Содержание

Введение

Тут должен быть краткий обзор предметной области (о моделировании (краткая классификация), мы выбираем имитационное моделирование, мелкозернистый параллелизм, его использования для моделирования) Цель, актуальность, план следующего изложения.

Необходимость постановки опыта (или эксперимента) над системой (под системой понимается совокупность объектов, функционирующих и взаимодействуюищих друг с другом для достижения определённой цели – в [2] написано, что такое определение даётся в [Шмидт, Тэйлор, 1970]) возникает постоянно и в самых различных в сферах деятельности, будь то физические процессы, экономические системы и т.д. Однако, во-первых, произвести эксперимент над реальной системой не всегда представляется возможным в силу того, что системы может либо не существовать, либо в результате эксперимента будет невозможным её возвращение в исходное состояние, а, во-вторых, эксперимент над реально существующей системой может потребовать слишком больших затрат на его осуществление. В таких случаях возможно прибегнуть к моделированию: построить модель – упрощённый вариант системы (что-нить о том, что этот вариант должен обладать требуемым свойствами) – и провести эксперимент уже над моделью. На этапе построения модели необходимо сделать выбор между физической моделью (создание упрощённого, но реального существующего варианта системы для проведение эксперимента) и математической (описание системы посредством отношений, которые определяют как система будет реагировать на изменения, если бы она существовала). Математическая модель может иметь точное аналитическое решение, дающее представление о том, как входные параметры влияют на систему в соответствие с построенной моделью. Однако описываемые модель отношения могут не дать простого аналитического решения и потребовать огромных вычислительных ресурсов, что сделает невозможным аналитическое моделирование или потребует большего упрощения системы. Альтернативный вариант в таком случае: изучение системы с помощью имитационного моделирования, то есть многократного испытания модели с нужными входными данным. Имитационные модели за последнее время получили широкое распространение благодаря появлению инструментов, упрощающих создание компьютерных программ, и увеличению и удешвлению компьютерных мощностей [1]. [2, стр. 23-34] сказать, что моделирование может быть физическим, аналитическим, но мы будем рассматривать имитационное Однако, моделирование невозможно без инструмента визуализации, позволяющего изучать модель и исследовать результаты смоделированного эксперимента: работа с объектами на данных микроуровне чрезвычайно сложна, т. к. он представляет из себя огромный массив ячеек с частицами, в то время как визуализация позволяет получить качественную картину ???. Целью данной работы является разработка подсистемы пользовательского интерфейса и визуализации данных среды имитационного моделирования, назначение которой – построение и исследование клеточно-автоматных моделей, широко применяющихся для исследования явлений естественных наук. К си-

стеме предъявляются требования,

Цель работы — обеспечить исследователя *МЗП* моделей физических процессов инструментом для построения и отладки таких моделей и визуализации данных в них. Хотя система должна обладать достаточной универсальностью и быть пригодной для широкого спектра таких моделей, в первую очередь рассматривается визуализация данных для клеточно-автоматных моделей газов. Актуальность. Рассматриваемый класс моделей характерен сложностью происходящих в них преобразований данных и большим объемом данных. Под большим объемом данных подразумевается, что он существенно превосходит размеры основной памяти. Сложность преобразований данных в моделях означает необходимость: 1) отладки модели в процессе ее создания и 2) создания режимов отображения, позволяющих исследователю увидеть как точную количественную картину некоторого фрагмента состояния модели, так и качественную картину протекающих в модели процессов. Если исследователь выбирает путь построения собственной программы визуализации, он неизбежно сталкивается с необходимостью реализации системных функций, сложность которых значительно выше, чем само описание модели. Таком образом, для эффективной работы исследователя необходим инструмент для построения, отладки и исследования моделей.

Далее в работе будет описан процесс разработки системы по этапам. В первую очередь были предъявлены требования к разрабатываемой системе, главными из которых являются: адекватность предметной области и открытость, в разделе «2 Формирование требований» будет подробно раскрыто каждое из требований, обусловлено наличие их предъявление к создаваемой системе. В разделе «3 Обзор существующих систем» представлен краткий обзор уже имеющихся систем визуализации в исследуемой предметной области, исследование их на соответсвие сформулированным ранее требованиям, сравнение по требуемым параметрам. В разделе «4 ???» разобрана архитектура разработанной системы, детально описан каждый компонент системы: тутнуженглагол цель создания данного модуля системы, какие из поставленных задач он решает и каким требованиями удовлетворяет, список основных функций, которые он реализовывает. Надо ли писать про заключение?

1 Формирование требований

Назначение системы — визуализация данных в МЗП моделях физических процессов. Наиболее существенное требование, требование адекватности системы проблемной области, заключается в создании удобного инструмента, содержащего все необходимые исследователю функции для изучения широкого спектра таких моделей.

Моделирования физических явлений с помощью моделей с мелкозернистым параллелизмом — динамично развивающаяся область, в которой постоянно возникают новые классы моделей. Для того, чтобы система могла применяться для них, она должна иметь открытую архитектуру, позволяя пополнять свой набор функций не только разработчикам, но и пользователям системы.

