Разработка адаптивного мультиплатформенного визуализтора результатов научных расчётов для высокопроизводительных вычислительных систем

Константин Рябинин
Механико-математический факультет
Пермский государственный национальный исследовательский университет, Пермь, Россия
kostya.ryabinin@gmail.com

Аннотация

В данной статье рассматривается вопрос создания системы визуализации результатов научных расчётов. Для такой системы предлагается использовать клиент-серверную архитектуру. Серверная часть может выполняться как на настольном компьютере, так и на высокопроизводительном вычислительном комплексе. Клиентская часть может выполняться как на настольном компьютере, так и на мобильном устройстве (смартфоне или планшетном компьютере). Процесс визуализации адаптивно распределяется между клиентом и сервером так, чтобы обеспечить оптимальную нагрузку и наибольшую скорость.

Ключевые слова: *Визуализация высокопроизводительных вычислений, мобильные платформы, OpenGLES, VTK, VES.*

1. ВВЕДЕНИЕ

В настоящее время расчётные задачи, возникающие в физике, химии, биологии и других естественных науках требуют всё более сложных средств визуализации. Происходит это потому, что усложняются математические модели, используемые в этих задачах, и традиционные средства отображения результатов, такие как графики, диаграммы и таблицы, оказываются недостаточно наглядными. Наглядность же представления результатов вычислений очень важна для исследователя [13].

На сегодняшний день существует большое количество программных пакетов и библиотек, служащих для визуализации результатов научных вычислений. Однако среди них достаточно мало мультиплатформенных решений, которые могли бы работать как под управлением операционных систем для настольных компьютеров (Windows, GNU / Linux, Mac OS X и др.), так и под управлением операционных систем для мобильных устройств (iOS, Android и др .). Традиционно мобильные устройства не рассматриваются как средства, пригодные для проведения сложной визуализации, однако, с ростом их популярности и улучшением их технических характеристик, этот вопрос нуждается в пересмотре.

Кроме того, большинство систем научной визуализации не обладают достаточной интеграцией с самими вычислителями. В то же время для исследователя, который имеет дело с вычислительно-сложной задачей, решаемой на удалённом высокопроизводительном сервере, такая интеграция является очень актуальной. Интеграция позволяет сократить время получения данных и адаптации их к конкретной системе визуализации, а также предоставляет возможность управлять вычислениями и видеть их результат при помощи единого интерфейса.

В контексте использования удалённых вычислительных систем высокой производительности встаёт вопрос об эффективной пересылке данных, подлежащих визуализации, а также балансировке нагрузки клиента и сервера. К решению этого вопроса существует три подхода [12]:

1. Визуализация в полном объёме выполняется на клиенте (клиент получает от сервера да нные, подлежащие визуализации).
2. Визуализация в полном объёме выполняется на сервере (клиент получает от сервера готовую картинку).
3. Визуализация распределена между клиентом и сервером.

На сегодняшний день наиболее распространёнными являются первые два подхода. Оба они обладают своими преимуществами, однако имеют и серьёзные недостатки.

Пользуясь первым подходом, можно достичь высокой степени интерактивности визуализации. Однако при этом к клиенту и к каналу связи клиента и сервера предъявляются повышенные требования: необходимо, в общем случае, передавать большие объёмы данных по сети, а затем выполнять построение изображения на локальной машине. В том случае, если в роли локальной машины выступает, например, мобильное устройство невысокой вычислительной мощности, подключенное к сети через низкоскоростное беспроводное соединение, этот подход неприемлем.

При втором подходе вычислительная нагрузка полностью перекладывается на сервер. С одной стороны, это позволяет значительно снизить системные требования для клиента, однако с другой – при обращении к серверу нескольких клиентов сразу нагрузка н а сервер может оказаться слишком высокой. Кроме того, хотя для передачи картинки требуется, в общем случае, меньше трафика, чем для передачи данных, подлежащих визуализации, при интерактивном изменении сцены нагрузка на сеть резко возрастёт. В условиях низкоскоростного соединения организация интерактивности и воспроизведение плавной анимации могут оказаться невозможными.

