Федеральное государственное бюджетное образовательное учреждение

высшего образования

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет имени***  ***Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

Факультет ***Информатика и системы управления***

Кафедра ***Компьютерные системы и сети (ИУ6)***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ**

**ПО ДИСЦИПЛИНЕ «ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММНЫХ СИСТЕМ»**

**НА ТЕМУ:**

**Программа моделирования физической среды методом частиц**

Студент гр. ИУ6-53**\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Воробьев Р.И.\_**\_**

(Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы,

доцент  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_Гуренко В.В.\_\_**\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2017

**Реферат**

Расчётно-пояснительная записка 37 страниц, 17 рисунков, 6 таблиц, 5 источников, 3 приложения.

МОДЕЛИРОВАНИЕ, МОЛЕКУЛЯРНАЯ ДИНАМИКА, МЕТОД ЧАСТИЦ, C++, Qt, OpenGL.

Объектом разработки является программа, моделирующая физическую среду на молекулярном уровне.

В процессе выполнения работы были разработаны и реализованы алгоритмы, описывающие взаимодействие частиц друг с другом и их поведение в веществе. Программа позволяет численно задавать параметры моделирования и в качестве результата визуально отображает процесс моделирования.

Задача реализована на языке программирования С++ с использованием фреймворка Qt и графической библиотеки OpenGL.

**Содержание**

Введение 6

1 Анализ предметной области, уточнение спецификаций 7

1.1 Разработка математической модели 7

1.2 Выбор архитектуры программного обеспечения 12

1.3 Выбор подхода к проектированию и разработке 12

1.4 Выбор модели жизненного цикла программы 12

1.5 Выбор языка и среды программирования 12

1.6 Объектная декомпозиция предметной области 13

1.7 Определение вариантов использования 14

2 Разработка пользовательского интерфейса 17

2.1 Разработка меню программы 17

2.2 Построение графов абстрактных диалогов 17

2.3 Построение графа состояний интерфейса 19

2.4 Разработка диаграммы классов интерфейсной части концептуального уровня 20

2.5 Разработка экранных форм 22

3. Разработка компонентов программного продукта 25

3.1 Разработка диаграммы классов предметной области концептуального уровня 25

3.2 Разработка диаграмм классов программы уровня реализации 26

3.2.1 Разработка диаграммы классов предметной области уровня реализации 26

3.2.2 Разработка диаграммы классов интерфейсной части уровня реализации 29

3.2.3 Разработка диаграммы класса графического представления 30

3.3 Схема компоновки программного продукта 31

4. Тестирование программного продукта 33

4.1. Выбор стратегии тестирования 33

4.2 Функциональное тестирование 33

4.3 Оценочное тестирование 35

Заключение 36

Список использованных источников 37

Приложение А. Техническое задание 38

Приложение Б. Руководство пользователя 44

Приложение В. Фрагмент исходного кода программы 51

**Введение**

Компьютерное моделирование – это современный и перспективный метод познания. Компьютерные модели проще и удобнее исследовать в случаях, когда реальные эксперименты затруднены из-за финансовых, физических или других препятствий. Конечно, моделирование нельзя противопоставлять натурным экспериментам, однако исследователям для понимания процессов, происходящих в веществе, иногда просто необходима визуализация или расчет движения атомов и структуры моделируемой системы. Программирование поведения сложных систем, состоящих из множества взаимодействующих составляющих, например, вещества, состоящего из атомов и молекул, является довольно трудной задачей моделирования.

На сегодняшний день существует довольно много программ для моделирования молекулярной динамики, однако в основном это иностранные пакеты программирования, для изучения которых требуется знание английского языка. К тому же подобные программы, например, программная система моделирования XMD, зачастую разрабатываются в узких целях моделирования для научных нужд, и многие из них не имеют визуализации самого процесса моделирования, поэтому они бывают сложными для понимания. Данная программа включает в себя простой механизм управления моделированием и рассчитана не на специалиста, а на рядового пользователя для ознакомления с процессами, происходящими в веществе на молекулярном уровне, или же просто в развлекательных целях.

**1 Анализ предметной области, уточнение спецификаций**

* 1. **Разработка математической модели**

Задача моделирования физической среды сводится к разработке математической модели, описывающей поведение взаимодействующих между собой атомов вещества. Среди множества методов решения данной задачи широко распространенным является метод частиц[1].

Одним из наиболее разработанных вариантов данного метода является метод молекулярной динамики[2], который на протяжении последних десятилетий интенсивно используется для исследования физико-химических свойств материалов. Метод основан на представлении вещества совокупностью взаимодействующих частиц (атомов или молекул), законы движения которых описываются классическими уравнениями динамики, а их взаимодействие можно представить в форме классических потенциальных сил, основным свойством которых является отталкивание при сближении и притяжение при удалении.

Уравнение, описывающее движение частиц, имеет следующий вид:

, (1)

где и – векторы положения и скорости k-ой частицы,

, , , , (2)

– масса частицы, – ускорение, получаемое частицей, и описывают консервативную и диссипативную составляющую взаимодействия между частицами, и описывают внешнее консервативное и диссипативное силовое поле. Рассмотрим подробнее каждый силовой фактор.

Консервативная составляющая взаимодействия определяется следующим образом:

, (3)

, (4)

где — скалярная сила взаимодействия между частицами, — потенциал взаимодействия. Величина является важнейшим силовым фактором.

В качестве потенциала взаимодействия был выбран потенциал Леннард-Джонса. Он представляет собой простую модель парного взаимодействия неполярных молекул, описывающую зависимость энергии взаимодействия двух частиц от расстояния между ними. Эта модель достаточно реалистично передаёт свойства реального взаимодействия сферических неполярных молекул и поэтому широко используется в расчётах и при компьютерном моделировании.

Потенциал Леннард-Джонса имеет следующий вид:

, (5)

где – расстояние между центрами частиц, – глубина потенциальной ямы, – расстояние, на котором сила взаимодействия становится равной нулю.

График данного потенциала изображен на рисунке 1.

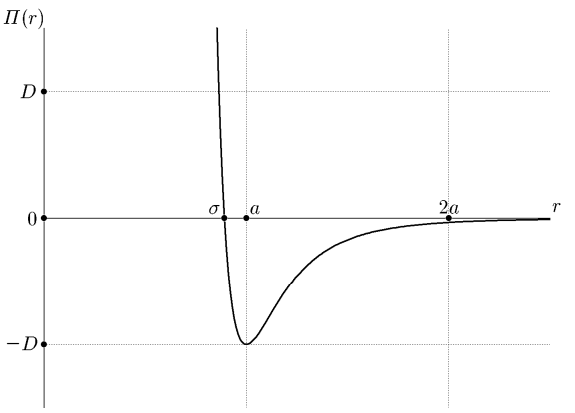


Рисунок 1 – График потенциала Леннард-Джонса

Подставив формулу (5) в формулу (4) и выполнив преобразования, получим:

, (6)

Диссипативная составляющая взаимодействия предназначена для описания внутренней диссипации тела. Поскольку тепловая энергия является характеристикой термодинамической системы, а не отдельных частиц, движение которых рассматривается в рамках разрабатываемой модели, диссипативной составляющей можно пренебречь.