Для МЗП моделей физических явлений характерно то, что адекватность результатов достигается только при очень больших размерах объектов: во-первых, не все являения могут быть заметны на моделях маленького размера, во-вторых, за счёт увеличения размера объекта достигается более точный результат.

Таким образом, основными требованиями, предъявляемыми к системе, являются:

- адекватность проблемной области;
- открытая архитектуа;
- обработка больших объёмов данных;

Рассмотрим более подробно, что включает в себя каждое из приведённых требований.

1.1 Адекватность проблемной области

Адекватность системы предметной области означает удовлетворение системой требованиям, которые предъявляет к ней исследователь в данной предметной области: так как система создаётся для работы с клеточно-автоматными моделями, то она должна реализовывать необходимые функции, требующиеся при изучении данной обалсти, а именно:

- построение моделей;
- управление проектами;
- исполнение модели;
- изучение модели;
- отладка модели.

1.1.1 Построение моделей

С точки зрения системы модель представляет из себя непосредственно свойства модели как физического объекта (т. е. все свойства модели: например, для моделей клеточных автоматов это будут правила, по которым которым осуществляются переходы в ячейках, массы частиц покоя и количество движущихся частиц) и объекты данных, над которым производятся вычисления. Объектом данных является структура, хранящая в себе состояние моделируемого объекта. Такая структура может быть как и многомерный массив так и нерегулярная структура данных.

Исследователю необходимо иметь возможность формировать и корректировать параметры модели и создавать и редактировать объекты данных. Для работы с моделью удобно наличие визуальной среды: графический пользовательский интерфейс упрощает работу с множеством параметров модели и с большими объектами данных.

Свойства модели могут быть значениями произвольного содержания (число, строка, массив значений и т. п.). Объект данных может быть как небольшого (обычно такие объекты используются при отладке модели), так и огромного размера. И в том, и в другом случае бывает необходимость редактировать и значения отдельных ячеек, и сразу большой области объекта данных.

1.1.2 Управление проектами

В процессе исследования возникает необходимость скорректировать параметры изучаемой модели, посмотреть на изменение поведения, сравнить данную модель с исходной по каким-либо характеристикам. В таких случаях удобно объединять две и более модели в один проект с возможностью сохранения и загрузки проекта.

Проект можно определить как группу логически связанных между собой моделей, при этом определение модели расширяется, в проекте с моделью так же ассоциируются пользовательские настройки (например, сохранение настроек режимов отображения для данной модели). пользовательские настройки, набор операций. В проект можно добавлять новые модели, удалять из него старые. Должна быть поддержка сохранения открытого проекта с возможностью последующей загрузки.

1.1.3 Исполнение модели

Одной из основных функций является счёт модели: это включает в себя передачу параметров модели и объекта данных на вычислитель, запуск модели на фиксированное число итераций, ожидание завершения счёта и последующее получение результирующих данных с вычислителя.

Под вычислителем понимается устройство, на котором происходит моделирование. Вычислителем может выступать такое устройство локального или удалённого компьютера, как центральный или графический процессор. Вычислитель при этом необязательно является однопроцессерной машиной: в его роли могут так же выступать

мультипроцессорные системы, системы с распределённой памятью, кластеры и ГРИД-системы.

1.1.4 Изучение модели

При работе с моделями пользователь системы нуждается в работате с объектом данных модели не только на микроуровне, но и с качественной картиной: иметь возможность выделить из всего объекта данных какую-либо конкретную характеристику. Так же пользователю необходимо дать возможность взаимодействовать с системой, быть не только наблюдетелем за процессом моделирования, но и иметь средства как-то влиять на него, т. е. обеспечить интерактивность.

Качественная картина может быть предоставлена за счёт поддержки различных режимов отображения (режимом отображения является функция, отображающая объект данных в изображение, которое впоследствии может быть выведено на экран монитора): например, это может быть режим осреднённых плотностей или профиль волны для исследования её поведения. Полезными могут быть не только режимы, визуализирующие какую-либо конкретную характеристику (плотность частиц, направление потоков), но и режимы, формирующие целостную картину: например, трёхмерный режим с возможностью масштабирования и вращения для просмотра объекта данных с разных ракурсов.

1.1.5 Отладка модели

При построении больших и сложных моделей могут возникать ошибки в их описании. В таких случаях модель приходится отлаживать: использовать какие-либо средства с целью выявления допущенных ошибок. Система в свою очередь должна помочь исследователю, предоставив такие средства.

В процессе отладки востребован не только запуск счёта модели на фиксированное, большое количество итераций, но и запуск модели в режиме пошагового исполнения: формирование модели и отправка её на вычислитель, запуск на небольшое количество шагов, получение результирующих данных с возможностью продолжить вычисления с того момента, где они были приостановлены. Пошаговое исполнение позволяет детальнее изучить поведение модели.

Так же при отладке востребованной может оказаться возможность модификации текущего объекта данных, передача изменений на вычислитель и продолжение счёта.

