Содержание

Введение

Благодаря беспрецедентно быстрому развитию вычислительной техники, моделирование стало одним из основных способов исследования физических процессов, наряду с экспериментом и теорией. Имитационное моделирование является одним из основных видов моделирования. Оно позволяет строить модели для широкого спектра физических явлений с такой сложностью протекающих в них процессов, которая не даёт возможность описать их в аналитической форме.

Одной из разновидностей имитационных моделей являются модели на основе алгоритмов с мелкозернистым параллелизмом (далее — МЗП-модели).

Мелкозернистый параллелизм, наряду с крупноблочным, является одним из двух основных видов параллелизма. Его отличительная черта — огромное число простых параллельно работающих процессорных элементов. К широко известным классам алгоритмов с мелкозернистым параллелизмом относятся классический клеточный автомат (КА) и его расширения (КА с окрестностью Марголуса [1], асинхронный КА, вероятностный КА), сеть КА, клеточно-нейронная сеть, дискретная динамическая сеть, системы Линденмайера. Алгоритмы и структуры с МЗП используются для моделирования физических [2], химических, биологических и социальных процессов, разнообразных вычислительных устройств (ассоциативный процессор, систолическая структура, однородная среда, универсальная вычислительная среда, конвейеры арифметических устройств) [3].

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 объёмом данных. Под большим объёмом данных подразумевается, что он существенно превосходит размеры основной памяти. Сложность преобразований данных в моделях означает необходимость:

1. отладки модели в процессе её создания;

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. Таким образом, для эффективной работы исследователя необходим инструмент для построения, отладки и исследования моделей.

В первой главе сформулированы требования, предъявляемые к разрабатываемой системе. Главными из них являются адекватность проблемной области, открытость архитектуры и возможность работы с большими объёмами данных.

Во второй главе приводится обзор существующих систем, имеющих средства визуализации моделей. Проведён анализ на предмет их соответствия полученным требованиям, который показал, что ни одна из рассмотренный систем не удовлетворяет им в полной мере.

В третьей главе даётся описание архитектуры системы. В частности, рассматриваются методы, используемые при проектировании системы, и её модульная структура.

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 МЗП-моделей — 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 явления 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 архитектура;
- обработка больших объёмов данных;

Далее будет подробно рассмотрено, 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 МЗП-моделями, то она должна реализовывать необходимые функции, требующиеся при изучении данной области, а именно:

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

1.1.1 Построение моделей

С точки зрения системы модель определяется правилами, задающими поведение моделируемых объектов, параметрами данных объектов (например, для моделей клеточных автоматов это будут правила, по которым которым осуществляются переходы в ячейках, массы частиц покоя и количество движущихся частиц) и объектом данных, над которым производятся вычисления. Объектом данных является структура, хранящая в себе состояние моделируемого объекта. Такая структура может быть как и многомерным массивов так и нерегулярной структурой 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 представлены значениями произвольных типов (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 значения как отдельных ячеек, так и их групп.

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 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. Основными функциями проекта являются сохранение открытого проекта с возможностью последующей загрузки.

1.1.3 Исполнение модели

Одной из основных функций является счёт модели, который 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 исполнение модели. Вычислителем может выступать такое устройство локального или удалённого компьютера, как центральный или графический процессор. Вычислитель при этом необязательно является однопроцессорной машиной: в его роли могут так же высту-

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

1.1.4 Изучение модели

При работе с моделями пользователь системы нуждается в визуализации объекта данных модели не только на микроуровне, но и на макроуровне, позволяющем увидеть качественную картину. Например, иметь возможность выделить из всего объекта данных какую-либо конкретную характеристику. Так же пользователю необходимо дать возможность быть не только пассивным наблюдателем, но и активно влиять на процесс моделирования, т. е. требуется обеспечить интерактивность системы.

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. На макроуровне полезными могут быть не только режимы, визуализирующие какую-либо конкретную численную 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 промежуточных данных с возможностью продолжить вычисления с того момента, где они были приостановлены. Пошаговое исполнение позволяет детальнее изучить поведение модели.

