Объектно-ориентированная методология разработки интегрированных приложений моделирования и визуализации *

Семенов В.А., Крылов П.Б., Морозов С.В.,
Роминов М.Г., Тарлапан О.А.

Аннотация

В статье описывается объектно-ориентированная методология построения интегрированных приложений моделирования и визуализации в различных предметных областях на единой концептуальной, инструментальной и программной основе. Разрабатываемые приложения имеют общую открытую архитектуру, которая включает объектно-ориентированное ядро, инвариантное по отношению к различным областям и проблемам, унифицированный графический интерфейс пользователя и специализированные прикладные библиотеки классов. Объектно-ориентированное ядро поддерживает представление итоговой графической сцены в виде композиции связанных типизированных данных и алгоритмов и реализует общие механизмы составления сценариев работы приложений и их интерпретации. Функциональность приложений определяется главным образом полнотой подключенных прикладных библиотек и семантикой их классов. Достоинства предложенной методологии демонстрируются примерами разработки на ее основе системы научной визуализации общего назначения и специализированных приложений для изучения ряда актуальных физических проблем.

1. Введение

В настоящее время компьютерные средства математического моделирования и визуализации научных расчетов получили широкое распространение и составляют неотъемлемую часть современных систем автоматизированного моделирования и проектирования (CAD/CAM/CAE). Разработка таких систем, сочетающих широкую предметную функциональность с развитыми графическими интерактивными вычислительными средствами, обычно требует чрезвычайно высоких затрат и сопряжена с решением целого круга проблем, включающего архитектурное проектирование, интеграцию функционально разнородных компонентов, организацию дружественного пользовательского интерфейса, обеспечение открытости, переносимости, эффективное использование на параллельных вычислительных системах.

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 разнообразных междисциплинарных исследований, обуславливают необходимость выработки общей методологии их построения. В этом случае проблема создания новых программных систем переводится в плоскость активного использования уже имеющихся программных компонентов и их возможной адаптации к новым задачам.

Поскольку, вырабатывая методологию, претендующую на существенную предметную общность, мы не можем конкретизировать семантику задач и назначение разрабатываемых программных компонентов, то будем исходить из следующих общих принципов построения приложений:

- наличие унифицированного программного ядра, инвариантного по отношению к различным предметным областям, прикладным задачам моделирования и визуализации, применяемым аппаратным платформам;

* Работа выполняется в рамках научно-исследовательских проектов «Объектно-ориентированная методология научной визуализации» (грант РФФИ 98–01–00321) и «Визуализация сложных физических явлений и математических объектов в виртуальной среде» (грант ИНТАС 96–0778)

- модульная организация или, другими словами, возможность построения конкретной прикладной системы из отдельного набора модулей, семантика которых соответствует сущностям рассматриваемой предметной области, способам их представления, программной реализации и, в конечном счете, определяет функциональность всей системы;
- обеспечение гибкой дисциплины связывания экземпляров модулей с целью составления и применения различных сценариев конструирования и функционирования сложных композиционных сцен;
- обеспечение механизмов обобщенной реализации модулей, при которой включаемые в систему новые модули могут взаимодействовать с имеющимися без строгой конкретизации контекста использования, а создание новых модулей может осуществляться на основе разработанных ранее;
- программная реализация модулей на основе современных информационных, графических, коммуникационных стандартов, с использованием, в частности, стандарта представления геометрии (STEP) [1], языка описания сцен виртуальной реальности (VRML) [2], графической библиотеки растеризации изображений (OpenGL) [3], интерфейса распределенных приложений (MPI) [4].

Использование объектно-ориентированного подхода для построения прикладных программных систем в соответствии с вышеперечисленными принципами имеет ключевое значение, поскольку предусматриваемые им механизмы инкапсуляции, наследования и полиморфизма могут быть конструктивно применены как при реализации унифицированного ядра, так и при разработке прикладных библиотек классов [5, 6].

Настоящая работа преследует цель представить объектно-ориентированную методологию создания интегрированных приложений математического моделирования и научной визуализации, опирающуюся на оригинальную открытую модульную архитектуру. В разделе 2 проводится объектный анализ и описывается ядро приложений. Раздел 3 посвящен некоторым аспектам унифицированной разработки интегрированных приложений с использованием данного ядра. В разделе 4 рассматривается система визуализации общего назначения OpenMV, построенная на основе предложенной методологии и разработанных программных средств и компонентов и иллюстрирующая их преимущества. В разделе 5 обсуждаются возможности специализации системы OpenMV для изучения сложных физических проблем.

2. Объектно-ориентированное ядро для моделирования и визуализации

Будем рассматривать итоговую графическую сцену приложения как композицию связанных типизированных данных, участвующих во всех процессах данного приложения, включая моделирование, визуализацию и растеризацию, а также алгоритмов, создающих, преобразующих, удаляющих эти данные и реализующих упомянутые выше процессы. Будем различать пассивные объекты-данные, которые контролируют только собственное поведение, и активные объекты-алгоритмы, которые могут управлять поведением других объектов в результате рассылки им сообщений в классическом объектно-ориентированном стиле. Создание итоговой сцены подразумевает определение экземпляров классов данных и алгоритмов, установление связей между ними, составление сценария из отдельных объектов данных и алгоритмов и его интерпретации. Подобный подход, основанный на подразделении объектов на активные и пассивные, отражает основную идею методологии Бэйлина, известную как объектно-ориентированная спецификация требований [7], и успешно зарекомендовал себя при создании приложений вычислительной математики [6].

Рассмотрим более подробно объектно-ориентированное ядро для моделирования и визуализации. Под ядром понимается система абстрактных и конкретных классов, выражающих ключевые сущности рассматриваемой предметной области, реализующие базовые методы манипулирования ими и служащих конструктивной основой для построения широкого круга приложений с использованием принципов объектно-ориентированного программирования.

Базовым абстрактным классом ядра является класс ***Object***, который выражает сущность произвольных данных и алгоритмов, участвующих в моделировании и визуализации. Объекты могут вводиться, конструироваться, связываться друг с другом, преобразовываться, запоминаться, визуализироваться, копироваться и уничтожаться в ходе работы приложения. Для единообразного манипулирования различными типами объектов и обеспечения функциональности ядра класс ***Object*** инкапсулирует идентификатор, номер версии и свое логическое местоположение в сцене (будет рассмотрено позже). Этими атрибутами обладают все объекты сцены, классы которых являются производными от ***Object***.

Помимо общих атрибутов каждый конкретный объект $obj \in Object$ имеет свой собственный набор атрибутов, определяющий его внутреннее состояние и поведение, а также набор типизированных соединений. Соединения являются внешними портами объектов, посредством которых они могут быть связаны с другими подобными объектами. Тип конкретного соединения $Link \subseteq Object$ определяет потенциальную возможность связи данного объекта с любым другим объектом $lobj \in LinkObject$, тип которого удовлетворяет типу соединения или, другими словами, является его подтипом $LinkObject \subseteq Link$.

Наличие связей в сцене характеризует функциональную зависимость ее объектов и, следовательно, необходимость их совместного рассмотрения и анализа. Каждая установленная связь определяет отношения использования главным объектом других вспомогательных объектов, с которыми он связан. Будем различать одиночные и множественные связи. Одиночная связь определяет отношение использования между парой объектов, рассматриваемых как основной и вспомогательный. Множественная связь используется в тех случаях, когда основной объект находится во взаимодействии с подмножеством однотипных вспомогательных объектов через одно соединение $lobj\_i \in LinkObject\_i \subseteq Link$, $i = 1, .., n$. Число объектов n, объединенных множественной связью, является произвольным и зависит только от конкретного сценария, реализуемого в приложении.