1.2 Открытая архитектура

Открытость архитектуры облегчает сопровождение системы и её адаптация к изменяющимся требованиям.

Свойство открытости включает в себя [2]:

- расширяемость;

- переносимость;
- интероперабельность;
- дружественность интерфейса.

1.2.1 Расширяемость

Пользователи уже используют свои форматы, они от них не уйдут, мы сами не может предоставить поддержку всего-всего, потому предоставляем средства для включения поддержки Система должна быть готова к изменениям, связанным с включением поддержки новых классов моделей, ранее не предусмотренных системой. Это обусловлено тем, что требуется создание универсального инструмента, реализующего не только необходимый на данный момент функционал, но и способного в будущем адаптироваться под новые требования, предоставляющего механизм для встраивания нового функционала.

Направлениями для расширения системы являются:

- режимы отображения;
- поддержка графических библиотек;
- коммуникация с подсистемой моделирования.

расписать направление расширяемости по пунктам Помимо поддержки новых форматов данных и новых классов моделей, так же должна быть возможность встраивать поддержку новых графических библиотек (например, отрисовку трёхмерных изображений на ОС Windows делать не с помощью OpenGL, а с использованием библиотеки Direct3D). Другим направлением расширением системы может быть добавление поддрежки альтернативных способов коммуникации с вычислителем: например, когда в качестве вычислителя вместо центрального процессора выступает видеокарта, то обмен данными уже невозможно организовать через общую память и потребуется реализовать передачу данных между оперативной памятью компьютера (к которой система визуализации имеет прямой доступ) и памятью графического ускорителя.

Всё это требует от системы масштабируемости по архитектуре: возможность вносить изменения, связанные с добавлением нового функционала, при этом минимизируя влияние (т.е. либо не требуя модификации, либо модификация существующей реализации должна быть незначительной) на другие, уже существующие компоненты системы.

1.2.2 Переносимость

С течением времени архитектуры процессоров и операционные системы меняются. Создание системы, привязанной к одной конкретной платформе, может в будущем потребовать значительных усилий для портирования системы под другую платформу

или вовсе привести к прекращению поддержки системы, когда используемая платформа устареет.

Второй проблемой при разработке платформозависимой системы является проблема взаимодействия с приложениями, привязанными к другим платформам. Это потребует от пользователей либо портировать их приложения на используемую системой платформу, либо отказаться от использования системы.

Таким образом, от системы требуется свойство переносимости — отсутствие зависимостей модулей системы от конкретной платформы. В системе допускается привязка отдельных реализаций некоторых компонентов к определённым платформам, но в целом она должна оставаться платформонезависимой.

Переносимость должна быть обеспечена на уровне исходного кода (код прилоежния не должен использовать платформозависимые библиотеки или функции или же функциональность, достигаемая с использованием данных библиотек, должна быть реализована для каждой платформы) и на уровне данных (работа с созданными проектами и моделями на одной платофрме должна быть возможна и на других платформах) [3].

1.2.3 Интероперабельность

Свойство интероперабельности означает способность системы взаимодействовать с окружением, в котором она запущена. Для подсистемы визуализации это, в первую очередь, способоность взаимодействовать с подсистемой моделирования.

Взаимодействие основывается на использовании общих форматов хранения данных, интерфейсов для взаимодействия на локальных системах или пртоколв для взаимодействия в распределённых средах. Благодаря интероперабельности система визуализации может использоваться как звено в цепи обработки информации, поулчая входыне данные от подсистемы моделирования, отрисовывая полученные данные и передвавая полученные изображения другим приложениям для последующей обработки.

1.2.4 Дружественность интерфейса

Интерфейс системы можно разделить на интерфейс пользователя и программный интерфейс [3].

Дружественность пользовательского интерфейса заключается в наличии интуитивнопонятного и комфортного взаимодействия системы с пользователем. Требование является необходиым для любой отчуждаемой от разработчиков программы, в которой присутствует функция взимодействия с пользователем.

Дружественность программного интерфейса относится к разработчикам системы (как и к тем, кто изначально разрабывает систему, так и к пользователям, которые будут расширять систему за счёт встраивания своих модулей) и означает упрощение разработки системы: интерфейсы, дающие возможности для расширения системы, должны требовать реализацию только минимального необходимого набора методов.

1.3 Обработка больших объёмов данных

Требование к поддержки больших объёмов данных обусловлено особенностью моделей физических явлений:

1. Многие явление могут быть незаметны на моделях маленького размера (например, вихри — они проявляются только на объектах больших размеров).

2. Как правило, при увеличиении размера объекта данных, при более сильном дроблении, качество моделирования повышается и результат получается более точным.

Под возможностью обработки больших объёмов данных понимается способность системы визуализировать объекты данных, размер которых превышают размер оперативной памяти компьютера.

2 Обзор существующих систем

Имитационное моделирование, а значит и необходимость в средствах визуализации, существует уже достаточно давно, потому следующим этапом после формирования требований был обзор уже существующих систем.