Внешнее консервативное силовое поле обычно рассматривается для задания внешних массовых силовых воздействий и распределено во всем объеме пространства. Для разрабатываемой модели единственным таким воздействием является гравитационное воздействие. Исходя из этого, внешнее консервативное силовое поле определяется следующим образом:

*,* (7)

где – масса частицы, – оператор Набла, – гравитационный потенциал. В рамках разрабатываемой модели вектор направлен вниз по оси Y и имеет неизменную величину и направление в любой точке пространства. Поскольку данный вектор характеризует ускорение, получаемое частицей массой под действием гравитационного поля, то для разрабатываемой модели его можно считать ускорением свободного падения.

Диссипативное силовое поле , как правило, используется для задания силовых граничных условий и локализовано вблизи некоторых поверхностей, часто являющихся границами области расчета. Также его применяют для отвода энергии из системы посредством внешней диссипации. Данное явление может обеспечить сила упругости, которая возникает при неупругом ударе частиц о границы поля моделирования.

Тогда согласно второму закону Ньютона:

, (8)

где – сила, проложенная частице, – производная импульса частицы по времени.

При неупругом ударе частицы о границу поля она теряет часть своей кинетической энергии. Тогда изменение импульса частицы за время отскока равно:

, (9)

где – коэффициент гашения удара, – масса частицы, – скорость частицы до удара.

Дифференцировав уравнение (9) по времени и подставив результат в формулу (8), получим формулу для расчета диссипативного воздействия:

, (10)

где – вектор положения частицы.

В результате разработана математическая модель, описывающая законы движения частиц в веществе. С точки зрения математики, процесс моделирования представляет собой решение задачи Коши для уравнения (1), где начальными условиями являются координаты и скорость каждой частицы.

На основе анализа предметной области построена концептуальная модель будущей программы (рисунок 2).

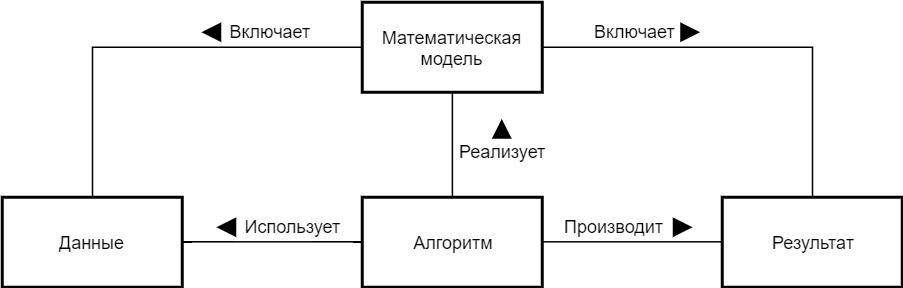


Рисунок 2 – Схема концептуальной модели программного продукта

Математическая модель в виде уравнения (1) включает в себя исходные данные (значения известных параметров) и результат (решение задачи Коши). Алгоритм реализует эту модель, используя данные для вычисления результата.

**1.2 Выбор архитектуры программного обеспечения**

Разрабатываемая программа рассчитана на одного пользователя, поэтому она имеет однопользовательскую архитектуру.

**1.3 Выбор подхода к проектированию и разработке**

В процессе анализа предметной области стало ясно, что в ней можно выделить отдельные сущности с уникальными параметрами и связи между ними. Исходя из этого, был выбран объектно-ориентированный подход к проектированию. В таком случае наиболее подходящим вариантом похода к разработке станет нисходящий подход.

**1.4 Выбор модели жизненного цикла программы**

Создаваемый программный продукт является прототипом и нацелен на будущее расширение как математической модели, так и программной реализации. Исходя из этого, наиболее подходящим вариантом модели жизненного цикла программы является спиральная модель. Такая модель позволит с каждой итерацией совершенствовать разрабатываемый программный продукт и расширять его функционал.

**1.5 Выбор языка и среды программирования**

Для разработки программного продукта был выбран кроссплатформенный фреймворк Qt. Он поддерживает разработку на языке программирования C++, который в свою очередь является объектно-ориентированным, как и выбранный подход к проектированию. Также Qt имеет собственную среду разработки Qt Creator. Она включает в себя визуальные средства для разработки интерфейса, а также имеет библиотеку для работы с OpenGL, с помощью которой можно графически отображать процесс моделирования.

**1.6 Объектная декомпозиция предметной области**

Представим разрабатываемый программный продукт в виде совокупности объектов, взаимодействующих между собой. Анализируя предметную область, можно выделить основные сущности предметной области и связи между ними[3]. Объектная декомпозиция программы представлена на рисунке 3.

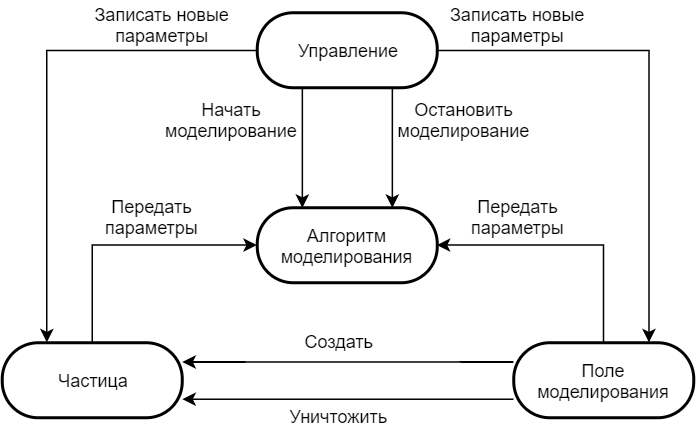


Рисунок 3 – Схема объектной декомпозиции программного продукта

Рассмотрим выделенные сущности и связи между ними.

Сущность «Управление» управляет работой программы. Она может устанавливать новые параметры сущностям «Частица» и «Поле моделирования», а также осуществлять запуск и остановку процесса моделирования.

Сущность «Поле моделирования» является областью, где происходит процесс моделирования. Сущность может создавать и удалять «Частицы» и передавать свои параметры «Алгоритму моделирования» для осуществления расчетов.

«Частица» является ключевой сущностью предметной области, поведение которой вычисляется и моделируется программой. «Частица» может передавать свои параметры «Алгоритму моделирования» для осуществления расчетов.