При отладке востребована возможность модификации текущего объекта данных, передача изменений на вычислитель и продолжение счёта.

1.2 Открытая архитектура

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

Свойство открытости включает в себя [4]:

- расширяемость;

- переносимость;
- интероперабельность;
- дружественность интерфейса.

1.2.1 Расширяемость

Потенциальные пользователи системы уже имеют набор активно используемых программ, устоявшиеся форматы данные в этим форматах. Требуется создание универсального инструмента, способного поддерживать существующие форматы и адаптироваться в будущем под новые требования. Поскольку не представляется возможным реализовать поддержку всех используемых форматов данных, обеспечить совместимость со всеми системами моделирования, то вместо этого необходимо предоставить пользователям механизм включения собственных реализаций работы с форматами данных, поддержки новых классов моделей, ранее не предусмотренных системой.

Направлениями для расширения системы являются:

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

Формат данных определяет правила кодирования информации. Существует огромное множество различных форматов данных, постоянно появляются новые, потому требуется механизмы, с помощью которых можно либо добавить в систему поддержку нового формата, либо добавить возможность конвертации нового формата в один из тех, с которыми система уже умеет работать.

В процессе исследования требуется смотреть на различного рода характеристики и невозможно заранее предугадать, какого рода данные предстоит визуализировать. Поддержка добавления новых режимов со стороны системы даёт исследователю инструменты для визуализирования любой характеристику, тем самым решая данную проблему.

Помимо поддержки новых форматов данных и новых режимов отображения, так же должна быть возможность встраивать поддержку новых графических библиотек (например, рисование трёхмерных изображений под ОС Windows делать не с помощью OpenGL, а с использованием библиотеки Direct3D).

Ещё одним направлением расширения системы является добавление поддержки 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 приложения 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 платформе должна быть возможна и на других платформах) [5].

1.2.3 Интероперабельность

Интероперабельность для подсистемы визуализации способность системы взаимодействовать с окружением, в котором она запущена. Для подсистемы визуализации это, в первую очередь, способность взаимодействовать с подсистемой моделирования.

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

1.2.4 Дружественность интерфейса

Интерфейс системы можно разделить на интерфейс пользователя и программный интерфейс [5].

Дружественность пользовательского интерфейса заключается в наличии интуитивно-понятного и комфортного взаимодействия системы с пользователем. Требование является необходим для любой отчуждаемой от разработчиков программы, в которой присутствует функция взаимодействия с пользователем.

Дружественность программного интерфейса относится к разработчикам системы (как и к тем, кто изначально разрабатывает 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 увеличении 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

На сегодняшний день существует немало средств моделирования и визуализации, потому следующим этапом после формирования требований был обзор уже существующих систем.

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

2.1 Mirek's Celebration

Система моделирования 1D и 2D клеточно-автоматных моделей. Имеет 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

Таблица 2: Соответствие требованиям системы WinALT

Адекватность пред. области	+	
Расширяемость	+	
Интероперабельность	+	
Платформонезависимость	??	Графическая среда платформозависимоа Win32, MFC. Ядро и консольная версия - пл не зав.
Дружественный интерфейс	+	
Большие объёмы данных	–	

Система визуального проектирования и отладки Система моделирования широкого спектра МЗП-моделей, в первую очередь алгоритмов и структур.

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	+	Кластеры

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. *где будем мы*

Рис. 1: Существующие системы визуализации

3 Архитектура системы

При проектировании системы использовался принцип модульности — подход к разработке архитектуры системы, при котором приложение разбивается на взаимодействующие между собой модули, каждый из которых выполняет одну функцию и содержит весь необходимый для этого исходный код. *[???]* За счёт такого разделения функционала системы облегчается её дальнейшая разработка и поддержка, т. к. внесение изменений в один модуль (как модификация существующей реализации, так и её расширение) остаются прозрачными для остальных.