Далее приведён обзор нескольких, наиболее подоходящих под предъявляемые требования, систем. Полный список обозреваемых систем и их соответствие требованиям привдён в Приложении А.

2.1 Mirek's Celebration

Система моделирования 1D и 2D клеточно-автоматных моделей. Имееет удобный, интуитивно понятный пользовательский интерфейс, хорошо документирована.

Таблица 1: Соответствие требованиям системы Mirek's Celebration

Адекватность пред. области	+	
Расширяемость	∓	Есть возможности встраивания DLL. Возможности расширения всё-таки ограничены.
Интероперабельность	??	
Платформонезависимость	–	Win32
Дружественный интерфейс	+	
Большие объёмы данных	–	

2.2 WinALT

Система моделирования разнообразных клеточно-автоматных моделей.

Таблица 2: Соответствие требованиям системы WinALT

Адекватность пред. области	–	Система рассчитана на проектирование МЗП алгоритмов и структур.
Расширяемость	+	
Интероперабельность	+	
Платформонезависимость	??	Win32
Дружественный интерфейс	+	
Большие объёмы данных	–	

2.3 Ansys Fluent

Система моделирования широкого спектра физических процессов, в том числе и газовой динамики, обладает богатыми возможностями моделирования и визуализации.

Таблица 3: Соответствие требованиям системы Ansys Fluent

Адекватность пред. области	–	Дифференциальные уравнения решаются классическими методами.
Расширяемость	+	Возможность встраивания пользовательских функций.
Интероперабельность	+	
Платформонезависимость	??	
Дружественный интерфейс	+	
Большие объёмы данных	??	

2.4 Выводы

Таким образом, ни одна из рассмотренных систем в полной мере не удовлетворяет всем предъявленным к ней требованиям и было решено создать собственную реализацию системы визуализации. На рисунке 1 графически изображена безысходность ситуации: есть множество систем, удовлетворяющих одному-двум свойствам, однако нет систему, удовлетворяющих одновременно всем предъявляемым требованиям.

Рис. 1: Существующие системы визуализации

3 Архитектура системы

На верхнем уровне архитектура системы представляет из себя четрые взаимодействующих друг с другом модуля:

- модуль взаимодействия с подсистемой моделирования;
- модуль поддержки графических библиотек;
- модуль поддержки различных режимов отображения;
- пользовательский интерфейс.

Схематически модульная архитектура представлена на Рис. 2 — система визуализации, разбитая на модули, а так же подсистема моделирования, взаимодействие с которой происходит через один из модулей.

Модульное программирование. ООП.

Рис. 2: Архитектура подсистемы визуализации

Такая модульная/откртая архитектура системы, разбивающая систему на отдельные, взимодействиующие между собой части, реализаций которых одна от другой никак не зависят, позволяет вносить изменения (как модификация существующей реализации, так и расширение за счёт *чего-то нового*) в отдельные компонеты прозрачно для остальных.

3.1 Модуль взаимодействия с подсистемой моделирования

Для обеспечения коммуникации подсистемы визуализации с подсистемой моделирования был создан интерфейс для подключения вычислителей.

В основные функции, решаемые данным модулем, входит запуск процесса моделирования в синхронном (запуск на счёт фиксированного количества итераций, ожидание завершения моделирования и получение результата) и в асинхронном (запуск на счёт

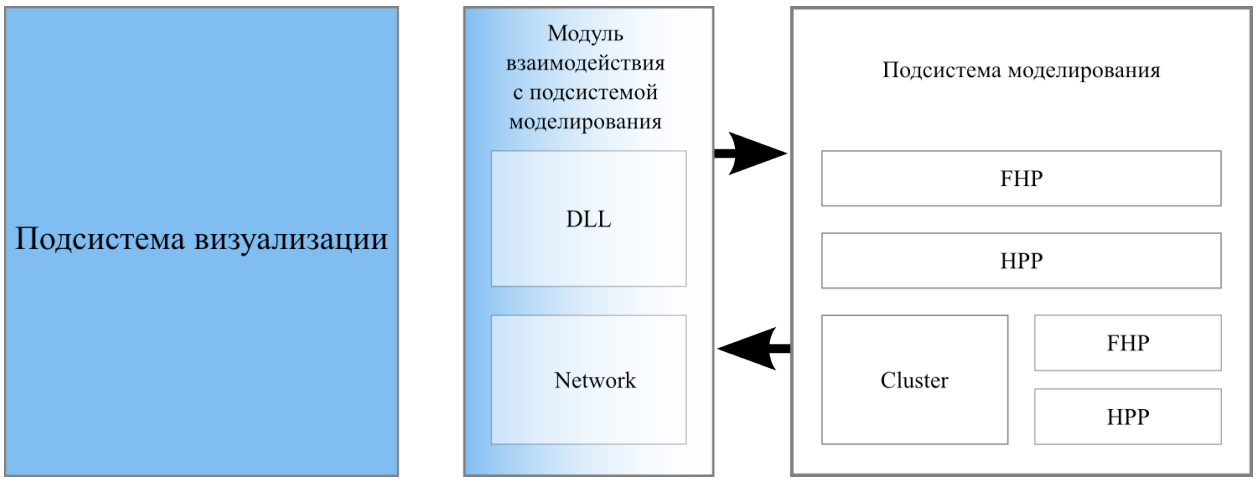


Рис. 3: Взаимодействие с подсистемой моделирования

с возможностью подключения к вычислителю в любой момент и получения текущего состояния *чего???*) режимах и реализация протокола передачи объекта данных визуализатору с вычислителя.