Для различия способов, с помощью которых объекты взаимодействуют между собой, все соединения и соответствующие им связи подразделяются на входные, выходные и смешанные. Участвуя в связях, основной объект использует данные и методы вспомогательных объектов и, следовательно, может изменять состояния связанных с ним объектов. Предполагается, что основной объект способен изменять состояния вспомогательных объектов через выходные и смешанные связи, и неспособен влиять на состояние входных объектов. Основной объект зависит только от входных и смешанных объектов и не зависит от выходных объектов. Основной и вспомогательные объекты, включенные в смешанную связь, являются взаимозависимыми. Таким образом, могут быть установлены описанные отношения зависимости, основанные на классификации соединений взаимодействующих объектов.

Инкапсулируя описанные выше свойства, класс ***Object*** определяет следующие группы методов, общие для всех его конкретных экземпляров:

- создание (конструирование, уничтожение, копирование, чтение из файла, вывод в файл),
- идентификация (идентификация объекта, его класса, базового класса, версии, верификация принадлежности к заданному типу),
- взаимосвязывание (получение количества соединений, их классов и типов, связывание объектов),
- получение информации о сцене (получение параметров сцены, логическое упорядочивание объектов в сцене).

Основными понятиями ядра являются также объекты-данные $dat \in Data \subset Object$ и объекты-алгоритмы $alg \in Algorithm \subset Object$. Абстрактные классы ***Data, Algorithm*** являются производными от ***Object*** и наследуют его свойства и поведение. Класс ***Data*** выражает сущность разнообразных научных данных, возникающих в приложениях моделирования и визуализации. Объекты-данные являются пассивными объектами, контролирующими только собственное поведение, и поэтому могут иметь только входы. Класс ***Algorithm*** представляет различные алгоритмы, преобразования, операции и вспомогательные утилиты, реализующие все процессы в приложениях моделирования и визуализации. Алгоритмы являются активными объектами, контролирующими не только собственное поведение, но и поведение связанных с

ними вспомогательных объектов (не обязательно данных). Алгоритмы могут не иметь входов, но обязательно имеют выходные и/или смешанные соединения.

Отличительная особенность алгоритмов заключается в возможности их активизации, которая реализуется при наступлении соответствующих событий в сцене. В этом случае сцена активизирует соответствующий метод выполнения алгоритма. Предполагается, что к моменту инициации все связи алгоритмов были установлены. В противном случае алгоритм не активизируется и рассматривается как объект-данное. При вызове метода запуска алгоритм рассылает сообщения связанным с ним объектам для получения информации о состоянии входных и смешанных объектов, выполнения необходимых операций над ними, а также для обновления состояния смешанных и создания выходных объектов.

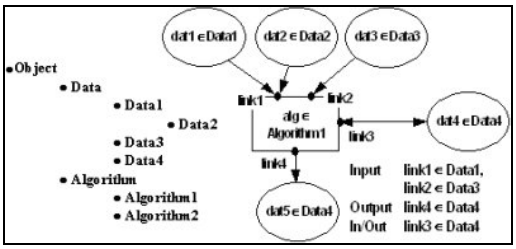

Рис. 1. Иерархия классов и схема связывания и взаимодействия объектов

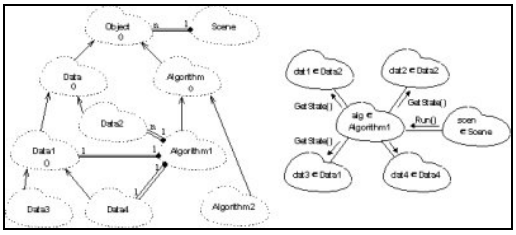

Рис. 2. Диаграммы классов и объектов в нотации Буча

На рис. 1 представлен пример взаимодействия объектов. Данный пример иллюстрирует, каким образом данные и алгоритмы могут связываться и взаимодействовать через типизированные соединения. Иерархия классов определяет отношения наследования между классами данных и алгоритмов. Диаграмма сцены полностью определяет схему связывания и характер взаимодействия объектов. В диаграмме экземпляры данных и алгоритмов показаны, соответственно, как овалы и прямоугольники. Соединения объектов промаркированы точками. Связи между объектами показаны стрелками. Подобная схема может быть представлена в терминах объектно-ориентированной методологии с использованием нотации Буча [8]. Для иллюстративности на рисунке 2 представлены диаграммы классов и объектов в нотации Буча для рассмотренного примера.

Согласно диаграмме сцены данные ***dat1***, ***dat2***, ***dat3*** связаны с алгоритмом ***alg*** через входные соединения. Множественное соединение ***link1*** типа ***Data1*** обеспечивает одновременную связь с ***dat1***, ***dat2***, удовлетворяющими его типу. Единственный объект ***dat3*** связан через простое соединение ***link2*** типа ***Data3***. Аналогичным образом объекты ***dat4***, ***dat5*** связываются через смешанное соединение ***link3*** и выходное соединение ***link4***. После активизации алгоритм ***alg*** обменивается сообщениями с ***dat1***, ***dat2***, ***dat3, dat4***, приводя к конструированию выходного объекта ***dat5*** и обновлению ***dat4***.

Отметим, что принятый способ направления стрелок соответствует потокам данных в сценарии. Это обстоятельство выявляет схожесть с традиционной диаграммой потока данных, широко используемой в системах визуализации и анимации. Однако рассматриваемая диаграмма сценария выражает более общую парадигму "сущности-связи", которая, на наш взгляд, является более предпочтительной вследствие возможностей задания более сложных типов взаимодействия объектов. Задача моделирования и визуализации аттрактора, обсуждаемая ниже, дает пример взаимодействия объектов, которое не может быть представлено парадигмой потоков данных, но может быть специфицировано непосредственно в терминах нашего подхода.

Наконец, класс ***Scene*** является контейнерным классом, поддерживающим упорядоченное представление (композицию) всех объектов данных и алгоритмов, включенных в сцену, и обеспечивающим широкую функциональность, необходимую для разрабатываемых приложений. Конструирование результирующей сцены осуществляется в результате непосредственного манипулирования ее объектами и активизации конвейера, составленного из ее отдельных алгоритмов. Алгоритмы конвейера сцены предварительно упорядочиваются и затем активизируются для решения частных задач

моделирования и визуализации. В ходе выполнения алгоритмов объекты сцены взаимодействуют друг с другом, что приводит к созданию новых объектов и обновлению уже существующих.

Экземпляры класса ***Scene*** используются для представления сцен как композиций данных, а также для спецификации и интерпретации сложных сценариев моделирования и визуализации. Необходимость включения в сцену и данных, и алгоритмов связана с итеративным характером создания итоговой картины, в ходе которого последовательно корректируются свойства данных, уточняется состав и последовательность применяемых методик моделирования и визуализации, корректируются их параметры, устанавливается подходящая камера вида. Перечисленные действия близко связаны друг с другом и обычно требуются для воспроизведения изучаемого явления более адекватным и выразительным способом.

