Рябинин К.В.

Методы и средства разработки адаптивных мультиплатформенных систем визуализации научных экспериментов

Диссертация

Пермский государственный национальный исследовательский университет

На правах рукописи

Рябинин Константин Валентинович

Методы и средства разработки адаптивных мультиплатформенных систем визуализации научных экспериментов

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

ДИССЕРТАЦИЯ
на соискание учёной степени
кандидата физико-математических наук

Научный руководитель
к.ф.-м.н.
Чуприна Светлана Игоревна

Пермь – 2014

Содержание

Введение

В настоящее время объекты исследования естественных и гуманитарных наук становятся всё более сложными и, соответственно, требуют всё более сложных средств для наглядного представления. Традиционные методы отображения данных, такие как построение диаграмм, графиков и таблиц, часто оказываются недостаточными для отражения всех сторон изучаемых объектов, процессов и явлений; более адекватными выступают методы построения специализированных схем или фотореалистичных изображений. Для этого активно используются мультимедийные возможности современных ЭВМ: средства трёхмерной и даже многомерной графики [1], звуковое сопровождение, построение стереоскопических изображений и т. д. в сочетании с использованием современных устройств ввода-вывода (например, шлемами виртуальной реальности и манипуляционными перчатками) и технологиями дополненной реальности.

В современной компьютерной графике выделяется целое направление так называемой «научной визуализации». М. Френдли (M. Friendly) определяет научную визуализацию как «область графики, в первую очередь ориентированную на отображение трёхмерных объектов (из области архитектуры, метеорологии, медицины, биологии и т. д.), при котором основной упор делается на реалистичность рендеринга объёмов, поверхностей, источников света и т. п., часто с наличием динамической (временной) компоненты» [2]. Таким образом, научная визуализация занимается вопросами точного и понятного человеку отображения объектов, процессов и явлений, представляющих научный интерес.

В противоположность «научной компьютерной графике» часто ставят так называемую «игровую компьютерную графику». С точки зрения средств реализации процесса рендеринга такое деление весьма условно, однако мультимедийные системы, предназначенные для развлекательных целей, зачастую оказываются плохо применимыми для отображения результатов научных экспериментов, и наоборот. Это происходит ввиду специфических целей научной

визуализации, а также ввиду различных требований, которые предъявляют «игровая» и «научная» компьютерная графика к вычислительным ресурсам и, следовательно, к элементной базе.

Разделение компьютерной графики на научную и игровую можно провести на основе:

1. Источника данных, подлежащих визуализации. В научной графике источником данных служит, как правило, научный эксперимент (результаты каких-либо измерений) или математическое моделирование некоторого явления. В игровой же графике изображение строится по некоторым синтетическим данным, описывающим «игровую вселенную» – виртуальный мир со своими законами и принципами, возможно, сильно отличающимися от действующих в реальности.
2. Характера данных, подлежащих визуализации. В научной графике данные максимально приближены к реальности, обладают, как правило, высокой детализацией и, соответственно, большим объёмом. Иногда допустимым считается потеря интерактивности в пользу высокой детализации (то есть происходит отказ от визуализации в реальном времени). В игровой графике, напротив, действует так называемое «правило минимализма»: объекты должны иметь минимально возможное разрешение, чтобы только сохранять визуальную привлекательность. Основной упор в игровой графике делается на интерактивность, даже если ради этого приходится отказываться от реалистичности.
3. Способов визуализации. В научной графике часто наряду с традиционным отображением поверхностей объектов используются также способы визуализации объёмов, сечений, поверхностей уровня, вокселей [3] и т. д. В игровой графике, как правило, используется только отображение поверхностей и, иногда, вокселей.
4. Специальных эффектов. Под специальными эффектами понимаются различного рода декоративные элементы: вспышки, системы частиц,

эффекты освещения, анимационные переходы между состояниями сцены, эффекты постобработки картинки и т. д. Наличие специальных эффектов в большей степени характерно для игровой графики, в научной же их применение является очень ограниченным: излишние спецэффекты могут нарушать реалистичность картинки или отвлекать внимание исследователя.

Сравнивая научную и игровую графику, в общем случае невозможно сказать, какая из них сложнее в реализации: большие объёмы входных данных и разнообразие способов визуализации в научной графике компенсируется сложностью спецэффектов и необходимостью достижения высокой интерактивности в игровой графике. Однако на этапе проектирования системы визуализации, как правило, определяют заранее, с каким типом графики она будет работать.

Несмотря на это, существуют решения общего назначения, одинаково хорошо применимые для решения как игровых, так и научных задач.

Компьютерная графика не ограничивается этими двумя направлениями, включая в себя и другие самостоятельные ветви, например, кинематографическую графику, то есть построение фотореалистичных изображений в режиме отложенной визуализации. Такая графика предназначена для создания мультипликационных фильмов или специальных эффектов для кинематографа. В данной работе, в соответствии с заявленной темой, будет рассматривается только научная графика реального времени и проблемы, связанные с ней.

К научной графике могут быть отнесены все виды диаграмм, визуализация в геоинформационных системах, визуализация динамики жидкостей и газов, отображение изолиний и изоповерхностей, визуализация многомерных данных и многое другое.

Научной визуализацией как направлением компьютерной графики активно занимаются в ведущих исследовательских центрах мира, например, в лаборатории компьютерной графики Массачусетского технологического института, центре научной визуализации университета Беркли, студии научной визуализации НАСА,

лаборатории компьютерной графики Мюнхенского технического университета, лаборатории компьютерной графики и вычислительной оптики Института прикладной математики им. М.В. Келдыша РАН, лаборатории компьютерной графики и мультимедиа факультета вычислительной математики и кибернетики Московского государственного университета им. М.В. Ломоносова, лаборатории численного анализа и машинной графики Института вычислительной математики и математической геофизики Сибирского отделения РАН и т. д. Среди зарубежных учёных большой вклад в развитие этого направления сделали М. Фрэндли [2], А. Сазерленд (I. Sutherland) [4], Б. Хэймс (B. Haimes) [5], Т. Джу (T. Ju) [6], К. Гручалла (K. Gruchalla) [7], М. Куттел (M. Kuttel) [8], С. Вольфрам (S. Wolfram) [9] и др. Среди российских – О.В. Джосан [10], А.В. Игнатенко [11], А.И. Сурин [12], Е.Л. Карташева [13], В.Л. Авербух [14], А.Е. Бондарев [15], В.А. Галактионов [16], А.Г. Волобой [17], В.Р. Васильев [18], Н.И. Вьюкова [18], В.Е. Турлапов [19], Л.А. Залогова [20], С.В. Смирнов [21], Е.И. Артамонов [22], А.В. Толок [23], Я.Д. Кузнецов [24], А.А. Зацаринный [25] и др. Этими учёными были сформулированы основополагающие принципы научной визуализации, составлены базовые алгоритмы синтеза изображений, отвечающие данным принципам, и разработаны программные системы, в которых эти алгоритмы были реализованы на практике.

На сегодняшний день существует большое количество программных пакетов и библиотек для научной визуализации, однако ряд проблем в этой области до сих пор не имеет эффективных решений.

Первая проблема состоит в том, что часть из этих систем являются узкоспециализированными, то есть пригодны лишь для ограниченного списка научных задач. У большинства универсальных систем, в свою очередь, отсутствуют эффективные механизмы, автоматизирующие интеграцию со сторонними решателями (*англ.* solvers) – программными (а иногда и программно-аппаратными) комплексами, которые генерируют исходные данные для построения изображения. Таким образом, перед исследователем часто возникает

выбор – либо разрабатывать систему визуализации для своей задачи с нуля, либо вручную производить конвертацию данных, получаемых от решателя, в формат, пригодный для существующих универсальных систем визуализации. Оба варианта в значительной степени повышают трудоёмкость научных экспериментов и зачастую заставляют исследователей обращаться за помощью к сторонним разработчикам программного обеспечения (ПО). Вследствие этого актуальной является задача разработки универсальных систем научной визуализации, поддерживающих автоматизированную интеграцию с разнородными решателями при помощи высокоуровневого графического интерфейса пользователя.

Вторая проблема заключается в относительно слабой проработанности вопроса создания распределённых систем визуализации. Часто решателем выступает некоторая ресурсоёмкая программа, выполняющаяся на удалённом компьютере и генерирующая большие объёмы выходных данных. В таких условиях адекватным является использование для системы визуализации клиент-серверной архитектуры. На сегодняшний день в большинстве клиент-серверных систем построение изображения происходит либо целиком на стороне клиента (с передачей всех необходимых данных по сети), либо целиком на стороне сервера (с передачей по сети лишь финального изображения). Однако оба этих подхода имеют ряд недостатков и актуальной является задача разработки способов их комбинации, то есть построения систем научной визуализации, использующих принцип распределённого между клиентом и сервером рендеринга.

Третья проблема – это малое количество мультиплатформенных решений, то есть таких систем визуализации, которые могли бы работать под управлением операционных систем (ОС) для настольных компьютеров (таких ОС, как Windows, GNU / Linux, OS X и др.), высокопроизводительных вычислительных комплексов (таких ОС, как HPC Windows, GNU / Linux и др.) и мобильных устройств (таких ОС, как iOS, Android и др.). Традиционно мобильные устройства не рассматриваются как средства, пригодные для построения сложных изображений

в реальном времени, однако с ростом их популярности и улучшением их технических характеристик этот вопрос нуждается в пересмотре. Актуальной является задача разработки средств и методов создания мультиплатформенных графических приложений, в частности – систем научной визуализации.

Четвёртая проблема состоит в том, что визуальное качество изображений, синтезируемых существующими системами научной визуализации, иногда оказывается недостаточно высоким ввиду ступенчатости границ объектов. Ступенчатость возникает из-за того, что визуализация является процессом дискретизации непрерывной математической модели сцены: в результате отображения на растр границы объектов перестают быть гладкими. Для решения данной проблемы применяются специальные алгоритмы сглаживания границ на изображении, однако в большинстве своём они снижают производительность визуализации и могут привести к потере интерактивности или потере плавности воспроизведения анимации. Актуальной является задача создания адаптивных алгоритмов сглаживания, способных обеспечивать высокое качество изображения без снижения производительности рендеринга, вне зависимости от программно-аппаратной платформы.

В контексте рассматриваемых проблем не затрагиваются вопросы т. н. Больших данных (*англ.* Big Data). Эта весьма актуальная проблематика выходит за рамки данного диссертационного исследования.

Целью диссертационного исследования является разработка методов создания мультиплатформенных систем научной визуализации, предоставляющих высокоуровневые средства интеграции со сторонними решателями.

Задачами, которые необходимо решить для достижения поставленной цели, являются:

1. Исследование современных средств научной визуализации и анализ существующих проблем.
2. Разработка методов:

   2.1. Интеграции систем научной визуализации со сторонними решателями.

2.2. Автоматической генерации мультиплатформенного графического интерфейса пользователя.
2.3. Адаптивного распределения процесса визуализации между узлами вычислительной системы.
2.4. Сглаживания ступенчатых границ объектов на изображении без существенного снижения производительности визуализации.

3. Реализация предложенных методов на практике путём разработки системы научной визуализации и применения её для решения реальных научных задач в различных предметных областях.

Научная новизна работы заключается в создании новых методов и средств для комплексного решения задач научной визуализации, включающего в себя:

1. Унифицированную организацию мультиплатформенных клиент-серверных систем научной визуализации, поддерживающих интеграцию на принципах адаптации со сторонними решателями, удовлетворяющими сформулированным ограничениям.
2. Автоматическую генерацию мультиплатформенного графического интерфейса пользователя по высокоуровневому описанию с решением проблемы двойного дизайна интерфейсов без снижения производительности визуализации.
3. Адаптивное распределение процесса визуализации между клиентом и сервером на основе эвристик, обеспечивающее поддержку работы системы на мобильных устройствах, подключенных по низкоскоростному сетевому соединению.
4. Адаптивное сглаживание границ объектов на изображении, обеспечивающее высокое качество изображения и в 3 раза более быстрый отклик системы на команды пользователя по сравнению аналогами.
5. Адаптивную мультиплатформенную систему научной визуализации, реализованную на основе предложенных методов и средств.

Достоверность полученных результатов подтверждается теоретической обоснованностью применяемых в исследовании научных методов, а также проверкой предложенных методов и средств путём создания на их основе адаптивной системы научной визуализации и тестированием этой системы при решении реальных прикладных задач из различных научных областей. Результаты тестирования подтвердили, что система может быть легко адаптирована к специфике различных решателей, а также обладает высокой производительностью визуализации на различных платформах.

Практическая значимость. Результаты диссертационного исследования могут быть использованы как методологическая база для создания систем научной визуализации, обладающих высокоуровневыми средствами интеграции со сторонними решателями. Разработанная система научной визуализации может выступать в качестве программной платформы для двумерной и трёхмерной визуализации научных экспериментов в различных областях знания, включая междисциплинарные исследования. Отдельные модули системы имеют самостоятельное значение и могут быть использованы для организации визуализации и мультиплатформенной переносимости при создании традиционных мультимедийных приложений.

Разработанные в рамках диссертационного исследования алгоритм адаптивного сглаживания границ объектов на изображении и система научной визуализации SciVi были внедрены в пермской IT-компании ООО «Ньюлана». Копии актов о внедрении представлены в приложении 1. Созданные при непосредственном участии автора диссертационного исследования библиотеки функций NGraphics и NChart3D, лежащие в основе программной системы SciVi, в комплексе и по отдельности используются в целом ряде программных продуктов ООО «Ньюлана», изготовленных по заказу компаний Hewlett Packard, Thomson Reuters, Roche, Citi Bank и Институт генетических исследований Genomics Institute of the Novartis Research Foundation и др. Указанные библиотеки, а также сама система SciVi зарегистрированы в Федеральной службе по

интеллектуальной собственности Роспатент. Копии соответствующих свидетельств регистрации представлены в приложении 2.

Методы исследования. В диссертационной работе используются методы и средства вычислительной геометрии и компьютерной графики для автоматического синтеза высококачественных двумерных и трёхмерных изображений и для отображения графического интерфейса пользователя. Для оценки сложности разработанных алгоритмов использован понятийный аппарат теории сложности алгоритмов. Для организации автоматической настройки на сторонние решатели использованы методы и средства онтологического инжиниринга (методологии представления, хранения и обработки знаний на основе онтологий). Проектирование и реализация системы осуществлены с применением методов параллельного и объектно-ориентированного программирования.

На защиту выносятся следующие полученные автором **научные результаты:**

1. Метод автоматизированной интеграции на принципах адаптации систем научной визуализации со сторонними решателями, основанный на онтологическом инжиниринге.
2. Метод автоматической генерации мультиплатформенного графического интерфейса пользователя, в отличие от известных аналогов решающий проблему двойного дизайна интерфейсов без снижения производительности визуализации.
3. Метод адаптивного распределения процесса визуализации между клиентом и сервером, основанный на применении эвристических правил.
4. Метод сглаживания границ объектов на изображении, обеспечивающий более высокое визуальное качество результата и в 3 раза более высокую производительность по сравнению с аналогами, а также допускающий мультиплатформенную реализацию.

5. Средства научной визуализации, включающие архитектуру адаптивных мультиплатофрменных систем научной визуализации и реализованную на её основе программную систему SciVi, которая, в отличие от аналогов, предоставляет высокоуровневые средства интеграции со сторонними решателями, безотносительно к программным средствам их реализации и типу решаемых ими научных задач.

Апробация работы и публикации. Основные положения диссертационной работы докладывались и обсуждались на следующих конференциях и семинарах:

1. 22-я Международная конференция по компьютерной графике и машинному зрению «ГрафиКон'2012», МГУ, Москва, 2012 г.
2. Всероссийская научно-практичесская конференция «Актуальные проблемы механики, математики, информатики», ПГНИУ, Пермь, 2012 г.
3. Межвузовская конференция «Междисциплинарные исследования – будущее науки XXI века», ПГНИУ, Пермь, 2013 г.
4. Международная конференция «International Conference on Computational Science 2013 – Computation at the Frontiers of Science», Барселона, Испания, 2013 г.
5. Всероссийская научно-практическая конференция молодых учёных «Современные проблемы математики и её прикладные аспекты», ПГНИУ, Пермь, 2013 г.
6. Совместный научный семинар кафедр ПУиИБ, ИТ, МОВС (ПГНИУ), ПИ (ПГГПУ) и ГК ИВС, ПГНИУ, Пермь, 2013 и 2014 гг.
7. II Всероссийская научно-практическая конференция с международным участием, с элементами научной школы для молодёжи «Высокопроизводительные вычисления на графических процессорах», ПГНИУ, Пермь, 2014 г.
8. Международная конференция «International Conference on Computational Science 2014 – Big Data meets Computational Science», Кэрнс, Австралия, 2014 г.

Основные результаты диссертации опубликованы в 16 работах, среди которых 13 статей, из них 5 в изданиях, включенных в перечень ведущих рецензируемых журналов, рекомендованных ВАК, в частности, 3 в изданиях, индексируемых Scopus и Web of Science.

Глава 1. Подходы к разработке систем визуализации научных экспериментов

Визуализация, как наглядное представление данных, является очень важной составляющей процесса научных исследований. Она позволяет представлять исходные данные, процесс и результаты опытов и экспериментов в образном виде, делая их более понятными для человека. На современном этапе развития естественных и гуманитарных наук объекты исследования, которыми приходится оперировать, настолько сложны, что традиционные средства визуализации – диаграммы, графики и таблицы – оказываются уже недостаточно наглядными. Наряду с этими средствами требуется использовать новые, основанные на мультимедийных возможностях ЭВМ. К такого рода новым средствам визуализации относятся трёхмерные модели и интерактивные виртуальные миры, позволяющие адекватно представлять пространственные и временные данные. Эти средства в настоящее время активно используются в сочетании с технологиями дополненной реальности, что позволяет в автоматическом режиме объединять образы, синтезированные на компьютере, с реальной картиной мира.

1.1. Специализированные аппаратные средства визуализации

Мультимедийные возможности современных компьютеров, расширяемые специализированными аппаратными средствами ввода и вывода информации, способны многократно увеличить реалистичность изображения и даже создать так называемый эффект присутствия, когда пользователю кажется, что синтетические объекты находятся рядом с ним. К такого рода аппаратным решениям, в первую очередь, относятся приборы, способствующие погружению в виртуальную реальность: шлемы виртуальной реальности, перчатки-манипуляторы, проекционные системы типа Cave, устройства захвата движений (например, Microsoft Kinect) и т. д. Кроме того, значительно улучшить визуальные характеристики итогового изображения способны также стереомониторы и мониторы высокого разрешения, например, мониторы на основе ретина-

дисплея [26]. Для организации высококачественной научной визуализации реалистичные трёхмерные модели объектов могут быть получены при помощи технологии трёхмерного сканирования, то есть автоматической оцифровки объектов реального мира.

Актуальной является задача разработки систем научной визуализации с поддержкой современных аппаратных средств ввода и вывода мультимедийной информации, таких, например, как трёхмерные сканеры или ретина-дисплеи.

1.2. Принципы построения систем научной визуализации

Важным требованием для системы визуализации является возможность простой настройки на источник данных. В том случае, если речь идёт о визуализации научных экспериментов, источником данных, как правило, выступает так называемый «решатель» – некоторая система, производящая сам эксперимент. В общем случае решателем может выступать программно-аппаратный комплекс. Если речь идёт о реальном эксперименте, комплекс снабжается необходимыми манипуляторами, датчиками и системами обработки сигнала с этих датчиков. В более простом случае, когда под экспериментом понимается математическое моделирование некоторого процесса, решателем выступает только компьютерная программа. В ходе своей работы решатель порождает некоторые числовые данные (математическое описание объектов и процессов), которые и должны быть представлены в наглядном для анализа виде. Для повышения эффективности исследовательского процесса, система визуализации должна иметь средства интеграции с решателем, то есть предоставлять возможность автоматически устанавливать связь с решателем, получать и отображать необходимые данные, а также позволять исследователю управлять решателем: изменять входные данные, приостанавливать и возобновлять вычислительный эксперимент и т. д., не изменяя его функциональность.

Сложность математических моделей, описывающих объекты и процессы в естественных, а иногда и в гуманитарных науках, всё чаще приводит к необходимости использования высокопроизводительных вычислительных комплексов, таких как суперкомпьютеры или облачные вычислители. В этом случае наличие в системе визуализации средств автоматизированной интеграции с решателем становится ещё более актуальным: интеграция даёт возможность сократить время получения данных и адаптации их к конкретному отображающему приложению, а также предоставляет возможность управлять вычислениями и видеть их результат при помощи единого графического интерфейса пользователя. Таким образом, учёный-исследователь может использовать систему визуализации, установленную у него на локальной ЭВМ (персональном компьютере или мобильном устройстве), как полноценное средство для организации доступа к решателю, который физически может быть расположен на удалённом высокопроизводительном сервере. При этом система визуализации должна обладать универсальностью и настраиваемостью, чтобы иметь возможность подключаться к различным решателям и взаимодействовать с ними при помощи общего интерфейса.

Для обеспечения интеграции с решателем системы визуализации чаще всего строятся на основе клиент-серверной архитектуры. Сервер находится на стороне решателя и взаимодействует с ним непосредственно, а клиент находится на стороне пользователя, предоставляя графический интерфейс управления и отображая результат визуализации. При этом встаёт вопрос об эффективной пересылке данных, подлежащих визуализации, а также о балансировке нагрузки клиента и сервера. Существует три основных подхода к разделению обязанностей клиента и сервера [27]:

1. Визуализация в полном объёме выполняется на клиенте (клиент получает от сервера данные, подлежащие визуализации).
2. Визуализация в полном объёме выполняется на сервере (клиент получает от сервера готовое изображение).

3. Визуализация распределена между клиентом и сервером.

На сегодняшний день чаще всего применяются первые два подхода [27]. Выбор между ними зависит от конкретной задачи, так как оба они имеют свои достоинства и недостатки.

Первый подход применим в том случае, когда, во-первых, объём исходных данных сравнительно мал, то есть пригоден для передачи по сети за приемлемое время, а во-вторых, клиент обладает достаточной вычислительной мощностью, чтобы самостоятельно произвести рендеринг итогового изображения. Положительной стороной подхода является обеспечение высокой интерактивности: после того, как все необходимые данные переданы клиенту, он может быстро перестраивать сцену в ответ на команды пользователя, не отправляя дополнительных запросов серверу. Отрицательной же стороной являются повышенные требования, предъявляемые к клиенту и каналу связи. В контексте визуализации научных данных, как правило, приходится иметь дело с большими объёмами информации и сложными алгоритмами визулизации, а значит, применимость этого подхода оказывается ограниченной – особенно в том случае, если клиентом выступает мобильное устройство невысокой вычислительной мощности, подключенное к сети через низкоскоростное беспроводное соединение. На принципах данного подхода, например, построена система просмотра трёхмерных моделей MeshLab [28].

Второй подход предполагает, что все действия по формированию изображения выполняются сервером, поэтому он применим в ситуации, когда производительность сервера значительно выше производительности клиента. Положительной стороной этого подхода является почти полное отсутствие системных требований для клиента: он должен лишь иметь возможность установить соединение с сервером и затем отображать передаваемые ему изображения. При этом, при большом числе клиентов, одновременно запрашивающих кадры визуализации, сервер может оказаться перегруженным. Кроме того, при интерактивном изменении сцены резко возрастает нагрузка на

сеть [29]. Например, при плавном вращении трёхмерной сцены необходимо получить от сервера кадры, отображающие каждый из углов поворота. Учитывая, что для создания эффекта плавности движения нужно отображать хотя бы 8 кадров в секунду [30] (а вообще говоря, чем больше, тем лучше), требование к скорости подготовки данных сервером и доставке кадров по сети оказываются очень высокими. Таким образом, в условиях низкоскоростного соединения организация интерактивности и воспроизведение плавной анимации могут оказаться невозможными. Кроме того, зачастую возникает ситуация дублирования передаваемой информации в случае, если, например, пользователь посмотрел сцену сначала под одним углом, затем – под другим, а после этого вновь вернулся к первоначальному ракурсу. Обычно не предусматривается никакой системы кеширования кадров, то есть полученные ранее данные будут запрашиваться и передаваться по сети повторно. На таких принципах построены технологии VNC [31], VirtualGL [14], RemoteFX [32] и vSGA [33].

Третий подход, по сути, является комбинацией первых двух. Он наиболее сложен в реализации, однако способен объединить в себе достоинства первых двух подходов, минимизируя при этом влияние их недостатков. Одной из первых российских разработок в области использования этого подхода является система для фотореалистичной визуализации LKernel [34]. Эта система является, по сути, предвестником облачных визуализаторов. Она выполняет основную (наиболее ресурсоёмкую) часть визуализации в вычислительном облаке, но при этом предоставляет клиенту возможность интерактивного изменения графической сцены.

Идея использования распределённого рендеринга в контексте создания систем научной визуализации заключается в адаптивном разделении процесса построения изображения между клиентом и сервером. Предполагается, что конкретная схема распределения задач, возникающих на пути построения итогового изображения, определяется при помощи некоторых эвристик. Например, на стороне высокопроизводительного сервера может быть выполнена

отрисовка наиболее сложных частей данных, а также произведено упрощение «сырых» данных, передаваемых клиенту. При балансировке нагрузки во внимание могут быть приняты такие факторы, как быстродействие клиента, загруженность сервера (количество подключенных к нему клиентов) и скорость соединения. Таким образом, может быть достигнута высокая интерактивность отображаемой сцены при сравнительно небольших требованиях к производительности клиента и ширине канала связи. Это означает, что при разработке мультиплатформенной системы научной визуализации следует ориентироваться именно на данный подход. Однако в доступной литературе достаточно слабо освящены вопросы его практической реализации, поэтому исследования в этой области являются на сегодняшний день очень актуальными.

Типичными подходами к рендерингу, которые используются в научной визуализации, является построение разнообразных графиков, диаграмм, изолиний, поверхностей уровня, визуализация векторных и скалярных полей, визуализация сечений и объёмов. Кроме того, распределение интересующих исследователя величин на поверхности изучаемых объектов отображается в виде закраски поверхностей трёхмерных моделей этих объектов различными цветами, либо путём наложения на эти поверхности соответствующих текстур. Для достижения высокой наглядности представляемых данных, описанные способы рендеринга могут комбинироваться с отображением различных дополнительных объектов, создающих контекст научной визуализации [18], например, интерьеров помещений, в которых измеряется или моделируется распределение температуры, или летальных аппаратов, для которых моделировалось обтекание воздухом. Дополнительными объектами при этом являются трёхмерные модели или целые трёхмерные сцены (группы трёхмерных объектов, возможно с анимацией).

Важным является также наличие интерактивности при взаимодействии пользователя с графической сценой: возможности осуществлять навигацию по сцене и применять к ней аффинные преобразования (смещение, масштабирование, поворот). Полезной функцией является также возможность просматривать сцену

на различных уровнях детализации. В том случае, если на сцене присутствует динамический (временной) компонент, то есть анимация, её скорость должна быть настраиваемой.

При отображении трёхмерных моделей средствами низкоуровневых библиотек стандартов OpenGL и Direct3D границы объектов на изображении оказываются ступенчатыми в силу того, что непрерывная математическая модель трёхмерных поверхностей отображается на растр. Для увеличения визуального качества результирующих изображений, в процесс визуализации добавляется дополнительный этап: сглаживание ступенчатых границ объектов. Этот этап, однако, требует дополнительных вычислительных ресурсов и снижает производительность визуализации, а также в некоторых случаях затрудняет поддержку эффективного параллелизма рендеринга [35]. Актуальной является задача разработки методов сглаживания границ объектов на изображении, пригодных для применения в системах научной визуализации, для обеспечения высокого визуального качества результатов рендеринга.

1.3. Особенности разработки систем научной визуализации на мобильных платформах

Мультимедийные средства вычислительной техники в настоящее время активно развиваются, предоставляя всё более широкие возможности для визуализации. При этом развитие идёт сразу в двух направлениях: как в сторону увеличения производительности, так и в сторону создания новых средств человеко-машинного взаимодействия вплоть до создания качественно новых типов ЭВМ. По данным [36] на начало 2014 года численность пользователей мобильных устройств (смартфонов и планшетных компьютеров) составляет около 1,5 миллиардов человек, то есть примерно 20% от общего населения планеты. При этом количество мобильных устройств на душу населения продолжает стремительно расти. Традиционно мобильные устройства занимали нишу средств коммуникации (непосредственной связи людей друг с другом и доступа в сеть Интернет), однако на сегодняшний день их технические характеристики

позволяют производить, в частности, и сложную визуализацию. Экономическая доступность (стоимость мобильного устройства в среднем в 6 раз ниже стоимости настольного компьютера), мобильность (масса мобильного устройства в среднем в 6 раз меньше массы ноутбука) и удобный интерфейс, основанный на жестах, голосовых командах, видеокамере и внутренних датчиках положения в пространстве, могут превратить эти устройства в мощные исследовательские инструменты. Такие инструменты могут использоваться учёными повсеместно, в частности, в полевых условиях (например, во время научно-исследовательских экспедиций), где настольные компьютеры оказываются недоступны.

В связи с этим, актуальной является разработка систем научной визуализации, являющихся мультиплатформенными, то есть работающими на настольных компьютерах, на высокопроизводительных вычислительных комплексах и на мобильных устройствах без модификации исходного кода. Именно поэтому задача достижения мультиплатформенности вынесена в качестве одной из первоочередных в данной работе.

С целью определения аппаратных ограничений мобильных платформ, которые необходимо учитывать при проектировании мультиплатформенных систем научной визуализации, было проведено исследование технических особенностей мобильных устройств. По результатам этого исследования было проведено сравнение современных мобильных устройств с современными настольными компьютерами.

Количественные различия в технических характеристиках мобильных устройств и настольных компьютеров представлены на рис. 1. Для удобства, за единицу принята числовая характеристика каждого из параметров мобильного устройства.

Качественные различия состоят в следующем:

1. Набор поддерживаемых функций графического конвейера (например, на мобильных устройствах отсутствует поддержка геометрических шейдеров).

2. Источник питания (мобильные устройства используют аккумуляторы, поэтому при написании мобильных приложений следует принимать во внимание экономию энергии, минимизируя время взаимодействие с сетью, количество операций ввода-вывода и время хранения в оперативной памяти больших объёмов данных).
3. Парадигма управления (на мобильных устройствах вместо клавиатуры и мыши используется сенсорный экран, а взаимодействие пользователя с приложением основано на жестах касаний).

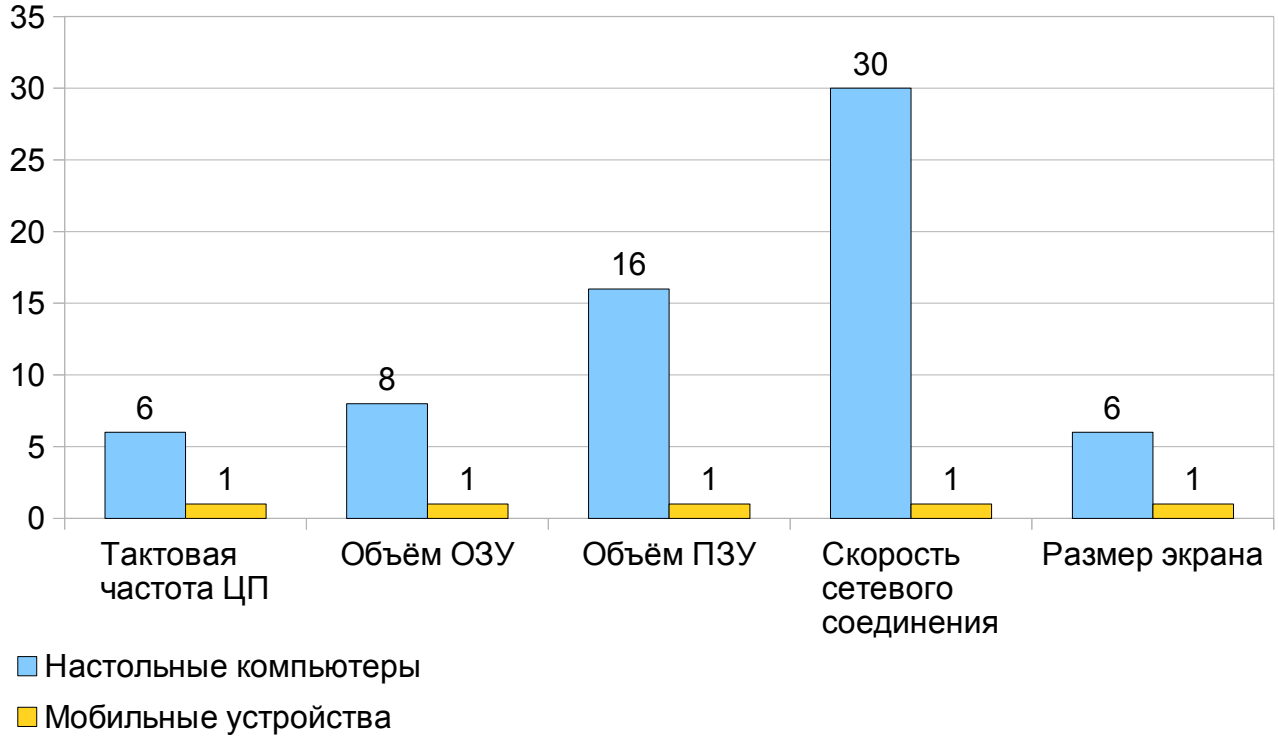


Рис. 1. Соотношение технических характеристик мобильных устройств и настольных компьютеров

В результате проведённого исследования наиболее развитых на сегодняшний день методов и средств научной визуализации было установлено, что они в недостаточной мере учитывают особенности платформ и в большинстве своём не готовы к переносу на мобильные устройства. Такая ситуация, отчасти, связана с тем, что мобильные устройства набрали достаточный потенциал для поддержки соответствия требованиям к системам научной визуализации сравнительно недавно (примерно с 2005 г. [11]). Следовательно, актуальной

является научно-исследовательская задача разработки методов и средств научной визуализации, совместимых с данным классом ЭВМ.

1.4. Краткий обзор систем и инструментов научной визуализации

1.4.1. Приложения для научной визуализации

Традиционно для наглядного представления научных экспериментов используются два класса программных пакетов: математические программные пакеты, интегрирующие в себе функции построения изображений (такие, как MathCad, MatLab, Mathematica, Maxima и др.) и пакеты, ориентированные исключительно на визуализацию (TecPlot, Origin, EasyPlot, IRIS Explorer, Surfer, Grapher, Voxler, Hesperus, ParaView, Avizo и др.) [13, 37, 38, 39]. Также в ряде случаев возможно использование средств визуализации систем автоматизированного проектирования (САПР), таких, как AutoCAD, Autodesk Inventor, ArchiCAD, DraftSight и др., см., например, [40].

Однако пакеты, интегрирующие в себе вычислительные инструменты и средства визуализации, не являются достаточно гибкими для решения с их помощью произвольных задач построения изображений. Они, как правило, плохо масштабируются в условиях высокопроизводительных вычислительных систем, так как ориентированы в основном лишь на настольные компьютеры, и, кроме того, предоставляют достаточно ограниченный спектр графических возможностей (чаще всего – построение диаграмм, графиков и поверхностей).

С другой стороны, пакеты, служащие исключительно для визуализации, как правило, не поддерживают автоматическую интеграцию с решателем. Управление решателем, получение исходных данных для построения изображения и приведение их к формату, приемлемому для данного вида систем (если решатель генерирует данные в некотором проприетарном, а не открытом формате), целиком ложится на пользователя или разработчика.

Так, например, высокоэффективная система интерактивной научной визуализации Modulight [41], поддерживающая распределённую визуализацию и

работу с решателями на высокопроизводительных вычислительных комплексах, требует, однако, написания дополнительного программного кода коммуникационных модулей для каждого нового решателя.

Примером подхода интеграции на уровне формата входных данных может служить одна из первых систем научной визуализации, разработанных в России, – система контекстной визуализации пространственных данных Visualizer [18], предоставляющая широкие графические возможности для отображения разнородных результатов измерений и математического моделирования в контексте среды обитания человека. Она входит в состав более крупной системы высококачественной фотореалистичной визуализации, включающей также программу математического моделирования распределения температуры и движения воздуха Flow [18]. Для подключения сторонних решателей (вместо Flow) требуется создавать конвертеры их выходных данных во входной формат системы Visualizer.

В некоторых случаях, для интеграции систем визуализации со сторонними решателями используется стороннее ПО. Примером такого ПО может служить библиотека FlowVR [42], предоставляющая функции для организации взаимодействия систем визуализации и систем виртуальной реальности, функционирующих на высокопроизводительных вычислительных комплексах. Однако эта библиотека не содержит высокоуровневого интерфейса для пользователя.

Частично автоматизация настройки на сторонние решатели достигнута в системе визуализации данных в виртуальных лабораториях VS-Sci [29], где используется модельно-ориентированный подход к описанию специфики источника данных (решателя или базы данных) и реализованы средства автоматической генерации Web-интерфейса управления графической сценой в зависимости от особенностей структуры данных. Однако данная система ориентирована только на двумерную визуализацию (построение графиков и диаграмм) и не поддерживает работу на мобильных устройствах.

В случае использования САПР проблема адаптации исходных данных становится ещё более острой, так как этот класс программного обеспечения, вообще говоря, служит для решения иных задач (таких, например, как построение чертежей и проведение сопутствующих расчётов).

Кроме того, общей проблемой всех вышеописанных программных пакетов является очень ограниченное число версий, работающих на мобильных устройствах. Существуют мобильные аналоги систем математических вычислений (PocketCAS [43], GraphCalc [44] и др.), однако чаще всего возможности встроенных в них визуализаторов также ограничены лишь построением графиков и поверхностей.

Детальный анализ наиболее популярных магазинов программного обеспечения для мобильных устройств App Store и Google Play Store показал почти полное отсутствие систем научной визуализации для мобильных устройств под управлением iOS и Android. Самой богатой функциональностью и высокой производительностью из найденных программных продуктов обладает KiwiViewer [45], который реализован для обеих указанных платформ. KiwiViewer активно развивается компанией KitWare и предоставляет достаточно широкие возможности эффективной визуализации больших объёмов данных, однако при этом не имеет средств автоматической интеграции с решателем (все данные, необходимые для визуализации, должны загружаться на мобильное устройство либо с настольного компьютера, либо с файлового сервера). Получение данных от решателя и адаптацию их к формату, пригодному для работы системы визуализации, должен осуществлять пользователь.

Результаты сравнительного анализа различных приложений научной визуализации представлены в табл. 1. Знак «+» означает присутствие указанного свойства, «-» – отсутствие. Под «базовой 3D-визуализацией» понимается возможность визуализации трёхмерных моделей и построения поверхностей уровня. Под «сложной 3D-визуализацией» понимается поддержка рендеринга объёмов и сечений.

Таблица 1. Сравнительный анализ приложений научной визуализации

Название	Открытый код	Поддерживаемые ОС					Эффективный параллелизм	Базовая 3D-визуализация	Сложная 3D-визуализации
		Windows	GNU/Linux	OS X	iOS	Android			
MathCad	-	+	-	-	-	-	-	-	-
MatLab	-	+	+	+	-	-	-	+	-
Mathematica	-	+	+	+	-	-	+	+	-
Maxima	+	+	+	+	-	-	-	-	-
TecPlot	-	+	+	+	-	-	-	+	+
Origin	-	+	-	-	-	-	-	-	-
EasyPlot	-	+	-	-	-	-	-	-	-
IRIS Explorer	-	+	+	+	-	-	-	+	+
Surfer	-	+	-	-	-	-	-	+	-
Grapher	-	+	-	-	-	-	-	-	-
Voxler	-	+	-	-	-	-	-	+	+
Hesperus	-	+	-	-	-	-	-	-	-
ParaView	+	+	+	+	-	-	+	+	+
Avizo	-	+	+	+	-	-	-	+	+
Modulight	-	+	+	+	-	-	+	+	+
Visualizer	-	+	+	+	-	-	-	+	+
VS-Sci	-	+	-	-	-	-	-	-	-
AutoCAD	-	+	-	+	+	+	-	+	+
Autodesk Inventor	-	+	-	-	-	-	-	+	+
ArchiCAD	-	+	-	+	-	-	-	-	-
DraftSight	+	+	+	+	-	-	-	+	-
PocketCAS	-	-	-	-	+	-	-	-	-
GraphCalc	-	-	-	-	+	-	-	-	-
KiwiViewer	+	-	-	-	+	+	-	+	+

На основе проведённого анализа сделан вывод о том, что на сегодняшний день достаточно малое число систем научной визуализации предоставляет средства эффективной интеграции со сторонними решателями. Кроме того, было

найдено очень мало систем визуализации, совместимых с мобильными устройствами. Таким образом, на основе проведённого исследования можно сделать вывод, что разработка мультиплатформенных систем научной визуализации, поддерживающих автоматизированную интеграцию с разнородными решателями, является актуальной задачей.

1.4.2. Библиотеки для научной визуализации

Для разработки систем научной визуализации существует целый ряд готовых библиотек и модулей. Наиболее популярными являются OpenDX, VTK, VizIt и ScientificVR. Подробный обзор этих программных средств приведён в диссертационной работе О. В. Джосан [27]. Согласно её исследованию, самой богатой функциональностью среди указанных библиотек обладает VTK [46]. Данная библиотека распространяется по свободной лицензии BSD, реализована на языке C++, её API доступен в различных языках программирования (Tcl, Perl, Python и Java). В качестве подсистемы вывода графики используются библиотеки стандарта OpenGL.

VTK ориентирована на настольные компьютеры, однако имеет ещё две модифицированных версии: pVTK и VES. pVTK обеспечивает поддержку параллельного рендеринга, что позволяет осуществлять эффективную визуализацию в условиях высокопроизводительной вычислительной системы. VES предназначена для мобильных устройств и использует в качестве подсистемы вывода графики библиотеки стандарта OpenGLES.

На основе библиотеки pVTK разработана свободно распространяемая кроссплатформенная система научной визуализации ParaView [38], обладающая удобным графическим интерфейсом пользователя и ориентированная на использование в высокопроизводительных вычислительных системах. ParaView имеет клиент-серверную архитектуру, что позволяет выполнять удалённую визуализацию. Клиентская часть данной системы может работать под управлением большинства операционных систем для настольных компьютеров.

Упомянутая ранее система KiwiViewer реализована на основе библиотеки VES и в каком-то смысле может считаться мобильной версией системы ParaView, хотя по факту реализует лишь базовую её функциональность. Однако, начиная с версии 2.0, KiwiViewer имеет средства прямой интеграции с ParaView (позволяет присоединяться к компьютеру, на котором установлена ParaView, и обмениваться с ней данными).

Также существует достаточно большое количество библиотек, предназначенных для построения диаграмм, графиков и поверхностей, ориентированных на применение в задачах визуализации научных данных. Например, Chaco [47], PyQwt [48], PyX [49], Matplotlib [50], MathGL [51] и др. Наиболее богатой функциональностью обладает MathGL, предоставляя среди прочего возможность построения поверхностей, изолиний и векторных полей с использованием аппаратного ускорения графики. Однако ориентированность на традиционные методы визуализации в данных библиотеках является их слабым местом. Так, например, в них отсутствует возможность отображения трёхмерных моделей с анимацией. Кроме того, для данных библиотек отсутствуют версии для мобильных устройств.

Результаты сравнительного анализа различных библиотек научной визуализации представлены в табл. 2. Использована нотация, аналогичная нотации, приведённой в табл. 1.

Адекватными программными средствами для организации поддержки сложной 3D-визуализации в мультиплатформенных системах научной визуализации являются библиотеки семейства VTK: библиотека для настольных компьютеров (VTK), для мобильных устройств (VES) и библиотека, обеспечивающая эффективный рендеринг в условиях высокопроизводительных вычислительных систем (pVTK).

Таблица 2. Сравнительный анализ библиотек научной визуализации

Название	Открытый код	Поддерживаемые ОС					Эффективный параллелизм	Базовая 3D-визуализация	Сложная 3D-визуализации
		Windows	GNU/Linux	OS X	iOS	Android			
OpenDX	+	+	+	+	-	-	+	+	+
VTK	+	+	+	+	+	+	+	+	+
VizIt	+	+	+	+	-	-	+	+	+
ScientificVR	-	+	+	+	-	-	-	+	+
Chaco	+	+	+	+	-	-	-	-	-
PyQwt	+	+	+	+	-	-	-	+	-
PyX	+	+	+	+	-	-	-	+	-
Matplotlib	+	+	+	+	-	-	-	+	-
MathGL	+	+	+	+	-	-	-	+	-

1.4.3. Модули графического расширения

Для визуализации трёхмерных и двумерных сцен могут быть использованы различные модули графического расширения, среди которых существует большое количество кроссплатформенных и мультиплатформенных решений. Примерами таких модулей служат Unreal Engine [53], Unity3D [54], SIO2 [55], OGRE [56], Irrlicht [57], Oolong [58], Cocos2D / Cocos3D [59], libGDX [60] и др. Все перечисленные модули являются мультиплатформенными, однако в основном ориентированы на создание игровых приложений и виртуальных миров. Как уже отмечалось выше, игровые приложения отличаются от систем научной визуализации тем, что, в основном, используют легковесные данные (оптимизированные низкополигональные модели персонажей и окружения), но при этом ориентированы на высокую динамичность сцены и обилие визуальных спецэффектов. Приложения научной графики, напротив, принимают на вход большие объёмы данных и сложные структуры, визуализация которых осуществляется с использованием специфических алгоритмов (основанных на построении сечений, проекций, представлении структур в разных масштабах и

уровнях детализации и т. д.). Чаще всего, сложные визуальные спецэффекты в научной графике не так важны и даже могут помешать работе исследователя. Исключением является только та ситуация, когда подобного рода эффекты необходимы для адекватного отражения специфики научного эксперимента. В связи с этим, на основе модулей графического расширения, ориентированных на разработку игровых приложений, не всегда удаётся создать эффективную систему научной визуализации.

Несмотря на это, в ряде случаев описанные средства всё-таки используются для решения задач научной визуализации. Так, например, OGRE был использован для моделирования биологических клеток [56].

Результаты сравнительного анализа различных модулей графического расширения представлены в табл. 3. Использованная нотация аналогична табл. 1.

Ввиду отсутствия поддержки средств визуализации сечений и объёмов у всех изученных модулей графического расширения, принято решение не использовать их при создании системы научной визуализации, остановив выбор на семействе библиотек VTK.

