу данных служит кнопка «Отмена».

При помощи кнопки «Удалить материал» можно полностью удалить материал из базы данных. В случае удаления из базы данных всех материалов происходит автоматическое повторное добавление всех стандартных материалов.

При нажатии кнопки «ОК» загружаются те, параметры материала, которые заданы в полях ввода в данный момент.

Также при помощи кнопок в верхней части окна пользователь может:

* добавить в базу данных новый материал;
* импортировать материалы из csv-файла;
* экспортировать базу данных в csv-файл.

Параметры наполнителя:

* ID - уникальный номер наполнителя, назначается автоматически, его невозможно изменить.
* Имя - строка. Уникально для каждой ткани. Может состоять из любых символов. Может быть отредактировано.
* Толщина - число с плавающей точкой. Измеряется в метрах (м).
* Проницаемость - число с плавающей точкой. Измеряется в квадратных метрах (м2).
* Пористость - число с плавающей точкой, от 0 до 1. Безразмерная величина.

## Окно «База данных связующих»

Для взаимодействия пользователя с базой данных связующих служит окно «База данных связующих» (Рисунок 20). Окна «База данных тканей» и «База данных связующих» имеют идентичный интерфейс и различаются только параметрами.

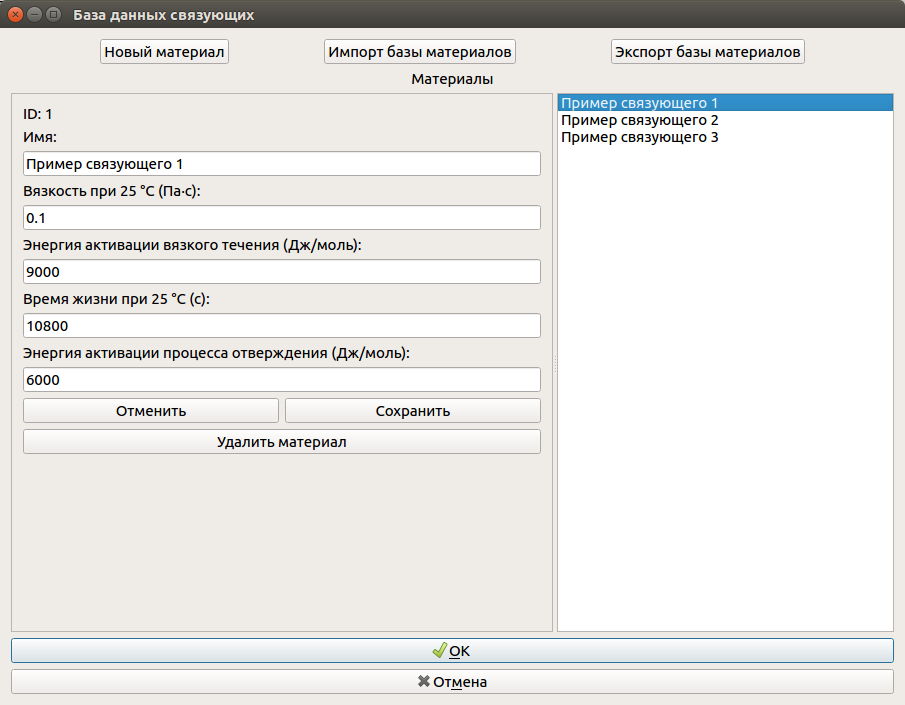


Рисунок 20 – Интерфейс взаимодействия с базой данных связующих

Каждое связующее характеризуется следующими параметрами:

* ID - уникальный номер связующего, назначается автоматически, его невозможно изменить.
* Имя - строка. Уникально для каждого типа связующего. Может состоять из любых символов. Может быть отредактировано.
* Номинальная вязкость – число с плавающей точкой. Динамическая вязкость жидкости, измеренная при температуре 25 °C (298,15 К). Измеряется в Паскалях на секунду (Па · с).
* Энергия активации вязкого течения – число с плавающей точкой. Используется для определения вязкости жидкости при заданной температуре. Измеряется в Джоулях в моль (Дж/моль).
* Время жизни связующего при 25 °C – число с плавающей точкой. Характеризует максимально возможный срок применения смешанного связующего, после которого оно станет желеобразным, и, в силу резкого роста вязкости, дальнейшая пропитка станет невозможной. Измеряется в секундах (с).
* Энергия активации процесса отверждения – число с плавающей точкой. Используется для определения времени жизни связующего при заданной температуре. Измеряется в Джоулях в моль (Дж/моль).

## Особенности реализации пользовательского интерфейса

Логика взаимодействия с интерфейсом пользователя, также как и программа моделирования, была реализована на языке C++.

### Визуальный редактор графического интерфейса

Для формирования окон, расположения в них и первичной настройки элементов управления использовался визуальный редактор, созданный по принципу WYSIWYG (от англ. *What You See Is What You Get*, «что видишь, то и получишь»). Такой редактор позволяет достаточно быстро, легко и, при этом, точно спроектировать пользовательский интерфейс. В результате работы визуального редактора генерируется XML-файл, в котором заданы позиции, параметры и иерархию расположения элементов пользовательского интерфейса. Перед компиляцией этот файл автоматически преобразуется препроцессором в C++ код.