Имея описанное представление сцены, можно модифицировать его вплоть до получения желаемого изображения. То же самое представление можно использовать для динамического моделирования и анимации полученных результатов. Эта возможность реализуема, поскольку однажды созданный сценарий может снова и снова использоваться для семантически эквивалентных наборов данных, сгенерированных приложением, а также введенных из файла или полученных от других процессов.

Функциональность класса ***Scene*** обеспечивается абстракциями данных и алгоритмов, специфичных для разрабатываемых частных приложений. Данный класс определяет следующие группы методов:

- создание (регистрация класса объекта, создание объекта данного класса, обновление версии объекта, копирование, удаление);
- идентификация (получение объекта по его идентификатору, выбор объектов, принадлежащих заданному типу);
- взаимосвязывание (связывание объекта с заданными вспомогательными объектами, автоматическое связывание объектов);
- интерпретация сценария (получение физического и реального времени, номера итерации; планирование сценария (ранжирование сцены, упорядочивание ее объектов), анализ латентных объектов, запуск отдельных алгоритмов и сценария в целом);
- обработка событий приложения.

Обсудим некоторые особенности манипулирования всей сценой и интерпретации соответствующего ей сценария.

Для упрощения процедуры связывания обеспечивается специальный механизм автоматического связывания объектов. Данный механизм основан на анализе типов, выполняемом для всех объектов сцены, и включении подходящих объектов в связи, помеченные как автоматические. Все приемлемые объекты включатся в множественные связи. Только первые из выбранных объектов включаются в одиночные связи. Подобный механизм оказывается чрезвычайно полезным в тех случаях, когда тип объекта предопределяет некоторые семантические действия, которые следует выполнить автоматически. Например, является удобным отрисовать геометрические объекты автоматически сразу после их конструирования и включения в сцену. В этом случае процедура отрисовки всей сцены может быть реализована как алгоритм с множественным входным соединением типа ***GeometryData*** $\subset$ ***Data***. Всякий раз, когда конструируется новый геометрический объект, он связывается с данным алгоритмом и может автоматически отрисовываться без каких-либо дополнительных усилий.

Определяя сценарий, пользователь располагает алгоритмы в той логической последовательности, в которой сцена должна их выполнять. Поскольку для больших сценариев это может вызвать затруднения, сцена позволяет упорядочивать алгоритмы автоматически. Так как классификация соединений объектов соответствует отношениям зависимости между ними, сцена способна анализировать связи между объектами для определения характера их зависимости и упорядочивания их в требуемом логическом порядке. Методы упорядочивания, предоставляемые классом ***Scene***, основаны на ранжировании сцены. Ранжирование преследует цель присвоить каждому объекту пару целых чисел (ранг и индекс), указывающих его местоположение в диаграмме сценария. Подобные диаграммы показывают объекты сцены как геометрические примитивы, соединенные стрелками, и часто используются в системах визуализации и анимации как

эффективные средства для графического представления сценариев и их визуального программирования.

Ранжирование устанавливает некоторый логический порядок следования объектов в сцене, при котором объекты с меньшим рангом не зависят от состояния объектов с большим или равным рангом, и, наоборот, объекты с большим рангом зависят только от состояния объектов с меньшим рангом. Процедура ранжирования просто формализуется и осуществляется путем последовательного приписывания рангов объектам в результате итеративного просмотра всей сцены. Нулевой ранг приписывается объектам, которые не имеют входных связей и не принимают участия в выходных связях других объектов. Для любого другого объекта ранг определяется как максимальный из рангов его входных объектов и объектов, связанных с ним через выходные соединения, плюс один при условии, что его собственные выходные объекты имеют неопределенный ранг. Смешанные связи последовательно рассматриваются как входные и выходные. Если ранг не может быть присвоен таким способом, и в сцене существуют объекты с неопределенным статусом, то это означает наличие как минимум одного цикла, и результат ранжирования может оказаться неоднозначным. В таких случаях ранг может быть присвоен первому объекту из оставшихся в соответствии с правилами, формализованными выше. Пользователь может влиять на способ, которым будут отранжированы объекты в обнаруженных циклах, предварительно упорядочив их. После окончания ранжирования объекты, имеющие один и тот же ранг, индексируются для определения их позиции в диаграмме сценария.

Способность сцены ранжировать и упорядочивать объекты является очень важной для корректной интерпретации сценария, поскольку алгоритмы должны активизироваться в логическом порядке, приводя к передаче данных через алгоритмический конвейер и получению итоговой картины. Если данный порядок был нарушен, то возможно появление различных типов ошибок, связанных с попытками активизировать алгоритмы без наличия на их входах подготовленных данных.

Если сценарий выполняется повторно, то на некоторых итерациях могут существовать латентные объекты. Латентными объектами называются такие, состояние которых не изменяется на текущей итерации. Если входные объекты алгоритма не изменились на текущей итерации сценария, активизация этого алгоритма не вызовет изменений выходных объектов и, следовательно, не имеет смысла. Анализ латентности позволяет исключать избыточные события активизации латентных алгоритмов и, таким образом, повысить эффективность интерпретации сценария в целом. Сцена осуществляет анализ латентности путем сравнения версий входных объектов с их версиями, сохраненными на предыдущей итерации. Анализ латентности должен выполняться при интерпретации сценария.

Наконец, класс ***Scene*** предоставляет метод для обработки типовых событий приложения. Такие события диспетчеризуются и напрямую рассылаются соответствующим методам сцены.

3. Разработка интегрированных приложений с использованием объектно-ориентированного ядра

Рассмотренное выше объектно-ориентированное ядро обладает, на наш взгляд, достаточной общностью и гибкостью для его использования при создании разнообразных приложений моделирования и визуализации на единой концептуальной, методологической, инструментальной и программной основе. В самом деле, при применении ядра не требуется модифицировать его всякий раз для приспособления к частной задаче, семантике специфических объектов, их отношениям и возможным деталям программной реализации. Лежащие в его основе механизмы связывания и взаимодействия объектов позволяют составлять и интерпретировать сложные сценарии, возникающие в реальных задачах, без какого-либо уточнения семантики конкретных объектов приложения. Следовательно, в большинстве случаев создание приложения сводится к разработке прикладной библиотеки классов для представления используемых научных данных и для реализации методов моделирования и визуализации, специфичных для конкретной предметной области, а также к реализации графического интерфейса пользователя. Принимая во внимание преимущества объектно-ориентированного программирования,

реализацию прикладных библиотек классов данных и алгоритмов можно значительно упростить за счет неформального использования принципов инкапсуляции, наследования и полиморфизма.

Обсудим самые общие вопросы создания приложений. При использовании рассмотренного объектно-ориентированного ядра разработка приложения сводится к проведению объектного анализа рассматриваемой предметной области, выделению специфичных для нее научных данных и алгоритмов, а также к проектированию и реализации специализированной библиотеки классов для представления этих объектов и манипулирования ими.