Таблица 3. Сравнительный анализ модулей графического расширения

Название	**Открытый код**	**Поддерживаемые ОС**					**Эффективный параллелизм**	**Базовая 3D-визуализация**	**Сложная 3D-визуализации**
		Windows	**GNU/Linux**	**OS X**	**iOS**	**Android**			
Unreal Engine	-	+	+	+	+	+	-	+	-
Unity3D	-	+	+	+	+	+	-	+	-
SIO2	-	+	+	+	+	+	-	+	-
OGRE	+	+	+	+	+	+	-	+	-
Irrlicht	+	+	+	+	+	+	-	+	-
Oolong	+	-	-	-	+	+	-	+	-
Cocos2D / 3D	+	-	-	-	+	-	-	+	-
libGDX	+	+	+	+	+	+	-	+	-

1.4.4. Средства создания графического интерфейса пользователя

При создании интерактивной системы визуализации важную роль играют не только средства рендеринга, но и пользовательский интерфейс. Для организации мультиплатофрменности этот интерфейс следует разрабатывать с использованием какой-либо библиотеки, легко адаптируемой под все целевые платформы.

Наиболее популярными библиотеками, предоставляющими средства построения графического интерфейса пользователя, являются Qt [61], GTK [62], Tk [63], Awt [64], Swing [65] и .NET / Mono [66].

GTK и Tk имеют реализации только под операционные системы для настольных компьютеров. Awt и Swing используются для создания интерфейсов Java-приложений, однако только для Java-машин, работающих на настольных компьютерах.

Для Qt помимо реализации под операционные системы для настольных компьютеров существуют также версии под операционные системы iOS (Qt-iPhone [67]) и Android (Necessitas [68] и Boot To Qt [69]). Однако в настоящий момент имеется ряд проблем со стабильностью и эффективностью работы этих библиотек, в особенности при комбинировании их с низкоуровневыми средствами вывода графики, например, с библиотеками стандарта OpenGL(ES). Так, например, скорость визуализации тестовой сцены, состоящей из $4{,}8 \cdot 10^6$ вершин, на устройстве iPad 3 без использования элементов интерфейса составляет 15 кадров в секунду (*англ.* frames per second, FPS), а при использовании элементов интерфейса библиотеки Qt-iPhone опускается до 8 FPS. Результат визуализации сцены представлен на рис. 2. Поскольку мультиплатформенные подсистемы рендеринга, как правило, основываются именно на OpenGL(ES), применимость Qt оказывается ограниченной.

.NET является набором библиотек и программным окружением для создания приложений для ОС Windows от компании Microsoft. Портирование приложений, созданных с использованием .NET, на другие платформы возможно при помощи наборов библиотек Mono от компании Xamarin. Для мобильных устройств

существует версия этого набора библиотек под названием MonoTouch. Языком реализации программ, использующих данные библиотеки, является C#. При этом в .NET и Mono присутствует возможность использования кода, написанного на C++, который обеспечивал бы высокую эффективность критичных ко времени мест, включая алгоритмы визуализации. Однако MonoTouch является платным продуктом, стоимость лицензии на одного сотрудника организации составляет $999 в год. Кроме того, весь код, отвечающий за создание графического интерфейса пользователя в программах для iOS и Android является платформенно-зависимым, то есть отсутствуют универсальные решения для разных платформ. В контексте создания интерфейсов MonoTouch выступает всего лишь обёрткой на языке C# для нативных средств создания графического интерфейса пользователя. В связи с этим он не полностью отвечает сформулированным в диссертационной работе требованиям мультиплатформенности.

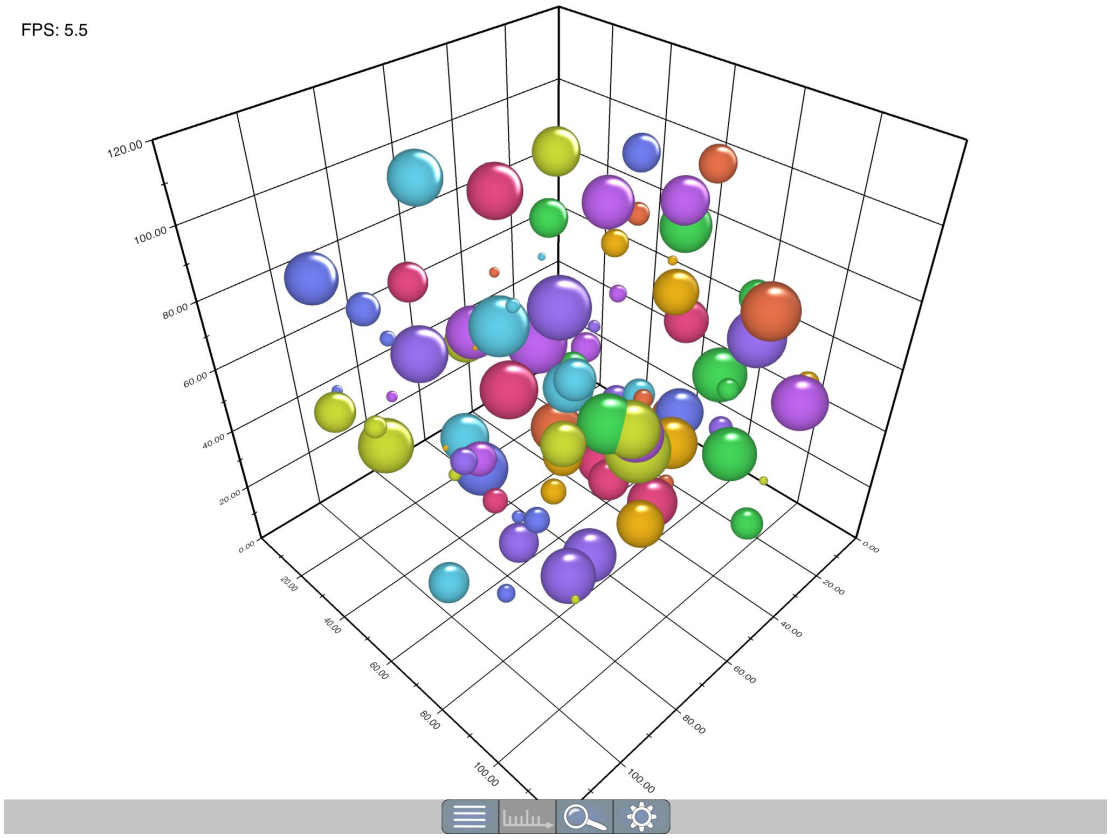


Рис. 2. Тестовая сцена, состоящая из $4{,}8 \cdot 10^6$ вершин, с использованием освещения по Фонгу, отображаемая в разработанной системе научной визуализации.

В контексте создания мультиплатформенных графических интерфейсов пользователя возникает ещё и так называемая «проблема двойного дизайна»: даже используя одну и ту же библиотеку элементов управления, программист вынужден проектировать и реализовывать различные интерфейсы для настольных компьютеров и для мобильных устройств. Такая необходимость вытекает из того, что эти два класса платформ используют различные парадигмы управления и отображения данных, как показано в разделе 1.3 данной главы. Например, основными средствами интерактивного взаимодействия с программой на настольном компьютере являются клавиатура и мышь, а на мобильном устройстве – сенсорный экран и управление при помощи жестов. Также различия заключаются и в дисплеях: экран мобильного устройства, как правило, значительно меньше, чем экран компьютера, и, следовательно, информация на нём должна располагаться иным образом, чтобы обеспечить требуемую эргономику.

Одним из возможных способов решения проблемы двойного дизайна и, как следствие, достижения мультиплатформенности, является разработка Web-приложений. Наиболее популярными на сегодняшний день технологиями в этой области являются HTML 5 (совместно с использованием JavaScript и CSS) [70] и Flash [71]. Обе эти технологии предоставляют программисту достаточно широкие возможности, однако приложения, созданные на их базе, выполняются внутри браузера, то есть в некоторой дополнительной программной среде. Это порождает целый ряд проблем, связанных с производительностью. В настоящее время поддержка указанных технологий со стороны браузеров активно развивается, и, соответственно, растёт сложность самих Web-приложений. Но в контексте научной графики, где имеют место большие объёмы данных и сложные алгоритмы визуализации, производительность оказывается пока ещё недостаточно высокой. Особенно остро проблема производительности встаёт на мобильных устройствах, где возможности браузеров ощутимо меньше, чем на настольных компьютерах. Так, например, под управлением iOS в настоящее время не поддерживается аппаратно-ускоренная графика средствами технологии WebGL. Под управлением

Android WebGL поддерживается только в мобильной версии браузера Google Chrome [72].

Несмотря на это, существуют достаточно перспективные проекты, например MoSync [73], в которых программисту предоставляется возможность создавать интерфейс мобильного приложения на HTML 5, а логику реализовывать на языке C++. Получаемое приложение может выполняться под управлением практически любой из популярных операционных систем для мобильных устройств. При этом, за счёт использования кода на C++, может быть достигнуто высокое быстродействие. Однако в контексте решения задач научной графики критичной является также скорость визуализации. Таким образом, при использовании MoSync (либо иных средств, основанных на HTML 5 или Flash), интерфейс, выполняющийся в браузере, является узким местом. Кроме того, MoSync ориентирован только на мобильные устройства, тогда как для достижения мультиплатформенности на стороне клиента необходима поддержка также и настольных компьютеров.

Для организации эффективного отображения графического интерфейса пользователя совместно со сложным рендерингом научных данных следует ориентироваться на использование модулей, основанных на стандарте OpenGL(ES). Тогда рендеринг сцены и интерфейса будет осуществляться одними и теми же низкоуровневыми средствами, что позволит избежать необходимости переключения между различными API, повысив тем самым эффективность реализации. Существует модуль графического расширения Clutter [74], служащий для создания интерфейсов на основе OpenGLES. Однако этот модуль не является в достаточной степени распространённым (он используется в операционной системе MeeGo), и его работоспособность под управлением популярных платформ ограничена.

Результаты сравнительного анализа различных средств создания графического интерфейса пользователя представлены в табл. 4. Использована нотация, аналогичная нотации для табл. 1.

Таблица 4. Сравнительный анализ средств создания графического интерфейса пользователя

Название	Открытый код	Поддерживаемые ОС				
		Windows	GNU / Linux	Mac OS X	iOS	Android
Qt	+	+	+	+	+	+
GTK	+	+	+	+	-	-
Tk	+	+	+	+	-	-
Awt / Swing	+	+	+	+	-	-
.NET / Mono	+	+	+	+	+	+
MoSync	+	+	+	+	+	+
Clutter	+	+	+	+	+	+

По результатам проведённого исследования можно сделать вывод, что рассмотренные средства, позволяющие создавать мультиплаторфменные приложения и графические интерфейсы пользователя в системах научной визуализации, не в состоянии в полной мере удовлетворить требованиям, сформулированным в диссертационной работе. Вследствие этого принято решение разработать собственную библиотеку, на основе которой обеспечивалась бы возможность создания мультиплатформенных приложений и решала проблему двойного дизайна графических интерфейсов, сохраняя при этом возможность эффективной реализации критичных ко времени алгоритмов и рендеринга с аппаратным ускорением графики, а также адаптация к особенностям различных платформ.

1.5. Выводы по главе

На основе проведённого исследования сделан вывод о том, что основными недостатками большинства популярных среди учёных-исследователей инструментов научной визуализации является отсутствие эффективных средств для:

1. Автоматизации процесса интеграции систем визуализации со сторонними решателями.

2. Организации мультиплатформенности, в частности – платформенно-независимого графического интерфейса пользователя.
3. Распределения процесса визуализации между клиентом и сервером с учётом особенностей инфраструктуры программного обеспечения и вычислительной сети.
4. Сглаживания границ объектов на изображении.

Наличие этих проблем затрудняет процесс автоматизированного создания высококачественных изображений, удовлетворяющих индивидуальным потребностям исследователей, работающих в различных научных сферах. Такие изображения необходимы учёным для представления динамики и результатов их экспериментов в наглядном виде с целью анализа, интерпретации, тиражирования (в том числе в контексте междисциплинарных исследований), поиска закономерностей, дефектоскопии и т. д. Следовательно, решение этих проблем является на сегодняшний день актуальной научной задачей.

Сформулированы следующие требования, которым должны удовлетворять системы научной визуализации, доступные конечным пользователям (учёным-исследователям, работающим в различных научных областях):

1. Поддержка развитых алгоритмов визуализации (отображение графиков, диаграмм, трёхмерных поверхностей, трёхмерных моделей, визуализация сечений и объёмов) и интерактивного взаимодействия со сценой (возможность навигации, просмотра на разных уровнях детализации, настройка скорости аниамции).
2. Совместимость с современными аппаратными средствами ввода-вывода мультимедийной информации, такими как трёхмерные сканеры и дисплеи высокой чёткости.
3. Автоматизированная интеграция на принципах адаптации со сторонними решателями.
4. Автоматическая настройка на особенности программно-аппаратной платформы, мультиплатформенность.

5. Автоматическое распределение процесса визуализации между вычислительными узлами системы.
6. Сглаживание границ объектов для обеспечения высокого визуального качества результирующего изображения.

Изучены особенности мобильных устройств с целью принятия архитектурных решений для организации мультиплатформенности.

Исследованы программные средства научной визуализации, сделан вывод о том, что для организации сложного рендеринга (отображения сечений и объёмов) перспективнее всего использовать свободно распространяемые библиотеки семейства VTK.

Проанализированы программные средства для создания мультиплатформенного графического интерфейса пользователя и сделан вывод о том, что существующие решения не обеспечивают достаточную эффективность и стабильность работы в контексте использования в системах научной визуализации. В связи с чем возникает необходимость разработки собственных библиотек для построения платформенно-независимого графического интерфейса пользователя.

Глава 2. Методы и средства разработки адаптивных мультиплатформенных систем научной визуализации

2.1. Описание модели адаптивных систем научной визуализации

Формально система научной визуализации и механизм её взаимодействия с решателем могут быть представлены следующим образом.

Пусть $I = \{i_1, i_2, \ldots, i_n\}$ – множество входных данных решателя, $n \in \mathbb{N}$, $O = \{o_1, o_2, \ldots, o_m\}$ – множество выходных данных решателя, $m \in \mathbb{N}$. Элементами множеств I и O могут выступать, вообще говоря, объекты различной природы, но чаще всего это числа или строки. Обозначим множество данных за Ξ, тогда $i_a \in \Xi$, $o_b \in \Xi$, $a = \overline{1,n}$, $b = \overline{1,m}$.

Решатель Σ может быть представлен как отображение на множестве данных вида:

$$\Sigma : \Xi \to \Xi, \tag{1}$$

$$O = \Sigma(I). \tag{2}$$

Пусть $\overline{U} = \{u_1, u_2, \ldots, u_k\}$ – множество поддерживаемых системой визуализации графических объектов сцены, $k \in \mathbb{N}$. Элементами множества $\overline{U}$ выступают математические модели визуальных объектов: трёхмерных тел, растровых изображений, диаграмм и т. д. Конкретная сцена представляет собой некоторое подмножество U множества $\overline{U}$.

Пусть $\overline{M} = \{m_1, m_2, \ldots, m_l\}$ – множество поддерживаемых системой визуализации элементов графического интерфейса пользователя, $l \in \mathbb{N}$, причём $m_d = \{\sigma_1^d, \sigma_2^d, \ldots, \sigma_{l_d}^d\}$ – множество состояний, которые может принимать элемент интерфейса m_d, $d \in \overline{1,l}$, $l_d \in \mathbb{N}$ в ответ на действия пользователя. Конкретный набор элементов интерфейса, служащих для управления визуализацией, представляет собой некоторое подмножество M множества $\overline{M}$, $M = \{m_{c_1}, m_{c_2}, \ldots, m_{c_t}\}$, $m_{c_g} \in \overline{M}$, $c_g \in \mathbb{N}$, $g = \overline{1,t}$, $t \in \mathbb{N}$, $t \le l$.

Пусть множество $\overline{S} = \{m_1 \times m_2 \times ... \times m_l\}$ – множество всех состояний, которые могут принимать все элементы интерфейса в ответ на действия пользователя. Тогда множество S' конкретных настроек графической сцены может быть записано как $S' = \{\sigma \mid \sigma \in m_{c_1} \times m_{c_2} \times ... \times m_{c_t}\}$.

Пусть множество $\overline{E} = \{e_1, e_2, ..., e_q\}$ – множество всех возможных действий пользователя, $q \in \mathbb{N}$. Во время очередного обращения к системе визуализации пользователь совершает некоторый набор действий E, являющийся подмножеством $\overline{E}$.

Модуль Γ организации интерактивного взаимодействия пользователя с графическим интерфейсом в системе визуализации может быть представлен как отображение множества всех возможных действий пользователя $\overline{E}$ и всех поддерживаемых элементов интерфейса $\overline{M}$ на множество состояний $\overline{S}$ этих элементов:

$$\Gamma : \overline{E} \times \overline{M} \to \overline{S}. \tag{3}$$

Тогда конкретные настройки сцены, установленные в ответ на конкретные действия пользователя в момент очередного его обращения к системе визуализации могут быть получены применением этого отображения к множеству M:

$$S' = \Gamma(E, M). \tag{4}$$

Множество $\overline{S}'$ всех возможных наборов настроек сцены, в свою очередь, может быть получено объединением вида

$$\overline{S}' = \bigcup_{E' \subset \overline{E}} \Gamma(E', M). \tag{5}$$

Визуализатор Φ может быть представлен как отображение множества всех поддерживаемых визуальных объектов $\overline{U}$ и всех возможных наборов настроек сцены $\overline{S}'$ на растровое изображение P, являющееся матрицей размерности $w \times h$, элементами которой выступают цвета, закодированные при помощи какой-либо цветовой модели (чаще всего – RGB):

$$\Phi : \overline{U} \times \overline{S}' \to P, \tag{6}$$

$$P: W \times H \to C, \tag{7}$$

где $W = \{1, 2, \ldots, w\}$, $w \in \mathbb{N}$,

$H = \{1, 2, \ldots, h\}$, $h \in \mathbb{N}$,

C – множество цветов используемой цветовой модели.

Получение изображения P' со сглаженными границами объектов может быть записано как суперпозиция визуализатора Φ и оператора сглаживания Δ:

$$\Delta: P \to P, \tag{8}$$

$$P' = \Delta(\Phi(\overline{U}, \overline{S}')). \tag{9}$$

Модуль Ψ взаимодействия системы визуализации с решателем может быть представлен как отображение входных и выходных данных решателя на множество визуальных объектов:

$$\Psi: \Xi \to \overline{U}. \tag{10}$$

Когда интеграция происходит лишь с данными, без участия решателя, получение множества U конкретных объектов, подлежащих визуализации можно записать следующим образом:

$$U = \Psi(I \cup O). \tag{11}$$

В случае, если система визуализации взаимодействует с решателем, получение множества U можно представить так:

$$U = \Psi(I \cup \Sigma(I)). \tag{12}$$

Для организации возможности управления решателем при помощи графического интерфейса пользователя системы визуализации необходимо расширить множество M дополнительными элементами. Для простоты записи обозначим новые элементы как отдельное множество N: $N = \{m_{c_{t+1}}, m_{c_{t+2}}, \ldots, m_{c_{t+n}}\}$.

Каждый элемент этого множества соответствует элементу множества I входных данных решателя. Тогда обратную связь системы визуализации с решателем можно записать в следующем виде: $U = \Psi(\Gamma(N) \cup \Sigma(\Gamma(N)))$.

Используя введённые обозначения, настроенную на конкретный решатель систему визуализации Y можно представить следующим образом:

$$Y = \langle \Phi, \overline{U}, \Delta, \Gamma, \overline{M}, \overline{E}, \Psi, M, N \rangle. \quad (13)$$

Составляющие модели $\Phi, \Delta, \Gamma, \overline{M}, \overline{E}, M$ унифицированы и не зависят от решателя:

- Φ – оператор визуализации, отображающий графические объекты множества $\overline{U}$. При необходимости, это множество может быть расширено новыми объектами для более наглядной демонстрации данных конкретного решателя, однако, при унифицированном представлении объектов, логика работы визуализатора при расширении множества $\overline{U}$ не изменится. Визуализатор обеспечивает необходимые способы рендеринга, включая распределение между узлами вычислительной сети с учётом всех необходимых особенностей инфраструктуры программно-аппаратного обеспечения.
- Δ – оператор сглаживания, воздействующий на готовое растровое изображение и порождающий новое изображение со сглаженными границами объектов. Его работа никак не связана ни с решателем. Технически, при конкретных программных реализациях логика его работы может быть связана с логикой работы визуализатора Φ, однако, в силу независимости визуализатора от решателя, оператор сглаживания также остаётся унифицированным и обеспечивает высокое визуальное качество итогового изображения.
- Γ – оператор для организации графического интерфейса, позволяющий пользователю задавать различные настройки при помощи графических элементов управления безотносительно к логике дальнейшего использования значений этих настроек. Данный оператор также поддерживает автоматическую генерацию платформенно-независимого интерфейса для поддержки работы систем на настольных компьютерах и мобильных устройствах.
- $\overline{M}$ – множемтво унифицированных элементов управления, таких, как кнопки, текстовые поля, текстовые метки, переключатели и т. д.

- $\overline{E}$ – множество поддерживаемых действий пользователя с элементами множества $\overline{M}$.
- M – однократно определённое множество элементов управления, необходимых для навигации по отображаемым сценам.
- Компоненты модели $\overline{U}, \Psi, N$ в общем случае зависят от решателя:
- $\overline{U}$ – множество объектов, которые служат наглядному представлению данных решателя.
- Ψ – оператор, осуществляющий конвертацию входных и выходных данных решателя в объекты, пригодные для визуализации модулем Φ.
- N – множество элементов управления, обеспечивающих обратную связь с решателем.

Для настройки системы визуализации на сторонний решатель необходимо определить множества $\overline{U}$, N и отображение Ψ.

Множество $\overline{U}$ является повторно используемым и в качестве него может быть использована пополняемая база знаний о визуальных объектах. Ключевое требование к его элементам – это унификация представления, чтобы при его расширении не возникло необходимости модифицировать программный код визуализатора. Для решения задачи унифицированного хранения, лёгкости расширяемости, удобства интерпретации, тиражируемости и, в то же время, обеспечения семантической мощности описаний принято решение использовать в качестве модели хранения знаний онтологии [75].

2.2. Применение методов онтологического инжиниринга для разработки адаптивных систем научной визуализации

Проведено исследование применимости методов онтологического инжиниринга к задаче описания визуальных объектов, используемых в системах научной визуализации. Элементы множества $\overline{U}$ связаны друг с другом в отношении родитель-потомок (с наследованием свойств), часть-целое и класс-экземпляр. В связи с этим сделан вывод о достаточности прикладной онтологии без аксиоматики для решения задач, поставленных в диссертационном

исследовании. Таким образом, $\overline{U} = \langle T_{\overline{U}}, R_{\overline{U}} \rangle$, где $T_{\overline{U}}$ – тезаурус визуальных объектов, а $R_{\overline{U}}$ – конечное множество связей между визуальными объектами.

Множество N может быть получено путём сопоставления типов элементов множеств I и $\overline{M}$ при помощи оператора Π выбора элементов из множества $\overline{M}$, пригодных для редактирования соответствующих данных:

$$\Pi : \Xi \times \overline{M} \to \overline{M}, \tag{14}$$

$$N = \Pi(I, \overline{M}). \tag{15}$$

Практическая реализация такого оператора в виде программного модуля является тривиальной задачей сопоставления.

Отображение Ψ может быть получено при помощи оператора Ω вида

$$\Omega : \Xi \times \overline{U} \to \overline{\Psi}, \tag{16}$$

где $\overline{\Psi}$ – множество операторов конвертации.

В этом случае:

$$\Psi = \Omega(I \cup O, \overline{U}). \tag{17}$$

В программной реализации системы визуализации отображение Ψ может быть сгенерировано в полуавтоматическом режиме, после того, как пользователь указал соответствие элементов множеств I и O свойствам объектов из множества $\overline{U}$. Подобное действие может осуществляться пользователем без программирования, путём сопоставления объектов в визуальной среде.

Таким образом, задача интеграции системы визуализации со сторонним решателем может быть решена в том случае, если известны множества входных и выходных данных решателя. Эти множества могут быть получены двумя основными способами: заданы пользователем (для этого также может быть использован высокоуровневый графический интерфейс), либо восстановлены путём автоматического анализа исходного кода решателя.

Пусть K_Σ – исходный код решателя Σ. Структуру входных и выходных данных решателя Σ может восстановить отображение вида:

$$\Theta : K_\Sigma \to \Xi, \tag{18}$$

$$D = \Theta(K_\Sigma), \tag{19}$$

$$D \equiv \langle I, O \rangle. \tag{20}$$

Для унификации программной реализации оператора Θ также может быть использован подход на основе онтологического инжиниринга. Оператор будет модифицирован следующим образом:

$$\Theta: K_\Sigma \times \overline{L} \to \Xi. \tag{21}$$

где $\overline{L}$ – онтология, описывающая синтаксические конструкции операторов ввода-вывода языков программирования.

В формуле (21) Θ обозначает унифицированный синтаксический анализатор, который управляется онтологией $\overline{L}$. За D_I далее будем обозначать структуру входных данных, а за D_O – структуру выходных данных, восстановленные при помощи оператора Θ.

Проведено исследование применимости методов онтологического инжиниринга к задаче синтаксического разбора исходного программного кода на заданном языке с целью извлечения входных и выходных переменных. Для автоматической генерации синтаксического анализатора необходимо представить формы Бэкуса-Наура (БНФ) этого языка в виде онтологии. Причём, в ходе диссертационного исследования выявлено, что можно ограничится лишь описанием конструкций ввода-вывода. Для представления БНФ, так же, как и для представления визуальных объектов, оказывается достаточно прикладной онтологии без аксиоматики. Таким образом, $\overline{L} = \langle T_{\overline{L}}, R_{\overline{L}} \rangle$, где $T_{\overline{L}}$ – тезаурус конструкций, а $R_{\overline{L}}$ – конечное множество связей между ними.

В результате проведённых исследований принято решение построить систему научной визуализации на базе модельно-ориентированного подхода с использованием принципов онтологического инжиниринга, чтобы организовать поддержку автоматизированной интеграции на принципах адаптации со сторонними решателями. Общая формальная модель такой системы может быть записана в следующем виде:

$$Y^* = \langle \Phi, \overline{U}, \Delta, \Gamma, \overline{M}, \overline{E}, \Theta, \overline{L}, \Omega, \Pi, M \rangle. \quad (22)$$

Интеграция системы со сторонним решателем Σ при наличии его исходного кода K_Σ может быть записана как

$$D(K_\Sigma) = \Theta(K_\Sigma, \overline{L}), \quad (23)$$

$$Y = \langle \Phi, \overline{U}, \Delta, \Gamma, \overline{M}, \overline{E}, \Omega(D_I(K_\Sigma), D_O(K_\Sigma), \overline{U}), M, \Pi(D_I(K_\Sigma), \overline{M}) \rangle. \quad (24)$$

По составу множество (23) эквивалентно множеству (13). То есть, при помощи введённых операторов и при наличии исходного кода решателя обобщённая система научной визуализации Y^* сводится к настроенной на решатель Σ системе Y.

При отсутствии исходного кода решателя, от пользователя требуется описать структуры входных и выходных данных D_I и D_O вручную, для чего система предоставляет высокоуровневый графический интерфейс. В этом случае интеграция системы со сторонним решателем Σ будет выглядеть как

$$Y = \langle \Phi, \overline{U}, \Delta, \Gamma, \overline{M}, \overline{E}, \Omega(D_I, D_O, \overline{U}), M, \Pi(D_I, \overline{M}) \rangle. \quad (25)$$

В рамках диссертационной работы предложены методы и средства разработки систем научной визуализации, удовлетворяющие сформулированным в требованиям, описанным в главе 1. На основе этих методов и средств реализована клиент-серверная система научной визуализации, названная SciVi [76, 77, 78, 79, 80, 81, 82, 83] и внедрённая в пермской IT-компании ООО «Ньюлана». Она построена на принципах архитектуры клиент-сервер, что даёт возможность организовать её взаимодействие со сторонними решателями – программами (или даже программно-аппаратными комплексами), которые генерируют данные, подлежащие отображению. Клиент SciVi может выполняться как на настольном компьютере, так и на мобильном устройстве, поддерживаются ОС GNU / Linux, Windows, OS X, iOS и Android. При необходимости не составит труда перенести клиентскую часть системы на любую другую ОС, для которой существует реализация библиотек стандарта OpenGL(ES), поскольку этот стандарт используется для организации рендеринга в SciVi. Сервер SciVi может

выполняться как на настольном компьютере, так и на высокопроизводительном вычислительном комплексе под управлением UNIX-подобной ОС. Настройка на сторонний решатель осуществляется в автоматизированном режиме при помощи высокоуровневых средств, основанных на онтологическом инжиниринге. Высокая интерактивность отображаемых системой сцен достигается путём адаптивного распределения процесса рендеринга между клиентом и сервером. Высокое качество итогового изображения достигается применением метода адаптивного сглаживания границ объектов.

Пользователь (научный сотрудник, исследователь) при помощи специальных высокоуровневых средств может зарегистрировать в системе SciVi свой решатель (программу, генерирующую подлежащие визуализации данные), а затем посредством клиента управлять работой этого решателя (запускать, останавливать решатель и изменять его входные данные) и просматривать результаты визуализации данных, получаемых от решателя.

Предлагаемые методы и средства разработки систем научной визуализации будут описаны далее на примере практической реализации системы SciVi.

2.3. Методы интеграции со сторонними решателями

Основной задачей систем научной визуализации является построение изображений на основе данных, которые генерируют сторонние решатели. Кроме того, важной задачей является является предоставление графического интерфейса с этим решателем, чтобы обеспечить для пользователя удобную среду, в которой он мог бы управлять визуальной сценой и генератором данных для этой сцены при помощи единого виртуального пульта. Решателями могут выступать сторонние программы и программно-аппаратные комплексы, принимающие на вход и отдающие на выходе наборы данных в некотором формате. Частным случаем этого является ситуация, когда решатель, как самостоятельный модуль, отсутствует, а у пользователя в наличии имеются лишь подлежащие визуализации

данные, сохранённые в некотором формате. Важной задачей является также возможность автономной работы с данным, ранее полученными от решателя.

На основе предлагаемого подхода, процесс интеграции систем научной визуализации со сторонним решателем сводится к автоматической генерации описания этого решателя и шаблона для графической сцены на основе настроек, заданных пользователем посредством высокоуровневого интерфейса. В разработанной системе SciVi за интеграцию отвечает отдельный модуль, написанный на языке Python и использующий для описания решателя и графической сцены методы онтологического инжиниринга. В приведённой ранее формальной модели системы научной (22) визуализации этот модуль представлен множеством $\langle \overline{U}, \Theta, \overline{L}, \Omega, \Pi \rangle$. Важной особенностью предлагаемого подхода является отсутствие необходимости изменения исходного кода системы SciVi и исходного кода решателя для осуществления интеграции. Настраиваемость и гибкость интеграционного модуля SciVi обеспечивается использованием расширяемой базы знаний на основе прикладных онтологий без аксиоматики.

Работа пользователя с интеграционным модулем осуществляется через Web-интерфейс, написанный на языке Python с использованием набора библиотек Django [84]. Web-интерфейс предоставляет пользователю личный кабинет, в котором он может зарегистрировать подлежащий визуализации ресурс (решатель или готовые данные), а затем работать с этим ресурсом через клиентское приложение. Функция регистрации не включена в клиентское приложение потому, что клиентом может выступать мобильное устройство, а регистрация ресурса предполагает отправку файлов на сервер. В связи этим пользователю пришлось бы сначала загрузить эти файлы на мобильное устройство, что, во-первых, включило бы процесс настройки дополнительный шаг, а во-вторых, не всегда представляется возможным ввиду потенциально большого объёма данных.

В результате регистрации ресурса интеграционным модулем автоматически генерируются XML-описание, состоящее из следующих частей:

1. Метаданные.

2. Описание интерфейса (может отсутствовать).
3. Шаблон графической сцены.

Метаданные включают в себя:

- название ресурса (решателя);
- расположение исполняемого файла решателя (если имеется);
- параметры командной строки для решателя (если имеются);
- описание структуры входных и выходных данных;
- имена или расположение входного и выходного файлов;
- расположение файла с описанием интерфейса решателя (если имеется);
- метаинформацию в нотации Dublin Core [85].

Название ресурса, исполняемый файл решателя, параметры командной строки, имена входного и выходного файлов (или сами файлы) и метаинформация в нотации Dublin Core запрашиваются у пользователя в вопросно-ответном режиме. Подробнее об использовании стандарта метаданных Dublin Core в SciVi рассказано в разделе 2.4. Описание структуры входных и выходных данных может быть сформировано в полностью автоматическом режиме в том случае, если доступен исходный код решателя (путём автоматического поиска в нём описаний переменных и конструкций ввода-вывода на основе онтологии $\overline{L}$ синтаксических конструкций языков программирования). Синтаксический анализ кода производит часть интеграционного модуля, обозначенная в формальной модели системы научной визуализации (22) как Θ. Правила для разбора исходного кода, написанного на конкретном языке программирования, декларируются онтологией, которая ниже будет рассмотрена более подробно. Если исходный код недоступен, описание структуры входных и выходных данных производится пользователем в вопросно-ответном режиме посредством высокоуровневого графического интерфейса.

Описание интерфейса автоматически строится на основании структуры входных данных решателя. За это построение отвечает часть интеграционного модуля, обозначенная в формальной модели системы научной визуализации (22)

как Π. Таким образом, SciVi позволяет исследователю управлять визуализацией данных и самим научным экспериментом при помощи единого интерфейса. В этом заключается одно из важнейших достоинств предлагаемой концепции интеграции систем научной визуализации с решателями.

Для отображения на стороне клиента графического интерфейса пользователя по сгенерированному описанию используется специально разработанная в рамках диссертационного исследования библиотека GUIBuilder. Её особенности будут подробно описаны в разделе 2.8.2.

Шаблон сцены автоматически генерируется на основании заданных пользователем назначений входных и выходных данных регистрируемого ресурса. За его генерацию, а также за предоставление пользователю необходимого графического интерфейса для указания назначений входных и выходных данных, отвечает часть интеграционного модуля, обозначенная в формальной модели системы научной визуализации (22) как Ω. Этим же программным средством в сгенерированный шаблон затем будут автоматически подставляться конкретные данные, считываемые из входного и выходного файлов решателя. Таким образом, SciVi получает возможность автоматически конвертировать данные научного эксперимента в вид, пригодный для построения сцены. В этом заключается второе достоинство предлагаемой концепции интеграции с решателями.

Предусмотрены следующие режимы работы SciVi:

1. Динамическая визуализация данных, генерируемых в процессе работы решателя:
   1.1. С управлением решателем.
   1.2. Без управления решателем.
2. Визуализация итоговых данных, сгенерированных решателем.

В первом режиме данные для визуализации создаются в динамике, во втором – считываются из заранее подготовленных файлов. В режиме 1.1 пользователь управляет работой решателя через сгенерированный для этого графический интерфейс, в результате чего данные для визуализации динамически

изменяются. Режим 1.2, по сути, сводится к режиму 2, так как принципиальное различие ними состоит только в скорости доступа к данным (генерация данных решателем чаще всего занимает значительно больше времени, чем считывание готовых данных из файлов).

Регистрируя ресурс, пользователь неявным или явным образом определяет, в каком режиме он собирается с ним работать. При регистрации возможны следующие ситуации:

1. Пользователь предоставляет исходный код решателя (и имеется соответствующая онтология $\overline{L}$ языка программирования). В этом случае описание структуры входных и выходных данных будет сформировано автоматически.
2. Пользователь предоставляет исполняемый файл решателя.
   2.1. Пользователь предоставляет готовые входные и выходные данные. В этом случае возможным оказывается использование всех режимов работы SciVi и пользователю задаётся соответствующий вопрос.
   2.2. Пользователь не предоставляет готовых данных. В этом случае возможны режимы работы 1.1 и 1.2 и пользователю задаётся соответствующий вопрос.
3. Пользователь не предоставляет исполняемый файл решателя, но предоставляет готовые входные и выходные данные. В этом случае возможен лишь режим 2, он считается выбранным неявно.

Возможные комбинации ситуаций и соответствующие им режимы работы приведены в табл. 5.

Таблица 5. Режимы работы SciVi

Наличие исполняемого файла решателя	Наличие исходного кода решателя	Наличие готовых данных	Автоматическое формирование описания структуры данных	Режим работы
+	+	+	+	1.1 / 1.2 / 2
+	-	+	-	1.1 / 1.2 / 2
+	+	-	+	1.1 / 1.2

+	-	-	-	1.1 / 1.2
-	+	+	+	2
-	-	+	-	2
-	+	-	не имеет смысла	-
-	-	-	-	-

Первым действием после разворачивании сервера SciVi является загрузка онтологии $\overline{L}$, описывающей конструкции ввода-вывода поддерживаемых языков программирования и онтологии $\overline{U}$ визуальных объектов, описывающей объекты графической сцены. Эти онтологии создаются инженером по знаниям, сохраняются в стандартном формате OWL [86] и загружаются администратором на сервер. Структура и назначение этих онтологий будет более подробно рассмотрена ниже.

Далее администратор регистрирует пользователей, которые затем смогут регистрировать в системе ресурсы (решатели и готовые данные).

Аутентифицированный пользователь получает возможность загрузить на сервер имеющиеся у него ресурсы и определить тем самым режим работы SciVi в соответствии с табл. 5. В качестве ресурса может выступать исполняемый файл решателя или файлы с готовыми данными.

В настоящее время решатель должен удовлетворять следующим ограничениям:

1. Имеется ровно один входной файл с описанием начальных условий эксперимента и один выходной файл с результатами эксперимента.
2. Работа идёт с объектами, набор параметров которых не изменяется в течение эксперимента.
3. Входной файл заполняется инкрементально, то есть ранее вычисленные результаты не изменяются.
4. Не требуется специальная установка на компьютер (решатель представляет собой самостоятельный, готовый к запуску исполняемый файл со всеми необходимыми динамическими библиотеками).

Первое ограничение является чисто техническим и может быть снято добавлением в архитектуру менеджера входных и выходных файлов.

Путём расширения онтологии $\overline{L}$ возможно перейти к поддержке не только других языков программирования, но и других источников данных, помимо файлов, например, к базам данных, потокам ввода-вывода, Web-ресурсам и т. д. Благодаря преимуществу методов онтологического инжиниринга [75], это не приводит к необходимости изменения исходного кода интеграционного модуля, так как, в соответствии с модельно-ориентированным подходом, правила синтаксического анализа кода решателя и извлечения данных для заполнения шаблона сцены генерируются на основе онтологии конструкций ввода-вывода языков программирования.

Для генерации интерфейса с решателем (если решатель имеется в наличии и пользователь выбрал режим работы с управлением) и шаблона сцены необходимо сформировать описание структуры входных и выходных данных.

При наличии исходного кода решателя формирование этой структуры производится в полностью автоматическом режиме путём поиска описаний переменных и выбора из них тех, которые передаются в качестве параметров операторам ввода-вывода. Правила извлечения нужных конструкций из исходного кода декларированы в онтологии конструкций ввода-вывода языков программирования. Фрагмент онтологии $\overline{L}$ в среде Protégé [75] представлен на рис. 3.

Онтология $\overline{L}$ может включать в себя описание произвольного количества языков. Необходимая часть онтологии для работы с конкретным языком определяется на основе расширения файла с исходным кодом. Для каждого языка в данной онтологии декларируются БНФ описания конструкций ввода-вывода, переменных, типов и правил их приведения. Встроенный в систему механизм логического вывода генерирует на основе этих БНФ регулярные выражения, при помощи которых осуществляется извлечение всех переменных с

их типами и всех операторов ввода-вывода из предоставленного исходного кода решателя.

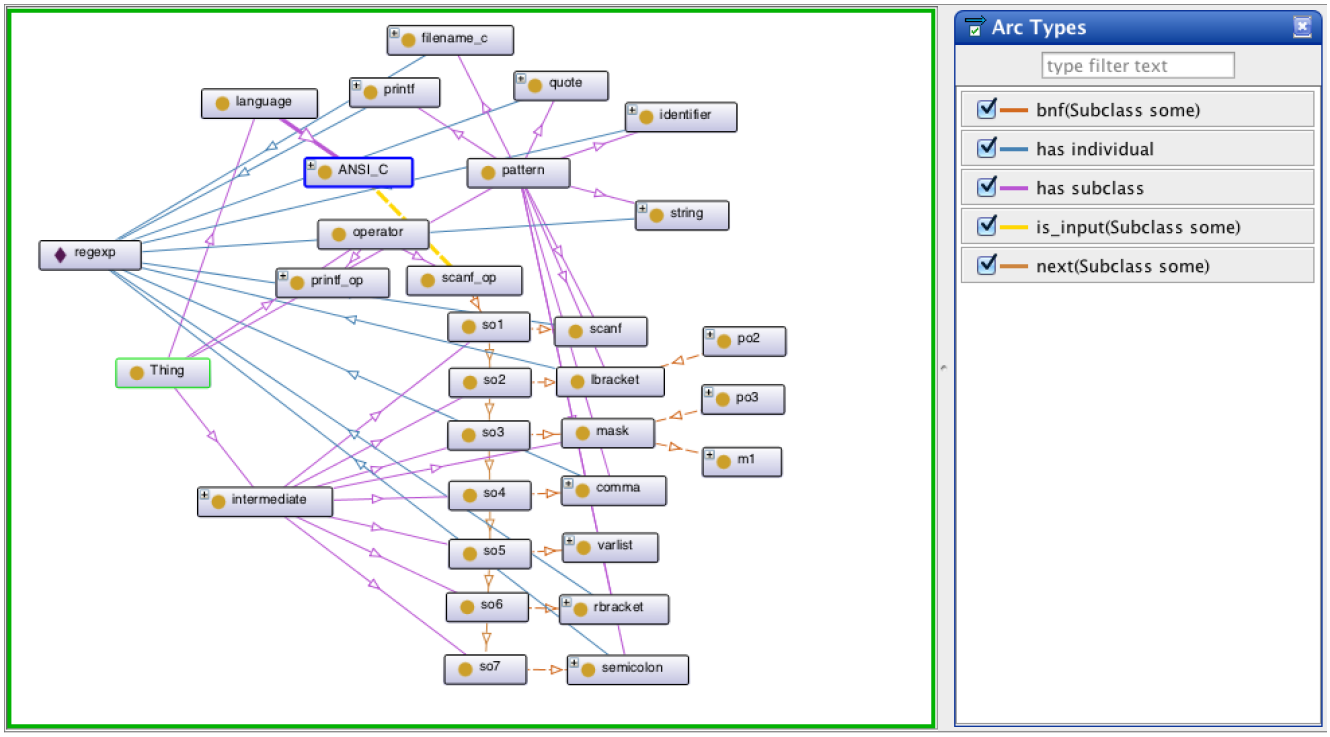


Рис. 3. Фрагмент онтологии L для языка C

Если язык программирования использует статическую типизацию, в итоговое описание структуры данных попадают только те переменные, для которых в исходном коде программы был определён тип и которые встретились в операторах ввода-вывода. Если язык использует динамическую типизацию, после автоматического разбора исходного кода пользователь должен выбрать тип для каждого элемента сформированной структуры данных из списка доступных типов.

Если в коде встретились параметры операторов вывода, не имеющие предварительного описания (то есть роль параметров играют выражения), их типы определяются на основе известных переменных. Для этого производится автоматический разбор найденного выражения с целью извлечения из него переменных и конструкций преобразования типов. Затем, на основе правил приведения типов, происходит определение типа значения. Для операторов ввода

наличие переменной-получателя вводимого значения является обязательным условием, поэтому трудности с определением типа не возникает.

На данный момент в онтологии SciVi описаны необходимые синтаксические конструкции языков программирования Fortran, C и C++. Эти языки были выбраны как наиболее популярные в среде экспериментаторов, разрабатывающих программы моделирования различных процессов. Трудоёмкость добавления поддержки нового языка не высока, так как задача расширения онтологии $\overline{L}$ сравнима с задачей написания для конкретного языка БНФ его конструкций ввода-вывода, описания переменных и приведения типов.

В том случае, если исходный код решателя недоступен, пользователь создаёт описание входных и выходных данных вручную с использованием высокоуровневого графического интерфейса. Форма для описания элемента данных показана на рис. 4.

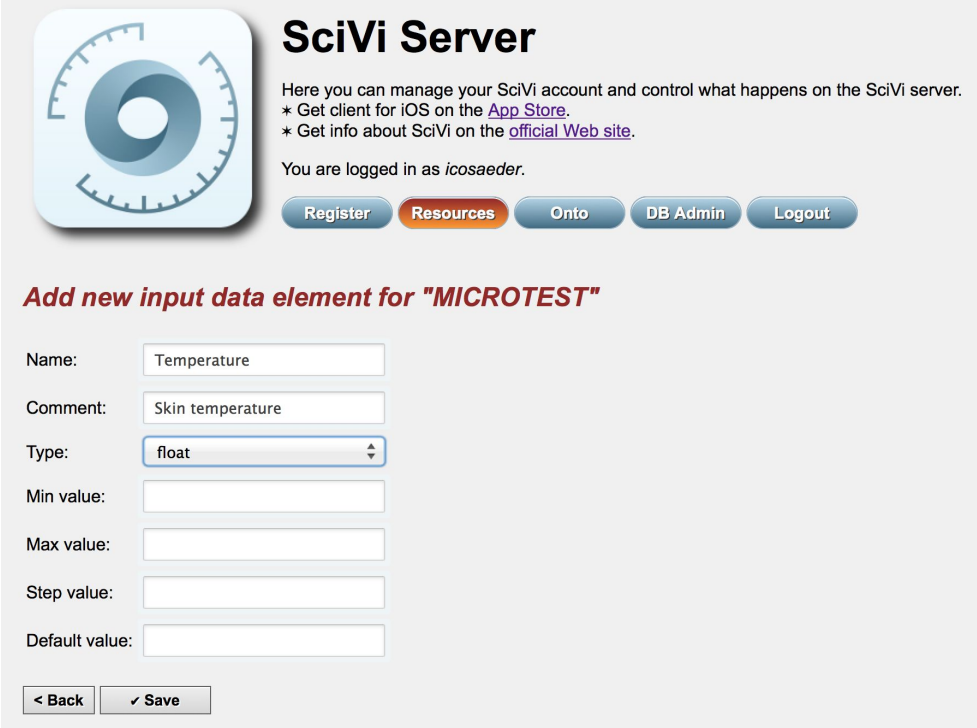


Рис. 4. Добавление описания нового элемента входных данных

На основе сформированного описания структуры входных и выходных данных автоматически создаются регулярные выражения для разбора входного и выходного файлов с данными. Порядок следования элементов структуры в файлах

определяется маской, которая создаётся автоматически во время разбора исходного кода (если он был предоставлен) и может быть отредактирована пользователем. В зависимости от режима работы системы SciVi, входной файл либо генерируется системой для решателя, либо загружается пользователем; выходной файл либо генерируется решателем, либо загружается пользователем. Извлечённые из этих файлов при помощи регулярных выражений значения затем попадают в шаблон сцены в процессе подготовки информации, необходимой для рендеринга.

