УДК 519.6

РАЗРАБОТКА ПОДХОДА К СОЗДАНИЮ СПЕЦИАЛИЗИРОВАННЫХ СИСТЕМ ВИЗУАЛИЗАЦИИ ДЛЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ НАУЧНЫХ ВЫЧИСЛЕНИЙ

А. И. Зенков
(ИММ УрО РАН)

Описывается один из возможных подходов к разработке средств научной визуализации для представления результатов моделирования с использованием высокопроизводительных вычислений, а также созданные на основе этого подхода унифицированные средства разработки специализированных систем научной визуализации.

Введение

Существуют несколько моделей научной визуализации, однако малоосвещенными остаются вопросы построения систем визуализации и выработки возможных критериев их оценки.

Работа содержит некоторые итоги исследований и разработок в области специализированных систем визуализации, предназначенных для представления данных, получаемых при численном решении ряда задач оптимального управления и дифференциальных игр [1—3]. В сотрудничестве с математиками разработан набор систем, хорошо учитывающих специфику задач и ориентированных на выделение важных для исследователя особенностей математических объектов. Существенным результатом этого сотрудничества можно считать созданный единый вычислительно-визуализационный комплекс, который позволил получить новые результаты в предметной области математики. Тем самым на практике проверены положения, связанные с теорией когнитивной визуализации [4, 5].

Однако отсутствие единого подхода к разработке программного обеспечения приводило к тому, что при реализации каждой новой постановки задачи приходилось заново создавать визуализационный модуль и реализовывать его связь с вычислительным модулем. Эта проблема послужила толчком к попытке унифицировать разработку специализированных систем визуализации.

Унифицированная система должна содержать модуль визуализации, общий для различных специализированных систем, и набор модулей, обеспечивающих описание и восстановление трехмерных сцен с учетом особенностей конкретных математических объектов. Для визуализации и исследования новых объектов пользователь-математик может разрабатывать собственные модули. Форматы входного потока данных такого модуля должны быть согласованы с соответствующей счетной программой, а выходного — с модулем визуализации.

Как указывалось выше, опыт работы с пользователями разработанных систем показал, что создаваемые средства визуализации должны отвечать требованиям когнитивной (способствующей мышлению) визуализации, так как именно за счет нее обеспечивается эффективная интерпретация результатов вычислений. Соответственно возникла необходимость в исследовании теоретических основ этого подхода.

В разд. 1 описывается авторский подход к построению систем когнитивной визуализации и требования к языку визуализации проектируемой унифицированной системы. В разд. 2 рассматривается структура программной реализации унифицированной системы визуализации.

1. Теоретические основы построения системы визуализации

В работе [2] предложено рассматривать графику как инструмент, не только выполняющий иллюстративную роль, но и способный стимулировать мышление исследователей и выявлять новые возможные подходы к предмету изуче-

ния. Компьютерную графику такого типа называют *когнитивной*, отличая ее от так называемой *иллюстративной*, которая фиксирует конечные результаты исследования. Понятие когнитивной визуализации рассматривается в [1]. Там же указывается на необходимость создавать новые методы представления данных и способы взаимодействия с изображением для разработки эффективных специализированных систем когнитивной графики.

Следует отметить важность когнитивной визуализации для представления результатов параллельных вычислений как на этапе отработки вычислительной модели, когда самому исследователю полностью не ясны особенности математических объектов, так и для визуализации в процессе вычислений по ходу "большого" параллельного счета. Можно говорить о когнитивности как об основном требовании при создании специализированных систем визуализации.

При изучении феномена когнитивной графики активно используется понятие метафоры. Всякая визуализация основана на сопоставлении изучаемых сущностей и визуальных объектов, их представляющих, т. е. на некоторой идее метафорического представления. Развитие ее служит основой любого языка визуализации [6]. Метафора осуществляет перевод (переформулирование) понятий математической модели с языка строгих математических конструкций, где возможности человеческой интуиции зачастую ограниченны, в визуальный язык (рис. 1). Как много новой информации об объекте изучения исследователь сможет получить из визуальной модели, определяется тем, насколько хорошо была разработана метафора.

