МФТИ
Кафедра системного программирования

УДК 681.3

Составитель Семенов В.А.
Рецензент
Доктор физико-математических наук, профессор
А.Н. Томилин

Открытая система для математического моделирования и научной визуализации / Учебно-методическое пособие /…

В пособии изложены вопросы разработки программных приложений математического моделирования и научной визуализации на основе системы с открытой модульной архитектурой. Подробно рассматриваются объектно-ориентированная архитектура системы, ее компонентный состав, связанная с ней технология программной инженерии, а также примеры разработанных на ее основе специализированных приложений моделирования и визуализации для решения ряда актуальных научно-технических задач.

Пособие предназначено для студентов старших курсов, аспирантов и преподавателей. Может быть использовано при выполнении практических задач и работ по курсу “Научная визуализация”.

ОТКРЫТАЯ СИСТЕМА ДЛЯ МАТЕМАТИЧЕСКОГО МОДЕЛИРОВАНИЯ И НАУЧНОЙ ВИЗУАЛИЗАЦИИ

ВВЕДЕНИЕ

В настоящее время компьютерные системы математического моделирования и визуализации научных расчетов получили широкое распространение. Создание таких систем, сочетающих широкую предметную функциональность с развитыми вычислительными графическими интерактивными средствами, сопряжено с решением целого круга проблем системного характера и обычно требует высоких затрат. Среди них следует особо выделить проблемы обеспечения открытости, мобильности, интероперабельности, масштабируемости в составе параллельных вычислительных комплексов, обеспечение возможностей коллективной работы в составе распределенных систем, организации дружественных графических пользовательских интерфейсов.

Разнообразие предметных областей науки и техники, в которых использование систем моделирования и визуализации является необходимым в силу невозможности или неэффективности проведения натурных экспериментов, а также высокая трудоемкость создания специализированных систем, в том числе и для проведения междисциплинарных исследований, обуславливают необходимость выработки общих методологических и технологических решений для их построения. В этом случае проблема создания новых приложений переводится в плоскость повторного использования уже имеющихся программных и инструментальных решений и их возможной адаптации к новым приложениям. Данные идеи востребованы и в разных формах реализованы в коммерческих и свободно распространяемых системах IRIS Explorer, IBM Data Explorer, AVS, Vis5D [1-4].

В настоящем пособии рассматриваются архитектура, принципы организации и функционирования разработанной в ИСП РАН открытой системы для математического моделирования и научной визуализации **OpenModeler&Visualizer** [5–8]. Поскольку сама система и связанная с ней оригинальная компонентная технология построения широкого класса программных приложений претендуют на существенную общность, эти вопросы приобретают важное учебно-методическое значение.

Главное внимание при этом уделяется аспектам открытости, обеспечивающим построение функционально развитых приложений моделирования и визуализации на единой концептуальной, методологической, программной и инструментальной основе.

КОНЦЕПЦИЯ

Что такое визуализация? Ответ на этот вопрос довольно нетривиален. Имеется множество частных примеров «правильной» визуализации, но до сих пор нет методического понимания, что же из того, что составляет хорошую визуализацию, являлось бы привлекательным для естественных моделей, человеческого восприятия и внутреннего зрения. Множество людей определяли понятие визуализации различными способами.

Рене Декарт (1637): «Воображение или визуализация, и, в частности, использование диаграмм, играет значительную роль в научных исследованиях».

Александр фон Гумбольд (1811): «Все, что относится к мере и количеству, может быть представлено геометрическими образами. Статистические проекты, которые обращаются к органам чувств, не утомляя разум, обладают преимуществом привлечения внимания к большему числу важных фактов».

Маккормик и др. (1987): «Визуализация — это вычислительный метод. Она преобразует символические образы в геометрические, позволяя исследователям наблюдать их

моделирование и вычисление. Визуализация предлагает метод для наблюдения невидимого. Она обогащает процесс научного открытия и способствует глубоким и непредвиденным проникновениям в суть вещей. … Много лет тому назад Ричард Хамминг заметил, что: «цель вычислений — это понимание, а не числа». Цель визуализации заключается в усилении существующих научных методов путем обеспечения нового научного понимания через визуальные методы».

Роговиц (1993): «Визуализация — это процесс отображения численных значений в воспринимаемое познаваемое пространство».

Кроме исторических проблем эволюции знаний, причина неоднозначности взглядов заключается в разнообразии исследуемых задач, существенных различиях моделей, подлежащих визуализации, и широком наборе методов и техник, которые бы потенциально подходили для достижения подобных целей. Тем не менее, интересно заметить, что все эти определения сводятся к идее трансформирования или отображения данных в некоторые воспринимаемые представления, допускающие четкую зрительную интерпретацию. Следовательно, отображение данных в зрительно связанные образы может рассматриваться как основополагающий принцип визуализации в целом и ее многочисленных приложений в частности.

Действительно, традиционный конвейер визуализации представим композицией отображений, соответствующих стадиям анализа прикладной задачи, предобработки прикладных данных, их визуализации, моделирования составленных визуальных сцен, и, наконец, растеризации сцен и генерации изображений. Здесь под прикладными данными понимается произвольная структурированная типизированная информация, используемая в различных научных и инженерных дисциплинах. На первой стадии конвейера проводится анализ прикладной задачи, возможно, осуществляемый на основе численного моделирования изучаемого явления или системы. На следующей стадии полученные данные про-

ходят дополнительную обработку для исключения возможной избыточности, а также для их улучшения. Стадия визуализации соответствует отображению данных из одной символической модели в другую, имеющую явное аудиовизуальное содержание, такое как пространство, время, форма, звук, и т.д. На следующей стадии полученное представление может быть дополнительно преобразовано в сцены моделирования виртуальной реальности, непосредственно поддерживаемые популярными графическими системами. На заключительной стадии происходит растеризация моделируемых сцен и генерация цифровых изображений, которые и являются конечной целью процесса визуализации.

Таким образом, все стадии могут рассматриваться как отображения одних данных в другие. Под более пристальным взглядом каждая из этих стадий может быть также представлена как композиция отображений, выполняющих частичные преобразования. Каждое отображение реализует некоторое правило, согласно которому типизированные элементы данных, принадлежащие одной информационной модели, должны быть трансформированы в соответствующие элементы другой модели. Из-за сложности и масштабности разработанных в последнее время многопрофильных информационных моделей STEP, MDA, охватывающих ключевые отрасли науки и промышленности, программирование приложений оказывается трудной задачей. Чтобы визуализировать прикладные данные эффективно и адекватно выбранным аспектам изучаемой проблемы, методы и сценарии визуализации должны опираться на подходящую визуальную метафору. Очевидно, что реализующие ее программы также сложны и нуждаются в более строгой структуризации с использованием некоторого общего формализма, содержательного для рассматриваемого класса программных систем моделирования и визуализации.