Разработка конкретного приложения может проходить в соответствии со следующей итерационной схемой:

- идентификация классов научных данных и алгоритмов, участвующих в постановке и решении задач моделирования и визуализации, на основе выделения ключевых абстракций предметной области;
- определение семантики выделенных классов, установление отношений общности между ними, таких как «общее–частное», «целое–часть», построение объектных классификаций (иерархий наследуемых классов) для данных и алгоритмов с вершинами в суперклассах ***Data*** и ***Algorithm*** соответственно;
- установление отношений использования между экземплярами классов на основе анализа семантической зависимости и представление их в виде типизированных входных, выходных и смешанных связей, выделение одиночных и множественных связей;
- идентификация специфических атрибутов, свойств и поведения частных объектов, реализация их классов, для конкретных алгоритмических классов реализация специфических методов выполнения;
- составление и тестирование различных сценариев работы с выделенной системой классов, возможное ее критическое переосмысление, выделение новых типов объектов и, как следствие, возврат к предыдущим этапам разработки. В исключительных ситуациях возможно расширение функциональности классов ***Scene***, ***Object***.

Законченное приложение включает объектно-ориентированное ядро, инвариантное по отношению к различным задачам и предметным областям, расширяемые библиотеки классов, специфичные для рассматриваемой прикладной области, и унифицированный графический интерфейс пользователя. В качестве стандартных инструментальных средств, допускающих перенос разрабатываемых приложений на различные аппаратные платформы с минимальными усилиями, могут использоваться язык программирования Си++, графическая библиотека OpenGL и стандартные интерфейсные библиотеки (такие как Motif, Qt, Gtk).Общая открытая модульная архитектура приложения приведена на рис. 3. Функциональность законченного приложения определяется главным образом подключаемыми прикладными библиотеками и семантикой их классов.

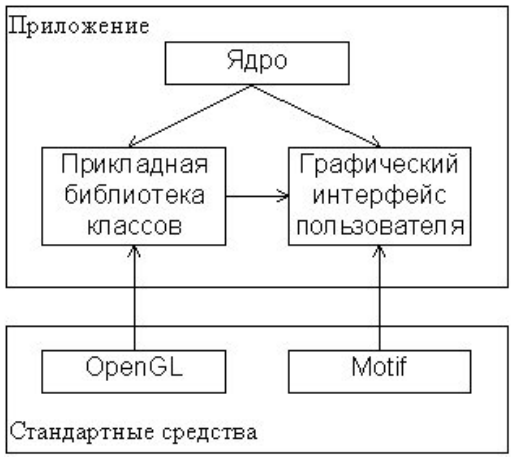

Рис. 3. Архитектура приложения

4. Система научной визуализации общего назначения OpenMV

4.1. Организация и состав системы

В соответствии с предложенной методологией на платформе UNIX/X Window была реализована система визуализации общего назначения OpenMV (Open Modeler&Visualizer). Система позволяет пользователю воспроизводить различные виды физических явлений посредством визуализации и анимации предварительно подготовленных данных моделирования. Система обеспечивает визуализацию распространенных типов данных с использованием широкого набора методов в рамках произвольных сценариев, задаваемых пользователем. Система

включает описанное выше объектно-ориентированное ядро, расширенное библиотеками научных данных и методов научной визуализации, и унифицированный графический интерфейс пользователя.

Библиотека научных данных построена как набор классов, производных от ядра и реализующих различные научные понятия, обычно используемые в приложениях вычислительной механики. Библиотека включает классы для манипулирования различными типами данных, такими как структурированные и неструктурированные поверхностные и объемные сетки, ломаные, наборы точек, физические поля, цветовые палитры, шкалы, наборы маркеров, ортогональные сечения, виды. Библиотека организована как единая иерархия конкретных и абстрактных классов, наследуемых от класса ***Data*** ядра системы. Фрагмент данной иерархии представлен на рис. 4.

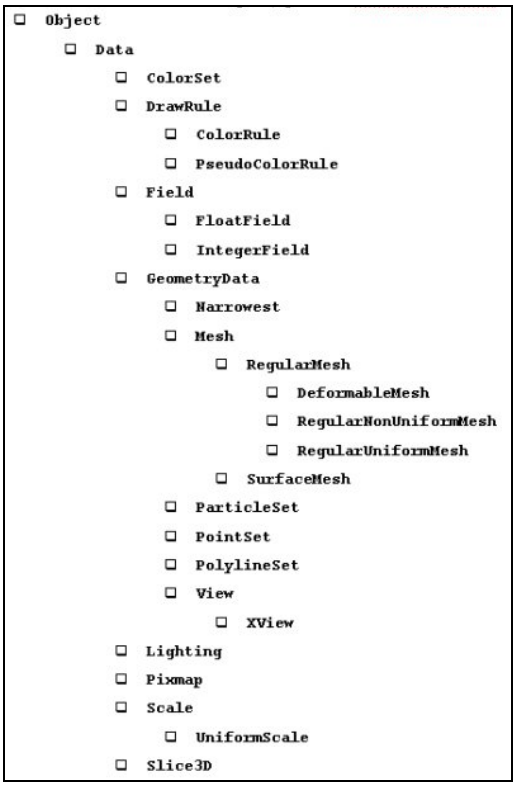

Рис. 4. Выдержка из иерархии классов библиотеки научных данных

Используемые абстракции позволяют манипулировать унифицированным образом различными геометрическими и топологическими типами данных. Все классы геометрических данных имеют обобщенную реализацию, позволяющую использовать их как в двумерных, так и в трехмерных случаях и обеспечивающую их стандартное отображение в контексте OpenGL. Класс физических полей также поддерживает обобщенное многокомпонентное представление, которое может соответствовать скалярным, векторным и тензорным полям, заданным на различных типах многомерных геометрических данных.

Методы отображения позволяют показывать произвольные геометрические данне как наборы точек, каркасное или твердотельное представление в соответствии с различными топологическими элементами (вершинами, ребрами, гранями). Закраска элементов может быть выполнена фиксированными цветами (правило закраски) или в зависимости от значений поля, заданных на выбранных элементах, а также указанной шкалы и цветовой палитры (правило псевдозакраски).

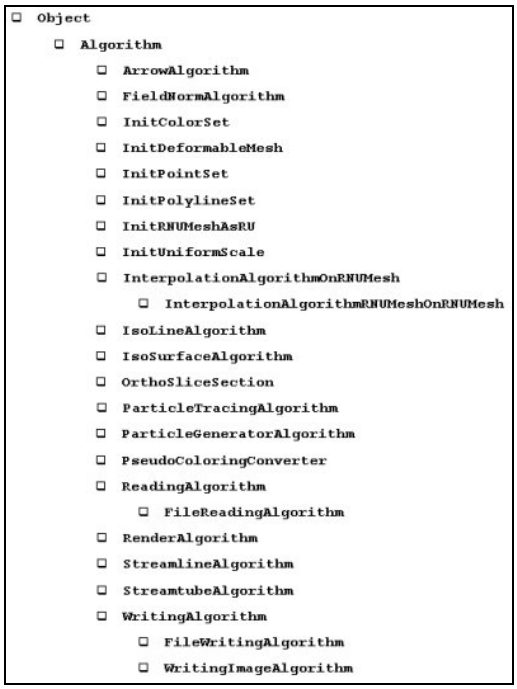

Рис. 5. Выдержка из иерархии классов библиотеки методов научной визуализации

Библиотека методов научной визуализации представляет собой набор производных от классов ядра алгоритмических классов, которые реализуют широко распространенные методы и способы, предназначенные для визуализации задач вычислительной механики. Библиотека включает классы для интерполяции полей, построения изолиний и изоповерхностей, конструирования линий и трубок тока, трассировки траекторий

частиц, конструирования ортогональных сечений, маркеров и вычисления норм полей. Библиотека построена как единая иерархия конкретных и абстрактных классов, наследованных от класса ***Algorithm*** ядра. Фрагмент иерархии классов представлена на рис. 5.