После того, как структура входных и выходных данных описана, пользователю предлагается настроить визуальную сцену и связать интересующие его элементы данных со свойствами её объектов. На этом этапе используется онтология визуальных объектов, которые могут быть использованы при построении итогового изображения. В формальной модели системы научной визуализации (22) данная онтология обозначена как $\overline{U}$. Она описывает типы экранов (взаимное расположение сцен на экране), типы сцен (контейнеры графических элементов, определяющие их взаимное расположение), элементы сцен (конкретные графические объекты) и их свойства (свойства графических объектов и глобальные свойства сцен и экранов, такие, как цвет фона, модельное время для воспроизведения анимации и т. д.). Фрагмент $\overline{U}$ в среде Protégé представлен на рис. 5.

Пользователь выбирает тип экрана и тип каждой из имеющихся на этом экране сцен (различные виды диаграмм и графиков, одиночная 3D-модель, регулярная трёхмерная сетка с расположенными в узлах 3D-моделями и т. д.), затем выбирает элементы сцен и устанавливает соответствие между их свойствами и элементами структуры входных и выходных данных.

Набор визуальных объектов может быть расширен путём описания новых сущностей или же добавления экземпляров уже описанных сущностей в онтологию $\overline{U}$. Так, например, в онтологии описана сущность «3D-модель», имеющая параметры «положение в пространстве» (координаты x, y, z),

«ориентация в пространстве» (углы поворота вокруг осей *OX*, *OY*, *OZ*, либо вектор направления), масштаб (по осям *OX*, *OY*, *OZ*), текстуру (двумерное изображение), шейдеры (вершинный и фрагментный для определения визуальных свойств материала) и т. д. Пользователь может загружать собственные трёхмерные модели и текстуры для них из файлов и посредством высокоуровневого интерфейса добавлять описания этих моделей в качестве экземпляров сущности «3D-модель», а затем использовать их на сценах. Это не приводит к необходимости внесения изменений в исходный код системы визуализации. В том случае, если пользователю потребуется работать, например, с четырёхмерными моделями, расширение онтологии повлечёт за собой необходимость доработки исходного кода визуализаторов клиента и сервера, однако нужно будет лишь добавлять функциональность, не переписывая ранее отлаженные механизмы.

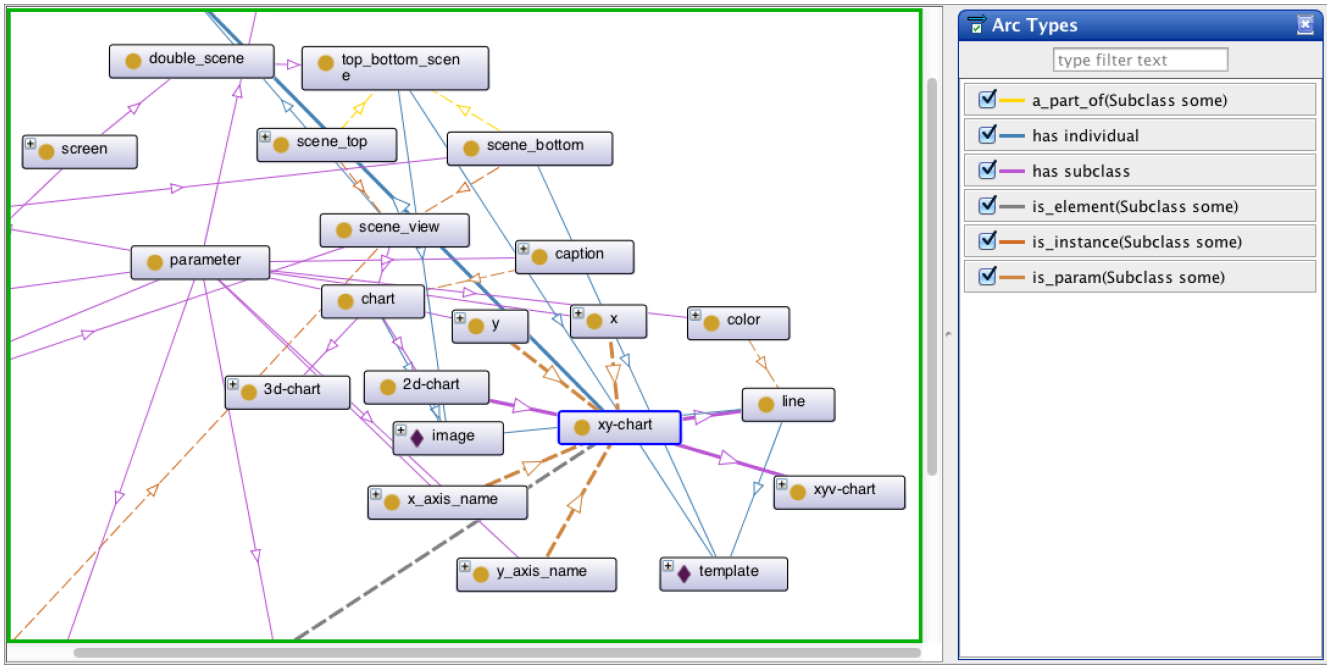


Рис. 5. Фрагмент онтологии $\overline{U}$ визуальных объектов

На основании настроек сцены, сделанных пользователем, SciVi автоматически генерирует шаблон сцены, представляющий собой XML-описание визуальных объектов и их параметров. По запросу клиента этот шаблон автоматически заполняется данными, извлечёнными из входного и выходного файлов решателя. Как уже отмечалось выше, при регистрации ресурса пользователь может предоставить эти файлы в готовом виде, либо же загрузить

исполняемый файл решателя для их динамической генерации. Заполненный данными шаблон (описание готовой сцены) автоматически передаётся модулю визуализации (реализации которого есть как на стороне клиента, так и на стороне сервера SciVi для организации распределённого рендеринга). Модуль визуализации строит по этому описанию итоговое изображение, отображаемое в клиентском приложении.

Процесс настройки системы SciVi на конкретный решатель схематично показан на рис. 6.

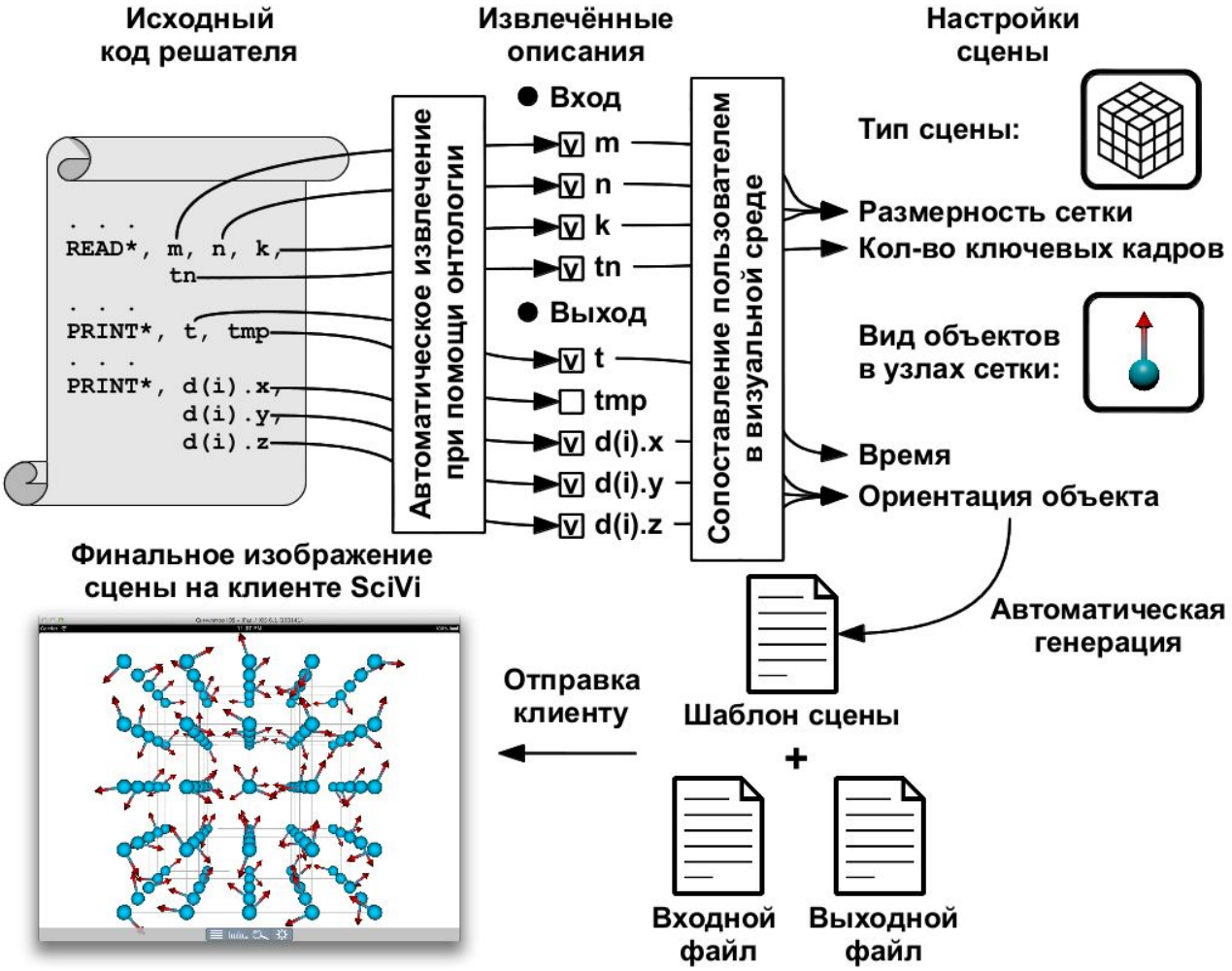


Рис. 6. Процесс интеграции системы SciVi с решателем

На основе исходного кода решателя (если он доступен) при помощи онтологии $\overline{L}$ автоматически восстанавливается структура входных и выходных данных решателя. Затем пользователь посредством высокоуровневого графического интерфейса устанавливает соответствие между интересующими его

элементами этой структуры и свойствами визуальных объектов, описанных в онтологии $\overline{U}$. На основании выполненных настроек, система автоматически генерирует шаблон сцены для визуализации, который в процессе работы решателя автоматически заполняется данными, извлекаемыми из входного и выходного файлов решателя. Сервер SciVi выполняет необходимую предварительную обработку данных, после чего отправляет описание сцены и сопутствующие мультимедийные ресурсы (3D-модели, текстуры и т. д.) клиенту SciVi. На стороне клиента строится итоговое изображение и обеспечивается интерактивное взаимодействие пользователя с графической сценой.

Пример описания сцены в соответствии с принятой в SciVi нотацией приведён в листинге 1. Результат построения изображения по приведённому описанию сцены показан на рис. 7. Полная документация по языку описания сцен приведена в приложении 3. Список поддерживаемых типов сцен приведён в приложении 4.

При помощи основанного на аппарате онтологий интеграционного модуля решается проблема настройки системы SciVi на разнородные решатели, имеющие различные форматы входных и выходных данных. Использование онтологии синтаксических конструкций ввода-вывода языков программирования позволяет автоматизировать процесс описания структуры входных и выходных данных решателя. Использование онтологии визуальных объектов позволяет настраивать систему на решатели из разных предметных областей, управляя внешним видом объектов. Процесс настройки не требует изменения исходного кода решателя. Место интеграционного модуля в архитектуре системы SciVi будет рассмотрено в разделе 2.7.1.

Листинг 1. Пример автоматически сгенерированного описания сцены для визуализации в SciVi

```
<scene background="#ffffff" initialRotX="-0.698131701"
initialRotY="3.40339204">
    <model id="landscape">
        <data
model="http://icosaeder.narod.ru/scivi/suzanne/suzanne.n3d"
```

```
vertex_shader="http://icosaeder.narod.ru/scivi/knot/knot.vsh"
fragment_shader="http://icosaeder.narod.ru/scivi/knot/knot.fsh"/>
        <position rotX="5.5" rotY="3.14"/>
    </model>
</scene>
```

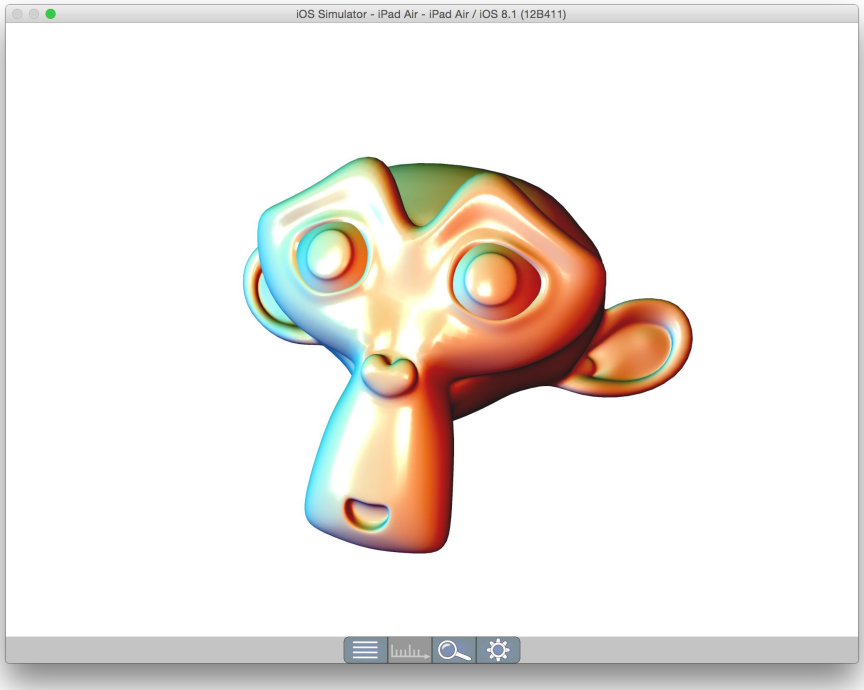


Рис. 7. Изображение, построенное в системе SciVi по описанию сцены из листинга 1

2.4. Использование стандарта Dublin Core

Для стандартизации форматов обмена данными между компонентами системы научной визуализации SciVi используется стандарт Dublin Core [85]. Это набор тэгов, которые могут быть использованы для описания разнообразных ресурсов. Он включает в себя два уровня, «простой» (*англ.* simple) и «квалифицированный» (*англ.* qualified).

«Простой» набор тэгов Dublin Core состоит из 15 элементов:

1. Title – название.
2. Creator – создатель.
3. Subject – тема.
4. Description – описание.
5. Publisher – издатель.

6. Contributor – внёсший вклад.
7. Date – дата.
8. Type – тип.
9. Format – формат документа.
10. Identifier – идентификатор.
11. Source – источник.
12. Language – язык.
13. Relation – отношения.
14. Coverage – покрытие.
15. Rights – авторские права.

«Квалифицированный» набор тэгов добавляет к 15 вышеперечисленным ещё три:

1. Audience – аудитория (зрители).
2. Provenance – происхождение.
3. RightsHolder – правообладатель.

Каждый тэг является опциональным и может повторяться неограниченное количество раз. Порядок следования роли не играет.

Все мультимедийные ресурсы (трёхмерные модели, текстуры, шейдеры и т. д.) аннотированы в соответствии с этим стандартом.

Метаданные в SciVi хранятся в XML-документах, чтобы обеспечить удобство чтения как для человека, так и для компьютера. Используется «квалифицированный» набор тэгов Dublin Core. Технически, каждый XML-файл в системе (например, XML-файл с описанием сцены или решателя) имеет внутри себя блок «dublinCore», в котором сохранены все необходимые тэги стандарта Dublin Core с соответствующей информацией. Каждый бинарный файл имеет соответствующий ему XML-файл с тегами Dublin Core.

Метаданные такого вида необходимы для различных целей, например, для облегчения поиска и переиспользования ресурсов: трёхмерные модели и шейдеры, служащие для повышения визуальной привлекательности сцен, часто

могут быть переиспользованы в различных задачах. Наиболее важными в контексте аннотирования мультимедийных ресурсов тэгами являются Title, Creator, Subject, Description, Publisher, Contributor, Audience и RightsHolder. Они использованы в метаданных для всех ресурсов, которые встретились в задачах, решённых в рамках данного диссертационного исследования.

2.5. Архитектура систем научной визуализации

Предлагаемая архитектура адаптивных мультиплатформенных клиент-серверных систем научной визуализации представлена на рис. 8.

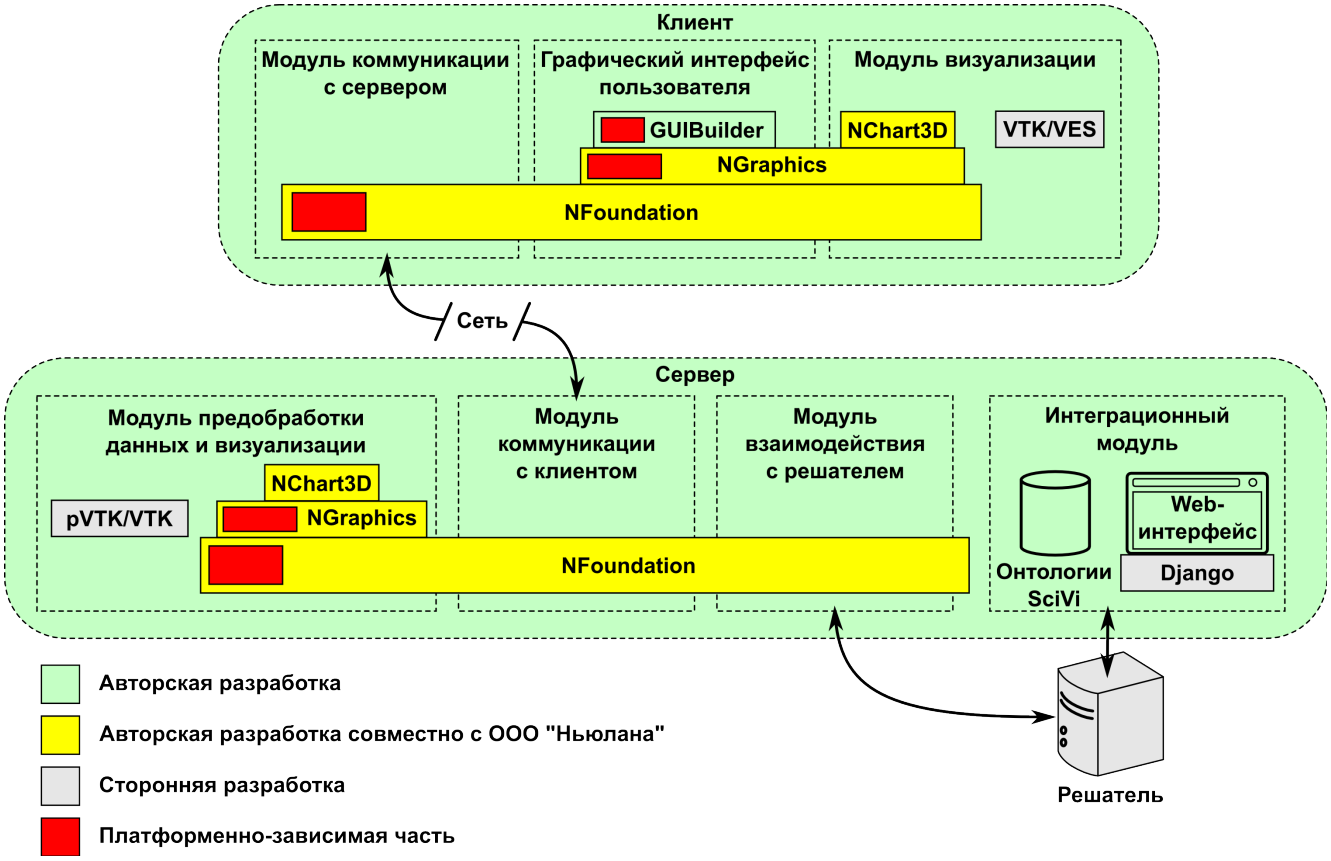


Рис. 8. Предлагаемая архитектура систем научной визуализации

Для организации абстрагирования от специфики конкретных ОС с целью достижения мультиплатформенности используется библиотека функций NFoundation [87], которая разработана в пермской IT-компании ООО «Ньюлана» в сотрудничестве с автором диссертационного исследования. Эта библиотека предоставляет уровень абстрагирования от ОС и инструменты для решения типовых задач. По своей архитектуре и функциональности она схожа с Boost [88]

и Apple Foundation [89]. В состав NFoundation входят методы для организации и упрощения следующих действий:

1. Динамическое распределение памяти: умные указатели и защищённое выделение памяти.
2. Организация объектов однократного создания (синглтонов) и классов, имеющих платформенно-зависимые реализации (с целью создания платформенно-независимых обёрток).
3. Работа с внешними ресурсами: независимые от ОС модули для чтения и записи файлов, потокового ввода-вывода, кодирования и декодирования изображений, разбора XML и т. д.
4. Работа с сетью: сокеты, шифрование, HTTP-запросы и т. д.
5. Управление потоками: создание, настройка и синхронизация потоков, вызов функций из одного потока в другом.
6. Объявление функций обратного вызова и делегатов: структуры данных для организации функций обратного вызова, в том числе таких, которые выполнялись бы в главном потоке, будучи вызванными из второстепенного, и абстрактные классы для организации интерфейсов.
7. Работа с контейнерами: динамические массивы, множества, словари и т. д.
8. Вычисление математических функций и решение типовых задач: общая математика, тригонометрия, основные алгоритмы сортировки и оптимизации.

NFoundation написана на языке C++, но кроме C++ её API доступен через соответствующие обёртки для языков Java, C#, Python и JavaScript.

Для организации платформенно-независимую абстракцию от низкоуровневого графического API используется библиотека NGraphics [90]. Она так же, как и NFoundation, разработана в ООО «Ньюлана» с непосредственным участием автора диссертационного исследования. NGraphics содержит методы, обеспечивающие следующую функциональность:

1. Поддержка стандартов OpenGL, OpenGLES, Direct3D 9 и Direct3D 11 для построения изображений.
2. Работа с вершинными и фрагментными шейдерами.
3. Потокобезопасность с возможностью размещать рендеринг и обработку системных событий в разных потоках.
4. Работа с графом сцены.
5. Аффинные преобразования на основе аппарата матриц и кватернионов.
6. Анимация изменения свойств объектов с применением различных типов интерполяции.
7. Рендеринг в текстуру и постобработка полученного изображения.
8. Средства растеризации текста и векторных контуров.
9. Организация интерактивности (используя ввод с клавиатуры, мыши и сенсорного экрана).
10. Поддержка визуализации на дисплеях высокой чёткости (учёт коэффициентов масштабирования для организации субпиксельной точности изображения).

Для построения традиционных диаграмм и графиков используется библиотека NChart3D [91, 92], базирующаяся на NGraphics. Средствами NChart3D могут быть построены интерактивные двумерные и трёхмерные диаграммы и графики следующих типов:

1. Вертикальные и горизонтальные колонки.
2. Диаграммы с областями.
3. Линейные диаграммы, ступенчатые диаграммы и графики функций вида $y = f(x)$, причём f может иметь разрывы.
4. Круговые и кольцевые диаграммы.
5. Пузырьковые диаграммы и диаграммы рассеивания.
6. Диаграммы рассеивания.
7. Японские свечи, OHLC-диаграммы и полосы.
8. Воронки.

9. Диаграммы, отображающие последовательности элементов.
10. Поверхности по функциям вида $y = f(x, z)$, причём f может иметь разрывы.
11. Диаграммы в полярных координатах.
12. Тепловые карты.

Внешний вид и краткое описание назначения поддерживаемых диаграмм приведены в приложении 5.

Тестовая сцена, представленная на рис. 2, также отображается средствами NChart3D: она является пузырьковой диаграммой, состоящей из 100 сфер, каждая из которых аппроксимирована примерно $1{,}6 \cdot 10^4$ треугольников.

Наряду с NGraphics для эффективного распределённого отображения больших объёмов данных, рендеринга сечений и объёмов используется сторонняя библиотека pVTK. Она распространяется по лицензии BSD, так же, как и все остальные библиотеки семейства VTK, и используется без модификаций.

Для построения графического интерфейса пользователя используется разработанная автором диссертационного исследования библиотека GUIBuilder [93], которая автоматически генерирует интерфейс для конкретной платформы по высокоуровневому описанию.

Для организации Web-интерфейса сервера использован сторонний набор библиотек функций Django, написанный на языке Python. Также Django является базой для модуля интеграции системы SciVi со сторонними решателями. Этот набор библиотек распространяется по лицензии BSD и использован в системе SciVi без модификаций.

2.6. Организация мультиплатформенности

Небольшое количество мультиплатформенных решений среди систем научной визуализации обусловлено отсутствием эффективных высокоуровневых средств поддержки мультиплатформенной переносимости.

Анализ популярных низкоуровневых технологий показал, что эффективнее всего реализовывать логику мультиплатформенного приложения на языке C++, так как программы, написанные на этом языке, во-первых, могут быть скомпилированы и запущены под управлением всех целевых платформ (GNU / Linux, Windows и OS X, а также iOS и Android), а во-вторых, отличаются высоким быстродействием. В операционных системах для настольных компьютеров и высокопроизводительных вычислительных комплексов, а также в iOS, есть возможность выполнения кода, написанного на C++, напрямую. В операционной системе Android, где основным языком разработки является Java, для этого используется технология JNI [94], позволяющая вызывать методы, написанные на C++, из кода, написанного на Java, и наоборот. Именно поэтому большая часть кода системы SciVi написана на языке C++. Для абстрагирования от операционной системы и решения типовых задач в SciVi используется библиотека NFoundation.

В качестве стандарта визуализации предлагается использовать OpenGL. Библиотеки рендеринга, поддерживающие данный стандарт, реализованы под все целевые платформы. На мобильных устройствах используется облегчённая версия стандарта – OpenGLES, являющаяся подмножеством OpenGL. Таким образом код, написанный для мобильных устройств, может быть исполнен на настольных компьютерах и высокопроизводительных вычислительных комплексах, но не наоборот. Разработка SciVi велась с учётом данного ограничения: использовались только те функции, которые доступны в OpenGLES. Такой подход, однако, не лимитировал графических возможностей системы, так как функции, недоступные в OpenGLES, относятся в основном к организации различного рода специальных эффектов, которые не являются необходимыми в контексте научной визуализации. Для обеспечения объектно-ориентированной надстройки над низкоуровневым графическим API используется библиотека NGraphics.

Анализ популярных мультиплатформенных библиотек для создания графического интерфейса пользователя выявил проблему низкой эффективности

интеграции элементов интерфейса со сценой, визуализируемой средствами аппаратно-ускоренной графики. Использование проанализированных мультиплатформенных библиотек создания графического интерфейса пользователя в сочетании с низкоуровневыми средствами вывода графики приводит к падению производительности рендеринга примерно на 12%. Более того, на мобильных устройствах зачастую эффективность рендеринга падает и при использовании стандартных элементов графического интерфейса пользователя, предоставляемых операционной системой. Так, например, результаты тестирования производительности рендеринга тестовой сцены (рис. 2) с использованием стандартных элементов интерфейса на устройстве iPad 3 под управлением операционной системы iOS 8 совпали с результатами, полученными при использовании элементов интерфейса из библиотеки Qt-iPhone: скорость визуализации снизилась с 15 до 8 FPS.

Падение производительности происходит из-за того, что элементы интерфейса и сцена отображаются через различное API, и операционная система тратит дополнительное время на переключение графических контекстов (системных структур данных, необходимых для рендеринга) и слияние полученных изображений. Особенно остро данная проблема стоит на мобильных устройствах, обладающих меньшей вычислительной мощностью по сравнению с настольными компьютерами. Это означает, что максимальной эффективности рендеринга можно достичь лишь в том случае, когда графический интерфейс пользователя отображается при помощи того же API, что и сцена, то есть актуальной является разработка библиотеки элементов интерфейса, использующей OpenGL(ES) в качестве стандарта визуализации.

Кроме того, с мультиплатформенным графическим интерфейсом пользователя связана также описанная ранее проблема двойного дизайна, то есть необходимость разрабатывать интерфейс по отдельности для настольных компьютеров и для мобильных устройств.

Проблема отсутствия адекватных поставленным задачам средств для создания графического интерфейса пользователя решена разработкой собственной библиотеки, названной GUIBuilder. Необходимость переключения графических контекстов исключается, так как GUIBuilder использует для рендеринга интерфейса то же самое графическое API, которое используется основным приложением для рендеринга трёхмерной сцены. Необходимость повторного проектирования интерфейса исключается, так как GUIBuilder принимает на вход высокоуровневое XML-описание интерфейса, в котором присутствуют только данные о типах управляющих элементов и их взаимном расположении. Все дальнейшие действия по отрисовке элементов интерфейса в привычном для конкретной платформы виде и поддержке соответствующей парадигмы управления GUIBuilder осуществляет автоматически. Реализации различных элементов интерфейса и различных парадигм управления входят в исходный код этой библиотеки в виде платформенно-зависимых модулей. Более подробно библиотека GUIBuilder описана в разделе 2.8.2.

Описанные средства позволяют системе SciVi выполняться на ЭВМ под управлением GNU / Linux, Windows, OS X, iOS и Android, а в будущем возможен перенос и на другие ОС.

2.7. Описание сервера

2.7.1. Архитектура

Архитектура сервера системы научной визуализации, организованной в соответствии с предлагаемым подходом, приведена на рис. 9. Сервер системы SciVi на 40% написан на языке C++ и на 60% – на языке Python.

Для работы с решателем ⑬ сервер использует XML-описание ⑧, за автоматическую генерацию которого отвечает интеграционный модуль ⑩, использующий онтологии ⑪. Пользователь может зарегистрировать нужный ему решатель через Web-интерфейс ⑨, при этом модуль ⑩ автоматически сгенерирует нужное XML-описание, сохранит его вместе с исполняемыми файлами решателя в

специальном репозитории на сервере и внесёт соответствующие данные в каталог решателей ⑦. Этот каталог доступен клиентам по протоколу HTTP. Регистрация включает в себя автоматический синтаксический анализ исходного кода решателя (при наличии), а также установление соответствий между элементами структуры входных-выходных данных решателя и свойствами визуальных объектов (как было описано в разделе 2.3). Вместо решателя пользователь может также загрузить готовые входные и / или выходные данные.

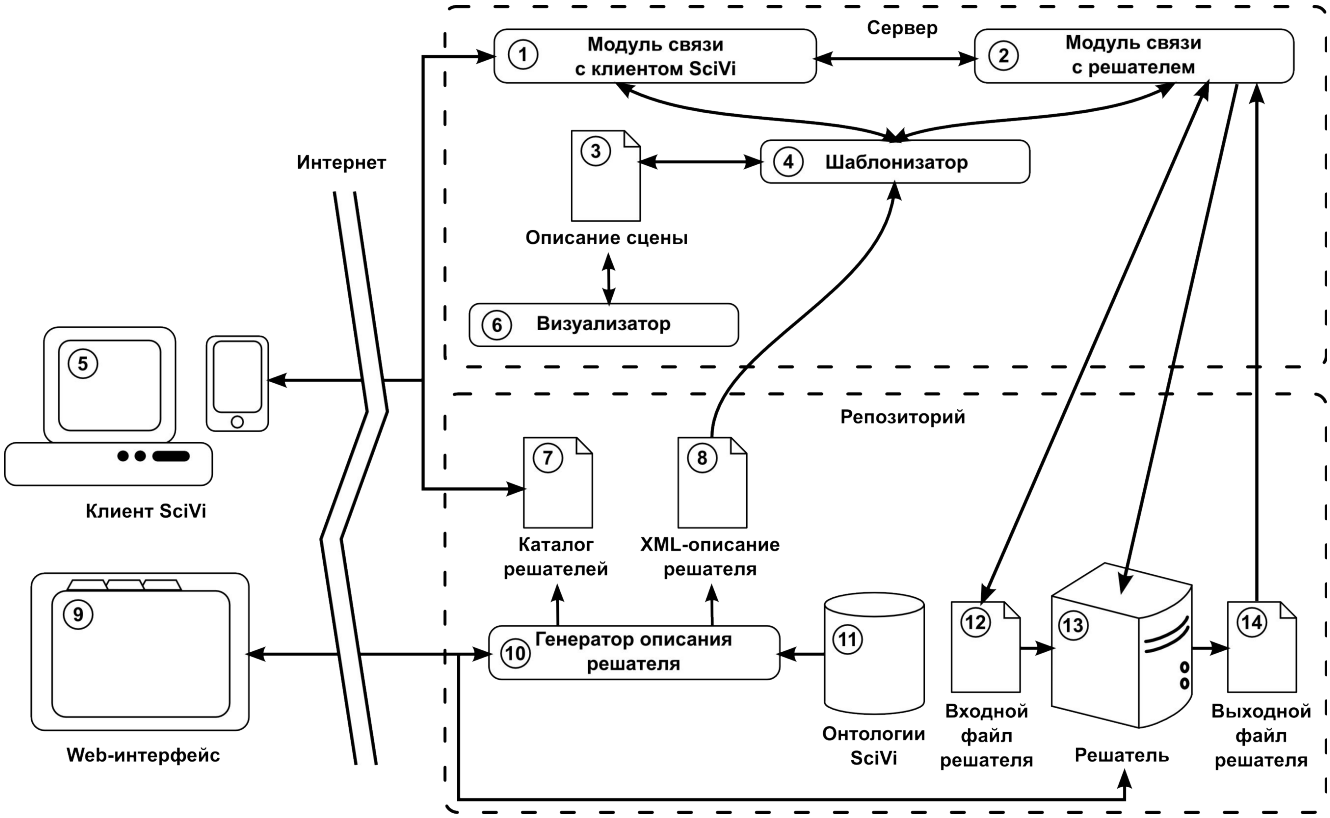


Рис. 9. Архитектура сервера SciVi

Клиентское приложение ⑤ производит аутентификацию на сервере при помощи имени пользователя и пароля (имя и пароль закрепляются за пользователем при регистрации) и предоставляет учёному-исследователю возможность в режиме реального времени отдавать команды решателю ⑬, если он был предварительно загружен на сервер. За трансляцию команд от клиента решателю отвечают модули коммуникации ① и ②. Для управления решателем служат параметры командной строки (задаваемые пользователем во время регистрации решателя) и входной файл ⑫. Решатель выводит результаты своей

работы в файл ⑭. Содержимое файлов ⑫ и ⑭ через модуль коммуникации ② подаётся на вход шаблонизатору ④, который, используя XML-описание ⑧, автоматически строит совместимое со SciVi описание подлежащей визуализации сцены ③.

Сервер автоматически производит планирование процесса рендеринга, эвристически учитывая производительность клиента, скорость соединения, собственную загруженность и особенности сцены. На сервере визуализируются в первую очередь те части сцены, на внешний вид которых не может повлиять пользователь. Например, если итогом визуализации является некоторая модель с наложенной на неё текстурой, а текстура представляет собой результат дополнительной визуализации, то текстура готовится на сервере. В случае нехватки производительности клиента или недостаточной для передачи всего объёма данных скорости соединения сервер автоматически осуществляет упрощение сцены. Коэффициент упрощения определяется эвристически. Визуализация и упрощение сцены осуществляются модулем ⑥, а затем все данные, предназначенные для клиента, передаются по сети при помощи коммуникационного модуля ①. Более подробно вопрос визуализации на стороне сервера будет рассмотрен в разделе 2.7.3.

2.7.2. Управление решателем

При помощи созданного на стороне клиента графического интерфейса пользователь задаёт входные данные, которые через коммуникационный модуль, обозначенный на рис. 9 ①, попадают на сервер. Параметры передаются по сети в виде сериализованного словаря (множества пар «ключ-значение»). Модуль коммуникации с клиентом передаёт этот словарь шаблонизатору, который, используя сгенерированное интеграционным модулем описание, создаёт входной файл для решателя.

Работой решателя управляет коммуникационный модуль, обозначенный на рис 9 ②. Он записывает подготовленный шаблонизатором входной файл по пути,

который указан в описании, а затем запускает решатель на выполнение. Если работа решателя завершается с ошибкой (код завершения отличен от 0), информация, выданная им в стандартный поток ошибок (stderr) передаётся клиенту в качестве отчёта о неудаче и отображается пользователю вместе с уведомлением о том, что задачу выполнить не удалось. Если же ошибок не возникло, сервер переходит в режим ожидания запроса данных со стороны клиента.

В настоящий момент инициатива запроса данных лежит на клиенте. который запрашивает данные для визуализации по команде пользователя.

Как только от клиента пришёл запрос на получение новой порции данных, коммуникационный модуль ② пытается открыть на чтение выходной файл решателя, путь к которому задаётся шаблонизатором. Если файл ещё не создан, или же в нём находится недостаточное количество данных, чтобы полностью заполнить цельный блок в шаблоне сцены, клиенту отправляется уведомление о том, что данные ещё не готовы. Иначе очередная считанная из выходного файла порция данных записывается в шаблон сцены.

Имеется возможность визуализировать на одной сцене и входные, и выходные данные эксперимента. После записи данных в шаблон, выполняются все необходимые действия по подготовке сцены для клиента, включая этап обработки данных и частичной визуализации на стороне сервера. Заключительным этапом является отправка сцены и всех сопутствующих мультимедийных ресурсов клиенту средствами коммуникационного модуля ①. Если решатель к моменту запроса данных успел завершить работу, клиенту вместе с финальным вариантом сцены отправляется уведомление об окончании выполнения задачи.

Таким образом, клиент может запрашивать данные в процессе их генерации решателем. SciVi предоставляет возможность визуализировать не только результат, но и весь процесс научного эксперимента.

Пользователь имеет возможность в любой момент времени остановить (и, при необходимости, перезапустить) работу решателя. Для этого имеются соответствующие элементы управления на генерируемом пользовательском интерфейсе системы. В случае остановки решателя, входной и выходной файлы не удаляются, и остаётся возможность визуализировать хранящиеся в них данные.

2.7.3. Обработка и визуализация данных

Подготовкой данных для отображения на стороне клиента занимается модуль визуализации ④, изображённый на рис. 9. Этот модуль основан на библиотеках pVTK, NGraphics и NChart3D, а также содержит в себе логику планирования распределённого процесса рендеринга и упрощения сцены. Планирование и упрощение происходит в соответствии с эвристиками, которые основываются на следующих начальных данных:

1. Тип клиента (настольный компьютер или мобильное устройство).
2. Быстродействие клиента (выражается в скорости визуализации тестовой сцены, состоящей из $4{,}8 \cdot 10^6$ вершин, см. рис. 2).
3. Скорость сетевого соединения с клиентом (определяется при помощи тестового запроса).
4. Количество подключенных в данный момент клиентов.
5. Объём доступной на сервере оперативной памяти.

Тип клиента, его быстродействие и скорость соединения определяются на стороне клиента и отправляются на сервер при первом запросе. Затем, когда одно состояние сцены уже было визуализировано, данные о быстродействии клиента и скорости соединения корректируются. Новые данные о скорости полностью замещают собой старые, так как скорость передачи данных может измениться с течением времени лишь под действием внешних факторов (загруженность сети) и не зависит от характера передаваемых данных. Новые данные о быстродействии, выведенные на основании подсчёта количества отображаемых кадров в секунду, которые способен отрисовать клиент на фактической (а не на тестовой) сцене

заменяют старые лишь в том случае, если оказалось, что клиент справляется с визуализацией медленнее, чем было спрогнозировано. Такая стратегия используется ввиду прямой зависимости скорости визуализации от характера сцены: из того, что клиент способен быстро отрисовать сцену на одном шаге эксперимента не следует, что он сможет так же быстро отрисовывать и все последующие состояния сцены (которые, скорее всего, будут сложнее, так как решатель генерирует данные инкрементально, постепенно увеличивая их объём). Под шагом эксперимента здесь понимается обращение клиента за новой порцией данных.

На каждом шаге модуль визуализации анализирует имеющиеся в текущий момент времени сведения о сцене, чтобы принять решение о распределении рендеринга и необходимости упрощения данных. Таким образом, конкретный вариант распределения при каждом следующем запросе сцены может отличаться от предыдущего.

Исходя из сравнения числа полигонов на текущей и тестовой сценах прогнозируется скорость визуализации текущей сцены клиентом. Для прогноза используется математическая модель, основанная на предположении об обратной пропорциональности числа вершин скорости визуализации. Такое предположение делается на основе того, что средняя сложность процесса рендеринга является линейной относительно числа вершин на сцене. Обоснование линейности приведено в главе 3. Таким образом, прогнозируемая скорость может быть получена по формуле

$$f_c = \frac{f_t n_t}{n_c}, \tag{26}$$

где f_c – прогнозируемая скорость визуализации (в кадрах в секунду),

n_c – число вершин на текущей сцене,

f_t – скорость визуализации тестовой сцены (в кадрах в секунду),

n_t – число вершин на тестовой сцене.

В качестве порогового значения «допустимой» скорости визуализации принято 8 FPS, так как, согласно исследованию [30], это минимальное значение частоты кадров, обеспечивающее плавное восприятие человеком отображения движений.

Если спрогнозированная скорость оказывается меньше пороговой, сервер принимает решение о необходимости упрощения сцены. Упрощение возможно путём уменьшения числа вершин в используемых на сцене моделях. Для этого используется сторонний алгоритм, описанный в [95]. Пример упрощения трёхмерной модели представлен на рис. 10.

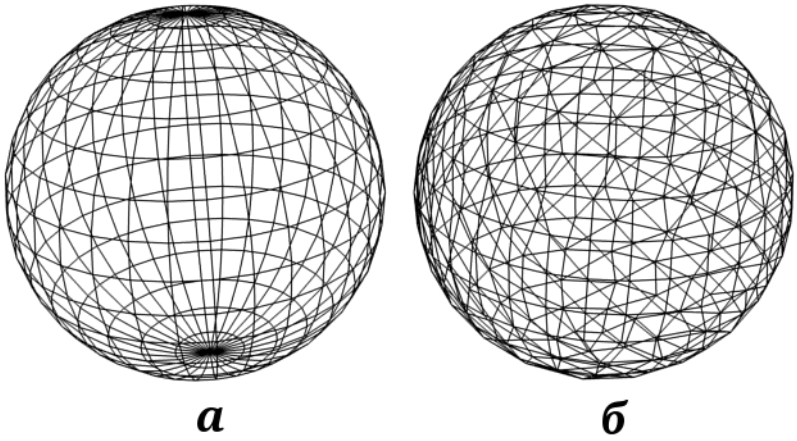


Рис. 10. Трёхмерная модель сферы (а) и результат её упрощения путём удаления 40% вершин (б)

Экспериментально установлено, что алгоритм упрощения сохраняет визуальное качество моделей в том случае, если изначально модели имели более 5000 вершин и количество вершин сокращается не более чем на 40%. Эти значения приняты в качестве пороговых в эвристике, определяющей применимость упрощения моделей. Вычисляется прогнозируемый максимум числа вершин, обеспечивающий скорость визуализации в 8 FPS, и если этот максимум достижим путём упрощения моделей, имеющих более 5000 вершин, не более чем на 40%, упрощение применяется. Обход моделей осуществляется в соответствии с жадным алгоритмом оптимизации, то есть в первую очередь упрощению подвергаются модели, имеющие максимальное количество вершин, и

процент упрощения для них также выставляется максимальным. Эксперименты показали, что она даёт приемлемые результаты. Если прогнозирование показывает, что упрощение моделей не приведёт к достижению минимально допустимой скорости, сервер выполняет визуализацию сцены полностью и отправляет клиенту лишь итоговое изображение. В этом случае клиент и сервер ведут себя так же, как при использовании технологии VNC [31].

На основе измеренной скорости соединения вычисляется время пересылки всех данных, необходимых клиенту. Принято решение установить пороговое время передачи данных равным 10 минутам. Если прогноз показывает, что сцена не сможет быть передана за 10 минут, определяется, возможно ли упростить модели и уменьшить размеры текстур. Экспериментально установлено, что минимальный размер текстур, при котором визуальное качество остаётся приемлемым, в среднем составляет 128x128. Текстуры уменьшаются пропорционально, начиная с самых крупных. Уменьшение происходит по шагам, соответствующим степеням двойки. То есть для большей стороны текстуры, имеющей длину w, находится такое максимальное натуральное $n \geq 7$, чтобы прогнозируемое время передачи сцены с этой текстурой, масштабированной с коэффициентом $\frac{2^n}{w}$, стало попадать в допустимые пределы, либо максимально приблизилось к ним. Уменьшение текстур производится после упрощения моделей, так как текстуры в большинстве случаев передают микрорельеф объектов и влияют на визуальное качество сильнее, чем геометрическая форма моделей. Более того, текстура высокого разрешения часто способна скрыть низкую полигональность модели.

Если решатель моделирует некоторый процесс, и в результате своей работы генерирует множество состояний сцены в разные моменты модельного времени, возможным способом уменьшения объёма передаваемых данных оказывается сокращение числа этих состояний. Состояния используются в качестве ключевых кадров для воспроизведения анимации; для организации плавных движений

производится линейная интерполяция изменившихся свойств объектов. Экспериментально установлено, что в среднем случае адекватную анимацию протекания моделируемого решателем процесса можно получить даже по 10% ключевых кадров. Таким образом, если упрощения моделей и текстур оказывается недостаточно для обеспечения нужной скорости передачи сцены, сервер прибегает к отбросу промежуточных ключевых кадров.

Если указанные способы не могут обеспечить необходимого времени передачи данных, и загрузка сервера ниже порогового значения, сервер производит рендеринг целиком. Пороговым значением нагрузки сервера выступает следующее соотношение:

$$m_s \cdot k < m_{available}, \tag{27}$$

где m_s – количество памяти, требуемое для загрузки всех мультимедийных ресурсов (моделей, текстур и шейдеров) сцены,

$m_{available}$ – количество доступной оперативной памяти,

k – количество подключенных в данный момент клиентов.

Это соотношение основано на «эвристике банкира»: делается предположение, что все остальные клиенты могут запросить столько же памяти, сколько данный, и запрос удовлетворяется только в том случае, если памяти будет достаточно для всех.

Если нагрузка сервера слишком высока, данные отправляются клиенту, несмотря на превышение порогового значения времени на передачу.

Если какие-либо текстуры моделей сцены являются результатом рендеринга другой сцены, и нагрузка сервера не выше пороговой, подготовка этих текстур выполняется на сервере.

