готовленном библиотечном файле. Работа с библиотечным элементом осуществляется как с единым целым аналогично работе с базовыми элементами. Определение библиотечных элементов позволяет осуществлять иерархический подход к проектированию.

Библиотечные элементы в списках задаются в виде ссылки на тело элемента и точки привязки элемента. Структура ссылки аналогична структуре базовых типов элементов. Помимо обеспечения возможности определения уровней иерархии, библиотечные элементы позволяют существенно сократить объем базы данных. Его дальнейшее сокращение достигается благодаря применению итераций библиотечных элементов, что позволяет весьма кратко описывать регулярные структуры. Итерации представляются ссылкой на библиотечный элемент, заданием точки привязки, шагов по осям X и Y, а также количества библиотечных элементов по осям X и Y.

Описания тел элементов представляют собой сортированные списки, аналогичные рабочим спискам, хранящиеся в отдельной длинной памяти. Поскольку с телами библиотечных элементов не работают непосредственно, а только через ссылки на тело элемента, то разбиение списков на диапазоны не является необходимым и поэтому не применяется.

Результат работы системы — файл с полным описанием топологии, на основе которого в дальнейшем может быть получена программа для управления фотопостроителем.

ЛИТЕРАТУРА

1. **Рассохин В. А., Трубкин В. А.** Универсальная мини-ЭВМ на базе процессора с разрядно-модульной организацией.— Автометрия, 1986, № 4.
2. **Остапенко А. М., Шеметов С. А.** Цветной графический дисплей.— Автометрия, 1984, № 4.

Поступило в редакцию 14 февраля 1986 г.

УДК 681.3.06

Б. Х. ЗИНГЕР
(Новосибирск)

ОБ АЛГОРИТМАХ ПОСТРОЕНИЯ ПЛОСКОСТИ, РАЗДЕЛЯЮЩЕЙ КОНЕЧНЫЕ МНОЖЕСТВА ТОЧЕК

В системах трехмерной машинной графики, использующих для удаления невидимых поверхностей алгоритмы приоритетного типа, плоскость, разделяющая модель сцены на две части, позволяет глобально решить вопрос о взаимных приоритетах объектов, находящихся в разных полупространствах, образуемых этой плоскостью, а именно объекты, располагающиеся в одном полупространстве с наблюдателем, не могут закрываться объектами из другого полупространства. Такую технику можно считать классической [1—3]. В данном сообщении предлагаются соображения, которые могут составить основу алгоритмов построения разделяющих плоскостей.

Определение. Плоскость $Ax + By + Cz + D = 0$ называется разделяющей множества U и V, если для любых $u \in U$ и $v \in V$ $d(u) \leqslant 0 \leqslant d(v)$ либо для любых $u \in U$ и $v \in V$ $d(v) \leqslant 0 \leqslant d(u)$. Здесь $d(p) = Ax + By + Cz + D$ при $p = (x, y, z)$.

Пусть имеется два конечных множества в R^3 $U = \{u_1, \ldots, u_n\}$ и $V = \{v_1, \ldots, v_m\}$. Для простоты дальнейших рассуждений допустим, что каждое из множеств содержит четыре точки, не лежащие в одной плоскости.

Утверждение 1. Если существует плоскость, разделяющая U и V, то существует разделяющая плоскость, проходящая через три точки множества $U \cup V$, не лежащие на одной прямой.

Утверждение 1 позволяет строить алгоритм, основанный на переборе всех линейно независимых троек точек из U и V и попытке построить на них разделяющую плоскость. Учитывая, что проверка разделяемости множеств U и V данной плоскости может привести еще к одному перебору всех точек, сложность данного алгоритма будет пропорциональна $(n + m)^4$. Можно предложить ряд приемов сокращения перебора.

Если в исходных множествах выделены плоские грани, то из вершин, принадлежащих одной грани, достаточно проверить одну тройку.

При проверке разделяемости множеств плоскостью можно выбирать точки поочередно из одного и из другого. Таким способом можно раньше обнаружить факт неразделяемости и перейти к проверке следующей тройки.