Рис. 1. Построение унифицированного блока системы визуализации

Язык визуализации можно рассматривать как набор графических образов, предъявляемых наблюдателю. Словарь этого языка составляют виды отображения, используемые в той или иной системе [6]. Значащими единицами его являются как сами визуальные образы, так и изменения их графических и неграфических атрибутов. При разработке языков визуализации основными моментами должны быть правильность и естественность интерпретации визуальных текстов. Изучение восприятия текстов на языках визуализации привело к следующему выводу: успех систем визуализации связан, как правило, с визуализацией ограниченного числа наиболее важных сущностей и реализацией разумного набора функциональных возможностей.

Анализ требований пользователей, предъявленных ими при проектировании разработанных систем, показывает, что для построения унифицированной системы весьма удобен язык, основанный на фотореалистической трехмерной графике. Такой язык определяется набором возможных видов отображений с изменяемыми атрибутами отдельных объектов. Структурными единицами его являются геометрические конструкции из трехмерных тел и поверхностей, отображаемых на экране с учетом их атрибутов (цвет, освещенность, прозрачность и т. д.). Следует отметить, что указанный язык является целостным, так как выбранные визуальные параметры геометрических тел хорошо сочетаются, создавая единый визуальный объект.

Определенность, фиксированность и относительная простота языка визуализации дают ряд дополнительных преимуществ: разработчику проще строить метафору визуализации, а пользователю — работать в интерактивном режиме, при обмене научными данными проще описывать получаемые изображения.

Таким образом, сформулирован ряд требований к языку визуализации для обеспечения когнитивности получаемой графики, которые должны быть реализованы при разработке системы.

Исходя из описанного выше подхода к организации визуализации при построении конкретной системы потребуется разработка только

лишь новой метафоры визуализации. Поскольку язык визуализации предполагается фиксированный, перевод описания сцены в изображение будет выполняться всегда одинаково.

2. Описание программной реализации системы

Основная задача при создании проекта системы заключалась в том, чтобы обеспечить ее максимальную гибкость и настраиваемость, избежав при этом функциональной избыточности и сложности применения. Для достижения этой цели использована идея построения системы в виде набора модулей, взаимодействующих между собой по заранее определенной схеме и оговоренным форматам данных.

Архитектура системы (рис. 2) состоит из двух основных частей:

1) подсистемы обработки исходных данных;
2) модуля визуализации.

Рис. 2. Архитектура системы

Подсистема обработки исходных данных. Это наиболее гибко настраиваемая часть системы, в которой сосредотачивается основная работа по созданию конкретной ее реализации. Назначение подсистемы — из файлов с исходными данными получить описание трехмерной сцены.

При этом необходимо:

- обеспечить простое использование типовых алгоритмов трехмерной компьютерной графики (триангуляция поверхностей, децимация поверхностей, построение нормалей и т. д.);
- дать пользователям декомпозировать их собственный процесс создания сцены.

Это достигается за счет введения понятия модуля. *Модуль* — это часть общего процесса построения сцены, которая выделяется в отдельный алгоритм и может использоваться независимо.

В данный момент в системе поддерживается 2 формата модулей:

1. *Исполняемые файлы операционной системы.* Эти программы должны соответствовать правилам вызова. Они могут создаваться в любой среде разработки и даже являться скриптовыми программами операционной системы. Для их создания требуются минимальные навыки программирования. Прием и передача данных в них осуществляется через файловую систему. Программе-модулю на вход в качестве параметров командной строки передаются имя файла с входными данными и имя файла с выходными данными, куда модуль должен поместить результат своей работы.
2. *Java-классы,* реализующие соответствующий интерфейс. Модули этого типа могут быть более эффективными с точки зрения производительности, и при их создании у программиста есть возможность пользоваться вспомогательными классами для упрощения программирования рутинных операций ввода/вывода. Однако в данном случае потребуется знание объектно-ориентированного подхода к программированию и языка Java.