Для клиентов, выполняющихся на мобильных устройствах, сервер производит разбиение трёхмерных моделей, количество вершин в которых превышает 2^{16}, на фрагменты, удовлетворяющие этому ограничению. Данное ограничение для мобильных устройств является аппаратным и оговорено стандартом низкоуровневых библиотек визуализации OpenGLES версии 2.0 [96].

Такое разбиение представлен на рис. 11 на примере трёхмерной модели пульта дистанционного управления, содержащей $2{,}6 \cdot 10^5$ вершин, полученной при помощи трёхменого сканера марки Roland LPX-600. Между фрагментами модели для наглядности искусственно добавлено расстояние. Для настольных компьютеров аппаратное ограничение на количество вершин в одной модели составляет 2^{32}, что в значительной степени превышает среднее количество вершин в используемых для визуализации моделях. Поэтому разбиение моделей для настольных компьютеров не производится.

Рис. 11. Трёхмерная модель, состоящая из 4 фрагментов в виде, отображаемом в системе SciVi (а), и с искусственно добавленным расстоянием между фрагментами (б)

Благодаря использованию описанных механизмов упрощения и разбиения, появляется возможность отображать сильно детализированные 3D-модели на различных платформах. Такие модели могут быть получены, например, посредством трёхмерного сканирования объектов реального мира. Для сканера Roland LPX-600 максимальное число вершин в модели составляет примерно $1{,}57 \cdot 10^9$. При помощи описанных алгоритмов визуализация подобных моделей на мобильных устройствах оказывается возможной.

Потенциально количество параметров эвристик, используемых для распределения рендеринга, может быть увеличено. Настройки работы эвристик

(например, пороговые значения или формулы для их вычисления) могут быть вынесены на интерфейс пользователя, чтобы обеспечить адаптируемость системы к различным ситуациям. Однако уже в настоящее время реализованные алгоритмы обеспечивают достаточно высокую производительность системы SciVi, даже когда клиентами выступают мобильные устройства, а способом их соединения является низкоскоростной EDGE.

Использование популярной на сегодняшний день технологии VNC в контексте низкоскоростных безпроводных соединений приводит к очень большим задержкам отклика на команды пользователя. Например, если используется мобильное устройство iPad 4, обладающее разрешением экрана 2048x1536, средний размер полноэкранного изображения при использовании сжатия без потерь составляет 1 Мб. Соединение по способу EDGE обеспечивает скорость в 474 кбит/с, следовательно, для передачи одного полноэкранного кадра потребуется около 17 с, что и обусловит высокую задержку на команды пользователя.

В худшем случае (при переходе в режим визуализации полностью на стороне сервера) предложенный метод обладает такой же продолжительностью задержки, однако в среднем случае она составляет от 125 до 60 мс.

Время первичной загрузки сцены при использовании предложенного метода в большинстве случаев не превышает 10 мин., в среднем случае составляет порядка 1,5 мин.

2.7.4. Коммуникация с клиентом

За коммуникацию с клиентом отвечает модуль (9), обозначенный на рис. 9. Передача сообщений по сети осуществляется посредством протокола прикладного уровня SVTP (*англ.* SciVi Transfer Protocol, протокол передачи данных в SciVi), разработанного специально для системы SciVi. Этот протокол является облегчённой версией HTTP, реализован поверх механизма сокетов и описывает формат сообщений, состоящих из заголовка и тела.

Заголовок представляет собой структуру, состоящую из следующих полей:

- Тип сообщения.
- Размер тела сообщения в байтах.
- Глобальный уникальный идентификатор задачи:
  - Идентификатор процесса решателя.
  - Идентификатор сессии.

Полная документация типов сообщений приведена в приложении 6. Глобальный уникальный идентификатор задачи вынесен в заголовок с целью оптимизации скорости получения находящейся в нём информации: в среднем 95% сообщений между клиентом и сервером так или иначе касаются работы решателя и требуют идентифицировать его процесс и сессию работы с ним.

Тело сообщения может содержать произвольный набор байтов, интерпретация которых происходит на основе типа, указанного в заголовке.

При помощи протокола SVTP осуществляется передача управляющих команд в виде сериализованных словарей (пар «ключ-значение»), файлов (при этом данные делятся на пакеты фиксированного размера) и метаинформации (в виде строк).

Достоинством SVTP по сравнению с HTTP является его лаконичность: за счёт своей привязанности к особенностям работы системы SciVi этот протокол позволяет уменьшить количество накладных расходов при передаче сообщений между клиентом и сервером. При необходимости данные, передаваемые по протоколу SVTP могут шифроваться согласно криптографическим протоколам SSL или TLS.

2.8. Описание клиента

2.8.1. Архитектура

Архитектура сервера системы научной визуализации, организованной в соответствии с предлагаемым подходом, приведена на рис. 12. Большая часть (порядка 85%) кода клиента SciVi написана на языке C++ и является

мультиплатформенной. Платформенно-зависимая часть представляет собой оболочку, инициализирующую ядро и графический контекст (платформенно-зависимые структуры данных, необходимые для работы низкоуровневого графического API). Кроме того, эта часть отвечает за диспетчеризацию событий ОС, например, низкоуровневых команд от пользователя, таких, как перемещение курсора мыши или касания сенсорного экрана.

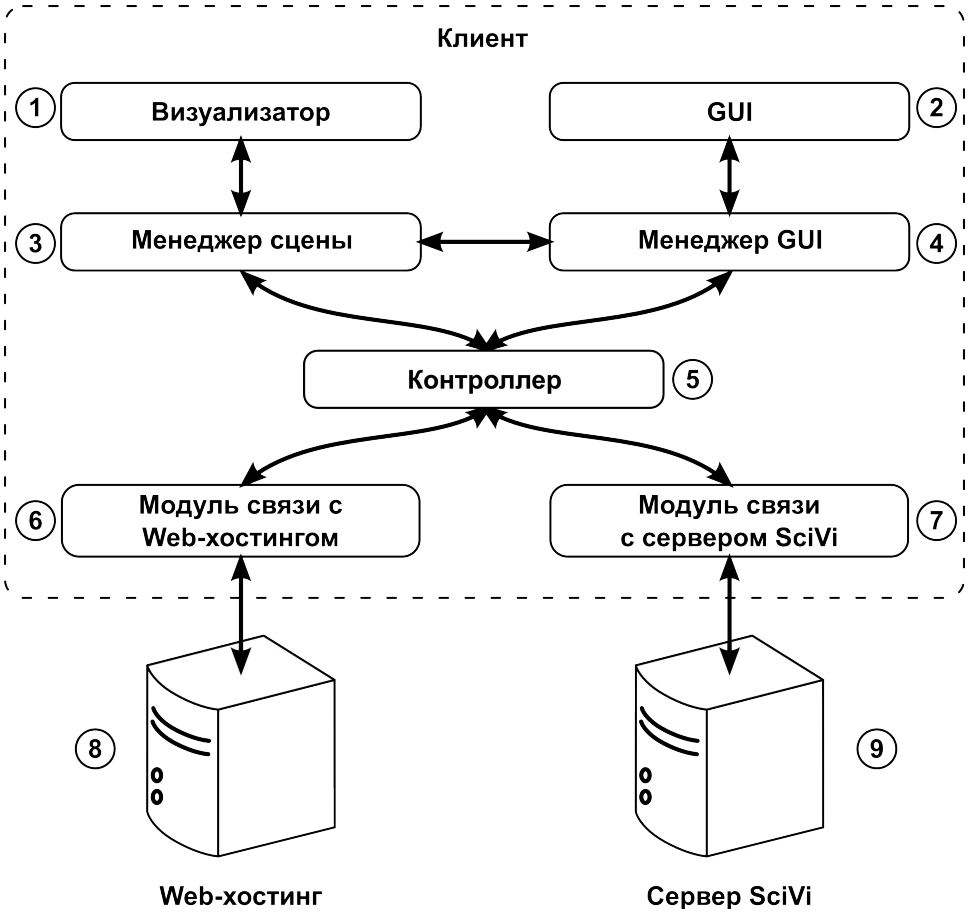


Рис. 12. Архитектура клиента SciVi

Клиент спроектирован в соответствии с шаблоном проектирования Model-View-Controller [97]. Его логика реализована модулем ⑤. Первым действием после запуска клиент через модуль коммуникации ⑥ запрашивает у Web-хостинга ⑧ список доступных ресурсов (решателей и готовых данных). Адрес Web-хостинга находится в настройках клиента и может быть изменён пользователем. Как правило, Web-хостинг находится там же, где и сервер системы SciVi (чаще всего список доступных ресурсов автоматически загружается на Web-хостинг

сервером, хотя может быть загружен и администратором вручную). Пользователь выбирает интересующий его ресурс (и, соответственно, сервер, ответственный за этот ресурс), указывает имя и пароль для аутентификации, и клиент предпринимает попытку соединения с сервером ⑨ посредством коммуникационного модуля ⑦. Если клиенту удаётся установить соединение и авторизоваться, он получает от сервера ⑨ данные, соответствующие режиму работы, связанному с выбранным ресурсом: либо готовые данные для визуализации, либо описание графического интерфейса для запрашиваемого решателя. В первом случае немедленно начинается процесс визуализации посредством модулей ③ и ①. Во втором – контроллер ⑤, получив описание интерфейса от коммуникационного модуля ⑦, передаёт его менеджеру интерфейсов ④, который строит соответствующий этому описанию графический интерфейс, используя библиотеку GUIBuilder. Пользователь, взаимодействуя с интерфейсом ②, задаёт необходимые настройки решателя, и они отправляются на сервер. Решатель, запущенный с учётом этих настроек, генерирует данные, которые обрабатывает сервер. Обработанные данные, в общем случае представляющие собой частично визуализированную сцену, отправляются клиенту. Контроллер ⑤ передаёт их менеджеру сцены ③, который, произведя необходимые приготовления (создав все нужные структуры данных), передаёт их визуализатору ①. Визуализатор, в свою очередь, строит итоговое изображение, и пользователь получает возможность интерактивно взаимодействовать с отображённой сценой. При этом его команды захватываются интерфейсом ②, через менеджер интерфейсов ④ передаются менеджеру сцены ③ и влияют на работу визуализатора ①.

2.8.2. Организация графического интерфейса пользователя

Графический интерфейс пользователя автоматически строится средствами библиотеки GUIBuilder. Эта библиотека, в отличие от аналогичных, обеспечивает высокую производительность при размещении элементов интерфейса поверх

отображаемой графической сцены, а также позволяет решить описанную ранее проблему двойного дизайна.

Высокое быстродействие достигается благодаря тому, что GUIBuilder использует для визуализации элементов интерфейса библиотеку NGraphics, которая, в свою очередь, работает на основе низкоуровневого графического API стандарта OpenGL(ES). Таким образом, интерфейс и основная сцена отображаются одной и той же подсистемой вывода графики, что исключает необходимость переключения графических контекстов и слияния изображений (как это происходит при использовании большинства других средств для построения пользовательских интерфейсов и даже стандартных средств ОС). Согласно результатам проведённых тестов, использование GUIBuilder вместо других средств построения интерфейсов при рендеринге достаточно сложных сцен (таких, например, как тестовая сцена, представленная на рис. 2) на мобильных устройствах позволяет увеличить производительность в среднем на 5-7 FPS. На настольных компьютерах ввиду большей вычислительной мощности прирост производительности значительно менее заметен и составляет в среднем 0,5 FPS.

Архитектура библиотеки GUIBuilder представлена на рис. 13.

Проблема двойного дизайна решается за счёт того, что сгенерированное описание интерфейса представлено декларативно на языке XML, при этом в нём присутствуют лишь такие высокоуровневые понятия, как «кнопка», «регулятор», «текстовая метка» и т. д. без детализации их внешнего вида и механизма взаимодействия с ними пользователя. За конкретные реализации тех или иных элементов управления отвечают платформенно-зависимые модули в составе GUIBuilder. На уровне программного кода к объектно-ориентированным абстракциям элементов интерфейса можно обращаться по идентификаторам, задаваемым в XML-описании. Каждая абстракция имеет ряд структур для присоединения функций обратного вызова, которые вызываются при возникновении тех или иных событий.

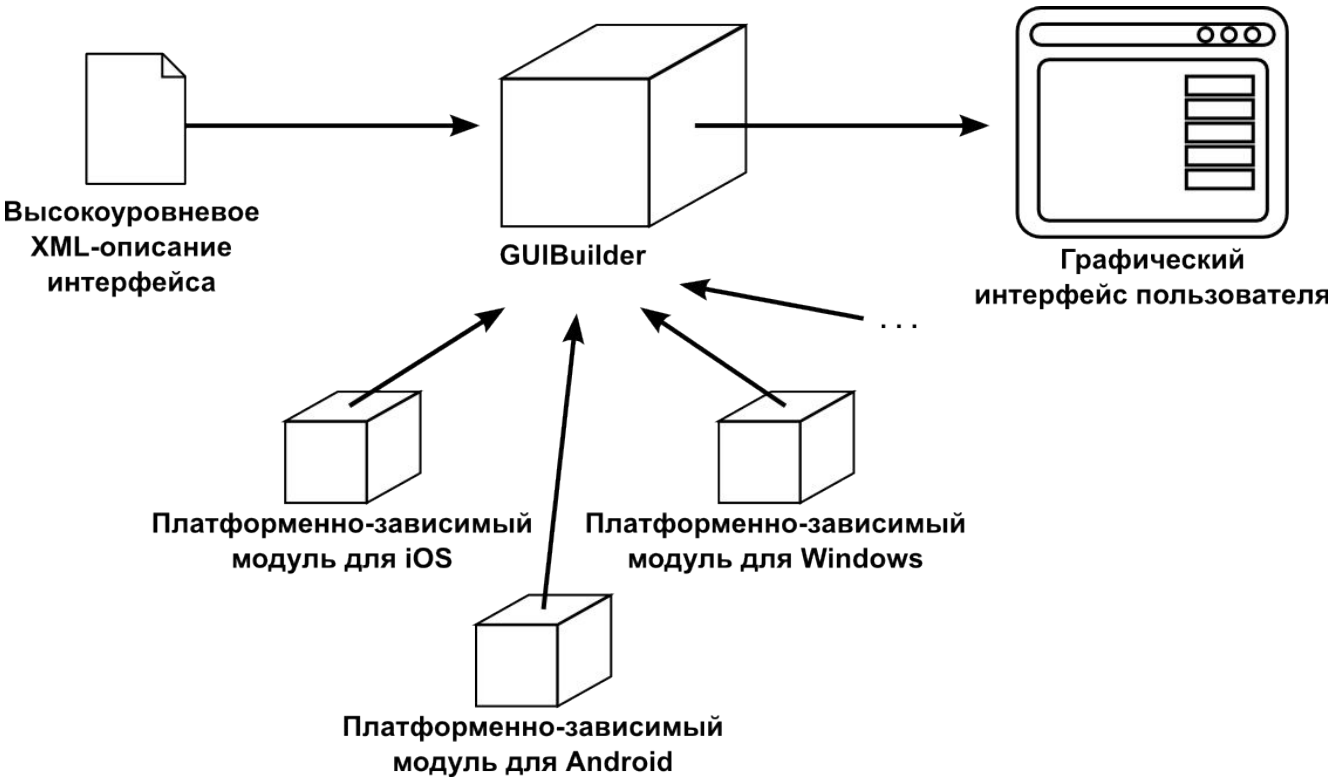


Рис. 13. Архитектура библиотеки построения графических интерфейсов пользователя GUIBuilder

Так, например, у объекта «кнопка» присутствует возможность установить функцию обратного вызова на событие «нажатие», при этом программисту не важно, на какой платформе и вследствие чего это событие произойдёт – либо на настольном компьютере из-за того, что по кнопке произошёл клик мышкой, либо на мобильном устройстве в результате касания сенсорного экрана. GUIBuilder полностью скрывает фактические механизмы взаимодействия пользователя с интерфейсом внутри платформенно-зависимых частей. Данная библиотека соответствует реализации оператора Γ в формальной модели системы научной визуализации (22). Множество конкретных элементов управления, реализация которых поддерживается библиотекой на уровне платформенно-зависимых модулей соответствует множеству $\overline{M}$. Множество поддерживаемых действий пользователя соответствует множеству $\overline{E}$.

Недостатком предложенного подхода является ограниченность настраиваемости внешнего вида интерфейса. Для модификации элементов интерфейса программист должен расширять GUIBuilder, добавив в иерархию

классов новые сущности. Однако в рамках создания системы научной визуализации настраиваемость внешнего вида интерфейса не является приоритетной: для управления решателями и отображаемой сценой определённых заранее стандартных элементов пользовательского интерфейса оказывается достаточно.

Пример описания графического интерфейса пользователя на языке разметки GUIBuilder приведён в листинге 2, а изображение построенное по этому описанию – на рис. 14. Полная документация по языку описания интерфейсов, используемого в GUIBuilder, приведена в приложении 7. Идеологически этот язык схож с языком XAML [98] от компании Microsoft, однако основывается на более простых и высокоуровневых примитивах.

Благодаря описанным преимуществам, простоте языка описания интерфейсов и небольшому объёму исходного кода, библиотека GUIBuilder оказывается адекватным средством создания графического интерфейса пользователя в мультиплатформенных системах научной визуализации.

Листинг 2. Пример описания интерфейса для библиотеки GUIBuilder

```
<tabbar id="tabbar"
        type="collapsable"
        background="#bbbbbbdd"
        tabbuttons="top">
    <tab id="zoomTab"
         image="zoom"
         layout="stack">
        <spacing height="small"/>
        <area layout="queue">
            <spacing width="small"/>
            <button id="discardZoom"
                    importance="0.5"
                    image="discardZoom"/>
            <spacing width="small"/>
            <area layout="stack"
                  importance="1">
                <slider id="zoomLine"
                        handler="handler"
                        body="slider"/>
                <dslider id="scaleLine"
                         handler="handler"
                         body="slider"
                         tick_width="2"/>
```

```
            </area>
        </area>
        <spacing height="small"/>
    </tab>
</tabbar>
```

Рис. 14. Графический интерфейс пользователя в SciVi, построенный библиотекой GUIBuilder по описанию, приведённому в листинге 2

2.8.3. Рендеринг сцены

Клиент получает от сервера XML-описание сцены, содержащее в себе ссылки на все мультимедийные ресурсы (трёхмерные модели, текстуры и шейдеры), описания положений и других свойств объектов, а также изменение этих свойств во времени, если на сцене должна присутствовать анимация. Мультимедийные ресурсы, как правило, также расположены на сервере и доступны по протоколу SVTP. Однако они могут находиться в любом открытом хранилище файлов, тогда получение осуществляется по протоколу HTTP.

Менеджер сцены, обозначенный на рис. 12 ③, выбирает, какую из библиотек визуализации использовать (VTK / VES, NGraphics или NChart3D) и строит из предоставляемых выбранной библиотекой объектов граф сцены по полученному описанию. Выбор осуществляется на основе характера сцены:

1. Если сцена содержит объекты, которые необходимо визуализировать с применением методов рендеринга сечений и / или объёмов, используется VTK / VES.
2. Если сцена представляет собой диаграмму или график, используется NChart3D.
3. В остальных случаях используется NGraphics.

Недостатком библиотек семейства VTK является отсутствие поддержки многомасштабного рендеринга, поэтому этих библиотек оказалось недостаточно для визуализации всех сцен, с которыми может работать система SciVi. Под многомасштабным рендерингом понимается метафора микроскопа, то есть возможность количественного и качественного масштабирования сцены. Кроме того, масштабирование применимо и к скорости воспроизведения анимации, что позволяет управлять временем отображения процессов, происходящих в визуализируемых сценах. Средства поддержки многомасштабного рендеринга реализованы автором диссертационного исследования на основе библиотеки NGraphics.

Количественное изменение масштаба метафорически соответствует подстройке фокуса линзы микроскопа и позволяет плавно изменять размер объектов сцены. При этом встроенными средствами библиотеки NGraphics автоматически изменяется детализация объектов: чем объект больше, тем более детальная модель используется для его представления, и наоборот. Таким образом, чем больше масштаб, тем более сложная геометрия должна быть визуализирована, но и тем больше объектов оказывается за пределами экрана. Объекты, не попавшие на экран, отбрасываются на основании отсечения по усечённой пирамиде видимости. NGraphics автоматически определяет, пересекает ли ограничивающий параллелепипед объекта усечённую пирамиду видимости, и если пересекает, объект отправляется на визуализацию, а в противном случае – отбрасывается. Такой подход резко увеличивает производительность визуализации на сложных сценах [99].

Качественное изменение масштаба метафорически соответствует смене линзы микроскопа и приводит к полной перестройке сцены. Данные для различных масштабов сцены генерирует решатель. Поддержка многомасштабности не является обязательным требованием для решателя, однако система визуализации должна предоставлять функциональность для работы и с данным способом изучения сцены. Пример переключения масштабов в SciVi приведён на рис. 15.

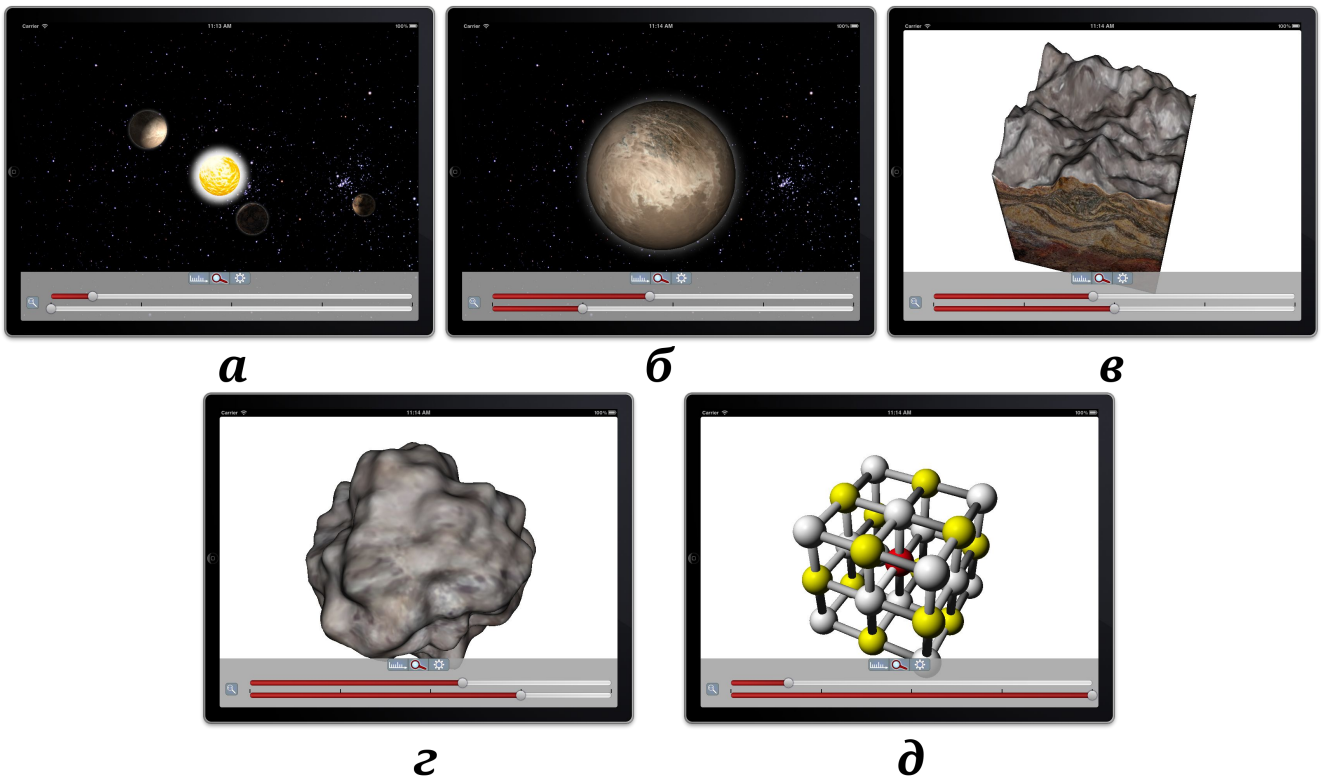


Рис. 15. Переключение между различными масштабами одной сцены в SciVi: модель планетарной системы (а), модель индивидуальной планеты (б), модель фрагмента ландшафта (в), модель камня (г), модель кристаллической решётки (д)

Возможность изменения скорости проигрывания анимации также является очень важной в контексте научной визуализации. При помощи анимации отображается протекание каких-либо моделируемых процессов, которые могут иметь различную реальную скорость. У пользователя же должна быть возможность ускорять и замедлять воспроизведение этой анимации, а также «проматывать» неважные для него фрагменты, чтобы сконцентрировать внимание на интересующих деталях изучаемого процесса.

Для управления анимацией на интерфейс клиента SciVi вынесены так называемая «линия времени» и регулятор скорости анимации, представленные на рис. 16. Деления на «линии времени» соответствуют состояниям сцены, которые были получены от решателя. Промежуточные состояния вычисляются системой визуализации средствами линейной интерполяции, что позволяет достичь плавной анимации.

Рис. 16. Линия времени и регулятор настройки скорости анимации в SciVi

SciVi поддерживает интерполяцию аффинных преобразований объектов сцены, а также интерполяцию любых однородных (*англ.* uniform) переменных для шейдеров, используемых на сцене.

Объекты сцены могут быть представлены как стандартными примитивами (например, сфера или куб), так и трёхмерными моделями в форматах 3DS [100], PLY [101] и N3D. N3D – это внутренний бинарный формат хранения трёхмерных моделей NGraphics, предполагающий описание вершин с минимальной необходимой метаинформацией. Документация этого формата представлена в приложении 8.

Материалы объектов (модель освещения, способ наложения текстуры и т. д.) и визуальные эффекты сцены описываются при помощи шейдеров, которые могут быть либо выбраны из встроенной библиотеки SciVi (например, для организации простого освещения по Фонгу), либо написаны пользователем на языке ESSL [102] (языке программирования шейдеров из стандарта OpenGLES). Поддерживаются вершинные и фрагментные шейдеры. Геометрические шейдеры не поддерживаются ввиду их отсутствия в стандарте OpenGLES. Для работы на настольных компьютерах (где используются библиотеки стандарта OpenGL) шейдеры должны быть написаны на языке GLSL [103] (языке программирования шейдеров из стандарта OpenGL), однако NGraphics выполняет конвертацию из ESSL в GLSL автоматически. Таким образом, если пользователь обладает соответствующей квалификацией, он имеет возможность добавить в систему свои шейдеры, реализовав их только на ESSL.

Для осуществления навигации по сцене используется два подхода:

1. Простое вращение, перемещение и масштабирование. Вращение осуществляется при помощи задания углов Эйлера.
2. Орбитальная камера (*англ.* arcball camera [104]), которая в большинстве случаев позволяет осуществлять вращение сцены более удобным для пользователя способом.

Между этими двумя режимами пользователь может свободно переключаться. Различные действия по изменению сцены могут быть настроены на различные команды управления (как для настольного компьютера, так и для мобильного устройства). Таким образом, пользователь посредством высокоуровневого интерфейса может легко адаптировать систему SciVi под конкретную задачу.

Совокупность стандартных элементов навигации по сцене соответствует множеству M, в формальной модели системы научной визуализации (22).

Хотя система SciVi позиционируется прежде всего как средство для автоматизации сложной научной визуализации, она поддерживает и традиционные

виды отображения данных, такие, как диаграммы и графики. Для этого используется описанная ранее библиотека NChart3D.

Библиотеки NGraphics, NChart3D и семейство библиотек VTK как на стороне клиента, так и на стороне сервера реализуют оператор визуализации Φ в составе формальной модели системы научной визуализации (22). Клиент и сервер используют эквивалентные средства визуализации, чтобы в процессе адаптивного разделения рендеринга была возможность производить синтез итогового изображения распределённо, полностью на стороне клиента или полностью на стороне клиента, в зависимости от конкретных условий (типа и быстродействия клиента, скорости сетевого соединения и загруженности сервера).

Описанные графические возможности делают систему SciVi пригодной для решения широкого круга научных задач, требующих как сложной (трёхмерной, многомасштабной) так и традиционной (диаграммы и графики) визуализации данных.

2.9. Выводы по главе

В рамках диссертационного исследования построена формальная модель системы научной визуализации, описывающая все необходимые составляющие для организации автоматизированной интеграции со сторонними решателями, адаптации к специфике различных платформ и обеспечения эффективной высококачественной визуализации. В соответствии с предложенной моделью разработаны методы и средства для создания систем научной визуализации. На их основе создана клиент-серверная система научной визуализации SciVi, внедрённая в пермской IT-компании ООО «Ньюлана». Эта система обладает средствами интеграции с решателями, является мультиплатформенной и в автоматическом режиме осуществляет распределение рендеринга между клиентом и сервером, обеспечивая тем самым высокую интерактивность отображаемой сцены и оптимальную загруженность вычислительных узлов. Сервер SciVi может выполняться на настольных компьютерах и высокопроизводительных

вычислительных комплексах под управлением UNIX-подобных ОС, клиент – на настольных компьютерах под управлением GNU / Linux, Windows и OS X и на мобильных устройствах под управлением iOS и Android.

SciVi предоставляет широкие возможности для наглядного отображения процесса и результатов научных экспериментов. Она имеет удобный интерфейс для регистрации разнородных решателей и готовых данных, подлежащих визуализации. Основанные на методах онтологического инжиниринга механизмы позволяют осуществить автоматизированную интеграцию с зарегистрированными решателями без внесения изменений в исходный код этих решателей, или в исходный код SciVi. Интеграция заключается в том, что SciVi, на основе описанных онтологей знаний, автоматически конвертирует входные и выходные данные решателя в вид, пригодный для визуализации, а также позволяет пользователю управлять работой решателя и отображением итоговой сцены при помощи единого графического интерфейса. Сцена при этом строится параллельно с работой решателя, динамически пополняясь по мере генерации решателем новых данных. Процесс визуализации адаптивно распределён между клиентом и сервером.

Для организации распределения используются эвристические правила, принимающие во внимание производительность клиента, скорость сетевого соединения, загруженность сервера и характер отображаемой сцены. Благодаря этому удалось достичь высокой производительности системы даже в тех условиях, когда клиентом выступает маломощное мобильное устройство, подключенное по низкоскоростному беспроводному сетевому соединению (в среднем скорость отклика системы на команды пользователя сведена к 60 мс). Для передачи данных используется специально разработанный и оптимизированный для системы SciVi сетевой протокол прикладного уровня SVTP.

Благодаря разработанным механизмам упрощения и разбиения 3D-моделей, система SciVi позволяет визуализировать на различных платформах данные, полученные при помощи трёхмерного сканирования.

Проблема построения человеко-машинного интерфейса, не зависящего от платформы и не снижающего производительности рендеринга графической сцены решается при помощи специально разработанной библиотеки GUIBuilder.

Средствами библиотеки NGraphics обеспечивается совместимость системы SciVi с дисплеями высокой чёткости, что, при наличии соответствующей элементной базы, позволяет получать изображения с субпискельной точностью.

Богатые возможности в отношении рендеринга графики, обеспеченные использованием библиотек семейства VTK, библиотеки NGraphics и библиотеки NChart3D позволяют использовать систему SciVi для решения задач в различных предметных областях, требующих как сложную трёхмерную визуализацию с поддержкой многомасштабности, рендеринга сечений и объёмов, сложных визуальных эффектов и анимаций, так и традиционную визуализацию в виде диаграмм и графиков.

Таким образом, система научной визуализации SciVi удовлетворяет сформулированным в главе 1 требованиям, является универсальной и способна увеличить производительность труда учёных-исследователей.

Глава 3. Адаптивное сглаживание границ и центрирование объектов сцены

Для увеличения визуального качества генерируемого изображения в рамках предлагаемого подхода к созданию систем научной визуализации предложено решение следующих важных задач:

1. Сглаживание границ объектов на изображении.
2. Автоматическое определение начального масштаба и положения объектов сцены таким образом, чтобы при заданных начальных углах поворота они наилучшим образом вписались в предназначенную для отображения сцены область экрана.

Практическая реализация предложенного решения осуществлена в системе SciVi. Представленное решение является универсальным и может быть использовано в любых системах компьютерной графики. Таким образом, оно представляют научную и практическую ценность. Разработанные для решения указанных задач алгоритмы были внедрены в пермской IT-компании ООО «Ньюлана» и использованы в целом ряде программных продуктов этой компании, например, в распространяемой на коммерческой основе мультиплатформенной библиотеке NChart3D, заказчиками которой, среди прочих, выступают компании Hewlett Packard, Thomson Reuters, Roche, Citi Bank и Институт генетических исследований Genomics Institute of the Novartis Research Foundation. Копия соответствующего акта о внедрении представлена в приложении 1.

3.1. Адаптивное сглаживание границ объектов на изображении

Поскольку растеризация полигонов трёхмерной сцены является процессом дискретизации, на итоговом изображении неизбежно возникает т. н. алиасинг (*англ.* aliasing) – ступенчатость границ объектов, выровненных по ячейкам буфера кадра. Как следствие, нарушается плавность контуров, то есть не обеспечивается достаточное для комфортного просмотра качество визуализации. Для решения

этой проблемы необходимо производить сглаживание границ, которое носит название антиалиасинг (*англ.* anti-aliasing). Существует большое количество алгоритмов антиалиасинга, некоторые из которых имеют аппаратную поддержку на современных видеокартах. Однако все алгоритмы привносят в процесс визуализации дополнительные накладные расходы, снижая общую производительность графического приложения. Особенно остро проблема производительности стоит на мобильных устройствах.

3.1.1. Проблемы системного сглаживания границ

Наиболее простым для прикладного программиста способом получения сглаженного изображения является использование системного антиалиасинга, предоставляемого большинством современных платформ на уровне встроенного API. В мобильных операционных системах iOS и Android, так же, как и в настольных операционных системах GNU / Linux, Windows и OS X, стандартным алгоритмом антиалиасинга является сглаживание на основе множественной выборки (*англ.* multisampling anti-aliasing, MSAA) [105, 106]. Стандартная реализация этого алгоритма даёт высококачественный результат, однако имеет ряд проблем:

1. Заметное снижение производительности визуализации.
2. Увеличение платформенно-зависимого кода.
3. Нежелательное размытие некоторых объектов.

Первая проблема является наиболее существенной. В результате измерений скорости визуализации было зарегистрировано в среднем трёхкратное снижение производительности при использовании системного антиалиасинга на мобильных устройствах. Так, например, тестовая сцена системы SciVi, представленная на рис. 2, которая состоит из $4{,}8 \cdot 10^6$ вершин, на устройстве iPad 3 без антиалиасинга визуализируется со скоростью 15 FPS (что является достаточным для воспроизведения плавной анимации), а со включенным системным антиалиасингом – со скоростью всего 5 FPS (что уже нарушает плавность

отображения движений согласно [30]). Все контрольные измерения осуществлялись на устройстве iPad 3, так как по состоянию на декабрь 2014 г. его производительность можно принять за среднестатистическую среди мобильных ЭВМ.

Результаты визуализации сцены без сглаживания и с использованием системного сглаживания представлены на рис. 17. Частичным решением проблемы падения производительности может быть выключение антиалиасинга на периоды динамического изменения сцены, так как ступенчатость границ у движущихся объектов менее заметна. Однако включение и выключение системного антиалиасинга в общем случае занимает до 1000 мс.

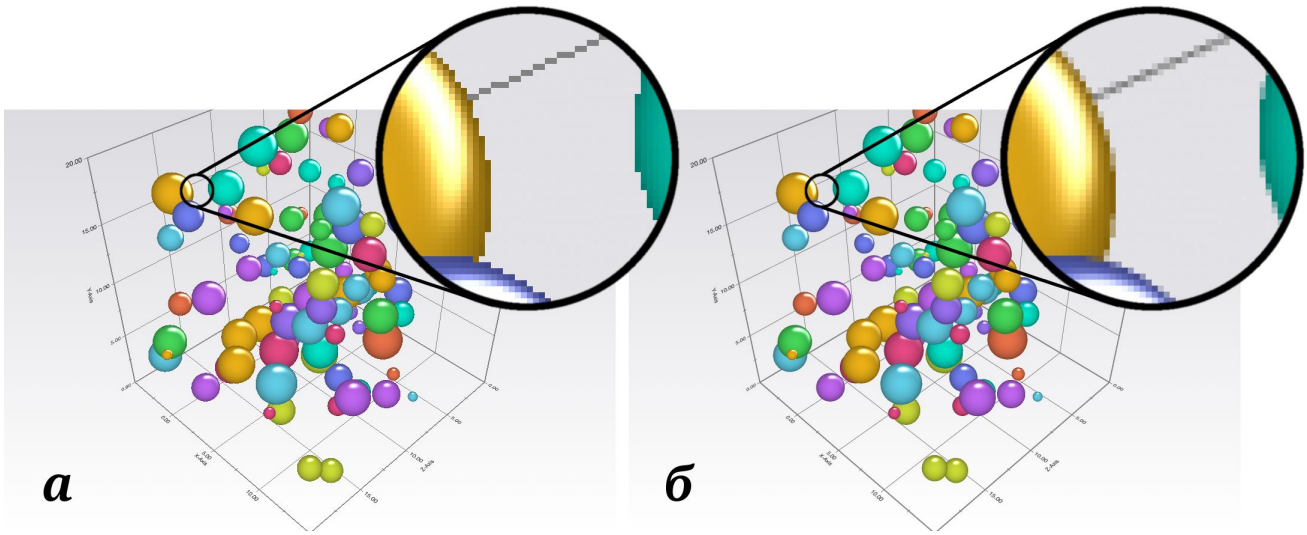


Рис. 17. Тестовая сцена SciVi без использования антилисинга (а) и с использованием системного антиалиасинга (б)

Такая задержка связана с необходимостью пересоздавать структуры данных, связанные с графическим контекстом, так как наличие системного антиалиасинга является неизменяемым параметром этих структур. Пересоздание же влечёт за собой необходимость синхронизации потока визуализации и главного потока приложения. Суммарное время, необходимое на эти операции, согласно произведённым измерениям колеблется от 200 до 1000 мс. Это означает, что начало каждой анимации в среднем на секунду будет отставать от породившего эту анимацию события, включая ситуацию изменения сцены в ответ на команды

пользователя. Столь длительная задержка неприемлема для графических систем реального времени.

Вторая проблема состоит в том, что программный код включения системного сглаживания отличается для iOS, Android и операционных систем настольных компьютеров. Следовательно, его использование привело бы к увеличению платформенно-зависимой части системы, что ввиду требования мультиплатформенности является крайне нежелательным.

Третья проблема проявляется лишь в ситуации, когда на сцене присутствуют объекты, стороны которых строго параллельны сторонам экрана. У линий, образованных сторонами таких объектов, отсутствует ступенчатость, а значит, к ним не нужно применять сглаживание. Однако в том случае, если координаты порождающих эти линии вершин в результате преобразований оказываются дробными, применение MSAA приводит к эффекту размытия. Пример такой ситуации представлен на рис. 18.

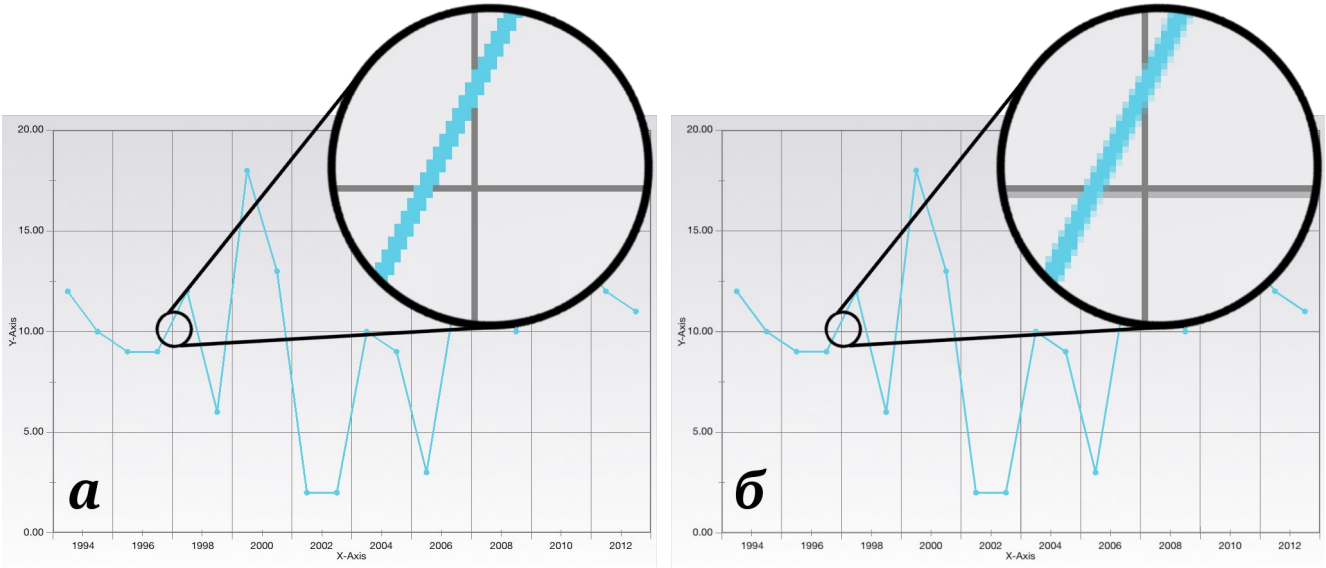


Рис. 18. Сцена без антиалиасинга (а) и сцена с системным антиалиасингом, порождающим нежелательные артефакты размытия вертикальных и горизонтальных линий (б)

Устранить размытие можно тремя основными способами:

1. Подбором координат таким образом, чтобы в результате преобразований они становились целочисленными.

2. Округлением координат в вершинном шейдере.
3. Исключением объектов со строго вертикальными и строго горизонтальными сторонами из множества, на которое воздействует антиалиасинг.

Первый способ устранения размытия не тривиален, а в общем случае и невозможен ввиду того, что начальные положения объектов и их дальнейшие преобразования определяются большим количеством факторов: например, в случае системы SciVi – выходными данными решателя и последующими командами пользователя. Накладывание ограничений на все эти факторы негативно сказалось бы на универсальности системы.

Второй способ, в свою очередь, может привести к потере точности позиционирования линий, а также нагружает вершинный шейдер дополнительными инструкциями.

Третий способ оказывается невозможен ввиду того, что системный антиалиасинг воздействует на весь буфер кадра целиком; его нельзя применить лишь к части сцены.

Перечисленные проблемы возможно решить лишь использованием стороннего алгоритма антиалиасинга, не связанного с системным. При этом используемый алгоритм должен удовлетворять следующим требованиям:

1. Визуальное качество результата не ниже, чем у системного антиалиасинга.
2. Независимость от платформы.
3. Возможность включения / выключения за время, не превышающее время прорисовки одного кадра.
4. Возможность воздействия лишь на часть объектов сцены.

3.1.2. Обзор наиболее распространённых алгоритмов сглаживания границ

Исторически первым алгоритмом антиалиасинга является сглаживание на основе увеличенного изображения (*англ.* supersampling anti-aliasing, SSAA) [106, 107]. Этот алгоритм базируется на достаточно очевидной идее о том, что сглаживание границ можно обеспечить путём увеличения разрешения

изображения. Визуализация сцены выполняется в буфер кадра, превосходящий по размеру разрешение экрана, а затем изображение из этого буфера уменьшается. Под экраном здесь (и далее) понимается та область, на которую в итоге будет выведено изображение. В процессе уменьшения происходит интерполяция соседних пикселей, что приводит к сглаживанию ступенек на границах объектов. Чем больше размер промежуточного изображения по сравнению с размером экрана, тем выше оказывается качество итога визуализации, однако увеличиваются и накладные расходы на дополнительные вычисления и хранение сопутствующих структур данных. SSAA очень прост в реализации, способен обеспечить высокое качество результата, однако в значительной степени снижает производительность графической программы.

Развитием SSAA является алгоритм сглаживания на основе множественной выборки (*англ.* multisampling anti-aliasing, MSAA) [105, 106]. Идея оптимизации состоит в том, что рассматривается расширенная выборка (*англ.* supersampling) лишь отдельных составляющих буфера кадра. Как правило, разрешение увеличивается у буферов глубины и трафарета, а разрешение буфера цвета берётся равным разрешению экрана. Данные для вычисления значения в каждой ячейки буфера цвета включают в себя значения глубины и трафарета, усреднённые по нескольким соседним точкам из соответствующих буферов. Таким образом, с одной стороны, цвет вычисляется по уточнённой выборке данных (что, в свою очередь уменьшает ступенчатость границ), а с другой – сокращается количество запусков фрагментного шейдера (оно остаётся таким же, как и при рендеринге картинки без сглаживания). MSAA является очень популярным алгоритмом, имеет аппаратную поддержку на большинстве современных видеокарт и используется в качестве встроенного системного средства сглаживания границ на многих платформах, например, на iOS, Android, GNU / Linux, Windows и OS X.

Развитием MSAA является алгоритм сглаживания на основе множественной выборки покрытия (*англ.* coverage-sampling anti-aliasing, CSAA) [106, 108], разработанный компанией NVidia. В этом алгоритме расширение выборки

производится не на всём буфере, а только в некоторой окрестности границ полигонов. Поиск границ осуществляется аппаратно, что позволяет увеличить производительность этого алгоритма по сравнению с MSAA. Выборка в окрестности пикселей, которые были идентифицированы как граничные, очень сильно зависит от фактического расположения соответствующих найденной границе полигонов. Таким образом, в некоторых случаях визуальное качество результата у CSAA оказывается выше, чем у MSAA, а в некоторых – ниже.

Другим направлением развития MSAA является т. н. отложенный MSAA (*англ.* deferred MSAA, DMSAA) [109]. Этот алгоритм использует расширенную выборку для фильтрации уже визуализированного изображения. Он состоит из трёх шагов:

1. Визуализация всей геометрии сцены без использования материалов и освещения, с записью только в буфер глубины. При этом используется обычный MSAA.
2. Визуализация сцены в обычном режиме без сглаживания.
3. Применение «отложенного антиалиасинга», в процессе которого используются данные глубины, собранные на первом шаге, и на их основе осуществляется фильтрация изображения.

Отложенный MSAA легко может быть реализован на настольном компьютере, обеспечивая высокое качество и достаточно высокую производительность. Однако он не совместим с мобильными устройствами под управлением iOS и Android, так как в используемых там библиотеках стандарта OpenGLES на данный момент отсутствует доступ на чтение к буферу глубины.