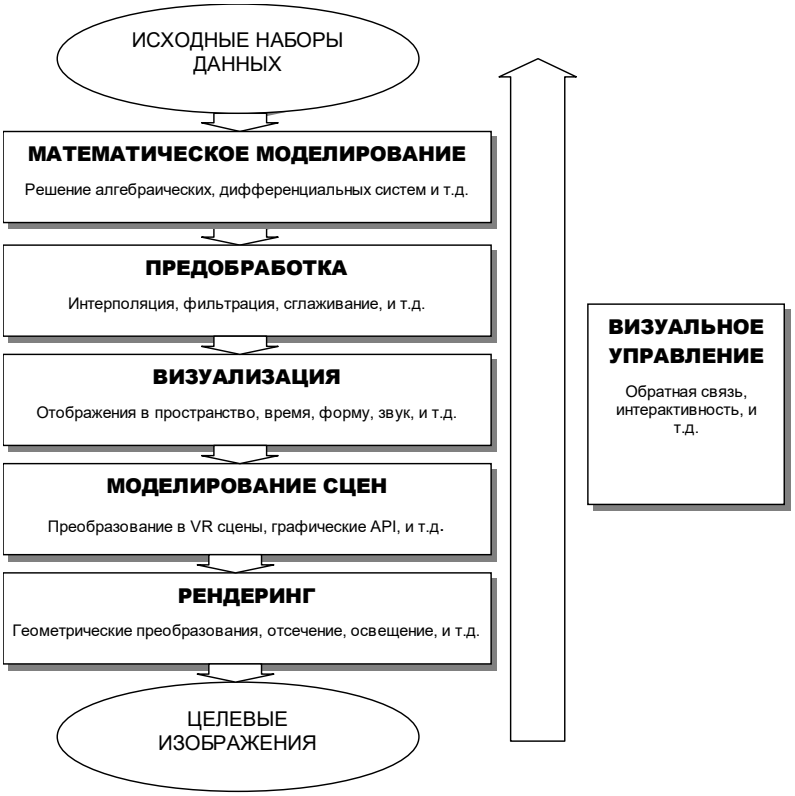

Рис. 1. Представление традиционного конвейера моделирования и визуализации как композиции отображений

Другая мотивация выработки формализма и следования ему при разработке программных приложений связана с требованием согласованного управления прикладными данными и их визуальными представлениями, существенным для интерактивных систем. В таких приложениях изменения визуальных сцен, вызванные реакциями пользователя, должны передаваться через все стадии конвейера визуализации в обратном порядке, чтобы привести прикладные данные в соответствие с их визуальным образом. Для достижения такой

цели могут применяться композиции обратных преобразований.

Например, при проектировании здания архитектор манипулирует визуальными примитивами, такими как ломаные, поверхности, твердые тела, цветовые палитры, непосредственно отображаемыми в графическом контексте приложения. Но эти манипуляции интерпретируются и обрабатываются как операции над прикладными данными, представляющими стены, окна, перекрытия, крыши и так далее, значимыми в этой прикладной области. Подобным образом, при изучении и исследовании магнитных, электрических, гравитационных полей посредством манипулирования визуальными сценами, геофизик фактически восстанавливает прикладные данные, соответствующие значимым в геологии свойствам горизонтов, скважин, областей и т.д.

Упомянутые выше обстоятельства побуждают взглянуть на дисциплины моделирования и визуализации через призму парадигмы композиции отображений, для того чтобы выработать формальный подход к разработке программных приложений и предложить конструктивные способы его реализации в рамках открытой системы.

АРХИТЕКТУРА И ПРИНЦИПЫ ОРГАНИЗАЦИИ СИСТЕМЫ

Общие требования

Система **OpenModeler&Visualizer** разработана, исходя из следующих общих требований, предъявляемых к способам построения и функционирования приложений с открытой модульной архитектурой:

* наличие унифицированного программного ядра, реализующего содержательную для рассматриваемой предметной области парадигму и инвариантного по отношению к широкому классу приложений моделирования и визуализации;

* компонентная организация, предполагающая возможность построения приложения с заданной функциональностью из отдельного набора прикладных компонентов;

* наличие развитых библиотек компонентов для поддержки стандартных типов прикладных данных и распространенных методов геометрического моделирования и научной визуализации;

* предусмотренные механизмы обобщенной реализации новых компонентов на основе разработанных ранее с использованием современных языковых, программных, графических, информационных, коммуникационных стандартов и соответствующих средств реализации;

* интерактивные возможности составления сложных визуальных сцен и поддержки альтернативных сценариев функционирования приложений на основе парадигмы композиции отображений.

Использование объектно-ориентированного подхода для реализации перечисленных принципов имеет ключевое значение, поскольку предусматриваемые им механизмы инкапсуляции, наследования и полиморфизма могут быть конструктивно применены для работы с прикладными компонентами при полной виртуализации их функциональной семантики.

Будем рассматривать сценарий функционирования приложения как упорядоченное множество типизированных ассоциативно связанных данных, участвующих в конвейере моделирования и визуализации, и алгоритмов, реализующих упомянутые выше стадии конвейера в результате преобразования данных. Будем различать пассивные объекты-данные, которые контролируют только собственное поведение, и активные объекты-алгоритмы, которые могут управлять поведением других объектов в результате активации их методов в классическом объектно-ориентированном стиле. Подобный подход следует методологии Бэйлина, известной как объектно-ориентированная спецификация требований, и представляется плодотворным для реализации открытой системы.

Объектно-ориентированное ядро

Объектно-ориентированное ядро открытой системы состоит из четырех основных классов *Entity*, *Data*, *Algorithm*, и *Scenario*, соответствующих понятиям прикладного объекта, данного, алгоритма и сценария приложения.

Для объектов класса *Entity* определены методы идентификации, методы интроспекции, сериализации и экстернализации, а также операции жизненного цикла: конструирование, копирование и удаление. В качестве атрибутов каждый экземпляр *Entity* может иметь ассоциативные связи с другими объектами. Тип связи $LinkedEntity \subseteq Entity$ определяет потенциальную возможность соединения объекта $object \in Entity$ с любым другим объектом соответствующего типа $linked\_object \in LinkedEntity$. Благодаря связям объекты приложения могут взаимодействовать. В частности, активируя методы ассоциируемых объектов, прикладной объект может изменить их состояния. Для различия способов взаимодействия все связи подразделяются на входные, выходные и смешанные. Через входные связи прикладной объект может узнать о состоянии ассоциируемых с ним объектов, не изменяя его. Используя выходные связи, прикладной объект реконструирует ассоциируемые объекты заново. Через смешанные связи он может изменить состояние уже существующих объектов. Информация о типе связей доступна через методы интроспекции прикладного объекта и используется соответствующими методами установления ассоциативных отношений между объектами.

Специализациями класса прикладного объект являются классы *Data*, *Algorithm* и *Scenario*, которые наследуют его свойства и поведение.

Класс $Data \subseteq Entity$ выражает сущность разнообразных прикладных данных, используемых приложениями моделирования и визуализации. Класс $Algorithm \subseteq Entity$ представляет методы, алгоритмы, преобразования, операции и вспомогательные утилиты, реализующие различные стадии кон-

вейера моделирования и визуализации. В отличие от пассивных объектов-данных алгоритмы являются активными объектами, контролирующими поведение связанных с ними объектов через соответствующие связи. Для этой цели в классе *Algorithm* предусмотрен специальный метод активации.