«Алгоритм моделирования» – сущность, которая реализует математическую модель предметной области, выполняя необходимые вычисления.

**1.7 Определение вариантов использования**

Поскольку программа использует однопользовательскую архитектуру, управляющей сущностью является Пользователь. В процессе анализа требуемой функциональности программного продукта были выделены следующие варианты использования[4]:

– задание параметров моделирования;

– моделирование физической среды;

– получение справки.

Таблица 1 – Вариант использования **Задание параметров моделирования**

Типичный ход событий

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1. Пользователь выбирает вариант задания параметров моделирования:  а) Если выбран ввод новых данных, см. раздел Ввод / редактирование параметров моделирования  б) Если выбрано задание по умолчанию, см. раздел Задание параметров по умолчанию.  3. Пользователь подтверждает запись данных. | 2. Система записывает данные в параметры моделирования.  6. Система принимает записывает данные в параметры моделирования. |

Таблица 2 – Раздел **Ввод / редактирование параметров моделирования**

Типичный ход событий

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1. Пользователь выбрал Ввод / редактирование параметров моделирования  3. Пользователь вводит новые данные | 2. Система ожидает ввод новых данных в соответствующие поля.  4. Система проверяет корректность введенных данных |

Альтернатива

4. Если обнаружены некорректные данные, система запрещает их ввод.

Таблица 3 – Раздел **Задание параметров по умолчанию**

Типичный ход событий

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1. Пользователь выбрал Задание параметров по умолчанию | 2. Система отображает данные по умолчанию в соответствующих полях. |

Таблица 4 – Вариант использования **Получение справки**

Типичный ход событий

|  |  |
| --- | --- |
| Действия пользователя | Отклик системы |
| 1. Пользователь выбрал Получение справки. | 2. Система открывает окно со справочной информацией. |

Вариант Задание параметров моделирования включает несколько вариантов, различающихся способом определения данных (ввод новых данных или задание по умолчанию), следовательно, они расширяют данный вариант. На основании выделенных вариантов использования была построена диаграмма вариантов использования, изображенная на рисунке 4.



Рисунок 4 – Диаграмма вариантов использования

Диаграмма позволяет наглядно представить ожидаемое поведение системы.

**2. Разработка пользовательского интерфейса**

Фреймворк Qt является объектно-ориентированным, а его среда разработки предоставляет большие возможности для создания графического оконного интерфейса[5].

**2.1 Разработка меню программы**

При проектировании интерфейса было решено организовать главное меню, структурная схема которого изображена на рисунке 5.



Рисунок 5 – Структурная схема меню

В нем пользователь может открыть непосредственно окно моделирования, окно со справочной информацией или выйти из программы.

**2.2 Построение графов абстрактных диалогов**

После запуска программы появляется главное меню. Пользователь может выбрать один из пунктов меню: «Начать моделирование», «Справка» или «Выход». Данный диалог управляется пользователем. Его граф изображен на рисунке 6.

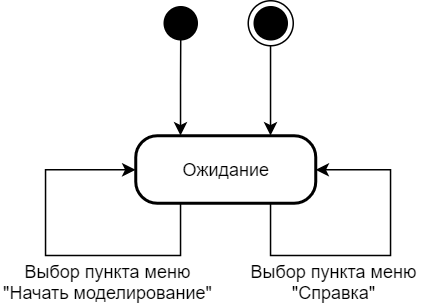


Рисунок 6 – Граф абстрактного диалога главного меню, управляемого пользователем

При запуске диалога «Моделирование» программа записывает все необходимые для моделирования параметры. Далее программа по запросу пользователя запускает процесс моделирования. Остановка процесса происходит также по запросу пользователя или же при изменении параметров моделирования.

На основе вышесказанного был построен граф абстрактного диалога «Моделирование» (рисунок 7).

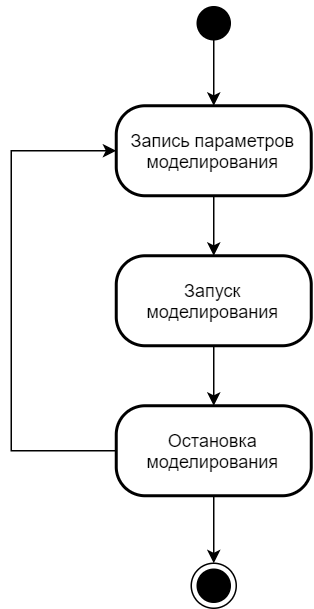


Рисунок 7 – Граф абстрактного диалога «Моделирование» комбинированного типа

Данный граф является комбинированным. Он учитывает как наличие сценария работы программы, так и отклонения от него по желанию пользователя.

**2.3 Построение графа состояний интерфейса**

В результате разработки интерфейса программы был построен граф состояний интерфейса (рисунок 8).

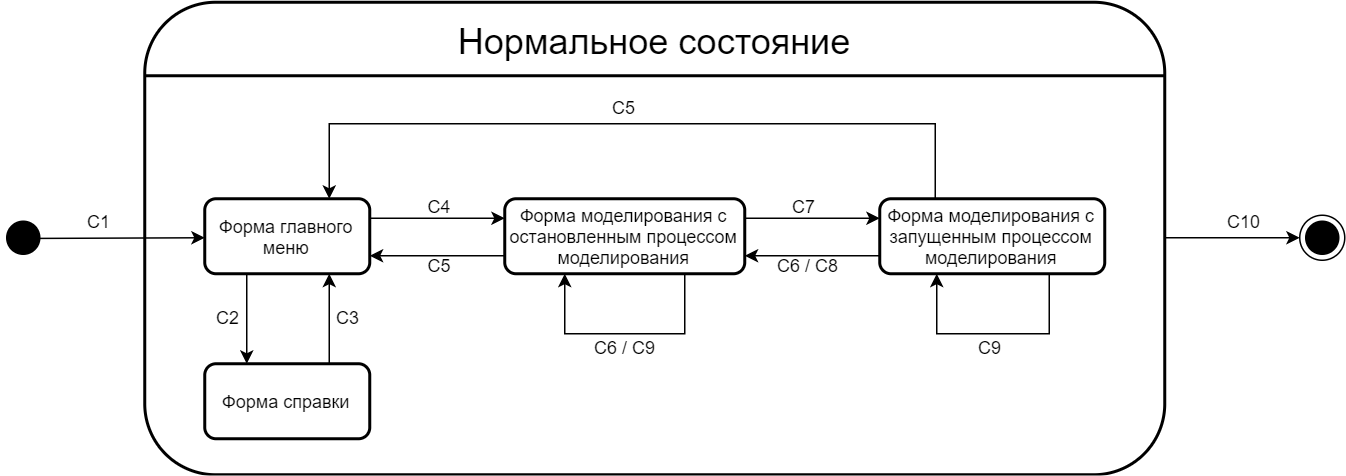


Рисунок 8 – Граф состояний интерфейса

На графе отмечены следующие переходы между состояниями:

C1 – инициализация программы;

C2 – нажатие кнопки «Справка» формы главного меню;

C3 – нажатие кнопки «Закрыть» формы справки;

C4 – нажатие кнопки «Начать моделирование» формы главного меню;

C5 – нажатие кнопки «Закрыть» формы моделирования;

C6 – нажатие кнопки «Принять» формы моделирования;

C7 – нажатие кнопки «Начать» формы моделирования;

C8 – нажатие кнопки «Стоп» формы моделирования;

C9 – ввод данных;

C10 – нажатие кнопки «Выход» формы главного меню / завершение программы.