На рисунке Рисунок 21 показана форма окна «Создание слоя», открытая в визуальном редакторе.

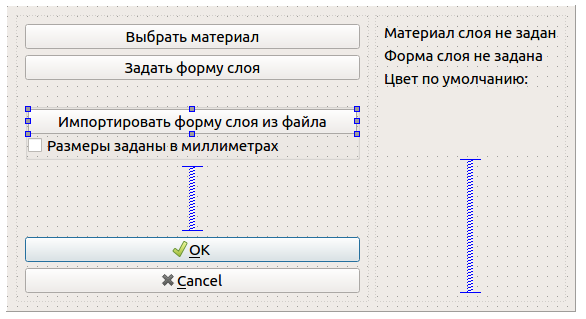


Рисунок 21 – Форма окна «Создание слоя», открытое в визуальном редакторе

### Метод взаимодействия интерфейса и алгоритма моделирования

Взаимодействие интерфейса пользователя и системы моделирования реализовано, по большей части, в соответствии с шаблоном проектирования MVP (от англ. *Model-View-Presenter*, «модель-вид-представитель»), Рисунок 22



Рисунок 22 – Шаблон проектирования «модель-вид-представитель»

Данный шаблон позволяет разделить ответственность в презентационной логике (отделения логики от отображения):

* *Модель* (англ. *Model*) – представляет собой логику работы приложения.
* *Вид* (англ. *View*) – реализует *отображение* данных (из Модели), обращается к Представителю за обновлениями.
* *Представитель* (англ. *Presenter*) – реализует взаимодействие между Моделью и Видом.

В данном случае в качестве модели выступает модуль, реализующий алгоритм моделирования *v\_sim*. В качестве вида выступают окна, создаваемые в визуальном редакторе. А в качестве представителя выступают классы, обеспечивающие связь пользовательского интерфейса с алгоритмом моделирования.

### Механизм сигналов-слотов

В программе была реализована структура, в которой интерактивные элементы вида сообщают представителю об изменении собственного состояния при помощи механизма сигналов-слотов. Модель сообщает представителю об изменении внутреннего состояния при помощи данного механизма.

Сигналы и слоты – подход, используемый в некоторых языках программирования и библиотеках, который позволяет реализовать шаблон «наблюдатель», минимизируя написание повторяющегося кода. Концепция заключается в том, что компонент может посылать сигналы, содержащие информацию о событии (например: был выделен текст «слово», была открыта вторая вкладка). В свою очередь, другие компоненты могут принимать эти сигналы посредством специальных функций – слотов. Система сигналов и слотов хорошо подходит для описания Графического интерфейса пользователя. Также механизм сигналов/слотов может быть применён для асинхронного ввода-вывода или уведомления о событиях. Благодаря метаобъектному компилятору автоматически генерируются классы C++, реализующие процессы передачи сигналов и их связывания со слотами.

## Внутренняя структура реализации пользовательского интерфейса

Диаграмма классов, отвечающих за интерфейс пользователя, представлена на рисунке Рисунок 23.