На рисунке 2 представлен пример взаимодействия объектов, иллюстрирующий каким образом данные и алгоритмы могут связываться и взаимодействовать через соответствующие типизированные связи в рамках некоторого сценария. Приведенная иерархия наследования классов определяет отношения обобщения/специализации между типами данных и алгоритмов ядра. В применяемой графической нотации сценария экземпляры данных и алгоритмов показаны, соответственно, как овалы и прямоугольники. Связи объектов промаркированы точками. Ассоциации показаны стрелками.

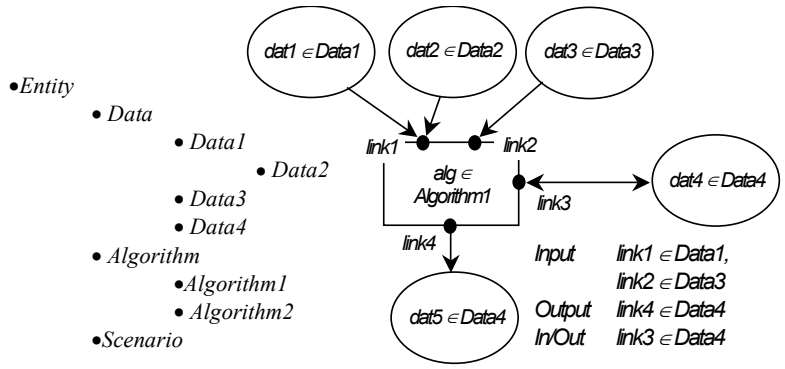

Рис. 2. Иерархия наследования классов ядра и диаграмма сценария приложения

Согласно диаграмме сценария, данные *dat1*, *dat2*, *dat3* связаны с алгоритмом *alg* через его входы. Множественная связь *link1* типа *Data1* при этом обеспечивает установление ассоциативных отношений с несколькими экземплярами данных *dat1*, *dat2*, удовлетворяющими ее типу. Объект *dat3* участвует в простой ассоциации через *link2* соответствующе-

го типа *Data3*. Аналогичным образом объекты *dat4*, *dat5* связаны с алгоритмом *alg* через смешанное соединение *link3* и выход *link4*. Согласно сценарию при активации алгоритма *alg* он вызывает соответствующие методы связанных с ним объектов *dat1*, *dat2*, *dat3*, *dat4*, приводя к конструированию выходного объекта *dat5* и обновлению состояния *dat4*.

Отметим, что применяемый способ ориентации стрелок соответствует потоку данных в сценарии. Это обстоятельство выявляет определенную схожесть с диаграммой потоков данных. Однако диаграмма сценария выражает более общую парадигму "композиции отображений", которая, на наш взгляд, является более предпочтительной для систем моделирования и визуализации общего назначения вследствие возможностей представления более сложных типов взаимодействия прикладных объектов. Приложения, обсуждаемые в пособии, дают подобные примеры сценариев.

Класс $Scenario \subseteq Entity$ реализует концепцию сценария приложения. Агрегируя списочный контейнерный класс, он поддерживает упорядоченное представление прикладных объектов и предоставляет всю необходимую функциональность приложения, которая предполагает предварительное составление (или загрузку) рабочего сценария и его последующую интерпретацию. Сценарий создается путем непосредственного манипулирования экземплярами данных и алгоритмов в результате их конструирования, редактирования значений атрибутов, установления ассоциативных отношений между ними и т.п. Интерпретация сценария осуществляется путем последовательной активации алгоритмов, приводящей к исполнению соответствующих этапов конвейера моделирования и визуализации и генерации итоговой графической сцены.

Имея описанное представление сценария, можно редактировать его вплоть до получения желаемого изображения. То же самое представление можно использовать для моделирования динамических процессов и анимации полученных результатов. Эта возможность поддерживается ядром,

поскольку однажды созданный сценарий может снова и снова использоваться для эквивалентных наборов данных, сгенерированных или импортированных приложением. Другая возможность заключается в множественном применении ранее составленных и апробированных сценариев в качестве прикладных объектов более сложных, иерархически организованных сценариев. В реализации ядра она обеспечивается наследованием классом *Scenario* всех рассмотренных выше методов базового класса ядра *Entity*, включая методы интроспекции. Необходимая параметризация сложного сценария связана с расширенной интерпретацией атрибутов всех его составляющих прикладных объектов.

Необходимо отметить, что описанный выше способ интерпретации сценария предполагает строгое упорядочивание экземпляров алгоритмов, которое может осуществляться как непосредственным манипулированием списком прикладных объектов, так и их автоматическим ранжированием. Для этого в классе реализованы соответствующие методы. В случае отсутствия циклов в диаграмме сценария данные методы обеспечивают однозначное упорядочивание экземпляров алгоритмов. При наличии циклов требуется дополнительное участие пользователя для исключения случаев некорректной интерпретации составленного сценария.

Необходимость включения в сценарий и данных и алгоритмов связана с итеративным характером создания итоговой графической сцены приложения, в ходе которого последовательно корректируются свойства прикладных данных, уточняется состав и последовательность применяемых методик моделирования и визуализации, корректируются их параметры, устанавливаются подходящие камеры вида, источники освещения и т.п. Перечисленные действия близко связаны друг с другом и обычно требуются для воспроизведения изучаемого явления более адекватным и выразительным способом.

Механизм событий

Для улучшения интерактивности приложения классом сценария реализуется специальный событийный механизм. В качестве событий рассматриваются операции конструирования, модификации или удаления прикладных объектов сценария. Стандартный обработчик событий предусматривает два типа действий: автоматическое связывание и автоматическую активацию.

Механизм автоматического связывания основан на анализе типов, который выполняется для всех вновь создаваемых и удаляемых прикладных объектов сценария, и включении их в подходящие по типу связи других объектов или исключении из них. Для управления этим механизмом связи объектов должны быть специально помечены, как автоматические. Все объекты включаются в автоматические множественные связи, если удовлетворяют их типу и не нарушают семантические условия размерности. В случае нарушений в них включаются только первые из потенциально подходящих кандидатов. Подобный механизм оказывается чрезвычайно полезным в тех случаях, когда тип объекта предопределяет некоторые действия, которые следует выполнить автоматически. Например, является удобным отобразить геометрические объекты сразу после их конструирования. Такой сценарий может быть реализован с помощью алгоритма растеризации сцены с множественной автоматической входной связью типа $GeometryData \subseteq Data$. Всякий раз, когда конструируется новый геометрический объект, он автоматически связывается с данным алгоритмом и может сразу отображаться без каких-либо дополнительных усилий.

Механизм автоматической активации предназначен для автоматической интерпретации сценария в случае возникновения каких-либо событий, связанных с его изменением. В приведенном выше примере изменение входной связи алгоритма растеризации, связанное с появлением в сценарии нового геометрического объекта, повлечет его автоматическую активацию и генерацию итогового графического изображе-

ния со вновь включенным объектом. Заметим, что общая реализация механизма предполагает полную интерпретацию сценария, поскольку активация одного алгоритма влечет изменение других объектов и порождает каскад новых событий, приводящих в свою очередь к активации других алгоритмов сценария.