Класс возможных вычислителей, с которыми может коммуницировать подсистема визуализации, программно ничем не ограничен. Таким образом, в качестве вычислителя может выступать *абсолютно любое устройство*: как процессор или видеокарта локальной машины, так и удалённый кластер, который, возможно, требует подключения по специфичному??? протоколу (вполне стандартна ситуация, когда доступ к кластеру осуществляется по SSH с необходимость авторизации).

Подключение нового вычислителя осуществляется написанием реализации созданного интерфейса. *Может, ссылку на приложение, которое создать для каждого модуля, где будет кратко описано, как создавать реализацию нового модуля?*. На данный момент реализована поддержка разделяемых библиотек для запуска счёта на локальной машине и в процессе разработки поддержка запуска счёта на удалённой машине, в том числе и для моделирования на кластере.

Т.о. система удовлетворяет свойству интероперабельности. <- *надо куда-то красивее вписать*.

3.2 Модуль поддержки графических библиотек

Рис. 4: Модуль поддержки графических библиотек

Посколько основной задачей инструмента визуализации является создание изображений из объекту данных, то в состав системы так же был включён модуль, отвечающий за поддержку различных графических библиотек. На данный момент существует множество библиотек, как для работы с двухмерной графикой (GTK, Qt, wxWidgets и многие другие) и с трёхмерной графикой (OpenGL, Direct3D и т.д.) и основной задачей модуля является предоставление общего интерфейса для рисования независимо от того, какая из библиотек используется в данный момент для формирования итогового изображения.

Модуль поддрежки графических библиотек, как и модуль взаимодействия с подсистемой моделирования, представлет из себя интерфейс, реализацией которого осуществляет подключения новых библиотек. В интерфейс входит рисование графических 2D и 3D примитивов (точки, линии, закрашивание областей), рисование более сложных объектов (различного рода кривые, объединение и пересечение объектов), поддержка прозрачности, если таковая есть в библиотеки. Иными словами, задачей интерфейса является предоставление универсального доступа к возможностям библиотек, скрыв при этом различия при работе с ними для других модулей.

За счёт введения интерфейса достигается расширяемость, а отсутствие привязанности интерфейса к каким-либо конкретным технологиям делает систему переносимой. Стоит отметить, что некоторые графические библиотеки являются платформозависимыми (например, возожность использовать Direct3D имеется только под операционной системой Microsoft Windows), однако, в целом это не делает систему непереносимой.

На данный момент используются следующие кроссплатформенные библиотеки:

- Модули QtGUI и QtWidgets из библиотеки Qt.
- Библиотека OpenGL для отрисовки 3D изображений.

3.3 Модуль поддержки режимов отображения

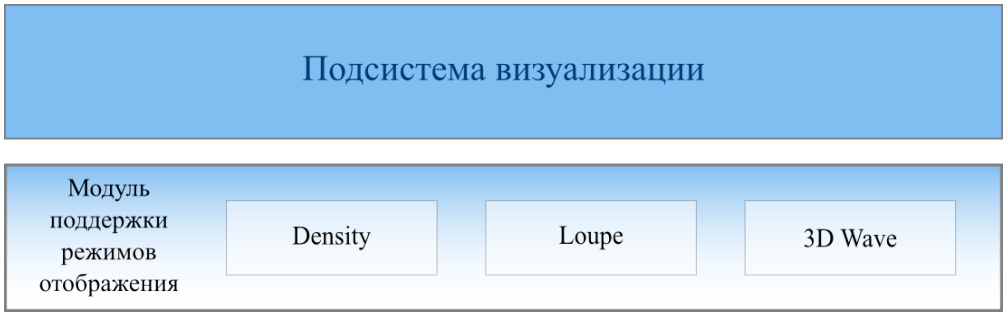


Рис. 5: Модуль поддержки графических библиотек

В разделе 1.1 одним из требований, предъявлемых к системе, является предоставление исследователю как возможность наблюдать за изменениями отдельных различных характеристик, так и возможность увидеть целостную картину. Данный модуль реа-

лизует поддержку различных режимов отображения и предоставляет возможности для встраивания новых режимов.

Единственная функция, которую реализует модуль поддержки режимов отображения, это отображение части или всего объекта данных в двухмерное или трёхмерное изображение. При отрисовывании изображения данный модуль использует интерфейс, предоставляемый модулем поддержки графических библиотек, а работа с объектом данных модели происходит через модуль взаимодействия с подсистемой моделирования.

