УДК 004.921

В.В. КРАСНОПРОШИН, Д.И. МАЗОВКА

ЗАДАЧА ВИЗУАЛИЗАЦИИ АБСТРАКТНЫХ МОДЕЛЕЙ

The paper investigates an approach to visualization problem solution based on the use of standard hardware and software means. The authors propose formalisms, within which the problems of data organization and visualization are effectively solved.

Визуализация является важной компонентой компьютерного моделирования реальных процессов и явлений. Как правило, это трудоемкая процедура, которая часто требует значительных вычислительных ресурсов [1].

В данной статье рассматривается подход к визуализации, основанный на использовании стандартных аппаратно-программных средств (АПС) [2]. В рамках предлагаемых формализмов можно эффективно решать проблемы организации данных, а также реализации процесса визуализации в целом.

Анализ проблемы и постановка задачи

В общем случае задача визуализации связана с построением некоторой последовательности кадров, отображающих изменения в моделируемой сцене.

Рассмотрим процесс построения кадров с использованием стандартных аппаратно-программных средств [1, 2] (рис. 1).

Рендер представляет собой структуры данных и наборы инструкций для построения кадра. Драйвер унифицирует методы работы с графическим оборудованием, которое формирует двумерное изображение на экране монитора.

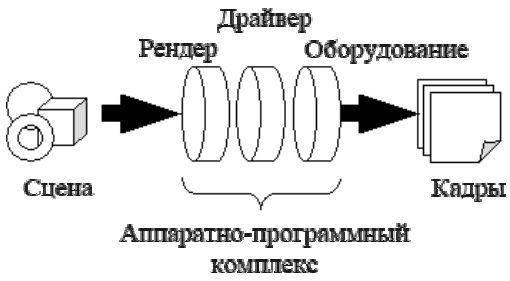

Рис. 1. Процесс визуализации

При использовании такого подхода необходимо решить две взаимосвязанные проблемы: рационального представления данных и эффективного использования стандартных механизмов.

Графическое оборудование является основным источником ограничений. В [2] показано, что производительность построения кадра существенно зависит от объема данных, частоты их отправки на оборудование, количества вызовов инструкций рендера и смен его состояния.

Рассмотрим процесс разработки систем визуализации, который, как правило, включает следующие этапы (см. например [3]) (рис. 2).

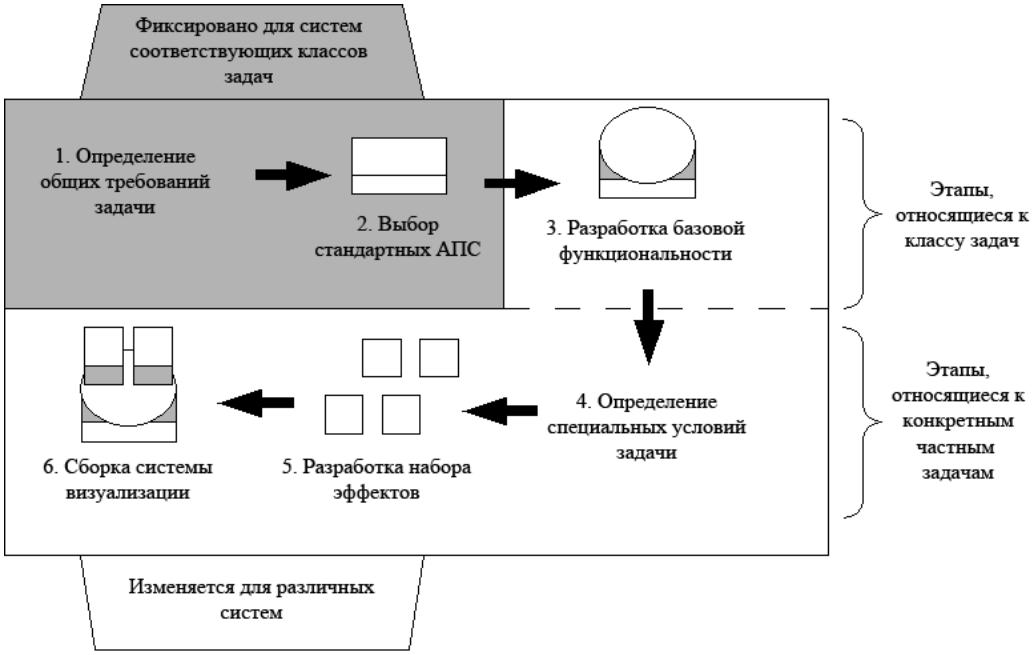

Рис. 2. Общий подход к разработке систем визуализации

Первые три этапа соответствуют свойствам всего класса решаемых задач, последующие отражают специфику конкретной задачи. Здесь под специальными условиями подразумевается выбор способов (эффектов) перевода абстрактных объектов модели в их визуальные образы.