Третий подход способен объединить в себе достоинства первых двух, минимизировав при этом влияние их недостатков. Основная идея заключается в том, чтобы часть визуализации выполнять на сервере, а часть – на клиенте. Средствами высокопроизводительного сервера может быть выполнена отрисовка наиболее сложных частей данных, а так же некоторое упрощение данных, передаваемых клиенту. Разделение нагрузки между клиентом и сервером может быть спланировано на основании быстродействия клиента, занятости сервера (количества других подключенных к нему клиентов) и скорости соединения.

Такой подход, однако, является наиболее сложным и допускает много различных вариантов реализации.

Целью данной работы является разработка системы визуализации результатов научных расчётов, основанной на адаптивном разделении предобработки данных и рендеринга между клиентом и сервером. Сервером для данной системы может выступать как обычный компьютер, так и высокопроизводительный массивно-параллельный вычислительный комплекс. Клиентами могут выступать

как настольные компьютеры под управлением различных операционных систем, так и мобильные устройства (смартфоны и планшетные компьютеры).

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

- анализ существующих разработок в области систем научной визуализации;
- анализ методов написания мультиплатформенных приложений;
- проектирование и реализация мультиплатформенного ядра системы визуализации (сервера и клиента);
- разработка средств автоматизации портирования графического интерфейса на различные платформы (платформы для настольных компьютеров и платформы для мобильных устройств);
- тестирование системы на реальных прикладных задачах.

2. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

Для наглядного представления научных данных исследователями традиционно используются математические программные пакеты, интегрирующие в себе функции построения изображений (такие, как MathCad, MatLab, Mathematica, Maxima и др.), либо пакеты, ориентированные исключительно на визуализацию (TechPlot, Origin, EasyPlot, IRIS Explorer, Surfer, Grapher, AMLab и др.) [11, 14]. Также в ряде случаев возможно использование систем автоматизированного проектирования (САПР).

Однако пакеты, интегрирующие в себе вычислительное ядро и визуализатор, являются недостаточно гибкими для эффективного решения с их помощью произвольных задач в высокопроизводительных вычислительных системах. Многие из них ориентированы лишь на настольные компьютеры и не могут эффективно работать на суперкомпьютерах. Кроме того, возможности визуализации в таких системах, как правило, ограничиваются построением двумерных и трёхмерных графиков.

Пакеты, ориентированные на одну только визуализацию, принимают на вход результаты вычислений в некотором формате и не предполагают автоматизированной коммуникации с программой-решателем. Поэтому пользователь должен самостоятельно осуществлять получение и адаптацию исходных данных.

Ещё более остро проблема адаптации данных стоит в случае использования САПР, так как этот класс программного обеспечения, вообще говоря, служит для решения иных задач.

Помимо этого, общей проблемой всех вышеописанных программных пакетов, является отсутствие для них мобильных версий. Существуют мобильные аналоги систем математических вычислений (PocetCAS, GraphCalc и др.), однако возможности встроенных в них визуализаторов также ограничены лишь построением графиков и поверхностей.

Детальный же анализ магазинов AppStore, Google Play Store и BlackBerry App World показал почти полное отсутствие систем научной визуализации для мобильных устройств под управлением iOS, Android и BlackBerry. Наиболее многофункциональной и производительной системой является KiviViewer [5], реализованная под iOS и Android. Эта система предоставляет возможность визуализировать относительно большие массивы геометрических данных, однако при этом не имеет прямой связи с ЭВМ, на которой производятся вычисления. Пересылку данных и адаптацию их к формату, пригодному для работы системы визуализации, должен осуществлять пользователь.

На уровне библиотек визуализации существует целый ряд различных решений, таких как OpenDX, VTK, VizIt, ScientificVR и др. Подробный обзор этих программных средств приведён в диссертационной работе О. В. Джосан [12]. Данные библиотеки предоставляют инструментарий для написания систем научной визуализации. Наиболее перспективной разработкой является библиотека VTK [2], имеющая расширение pVTK, которое добавляет функции распараллеливания процесса визуализации. Таким образом, при помощи данной библиотеки может быть осуществлено параллельное построение изображения в высокопроизводительной вычислительной системе.