Ещё одним способом сглаживания границ является морфологический антиалиасинг (*англ.* morphological anti-aliasing, MLAA) [110]. Он не зависит от графического конвейера, поэтому может быть использован в подходах, основанных как на растеризации полигонов, так и на трассировке лучей. По своей сути MLAA является фильтром постобработки изображения. Его идея состоит в поиске на изображении фрагментов ступенчатых границ по заранее определённым

шаблонам. Затем цвета пикселей найденных фрагментов усредняются по определённым правилам, зависящим от соответствующих шаблонов. Этот алгоритм может быть эффективно распараллелен на GPU и в этом случае обеспечивает высокую производительность. Однако на мобильных устройствах он оказывается неприменим ввиду того, что они на данный момент не имеют GPU общего назначения (*англ.* general-purpose graphics processing units, GPGPU).

Развитием MLAA является морфологический антиалиасинг с субпиксельной точностью (*англ.* subpixel morphological anti-aliasing, SMAA) [111], разработанный в компании CryTech. Данный алгоритм использует локальный анализ контраста на изображении, чтобы обеспечить более надёжный поиск границ, а также более эффективные приёмы для обработки острых углов и диагональных линий. За счёт этого удаётся улучшить распознавание контуров на изображении, увеличивая качество сглаживания с сохранением резкости объектов. Однако в силу проблем с производительностью SMAA, так же как и MLAA, на данный момент не может быть использован на мобильных устройствах в графических системах реального времени.

В том случае, если на сцене присутствует движение, может быть использован временной антиалиасинг (*англ.* temporal anti-aliasing, TXAA) [112]. Его идея состоит в использовании выборки данных на соседних кадрах анимационной последовательности, чтобы получить расширенную выборку по времени. TXAA позволяет сгладить границы движущихся объектов, если разность между соседними кадрами в анимационной последовательности достаточно мала. Алгоритм может быть обобщён и на неподвижные сцены: движение имитируется путём нескольких рендерингов с эффектом дрожания камеры. Математически этот эффект достигается путём внесения небольших смещений в матрицу проекции на каждом кадре. Затем полученные изображения усредняются. Результирующее изображение, как правило, обладает высоким визуальным качеством, иногда даже превосходящим результат MSAA. Однако проблемами TXAA являются производительность и затраты памяти, так как он требует произвести несколько

рендерингов для сглаживания одного изображения. В контексте мобильных устройств, где вызовы API-функций отрисовки занимают достаточно много времени, более оптимальным оказывается произвести визуализацию одного изображения с большим разрешением (принцип SSAA), чем визуализацию нескольких изображений с разрешением экрана.

Ещё одним алгоритмом, основанным на постобработке изображения, является быстрый аппроксимированный антиалиасинг (*англ.* fast approximate anti-aliasing, FXAA) [113]. Он разработан Т. Лоттесом (T. Lottes) из компании NVidia. Идея алгоритма заключается в том, чтобы в процессе однопроходной фильтрации найти на изображении ступеньки границ и размыть их в перпендикулярном к ним направлении. Благодаря тому, что алгоритм работает с полностью подготовленным изображением, он в состоянии сгладить абсолютно все имеющиеся границы, включая границы внутри областей, подвергнувшихся альфа-смешиванию, и границы, являющиеся следствием эффектов освещения – теней, бликов и т. д. Пиксели в таких областях оказываются недоступными для алгоритмов, основанных на обработке геометрических данных, таких как MSAA. Однако это достоинство порождает и соответствующий недостаток: текстуры, выровненные на пиксели экрана (например, части оформления экрана, текст и т. д.) также подвергаются воздействию FXAA и могут быть размыты. Чтобы решить эту проблему, необходимо использовать послойный рендеринг. Сначала следует визуализировать объекты, нуждающиеся в сглаживании, применить к ним FXAA, а затем поверх отобразить всё то, что не требует сглаживания. FXAA основан на конволюции изображения, поэтому может быть реализован во фрагментном шейдере и быстро интегрирован в любую графическую систему. Его исходный код был опубликован Т. Лоттесом под лицензией Public Domain [114].

На основе проведённого обзора составлена следующая классификация алгоритмов сглаживания границ:

1. Алгоритмы времени рендеринга (например, SSAA, MSAA и CSAA);
2. Алгоритмы постобработки (например, MLAA, SMAA и FXAA);

3. Алгоритмы, использующие многопроходный рендеринг (например, DMSAA и TXAA).

В результате анализа сделан вывод, что в контексте темы диссертационного исследования из списка известных алгоритмов для использования на мобильных устройствах наиболее хорошо подходят SSAA и FXAA ввиду их совместимости с аппаратной базой мобильных устройств, приемлемой производительности и высокого визуального качества. Для разработки мультиплатформенных систем научной визуализации решено использовать именно их [115, 116].

3.1.3. Оценка сложности алгоритма визуализации сцены

Для прогнозирования производительности алгоритмов антиалиасинга, необходимо оценить их теоретическую сложность. Поскольку антиалиасинг является дополнительным действием в рендеринге сцены, он всегда будет увеличивать сложность алгоритма визуализации и при этом в общем случае зависеть от входных параметров этого алгоритма. Поэтому первым шагом следует оценить сложность самой визуализации.

Пусть A – алгоритм визуализации. Предполагается, что визуализация сцены происходит средствами программируемого графического конвейера, например, с использованием библиотеки стандарта OpenGL. Однако, ввиду схожих принципов организации различных низкоуровневых графических API, полученный результат будет справедлив и для библиотек других стандартов. Кроме того, предполагается, что графический конвейер содержит только два программируемых этапа – этап преобразования вершин (вершинный шейдер) и этап расцвечивания фрагментов (фрагментный шейдер). Такая схема работы является типичной для систем, совместимых со стандартом OpenGLES 2.0, к которым относится и система научной визуализации SciVi. В этом случае время, требуемое на визуализацию одного кадра без учёта параллелизма на GPU, может быть определено как

$$T(A) = \sum_{v=1}^{n} \zeta(v) + \sum_{f=1}^{m} \xi(f) + T_s, \tag{28}$$

где n – число вершин на сцене,

$\zeta(v)$ – время выполнения вершинного шейдера для вершины v,

m – число фрагментов, получившихся в результате растеризации сцены,

$\xi(f)$ – время выполнения фрагментного шейдера для фрагмента f,

T_s – совокупное время на системные операции, такие как очистка буферов, переключение активного и видимого буферов в процессе организации двойной буферизации, время на вызовы API-функций и т. п.

Последнее слагаемое в формуле (28) в большинстве случаев оказывается достаточно мало относительно суммарного времени выполнения шейдеров и слабо зависит от количества обрабатываемых вершин и фрагментов, поэтому в оценке асимптотической сложности его можно опустить. Различия во времени выполнения вершинного шейдера для разных вершин в большинстве случаев оказываются незначительными, поэтому асимптотическую сложность этого шейдера можно оценить как $O(1)$. Различия во времени выполнения фрагментного шейдера для разных фрагментов также можно считать постоянным и аналогично оценить асимптотическую сложность этого шейдера как $O(1)$. Следовательно, асимптотическая сложность визуализации может быть выражена как

$$\tau(A) = O(n + m). \tag{29}$$

Число вершин, обрабатываемых вершинным шейдером, для каждой сцены известно заранее.

Число фрагментов, обработанных фрагментным шейдером, зависит от начального расположения полигонов и применённых к ним преобразований. В общем случае определить его заранее невозможно, поэтому необходимо рассмотреть минимальное, максимальное и среднее значения, определив тем самым минимальную, максимальную и среднюю сложности визуализации.

Минимальным числом фрагментов является $m = 0$, что соответствует ситуации, когда все полигоны в результате преобразований оказались за пределами экрана.

Пусть вершины объединяются только в треугольники (как того требуют стандарты низкоуровневого графического API). Пусть также отсутствует переиспользование вершин, то есть каждая вершина входит в один и только один треугольник. Тогда число треугольников на сцене $p = \frac{n}{3}$. Любая сцена, состоящая из многоугольников с числом углов больше 3 и любая сцена, где одна вершина может входить в несколько многоугольников, на этапе преобразования вершин сводится к описанному случаю, поэтому сделанное допущение не нарушает общности.

Максимальным числом фрагментов является $m = p \cdot w \cdot h$, здесь и далее w, h – ширина и высота экрана в пикселях соответственно. Такая ситуация возможна в том случае, если каждый полигон закрывает собой весь экран, и все полигоны ориентированы на наблюдателя (либо отключено отсечение задних граней) и выводятся в порядке убывания глубины (либо отключен тест буфера глубины).

Обе описанных граничных ситуации являются очень редкими и почти не встречаются на практике. Согласно произведённым наблюдениям на примере различных систем, в том числе, системы SciVi, чаще всего полигонами бывает занята лишь половина пространства экрана. При этом каждый фрагмент переписывается в среднем k раз, где k – среднее число объектов на сцене, стоящих друг за другом и выводящихся, начиная с самого дальнего (в том случае, если включены отсечение задних граней и тест глубины). Таким образом, число фрагментов в среднем случае составляет

$$m = \frac{k \cdot w \cdot h}{2}. \tag{30}$$

Как правило, числовое значение $\frac{k}{2}$ бывает достаточно мало по сравнению с w, h и n, поэтому при оценке асимптотической сложности его можно опустить.

Здесь и далее среднее число вызовов фрагментных шейдеров предполагается равным $w \cdot h$. На основе этого предположения оценка (29) примет вид

$$\tau(A) = O(n + w \cdot h). \tag{31}$$

Полученная оценка средней сложности визуализации сцены использована для анализа сложности алгоритмов сглаживания.

3.1.4. Предлагаемый метод сглаживания границ

Как уже отмечалось выше, из списка проанализированных алгоритмов антиалиасинга по критерию совместимости с мобильными устройствами были выбраны SSAA и FXAA. Однако по отдельности эти алгоритмы не обеспечивают достаточного визуального качества результата, поэтому был разработан новый алгоритм как суперпозиция существующих (с модификациями, которые будут описаны ниже). Результирующие изображения, полученные без использования сглаживания и с использовании различных алгоритмов сглаживания представлены на рис. 19.

Алгоритм SSAA прост в реализации, однако имеет ряд проблем:

1. Ограниченность размера промежуточного изображения.
2. Снижение производительности визуализации и увеличение затрат памяти.
3. Недостаточно высокое визуальное качество, в особенности на мобильных устройствах.

Первая проблема состоит в том, что максимальный размер визуализируемого изображения ограничен на аппаратном уровне. Для того, чтобы осуществить сглаживание методом SSAA, сначала сцена визуализируется в текстуру, стороны которой в s раз больше сторон экрана. Коэффициент масштабирования s при этом называется коэффициентом суперсемплинга (*англ.* supersampling ratio). Затем эта текстура накладывается на прямоугольник размером с экран, и границы объектов сглаживаются за счёт фильтрации при уменьшении изображения.

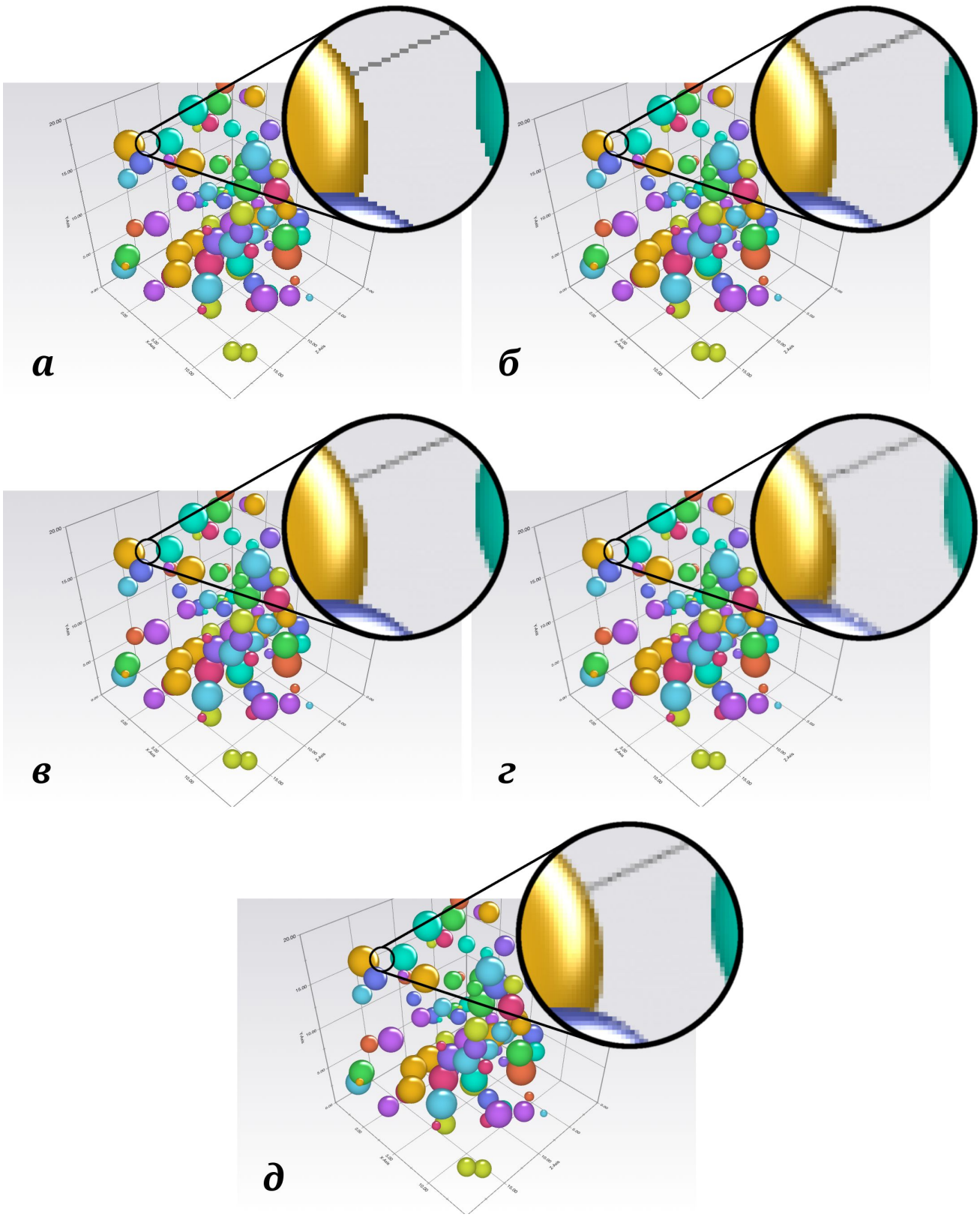


Рис. 19. Тестовая сцена без сглаживания (а), со сглаживанием методами MSAA (б), SSAA (в), FXAA (г) и суперпозицией методов SSAA и FXAA (д)

Чем больше s и чем более качественным является алгоритм фильтрации, тем выше качество результата. Однако максимальный размер текстуры имеет аппаратное ограничение.

Например, максимальный размер текстуры на большинстве мобильных устройств составляет 2048x2048 пикселей, хотя на устройствах iPad поколения 3 и выше поддерживаются текстуры размером в 4096x4096 пикселей. При этом, однако, и разрешение экрана на таких устройствах также является достаточно высоким: iPad 3 и iPad 4 имеют ретина-дисплей [26] с разрешением 2048x1536 пикселей. Следовательно, максимальный коэффициент суперсэмплинга для этих устройств равен 2.

Устройство iPad 2 не имеет ретина-дисплея, разрешение его экрана составляет 1024x768 пикселей, однако максимальный поддерживаемый размер текстуры равен 2048x2048, то есть максимальный коэффициент суперсэмплинга для него также равен 2. С другой стороны, разрешение экрана мобильного устройства Asus Transformer TF101 составляет 1280x800, при этом максимальный размер текстуры так же, как и у iPad 2, равен 2048x2048, следовательно, максимальный коэффициент суперсэмплинга составляет всего 1,6. Для увеличения коэффициента суперсемплинга можно производить рендеринг сцены по частям, однако, во-первых, использование нескольких проходов визуализации ещё больше снижает производительность, а во-вторых, в общем случае неизбежно возникновение дефектов изображения на границах текстур ввиду ограниченной точности чисел с плавающей точкой. Поэтому в предлагаемом алгоритме используется адаптивное вычисление коэффициента суперсемплинга по формуле

$$s = \min\left\{\max\left\{s',1\right\},2\right\}, \tag{32}$$

$$s' = \begin{cases} \dfrac{w_t}{w}, & w > h \\ \dfrac{h_t}{h}, & w \le h \end{cases}, \tag{33}$$

где w_t, h_t – максимальные ширина и высота текстуры в пикселях.

Коэффициент s заключается в отрезок $[1;2]$ потому, что, согласно проведённым экспериментам, приемлемое сочетание скорости и качества достигается при $s = 2,$ и в соответствии с идеей алгоритма SSAA не имеет смысла использовать $s < 1$. Ситуация, когда максимальный поддерживаемый размер текстуры на устройстве меньше разрешения экрана, является исключительной, и в этом случае антиалиасинг запрещается полностью.

Вторая проблема алгоритма SSAA состоит в снижении производительности визуализации и увеличении затрат памяти. При использовании SSAA в процессе подготовки промежуточной текстуры количество вызовов фрагметного шейдера увеличивается в s^2 раз (так как увеличивается площадь изображения). Затем, при выводе прямоугольника с подготовленной текстурой происходит 6 дополнительных вызовов вершинного шейдера (для обработки 6 вершин 2 треугольников, из которых формируется прямоугольник) и $w \cdot h$ вызовов фрагментного шейдера (вычисление цвета каждой точки растеризованного прямоугольника). Таким образом, среднее число итераций шейдеров при использовании SSAA для сглаживания границ объектов сцены составляет $(n+6)+(s^2+1)w \cdot h,$ и оценка асимптотической сложности имеет вид

$$\tau(SSAA) = O(n + s^2 \cdot w \cdot h). \quad (34)$$

Скорость визуализации полноэкранного изображения на устройстве iPad 3 с использованием SSAA составляет 4,5 FPS. Результат представлен на рис. 19 (в). Производительность SSAA близка к производительности системного MSAA: как отмечалось выше, тестовая сцена с использованием MSAA перерисовывается на iPad 3 со скоростью 5 FPS. Затраты оперативной памяти при использовании дополнительного буфера кадра размером 4096x4096, содержащего буфер цвета и 16-битный буфер глубины, составляют всего 5 Кб, так как основной объём данных хранится в видеопамяти. Следовательно, можно утверждать, что SSAA пригоден для мобильных устройств, так как его производительность близка к производительности системного MSAA. На настольных компьютерах, ввиду их

большей вычислительной мощности, производительность этого алгоритма оказывается как минимум не ниже, чем на мобильных устройствах. Это означает, что он может быть использован в мультиплатформенных системах.

Третья проблема SSAA является самой серьёзной: качество итогового изображения, полученного с помощью SSAA, в общем случае уступает качеству изображения, полученного с использованием системного MSAA. На настольных компьютерах поддерживаются текстуры большого разрешения, а значит, можно выставлять большие значения для коэффициента суперсемплинга. Кроме того, поддерживаются различные режимы фильтрации текстур, способные повысить качество сглаживания в процессе уменьшения изображения. Однако на мобильных устройствах размер текстур, как уже было отмечено, сильно ограничен, а для фильтрации используется лишь билинейная интерполяция. Следовательно, в силу объективных ограничений один SSAA не может обеспечить требуемое визуальное качество.

Алгоритм FXAA является фильтром однопроходной постобработки и может быть реализован во фрагментном шейдере, который затем используется для наложения текстуры с изображением сцены на прямоугольник размером с экран. Таким образом, он привносит в обработку сцены 6 дополнительных вызовов вершинного и $w \cdot h$ дополнительных вызовов фрагментного шейдера (что соответствует визуализации прямоугольника), то есть среднее количество итераций шейдеров при использовании FXAA составляет $(n+6)+2 \cdot w \cdot h.$ С учётом формулы (29) асимптотическая сложность FXAA не отличается от обычной визуализации и выражается формулой (31). Однако на практике удвоение числа вызовов фрагментного шейдера даёт ощутимое снижение производительности.

Как отмечалось выше, исходный код шейдера FXAA распространяется по лицензии Public Domain, поэтому может быть легально использован в любых проектах. Первоначальный вариант шейдера FXAA, реализованный Т. Лоттесом, ориентирован на настольные компьютеры и имеет большое количество настроек,

не совместимых со стандартом OpenGLES. Упрощённая версия шейдера FXAA предоставлена под той же лицензией Дж. Гинотом (J. Guinot) [117]. Хотя представленный код изначально не совместим с ESSL, адаптация состоит лишь в замене функций, которые не поддерживаются в ESSL, на соответствующие аналоги. Результат работы кода, адаптированного таким образом к использованию на мобильных устройствах, представлен на рис. 19 (г). Скорость перерисовки тестовой сцены на устройстве iPad 3 составила 8 FPS. При этом качество сильно уступает как SSAA, так и системному MSAA.

Автором диссертационного исследования произведён ряд оптимизаций кода шейдера FXAA, в результате которых удалось, во-первых, увеличить его производительность (на тестовой сцене скорость перерисовки возросла до 10 FPS), а во-вторых, повысить визуальное качество.

Первым шагом оптимизации является перенос части вычислений из фрагментного шейдера в вершинный, так как вершинный шейдер выполняется всего 6 раз (чтобы вычислить координаты вершин двух треугольников, составляющих полноэкранный прямоугольник), а фрагментный – столько раз, сколько пикселей на итоговом изображении. В вершинный шейдер был перенесён расчёт смещения текстурных координат для получения доступа к пикселям, соседним по отношению к обрабатываемому. Эти смещения передаются во фрагментный шейдер в виде varying-переменных. Кроме того, все векторы-константы были записаны с атрибутом const, что исключает вызов их конструктора во время выполнения шейдера. Согласно проведённым тестам производительности, компилятор шейдеров на операционной системе iOS не оптимизирует постоянные векторы, если они не промаркированы данным атрибутом. Прирост производительности после этих оптимизаций составил примерно 0,4 FPS.

Следующим шагом оптимизации является устранение всех обращений к отдельным компонентам векторов в пользу обработки векторов целиком. Дело в том, что обращение к компонентам приводит к добавлению в машинный код

шейдера дополнительных инструкций по декомпозиции, что на большом количестве вызовов фрагментного шейдера (порядка 4 миллионов раз за кадр на iPad 3) снижает скорость визуализации на 0,1-0,2 FPS.

Дальнейшая оптимизация касается изменения всех формул таким образом, чтобы они удовлетворяли шаблону вида

$$D = A \cdot B + C, \tag{35}$$

где D, A, B, C – действительные числа.

Большинство графических процессоров обладают так называемыми инструкциями умножения-сложения (*англ.* multiply-add, MAD) [118], что позволяет вычислять выражения вида (35) за один такт. Некоторые компиляторы шейдеров производят оптимизацию формул автоматически. Однако, согласно проведённым наблюдениям, компилятор, используемый на iOS, такой функцией не обладает, поэтому формулы были преобразованы вручную. Переход к формулам указанного вида привёл к повышению производительности примерно на 0,7 FPS.

Ещё одной проблемой, замедляющей выполнение шейдера FXAA, является наличие в нём динамических ветвлений. Так как графические процессоры построены на принципах SIMD-архитектуры [119], они поддерживают лишь параллелизм по данным. Наличие ветвлений в коде приводит к тому, что потенциально разные данные могут требовать выполнения разных инструкций, а это, в свою очередь, влечёт за собой нарушение параллелизма и снижение производительности. Для организации условных вычислений без ветвлений на GPU используется инструкция условного перемещения (*англ.* conditional move) [119], которая позволяет копировать данные из одного места в другое только в том случае, если истинно некоторое условие. На уровне шейдерного языка, обеспечивающего абстракцию от графического процессора, для организации условных вычислений используются функции поиска минимума / максимума (*англ.* min, max) и пороговые функции (*англ.* step). Замена

ветвлений этими функциями обеспечила прирост производительности примерно на 0,7 FPS.

Кроме того, для линейной интерполяции использована специальная встроенная функция (*англ.* mix), имеющая аппаратную поддержку. Это увеличило производительность ещё примерно на 0,1 FPS.

Общий прирост производительности визуализации с использованием FXAA после произведённых оптимизаций составил примерно 2 FPS, то есть скорость перерисовки сцены с 8 FPS возросла до 10.

Кроме того, был модифицирован и сам алгоритм FXAA. В оригинале этот алгоритм для каждой точки определяет новый цвет, исходя из её начального цвета и цвета точек, соседних с ней. При этом новый цвет полностью заменяет старый. В предложенной модификации точка приобретает цвет, являющийся усреднённым значением начального и вычисленного (используется цветовая модель RGB). Такой подход увеличивает визуальное качество результирующего изображения за счёт снижения шума на границах объектов.

Однако даже с описанной модификацией качество у FXAA на мобильных устройствах остаётся ниже, чем у MSAA. Проблема низкого качества была решена путём суперпозиции алгоритмов SSAA и FXAA. Результат работы этих двух алгоритмов после их суперпозиции представлен на рис. 19 (д). Согласно произведённому сравнению, такая суперпозиция обеспечивает качество не хуже, а иногда даже лучше чем MSAA.

Идея организации суперпозиции состоит в том, что сначала сцена визуализируется в текстуру, размер которой превосходит размер экрана с коэффициентом масштабирования s, вычисленным по формулам (32), (33), то есть используется SSAA, а затем эта текстура накладывается на прямоугольник размером с экран с использованием шейдера FXAA. Теоретическая сложность суперпозиции равна теоретической сложности SSAA и выражается формулой (34), так как количество итераций шейдеров не изменилось, а изменился лишь программный код вершинного и фрагментного шейдеров, при помощи которых

выводится прямоугольник. Этот теоретический результат был подтверждён на практике: скорость прорисовки тестовой сцены с использованием суперпозиции SSAA и FXAA на iPad 3 составила 4,5 FPS, что равно скорости при использовании SSAA.

Для экономии энергии и предотвращения перегрева мобильного устройства визуализация производится только в моменты изменения сцены. Изменения могут быть вызваны либо командами пользователя, либо событиями, произошедшими в процессе исполнения программы (срабатывание таймера, получение новых данных по сети и т. д.). Все изменения визуальных свойств объектов автоматически отслеживаются на уровне библиотеки NGraphics и приводят к планированию очередной перерисовки.

Сглаживание автоматически отключается на периоды обновления сцены, чтобы обеспечить воспроизведение плавной анимации и быстрый отклик системы на команды пользователя. Как только обновление заканчивается, сглаживание автоматически включается вновь. Таким образом, все кадры, в которых отображаются промежуточные состояния меняющихся объектов, визуализируются без использования антиалиасинга, а сглаживается только изображение на финальном кадре. При этом задержка реакции системы на изменение состояния сцены составляет не больше, чем время прорисовки одного кадра, то есть порядка 60 мс. Это, в свою очередь, удовлетворяет оговоренному ранее минимуму скорости для графических систем реального времени [30].

При этом на уровне API библиотеки NGraphics доступна настройка, включающая постоянный антиалиасинг. Эта настройка может быть использована в том случае, если сцена является достаточно простой и быстро визуализируется даже с постоянно включенным сглаживанием. Оценить максимальную сложность сцены, допустимую для постоянной работы антиалиасинга, является крайне нетривиальной задачей, так как на производительность рендеринга влияет не только количество вершин и полигонов сцены, но и сложность используемых шейдеров, количество вызовов графического API для отправки данных на

конвейер, количество изменений состояний графического конвейера (машины состояний), объём сопутствующих вычислений при трансформациях объектов и т. п. В связи с этим настройка постоянного антиалиасинга вынесена на публичный интерфейс библиотеки, а не определяется автоматически.

Для решения проблемы комбинирования на одной сцене объектов, требующих и не требующих сглаживания, используется послойная визуализация. Результат работы послойной визуализации в сравнении с результатом работы системного MSAA представлен на рис. 20.

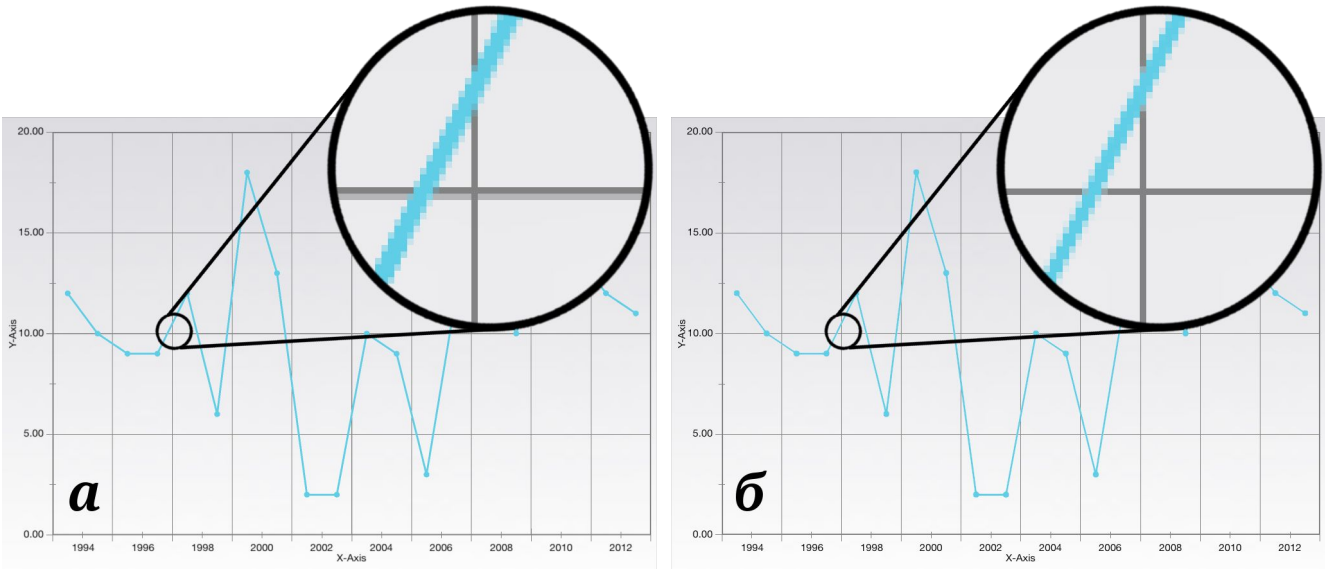


Рис. 20. Сцена, полученная с использованием системного MSAA (а) и предложенной суперпозиции SSAA и FXAA в комбинации с послойной визуализацией (б)

На рис. 20 продемонстрирована сцена, аналогичная сцене на рис. 18, где было продемонстрировано различие изображений, полученных без сглаживания и с использованием системного MSAA. Линии сетки на этой сцене параллельны сторонам экрана и потому выровнены на пиксели изображения. Согласно настройкам, они должны иметь толщину в один пиксель, однако сглаживание иногда может размывать их. На увеличенном фрагменте рис. 20 (а) видно, что вертикальная линия имеет толщину в один пиксель, а горизонтальная линия оказалась размыта: под ней проходит ещё один ряд пикселей меньшей интенсивности цвета. В случае использования системного MSAA такая ситуация в

общем случае неизбежна, так как этот алгоритм воздействует на все объекты сцены.

Как уже отмечалось выше, параллельные сторонам экрана линии оказываются размытыми, если порождающие их вершины получили в результате преобразований дробные координаты. Подбор начальных координат и их преобразований таким образом, чтобы в результате получались целые числа, является нетривиальной задачей ввиду того, что преобразованиями сцены, вообще говоря, управляет пользователь, а начальные координаты объектов определяются входными данными (в случае системы SciVi – данными, полученными от решателя). Принудительное округление также не представляется возможным, поскольку финальный этап преобразований (преобразование порта просмотра) происходит на неуправляемом этапе графического конвейера, поэтому у программиста нет возможности его переопределить. Принудительное округление на этапе работы вершинного шейдера, в свою очередь, может привести к потере точности координат и неверному позиционированию соответствующих вершин.

Предложенная послойная визуализация позволяет автоматически группировать объекты в слои, исходя из их взаимного расположения и необходимости сглаживания. Затем слои последовательно визуализируются, а сглаживание включается только для тех из них, для которых оно необходимо. На рис. 20 (б) линии графика относятся к слою переднего плана, для которого включено сглаживание, а линии сетки находятся на слое заднего плана, для которого сглаживание отключено.

Ограничение послойного рендеринга состоит в том, что требующие и не требующие сглаживания объекты не могут пересекаться, так как сглаживаемые объекты визуализируются в текстуру, которая затем накладывается на прямоугольник, размер которого равен размеру экрана.

Ещё одним ограничением описанного подхода является затруднённая совместимость с полупрозрачными объектами. На рис. 21 представлена сцена, составленная из трёх слоёв.

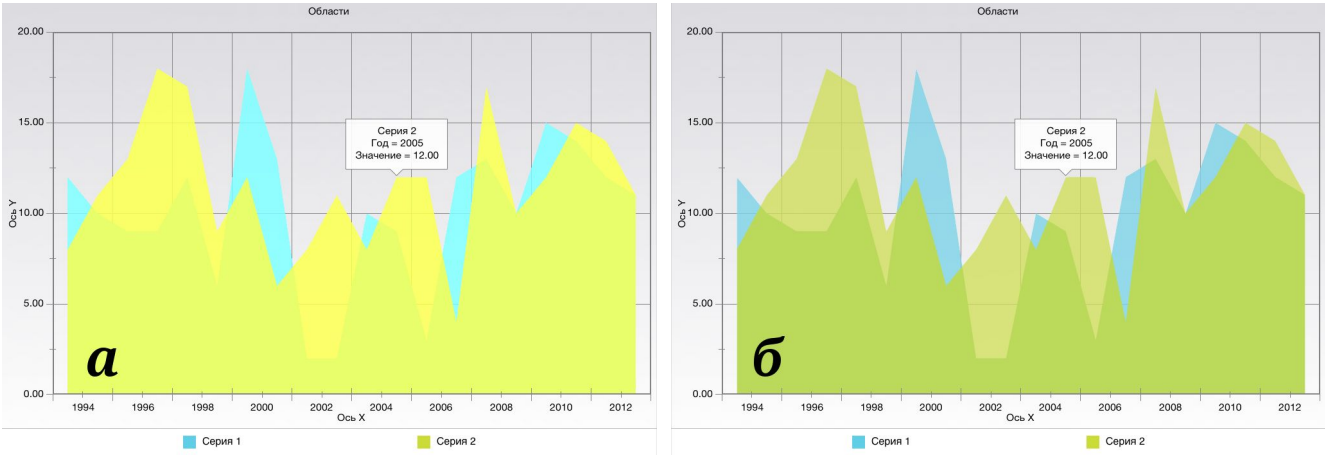


Рис. 21. Визуализация сцены с использованием альфа-смешивания без антиалиасинга (а) и с антиалиасингом, обеспеченным суперпозицией SSAA и FXAA (б)

Нижний слой содержит сетку графика, средний слой содержит две серии данных, представленных закрашенными областями, а верхний слой содержит заголовок, легенду и всплывающую подсказку. Слои отображаются друг за другом с использованием альфа-смешивания цветов по формуле:

$$\begin{cases} c = \alpha_{src} \cdot c_{src} + (1 - \alpha_{src}) \cdot c_{dst}, \\ \alpha = \alpha_{src} + (1 - \alpha_{src}) \cdot \alpha_{dst} \end{cases} \quad (36)$$

где c – результирующий вектор красной, зелёной и синей составляющих цвета точки,

c_{dst} – вектор красной, зелёной и синей составляющих цвета точки объекта, который был визуализирован раньше текущего (данные о точке из буфера цвета),

c_{src} – вектор красной, зелёной и синей составляющих цвета точки текущего объекта (цвет точки, вычисленной в результате прохода текущей порции данных по графическому конвейеру),

α – результирующее значение альфа-канала,

α_{dst} – значение альфа-канала точки объекта, который был визуализирован раньше текущего,

α_{src} – значение текущего альфа-канала точки.

Все значения лежат в отрезке $[0;1]$.

На рис. 21 (а) антиалиасинг отключен для всех слоёв. На рис. 21 (б) антиалиасинг включен для слоя, содержащего полупрозрачные диаграммы-области. Как можно заметить, результаты альфа-смешивания на продемонстрированных изображениях очень сильно различаются. Различие объясняется тем, что альфа-смешивание не коммутативно, не ассоциативно и в представленных на рис. 21 ситуациях выполняется в различном порядке.

При условии отсутствия антиалиасинга порядок визуализации групп объектов таков:

1. Фон (градиентная заливка и сетка графика).
2. Первая диаграмма-область.
3. Вторая диаграмма-область.
4. Легенда, заголовок и всплывающая подсказка.

Объекты из первой группы составляют первый слой, объекты из второй и третьей группы – второй и объекты из четвёртой – третий. Порядок вывода объектов из четвёртой группы в данном примере не имеет значения, так как они не пересекаются. В том случае, если для второго слоя включен антиалиасинг, порядок визуализации будет таким:

1. Фон (градиентная заливка и сетка графика).
2. Прямоугольник, на который наложена текстура с результатом визуализации первой и второй диаграммы-области после применения антиалиасинга.
3. Легенда, заголовок и всплывающая подсказка.

Порядок альфа-смешивания диаграмм-областей и фона в приведённых последовательностях визуализации различается, а, следовательно, различаются и результирующие изображения. Однако для реализации идеи отключения антиалиасинга на периоды динамического изменения сцены необходимо, чтобы эти изображения были максимально похожи, так как от их сходства напрямую зависит эффект плавности перехода от сглаженной версии изображения к несглаженной. Для обеспечения идентичности, в свою очередь, необходимо, чтобы порядок альфа-смешивания (а, следовательно, и порядок визуализации) не

изменялся. Единственным эффективным способом сделать порядок неизменным является использование рендеринга в текстуру в обои случаях – то есть даже тогда, когда антиалиасинг отключен. В случае отключенного антиалиасинга, однако, с текстурой не производится никаких дополнительных преобразований, поэтому накладные расходы сводятся к минимуму. По результатам выполненных измерений, использование дополнительного рендеринга в текстуру снижает производительность не более чем на 3 FPS, что в большинстве случаев является приемлемым и не нарушает плавности отображения движений. Библиотека NGraphics автоматически определяет, присутствуют ли на слое со включенным антиалиасингом объекты, для которых $\alpha < 1$ (то есть будет производиться альфа-смешивание), и если такие объекты были обнаружены, использует для данного слоя принудительный рендеринг в текстуру.

Предложенный метод адаптивного сглаживания позволил обеспечить высокое качество результирующего изображения, не увеличивая объёма платформенно-зависимого кода графической системы и не снижая производительности рендеринга в моменты, критичные к скорости перерисовки сцены. Предложенная суперпозиция алгоритмов SSAA и FXAA является практической реализацией оператора Δ, введённого в формальной модели системы научной визуализации (22).

3.2. Центрирование объектов на экране

Визуализация сцены и элементов графического интерфейса пользователя выполняется единообразно, чтобы исключить накладные расходы на переключение графических контекстов и тем самым повысить производительность рендеринга. В соответствии со сгенерированным описанием графического интерфейса, на экране выделяются специальные зоны для элементов интерфейса и для объектов сцены. Пользователь может перемещать объекты сцены по всему экрану, что, в частности, может приводить к перекрытиям этих объектов элементами интерфейса (элементы интерфейса в этом случае

отображаются поверх объектов сцены). Однако первый показ сцены таков, что все её объекты точно вписаны в отведённое для её отображения пространство.

Элементы интерфейса отображаются при помощи текстурированных прямоугольников в ортогональной проекции. Для отображения сцены предлагается использовать два режима: двумерный и трёхмерный, которые автоматически переключаются в зависимости от характера данных, подлежащих визуализации.

В двумерном режиме отображения сцены, согласно предложенному подходу к визуализации, используется ортогональная проекция с параллелепипедом видимости, центр которого находится в нуле, длина и ширина равны 2, а глубина вычисляется на основании глубины сцены. Хотя сцена визуализируется как двумерная, она может содержать трёхмерные объекты, глубина которых должна быть учтена, чтобы избежать отсечения ближней или дальней плоскостью параллелепипеда видимости. Матрица камеры при этом заполняется следующим образом:

$$M_{camera} = \begin{pmatrix} 2\cdot\frac{w_d}{w} & 0 & 0 & 2\cdot\frac{x}{w}-1 \\ 0 & 2\cdot\frac{h_d}{h} & 0 & 2\cdot\frac{y}{h}-1 \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (37)$$

где, w_d, h_d – ширина и высота области для отображения сцены в экранных координатах,

x, y – экранные координаты левого нижнего угла этой области,

z – половина максимальной протяжённости по оси Z использованных на сцене трёхмерных моделей.

Данная матрица производит смещение сцены в соответствующую зону на экране и масштабирование сцены к размеру этой зоны. В дальнейшем все преобразования для объектов строятся по принципу:

$$M_{transform} = M_{camera} \cdot M_{object}, \tag{38}$$

где $M_{transform}$ – итоговая матрица преобразования, действующая на объект,

M_{object} – матрица преобразования объекта в пространстве сцены.

В трёхмерном режиме отображения сцены, согласно предложенному подходу к визуализации, используется перспективная проекция, усечённая пирамида видимости для которой имеет следующие характеристики: центр определяется по формуле:

$$\left(\frac{2 \cdot x + w_d}{w} - 1, \left(\frac{2 \cdot y + h_d}{h} - 1 \right) \cdot \frac{w}{h}, \ 0 \right), \tag{39}$$

глубина определяется на основе глубины сцены, а соотношение сторон равно $\frac{w}{h}$.

Обозначения в формуле (39) аналогичны обозначениям в формуле (37).

Традиционно, при смещении объектов центр области видимости остаётся в начале системы координат. Однако в этом случае, при смещениях, достаточно больших относительно размеров экрана, в результате перспективной проекции будут наблюдаться заметные искажения объектов. Данная ситуация продемонстрирована на рис. 22 (а). Чтобы избежать такого эффекта, предлагается смещать центр области видимости. В этом случае искажений объектов не возникает, как показано на рис. 22 (б).

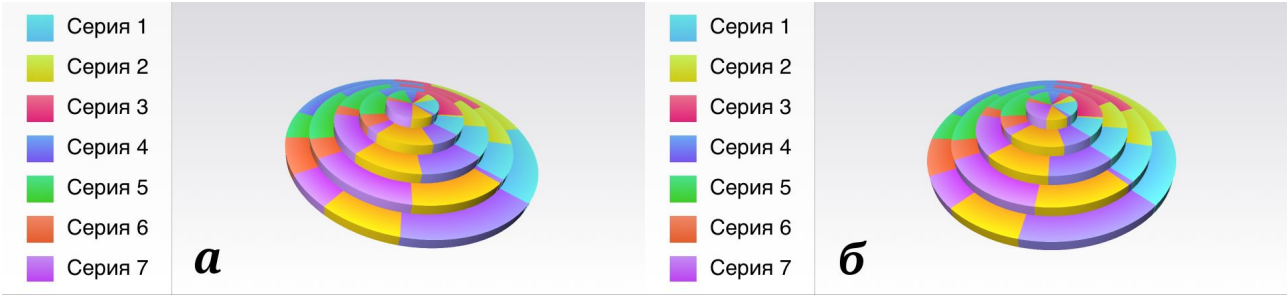


Рис. 22. Сцена со смещением объектов (а) и смещением проекции (б)

Совмещение центров области видимости и области, предназначенной для отображения сцены, предлагается осуществлять при помощи вектора смещения (39). Рассмотрим его вывод.

Пусть дана экранная система координат, имеющая начало в точке $O(0,0)$, w, h – ширина и высота экрана в пикселях.

Пусть дана произвольная прямоугольная область ширины $w_d \le w$, высоты $h_d \le h$ (пиксели) с началом в точке $D(x, y)$, $x \ge 0$, $y \ge 0$. Пусть данная область предназначена для отображения сцены (предположим, что вокруг неё располагаются элементы управления, которые при первом показе сцены должны присутствовать на экране, но не должны перекрывать объекты сцены). Эта область обозначена красным контуром на рис. 23. Экран обозначен чёрным контуром.

Пусть r, l, b, t – расстояния в пикселях от области сцены до границ экрана, C_s – центр экрана, C_d – центр области сцены.

Направление осей координат и положение точки начала координат соответствуют описанным в стандарте OpenGL(ES).

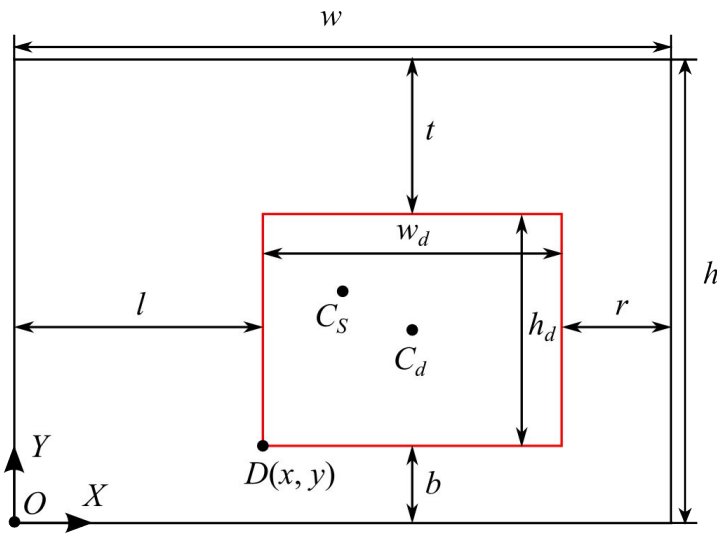


Рис. 23. Область для отображения сцены на экране

Тогда, чтобы совместить точки C_s и C_d необходимо сместить область сцены на $\frac{l-r}{2}$ по оси X и на $\frac{b-t}{2}$ по оси Y. Расстояния r, l, b, t могут быть выражены следующим образом:

$$\begin{cases} l = x \\ r = w - (x + w_d) \\ b = y \\ t = h - (y + h_d) \end{cases}. \tag{40}$$

Смещение воздействует на центр усечённой пирамиды видимости, которая, в соответствии с использованным аппаратом проекций, задаётся в нормированных координатах устройства (*англ.* normalized device coordinates, NDC) [120]. Так как NDC представляет собой куб с центром в начале координат и длиной стороны, равной 2, который отображается на экран, расстояния по осям *X* и *Y* могут быть переведены из экранных координат в NDC по формулам:

$$\begin{cases} \delta_x^{NDC} = \dfrac{2 \cdot \delta_x}{w} \\ \delta_y^{NDC} = \dfrac{2 \cdot \delta_y}{h} \cdot \dfrac{w}{h} \end{cases}. \tag{41}$$

Следовательно, смещение (40) в NDC имеет вид:

$$\begin{cases} \dfrac{l - r}{2} = \dfrac{2 \cdot x + w_d}{w} - 1 \\ \dfrac{b - t}{2} = \left(\dfrac{2 \cdot y + h_d}{h} - 1 \right) \cdot \dfrac{w}{h} \end{cases}. \tag{42}$$

С учётом нулевого смещения по оси *Z*, вектор смещения запишется как (39).