**2.4 Разработка диаграммы классов интерфейсной части концептуального уровня**

После анализа графа абстрактных диалогов и графа состояний интерфейса была построена диаграммаклассов интерфейсной части концептуального уровня с использованием готовых графических компонентов Qt (рисунок 9).

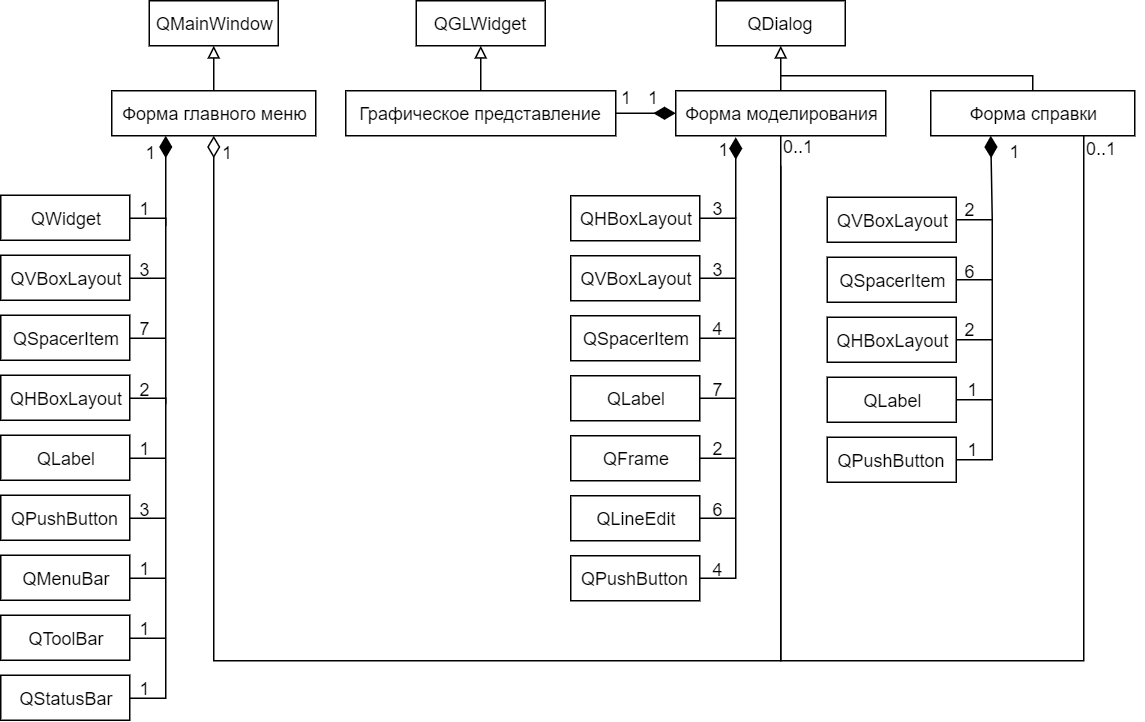


Рисунок 9 – Диаграммаклассов интерфейсной части концептуального уровня

Данная диаграмма уточняет отношения между интерфейсными классами.

Форма главного меню наследуется от класса QMainWindow, который представляет собой заготовку главного окна приложения. Она включает в себя ссылки на форму моделирования и форму справки, которые в свою очередь наследуются от класса QDialog. Эти формы могут создаваться и уничтожаться из главного меню в процессе работы программы, поэтому они связаны с формой главного меню отношением агрегации. Все формы содержат виджеты, используемые для создания графического интерфейса окон и ввода-вывода информации. Они включены в формы отношением композиции, поскольку создаются и уничтожаются только вместе с формами.

Класс графического представления процесса моделирования наследуется от класса QGLWidget, который реализует функции библиотеки OpenGL для вывода графики. Он также включен в форму моделирования отношением композиции по той же причине, что и остальные виджеты.

**2.5 Разработка экранных форм**

На основе диаграммы классов интерфейсной части концептуального уровня были разработаны экранные формы интерфейса.

Форма главного меню изображена на рисунке 10.

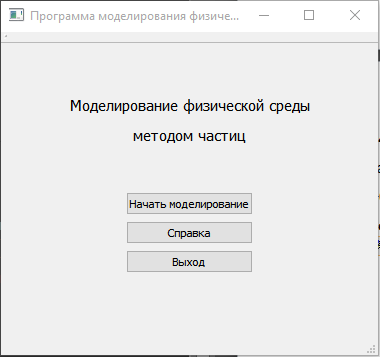


Рисунок 10 – Форма главного меню

На ней находится название программного продукта и 3 кнопки: «Начать моделирование», «Справка» и «Выход». При нажатии на кнопку «Начать моделирование» открывается форма моделирования. При нажатии на кнопку «Справка» открывается форма справки. При нажатии на кнопку «Выход» программа завершает работу.

Форма моделирования изображена на рисунке 11.

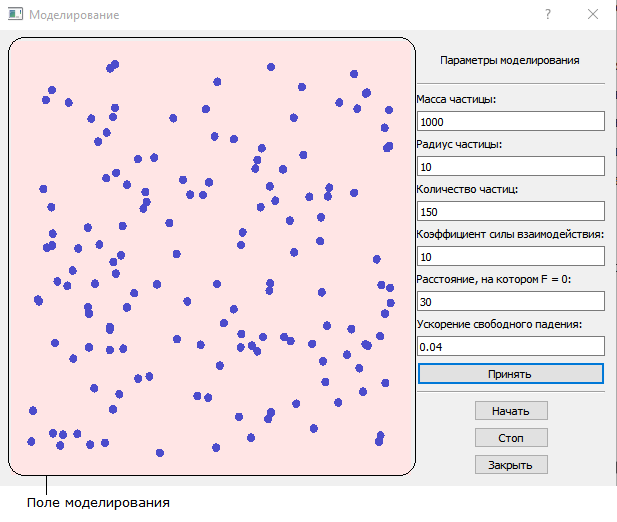


Рисунок 11 – Форма моделирования

На ней находятся следующие компоненты: поле моделирования, поля для ввода параметров моделирования и названия соответствующих параметров, кнопки «Принять», «Начать», «Стоп» и «Закрыть». При открытии формы в поля записываются значения параметров моделирования по умолчанию. Пользователь может ввести новые значения параметров и применить их, нажав на кнопку «Принять». При нажатии на кнопку «Начать» происходит запуск моделирования, а при нажатии на кнопку «Стоп» – остановка моделирования. Пользователь может закрыть форму нажатием кнопки «Закрыть».