При разработке системы использовалась парадигма объектно-ориентированного программирования **??**. В качестве языка разработки был выбран C++ как язык, поддерживающий объектно-ориентированную парадигму и содержащий в себе средства для разработки модульных приложений

На верхнем уровне модульная структура приложения системы представляет из себя четыре взаимодействующих друг с другом модуля (Рис. 2):

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

3.1 Модуль взаимодействия с подсистемой моделирования

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

Рис. 2: Архитектура подсистемы визуализации

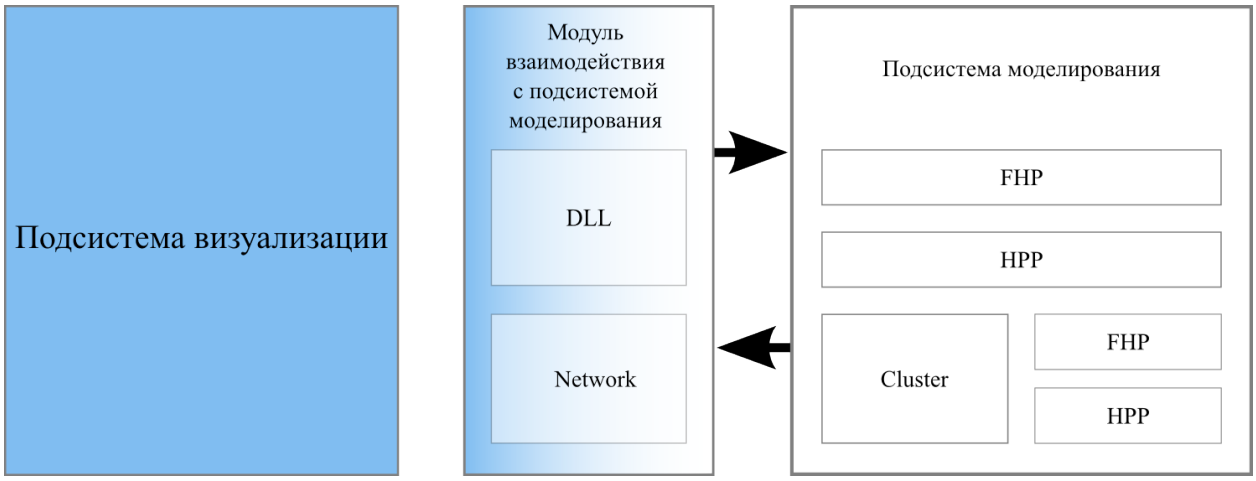


Рис. 3: Взаимодействие с подсистемой моделирования

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 с возможностью подключения к вычислителю в любой момент и получения текущего состояния модели) режимах и реализация протокола передачи объекта данных с подсистемы моделирования в подсистему визуализации.

Класс возможных вычислителей, с которыми может взаимодействовать подсистема визуализации, программно ничем не ограничен. Таким образом, в качестве подсистемы моделирования может выступать приложение, запущенное на абсолютно любом устройстве: как процессор или видеокарта локальной машины, так и удалённый кластер, с которым, возможно, требуется общение по уже существующему протоколу (вполне стандартна ситуация, когда доступ к кластеру осуществляется по SSH с необходимостью авторизации).

Подключение нового вычислителя осуществляется написанием реализации созданного интерфейса. Подробнее о реализации данного модуля написано в приложении Б.

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

Поскольку 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), так и с трёхмерной графикой (OpenGL, Direct3D и т.д.). Основной задачей модуля является предоставление общего интерфейса для рисования независимо от того, какая из библиотек используется в данный момент для формирования итогового изображения.

Модуль поддержки графических библиотек, как и модуль взаимодействия с подсистемой моделирования, включает в себя интерфейс, через реализацию которого осуществляется подключение новой библиотеки. 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, возможность 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 рисования 3D изображений.