С помощью данного модуля достигается расширяемость системы: появляется возможность встраивать новые режимы отображения, при этом не затрагивая обмен данных с вычислительной подсистемой и не внося изменения в существующие форматы данных.

Реализации поддержки режима отображения включает в себя реализацию двух интерфейсов, отвечающих за рисование изобораження и за пользовательский интерфейс с настройками режима отображения. Подробнее о поддеркже режимов отображения написано в Приложении В. Для поддержки различных режимов отображения был так же разработан интерфейс для их подключения, что делает систему расширяемой.

На данный момент релизованы:

- Режим среза для изучения профиля волны.
- Режим осреднения (статистически-обобщённый).
- Режим лупы, позволяющий рабоатть с объектами данных на микроуровне.
- 3D режим для получения полноценной картины.

3.4 Модуль пользовательского интерфейса

Для взаимодействия с пользователем системы был реализован графический интерфейс. Задачей модуля было предоставить пользователю простой, но в тоже время функиональный, удобный, интуитивно-понятный интерфейс. Графический интерфейс направлен на удовлетворение свойству адекватности предметной области: через него пользователь осуществляет работу с проектами и моделями, запуск и наблюдение за процессом моделирования, получение и исследование результатов моделирования, наличие средств для отладки модели.

Пользовательский интерфейс представляем из себя MDI (multiple document interface — многодокументный интерфейс) окно, внутри которого расположены панели управления, dockable-окна с различными настройками и окна моделей, на которых визуализируются объекты данных. Пример запущенного приложения с открытыми окнами модели представлен на рисунке 6. Пример сценария работы с моделью представлен в Приложении Г.

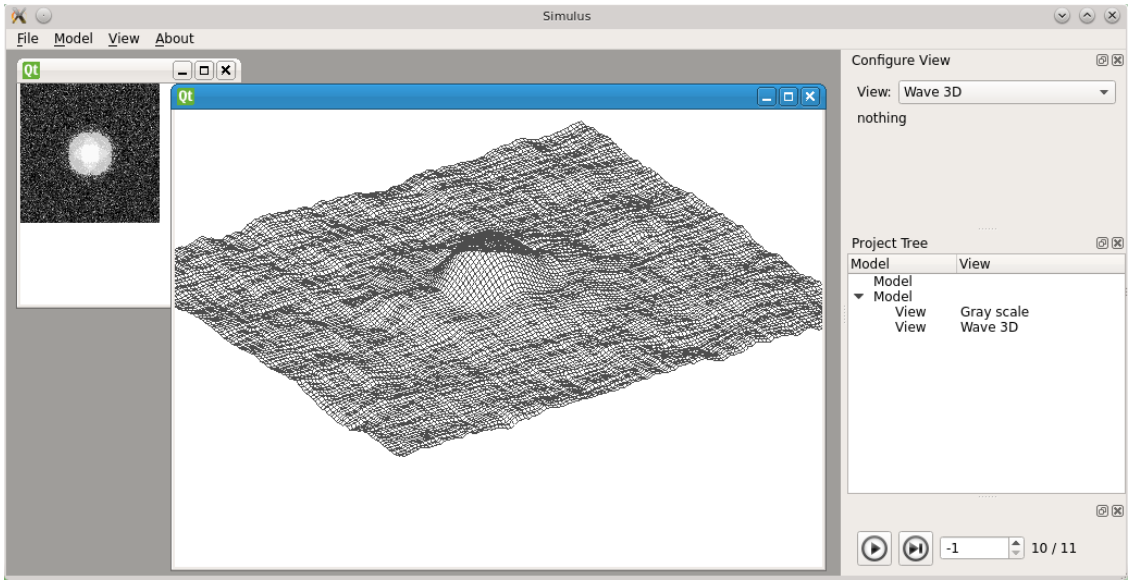


Рис. 6: Пользовательский интерфейс подсистемы визуализации

Заключение

В результате работы была разработана архитектура подсистемы визуализации системы имитационного моделирования, на основе разработанной архитектуры была написана реализация системы. Был разработан и опробован на чём? механизм взаимодействия с моделью. Разработан механизм встраивания режимов визуализации, на базе которого построено несколько режимов каких? Разработана графическая оболочка системы. В планах реализация механизма удалённого исполнения модельных программ, запуск счёта модели на кластере ССКЦ.

Приложение А. Сравнение систем визуализации

(Обязательное)

Таблица 4: Соответствие систем визулизации предъявленным требованиям

#	**Система**	**А**[1]	**Р**[2]	**И**[3]	**П**[4]	**Д**[5]	**Б**[6]
1	Mirek's Celebration	+	∓				
2	WinALT	+	+				

1) Адекватность предметной области
2) Расширяемость
3) Интерактивность
4) Переносимость
5) Дружественность пользовательского интерфейса
6) Поддержка больших объёмов данных

Приложение Б. Реализация модуля взимодействия с подсистемой моделирования

(Рекомендуемое)

Одной из основных функций системы является коммуникация с подсистемой моделирования. В основном этом передача модели (свойства модели и объект данных) и управляющие команды (запуск, остановка, получение информации о состоянии процесса моделирования и т. п.).