Преобразование камеры в трёхмерном режиме отображения сцены включает в себя смещение объектов в плоскостях *XOY*, *YOZ* и *XOZ*, масштаб и повороты вокруг осей *X, Y* и *Z*. При помощи преобразования камеры осуществляется пользовательская навигация по сцене. Начальные значения смещения и масштаба автоматически вычисляются таким образом, чтобы при заданных начальных углах поворота объекты сцены заняли максимальную площадь в отведённой под сцену области, но при этом не вышли за её границы. Вычисление производится на основе бинарного поиска и эвристик. Диапазон поиска смещения обусловлен особенностями NDC: координаты *x* и *y* могут изменяться только от -1 до 1. Диапазон поиска масштаба задаётся эвристикой: экспериментально было

установлено, что в большинстве случаев нужный масштаб лежит в пределах от 0,1 до 3. Эвристикой также задаётся порядок поиска значений смещения и масштаба. Экспериментально установлено, что наиболее качественный результат с минимальным количеством «холостых» итераций (итераций поиска, не дающих улучшения результата визуализации) достигается при следующем порядке поиска оптимальных значений смещения и масштаба:

1. Найти смещение по *x* и *y*, обеспечивающее центровку объектов сцены.
2. Найти масштаб, обеспечивающий максимальную близость объектов сцены к границам отведённой области.
3. Повторить шаг (1).
4. Повторить шаг (2).

Для того, чтобы обеспечить максимальную эффективность бинарного поиска, в предлагается использовать не сами объекты сцены, а параллелепипед, описанный вокруг них. На каждой итерации поиска по текущим значениям смещения, масштаба и углов поворота строится матрица камеры, перемножается с матрицей проекции, а результат последовательно умножается на координаты всех 8 вершин ограничивающего параллелепипеда. Так определяются координаты этих вершин в NDC. Аппликаты полученных координат отбрасываются, а из нормированных абсцисс и ординат выбираются минимальные и максимальные.

Критерием для бинарного поиска оптимального смещения является равенство расстояний по осям *X* и *Y* от преобразованного ограничивающего параллелепипеда до границ заданной области:

$$\begin{cases} \left| \frac{x_{min} + x_{max} - l_{NDC} + r_{NDC}}{2} \right| < \epsilon \\ \left| \frac{y_{min} + y_{max} - b_{NDC} + t_{NDC}}{2} \right| < \epsilon \end{cases}, \tag{43}$$

где x_{min}, y_{min} и x_{max}, y_{max} – соответственно минимальные и максимальные координаты преобразованных вершин ограничивающего параллелепипеда,

$l_{NDC}, r_{NDC}, b_{NDC}, t_{NDC}$ – расстояния от границ области до границ экрана, приведённые к NDC,

ϵ – эвристический параметр, отвечающий за точность поиска.

Критерием для бинарного поиска оптимального масштаба является равенство заданной для отображения сцены области прямоугольнику, описанному вокруг проекции преобразованных вершин ограничивающего параллелепипеда на плоскость *XOY*:

$$\left|\max\left\{x_{min}-l_{NDC},\ \ x_{max}+r_{NDC},\ \ y_{min}-b_{NDC},\ \ y_{max}+t_{NDC}\right\}\right|<\epsilon. \qquad (44)$$

Для обоих критериев экспериментально подобрано значение точности поиска $\epsilon = 0.01$.

Максимальное количество итераций описанного алгоритма составляет

$$n_i = 2\cdot\log_2\left(\frac{offset_{max}-offset_{min}}{\epsilon}\right)+2\cdot\log_2\left(\frac{zoom_{max}-zoom_{min}}{\epsilon}\right), \qquad (45)$$

где $offset_{min}, offset_{max}, zoom_{min}, zoom_{max}$ – диапазоны поиска смещения и масштаба соответственно.

Описанный алгоритм имеет логарифмическую сложность.

С учётом указанных ранее числовых значений, максимальное количество итераций поиска равно 32. Описанный алгоритм не снижает производительность рендеринга, так как, во-первых, выполняется только один раз для каждой сцены (чтобы определить преобразование камеры для первого показа), а во-вторых в среднем время его исполнения на устройстве iPad 3 составляет всего 1 мс. При этом он позволяет определить оптимальное положение и размер для произвольной группы трёхмерных объектов, для которых заданы определённые углы поворота и определённая проекция.

Благодаря разработанному алгоритму удалось автоматизировать выбор наиболее выгодного начального расположения для произвольной сцены по заданной области, если площадь области превышает 1 пиксель. Ограничение на площадь вытекает из соображений здравого смысла.

3.3. Тестирование производительности

Сглаживание границ объектов обеспечивает ощутимое увеличение визуального качества изображения. Было произведено сравнение результата визуализации одной и той же сцены (результата моделирования поведения жидкости в ограниченном объёме, решатель: программа OpenFOAM [121]) в системе SciVi и в одной из самых популярных на сегодняшний день систем научной визуализации ParaView. Итоговые изображения приведены на рис. 24.

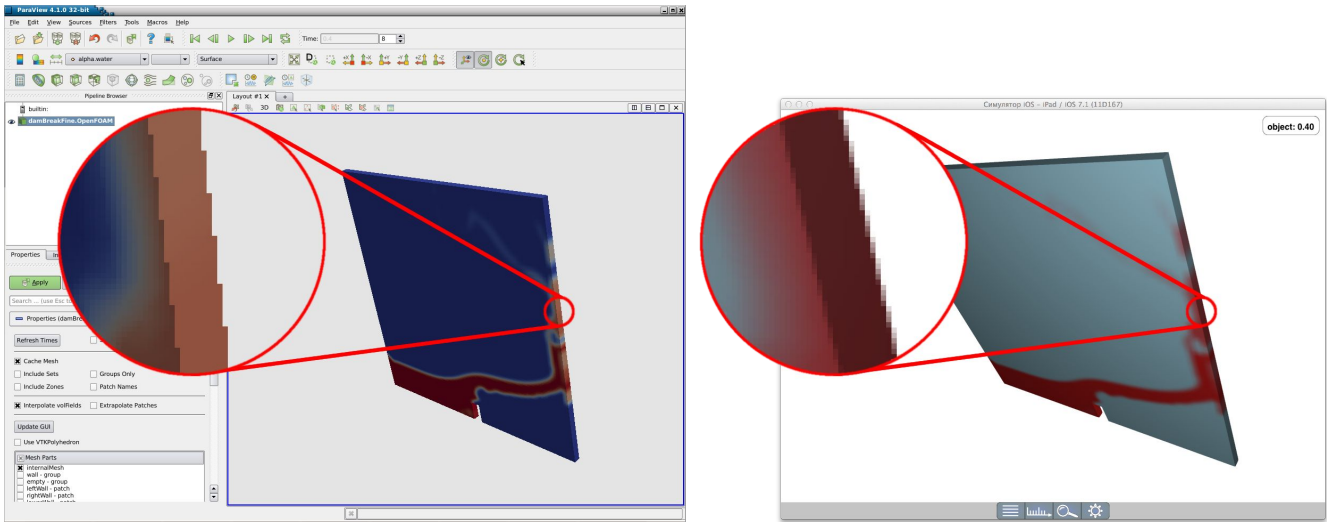


Рис. 24. Визуализация результатов моделирования поведения жидкости в ограниченном объёме в системе ParaView (слева) и в системе SciVi (справа)

Система ParaView не поддерживает сглаживание границ объектов [35], в силу чего визуальное качество изображений, построенных в ней, оказывается ниже, чем в системе SciVi.

С целью проверки целесообразности использования предложенного метода сглаживания границ объектов было проведено тестирование производительности на сценах различной сложности. Тестирование производилось на устройстве iPad 3. Измерялось время задержки отклика системы на команды пользователя при визуализации сцен с различным числом вершин. Результаты тестирования представлены на графике, изображённом на рис. 25. Зелёная зона (от 0 до 50 мс) на этом графике – это зона максимального комфорта, когда пользователь не ощущает задержек отклика. Жёлтая зона (от 50 до 125 мс) – это зона достаточного комфорта, когда задержка отклика, хотя и заметна пользователю, но не нарушает

восприятие динамики сцены. Красная зона (от 125 мс) – это зона отсутствия комфорта, когда задержка отклика нарушает восприятие динамики сцены. Зоны выделены условно на основании исследования [30].

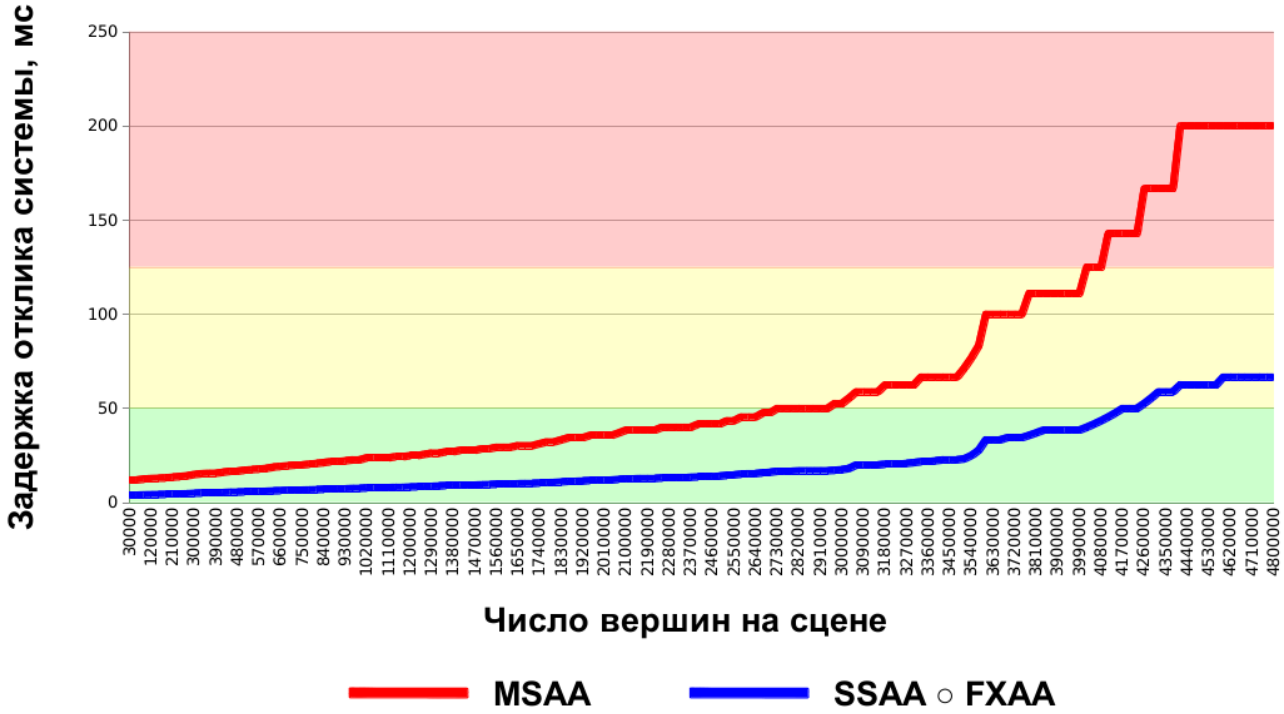


Рис. 25. График задержки отклика системы при использовании сглаживания на основе системного MSAA и предложенной суперпозиции SSAA и FXAA

По результатам тестирования можно сделать вывод о том, что предложенный алгоритм сглаживания обеспечивает в среднем в три раза большую производительность работы системы визуализации и даже на сложных сценах (содержащих более 4 млн. вершин) сохраняет комфортность восприятия динамики сцены пользователем. Следовательно, с позиций производительности использование предложенного алгоритма сглаживания более целесообразно, чем использование системного.

С учётом выполненных экспериментов, было произведено сравнение производительности системы SciVi с производительностью систем ParaView и KiwiViewer (наиболее полнофункциональной из доступных систем научной визуализации для мобильных устройств). Производительность измерялась как величина, обратная времени задержки отклика (в мс). Было установлено, что SciVi

не уступает по производительности ParaView (разница во времени отклика в среднем нулевая) и обладает в три раза большей производительностью, чем KiwiViewer (при включенном сглаживании границ; при выключенном сглаживании производительность системы KiwiViewer равна производительности системы SciVi).

3.4. Выводы по главе

На основе проведённых исследований выделены недостатки встроенного платформенно-зависимого антиалиасинга. Сделан вывод о том, что для поддержки высокой производительности рендеринга, мультиплатформенности и, вместе с тем, высокого визуального качества результирующего изображения следует отказаться от встроенных средств антиалиасинга в пользу сторонних алгоритмов.

Проанализирован ряд алгоритмов антиалиасинга и выбраны два наиболее подходящих для использования в контексте реализации мультиплатформенных систем научной визуализации – SSAA и FXAA. Предложен новый метод адаптивного сглаживания границ объектов на изображении, основанный на суперпозиции модификаций этих алгоритмов.

Адаптивность сглаживания предложенным методом к аппаратному обеспечению состоит в том, что коэффициент суперсэмплинга вычисляется автоматически на основе максимального поддерживаемого размера текстуры. За счёт этого метод применим на большом множестве различных мобильных устройств и настольных компьютеров.

Адаптивность сглаживания предложенным методом к процессу отображения состоит в том, что сглаживание автоматически отключается на периоды динамического изменения сцены. За счёт этого сохраняется высокая производительность в критичные к скорости визуализации моменты отображения сцены и тем самым обеспечивается плавность воспроизводимой анимации.

Адаптируемость сглаживания предложенным методом к конкретной отображаемой сцене состоит в послойной группировке объектов сцен и

возможностью определять, какие из слоёв нуждаются в сглаживании, а какие нет. Это позволяет комбинировать объекты со сглаженными и чёткими границами на одном изображении, избегая тем самым эффектов нежелательного размытия.

Модификации алгоритма FXAA состоят в оптимизации его работы на графическом процессоре и введении в него дополнительных шагов обработки изображения, служащих повышению визуального качества результата. Оптимизация достигнуты благодаря тому, что:

1. Часть вычислений перенесена из фрагментного шейдера в вершинный.
2. Всюду, где возможно, использованы векторные инструкции.
3. Все формулы приведены к виду, совместимому с MAD-инструкциями GPU.
4. Все динамические ветвления заменены на пороговые функции.
5. Для реализации линейной интерполяции используются только аппаратно-поддерживаемые функции.

В результате удалось увеличить скорость работы алгоритма в среднем на 20%. Визуальное качество результата работы FXAA увеличено путём смешивания исходного значения цвета каждой точки изображения с цветом, вычисленным в ходе алгоритма. Такое смешивание снижает шум, появляющийся на границах объектов.

Алгоритм SSAA модифицирован путём введения в него переменного коэффициента увеличения изображения, который на каждой конкретной платформе автоматически определяется в зависимости от максимального поддерживаемого размера текстуры и разрешения экрана.

Произведена теоретическая оценка сложности процесса визуализации без использования сглаживания и с использованием каждого из предложенных алгоритмов. Произведено тестирование производительности работы системы научной визуализации в различных условиях. Результаты теоретических оценок и практических измерений представлены в табл. 6.

Таблица 6. Оценка характеристик процесса визуализации в системе SciVi

Способ визуализации	**Без сглаживания**	**SSAA**	**FXAA**	**SSAA○FXAA**
Асимптотическая сложность	$O(n+w\cdot h)$	$O(n+s^2\cdot w\cdot h)$	$O(n+w\cdot h)$	$O(n+s^2\cdot w\cdot h)$
Среднее количество итераций шейдеров	$n+w\cdot h$	$(n+6)+$ $(s^2+1)\cdot w\cdot h$	$(n+6)+$ $2\cdot w\cdot h$	$(n+6)+$ $(s^2+1)\cdot w\cdot h$
Скорость, FPS	15	4,5	10	4,5

Предложенный метод антиалиасинга решает проблемы, присущие встроенному системному методу, сохраняя мультиплатформенность кода, высокое визуальное качество итогового изображения и высокую производительность графического приложения. Метод был интегрирован в систему научной визуализации SciVi и протестирован на различных ЭВМ под управлением iOS, Android, Windows, GNU / Linux и OS X. В результате визуальное качество итоговых изображений, генерируемых системой SciVi, оказывается выше чем у некоторых популярных на сегодняшний день систем научной визуализации, например системы ParaView (в которой сглаживание границ объектов не поддерживается). Производительность визуализации в системе SciVi на настольных компьютерах не ниже, чем в аналогичных системах, а на мобильных устройствах при включенном сглаживании границ объектов оказывается в 3 раза выше.

Предложен алгоритм для выбора выгодного с точки зрения качества визуализации первоначального положения сцены. Он позволяет определить оптимальное положение и размер для произвольной группы трёхмерных объектов, для которых известны соответствующие углы поворота и проекции. При помощи данного алгоритма был автоматизирован выбор т. н. «вида по умолчанию» – положения сцены, в котором она отображается до начала каких-либо навигационных действия со стороны пользователя.

Глава 4. Применение разработанной системы для визуализации научных данных различной природы

Система научной визуализации SciVi, разработанная на базе предложенного в диссертационном исследовании подхода, была применена для построения наглядных изображений при решении задач из различных предметных областей:

1. Моделирование вращения магнитных моментов наночастиц в магнитном поле (физика).
2. Мониторинг изменения цен на валютной бирже (экономика).
3. Множественное выравнивание последовательностей ДНК и построение филогенетических деревьев (генетика).
4. Моделирование поведения жидкости в ограниченном объёме (гидродинамика).
5. Измерение колебания кожной температуры человека (медицина).
6. Измерение скорости передачи данных по компьютерной сети (информационные технологии).

Лёгкость интеграции со сторонними решателями, которые являлись источниками данных в этих задачах, а также успешные результаты визуализации подтверждают жизнеспособность предложенного подхода.

Результат решения задачи (6) получен при внедрении системы SciVi в пермской IT-компании ООО «Ньюлана». Копия соответствующего акта о внедрении представлена в приложении 1.

4.1. Моделирование вращения магнитных моментов наночастиц в магнитном поле

Решателем в данной задаче выступает параллельная программа ManetoDynamics-F [122], разработанная Т.С. Белозёровой в рамках гранта РФФИ-Пермский Край 11-07-96007 р_урал_а «Современные вычислительные и информационные технологии в исследованиях магнитодинамики и когерентных процессов в наномагнитных структурах» (2011-2013, руководители А.Г. Деменев,

Е.К. Хеннер). Языком реализации является Fortran. Параллелизм достигается путём применения технологии OpenMP.

Программа MagnetoDynamics-F тестировалась на суперкомьютере Тесла ПГУ, расположенном в пермского Государственного национального исследовательского университета. Входные данные программы MagnetoDynamics-F описывают характеристики магнитного поля, структуру сетки, в узлах которой расположены магнитные наночастицы, и количество квантов модельного времени. Выходные данные представляют собой массив трёхкомпонентных векторов, задающих ориентацию частиц в трёхмерном пространстве в конкретный момент времени. Для отображения результатов моделирования SciVi была настроена на структуру сцены «трёхмерная сетка», в узлах которой расположены сферы, символизирующие наночастицы. У каждой сферы имеется стрелка, указывающая направление магнитного момента (во введённой в модели системе координат, где ось *OX* направлена вправо, ось *OY* – вверх, а ось *OZ* – на наблюдателя). Клиенту SciVi передаётся описание сцены, содержащее данные об ориентации всех моделируемых частиц в каждый из моментов модельного времени. Для воспроизведения плавной анимации промежуточные ориентации вычисляются в динамике методом сферической интерполяции кватернионов. Один кадр анимационной последовательности, полученной в результате визуализации, представлен на рис. 26.

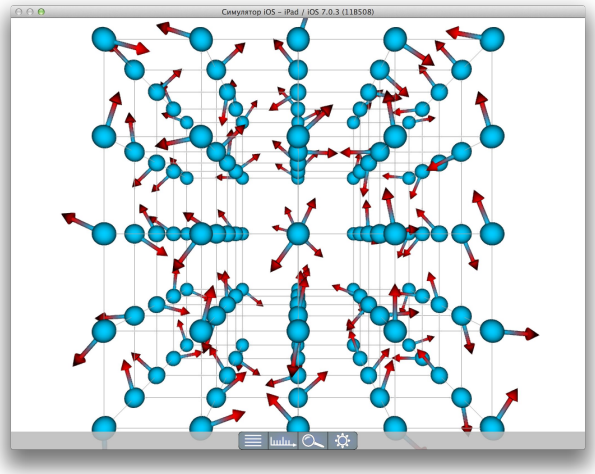


Рис. 26. Визуализация ориентации магнитных моментов наночастиц в системе SciVi

4.2. Мониторинг изменения цен на валютной бирже

Решателем в данной задаче выступает приложение BTCeXplorer, написанное автором диссертационного исследования на языке Java. Это приложение с периодичностью 1 раз в 5 минут запрашивает по сети Интернет данные о состоянии валютной биржи BTC-e, используя API онлайн-сервиса этой биржи, и сохраняет их в свою базу данных. Затем, по запросу пользователя, оно выдаёт данные за определённый промежуток времени. Входными данными выступают дата и время начала и конца интересующего пользователя периода. Выходными данными является массив значений максимальной и минимальной цены, а также цены открытия и закрытия на каждый из пятиминутных периодов внутри заданного пользователем временного промежутка. Для отображения изменений цен на бирже SciVi была настроена на структуру сцены «двумерный график» типа «японские свечи». Данный тип является классическим для визуализации состояния биржи и позволяет наглядно показать минимумы и максимумы цен, а также цены открытия и закрытия в заданные периоды времени. По оси абсцисс откладывается время, по оси ординат – значение цены в долларах. В описанной задаче не используются средства анимации, однако присутствует интерактивность: пользователь может изменять масштаб по осям и прокручивать график, чтобы изучить отдельные его части. Результат визуализации представлен на рис. 27.

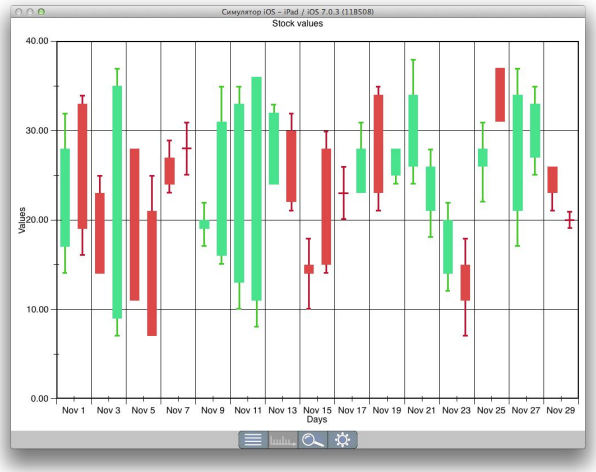


Рис. 27. Визуализация изменения цен на валютной бирже в системе SciVi

4.3. Множественное выравнивание последовательностей ДНК и построение филогенетических деревьев

Решателем в данной задаче выступает свободно распространяемая программа Clustal [123], написанная на языке C++, предоставляющая возможность осуществлять множественное выравнивание последовательностей ДНК.

Входными данными для программы Clustal выступает массив секвенированных цепочек ДНК, записанных в виде строк в алфавите $\{A, T, G, C, N\}$, где каждый символ соответствует азотистому основанию: A – аденин, T – тимин, G – гуанин, C – цитозин, N – нераспознанное основание. Множественное выравнивание заключается в добавлении в эти цепочки пробелов таким образом, чтобы получившиеся новые цепочки минимально отличались друг от друга. На основании их сходства эксперт-генетик может сделать вывод о степени родства организмов, которым принадлежат эти ДНК.

Выходными данными программы Clustal является массив выровненных цепочек ДНК, алфавит в которых расширен символом пробела, а также структура филогенетического дерева, выражающего степень родства организмов, являвшихся донорами цепочек.

Генетический материал в данной задаче был предоставлен Институтом экологии и генетики микроорганизмов Уральского отделения Российской Академии наук (г. Пермь). Материал получен в результате секвенации ДНК изучаемых в институте штаммов бактерий.

Для визуализации цепочек ДНК SciVi была настроена на структуру сцены «двумерный график» типа «последовательность». Каждое азотистое основание отображается в виде прямоугольника заданного цвета. Пробелы отображаются серыми прямоугольниками. По оси абсцисс откладываются порядковые номера азотистых оснований, по оси ординат – цепочки. Для визуализации филогенетического дерева используется структура сцены «двумерный график» типа «дерево». Результат визуализации представлен на рис. 28.

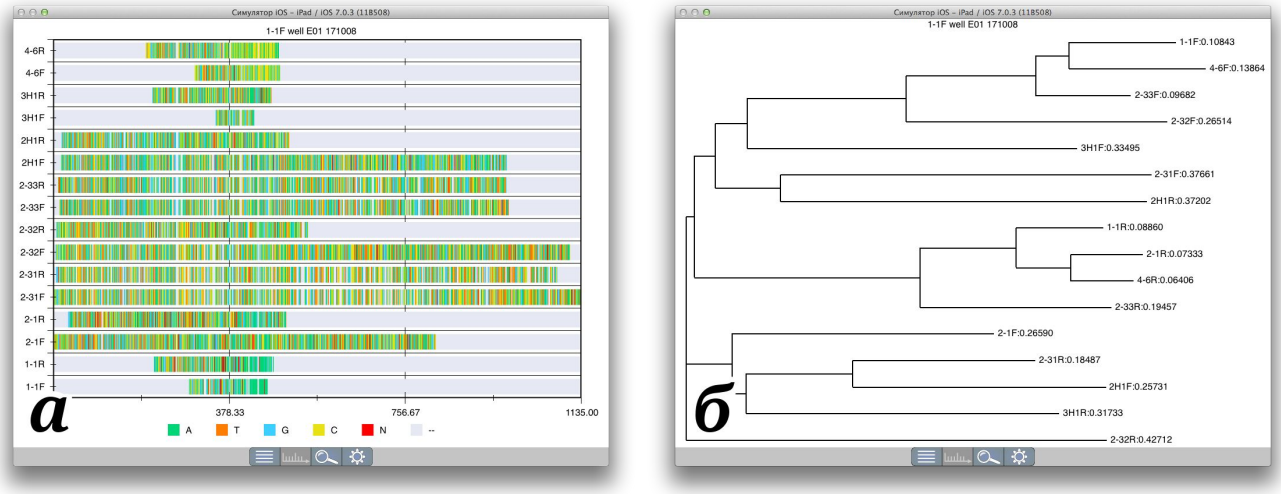


Рис. 28. Визуализация последовательностей ДНК (а) и филогенетического дерева (б) в системе SciVi

4.4. Моделирование поведения жидкости в ограниченном объёме

Решателем в данной задаче выступает свободно распространяемая система моделирования OpenFOAM [121], написанная на языке C++. Эта программа предоставляет большое количество алгоритмов для решения задач гидро- и газодинамики. Входными данными для неё являются описание характеристик жидкости (газа), области моделирования (как правило – некоторой полости), модельного времени и настроек алгоритма моделирования. Выходными данными являются описания трёхмерных моделей областей моделирования, в узлах полигональной сетки которых сохранены значения выбранных пользователем характеристик жидкости (газа) в каждый момент модельного времени.

Для визуализации в системе SciVi был взят один из обучающих примеров OpenFOAM, в котором моделируется процесс разлива столба воды в полости сложной формы. Для отображения сгенерированных OpenFOAM трёхмерных моделей в SciVi была выбрана структура сцены «трёхмерная модель». Значения плотности воды в точках внутреннего пространства моделируемой полости конвертируются сервером SciVi в данные цветовой маркировки узлов модели. Цветовая шкала задаётся при настройке SciVi на решатель. Результат визуализации представлен на рис. 29. В данном примере малой плотности

соответствует голубой цвет, высокой плотности – красный. Таким образом, по форме красного пятна на поверхности трёхмерной модели можно определить положение жидкости в изучаемой полости.

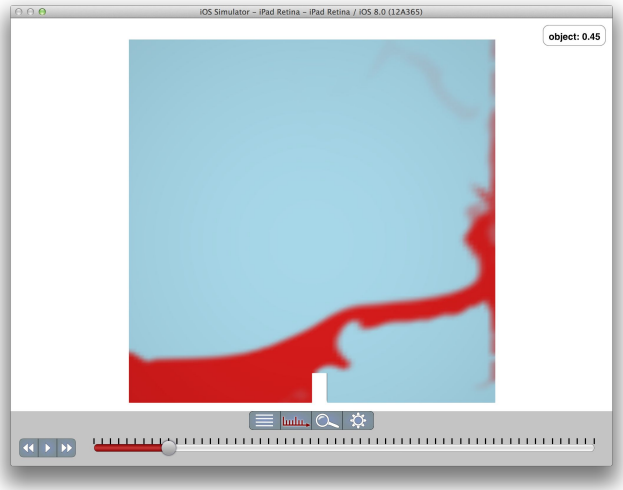


Рис. 29. Визуализация жидкости в ограниченном объёме в системе SciVi

4.5. Измерение колебания кожной температуры человека

Решателем в данной задаче выступает программно-аппаратный комплекс Микротест [124, 125, 126], осуществляющий измерение и вейвлет-анализ колебаний кожной температуры пальца руки человека с использованием медицинского диагностического прибора. Основная цель такого анализа состоит в ранней диагностике различных заболеваний (таких как панкреатит или диабет) путём изучения микроциркуляции крови пациента.

Входными данными для программной части решателя является массив температур, измеренных с определённой периодичностью. Измерения производятся аппаратной частью решателя. Выходными данными является результат вейвлет-анализа полученных данных, представляющий собой массив коэффициентов вейвлет-плоскости.

Визуализация массива температур и вейвлет-плоскости позволяет экспертам в области медицины интерпретировать результаты измерений, способствует выявлению закономерностей и в конечном счёте улучшает диагностику пациентов.

Интеграция SciVi с решателем производилась на уровне данных (без прямого взаимодействия с приложением, генерирующим их). Для отображения массива температур в SciVi была выбрана структура сцены «двумерный график» типа «линия». По оси абсцисс откладывается время, в которое произведено измерение, а по оси ординат – температура в градусах Цельсия. Для отображения вейвлет-плоскости выбрана структура сцены «двумерный график» типа «тепловая карта». По оси абсцисс откладывается время, по оси ординат – масштаб, а на цвет плоскости влияют значения, полученные в результате вейвлет-анализа (преобразование в цвет происходит по шкале, заданной в процессе настройки SciVi на решатель). Результат визуализации представлен на рис. 30.

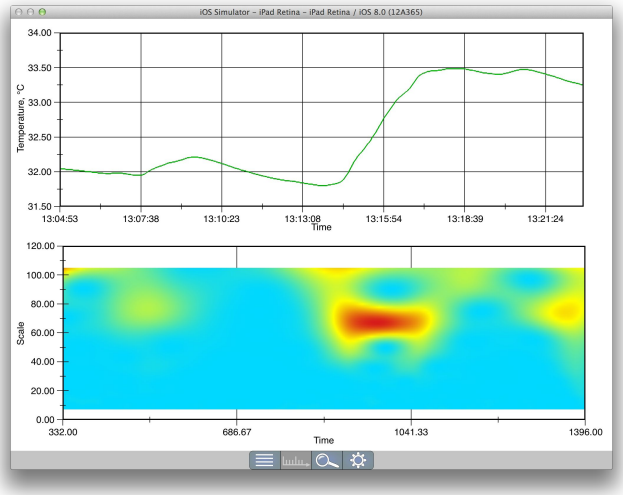


Рис. 30. Визуализация массива температур и вейвлет-плоскости в системе SciVi

4.6. Измерение скорости передачи данных по сети

Решателем в этой задаче выступает программа Remotix [127], разработанная в пермской IT-компании ООО «Ньюлана». Remotix – это программа для удалённого управления компьютером. В функции, доступные пользователю, не входит измерение скорости передачи данных по сети, однако её анализ был необходим для отладки механизмов сетевого взаимодействия. Была создана тестовая версия программы Remotix, которая выводила необходимую информацию в журнал, сохраняемый в файле. Затем файл журнала обрабатывался сервером SciVi, как готовые выходные данные решателя. Таким образом, в этой

задаче, так же как и в задаче измерения кожной температуры, SciVi взаимодействовала не напрямую с решателем, а только с заранее созданными им файлами. В журнале, помимо прочей, не подлежащей визуализации информации, находился массив чисел и символов, описывающий скорость передачи данных на различных участках сети, которые кодировались двумя координатами (в нотации «буква-цифра»). Благодаря произведённым настройкам, сервер SciVi автоматически вычленял этот массив из общей структуры журнала и заполнял его элементами соответствующий шаблон сцены.

Для визуализации SciVi была настроена на структуру сцены «трёхмерный график» типа «гистограмма». На осях абсцисс и аппликат располагались координаты участков сети, на оси ординат – скорость в кб/с. Результат визуализации представлен на рис. 31.

Использование визуализации средствами SciVi позволило выявить проблемные места в сети и, как следствие, путём целенаправленной оптимизации повысить производительность тестируемого сетевого протокола и эффективности тестируемой топологии сети.

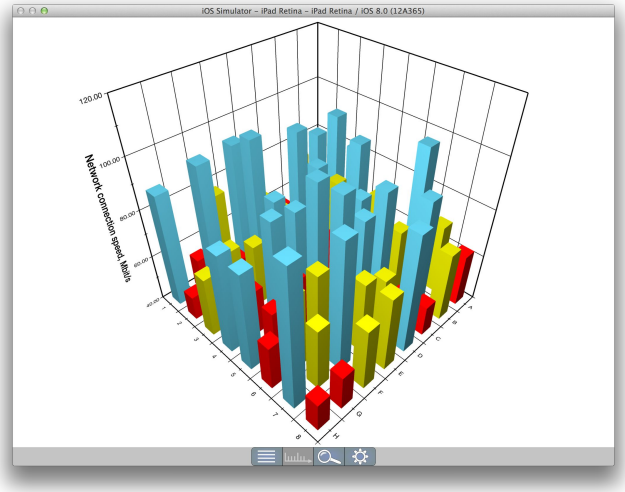


Рис. 31. Визуализация значений скорости передачи данных на различных участках сети в системе SciVi

4.7. Выводы по главе

Система визуализации SciVi была успешно применена для визуализации данных в реальных научных задачах из различных предметных областей. В процессе решения задач визуализации SciVi была успешно адаптирована к разнородным сторонним решателям. Данный факт доказывает универсальность и гибкость SciVi, а следовательно, адекватность и практическую значимость разработанного в рамках диссертационного исследования подхода к созданию систем научной визуализации.

Можно также сделать вывод о том, что SciVi применима в междисциплинарных или обучающих проектах, позволяя учёным из различных областей знания наглядно представлять результаты своих исследований и тем самым проще находить общий язык с коллегами, интерпретировать и проверять результаты экспериментов и выявлять скрытые закономерности в данных.

Заключение

Научная визуализация в настоящее время является актуальным и активно развивающимся направлением компьютерной графики. Её востребованность обусловлена потребностью учёных-исследователей в автоматизированном создании высококачественных изображений и интерактивных анимационных роликов средствами ЭВМ, которые служили бы целям наглядной демонстрации и анализа хода и результатов научных экспериментов.

Несмотря на достаточно глубокую проработанность вопросов научной визуализации, в этой области существует ряд нерешённых проблем. В ходе анализа средств научной визуализации, доступных конечным пользователям (учёным-исследователям), были выявлены следующие основные недостатки:

1. Отсутствие средств высокоуровневой интеграции со сторонними решателями.
2. Малое число мультиплатформенных решений.
3. Относительно слабая проработанность вопросов распределённой научной визуализации.
4. Падение качества результирующего изображения из-за ступенчатости границ объектов при визуализации трёхмерных моделей и недостаточная эффективность алгоритмов сглаживания границ.

Разработка методов и средств для комплексного решения указанных проблем представляет собой актуальную научную задачу. Эта задача была решена в результате проведения данного диссертационного исследования. Целью исследования ставилась разработка методов и средств для создания адаптивных мультиплатформенных клиент-серверных систем визуализации научных экспериментов. Были достигнуты следующие основные результаты:

1. Предложены новые метод и средства организации мультиплатформенных клиент-серверных систем научной визуализации с адаптивным распределением процесса рендеринга между клиентом и сервером, а также с поддержкой интеграции на принципах адаптации со сторонними

решателями. Впервые для адаптации систем научной визуализации к сторонним решателям использованы методы онтологического инжиниринга.

2. Созданы новые метод и средства построения мультиплатформенного графического интерфейса пользователя, отличающиеся от известных тем, что решают проблему двойного дизайна без снижения эффективности визуализации.
3. Предложены метод и средства адаптивной распределённой визуализации, позволяющие автоматически учитывать инфраструктуру вычислительной сети и сетевого ПО на основе эвристических правил. В отличие от существующих решений, предложенные метод и средства обеспечивают высокую интерактивность графических сцен в условиях использования маломощных мобильных устройств, подключенных к серверу по низкоскоростному беспроводному сетевому соединению.
4. Разработан новый метод сглаживания границ объектов на изображении, отличающийся от известных аналогов тем, что, путём адаптивного учёта аппаратных особенностей платформы, обеспечивает в 3 раза более быстрый отклик системы на команды пользователя и, одновременно с этим, высокое визуальное качество итогового изображения, а также допускает мультиплаторфменную реализацию.
5. На основе предложенных методов и средств разработана система научной визуализации SciVi. Система успешно протестирована на различных типах настольных компьютеров и мобильных устройств под управлением ОС Windows, GNU / Linux, OS X, iOS и Android, а также использована для решения задач научной визуализации в различных предметных областях и внедрена в пермской IT-компании ООО «Ньюлна».

В ходе диссертационного исследования решены все поставленные задачи и, тем самым, достигнута поставленная цель. Предложено комплексное решение отмеченных проблем в области научной визуализации, и жизнеспособность этого решения проверена на практике путём разработки системы визуализации научных

экспериментов, которая успешно протестирована на практике в ходе решения реальных прикладных задач.

В перспективе система SciVi может быть усовершенствована путём:

1. Расширения онтологии конструкций ввода-вывода языков программирования и онтологии визуальных объектов с целью увеличения числа поддерживаемых языков программирования решателей и добавления в репозиторий системы новых типов визуальных объектов для отображения разнородных данных.
2. Поддержки дополнительных способов рендеринга, например, рендеринга многомерных объектов.

Перспективным является также исследование вопросов применимости разработанных методов и средств создания систем научной визуализации в контексте решения задач обработки Больших данных (*англ.* Big Data).

Список сокращений и условных обозначений

Φ	Оператор визуализации.
$\overline{U}$	Онтология визуальных объектов.
Δ	Оператор сглаживания.
Γ	Оператор интерактивного взаимодействия.
$\overline{M}$	Множество поддерживаемых элементов интерфейса.
$\overline{E}$	Множество поддерживаемых действий пользователя.
Θ	Оператор синтаксического анализа, управляемый онтологией $\overline{L}$.
$\overline{L}$	Онтология синтаксических конструкций ввода-вывода языков программирования.
Ω	Оператор генерации конвертера.
Π	Оператор выбора элементов управления.
M	Множество элементов управления для организации навигации по сцене, $M \subset \overline{M}$.
3G	3 Generation.
API	Application Programming Interface.
CSAA	Coverage-Sampling Anti-Aliasing.
DMSAA	Deferred MultiSampling Anti-Aliasing.
EDGE	Enhanced Data rates for GSM Evolution.
FPS	Frames Per Second.
FXAA	Fast approXimate Anti-Aliasing.
GPGPU	General-Purpose Graphics Processing Units.
GPRS	General Packet Radio Service.
GSM	Global System for Mobile Communications.
HTML	Hyper Text Markup Language.
HTTP	Hyper Text Transfer Protocol.
JNI	Java Native Interface.
MAD	Multiply-Add.
MLAA	MorphoLogical Anti-Aliasing.

MSAA	MultiSampling Anti-Aliasing.
MVC	Model-View-Controller.
NDC	Normalized Device Coordinates.
OWL	Web Ontology Language.
RGB	Red Green Blue.
SMAA	Subpixel Morphological Anti-Aliasing.
SSAA	SuperSampling Anti-Aliasing.
TXAA	Temporal Anti-Aliasing.
VNC	Virtual Network Computing.
WiFi	Wireless Fidelity.
XML	eXtensible Markup Language.
БНФ	Формы Бэкуса-Наура.
ДНК	Дезоксирибонуклеиновая кислота.

Список терминов

3 generation (3G): Технологии мобильной связи 3-го поколения; набор услуг, который объединяет как высокоскоростной мобильный доступ с услугами сети Интернет, так и технологию радиосвязи, которая создаёт канал передачи данных.

application programming interface (API): Набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах. Используется для написания приложений.

алиасинг (Aliasing): Появление ступенек на границах объектов в результате растеризации.

антиалиасинг (Anti-Alising): Процесс устранения алиасинга.

cave: Программно-аппаратный комплекс, предназначенный для демонстрации виртуального мира, представляющий собой комнату, на стены которой проецируются генерируемые компьютером изображения с целью достижения у наблюдателя «эффекта присутствия».

coverage-sampling anti-aliasing (CSAA): Алгоритм сглаживания границ объектов на изображении, основанный на расширенной выборке данных из буферов глубины и трафарета вблизи границ полигонов.

deferred multisampling anti-aliasing (DMSAA): Алгоритм отложенный MSAA, основанный на многопроходной фильтрации.

Direct3D: Спецификация и библиотека низкоуровневого графического API для ОС Windows от компании Microsoft.

Dublin Core: Словарь понятий, предназначенный для унификации метаданных описания широкого диапазона различных ресурсов.

enhanced data rates for GSM evolution (EDGE): Цифровая технология беспроводной передачи данных для мобильной связи, развитие технологии GPRS.

extensible markup language (XML): Расширяемый язык разметки.

fast approximate anti-aliasing (FXAA): Алгоритм быстрого аппроксимированного сглаживания границ объектов на изображении, идея которого состоит в поиске границ объектов и размытии их в перпендикулярном к ним направлении.

frames per second (FPS): Показатель скорости обновления изображения на экране, выраженный в числе кадров, сменяющих друг друга за секунду.

general packet radio service (GPRS): Надстройка над технологией мобильной связи GSM, осуществляющая пакетную передачу данных.

general-purpose graphics processing units (GPGPU): Техника использования графического процессора для выполнения расчётов общего назначения (которые обычно производит центральный процессор).

global system for mobile communications (GSM): Глобальный стандарт цифровой мобильной сотовой связи, с разделением каналов по времени и частоте.

HTTP-запрос: Запрос согласно протоколу HTTP.

hyper text markup language (HTML): Стандартный язык разметки документов в сети Интернет.

hyper text transfer protocol (HTTP): Сетевой протокол передачи данных прикладного уровня, изначально предназначенный для передачи Интернет-страниц в формате HTML, в настоящее же время обобщённый для передачи произвольных данных.

iPad: Планшетный компьютер, разрабатываемый и поддерживаемый компанией Apple.

java native interface (JNI): Стандартный механизм для запуска кода под управлением виртуальной машины Java, который написан на языках C / C++ или Ассемблера и скомпонован в виде динамических библиотек. Даёт возможность вызова функции C / C++ из программы на Java, и наоборот.

model-view-controller (MVC): Шаблон проектирования, предписывающий разделение системы на модули взаимодействия с данными (модель, model), отображения данных (представление, view) и управления (контроллер, controller).

morphological anti-aliasing (MLAA): Алгоритм морфологического сглаживания границ объектов на изображении, основанный на поиске шаблонов границ и размытия их по правилам, соответствующим щаблонам.

multisampling anti-aliasing (MSAA): Алгоритм сглаживания границ объектов на изображении, основанный на расширенной выборке данных из буферов глубины и трафарета.

normalized device coordinates (NDC): Нормированные координаты устройства, куб с длиной стороны, равной 2, центр которого расположен в начале декартовой системы координат. В растеризации участвуют лишь те части полигонов, которые в результате всех преобразований попали в данный куб.

OpenGL: Спецификация, определяющая независимый от языка программирования и платформы программный интерфейс для написания приложений, использующих двумерную и трёхмерную компьютерную графику.

OpenGLES: Облегчённая версия OpenGL, ориентированная на мобильные устройства.

SIMD-архитектура (Single Instruction, Multiple Data): Архитектура ЭВМ, построенная на принципе параллелизма по данным.

subpixel morphological anti-aliasing (SMAA): Алгоритм SMAA по уточнённой выборке, спользующий более качественные способы поиска границ

supersampling anti-aliasing (SSAA): Алгоритм сглаживания границ объектов на изображении на основе увеличенного изображения, разрешение которого превышает размер экрана и при показе уменьшается с применением фильтрации.

temporal anti-aliasing (TXAA): Алгоритм сглаживания границ объектов на изображении по времени, использующий несколько кадров анимационной последовательности для осуществления расширенной выборки.

virtual network computing (VNC): Система удалённого доступа к рабочему столу компьютера. Управление осуществляется путём передачи нажатий клавиш

на клавиатуре и движений мыши с одного компьютера на другой и ретрансляции содержимого экрана через компьютерную сеть.

web ontology language (OWL): Язык описания онтологий, позволяющий описывать классы и отношения между ними, присущие веб-документам и приложениям.

Web-интерфейс: Совокупность средств, при помощи которых пользователь взаимодействует с Web-сайтом или Web-приложением через браузер.

Web-приложение: Приложение, ядро которого находится на сервере, а взаимодействие с клиентами осуществляется через Web-интерфейс.

Web-сайт: Система электронных документов (файлов данных и кода) частного лица или организации в компьютерной сети под общим адресом.

Web-хостинг: Услуга по предоставлению вычислительных мощностей для размещения информации на сервере, постоянно находящемся в сети Интернет (или локальной сети организации).

WebGL: Облегчённая версия OpenGL, ориентированная на Web-приложения.

wireless fidelity (WiFi): Торговая марка Wi-Fi Alliance для беспроводных сетей на базе стандарта IEEE 802.11.

XML-описание: Описание чего-либо, выполненное средствами языка разметки XML.

XML-тэг: Синтаксическая конструкция языка разметки XML, предназначенная для описания некоторой сущности. Может иметь неограниченное количество параметров.

азотистое основание: Органическое соединение, входящее в состав ДНК, например аденин, гуанин, цитозин, тимин.

альфа-канал: Значение, связанное с каждой точкой изображения и определяющее коэффициент для альфа-смешивания цвета этой точки с другим цветом. Чаще всего значение альфа-канала точки изображения характеризует прозрачность этой точки.