3.3 Модуль поддержки режимов отображения

Рис. 5: Модуль поддержки графических библиотек

В разделе 1.1 одним из требований, предъявляемых к системе, является предоставление исследователю как возможность наблюдать за изменениями отдельных различных характеристик, так и возможность увидеть целостную картину. Данный модуль реализует 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 рисовании 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 интерфейсов 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 интерфейсов, отвечающих за рисование изображения и за пользовательский интерфейс с настройками режима отображения. Подробнее о поддержке режимов отображения написано в Приложении В.

На данный момент реализованы:

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

3.4 Пользовательский интерфейс

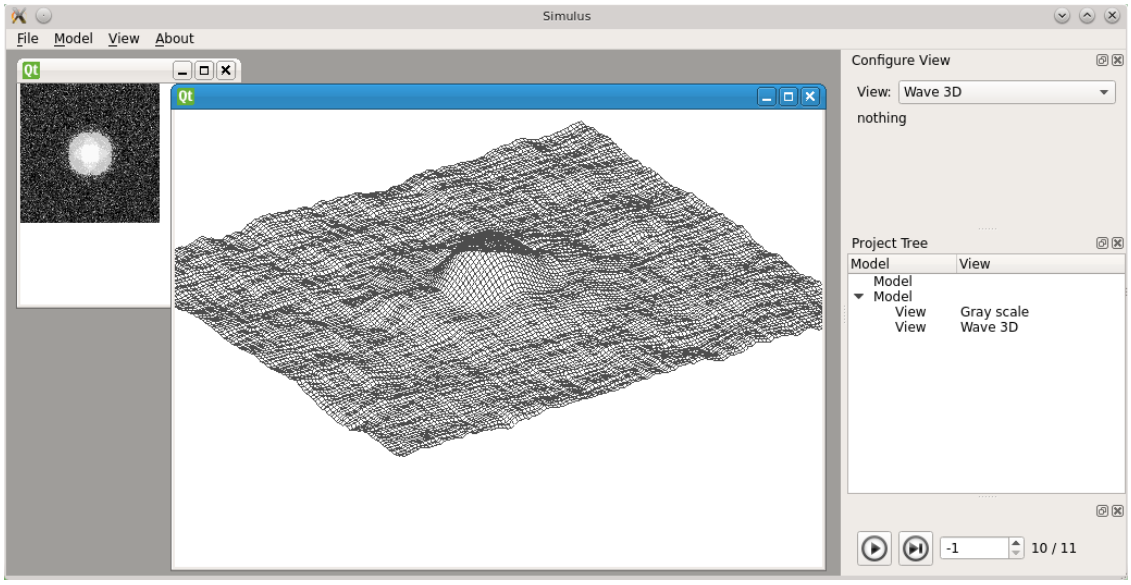


Рис. 6: Пользовательский интерфейс подсистемы визуализации

Для взаимодействия с пользователем системы был реализован графический интерфейс. Задачей модуля было предоставить пользователю простой, но в тоже время функциональный, 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.

Заключение

В результате работы были сформулированы требования к подсистеме визуализцаии, разработана архитектура данной подсистемы и построены реализации её ключевых модулей: модуля взаимодействия с подсистемой моделирования, модуля поддержки режимов отображения, модуля поддержки графических библиотек и пользовательский интерфейс. Произведена стыковка подсистемы визуализации с запускаемой на локальной машине счётной подсистемой, реализующей клеточно-автоматную гидродинамическую модель (HPP), дополненную взвешенными частицами покоя.

Реализация подсистемы может быть использована для исследования волновых процессов в средах с разными волновыми характеристиками с помощью клеточно-автоматных моделей газовой динамики. В частности, разработаны режимы визуализации поля скоростей и амплитуд волн в ограниченных пространствах разной конфигурации.