Реализованный набор алгоритмов позволяет интерполировать поля, заданные на структурированных сетках, на любые другие структурированные сетки или произвольные геометрические данные; извлекать изолинии из скалярных полей, заданных на поверхностных сетках; извлекать изоповерхности из скалярных полей, заданных на структурированных объемных сетках; конструировать линии и трубки тока для стационарных векторных полей, заданных на структурированных сетках; трассировать траектории частиц для нестационарных векторных полей, заданных на структурированных объемных сетках; конструировать ортогональные сечения полей, заданных на структурированных объемных сетках; конструировать маркеры для скалярных и векторных полей, заданных на произвольных геометрических данных, вычислять различные нормы для полей.

Кроме методов визуализации, данная библиотека обеспечивает специальные классы для растеризации геометрических данных в графическом контексте OpenGL в соответствии с заданными правилами закраски и псевдозакраски, а также вспомогательные классы для сохранения/восстановления объектов сцены в файлы/из файлов внутреннего формата и сохранение созданных изображений сцены в растровом формате PPM.

Полнота включенных библиотек классов соответствует функциональности системы общего назначения и обеспечивает ее развитые возможности для визуализации широкого круга задач вычислительной механики, включая гидродинамику, химическую кинетику, релятивистскую механику и т.п.

Интерфейс системы предоставляет унифицированные диалоги для составления и интерпретации сложных сценариев, а также для редактирования объектов, окна просмотра трехмерной сцены, меню и панели инструментов. Для ясности и удобства использования диалоги сцены отображают иерархию зарегистрированных в системе прикладных классов, диаграмму текущего сценария и отфильтрованный список объектов сценария. Для создания, редактирования и связывания отдельного объекта внутри сценария пользователь устанавливает требуемые значения их открытых атрибутов и соединений через соответствующий унифицированный диалог редактирования. Унификация диалога достигается за счет применяемой спецификации атрибутов и соединений как членов классов данных и алгоритмов. Применяемая унификация интерфейса не препятствует его специализации для частных целей. Снимок экрана системы представлен на рис.6.

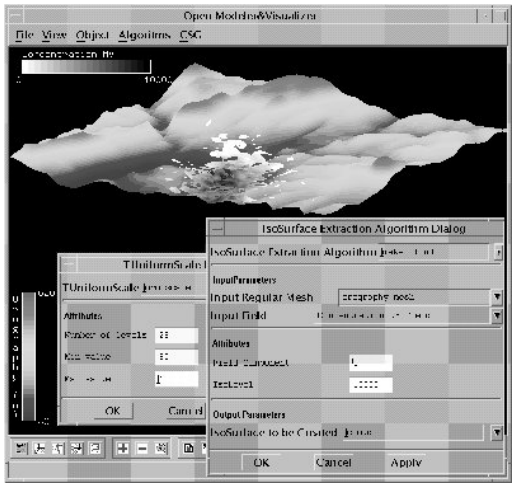

Рис. 6. Снимок экрана системы OpenMV с полученным изображением и диалогами редактирования объектов

Через интерфейс пользователь выбирает соответствующий сценарий для изучаемой задачи, корректирует его параметры (уточняет параметры постановки задачи, настраивает используемые методы визуализации, устанавливает подходящие виды сцены, размещает источники света и т.д.), активизирует его и наблюдает возникающие и визуализируемые результаты как анимированное трехмерное изображение сцены. Итоговое изображение генерируется и анимируется посредством повторяющейся интерпретации выбранного сценария. Интерпретация может либо непосредственно инициироваться пользователем, либо осуществляться в результате некоторых изменений в сценарии, приводя к автоматическому обновлению итоговой сцены. Эта возможность позволяет интерактивно в визуальном режиме изучать решаемую задачу в зависимости от параметров ее постановки. Квалифицированные пользователи могут составлять собственные сценарии, интегрируя различные

компоненты моделирования и визуализации для изучения сложных процессов и явлений.

В заключение перечислим основные возможности системы OpenMV. Как система визуализации общего назначения OpenMV обеспечивает:

- визуализацию широко распространенных геометрических и физических типов данных, включая широкое множество сеток различной топологической организации, скалярных, векторных, тензорных полей, шкал, сечений и т.п.;
- использование различных методов визуализации, включая извлечение изолиний и изоповерхностей, конструирование линий и трубок тока, отображение полей с применением методов стрелок, частиц, псевдозакраски;
- визуализацию и анимацию сложных композиционных сцен;
- интерактивное манипулирование отдельными объектами сцены;
- написание и повторное использование сценариев для типовых задач визуализации;
- сохранение используемых данных, методов и сценариев в файлах.

Благодаря развитым возможностям, а именно:

- комбинированию сценариев, определяемых пользователем, со встроенными;
- объектно-ориентированному визуальному программированию сложных сценариев;
- трехмерному многооконному интерфейсу;
- манипулированию с многокомпонентными полями;
- эволюции (адаптируемости, расширяемости) посредством расширения прикладных библиотек данных и алгоритмов,

система предоставляет удобные и эффективные средства для построения специализированных интегрированных приложений моделирования и визуализации, включая системы CAD/CAM/CAE. Открытая архитектура системы позволяет расширять функциональность путем включения новых прикладных библиотек классов для представления научных данных и алгоритмов моделирования, специфичных для разрабатываемых приложений. Унифицированный пользовательский интерфейс может быть относительно просто модифицирован для приспособления к специфике приложений. В следующем разделе мы обсудим эволюционные возможности системы OpenMV.

4.2. Некоторые примеры визуализации задач вычислительной механики

Рассматриваемая система OpenMV может применяться для изучения широкого класса задач вычислительной механики с использованием различных сценариев визуализации. Построение конкретного сценария осуществляется путем конструирования экземпляров соответствующих научных данных и алгоритмов визуализации и установления связей между ними в соответствии с общей дисциплиной связывания объектов. В определенном смысле получаемое представление сценария напоминает диаграмму потоков данных в системах визуализации общего назначения, таких как AVS, Data Explorer, IRIS Explorer [9], за исключением того, что в нем непосредственно участвуют и сами преобразуемые данные. Интерпретация сценария выполняется путем повторяющейся последовательной активизации алгоритмов, которые должны быть предварительно упорядочены в соответствии с установленными отношениями зависимости между объектами сцены. Будучи последовательно активизированы, алгоритмы сценария конструируют новые объекты и обновляют существующие, приводя к получению итогового изображения в статическом режиме и к его анимации в динамическом режиме. Однажды сконструированный сценарий может быть впоследствии снова и снова применен для изучения аналогичной задачи.

Разработанная система визуализации была успешно использована для изучения актуальных научных проблем, связанных с математическим моделированием сложных физических явлений, таких как:

- экологические последствия процесса горения метана;
- моделирование процессов переключения в полупроводниковых кристаллах с оптической бистабильностью;
- обтекание открытой каверны набегающим потоком сверхзвукового газа.

Более подробную информацию о данных приложениях, дополнительные изображения и фильмы, описания других задач, изученных посредством OpenMV,

можно найти в [10]. Строгая математическая постановка задач и детали численного моделирования приводятся в [11, 12].