альфа-смешивание: Смешивание цветов соответствующих точек двух изображений с учётом значений альфа-каналов этих точек. Существует большое количество различных законов смешивания. Чаще всего альфа-смешивание используется для организации полупрозрачности изображений.

анимация: Последовательный показ нескольких кадров (анимационной последовательности), содержимое которых постепенно изменяется. Тем самым демонстрируется изменение графической сцены с течением времени.

Аутентификация – процедура проверки подлинности (например проверка подлинности пользователя путём сравнения введённого им пароля с паролем в базе данных пользователей).

буфер глубины: Структура данных для хранения глубины каждой точки итогового изображения. Является частью буфера кадра и заполняется в процессе рендеринга.

буфер кадра: Множество структур данных для хранения промежуточных результатов рендеринга и итогового изображения.

буфер трафарета: Структура данных для хранения значений, связанных с каждой точкой итогового изображения, интерпретацией которых управляет программист. Является частью буфера кадра и заполняется в процессе рендеринга.

буфер цвета: Структура данных для хранения цвета каждой точки итогового изображения. Является частью буфера кадра и заполняется в процессе рендеринга.

вершина трёхмерной модели: Точка в пространстве, являющаяся узлом сетки трёхмерной модели.

вершинный шейдер: Микропрограмма для графического процессора, отвечающая за преобразование вершин трёхмерных моделей.

видеопамять: Память видеокарты, предназначенная для хранения промежуточных данных и результата визуализации.

виртуальная реальность: Виртуальный мир в котором введены некоторые законы, ограничивающие действия пользователя, либо придающие этим

действиям некоторый смысл. Виртуальная реальность существует на базе некоторой программной системы (или комплекса систем), которая, в свою очередь, предусматривает определённые механизмы взаимодействия с пользователем.

виртуальный мир: Виртуальное пространство (как правило, имеющее графический образ), по которому пользователь может осуществлять навигацию в реальном времени, а также взаимодействовать с заполняющими это пространство объектами.

воксель: Элемент объёма, используемый для описания и / или отображения трёхмерных объектов.

геоинформационная система: Информационная система, данные в которой имеют привязку к географическим координатам (географической карте).

геометрический шейдер: Микропрограмма для графического процессора, отвечающая за сборку примитивов, из которых состоят трёхмерные модели.

грань трёхмерной модели (полигон): Фрагмент плоскости, ограниченный рёбрами трёхмерной модели. Поверхности любых трёхмерных объектов в векторной трёхмерной графике аппроксимируются такими фрагментами.

граф сцены: Структура данных, определяющая связи между объектами графической сцены.

графическая сцена: Пригодное для визуализации математическое описание множества объектов, определяющее их форму, визуальные характеристики и положение в пространстве.

графический конвейер: Цепочка преобразований данных, состоящая из выделенных ступеней (этапов), как правило, имеющих аппаратную поддержку. Служит для преобразования математической модели графической сцены в результирующее изображение (то есть для произведения рендеринга).

графический контекст: Совокупность системных структур данных, необходимых для отображения сцены при помощи графического API.

двойная буферизация: Механизм организации плавной анимации, предполагающий наличие двух страниц видеопамяти – активной и видимой. В то время, как происходит рендеринг в активную страницу, видима другая. Как только рендеринг закончился, страницы меняются ролями и процесс повторяется.

дезоксирибонуклеиновая кислота (ДНК): Макромолекула, обеспечивающая хранение, передачу из поколения в поколение и реализацию генетической программы развития и функционирования живых организмов.

делегат: Объект, адрес на который передан некоторому модулю с целью организации механизма оповещения о том, что произошло какое-либо событие: как только оговоренное событие происходит, модуль передаёт управление некоторому методу делегата. Часто для организации делегатов используются интерфейсы. Тогда делегатом может стать объект произвольного класса, реализующего нужный интерфейс.

диабет: Общее название заболеваний, сопровождающихся обильным выделением мочи.

дополненная реальность: Реальная картина мира, в которую в автоматическом режиме были добавлены некоторые искусственные элементы, связанные, однако, с реальными объектами. Чаще всего дополненная реальность строится при помощи видеокамеры, которая обеспечивает захват реальной картины мира, системы распознавания образов, которая выделяет в видеопотоке интересующие объекты, и системы визуализации, которая накладывает на кадры видеопотока некоторые синтетические образы, внешний вид и расположение которых диктуются информацией о распознанных объектах и внутренней логикой системы дополненной реальности.

захват движения (Motion Capture): Процесс автоматизированного преобразования движений человека или животного в движения соответствующего компьютерного персонажа. Захват движений выполняется либо при помощи оптических датчиков (например, при помощи устройства Microsoft Kinect), либо

при помощи датчиков, устанавливаемых непосредственно на теле того, чьи движения захватываются.

защищённое выделение памяти: Технология управления динамическим распределением памяти, при которой происходит журнализация всех выделений и освобождений памяти с целью автоматического отслеживания утечек.

изолиния: Линия, в каждой точке которой измеряемая величина сохраняет одинаковое значение.

изоповерхность: Поверхность, в каждой точке которой измеряемая величина сохраняет одинаковое значение.

инструкция умножения-сложения (multiply-add, MAD): Инструкция графического процессора, которая за один такт выполняет вычисление выражения вида $D = A \cdot B + C$, где A, B, C, D – действительные числа.

интерактивность: Свойство программной системы быть управляемой пользователем в режиме реального времени.

конволюция: Операция свёртки.

кроссплатформенность: Способность программной системы выполняться под управлением различных операционных систем в рамках одного класса устройств.

лицензия BSD: Лицензионное соглашение на программное обеспечение, по которому распространяется исходный код, допускается его модификация и использование в проектах с закрытым исходным кодом без лицензионных отчислений автору, но с указанием факта использования, изначального автора продукта и авторов всех модификаций.

лицензия Public Domain: Лицензионное соглашение, согласно которому имущественные авторские права на программный продукт отсутствуют, то есть программный продукт является общественным достоянием. Допускаются любые модификации такого программного продукта и использование его в любых проектах. Запрещается лишь присвоение авторства и изменение лицензии оригинального продукта.

максимальная цена: Максимальная цена товара на бирже за заданный промежуток времени.

манипуляционные перчатки (Data Glove): Устройство, предназначенное для преобразования движений кистей рук и пальцев пользователя в команды компьютерной системе. Представляет собой перчатки, оснащённые датчиками.

матрица проекции: Матрица преобразования, обеспечивающая трансформацию координат объектов в процессе проецирования на экран.

машина состояний: Глобальное хранилище настроек подсистемы вывода графики.

минимальная цена: Минимальная цена товара на бирже за заданный промежуток времени.

модуль графического расширения (графический движок): Библиотека, являющаяся надстройкой над низкоуровневым графическом API (например, отвечающим стандарту OpenGL или Direct3D) и упрощающая процесс написания графических систем.

мультимедийный ресурс: Общее название для всех внешних ресурсов (например, трёхмерных моделей, текстур и шейдеров), используемых в мультимедийной системе.

мультиплаформенность: Способность программной системы выполняться под управлением различных операционных систем в рамках различных классов устройств.

наночастица: Частица вещества размером порядка 10^{-9} метра.

низкополигональная трёхмерная модель: Трёхмерная модель, содержащая сравнительно небольшое количество полигонов, из-за чего не обладающая высокой детализацией.

облачный вычислитель: Легко масштабируемая вычислительная система, представляющая собой объединение большого количества географически разделённых вычислительных узлов.

ограничивающий параллелепипед: Параллелепипед, описанный около некоторого объекта или группы объектов.

однородные переменные (uniform-переменные): Переменные в шейдере, значения которым задаются из основного приложения. Однородные переменные являются основным средством настройки шейдера.

онтологический инжиниринг: Методология и технология представления, хранения и обработки знаний при помощи онтологий.

онтология: Множество понятий из некоторой предметной области, связей между ними и аксиом.

орбитальная камера (Arcball-Camera): Модель перемещения камеры, при которой камера движется по сфере заданного радиуса, и при этом ось взгляда всегда проходит через центр этой сферы. Такое поведение камеры является достаточно удобным для того, чтобы осмотреть группу объектов со всех сторон, а потому часто используется в программах просмотра трёхмерных моделей.

освещение по Фонгу: Модель освещения в компьютерной графике, при которой цвет каждой точки поверхности определяется по формуле: $c = c_d \cdot \max\left\{\cos\alpha, 0\right\} + c_s \cdot \max\left\{\cos^N\beta, 0\right\}$, где c – цвет точки, c_d – диффузный цвет материала поверхности, α – угол между вектором, указывающим направление к источнику света и нормалью к поверхности, c_s – цвет зеркального блика материала поверхности, N – показатель интенсивности зеркального блика материала поверхности, β – угол между направлением отражённого света и вектором, указывающим направление к наблюдателю.

отсечение задних граней: Исключение граней из растеризации в том случае, если они развёрнуты к наблюдателю своей тыльной стороной.

отсечение по усечённой пирамиде видимости: Исключение из списка рендеринга объектов, которые в результате преобразований размещения на сцене не попадают в усечённую пирамиду видимости, то есть заведомо не будут видны наблюдателю.

панкреатит: Группа заболеваний и синдромов, при которых наблюдается воспаление поджелудочной железы.

параллелепипед видимости: Область трёхмерного пространства, видимая наблюдателю и в условиях использования параллельной проекции представляющая собой параллелепипед.

планшетный компьютер: Легковесный мобильный компьютер, как правило, оснащённый лишь сенсорным экраном и по размеру примерно равный тетради формата A5.

поверхность уровня: Поверхность, в каждой точке которой измеряемая величина не больше (не меньше) порогового значения.

постобрботка: Дополнительная обработка результата рендеринга гарфической сцены с целью привнесения дополнительных визуальных эффектов.

преобразование камеры: Первое аффинное преобразование сцены, метафорически соответствующее размещению наблюдателя (виртуальной камеры). Чаще всего данное преобразование, так же, как и все остальные, выражается через матрицу преобразования.

преобразование порта просмотра: Финальный этап преобразований графического конвейера, в результате которого координаты объектов приводятся к экранным.

разнообразные переменные (varying-переменные): Переменные в шейдере, значения которых определяются шейдером более ранней ступени графического конвейера. Разнообразные переменные являются средством передачи данных от шейдера к шейдеру вперёд по графическому конвейеру.

растеризация: Процесс подбора точек растрового изображения для того, чтобы визуализировать некоторый объект, имеющий математическое описание формы и положения.

ребро трёхмерной модели: Отрезок, соединяющий две вершины трёхмерной модели.

рендеринг: Процесс визуализации сцены, построение изображения по математической модели.

рендеринг в текстуру: Визуализация сцены, выполняемая не с целью непосредственного показа результирующего изображения на экране, а с целью использования этого изображения в качестве текстуры в процессе другой визуализации.

рендеринг объёмов: Способ визуализации трёхмерных объектов, при котором отображается не только их поверхность, но и весь их объём.

рендеринг сечений: Способ визуализации трёхмерных объектов, при котором отображается не весь объект, а сечение его некоторой плоскостью.

репозиторий: Место, где хранятся и поддерживаются какие-либо данные.

ретина-дисплей: Запатентованный компанией Apple дисплей высокой чёткости, которая обеспечивается путём уменьшения площади индивидуальных пикселей.

решатель: Программа или программно-аппаратный комплекс, производящий некоторый эксперимент и генерирующий подлежащие визуализации данные.

секвенация ДНК: Процесс расшифровки молекул ДНК, извлекаемых из биологического материала, с применением специального программного и аппаратного обеспечения.

сериализация: Процесс преобразования какой-либо структуры данных в последовательность битов с целью сохранения на определённом носителе информации или передачи по компьютерной сети.

синглтон: Объект некоторого класса, существующий в приложении в единственном экземпляре (создать второй экземпляр этого класса невозможно из-за установленных программистом ограничений), к которому организуется возможность глобального доступа.

смартфон: Мобильный телефон с интегрированным компьютером.

сокет: Программный интерфейс для обеспечения обмена данными между процессами.

стандартный поток ошибок (stderr): Поток вывода в UNIX-подобных ОС, зарезервированный для диагностических и отладочных сообщений, а также сообщений об ошибках.

стереомонитор: Монитор, способный отображать стереоскопические изображения. Чаще всего для этого используется либо поляризационная технология (левый и правый кадры стереопары отображаются светом разной поляризации, а разделение для человека осуществляется при помощи очков с поляризационными светофильтрами), либо технология параллакс-барьера (монитор отображает изображение, в котором чередуются строчки левого и правого кадров стереопары, перед монитором располагается решётка из лентикулярных линз, обеспечивающая наблюдателю такой параллакс, чтобы одним глазом он видел только строчки одного кадра, а другим – только другого), либо эклипсная (затворная) технология (монитор на большой скорости показывает сначала левый, затем правый кадры стереопары, а очки наблюдателя, оснащённые жидкокристаллическими дисплеями, изменяют прозрачность стёкол так, чтобы правый глаз оказался «закрыт», когда монитор отображает левый кадр, и наоборот).

стереоскопическое изображение: Плоское изображение, которое, тем не менее, создаёт у наблюдателя эффект восприятия протяжённости пространства.

суперкомпьютер: Компьютер, обладающий значительно более высокой производительностью, чем обычный персональный.

текстура: Изображение, по определённому правилу накладываемое на определённую поверхность с целью придания ей особых визуальных качеств.

текстурные координаты: Координаты в системе, введённой на текстурном изображении. По таким координатам происходит выборка цвета из текстуры.

тест буфера глубины: Сравнение глубины обрабатываемой точки изображения со значением в буфере глубины с целью выяснения, перекрывается

ли обрабатываемый трёхмерный объект в данной точке уже обработанным объектом. В том случае, если перекрытия нет, для точки вычисляется её цвет, а значение её глубины записывается в буфер глубины. Если же перекрытие имеет место, точка отбрасывается как невидимая.

трассировка лучей: Алгоритм рендеринга, при котором от наблюдателя на сцену запускаются лучи и определяются их столкновения с объектами. Цвета точек изображения определяются на основе цветов объектов, с которыми произошло столкновение соответствующих лучей.

трёхмерная модель (3D-модель): Математическое описание объёмного объекта. В векторной трёхмерной графике трёхмерные модели представляют собой множество вершин, рёбер и граней.

трёхмерное сканирование: Процесс автоматизированного построения трёхмерных моделей объектов реального мира.

умные указатели: Технология управления динамическим распределением памяти, при которой указатели на объекты представляют собой специализированные структуры данных, отвечающие за подсчёт ссылок. Объекты, ссылок на которые не осталось, автоматически удаляются, что снижает риск возникновения утечек памяти.

усечённая пирамида видимости: Область трёхмерного пространства, видимая наблюдателю и в условиях использования перспективной проекции представляющая собой усечённую пирамиду.

условное перемещение (conditional move): Инструкция процессора, в результате которой данные из одного места копируются в другое только в том случае, если истинно некоторое условие.

филогенетическое дерево: Древовидная структура данных, передающая родственные связи между живыми организмами (или иными родственными сущностями).

фильтр постобработки: Конкретный алгоритм для постобработки изображения.

фильтрация текстуры: Вычисление цвета точки текстуры по заданным текстурным координатам.

формы Бэкуса-Наура (БНФ): Формальная система описания синтаксиса языка, в которой одни синтаксические категории последовательно определяются через другие.

фоторелистичное изображение: Искусственно созданное изображение каких-либо объектов, передающее их внешний вид с максимальной реалистичностью, в идеале не отличимое от фотографии.

фрагментный шейдер: Микропрограмма для графического процессора, отвечающая за определение цветов точек итогового изображения.

функция обратного вызова: Функция, адрес на которую был передан некоторому модулю с целью организации механизма оповещения о том, что произошло какое-либо событие: как только оговоренное событие происходит, модуль передаёт управление по сообщённому ему адресу.

цветовая модель RGB: Цветовая модель, в которой цвет кодируется при помощи интенсивности излучения красного (*англ.* red, R), зелёного (*англ.* green, G) и синего (*англ.* blue, B) света.

цена закрытия: Цена товара на бирже на конец заданного промежутка времени.

цена открытия: Цена товара на бирже на начало заданного промежутка времени.

шейдер: Микропрограмма для графического процессора, определяющая алгоритм работы некоторой ступени графического конвейера.

шлем виртуальной реальности (Head-Mounted Display): Устройство для демонстрации генерируемых на компьютере изображений, включающее в себя очки, вместо стёкол в которых располагаются небольшие дисплеи. Такое устройство позволяет увеличить «эффект присутствия» наблюдателя, так как изолирует его от посторонних зрительных раздражителей, переключая всё его внимание на изображения, создаваемые компьютером. При условии качественного

исполнения, шлем виртуальной реальности способен создать иллюзию того, что человек видит мир глазами виртуального персонажа (т. н. «эффект погружения»).

язык ESSL: Язык программирования шейдеров, входящий в спецификацию OpenGLES.

язык GLSL: Язык программирования шейдеров, входящий в спецификацию OpenGL.

Литература

1. Бондарев А.Е., Галактионов В.А. Анализ многомерных данных в задачах многопараметрической оптимизации с применением методов визуализации // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2012. – К. 2, Т. 4, №2. – С. 1-13.
2. Friendly M. Milestones in the history of thematic cartography, statistical graphics, and data visualization [Электронный ресурс]. – York University, 2009. – 79 p. – URL: http://www.math.yorku.ca/SCS/Gallery/milestone/milestone.pdf.
3. Crassin C. GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes / PhD Thesis from Universite de Grenoble. – Grenoble, 2011. – 207 p.
4. Sutherland I. Sketchpad, A Man-Machine Graphical Communication System / PhD Thesis from Massachusetts Institute of Technology. – Cambridge, 1963. – 149 p.
5. Haimes B. Homepage [Электронный ресурс]. – Massachusetts Institute of Technology. – URL: http://raphael.mit.edu/haimes.html.
6. Ju T. Homepage [Электронный ресурс]. – Washington University in St. Louis. – URL: http://www.cs.wustl.edu/~taoju/.
7. Gruchalla K. Homepage [Электронный ресурс]. – University of Colorado at Boulder. – URL: http://kenny.gruchalla.org/.
8. Kuttel M. Homepage [Электронный ресурс]. – University of Cape Town. – URL: http://people.cs.uct.ac.za/~mkuttel/.
9. Wolfram S. Homepage [Электронный ресурс]. – Wolfram Research. – URL: http://www.stephenwolfram.com/.
10. Dzhosan O.V, Popova N.N., Korzh A.A. Hierarchical Visualization System for High Performance Computing // pros. conf. ParCo 2009. – France, Lyon, 2009.

11. Игнатенко А.В. Методы интерактивной визуализации и обработки трехмерных данных на основе изображений / дис. … канд. физ.-мат. наук: 05.13.11. – М., 2005. – 148 с.
12. Сурин А, Сурина О. К иммерсивной визуализации математики // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2010. – К. 2, Т. 2, №2. – С. 1-19.
13. Карташева Е.Л., Багдасаров Г.А., Болдарев А.С., Гасилова И.В., Дьяченко С.В., Ольховская О.Г., Шмыров В.А., Болдырев С.Н., Гасилов В.А. Визуализация данных вычислительных экспериментов в области 3D моделирования излучающей плазмы, выполняемых на многопроцессорных вычислительных системах с помощью пакета Marple // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2010. – К. 1, Т. 2, №1. – С. 1-25.
14. Авербух В.Л., Байдалин А.Ю., Бахтерев М.О., Васёв П.А., Казанцев А.Ю., Манаков Д.В. Опыт разработки специализированных систем научной визуализации // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2010. – К. 4, Т. 2, №4. – С. 27-39.
15. Бондарев А.Е. Оптимизационный анализ нестационарных пространственно-временных структур с применением методов визуализации // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2011. – К. 2, Т. 3, №2. – С. 1-11.
16. Галактионов В.А. Программные технологии синтеза реалистичных изображений / дис. … докт. физ.-мат. наук: 05.13.11. – М., 2006. – 236 с.
17. Волобой А.Г. Программные технологии автоматизации построения реалистичных изображений / дис. … докт. физ.-мат. наук: 05.13.11. – М., 2012. – 248 с.

18. Васильев В.Р., Волобой А.Г., Вьюкова Н.И., Галактионов В.А. Контекстная визуализация пространственных данных // Информационные технологии и вычислительные системы. – М., 2004. – №4. – С. 25-34.
19. Гаврилов Н.И., Турлапов В.Е. Подходы к оптимизации GPU-алгоритма volume raycasting для применения в составе виртуального анатомического стола // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2012. – К. 2. Т. 4. № 2. – С. 21-56.
20. Залогова Л.А. Компьютерная графика (учебное пособие и практикум) // Успехи современного естествознания. – 2010. – № 9 – С. 77-79.
21. Смирнов С.В. Модели и методы автоматизированного проектирования информационных систем со сложно структурированными графическими данными / дис. … канд. техн. наук: 05.13.11. – М., 2005. – 130 с.
22. Артамонов Е.И., Разумовский А.И., Курков С.В., Курков А.С. Импорт графических данных в программе расчёта конструкций методом конечных элементов «Зенит-95» // Труды. V-й международной конференции «Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM – 2005) / под ред. Е.И. Артамонова. – М.: Институт проблем управления РАН, 2005.
23. Максименко-Шейко К.В., Толок А.В., Шейко Т.И. R-функции как аппарат в приложениях фрактальной геометрии // Прикладная информатика. – М.: НОУ ВПО «МФПУ «Синергия», 2010. – №6 (30). – С. 21-28.
24. Авербух В.Л., Бахтерев М.О., Васёв П.А., Кузнецов Я.Д. Развитие программных средств научной визуализации // Параллельные вычислительные технологии (ПаВТ'2014): труды международной научной конференции (1–3 апреля 2014 г., г. Ростов-на-Дону). – Челябинск: Издательский центр ЮУрГУ, 2014. – С. 359.

25. Зацаринный А.А., Чупраков К.Г. Некоторые аспекты выбора технологии для построения систем отображения информации ситуационного центра // Информатика и её применения. – 2010. – №3. – С. 62-71.
26. The United States Patent and Trademark Office. Retina [Электронный ресурс]. – 2011. – URL: http://tsdr.uspto.gov/#caseNumber=85056807&caseType=SERIAL_NO&searchType=statusSearch.
27. Джосан О.В. Исследование и разработка методов и программных средств визуализации результатов научных вычислений для массивно-параллельных вычислительных систем / дис. … канд. физ.-мат. наук: 05.13.11. – М., 2009. – 131 с.
28. Abela R.L., Parfittc S., Ashtone N., Lewisf S.G., Scotte B., Stringerd C. Digital preservation and dissemination of ancient lithic technology with modern micro-CT // Computers & Graphics. – Elsevier, 2011. – Vol. 35, I. 4. – P. 878-884.
29. Зелепухина В.А. Обработка и визуализация данных в виртуальных лабораториях удаленного доступа на основе модельно-ориентированного подхода / дис. … канд. техн. наук: 05.13.18. – М., 2008. – 166 с.
30. Keval H., Sasse M.A. To catch a thief – you need at least 8 frames per second: the impact of frame rates on user performance in a CCTV detection task // Proceedings of the 16th ACM international conference on Multimedia. – ACM, 2008. – P. 941-944.
31. Richardson T., Stafford-Fraser Q., Wood R.R., Hopper A. Virtual Network Computing // IEEE Internet Computing. – 1998. – Vol. 2, I. 1. – P. 33-38.
32. Jang S.-M., Choi W.-H., Kim W.-Y. An Efficient Application Virtualization Mechanism using Separated Software Execution System // International Journal of Software Engineering and Its Applications. – 2012. – Vol. 6, No. 4. – P. 257-264.

33. VMWare. Graphics Acceleration in View Virtual Desktops [Электронный ресурс]. – 2014. – 32 p. – URL: https://www.vmware.com/files/pdf/techpaper/vmware-horizon-view-graphics-acceleration-deployment.pdf.
34. Барладян Б.Х., Волобой А.Г., Вьюкова Н.И., Галактионов В.А., Дерябин Н.Б. Моделирование освещенности и синтез фотореалистичных изображений с использованием Интернет технологий // Программирование. – М., 2005. – №5. – С. 66-80.
35. Moreland K. [Paraview] anti aliasing [Электронный ресурс]. – 2011. – URL: http://public.kitware.com/pipermail/paraview/2011-June/022062.html.
36. QuantumUmbrella Ltd. Phone Count [Электронный ресурс]. – 2014. – URL: http://www.phonecount.com/pc/count.jsp.
37. Виноградов М.А. Современные средства визуализации и обработки двумерных научных данных [Электронный ресурс]. – AMLab, 2002. – URL: http://www.amlab.ru/paper_max.shtml.
38. Ahrens J., Geveci B., Law C. ParaView: An End-User Tool for Large Data Visualization / edited by C.D. Hansen, C.R. Johnson. – Elsevier, 2005. – 984 p.
39. FEI Visualization Sciences Group. Avizo Fire for Core samples and Digital rock physics [Электронный ресурс]. – 2013. – 4 p. – URL: http://www.vsg3d.com/sites/default/files/related/fei_sh_core_sample_digital_rock_physics.pdf.
40. Котов Д.С. Математическое и алгоритмическое обеспечение для системы визуализации в САПР / дис. … канд. техн. наук: 05.13.12. – Владимир, 2011. – 197 с.
41. Limet S., Poquet M., Robert S. Modulight: A Framework for Efficient Dynamic Interactive Scientific Visualization // Procedia Computer Science. – Elsevier, 2014. – Vol. 29. – P. 692-702.
42. Allard J., Gouranton V., Lecointre L., Limet S., Melin E., Raffin B., Robert S. FlowVR: A Middleware for Large Scale Virtual Reality Applications //

Proceedings of Euro-Par 2004 Parallel Processing. – Springer, 2004. – Vol. 3149 of LNCS. – P. 497–505.

43. Alm D. PocketCAS [Электронный ресурс]. – 2014. – URL: http://pocketcas.com/.
44. Jockusch W. Free Graphing Calculator [Электронный ресурс]. – 2014. – URL: https://itunes.apple.com/us/app/free-graphing-calculator/id378009553?mt=8.
45. Yuan J., Knapp G., Liner S., Hynes C., Olatinwo B., Brooks A., Ekpo E., Ge J., Ham K., Beaird R., Butler J.G. Data Workflow and Visualization: X-Ray Interferometry, Vistrails and IPad/Android Collaboration // Proceedings of Louisiana EPSCoR RII LA-SiGMA Symposium. – 2013. – P. 17-20.
46. Ahrens J., Law C., Schroeder W., Martin K., Papka M. A Parallel Approach for Efficiently Visualizing Extremely Large, Time-Varying Datasets // Los Alamos National Laboratory. – 2000. – Technical Report #LAUR-00-1620.
47. Enthought, Inc. Chaco [Электронный ресурс]. – 2013. – URL: http://code.enthought.com/projects/chaco/.
48. Vermeulen G., Colclough M. PyQwt [Электронный ресурс]. – 2010. – URL: http://pyqwt.sourceforge.net/.
49. Lehmann J., Schindler M., Wobst A. PyX [Электронный ресурс]. – 2013. – URL: http://pyx.sourceforge.net/index.html.
50. Hunter J. Matplotlib [Электронный ресурс]. – 2012. – URL: http://matplotlib.org/.
51. MathGL Group. MathGL [Электронный ресурс]. – 2014. – URL: http://mathgl.sourceforge.net/doc_en/Main.html.
52. Unreal Engine [Электронный ресурс]. – 2014. – URL: http://www.unrealengine.com/.
53. Hu J.W., Feng C., Liu Y., Zhu R.Y. UTSE: A Game Engine-Based Simulation Environemnt for Agent // Applied Mechanics and Materials. – 2014. – P. 2142-2145.

54. Mat R.C., Shariff A.R.M., Zulkifli A.N., Rahim M.S.M, Mahayudin M.H. Using game engine for 3D terrain visualisation of GIS data: A review // IOP Conference Series: Earth and Environmental Science. – 2014. – Vol. 20, I. 1. – P. 1-11.
55. SIO2Interactive. SIO2 Engine [Электронный ресурс]. – 2014. – URL: http://sio2interactive.com/index.php.
56. Shi H.W., Yang X.Y., Lin H.P. OGRE Engine Based on Three-Dimensional Modeling of Biological Cells // Advanced Materials Research. – 2010. – Vol. 159. – P. 662-666.
57. Pejsa T., Pandzic I.S. Architecture of an Animation System for Human Characters // Telecommunications, 2009. ConTEL 2009. – Zagreb: IEEE, 2009. – P. 171-176.
58. Oolong [Электронный ресурс]. – 2010. – URL: http://code.google.com/p/oolongengine/.
59. Hollings B. Cocos2D / Cocos3D [Электронный ресурс]. – 2014. – URL: https://github.com/cocos3d/cocos3d/.
60. Zechner M. libGDX [Электронный ресурс]. – 2014. – URL: http://libgdx.badlogicgames.com/.
61. Blanchette J., Summerfield M. C++ GUI Programming with Qt 4 / 2nd Edition. – Prentice Hall, 2008. – 572 p.
62. Krause A. Foundations of GTK+ Development. – Apress, 2007. – 630 p.
63. Welch B., Jones K. Practical Programming in Tcl and Tk / 4nd Edition. – Prentice Hall, 2003. – 960 p.
64. Zukowski J. Java AWT Reference. – O'Reilly Media, 1997. – 1074 p.
65. Loy M., Eckstein R., Wood D., Elliott J., Cole B. Java Swing / 2nd Edition. – O'Reilly Media, 2002. – 1280 p.
66. Osborn J. Mono and the .NET Linux Opportunity [Электронный ресурс]. – Novell, 2010. – Technical White Paper. – 23 p. – URL: http://www.novell.com/docrep/2010/02/Mono_and_the_Net_Linux_Opportunity_en.pdf.

67. Linares-Pellicer J., Carrasquer-Moya E., Esparza-Peidro J., Tormo-Llácer J. et al. MobWeb Erasmus Intensive Programme 2.0. New Trends In Mobile And Web Development 2012 / Edited by J. Linares-Pellicer, R. Doina Zmaranda, M. Welin. – Tampereen yliopistopaino Oy, 2012. – 528 p.
68. Paramonov I., Vasilev A., Kozhemyakin N., Timofeev I., Krylov E., Subbotkin A., Korzun D., Galov I. Cross-platform Development of Smart Conference Clients // Roceeding Of The 11th Conference Of Fruct Association. – St.-Petersburg, 2012. – P. 197-198.
69. Sletta G. Introducing Boot to Qt – A Technology Preview [Электронный ресурс] // QtBlog. – 2013. – URL: http://blog.qt.digia.com/blog/2013/05/21/introducing-boot-to-qt-a-technology-preview/.
70. Pilgrim M. HTML5: Up and Running. – O'Reilly Media, 2010. – 222 p.
71. Adobe Creative Team. Adobe Flash Professional CS5 Classroom in a Book. – Adobe Press, 2010. – 384 p.
72. Piltch A. How to Enable WebGL Support on Chrome for Android [Электронный ресурс]. – 2013. – URL: http://blog.laptopmag.com/how-to-enable-webgl-support-on-chrome-for-android/.
73. Panth S., Jivani M. Device Control in an Ad-hoc Network Environment by using MoSync for Multiple Platform Mobile Application Development // International Journal of Computer Science & Engineering Technology (IJCSET). – 2013. – Vol. 4 No. 8. – P. 1145-1152.
74. Crits-Christoph A. Innovative Interfaces with Clutter [Электронный ресурс] // Linux Journal. – 2009. – URL: http://www.linuxjournal.com/magazine/innovative-interfaces-clutter/.
75. Noy N.F., McGuinness D.L. Ontology Development 101: A Guide to Creating Your First Ontology [Электронный ресурс]. – Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001. – 25 p. – URL: http://protege.stanford.edu/publications/ontology_development/ontology101.pdf.

76. Рябинин К.В. Программа для ЭВМ SciVi: свидетельство о государственной регистрации программы для ЭВМ № 2014616257 / Зарег. 18.06.2014, опубл. 20.07.2014. Официальный бюллетень Федеральной службы по интеллектуальной собственности (Роспатент) «Программы для ЭВМ. Базы данных. Топологии интегральных микросхем», №7 (93), 2014, 1 с.
77. Ryabinin K, Chuprina S. Adaptive Scientific Visualization System for Desktop Computers and Mobile Devices // Procedia Computer Science. – Elsevier, 2013. – Vol. 18. – P. 722-731.
78. Ryabinin K, Chuprina S. Development of Multiplatform Adaptive Rendering Tools to Visualize Scientific Experiments // Procedia Computer Science. – Elsevier, 2014. – Vol. 29. – P. 1825-1834.
79. Рябинин К.В. Разработка мультиплатформенной клиент-серверной системы для научной визуализации // Научно-технический вестник Поволжья. – Казань: Научно-технический вестник Поволжья, 2013. – №2. – С. 197-203.
80. Рябинин К.В. Разработка адаптивного мультиплатформенного визуализатора результатов научных расчётов для высокопроизводительных вычислительных систем // Труды 22-й Международной конференции по компьютерной графике и машинному зрению «ГрафиКон'2012». – М., 2012. – С. 193-198.
81. Рябинин К.В. Разработка адаптивного мультиплатформенного визуализтора результатов научных расчётов для высокопроизводительных вычислительных систем // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2012. – К. 4, Т. 4, №4. – С. 17-29.
82. Рябинин К.В. Разработка мультиплатформенной системы научной визуализации // Междисциплинарные исследования: сб. матер. науч.-практ. конф. (Пермь, 9 – 11 апреля 2013 г.) / гл. ред. Ю.А. Шарапов; Перм. гос. нац. ис-след. ун-т. – Пермь, 2013. – Т. 1. – С. 179-182.

83. Рябинин К.В. Подход к разработке системы визуализации результатов высокопроизводительных научных вычислений // Высокопроизводительные вычисления на графических процессорах: материалы II Всероссийской науч.-практ. конф. с междунар. участием с элементами науч. шк. для молодёжи, 2 – 6 июня 2014 г. / отв. за вып. С.В. Русаков, М.М. Бузмакова; Перм. гос. нац. исслед. ун-т. – Пермь, 2014. – С. 50-55.
84. Forcier J., Bissex P., Chun W. Python Web Development with Django. – Addison-Wesley Professional, 2008. – 408 p.
85. Powell A. Metadata for the Web. RDF and the Dublin Core [Электронный ресурс]. – 1998. – URL: http://www.ukoln.ac.uk/metadata/presentations/ukolug98/paper/intro.html.
86. Patel-Schneider P.F., Hayes P., Horrocks I. OWL Web Ontology Language Semantics and Abstract Syntax [Электронный ресурс]. – W3C, 2004. – URL: http://www.w3.org/TR/2004/REC-owl-semantics-20040210/.
87. Филоненко Ф.С. Программа для ЭВМ NFoundation: свидетельство о государственной регистрации программы для ЭВМ № 2014613793 / Зарег. 07.04.2014, опубл. 20.05.2014. Официальный бюллетень Федеральной службы по интеллектуальной собственности (Роспатент) «Программы для ЭВМ. Базы данных. Топологии интегральных микросхем», №5 (91), 2014, 1 с.
88. Karlsson B. Beyond the C++ Standard Library: An Introduction to Boost. – Addison-Wesley Professional, 2005. – 432 p.
89. Sadun E. iOS Drawing: Practical UIKit Solutions. – Addison-Wesley Professional, 2013. – 308 p.
90. Рябинин К.В., Филоненко Ф.С., Полотнянщиков И.С. Программа для ЭВМ NGraphics: свидетельство о государственной регистрации программы для ЭВМ № 2014613796 / Зарег. 07.04.2014, опубл. 20.05.2014. Официальный бюллетень Федеральной службы по интеллектуальной собственности

(Роспатент) «Программы для ЭВМ. Базы данных. Топологии интегральных микросхем», №5 (91), 2014, 1 с.

91. Рябинин К.В., Чудинов Е.А. Программа для ЭВМ NChart3D: свидетельство о государственной регистрации программы для ЭВМ № 2014616258 / Зарег. 18.06.2014, опубл. 20.07.2014. Официальный бюллетень Федеральной службы по интеллектуальной собственности (Роспатент) «Программы для ЭВМ. Базы данных. Топологии интегральных микросхем», №7 (93), 2014, 1 с.
92. Рябинин К.В. Разработка мультиплатформенной библиотеки построения и визуализации диаграмм // Научная визуализация. – М.: Национальный исследовательский ядерный университет МИФИ, 2014. – К. 1, Т. 6, №1. – С. 51-67.
93. Рябинин К.В. GUI-фреймворк для настольных компьютеров и мобильных систем // Актуальные проблемы механики, математики, информатики: сб. тез. науч.-практ. конф. (Пермь, 30 октября – 1 ноября 2012 г.) / гл. ред. В.И. Яковлев; Перм. гос. нац. исслед. ун-т. – Пермь, 2012. – С. 195.
94. Liang S. The Java Native Interface. Programmer's Guide and Specification. – An imprint of Addison Wesley Longman, Inc., 1999. – 318 p.
95. Wang X., Yang B.-L. 3D Models Simplification Algorithm for Mobile Devices // Journal Of Software. – 2012. – Vol. 7, No. 11. – P. 2599-2605.
96. Munshi A., Leech J. OpenGLES Common Profile Specification. Version 2.0.25 (Full Specification) [Электронный ресурс]. – The Khronos Group Inc., 2010. – P. 21-22. – URL: https://www.khronos.org/registry/gles/specs/2.0/es_full_spec_2.0.25.pdf.
97. Рогачев С.Н. Обобщенный Model-View-Controller. Каркас на основе шаблона проектирования MVC в исполнении Generic Java и C# [Электронный ресурс]. – RSDN, 2007. – URL: http://rsdn.ru/article/patterns/generic-mvc.xml.
98. Nathan A. Windows 8 Apps with Xaml and C# Unleashed. – Sams, 2012. – 624 p.

99. Assarsson U., Möller T. Optimized View Frustum Culling Algorithms for Bounding Boxes // Journal of Graphics Tools. – Taylor & Francis, 2000. – Vol. 5. – P. 9-22.
100. Lewis J. 3DS File Format [Электронный ресурс]. – 1998. – URL: http://www.the-labs.com/Blender/3DS-details.html.
101. Bourke P. PLY – Polygon File Format [Электронный ресурс]. – URL: http://paulbourke.net/dataformats/ply/.
102. Simpson R.J. The OpenGL ES Shading Language [Электронный ресурс]. – The Khronos Group Inc., 2009. – 119 p. – URL: https://www.khronos.org/files/opengles_shading_language.pdf.
103. Kessenich J. The OpenGL Shading Language [Электронный ресурс]. – Khronos Group Inc., 2014. – 209 p. – URL: https://www.opengl.org/registry/doc/GLSLangSpec.4.50.pdf.
104. Smith B. Arcball [Электронный ресурс]. – 2006. – URL: http://rainwarrior.ca/dragon/arcball.html.
105. 3D Center. Multisampling Anti-Aliasing: A Closeup View [Электронный ресурс]. – 3D Center, 2003. – URL: http://alt.3dcenter.org/artikel/multisampling_anti-aliasing/index_e.php.
106. Woligroski D. Anti-Aliasing Analysis, Part 1: Settings And Surprises [Электронный ресурс] // Tom's Hardware. – 2011. – URL: http://www.tomshardware.com/reviews/anti-aliasing-nvidia-geforce-amd-radeon,2868.html.
107. Lizandra M.C.J. Graphic libraries for Windows programming // Crossroads, the ACM Student Magazine (ACM). – 2000. – Vol. 6 I. 4. – P. 14-18.
108. Young P. CSAA (Coverage Sampling Antialiasing) [Электронный ресурс]. – NVidia. – URL: http://www.nvidia.ru/object/coverage-sampled-aa.html.
109. Pettineo M. Deferred MSAA [Электронный ресурс]. – 2010. – URL: http://mynameismjp.wordpress.com/2010/08/16/deferred-msaa/.

110. Reshetov A. Morphological Antialiasing [Электронный ресурс] // Proceedings of the 2009 ACM Symposium on High Performance Graphics. – 2009. – 8 p. – URL: http://visual-computing.intel-research.net/publications/papers/2009/mlaa/mlaa.pdf.
111. Jimenez J., Echevarria J.I., Sousa T., Gutierrez D. SMAA: Enhanced Morphological Antialiasing [Электронный ресурс] // Computer Graphics Forum (Proc. EUROGRAPHICS 2012). – 2012. – Vol. 31, No. 2. – URL: http://www.iryoku.com/smaa/.
112. NVidia. Temporal Anti-Aliasing (TXAA) [Электронный ресурс]. – NVidia. – URL: http://www.geforce.com/hardware/technology/txaa/technology/.
113. Lottes T. Fast Approximate Anti-aliasing [Электронный ресурс]. – NVidia, 2009. – 15 p. – URL: http://developer.download.nvidia.com/assets/gamedev/files/sdk/11/FXAA_WhitePaper.pdf.
114. Subhransu B., Sigman D. Using NVidia FxAA in my code: Whats the licensing model? [Электронный ресурс] // StackOverflow. – 2012. – URL: http://stackoverflow.com/questions/12170575/using-nvidia-fxaa-in-my-code-whats-the-licensing-model.
115. Рябинин К.В. Адаптивное сглаживание границ объектов на изображении для мобильных устройств // Вестник компьютерных и информационных технологий. – М.: Спектр, 2014. – №8. – С. 23-28.
116. Рябинин К.В. Динамический антиалиасинг на мобильных устройствах // Современные проблемы математики и её прикладные аспекты – 2013: сб. тез. науч.-практ. конф. (Пермь, 29 – 31 октября 2013 г.) / гл. ред. В.И. Яковлев; Перм. гос. нац. исслед. ун-т. – Пермь, 2013. – С. 25.
117. Guinot J. (Tested) Fast Approximate Anti-Aliasing (FXAA) Demo (GLSL) [Электронный ресурс]. – Geeks3D, 2011. – URL: http://www.geeks3d.com/20110405/fxaa-fast-approximate-anti-aliasing-demo-glsl-opengl-test-radeon-geforce/3/.

118. OpenGL. GLSL Optimizations [Электронный ресурс]. – OpenGL, 2013. – URL: https://www.opengl.org/wiki/GLSL_Optimizations#Get_MAD.
119. Khailany D., Dally W.J., Rixner S., Kapasi U.J., Mattson P., Namkoong J., Owens J.D., Towles B., Chang A. Imagine: Media Processing with Streams // IEEE Micro. – 2001. – P. 35-46.
120. Райт-мл. Р., Липчак Б. OpenGL Суперкнига. – М.: Вильямс, 2006. – С. 177-203.
121. Weller H.G., Tabor G., Jasak H., Fureby C. A tensorial approach to computational continuum mechanics using object orientated techniques // Computers in Physics. – 1998. – Vol. 12, No. 6. – P. 620-631.
122. Деменев А.Г., Белозерова Т.С., Харебов П.В., Хеннер Е.К., Хеннер В.К. О решении задач магнитодинамики и когерентных процессов в наномагнитных структурах на суперкомпьютере // Высокопроизводительные параллельные вычисления на кластерных системах. Материалы XII Всероссийской конференции (Н. Новгород, 26-28 ноября 2012 г.). – Н. Новгород: НГГУ, 2012. – С.122-124.
123. Larkin M.A. et al. ClustalW and ClustalX version 2 // Bioinformatics. – 2007. – Vol. 23, Iss. 21 – P. 2947-2948.
124. Podtaev S, Morozov M., Frick P. Wavelet-based correlations of skin temperature and blood flow oscillations // Cardiovascular Engineering. – 2008. – P. 1567-8822.
125. Podtaev S., Smirnova E., Popov A., Ershova A., Zhukova E. Skin temperature fluctuations at contralateral cooling test in type 2 diabetic patiens // 9th World congress for microcirculation. – Paris, France, 26th - 28th September 2010. – P. 77.
126. Smirnova E., Podtaev S., Mizeva I., Loran E. Assessment of endothelial dysfunction in patients with impaired glucose tolerance during a cold pressor test // Diabetes and Vascular Disease Research. – 2013. – Vol. 10, No. 6. – P. 489-497.

127. Филоненко Ф.С., Поспелов В.В., Сарафанов П.С., Рябинин К.В., Полотнянщиков И.С. Программа для ЭВМ Remotix: свидетельство о государственной регистрации программы для ЭВМ № 2014613795 / Зарег. 07.04.2014, опубл. 20.05.2014. Официальный бюллетень Федеральной службы по интеллектуальной собственности (Роспатент) «Программы для ЭВМ. Базы данных. Топологии интегральных микросхем», №5 (91), 2014, 1 с.

Приложение 1. Копии актов о внедрении результатов диссертационного исследования

Общество с ограниченной ответственностью
«Ньюлана»

«УТВЕРЖДАЮ»
Генеральный директор
Егоркина Наталья Борисовна
«19» февраля 2014 г.

АКТ О ВНЕДРЕНИИ
результатов диссертационной работы на соискание степени кандидата физико-математических наук Рябинина Константина Валентиновича

Комиссия в составе:

председатель:	Поспелов Владислав Владиславович
члены комиссии:	Репин Никита Ильич
	Чудинов Евгений Александрович

составили настоящий акт о том, что программный продукт «SciVi» (система научной визуализации), разработанный К.В. Рябининым в рамках диссертационной работы «Методы и средства разработки адаптивных мультиплатформенных систем визуализации научных экспериментов», представленной на соискание учёной степени кандидата физико-математических наук по специальности 05.13.11, основанный на созданной в ООО «Ньюлана» библиотеке визуализации NGraphics, использован в деятельности ООО «Ньюлана» для решения следующих задач:

1. Визуализация соотношения скоростей передачи данных по сети с целью отладки сетевого протокола и поиска узких мест в сетевой топологии при тестировании распространяемой на коммерческой основе программы удалённого управления компьютерами Remotix.
2. Отладка многопоточной визуализации данных в библиотеке NGraphics.
3. Отладка средств организации мультиплатформенного графического интерфейса пользователя при помощи библиотеки NGraphics.

Использование указанного программного продукта позволило:

1. В результате произведённых экспериментов с использованием SciVi в качестве системы визуализации повысить производительность тестируемого сетевого протокола и эффективности тестируемой топологии сети, оптимизировав тем самым работу программы Remotix.
2. В результате эксплуатации и тестирования SciVi повысить стабильность средств многопоточной визуализации в библиотеке NGraphics.
3. В результате эксплуатации и тестирования SciVi повысить стабильность средств отображения мультиплатформенного графического интерфейса пользователя в библиотеке NGraphics.