Каждый модуль должен быть описан (рис. 3) в конфигурационном файле системы. Для каждого модуля указывается его тип, имя программы или Java-класса (в зависимости от типа модуля) и тип файла данных, с которыми он может работать. Тип файла данных определяется двумя способами:

- сигнатурами (строками), содержащимися в начале файла;
- расширением имени файла.

```
<handlers>
  <handler
  code="tubesHandler"
  type="external"
  start-string="modules\tubes.zai\dotube.
  exe"
  signature="tbc1"/>
  <handler
  code="KumkovsTubesHandler"
  type="external"
  start-string="modules\tubes\Tubes.exe"
  signature="Tubes (2002-05-01)"/>
</handlers>
```

Рис. 3. Пример описания модулей

Для всех конфигурационных файлов используется формат XML. Он дает ряд преимуществ:

- легкую читабельность и модифицируемость пользователями любого уровня (простой текстовый формат);
- обеспечение синтаксической корректности описаний за счет дескриптивных схем, без создания проверочного кода;
- удобный API для доступа к данным из кода.

После подготовки модулей необходимо настроить конвейер обработки данных, т. е. последовательность, в которой модули будут вызываться. Это делается в специальной секции конфигурационного файла.

Для файлов данных определенного типа описывается последовательная цепочка вызовов модулей (рис. 4). Для файлов другого типа может быть описана своя цепочка. Таким образом, при запуске системы на конкретном файле с данными производится поиск соответствующей цепочки в конфигурационном файле и, если она найдена, последовательно вызываются все модули из цепочки. Результат работы предыдущего модуля передается на вход следующему. При этом в промежуточных файлах контролируется только их тип (сигнатура или расширение), чтобы гарантировать, что следующий модуль сможет работать с этим типом. Регламентируется только формат итогового файла. Это должно быть описание трехмерной сцены.

Схема работы подсистемы обработки исходных данных приведена на рис. 5, где D — файлы с данными; P — вызовы модулей; L — описания трехмерных сцен.

Следует отметить, что модули — это и есть существенная часть процесса визуализации в разрабатываемой системе. Алгоритмы модулей определяют метафору и способы визуализации конкретных математических объектов. Поэтому

```
<file_types>
  <file_type signature="tbc1">
  <handler_call handler_code="tubesHandler"
  sequence="1"/>
  <handler_call handler_code="KumTubesHand-
  ler" sequence="2"/>
  </file_type>
<file_types>
```

Рис. 4. Пример цепочки вызовов модулей для файлов данных определенного типа

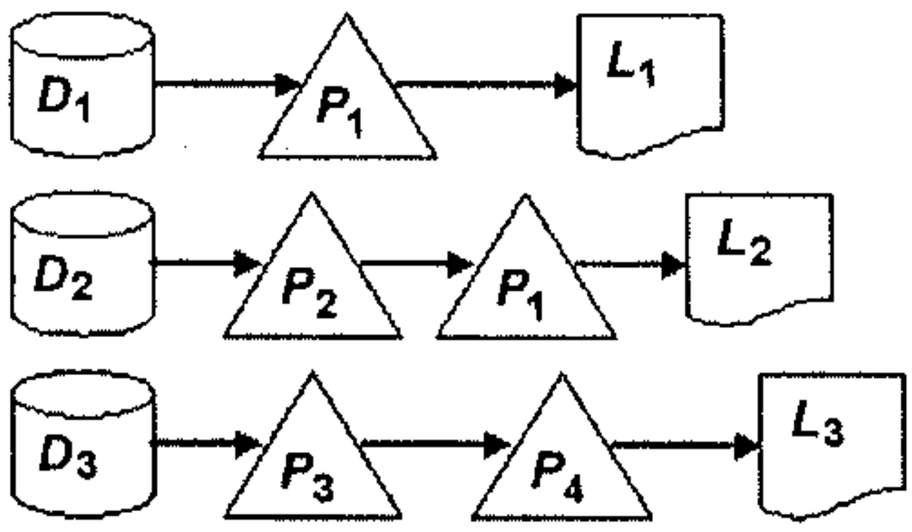

Рис. 5. Схема работы подсистемы обработки исходных данных

модули должны создаваться при непосредственном участии математиков-пользователей системы.

Описание сцены. Трехмерная сцена — это набор объектов и групп этих объектов. Всего в распоряжении пользователя 3 типа объектов:

1) поверхность — задается набором треугольников и нормалей в каждой точке поверхности;
2) ломаная линия — задается набором отрезков в пространстве;
3) точки — задаются своими координатами в пространстве.