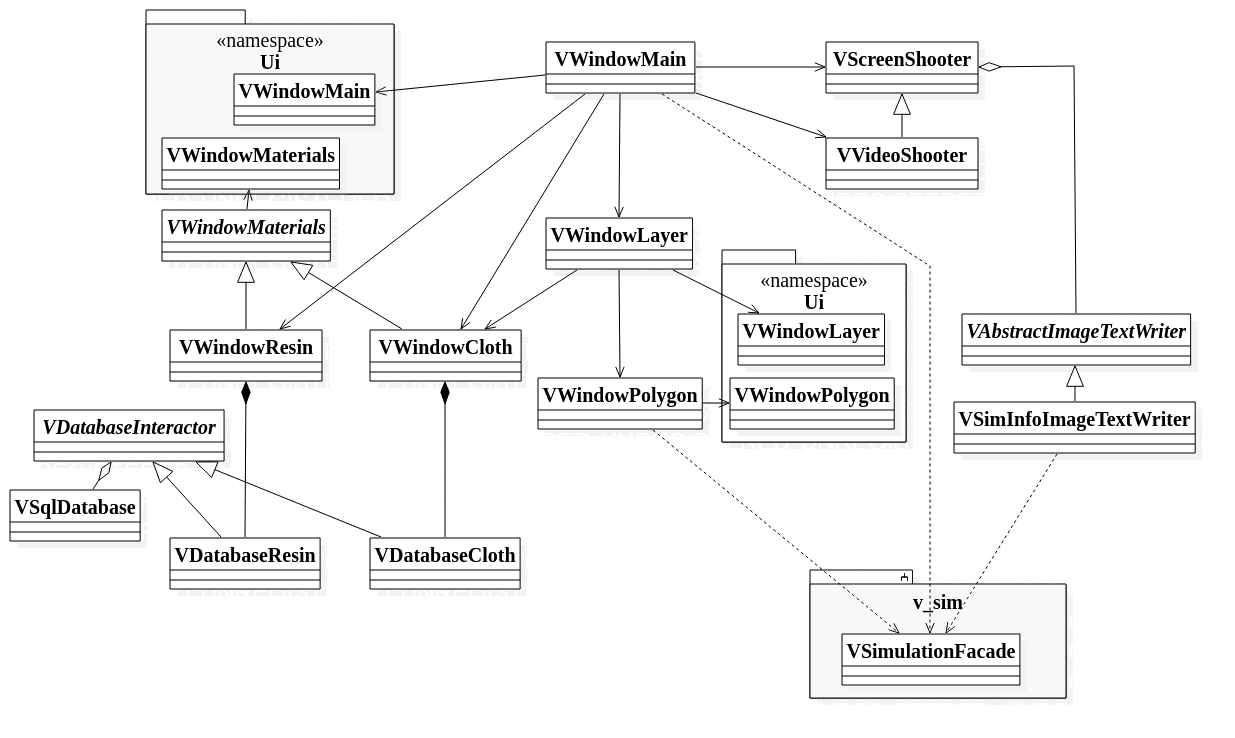


Рисунок 23 – Диаграмма классов интерфейса пользователя

### Классы вида (форм окон)

В пространстве имён *Ui* находятся классы, определяющие свойства и элементы окон. Данные классы *Виду*. Классы из этого пространства имён генерируются автоматически из созданных в визуальном редакторе форм окон. Пространство имён *Ui* содержит четыре класса.

***Ui::VWindowMain*** – определяет элементы формы основного окна.

***Ui::VWindowMaterials*** – определяет элементы форм окон «База данных тканей» и «База данных связующих».

***Ui::VWindowLayer*** – определяет элементы формы окна «Создание слоя».

***Ui::VWindowPolygon*** – определяет элементы формы окна «Задание границ слоя».

### Классы представители

Классы *представители* предназначены для реализации взаимодействия форм окон с внутренним представлением программы (с *моделью*).

***VWindowMain*** – обеспечивает взаимодействие главного окна и алгоритма моделирования процесса пропитки.

***VWindowMaterials*** – абстрактный класс. Служит для обеспечения взаимодействия форм окон «База данных тканей» и «База данных связующих» с соответствующими таблицами базы данных.

***VWindowCloth*** – класс для обеспечения взаимодействия формы окна «База данных тканей» с соответствующей таблицей базы данных. Наследуется от *VWindowMaterials*.

***VWindowResin*** – класс для обеспечения взаимодействия формы окна «База данных связующих» с соответствующей таблицей базы данных. Наследуется от *VWindowMaterials*.

***VWindowLayer*** – обеспечивает логику работы окна «Создание слоя».

***VWindowPolygon*** – обеспечивает логику работы окна «Задание границ слоя» и его взаимодействие с заданными в модуле симуляции параметрами.

### Классы взаимодействия с базой данных

Для обеспечения взаимодействия программы с базой данных наполнителей и связующих создано несколько классов.

***VSqlDatabase*** – класс, отвечающих за создание, открытие и закрытие подключения к базе данных. Реализован в соответствии с шаблоном проектирования «Одиночка».

***VDatabaseInteractor*** – абстрактный класс, определяющий интерфейс и реализующий основные методы чтения и записи в таблицы наполнителей и связующих базы данных.

***VDatabaseCloth*** –класс, реализующий методы чтения и записи в базу данных, специфичные для таблицы наполнителей. Наследуется от *VDatabaseInteractor*.

***VDatabaseResin*** –класс, реализующий методы чтения и записи в базу данных, специфичные для таблицы связующих. Наследуется от *VDatabaseInteractor*.

### Классы записи процесса пропитки

Для записи серий снимков и видео с нанесённой поверх кадров информацией о ходе процесса пропитки созданы следующие классы.

***VScreenShooter*** – класс, реализующий запись серий снимков с заданным интервалом времени.

***VVideoShooter*** – класс, реализующий запись серий снимков с заданным интервалом времени с последующим формированием по ним видеофайла. Наследуется от *VScreenShooter.*

***VAbstractImageTextWriter*** – абстрактный класс, реализующий добавление на изображение какого-либо текста.

***VSimInfoImageTextWriter*** – класс, реализующий добавление на изображение текста, содержащего информацию о текущем состоянии процесса пропитки. Наследуется от *VAbstractImageTextWriter*.

## Тестирование и отладка учебно-демонстрационной программы

Для разработанной программы проводилось, в первую очередь функциональное тестирование. Многократно проводилась пропитка деталей с самыми различными конфигурациями, выставлялись различные значения параметров. Регулярно производились тесты, заключающиеся в осуществлении взаимодействия со всеми элементами управления в произвольном порядке и записи в поля ввода произвольных значений. Был достигнут результат, при котором программа продолжала исправно функционировать после всех выполненных последовательностей действий.

Системное тестирование программы регулярно выполнялось на трёх операционных системах (ОС): Microsoft Windows 7, Linux Ubuntu 18.04 и macOS Heigh Sierra. На каждой из приведённых систем полностью проверялся на работоспособность весь набор функций программы, а также идентичность их работы на различных платформах. Также по окончании разработки была полностью протестирована работоспособность программы в системах Microsoft Windows 10 и Linux Ubuntu 16.04. На рисунках Рисунок 24, Рисунок 25 и Рисунок 26 представлен пример, в котором было произведено моделирование пропитки одной и той же преформы в трёх различных ОС в течение 45 минут. Как можно судить по снимкам экрана, итоговый результат во всех случаях результаты идентичны.