Рассмотрим пример сценария визуализации результатов моделирования сверхзвукового потока вязкого газа над открытой прямоугольной каверной. Входными данными, поступающими от программы моделирования, являются регулярные сетки и заданные на них поля давления и скорости.

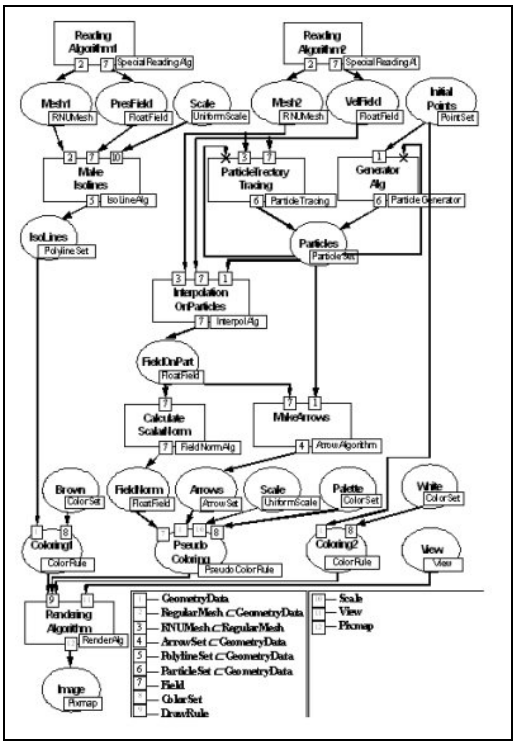

Рис. 7. Диаграмма сценария визуализации, предназначенного для изучения сверхзвукового потока вязкого газа над открытой прямоугольной каверной

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 псевдозакраски исходное поле скорости интерполируется на набор частиц и вычисляется норма поля, соответствующая модулю скорости поля. Для информативности дополнительно визуализируется поле давлений с применением контурного метода. Правила отрисовки 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 рис.7. Полученное итоговое изображение показано на рис. 8.

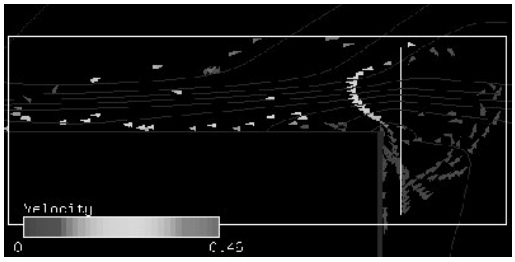

Рис. 8. Итоговое изображение для задачи динамики сверхзвукового газа

Необходимо отметить, что большинство методов визуализации допускает обобщенную реализацию, не зависящую от конкретных типов входных данных. Например, в приведенном примере алгоритм выделения изолиний реализован с применением абстракции регулярных сеточных данных, на которых задано исходное скалярное поле. В этом случае алгоритм применим в равной степени 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. Данное обстоятельство оказывается существенным для функционального расширения открытой системы, поскольку добавление новых производных типов данных и алгоритмов не приводит к необходимости создания новых версий ранее реализованных классов.

Другая задача связана с оценкой последствий процесса горения метана, имеющей важное экологическое значение. Фактор распространения потока воздуха является существенным для адекватного моделирования протекающих процессов химической кинетики для многокомпонентной реагирующей смеси. Рис. 9 демонстрирует потоки газа, воспроизведенные посредством линий тока и стрелок. Полученные линии тока и стрелки окрашены в соответствии с амплитудой скорости.

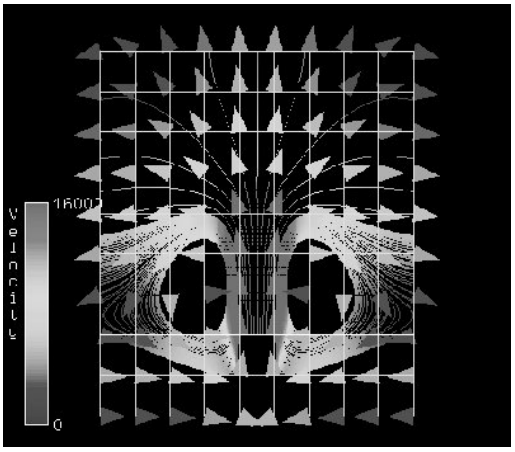

Рис. 9. Итоговое изображение для задачи химической кинетики

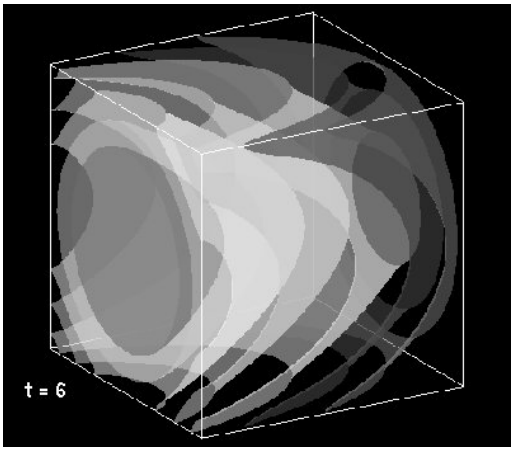

Рис. 10. Итоговое изображение для задачи оптики

Третья задача, связанная с моделированием процессов переключения в полупроводниковых кристаллах с оптической бистабильностью, является важной для конструирования новых полупроводниковых устройств, являющихся базой для будущих оптических ЭВМ и сетей. Процесс перехода распределения носителей заряда в оптических кристаллах в устойчивое состояние исследуется в зависимости от интенсивности лазерного излучения и диффузии носителей. Данный процесс визуализируется методом извлечения изоповерхностей (см. рис. 10). Прозрачные изоповерхности окрашены с использованием радужной палитры.

5. Некоторые специализированные интегрированные приложения моделирования OpenMV

5.1. Динамика релятивистских струн

Для изучения и иллюстрации теоретически установленных явлений динамики релятивистских струн [13] было разработано и реализовано в ОС UNIX/X Window специальное интегрированное интерактивное приложение как специализация системы визуализации общего назначения OpenMV. Данное приложение позволяет пользователю воспроизводить различные типы явлений динамики открытых и замкнутых струн посредством их моделирования, визуализации и анимации.

Поскольку стандартная версия OpenMV предназначена для научной визуализации задач вычислительной механики и предоставляет развитые библиотеки научных данных и методов визуализации, данное специализированное приложение было разработано как эволюция стандартной версии OpenMV, расширенное прикладными библиотеками классов для задач динамики струн.

Разработанная библиотека динамики струн включает специальные и универсальные классы алгоритмов, предназначенных для:

- конструирования открытых и замкнутых струн с полями распределения плотности энергии и момента по заданным опорным кривым произвольной геометрии;
- модификации структурированных поверхностных сеток путем добавления лент ячеек, основанных на заданной ломаной (для реконструкции мировых листов по обновленным открытым и замкнутым струнам);
- конструирования мировых листов открытых и замкнутых струн с полями распределения плотности энергии и момента непосредственно по заданным опорным кривым;
- интегрирования математических функций неявными методами Рунге–Кутта;

а также вспомогательные классы данных для:

- преобразования полей распределения плотности энергии и момента, заданных

на геометрических кривых, в математические функции;
- манипулирования с опорными кривыми (NURBS).