Однако в большинстве практических случаев изменения носят локальный латентный характер, а состояние большинства прикладных объектов сценария не меняется. Поэтому для оптимизации повторно интерпретируемых сценариев используется принцип латентности. Латентными называются прикладные объекты сценария, состояние которых не изменилось на текущей итерации. Если все входные и смешанные объекты алгоритма являются латентными, повторная активация алгоритма не вызовет каких-либо изменений и, следовательно, не имеет смысла. Анализ латентности позволяет исключить обработку избыточных событий и, таким образом, повысить эффективность интерпретации сценария в целом. При интерактивном изучении сложных явлений и систем, связанном с многократной активацией алгоритмов моделирования и визуализации, этот аспект является принципиальным.

Шаблон приложений

Наконец, класс *Application* предоставляет типовое решение (шаблон) для построения пользовательских графических приложений моделирования и визуализации на основе программного ядра, унифицированного графического интерфейса пользователя и расширяемых библиотек прикладных компонентов и визуальных интерфейсов (рис. 3). В качестве стандартных средств реализации, допускающих перенос разрабатываемых приложений на различные аппаратные платформы с минимальными усилиями, использовались язык программирования Си++, графическая библиотека OpenGL и интерфейсная библиотека Qt.

Шаблон предусматривает необходимые средства для регистрации подключаемых библиотек прикладных компонентов и визуальных интерфейсов и управления ими в ходе пользовательских сессий.

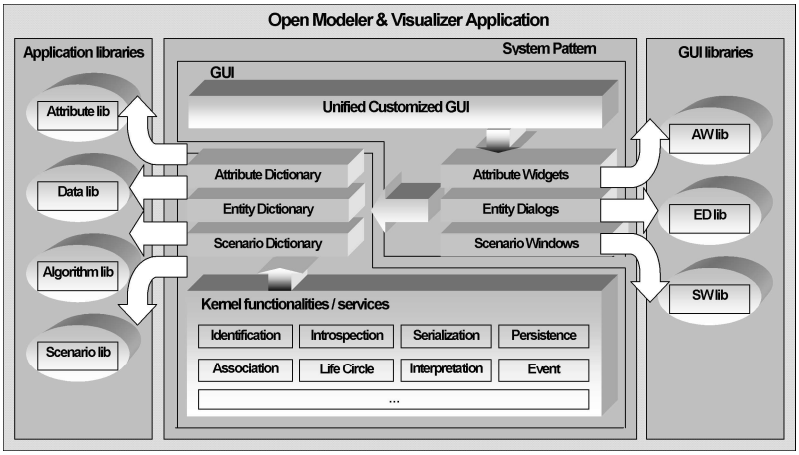

Рис. 3. Архитектура и компонентный состав открытой системы

Ядро (нижний прямоугольник) реализует выше описанные системные функции приложений. Графический интерфейс пользователя (верхний прямоугольник) предоставляет унифицированные окна для составления и интерпретации сценариев, для просмотра и редактирования экземпляров данных и алгоритмов, а также элементы для манипулирования отдельными атрибутами прикладных объектов. При этом каждому зарегистрированному в приложении прикладному компоненту (левый прямоугольник) может быть поставлено в соответствие несколько визуальных интерфейсов (правый прямоугольник), которые будут активироваться при попытке просмотра и редактирования прикладных объектов соответствующего типа.

Через графический интерфейс пользователь выбирает соответствующий сценарий для изучаемой задачи, корректирует его параметры, активизирует его и наблюдает результаты как анимированное трехмерное изображение. Итоговое изображение генерируется посредством интерпретации выбранного сценария, которая может быть инициирована непосредственно пользователем или осуществляться автоматически в результате обработки событий, происшедших в сценарии. Последняя возможность позволяет интерактивно в визуальном режиме изучать решаемую задачу в зависимости от параметров ее постановки. Квалифицированные пользователи могут составлять собственные сценарии для изучения сложных процессов и явлений, используя экземпляры компонентов моделирования и визуализации.

КОМПОНЕНТНАЯ МЕТОДОЛОГИЯ ПОСТРОЕНИЯ ПРИЛОЖЕНИЙ

При использовании рассмотренных выше архитектуры и шаблона разработка приложений сводится к проведению объектного анализа рассматриваемой предметной области, выделению специфичных для нее сущностей, реализации специализированных компонентов для их представления и их регистрации в шаблоне приложения. Разработка конкретного приложения может проходить в соответствии со следующей итерационной схемой:

* идентификация классов данных и алгоритмов, участвующих в постановке и решении задач моделирования и визуализации, на основе выделения ключевых абстракций предметной области. Для типовых случаев можно воспользоваться уже имеющимися библиотеками компонентов общего назначения и свести анализ к особенностям разрабатываемого приложения;

* определение семантики выделенных классов, установление отношений общности между ними, таких как «общее–частное», «целое–часть»; установление ассоциативных

отношений использования между классами и представление их в виде типизированных входных, выходных и смешанных связей; идентификация специфических атрибутов, свойств прикладных объектов и реализация их классов как наследников *Data* или *Algorithm*;

* разработка визуального представления прикладных объектов и реализация соответствующих визуальных интерфейсов для просмотра и манипулирования ими;

* регистрация разработанных прикладных компонентов и визуальных интерфейсов в шаблоне приложений;

* составление и тестирование различных сценариев работы приложения с выделенной системой сущностей, возможное ее критическое переосмысление, выделение новых типов объектов и, как следствие, возврат к предыдущим этапам разработки.

Для упрощения разработки типовых приложений открытая система предусматривает использование библиотек компонентов общего назначения для представления научных и инженерных данных, а также для реализации распространенных методов геометрического моделирования и научной визуализации.

Библиотека компонентов для представления прикладных данных

Библиотека научных и инженерных данных организована в виде иерархии наследуемых конкретных и абстрактных классов, которые реализуют понятия, традиционно используемые в приложениях численного моделирования. Библиотека предоставляет классы для манипулирования с такими типами прикладных данных, как наборы точек, кривые и ломаные, структурированные и неструктурированные поверхностные и объемные сетки, физические поля, цветовые палитры, шкалы, виды, плоскости сечений, наборы маркеров, текстуры, источники освещения. Фрагмент данной иерархии представлен на рисунке 4.

- *Entity*
    - *Data*
        - *GeometryData*
            - *PointSet*
            - *PolylineSet*
            - *SurfaceMesh*
                - *GenericSurfaceMesh*
                - *TriangularMesh*
                - *QuadricMesh*
            - *VolumeMesh*
                - *RegularMesh*
                    - *UniformMesh*
                    - *NonuniformMesh*
                    - *DeformedMesh*
                - *NonregularMesh*
                    - *GenericVolumeMesh*
                    - *TetrahedronMesh*
            - *GlyphSet*
        - *Field*
            - *ScalarField*
            - *VectorField*
            - *TensorField*
        - *Scale*
            - *LinearScale*
            - *OctaveScale*
            - *DecadeScale*
        - *Lighting*
        - *Camera*
        - *Palette*
        - *Texture*
        - *DrawRule*
            - *ColorRule*
            - *PseudoColorRule*

Рис. 4. Фрагмент библиотеки научных и инженерных данных

Все классы геометрических данных имеют обобщенную реализацию, позволяющую использовать их как в пространственно двумерных, так и в трехмерных случаях. Класс физических полей также поддерживает обобщенное многокомпонентное представление, которое соответствует скалярным, векторным и тензорным полям, заданным на различных типах многомерных геометрических данных.