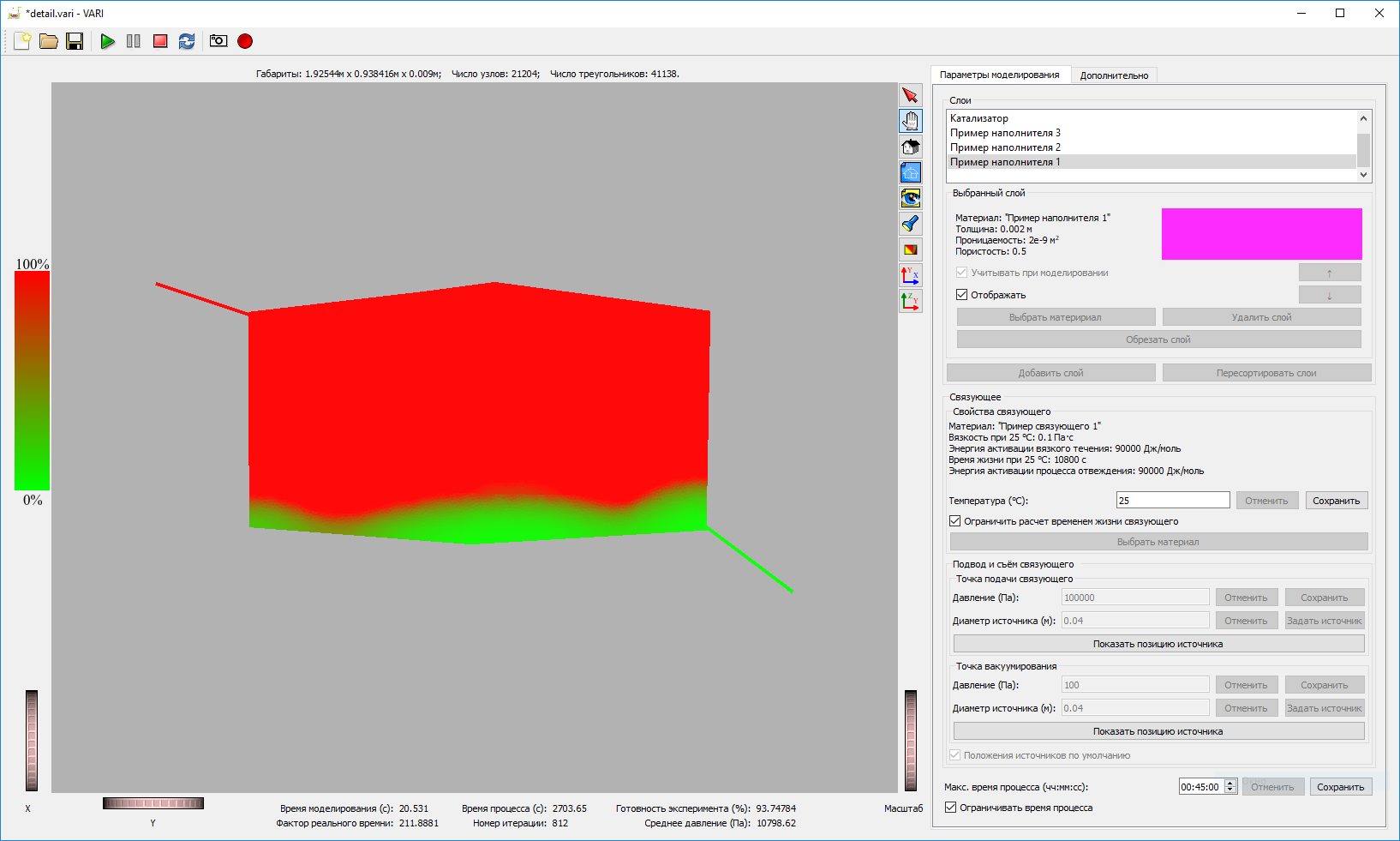


Рисунок 24 – Пример работы программы в Microsoft Windows 10

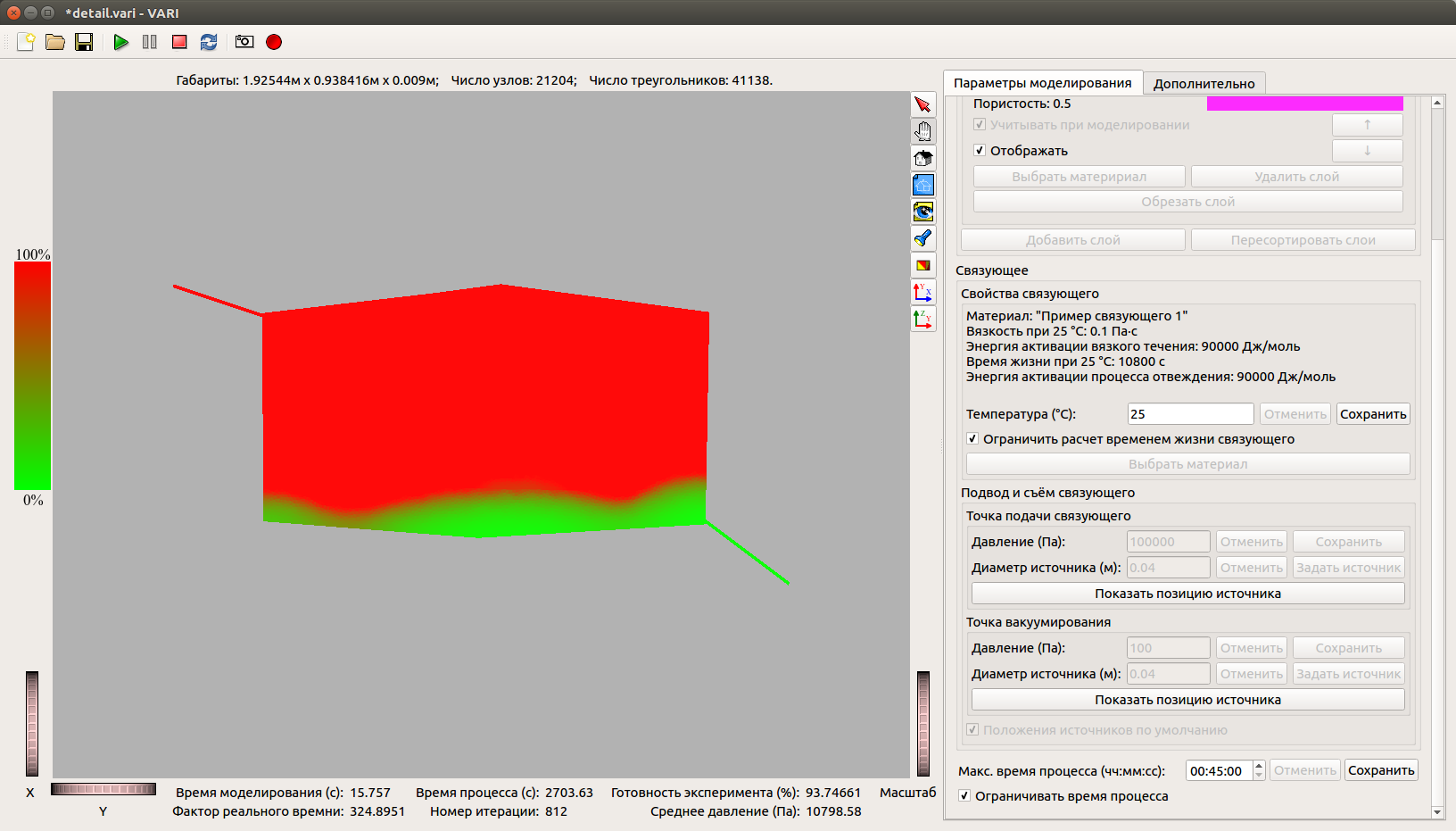


Рисунок 25 – Пример работы программы в Linux Ubuntu 16.04

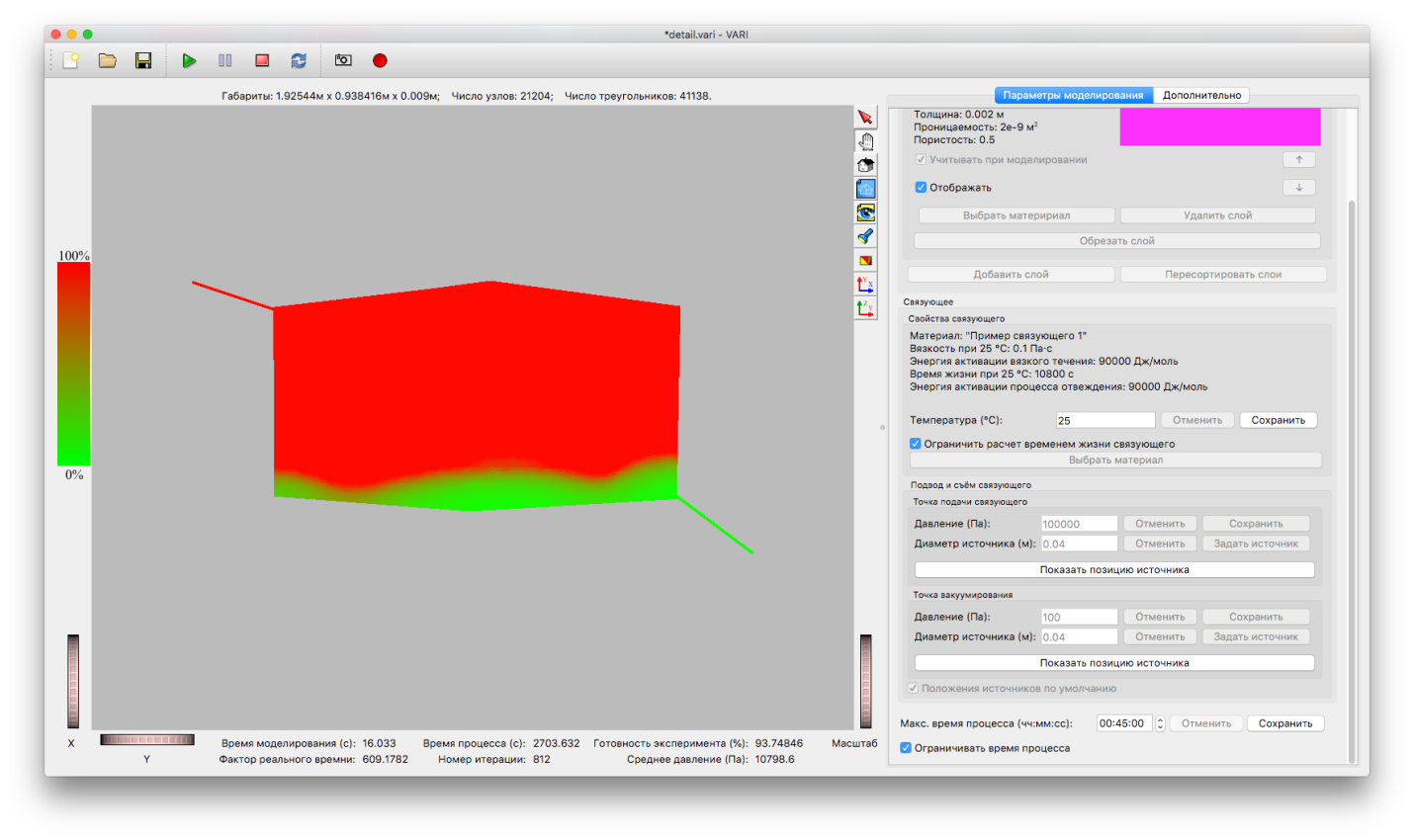


Рисунок 26 – Пример работы программы в macOS High Sierra

Также было уделено внимание тестированию производительности. Ниже приведены результаты одного из тестов, в котором осуществлялась пропитка детали, состоящей из слоёв с габаритами 2 x 1 м . При формировании сетки для вычислений был взят средний шаг 0,015 м . Моделирование производилось на компьютере со следующими характеристиками: восьмиядерный процессор AMD FX-8120, видеокарта AMD Radeon R9 200 Series с 2 ГБ памяти, оперативная память 16 ГБ, операционная система Linux Ubuntu 16.04. Результаты измерений представлены в Таблица 2.

Таблица 2 – Измерение времени моделирования деталей с различным числом слоёв

|  |  |  |  |
| --- | --- | --- | --- |
| Число слоёв | Число узлов | Число треугольников | Время моделирования, с |
| 1 | 10076 | 19784 | 61 |
| 2 | 20152 | 39568 | 222 |
| 4 | 40304 | 79136 | 724 |
| 6 | 60456 | 118704 | 1724 |

На Рисунок 27 представлен график зависимости времени моделирования от числа узлов, соответствующий данным из таблицы Таблица 2.