Форма справки изображена на рисунке 12.

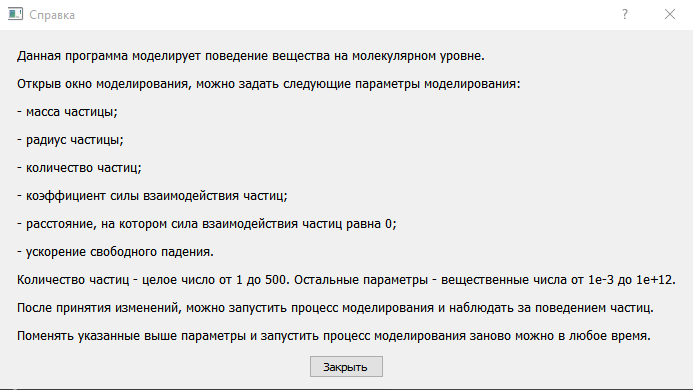


Рисунок 12 – Форма справки

На форме находится справочная информация о данной программе и кнопка «Закрыть», при нажатии на которую закрывается форма справки.

**3. Разработка компонентов программного продукта**

**3.1 Диаграмма классов предметной области концептуального уровня**

На основе концептуальной модели и объектной декомпозиции была построена диаграмма классов предметной области концептуального уровня (рисунок 13).

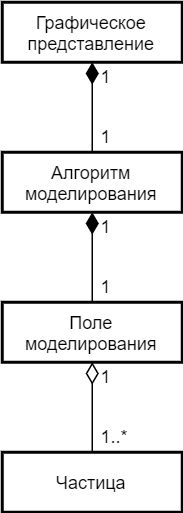


Рисунок 13 – Диаграмма классов предметной области концептуального уровня

В диаграмме классов концептуального уровня произошло уточнение отношений между классами.

Класс «Частица» представляет собой ключевую сущность программы. На уровне реализации он будет иметь параметры, характеризующие только свойства частицы.

Класс «Поле моделирования» представляет пространство, на котором отображается процесс моделирования. Этот класс связан с классом «Частица» отношением агрегации, поскольку объекты класса «Частица» могут создаваться и уничтожаться не только вместе с самим объектом «Поле моделирования», их количество на поле может меняться при изменении параметров моделирования.

Класс «Алгоритм моделирования» реализует математическую модель предметной области, в нем непосредственно происходит процесс моделирования. Данный класс связан с классом «Поле моделирования» отношением композиции, поскольку объект «Поле моделирования» создается и уничтожается только вместе с объектом «Алгоритм моделирования».

Класс «Графическое представление» визуально отображает результат процесса моделирования. Фреймворк Qt предоставляет наборы готовых классов для работы с графикой, поэтому данный класс будет реализован на одном из таких классов. Класс «Графическое представление» связан с классом «Алгоритм моделирования» отношением композиции, поскольку объект «Алгоритм моделирования» создается и уничтожается только вместе с объектом «Графическое представление».

**3.2 Разработка диаграмм классов программы уровня реализации**

После анализа диаграмм классов концептуального уровня стало ясно, что нельзя четко разграничить интерфейсную часть и предметную область. Визуализация графического представления процесса моделирования происходит через класс GLWidget, наследуемый от QGLWidget. Данный класс реализует функции библиотеки OpenGL и предназначен для вывода графики. Он является связующим между интерфейсом и предметной областью.

**3.2.1 Разработка диаграммы классов предметной области уровня реализации**

После уточнения полей и методов классов предметной области была построена диаграмма классов предметной области уровня реализации, изображенная на рисунке 14.