Общим недостатком подхода является то, что при решении каждой новой задачи необходимо не только вновь реализовывать этапы 4–6, но и адаптировать к конкретной задаче базовую функциональность. Все это требует дополнительных затрат при использовании открытых пакетов или коммерческих систем [3].

Таким образом, существует потребность в формализации задачи визуализации с целью унификации процесса ее описания и в разработке алгоритма, позволяющего строить инвариантное относительно частных задач решение.

Предлагаем один из возможных путей решения данной проблемы.

Каноническое представление задачи визуализации

В содержательном смысле задача визуализации состоит в следующем. Пусть задана некоторая абстрактная модель M. Требуется разработать алгоритм, который на основе совокупности абстрактных объектов модели M строит двумерное изображение в виде их геометрических образов. Такие алгоритмы будем называть эффектами визуализации.

Проведем формализацию описанной задачи, для чего введем необходимые понятия и обозначения.

Множество $S = \{O_i, i = \overline{1,n}\}$ визуализируемых объектов модели *M* назовем сценой. Обозначим через $A = \{A_j, j = \overline{1,m}\}$, $W = \{W_k, k = \overline{1,l}\}$ и $E = \{E_k, k = \overline{1,g}\}$ атрибуты объектов, наборы визуальных образов и эффекты визуализации соответственно.

С учетом этого задачу визуализации можно записать в виде следующей онтологической модели [4]:

$$Z_0 =< O, A, W, E >,$$

где по заданным множествам *O*, *A*, *W* необходимо построить набор эффектов $E : \{O, A\} \to W$.

Для более детального описания задачи уточним понятие эффектов визуализации.

Определим предикат:

$$P_s(O_i) = \begin{cases} 1, A_s \subseteq O_i, \\ 0, A_s \not\subset O_i, \end{cases}$$

где $A_s = \{A_1^s, A_2^s, ..., A_{m_s}^s\}$ – подмножество атрибутов из *A*.

Обозначим через $C_s = \{O_i \in S \,|\, P_s(O_i) = 1\}$ подмножество сцены *S* (класс), соответствующее предикату P_s. Упорядоченное множество классов $T = (C_1, C_2, ..., C_q)$ назовем шаблоном.

Группой визуализации из сцены *S* по шаблону $T = (C_1, C_2, ..., C_q)$ назовем упорядоченное множество объектов сцены: $G_j = (O_1, O_2, ..., O_q)$, $O_i \in C_i, i = \overline{1,q}$.

Тогда эффект визуализации можно записать в виде тройки:

$$E_k =< \{P_i\}, T, EA >,$$

где отображение $EA : G \to W$ задает алгоритм перевода групп визуализации из сцены *S* по шаблону *T* в их геометрические образы.

Пусть теперь $S = \{O_i, i = \overline{1,n}\}$ – некоторая сцена и $E = \{E_k, k = \overline{1,g}\}$ – соответствующее ей множество эффектов, представление задачи визуализации в виде $Z =< S, E >$ назовем каноническим.

Нетрудно показать, что любая задача визуализации достаточно просто приводится к ее каноническому виду.

В частности, это можно сделать с помощью следующего алгоритма.

Алгоритм 1.

1. Выделить множество объектов задачи и их атрибуты.
2. Определить объекты, подлежащие визуализации.
3. Описать необходимые эффекты визуализации.
4. Для каждого эффекта визуализации:
   4.1. Задать множество классов эффекта.
   4.2. Задать шаблон эффекта.
   4.3. Описать алгоритм перевода.

Таким образом, нами построено каноническое представление задачи визуализации. Покажем, что это позволяет не только формализовать процесс ее описания, но и унифицировать процесс решения с использованием аппаратно-программных средств, вывести его методологию на более высокий уровень абстракции и построить алгоритм, инвариантный относительно частных задач.

Далее под задачей визуализации будет пониматься ее каноническое представление.

Формализация процесса построения кадров

Рассмотрим подробно процесс построения кадров с использованием стандартных АПС, изображенный на рис. 1.

Рендер, как интерфейс аппаратно-программного комплекса, задается множеством инструкций, посредством которых можно управлять визуализацией, структурами данных и исполняющим модулем. Формально его можно представить в виде тройки

$$R =< RT, RI, RE >,$$

где $RT = \{T_1, T_2, ..., T_m\}$ — набор типов данных, $RI = \{F_1, F_2, ..., F_n\}$ – набор инструкций, а *RE* – исполняющий модуль. В свою очередь множество *RI* разделяется на два непересекающихся подмножества инструкций: смены состояния *RIS* и построения (отрисовки) кадра *RID*.

Общая схема работы с рендером включает следующие основные этапы:

• представление визуализируемых данных типами *RT*: $\{D_1, D_2, ..., D_k\}$;

• загрузка данных с помощью инструкций *RIS* в исполняющий модуль *RE*;

• последовательный вызов инструкций *RID*.