Специальные классы типа *DrawRule* используются для задания способов графического представления геометрических объектов сцены. В частности, они позволяют показывать геометрические данные как наборы точек, каркасное или твердотельное представление. При этом закраска элементов (вершин, ребер, граней) может быть выполнена фиксированными цветами или в зависимости от значений поля, заданных на выбранных топологических элементах, в соответствии с указанной шкалой и цветовой палитрой (псевдозакраска).

Библиотека геометрического моделирования

Для построения программных приложений, связанных с трехмерным геометрическим моделированием, может быть использована библиотека компонентов конструктивного твердотельного моделирования (CSG). CSG-модель описывает сложное тело в терминах теоретико-множественных (булевых) операций над множеством элементарных тел, к числу которых относятся параллелепипед, сфера, цилиндр, конус, призма, тор и др. Обычно конструктивная модель представляют древовидной структурой, внутренние узлы которой определяют операции объединения, пересечения или разности, а листья — элементарные тела с учетом их положения в пространстве.

Библиотека моделирования организована в виде иерархии наследуемых классов, фрагмент которой изображен на рис. 5. Элементарные тела и теоретико-множественные операции представлены в ней соответствующими типами *CSGElement* и *CSGOperator*, наследуемыми от базового *CSGObject*. Подобная схема наследования позволяет доволь-

но естественно реализовать данную модель в рамках объектно-ориентированного подхода. Элементарные тела реализуются как экземпляры данных без связей, а операции — как экземпляры данных с парой связей типа *CSGObject*, соответствующих операндам, которыми могут быть как элементарные тела, так и другие операции CSG-модели. Результирующая операция, составляющая вершину дерева, соответствует самому геометрическому телу, которое в свою очередь может использоваться в дальнейшем сценарии приложения.

- *Entity*
  - *Data*
    - *GeometryData*
      - *CSGObject*
        - *CSGElement*
          - *Box*
          - *Sphere*
          - *Cylinder*
          - *Cone*
          - *Prism*
          - *Torus*
          - *Polyhedron*
        - *CSGOperator*
          - *Union*
          - *Subtraction*
          - *Intersection*
  - *Algorithm*
    - *CSG-BR-Algorithm*

Рис. 5. Фрагмент разработанной библиотеки компонентов для твердотельного геометрического моделирования

Для отображения геометрических тел, заданных конструктивной моделью, необходимо построить их граничное представление (BR). Для этого предназначен специальный алгоритмический компонент *CSG-BR-Algorithm*, осуществляющий аппроксимацию граничного представления элементарных тел с помощью класса многогранников *Polyhedron* и

последовательное применение к ним теоретико-множественных операций. Его реализация связана с использованием метода классификации точек области относительно геометрических тел. С этой целью для объектов типа *CSGObject* виртуально определен метод классификации, который непосредственно реализуется в наследуемых конкретных классах (для элементарных тел метод реализуется на основе простейших геометрических соотношений, для операций — путем логического анализа результатов классификации точки для каждого операнда).

Содержательные примеры геометрического моделирования с использованием CSG и BR представлений можно найти на web-странице проекта по визуализации промышленных моделей [10].

Библиотека научной визуализации

Библиотека компонентов для визуализации научных расчетов представляет собой набор абстрактных и конкретных классов, наследуемых от класса ядра *Algorithm*. Фрагмент иерархии классов представлен на рисунке 6. Компоненты реализуют распространенные методы обработки и визуализации научных и инженерных данных, такие как методы интерполяции, построения изолиний и поверхностей уровней, конструирования линий и трубок тока, трассировки траекторий частиц, сечений, маркеров и другие.

Реализованный набор алгоритмов позволяет интерполировать значения полей, заданных на регулярных и нерегулярных сетках, на любые другие сетки и произвольные геометрические данные. Специальная группа алгоритмов предназначена для инициализации типовых объектов, часто используемых в пользовательских сценариях, например, стандартные палитры, шкалы. Важную группу алгоритмов реализуют классы, обеспечивающие извлечение линий уровня из скалярных полей, заданных на поверхностных сетках, а также извлечение поверхностей уровней из скалярных полей, заданных на структурированных объемных сетках. Для визу-

ализации векторных полей библиотека предоставляет набор классов, позволяющих конструировать линии и трубки тока в стационарном случае, а также трассировать траектории частиц в нестационарном случае. Библиотека также позволяет вычислять различные нормы для полей; по значениям полей конструировать маркеры; создавать сечения отдельных геометрических фрагментов и сцен.

- *Entity*
    - *Algorithm*
        - *InitializationAlgorithm*
            - *InitColorSet*
            - *InitUniformScale*
            - *ParticleGenerator*
        - *InterpolationAlgorithm*
            - *InterpolationRNU-RNUMeshAlgorithm*
        - *ScalarFieldVisualizationAlgorithm*
            - *IsoLineAlgorithm*
            - *IsoSurfaceAlgorithm*
        - *VectorFieldVisualizationAlgorithm*
            - *StreamlineAlgorithm*
            - *StreamtubeAlgorithm*
            - *ParticleTracingAlgorithm*
        - *TensorFieldVisualizationAlgorithm*
            - *FieldNormAlgorithm*
            - *GlyphAlgorithm*
        - *OrthoSliceSectionAlgorithm*
        - *RenderingAlgorithm*
        - *ReadingAlgorithm*
            - *FileReadingAlgorithm*
        - *WritingAlgorithm*
            - *FileWritingAlgorithm*
            - *PPMImageWritingAlgorithm*

Рис. 6. Фрагмент разработанной библиотеки методов научной визуализации

Кроме методов визуализации данная библиотека предоставляет классы для растеризации геометрических сцен в графическом контексте приложения в соответствии с назначенными правилами отображения, а также вспомогательные классы для сохранения/восстановления объектов сцены в файлы/из файлов внутреннего формата и сохранение созданных изображений сцены в растровом формате PPM.

Необходимо отметить, что большинство реализованных методов геометрического моделирования и визуализации имеет обобщенную реализацию, независящую от конкретных типов входных данных. Например, метод маркированных кубов для выделения поверхности уровня скалярного поля реализован с применением абстракции топологически регулярных сеточных данных. В силу этого соответствующих программный компонент применим как к равномерным, неравномерным ортогональным сеткам, так и к деформированным сеткам. Привлечение механизма абстрактных типов данных существенно облегчает построение сложных композиционных сценариев с использованием относительно небольшого репертуара методов и реализующих его компонентов. Данное обстоятельство оказывается существенным для функциональной эволюции открытой системы, поскольку добавление новых производных типов данных и алгоритмов не приводит к необходимости создания новых версий ранее реализованных классов.