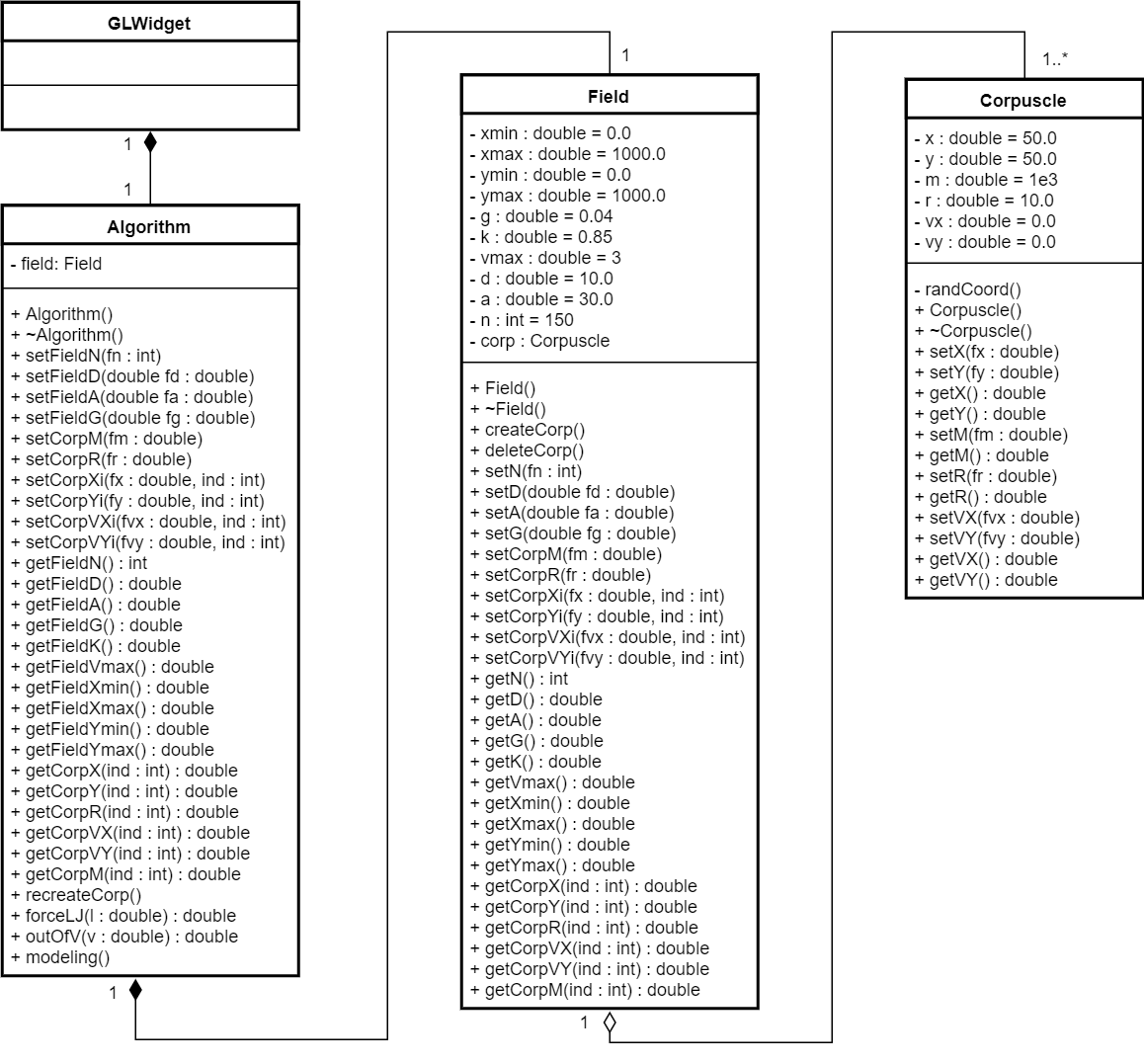


Рисунок 14 – Диаграмма классов предметной области уровня реализации

Класс ***Corpuscle*** представляет сущность «Частица».

Описание полей класса:

* координаты X и Y;
* масса;
* радиус;
* проекция скорости на оси X и Y.

Описание методов класса:

* конструктор и деструктор класса;
* методы для чтения и записи полей класса;
* метод, генерирующий частице случайные координаты.

Класс ***Field*** представляет сущность «Поле моделирования».

Описание полей класса:

* координаты границ поля;
* ускорение свободного падения;
* коэффициент гашения удара;
* максимально допустимую скорость;
* параметры d и a потенциала Леннард-Джонса;
* количество частиц на поле;
* массив частиц (объектов класса Corpuscle).

Описание методов класса:

* конструктор и деструктор класса;
* методы для создания и уничтожения массива частиц;
* методы для чтения и записи полей класса и полей объектов-частиц.

Класс ***Algorithm***. Класс представляет сущность «Алгоритм» и является программной реализацией математической модели. Единственным полем класса является объект класса Field, в котором происходит процесс моделирования. Описание методов класса:

* конструктор и деструктор класса;
* методы для чтения и записи полей класса, полей объекта «Поле моделирования» и полей объектов-частиц;
* метод для пересоздания массива частиц на поле моделирования;
* метод, вычисляющий силу взаимодействия между частицами, описываемую потенциалом Леннард-Джонса;
* метод проверки превышения частицей максимально допустимой скорости;
* метод, вычисляющий координаты и скорости частиц для следующей итерации в соответствии с математической моделью.

**4.3 Разработка диаграммы классов интерфейсной части уровня реализации**

После уточнения полей и методов интерфейсных классов и на основе спроектированных экранных форм была построена диаграммаклассов интерфейсной части уровня реализации, которую можно увидеть на рисунке 15.

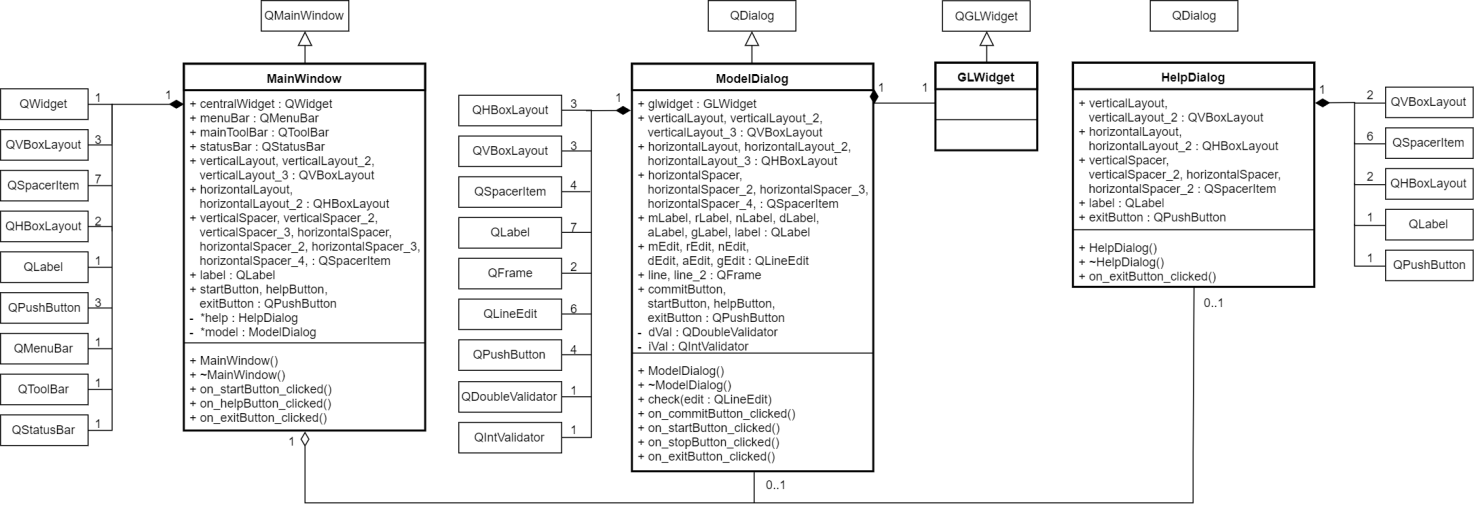


Рисунок 15 – Диаграммаклассов интерфейсной части уровня реализации

Класс MainWindow. Является реализацией формы главного меню. Помимо готовых графических компонентов, описание которых находится в пункте 2.5, полями данного класса являются класс ModelDialog, реализующий форму моделирования и класс HelpDialog, реализующий форму справки. Каждый класс содержит слоты для сигналов нажатия на кнопки в соответствующих формах.

Класс ModelDialog также содержит 2 объекта классов QDoubleValidator и QIntValidator, которые служат для задания формата ввода данных в поля параметров моделирования. Параметры моделирования имеют следующие границы допустимых значений: количество частиц – целое число от 1 до 500, остальные параметры – вещественные числа от 1e-3 до 1e+12. А метод check() проверяет введенные значения на предмет выхода их за допустимые границы.

**3.2.3 Разработка диаграммы класса графического представления**

Диаграмма класса графического представления GLWidget. Она представлена на рисунке 16.