На основе библиотеки pVTK разработана свободно распространяемая кроссплатформенная система научной визуализации ParaView [3], обладающая графическим интерфейсом пользователя и ориентированная на использование в массивно-параллельных 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, однако версии для мобильных устройств на данный момент не существует.

Благодаря широким функциональным возможностям, библиотека pVTK может быть использована в качестве параллельного визуализатора на стороне сервера. Однако на стороне клиента данную библиотеку возможно применить лишь в качестве вспомогательного визуализатора: ориентированность этого средства на настольные компьютеры не позволяет использовать его на мобильных платформах в готовом виде.

Также старые версии pVTK не предоставляли средств для осуществления стереоскопического рендеринга. Учитывая высокую популярность стереоскопии и всё увеличивающееся количество устройств, позволяющих демонстрировать стереоизображения пользователю (стереомониторы, шлемы виртуальной реальности и т. д.), её поддержка необходима и была добавлена в новых версиях библиотеки.

Ещё одним минусом pVTK является отсутствие средств воспроизведения сложной анимации, то есть динамики поведения визуализируемых систем [12]. В то же время наглядное представление динамики с возможностью управлять масштабом времени (интерактивно ускорять и замедлять отображаемый процесс) – очень важная функция программы визуализации для исследователя.

Ранее упомянутая система KiviViewer основана на библиотеке VES [10], которая, в свою очередь, является адаптацией библиотеки VTK для мобильных устройств. Наследуя от VTK широкие возможности в области визуализации научной графики, библиотека VES может быть использована в качестве основы визуализатора на стороне мобильного клиента.

Для визуализации трёхмерных и двумерных сцен могут быть использованы различные модули графического расширения, среди которых существует большое количество кроссплатформенных решений. Примерами таких модулей служат Unity3D, SIO2, OGRE, Irrlicht, oolong, Cocos2D/Cocos3D, libGDX и др. Все перечисленные модули способны работать на мобильных устройствах, однако они ориентированы на создание игровых приложений и не подходят для визуализации научной графики. Игровые приложения отличаются от систем научной визуализации тем, что 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 (построение сечений, проекций, представление структур в разных масштабах и разных уровнях детлизации и т. д.). Сложные визуальные эффекты в данном случае не так важны, а зачастую могут только помешать, отвлекая внимание исследователя. Таким образом, систему научной визуализации не следует основывать на модуле графического расширения, ориентированном на игровые приложения.

При создании интерактивной системы визуализации важную роль играет не только подсистема рендеринга, но и пользовательский интерфейс. Для организации кроссплатофрменности, графический интерфейс следует разрабатывать с использованием какой-либо библиотеки, реализованной под все целевые платформы.

На сегодняшний день существует относительно немного кроссплатформенных библиотек для создания графических интерфейсов пользователя. Наиболее популярными являются Qt, GTK, Tk и Awt / Swing. GTK и Tk имеют реализацию только под операционные системы для настольных компьютеров. Awt и Swing используются для создания интерфейсов Java-приложений, однако тоже только для Java-машин, работающих на настольных компьютерах. Для Qt существуют реализации под операционные системы iOS (Qt-iPhone [9]) и Android (Necessitas [7]). Однако эти реализации находятся ещё в стадии разработки и отладки, а потому могут испытывать проблемы стабильности работы.

Кроме того, используя Qt в качестве библиотеки для создания приложения, которое работало бы под управлением операционных систем для настольных компьютеров и для мобильных устройств, программист сталкивается с проблемой различного дизайна интерфейса под различные платформы. Дело в том, что мобильные устройства и настольные компьютеры используют различные парадигмы управления и отображения данных. Так, например, основными средствами контроля над 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, а значит и информация на н ём должна располагаться иным образом, чтобы обеспечить требуемую эргономику. Поэтому, даже используя какую-либо из существующих кроссплатформенных библиотек, программисту придётся создавать дизайн приложения дважды – под разные типы устройств.