Если предварительно построить выпуклые оболочки обоих множеств (с использованием, например, алгоритма [4]), то достаточно рассматривать только точки,

являющиеся вершинами полученных выпуклых многогранников. Применяя к множествам вершин утверждение 1 и учитывая, что для любой тройки вершин, по крайней мере, две принадлежат одному многограннику, получаем, что существует разделяющая плоскость, проходящая через грань либо через вершину одного многогранника и ребро другого. Отсюда следует, что сложность алгоритма построения разделяющей плоскости для выпуклых многогранников не превосходит порядок $(n+m)^3$.

Утверждение 2. Если два выпуклых многогранника отделимы, то найдется два непараллельных ребра этих многогранников такие, что плоскость, проходящая через одно из ребер параллельно другому, будет разделяющей. (Заметим, что выбор двух ребер, принадлежащих одной из граней выпуклой оболочки, определяет плоскость, проходящую через грань.)

Учитывая, что два множества отделимы тогда и только тогда, когда отделимы их выпуклые оболочки, а выпуклая оболочка конечного множества точек является выпуклым многогранником, утверждение 2 будет обоснованием для алгоритма, базирующегося на построении выпуклых оболочек. Перебор производится по всем граням обеих выпуклых оболочек, а затем по парам ребер, принадлежащих разным выпуклым оболочкам.

Укажем еще ряд приемов, которые при реализации алгоритмов в некоторых специальных случаях позволяют либо сократить объем вычислений, либо обнаружить разделяющую плоскость на более ранних этапах. Если в реализации алгоритма требуется ввод информации об исходных множествах, то при вводе легко провести тест на минимум и максимум по отдельным координатам всех вершин, т. е. заключить множества в параллелепипеды с гранями, параллельными координатным плоскостям. В случае отделимости параллелепипедов исходные множества также будут отделимы. Такой случай достаточно часто встречается в реальной практике.

Если одно из исходных множеств представляет собой совокупность вершин многогранника, являющегося трехмерным телом, то легко показать, что выпуклую оболочку достаточно строить на тех гранях множеств, которые «освещаются» источником, расположенным в одной из вершин второго множества. При определении «освещенности» проверяются только источник и сама грань без учета закрытия другими гранями.

Описанный здесь подход опробирован в задачах предобработки данных для системы синтеза визуальной обстановки.

ЛИТЕРАТУРА

1. **Sutherland I. E., Sproul R. F., Schumacker R. A.** A characterization of ten hidden surface algorithms.— Comp. Surv., 1974, v. 6, N 1.
2. **Fuchs H., Kedem Z. M., Naylor B.** Predetermining visibility priority in 3-*D* scenes.— Computer Graphics, 1979, v. 13, N 2.
3. **Fuchs H., Adram G. D., Grant E. D.** Near real-time shaded display of rigid objects.— Computer Graphics, 1983, v. 17, N 3.
4. **Дейкстра Э.** Дисциплина программирования.— М.: Мир, 1978.

Поступило в редакцию 4 марта 1986 г.

УДК 681.3.06

А. В. ИОФФЕ

(Новосибирск)

ТЕСТИРОВАНИЕ, ДИАГНОСТИКА И НАЛАДКА ЦИФРОВЫХ УСТРОЙСТВ С ИСПОЛЬЗОВАНИЕМ ИЕРАРХИЧЕСКОЙ СХЕМЫ ПРОГРАММНЫХ МОДЕЛЕЙ

В данной заметке рассмотрен подход к разработке автоматизированных средств тестирования аппаратуры, использованный при создании синтезирующих систем визуализации, описанных в [1—3].

Для устройства, подлежащего тестированию, строится иерархическая система программных моделей. Модели самого верхнего уровня описывают поведение устройства в целом или какой-то функционально-независимой его части в терминах сигналов в основных входных и выходных контрольных точках. Модели более низких уровней относятся к подсхемам, представленным меньшим объемом оборудования, и описывают сигналы на внутренних контрольных точках. Следует подчеркнуть, что программные модели не обязательно должны непосредственно имитировать работу оборудования: от них требуется лишь точность расчета состояний в контрольных точках в соответствии с функционированием устройства. За счет этого модели могут