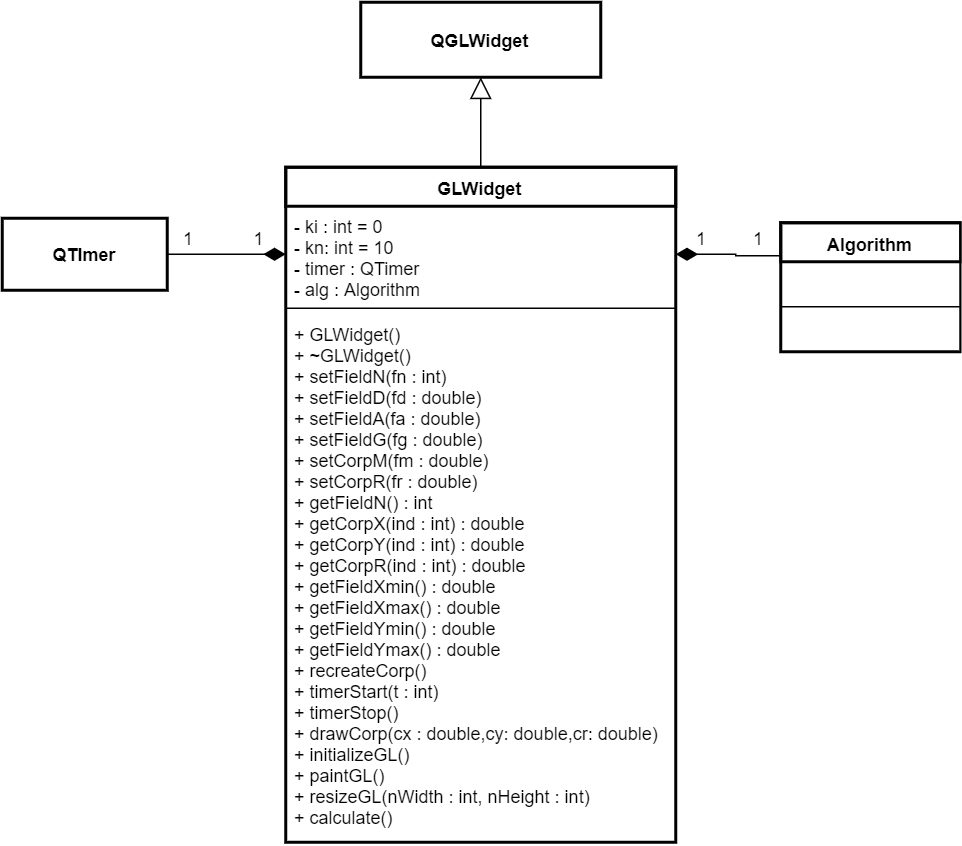


Рисунок 16 – Диаграмма класса GLWidget

Класс ***GLWidget***. Служит для визуализации процесса моделирования. По срабатыванию таймера происходит выполнение одной итерации моделирования, однако визуализация изображения происходит лишь через несколько итераций. Таким образом, при определенной частоте дискретизации картинка воспринимается непрерывно человеческим глазом, а процессор не нагружается избыточной отрисовкой изображения.

Описание полей класса:

* счетчик шагов дискретизации;
* частота дискретизации;
* объект-таймер;
* объект-алгоритм.

Описание методов класса:

* конструктор и деструктор;
* методы для чтения и записи полей класса, а также полей объекта-алгоритм, поле моделирования и частиц;
* метод для пересоздания массива частиц на поле моделирования;
* методы запуска и остановки таймера;
* метод, рисующий частицу;
* метод, выполняющий итерацию моделирования.

Также класс содержит переопределенные методы класса QOpenGL:

* initializeGL() – устанавливает начальные значения контекста рендеринга;
* paintGL() – рисует изображение;
* resizeGL(int nWidth, int nHeight) – отвечает за изменение размеров окна виджета.

**3.3 Схема компоновки программного продукта**

На основе включения внутренних файлов проекта была построена схема компоновки программы (рисунок 17).

