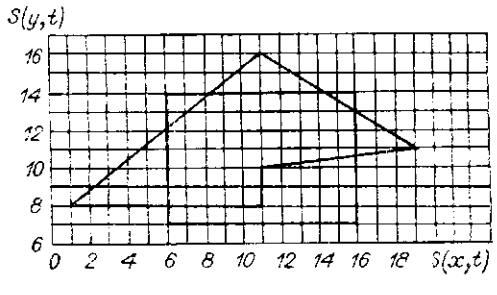


Рис. 3

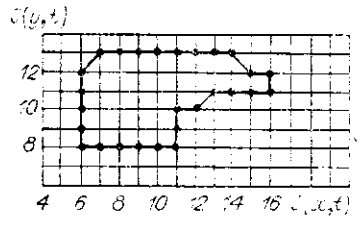


Рис. 4

исходного многоугольника. Окончание работы алгоритма совпадает в этом случае с возвратом в начальную точку преобразуемого графического объекта. Перечисленные особенности позволяют рекомендовать разработанный алгоритм отсечения для использования в специализированных графических системах реального времени.

СПИСОК ЛИТЕРАТУРЫ

1. **Фоли Дж., вэн Дэм А.** Основы интерактивной машинной графики.— М.: Мир, 1985.— Кн. 2.
2. **Гилой В.** Интерактивная машинная графика: Структуры данных, алгоритмы, языки.— М.: Мир, 1981.
3. **Павлидис Т.** Алгоритмы машинной графики и обработки изображений.— М.: Радио и связь, 1986.
4. **Кларк Д. Х., Дэйвис Т.** Универсальная рабочая станция инженера-конструктора // Электроника.— 1983.— 56, № 20.
5. **Байков В. Д., Смолов В. Б.** Специализированные процессоры: Итерационные алгоритмы и структуры.— М.: Радио и связь, 1985.
6. **Айдемиров И. А., Хачумов В. М., Шабалов Д. В.** Применение алгоритмов Волдера для быстрого преобразования «вектор — растр» // Изв. вузов. Приборостроение.— 1988.— **XXXI**, № 7.
7. **Батурицкий М. А.** Применение аппаратного вычисления элементарных функций по алгоритму Волдера в СЦВМ для обработки данных в реальном времени // Проектирование специализированных процессоров ЭВМ.— Л.: ЛГУ, 1986.— (Организация вычислительных структур и процессов.— Вып. 10).

Поступила в редакцию 28 ноября 1988 г.

УДК 681.3.06

А. В. БУШМЕЛЕВ, Е. П. КУЗЬМИН
(Москва)

СЖАТИЕ ИЕРАРХИЧЕСКИХ СТРУКТУР

Пространственные иерархические структуры в настоящее время широко применяются в системах машинной графики для представления трехмерной геометрической информации [1, 2]. Основным недостатком подобного представления является большой объем памяти, требующийся для хранения структур.

В работе предложены новые иерархические структуры — адаптивные бинарные деревья и бинарные сети, позволяющие значительно понизить количество узлов иерархических представлений.

Введение. Общая схема получения пространственных иерархических структур выглядит следующим образом. Рассмотрим мировое пространство — прямоугольный параллелепипед, содержащий трехмерную

сцену. Разделим его на несколько равных частей (узлов), каждая из которых, в свою очередь, является параллелепипедом. Узел может располагаться внутри или вне объектов сцены (томогенные узлы) либо пересекаться с границами объектов (гетерогенные узлы). Узлы последнего типа снова делятся подобным образом, и процесс продолжается рекурсивно. Представляя его графом, и получаем иерархическую структуру. Процесс деления завершается при выполнении одного из следующих условий.

1. Сцена внутри узла «простая» (малое количество примитивов в кубике либо они принадлежат определенным классам). Это представление — наполненное иерархическое дерево. Терминальные узлы содержат ссылки на структуры.

2. Достигнуты заданные минимальные размеры узла; при этом по соглашению терминальные узлы считаются либо пустыми, либо лежащими внутри тела. Это **представление** — чистое иерархическое дерево [3].