Группы объектов могут включать в себя другие объекты и другие группы. В результате получается иерархическая структура объектов, где в качестве родительских элементов выступают группы (рис. 6). Например, поверхности представляются как группы смежных треугольников с подобранными нормалями в вершинах (для передачи гладкости поверхности). Трехмерные тела представляются своей границей.

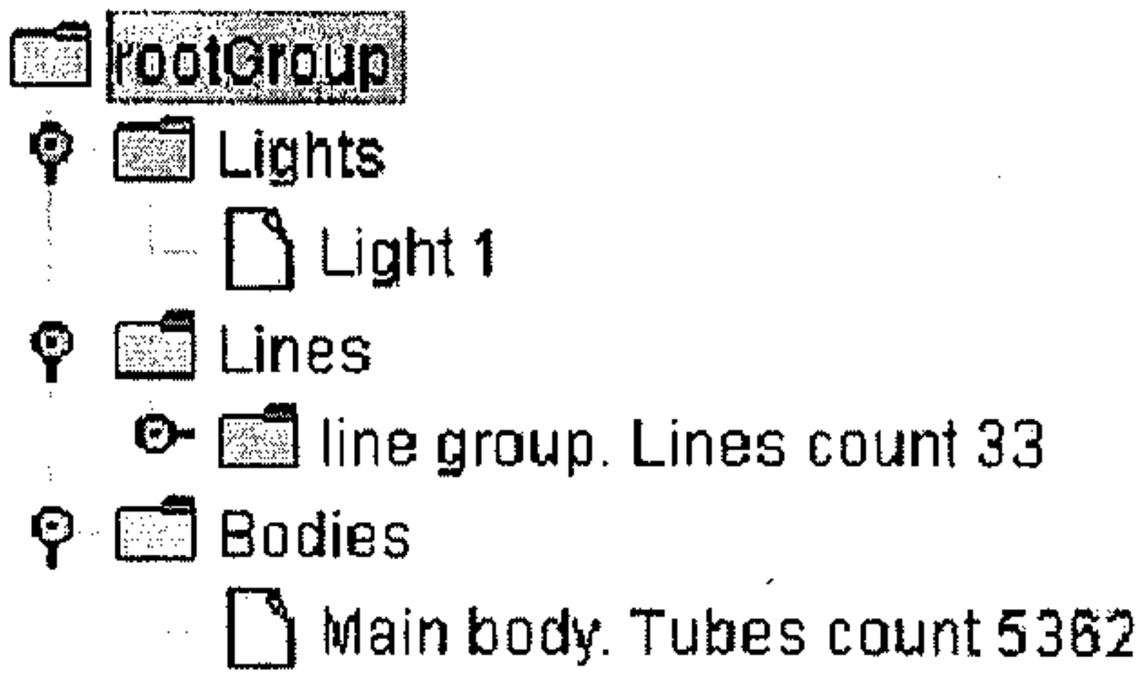

Рис. 6. Иерархическая структура объектов

Для каждого объекта пользователь независимо может задавать следующие атрибуты:

- тип отрисовки (*mesh/solid/wireframe*): *mesh* — режим, при котором отрисовывается проволочная модель объекта; *solid* — режим изображения объектов как трехмерных тел; *wireframe* — дополнительный режим "упрощенного" проволочного каркаса (задается дополнительно и может быть полезен при исследовании сцен с большим числом объектов);
- степень прозрачности;
- цвет (отметим, что каждая поверхность имеет две стороны, которые могут независимо менять свой цвет);
- наличие отсекающих плоскостей.

Окончательно сформированная сцена перед поступлением в визуализатор записывается в файл в определенном формате. В данный момент реализовано два формата. Первый — текстовый формат на основе XML: используется пока для тестовых целей, в перспективе — для реализации связи с форматом VRML. Второй — двоичный формат: хорошо подходит для хранения больших сцен, так как данные хранятся очень компактно. Второй формат имеет такую же структуру, что и первый, только вместо тэгов использует специальные открывающие и закрывающие сигнатуры. Оба формата очень просты и удобны для пользователя.