Как было сказано в разделе 3.1, добавление поддержки вычислителя возможно за счёт реализации интерфейса, код которого приведён в листинге 1.

Листинг 1: Интерфейс подключения вычислителя

```
1  class Config : public QObject
2  {
3      Q_OBJECT
4
5      Config(const Config& );
6      Config& operator=(const Config& );
7
8  protected:
9
10     int current_iteration_id;
11
12     Config();
13
14     void preSerialize  (QDataStream& stream);
15     void preDeserialize(QDataStream& stream);
16
17 public:
18
19     virtual ~Config();
20
21     virtual int nextIteration();
22     virtual int setIteration(int iteration) = 0;
23     virtual int getIterationsCount();
24
25     virtual void* getData(void* data_type = NULL) = 0;
26     virtual int getDimSize(int dim) const = 0;
27     virtual int getDimSizeX() const;
28     virtual int getDimSizeY() const;
29     virtual int getDimSizeZ() const;
30
31     virtual void serialize  (QDataStream& );
32     virtual void deserialize(QDataStream& );
33 };
```

Интерфейс представляет из себя класс с набором виртуальных методов, который можно логически разделить на 3 части:

1. управление счётом (запуск, остановка);
2. работа с объектом данных;
3. сохранение и загрузка модели.

Бóльшая часть виртуальных уже имеет простую реализацию и для добавления поддеркжи нового вычислителя достаточно реализовать всего три метода. За что отвечает каждый из методов и какие из них необходимо реализовывать будем рассмотрено далее.

Б.1 Управление счётом

Запуск и остановка процесса моделирования происходит с использованием следующих функций:

```
21      virtual int nextIteration();
22      virtual int setIteration(int iteration) = 0;
23      virtual int getIterationsCount();
```

Основным является метод setIteration(int iteration) и требует определения при реализации интерфейса. Вызов данного метода означает запуск счёта от последней посчитанной итерации до итерации номер iteration.

Метод nextIteration() запускает счёт на одну итерацию. В реализации по умолчанию он содержит внутри вызов setIteration от номера текущей итерации плюс 1 и позволяет упростить написание и чтение кода.

getIterationsCount() возвращает количество посчитанных итераций.

Б.2 Работа с объектом данных

Получение объекта данных происохдит с ипользованием следующих функций:

```
25      virtual void* getData(void* data_type = NULL) = 0;
26      virtual int getDimSize(int dim) const = 0;
27      virtual int getDimSizeX() const;
28      virtual int getDimSizeY() const;
29      virtual int getDimSizeZ() const;
```

Метод getData(void* data_type) возвращает часть или весь объект данных. Формат возвращаемого значения зависит от параметра data_type.

Метод getDimSize(int dim) иммет смысл, когда объект данных является многомерным массивом (в том числе одномерным) и возвращает количество элементов в размерности dim. Методы getDimSizeX(), getDimSizeY() и getDimSizeZ() работают аналогично getDimSize(int dim) для размерности соответственно 0, 1 и 2. Во-первых, они улучшают читаемость кода, во-вторых, в некоторых случаях могут положительно повлиять на производительность.

Б.3 Сохранение и загрузка модели

```
31      virtual void serialize  (QDataStream& );
32      virtual void deserialize(QDataStream& );
```

Сериализация и десериализация объектов применяется соответственно при сохранении и загрузки проекта. Реализацию данных методов можно оставить пустой, в таком случае проект, в котором используется данная реализация работы с подсистемой вычисления, будет невозможно сохранить и загрузить.

Приложение В. Реализация модуля поддержки режимов отображения

(Рекомендуемое)

Добавление поддержки нового режима отображения осуществляется реализацией двух интерфейсов, один из которых отвечает за рисования изображения, а другой предоставляет пользовательский интерфейс для конфигурации данного режима.

В.1 Рисование изображения

Листинг 2: Интерфейс добавления поддержки режима отображения

```
1  class Renderer : public QObject
2  {
3      Q_OBJECT
4
5      Renderer(const Renderer& );
6      Renderer& operator=(const Renderer& );
7
8  protected:
9      Renderer();
10     Config *config;
11     GraphicBuffer *buffer;
12
13 public:
14     virtual ~Renderer();
15
16     virtual void setParameters(RendererGUI *);
17
18     virtual void setConfig(Config *_config);
19     virtual void setBuffer(GraphicBuffer *_buffer);
20     virtual Config* getConfig() const;
21     virtual GraphicBuffer* getBuffer() const;
22
23     virtual void prepare() = 0;
24     virtual void draw(void *device) = 0;
25 };
```

Класс содержит два свойства: Config *config — указатель на объект, реализующий взаимодействие с вычислительной подсистемой, и GraphicBuffer *buffer — указатель на графический буффер, который используется дли рисования. Для обоих свойств определены методы set и get.