Будем рассматривать чистые иерархические структуры, поскольку полученные для них результаты легко переносятся и на наполненные структуры.

Восьмеричные деревья. При делении кубического мирового пространства на восемь одинаковых частей получаем структуру, называемую **восьмеричным деревом**. Процедура получения восьмеричных деревьев достаточно хорошо исследована. В ее основе лежит набор алгоритмов определения пересечения геометрических примитивов с кубом. Следует отметить, что куб в этих проверках может быть заменен на некие пробные тела (шары, цилиндры), содержащие этот куб, что позволяет значительно ускорить процесс генерации.

Существенный недостаток этого подхода — большой объем ресурсов памяти, необходимых для хранения структуры. Рассмотрим единичный куб и назовем разрешением величину, обратную размеру минимального кубика. Тогда количество узлов восьмеричного дерева растет пропорционально квадрату разрешения [4]. При высоком разрешении дерево содержит миллионы узлов.

В качестве примера рассмотрим шар единичного радиуса с центром в вершине единичного мирового куба. Тогда с ростом разрешения R коэффициент пропорциональности $K = N/(R \times R)$ (где N — количество узлов дерева) изменяется следующим образом:

R	8	16	32	64	128	256	512	1024
K	2,8	2,9	3,0	3,1	3,1	3,2	3,2	3,2

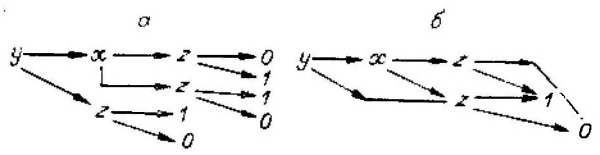
Бинарные деревья. Рассмотрим другой подход к построению иерархических структур. Будем делить мировой параллелепипед пополам плоскостями, параллельными его граням. Выбирая плоскости деления циклически, получаем структуру, называемую **бинарным деревом.** В этом случае на той же сцене с ростом разрешения количество узлов и коэффициент пропорциональности изменяются следующим образом:

R	8	16	32	64	128	256	512	1024
K	1,4	1,6	1,7	1,7	1,8	1,8	1,8	1,8

Бинарное дерево позволяет снизить количество узлов в представлении за счет того, что гомогенные узлы удается отбросить на более раннем этапе.

Еще более снизить количество узлов можно, если порядок выбора оси деления не фиксирован, как в традиционных бинарных деревьях, а обусловлен расположением сцены в мировом кубе. Построенная структура — **адаптивное бинарное дерево** — позволяет понизить количество узлов. Существенным вопросом становится порядок выбора плоскости деления.

Можно предложить ряд простых правил для определения плоскости деления:



Построение бинарной сети:
а — бинарное дерево, *б* — соответствующая ему бинарная сеть

необходимо производить деления плоскостями, если при этом возникают гомогенные узлы;

деление осуществляется плоскостью, перпендикулярной оси с наименьшим отношением; рассмотрим отношения проекций границ сцены в узле на три оси к длинам соответствующих сторон.

При использовании этих правил для сцены с шаром наши результаты примут следующий вид:

R	8	16	32	64	128	256	512	1024
K	1,0	1,2	1,2	1,3	1,35	1,4	1,4	1,4

Следует отметить, что можно и дальше уменьшать объем дерева, но при применении более сложных правил размер уменьшения не сопоставим с увеличением времени генерации дерева.

Адаптивные бинарные деревья позволяют значительно (на несколько порядков) сократить количество узлов дерева, если большинство граней объектов сцены параллельно ребрам мирового пространства (архитектурные сцены). Например, адаптивное дерево для плоскости, параллельной грани куба, состоит из нескольких десятков узлов, а бинарные и восьмеричные деревья содержат несколько сотен тысяч узлов.

Адаптивные бинарные деревья можно хранить как в линейном, так и в регулярном представлении.