Ещё один способ достижения корссплатформенности – написание Web-приложений. Наиболее популярными на сегодняшний день технологиями в этой области являются HTML 5 (совместно с использованием JavaScript и CSS) и Flash. Обе эти технологии предоставляют программисту широкие возможности, однако ввиду того, что итоговое приложение выполняется в браузере (то есть в некоторой программной прослойке), возможен целый ряд проблем с производительностью и стабильностью работы. В настоящее время поддержка этих технологий со стороны браузеров активно развивается, вводятся различные оптимизации, и, соответственно, растёт сложность самих Web-приложений. Но в контексте научной графики, где имеют место большие объёмы данных, производительности Web-приложений пока ещё не достаточно. В особенности проблема производительности встаёт на мобильных устройствах, так как их вычислительная мощность значительно ниже, чем у стационарных компьютеров. Из-за этого и поддержка Web-технологий для мобильных устройств оказывается в значительной степени сокращённой.

Несмотря на это, существует отдельный достаточно перспективный проект MoSync [6], предоставляющий программисту возможность создавать интерфейс мобильного приложения на HTML 5, а логику реализовывать на языке C++. Получаемое приложение может выполняться под управлением любой из наиболее 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 критичным является так же быстрая визуализация на уровне интерфейса. Таким образом, при использовании фреймворка MoSync (либо иных основанных на HTML 5 или Flash фреймворков), интерфейс, выполняющийся в браузере, станет узким местом. Кроме того, MoSync ориентирован только на мобильные устройства, тогда как для достижения мультиплатформенности необходима поддержка и настольных компьютеров.

Для эффективного рендеринга на мобильных устройствах под управлением таких операционных систем, как iOS и Android, используются библиотеки стандарта OpenGLES [8] – адаптации стандарта OpenGL к мобильным платформам. По своей функциональности OpenGLES предназначен для отображения трёхмерной графики, но с его помощью может быть создан и обычный двумерный интерфейс пользователя. Однако этот стандарт описывает лишь низкоуровневые функции работы с графикой, из-за чего создание пользовательского интерфейса, как и любых других сложных графических композиций, на его основе трудоёмко. Возникает необходимость разработки программных прослоек, называемых модулями графического расширения. Такие прослойки покрывают функциональность некоторой низкоуровневой графической библиотеки, предоставляя программисту высокоуровневое API. Чаще всего, модули графического расширения разрабатываются в объектно-ориентированной парадигме, так как она хорошо сочетается с терминами компьютерной графики – представление графической сцены в виде множества объектов, свойства и поведение которых объединены в некоторой иерархии классов.

Так, например, существует модуль графического расширения Clutter [1], 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 наиболее популярных платформ ограничена.

3. ПРЕДЛАГАЕМОЕ РЕШЕНИЕ

На основе проведённого теоретического исследования можно сделать два важных вывода:

1. Не было создано ещё ни одной мультиплатформенной (работающей на мобильных устройствах и на настольных компьютерах одновременно) системы научной визуализации, которая бы получила широкое распространение.
2. На сегодняшний день не существует эффективного решения задачи написания мультиплатформенного приложения, которое работало бы и на мобильных устройствах и на стационарных компьютерах.

Продолжая анализ, можно утверждать, что ситуация, описанная в выводе (1) частично обусловлена проблемой, сформулированной в выводе (2). Таким образом, создание мультиплатформенной системы научной визуализации следует начать с решения проблемы достижения мультиплатформенности вообще.

Анализ существующих технологий показал, что логика мультиплатформенного приложения может быть реализована на языке C++, так как код, написанный на этом языке, так или иначе может выполняться как на мобильных платформах iOS и Android, так и под управлением операционных систем для настольных компьютеров (Windows, GNU / Linux, Mac OS X и др.). В операционных системах для настольных компьютеров, а так же в iOS, есть возможность выполнения кода, написанного на C++, напрямую. В операционной системе Android для этого используется технология JNI [4], позволяющая вызывать методы, написанные на C++, из кода, написанного на Java, и наоборот.