Рисунок 27 – Зависимость времени моделирования от числа узлов сетки

Нелинейная зависимость времени моделирования от числа узлов объясняется тем, что узлы в различных слоях взаимодействуют друг с другом, на что уходят дополнительные вычислительные рессурсы.

Результаты тестирования производительности свидетельствуют о том, что достигнутый уровень производительности является вполне комфортным для использования программы.

При нахождении ошибок в работе программы осуществлялась её отладка. Для этой цели использовались программы-отладчики GNU Debugger в Linux и macOS и Microsoft Visual Studio Debugger в Windows.

# Разработка элементов базы данных материалов учебно-демонстрационной программы

При формировании базы данных выполняются измерения основных характеристик наполнителей и связующих. Для вычисления значений требуемых параметров проводятся эксперименты. В результате формируется база данных, включающая две таблицы: наполнители и связующие. Изначально в таблицу закладывается 5 типов наполнителей и 3 типа связующих.

Для хранения базы данных выбрана система управления базами данных (СУБД) SQLite. Основными преимуществами данной СУБД являются:

* Отсутствие необходимости настройки сервера СУБД.
* Возможность простого распространения со своим продуктом.
* Полностью свободная лицензия.
* Кроссплатформенность.
* Высокая скорость (на простых операциях в разы быстрее MySQL и на порядок быстрее PostgreSQL [4]).
* Безопасность. База данных хранится в одном файле, права доступа к которому можно контролировать стандартными средствами операционной системы.
* Возможность использовать разные языки программирования, в том числе C++.

## Методика измерения параметров наполнителей

### Методика измерения проницаемости

В общем случае, когда среда является анизотропной, проницаемость характеризуется тензором [*K*]. В [5] приведён одномерный канальный метод, позволяющий определить плоскостные компоненты тензора проницаемости в главной системе координат. В случае изотропной среды достаточно определить только одну компоненту тензора.

Эксперимент проводится путём осуществления вакуумной инфузии прямоугольных образцов. Смола поступает с одного края образца через линейный источник, на противоположной стороне поддерживается постоянный уровень вакуума. Экспериментально найденная зависимость координаты фронта от времени подставляется в аналитическое выражение для проницаемости. На рисунке Рисунок 28А представлена схема эксперимента, на рисунке Рисунок 28Б – пример экспериментальной установки. На рисунке Рисунок 28 введены следующие обозначения: 1 - пропитанная зона, 2 – фронт пропитки, 3 – непропитанная зона.