Таким образом, для визуализации сцены $S = \{O_1, O_2, ..., O_n\}$ необходимо последовательно решить три подзадачи: множество ее объектов представить типами рендера *RT*, построить последовательность инструкций, соответствующую множеству эффектов *E* и реализовать ее исполняющим модулем *RE*.

Рассмотрим подробно каждую из этих подзадач.

Пусть G_j – группа визуализации для эффекта E_k. Предположим, что для любой G_j существует алгоритм (определяемый E_k), переводящий атрибуты объектов из G_j в набор данных рендера $\{D_1, D_2, ..., D_k\}$, $D_i \in RT$, $i = \overline{1,k}$, и строящий последовательность инструкций $(F_1, F_2, ..., F_t)$, $F_i \in RI$, $i = \overline{1,t}$.

В простейшем случае последовательность $(F_1, F_2, ..., F_t)$ (см. [2]) содержит *k* инструкций по установке источников данных, и одну, инициирующую их отрисовку, т. е. всего $t = k + 1$ инструкция. Поэтому $(F_1, F_2, ..., F_t)$ можно представить в виде пары отрисовки $(Set, Draw)$, где $Set = (F_1, ..., F_k) \subset RIS$ инструкции установки для набора данных $\{D_1, D_2, ..., D_k\}$, а $Draw = F_{k+1} \in RID$ – инструкция отрисовки. В общем случае построенная для группы G_j последовательность может содержать несколько установок и отрисовок:

$$(F_1, F_2, ..., F_t) = ((Set_1, Draw_1), ..., (Set_p, Draw_p)).$$

Результирующую последовательность инструкций рендера можно представить в виде:

$$Seq = ((Set_1, Draw_1), (Set_2, Draw_2), ..., (Set_N, Draw_N)).$$

Таким образом, для отображения канонической задачи *Z* в виде кадра необходимо построить множество $\{D_i\}$ и последовательность инструкций *Seq*.

Общую схему алгоритма построения кадра можно описать следующим образом.

Алгоритм 2.

1. Этап подготовки.
   Для сцены *S*:
   1.1. Объединить простые объекты в кластеры.
   1.2. Упростить и фрагментировать сложные объекты.
   1.3. Подготовить вспомогательные данные для групп визуализации.
2. Этап моделирования процессов и визуализации.
   2.1. Обновить значения динамических атрибутов объектов сцены.
   2.2. Для каждого моделируемого эффекта визуализации E_k:
      2.2.1. Оптимизировать текущее состояние сцены.
      2.2.2. Сформировать группы визуализации и выполнить перевод.
      2.2.3. Построить последовательность инструкций и провести ее оптимизацию.
      2.2.4. Выполнить последовательность инструкций и получить кадр.
   2.4. Перейти к шагу 2.1.

Нетрудно видеть, что эффективность данного алгоритма существенным образом зависит от решения указанных подзадач. Каждая из них имеет собственную природу, и при этом они могут решаться независимо друг от друга. Например, первые две можно решать с помощью декомпозиции и упрощения данных путем объединения объектов и анализа их видимости [5], а оптимизация последовательности пар отрисовки является типичной задачей теории расписаний [6] (решение указанных задач выходит за рамки настоящей работы).

Таким образом, введение канонического представления задачи визуализации позволяет строить эффективные алгоритмы указанных подзадач, которые в совокупности образуют алгоритм, инвариантный относительно частных задач. Все это в свою очередь позволяет построить универсальную программную библиотеку, автоматизирующую весь процесс получения кадров в рамках исследуемого подхода.

* * *

В результате рассмотрения проблемы визуализации динамических моделей на основе стандартных аппаратно-программных средств выявлены основные проблемы, влияющие на эффективность процесса визуализации и сформулированы две базовые подзадачи: организации данных и построения кадров. В рамках первой из них была построена каноническая задача, что позволило унифицировать процесс построения кадров и сделать его инвариантным относительно практических задач визуализации.

1. Shirley P. Fundamentals of Computer Graphics. Wellesley, 2002. P. 51.
2. DirectX Software Development Kit Documentation, Microsoft Corporation, 2008.
3. Gregory J. Game engine architecture. Wellesley, 2009. P. 5.
4. Краснопрошин В.В., Маркова О.А., Вальвачев А.Н. // Информатика. Мн., 2007. № 3. С. 124.
5. Akenine-Möller T., Haines E., Hoffman N. Real-time rendering. 3rd edition. Wellesley, 2008. P. 645.
6. Севастьянов С.В. Введение в теорию расписаний. Новосибирск, 2003.

Поступила в редакцию 26.01.10.

Виктор Владимирович Краснопрошин – доктор технических наук, профессор, заведующий кафедрой математического обеспечения автоматизированных систем управления,

Дмитрий Игоревич Мазовка – магистрант факультета прикладной математики и информатики.