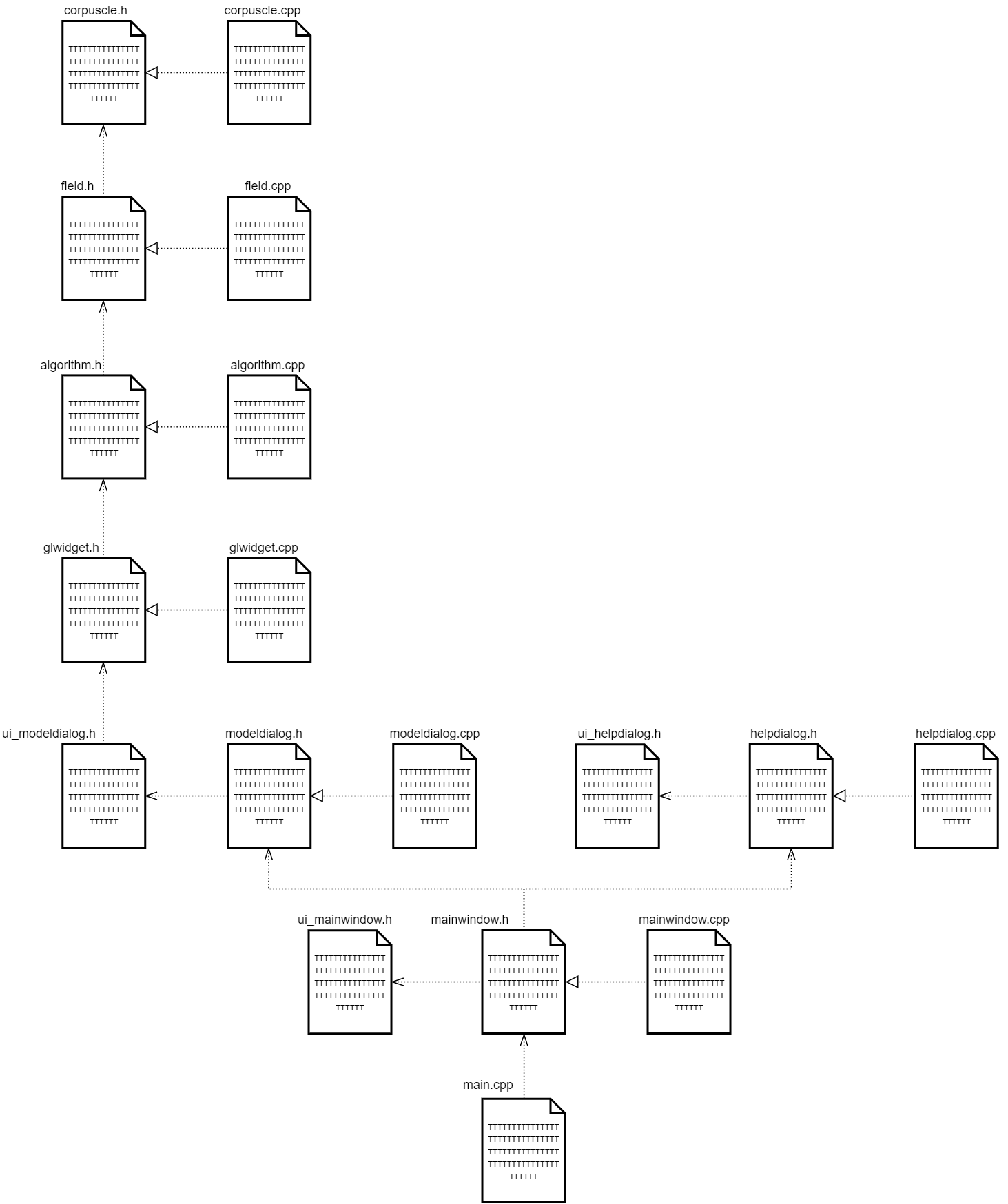


Рисунок 17 – Схема компоновки программного продукта

Программа состоит из 18 модулей: 7 заголовочных и исходных файлов соответствующих разработанных классов, 3 заголовочных файлов форм и файла main.cpp запускающего программу.

**4 Тестирование программного продукта**

**4.1 Выбор стратегии тестирования**

Существует два различных подхода к тестированию программного продукта: структурный и функциональный.

В основе структурного подхода лежит концепция максимально полного тестирования всех маршрутов программы. Однако даже в программе среднего уровня сложности число неповторяющихся маршрутов может быть очень велико, поэтому для данного программного продукта такой подход к тестированию является нерациональным.

Функциональный подход основывается на том, что структура программы не известна («черный ящик»). В этом случае тесты строят, опираясь на функциональные спецификации. При тестировании программного продукта необходимо проверить, насколько верно программа реализует заявленные в техническом задании функции.

**4.2 Функциональное тестирование**

Рационально осуществить тестирование программы по принципу «чёрного ящика», т.е. организовав функциональное тестирование. Рассмотрим подробнее 2 метода функционального тестирования.

Метод эквивалентного разбиения предполагает разбиение всех возможных наборов входных данных программы на конечное число групп – классов эквивалентности. Причем каждую группу строят по принципу обнаружения одних и тех же ошибок. Данное разбиение является эвристическим процессом, однако рекомендуется выделять в отдельные классы эквивалентности наборы, содержащие допустимые и недопустимые значения некоторого параметра.

Все значения входных данных в программе представляют собой числа. Для данного метода можно выделить два класса эквивалентности для значений входных переменных: числа, и не числа. В таблице 5 показаны результаты тестирования методом эквивалентного разбиения.

Таблица 5 – Результаты тестирования методом эквивалентного разбиения

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Ожидаемый результат** | **Результат** | **Вывод** |
| Ввод в любое поле числового значения | Ввод числового значения | Ввод числового значения | Успешно |
| Ввод в любое поле нечислового значения | Ошибка ввода | Игнорирование вводимых данных | Успешно |

Метод анализа граничных значений предполагает обнаружение ошибок на границах классов эквивалентности входных данных. Для данного метода можно выделить два класса эквивалентности для значений входных переменных: целые числа (от 1 до 500) и вещественные числа (от 1e-3 до 1e+12). В таблице 6 показаны результаты тестирования методом анализа граничных значений.

Таблица 6 – Результаты тестирования методом анализа граничных условий

|  |  |  |  |
| --- | --- | --- | --- |
| **Тест** | **Ожидаемое значение** | **Результат** | **Вывод** |
| Ввод в поле «Количество частиц» значения 1 | 1 | 1 | Успешно |
| Ввод в поле «Количество частиц» значения 0 | 1 | 1 | Успешно |
| Ввод в поле «Количество частиц» значения 500 | 500 | 500 | Успешно |
| Ввод в поле «Количество частиц» значения 501 | 500 | 500 | Успешно |

Продолжение таблицы 6

|  |  |  |  |
| --- | --- | --- | --- |
| Ввод в поле любое поле, кроме поля «Количество частиц», значения 0.001 | 0.001 | 0.001 | Успешно |
| Ввод в поле любое поле, кроме поля «Количество частиц», значения 0.0009 | 0.001 | 0.001 | Успешно |
| Ввод в поле любое поле, кроме поля «Количество частиц», значения 1000000000000 | 1000000000000 | 1000000000000 | Успешно |
| Ввод в поле любое поле, кроме поля «Количество частиц», значения 1000000000000.1 | 1000000000000 | 1000000000000 | Успешно |

**4.3 Оценочное тестирование**

Была выполнена проверка удобства эксплуатации на уровне интерфейса на выборке экспертов в количестве десяти студентов группы ИУ6-53. Было выяснено визуальное неудобство розового фона поля моделирования.

**Заключение**

В результате разработки была спроектирована и реализована программа моделирования физической среды методом частиц. Заявленная функциональность реализована в полной мере в соответствии с требованиями в техническом задании.

Данная программа может быть использована для изучения процессов межмолекулярного взаимодействия в различных веществах.

В перспективе планируется расширение и усложнение алгоритма, описывающего взаимодействие частиц, а также количественных параметров результатов моделирования в текстовом виде.

**Список использованных источников**

1. Кривцов А.М. Метод частиц и его использование в механике деформируемого твердого тела  / Кривцов А.М., Кривцова Н.В. // Дальневосточный математический журнал ДВО РАН. – 2002. – Т. 3, № 2. – с. 254-276.
2. Холмуродов Х.Т. Методы молекулярной динамики для моделирования физических и биологических процессов / Холмуродов Х.Т., Алтайский М.В., Пузынин И.В., Дардин Т., Филатов Ф.П. // Физика элементарных частиц и атомного ядра. – 2003. – Т. 34, № 2. – с. 474-510.
3. Иванова Г.С. Технология программирования: учебник / Г.С. Иванова. – 3-е изд., стер. – М.: КНОРУС, 2016. – 334 с. – (Бакалавриат). – ISBN 978-5-04734-7
4. Иванова Г.С. Методические указания по выполнению курсовой работы по дисциплине «Технология разработки программных систем» / Иванова Г.С., Ничушкина Т.Н., Пугачев Е.К., Самарев Р.С. – Москва, 2013. – 23 с.
5. Макс Шлее. Qt 5.3. Профессиональное программирование на C++ / Макс Шлее – Санкт-Петербург, 2015. – 928 с. – ISBN 978-5-9775-3346-1.