К числу приоритетных направлений дальнейшего развития можно отнести:

1. разработку режимов визуализации;

2. обеспечение возможности взаимодействия со счётными программами, исполняющимися на удалённых машинах, кластерах и ГРИД системах. В первую очередь такая возможность будет реализована для кластера Сибирского Суперкомпьютерного Центра (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 взаимодействия 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 методов уже имеет простую реализацию и для добавления поддержки 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 происходит с использованием 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 использоваться 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 перерисовывать каждый раз, когда изменяются параметры режима, однако данные может оказаться достаточным обработать только один раз, т. о. получается выигрыш 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 запуска процесса моделирования

Работа с моделью начинается с добавления модели в проект — в систему загружаются все необходимые функции по работе с загружаемой моделью, после чего пользователь имеет возможность управлять созданной моделью. Пример загрузки модели показана на Рис. 7 и на Рис. 8 — здесь выбирается динамически-подключаемая библиотека с реализацией модели и указывается файл с начальной конфигурацией.

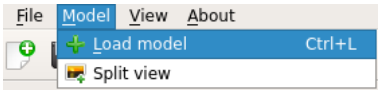


Рис. 7: Загрузка модели в проект

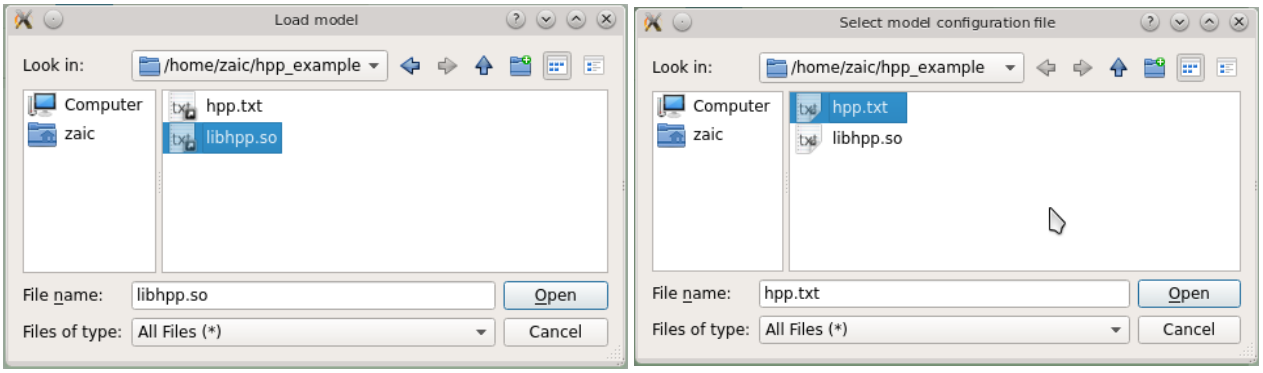
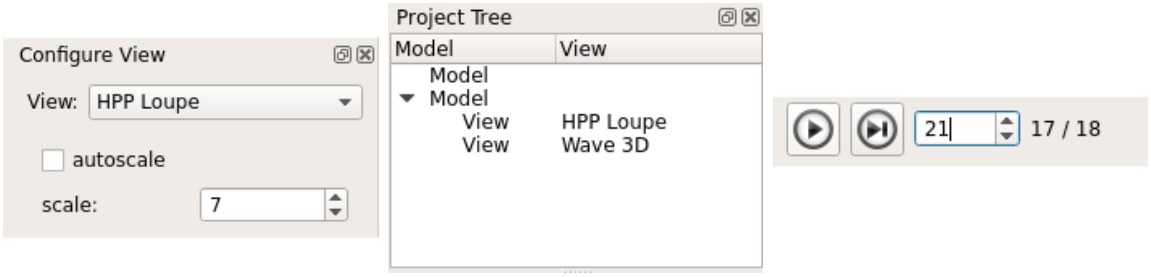


Рис. 8: Загрузка модели в проект

Далее пользователь может запустить модель на счёт с помощью элемента управления, представленного на Рис. 9 в). Может выбирать режимы отображения и настраивать их параметры (Рис. 9 а)).



а) б) в)

Рис. 9: Окна работы с моделью

Литература

[1] Toffoli T. *Cellular Automata Machines: A New Environment for Modelling.* MIT Press, 1987.

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

[3] Ilachinski A. *Cellular automata: a discrete universe.* Singapore: World Scientific, 2001.

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

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

попр