ПРИЛОЖЕНИЯ СИСТЕМЫ

Полнота разработанных и рассмотренных выше библиотек компонентов соответствует функциональности системы визуализации общего назначения, подобной IRIS Explorer, IBM Data Explorer, AVS, Vis5D, и обеспечивает ее развитые возможности для изучения широкого круга явлений, процессов в таких областях, как динамика жидкости и газа, механика деформируемого тела, химическая кинетика,

релятивистская механика и многих других. Вместе с тем, открытая система предоставляет развитые инструментальные средства для построения специализированных интегрированных приложений моделирования и визуализации.

В настоящем разделе, следуя представленным рабочим сценариям системы, рассматриваются некоторые примеры таких приложений. Более подробный список приложений, разработанных на основе открытой системы, необходимые ссылки на математические постановки задач, применяемые численные методы решения можно найти на web-странице проекта по научной визуализации [9].

Визуализация сверхзвукового течения вязкого газа над открытой прямоугольной каверной

Задача имеет важное значение для проектирования ряда авиационных и космических систем и связана с численным решением пространственно двухмерных стационарных уравнений Навье-Стокса. Исходными данными, поступающими от программы численного моделирования, являются регулярные сетки и заданные на них поля давления, температуры и скорости потока газа.

Для получения полной картины возвратного течения вблизи угла каверны применяется метод трассировки траекторий частиц. Частицы маркируются стрелками, окрашиваемыми в соответствии с амплитудой скорости. Для применения метода цветовой заливки исходное поле скорости интерполируется на набор частиц и вычисляется его норма. Для информативности дополнительно визуализируется поле давлений с применением контурного метода. Правила отображения задаются как входные параметры алгоритмов отображения, которые выполняют совместный анализ геометрических объектов сцены и осуществляют их закраску в указанном виде.

Порядок применения алгоритмов и получения промежуточных результатов легко прослеживается по диаграмме сценария, представленной на рисунке 7.

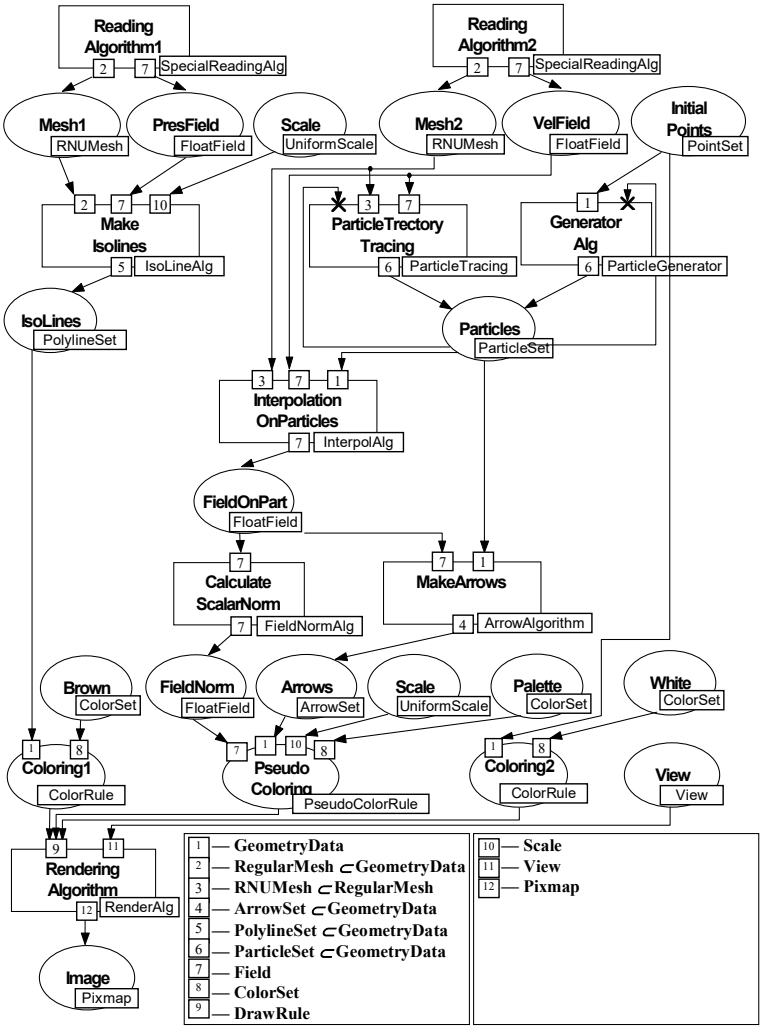

Рис. 7. Диаграмма сценария визуализации, предназначенного для изучения задач газовой динамики

Сгенерированные приложением изображения приводятся на рисунке 8.

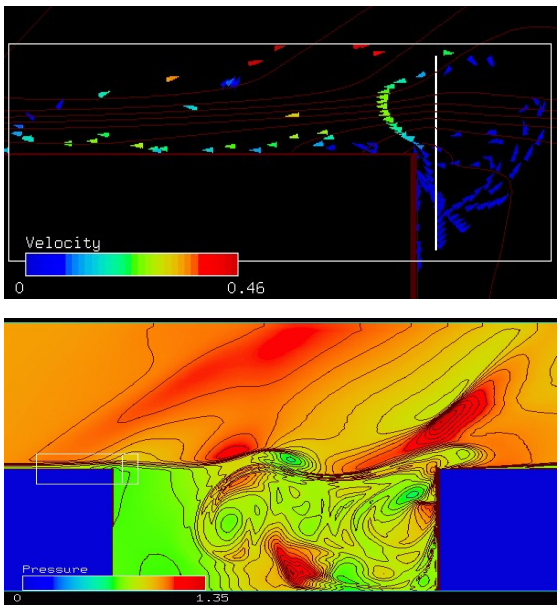

Рис. 8. Применение методов визуализации для изучения сверхзвукового течения вязкого газа вблизи открытой каверны.

Визуализация переходных процессов в оптически бистабильных полупроводниковых кристаллах

Задача связана с моделированием переходных процессов в полупроводниковых кристаллах с оптической бистабильностью и представляет интерес для разработчиков оптических систем, в частности, запоминающих устройств. Для исследуемой двухпучковой схемы, основанной на нелинейном поглощении лазерного излучения в кристаллах микронных размеров, существенна следующая информация: гарантированный переход во второе устойчивое состояние (затруднен, когда подвижность носителей заряда высока), время данного перехода (определяет быстродействие элемента) и

профиль интенсивности информационного пучка на выходе из кристалла (симметричный профиль повышает качество регистрации переключения и позволяет снизить размеры элемента).