Визуализация данных (трёхмерных структур, двумерных графиков и диаграмм разных типов) может быть осуществлена с использованием открытых библиотек VTK и VES. Эти библиотеки эквивалентны на уровне интерфейса, поэтому переключение между ними не потребует написания большого количества дополнительного кода. Для клиентов системы визуализации, работающих на мобильных устройствах, будет использована библиотека VES, а для настольных клиентов – библиотека VTK.

Однако при использовании этих библиотек должна быть решена проблема организации сложной анимации (для наглядного изображения динамики системы) и проблема поддержки стереоскопического рендеринга.

Сложная анимация может быть достигнута интерполяцией входных данных между некоторыми заранее выделенными состояниями (ключевыми кадрами). Такие состояния могут быть отмечены программой-решателем, которая решает исходную вычислительную задачу на стороне сервера.

Новые версии VTK поддерживают стереоскопическую визулизацию в разных форматах, однако следует решить вопрос совместимости с разными типами устройств вывода графической информации (стереомониторы, шлемы виртуальной реальности и т. д.).

В качестве решения проблемы унифицированного создания графического интерфейса пользователя был разработан модуль графического расширения, основанный на стандарте OpenGLES. Так как OpenGLES является подмножеством OpenGL, модуль работоспособен под управлением операционных систем как для мобильных устройств, так и для настольных компьютеров.

Разработанный модуль предоставляет возможность быстро создавать элементы двумерного и трёхмерного интерфейса (использовать пространственную анимацию и объёмные модели элементов интерфейса), предоставляя программисту удобную объектную модель, полностью скрывающую вызовы функций библиотеки стандарта OpenGLES. Языком реализации модуля является C++. Модуль был протестирован на операционных системах iOS, Android, Windows, GNU / Linux и Mac OS X.

Однако сам по себе фреймворк создания произвольных интерфейсов, как это было показано ранее, ещё не обеспечивает лёгкого достижения мультиплатформенности. Проблема заключается в необходимости создания разного дизайна приложений для мобильных устройств и настольных компьютеров.

Для решения этой проблемы предлагается разработать язык декларативного описания интерфейсов, который позволил бы описать лишь наиболее общие черты интерфейса. Затем, интерпретатор этого языка создал бы интерфейс, отвечающий декларированным чертам, но с учётом особенностей текущей платформы. Идейно такой язык схож с языком XAML от компании Microsoft, однако он должен предоставлять более общие описания, чтобы по одной и той же декларации мог быть построен интерфейс как для мобильной, так и для настольной версии приложения.

Системой научной визуализации должны поддерживаться многомасштабность по размеру отображаемой структуры и по времени протекания в этой структуре изучаемых процессов.

Для достижения многомасштабности по размеру, предлагается использовать метафору микроскопа. В программе выделяются два способа изменения масштаба сцены – количественный и качественный.

Количественное масштабирование непрерывно и позволяет увеличивать или уменьшать текущую структуру в определённых пределах. Такое масштабирование метафорически соответствует подстройке фокуса у микроскопа. При этом с изменением коэффициента увеличения меняется уровень детализации сцены: когда части структуры становятся меньше определённого порогового значения, их детализация уменьшается. Оказавшиеся за пределами усечённой пирамиды видимости фрагменты структуры временно исключаются со сцены и не поступают на графический конвейер. Таким образом достигается оптимальная нагрузка на центральный и графический процессоры и повышается производительность.

Качественное масштабирование изменяется дискретно и приводит, вообще говоря, к полной перестройке сцены. Переход от одного масштаба к другому метафорически соответствует смене линзы у микроскопа. Разные уровни масштаба должны иметь поддержку также на стороне программы-решателя, так как именно от неё система визуализации получает данные о структуре отображаемого объекта. Не все программы-решатели обязаны поддерживать многомасштабность обсчитываемой структуры, однако в том случае, если такая поддержка имеется, для исследователя очень актуально иметь возможность быстро переключаться между масштабами в системе визуализации.