Визуализатор. Непосредственно вывод изображения и работу с изображением в интерактивном режиме осуществляет программа-визуализатор.

Этот модуль является наиболее трудоемким с точки зрения программирования, так как он должен обладать достаточно развитой функциональностью и обеспечивать интерактивную работу пользователя. Ранее при построении систем визуализации "с нуля" [1] вся работа с исходными данными и непосредственно вывод изображения реализовывались каждый раз заново. Основной проблемой являлось создание мощной и удобной программы интерактивной работы с изображением. При изложенном подходе при разработке новой системы уже имеется готовый модуль, который независимо может совершенствоваться и расширяться [7].

Модуль визуализации реализует пользовательский интерфейс управления трехмерными объектами и процедуру считывания наборов трехмерных объектов из файлов-описаний сцен, созданных ранее. Он обеспечивает интерактивные средства управления выводом изображения и изменения видов отображения информации. Пользователь имеет возможность быстро изменять положение и ориентацию объекта в трехмерном пространстве, масштаб вывода изображения, количество и положение источников света и секущих плоскостей, ракурс вывода, а также отображать объекты в различных режимах прозрачности.

В данный момент система находится в стадии опытной эксплуатации. На базе описанного подхода реализованы системы визуализации для двух задач.

Значительные результаты получены при построении визуализации множества функции цены и сингулярных поверхностей для задачи теории дифференциальных игр. Использование в предлагаемом подходе принципов специализированности позволило получить и увидеть такие детали поверхностей, которые невозможно было увидеть при помощи универсальных систем визуализации [8]. Эти детали дали возможность существенно продвинуться в исследовании данной проблемы.

Сделан также первый вариант визуализации для задачи оценивания множеств достижимости линейных многошаговых систем с фазовыми ограничениями. Специфика этой задачи заключается в многомерности получаемых геометрических объектов. Вырабатываются дополнительные виды отображения для эффективной работы с такими объектами.

В завершающей фазе разработки находятся несколько новых возможностей системы: редактируемая история, сохранение сцены, возможность создания параллельных иерархий объектов с помощью иерархических тэгов.

Основой модуля отображения трехмерной графики является высокоуровневая библиотека Java3D фирмы Sun. Выбор этой библиотеки позволил значительно уменьшить время, затраченное на разработку, и получить возможность использовать современное оборудование для виртуальной реальности. Java3D также является многоплатформенной библиотекой, что значительно расширяет круг пользователей системы и делает возможным ее использование через Интернет.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (код проекта 01-07-90210).

Список литературы

1. *Авербух В. Л., Зенков А. И., Исмагилов Т. Р., Манаков Д. В., Пыхтеев О. А., Юртаев Д. А.* Разработка специализированных систем научной визуализации // Алгоритмы и программные средства параллельных вычислений. 2000. Вып. 4. С. 3—23.

2. *Авербух В. Л.* Метафоры визуализации // Программирование. 2001. № 5. С. 13—17.

3. *Зенков А. И.* Разработка унифицированного модуля для специализированных систем научной визуализации // Алгоритмы и программные средства параллельных вычислений. 2001. Вып. 5. С. 3—23.

4. *Зенкин А. А.* Когнитивная компьютерная графика. М.: Наука, 1991.

5. *Zenkin Alexander A., Zenkin Anton A.* Cognitive computer visualization for scientific discoveries in mathematics, logic, philosophy, psychology, and education, and foundations of mathematics. http://www.com2com.ru/alexzen/Cognitive Reality/Cognitive Reality.html.

6. *Zenkov A. I.* The specialized systems of scientific off-line visualization // Тр. 11 Межд. конф. по компьютерной графике и машинному зрению "Графикон 2001". Нижний Новгород: НГГУ, 2001. С. 86—87.

7. *Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж.* Приемы объектно-ориентированного проектирования. Паттерны проектирования: Пер. с англ. С-Пб.: Питер, 2001.

8. *Sanglard H., Nageli H.-H.* An end user oriented platform for scientific visualization // IEEE Conference on Information Visualization (IV'97). 1997. P. 235.