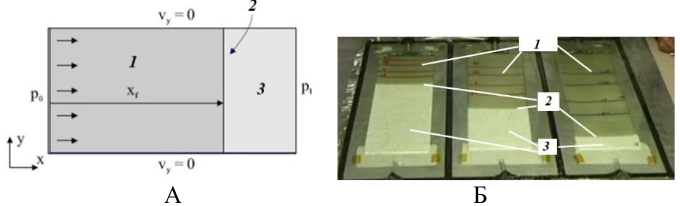


Рисунок 28 – Схема одноканального метода (А)  
и экспериментальная установка (Б).

Проницаемость вдоль направления *x* определяется по следующей формуле:

*,* (1.12)

[*Kx*] = [ (Па · с · м2) / (Па · с) ] = [м2],

где:

*t0* - момент времени начала измерения, с;

*tf* - момент времени окончания измерения, с;

∆*t* = *tf - t0* - время измерения процесса, с;

*xf*(*t*) - координата фронта пропитки (начало координат совпадает с положением источника), м;

µ - вязкость смолы, Па·с;

∆*p* - разность давлений на источнике впуска связующего и давления на фронте, Па:

∆*p = patm - pf ,*

*patm* - давление в источнике впуска связующего, Па;

*pf* - давление вакуумирования (давление на фронте течения), Па.

### Методика измерения пористости

Методы исследования пористых структур описаны в [6].

Пористость материала, не содержащего закрытых пор (не сообщающихся с поверхностью тела), можно измерить следующим образом:

1) Измерить габариты образца и по результатам измерений вычислить его общий объём *V.*

2) Поместить образец в жидкостный волюмометр (сосуд, проградуированный в единицах объёма). Рабочая жидкость должна обладать хорошей смачивающей способностью и не взаимодействовать с телом. Объём скелета образца *VТ* определяется по объёму вытесненной им жидкости.

3) По полученным значениям вычислить величину пористости по формуле (1.13):

. (1.13)

## Методика измерения параметров связующих

### Методика измерения вязкости

Динамическая вязкость – характеристика вещества, численно равная силе трения, возникающей между двумя слоями жидкости площадью по 1 м2 каждый при градиенте скорости, равном 1 м/с на метр. Размерность динамической вязкости [µ] = [Па·с].

Коэффициент динамической вязкости зависит от природы жидкости и для жидкости с повышением температуры уменьшается.

Для измерения вязкости (вискозиметрии) применяют ряд экспериментальных методов. Измерить динамическую вязкость при определённой температуре можно при помощи метода Стокса [7].

Для определения вязкости жидкости по методу Стокса берётся высокий цилиндрический сосуд с исследуемой жидкостью (Рисунок 29). На сосуде имеются две кольцевые метки А и В, расположенные на расстоянии *l* друг от друга. Уровень жидкости должен быть выше верхней метки на *l*0 = 4…5 см, чтобы к моменту прохождения шарика мимо верхней метки его скорость можно было считать установившейся.

Бросая шарик с радиусом *r* и плотностью ρ в сосуд, наполненный исследуемой жидкостью с известной плотностью ρ0, отмечают по секундомеру время *t* прохождения шариком расстояния *l* = АВ между двумя метками.

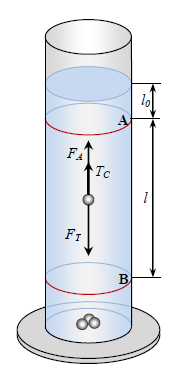


Рисунок 29 – Лабораторная установка для измерения вязкости.

При падении шарика радиусом *r* в цилиндрической трубе радиусом *R*0, высотой *h* с учётом влияния границ формула Стокса преобразуется к виду:

, (1.14)

*,*

где:

*g* = 9,8 м/с2 – ускорение свободного падения;

*r* – радиус шарика, м;

*R*0 – радиус сосуда, м;

*h* – высота сосуда, м;

ρ – плотность шарика, кг/м3;

ρ0 – плотность шарика, кг/м3;

*l* – расстояние между метками сосуда, м;

*t* – время прохождения шариком расстояния между метками, с.

Таким образом, зная плотности материала шарика и жидкости, радиусы шарика и сосуда, скорость установившегося движения шарика, по формуле (1.14) можно вычислить динамическую вязкость жидкости.

Для занесения номинального значения динамической вязкости в базу данных её измерение следует выполнять при температуре, соответствующей стандартному состоянию, – 25 °C (298,15 К).

### Методика измерения энергии активации вязкого течения связующего

Вязкость связующего имеет зависимость от температуры. Связь вязкости и температуры может быть выражена при помощи формулы Френкеля-Андраде (1.8):

,

Используя формулу (1.8) возможно определить **энергию активации вязкого течения по экспериментальным данным.**

По приведённой в пункте 4.2.1 методике определяют вязкость полимерного связующего при значениях температуры, различающихся более чем на 20 К, за результат измерений принимают среднее значение не менее 3 экспериментов при каждой температуре.