Генеральный директор ООО «Ньюлана»
Н.Б. Егоркина

МП

Председатель комиссии
В.В. Поспелов

Члены комиссии
Н.И. Репин
Е.А. Чудинов

Общество с ограниченной ответственностью
«Ньюлана»

«УТВЕРЖДАЮ»
Генеральный директор
Егоркина Наталья Борисовна
«19» февраля 2014 г.

АКТ О ВНЕДРЕНИИ

результатов диссертационной работы на соискание степени кандидата физико-математических наук Рябинина Константина Валентиновича

Комиссия в составе:

председатель: Поспелов Владислав Владиславович
члены комиссии: Репин Никита Ильич
Чудинов Евгений Александрович

составили настоящий акт о том, что алгоритм адаптивного сглаживания границ объектов на изображении и его мультиплатформенная реализация, разработанные К.В. Рябининым в рамках диссертационной работы «Методы и средства разработки адаптивных мультиплатформенных систем визуализации научных экспериментов», представленной на соискание учёной степени кандидата физико-математических наук по специальности 05.13.11, интегрированы в библиотеку визуализации NGraphics и использованы в деятельности ООО «Ньюлана» для решения задачи увеличения визуального качества изображений, генерируемых распространяемой на коммерческой основе библиотекой построения диаграмм NChart3D.

Библиотека NChart3D создаётся и поддерживается по заказу компаний Hewlett Packard, Thomson Reuters, Roche, Citi Bank, Институт генетических исследований Genomics Institute of the Novartis Research Foundation и др.

Использование указанного алгоритма позволило:

1. Устранить нежелательную ступенчатость границ объектов на изображении, генерируемом NChart3D, и тем самым увеличить визуальную привлекательность этого изображения.
2. Увеличить производительность визуализации в NChart3D.

МП

Генеральный директор ООО «Ньюлана»
Н.Б. Егоркина

Председатель комиссии
В.В. Поспелов

Члены комиссии
Н.И. Репин
Е.А. Чудинов

Приложение 2. Копии свидетельств регистрации программ для ЭВМ, созданных по материалам диссертационного исследования

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2014613796

NGraphics

Правообладатель: ***Общество с ограниченной ответственностью «Ньюлана» (RU)***

Авторы: ***Рябинин Константин Валентинович (RU), Филоненко Филипп Сергеевич (RU), Полотнянщиков Иван Сергеевич (RU)***

Заявка № **2014611639**
Дата поступления **28 февраля 2014 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***07 апреля 2014 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2014616258

NChart3D

Правообладатель: ***Общество с ограниченной ответственностью «Ньюлана» (RU)***

Авторы: ***Рябинин Константин Валентинович (RU), Чудинов Евгений Александрович (RU)***

Заявка № **2014611642**
Дата поступления **28 февраля 2014 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***18 июня 2014 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2014616257

SciVi

Правообладатель: ***Общество с ограниченной ответственностью «Ньюлана» (RU)***

Автор: ***Рябинин Константин Валентинович (RU)***

Заявка № **2014611640**

Дата поступления **28 февраля 2014 г.**

Дата государственной регистрации

в Реестре программ для ЭВМ ***18 июня 2014 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

Приложение 3. Документация по языку описания сцен

Описание сцены генерируется шаблонизатором на основе входных и выходных данных решателя. По данному описанию строится сцена и производится её визуализация. Описание производится на языке XML.

Пример XML-документа, содержащего описание сцены:

```
<scene timestamp_count="10"
background="http://myhost.com/background.png"
       initialRotX="-0.698131701" initialRotY="3.40339204">

    <model id="sun">
        <data model="sphere" texture="http://myhost.com/sphere.png"
              vertex_shader="http://myhost.com/myshader.vsh"
              fragment_shader="http://myhost.com/myshader.fsh"/>
        <position rotY="0"/>
        <shader key="u_turbulence" value="0"/>
        <animation>
            <timestamp id="9">
                <position rotY="1.57"/>
                <shader key="u_turbulence" value="1.57"/>
            </timestamp>
        </animation>
        <model id="halo0">
            <data model="billboard"
texture="http://myhost.com/billboard.png"/>
            <position scaleX="3.5" scaleY="3.5" scaleZ="3.5"/>
        </model>
    </model>

</scene>
```

Документация разметки:

- Тэг `<scene>` – агрегатор сцены.
  - Поля:
    - `timestamp_count` – количество шагов анимации.
    - `background` – hex-код цвета фона или путь к файлу фона.
    - `initialRotX` – начальный поворот всех объектов по оси X (в радианах).
    - `initialRotY` – начальный поворот всех объектов по оси Y (в радианах).

        - `initialRotZ` – начальный поворот всех объектов по оси Z (в радианах).
        - `arcball` – логический флаг, определяющий, включен ли по умолчанию режим аркбол-камеры. Допустимые значения:
            - `true` – режим аркбол-камеры включен.
            - `false` – режим аркбол-камеры выключен. Значение по умолчанию.
        - `horPan` – индекс оси, вдоль которой осуществляется перемещение сцены при горизонтальном пане. Допустимые значения:
            - `-1` – нет плоскости, равносильно отсутствию поля;
            - `0` – ось X;
            - `1` – ось Y;
            - `2` – ось Z.
        - `vertPan` – индекс оси, вдоль которой осуществляется перемещение сцены при вертикальном пане. Допустимые значения:
            - `-1` – нет плоскости, равносильно отсутствию поля;
            - `0` – ось X;
            - `1` – ось Y;
            - `2` – ось Z.
    - Потомки:
        - `<model>`.
        - `<chart>`.
- Тэг `<model>` – описание модели, агрегатор анимаций. Может содержать другие модели, которые будут является потомками данной в иерархии сцены (наследовать аффинные преобразования).
    - Поля:
        - `id` – идентификатор модели.

- Потомки:
  - <data>.
  - <pivot>.
  - <position>.
  - <shader>.
  - <animation>.
  - <model>.

- Тэг <data> – описание данных для создания модели.
  - Поля:
    - model – используемая модель. Допустимые значения:
      - sphere – сфера;
      - billboard – биллборд;
      - путь к файлу формата n3d;
      - путь к файлу формата 3ds;
      - путь к файлу формата ply (должен содержать только треугольные грани).
    - shader – используемый шейдер. Допустимые значения:
      - texturing – наложение текстуры без освещения;
      - phong – наложение текстуры с освещением по Фонгу, источник света находится в камере;
      - phong_vc – использование цвета из вершин, освещение по Фонгу, источник света находится в камере;
      - line – закраска линий без освещения.
    - vertex_shader – путь к файлу, хранящему код вершинного шейдера на языке ESSL. Допустимые параметры шейдера (размерность атрибутов может изменяться по желанию программиста, главное, чтобы были соблюдены их порядок и размерность в данных вершин

предоставляемой модели; на семантику атрибутов ограничений не накладывается; типы и семантика юниформов закреплены жёстко; возможно также введение новых юниформов, имена и значения которых описываются при помощи тэга <shader>):

- `attribute vec4 a_veretx` – координаты вершины;
- `attribute vec4 a_normal` – координаты нормали;
- `attribute vec4 a_texCoord` – координаты текстуры;
- `attribute vec4 a_globalTexCoord` – глобальные координаты текстуры;
- `attribute vec4 a_diffuse` – диффузный цвет;
- `attribute vec4 a_specular` – цвет зеркального блика;
- `attribute vec4 a_ambient` – цвет окружающей подсветки;
- `attribute vec4 a_borderDiffuse` – диффухный цвет контура;
- `attribute vec4 a_borderSpecular` – цвет зеркального блика контура;
- `attribute vec4 a_borderAmbient` – цвет окружающей подсветки контура;
- `uniform sampler2D u_map_0` – дискретизатор текстуры;
- `uniform mat4 u_mMatrix` – матрица модели;
- `uniform mat3 u_mnMatrix` – матрица нормалей от матрицы модели;
- `uniform mat4 u_mvMatrix` – произведение матриц вида и модели;
- `uniform mat3 u_nMatrix` – матрица нормалей от произведения матриц вида и модели;
- `uniform mat4 u_mvpMatrix` – произведение матриц вида, модели и проекции.

        - `fragment_shader` – путь к файлу, хранящему код фрагментного шейдера на языке ESSL. Допустимые юниформы аналогичны описанным для вершинного шейдера.
        - `polygonType` – тип полигонов. Допустимые значения:
            - `triangles` – треугольники (значение по умолчанию);
            - `lines` – линии (вершины *n* и *n* + 1 соединяются отрезком).
        - `texture` – путь к файлу с текстурой в формате png или jpeg.
- Тэг `<pivot>` – опорная точка модели.
    - Поля:
        - `x` – абсцисса (в юнитах сцены).
        - `y` – ордината (в юнитах сцены).
        - `z` – аппликата (в юнитах сцены).
        - `rotX` – поворот вокруг оси X (в радианах).
        - `rotY` – поворот вокруг оси Y (в радианах).
        - `rotZ` – поворот вокруг оси Z (в радианах).
        - `scaleX` – масштаб по оси X.
        - `scaleY` – масштаб по оси Y.
        - `scaleZ` – масштаб по оси Z.
        - `dirX` – абсцисса вектора направления (в юнитах сцены). Вектор направления альтернативен повороту и использует для вращения кватернионы вместо матриц.
        - `dirY` – ордината вектора направления (в юнитах сцены). Вектор направления альтернативен повороту и использует для вращения кватернионы вместо матриц.
        - `dirZ` – аппликата вектора направления (в юнитах сцены). Вектор направления альтернативен повороту и использует для вращения кватернионы вместо матриц.

- Тэг `<position>` – позиция модели. Может иметь собственные координаты, повотор и масштаб; откладывается от опорной точки.
  - Поля аналогичны полям опорной точки.
- Тэг `<shader>` – описание юниформа для используемого в модели шейдера.
  - Поля:
    - `key` – название юниформа.
    - `value` – значение типа float для юниформа.
- Тэг `<animation>` – агрегатор для ключевых кадров анимации.
  - Потомки:
    - `<timestamp>`.
- Тэг `<timestamp>` – описание ключевого кадра анимации. Ключевые кадры отображаются тиками на линии времени, промежуточные состояния сцены вычисляются при помощи линейной интерполяции состояний в соседних ключевых кадрах. Начальным состоянием считается состояние, которое формируется за пределами блока `<animation>`.
  - Поля:
    - `id` – номер ключевого кадра.
  - Потомки:
    - `<pivot>`.
    - `<position>`.
    - `<shader>`.
- Тэг `<chart>` – описание диаграммы, которая строится при помощи модуля NChart3D.
  - Поля:
    - `caption` – заголовок диаграммы.
  - Потомки:
    - `<axis>`.

        - <legend>.
        - <seriesArray>.
- Тэг <axis> – описание осей диаграммы.
    - Поля:
        - knid – вид оси. Допустимые значения:
            - x – ось абсцисс;
            - y – ось ординат;
            - z – ось аппликат.
        - name – заголовок оси.
        - offset – логический флаг, определяющий наличие сдвига на оси. Допустимые значения:
            - true – сдвиг на оси присутствует.
            - false – сдвиг на оси отсутствует. Значение по умолчанию.
        - beautifyMinMax – логический флаг, определяющий, нужно ли вычислять «красивые» значения минимума, максимума и шага на основании точек графика. Допустимые значения:
            - true – вычисление нужно. Значение по умолчанию.
            - false – вычисление не нужно.
    - Потомки:
        - <tick>.
- Тэг <tick> – описание тика на оси.
    - Поля:
        - name – подпись тика.
- Тэг <legend> – описание легенды диаграммы.
    - Поля:
        - visible – логический флаг, определяющий видима ли легенда. Допустимые значения:

        - `true` – легенда видима. Значение по умолчанию.
        - `false` – легенда невидима.
- Тэг `<seriesArray>` – агрегатор серий данных диаграммы.
    - Потомки:
        - `<series>`.
- Тэг `<series>` – описание серии данных диаграммы.
    - Поля:
        - `id` – идентификатор серии.
        - `type` – тип серии. Допустимые значения:
            - `candlestick` – свечи;
            - `sequence` – последовательность;
            - `phylogeneticTree` – филогенетическое дерево.
        - `positiveColor` – hex-код цвета для положительных значений.
        - `positiveBorderColor` – hex-код цвета для контура положительных значений.
        - `negativeColor` – hex-код цвета для отрицательных значений.
        - `negativeBorderColor` – hex-код цвета для контура отрицательных значений.
        - `borderThickness` – толщина контура в пикселях.
        - `color` – hex-код основного цвета.
        - `textColor` – hex-код цвета текста.
        - `lineThickness` – толщина линий в пикселях.
    - Потомки:
        - `<point>`.
- Тэг `<point>` – описание точек диаграммы.

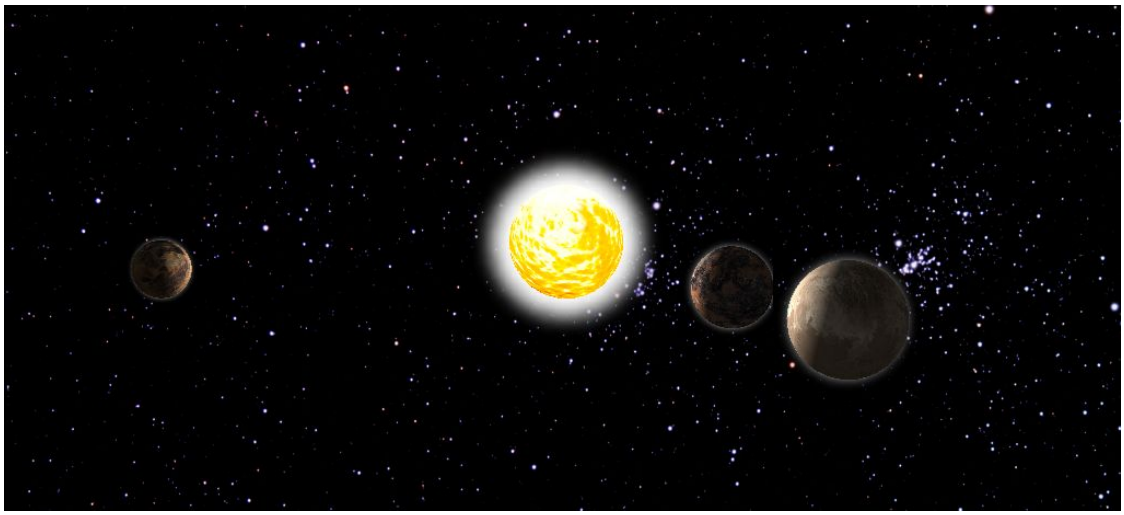
- Поля:
  - `open` – значение курса открытия.
  - `close` – значение курса закрытия.
  - `low` – минимальное значение курса.
  - `high` – максимальное значение курса.
  - `x` – абсцисса.
  - `y` – ордината.
  - `z` – аппликата.

Приложение 4. Поддерживаемые типы графических сцен

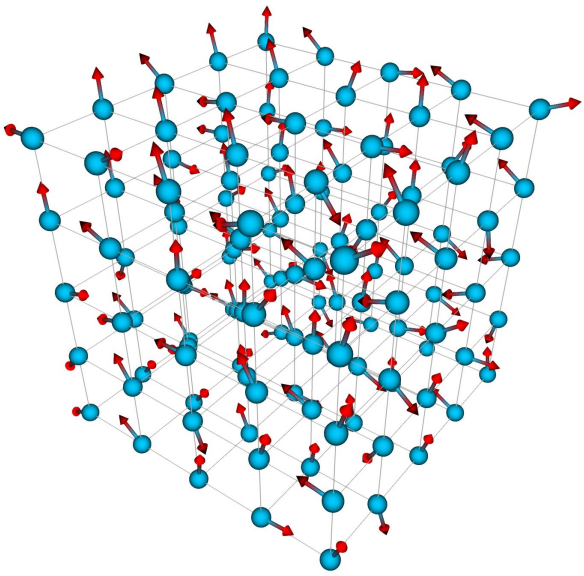
Выбор типа сцены осуществляется на этапе настройки системы научной визуализации SciVi на конкретную задачу. Поддерживаются следующие типы сцен:

1. Одиночная 3D-модель (встроенный примитив или модель, загруженная из файла).

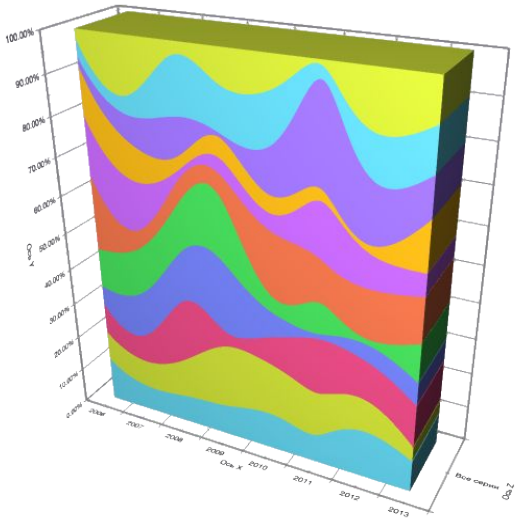
2. Множество трёхмерных моделей, иерархически связанных друг с другом (встроенных примитивов или моделей, загруженный из файлов).



3. Пространственная решётка, в узлах которой располагаются трёхмерные модели (встроенные примитивы или модели, загруженные из файлов)



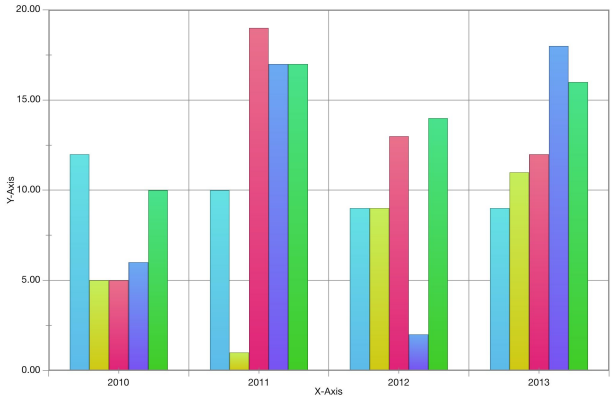
4. Диаграмма или график. Список поддерживаемых типов диаграмм и графиков представлен в приложении 5. Диаграммы и графики могут комбинироваться по несколько на одном экране.

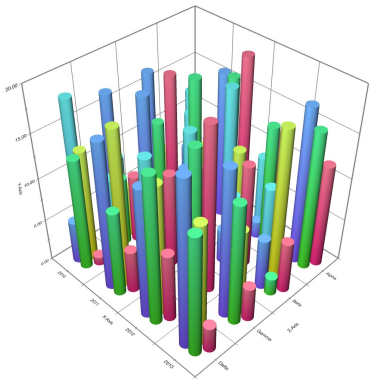


Приложение 5. Поддерживаемые типы диаграмм и графиков

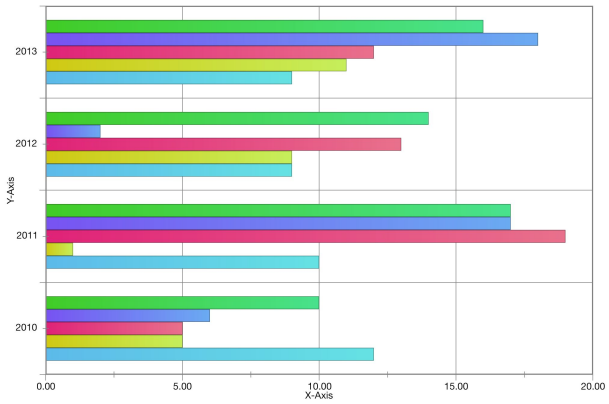
Библиотека NChart3D поддерживает следующие типы диаграмм и графиков (большинство типов доступны в двумерном и трёхмерном представлении):

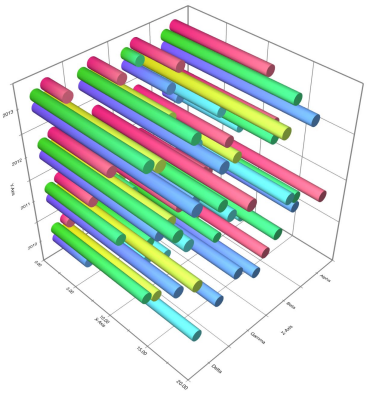
1. Вертикальные колонки (могут быть использованы для построения гистограмм):



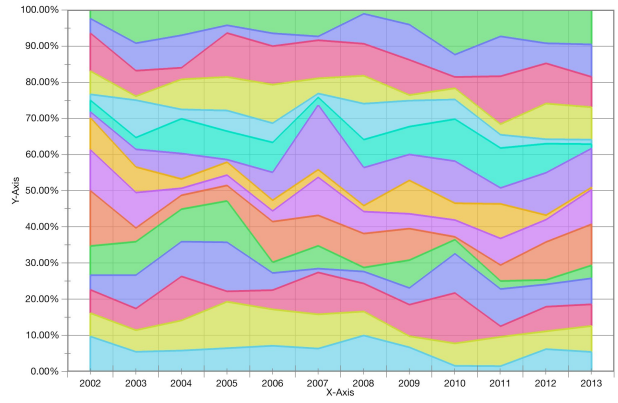


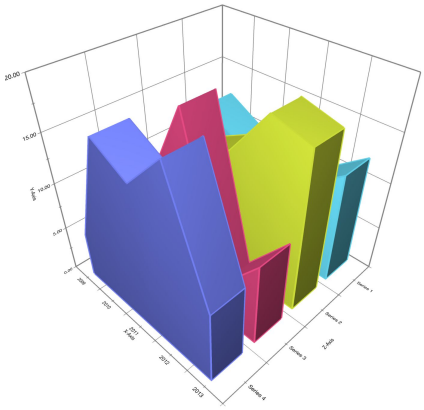
2. Горизонтальные колонки (могут быть использованы для построения горизонтальных гистограмм):



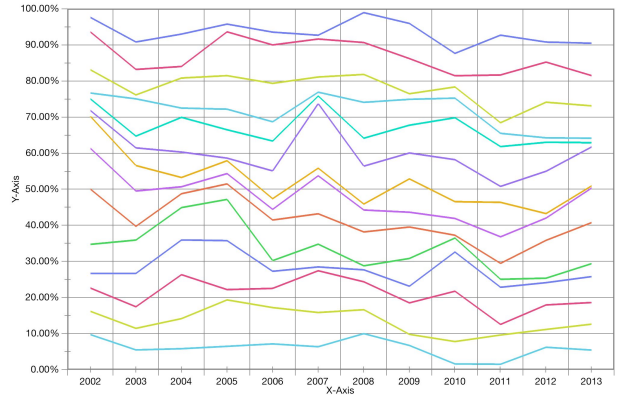


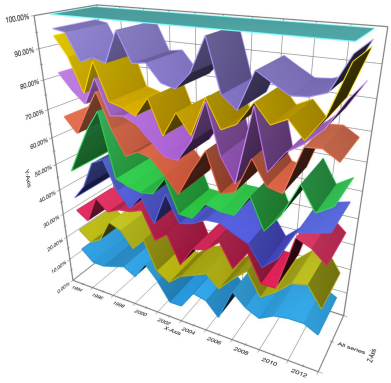
3. Диаграммы с областями (могут быть использованы для построения непрерывных гистограмм):



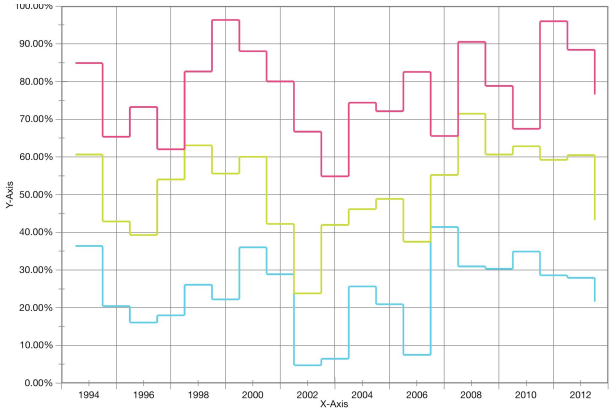


4. Линейные диаграммы (могут быть использованы для построения графиков функций):

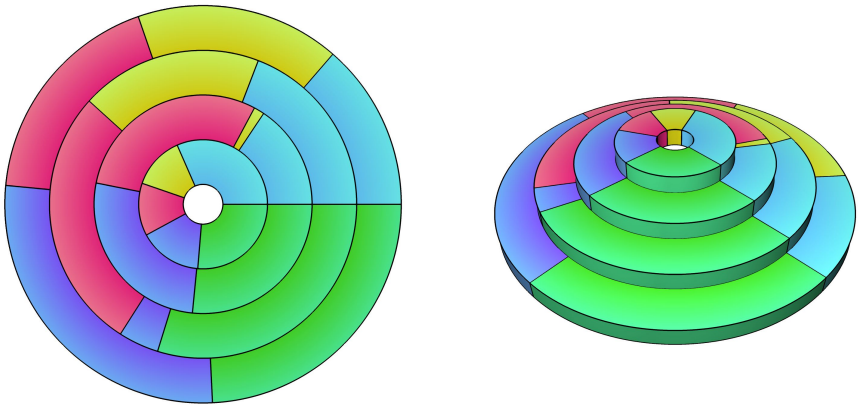




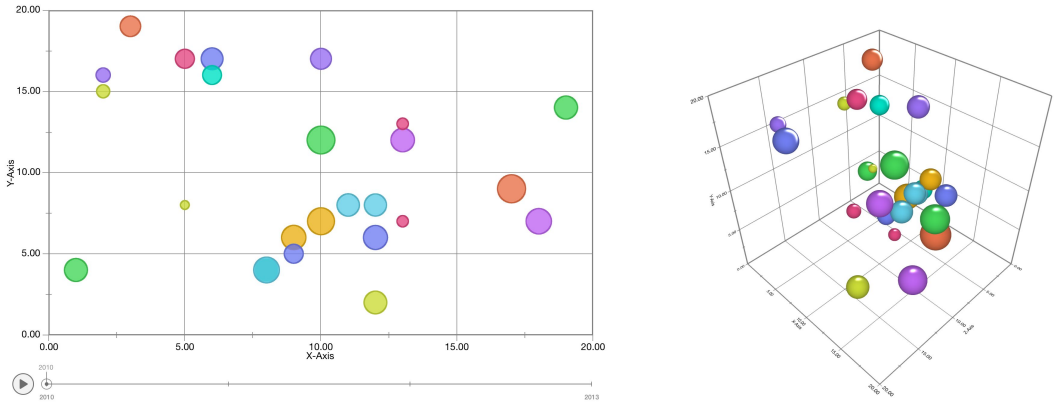
5. Ступенчатые диаграммы (могут быть использованы для построения диаграммы дискретного сигнала):



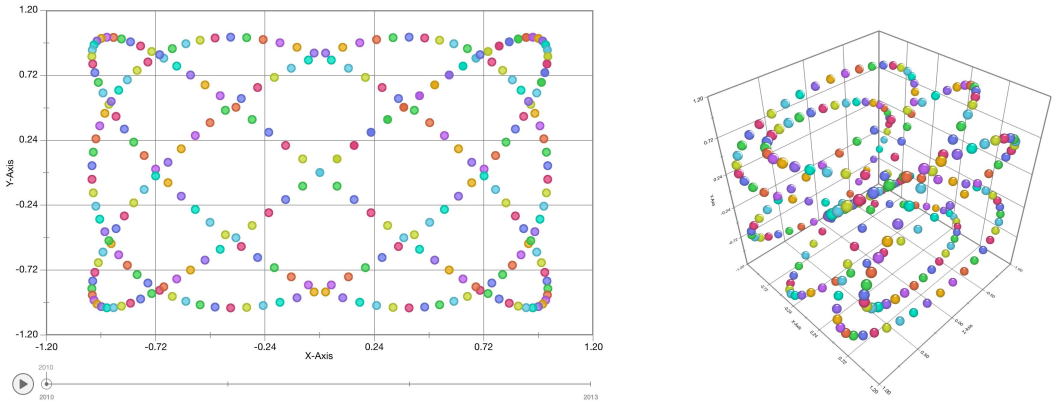
6. Круговые и кольцевые диаграммы (могут быть использованы для отражения долевого вклада показателей):



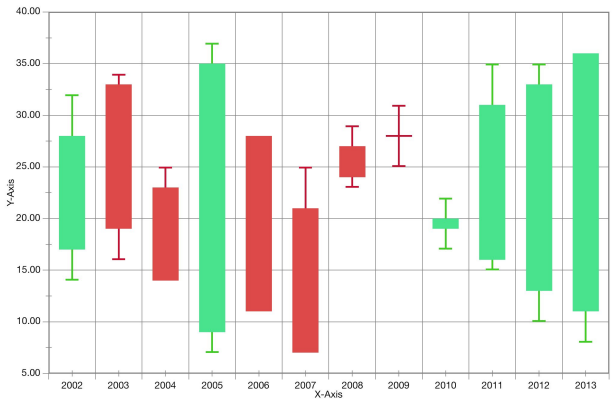
7. Пузырьковые диаграммы (могут быть использованы для отображения многомерных данных):

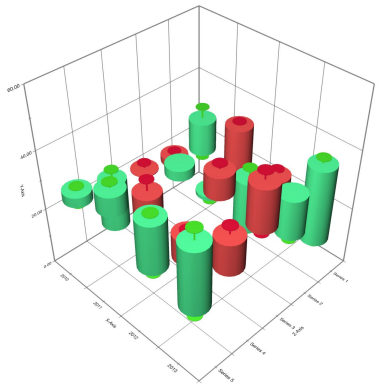


8. Диаграммы рассеивания (могут быть использованы для отображения множество точек на плоскости или в пространстве):

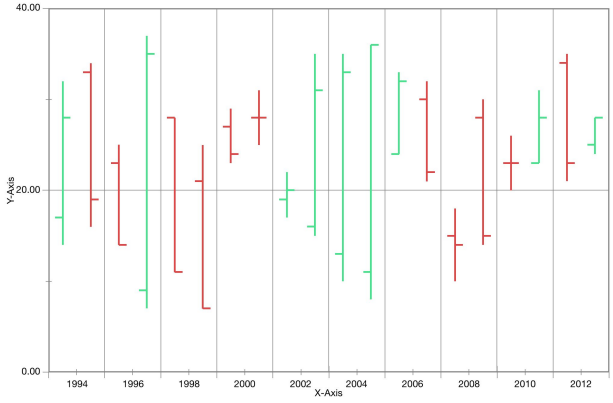


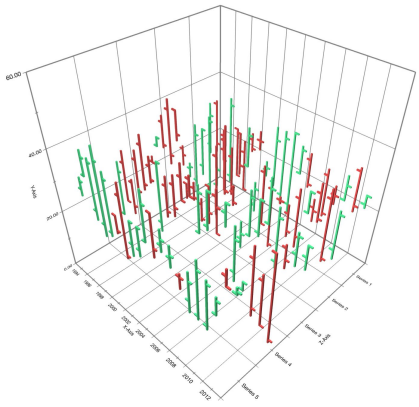
9. Японские свечи (могут быть использованы для отображения колебаний цен на рынке):



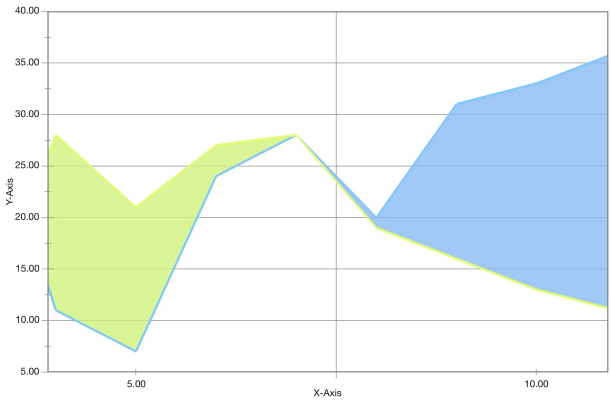


10. OHLC-диаграммы (могут быть использованы для отображения колебаний цен на рынке):

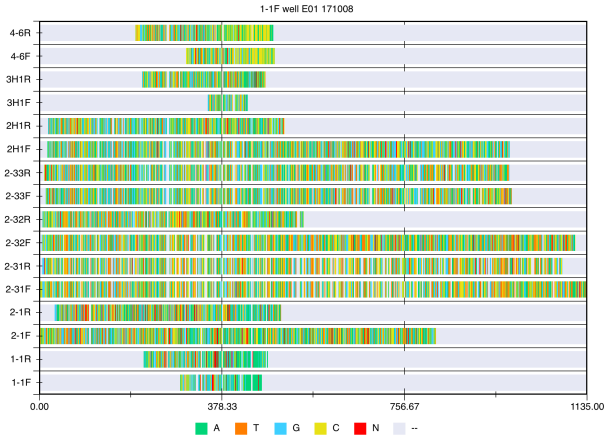




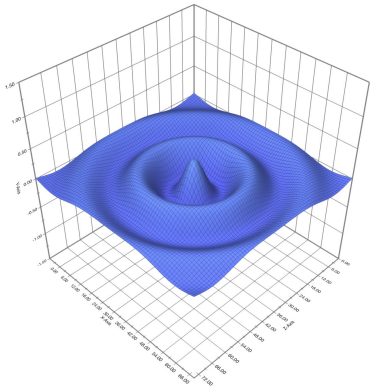
11. Полосы (могут быть использованы для отображения колебаний цен на рынке):



12. Диаграммы, отображающие последовательности элементов:



13. Поверхности (могут быть использованы для отображения поверхностей уровня):



Приложение 6. Документация типов сообщений протокола SVTP

Протокол SVTP поддерживает следующие типы сообщений:

- Общие сообщения:
  - SciViNetworkMessageTypeIncomplete – неполное сообщение, требуются дополнительные данные.
- Запросы:
  - SciViNetworkMessageTypeRequestData – сообщение запроса данных;
  - SciViNetworkMessageTypeRequestTaskStart – команда на запуск решателя;
  - SciViNetworkMessageTypeRequestTaskStatus – запрос состояния решателя;
  - SciViNetworkMessageTypeRequestTaskCancel – команда на остановку решателя.
- Ответы:
  - SciViNetworkMessageTypeReturnData – сообщение, содержащее данные (ответ на сообщение SciViNetworkMessageTypeRequestData);
  - SciViNetworkMessageTypeReturnTaskStart – сообщение, подтверждающее запуск решателя (ответ на сообщение SciViNetworkMessageTypeRequestTaskStart);
  - SciViNetworkMessageTypeReturnTaskCancel – сообщение, подтверждающее остановку решателя (ответ на сообщение SciViNetworkMessageTypeRequestTaskCancel);
  - SciViNetworkMessageTypeReturnTaskWorking – сообщение о том, что решатель продолжает работу (ответ на сообщение SciViNetworkMessageTypeRequestTaskStatus);

- SciViNetworkMessageTypeReturnTaskDone – сообщение о том, что решатель завершил работу (ответ на сообщение SciViNetworkMessageTypeRequestTaskStatus);
- SciViNetworkMessageTypeReturnTaskFailed – сообщение о том, что в процессе работы решателя возникла ошибка (ответ на сообщение SciViNetworkMessageTypeRequestTaskStatus).

Приложение 7. Документация по языку описания графического интерфейса пользователя

Для решения проблемы двойного дизайна интерфейс описывается декларативно при помощи высокоуровневых примитивов, а затем в автоматическом режиме интерпретируется и отображается библиотекой GUIBuilder. Описание производится на языке XML.

Пример XML-документа, содержащего описание интерфейса:

```
<interface solver_name="MagnetoDynamics-F">
     <mdslider id="xDim" mask="Number on X: %d" min="1" max="10"
          val="5" step="1" button_image="close" handler="handler"
          body="slider" background="#ddf2ff" mid="2"/>
     <mdslider id="yDim" mask="Number on Y: %d" min="1" max="10"
          val="5" step="1" button_image="close" handler="handler"
          body="slider" background="#c7e1f2" mid="3"/>
     <mdslider id="zDim" mask="Number on Z: %d" min="1" max="10"
          val="5" step="1" button_image="close" handler="handler"
          body="slider" background="#ddf2ff" mid="4"/>
     <mdslider id="tDim" mask="Number of timestamps: %d" min="1"
          max="300" val="300" step="1" button_image="close"
          handler="handler" body="slider" background="#c7e1f2"
          mid="1"/>
</interface>
```

Документация разметки:

- Тэг `<interface>` – агрегатор интерфейса.
  - Поля:
    - `solver_name` – название решателя.
  - Потомки:
    - `<tabbar>`.
    - `<area>`.
    - `<dialog>`.
    - `<button>`.
    - `<slider>`.
    - `<dslider>`.
    - `<mdslider>`.

        - `<list>`.
        - `<progressbar>`.
        - `<spacing>`.
- Абстрактный элемент (не имеет тэга).
    - Поля:
        - `id` – строковый идентификатор элемента, который может быть использован для получения доступа к данному элементу в коде приложения.
        - `importance` – числовой показатель важности. В том случае, если область на экране слишком мала, чтобы отобразить все присутствующие в описании элементы, элементы с меньшим значением данного параметра будут скрыты.
        - `padding` – внутренний отступ области в пикселях в формате `left,right,bottom,top`, где `left` – левый, `right` – правый, `bottom` – нижний и `top` – верхний отступы.
        - `mid` – метаинформация элемента (например, идентификатор параметра решателя, с которым ассоциирован элемент; модулем GUIBuilder игнорируется).
- Тэг `<area>` – наследник абстрактного элемента, область, которая может располагать внутри себя элементы интерфейса (в частности – другие области).
    - Поля:
        - `layout` – способ расположения элементов. Допустимые значения:
            - `stack` – элементы располагаются один над другим.
            - `queue` – элементы располагаются один за другим.
        - `align` – выравнивание элементов. Допустимые значения:
            - `center` – выравнивание по центру.

        - `left` – выравнивание по левому краю.
        - `right` – выравнивание по правому краю.
    - Потомки:
        - `<tabbar>`.
        - `<area>`.
        - `<dialog>`.
        - `<button>`.
        - `<slider>`.
        - `<dslider>`.
        - `<mdslider>`.
        - `<list>`.
        - `<progressbar>`.
        - `<spacing>`.
- Тэг `<tabbar>` – наследник абстрактного элемента, область со вкладками. Каждая вкладка может располагать внутри себя элементы интерфейса.
    - Поля:
        - `type` – тип области со вкладками. Допустимые значения:
            - `fixed` – область, зафиксированная на экране. В каждый момент времени активна ровна одна вкладка.
            - `collapsable` – область, которая автоматически скрывается, если не активна ни одна вкладка.
        - `tabbuttons` – положение вкладок в области. Допустимые значения:
            - `left` – вкладки слева, при скрытии область уезжает вправо так, чтобы от неё на экране осталась только полоска со вкладками.
            - `right` – вкладки справа, при скрытии область уезжает влево так, чтобы от неё на экране осталась только полоска со вкладками.

        - `bottom` – вкладки внизу, при скрытии область уезжает вверх так, чтобы от неё на экране осталась только полоска со вкладками.
        - `top` – вкладки вверху, при скрытии область уезжает вниз так, чтобы от неё на экране осталась только полоска со вкладками.
    - `background` – hex-код цвета фона (см. раздел «Работа с цветом»).
  - Потомки:
    - `<tab>`.

- Тэг `<tab>` – наследник `<area>`, вкладка, которая отображается в области вкладок и может располагать внутри себя элементы интерфейса.
  - Поля:
    - `image` – название изображения или путь к файлу изображения, которое должно быть использовано в качестве пиктограммы вкладки. Производится попытка загрузки набора изображений для отображения активного, неактивного и запрещённого состояний вкладки (см. раздел «Работа с изображениями»).
- Тэг `<dialog>` – наследник `<area>`, диалоговое окно, которое может быть показано на экране поверх всего остального графического интерфейса пользователя (в виде обычного или в виде модального диалога).
  - Поля:
    - `text` – заголовок диалога.
    - `text_color` – hex-код цвета текста заголовка диалога (см. раздел «Работа с цветом»).
- Тэг `<button>` – наследник абстрактного элемента, кнопка.
  - Поля:
    - `image` – название избражения, отображаемого на кнопке. Производится попытка загрузки набора изображений для отображения

обычного, нажатого, и запрещённого состояний кнопки (см. раздел «Работа с изображениями»).

- `is_checker` – флаг, определяющий, является ли кнопка переключателем. Допустимые значения:
  - `true` – кнопка является переключателем.
  - `false` – кнопка не является переключателем. Данное значение является значением по умолчанию.
- `text` – текст, отображаемый на кнопке.
- `size` – предпочтительный размер кнопки в пикселях в формате `width,height`, где `width` – ширина, `height` – высота. Значение `0` для ширины или высоты означает необходимость автоматического определения. Значение по умолчанию – `0,0`.

- Тэг `<slider>` – наследник абстрактного элемента, слайдер.
  - Поля:
    - `handler` – название изображения, используемого для ручки слайдера. Производится попытка загрузки набора изображений для отображения обычного, активного, и запрещённого состояний ручки слайдера (см. раздел «Работа с изображениями»).
    - `body` – название изображения, используемого для полосы прокрутки слайдера. Производится попытка загрузки только одного изображения (для состояния «enabled», см. раздел «Работа с изображениями»).
- Тэг `<dslider>` – наследник `<slider>`, дискретный слайдер.
  - Поля:
    - `tick_width` – толщина (в пикселях) рисок, отображающих шаги слайдера.
- Тэг `<mdslider>` – наследник `<dslider>`, дискретный слайдер с текстовой меткой, отображающей текущее значение и кнопкой сброса значения на умолчание.

- Поля:
  - `mask` – маска вывода в нотации функции printf.
  - `min` – минимальное допустимое значение.
  - `max` – максимальное допустимое значение.
  - `val` – значение по умолчанию. Должно быть в диапазоне от `min` до `max`.
  - `step` – шаг изменения значения.
  - `button_image` – название изображения для кнопки, отвечающей за сброс значения на умолчание. Производится попытка загрузки набора изображений для отображения обычного, нажатого, и запрещённого состояний кнопки (см. раздел «Работа с изображениями»).
  - `background` – hex-код цвета фона (см. раздел «Работа с цветом»).
- Тэг `<list>` – наследник `<area>`, список с прокруткой.
- Тэг `<progressbar>` – наследник абстрактного элемента, показатель процента выполнения процесса.
  - Поля:
    - `body` – название изображения, используемого для полосы показателя процента выполнения процесса. Производится попытка загрузки только одного изображения (для состояния «enabled», см. раздел «Работа с изображениями»).
- Тэг `<spacing>` – наследник абстрактного элемента, разделитель.
  - Поля:
    - `width` – размер разделителя. Допустимые значения (фактический размер определяется на основе фактической величины области, в которой располагаются разделяемые элементы):
      - `small` – небольшой разделитель.
      - `medium` – разделитель средней величины.

- `large` – крупный разделитель.

Работа с цветом:

Цвет задаётся hex-кодом в одном из возможных форматов:

- `#RRGGBB`, где `RR` – количество красного, `GG` – количество зелёного, `BB` – количество синего
- `#RRGGBBAA`, где первая часть аналогична предыдущему формату, а `AA` – значение альфа-канала.

Работа с изображениями:

По заданному названию name производится попытка загрузки изображений для неактивного, активного (в случае кнопок – нажатого) и запрещённого состояний соответствующего элемента интерфейса.

Для состояний используется следующая система именования:

- `name_enabled` – неактивное состояние.
- `name_pushed` – активное состояние.
- `name_disabled` – запрещённое состояние.

Изображение для каждого состояния state, полученного из названия name, загружается по стандартному пути поиска из файла со следующим именем:

- `state_desktop.png` – если приложение выполняется на настольном компьютере.
- `state_pad.png` – если приложение выполняется на планшетном компьютере.
- `state_phone.png` – если приложение выполняется на смартфоне.
- `state_phone_retina.png` – если приложение выполняется на смартфоне с ретина-дисплеем.

Например, если указано название `play` и приложение выполняется на настольном компьютере, для неактивного состояния элемента интерфейса будет произведена попытка загрузки файла `play_enabled_desktop.png`.

Приложение 8. Документация по формату хранения трёхмерных моделей N3D

Формат N3D – формат бинарного представления трёхмерных моделей с минимальной метаинформацией, совместимый с библиотекой визуализации NGraphics.

Структура формата:

- Строка «N3D» (3 байта) – идентификатор формата.
- Размер индекса в байтах (4 байта) – целое число типа unsigned int, определяющее размер типа индексов модели (как правило, для настольных компьютеров размер индекса равен 4, что соответствует типу индексов unsigned int, а для мобильных устройств – 2, что соответствует unsigned short).
- Метаинформация модели в виде структуры:
  - Ключ (4 байта) – число типа int, определяющее идентификатор модели.
  - Количества компонентов вершины (4 байта) – число типа unsigned int, определяющее количество значений типа float, связанных с одной вершиной модели.
  - Количество компонентов координат (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих координатам вершины модели.
  - Количество компонентов нормали (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих нормали вершины модели.
  - Количество компонентов текстурных координат (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих текстурным координатам вершины модели.

- Количество компонентов вторичных координат (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих вторичным координатам вершины модели.
- Количество компонентов вторичных текстурных координат (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих вторичным текстурным координатам вершины модели.
- Количество компонентов диффузной составляющей материала (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих диффузной составляющей материала вершины модели.
- Количество компонентов зеркальной составляющей материала (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих диффузной составляющей материала вершины модели.
- Количество компонентов окружающей подсветки (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих окружающей подсветке вершины модели.
- Количество компонентов диффузной составляющей контура (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих диффузной составляющей контура, проходящего через вершину модели.
- Количество компонентов зеркальной составляющей контура (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих зеркальной составляющей контура, проходящего через вершину модели.
- Количество компонентов окружающей подсветки контура (4 байта) – число типа unsigned int, определяющее количество значений типа float, соответствующих окружающей подсветке контура, проходящего через вершину модели.

    - Точность (4 байта) – число типа float, характеризующее точность модели в том случае, если используются уровни дателизации.
    - Флаг динамического вершинного буфера (1 байт) – флаг типа bool, определяющий, должен ли вершинный буфер для модели создаваться как динамический (актуально для моделей, данные вершин которых должны изменяться с течением времени).
    - Флаг динамического индексного буфера (1 байт) – флаг типа bool, определяющий, должен ли индексный буфер для модели создаваться как динамический (актуально для моделей, данные индексов которых должны изменяться с течением времени).
- Количество вершин (4 байта) – целое число типа unsigned int, определяющее количество вершин в модели.
- Количество индексов (4 байта) – целое число типа unsigned int, определяющее количество индексов в модели.
- Массив вершин (длина определяется на основе метаинформации и количества вершин, тип значений – float).
- Массив индексов (длина определяется на основе количества вершин, тип – на основе размера типа).