Основными методами являются void prepare() и void draw(void *device). Первый из них вызывается при получении объекта данных от вычислителя, в нём совершаются предварительные вычисления (например, этот метод может использова для вычисления осреднённых значений), которые затем понадобятся при рисовании изображения.

Второй метод вызывает каждый раз, когда требуется заново отрисовать объект данных. Такое разделение обусловлено тем, что один объект данных требуется отрисовывать каждый раз, когда изменяются параметры режима, однако данные может оказаться достаточным обработать только один раз, т. о. получается выигрышь в производительности. Оба данных метода требуется реализовывать при добавлении поддержки нового режима.

Метод setParameters(RendererGUI *) используется для сопоставления режима отображения с настройками из пользовательского интерфейса.

B.2 Графический пользовательский интерфейс

Листинг 3: Реализация графического интерфейса для настроек режима отображения

```
1  class RendererGUI : public QObject
2  {
3      ...
4
5  public:
6      Q_INVOKABLE RendererGUI(Renderer *_rend);
7      virtual ~RendererGUI();
8
9      virtual Renderer* getRenderer() const;
10
11     virtual QString getName() const;
12     virtual QWidget* getWidget() const;
13 };
```

В листинге 3 приведён интерфейс для создания графического пользовательского интерфейса с настройками режима отображения, для краткости в листинге опущены private и protected члены класса как несущественные для ознакомления.

В конструкторе RendererGUI(Renderer *_rend) создаётся оконный интерфейс, который связывается с объектом _rend, реализующий режим отображения.

Метод getName() возвращает название режима отображения, которое будет использовано в графическом интерфейсе. Метод getWidget() возвращает указатель на окно пользовательского интерфейса.

Ссылка на картинку с примером графического интерфейса

Приложение Г. Сценарий построения и отладки модели пользователем

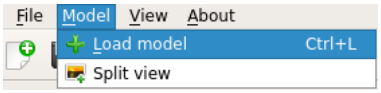


Рис. 7: Загрузка модели в проект

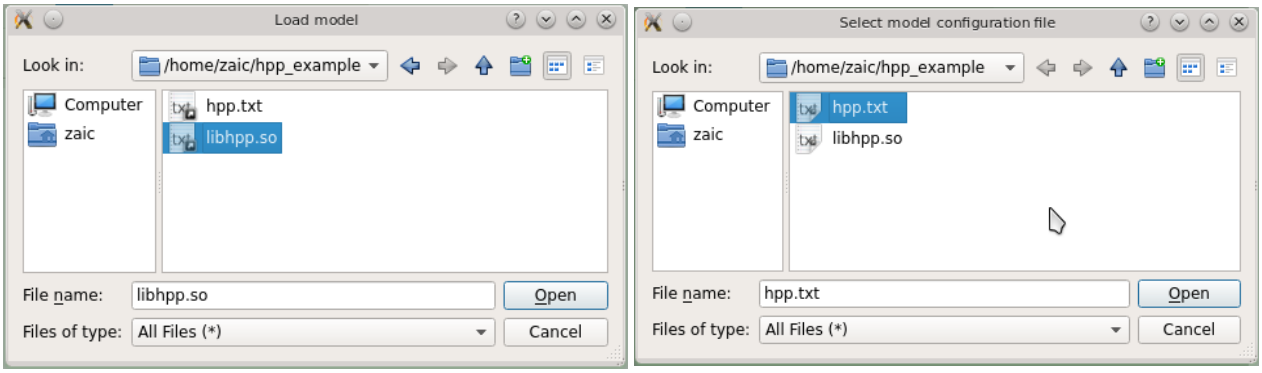
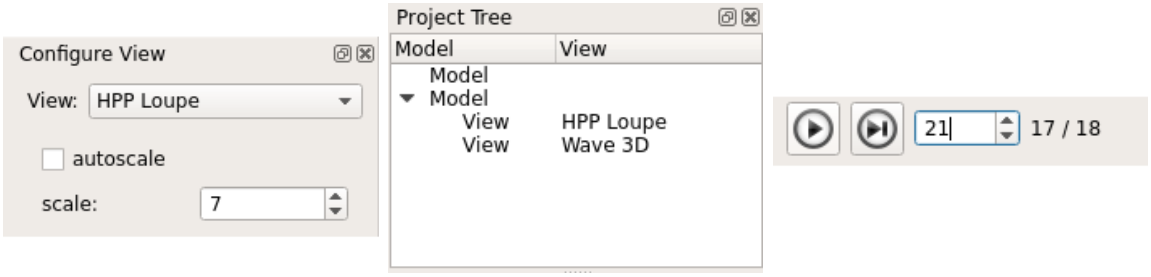


Рис. 8: Загрузка модели в проект



а) б) в)

Рис. 9: Окна работы с моделью

Литература

[1] Лоу А. М. *Имитационное моделирование*. СПб: Питер, 2004.

[2] Филинов Е. Выбор и разработка концептуальной модели среды открытых систем. *Открытые системы*, (6), 1995.

[3] IEEE Std 1003.0-1995 IEEE Guide to the POSIX® Open System Environment (OSE), 1995.