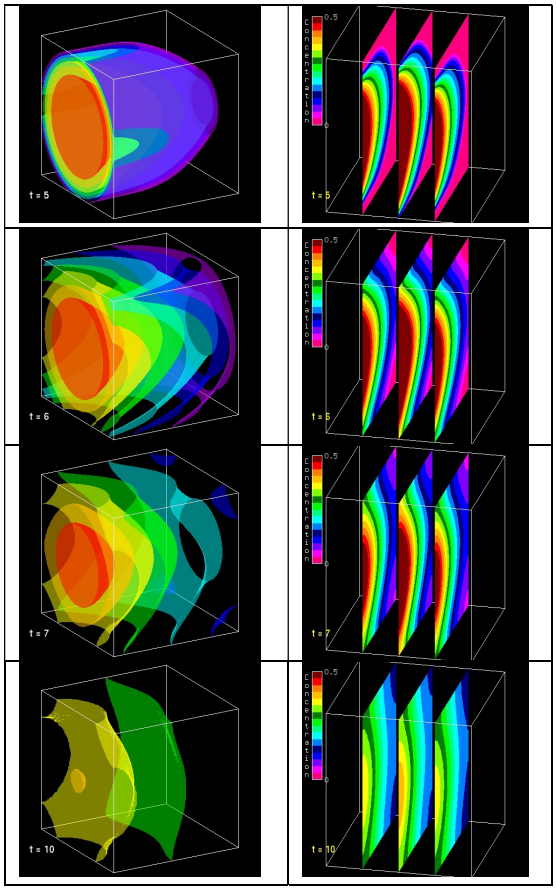

Рис. 9. Применение методов визуализации для изучения динамики переходных процессов в полупроводниковых оптически бистабильных кристаллах

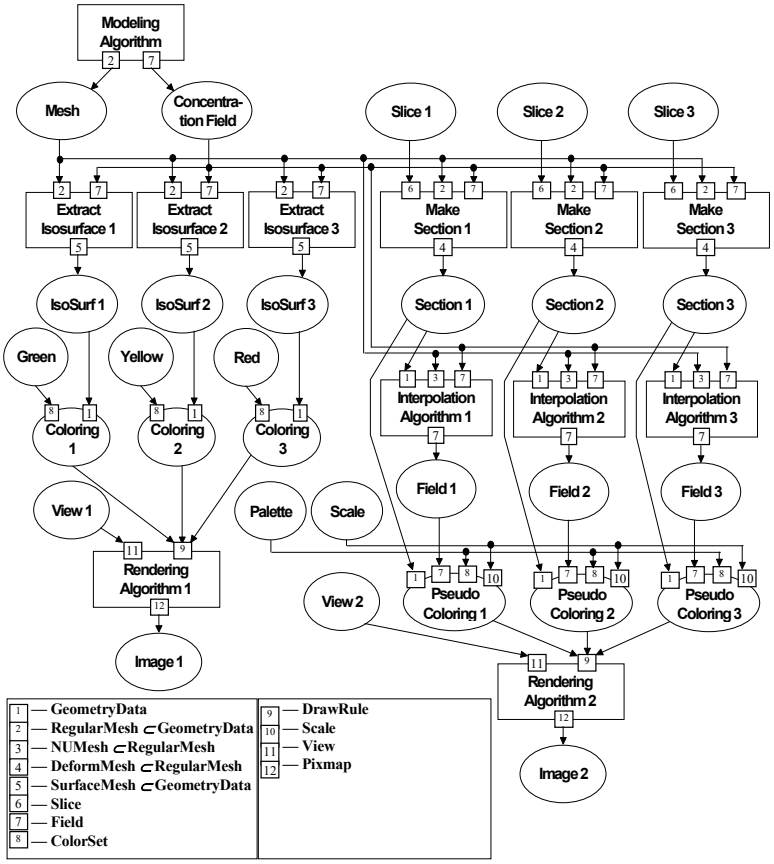

Рис. 10. Диаграмма сценария визуализации, предназначенного для изучения переходных процессов в полупроводниковых кристаллах

Приведенная на рисунке 9 серия изображений иллюстрирует процесс переключения кристалла при заданном значении коэффициента диффузии носителей заряда. Методами выделения поверхностей уровня и сечений визуализируется динамическое пространственно трехмерное поле концентрации носителей заряда. Применяемая техника отобра-

жения уровней концентрации как полупрозрачных окрашиваемых поверхностей дает пространственное ощущение геометрии распределения зарядов и позволяет более детально исследовать переходные процессы. На изображениях в правой колонке поле концентрации носителей заряда визуализируется на сечениях кристалла с использованием традиционной техники псевдозакраски.

Диаграмма сценария для изучения задачи с помощью описанных методик представлена на рисунке 10.

Моделирование генератора с нелинейной инерционностью

Приложение предназначено для изучения поведения генератора с нелинейной инерционностью. Математической моделью генератора является система обыкновенных дифференциальных уравнений (ОДУ). Для изучения поведения данной модели и определения притягивающей области в фазовом пространстве (странного аттрактора) возможно реконструировать фазовые траектории, соответствующие различным начальным условиям соответствующей задачи Коши. Множество начальных условий может быть задано посредством конструирования сложного твердого тела в интересующей области пространства и генерирования точек на его граничном представлении. Затем набор задач Коши решается численным методом, и полученные результаты визуализируются посредством окрашивания фазовых траекторий в цвета в соответствии, например, с величиной производной. Для ее пространственного анализа может применяться рассмотренный выше метод выделения поверхностей уровня.

Состав необходимых компонентов для реализации данного сценария и его диаграмма показаны на рисунках 11, 12. Обсудим особенности сценария. Неопределенный характер возможных постановок задачи Коши, связанный со свойствами жесткости дифференциальных уравнений, их смешанным типом, структурной разреженностью, может приводить к необходимости изменения вычислительной стратегии.

Например, при решении жестких систем ОДУ обычно применяются неявные методы, реализация которых связана с решением нелинейных алгебраических уравнений на каждом шаге временной дискретизации. Для решения последних методами ньютоновского типа необходимо применить тот или иной метод решения систем линейных алгебраических уравнений, например, метод LU-факторизации. Существенно, что выбор каждого метода в сценарии приложения может осуществляться независимо и в конечном итоге определяться особенностями решаемой конкретной задачи.

- ***Entity***
  - ***Data***
    - ***Vector***
    - ***Matrix***
    - ***Function***
      - ***AttractorFunction***
  - ***Algorithm***
    - ***FunctionEvaluationAlgorithm***
    - ***ODEImplicitAlgorithm***
      - ***RungeKuttaAlgorithm***
    - ***NonlinearSystemAlgorithm***
      - ***NewtonAlgorithm***
    - ***LinearSystemAlgorithm***
      - ***LUAlgorithm***

Рис. 11. Фрагмент библиотеки компонентов для моделирования генератора

Описанная редукционная схема решения систем ОДУ естественно реализуется средствами открытой системы. Методы решения систем ОДУ, систем нелинейных и линейных алгебраических уравнений представляются соответствующими алгоритмическими классами. Входами перечисленных алгоритмов являются данные, участвующие в математических постановках соответствующих задач, выходами — результаты их численного решения.