Многомасштабность по времени позволяет динамически настраивать скорость воспроизведения анимации структуры. Разные процессы, которые могут быть интересны для исследователя, могут иметь самую разную реальную скорость протекания – от долей секунды (если, например, речь идёт о моделировании квантовых взаимодействий) до миллионов лет (если речь идёт о моделировании планетарной системы). Более того, в одной системе могут протекать сразу несколько процессов с разной, возможно, переменной, скоростью. Для исследователя актуально иметь возможность быстро, во время работы визуализатора, изменять скорость воспроизведения анимации того или иного процесса, ставить воспроизведение на паузу или возобновлять, а также быстро проматывать анимацию к некоторой временной позиции. В связи с этим на интерфейс системы должна быть вынесена линия времени, предоставляющая возможность перемотки и настройки скорости воспроизведения анимации.

Следующей важной задачей является управление программой-решателем непосредственно с интерфейса системы визуализации. Сложность добавления такой функциональности заключается в том, что для каждой конкретной задачи, вообще говоря, существует свой

решатель, имеющий свой набор команд управления. Очень часто под конкретную научную вычислительную задачу пишется отдельная программа, не имеющая почти никакого собственного интерфейса, и лишь принимающая на вход некоторые исходные данные. В этом случае система визуализации должна позволять формировать необходимые исходные данные и давать команду на запуск либо остановку расчётов.

Для каждого конкретного решателя предлагается создавать описание интерфейса управления (формы ввода данных и, если это необходимо, дополнительных команд, которые могут быть переданы решателю прямо в процессе счёта). Описание осуществляется на том же декларативном языке, на котором создаётся интерфейс всей системы визуализации. Далее файлы с описаниями интерфейсов различных решателей, а так же с необходимой мета-информацией (IP-адрес сервера, на котором находится решатель, параметры подключения и т. д.) размещаются на некотором Web-хостинге, откуда по запросу скачиваются системой визуализации. После того, как файл с описанием интерфейса решателя скачан, система может построить форму управления и отдать решателю команду начать вычисления по данным, которые ввёл в эту форму пользователь.

В качестве посредника между клиентской частью системы визуализации и программой-решателем выступает серверная часть системы. Для того, чтобы сервер мог транслировать команды от клиента решателю, последний должен поддерживать некоторый стандартный интерфейс управления. В качестве такого интерфейса предлагается использовать подход передачи управляющих команд в виде ключей, с которыми запускается решатель. Такой подход позволяет сделать решатель максимально изолированным от системы, и не принуждает создателей решателя загромождать его код какими-либо механизмами взаимодействия. В этом случае, серверу также необходимо иметь описание команд решателя, ставящее в соответствие запросы, приходящие от клиента, ключам решателя. Такое описание легко может быть составлено вручную на языке XML.

Более сложным является управление решателем в процессе его работы. В том случае, если это необходимо, наиболее простым вариантом будет использование механизма межпроцессного взаимодействия через сигналы. Такой подход также не потребует от создателей решателя большого количества дополнительного кода.

По мере вычисления, решатель формирует некоторые выходные данные, которые должны быть визуализированы. Предлагается осуществить разделение процесса визуализации между клиентом и сервером. Для визуализации на стороне сервера предлагается использовать библиотеку pVTK (параллельный вариант библиотеки VTK). При помощи этой библиотеки можно осуществить эффективный рендеринг на высокопроизводительной вычислительной системе.

Разделение рендеринга должно осуществляться эвристически, на основе данных о вычислительной мощности клиента, скорости соединения и загруженности сервера. Кроме того, во внимание необходимо принимать характер визуализируемых данных. На стороне сервера имеет смысл осуществлять визуализацию в первую очередь тех данных, которые не будут изменяться в процессе интерактивного взаимодействия пользователя с графической сценой на клиенте. Например, если итогом визуализации должна являться некоторая трёхмерная модель с наложенной на неё текстурой, причём текстура также представляет собой результат визуализации, то эту текстуру имеет смысл подготовить на сервере.