По результатам эксперимента можно рассчитать энергию активации вязкого течения, используя следующую формулу:

,

*,*

гдеR – универсальная газовая постоянная **(**R **= 8,314 Дж/(моль · К); µ1, µ 2** – экспериментальные значения динамической вязкости, Па·с; Т**1,** Т**2** – значения температуры, при которых определена вязкость, К.

### Методика измерения времени жизни связующего

При тепловом воздействии на **термореактивные полимеры** их **вязкость** с некоторого момента времени начинает интенсивно возрастать, что свидетельствует о начале процесса гелеобразования.

Использовать связующее для изготовления изделий или для пропитки необходимо в промежутке времени от момента приготовления связующего до момента, когда оно переходит в гелеобразное состояние. Такой промежуток времени называется временем жизни (жизнеспособность связующих).

Для экспериментального определения времени гелеобразования используют прибор, схема которого приведена на рисунке Рисунок 30.

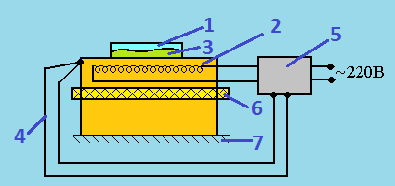


Рисунок 30 – Схема приспособления для определения времени гелеобразования термореактивных связующих

На рисунке Рисунок 30 используются следующие обозначения:1 – кювета; 2 – обогреваемая плита; 3– исследуемая жидкость; 4 – термопара; 5– регулятор температур; 6 – асбестовая прокладка; 7 – стол.

За время гелеобразования принимают промежуток времени (в секундах) от момента заливки связующего в емкость до момента, когда вытягиваемое из емкости связующее обрывается (вытягивание не выше 20 мм над поверхностью связующего).

Для занесения номинального значения времени жизни связующего в базу данных его измерение следует выполнять при температуре, соответствующей стандартному состоянию, – 25 °C (298,15 К).

### Методика измерения энергии активации процесса отверждения

**Зависимость времени гелеобразования от температуры** идентична зависимости вязкости от температуры и описывается формулой (1.11):

.

Используя формулу (1.11) можно определить **энергию активации отверждения по экспериментальным данным**.

По предложенной в пункте 4.2.3 методике определяют время жизни связующего при значениях температуры, различающихся более чем на 20 К, за результат измерений принимают среднее значение не менее 3 экспериментов при каждой температуре.

По результатам эксперимента можно рассчитать энергию активации отверждения, используя следующую формулу:

,

*,*

гдеR – универсальная газовая постоянная **(**R **= 8,314 Дж/(моль · К)**; ***t*ж1, *t*ж2** – экспериментальные значения времени жизни, с; Т**1,** Т**2** – значения температуры, при которых определена вязкость, К.

# Достигнутые результаты и их оценка

В результате выполнения работ по 2 (заключительному) этапу НИОКР завершена разработка программной составляющей разрабатываемого комплекса (учебно-демонстрационной программы), в частности:

* Разработаны алгоритмы учебно-демонстрационной программы;
* Разработан интерфейс пользователя учебно-демонстрационной программы;
* Разработана учебно-демонстрационная программа на основе созданных алгоритмов и интерфейса пользователя;
* Разработана методика экспериментов по определению характеристик материалов, закладываемых в базу данных.

По результатам выполнения второго этапа НИОКР можно сделать следующие выводы:

* Разработанный алгоритм моделирования процесса вакуумной инфузии позволяет выполнить оценку характера пропитки преформы, состоящей из слоёв с различными характеристиками;
* Благодаря разработанным и применённым алгоритмам достигнута высокая производительность выполнения моделирования, достаточная для комфортного использования программы на любых современных компьютерах;
* Разработанный графический интерфейс является понятным с точки зрения пользователя и позволяет ему создавать и моделировать пропитку деталей с достаточно сложной структурой;
* Обеспечение кроссплатформенности позволяет использовать программу в любой из трёх наиболее распространённых на сегодняшний день операционной системе для персональных компьютеров (Microsoft Windows, GNU/Linux, macOS).
* Разработанная методика проведения экспериментов по определению характеристик материалов позволяет составить базу данных наполнителей и связующих.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Henne M., Barandun G. A. Simulation of LCM processes using cellular automats //The 10th International Conference on Flow Processes in Composite Materials (FPCM10) – 2010.
2. Френкель Я. И. Кинетическая теория жидкостей. Наука. – 1975.
3. Уильямс Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. – Litres, 2017.
4. Database Speed Comparison [Электронный ресурс] URL: https://www.sqlite.org/speed.html (дата обращения: 25.04.2019).
5. Щеглов Б. А., Сафонов А. А. Теоретические основы и прикладные задачи технологии композитов. – 2015.
6. Фандеев В. П., Самохина К. С. Методы исследования пористых структур //Интернет-журнал Науковедение. – 2015. – Т. 7. – №. 4 (29).
7. Никулин С.С., Чех А.С. Определение вязкости жидкости методом Стокса: методические указания. – Тамбов: Изд-во ГОУ ВПО ТГТУ, 2011.