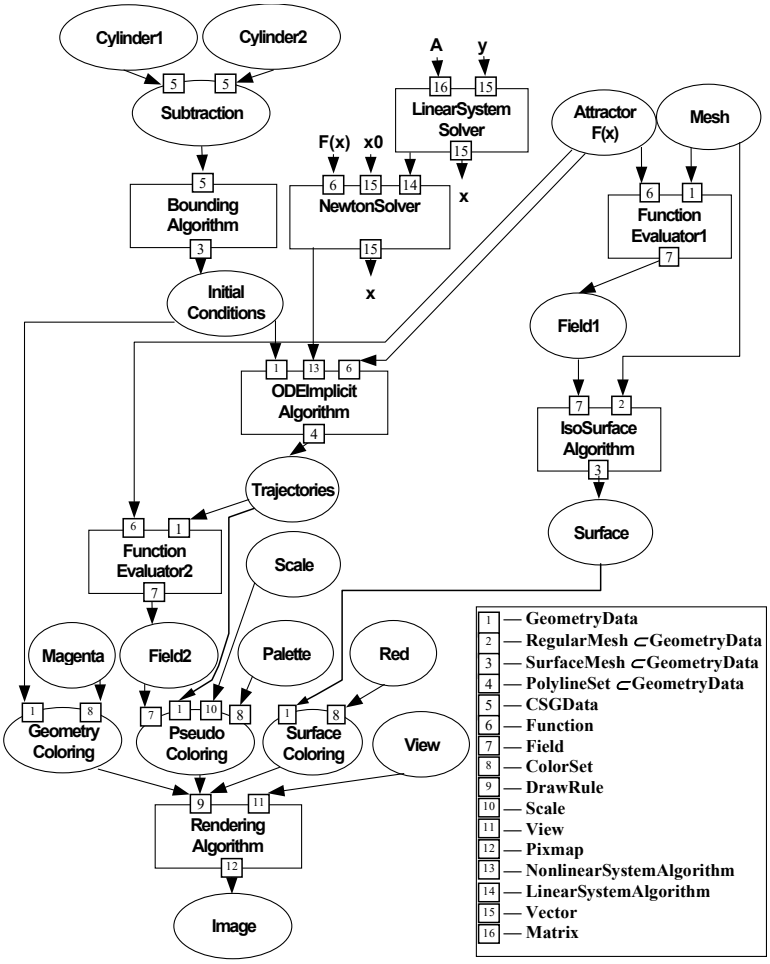

Рис. 12. Диаграмма сценария моделирования и визуализации для изучения поведения генератора с нелинейной инерционностью

Заметим, что входные и выходные связи задействованы только у основного алгоритмического объекта — метода решения системы ОДУ, который активируется в результате

наступления соответствующих событий в сценарии. Ассоциативные отношения между остальными объектами-алгоритмами используются непосредственно при численном решении вспомогательных задач.

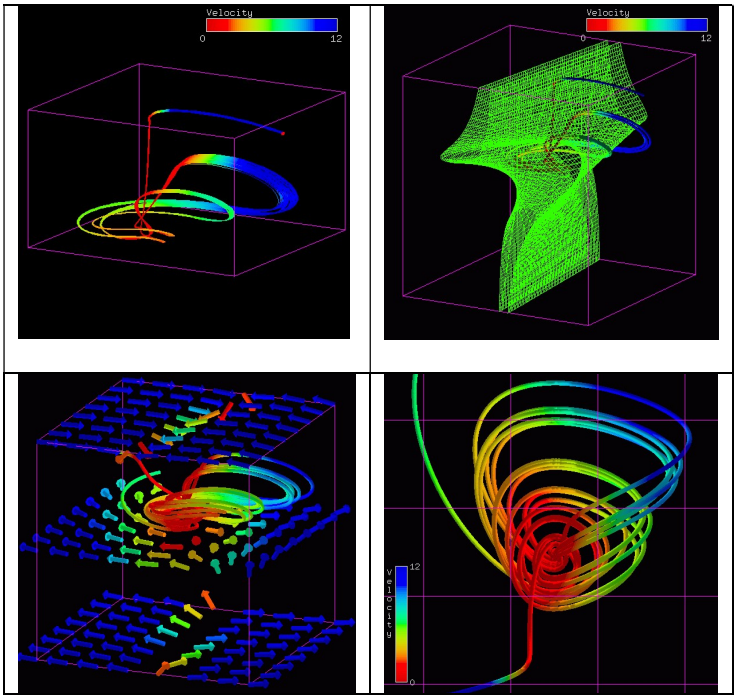

Рис. 13. Применение методов моделирования и визуализации для изучения поведения генератора с нелинейной инерционностью

На рисунке 13 представлены некоторые изображения, которые сгенерированы приложением, разработанным на основе открытой системы. Два верхних изображения получены с помощью описанного выше сценария. Нижние изображения иллюстрируют иные возможности визуального изучения математической модели генератора с использованием методики

маркировки векторного поля в разных сечениях интересующей области фазового пространства и реконструкции трубок фазовых траекторий.

ЗАКЛЮЧЕНИЕ

Таким образом, рассмотрены архитектура, принципы организации и функционирования разработанной в ИСП РАН открытой системы для математического моделирования и научной визуализации **OpenModeler&Visualizer**. Система предоставляет развитые возможности для построения сложных программных приложений на основе компонентно-ориентированной методологии и предусмотренных средств разработки. Система апробирована в ходе реализации ряда актуальных научно-технических проектов и представляется перспективной для построения сложных многофункциональных междисциплинарных интегрированных комплексов в существенно разных предметных областях.

Принципы организации и функционирования системы, а также примененные в ней программно-инструментальные решения имеют важное учебно-методическое значение. А сама система и созданные на ее основе приложения могут использоваться при выполнении практических работ по курсу “Научная визуализация”. Технические характеристики системы, документация, сценарии и примеры использования могут быть получены по запросу с web-страниц проектов по научной и промышленной визуализации ИСП РАН [9, 10]

СПИСОК ЛИТЕРАТУРЫ

1. W. Brodlie, J. R. Gallop, A. J. Grant, J. Haswell, W. T. Hewitt, S. Larkin, C. C. Lilley, H. Morphet, A. Townend, J. Wood, H. Wright, Review of Visualization Systems Advisory Group on Computer Graphics. Technical Report 1999, www.agocg.org/cd/reports/visual/vissyst/DOGBOO_2.HTMK.

2. IBM Open Data Explorer Web Site, http://www.research.ibm.com/dx.
3. IRIS Explorer Centre of Excellence. Web Site, http://www.comp.leeds.ac.uk/iecoe.
4. Vis5D. Vis5D Web Site, http://www.ssec.wisc.edu/~billh/vis5d.html.
5. Семенов В.А., Крылов П.Б., Морозов С.В., Роминов М.Г., Тарлапан О.А. Объектно-ориентированная методология разработки интегрированных приложений моделирования и визуализации. // Труды Института системного программирования РАН — М.: Открытые системы, 2000, с. 101-114.
6. Семенов В.А., Крылов П.Б., Морозов С.В., Тарлапан О.А. Объектно-ориентированная архитектура для приложений математического моделирования и научной визуализации. // Программирование, 2000, N 2, с.29-40.
7. Semenov V., Morozov S., Tarlapan O., Belyaeva A. Component-based development of scientific computing applications in OpenMV environment. // Proceedings of 16th IMACS World Congress, Lausanne, Switzerland, August 21–25, 2000, ed. Deville M., Owens R., ISBN 3-9522075-1-9.
8. Ivannikov V., Morozov S., Semenov V., Tarlapan O., Rasche R., Jung T. Parallel object-oriented modeling and visualization in OpenMV environment. // Proceedings of GraphiCon'99, Moscow, August 26 - September 1, 1999, pp. 206-213.
9. Проект научной визуализации ИСП РАН, http://www.ispras.ru/~3D
10. Проект визуализации промышленных моделей ИСП РАН, http://www.ispras.ru/~step.