Бинарные сети. Применение адаптивных бинарных деревьев позволяет во многих случаях добиться требуемого уменьшения количества узлов, но рост в общем случае остается квадратичным.

Выход состоит в построении на основе бинарного дерева **бинарной сети**, позволяющей избежать дублирования информации, содержащейся в дереве. Действительно, если два поддерева бинарного дерева совпадают, то можно хранить только одну копию поддерева, перебросив на нее все ссылки на данное поддерево.

Совпадающие поддеревья строятся, начиная с терминальных узлов. На рисунке приведен пример построения сети по дереву.

Следует отметить, что выбор бинарного дерева для построения сети почти не влияет на ее объем.

С помощью бинарной сети удается значительно уменьшить количество узлов структуры. Для сцены с шаром наши результаты примут вид

R	8	16	32	64	128	256	512	1024
K	—	—	0,4	0,3	0,3	0,2	0,15	0,1

Важное свойство бинарной сети — медленный рост количества узлов с ростом разрешения (менее чем квадратичный).

Алгоритмы обработки сети: теоретико-множественные операции, геометрические преобразования, визуализация и другие — не отличаются от аналогичных алгоритмов для бинарных деревьев, так как с точки зрения обрабатывающей программы их представления совпадают.

Для хранения бинарной сети необходимо регулярное представление.

Заключение. Полученные результаты легко переносятся как на случай больших размерностей, где иерархические структуры применяются сейчас крайне слабо из-за их объема, так и на плоские растровые изображения и разреженные матрицы.

Представляют большой интерес асимптотические оценки количества узлов в оптимальных адаптивных деревьях и сетях, а также быстрые алгоритмы их построения.

СПИСОК ЛИТЕРАТУРЫ

1. **Meagner D.** Geometric modeling using oct-tree encoding // Comp. Graph. and Image Proc.— 1982.— 18, N 2.— P. 129.
2. **Sammet H., Tamminen M.** Bintrees, CSG trees and time // Comp. Graph.— 1985.— 19, N 3.— P. 121.
3. **Jackins C. L., Tanimoto S. L.** Oct-trees and their use in representing three-dimensional objects // Comp. Graph. and Image Proc.— 1980.— 14, N 3.— P. 249.
4. **Walsh T. R.** On the size of quadtrees generalized to d-dimensional binary pictures // Comp. and Maths. with Appl.— 1985.— 11, N 11.— P. 1089.

Поступила в редакцию 17 января 1990 г.

УДК 681.3.06

Е. П. КУЗЬМИН
(Москва)

ВИЗУАЛИЗАЦИЯ ИЕРАРХИЧЕСКИХ СТРУКТУР

Введение. В отличие от трассировки ячеистых структур [1, 2], где получены эффективные целочисленные алгоритмы генерации трассы луча, существующие методы построения трассы луча для иерархических структур [3, 4] основаны на вещественной арифметике и многократно используют поиск по дереву, что ограничивает область их применения лишь наполненными деревьями низкого разрешения.

В этих алгоритмах определяется последовательность терминальных узлов дерева методом поиска узла, содержащего фиксированную точку. Данный подход приводит к высокой зависимости времени трассировки от разрешения сцены, что дает основание считать подобный метод неперспективным [1].

Построение трассы луча. Рассмотрим трассируемый луч в мировом пространстве с иерархией. Алгоритм построения трассы этого луча основан на процедурной генерации иерархического дерева трассы луча на основе иерархического дерева сцены. Генерация производится в направлении удаления от наблюдателя до прихода на поверхность или выхода из мирового куба.

Тогда алгоритм построения трассы выглядит следующим образом:

```
трассировка (узел, луч)
{
  для (всех узлов_сыновей в порядке удаления от наблюдателя)
   выполнить
   {
      если (узел_сын не пуст и пересекается с лучом)
      {
     если (узел_сын — терминальный)
      {
    трассировка_узла (узел_сын, луч)
      }
     иначе
      {
     трассировка (узел_сыи, луч)
      }
      }
   }
}
```