Задача разделения рендеринга полностью ложится на сервер системы визуализации. Решив её , он выполняет свою часть визуализации и отправляет часть «сырых» (подлежащих визуализации) данных вместе с результатом своей визуализации клиенту.

«Сырые» данные в общем случае так же должны проходить некоторую предобработку на сервере. Как правило, объём их слишком велик, и передавать их по сети слишком затратно, равно как и обрабатывать на стороне клиента. Поэтому в обязанности сервера входит выполнение упрощения набора данных, полученных от программы-решателя. Коэффициент упрощения также должен определяться эвристически, исходя из вычислительной мощности клиента и скорости соединения.

Настройку качества визуализации необходимо вынести на графический интерфейс клиента, чтобы пользователь мог, при желании, полностью отключить упрощение данных, так как в ряде случаев из-за упрощения могут теряться существенные для исследователя детали.

4. ЗАКЛЮЧЕНИЕ

На данный момент полностью реализованным является фреймворк для создания интерфейсов, базирующийся на стандарте OpenGLES (и совместимый с со стандартом OpenGL). Этот фреймворк позволяет создавать все базовые элементы графического интерфейса пользователя и был протестирован под управлением операционных систем iOS, Android, Windows, GNU / Linux и Mac OS X. Остальные части системы находятся на этапе проектирования или реализованы частично.

Также изучена возможность использования библиотек VTK, pVTK и VES для научной визуализации.

В качестве прикладной задачи для отладки системы планируется рассмотреть расчёт затухания звуковой волны в шумоподавляющем пористом материале. Для решения данной задачи уже имеется математическая модель и программа-решатель [15].

5. ССЫЛКИ

[1] *Clutter project. http://www.clutter-project.org/*

[2] James Ahrens, Charles Law,Will Schroeder, Ken Martin, Michael Papka. *Parallel processing with VTK. Los Alamos National Laboratory - Technical Report#LAUR-00-1620. 2000.*

[3] James Ahrens, Berk Geveci, Charles Law. *ParaView: An End-User Tool for Large Data Visualization. Edited by C.D. Hansen, C.R. Johnson. Elsevier. 2005.*

[4] *Java Native Interface. http://java.sun.com/docs/books/jni/.*

[5] *KiwiViewer project. http://www.kiwiviewer.org/.*

[6] *MoSync project. http://www.mosync.com/what-is-mosync.*

[7] *Necessitas project. http://sourceforge.net/p/necessitas/home/necessitas/.*

[8] *OpenGLES. http://www.khronos.org/opengles/.*

[9] *Qt-iPhone project. http://www.qt-iphone.com/Introduction.html.*

[10] *VES library. http://www.vtk.org/Wiki/VES.*

[11] Максим Виноградов. *Современные средства визуализации и обработки двумерных научных данных. Московский государственный технический университет имени Н.Э. Баумана. 2002. http://www.amlab.ru/paper_max.shtml.*

[12] Оксана Джосан. *Исследование и разработка методов и программных средств визуализации результатов научных вычислений для массивно-параллельных вычислительных систем. Московский государственный университет имени М.В. Ломоносова. 2009.*

[13] *Научная визуализация. Virtual Environment Group. http://www.ve-group.ru/vr13_127.html.*

[14] Е. Л. Карташева, Г .А. Багдасаров, А.С. Болдарев, И.В.Гасилова, С. В. Дьяченко, О .Г. Ольховская, В. А . Шмыров, С.Н. Болдырев, В.А. Гасилов. *Визуализация данных вычислительных экспериментов в области 3D моделирования излучающей плазмы, выполняемых на многопроцессорных вычислительных системах с помощью пакета Marple. Институт математического моделирования РАН. http://sv-journal.com/2010-1/01/index.html.*

[15] Александр Синер. *Методика выбора звукопоглощающих конструкций для турбомашин на основе математического моделирования. Пермский государственный университет. 2010.*

Об авторах

Константин Рябинин – аспирант кафедры математического обеспечения вычислительных систем ПГНИУ.
Его адрес: kostya.ryabinin@gmail.com.