На рис. 11 приведен пример сценария, предназначенного для изучения задач динамики релятивистских струн. Данный сценарий определяет совокупность данных и алгоритмов, позволяющих пользователю устанавливать и редактировать опорные кривые, конструировать открытые и замкнутые струны, реконструировать соответствующие мировые листы и визуализировать полученные результаты одновременно в нескольких видах как анимированные трехмерные геометрические сцены, закрашенные в соответствии с распределениями плотности энергии и момента. Пользователь может легко модифицировать данный сценарий путем изменения значений атрибутов объектов, замены отдельных методов моделирования и визуализации другими или даже путем замены целых фрагментов, чтобы приспособить сценарий для конкретных целей.

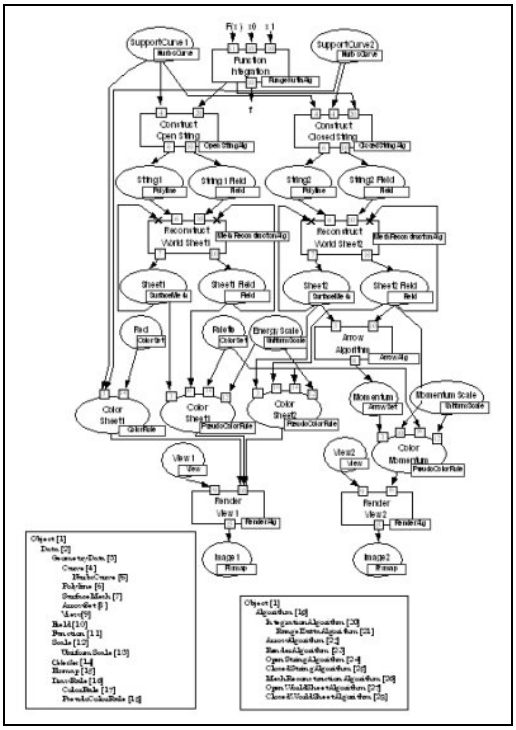

Рис. 11. Диаграмма сценария для изучения задач динамики релятивистских струн с иерархиями используемых классов данных и алгоритмов

Снимок экрана законченного интегрированного приложения приведен на рис. 12 для иллюстрации принципов унификации пользовательского интерфейса. На рисунке показаны изображения, полученные в результате интерпретации описанного выше сценария.

5.2. Моделирование генератора с инерционной нелинейностью

Следующий пример иллюстрирует, каким образом сложный сценарий, интегрирующий различные компоненты для геометрического моделирования, решения математических задач, визуализации и отображения, может составляться и интерпретироваться внутри приложения, построенного на основе описанного выше ядра, расширенного соответствующими прикладными библиотеками классов в соответствии с общей методологией. Представленный сценарий предназначен для изучения поведения генератора с инерционной нелинейностью. Моделью данного генератора является система обыкновенных дифференциальных уравнений (ОДУ) со странным аттрактором [14].

Для изучения поведения данной модели и определения притягивающей области в фазовом пространстве удобно сконструировать фазовые траектории, соответствующие различным начальным условиям задачи Коши. Множество начальных условий может быть задано посредством конструирования сложного твердого тела в интересующей области пространства в соответствии с конструктивной твердотельной геометрической моделью (CSG) [15] и генерирования точек на его граничном представлении. Затем множество задач Коши может быть решено численным методом, и полученные результаты могут быть визуализированы посредством окрашивания фазовых траекторий в соответствии с величинами производной. Для выявления свойств функции ОДУ может быть также сконструирована изоповерхность, соответствующая заданной величине производной. Данная изоповерхность выделяется из поля величин производных, определяемого по правой части системы ОДУ, реализуемой классом Attractor $\subset$ Function. Описанный сценарий моделирования и визуализации показан на рис. 14, полученное итоговое изображение — на рис. 13.

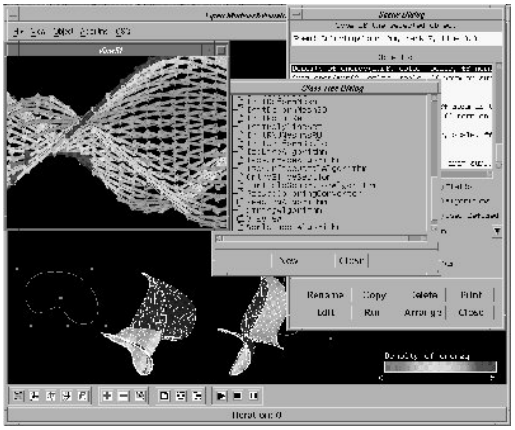

Рис. 12. Снимок экрана приложения для изучения задач динамики релятивистских струн с полученными изображениями

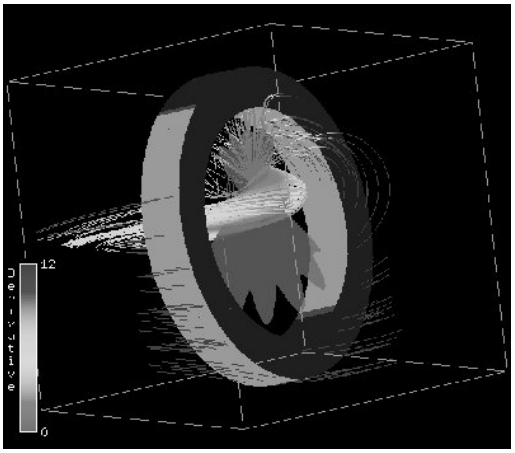

Рис. 13. Итоговое изображение для задачи моделирования генератора с инерционной нелинейностью

Обсудим достигнутые преимущества нашего подхода, ограничиваясь фрагментом сценария, связанным с геометрическим моделированием. CSG модель определяет сложное тело в терминах теоретико-множественных операций (объединение, вычитание, пересечение) над элементарными телами (параллелепипед, сфера, цилиндр, конус, призма, тор). CSG модель может быть естественным образом реализована в рамках предложенной методологии. Элементарные тела и теоретико-множественные операции должны рассматриваться как пассивные объекты, представленные классами ***CSGElement*** $\subset$ ***CSGData*** $\subset$ ***Data*** и ***CSGOperation*** $\subset$ ***CSGData*** $\subset$ ***Data*** соответственно. Геометрические элементы не имеют связей. Операции имеют пару входов, через которые они связываются с операндами. Поскольку операндами могут являться как элементарные тела, так и сами операции, данные связи имеют тип CSGData.

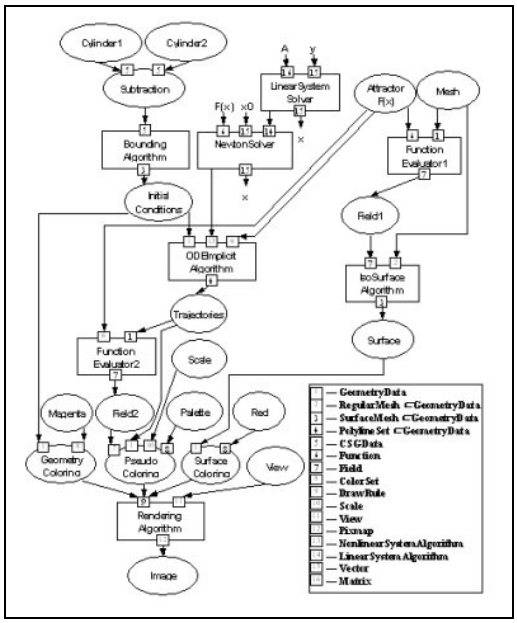

Рис. 14. Диаграмма сложного сценария моделирования и визуализации, предназначенного для изучения поведения генератора с инерционной нелинейностью

Взаимодействуя друг с другом, как CSG элементы, так и операции способны классифицировать точки по принадлежности к соответствующим геометрическим телам. Для элементов подобная классификация основывается на простейших геометрических соотношениях, приписанных данному типу примитива, для операций — на логическом анализе результатов, полученных для каждого операнда. Результирующая операция соответствует сложному твердому телу, обладает необходимой функциональностью для классификации точки и может использоваться как часть более сложных сценариев.

Обсудим фрагмент сценария, связанный с численным решением нелинейных дифференциальных уравнений. Неопределенный характер решения, связанный со свойствами жесткости, смешанным типом уравнений, их структурной разреженностью, приводит к необходимости изменения отдельных методов решения и применения различных вычислительных стратегий. Например, при решении жестких обыкновенных дифференциальных уравнений (ОДУ) обычно применяются неявные методы, реализация которых связана с решением нелинейных алгебраических уравнений на каждом шаге временной дискретизации [16].

Для решения последних методами ньютоновского типа необходимо применить тот или иной метод решения систем линейных алгебраических уравнений. Существенно, что выбор каждого метода решения может осуществляться независимо друг от друга и в конечном итоге определяется численными особенностями вспомогательных задач.

Описанная редукционная схема решения систем ОДУ естественно реализуется в рамках обсуждаемой методологии. Методы решения систем ОДУ, нелинейных и линейных алгебраических уравнений представляются, соответственно, классами ***ODEImplicitAlgorithm***, ***NewtonSolver***, ***LinearSystemSolver***, производными от ***Algorithm***. Входами перечисленных алгоритмов являются данные, участвующие в математических постановках соответствующих задач, выходами — результаты их численного решения. Заметим, что входы и выходы задействованы только у основного алгоритмического объекта — метода решения системы ОДУ, который активизируется в результате наступления соответствующих событий в сцене. Отношения использования между остальными объектами-алгоритмами передают редукционные связи. Для алгоритмических объектов, участвующих в них, входы и выходы не определены, поскольку эти объекты используются друг другом, но не активизируются непосредственно сценой.

Будучи разработанным в рамках системы OpenMV, расширенной классами для представления CSG модели и решения вычислительных задач, данный сценарий может быть гибко приспособлен к решению частных задач посредством модификации отдельных фрагментов, данных и алгоритмов. Единственное ограничение состоит в том, что изменяемые объекты должны удовлетворять установленной типизации связей. Данное свойство гарантирует, что тот же самый сценарий может корректно использоваться в комбинации с новыми типами данных и алгоритмов, расширяющими разработанные библиотеки классов. Объединяя достигнутые возможности объектно-ориентированной разработки прикладных библиотек классов, составления и модификации сложных сценариев, OpenMV обеспечивает широкие возможности повторного использования как разработанных компонентов программного обеспечения, так и составленных сценариев.

6. Заключение

Таким образом, рассмотрена объектно-ориентированная методология для построения интегрированных приложений моделирования и визуализации. Данная методология обеспечивает общую дисциплину и гибкие программные средства для унифицированной разработки приложений в существенно различных научных и промышленных областях на единой концептуальной, инструментальной и программной основе. Разработанные приложения имеют общую открытую модульную архитектуру, которая включает полностью инвариантное ядро, расширяемые библиотеки прикладных классов и графический интерфейс пользователя. Функциональность созданного приложения определяется главным образом полнотой включаемых прикладных библиотек и семантикой их классов.

В соответствии с предложенной методологией была реализована система научной визуализации общего назначения OpenMV под ОС UNIX/X Window. Система обеспечивает визуализацию и анимацию широко распространенных научных данных стандартными методами визуализации в рамках произвольных сценариев, определяемых пользователем. OpenMV обеспечивает представление различных типов научных данных, таких как сетки, поля, шкалы, палитры, изображения, а также реализует различные методы визуализации и отображения, включая методы извлечения изолиний и изоповерхностей, построения линий тока, маркеров, конструирования сечений, псевдозакраски. Открытая архитектура системы позволяет разрабатывать многочисленные интегрированные интерактивные приложения для изучения разнообразных физических задач.

Область потенциальных приложений предложенной методологии чрезвычайно широка и может охватывать математическое моделирование, интегрированные системы CAD/CAM/CAE, технологии моделирования виртуальной реальности, анимационные системы. В дальнейшем планируется сосредоточить работу на создании законченных приложений в близких междисциплинарных областях.

Литература

1. STEP Product Data Representation and Exchange. ISO CD 10303-42, 1994.
2. The Virtual Reality Modeling Language. ISO/IEC 14772-1:1997.
3. OpenGL programming guide: the official guide to learning OpenGL, version 1.1/ Mason Woo, Jackie Neider, Tom Davis, 2nd ed. Addison Wesley, 1996.
4. MPI: A Message-Passing Interface Standard, Message-Passing Interface Forum, May 5, 1994.
5. Object-Oriented and Mixed Programming Paradigms: new directions in computer graphics / Peter Wisskirchen (ed.), Springer, 1996.
6. Семенов В.А. Объектная систематизация и парадигмы вычислительной математики. // Программирование. 1997, № 4, с. 14–25.
7. S.C. Bailin, An Object-Oriented Requirements Specification Method, Comm.ACM, Vol. 32, No. 5, 1989, pp.608–623.
8. G. Booch, Object Oriented Design With Applications, Behjamin/Cummings Publishing Company, 1991.
9. S. Belien et al. Comparison of Visualization Techniques and Packages with applications to plasma physics. http://www.sara.nl/Consumer.Report/ Report.html
10. The ISP RAS Scientific Visualization Group Home Page, http://www.ispras.ru/~3D
11. Chetverushkin B.N., Iakobovski M.V., Kornilina M.A., Malikov K.Yu., Romanukha N.Yu. Ecological after-effects numerical modelling under methane combustion. In: Mathematical Models of Non-Linear Excitations, Transfer, Dynamics, and Control in Condensed Systems and Other Media. Ed. By L.A Uvarova, A.E. Arinstein, and Latyshev. Plenum Press, New York, 1998.
12. Yu. N. Karamzin, T.A. Kudryashova, S.V. Polyakov and I.G. Zakharova. Simulation of 3D optical bistability problem on parallel computer systems. Matematicheskoe modelirovanie, in «Fundamental physical and mathematical problems and modelling of technical and technological systems» (ed. L.A.Uvarova), pp. 117-124. Published by Moscow state technology university «Stankin», Moscow, 1999.
13. V. Burkin, S. Klimenko, I. Nikitin. Visualization and animation of relativistic string dynamics, Programming and computer software, 1998, V.24, —¹ 6, pp. 320–328.
14. Анищенко В.С. Сложные колебания в простых системах. — М.: Наука, 1990.
15. Requicha A.A.G. and Voelcker H.B., Solid Modeling: Current Status and Research Directions. — IEEE CG.A., Vol.3, No.7, 1983, pp.25–37.
16. Арушанян О.Б., Залеткин С.Ф. Численное решение обыкновенных дифференциальных уравнений на Фортране. — М.: изд. МГУ, 1990.